Tensor Product B-Spline Mesh Generation for Accurate Surface Visualizations in the NIST Digital Library of Mathematical Functions

Bonita Saunders and Qiming Wang

National Institute of Standards and Technology, Gaithersburg MD 20899, USA
bonita.saunders@nist.gov
http://math.nist.gov/~BSaunders

Abstract. We discuss the use of tensor product B-spline meshes to design accurate three dimensional graphs of high level mathematical functions for the National Institute of Standards and Technology (NIST) Digital Library of Mathematical Functions. The graph data is placed inside a web based format such as VRML (Virtual Reality Modeling Language) or X3D (Extensible 3D) to create an interactive visualization environment where users can carefully examine complicated function attributes such as zeros, branch cuts, poles and other singularities. We describe the grid generation technique and its effectiveness for creating clear and informative visualizations.

1 Introduction

One of the most cited publications from the National Institute of Standards and Technology (NIST) is the Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables first published in 1964 [1]. Currently, NIST is completely rewriting the handbook and will release it in both hardcopy and web-based format as the NIST Digital Library of Mathematical Functions (DLMF). The DLMF will include formulas, methods of computation, references, and software information for nearly forty high level, or special, mathematical functions. The website will feature interactive navigation, a mathematical equation search and dynamic interactive visualizations.

This paper discusses our use of mesh generation techniques to facilitate the accurate 3D plotting of functions over irregular, discontinuous, or multiply connected domains for the visualizations. By modifying an algebraic tensor product spline mesh generation algorithm that we originally designed for problems in aerodynamics and solidification theory, we have created boundary/contour fitted computational grids that capture significant function features such as zeros, poles, branch cuts and other singularities.

While the DLMF visualizations are not designed to compete with the on-demand computational and plotting capabilities provided by many well-known computer algebra packages, the motivation for our work stems from some of the inadequacies of such packages. Commercial packages often have many built-in special functions, but their 3D plots are generally over a rectangular Cartesian mesh, leading to poor and misleading graphs. Also, many packages have trouble properly clipping a function, that is, accurately restricting the graphical representation to the points lying within the range

M. Dæhlen et al. (Eds.): MMCS 2008, LNCS 5862, pp. 385–393, 2010.

of interest. Furthermore, some packages may display the properly clipped graph inside the package, but provide no acceptable way to export the clipped data for use outside the package. This is particularly important if the data needs to be transformed to a web-based format such as X3D (Extensible 3D) or VRML (Virtual Reality Modeling Language).

We show how our mesh generation algorithm eliminates or lessens the severity of many of these problems. We also talk about progress in making the method more adaptive and discuss other possibilities for improvements.

2 Constructing 3D Graphs for Interactive Visualizations in a Digital Library

The first release of the NIST DLMF will consist of thirty eight chapters authored by special function experts throughout the U. S. and abroad. As primary developers of the visualizations that will be an integral part of the DLMF, we have chosen the number and location of graphics for each chapter after consultation with the authors and DLMF editors. To ensure data accuracy, the plot data is being validated by computing each function by at least two different methods, using commercial packages, publicly available codes, or the author's personal codes. However, another concern is plot accuracy, that is, we want to make sure that the displayed plot accurately represents the graph of the function.

Most commercial packages produce very accurate 2D plots. They typically handle discontinuities automatically or fairly easily with special options, and the packages properly cut, or clip, curves so that only points within the desired range appear. Unfortunately, this is often not the case in 3D. In Figure 1, the plot of the Struve function $L_\nu(x)$, rendered using a popular commercial package, illustrates some of the problems we have encountered. We wanted the plot to show the graph for function values less than or equal to 10, but there is a flat area, or shelf, where values greater than 10 have been set to 10. Also, areas near the poles are not resolved very well. Although the shelf-like area may not concern many users, it might be confusing to students and others unfamiliar with the behaviour of the function. Inputting a command to remove the shelf produces a saw-tooth area that can be even more misleading.

