2011 12th International Conference on Computer-Aided Design and Computer Graphics

CurveNetSurf: Creating Surfaces from Curve Networks

Leilei Gao Lifeng Zhu Guoping Wang
Graphics and Interactive Technology Lab, Peking University

Abstract

In conceptual design of models, designers usually express their ideas in curve networks, without generating final 3D surfaces. In this paper, we propose a method to utilize the structures and properties of curve networks to create mesh surfaces with controllable sharp features. We demonstrated our method on various curve networks, the created surfaces are both visually plausible and faithfully follow the input curve networks.

1. Introduction

3D computer modeling is becoming a common tool for designers and artists, and even amateurs. However, professional modeling packages, such as [1][2], are not intuitive and usually require hours of tedious works to create a simple surface. Artists and designers mainly draw characteristic curves to express their ideas. Tools that can directly create surfaces from such curves are more preferred, such as FiberMesh [3]. Different from those existing methods, which only use those curves as local positional constraints, we utilize the structures and differential properties of curve networks to create surfaces, which smoothly interpolate differential properties of curve networks.

Our key observation is that curve networks provide most desired features of surfaces, the areas in between them are often smooth blending of these curves. Therefore, positions and properties of these curves, such as normals, must be respected and used to guide the creation of surface patches in between them. Guided by this observation, we propose a method for creating surfaces from curve networks. First, we analyze the structure and properties of a curve network and decompose it into a set of disconnected sub-structures. Then, a coarse patch is created for each sub-structure. At last, we construct the final surface from a harmonic field defined by the analyzed or supplied differential properties of the curve network.

In the remainder of this paper, we first discuss related works and introduce relationships between curve networks and respective surfaces. Then, an algorithm is presented to create surfaces from curve networks. We evaluate the results in Section 5.

2. Related Work

Sketch-based 3D modeling. A comprehensive survey on sketch-based 3D modeling can be seen in [4][5]. Here we only focus on methods most related to our work. Popular sketch-based modeling systems [3][6][7] can only create few bulbous 3D shapes, since inflation methods are used. But our method can create surfaces from curve networks for complex shapes. FreeDrawer [8] approximate a network of spline curves by a b-spline surface, using a complex fitting process.

Our method is also relevant to patching parametric surfaces. In parametric surface modeling, surfaces are mostly created from triangular or quadrilateral domains. Additional requirements, such as the continuity constraints across the patch boundaries, complicate the construction [9]. However, our work interpolates and blends the surface patches in a discrete way. The shape of the network is not restricted to be triangular or quadrilateral, and complicated constraint management is avoided.

Surface Reconstruction. To recover a surface from a curve network is similar to reconstruction from lower dimensional data sets, because a 2D manifold has to be constructed from a lower dimensional data set, i.e., a set of curves. Recently, surface reconstruction from point clouds are intensively studied for reverse engineering or data acquisition [10][11][12]. Better reconstruction results can be obtained when normals of points are additionally supplied [13][14], or various prior knowledge is provided [15][16]. Besides point clouds, a set of projected contours can be obtained from images [17]. A popular method to reconstruct these surfaces is to find an implicit representation to the data set, and then convert it into surface meshes using standard techniques, e.g. marching cubes [18]. However, it is difficult to get an implicit representation for curve networks, because the curve network structure is sparser than point clouds or voxels from the images. A similar work presented in [19]. [20] reconstructs surfaces from cross-sections. However, for

978-0-7695-4497-7/11 $26.00 © 2011 IEEE
DOI 10.1109/CAD/Graphics.2011.29

most cases, it is hard to extract cross-sections from the curve network abstraction.

Poisson Editing. Poisson editing is an efficient algorithm for performing geometric processing in the gradient domain. After it is introduced by Yu et al [21], it has been successfully used in wide varieties of geometric applications, include shape morphing[22], deformation transfer[23], paramerization[24], hole filling[25] and recently, remeshing for equivalent class[26]. In this paper, we adopt Poisson editing for the shape design from curve networks.

