Slovenská technická univerzita v Bratislave
Stavebná fakulta

Mgr. Marián Šagát

Autoreferát dizertačnej práce

Evolving manifolds in practical applications

na získanie akademického titulu *philosophiae doctor (PhD.)*
v doktorandskom študijnom programe
9.1.9 Aplikovaná matematika

Bratislava 2019

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia na Katedre matematiky a deskriptívnej geometrie SvF STU v Bratislave.

Predkladateľ: Mgr. Marián Šagát
Katedra matematiky a deskriptívnej geometrie
SvF STU, Bratislava

Školiteľ: Doc. Mgr. Mariana Remešíková,PhD.
Katedra matematiky a deskriptívnej geometrie
SvF STU, Bratislava

Oponenti: Prof. RNDr. Daniel Ševčovič, DrSc.
Dekan Fakulty matematiky, fyziky a informatiky
Univerzita Komenského v Bratislave

Doc. Mgr. Ivan Cimrák, Dr.
Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Doc. RNDr. Zuzana Krivá, PhD.
Katedra matematiky a deskriptívnej geometrie
SvF STU, Bratislava

Autoreferát bol rozoslaný dňa .....

Obhajoba dizertačnej práce sa koná dňa ..... o ..... hod. na Katedre matematiky a deskriptívnej geometrie SvF STU v Bratislave, Radlinského 11.

prof. Ing. Stanislav Unčík, PhD.
dekan Stavebnej fakulty

Abstract

This work deals with numerical solution of geodesic curvature flow of curves in the Lagrangian framework using a finite difference method leading to a semi-implicit numerical scheme. The primary focus is on solving the geodesic curvature flow of open curves and closed curves are mentioned marginally. For practical and numerical reasons, the basic model is enriched with a tangential term. Geodesic curvature flow is used for the approximation of open geodesies on surfaces, especially with an emphasis on discrete surfaces. The presented approximative method for finding open geodetics is experimentally compared with several known methods. It is also implemented in Rhinoceros CAD software. Along with other known methods, it is used to generate cutting patterns for tensile membrane structures in civil engineering. Many results and experiments are presented.

Keywords: geodesic, geodesic curvature flow, curve evolution, cutting patterns, tensile membrane structures, Rhinoceros software, Grasshopper plug-in, Grasshopper component

Abstrakt

Táto práca sa zaoberá numerickým riešenením vývoja kriviek podľa geodetickej krivosti v Lagrangeovskej formulácii na plochách pomocou metódy konečných diferencií vedúcim na semiimplicitnú numerickú schému. Prioritne sa zameriava na riešenie modelu vývoja otvorených kriviek podľa geodetikej krivosti, okrajovo sa venuje uzavretým krivkám. Z praktických a numerických dôvodov je model obohatený o tangenciálny člen. Tok podľa geodetickej krivosti sa používa na approximáciu otvorených geodetík na plochách, aj na hladkých, ale hlavne s dôrazom na diskrétne plochy. Prezentovaná aproximatívna metóda na hľadanie otvorených geodetík je experimentálne porovnávaná s viacerými známymi metódami. Je tiež implementovaná v Rhinoceros CAD softvére. Spolu s ďalšími známymi metódami je použitá na generovanie strihových plánov pre konštrukciu plachtových membránových štruktúr v stavebníctve. Je prezentovaných veľa experimentálnych výslekov.

Kľúčové slová: geodetika, tok podľa geodetickej krivosti, vývoj krivky, strihové plány, plachtové membránové štruktúry, Rhinoceros softvér, Grasshopper plug-in, Grasshopper komponent

Contents

Chapter 1

Introduction

A geodesic on a surface is defined as a curve with zero geodesic curvature in each point. Every shortest path on a surface is a geodesic. The problem of the extremal path, i.e. the geodesic problem, is occurring in many applications in different fields. In practice, geodesics occur either as smooth curves on smooth surfaces (in engineering mainly on NURBS surfaces) or as discrete curves on discrete surfaces (meshes).

One of the main sources of applications involving geodesics is, of course, industry, where the shortest paths are used for example in programming of CNC machines (6), filament winding technologies (32), or in examination of polymer material properties (5). Other industrial applications of geodesics are in manufacturing of fuselage of an aircraft, CNC machines (30), tent manufacturing, cutting and painting path finding, fiberglass tape windings in pipe manufacturing, textile manufacturing (15), ship design, NC machining (22). Another important field of application of geodesics is computer graphics, computer vision and CAD/CAM technologies (19), (14), (4), (31). Geodesics are of great importance in robotics (7), (40) and in geography related fields such as seismology (33) or geographical information systems (GIS) (4). Our main motivation comes from manufacturing of tensile membrane structures in civil engineering. An example of such a structure is shown in Figure 1.1.

Figure 1.1: An example of a tensile membrane structure, a tram station roof in Vienna, Austria.

Tensile membrane structures are in general designed as non-developable surfaces, however, in reality they are assembled from planar pieces of textile or other similar material. During the manufacturing process the CAD model of the membrane structure is cut into smaller elements that are then developed into planar pieces. The original curved elements can be, in general, developed only with a certain error and it is important to minimize the distortion between the curved element and its planar counterpart as much as possible. The planar elements are usually cut from a fabric roll of a relatively narrow width and in an efficient manufacturing process the waste of material should be minimized. Therefore, the elements to be cut should preferably have straight or only slightly curved borders. If an element with geodesic borders is narrow enough, after developing we get

a planar strip with almost straight borders. This is the main advantage of cutting along geodesics and it allows optimal usage of the material as well as easier dealing with the shear loading and aesthetic criteria. A more detailed insight in the cutting pattern generation problem and using geodesic cuts can be found for example in the paper (10) or other works by the same authors.

1.1 Related work

The problem of finding geodesics is widely studied for many years. Depending on the accuracy there are two types of methods which are used for finding the shortest paths on meshes: exact algorithms and approximation algorithms. The most of exact algorithms are using an unfolding technique. The idea of unfolding is based on the well-known property of the shortest paths (14) – if we have a geodesic path that connects two points along an edge sequence, then the planar unfolding of the path along this sequence is a straight line segment.

The largest groups of approximation methods are the group of differential equation methods and graph based methods.

The curvature flow of curves in the plane has been studied for many years by many researchers. The geodesic curvature flow on surfaces, also called curve shortening flow (39), has been studied less, but one can find papers with theoretical results (21), (8), (9), (24). The evolution is called curve shortening flow because the flow lines in the space of closed curves are tangent to the gradient of the length functional of the curve (9). When a curve is evolving by this flow, its length is monotonously strictly decreasing (8), (21). Grayson (21), (9), proved that any embedded curve during the curve shortening flow either shrinks to a point in a finite time or its curvature converges to zero in the C^∞ norm as $t \to \infty$.

The geodesic curvature flow can be mathematically formulated in several different ways. For example, in the work of Osher and Sethian (28), the geodesic curvature flow algorithms are based on Hamilton-Jacobi formulations. Probably the most popular approach with several practical advantages is the level set (otherwise called Eulerian) formulation (17), (39), where the evolving curve is considered to be a level set of a higher dimensional map. The geodesic curvature flow on parametric surfaces is studied in (36), (35). Another possibility is the direct (Lagrangian) approach that directly prescribes the evolution of each point of the curve. This approach has been elaborated for example in the papers by Mikula et al. (24), (27) that use the Lagrangian formulation of geodesic curvature flow. In the former paper, the authors reduce the dimension of the problem for closed curves by projection into a plane and in the latter paper they solve the problem by an explicit method for geodesic curvature flow of open curves on an analytically given surface. Barrett et al. (2) use the Lagrangian formulation of geodesic curvature flow for closed curves and present its parametric finite element approximations. They study more general flows, e.g. gradient flows of curves on manifolds, as well. Recently, in the work (3), the finite element method for evolving closed curves on triangulated surfaces has been explored. The authors use Lagrangian framework and solve topology changes of evolving closed curves, too.

Each of the approaches has its advantages and drawbacks. The authors of the paper (39) very well explain the differences between the Lagrangian and the Eulerian framework and some pros and cons. In order to model the geodesic curvature flow, they used the Eulerian framework which works particularly well for closed curve evolution and benefits from its ability to easily handle topology changes. On the other hand, it has difficulties with open curves and therefore they model only geodesic curvature flow of closed curves. Actually we found only one article (18) using the level set method for geodesic curvature flow of open curves, however, only for surfaces which can be injectively projected into plane and the problem is solved only on a rectangular grid. On the other hand, the Lagrangian framework handles both open and closed curves quite well. Moreover, the Lagrangian framework is a direct approach that does not need to add another dimension like the level set method. Therefore, we can expect the algorithms to be faster and also less demanding with respect to memory consumption. In contrast to the Eulerian framework, in the Lagrangian framework for closed curves it is necessary to solve the problem of changing the topology of the evolving curve in addition. This can be quite tricky, though efficient algorithms dealing with topology changes already exist at least for plane curves (1). Other drawbacks of the Lagrangian approach appear in the discrete setting. First, the points of the evolving discretized curve can deviate from the mesh on which they should be situated. Second, the points of a discrete curve can meet and form instabilities and self-crossings of the curve during the evolution. The first issue is

solved by back-projection to the mesh. The second issue is standardly solved by tangential redistribution of points. To the best of our knowledge, only the authors (24), (2), (3), (25) studied geodesic curvature flow of closed curves by the Lagrangian approach and related problems which arise from Lagrangian approach such as tangential redistribution of points and topology changes. The topic of tangential redistribution is solved in the works (20), (12), (13), too.

1.2 Overview of our work

We propose an algorithm based on numerical solution of the geodesic curvature flow in the Lagrangian setting called Discrete Langrangian Algorithm (DLA). We chose to apply the geodesic curvature flow, since, as we have said before, it shrinks the curve as fast as it can and it uses only local information. Thus the performance of the algorithm is independent of the size of the mesh, if we exclude the construction of the initial approximation from our consideration. With an appropriately set time discretization step, the algorithm quickly converges to an approximation of a geodesic that is accurate enough for practical purposes. To further motivate the use of a Lagrangian evolution, let us note that in our specific application of cutting pattern design, we practically deal only with open geodesics connecting two points. Moreover, if we choose a reasonable initial condition, no topology changes during the evolution are expected. These facts make the Lagrangian framework the approach of our first choice.