Experienced users can input alternative commands to successfully clip the function properly, but we found that this was still not sufficient for our requirements. Since the computation of the function is over a rectangular Cartesian mesh, the figure can look fine when viewed inside the package, but produce a very irregular color map when the data is transformed to another format, such as VRML (Virtual Reality Modeling Language) for viewing on the web. VRML and its successor, X3D (Extensible 3D) are standard 3D file formats for interactive web-based visualizations [12], [13], [14]. Freely available plugins can be downloaded for most platforms. Of course the figure can be improved by using a much larger number of mesh points, but large data files hamper the performance of our visualizations. The rendering problems can be eliminated or decreased in severity by computing the function over a specially designed boundary/contour fitted mesh. While various unstructured techniques such as Voronoi or Delaunay triangulations and quadtree designs might be used to construct such a mesh, a structured

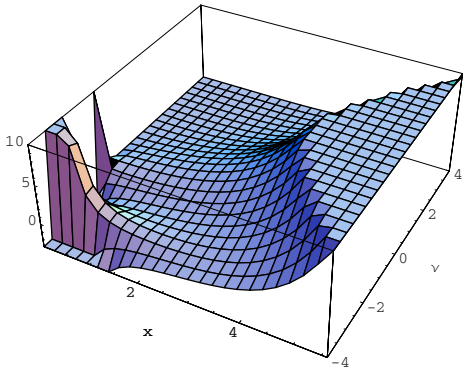

Fig. 1. Plot of Struve function L_v illustrating bad clipping and poor resolution of poles

design allowed us to write more efficient code to implement the interactive capabilities for the visualizations. Whether a structured or unstructured grid is used, it is important that the interior grid lines not stray too far from the contours of the function being plotted. The next section discusses the tensor product B-spline mapping we have used to produce our meshes.

3 Grid Generation Mapping

The complexity of the grid generation problem depends not only on the attributes of the computational domain of the special function, but also on the behaviour of the function. Function domains can range from simple rectangles to complicated multipy connected domains with poles or branch cuts; however, large function gradients can further complicate the creation of a suitable computational grid. Our mesh generation technique is based on an algorithm developed by one of the authors for meshes to be used in solving partial differential equations (pdes) related to aerodynamics and solidification theory [4], [5], [6]. The algorithm constructs a curvilinear coordinate system defined by a mapping $\mathbf{T}$ from the unit square I_2 to a physical domain of arbitrary shape. We let

$$\mathbf{T}(\xi,\eta) = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) \\ \sum_{i=1}^{m}\sum_{j=1}^{n} \beta_{ij} B_{ij}(\xi,\eta) \end{pmatrix}, \tag{1}$$

where $0 \le \xi,\eta \le 1$ and each B_{ij} is the tensor product of cubic B-splines. Hence, $B_{ij}(\xi,\eta) = B_i(\xi)B_j(\eta)$ where B_i and B_j are elements of cubic B-spline sequences associated with finite nondecreasing knot sequences, say, $\{s_i\}_1^{m+4}$ and $\{t_j\}_1^{n+4}$, respectively. To obtain the initial α_{ij} and β_{ij} coefficients for $\mathbf{T}$, we first construct the transfinite blending function interpolant that matches the boundary of the physical domain. The interior coefficients are obtained by evaluating the interpolant at knot averages as described in [2] to produce a shape preserving approximation that reproduces straight lines and preserves convexity. For the boundary coefficients, we use the same technique

if the boundary side is a straight line or use DeBoor's SPLINT routine [2] to find coefficients that produce a cubic spline interpolant of the boundary. For simple boundaries, the initial grid is often sufficient, but for more complicated or highly nonconvex boundaries we can improve the grid by using a variational technique to find coefficients that minimize the functional

$$F = \int_{I_2} \left(w_1 \left\{ \left(\frac{\partial J}{\partial \xi} \right)^2 + \left(\frac{\partial J}{\partial \eta} \right)^2 \right\} + w_2 \left\{ \frac{\partial \mathbf{T}}{\partial \xi} \cdot \frac{\partial \mathbf{T}}{\partial \eta} \right\}^2 + w_3 \{uJ^2\} \right) dA \qquad (2)$$

where $\mathbf{T}$ denotes the grid generation mapping, J is the Jacobian of the mapping, w_1, w_2 and w_3 are weight constants, and u represents external criteria for adapting the grid. The integral controls mesh smoothness, orthogonality, and depending on the definition of u, the adaptive concentration of the grid lines. When solving pdes, u might be the gradient of the evolving solution or an approximation of truncation error. Ideally, for our problem, we want u to be based on curvature and gradient information related to the function surface. We have made a slight modification of the integral we gave in [7]. The adaptive term $w_3\{Ju\}$ has been replaced by $w_3\{uJ^2\}$. The new term produces a better concentration of grid lines, and also can be shown to be equivalent to the adaptive term used in the variational technique of Brackbill and Saltzman [3], [8].

