International Journal of Pure and Applied Mathematics

Volume 51 No. 4 2009, 567-595

VARIATIONAL CALCULUS AND MULTIGRID DYNAMICAL PROGRAMMING ON EXPO-RATIONAL TENSOR-PRODUCT SURFACES

Lubomir T. Dechevsky[1 §], Xiang F. Bu[2], Børre Bang[3], Arne Laksā[4]

[1,2,3,4]Institute for Information, Energy and Space Technology
Narvik University College
2, Lodve Lange's St., P.O. Box 385, N-8505, Narvik, NORWAY

[1]e-mail: ltd@hin.no
url: http://ansatte.hin.no/ltd/
[2]e-mail: buxiangfeng@hotmail.com
[3]e-mail: bb@hin.no
url: http://ansatte.hin.no/bb/
[4]e-mail: ala@hin.no
url: http://ansatte.hin.no/ala/

Abstract: In [6] the problem about finding geodesics on parametric surfaces with a given known parametrization was considered. The purpose of the present work is to study the possibility to compute geodesics and possibly some other types of optimal curves on surfaces in the more general situation when a parametrization of the surface is not readily available, but information about this surface is only given on a discrete rectangular set of points. This information may include only position of the point on the surface, or it may also include information about derivatives of the parametrized surface. From the discrete set of points, by interpolation, a parametrization of the surface is generated which interpolates the given data. For this purpose we use expo-rational B-spline (ERBS) surfaces. A remarkable advantage of ERBS compared to traditional tools of computer-aided geometric design, such as, polynomial B-splines and NURBS, is that ERBS tensor-product surfaces have C^∞-smooth parametrization even when they are only G^0-continuous.

Received: April 17, 2008

§Correspondence author

We confirm the remarks made in section 6 of [6] that the property "C^∞-smooth *non-regular* parametrization of a G^0-continuous surface" can be used to obtain efficient and elegant extension of the methods for exact or numerical minimization of smooth functionals to the case of non-smooth ones.

The direct optimization method of multigrid dynamic programming type, proposed in [6] and [7] is completed with an implementation of the A**-method introduced here, which is an upgrade of the so-called A*-search method (see [10]). The main difference between the A**-method for global search, and its analogue in [6] and [7] is that the cross sections of the dynamic programming are smaller in size than in [6] which leads to lower computational cost for achieving results which are very close to the results of [6]. Thus, the dynamical programming method proposed here can be used for verification of the dynamical programming method proposed in [6] and [7], and vice versa.

Here several types of cost functionals are considered; however, similar to [6], only the case of shortest length (geodesics) is considered in detail. Similar to [7], constraints of several types are studied, both of equality or inequality type, smooth or non-smooth.

One of the advantages of the present algorithm, compared to the methods on triangulated surfaces as discussed in [14], is its easy upgrading to 3D volume deformations and higher-dimensional manifolds. The use of expo-rational B-splines allows a simple uniform classical variational approach to the handling of both very smooth and non-smooth geometrical data, including cases where subgradient methods for non-smooth optimization do not work (for example, for Lip-α cost functionals and/or constraints with $0 < \alpha < 1$).

The extension of the definition of expo-rational B-splines for triangulated manifolds (see [5]) allows the method considered here for tensor-product surfaces to be extended for general triangulated manifolds. It also allows to upgrade the method of geodesic Bezier curves proposed in [11] so that the challenges of handling non-smooth and very smooth manifolds (see Subsection 7.1 in [11]) are simultaneously overcome in a uniform way. The uniformity and robustness of the expo-rational B-spline parametrization has considerable potential even in cases where parametric models are considered less efficient than implicit and other non-parametric geometric representations (see, e.g., [13]).

The animated 3D visualization used for the illustrations given here is based on *C++/OpenGL*, using the in-house library *GM_lib* developed at Narvik University College, Norway, and *Qt* window handling.

AMS Subject Classification: 49L20, 53C22, 65D07, 26B35, 33F05, 41A05, 41A15, 41A30, 41A55, 41A63, 53A04, 53A05, 65D05, 65D17, 65D18, 65Y25,

68U05, 68U07
Key Words: local, global, optimization, algorithm, discrete time, continuous time, dynamical system, variational calculus, optimal control, Bellman's optimality principle, dynamical programming, direct optimization method, cost-functional, smooth, non-smooth, Lipschitz, cuspidal, discontinuous, additive, constraint, equality, inequality, computational complexity, grid refinement, resolution tolerance, multigrid, triangulation, parametric curve, parametric surface, computer-aided geometric design, global geodesics, numerical integration, error, stability, sensitivity, robustness, small perturbation

1. Introduction

In [6] and [7] we considered a variety of problems of variational calculus and optimal control theory with applications to geometric modeling with parametric curves, surfaces, volume deformations and higher dimensional manifolds. As a numerical method for approximate computation of the extremals, we proposed a fast multigrid dynamical programming algorithm. In [6] we used this method for the fast computation of geodesics on parametric surfaces, and Section 7 of [6] contains a detailed comparison of this algorithm with current state-of-the-art methods for exact and approximate computation of geodesics on triangulated surfaces, as described in [14] and the references therein. We pointed out that the algorithm from [6] has numerous significant advantages compared to the methods discussed in [14]. In the present communication we continue this analysis in the case when the original surface is approximated (by interpolation or by fitting) with a tensor-product expo-rational surface (see [8]).

While in [6] we considered the problem about finding geodesics on parametric surfaces with a given known parametrization, in the present paper we show how to compute geodesics (and possibly other types of optimal curves on surfaces) in the more general situation when a parametrization of the surface is not readily available, but information about this surface is only given on a discrete rectangular set of points. This information may include only position of the point on the surface, or it may also include information about derivatives of the parametrized surface. From the discrete set of points, by interpolation, we shall generate a parametrization of the surface which interpolates the given data. For this purpose we shall use expo-rational B-spline (ERBS) surfaces.

The animated 3D visualization used for the illustrations given here is based on *C++/OpenGL*, using the in-house library *GM_lib* developed at Narvik University College, Norway, and Trolltech *Qt* for the interfaces. The development of the software application used for generating the images in Figures 2–9 in the sequel of this paper was part of the results obtained in the M. Sc. Diploma thesis [3].

The colour graphical images appear as grey-scale in the journal version of this exposition, which may potentially cause some problems in the interpretation of references to colour in the captions to the images and in the analysis of the images in the main body of the text. We have nevertheless opted to rely on the geometric intuition of the reader and to make only minimal changes in these references.

2. The Multigrid Method of [4], [6] and [7]

In this section we provide the necessary minimum of information about the mathematical model developed in [6] and the global optimization method proposed in [4], for the sake of self-consistency if the exposition. We present only the components of the mathematical model and the main steps of the method. For details and discussion, we refer to [6] (Sections 2-4, 7) and [4].

2.1. The Mathematical Model of [6]

2.1.1. Components of the Model

1. Variables.

Vector of state in moment T_i :

$$\overrightarrow{x}_i,\ \dim \overrightarrow{x}_i = l_i,\ i = 0, 1, ..., N \tag{1}$$

Vector of control in moment T_i :

$$\overrightarrow{u}_i,\ \dim \overrightarrow{u}_i = m_i,\ i = 0, 1, ..., N-1 \tag{2}$$

2. Constraints.

Phase equalities: The state x_i depends on the control u_i, as follows:

$$\begin{aligned} x_{i+1} &= f_i\left(\overrightarrow{x}_0, \overrightarrow{x}_1, ..., \overrightarrow{x}_i; \overrightarrow{u}_0, \overrightarrow{u}_1, ..., \overrightarrow{u}_i\right) \\ i &= 0, 1, ..., N-1 \end{aligned} \tag{3}$$

International Journal of Pure and Applied Mathematics (ISSN 1311-8080)
Volume 51, Issue 4 (2009), 567 – 595
2009 Academic Publications

