A geometric idea to solve the eikonal equation

Martin Peternell[*]
Institute of Discrete Mathematics and Geometry
Vienna University of Technology

Tibor Steiner[†]
Institute of Discrete Mathematics and Geometry
Vienna University of Technology

Abstract

Given a closed plane curve $\mathbf{c}(t) = (c_1, c_2)(t) \in \mathbb{R}^2$ and associated function values $g(t)$ we present a geometric idea and an algorithm to solve the equation $\|\nabla f\| = a = \text{const.}$ with respect to the boundary values $g(t)$ along the boundary $\mathbf{c}(t)$. This is equivalent to finding a developable surface D of constant slope $a = \tan\alpha$ through the spatial curve C determined by $(c_1, c_2, g)(t)$. The presented method constructs level curves of the surface D. We put some emphasis on the treatment of the singularities of the solution which are D's self intersections.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; J.6 [Computer–Aided Engineering]: Computer–Aided Design—(CAD)

Keywords: developable surface, distance function, eikonal equation, level curves.

1 Introduction

This article presents geometric ideas to solve the eikonal equation $\|\nabla f\| = a = \text{const.}$ for given boundary data. These data consist of a closed plane curve $\mathbf{c} = (c_1, c_2)(t) \in \mathbb{R}^2$ and function values $g(t)$ along $\mathbf{c}$.

The eikonal equation is a well studied subject in numerical analysis. The *fast marching* [Sethian 1999] and *fast sweeping* [Osher and Fedkiw 2003] *methods* are probably the most famous techniques in this context. Often the eikonal equation $\|\nabla f\| = a = \text{const.}$ is solved for given boundary data consisting of an interface in the form of a curve $\mathbf{c} \in \mathbb{R}^2$ and $g = 0$ along $\mathbf{c}$. The construction of a solution is equivalent to evaluate the distance function of the curve $\mathbf{c}$, which is in fact a planar problem.

The solution to the eikonal equation is known in advance, or more precisely, the solution $f(x,y)$ determines a developable surface D of constant slope a, where D is represented by the graph surface $(x, y, f(x,y))$. Since geometric properties of developable surfaces of constant slope are quite well understood, we present a technique motivated by the constructive geometric properties of these surfaces. We will discuss the special case of boundary values $g = 0$ along the boundary curve $\mathbf{c}$, but this paper focuses on the more general case where these values $g(t)$ are not constant. Thus, D is a surface of constant slope through the spatial curve C given by $(c_1, c_2, g)(t)$. The function $\gamma(t) = \dot{g}(t)/\|\dot{\mathbf{c}}(t)\|$ along the boundary curve $\mathbf{c}$ is the slope of the spatial curve C, i.e. the slope of its tangent lines. Real solutions exist exactly if $|\gamma(t)| \le a$ holds for all t. Particular emphasis will be laid on the treatment of the singularities of f which are represented by the self intersections of D.

The proposed method computes level curves of the developable surface D at prescribed heights. Each surface strip of D which lies between two adjacent level curves is represented as a triangular mesh. This idea is a generalization of a scan-line algorithm and uses a moving horizontal plane instead of the line. The horizontal level curves will be trimmed. This leads to a smoothing of the singularities of the surface D, and finally we study methods to improve the shape of D near these self intersections.

This research has been motivated by a project with a civil engineering company for the planning of artificial terrain of roads and excavations. The primary problem has been the following: Consider a digital terrain model and a curve or a polygon C. We want to construct a developable surface D of constant slope (with respect to the horizontal planes) which passes through the given curve C. Thereby, C is considered as the boundary of a road or of an excavation. The surface D has to be represented by a triangulation and the intersection of D and the given terrain has to be computed. If the slope γ of C is small, like for the boundary of a road, one can use a modification or correction of the method for horizontal input curves $\mathbf{c}$ (evaluation of the distance function).

The paper is organized as follows: In section 2 we briefly recall some geometric properties of developable surfaces. Sections 3 and 4 describe the practical implementation issues for horizontal and general input curves, respectively. Finally, section 5 deals with the handling of global self intersections.