Curved Triangle. Curved triangles [27][28] are a set of local operations, which refine the triangles guided by the vertex normals. Cubic Bezier patches [27] or quadratic patches [28][29] are used to interpolate the vertex positions and its normals. They can be regarded as techniques creating surfaces from triangular networks. While curved triangles focus on improving the visual smoothness for rendering meshes, our work focuses on shape modeling. Moreover, our work is a generalization of curved triangles. Polygonal networks, rather than triangular networks, are interpolated to create desirable shapes. Compared to triangular networks, polygonal networks provide more possibilities to create complex shapes with only a few curves.

3. Curve networks and respective surfaces

We focus on curve networks used in concept design, which outline 3D shapes through a set of feature curves. In such curve networks, curves are served as potential features, and the outlined shape can be regarded as a smooth blending of these curves. Considering curve networks for different shapes may be very different, we make a constraint about input curve networks that they should be able to be decomposed into a set of curve loops. For few shapes, some curves need to be added to conform to this constraint. This assumption is minor, because good structural curves always contain a lot of curve loops.

Based on the assumption, we propose a method to decompose a curve network into a set of curve loops. In-between of these curve loops forms a set of surface patches, and its assembly approximates the shape characterized by the curve network. Thus, the problem is reduced to how to design a surface patch from its boundary loop and how to smoothly blend them.

Considering various 3d curve networks either created by artists [30] or extracted from shape analysis algorithms [31], we find that a good curve network abstraction locates the curves around the surface features where the normals are either discontinuous or changing rapidly, and normals of the surface patches bounded by curves do not change vastly, as shown in Figure 1. Noises in the scanned vase cause few red points in dark blue regions, which rarely exist in conceptual design models. Because the surface patches have various shapes, the vertex normals are not always linearly or quadratic distributed on the surface. We make a more general assumption that the vertex normals' distribution is harmonic. Note that, if the distribution of the normals is harmonic, the variance of the normals is minimized locally, which can be guaranteed by a good curve network abstraction.

Figure 1: The color map of normal variations.
(Values near zero are in dark blue).

4. Algorithm

From the discussion in section 3, we can formulate our algorithm to create surfaces from input curve networks now.

4.1. Curve Network Decomposition & Analysis

To deal with various representations for curve networks, a uniform re-sampling is performed to convert an input curve network into a connected vertex graph G, as shown in Figure 2. Arc parameterization and chord parameterization is used for parametric curves and discrete curves respectively.

In a graph G, each vertex connected to more than two vertexes is identified as a cross point. To identify loops in G, each edge is split into two opposite half edges. All half edges form a set E'. Based on our assumption about G, it is easy to find that each half edge must exist in only one loop, and all half edges in all loops form E'.

We find all curve loops in G as follows. When there is still a half edge in E', we pick it out, and follow it to find a loop in a counter clock wise way. Encountering

a cross vertex, we select the one with smallest angle of the former edge. When a loop is identified, all its half edges are deleted from E'. When E' becomes an empty set, all loops are identified. Note that extra boundary loops are also identified, if a curve network is for a non-closed shape. They can be removed interactively.

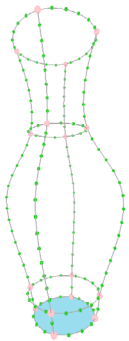

Figure 2: A curve network. Cross points (pink), ordinary points (green) and a bottom loop (blue).

If the vertex normals on the curve network are not provided, we estimate the vertex normals for each loop respectively. Each curve of a loop is labeled as a smooth feature by default, unless it is labeled as a sharp feature by designers explicitly. The normals of its two neighboring cross vertices of a curve are calculated first. For a smooth curve, they are defined as angle weighted average of their normals in all loops. For a sharp curve, the normal of each neighboring cross point v is defined as cross product of the vectors pointing from v to its neighboring vertices in the loop. Then, normals of non-cross vertices on the curve are linearly interpolated from its neighboring cross vertices.

