Computer-Aided Design 46 (2014) 252–257

Contents lists available at ScienceDirect

Computer-Aided Design

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

Technical note

Precise convex hull computation for freeform models using a hierarchical Gauss map and a Coons bounding volume hierarchy

Yong-Joon Kim [a], Myung-Soo Kim [b,*], Gershon Elber [a]

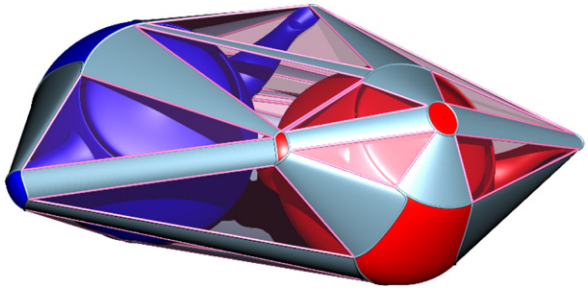
[a] Computer Science Department, Technion, Haifa 32000, Israel

[b] School of Computer Science and Engineering, Seoul National University, Seoul 151-744, Republic of Korea

HIGHLIGHTS

- Present an interactive-speed algorithm for computing the precise convex hull of freeform geometric models.
- Employing two pre-built data structures, a hierarchical Gauss map and a Coons bounding volume hierarchy, we develop an efficient culling technique that can eliminate the majority of redundant surface patches.
- Construct the precise convex hull boundary using numerical methods.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Convex hulls
Freeform surfaces
Support distance
Upper envelope
Stereographic projection
Tritangent
Developable scroll

ABSTRACT

We present an interactive-speed algorithm for computing the precise convex hull of freeform geometric models. The algorithm is based on two pre-built data structures: (i) a Gauss map organized in a hierarchy of normal pyramids and (ii) a Coons bounding volume hierarchy (CBVH) which effectively approximates freeform surfaces with a hierarchy of bilinear surfaces. For the axis direction of each normal pyramid, we sample a point on the convex hull boundary using the CBVH. The sampled points together with the hierarchy of normal pyramids serve as a hierarchical approximation of the convex hull, with which we can eliminate the majority of redundant surface patches. We compute the precise trimmed surface patches on the convex hull boundary using a numerical tracing technique and then stitch them together in a correct topology while filling the gaps with tritangent planes and bitangent developable scrolls. We demonstrate the effectiveness of our algorithm using experimental results.

1. Introduction

Convex hull computation has a long history of development and it plays an important role in the design of many efficient geometric algorithms [1]. Algorithms for convex objects are usually more efficient than for non-convex ones. Consequently, geometric tests on convex hulls are often applied beforehand to filter out simple cases where no further computation is needed for general non-convex objects. For example, the shape matching between two convex hulls is typically a necessary condition for the matching between two objects under comparison.

To the best of our knowledge, there is no previous interactive-speed algorithm for computing the convex hull of freeform geometric models. In this work, we present an efficient solution to this problem using two pre-built data structures: one for a hierarchy of bilinear surface approximations and the other for a hierarchy of unit normal bounds on the Gaussian sphere.

A simple way of characterizing the convex hull boundary is as a set of surface points which are extremal along each normal direction [2–4]. The support distance function, mapping each surface point to the inner product of the position vector and the unit normal vector, is an equivalent way of characterizing the extremality condition. Using the upper envelope of the support

* Corresponding author.
E-mail address: mskim@snu.ac.kr (M.-S. Kim).

0010-4485/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cad.2013.08.041

distance functions, Kim et al. [3] developed a real-time convex hull algorithm for planar freeform curves. We take a similar approach, but the main difference is in the hierarchical representations of the surface approximations and the Gaussian map.

The main contributions of this work can be summarized as follows:

- A simple technique is introduced that constructs an approximate convex hull boundary as color-encoded images of the upper envelope using graphics rendering hardware.
- An efficient interior culling algorithm is proposed that can eliminate the majority of redundant surface patches.
- The implemented algorithm can compute the convex hull of freeform geometric models in an interactive speed.

2. Related work

Early convex hull algorithms considered only discrete objects such as points, lines, polygons, or polyhedra [1]. Theoretical algorithms (with no implementation) have been designed for planar curves [5–7]. Practical algorithms with full implementation were developed in [2,8,4]. Elber et al. [2] presented a robust implementation for planar freeform curves. Seong et al. [4] extended the result to freeform space curves and surfaces. They reduced the convex hull computation to the problem of solving systems of polynomial equations. Nevertheless, it has been quite difficult to implement these algorithms for real-time performance.

Employing a G^1-biarc approximation of planar curves [9], Kim et al. [3] introduced a real-time convex hull algorithm for planar freeform curves. The efficiency improvement is based on a pre-built BVH of G^1-continuous arcs tightly fitting the given planar curves. Kim et al. [10] introduced the CBVH for NURBS surfaces and also demonstrated real-time performance in collision detection and minimum distance computation among many NURBS surfaces. Using the CBVH, Kim et al. [11] developed an interactive-speed algorithm for computing the Hausdorff distance between two NURBS surfaces. In the current work, we employ the CBVH for developing an interactive-speed convex hull algorithm for NURBS surfaces.

Sederberg et al. [12,13] introduced the concepts of normal cone/pyramid and surface bounding cone/pyramid for the loop detection in surface–surface intersection. In this work, we employ these geometric concepts for an efficient elimination of redundant surface patches in the convex hull computation.