To overcome the disadvantages of the Lagrangian approach that remain, our procedure includes an algorithm for back-projecting the evolving curve to our mesh as well as a method for tangential redistribution of the discretization points during the evolution. For a better practical applicability, we designed our method so that it is able to deal not only with simple meshes but also with more challenging triangular surfaces, for example large meshes or the ones with sharp corners. We also propose a procedure for an automatic choice of time step, in case it is convenient.

In order to make our algorithm accessible to a broader community of users, we created a Grasshopper assembly library for Rhinoceros called MeshPaths (23) that includes our method for finding geodesics.

This work is organized as follows. Chapter 2 describes the analytical model of geodesic curvature flow and some of its features and it also describes the geodesic curvature flow model extended by a tangential term. The next Chapter 3 is devoted to the discrete model of geodesic curvature flow and the derivation of a numerical scheme. Chapter 4 briefly describes auxiliary algorithms and implementation details of our Discrete Lagrangian Algorithm (DLA). The last Chapter 5 presents chosen experimental results from thesis: comparisons DLA with FMM method and example of cutting pattern generation with comparison between geodesic and non-geodesic cutting patterns. The implementation for cutting patterns generations is made in Grasshopper as library called CuttingPatternDesign library.

Chapter 2

Geodesic Curvature Flow

2.1 Mathematical model

Let $\boldsymbol{\gamma} : I \to S$ be a smooth curve, where $I \subset \mathbb{R}$ is an interval, on a smooth surface S embedded in $\mathbb{R}^3$. If we denote by κ the curvature of the curve $\boldsymbol{\gamma}$ and by $\boldsymbol{N}$ its principal normal, then its curvature vector is $\boldsymbol{K} = \kappa \boldsymbol{N}$. Now let $\boldsymbol{N^V}(u)$ be the unit normal to the curve in the tangent space of S at the point $\boldsymbol{\gamma}(u)$. Then $\boldsymbol{N^V} = \boldsymbol{N^S} \times \boldsymbol{T}$, where $\boldsymbol{N^S}(u)$ is the unit normal to S at $\boldsymbol{\gamma}(u)$ and $\boldsymbol{T}$ is the unit tangent vector to the curve $\boldsymbol{\gamma}$. The geodesic curvature vector is defined as the projection of the curvature vector $\boldsymbol{K}$ in the direction of $\boldsymbol{N^V}$, which means

$$\boldsymbol{K_g} = (\boldsymbol{K} \cdot \boldsymbol{N^V})\boldsymbol{N^V}.$$

The scalar geodesic curvature is then defined as

$$K_g = \boldsymbol{K} \cdot \boldsymbol{N^V}.$$

Geodesic curvature flow is a motion of a curve evolving by geodesic curvature on a surface. Let

$$\boldsymbol{\gamma} = \boldsymbol{\gamma}(u,t) : I \times \langle 0, T \rangle \to S,\ I \subset R$$

be a curve evolving by geodesic curvature K_g on surface S embedded in R^3. The evolution of the curve by its geodesic curvature is described by

$$\partial_t \boldsymbol{\gamma} = K_g \boldsymbol{N^V}, \tag{2.1}$$

where $\boldsymbol{N^V}$ is the unit normal to the curve in the tangent space of S, $\boldsymbol{N^V} = \boldsymbol{N^S} \times \boldsymbol{T}$, $\boldsymbol{N^S}$ is the unit normal to S, $\boldsymbol{T}$ is a unit tangent vector to the curve $\boldsymbol{\gamma}$. This is a motion only in the normal direction. For our purposes we will consider a model with an additional tangential term

$$\partial_t \boldsymbol{\gamma} = K_g \boldsymbol{N^V} + \alpha \boldsymbol{T}. \tag{2.2}$$

If we consider an evolving closed curve, the model (2.1) or (2.2) is accompanied by the periodic boundary condition, otherwise we consider the Dirichlet boundary condition, i.e. fixed boundary points.

2.2 Curve-shortening property of the geodesic curvature flow

A curve evolving by geodesic curvature is monotonously becoming shorter during the evolution. This shortening property of geodesic curvature flow is crucial for us, because we want to use the geodesic curvature flow for finding the shortest path on the surfaces. The proof that the length $L = L(t) = L(\boldsymbol{\gamma}(t))$ is monotonously strictly decreasing in time can be found in (8), (21). We prove it in a different way, we find the expression for the length of an arbitrary evolving curve on surface and the length of a curve evolving by geodesic curvature is only a special case.

Theorem 2.1. *The geodesic curvature flow of a curve strictly monotonously decreases its length until the flow*

stops, i.e.

$$L_t < 0,\ \forall t \in \langle 0, T\rangle.$$

Proof. Without any loss of generality we can consider only the motion of a curve on a surface in the normal direction to the curve $\boldsymbol{N^V}$, because the tangential motion does not affect the length of the curve, supposed that the boundary points are fixed. Let the curve $\boldsymbol{\gamma} = \boldsymbol{\gamma}(u,t)$ evolve on the surface S with a unit normal $\boldsymbol{N^S}$. Let us denote the tangential vector of the curve $\boldsymbol{\gamma}$ by $\boldsymbol{T}$, the geodesic curvature vector by $\boldsymbol{K_g}$. Standardly $\left\|\boldsymbol{N^S}\right\| = \|\boldsymbol{T}\| = 1$. Let the curve move on the surface by the equation

$$\boldsymbol{\gamma}_t = \beta \boldsymbol{N^V}. \tag{2.3}$$

By differentiating $\|\boldsymbol{\gamma}_u\|^2 = \boldsymbol{\gamma}_u \cdot \boldsymbol{\gamma}_u$ with respect to the time variable we get

$$2\,\|\boldsymbol{\gamma}_u\| \cdot \|\boldsymbol{\gamma}_u\|_t = \boldsymbol{\gamma}_{ut} \cdot \boldsymbol{\gamma}_u + \boldsymbol{\gamma}_u \cdot \boldsymbol{\gamma}_{ut},$$

so we obtain

$$\|\boldsymbol{\gamma}_u\|_t = \boldsymbol{\gamma}_{ut} \cdot \frac{\boldsymbol{\gamma}_u}{\|\boldsymbol{\gamma}_u\|} = \boldsymbol{\gamma}_{ut} \cdot \boldsymbol{T}. \tag{2.4}$$

The length $L = L(t)$ of the curve is equal to $\int\limits_I \|\boldsymbol{\gamma}_u\|\,\mathrm{d}u$, so its time derivative is

$$L_t = \int\limits_I \|\boldsymbol{\gamma}_u\|_t\,\mathrm{d}u.$$

Therefore we need to express $\|\boldsymbol{\gamma}_u\|_t$. From (2.3) we get

$$\boldsymbol{\gamma}_{ut} = \boldsymbol{\gamma}_{tu} = (\beta \boldsymbol{N^V})_u$$

and

$$\begin{aligned} \|\boldsymbol{\gamma}_u\|_t = \boldsymbol{\gamma}_{ut} \cdot \boldsymbol{T} = (\beta \boldsymbol{N^V})_u \cdot \boldsymbol{T} = (\beta_u \boldsymbol{N^V} + \beta \boldsymbol{N^V_u}) \cdot \boldsymbol{T} = \\ = \beta_u \boldsymbol{N^v} \cdot \boldsymbol{T} + \beta \boldsymbol{N^V_u} \cdot \boldsymbol{T} = \beta \boldsymbol{N^V_u} \cdot \boldsymbol{T}, \end{aligned} \tag{2.5}$$

since $\boldsymbol{N^V} \perp \boldsymbol{T}$. Using

$$\boldsymbol{N^V_u} = (\boldsymbol{N^S} \times \boldsymbol{T})_u = \boldsymbol{N^S_u} \times \boldsymbol{T} + \boldsymbol{N^S} \times \boldsymbol{T}_u$$

we can modify (2.5) to

$$\|\boldsymbol{\gamma}_u\|_t = \beta \boldsymbol{N^V_u} \cdot \boldsymbol{T} = \beta(\boldsymbol{N^S_u} \times \boldsymbol{T} \cdot \boldsymbol{T} + \boldsymbol{N^S} \times \boldsymbol{T}_u \cdot \boldsymbol{T}) = \beta \boldsymbol{N^S} \times \boldsymbol{T}_u \cdot \boldsymbol{T}, \tag{2.6}$$

where

$$\boldsymbol{T}_u = \boldsymbol{T}_s \cdot s_u = \left| s = \int \|\boldsymbol{\gamma}_u\|\,\mathrm{d}u \Rightarrow s_u = \|\boldsymbol{\gamma}_u\| \right| = \boldsymbol{T}_s \cdot \|\boldsymbol{\gamma}_u\|$$

and from Frenet–Serret formulas we get $\boldsymbol{T}_s = k\boldsymbol{N} = \boldsymbol{K}$ and therefore $\boldsymbol{T}_u = \|\boldsymbol{\gamma}_u\|\,\boldsymbol{K}$. The relationship (2.6) can be modified in following way

$$\begin{aligned} \|\boldsymbol{\gamma}_u\|_t = \beta(\boldsymbol{N^S} \times \boldsymbol{T}_u \cdot \boldsymbol{T}) = \beta(\boldsymbol{N^S} \times (\|\boldsymbol{\gamma}_u\|\,\boldsymbol{K})) \cdot \boldsymbol{T} = \\ = \beta\,\|\boldsymbol{\gamma}_u\|\,\boldsymbol{T} \cdot (\boldsymbol{N^S} \times \boldsymbol{K}) = \beta\,\|\boldsymbol{\gamma}_u\|\,\boldsymbol{N^S} \cdot (\boldsymbol{K} \times \boldsymbol{T}) = \beta\,\|\boldsymbol{\gamma}_u\|\,\boldsymbol{K} \cdot (\boldsymbol{T} \times \boldsymbol{N^S}) = \\ = \beta\,\|\boldsymbol{\gamma}_u\|\,\boldsymbol{K} \cdot (-\boldsymbol{N^V}) = -\beta\,\|\boldsymbol{\gamma}_u\|\,\|\boldsymbol{K_g}\|. \end{aligned}$$