4.2. Initial Surface Generation

We adopt the advancing front mesh method [32] to create a surface patch for each loop. All initial surface patches form an initial surface altogether.

4.3. Harmonic field guided surface refinement

We compute the harmonically-distributed normals of the initial surface to refine it. Our surface refinement can be thought as an ingredient to [21], a basic mesh editing framework.

General Differential setting. In the harmonic field construction and surface refinement, we considering the steady-state elliptic equation

$$\nabla^2 u = f \qquad (1)$$

with appropriate boundary conditions. If $f \equiv 0$, it is the Laplace equation. Otherwise, it is the Poisson equation. The discrete format is well-known and can be solved by a sparse linear system

$$A\mathbf{x} = \mathbf{b}, \qquad (2)$$

where the coefficient matrix A represents the discrete Laplace-Beltrami operator matrix [33][34].

Harmonic normal distribution. Given the normals on a loop, we interpolate them in a smooth way over the loop surface patch by using harmonic fields. A harmonic function f satisfies the Laplace equation $\nabla^2 f = 0$. Three harmonic fields are constructed, each for one dimension of the normals. We then solve the Laplace equation $Af = 0$ for each harmonic field f, with the boundary conditions defined by the vertex normals of the loop. Each normal is normalized after its three coordinates are obtained.

Surface refinement. Based on the results of [35][36], the discrete Poisson equation on a triangular meshes is formulated as follows. A discrete potential scalar field is defined as a piecewise linear function

$$f(v) = \sum_i f_i \Phi_i(v),$$

with $\Phi_i(v)$ being the piecewise linear basis function valued 1 at vertex v_i and 0 at all other vertices, and f_i being the value of f at v_i. Its discrete gradient is

$$\nabla f(v) = \sum_i f_i \nabla \Phi_i(v), \qquad (3)$$

which is a piecewise constant vector field, e.g., inside each triangle the gradient is constant vector. For such a piecewise constant vector field $\mathbf{w}$, the discrete divergence at vertex v_i is defined as

$$(\mathrm{Div}\,\mathbf{w})(v_i) = \sum_{T_k \in N_T(v_i)} \nabla \Phi_{ik} \cdot \mathbf{w}_k |T_k|, \qquad (4)$$

where $N_T(v_i)$ is the set of all triangles immediately adjacent to the vertex i, $\nabla \Phi_{ik}$ is the gradient of basis function at vertex i in triangle T_k, $\mathbf{w}_k$ is the constant vector in T_k, $|T_k|$ is the area of triangle. Then, the discrete Laplacian operator is formulated as

$$\Delta f = \mathrm{Div}(\nabla f).$$

The well-known cotangent weights Laplacian operator for triangular surface mesh, e.g.,

$$\Delta f(v_i) = \sum_{T \in N_T(v_i)} \frac{1}{2}\left(\cot\alpha_j + \cot\beta_j\right)(f_i - f_j), \qquad (5)$$

where α_j and β_j are the two angles opposite to the edge (v_i, v_j) [37]. f_i is one dimensional of the coordinates of the vertex indexed i on our initial surface. Finally, the discrete Poisson equation is expressed as

$$\Delta f \equiv \mathrm{div}(\nabla f) = \mathrm{div}\,\mathbf{w}. \qquad (6)$$

To utilize the harmonic normals calculated above, each triangle is rotated around its center separately, to be consistent with its calculated normal. Then, a

fragmented, discontinuous mesh is yielded, which provide a guidance vector field for Poisson equation.

With the guidance vector field, the initial surface can be refined based on the Poisson equation. Use Eq. (3) to calculate gradients of each vertex on its adjacent triangles. Then the divergence of each vertex on the input curve network is calculated, according to Eq. (4). Matrix A can be computed using Eq. (5) and **b** is initialized by divergences of all points on the input curve network. Finally, we solve the Poisson equation to obtain the new vertex coordinates, which defines the geometry of the optimized surface.