2 Some mathematical background

The discussion of geometric properties of developable surfaces of constant slope distinguishes between horizontal boundary curves ($g = 0$) and general boundary curves C with $g(t) \neq 0$. Throughout the paper we assume that C's slope satisfies $|\gamma(t)| \le a$ with $a > 0$.

2.1 Plane input curve

Let C be a given horizontal curve, lying in a plane $E: z = z_0$, and let D be the developable surface of constant slope $a = \tan\alpha$ passing through C. The slope of D's tangent planes equals a and so D can be generated as the envelope of planes with slope a passing through the tangent lines of C. The surface D is the graph of the function

$$f(\mathbf{x}) = z_0 \pm \text{dist}(\mathbf{x}, \mathbf{p}') \tan\alpha, \tag{1}$$

where $\mathbf{x} = (x, y, 0)$ is an arbitrary query point in the xy-plane, $\mathbf{p} = (p_1, p_2, z_0) \in C$ is the closest point to $\mathbf{x}$ and α with $0 < \alpha < \pi/2$ is the angle between D's tangent planes and the xy-plane. The projection of $\mathbf{p} \in C$ onto $z = 0$ is denoted by $\mathbf{p}' = (p_1, p_2, 0) \in C'$. The function $f(\mathbf{x})$ satisfies

$$\|\nabla f\| = \tan\alpha. \tag{2}$$

[*]e-mail: martin@geometrie.tuwien.ac.at
[†]e-mail: tibor@geometrie.tuwien.ac.at

For $\alpha = \pi/4$, (2) is called *eikonal equation*. This can always be achieved by an appropriate scaling of the function values. Because of the occurring singularities, f is a solution in the weak sense.

The surface D consists of two sheets D^-, D^+ which are given by the different signs in (1). Concerning the applications, we are only interested in one of these two solutions and denote it by D.

Fig. 1 shows a strip of a developable surface of constant slope through a closed curve C.

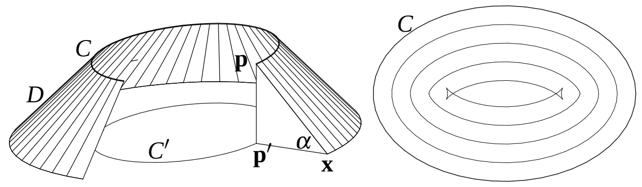

Figure 1: Left: Developable surface through an ellipse C, evaluation of the distance function of C' at $\mathbf{x}$. Right: Ellipse with untrimmed interior offset curves.

The geometric properties of developable surfaces are well known, and can be found in many textbooks, see for instance [Pottmann and Wallner 2001]. We list some important facts:

- The developable surface D of constant slope a through C is the envelope of cones of revolution G with slope a, whose vertices move along C, see Fig. 2.
- The surface D is the envelope of a one-parameter family of planes which are tangent to D and G along the generating lines of D. Thus, D is also generated by these lines.

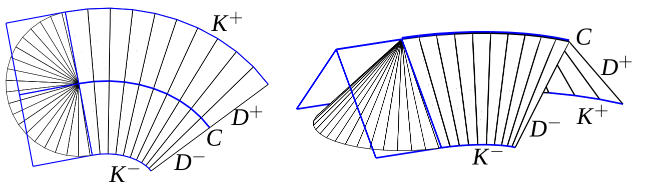

Figure 2: Top view and axonometric view of the developable surface through a horizontal curve, tangent cone and tangent planes along two generators.

- The vertical projections of the contour lines of D onto the xy-plane are offset curves of the projection of the input curve C'. The singularities of the untrimmed offset curves are located at the evolute C^* of C'.
- The evolute C^* is the vertical projection of the singular curve S of D onto the xy-plane.
- The medial axis M of C' is the orthogonal projection of those parts of the self intersection S of the surface D onto the xy-plane which are visible from above.
- The normals of the curve C' are the vertical projections of the generating lines of the surface D. These normals envelope the projection S' of the singular curve S of D.

2.2 Spatial input curve

Let C be a spatial curve, parametrized by $(c_1, c_2, g)(t)$ and let D be a developable surface of constant slope a through C. In order to obtain real solutions for D, it has to be required that $|\gamma| \leq a$, where $\gamma = \dot{g}(t)/\|\dot{\mathbf{c}}(t)\|$ denotes C's slope and $\mathbf{c}(t) = (c_1, c_2)(t)$.