Finally

$$L_t = \int\limits_I \|\boldsymbol{\gamma}_u\|_t\,\mathrm{d}u = -\int\limits_I \beta\,\|\boldsymbol{K_g}\|\,\|\boldsymbol{\gamma}_u\|\,\mathrm{d}u = -\int\limits_I \beta K_g\,\mathrm{d}s.$$

Specifically for the geodesic curvature flow i.e. $\beta = K_g$ it holds

$$L_t = -\int\limits_I K_g^2\,\mathrm{d}s < 0 \tag{2.7}$$

for $K_g \neq 0$. Equation (2.7) means that the length of a curve evolving by geodesic curvature is strictly decreasing during evolution and the evolution stops, when $\boldsymbol{K_g} \equiv 0$. □

2.3 Tangential redistribution of points of a curve

Using only the normal evolution of curve points, some difficulties can arise after discretization – the distance between the discretization points can become too small or even self-intersections of the discretized curve can appear. This can lead to instabilities or crossings. Therefore, for practical purposes, we use the extended model (2.2) with an additional tangential term. We develop a formula for the coefficient $\alpha = \alpha(u,t)$ in (2.2). In the papers (26), (24) there are derived equations for a redistribution of points of evolving curves. We use the same ideas and techniques for our model here and derive equations for tangential redistribution of points of a curve evolving by its geodesic curvature. Let the relative length (ratio of the local length and the length) be

$$\frac{\|\boldsymbol{\gamma}_u\|(u,t)}{L(t)} = f(u,t).$$

Then $\|\boldsymbol{\gamma}_u\| = f \cdot L$ and after differentiating we get

$$\|\boldsymbol{\gamma}_u\|_t = f_t \cdot L + f \cdot L_t. \tag{2.8}$$

In Section 2.2 We have already found the formula for $\|\boldsymbol{\gamma}_u\|_t$ for the evolution without tangential motion. Therefore we get this formula for case with the tangential motion easily. Since

$$(\alpha \boldsymbol{T})_u \cdot \boldsymbol{T} = (\alpha_u \boldsymbol{T} + \alpha \boldsymbol{T}_u) \cdot \boldsymbol{T} = \alpha_u + \alpha \boldsymbol{T}_u \cdot \boldsymbol{T} = \alpha_u + \alpha \|\boldsymbol{\gamma}_u\| \boldsymbol{K} \cdot \boldsymbol{T} = \alpha_u$$

we have

$$\|\boldsymbol{\gamma}_u\|_t = -\beta \|\boldsymbol{\gamma}_u\| \|\boldsymbol{K_g}\| + \alpha_u = -\|\boldsymbol{\gamma}_u\| K_g^2 + \alpha_u. \tag{2.9}$$

Using the fact that $\alpha_u = \alpha_s \cdot s_u = \alpha_s \cdot \|\boldsymbol{\gamma}_u\|$ we get

$$\|\boldsymbol{\gamma}_u\|_t = -\|\boldsymbol{\gamma}_u\| K_g^2 + \alpha_s \cdot \|\boldsymbol{\gamma}_u\|. \tag{2.10}$$

By comparing the equations (2.8) and (2.10) we get the equation

$$-\|\boldsymbol{\gamma}_u\| K_g^2 + \alpha_s \cdot \|\boldsymbol{\gamma}_u\| = f_t \cdot L + f \cdot L_t. \tag{2.11}$$

Further, $\dfrac{f(t)}{\|\boldsymbol{\gamma}_u\|} = \dfrac{1}{L(t)}$ and from the equation (2.11) we get

$$\alpha_s = K_g^2 + \frac{L_t}{L} + \frac{f_t}{f}. \tag{2.12}$$

We choose the relative length $f(t)$ in a similar manner as the authors in (26) $f(t) = \mathrm{e}^{-k_0 t} + 1$, which guarantees the asymptotically uniform redistribution of points. Here k_0 is a constant coefficient of the redistribution speed. Then

$$f_t = \mathrm{e}^{-k_0 t}(-k_0) = (f-1) \cdot (-k_0) = (1-f)k_0$$

and

$$\frac{f_t}{f} = \left(\frac{1}{f} - 1\right) k_0 = \left(\frac{L}{\|\boldsymbol{\gamma}_u\|} - 1\right) k_0. \tag{2.13}$$

Substituting (2.13) into the equation (2.12) and using the expression for L_t, which we have found in Section (2.2) we get

$$\alpha_s = K_g^2 - \frac{1}{L} \int_I K_g^2 \, \mathrm{d}s + \left(\frac{L}{\|\boldsymbol{\gamma}_u\|} - 1\right) k_0. \tag{2.14}$$

Equation (2.14) is formally the same as in the works (27), (26), but with geodesic curvature.

Chapter 3

Discrete Geodesic Curvature Flow

In order to construct a numerical scheme for solving the equations (2.1) and (2.2), we choose a semi-implicit time discretization and a finite difference space discretization approach. The semi-implicit scheme is more stable than the explicit scheme and its advantage with respect to the fully implicit scheme is that it leads to a linear system of equations instead of non-linear.

We start by describing the discretization of the basic model (2.1) with no tangential redistribution and the tangential redistribution term and its discrete version will be added later.

Time discretisation First, let us discretize the equation (2.1) in time. For this purpose, let us use a uniform discretization with time step h_τ. The principle of our semi-implicit approximation is that the curvature vector $\boldsymbol{K}$ is taken from the new time step, while the normal $\boldsymbol{N}^{\boldsymbol{V}}$ is taken from the actual time step. This choice is made in order to be able to use the already known values of non-linear terms. The discretized equation has the form

$$\frac{\boldsymbol{\gamma}^{j+1}-\boldsymbol{\gamma}^{j}}{h_\tau}=\left(\boldsymbol{K}^{j+1}\cdot\left(\boldsymbol{N}^{\boldsymbol{V}}\right)^{j}\right)\left(\boldsymbol{N}^{\boldsymbol{V}}\right)^{j}, \tag{3.1}$$

where $\boldsymbol{\gamma}^j$ is the approximation of $\boldsymbol{\gamma}(\cdot, jh_\tau)$ and $\boldsymbol{K}^{j+1}$, $\left(\boldsymbol{N}^{\boldsymbol{V}}\right)^j$ are understood analogously.

Space discretisation Now let us assume that the surface S is approximated by a triangular mesh $\mathfrak{T}(S)$ and let us also discretize the curve $\boldsymbol{\gamma}^j$ by $n+2$ nodal points $\boldsymbol{\gamma}_i^j \approx \boldsymbol{\gamma}(u_i,t_j)$, $i=0,\ldots,n+1$. The values $\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_{n+1}$ are determined according to the boundary conditions, the rest of the points will be the uknowns of the resulting linear system. In our approximation scheme, we will also use the points

$$\boldsymbol{\gamma}_{i-\frac{1}{2}}^j=\frac{\boldsymbol{\gamma}_{i-1}^j+\boldsymbol{\gamma}_i^j}{2}, \qquad \boldsymbol{\gamma}_{i+\frac{1}{2}}^j=\frac{\boldsymbol{\gamma}_{i+1}^j+\boldsymbol{\gamma}_i^j}{2}.$$

The curvature vector at the point (u_i,t_j) is defined as

$$\boldsymbol{K}(u_i,t_j)=\frac{\partial \boldsymbol{T}}{\partial s}(u_i,t_j),$$

where s denotes the arc-length of the curve. Therefore we can take the approximation

$$\boldsymbol{K}_i^{j+1}=\frac{\boldsymbol{T}_{i+1/2}^{j+1}-\boldsymbol{T}_{i-1/2}^{j+1}}{\frac{h_{i+1}^j+h_i^j}{2}}, \tag{3.2}$$

where we use approximations of the unit tangent vectors in the form

$$\boldsymbol{T}_{i+1/2}^{j+1}=\frac{\boldsymbol{\gamma}_{i+1}^{j+1}-\boldsymbol{\gamma}_i^{j+1}}{h_{i+1}^j}, \qquad \boldsymbol{T}_{i-1/2}^{j+1}=\frac{\boldsymbol{\gamma}_i^{j+1}-\boldsymbol{\gamma}_{i-1}^{j+1}}{h_i^j}, \tag{3.3}$$

with $h_i^j=\left\|\boldsymbol{\gamma}_i^j-\boldsymbol{\gamma}_{i-1}^j\right\|$. In order to approximate the normal $\boldsymbol{N}^{\boldsymbol{V}}(u_i,t_j)$, we first need to construct an approx-

imation of the surface normal $\boldsymbol{N^S}$ in the point $\boldsymbol{\gamma}(u_i,t_j)$. If the point $\boldsymbol{\gamma}_i^j$ happens to be a vertex of the mesh, the corresponding surface normal is approximated by the arithmetic mean of the normals of the neighboring triangles. If the point lies on an edge of the mesh, we use the normal of one of the neighboring triangles. The cases when the angle between these two triangles is too large are treated separately as described in Section 4. However, we have to mention that except for the initial condition that usually consists of mesh vertices, the points of the evolving curve are (after projection) almost always situated inside a mesh triangle. In this case, naturally, $\left(\boldsymbol{N^S}\right)_i^j$ is the normal of this triangle. Having approximated $\boldsymbol{N^S}$, we take

$$\left(\boldsymbol{N^V}\right)_i^j = \left(\boldsymbol{N^S}\right)_i^j \times \boldsymbol{T}_i^j, \qquad \boldsymbol{T}_i^j = \frac{\boldsymbol{\gamma}_{i+1}^j - \boldsymbol{\gamma}_{i-1}^j}{\left\|\boldsymbol{\gamma}_{i+1}^j - \boldsymbol{\gamma}_{i-1}^j\right\|}.$$

