Computer Aided Geometric Design 62 (2018) 91–103

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Fast and robust Hausdorff distance computation from triangle mesh to quad mesh in near-zero cases

Yunku Kang[a], Min-Ho Kyung[b], Seung-Hyun Yoon[c], Myung-Soo Kim[a,*]

[a] Department of Computer Science and Engineering, Seoul National University, Seoul 08826, South Korea
[b] Department of Digital Media, Ajou University, Suwon 16499, South Korea
[c] Department of Multimedia Engineering, Dongguk University, Seoul 04620, South Korea

A R T I C L E I N F O

Article history:
Available online 23 March 2018

Keywords:
Hausdorff distance
Shape matching
Quad mesh

A B S T R A C T

We present an algorithm that computes the one-sided Hausdorff distance from a triangle mesh to a quad mesh. Our algorithm is much more robust than previous ones in the sense that memory requirement is vastly reduced, by avoiding storing combinatorial pairs of each two input model's parts. Meanwhile, point projection via uniform grid greatly accelerates the algorithm. Experimental results show that even for cases where the Hausdorff distance is near zero, its precise computation is done in an interactive speed, while memory consumption is easily manageable.

1. Introduction

Quadrilateral meshes' structural regularity is a major advantage over unstructured triangle meshes in that quad meshes can better represent or preserve principal curvature and feature curves. The structure can be exploited to efficiently create bounding volume hierarchies (Kim et al., 2011), data structure used to accelerate many geometric operations. The regularity can also be regarded as a form of parameterization, which further means that it can easily be converted into parameterized surfaces. In finite element analysis, hexahedral meshes, whose boundary is a quad mesh, are preferable to tetrahedral meshes (Benzley et al., 1995; Cottrell et al., 2009). As such, approximating geometric models with quad meshes has become a widely researched subject, as surveyed by Bommes et al. (2013). The Hausdorff distance is the measure most often used to determine the approximation quality of such quad meshes.

The Hausdorff distance (HD) between two geometric models is an interesting measure with various application possibilities. It becomes zero only when the two models are identical and positioned exactly the same, which implies that the HD can be used for shape matching and as an approximation error (Alt and Guibas, 1999). However, the computation itself is a challenging matter, especially when the HD is near zero, as eliminating parts of the models that do not contribute to the HD becomes harder. Unfortunately, near-zero cases are when HD computation is most relevant, as one would aim to minimize the HD when creating an approximation or finding a match. Such cases also require continuously growing number of iterations and memory space, threatening the proper execution of HD computation algorithms. Previous computation algorithms for HD between mesh models (Aspert et al., 2002; Bartoň et al., 2010; Cignoni et al., 1998; Guthe et al., 2005; Tang et al., 2009) fail in that regard, with even the most recent results occasionally ending up crashing in near-zero cases.

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

https://doi.org/10.1016/j.cagd.2018.03.017
0167-8396/© 2018 Elsevier B.V. All rights reserved.

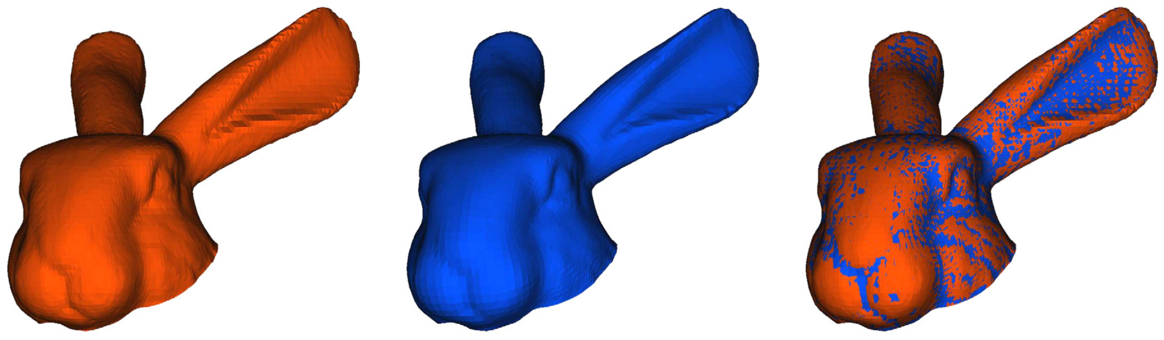
(a) Original triangle mesh. (b) Approximated quad mesh. (c) Superposed.