The surface D is represented as graph (x, y, f) of a function $f = f(x,y)$ which solves the eikonal equation $\|\nabla f\| = a$ and satisfies the boundary condition $f(c_1, c_2) = g$ for all points (c_1, c_2) on the projection C' of C.

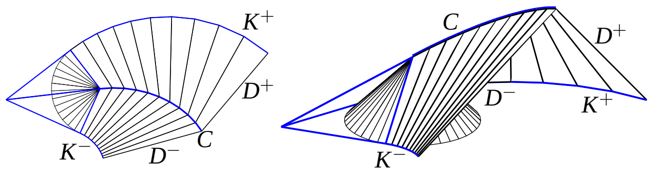

Figure 3: Developable surface through a space curve C with level curves K^+, K^- of the two sheets D^+, D^-. Tangent cone and tangent planes along two generators.

Fig. 3 shows a top view and an axonometric view of the two sheets D^+, D^- of a developable surface through a general space curve C. The level curves in the plane H are denoted by K^+, K^-. We observe the following properties of D:

- The developable surface D of constant slope a through C is the envelope of cones of revolution G with slope a, whose vertices move on C.
- The surface D is the envelope of a one-parameter family of planes of slope a which pass through C's tangent lines. These planes are tangent to D along its generating lines, and are tangent to the cones G, too. Thus, D contains a one parameter family of lines with constant tangent planes along them.
- The generating lines are tangent to the singular curve S of D, their orthogonal projections are tangent to the projection S', see Fig. 4.
- The generating lines are orthogonal trajectories of D's level curves.

Fig. 4 shows a top view and an axonometric view of a developable surface of constant slope passing through a (non-horizontal) parabola C.

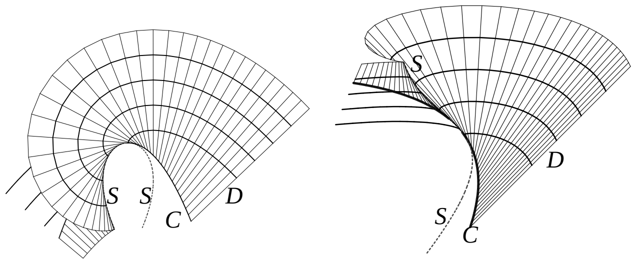

Figure 4: Developable surface D through a space curve C. The singular curve S (curve of regression) is displayed in dotted linestyle. Generating lines and level curves are shown.

3 The practical implementation for horizontal input curves

Now we go into more detail concerning the practical implementation. Given a horizontal input curve C, lying in the plane $E : z = z_0$, we want to compute the developable surface D of constant slope $a = \tan\alpha$ passing through C. We assume that C is given by a list of (ordered) data points $\mathbf{p}_i = (x_i, y_i, z_0)$, $i = 1, \ldots, M$. This sampling has to be dense enough to represent all important features of the curve. For convenience we describe the construction for simple closed input curves only.

The algorithm consists of three main steps:

- Define a domain U for the evaluation of the function f which determines the surface D.
- Define query points $\mathbf{x} = (x, y)$ to evaluate the distance function of the curve C' and compute f.
- Represent D by a triangular mesh with points $(x, y, f(x, y))$ as vertices. The boundary curves of U are constraints for the triangulation.

3.1 Details of the implementation

Domain U**:** The shape and the size of the domain U are mainly determined by the application. If no requirements are made, U can be bounded by C' and a trimmed offset curve C'_d of C' for a suitably chosen distance d.

Query points: The query points $\mathbf{x} \in U$ are chosen as points on a regular grid. The grid size shall not exceed the average segment length of C. We propose to start with a dense sampling of the input curve C, since the number of input data points is much smaller than the number of grid points $\mathbf{x}$ for evaluation of the distance function. By refining the grid it is possible to improve the accuracy of the representation of the surface D.