Constraint inequalities: Under the constraints (in moment T_i),

$$\begin{aligned} \overrightarrow{g}_{-1}(\overrightarrow{x}_0) &\geq 0 \qquad (4)\\ \overrightarrow{g}_0(\overrightarrow{x}_0, \overrightarrow{x}_1; \overrightarrow{u}_0) &\geq 0 \\ \overrightarrow{g}_0(\overrightarrow{x}_0, \overrightarrow{x}_1, \overrightarrow{x}_2; \overrightarrow{u}_0, \overrightarrow{u}_1) &\geq 0 \\ \\ \overrightarrow{g}_i(\overrightarrow{x}_0, \overrightarrow{x}_1, ..., \overrightarrow{x}_{i+1}; \overrightarrow{u}_0, \overrightarrow{u}_1, ..., \overrightarrow{u}_i) &\geq 0 \\ \dim \overrightarrow{g}_i = n_i, \ i = 0, 1, ..., N-1. \end{aligned}$$

3. Cost Functional. it must be additive, i.e., of the form

$$G(\overrightarrow{x}_0, \overrightarrow{x}_1, ..., \overrightarrow{x}_N; \overrightarrow{u}_0, \overrightarrow{u}_1, ..., \overrightarrow{u}_{N-1}) = \sum_{k=0}^{N-1} F_k(\overrightarrow{x}_k, \overrightarrow{u}_{k-1}; \overrightarrow{x}_{k+1}, \overrightarrow{u}_k) + F_N(\overrightarrow{x}_N) .$$

2.1.2. Extremal Problem

The model is optimized by minimization or maximization of (**??**) in $\overrightarrow{u}_0, \overrightarrow{u}_1, ...,$ $\overrightarrow{u}_{N-1}$, under the constraints (3) and (4).

2.2. The Direct Optimization Method of [4] and [6]

A direct optimization method is an optimization method such that:

1. the exact solution of the optimization problem is being approximated via a solution generated by this method;

2. every approximate solution generated by the method satisfies all the constraints of the optimization problem (or, in generalized form, it satisfies approximations to the constraints of the problem);

3. the optimization problem is correctly posed in the sense of Tikhonov (see, e.g., [15]).

Step methods for optimal control based on Bellman's optimality principle form a particular class of direct optimization methods.

Bellman's optimality principle can be formulated, as follows: *Every (connected) part of an optimal trajectory with respect to a given cost functional is also optimal with respect to the same functional, for its respective initial and end point.*

The numerical implementations of Bellman's optimality principle in optimizing continual dynamical systems and their discrete approximations, as well as in optimizing discrete dynamical systems 'per se', result in a family of algorithms called *dynamical programming* algorithms. For more details, we refer to [1], [2] and [12].

In brief, the idea of the multigrid method of [4], [6] and [7] is, as follows:

Step 1. *Within the usual dynamical programming paradigm, select a coarse (low-resolution) global mesh. The dynamical programming algorithm over this mesh will have low computational complexity and low execution time.*

Step 2. *Using the globally optimal solution on the coarse mesh, obtained at step 1 (or a finer resolution mesh obtained on a refinement iteration - see step 3 - and used here in the same way as the coarse mesh on step 1), as an initial approximation to the exact solution of the considered problem, define a finer-resolution local mesh around the initial approximation.*

Step 3. *Check if the globally optimal solution on the new mesh coincides with the initial approximation. If not, go to step 2; if yes, take the globally optimal solution on the current mesh to be the sought approximation to the exact global solution within the resolution of the finest mesh used.*

Provisionally, the stopping criteria at step 3 can be related to a cost tolerance applied to the difference between the cost values on two subsequent iterations. This modification should be used only when achieving a 'good cost value' is enough, and the exact global extremum on the mesh is of secondary importance.

3. Cost Functionals and Constraints

3.1. Geodesics

In [6] we used the following cost functional for computing the geodesic line between two points on a C^2-smooth surface.

Let $\overrightarrow{r}(u,v)$ be the parametrization of a surface in 3D, $dim\ \overrightarrow{r} = 3$, where $(u,v) \in I \times J$, I and J being (open or closed) intervals. Let P and Q be two points on the surface, with $\overrightarrow{O}P = \overrightarrow{r}(u_0,v_0)$, $\overrightarrow{O}Q = \overrightarrow{r}(u_1,v_1)$, O being the origin in 3D. Assume also that $\overrightarrow{r} \in C^1(I \times J)$. The geodesic on the surface, between P and Q, is the surface curve $\zeta(t) = \overrightarrow{r}(u(t),v(t))$, $t \in [t_0,t_1]$, $u(t_0) = u_0$, $v(t_0) = v_0$, $u(t_1) = u_1$, $v(t_1) = v_1$, which minimizes the continual variational functional

$$\int_{t_0}^{t_1} \left| \dot{\overrightarrow{\zeta}}(\tau) \right| d\tau = \int_{t_0}^{t_1} [E\left(u(\tau),v(\tau)\right)\dot{u}(\tau)^2 \tag{5}$$

$$+2F\left(u(\tau),v(\tau)\right)\dot{u}(\tau)\dot{v}(\tau) + G\left(u(\tau),v(\tau)\right)\dot{v}(\tau)^2]^{1/2} d\tau$$

where $<.,.>$ is the inner product in 3D $-$ space , and

$$E\left(u(\tau),v(\tau)\right) = \left\langle \overrightarrow{r}_u(u(\tau),v(\tau)), \overrightarrow{r}_v(u(\tau),v(\tau))\right\rangle,$$

$$F\left(u(\tau),v(\tau)\right) = \left\langle \overrightarrow{r}_u(u(\tau),v(\tau)), \overrightarrow{r}_v(u(\tau),v(\tau))\right\rangle,$$

$$G\left(u(\tau),v(\tau)\right) = \left\langle \overrightarrow{r}_u(u(\tau),v(\tau)), \overrightarrow{r}_v(u(\tau),v(\tau))\right\rangle,$$

where E, F, G, are the coefficients of the Riemannian metric (or First fundamental form) on the surface.

The discrete model from Section 2 approximating this continual variational functional has the following components.

State vector

$$\overrightarrow{x}_i = \begin{pmatrix} x_{1,i} \\ x_{2,i} \end{pmatrix};\ i = 0,1,...,N; N = 2^n \tag{6}$$

Control vector

$$\overrightarrow{u}_i = \begin{pmatrix} u_{1,i} \\ u_{2,i} \end{pmatrix};\ i = 0,1,...,N \tag{7}$$

Phase equalities

$$\overrightarrow{x}_{i+1} = \overrightarrow{u}_i;\ i = 0,...,N-1 \tag{8}$$

Dimensions

$$\begin{array}{rcll} l_i & = & 2 & i = 0,...,N \\ m_i & = & 2 & i = 0,...,N-1 \\ n_i & = & 2 & i = 0,...,N \end{array}$$

Constraint inequalities

$$\left.\begin{array}{r} \overrightarrow{x}_0 - \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \geq 0 \\ -\overrightarrow{x}_0 + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \geq 0 \end{array}\right\} \Leftrightarrow \begin{array}{r} x_{1,0} - u_0 \geq 0 \\ x_{2,0} - v_0 \geq 0 \\ -x_{1,0} + u_0 \geq 0 \\ -x_{2,0} + v_0 \geq 0 \end{array}$$

$$\left.\begin{array}{r} \overrightarrow{x}_N - \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \geq 0 \\ -\overrightarrow{x}_N + \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \geq 0 \end{array}\right\} \Leftrightarrow \begin{array}{r} x_{1,N} - u_1 \geq 0 \\ x_{2,N} - v_1 \geq 0 \\ -x_{1,N} + u_1 \geq 0 \\ -x_{2,N} + v_1 \geq 0 \end{array}$$

$$\left.\begin{array}{l} \overrightarrow{x}_i - \begin{pmatrix} a \\ c \end{pmatrix} \geq 0 \\ -\overrightarrow{x}_i + \begin{pmatrix} b \\ d \end{pmatrix} \geq 0 \end{array}\right\} \Leftrightarrow \begin{array}{c} x_{1,i} - a \geq 0 \\ x_{2,i} - c \geq 0 \\ -x_{1,i} + b \geq 0 \\ -x_{2,i} + d \geq 0 \end{array}$$

The state and the control vector correspond to (u,v)-parametrizations of the surfaces.