To avoid solving the Euler equations for the variational problem, F is approximated in the computer code by the sum

$$\begin{aligned} G = & \sum_{i,j} w_1 \left[\left(\frac{J_{i+1,j} - J_{ij}}{\triangle \xi} \right)^2 + \left(\frac{J_{i,j+1} - J_{ij}}{\triangle \eta} \right)^2 \right] \triangle \xi \triangle \eta \\ & + \sum_{i,j} w_2 Dot_{ij}^2 \triangle \xi \triangle \eta \\ & + \sum_{i,j} w_3 u_{ij} J_{ij}^2 \triangle \xi \triangle \eta \end{aligned} \qquad (3)$$

where J_{ij} is the Jacobian value, u_{ij} is the value of u, and Dot_{ij} is the dot product of $\partial \mathbf{T}/\partial \xi$ and $\partial \mathbf{T}/\partial \eta$ at mesh point (ξ_i, η_j) on the unit square. When $w_3 = 0$, G is actually a fourth degree polynomial in each spline coefficient so the minimum can be found by using a cyclic coordinate descent technique which sequentially finds the minimum with respect to each coefficient. This technique allows the minimization routine to take advantage of the small support of B-splines when evaluating the sums that comprise G.

In Figure 2 we applied the algorithm to a puzzle shaped domain. The initial grid, constructed using linear Lagrange polynomials for the blending functions, is shown on the left. Note that the grid lines overlap the nonconvex boundary. The grid on the right shows the mesh obtained after the spline coefficients are modified to minimize G with weights $w_1 = 1$, $w_2 = 0.5$, and $w_3 = 0$. The overlapping grid lines have been pulled into the interior. When a grid line folds over a boundary, the Jacobian value changes sign. The Jacobian terms of G try to minimize the changes in the Jacobian between adjacent grid cells. Consequently, lines outside the boundary are pulled into the interior. In the original code, this effect was obtained by trying to compute the interval for each coefficient that would guarantee that the Jacobian remained positive [10]. However,

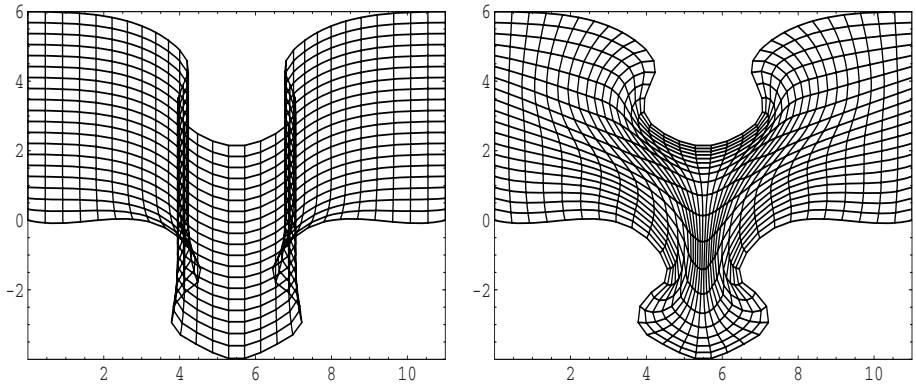

Fig. 2. Initial and optimized puzzle grids

experimenting showed this to be a time consuming procedure that was not needed in most cases. Gonsor and Grandine use the same idea to obtain similar results in [11].

4 Results

We have used boundary/contour fitted grid generation to produce computational grids for over two hundred 3D visualizations for the NIST DLMF. In Figure 3 we show a snapshot of the VRML rendering of the real-valued Struve function $L_\nu(x)$. The data values were obtained by computing the function over the contour grid shown on the left. To create the grid boundary we found the contour, or level, curves where $L_\nu = 10$, the maximum height we wanted to display, and $L_\nu = -3$, the minimum height. We then connected the curves to parts of a rectangle to form a closed boundary for the grid generation algorithm. In contrast to Figure 1, we show that the boundary/contour fitted grid produces a nicely clipped function and very smooth color map. The dark horizontal bar is actually a dense concentration of grid lines needed to accurately resolve the area around a pole. Figure 4 shows a density plot of L_ν that a user may obtain by scaling the surface down so that the vertical component is near zero. This is one of several interactive features that will be available to users of the DLMF.