Evaluation of the distance function: Assume we have chosen N query points $\mathbf{x} \in U$, we evaluate the distance $\text{dist}(\mathbf{x}, \mathbf{p}')$ where $\mathbf{p}'$ is the closest point to $\mathbf{x}$ on C'. The function f which defines D, evaluates to

$$f(\mathbf{x}) = z_0 - \text{dist}(\mathbf{x}, \mathbf{p}') \tan\alpha. \tag{3}$$

The closest point $\mathbf{p}'$ is computed as *nearest neighbor* to the query point $\mathbf{x}$. For this, one may use the implementation of D. Mount [Arya et al. 1998].

Constrained triangulation: The surface is represented by a constrained triangular network T_D which shall contain also the list of segments of the boundary curves C and Q as edges of the triangulation. Here, Q is the intersection of the vertical cylinder through C'_d with D. The z-values of Q are computed by $z_0 - d\tan\alpha$. Our test implementation uses the program *Triangle* by J.R. Shewchuk, see [Shewchuk 1996; Shewchuk 2002].

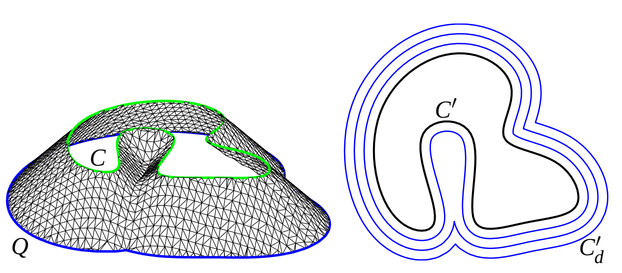

Figure 5: Left: Triangular representation of the developable surface through a horizontal curve C. Rigth: Projection C' and trimmed offset curves C'_d for different distances.

Fig. 5 shows an axonometric view of the developable surface D of constant slope $a = 1$ through the curve C. The domain of evaluation U is bounded by the top view C' of the input curve C and a trimmed offset C'_d at a certain distance. The right hand side figure shows different trimmed offset curves C'_d of the top view C' of the input curve C.

What we did not discuss so far is the trimming operation of offset curves. An approach which is based on triangulations and which works well in our framework will be discussed in the next section.

3.2 Trimming of offset curves

We are given a discretized input curve C' by a list of points $\mathbf{p}_i$, $i = 1, \ldots, M$ which form a simple polygon. The trimmed offset curve C'_d at signed distance d is to be computed and has to be free of self intersections.

To trim the offset curves we apply a triangulation based approach. For all segments $\mathbf{p}_i\mathbf{p}_{i+1}$, with $i = 1, \ldots, M$ we compute the parallel segment $\mathbf{q}_i\mathbf{q}_{i+1}$ at oriented distance d with $\mathbf{q}_i = \mathbf{p}_i + d\mathbf{n}_i$ and $\mathbf{n}_i$ as unit normal of the i-th segment. The end points of the translated segments do not conincide in general but these segments will intersect or there will be gaps between them, see Fig. 6.

Instead of translating segments one can move the vertices $\mathbf{p}_i$ to positions $\mathbf{q}_i = \mathbf{p}_i + d\mathbf{n}_i$, where $\mathbf{n}_i$ is the average normal of adjacent segments of $\mathbf{p}_i$. Trimming of the offsets is necessary in both cases.

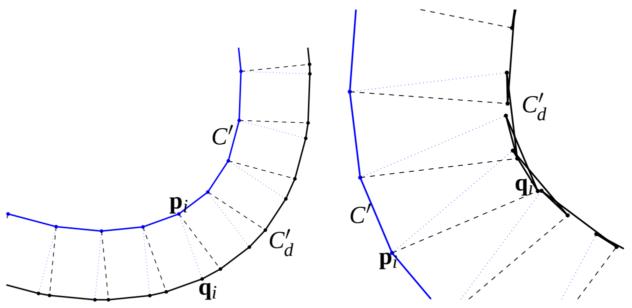

Figure 6: Parallel segments in convex and non-convex regions of the input curve C'.

The end points of the translated segments $\mathbf{q}_i\mathbf{q}_{i+1}$ are collected to form a curve $\tilde{C}_d$ which usually has self intersections. To trim these self intersections, we perform a constrained triangulation with the data points $\mathbf{p}_i, \mathbf{q}_i$ as input points and the segments of these curves as constraint segments of the triangulation. The exterior boundary of the triangular net is then a sufficiently good representation of the trimmed offset curve C'_d of C', see right hand side of Fig. 5. The intersection points of crossing segments $\mathbf{q}_i\mathbf{q}_{i+1}$ are computed and inserted as additional data points.