Cost functional

To construct the cost functional for the discrete model, we approximate the integral in (5) by the Trapezoid quadrature rule

$$\begin{aligned} F_i(\overrightarrow{x}_i, \overrightarrow{u}_{i-1}, \overrightarrow{x}_{i+1}, \overrightarrow{u}_i) = \frac{1}{2}[&E(x_{1,i}, x_{2,i})(u_{1,i} - u_{1,i-1})^2 \\ &+ 2F(x_{1,i}, x_{2,i})(u_{1,i} - u_{1,i-1})(u_{2,i} - u_{2,i-1}) \\ &+ G(x_{1,i}, x_{2,i})(u_{2,i} - u_{2,i-1})^2]^{1/2} \\ &+ \frac{1}{2}[E(x_{1,i+1}, x_{2,i+1})(u_{1,i} - u_{1,i-1})^2 \\ &+ 2F(x_{1,+1i}, x_{2,i+1})(u_{1,i} - u_{1,i-1})(u_{2,i} - u_{2,i-1}) \\ &+ G(x_{1,+1i}, x_{2,i+1})(u_{2,i} - u_{2,i-1})^2]^{1/2}. \end{aligned} \tag{9}$$

For integrand with low regularity (e.g., Lip $\alpha, 0 < \alpha < 1$ or Lipshitz), using the Trapezoid rule (2-point closed Newton–Cotes quadrature rule), which has order of convergence 2 on C^2, is sufficient. However, whenever the surface has regularity C^m, m≥3, we use μ iterations of the Romberg quadrature process, where

$$\int_{t_i}^{t_{i+1}} F(t)dt \approx \frac{t_{i+1} - t_i}{2^\mu} \left(\frac{1}{2} F_i + \sum_{v=1}^{2^\mu - 1} F(t_{v,i}) + \frac{1}{2} F_{i+1} \right) = I_{0,\mu}, \tag{10}$$

$$\begin{aligned} &\mu = 0, 1, 2, \ldots, \theta_{\mathrm{v}} = \tfrac{v}{2^\mu}, \mathrm{v} = 1, 2, \ldots, 2^\mu - 1 \\ &t_{v,i} = (1 - \theta_v)\, t_i + \theta_v \cdot t_{i+1} \\ &\overrightarrow{x}_{v,i} = (1 - \theta_v)\, \overrightarrow{x}_i + \theta_v \cdot \overrightarrow{x}_{i+1} \\ &\overrightarrow{u}_{v,i} = (1 - \theta_v)\, \overrightarrow{u}_{i-1} + \theta_v \cdot \overrightarrow{u}_i \end{aligned}$$

and, for $k = 0, 1, \ldots, \mu$,

$$I_{k,\mu} = \frac{1}{2^{2k} - 1} \left(2^{2k} I_{k-1,\mu+1} - I_{k-1,\mu} \right), \tag{11}$$

and on the μ-th Romberg iteration $I_{\mu,0}$ is accepted as the Romberg quadrature formula for $\int_{t_i}^{t_{i+1}} F(t)dt$. It has order of convergence $2\mu + 2$ where $\mu : \; m \leq 2\mu + 2 \leq m + 1$, so, for $F \in C^m$, $\left[\frac{m+1}{2}\right] - 1$ Romberg iterations are being performed, with [.] being the Gaussian bracket. Note that the simplest case $\mu = 0$ corresponds to the Trapezoid rule. For more details on the version of Romberg integration used here, see [6].

3.2. Other Cost Functionals

Some of the criteria often used in relevance to parametrized manifolds and, more generally, to parametric representations of dynamical systems, are given below.

- Length of curve (this type was considered in separate in the previous subsection)
- Shortest time type
- Energy type
- Entropy type
- Resource type
- Tikhonov regularization type (for example, a norm in a Sobolev, Besov or Triebel-Lizorkin space, measuring smoothness), depending on one or several small parameters
- Others

The method to compute the cost function of the other criteria types is essentially the same, as long as each of these criteria is additive, or is reduced to iterative computation of a sequence of additive criteria. For example, for shortest time:

$$Cost = \sum_{i=0}^{N-1} T(P_i, P_{i+1}) \to min \tag{12}$$

where the function $T(,)$ is the time from the point P_i to P_{i+1}, defined by

$$T(P_i, P_{i+1}) = \frac{L(P_i, P_{i+1})}{v}; \tag{13}$$

and where $L(,)$ is length by formula (5), (9) or (10), and v is the constant speed in the parametrization. A general method to compute criteria depending on time is given in [7] where the time is treated as control parameter (that is, it is one of the parameters to optimize in the cross sections of the dynamical programming algorithm). Here we consider the simpler case of natural parametrization when the speed is equal to 1 (or to a constant $v \neq 0$). In Figure 6 is given a model situation about shortest time trajectory between two points in the case of three different "permeable" media with different constant speeds (the blue (light) region has high speed, the yellow (medium gray-scale) region has middle speed and the violet (dark) region has low speed). This setting can be used, e.g., for ray tracing.

3.3. Constraints

All enhancements proposed in this paper are valid for the full range of types of constraints considered in [6] and [7]. As discussed in [6], [7], the great diversity of the range of possible types of constraints is due to the fact that we are using a *direct optimization method*. The only limitations which may arise can be ones *due to the assumption about additivity of the cost functional*. For example, if a highway project has to be optimized with respect to cut and fill masses, given a surface describing the terrain, constraints like steepness of ascent/descent and radius of curvature would be local and, thus, compatible with the additivity of the cost functional, while adding a cost component and/or constraints related to the exploitation expenses for the use of the new highway over different periods of time would be global in nature and would thus violate the requirement about additivity of the cost functional. However, even for this type of cost functionals and/or constraints it is possible to design an iterative upgrade of the present direct method where on every iteration the cost functional is additive, and the constraints are compatible with the additivity assumption. A detailed consideration of this upgrade is, however, beyond the scope of the present study, and will not be considered here. In the graphical examples given in the sequel of this paper we shall consider only 3 simple model types of constraints which are all compatible with the assumption about additivity of the cost functional, as follows:

- *Forbidden areas*: areas on the surface where curves are not allowed to go through. Example: the solution of cases (a-d) in Figure 3 corresponds to such a constrained problem.
- *Constraints on the speed of parametrization.* Example: the solution of cases (a-d) in Figure 6 corresponds to such a constrained problem.
- *Angle constraints* (which can be reformulated as *curvature constraints* – see [7]). Example: in Figure 1, there is an example of minimal angle restriction. The angles α and β are not allowed to be too small (say,

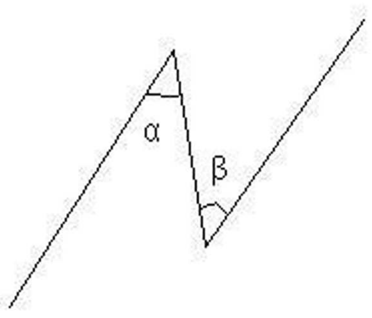

Figure 1: Minimal angle constraint. Each of the angles must not be less than the minimal angle constraint (a small positive value (say, in degrees (^{o}) – see subsection 3.3)).

values $< 50^o$ are forbidden). A typical example of such constraints are the limitations for an automobile road or a railway, where the limitations depend on the normative data base for the respective "class of the road" (higher or lower speed, longer or shorter vehicles, width of the road versus its curvature, and so on). For examples when such a constraint influences the form of the curve, see [7].

4. Expo-Rational B-Splines

4.1. Definition

An expo-rational B-spline function $f(t)$ is defined by:

$$f(t) = \sum_{k=1}^{n} l_k(t) B_k(t) \tag{14}$$

where

$$B_k(t) = \begin{cases} \int\limits_{t_{k-1}}^{t} \varphi_{k-1}(s)ds, & t_{k-1} < t \le t_k \\ 1 - \int\limits_{t_k}^{t} \varphi_k(s)ds, & t_k < t < t_{k+1} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

with

$$\varphi_k(t) = \frac{e^{-\beta_k \frac{|t-((1-\lambda_k)t_k+\lambda_k t_{k+1})|^{2\sigma_k}}{((t-t_k)(t_{k+1}-t)^{\gamma_k})^{\alpha_k}}}}{\int_{t_k}^{t_{k+1}} e^{-\beta_k \frac{|s-((1-\lambda_k)t_k+\lambda_k t_{k+1})|^{2\sigma_k}}{((s-t_k)(t_{k+1}-s)^{\gamma_k})^{\alpha_k}}} ds}, \tag{16}$$

where

$$\alpha_k > 0, \quad \beta_k > 0, \quad \gamma_k > 0, \quad 0 \le \lambda_k \le 1, \quad \sigma_k \ge 0,$$

on a strictly increasing knot vector $\overrightarrow{t} = \{t_0, ..., t_{n+1}\}$, and where

$$l_k(t), \quad k = 1, ..., n,$$

are local functions on their respective domains (t_{k-1}, t_{k+1}).

