Available online at www.sciencedirect.com
ScienceDirect
Computer-Aided Design 38 (2006) 1134–1144
COMPUTER-AIDED DESIGN
www.elsevier.com/locate/cad

Robust uniform triangulation algorithm for computer aided design

Hovhannes Sadoyan[a], Armen Zakarian[b,*], Vahram Avagyan[b], Pravansu Mohanty[c]

[a] *Autodesk Inc., Novi, MI 48375, United States*
[b] *Department of Industrial and Manufacturing Systems Engineering, University of Michigan – Dearborn, Dearborn, MI 48128, United States*
[c] *Department of Mechanical Engineering, University of Michigan – Dearborn, Dearborn, MI 48128, United States*

Received 16 June 2005; accepted 29 June 2006

Abstract

This paper presents a new robust uniform triangulation algorithm that can be used in CAD/CAM systems to generate and visualize geometry of 3D models. Typically, in CAD/CAM systems 3D geometry consists of 3D surfaces presented by the parametric equations (e.g. surface of revolution, NURBS surfaces) which are defined on a two dimensional domain. Conventional triangulation algorithms (e.g. ear clipping, Voronoi-Delaunay triangulation) do not provide desired quality and high level of accuracy (challenging tasks) for 3D geometry. The approach developed in this paper combines lattice tessellation and conventional triangulation techniques and allows CAD/CAM systems to obtain the required surface quality and accuracy. The algorithm uses a Cartesian lattice to divide the parametric domain into adjacent rectangular cells. These cells are used to generate polygons that are further triangulated to obtain accurate surface representation. The algorithm allows users to control the triangle distribution intensity by adjusting the lattice density. Once triangulated, the 3D model can be used not only for rendering but also in various manufacturing and design applications. The approach presented in this paper can be used to triangulate any parametric surface given in $S(u, v)$ form, e.g. NURBS surfaces, surfaces of revolution, and produces good quality triangulation which can be used in CAD/CAM and computer graphics applications.

Keywords: Triangulation; CAD/CAM; Trimming; 3D geometry; Tessellation

1. Introduction

This paper introduces a robust generalized triangulation technique for the uniform triangulation of the domain that can be used in CAD/CAM systems for 3D geometry visualization. In most CAD/CAM systems 3D geometry is presented by the parametric equations that use additional trimming curves to define 3D surface domains. Modern 3D visualization hardware (e.g., ATI and NVIDIA graphics cards with OpenGL and DirectX support) are not capable of visualizing arbitrary trimmed parametric surfaces and coping with the miscellaneous trimming curves, curve intersections, overlappings and other exceptional issues. In CAD/CAM systems, to visualize a 3D geometry presented by the parametric surfaces, triangulation algorithms are used to convert the trimmed parametric surface to a set of triangles that represent given surfaces. Triangles of the triangulated surfaces are further transferred by the 3D design software to the graphics hardware for visualization on a computer display.

The main problem in any parametric visualization process is the generation of triangles that represent a given trimmed parametric surface in a smooth and accurate manner. The secondary problem is the performance of the triangulation algorithm. Surface triangulation should be fast enough to be used in real-time complex 3D surface modelling applications. Complex 3D models may consist of thousands of trimmed surfaces and for these models triangulation should be applied once, during the loading process, rather than for each frame of rendering (e.g. 30 times/s).

Literature describes various polygon triangulation algorithms with their own advantages and disadvantages. Some of the algorithms can be used to triangulate simple polygons [1–6], while several others are used to triangulate polygons with holes [7–9]. Also, there are several fast view dependent algorithms that achieve speed by storing and reusing the information

* Corresponding address: University of Michigan Deraborn, Department of Industrial and Manufacturing Systems Engineering, 4901 Evergreen Rd, 48128 Dearborn, MI, United States. Tel.: +1 313 593 5244; fax: +1 313 593 3692.
E-mail address: zakarian@umich.edu (A. Zakarian).

0010-4485/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2006.06.008

from frame to frame [3,7,10,11] and algorithms that first tessellate the trimming domain before they triangulate and map it on the surface [6,9,12–15]. The algorithm presented in this paper follows the latter approach. It uses a Cartesian lattice to divide the parametric domain into adjacent rectangular cells. These cells are used to generate polygons that are further triangulated to obtain accurate surface representation. The algorithm allows users to control the triangle distribution intensity for various surfaces of a 3D model by adjusting the lattice density or level of detail (LOD). Once triangulated, the 3D model can be used not only for rendering but also in various manufacturing and design applications. The approach presented in this paper can be used to triangulate any parametric surface given in $S(u, v)$ form, e.g., NURBS surfaces, surfaces of revolution, and produces good quality triangulation which can be used in CAD/CAM and computer graphics applications.