The advantages of this method are its good performance and the fact that it works for all input curves and offset distances. Obviously, the point density of C plays an important role and we want to note that the sampling in highly curved regions of C' should be dense enough, to represent all features of the curve. In order to guarantee a similar point density on the trimmed offset, C'_d can be resampled.

4 The practical implementation for spatial input curves

We have already noted that the developable surface D of constant slope a through C will be represented by a triangular mesh based on the level curves of D. The input curve C is given by a list of data points $\mathbf{p}_i \in \mathbb{R}^3$. The method we propose is a generalization of a scan-line algorithm. In 2D, a y-parallel line is moved in x-direction and certain events, like when the moving line touches the object, are considered. The type of events depends on the type of application.

The basic idea is the following: We consider a horizontal plane $H: z = c$ and move this plane from above downwards, in several steps. These planes carry the level curves of D. We start at a level, where H is entirely above the input curve C, passing by the levels where H intersects C, and let it move downwards to a level where H is entirely below C. If H is completely above the curve C, nothing notable happens. If H intersects C, then H also intersects the developable surface D and $K = D \cap H$ is a level curve of D. All the level curves K project to top views K' which are offset curves of each other.

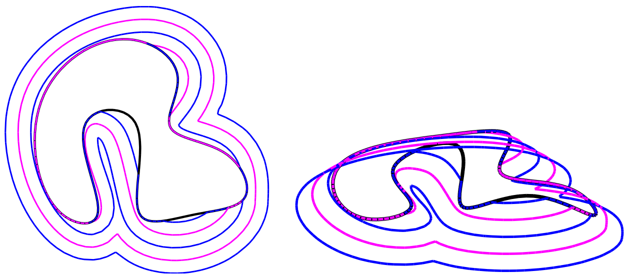
Figure 7: Top view and axonometric view of the trimmed level curves of a developable surface.

Now we consider D as the envelope of its generating lines L. It is quite obvious that we can replace C by one of the level curves K and the envelope D will not change. Thus, we can also replace parts of C by appropriate horizontal curve segments $G_j \subset D$ being defined later. In order to represent those parts and self intersections of D which are visible from above, the level curves will be trimmed, see Fig. 7.

The basic steps of the algorithm are the following. At first we choose $n+1$ levels $z_0, z_1, \ldots, z_n$, where $z_n \geq z_{max}$. Let $F = C$ be the input curve. Within a loop over all z-levels starting from z_{n-1} down to z_0, we proceed as follows:

1. Determine those parts A_j of F which lie above the current level $H: z = z_i$.
2. Pass the developable surface patches D_j through the curve segments A_j and compute the intersection $G_j = D_j \cap H$ of the patch D_j and the current level H, see Fig. 8.
3. Trim the horizontal curve segments G_j.
4. Replace the curve segments A_j in F by the horizontal curve segments G_j and store this collection as curve E_i. Define this as new input curve $F = E_i$.
5. Choose the next z-level and restart at 1.

This construction yields a list of curves E_i which are defined during the above loop. Each curve E_i consists of horizontal components at the level $z = z_i$ and possibly of components lying below the current level. The components below the level are part of the original input curve C.

Finally a constrained triangulation is performed with C and E_i, $i = 1, \ldots, n-1$ as input. The data points as well as the segments of C and E_i are considered, in order to represent D's level curves exactly.

4.1 Details for general input curves

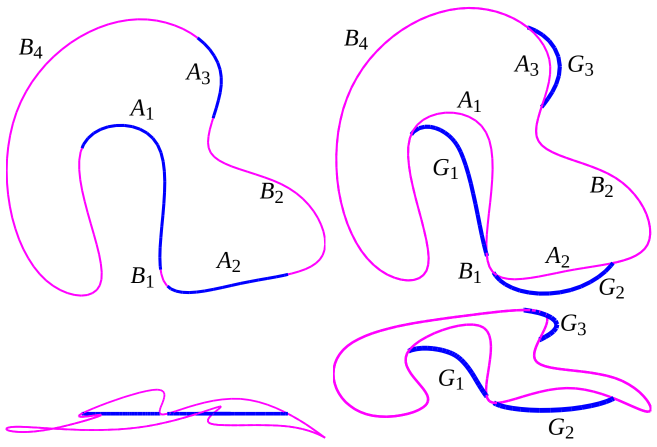

Figure 8: Components A_j of C above the current level and corresponding horizontal components G_j. Top: top views. Bottom: front view and axonometric view.