3. Convex hull for freeform surfaces

We first consider a simple technique for approximating the convex hull using graphics rendering and then move to an efficient algorithm for computing the precise convex hull.

3.1. Approximate convex hull for freeform surfaces

Given a set of regular freeform surfaces $S_i(u, v)$, $0 \le u, v \le 1$, $i = 1, \ldots, K$, and their unit normal vectors $\mathbf{n}_i(u, v)$, we define the bivariate support distance functions using the inner product between the position vector and the unit normal vector:

$$f_i(u, v) = \langle S_i(u, v), \mathbf{n}_i(u, v) \rangle ,$$

for all $0 \le u, v \le 1$ and $i = 1, \ldots, K$. The upper envelope of these functions characterizes the convex hull boundary [3], which can be parameterized by the unit normals $\mathbf{n} \in S^2$:

$$\mathcal{U}(\mathbf{n}) = \max_i \{f_i(u, v) \mid \mathbf{n}_i(u, v) = \mathbf{n}\}.$$

This is a mapping of each unit normal $\mathbf{n}$ to the outermost surface point $S_i(u, v)$ that has the maximum support distance along the direction of $\mathbf{n}$ and thus contributes to the convex hull boundary.

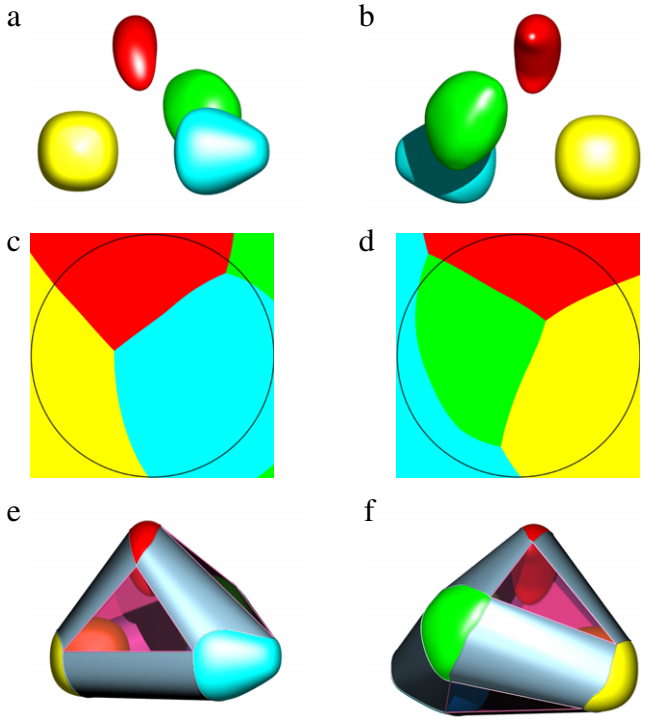


Fig. 1. Convex hull computation for freeform surfaces: (a)–(b) four freeform surfaces, (c)–(d) the upper envelope of $f_i(u, v)$ shown as color images on the stereographic projections from the south/north pole $(0, 0, \mp 1)$, and (e)–(f) the corresponding convex hull on the top/bottom of the four surfaces.

To represent the upper envelope as images, we can reparameterize the Gaussian sphere S^2 in rectangular coordinates by employing the stereographic projections of S^2 from the south/north pole $(0, 0, \mp 1)$ to the xy-plane $(z = 0)$:

$$(\bar{n}_x, \bar{n}_y, 0) = \left(\frac{n_x}{n_z \pm 1}, \frac{n_y}{n_z \pm 1}, 0 \right),$$

for all $\mathbf{n} = (n_x, n_y, n_z) \in S^2 \setminus (0, 0, \mp 1)$. The upper envelope function $\mathcal{U}$ is then locally parameterized as $\mathcal{U}(\bar{n}_x, \bar{n}_y)$.

For approximating the upper envelope $\mathcal{U}$, we generate a set of surface sample points $S_i(u, v)$ and compute the corresponding support distances $f_i(u, v)$. The largest value $f_i(u, v)$ at the corresponding location $(\bar{n}_x, \bar{n}_y)$ is then stored as the upper envelope $\mathcal{U}(\bar{n}_x, \bar{n}_y)$ in the graphics rendering hardware depth buffer. At the same time, the corresponding surface identity S_i and the parameter value (u, v) are stored in the frame buffer as an encoded color value. The color image in the frame buffer contains the surface patches that appear on the convex hull boundary and their connectivity with other surface patches sharing common tangent developable surfaces and common tritangent planes.

Fig. 1(a)–(b) show the top and bottom views of four freeform surfaces. The upper envelope is generated as two rendered images of Fig. 1(c)–(d), each clipped to the square: $-1 \le \bar{n}_x, \bar{n}_y \le 1$, from the stereographic projections from the south/north pole. In Fig. 1(d), the vertex in the upper left corner is outside the inscribed circle, which means that this vertex is the projection of a unit normal vector $(n_x, n_y, n_z) \in S^2$, in the upper hemisphere, $n_z > 0$. The projection of (n_x, n_y, n_z) from the south pole is the upper right vertex of Fig. 1(c), which is contained in the inscribed circle.

By processing these images, we can extract curves that separate adjacent regions corresponding to different surface patches. Each point on a separating curve matches two corresponding surface patches and the surface locations that share the same tangent planes. Moving along the separating curve $C(t)$ on the image,