Computer-Aided Design 43 (2011) 573–585

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Localized construction of curved surfaces from polygon meshes: A simple and practical approach on GPU

Yuen-Shan Leung, Charlie C.L. Wang [*], Yunbo Zhang

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, China

ARTICLE INFO

Article history:
Received 22 May 2010
Accepted 17 January 2011

Keywords:
Curved surface
Localized construction
Hardware acceleration
G^1 continuity
Sharp feature

ABSTRACT

We present a method for refining n-sided polygons on a given piecewise linear model by using local computation, where the curved polygons generated by our method interpolate the positions and normals of vertices on the input model. Firstly, we construct a Bézier curve for each silhouette edge. Secondly, we employ a new method to obtain C^1 continuous cross-tangent functions that are constructed on these silhouette curves. An important feature of our method is that the cross tangent functions are produced solely by their corresponding facet parameters. Gregory patches can therefore be locally constructed on every polygon while preserving G^1 continuity between neighboring patches. To provide a flexible shape control, several local schemes are provided to modify the cross-tangent functions so that the sharp features can be retained on the resultant models. Because of the localized construction, our method can be easily accelerated by graphics hardware and fully run on the *Graphics Processing Unit* (GPU).

1. Introduction

In many applications, polygonal meshes have become an important representation of computer generated objects for visual effects. They are simple and versatile, but the lack of continuity between neighboring polygons on the models would be a problem. To improve the visual quality, a common strategy is to subdivide the input mesh surface into a finer mesh, which however quickly increases the consumption of memory and transmission time (either through the network or from main memory to the graphics hardware). To overcome this difficulty, an ideal way is that we transmit the relatively coarse mesh during communication, and refine the coarse mesh only when it is about to be displayed. Specifically, we wish to send a coarse input mesh M^0 to the graphics hardware and a smooth dense mesh M^r is then produced in real time through interpolating M^0 by using the parallel computational power of *Graphics Processing Unit* (GPU). The input mesh M^0 is composed of n-sided polygons (with $n \geq 3$). To accomplish this purpose, the proposed method should satisfy the following requirements.

- The resultant surface M^r interpolates the vertices on M^0 as well as their normal vectors.
- M^r possesses G^1 continuity.
- The construction procedure must be computed locally on each facet of M^0 so that it can employ the strength of GPUs extensively.

Moreover, unlike the GPU-based subdivision schemes [1–3] and the visualization based normal processing [4], we intend to form a continuous parametric patch representation on M^r, which is able to evaluate surfaces at arbitrary parametric points — this is important for those non-visualization applications (e.g., distance query and evaluation).

In this paper we offer a local construction approach to create Gregory patches on every n-sided polygon of a coarse mesh M^0. These n-sided Gregory patches require vertices and normal vectors on M^0 and maintain G^1 with their adjoining patches. To build the surface, Bézier curves are first substituted for every silhouette edge. Then, we customize C^1 continuous cross-tangent functions that exclusively rely on boundary curves and are independent of the vicinity facets. Therefore, a Gregory patch can be constructed. Because of this localized scheme, our approach can be processed and accelerated on graphic hardware. Fig. 1 shows an example of curved polygons constructed by our approach. To extend the basic construction method, we introduce several schemes to produce sharp features (e.g., the examples shown in Fig. 15). The outcome mesh surface M^r with sharp features can be displayed by tessellating the reconstructed Gregory patches on the GPU.

1.1. Related work

The work presented in this paper relates to the existing research in several aspects, including localized parametric patch construction approaches, GPU-based subdivision surface evaluation methods, surface tessellation schemes on the GPU, and n-sided patch researches. They are reviewed below.

[*] Corresponding author. Tel.: +852 26098052; fax: +852 26036002.
E-mail address: cwang@mae.cuhk.edu.hk (C.C.L. Wang).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.01.010

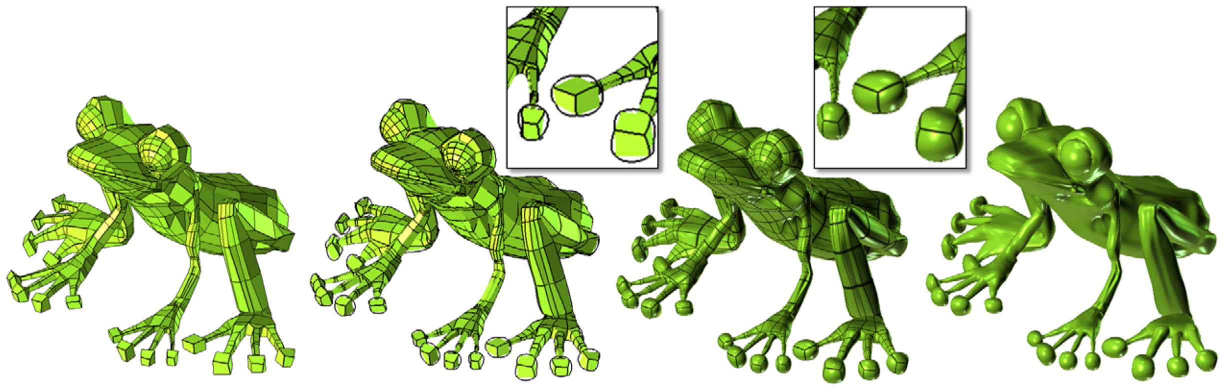

Fig. 1. An example of our localized construction of curved polygons. The first image (far left) illustrates an input control mesh. The second image (middle left) shows the silhouette curves generated on the edges of the input control mesh. The boundaries of Gregory patches are displayed by grids in the third image (middle right) and the final (far right) image — the surface of curved polygons generated from the input control mesh.