Now, plugging the formulas (3.2) and (3.3) into the equation (3.1), we obtain

$$\frac{h_{i+1}^j + h_i^j}{2}\ \frac{\boldsymbol{\gamma}_i^{j+1} - \boldsymbol{\gamma}_i^j}{h_\tau} = \left\{ \left[\frac{\boldsymbol{\gamma}_{i+1}^{j+1} - \boldsymbol{\gamma}_i^{j+1}}{h_{i+1}^j} - \frac{\boldsymbol{\gamma}_i^{j+1} - \boldsymbol{\gamma}_{i-1}^{j+1}}{h_i^j} \right] \cdot \left(\boldsymbol{N^V}\right)_i^j \right\} \left(\boldsymbol{N^V}\right)_i^j. \tag{3.4}$$

Fully discrete formulation In order to provide a fully discrete formulation of our evolution model, we use an auxiliary notation

$$\boldsymbol{\gamma}_{i-1}^{j+1} = \boldsymbol{a}, \qquad \boldsymbol{\gamma}_i^{j+1} = \boldsymbol{b}, \qquad \boldsymbol{\gamma}_{i+1}^{j+1} = \boldsymbol{c}, \qquad \left(\boldsymbol{N^V}\right)_i^j = \boldsymbol{d}, \tag{3.5}$$

where $\boldsymbol{a} = (a_1, a_2, a_3)$. We consider all vectors in this section to be column vectors. After using this notation in (3.4) we get

$$\frac{h_{i+1}^j + h_i^j}{2}\ \frac{\boldsymbol{b} - \boldsymbol{\gamma}_i^j}{h_\tau} = \left\{ \left[\frac{\boldsymbol{c} - \boldsymbol{b}}{h_{i+1}^j} - \frac{\boldsymbol{b} - \boldsymbol{a}}{h_i^j} \right] \cdot \boldsymbol{d} \right\} \boldsymbol{d}. \tag{3.6}$$

After a simplification we have

$$\left[\frac{\boldsymbol{c} - \boldsymbol{b}}{h_{i+1}^j} - \frac{\boldsymbol{b} - \boldsymbol{a}}{h_i^j} \right] \cdot \boldsymbol{d} = \frac{\boldsymbol{c} \cdot \boldsymbol{d}}{h_{i+1}^j} + \left(-\frac{1}{h_{i+1}^j} - \frac{1}{h_i^j} \right) \boldsymbol{b} \cdot \boldsymbol{d} + \frac{\boldsymbol{a} \cdot \boldsymbol{d}}{h_i^j}. \tag{3.7}$$

We can use the identity

$$(\boldsymbol{c} \cdot \boldsymbol{d})\, \boldsymbol{d} = \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{c}$$

and rewrite the system (3.6) in the the form

$$\omega[\boldsymbol{b} - \boldsymbol{\gamma}_i^j] = \omega_+ \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{c} + (-\omega_+ - \omega_-) \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{b} + \omega_- \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{a}, \tag{3.8}$$

where we used new variables

$$\omega_- = \frac{1}{h_i^j}, \qquad \omega_+ = \frac{1}{h_{i+1}^j}, \qquad \omega = \frac{h_{i+1}^j + h_i^j}{2h_\tau}. \tag{3.9}$$

After a rearrangement we get the system

$$\omega \boldsymbol{b} - \omega_- \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{a} + (\omega_+ + \omega_-) \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{b} - \omega_+ \boldsymbol{d} \cdot \boldsymbol{d}^T \cdot \boldsymbol{c} = \omega \boldsymbol{\gamma}_i^j. \tag{3.10}$$

Now, returning to the original indexed notation according to (3.5) and denoting $\boldsymbol{d}\boldsymbol{d}^T = D_i^j$, we get a compact block three-diagonal matrix form of the linear system that we have to solve. The i-th block row is of the form

$$\left(A_i^j \mid B_i^j \mid C_i^j \right) \begin{pmatrix} \boldsymbol{\gamma}_{i-1}^{j+1} \\ \hline \boldsymbol{\gamma}_i^{j+1} \\ \hline \boldsymbol{\gamma}_{i+1}^{j+1} \end{pmatrix} = \omega \boldsymbol{\gamma}_i^j, \tag{3.11}$$

where

$$A_i^j = -\omega_- \, D_i^j, \qquad B_i^j = \, \omega I + (\omega_+ + \omega_-) \, D_i^j, \qquad C_i^j = -\omega_- \, D_i^j. \tag{3.12}$$

For a closed curve, the periodic boundary conditions $\boldsymbol{\gamma}_0^j = \boldsymbol{\gamma}_n^j$ and $\boldsymbol{\gamma}_{n+1}^j = \boldsymbol{\gamma}_1^j$ are added to the discretized system. In the case of an open curve, the model is accompanied by Dirichlet boundary conditions and two unknowns are excluded using $\boldsymbol{\gamma}_0^j = \boldsymbol{\gamma}_L$, $\boldsymbol{\gamma}_{n+1}^j = \boldsymbol{\gamma}_R$, where $\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R$ are fixed points – the endpoints of the geodesic.

Discretization of the tangential term The tangential term $\alpha \boldsymbol{T}$ is approximated as

$$\alpha(u_i,t_j)\boldsymbol{T}(u_i,t_j) \approx \alpha_i^j \boldsymbol{T}_i^j = \alpha_i^j \, \frac{\boldsymbol{\gamma}_{i+1}^j - \boldsymbol{\gamma}_{i-1}^j}{h_{i+1}^j + h_i^j},$$

where α_i^j is obtained from the discretized version of (2.14) that reads

$$\alpha_{i+1}^j = \alpha_i^j + \left(\left(K_g^2\right)_i^j - \frac{1}{L^j} \sum_{m=1}^{n+1} \left(K_g^2\right)_m^j h_m^j \right) h_i^j + \left(\frac{L^j}{n+1} - h_i^j \right) k_0. \tag{3.13}$$

Here we set $\alpha_0^j = 0$ and

$$L^j = \sum_{i=1}^{n+1} h_i^j$$

is the approximation of the total length of the curve $\boldsymbol{\gamma}^j$.

The tangential speed in the form (3.13), being based on a rigorous mathematical reasoning, guarantees that the space discretization grid approaches a uniform discretization as the curve evolves. However, the formula requires a non-negligible amount of arithmetic operations in each time step. Now, taking in account our specific situation, we can consider using a simplified version of (3.13). Let us recall that as our curve evolves, its geodesic curvature converges to zero. Moreover, in practical applications, the initial condition for the evolution model is usually a curve, which is already quite close to the searched geodesic (most often the shortest path found by the Dijkstra algorithm). That means we start with a curve whose geodesic curvature is quite close to zero. If this is the case and if our primary interest is a fast computation, we can approximate the tangential speed α by omitting the terms containing K_g, that means we can use

$$\alpha_{i+1}^j = \alpha_i^j + \left(\frac{L^j}{n+1} - h_i^j \right) k_0. \tag{3.14}$$

Numerous practical experiments with open curves that we performed show that the formula (3.14) is sufficient in a lot of situations.

By addition of the tangential term to our model, we get a linear system of the form (3.11) with almost the same coefficients. The only change takes place on the right-hand side, namely it is equal to

$$\omega \boldsymbol{\gamma}_i^j + \alpha_i^j \left(\boldsymbol{\gamma}_{i+1}^j - \boldsymbol{\gamma}_{i-1}^j \right) /2,$$

where α_i^j is the redistribution speed expressed by either (3.13) or (3.14).

The same consideration can be made for closed curves. Let the discretized closed curve have n nodes. Then the formula (3.14) is in the form:

$$\alpha_{i+1}^j = \alpha_i^j + \left(\frac{L^j}{n} - h_i^j \right) k_0. \tag{3.15}$$

Chapter 4

Implementation Details of Discrete Lagrangian Algorithm

We will briefly mention some implementation details. For efficiency, we used C ++ programming language with STL libraries. We use the Dijkstra algorithm to determine the initial approximation for our DLA algorithm. To achieve its effective implementation, we used the structure the so-called container adapter `std::priority_queue` and created a structure for the triangulated surface to get neighbours to the given vertex immediately. Another condition which we require from the created structure for the triangulated surface was to immediately get the vertices incident to a given edge, which in turn ensures the efficiency of the PTA algorithm that we use to project of evolving curve to the surface.

We use the BiCGStab (biconjugate gradient stabilized) method (38) for solving the linear system (3.11) of our DLA method. The PTA algorithm is described in detail in the thesis. Since the points of two consecutive iterations of the evolving curve are relatively close to each other on the triangulated surface, PTA proves to be a suitable algorithm for obtaining an unknown point position as well as its projection on the triangulated surface. By back-projection of the points of the evolving curve, we have ensured that its points lie on the surface, but the lines of the adjacent points of the curve can still lie outside the surface. In order to make comparisons on triangulated surfaces with known methods, it is essential that the nodes of the discretized curve lie on the edges of the triangulated surface. Again, we used PTA algorithm for it. The comparisons of our DLA algorithm with several known algorithms are presented in the thesis in detail.

For practical reasons, it is necessary to have an automatic choice of the time step. Based on the condition of diagonal dominance of the system matrix 3.12 of our DLA algorithm, we have estimated the time step h_τ^j by formula

$$h_\tau^j = \min_i \left(\frac{\sqrt{2}}{4} h_{i+1}^j h_i^j \right) . \tag{4.1}$$

In our experiments, we chose a uniform time step calculated from the matrix in the first iteration, i.e. $h_\tau = h_\tau^0$. A large number of practical experiments have shown that we can use at least 100 times bigger the time step than the one determined by the formula (4.1).

We chose the stopping criterion as follows. The geodetic curvature flow stops when the average absolute value of the geodesic curvature at the points of the evolving curve falls below a certain threshold of ε or if the number of iterations reaches the chosen maximum value of N_{max}.