Some curves on the curve network represent sharp features on the surface. We allow users to adjust the vertex normals on a curve network where the vertices or curves are required to be a corner or sharp edge. By assigning desirable normals for a vertex in different loops, a sharp feature can be created.

5. Results & Discussion

We test on curve networks of various models. Figure 3 gives an illustration about the whole process.

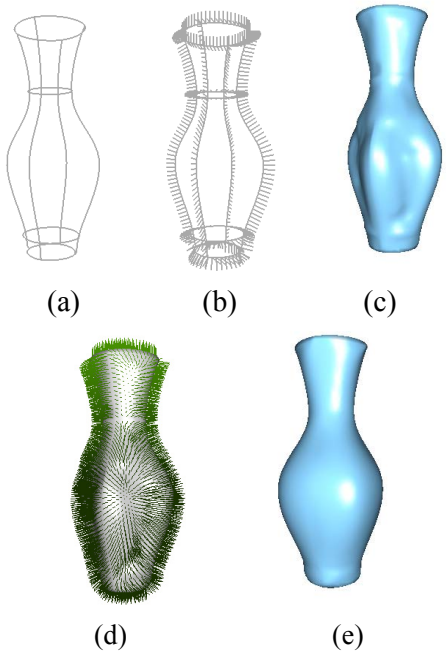

Figure 3: CurveNetSurf. An input curve network (a), normals on the curve network(b), the initial surface (c), harmonic distributed normals (d), the final surface (e).

To test whether the refined surface obeyed the normal harmonic fields, we measure the angle variations between the face normals of the refined surface and their respective normals defined by the harmonic fileds, as shown in Figure 4. Besides some variations near the top and bottom sharp feature curves, almost no variations exist, the refined surface smoothly blends the curve network within each loop and over the smooth curves.

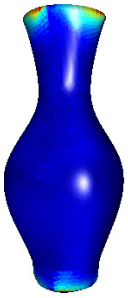

Figure 4: Normal variation color maps.

In Figure 5, we show a comparison with Fiber Mesh. Because Fiber Mesh uses a global optimization to create surfaces, sharp curves can't guarantee to produce sharp features. However, sharp features of surfaces can be easily created using our method, as shown in Figure 5d. More examples are shown in Figure 7.

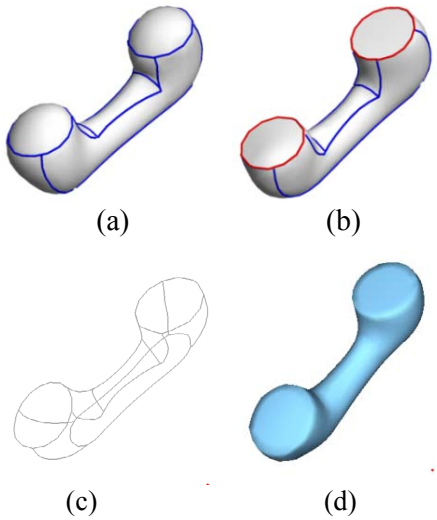

Figure 5: A comparison to Fiber Mesh. A telephone listener model created by FiberMesh, with the blue curves as smooth constraints (a), two top curves are changed into sharp constraints (b). A curve network (c) is manually extracted from (a), and used by CurveNetSurf to create a surface (d).

The proposed method has been implemented with VC++9.0, using the LAPACK library [38] to do numerical computing. All experimental results in this paper were obtained on a 2.66GHz Intel Core2 Duo CPU with 4.0GB memory. The complexity and creation time is summarized in Table 1. For a local

editing, those equations can be solved in less than one second for the curve networks we used.