This means that $B_k(t)$ are defined with a minimal support by the three knots t_{k-1}, t_k, t_{k+1}. The domain of $f(t)$ is $(t_1, t_n]$.

The default intrinsic parameters are $\alpha_k = \beta_k = \gamma_k = \sigma_k = 1$ and $\lambda_k = \frac{1}{2}$.

4.2. Basic Properties

There are five basic properties of the basis function $B_k(t)$.

1. $B_k(t) \begin{cases} > 0, & t_{k-1} < t < t_{k+1} \\ = 0, & \text{otherwise} \end{cases}$ for $k = 1, ..., n$,
2. $\sum_{k=1}^{n} B_k(t) = 1$
   i.e. for $t_k < t \le t_{k+1}$ then $B_k(t) + B_{k+1}(t) = 1$,
3. $\begin{array}{lll} \text{if} & t_{k-1} < t_k & \text{then } B_k(t_k) = 1 \\ \text{else if } & t_{k-1} = t_k & \text{then } \lim\limits_{t \to t_k+} B_k(t) = 1 \end{array}$ for $k = 1, ..., n$,
4. $\frac{d^j}{dt^j} B_k(t_i) = 0$ for $j = 1, 2, ...,$ $i = 1, ..., n$ and for $k = 1, ..., n$,
5. $B_k(t)$ is C^∞ on (t_{k-1}, t_{k+1}).

4.3. Tensor Product Surfaces

For tensor product surfaces we have the following general formula for expo-rational B-splines:

$$S(u,v)=\sum_{i=1}^{n}\sum_{j=1}^{m}s_{ij}(u,v)B_i(u)B_j(v), \tag{17}$$

where $s_{ij}(u,v), \quad i=1,..,n \quad j=1,..m,$ are $n\times m$ local patches. The formula resembles the usual B-spline tensor product surface, except that $s_{ij}(u,v)$ are not points but surfaces.

An expo-rational B-spline tensor product surface is a blending of local patches. A tensor product surface with $n\times m$ local patches can be divided into $n-1\times m-1$ "rectangular" parts which is a blending of a "quarter" of 4 local patches (except for the corner patches which are complete).

For polynomial-based local surfaces we use again two different representations: Hermite interpolation

$$f(t)=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{r=0}^{k_i}\sum_{s=0}^{k_j}p_{ijrs}T_r(\frac{u-u_{i-1}}{u_{i+1}-u_{i-1}})T_s(\frac{v-v_{j-1}}{v_{j+1}-v_{j-1}})B_i(u)B_j(v), \tag{18}$$

where $T_j(x)$ is defined by

$$T_j(x)=\frac{x^j}{j!}. \tag{19}$$

is the Taylor monomial basis;

and Bernstein-Bezier type fitting

$$f(t)=\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{r=0}^{k_{1i}}\sum_{s=0}^{k_{2j}}c_{ijrs}b_{k_{1i},r}(\frac{u-u_{i-1}}{u_{i+1}-u_{i-1}})b_{k_{2j},s}(\frac{v-v_{j-1}}{v_{j+1}-v_{j-1}})B_i(u)B_j(v), \tag{20}$$

where $b_{k_i,j}(x)$ is defined by

$$b_{k_i,j}(x)=\left(\begin{array}{c}k_i\\ j\end{array}\right)x^j(1-x)^{k_i-j}. \tag{21}$$

is the Bernstein polynomial basis of degree k_i. Our current application generates a surface by using Hermite interpolation, then transforms the local patches to Bezier type, and then editing can be done on the geometry.

A remarkable property of ERBS curves and surfaces is that for the case of Lagrange interpolation (Hermite interpolation of smoothness order 0) the resulting ERBS curves and surfaces geometrically coincide with the respective curves and surfaces obtained by Lagrange interpolation with piecewise affine (i.e., first degree algebraic polynomial) B-splines; however, the ERBS parametrization of these G^0-continuous geometric manifolds is C^∞-smooth. *This is what makes possible to extend the use of classical variational calculus based on the Euler-Lagrange and Hamiltonian formalisms to the case of of piecewise affine manifolds which are only* G^0*-continuous.*

5. Modelling the Cost Function for Discrete Data

In the previous section we considered Hermite interpolation and Bernstein-Bezier interpolation for the local surface. Now we will discuss another way of interpolation – the Lagrange interpolation and Newton interpolation, due to the discrete character of the data sets available for the surface. Namely, the parametrization $\mathbf{r}(u,v)$, $(u,v) \in [a,b] \times [c,d]$, $-\infty < a < b < \infty$, $-\infty < c < d < \infty$,, of the surface is assumed to be known only in a discrete (rectangular) grid of points

$$\{(u_i, v_j) : a \leq u_0 < u_1 < \cdots < u_n \leq b,\ c \leq v_0 < v_1 < \cdots < v_m \leq d\},$$

where the distances $u_i - u_{i-1}$ and $v_j - v_{j-1}$ are "sufficiently small". We also assume that only information about the values of $\mathbf{r}(u_i, v_j)$ is available, i.e., Lagrange interpolation (in the case of Hermite interpolation in the knots of the grid we can use directly (18), (19) The case of Lagrange interpolation can be implemented in one of the following two ways.

- It is possible to use directly the Taylor expansion of the local curve and surface patches in (18), (19) for Hermite interpolation with Hermite multiplicities $k_i = k_j = 0$ for all i, j

- It is possible to 'emulate' the Hermite data in (18), (19) by replacing the Taylor polynomials in (18), (19) with Lagrange polynomials

$$L_n(x) = \sum_{k=0}^{n} l_k(x) f_k = \sum_{k=0}^{n} \frac{\lambda_k(x)}{\lambda_k(x_k)} f_k \tag{22}$$

  where

$$\lambda_k(x) = (x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n) \tag{23}$$

  or Newton interpolation form

$$\begin{aligned} N_n(x) = f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] \\ + \cdots (x - x_0)(x - x_1) \cdots (x - x_{n-1}) f[x_0, \cdots, x_n]. \end{aligned} \tag{24}$$

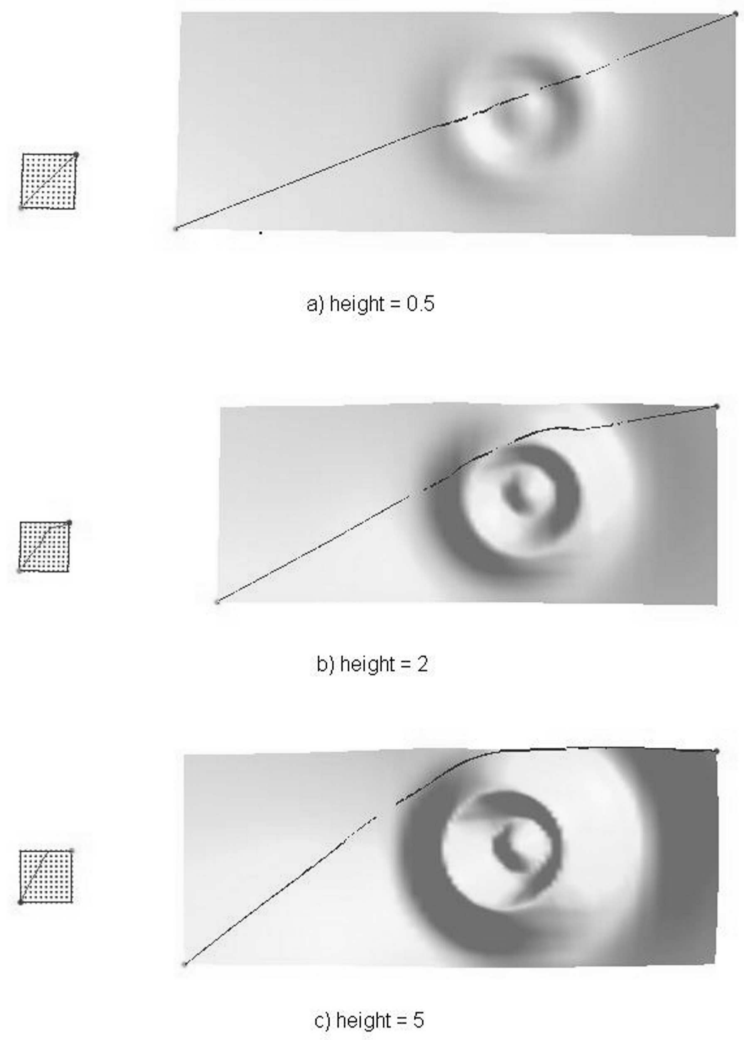

Figure 2: Geodesics (right-hand row of images) with the same terminal points in phase space (the parametric domain) (left-hand row of images) for Mexican Hat surfaces 'with different depth of the hat'.