In addition to the C ++ implementation, we have also made an implementation in C# programming language for the Rhinoceros Grasshopper plug-in. Grasshopper is a graphical editor for Rhinoceros software. It can also be interpreted as a visual programming language where components represent objects and algorithms themselves. The program is created by joining these components and filling inputs for these components. An example of such a program can be seen in Figure 4.1, where our geodetic component is visualized on the right hand side and a discretized surface along with computed geodesics on the left side of figure. We implemented DLA as two components, one for Mesh, which is a Rhinoceros representation of the triangulated surface and one for Surface, which represents the analytically given surfaces, of which the largest class is

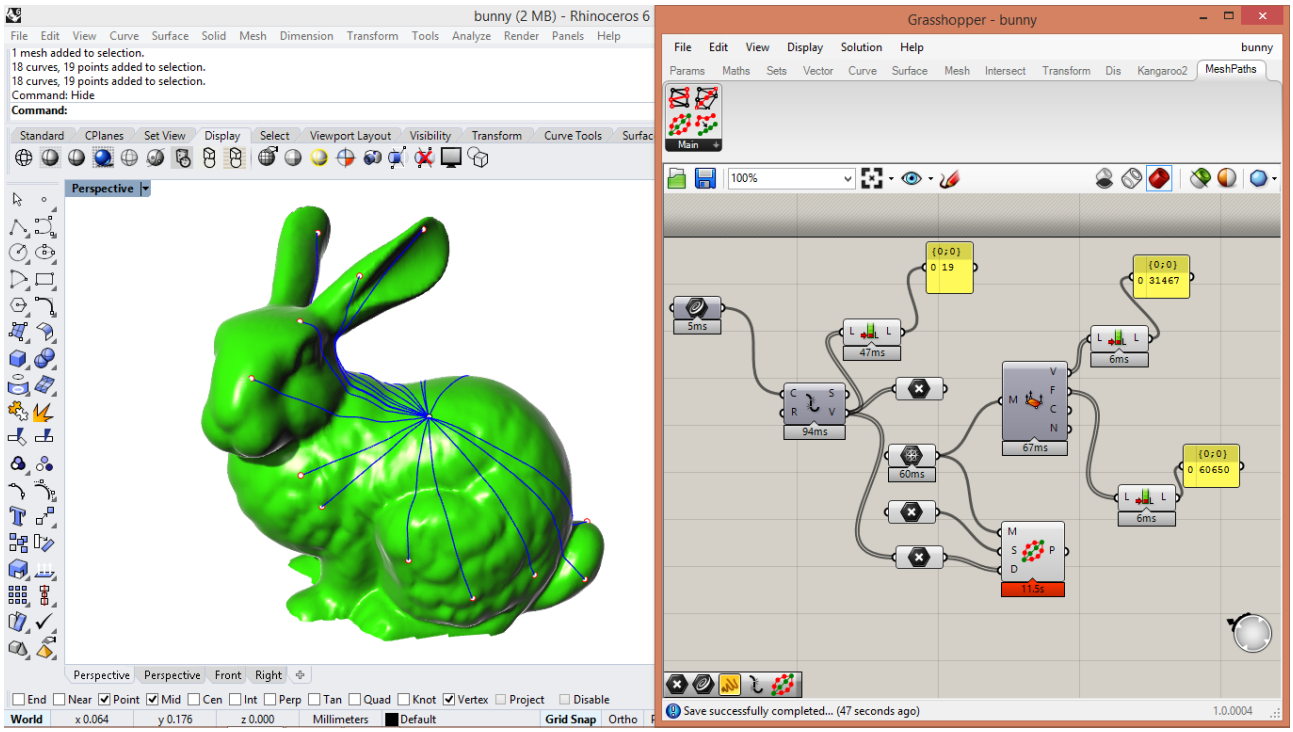

Figure 4.1: An a example of a surface visualization in Rhinoceros software (left) and Grasshopper graphic user interface with visual programming components (right).

NURBS. Both are part of the MeshPaths Grasshopper Library, which is presented along with documentation and examples in (23). Because the implementation is in C# and for Rhinoceros software, there are major implementation differences compared to the original C ++ implementation. In this implementation, we use `Rhino.Geometry.Mesh`, which is a structure that represents a triangulated surface in Rhinoceros software, and it is possible to use all known operations on it, such as looking for neighbours for a given vertex, etc. We use the Mesh.ClosestMeshPoint method on the Mesh structure instead of the PTA algorithm. The implementation of DLA for analytically defined surfaces, i.e. the implementation of a component on the Surface structure, differs mainly in the choice of initial iteration and Surface is used instead of Mesh. An example of geodesics created by our DLA component can be seen in Figures 4.2, 4.3.

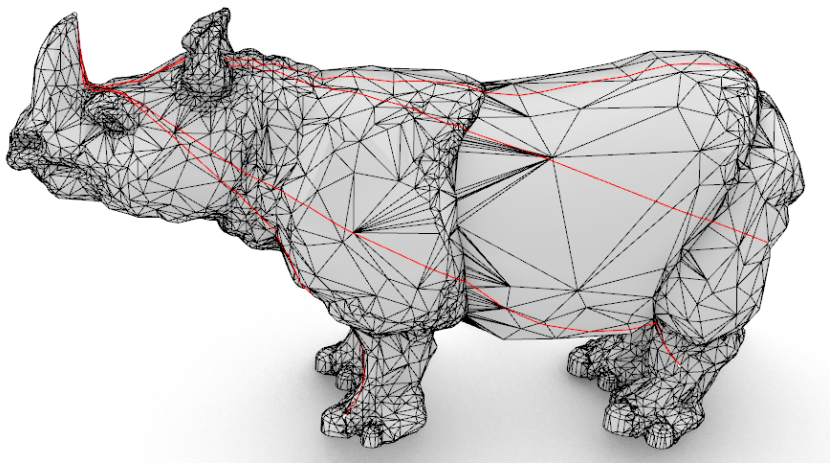

Figure 4.2: Geodesics on the surface with the highlighted irregular triangulated surface.

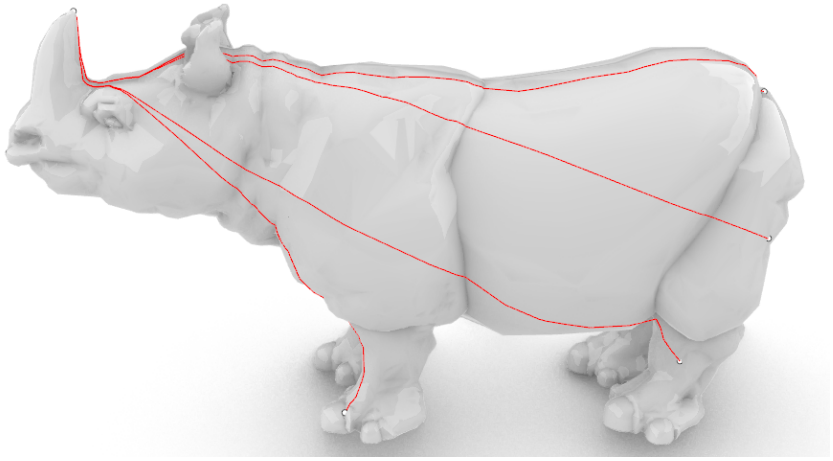

Figure 4.3: Geodesics on the same surface as in Figure 4.2 without displayed edges of the triangulated surface.

Chapter 5

Experiments and results

In this chapter we present results of several experiments. All computations were performed on a Lenovo B50 machine with Intel(R) Pentium(R) CPU N3540 2.16GHz processor, 8GB RAM and a 64-bit operation system.

5.1 Comparison with the Fast Marching Method

In order to demonstrate the efficiency and performance of our DL algorithm, we provide a comparison with the algorithm by Kimmel and Sethian (16) based on the Fast Marching Method (FMM). The FMM is one of the most popular methods for numerical computation of distance functions. The distance function to a certain point can be used to backtrack geodesic paths beginning in that point, which is exactly what the authors of the mentioned paper do.

We used a Matlab implementation of FMM from Matlab Toolbox Fast Marching (29). As for the DLA, we used the tangential redistribution model with the simplified tangential speed (3.14). The initial condition for the method was the Dijsktra shortest path and the time step was set automatically by the procedure described in Chapter 4. The endpoints of the geodesics used for testing were generated randomly. Each experiment has a detailed description providing the size of the mesh, the number of computed geodesics, the CPU time needed for the computations (these values are also listed in Table 5.1) and, of course, the evaluation of the results. We also list the CPU times needed to initialize the mesh structure in table. As we have shown in thesis, back-projection of curve points and mesh structure initialisation took only a small portion of the overall CPU time, up to about 3% altogether.

First, we performed experiments on a triangulated elephant surface with 24955 vertices and 49918 faces, which is a sample mesh in Matlab Toolbox Fast Marching (29). The computed geodesics are shown are shown in Figure 5.1 and we can see that DLA and FMM give visually almost the same approximations of geodesics. As for the computational time, the computation by DLA including the construction of the initial conditions took 9.74 seconds and for FMM algorithm it lasted 10.73 seconds. The stopping threshold ε was set to $\varepsilon = 0.002$. This value was chosen since it provided results of a quality comparable to FMM, which specifically means that the lengths of the approximate geodesics were about the same as those obtained by FMM. Actually, the lengths of the approximations of geodesics computed by DLA were shorter than the FMM approximations in 88 out of 100 experiments; however, the relative differences of the lengths are less than 1% in all cases, see the graph in Figure 5.2. The PTA algorithm was called 136941 times and exceptions occurred in 107 cases, i.e. the brute force searching algorithm was called 107 times. The tangential redistribution speed for this experiment was set to $k_0 = 100.0$. Second, we performed experiments on a triangulated torus, see Figure 5.3. And the third comparison was made on the triangular mesh of Stanford bunny. The results are presented in Figure 5.4. Other data from these two experiments with the same meaning as in the first experiment are presented in the table 5.1. To provide a complete image of the performance of our algorithm, we also note that in all experiments how long the mesh initialization took, see the mesh DLA column of table 5.1.