Table1. Computing time

Model	triangles	vertices	Laplace Eq.	Poisson Eq.
Chair	732	1460	0.0176s	0.1493s
Vase	4690	2347	0.0358s	0.2694s
Phone	2090	4176	1.4129s	1.3833s
Dolphin	6886	13768	42.2257s	40.154s
Bird	6706	13408	41.0531s	39.221s
Donut	2557	5114	1.4736s	1.4234s

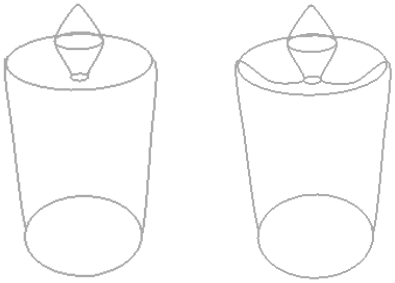

Figure 6: Curve networks of a handle.
(left: non-valid, right: valid)

Limitations. This work is just a first step, and it has some limitations. For example, we assume each input curve network is manifold and made up of a set of loops. For few shapes, some curves need to be added to conform to this constraint, as shown in Figure 6. However, these added are actually needed, since they characterized the slope of the blending surface between the two loops. Another issue relates to the normal interpolation on a curve. Although the linear interpolation is enough for most examples we used, for a long arc curve with many normal variations, it may fail. We resolve this problem by labeling some non-cross vertices on such a curve as cross points to cut the curve into segments with less variation.

6. Conclusion

In this paper, we introduced a simple and efficient algorithm for creating surfaces from curve networks. The algorithm produces plausible surfaces that satisfy the characterizations of curve networks. In addition, the final surface can be interactively modified in a simple way by few adjustments, to obtain surfaces with different sharp features. In terms of future work, we would like to investigate curve networks in a more general structure.

Acknowledgement

This work is supported by the National Basic Research Program of China (GrantNo. 2010CB328002) and the National Natural Science Foundation of China (Grant No. 90915010, 60925007, 60833007).

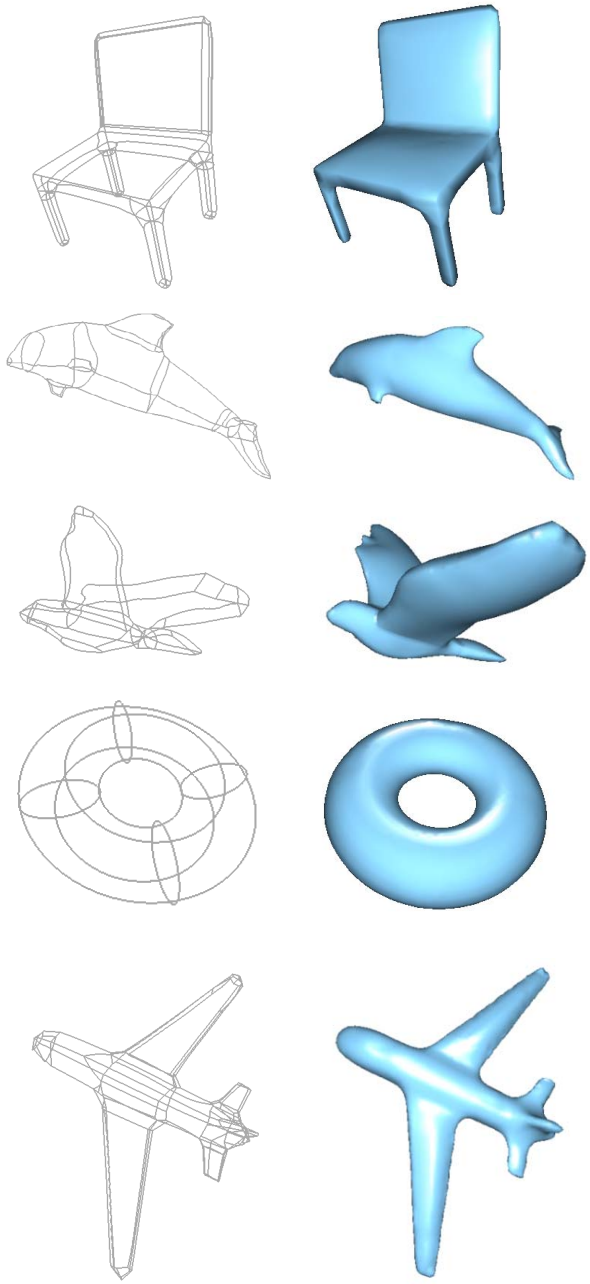

Figure 7: Surfaces created using CurveNetSurf.
(left: curve networks, right: created surfaces)

References

[1] 3DS MAX, 2010, Autodesk, http://usa.autodesk.com/3dsmax.