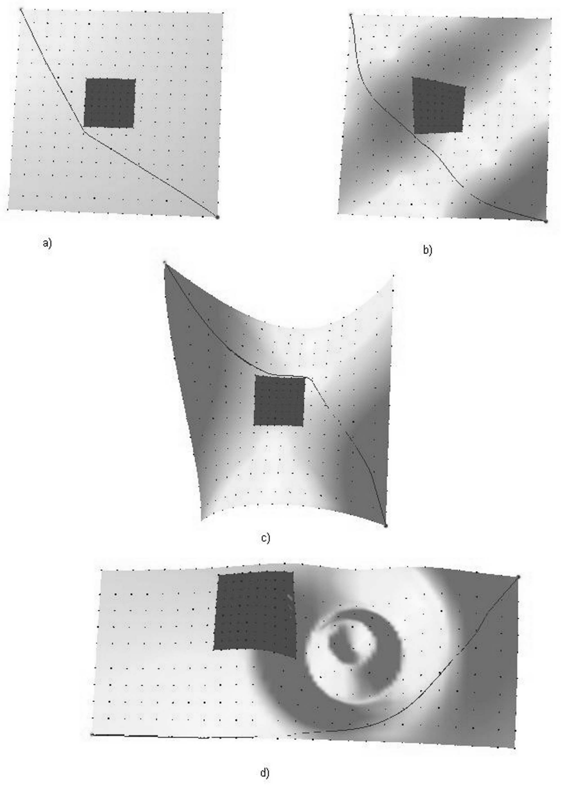

Figure 3: Constrained geodesics. The red (dark) area is forbidden. The parametric domain (phase space) and the 3D surface lifted over it are superimposed. The forbidden area used in the dynamical programming algorithm is the planar rectangle in the parametric domain; the forbidden area used in the visualization is the red (dark) tensor-product patch on the 3D surface which is the part of the surface defined over the forbidden area in the parametric domain. Note that the presence of forbidden area may result in considerable, global, change of the geodesic (including, possibly, changes in its topology). (a) Linear (affine) surface (i.e., plane); (b-d) nonlinear (non-affine) surfaces. In the case of the Mexican Hat surface (d), note that the unconstrained geodesic would pass 'on the other side of the top of the hat', compared to the constrained geodesic, given here.

using the divided differences

$$f[x_0,\cdots,x_k] = \frac{f[x_1,\cdots,x_k] - f[x_0,\cdots,x_{k-1}]}{x_k - x_0}. \tag{25}$$

Here $x = u$ or $x = v$; the parameter n in (22), (24) corresponds to a multiplicity k_i or k_j in (18), (19). In case of uniform grid in u and/or v, Newton forward and backward finite difference interpolation forms can be used in the place of (22), (24). In this way, the coefficients in the Taylor expansion, which have the meaning of derivatives, are being replaced by the respective coefficients of the Lagrange expansion in the Newton interpolation form, which have the meaning of respective divided differences corresponding to the derivatives in the Taylor expansion. The resulting ERBS tensor-product surface is continuous, but may not be geometrically smooth (G-smooth); however, it retains C^2-smooth parametrization, and the method of [6] can still be applied to compute the geodesics also in the geometrically non-smooth case.

5.1. Narrowing the Cross Sections of the Grid

So far in this exposition we have considered enhancements related to the cost functional and the constraints of the optimization problem. In this section we shall consider an enhancement of the search in the cross sections of the dynamical programming grid. This enhancement can be used at the initial stage of global search in the initial coarse global grid, as defined in [6], but its main positive impact for reducing the overall complexity of the multigrid algorithm is in the the second stage of iterative local search, i. e., in the global search within the local iterative refinements of the grid (see [6]). The approach we shall be using here is a variant of the A*-search algorithm discussed in [10], and for brevity we shall call it 'the A**-method'. The idea of this method is to narrow the cross sections of the dynamical programming mesh using a modification of the 'divide and conquer' approach, as follows.

1. Set the boundary and constraints on the grid.

2. From one vertex the grid it is only possible to go to its neighbours.

3. Begin with the initial vertex of the grid (the only vertex in the initial cross section of the current dynamical programming grid).

4. For every vertex, compute the cost values from each of its neighbors, and record the minimal value and the neighbour.

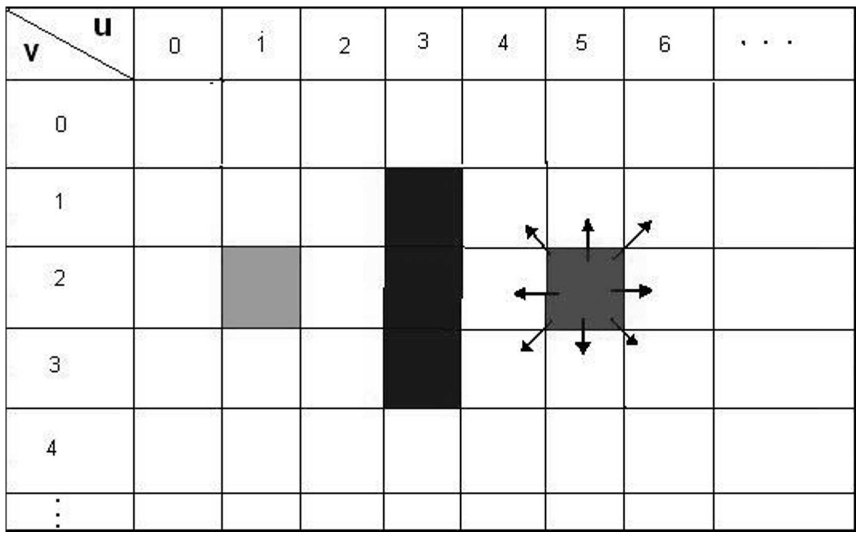

Figure 4: Matrix model of the optimization grid. The rectangle in i-th row and j-th column is the (i, j)-th vertex of the rectangular dynamical programming grid in the parametric domain (phase space).

5. After performing these computations for all vertices, according to the kept records, construct the optimal curve from the starting vertex to the end point.

As an illustration, let us consider in detail a simple model example. In Figure 4 is given a matrix model of the surface. The green square vertex (left) is the start vertex, the red square vertex (right) is the end vertex. The blue (dark) vertices between them correspond to a constrained region (for example, forbidden areas or areas with restriction on the speed of parametrization). This information is saved in a matrix.