Determine parts of C above H: We compute those curve parts of C which are above the current z-level $H_i: z = z_i$. The intersection points $C \cap H_i$ are computed and are inserted into the list of data points $\mathbf{p}_i$ representing C. Thus, the curve is split into a list of curve components $A_1, \ldots, A_a$, $B_1, \ldots, B_b$. We store these curve components in the right ordering, but with flags telling which component is above the current z-level.

Developable surface through a curve: For each curve component A_j the generating lines of the developable surface patch D_j passing through A_j are considered. The intersection points of these lines and the current plane H determine the curve arc $G_j = D_j \cap H$. Since the end points of the components A_j are contained in H, the arcs A_j and G_j agree in their end points.

Trimming of the horizontal curve G_j: G_j is stored as a list of segments, similar to the offset curve construction. These segments overlap in non convex regions of the input curve A_j, and thus have to be trimmed. The applied procedure is similar to the one which is used for trimming offset curves.

Replace A_j by the horizontal curve G_j: We substitute the components A_j by the trimmed curve components G_j and store the collection as new input curve for the next step in the construction. Fig. 8 illustrates this procedure showing top view and front views of the curve components.

Constrained triangulation: A constrained triangulation is performed with data points and segments of the input curve C and level curves K_i as input.

Fig. 9 shows the top views of C and the level curves $K_1, \ldots, K_4$ on the left hand side and the axonometric view of the triangulation representing D on the right hand side. Most of the triangles shown in Fig. 9 have the correct slope, but close to the self intersection the

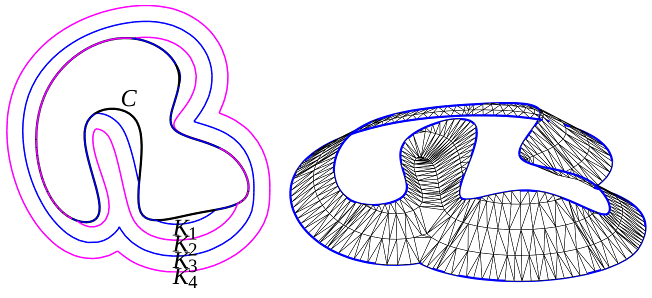

Figure 9: Top view of level curves and axonometric view of the triangular representation of the developable surface D.

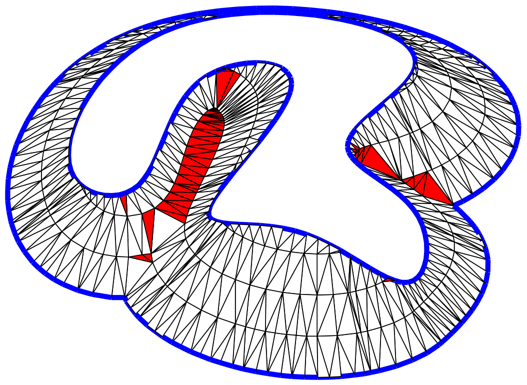

Figure 10: Triangles of wrong slope of Fig. 9.

triangles are not correct. In Fig. 10 all triangles are shaded whose slope differs from the required slope a by more than 1°.

We describe some possibilities to improve the representation of the developable surface. The first and obvious one is to increase the number of level curves which are computed in the construction. We want to note that the level curve computation is costly and it is always desirable to minimize the number of level curves which have to be computed. Results on increasing the number of level curves and the limitations of the currently implemented version are displayed in Fig. 11. Since the computed level curves are restricted to possess one component only, relevant parts of them are trimmed. These parts are denoted by X and Y in Fig. 11. Thus even by increasing the number of level curves, the developable surface contains horizontal triangles in these regions (X and Y). To solve this problem and to model these features, the method has to be extended in the way that level curves can consist of more than one component.