Vlachos et al. presented a method in [5] to construct curved point-normal (PN) triangles based only on the three vertices and three vertex normals of given flat triangles. The main principle of their approach is based on substituting the geometry of a three-sided cubic Bézier patch for the triangles' flat geometry, and a quadratically varied normal for Gouraud shading. Visually smooth models can be generated; however, these methods do not preserve G^1 continuity across the boundary of neighboring PN triangles. Similar to the PN-triangle approach [5], some work has been done to improve the appearance of a smooth surface generated from triangles [4,6]. Again, neither of these approaches provides the shape of smooth surfaces preserving G^1 continuity across the boundaries. Another limitation is that only triangular meshes are supported. The approach of Volino and Magnenat-Thalmann [7] can generate the substitute smooth geometry for n-sided polygons, but still cannot preserve tangent continuity across boundaries. The work presented in this paper also relates to the research of generating surface which interpolate curve networks (e.g., [8–10]). Nevertheless, in order to borrow the parallel computing power on GPUs, a localized construction approach needs to be exploited.

A subdivision surface provides an effective way to convert an input coarse mesh surface into a fine surface with high resolution. Although a subdivision surface can be composed of an infinite set of polynomial patches (especially essential for a complex model), more efforts need to be made to evaluate subdivision surfaces using GPU (Refs. [1–3]). Stam [11] presented a method to exactly evaluate Catmull–Clark surfaces using the framework provided by Halstead et al. [12], but this method can only operate on quadrilateral patches with at most one extraordinary vertex. To apply it to surfaces with triangles or patches with more than one extraordinary vertex, the surface must be subdivided up to two times to provide sufficiently separate extraordinary vertices. Due to the considerable resource requirements, many researchers have investigated methods for approximating subdivision surfaces. Recently, Loop et al. [13] developed a new method which replaces those irregular patches with a single rational patch that preserves G^1 continuity across the boundary to the surrounding patches. However, only quad/triangle patches are allowed to be constructed in their approaches. A more relevant approach by Christoph et al. [14] is to define a triangle patch with its vertex normals and the three edge neighbor triangles. Although they achieved G^1 continuity on GPU efficiently, the transmitted data still spend a lot of resources on the connectivity information and the surfaces are limited to triangle patches. On the contrary, we employ n-sided Gregory patch interpolation [15,16] to generate parametric surfaces that interpolate the positions and normal vectors on the vertices of M^0. For an n-sided polygon, only $2n$ vectors are required to perform an evaluation on a smooth surface — when having sharp edges, at most $4n$ vectors plus a 2-bit number are needed. This contributes to a significant improvement in the speed of data transmission.

Another relevant research line is the adaptive tessellation method on GPU [17–20]. Dyken et al. [17] presented an algorithm for detecting and extracting the silhouette edges of a triangle mesh in real time using GPU. The smooth silhouette is reconstructed through a continuous blend between Bézier patches with varying level of details. Their recent work in [20] introduced an adaptive tessellation scheme for surfaces consisting of parametric patches. However, different from ours (for n-sided polygons), their approaches only work on triangular mesh surfaces. The method of Boubekeur and Schlick [18] was only for mesh refinement on triangles too. The recent work of Schwarz and Stamminger [19] provides a new framework for performing on-the-fly crack-free adaptive tessellation of surface primitives completely on the GPU. Their implementation is based on CUDA of nVIDIA whereas the surface reconstructed in our approach is generated by the GL shading language, which can be supported by graphics hardware with Shader Model 4.0 in a variety of brands.

Regarding the study on n-sided surfaces, many researchers have put much effort in this area for many years. Piegl et al. [21] presented an algorithm to fill an n-sided region with G^k continuous NURBS patches, whereas Wang et al. [22] used Varady patches to fit an arbitrary n-sided region. Other approaches [23,24] described techniques to insert a region with multiple patches. The smoothness of the filled patches along the shared edges requires some special treatment. Basically, the computations involved in the methods of [21–24] used iterative algorithms or constrained optimization, which are too computationally expensive to fit in the pipeline of shader programs running on the graphics hardware. Our method aims at improving visual quality with as less transmitted data as possible. Therefore we provide an algorithm which is able to tackle the twist incompatibility and boundary incompatibility in a local manner and result in G^1 continuous surfaces. There are also some other works in literature that can generate G^1 interpolating polynomial surfaces (e.g., [25–27]). However, how to evaluate them locally to fit the framework of parallel computing on a GPU is still an open problem to be solved.

Prior to this work, we presented a Gregory patch based smooth force rendering method in [28]. The approach presented in this paper is extended from that algorithm by (1) providing a formal proof and analysis about the correctness of the basic local construction scheme, (2) investigating the extended schemes for the flexible shape control and (3) developing a GPU-based algorithm.