Before computing, set the value of the cost function for the end vertex at 0, and for the other vertices equal to -1. The matrix will be:

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & \mathbf{-1} & -1 & -1 & \mathbf{0} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

On each of the subsequent iterations, assume that it costs 1 to move from the end vertex to its neighbors. On the 1-st iteration the matrix will be:

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & \mathbf{0} & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

On the 2-nd iteration it becomes relevant that the points (3,1), (3,2) and (3,3) are constrained. The matrix will be:

$$\begin{pmatrix} -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & \mathbf{0} & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 2 & 2 & 2 & 2 \end{pmatrix}$$

On the 3-rd iteration, the matrix will be:

$$\begin{pmatrix} -1 & -1 & 3 & 2 & 2 & 2 & 2 \\ -1 & -1 & 3 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & \mathbf{0} & 1 \\ -1 & -1 & 3 & -1 & 1 & 1 & 1 \\ -1 & -1 & 3 & 2 & 2 & 2 & 2 \end{pmatrix}$$

On the 4-th iteration, the matrix becomes:

$$\begin{pmatrix} -1 & 4 & 3 & 2 & 2 & 2 & 2 \\ -1 & 4 & 3 & -1 & 1 & 1 & 1 \\ -1 & 4 & 4 & -1 & 1 & \mathbf{0} & 1 \\ -1 & 4 & 3 & -1 & 1 & 1 & 1 \\ -1 & 4 & 3 & 2 & 2 & 2 & 2 \end{pmatrix}$$

On the last, 5-th iteration, the matrix will be:

$$\begin{pmatrix} 5 & 4 & 3 & 2 & 2 & 2 & 2 \\ 5 & 4 & 3 & -1 & 1 & 1 & 1 \\ 5 & \mathbf{4} & 4 & -1 & 1 & \mathbf{0} & 1 \\ 5 & 4 & 3 & -1 & 1 & 1 & 1 \\ 5 & 4 & 3 & 2 & 2 & 2 & 2 \end{pmatrix}$$

After the completion of the last iteration, the value of the start vertex (1,2) is found to be 4, and the path (1,2)–(2,1)–(3,0)–(4,1)–(5,2) is found to be optimal. In this example, there is also another optimal path (1,2)–(2,3)–(3,4)–(4,3)–(5,2), but the present version of the proposed A**-method only finds one of the solutions.

From this analysis it becomes clear that the proposed A** algorithm is equivalent to the dynamical programming algorithm on the coarse grid in [6] but in this case the 0th, 1st, 2nd, etc, cross-section in the dynamical programming algorithm is the 0th, 1st, 2nd etc "neighborhood" of the initial of the two boundary (terminal) points of the optimal curve in quest. For the example given above:

- 0th neighbourhood = (5,2) (marked with 0 in the first matrix)
- 1st neighbourhood = (4,1), (5,1), (6,1), (4,2), (6,2), (4,3), (5,3), (6,3) (marked with 1 in the second matrix)
- 2nd neighbourhood = (3,0), (4,0), (5,0), (6,0), (3,4), (4,4), (5,4), (6,4) (marked with 2 in the third matrix)
- 3rd neighbourhood = (2,0), (2,1), (2,3), (2,4) (marked with 3 in the matrix)
- 4th neighborhood = (2,2), (1,0), (1,1), (1,2), (1,3), (1,4) (marked with 4 in the matrix)
- 5th neighbourhood = (0,0),(0,1),(0,2),(0,3),(0,4) (marked with 5 in the matrix)

The cross section of the A**-method (dynamical programming algorithm) is obtained only after the initial vertex has been assigned into a neighborhood. In the example above, this vertex is (1,2) and it falls in the 4-th neighborhood. Then the dynamical programming algorithm for the above example has 5 cross sections (cross section 0-4) which are the 0th, 1st, 2nd, 3rd, 4th neighborhood, respectively while the 5th neighborhood is ignored. In comparison with this, in the coarse global search in [6] the 0th and the 4th cross section would be (5,2) and (1,2), respectively, while the 1st, 2nd and 3rd cross section would each be the whole matrix in the above example. In the method used here, the cross sections are smaller in size (the last 4-th cross section is also constrained only to (1,2) because this is the initial point.)

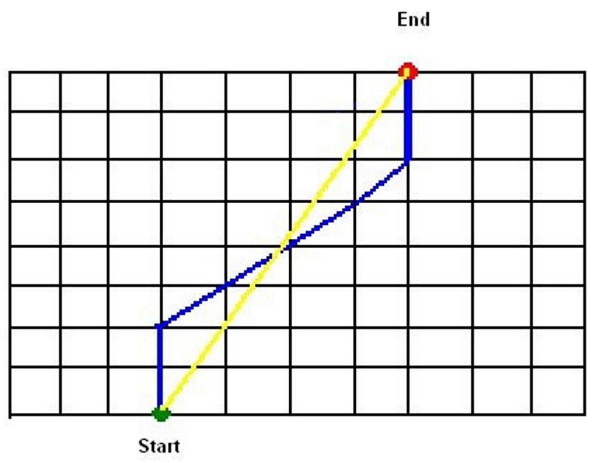

Figure 5: According to the A**-method, the blue (darker) curve is the approximate optimal solution (in phase space). The yellow (lighter) curve is the accurate optimal solution (in phase space). The corresponding approximate and accurate geodesics on the 3D surface are obtained by 'lifting' the two planar curves given here to their respective curves on the 3D surface.

The global optimal curve can be found using this method, but due to the high complexity of the 'divide and conquer' search the grid must be rather coarse, and the respective result will be obtained with low resolutiion (see, e.g., Figure 5). The result can be improved by using the A**-method for the search in the local refinements of the grid, as discussed in the beginning of this section.

When the parametrization is periodic (for example sphere, torus, other surfaces without boundary) it is necessary to also define the relationship of a neighbour on the boundary to the neighbours in the "next period" of the parametric domain.

In conclusion of this section, let us emphasize that the enhancement based on the A**-method is completely independent of the enhancements related to the cost functional and the constraints, as discussed in the previous sections. It is possible and meaningful to implement any one of these enhancements without implementing the others.

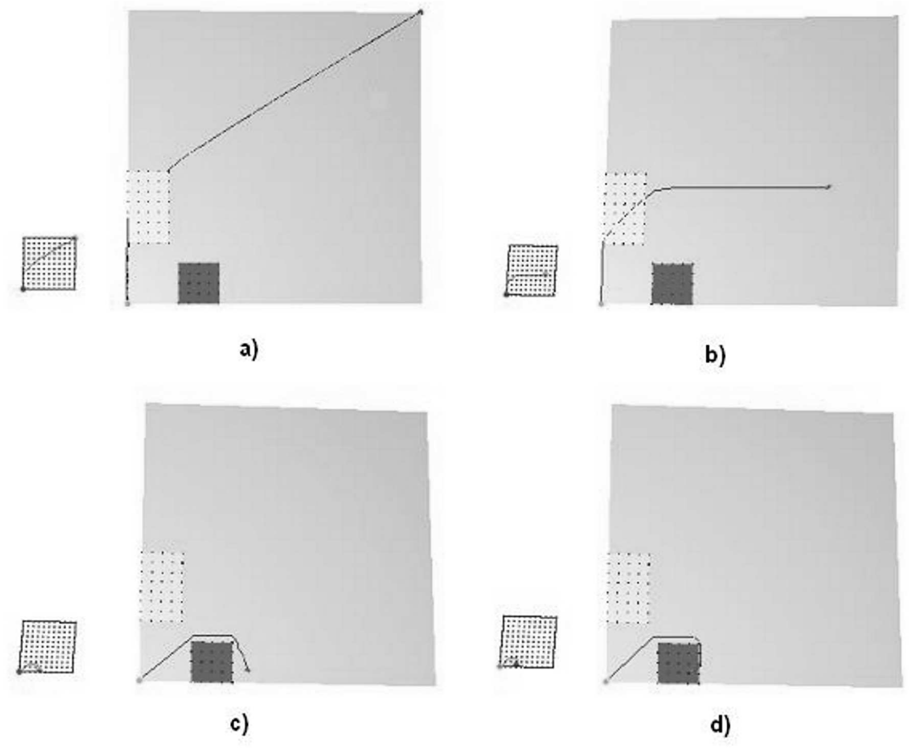

Figure 6: Shortest-time curves with constraints. The light blue (light) one is a high-speed region. The violet (dark) one is a slow-speed region. The parametric domain (phase space) and the 3D surface lifted over it are superimposed, similar to Figure 3. However, in the present figure a separate reduced copy of the parametric domain is given 'at the south-west corner' of the picture of each of the cases (a-d), with the graph of the optimal trajectory in phase space.