Figure 5 shows a grid and surface for the modulus of the complex digamma, or psi, function $|\psi(z)|$. The top and bottom halves of the grid were generated separately, with an exponential function used to concentrate the grid points near $y = 0$. The accuracy of the contours satisfying $|\psi(z)| = 6$ is illustrated in the surface shown on the right. Since the coefficients define an explicit curvilinear coordinate system mapping, we can create a coarser, or finer, grid simply by evaluating fewer, or more, points on the unit square. Also, notice that as was seen in Figure 3, the grid spacing does not appear to be smooth in some areas. To guarantee that key boundary or contour points, such as those at zeros or corners, are maintained regardless of grid size, we identify "fixed points," that is, grid points that must always be kept. Grid lines are always drawn through these points. In most cases, the resulting discontinuities in cell spacing do not affect the quality of the

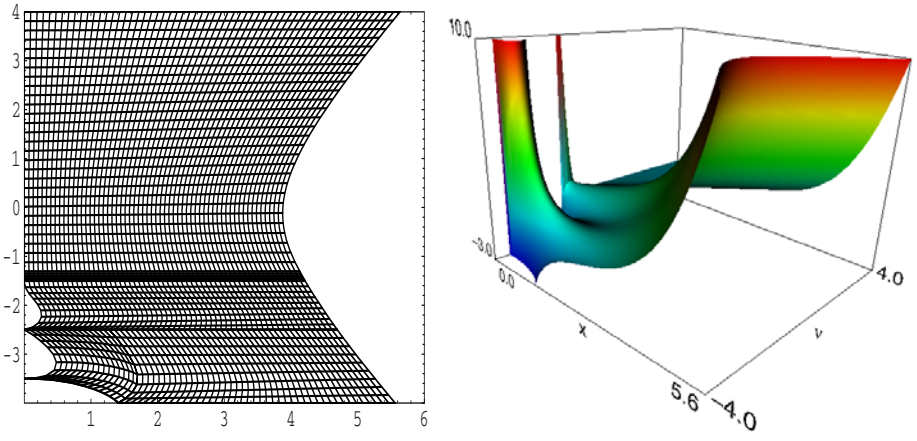

Fig. 3. Grid and Plot of Struve Function L_ν

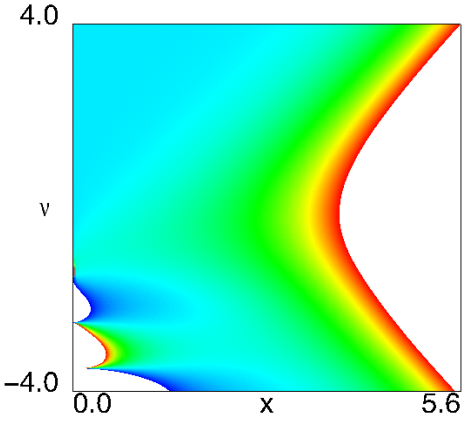

Fig. 4. Density Plot of Struve Function L_ν

visualizations significantly, but this is an aspect of the technique that might be improved by carefully chosen adaptive weights.

All the visualizations in the NIST DLMF represent either real-valued or complex-valued functions of the form, $w = f(x,y)$. For complex-valued functions, the user has the option of using a height based color mapping where height = $|w|$, or a mapping based on the phase, or argument, of w. Figure 6 shows a plot of the modulus of the Hankel function $H_1^{(1)}(z)$ with a phase based color map. On the left is a phase density plot obtained by scaling the figure down in the vertical direction. The branch cut, zeros and pole are visible in both figures.