[2] MAYA, 2010, Autodesk, http://usa.autodesk.com/maya.

[3] Andrew Nealen, Takeo Igarashi, Olga Sorkine and Marc Alexa, "FiberMesh: Designing Freeform Surfaces with 3D Curves", ACM Transactions on Computer Graphics, ACM SIGGRAPH 2007, San Diego, USA, 2007.

[4] Matthew T. Cook, and Arvin Agah, "A survey of sketch-based 3-D modeling techniques", Interacting with Computers, Volume 21, Issue 3, July 2009, Pages 201-211.

[5] Olsen, L., Samavati, F.F., Sousa, M.C., and Jorge, J.A., "Sketch-based modeling: A survey", Computer & Graphics, Volume 33, Issue 1, Feb 2009, Pages 85-103.

[6] Igarashi, Takeo, Matsuoka, Satoshi, Tanaka, Hidehiko, "Teddy: a sketching interface for 3D freeform design", In: ACM SIGGRAPH 1999, ACM, Los Angels, California.

[7] Schmidt, R., Wyvill, B., Sousa, M., Jorge, J., "ShapeShop: Sketchbased solid modeling with blobtrees", In Eurographics Workshop on Sketch-Based Interfaces and Modeling, 2005, pp. 53–62.

[8] Wesche, Gerold, Seidel, Hans-Peter, "FreeDrawer: a free-form sketching system on the responsive workbench", In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM Press, 2001, pp. 167–174.

[9] G.Farin, J.Hoschek and M.-S. Kim, Handbook of computer aided geometric design, Elisevier Science B.V. 2002.

[10] Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., Mccallum B. C., Evans T. R, "Reconstruction and representation of 3D objects with radial basis functions", In Siggraph (2001), pp. 67–76.

[11] Hornung A., Kobbelt L., "Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information", In Eurographics Symposium on Geometry Processing (2006), pp. 41–50.

[12] Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or, "Interactive topology-aware surface reconstruction", ACM Transactions on Graphics (TOG), Volume.26 No.3, July 2007.

[13] Michael Kazhdan , Matthew Bolitho , Hugues Hoppe, "Poisson surface reconstruction", Proceedings of the fourth Eurographics symposium on Geometry processing, Cagliari, Sardinia, Italy, June 26-28, 2006.

[14] J. Schreiner, C. Scheidegger, S. Fleishman, and C. Silva, "Direct (Re)Meshing for Efficient Surface Processing", Computer Graphics Forum (Proceedings of Eurographics 2006), Volume 25, Issue 3, 2006, pp.527—536.

[15] Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, Baoquan Chen, "SmartBoxes for Interactive Urban Reconstruction", ACM Trans. Graph., Volume 29, Issue 4, July 2010, pp. 93:1--93:10.

[16] Ran Gal and Ariel Shamir and Tal Hassner and Mark Pauly and Daniel Cohen-Or, D., "Surface reconstruction using local shape priors", In Proc. of Eurographics Symp. on Geometry Processing, 2007, pp. 253–262.

[17] Seitz S. M., Curless B., Diebel J.,Szeliski D. S. R., A comparison and evaluation of multi-view stereo reconstruction algorithms, CVPR 2006, Vol. 1, pp. 519-526.

[18] Lorensen W, Cline H., "Marching cubes: a high resolution 3D surface construction algorithm", Computer Graphics, 21(4), 1987, pp. 163–9.

[19] T. Ju, J. Warren, J. Carson, G. Eichele, C. Thaller, W. Chiu, M. Bello and I. Kakadiaris, "Building 3d surface networks from 2d curve networks with application to anatomical modeling", Proceedings of Pacific Graphics 2005, 21(8-10), pp.764-773.