6. Examples

In Figure 2 (in the right hand column) are given the computed geodesic lines for a surface with the same parametrization, $z = f(x, y)$ but with different scaling of the vertical coordinate z. In the left-hand column are given the respective optimal curves in the parametric phase space $\{(x, y)\}$

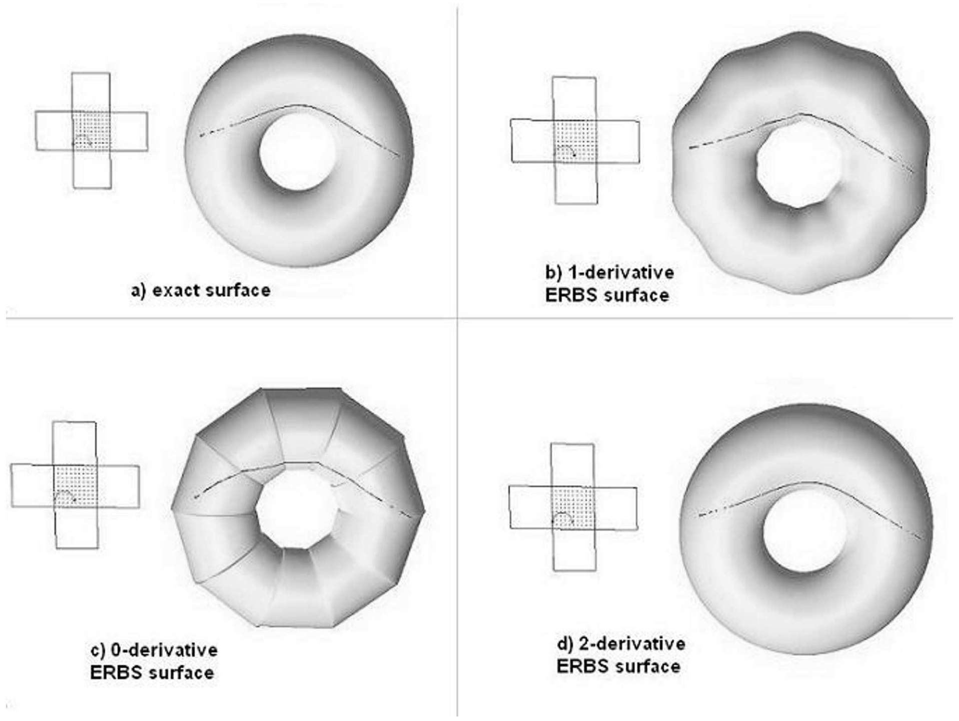
a) exact surface
b) 1-derivative ERBS surface
c) 0-derivative ERBS surface
d) 2-derivative ERBS surface

Figure 7: Geodesics on the exact torus surface and approximate geodesics on its approximating ERBS surfaces (the right-hand images in pictures (a-d)). In the left-hand side of each of pictures (a-d) are the images of the parametric domain (phase space), extended with neighbouring copies in each of the two parameters of the surface, due to the periodicity of the parametrization of the torus in each of the two parameters. Note that the ERBS surfaces in (b), (c) and (d) are only G^1-smooth, G^0-continuous and G^2-smooth, respectively, but their parametrizations are all C^∞-smooth.

In Figure 3 is given geodesic for the same surface as in Figure 2, but now in the presence of a constraint.

In Figure 6 are given shortest-time optimal curves in the presence of 'fast', 'average' and 'slow' environment. This type of problem has applications related to Fermat's variational principle in the geometric optics of wave diffraction and deflection.

In Figures 7 and 8 are given examples of geodesics on smooth surfaces and

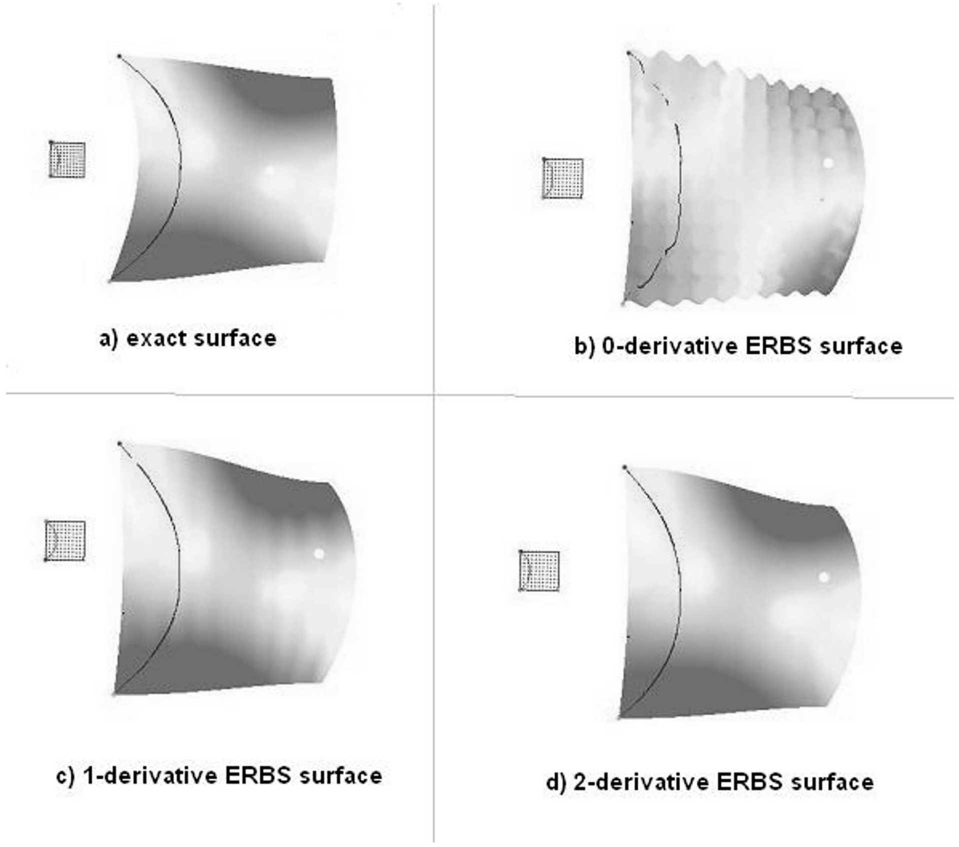
a) exact surface
b) 0-derivative ERBS surface
c) 1-derivative ERBS surface
d) 2-derivative ERBS surface

Figure 8: Geodesics on the exact saddlepoint surface and approximate geodesics on its approximating ERBS surfaces (the right-hand images in pictures (a-d)). In the left-hand side of each of pictures (a-d) are the images of the parametric domain (phase space). (The parametrization is not periodic in any of the two parameters of the surface, and no extensions of the parametric domain are needed.) Note that the ERBS surfaces in (b), (c) and (d) are only G^0-continuous, G^1-smooth and G^2-smooth, respectively, but their parametrizations are all C^∞-smooth.

their G^2-smooth (Hermite interpolation, derivatives up to order 2), G^1-smooth (Hermite interpolation, derivatives up to order 1), and G^0-continuous (Lagrange interpolation, 'derivatives of order 0'). It is seen that the optimal curve in the parametric phase space and the geodesic curve on the surface are successfully computed on the base of the C^2 model (5) also in the geometrically non-smooth