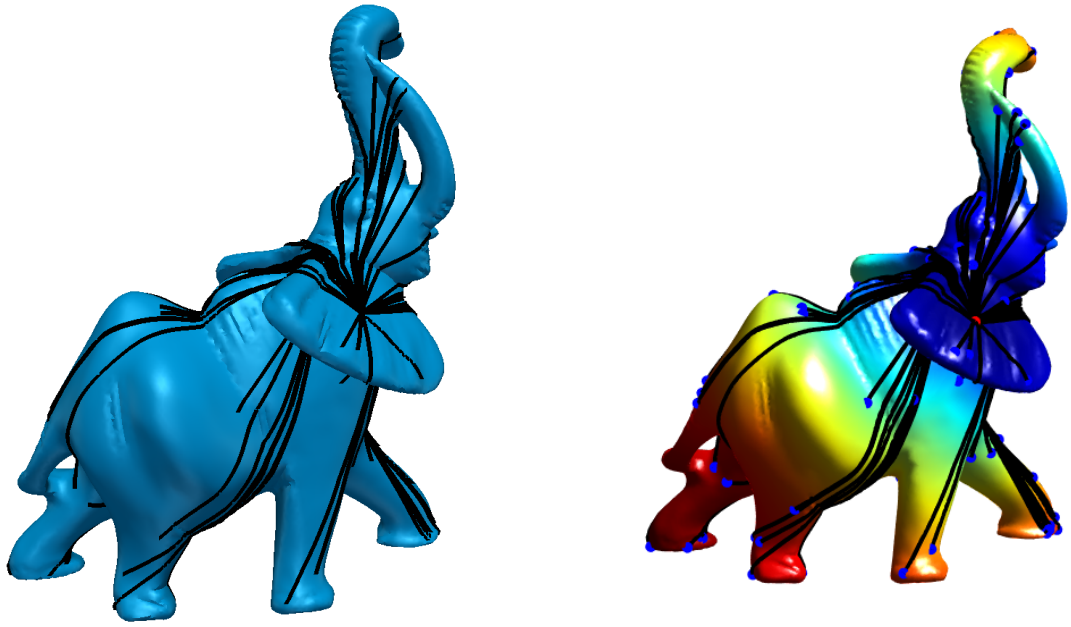

Figure 5.1: The approximate geodesics computed by the DLA (left) and FMM (right) algorithms on the elephant mesh.

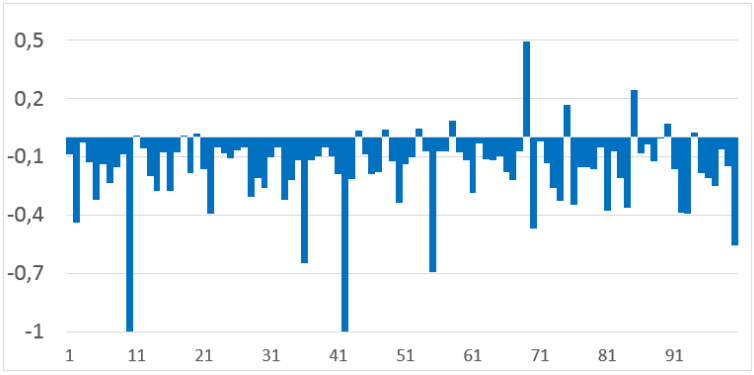

Figure 5.2: Comparison of the lengths of the approximate geodesics obtained by DLA and FMM for the elephant mesh. The relative difference of the DLA and FMM length $(100 * (L_{DLA} - L_{FMM})/L_{FMM})$ is depicted for each of the 100 computed approximations of geodesics.

mesh	vertices	faces	mesh DLA	ε	k_0	CPU DLA	CPU FMM	$L_{DLA} < L_{FMM}$
elephant	24955	49918	0.182 s	0.002	100.0	9.74 s	10.73 s	88/100
torus	15045	30090	0.039 s	0.002	1.0	5.82 s	6.56 s	99/100
bunny	31467	60650	0.196 s	0.002	100.0	11.25 s	11.41 s	68/91

Table 5.1: Comparison of DLA with FMM. The table shows the computation time needed by both algorithms and the number of approximate geodesics computed by DLA that were shorter than their FMM counterparts.

To conclude, we would say that our algorithm provides results of about the same quality (with respect to speed and accuracy) as the fast marching method. As for the difficulty of implementation, we would also say that both algorithms are at about the same level. For sure, both methods could be still tuned for ideal values of parameters for each test and maybe both implementations could be optimized even more. However, we

think the presented results demonstrate that the Discrete Lagrangian Algorithm is a good alternative to the Fast Marching Method.

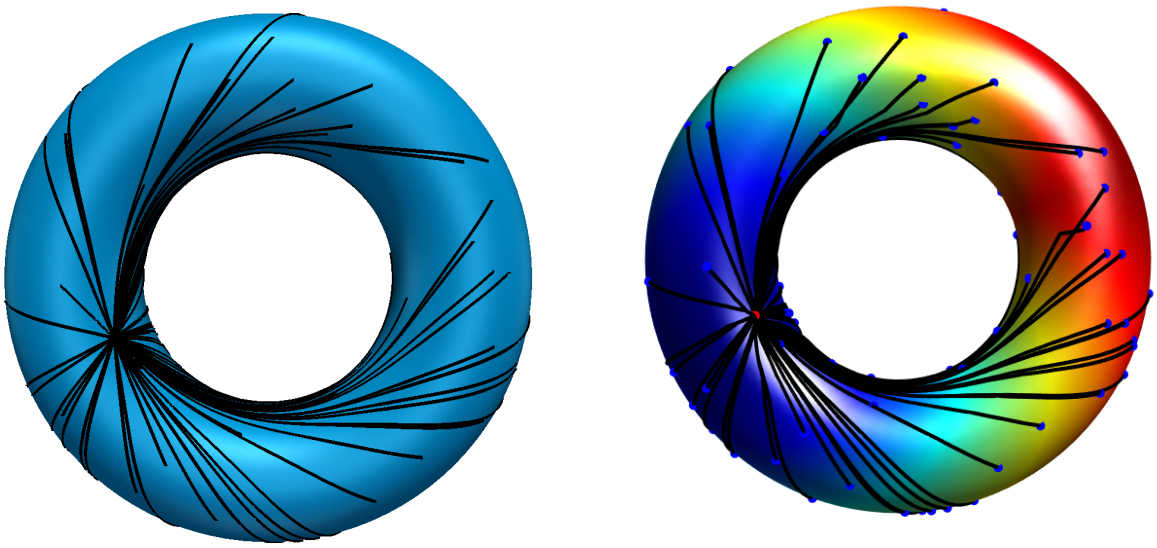

Figure 5.3: The approximate geodesics computed by the DLA (left) and FMM (right) algorithms on a triangulated torus.

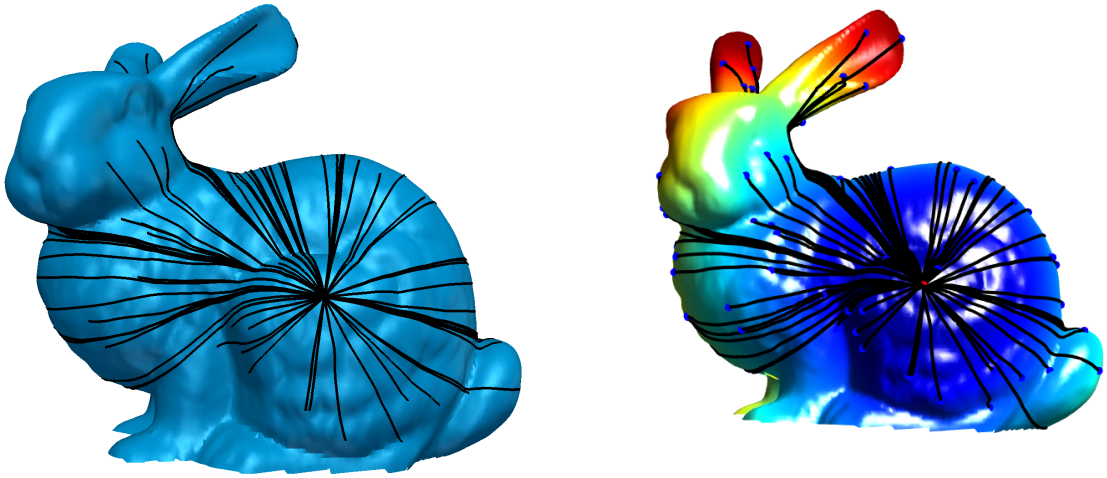

Figure 5.4: The approximate geodesics computed by the DLA (left) and FMM (right) algorithms on the Stanford bunny mesh.

5.2 Examples of a cutting pattern design

Our goal was to design a cutting pattern for a given membrane structure model that is represented by a triangulated surface. This means that we want to find such shapes of planar strips that form the desired 3D shape after joining. We proceeded as follows. On the given triangulated surface, we found geodesics that serve as cutting curves for the cutting pattern generation. Consider cutting geodetics in Figure 5.5.

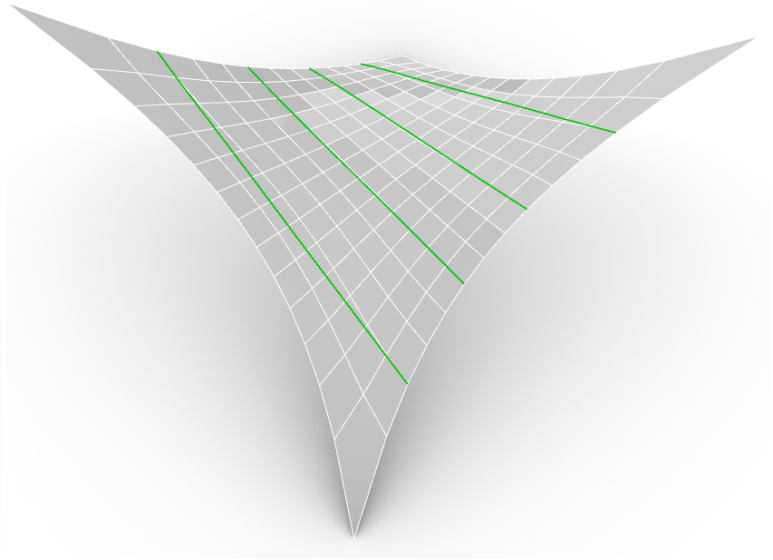

Figure 5.5: The geodesics for the initial polygonal front generations

These curves divide the surface into parts that we need to re-triangulate so that the union of these parts once again constitutes one triangulated surface. That is, the cutting curve nodes are the vertices of the newly formed triangulated surface. Therefore, we proceed as follows. We find the boundaries of the parts that we get by dividing the surface by cutting geodesics and dividing them almost equally by discrete step δ_T. We call such closed discrete curves polygonal fronts. These polygonal fronts are the initial condition for the triangularization algorithm that we have implemented according to the article(11). The polygonal fronts, together with the newly created triangulation (blue colour), can be seen in Figure 5.6.