[20] Liu, L., Bajaj, C., Deasy, J. O., Low, D. A., and Ju, T., Surface Reconstruction From Non-parallel Curve Networks, Computer Graphics Forum, Blackwell Publishing Ltd, Vol. 27, 2008, pp. 155-163.

[21] Yu, Y., Zhou, K., Shi, X., Bao, H. Guo, B., Shum, H., "Mesh editing with Poisson-based gradient field manipulation", In: processing of SIGGRAPH, Los Angeles, California, USA, 8–12 August 2004, pp. 644–651.

[22] Xu, D., Zhang, H., Wang, Q., and Bao, H., "Poisson shape interpolation", In SPM '05: Proc. ACM Symp. on Solid and Physical Modeling, 2005, pp. 267–274.

[23] Robert W. Sumner and Jovan Popovic, "Deformation transfer for triangle meshes", ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH), 23(3), 2004, pp. 399–405.

[24] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, Steven J. Gortler, "A Local/Global Approach to Mesh Parameterization", Computer Graphics Forum (Proc. Eurographics Symposium on Geometry Processing (SGP)), 27(5), 2008, pp. 1495-1504,

[25] Zhao, Wei and Gao, Shuming and Lin, Hongwei, "A robust hole-filling algorithm for triangular mesh", Visual Computer, Volume 23, Issue 12, 2007, pp. 987—997.

[26] Mayank Singh and Scott Schaefer, “Triangle surfaces with Discrete Equivalence Classes”, ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH), 29(4) 2010.

[27] Vlachos, A., Peters, J., Body, C., and Mitchell, J., Curved PN triangles. Proceedings of ACM Symposium on Interactive 3D, 2001, pp.159–166.

[28] Boubekeur, T., and Alexa, M., “Phong tessellation, ACM Trans. Graph. 27, 5, 2008, pp.1–5.

[29] Boubekeur, T., and Schilick, C., “QAS: Real-time quadratic approximation of subdivision surfaces”, In Proceedings of Pacific Graphics 2007, pp. 453–456.

[30] Schmidt, Ryan and Khan, Azam and Singh, Karan and Kurtenbach, Gord, “Analytic drawing of 3D scaffolds”, ACM Trans. Graph, Volume 28, Issue 5, December 2009, pp. 112-121.

[31] Fernando de Goes, Siome Goldenstein, Mathieu Desbrun, Luiz Velho, “Exoskeleton: Curve network abstraction for 3D shapes”, Computers & Graphics, Volume 35, Issue 1, pp. 112-121.

[32] George, L.P., Seveno, E., “The advancing-front mesh generation method revisited”, Int. J. Numer. Methods Eng. 37(7), 1994, pp.3605–3619.

[33] Pinkall, U., Polthier, K., “Computing discrete minimal surfaces and their conjugates”, Comput. Aided Des, Volume 25, Issue 4, 1993, pp.225–232.

[34] Meryer, M., Desbrun, M., Schr -Oder, P., and Barr, A., “Discrete differentialgeometry operators for triangulated 2-manifolds”, In Proc. Vis-Math, 2002, pp. 35–57.

[35] K. Polthier and E. Preuss, “Identifying Vector Fields Singularities Using a Discrete Hodge Decomposition,” Visualization and Math., Springer Verlag, 2002.

[36] Tong, Y., Lombeyda, S., Hirani, A. N., and Desbrun, “Discrete multiscale vector field decomposition”, ACM Trans. Graph., Volume 21, Issue 3, July 2003, pp.445-452.

[37] Hirani, Anil N., “Discrete exterior calculus”, Dissertation (Ph.D.), California Institute of Technology, 2003.

[38] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, Jack J. Dongarra, J. Du Croz, S. Ham-marling, A. Greenbaum, A. McKenney, and D. Sorensen, “LAPACK Users' guide (third ed.)”, Societyfor Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.