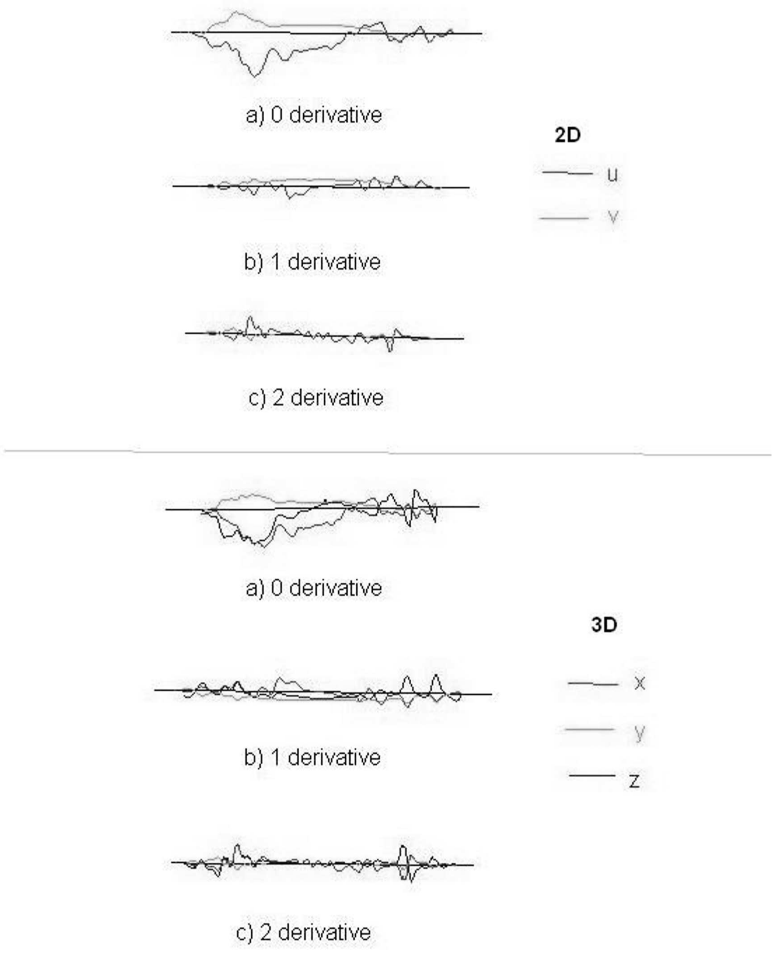

Figure 9: The error of geodesic computation between the exact saddle-point surface and its approximating ERBS surfaces. The 2D case refers to the planar optimal curve in the parametric domain (phase space); the 3D case refers to the geodesic (generally, non-planar) curve obtained by lifting the planar curve in the 2D case to its image on the 3D surface.

case. (For more details about the comparison between analytic and geometric regularity (C- and G-regularity) see, e. g., [9].)

The multiple copies of the parametric phase space in the case of the torus on Figure 7 are due to the periodicity of the parametrization, see [6].

In Figure 9 is visualized the error made by approximating the exact saddle-point surface in Figure 8 (a) by the respective surface on Figure 8 (b), (c) and (d). The upper half of Figure 9 refers to the error in the computation of the optimal curve in parametric phase space; the lower half of Figure 9 visualizes the computation error for the geodesic line on the surface.

7. Conclusion and Future Work

This paper discusses and implements two main tasks: making ERBS surfaces from discrete rectangular data sets (which can be generated from parametric surfaces or be the result of measurements or numerical computations), and finding the optimal curves on the surfaces. Of course, there are many methods to solve these problems. Using the method proposed here, we can solve the problems accurately and efficiently, using a uniform approach for both smooth and non-smooth surfaces.

Further work includes, but is not limited to, the following:

- using the ERBS with scattered data (triangulated domains);
- other types of cost criteria;
- other constraints.

The most important conclusion made during the work on this study, which from the very beginning was also the main reason to choose ERBS over other more established tools of geometric modeling (such as, classical polynomial B-splines or NURBS) for generating parametrization by interpolating discrete data sets, is that ERBS provide a sufficiently smooth parametrization (even C^{∞}-smooth one) even in the cases when the surface is only G^0-continuous. A remarkable consequence of this is that we are able to obtain optimal curves using standard methods designed for smooth surfaces even in the case when these surfaces are not more than G^0-continuous. Here we considered only tensor-product surfaces, and the G^0-continuous surfaces were obtained by Lagrange interpolation (only positions interpolated at each data point; no derivatives). In this case the surface consists of patches which are bilinear, and are connected continuously, but not smoothly, together. An even more interesting case in this aspect is, if, instead of tensor-product surfaces, one studies triangulated surfaces, in which case the patches will be linear, again connected together continuously, but not smoothly, at the edges. For this case, the classical methods for solving variational problems (including direct methods like dynamic programming) will provide much more efficient methods for computing geodesics or other optimal curves on triangulated surfaces than the methods in [14] (see section 6 in [6]). Another advantage of the present algorithm compared to the methods on triangulated surfaces, as discussed in [14] is its easy upgrading to 3D volume deformations and higher-dimensional manifolds.

The use of expo-rational B-splines allows a simple uniform classical variational approach to the handling of both very smooth and non-smooth geometrical data, including cases where subgradient methods for non-smooth optimization do not work (for example, for cost functionals and/or constraints in the Besov (Hölder) space Lip-α with $0 < \alpha < 1$).

The extension of the definition of expo-rational B-splines for triangulated manifolds (see [5]) allows the method considered here for tensor-product surfaces to be extended for general triangulated manifolds. It also allows to upgrade the method of geodesic Bezier curves proposed in [11] so that the challenges of handling non-smooth and very smooth manifolds (see Subsection 7.1 in [11]) are simultaneously overcome in a uniform way. The uniformity and robustness of the expo-rational B-spline parametrization has considerable potential even in cases where parametric models are considered less efficient than implicit and other non-parametric geometric representations (see, e.g., [13]).

Returning on the topic of this study in the context of triangulated surfaces is, therefore, one of the main directions of future work in connection with computing optimal curves on surfaces.

Acknowledgements

This work was partially supported by the 2006 and 2007 Annual Research Grants of the R&D Group for Mathematical Modeling, Numerical Simulation and Computer Visualization at Narvik University College, Norway.

The research of Xiang F. Bu was part of the work which he did under the supervision of Lubomir T. Dechevsky, and with co-supervision by Børre Bang and Arne Lakså, in relation with his M. Sc. Diploma thesis at Narvik University

College.

References

[1] R. Bellman, *Dynamic Programming*, Dover Publications, Mineola, NY (2003).

[2] R.E. Bellman, S.E. Dreyfus, *Applied Dynamic Programming*, Princeton Univ. Press, Princeton, NJ (1962).

[3] X.F. Bu, *Dynamical Programming for Computing of Optimal Curves on Expo-rational B-spline (ERBS) Surfaces*, M.Sc. Diploma Thesis, Narvik University College, Main supervisor: L.T. Dechevsky; Co-supervisors: B. Bang, A. Lakså). Narvik, Norway (2007).

[4] L.T. Dechevski, G.P. Andreeva, I.V. Daskalov, On a method for computer-aided local and global optimization of railway vertical alignment, In: *Numerical Methods and Applications'1984* (Ed-s: Bl.H. Sendov, R.D. Lazarov), Publishing House of the Bulgarian Academy of Sciences, Sofia (1985), 232-241.

[5] L.T. Dechevsky, Generalized expo-rational B-splines for Hermite interpolation on scattered-point sets in domains in $\mathbb{R}^n$, II: Simplicial and convex polygonal partitions, duality, *International Journal of Pure and Applied Mathematics*, To Appear.

[6] L.T. Dechevsky, L.M. Gulliksen, A multigrid dynamical programming algorithm for discrete dynamical systems and its applications to numerical computation of global geodesics, *International Journal of Pure and Applied Mathematics*, **33**, No. 2 (2006), 257-285.

[7] L.T. Dechevsky, L.M. Gulliksen, Application of a multigrid dynamical programming algorithm to optimal parametrization, and a model solution of an industrial problem, *International Journal of Pure and Applied Mathematics*, **33**, No. 3 (2006), 381-406.

[8] L.T. Dechevsky, A. Lakså, B. Bang, Expo-rational B-splines, *International Journal of Pure and Applied Mathematics*, **27**, No. 3 (2006), 319-369.

[9] G. Farin, *Curves and Surfaces for CAGD. A Practical Guide*, 5-th Edition, Morgan Kaufmann Publ. and Academic Press, San Francisco and San Diego, CA (2002).

[10] http://en.wikipedia.org/wiki/A*

[11] D. Martinez, P. Carvalho, L. Velho, Geodesic Bezier curves: a tool for modeling on triangulations, *Technical Report* TR-07-04, April 07, Relatório Técnico, Laboratório VISGRAF, Instituto de Matemática Pura e Aplicada, Rio de Janeiro, Brasil.

[12] N.N. Moiseev, *Elements of the Theory of Optimal Systems*, Nauka, Moscow (1975), In Russian.

[13] N. Paragios, R. Deriche, Geodesic active regions and level set methods for motion estimation and tracking, *Computer Vision and Image Understanding*, **97** (2005), 259-282.

[14] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, H. Hoppe, Fast exact and approximate geodesics on meshes, *ACM Transactions on Graphics*, **44**, No. 3 (2005), 553-560.

[15] M.M. Vainberg, *Functional Analysis*, Prosveshchenie, Moscow (1979), In Russian.