Fig. 1. An example of triangle mesh approximation by a quad mesh. Measuring the small Hausdorff distance is a way to assess the quality of the approximation. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

For NURBS surfaces, the algorithm of Kim et al. (2013) is one example that successfully computes near-zero HD in real time. But the situation is much more complex for mesh models, which are unlike smooth NURBS surfaces. This may be surprising, since they are the simplest form of surface representation. Consider offsetting one of the input models in the surface normal direction by a continuously increasing radius. When the radius reaches the HD is the point where the offset first envelops the other model. But the discreteness of meshes leads to complex offset representation; vertices become spheres and edges become cylinders. The increased number of combinatorial possibilities complicates the analysis of the offset's relationship with the models.

This hinders quad mesh-to-triangle mesh HD computation, but we are able to take advantage of the surface-like properties of the quad mesh and present a fast and robust algorithm for the computation of the one-sided HD from a triangle mesh to a quad mesh, guaranteed within a very small error bound, even for near-zero HD cases. Experimental results show that this error bound can be arbitrary, up to machine precision without any significant performance drawback. While the one-sided HD is not the HD proper — it should be two-sided — its computation is the first half of the whole HD computation, and thus a crucial step. It can nevertheless be used, to some degree, for the evaluation of approximated quad meshes, aiding in their refinement.

For surfaces in general, the one-sided HD from surface A to surface B is defined as:

$$h(A,B) = \max_{\mathbf{p}\in A} \min_{\mathbf{q}\in B} \|\mathbf{p}-\mathbf{q}\|. \tag{1}$$

In other words, it is the maximum among points in A of the minimum distance to B. As mentioned, it can be used as a measure of B's approximation power of A. Having small $h(A,B)$ means good approximation, since for any point on A, there is a point on B within the radius $h(A,B)$. Thus, fast and accurate computation of the HD aids in the evaluation of the quality of an approximation, and we do this for the case where A is a triangle mesh and B is a quad mesh.

We iteratively split certain relevant pieces of A into smaller pieces, managing them in a queue. Eliminating pieces known to be irrelevant to the HD is a core operation, and their identification (Sections 5.2 and 5.3) is one of our main contributions. Unlike the approach introduced by Tang et al. (2009), the A pieces are the *only* ones enqueued, each by itself, instead of as combinatorial pairs with pieces of B. This greatly reduces the memory requirement necessary for HD computation, resulting in a more robust algorithm, even when the HD is near zero. Meanwhile, assigning the quads of B into a uniform grid allows much faster point projection, which is a bottleneck operation in HD computation.

Since we are dealing in $\mathbb{R}^3$, a quadrilateral is in general non-planar, and is therefore, strictly speaking, a *skew quadrilateral* whose interior surface is up to interpretation. One popular solution in graphics is to split the quad into two triangles, by adding a diagonal edge. As there are two diagonals, this is once again a matter of choice, possible criteria for which include diagonal length and dihedral angle. As long as a split is given for each quad, our algorithm can handle them. One may also consider each quad to be a bilinear surface interpolating its four vertices. For generalization, we cover both forms of interpretation with differences in algorithm details.

To simplify illustrating our algorithm, in Section 3, we first describe the general framework for finding the HD between two surfaces without any assumption about the properties of the surfaces. Then we apply it to the simple problem of computing the HD from a single triangle to a single quad by specifying its ambiguous elements, in Section 4. This is extended to our main problem in Section 5.

2. Related work

Past research on Hausdorff distance computation that does not merely involve point sets is in general easily traceable, likely due to its difficulty. It was first studied for the case of planar polygons with a linear-time algorithm by Atallah (1983), albeit restricted to non-overlapping, convex cases. Planar polygons and polygonal curves were further studied by Alt et al. (1995), with an $O(n \log n)$ time complexity algorithm. Alt et al. (2003), Godau (1999) have also explored HD computation for