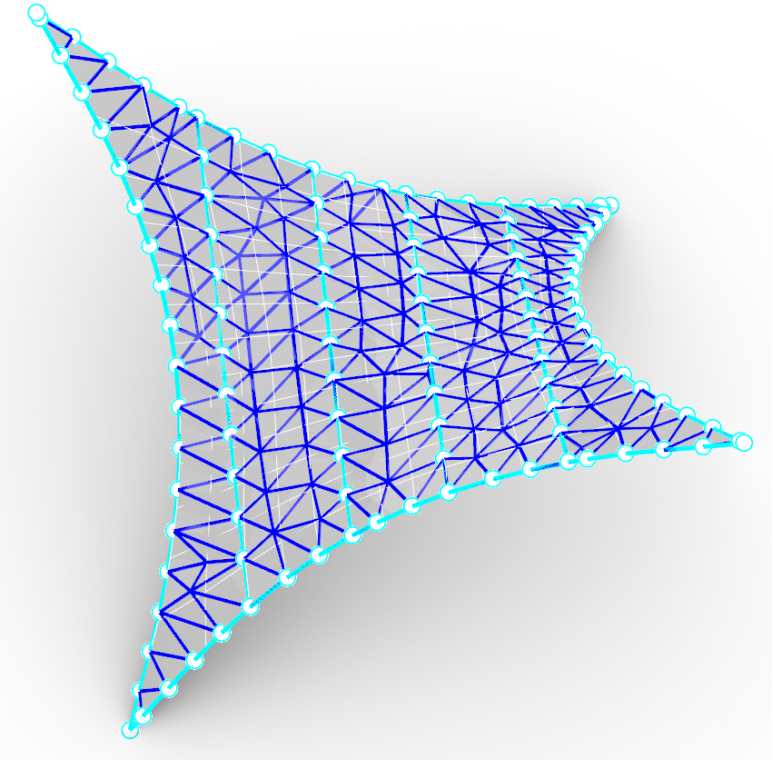

Figure 5.6: The mesh obtained by the marching triangulation algorithm for geodesic cutting curves.

After receiving the required triangulation, we project the individual parts of it into the plane. We use an

algorithm for each of these projections (37), which iteratively minimizes the difference between the edge length of the original and the projected triangulation. This will provide a cutting pattern for each cutting section. The final geodesic cutting pattern can be seen in Figure 5.7. In the same way, we constructed a non-geodetic cutting pattern for the same triangulated surface. We chose the cutting curves in a standard way, i.e. the curves formed by the edges of the triangulated surface. We calculated material waste for both created cutting patterns, the results are presented in Table 5.2.

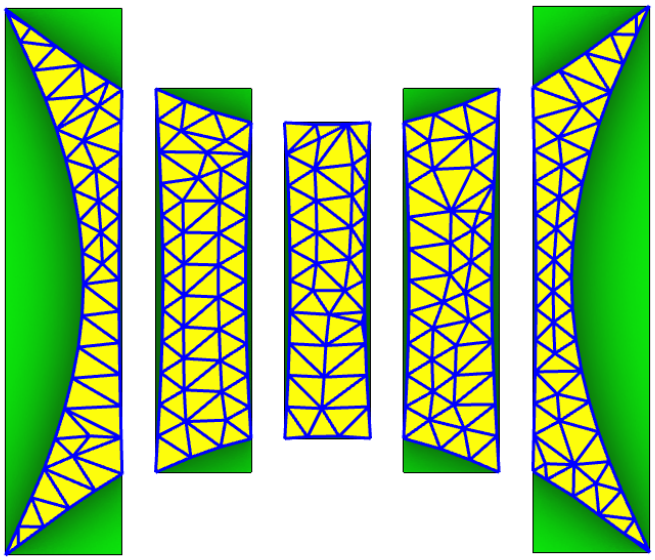

Figure 5.7: The individual strips of cutting pattern for triangulated surface shown in Figure 5.6 after developing into plane constructed from geodesics cutting lines. The corresponding bounding boxes are shown too.

Cutting	Strip 1	Strip 2	Strip 3	Strip 4	Strip 5	Overall
geodesic	18.732	3.209	1.358	3.195	18.771	45.265
layout curves	20,301	4.340	1.526	4.354	20.313	50.834

Table 5.2: Cutting pattern experiment - the waste of material corresponding to the geodesic and non-geodesic cutting patterns in area units. We show the waste for each of the five strips and the overall waste. The strips are numbered from left to right.

In this case, the total material saving using the geodesic cutting pattern is 12.3%.

References

[1] M Balazovjech, K Mikula, and M Petrášová Proceedings of ALGORITMY. 2012. . . . . Lagrangean method with topological changes for numerical modelling of forest fire propagation. *iam.fmph.uniba.sk*. URL http://www.iam.fmph.uniba.sk/algoritmy2012/zbornik/5Balazovjech.pdf.

[2] John W. Barrett, Harald Garcke, and Robert Nürnberg. Numerical approximation of gradient flows for closed curves in Rd. *IMA Journal of Numerical Analysis*, 30(1):4–60, jan 2010. ISSN 02724979. doi: 10.1093/imanum/drp005. URL https://academic.oup.com/imajna/article-lookup/doi/10.1093/imanum/drp005.

[3] Heike Benninghoff and Harald Garcke. Segmentation and Restoration of Images on Surfaces by Parametric Active Contours with Topology Changes. *Journal of Mathematical Imaging and Vision*, 55(1):105–124, may 2016. ISSN 15737683. doi: 10.1007/s10851-015-0616-6. URL http://link.springer.com/10.1007/s10851-015-0616-6.

[4] Prosenjit Bose, Anil Maheshwari, Chang Shu, and Stefanie Wuhrer. A survey of geodesic paths on 3D surfaces, apr 2011. ISSN 09257721. URL http://arxiv.org/abs/0904.2550http://dx.doi.org/10.1016/j.comgeo.2011.05.006.

[5] R. Brémond, D. Jeulin, P. Gateau, J. Jarrin, and G. Serpe. Estimation of the transport properties of polymer composites by geodesic propagation. *Journal of Microscopy*, 176(2):167–177, nov 1994. ISSN 00222720. doi: 10.1111/j.1365-2818.1994.tb03511.x. URL http://doi.wiley.com/10.1111/j.1365-2818.1994.tb03511.x.

[6] Youdong Chen and Ling Li. Smooth Geodesic Interpolation for Five-Axis Machine Tools. *IEEE/ASME Transactions on Mechatronics*, 21(3):1592–1603, jun 2016. ISSN 1083-4435. doi: 10.1109/TMECH.2016.2521683. URL http://ieeexplore.ieee.org/document/7393593/.

[7] Youdong Chen, Ling Li, and Wei Tang. An improved geodesic algorithm for trajectory planning of multi-joint robots. *International Journal of Advanced Robotic Systems*, 13(5):172988141665774, sep 2016. ISSN 1729-8814. doi: 10.1177/1729881416657742. URL http://journals.sagepub.com/doi/10.1177/1729881416657742.

[8] Michael E. Gage. Curve shortening on surfaces. *Annales scientifiques del l'É.N.S.*, 2(2):229–256, 1990. ISSN 0012-9593. doi: 10.24033/asens.1603. URL http://www.numdam.org/item?id=ASENS_1990_4_23_2_229_0.

[9] Matthew A. Grayson. Shortening Embedded Curves. *The Annals of Mathematics*, 129(1):71, jan 1989. ISSN 0003486X. doi: 10.2307/1971486. URL https://www.jstor.org/stable/1971486?origin=crossref.

[10] Lothar Gruendig, Erik Moncrieff, Peter Singer, and Dieter Ströbel. High-performance cutting pattern generation of architectural textile structures. *Proc. of International Colloquium on Computation of Shell And Spatial Structures.*, 2000. URL https://www.technet-gmbh.com/fileadmin/user_upload/technet/Publikationen/Easy/Cutpat2.pdf.

[11] Erich Hartmann. A marching method for the triangulation of surfaces. *The Visual Computer*, 14(3):95–108, jul 1998. ISSN 01782789. doi: 10.1007/s003710050126. URL http://link.springer.com/10.1007/s003710050126.

[12] Thomas Y. Hou, John S. Lowengrub, and Michael J. Shelley. Removing the stiffness from interfacial flows with surface tension. *Journal of Computational Physics*, 114(2):312–338, oct 1994. ISSN 10902716. doi: 10.1006/jcph.1994.1170. URL https://www.sciencedirect.com/science/article/pii/S0021999184711703.

[13] Thomas Y. Hou, Isaac Klapper, and Helen Si. Removing the Stiffness of Curvature in Computing 3-D Filaments. *Journal of Computational Physics*, 143(2):628–664, jul 1998. ISSN 00219991. doi: 10.1006/jcph.1998.5977. URL https://www.sciencedirect.com/science/article/pii/S0021999198959770.

[14] Takashi Kanai and Hiromasa Suzuki. Approximate shortest path on a polyhedral surface and its applications. *CAD Computer Aided Design*, 33(11):801–811, sep 2001. ISSN 00104485. doi: 10.1016/S0010-4485(01)00097-5. URL https://www.sciencedirect.com/science/article/pii/S0010448501000975.

[15] Emin Kasap, Mustafa Yapici, and F. Talay Akyildiz. A numerical study for computation of geodesic curves. *Applied Mathematics and Computation*, 171(2):1206–1213, dec 2005. ISSN 00963003. doi: 10.1016/j.amc.2005.01.109. URL https://www.sciencedirect.com/science/article/pii/S0096300305001700.

[16] R Kimmel and J A Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, jul 1998. ISSN 0027-8424. doi: 10.1073/pnas.95.15.8431. URL http://www.ncbi.nlm.nih.gov/pubmed/9671694http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC21092http://www.pnas.org/cgi/doi/10.1073/pnas.95.15.8431.