Alternative methods which increase the number of evaluation points locally or which try to compute the self intersection, are discussed in the next section.

5 Handling self intersections

Let T be a triangulation of a developable surface as shown in Fig. 10. We propose a method to improve the representation of the surface near self intersections. For that the resolution is increased in regions containing nearly horizontal triangles. The concept works as follows:

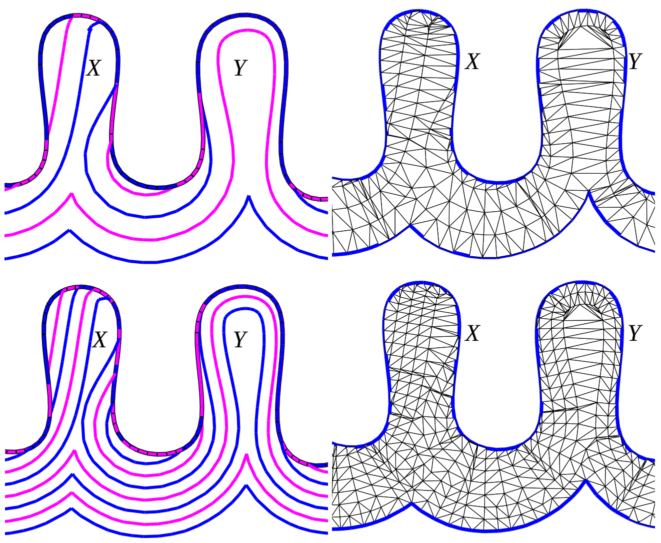

Figure 11: Top view of level curves and the triangular representation of a developable surface.

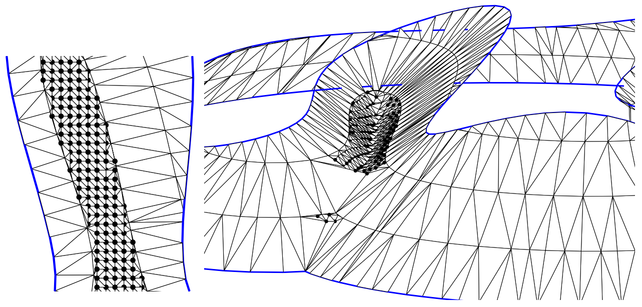

Figure 12: Left: Top view of selected grid points. Right: Improved representation of selected region.

- Find nearly horizontal triangles and mark them. Determine connected components of the marked triangles at height z_i.
- For each connected component we use a fine grid and pick those grid points which are contained in the component.
- Compute new function values for the chosen grid points. This is similar to the case of horizontal input curves. Since each connected component of marked triangles is at constant height z_i, nearly all segments of its boundary are contained in the level curves K_i of D. Thus, the new function values are computed as shortest distances from segments of these level curves, see Fig. 12.

Another method to improve the behavior of the surface close to the singularities tries to model the self intersections. To perform this one can proceed as follows:

- Find triangles with wrong slope and mark them. Determine connected components of the marked triangles.
- For each connected component R_i we determine its boundary b_i. If the shape of component R_i is curve-like, we compute its medial axis m_i. This medial axis can have different branches, depending on whether R_i is a simple curve-like shape or not. The boundary b_i of R_i consists of parts of level curves and parts of the given input curve C.

- Assume that m_i is a simple curve and consists only of one branch, see Fig. 13. Now we consider vertical planes perpendicular to the medial axis m_i and denote them as profile-planes. Each profile-plane intersects the boundary b_i of R_i in at least two points $\mathbf{b}_1, \mathbf{b}_2$. Now we consider lines l_1, l_2 with slope a (same slope as the surface) through these boundary points $\mathbf{b}_1, \mathbf{b}_2$. As these lines lie close to the developable surface D, their intersection point $\mathbf{s}$ lies close to the self intersection of D.
- Build a polygon S from these intersection points $\mathbf{s}$ and use S as a constraint boundary for the final triangulation of the developable surface D.

We note that this method can be applied to a region R_i whose medial axis m_i consists of several branches. According to our experience this can become arbitrarily complicated and we propose to use the previous method (refinement of the region by using a regular grid) in these cases.