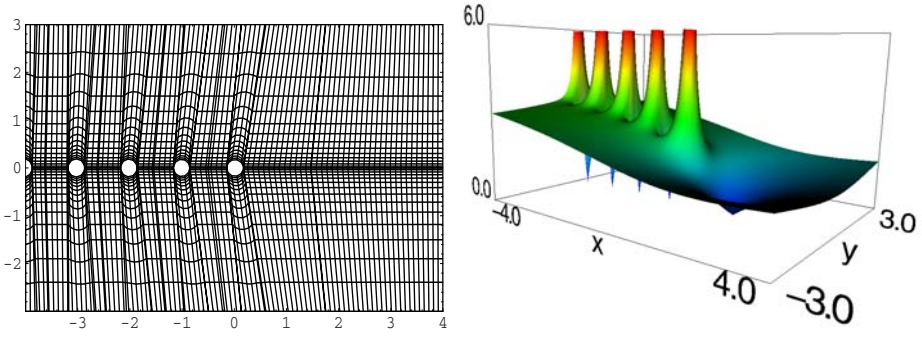

Fig. 5. Grid and Plot of Complex Digamma Function $|\psi(z)|$

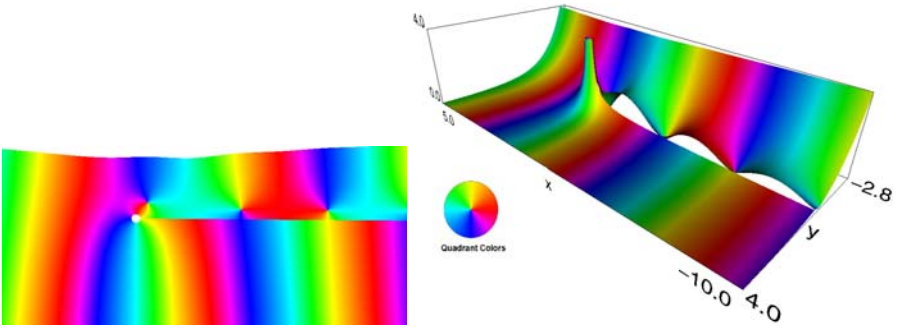

Fig. 6. Plot of Hankel function $H_1^{(1)}$ with phase colormap

In Figure 7 we attracted grid points in an equally spaced square grid to a circle and a sine curve, respectively. To obtain the attraction to the circle we defined the u in the adaptive term of our functional by

$$u(\xi,\eta) = e^{-500[(\xi-.5)^2+(\eta-.5)^2-\frac{9}{64}]^2} \tag{4}$$

and for attraction to the sine curve we defined

$$u(\xi,\eta) = e^{-100[0.35(\sin(2\pi\xi)+1.35)-\eta]^2}. \tag{5}$$

The parameters were chosen so that the curves lie completely within the unit square. Of course these definitions guarantee that the grid lines are attracted to a circle (or sine curve) in I_2 but not in the xy plane. Choosing u in this manner allowed us to quickly test the affect of u with few modifications to our code since the functional approximation G will still be a fourth degree polynomial in each spline coefficient. To show the affect of u we simply mapped the unit square to a larger square.

In order to reach our goal of adapting the grid lines to function curvature data, we must update the code to allow u to have a nonlinear dependence on the spline coefficients. We have made some progress in this direction by incorporating Nash's truncated