[17] R. Kimmel, A. Amir, and a M Bruckstein. Finding shortest paths on surfaces using level sets propagation, jun 1995. ISSN 01628828. URL http://ieeexplore.ieee.org/document/387512/http://www.cs.technion.ac.il/~ron/PAPERS/geodesics_pami1995.pdf.

[18] Ron Kimmel. Finding shortest paths on surfaces by fast global approximation and precise local refinement. *Proceedings of SPIE*, 10:198–209, 1995. ISSN 0277786X. doi: 10.1117/12.198608. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.1851http://link.aip.org/link/?PSI/2356/198/1&Agg=doi.

[19] Ron Kimmel, Doron Shaked, Nahum Kiryati, and Alfred M. Bruckstein. Skeletonization via Distance Maps and Level Sets. *Computer Vision and Image Understanding*, 62(3):382–391, nov 1995. ISSN 1077-3142. doi: 10.1006/CVIU.1995.1062. URL https://www.sciencedirect.com/science/article/pii/S1077314285710624.

[20] Masato Kimura. Numerical analysis of moving boundary problems using the boundary tracking method. *Japan Journal of Industrial and Applied Mathematics*, 14(3):373–398, oct 1997. ISSN 0916-7005. doi: 10.1007/BF03167390. URL http://link.springer.com/10.1007/BF03167390.

[21] Li Ma and Dezhong Chen. Curve shortening in a Riemannian manifold. *Annali di Matematica Pura ed Applicata*, 186(4):663–684, jun 2007. ISSN 03733114. doi: 10.1007/s10231-006-0025-y. URL http://link.springer.com/10.1007/s10231-006-0025-y.

[22] Takashi Maekawa. Computation of Shortest Paths on Free-Form Parametric Surfaces. *Journal of Mechanical Design*, 118(4):499, dec 1996. ISSN 10500472. doi: 10.1115/1.2826919. URL http://mechanicaldesign.asmedigitalcollection.asme.org/article.aspx?articleid=1444939.

[23] MeshPaths. MeshPaths GrassHopper assembly for Rhinoceros, 2018. URL http://www.food4rhino.com/app/geodesic-lines-meshes-meshpaths.

[24] Karol Mikula and Daniel Ševčovič. Computational and qualitative aspects of evolution of curves driven by curvature and external force. *Computing and Visualization in Science*, 6(4):211–215, apr 2004. ISSN 14330369. doi: 10.1007/s00791-004-0131-6. URL http://link.springer.com/10.1007/s00791-004-0131-6.

[25] Karol Mikula and Daniel Ševčovič. Evolution of curves on a surface driven by the geodesic curvature and external force. *Applicable Analysis*, 85(4):345–362, apr 2006. ISSN 0003-6811. doi: 10.1080/00036810500333604. URL http://www.tandfonline.com/doi/abs/10.1080/00036810500333604.

[26] Karol Mikula and Jozef Urban. A new tangentially stabilized 3D curve evolution algorithm and its application in virtual colonoscopy. *Advances in Computational Mathematics*, 40(4):819–837, aug 2014. ISSN 10197168. doi: 10.1007/s10444-013-9328-x. URL http://link.springer.com/10.1007/s10444-013-9328-x.

[27] Karol Mikula, Mariana Remešíková, Peter Sarkoci, and Daniel Ševčovič. Manifold Evolution with Tangential Redistribution of Points. *SIAM Journal on Scientific Computing*, 36(4):A1384–A1414, jan 2014. ISSN 1064-8275. doi: 10.1137/130927668. URL http://epubs.siam.org/doi/10.1137/130927668http://epubs.siam.org.proxy2.library.illinois.edu/doi/abs/10.1137/130927668.

[28] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, nov 1988. ISSN 0021-9991. doi: 10.1016/0021-9991(88)90002-2. URL https://www.sciencedirect.com/science/article/pii/0021999188900022.

[29] Gabriel Peyre. Matlab Toolbox Fast Marching version 1.2 (5.8 MB): A toolbox for the computation of the Fast Marching algorithm in 2D and 3D. URL https://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching?requestedDomain=www.mathworks.com.

[30] B. Porter. Genetic computation of geodesics on three-dimensional curved surfaces. In *1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, volume 1995, pages 448–453. IEE, 1995. doi: 10.1049/cp:19951090. URL http://digital-library.theiet.org/content/conferences/10.1049/cp_19951090.

[31] Dao T P Quynh, Ying He, Shi Qing Xin, and Zhonggui Chen. An intrinsic algorithm for computing geodesic distance fields on triangle meshes with holes. In *Graphical Models*, volume 74, pages 209–220. Academic Press, jul 2012. doi: 10.1016/j.gmod.2012.04.009. URL https://www.sciencedirect.com/science/article/pii/S152407031200029X?via%3Dihub.

[32] G. V.V. Ravi Kumar, Prabha Srinivasan, V. Devaraja Holla, K. G. Shastry, and B. G. Prakash. Geodesic curve computations on surfaces. *Computer Aided Geometric Design*, 20(2):119–133, may 2003. ISSN 01678396. doi: 10.1016/S0167-8396(03)00023-2. URL https://www.sciencedirect.com/science/article/pii/S0167839603000232.

[33] N. Rawlinson and M. Sambridge. Wave front evolution in strongly heterogeneous layered media using the fast marching method. *Geophysical Journal International*, 156(3):631–647, mar 2004. ISSN 0956540X. doi: 10.1111/j.1365-246X.2004.02153.x. URL https://academic.oup.com/gji/article-lookup/doi/10.1111/j.1365-246X.2004.02153.x.

[34] Marián Šagát. mariansagat.github.io, 2017. URL https://mariansagat.github.io/.

[35] Alon Spira and Ron Kimmel. Geometric curve flows on parametric manifolds. *Journal of Computational Physics*, 223(1):235–249, apr 2007. ISSN 0021-9991. doi: 10.1016/J.JCP.2006.09.008. URL https://www.sciencedirect.com/science/article/pii/S0021999106004347.

[36] Alon Spira, Alon Spira, and Ron Kimmel. Geodesic Curvature Flow on Parametric Surfaces. *IN CURVE AND SURFACE DESIGN: SAINT-MALO 2002*, pages 365—-373, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.2385.

[37] Z Tabarrok, B and Qin. Form finding and cutting pattern generation for fabric tension structures. *Computer-Aided Civil and Infrastructure Engineering*, 8(5):377—-384, 1993.

[38] E. Topsakal, R. Kindt, K. Sertel, and J. Volakis. Evaluation of the BICGSTAB(l) algorithm for the finite-element/boundary-integral method. *IEEE Antennas and Propagation Magazine*, 43(6):124–131, 2001. ISSN 10459243. doi: 10.1109/74.979531. URL http://ieeexplore.ieee.org/document/979531/.

[39] Chunlin Wu and Xuecheng Tai. A level set formulation of geodesic curvature flow on simplicial surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):647–662, jul 2010. ISSN 10772626. doi: 10.1109/TVCG.2009.103. URL http://www.ncbi.nlm.nih.gov/pubmed/20467062http://ieeexplore.ieee.org/document/5226631/.

[40] Kun-Lin Wu, Ting-Jui Ho, Sean A. Huang, Kuo-Hui Lin, Yueh-Chen Lin, and Jing-Sin Liu. Path Planning and Replanning for Mobile Robot Navigation on 3D Terrain: An Approach Based on Geodesic. *Mathematical Problems in Engineering*, 2016:1–12, oct 2016. ISSN 1024-123X. doi: 10.1155/2016/2539761. URL https://www.hindawi.com/journals/mpe/2016/2539761/.

Zoznam publikačnej činnosti

REMEŠÍKOVÁ, Mariana - ŠAGÁT, Marián - NOVYSEDLÁK, Peter. Discrete Lagrangian Algorithm for Finding Geodesics on Triangular Meshes. Applied Mathematical Modelling. Date Submission: 5 November 2018

ŠAGÁT, Marián - REMEŠÍKOVÁ, Mariana. Approximation of geodesics on triangulated surfaces based on Lagrangian curve evolution. In Advances in Architectural, Civil and Environmental Engineering [elektronický zdroj] : 26th Annual PhD Student Conference on Architecture and Construction Engineering, Building Materials, Structural Engineering, Water and Environmental Engineering, Transportation Engineering, Surveying, Geodesy, and Applied Mathematics. 26. October 2016, Bratislava. 1. vyd. Bratislava : Slovenská technická univerzita v Bratislave, 2016, CD-ROM, s. 58-65. ISBN 978-80-227-4645-8.

ŠAGÁT, Marián - REMEŠÍKOVÁ, Mariana. Discrete Lagrangian algorithm for finding geodesics and its comparisons with the fast marching method. In Advances in Architectural, Civil and Environmental Engineering [elektronický zdroj] : 27th Annual PhD Student Conference on Applied Mathematics, Applied Mechanics, Geodesy and Cartography, Landscaping, Building Technology, Theory and Structures of Buildings, Theory and Structures of Civil Engineering Works, Theory and Environmental Technology of Buildings, Water Resources Engineering. 25. October 2017, Bratislava, Slovakia. 1. vyd. Bratislava : Spektrum STU, 2017, CD-ROM, s. 54-61. ISBN 978-80-227-4751-6.

ŠAGÁT, Marián - REMEŠÍKOVÁ, Mariana. Implementation of approximate discrete Lagrangian algorithm for computing geodesics on smooth surfaces and on polygonal meshes for Rhinoceros. In Advances in Architectural, Civil and Environmental Engineering [elektronický zdroj] : 28th Annual PhD Student Conference on Applied Mathematics, Applied Mechanics, Building Technology, Geodesy and Cartography, Landscaping, Theory and Environmental Technology of Buildings, Theory and Structures of Buildings, Theory and Structures of Civil Engineering Works, Water Resources Engineering. October 24th 2018, Bratislava. 1. vyd. Bratislava : Spektrum STU, 2018, CD-ROM, s. 32-37. ISBN 978-80-227-4864-3.