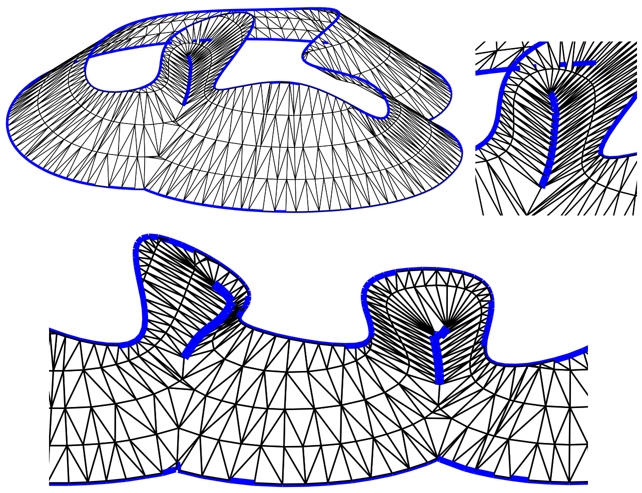

Figure 13: Self intersections of developable surfaces

Conclusion

So far we have presented methods to compute the developable surface D of constant slope a through a curve C. If C lies in a horizontal plane, the task is simply solved by evaluating the distance function of the top view C' of C. In case of general input curves we have proposed a generalization of scan-line algorithms by stepwise computation of the trimmed level curves of the developable surface D. By triangulating these level curves we obtain useful models for the developable surface D.

The test implementation has been performed in *Matlab*. The constrained triangulation and the nearest neighbors are computed with help of the programs *Triangle* by J.R. Shewchuk and *ANN* (approximate nearest neighbor searching) by D. Mount.

In applications it is often required to compute a developement of a developable surface D or to compute the volume enclosed by D (represented by $f(x,y)$) and a horizontal plane. At least the computation of volumes can be carried out in a straightforward manner using the proposed representation of D. The development to a plane works basically but close to self intersections the developability condition might not be fulfilled.

Acknowledgements: This work has been carried out partially within the Kplus competence center *Advanced Computer Vision*.

References

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM 45*, 891–923.

AUMANN, G. 2003. A simple algorithm for designing developable Bézier surfaces. *Computer Aided Geometric Design 20, 8–9*, 601–619.

CHALFANT, J., AND MAEKAWA, T. 1998. Design for manufactoring using b-spline developable surfaces. *Journal of Ship Research 42, 3*, 207–215.

CHU, C.-H., AND SÉQUIN, C. 2002. Developable Bézier patches: properties and design. *Computer-Aided Design 34, 7*, 511–527.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational Geometry, Algorithms and Applications*. Springer, Berlin, Heidelberg.

FREY, W., AND WAMPLER, C. 1999. Boundary triangulations approximating developable surfaces. Tech. Rep. 8997, GM Research and Development Center.

HOSCHEK, J., AND LASSER, D. 1993. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Wellesley, MA.

HOSCHEK, J., AND POTTMANN, H. 1995. Interpolation and approximation with developable B–spline surfaces. In *Mathematical Methods for Curves and Surfaces*, M. Daehlen, T. Lyche, and L. Schumaker, Eds. Vanderbilt University Press, Nashville, TN, 255–264.

KLEIN, R. 1997. *Algorithmische Geometrie*. Addison-Wesley, Bonn.

OSHER, S., AND FEDKIW, R. 2003. *Level Set Methods and Dynamic Implicit Surfaces*. Springer.

PARK, F., YU, J., CHUN, C., AND RAVANI, B. 2002. Design of developable surfaces using optimal control. *Transactions of the ASME, Journal of Mechanical Design 124*, 602–608.

POTTMANN, H., AND WALLNER, J. 1999. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design 16*, 539–556.

POTTMANN, H., AND WALLNER, J. 2001. *Computational Line Geometry*. Springer, Berlin-Heidelberg-New York.

SETHIAN, J. 1999. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

SHEWCHUK, J. R. 1996. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds., vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May, 203–222. From the First ACM Workshop on Applied Computational Geometry.

SHEWCHUK, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22, 1-3*, 21–74.