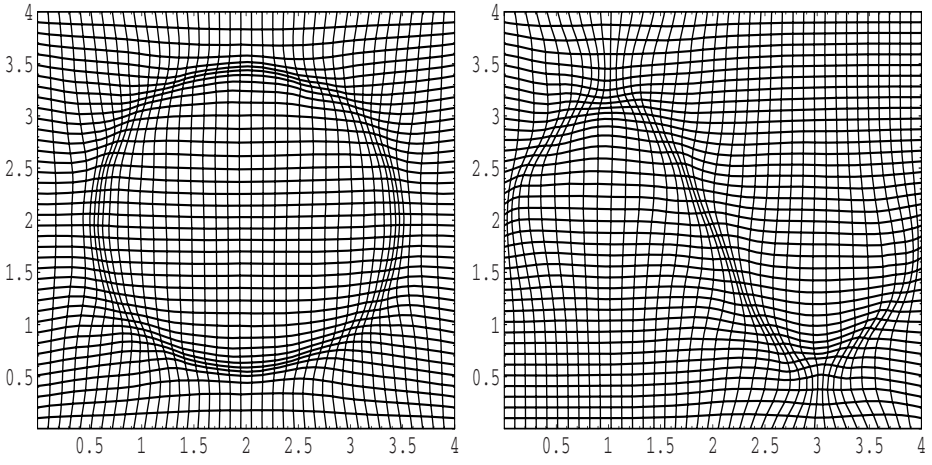

Fig. 7. Grids adapted to circular and sinusoidal shapes

Newton algorithm code described in [9], but additional modifications to take better advantage of the small support of B-splines are needed. We also need to look at the movement of the boundary points. Minimization of G should also include adjustment of the boundary coefficients, but this must be done very carefully so that we obtain a reparameterization of the boundary curves without altering the shape of the boundary. Then we can concentrate on defining an expression for u that captures the appropriate function gradient information. We are acutely aware that the specialized nature of some high level mathematical functions may mean that accessing and linking the codes needed to compute gradient and curvature data may not be a trivial task.

5 Conclusions

We have used boundary/contour fitted grid generation to complete over two hundred interactive 3D visualizations for the NIST Digital Library of Mathematical Functions. Our tensor product B-spline technique has been effective in addressing many problems that appear in commercial packages such as the inaccurate resolution of poles, bad clipping, and poor color mappings. The clarity of the surface color map is still an issue in areas where the gradient is large, but we believe this can be improved by using an adaptive technique based on function gradient and curvature information. The adaptive method should also result in smaller data file sizes, which can improve the efficiency of some of our interactive features.

In spite of tackling several issues simultaneously, including the production of the visualizations, data validation, the availability of VRML/X3D plugins, and accessibility on major platforms, we have made steady progress toward the development of the adaptive code. We have successfully adapted our grids to curves defined on the unit square and had some success with adaption to more general curves where the definition is influenced by the spline coefficients.

Our future work will include incorporating the movement of the boundary coefficients in the minimization functional, describing suitable adaption criteria that will capture gradient and curvature information, and integrating the computation of this function information with the grid generation code.

Disclaimer

All references to commercial products are provided only for clarification of the results presented. Their identification does not imply recommendation or endorsement by NIST.

References

1. Abramowitz, M., Stegun, I.A. (eds.): Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables. National Bureau of Standards Applied Mathematics Series, vol. 55. U.S. Government Printing Office, Washington (1964)
2. de Boor, C.: A Practical Guide to Splines, Revised Edition. Springer, New York (2001)
3. Brackbill, J.U., Saltzman, J.S.: Adaptive zoning for singular problems in two dimensions. J. Comput. Phys. 46, 342–368 (1982)
4. Saunders, B.V.: A boundary conforming grid generation system for interface tracking. Computers Math. Applic. 29, 1–17 (1995)
5. Saunders, B.V.: The application of numerical grid generation to problems in computational fluid dynamics. Council for African American Researchers in the Mathematical Sciences, Contemporary Mathematical Series, vol. III, 275. American Mathematical Society (2001)
6. Saunders, B.V., Wang, Q.: From 2d to 3d: numerical grid generation and the visualization of complex surfaces. In: Soni, B.K., et al. (eds.) Proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations, Whistler, British Columbia, Canada, September 25-28 (2000)
7. Saunders, B.V., Wang, Q.: From B-spline mesh generation to effective visualizations for the nist digital library of mathematical functions. In: Chenin, P., et al. (eds.) Curve and Surface Design: Avignon 2006. Nashboro Press, Brentwood (2007)
8. Thompson, J.F., Warsi, Z.U.A., Mastin, C.W.: Numerical Grid Generation: Foundations and Applications. North Holland, New York (1985)
9. Nash, S.G.: Newton-type minimization via the lanczos method. SIAM Journal on Numerical Analysis 21(4), 770–788 (1984)
10. Saunders, B.V.: Algebraic grid generation using tensor product B-splines, NASA CR-177968 (1985)
11. Gonsor, D., Grandine, T.: A curve blending algorithm suitable for grid generation. In: Lucian, M., et al. (eds.) Geometric Modeling and Computing: Seattle 2003. Nashboro Press, Brentwood (2004)
12. Carey, R., Bell, G.: The Annotated VRML 2.0 Reference Manual. Addison-Wesley, Boston (1997)
13. Brutzman, D., Daly, L.: Extensible 3D Graphics for WEB Authors. Morgan Kaufmann (Elsevier), San Francisco (2007)
14. VRML. The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997 (1997)