2. Related work

Kumar and Manocha [7,12] present a view dependent algorithm for interactive display of trimmed NURBS surfaces. The algorithm converts the NURBS surfaces/curves into Bezier surfaces/curves and optimizes the triangulation process. It uses tight bounds to tessellate Bezier surfaces uniformly into cells, partition the cells into trimmed and untrimmed regions, and triangulate the trimming regions. Each cell is triangulated by tracing the trimming curves and computing the trimming regions of each cell. The algorithm uses back face culling to optimize the number of triangles generated, and uses coherence between successive frames to perform incremental computation at each frame. Similar to the approach presented in [7], the triangulation algorithm developed in this paper partitions the domain into adjacent rectangular (i.e. uniform) cells and triangulates each cell individually. In contrast to the algorithm developed in [7], the approach developed in this paper is not view dependent and can be applied to any parametric surface represented in $S(u, v)$ form, including NURBS surfaces and surfaces of revolution. Furthermore, triangles generated by our triangulation algorithm are located in a single cell, while the triangles generated by Kumar and Manocha [7] approach may intersect (overlap) with multiple cells.

Kumar [13] presents another view dependent triangulation algorithm based on Bezier surfaces and trapezoidation. The algorithm eliminates or replaces long and thin triangles with well distributed ones. The triangulation process is optimized by using information from one frame to the next, which requires real-time calculations for each frame rendering.

Bern and Eppstein [1] and Bern et al. [8] present polynomial-time triangulation algorithms that allows one to triangulate n-sided polygons with $O(n^2)$ non-obtuse triangles (i.e. triangles with small angles) and an algorithm that triangulates polygons with holes. Although the algorithms generate triangles with small angles, the disadvantage of this approach is that it cannot be used to triangulate trimming surfaces and cannot generate uniformly distributed triangles for surfaces. However, the method can be used in the proposed approach as an auxiliary triangulation procedure for generating polygons that do not affect uniform triangle distribution. To triangulate a polygon, the algorithm developed in [8] first populates the domain with non-overlapping circles, then divides the domain into small polygons by adding edges between the centres of the circles and points of tangency on their boundaries. It triangulates small polygons using Steiner points. Eppstein [16] introduces an improved technique for the circle population that increases the efficiency of the algorithm described in [8].

Cho [3] presents an unstructured triangulation algorithm that approximates a set of mutually non-intersecting simple trimmed NURBS surface within a user specified geometric tolerance. The algorithm is based on an unstructured Delaunay mesh approach that leads to an efficient adaptive triangulation. It constructs a 2D triangulation domain, which sufficiently preserves the shape of triangles when mapped into a 3D space.

Rockwood [11] presents a modular approach to render trimmed NURBS surfaces in real time by a uniform view-dependent tessellation per patch. Discretization anomalies are minimized regardless of the view motion to obtain high quality 3D geometry. This concept is used in [9] to triangulate trimmed surfaces. The approach converts all surfaces into Bezier patches bounded by the trimming curves that require additional calculations in the triangulation workflow.

Piegl [6] presents an algorithm for the piecewise triangular approximation of a trimmed NURBS surface. The approach presented in [6] is similar to the one developed in this paper. The algorithm generates deviation-based subdividing rectangles and triangulates them without requiring NURBS surfaces to be differentiable. The difference between this approach and the one presented in this paper is that Piegl [6] generates rectangular regions of the domain based on the derivation properties of the NURBS surface while the approach presented in this paper creates an adjacent rectangular (uniform) lattice based on the curvature of the parametric surface (NURBS, Bezier, or surface of revolution). Furthermore, Piegl [6] uses a NURBS surface approximation (4-point cubic) to expedite the NURBS computation, while our approach emphasizes efficient domain triangulation and maintains the original surface without distorting it. The latter is important since in many applications surface distortion may cause undesired artifacts.

Abi-Ezzi [9] presents a technique that uses a graphical compilation to enable fast dynamic tessellation of trimmed NURBS surfaces under highly varying transforms. The approach uses a concept of graphical data compilation to absorb the main complexities encountered in trimming and processes NURBS surfaces into compact, view-independent forms for fast per-frame triangle extraction. Complex trimming curves are divided into simple trapezoidal v-regions that are specially designed to facilitate tessellation before they are rendered. It also detects and resolves all cases of degeneracies in the original surface primitive.

Eberly [4] introduces a simple ear clipping triangulation algorithm with $O(n^2)$ asymptotic upper bound based on the theory of ears for polygons [17]. Ear clipping is used