Computer Science and Application 计算机科学与应用, 2013, 3, 407-410
http://dx.doi.org/10.12677/csa.2013.39070 Published Online December 2013 (http://www.hanspub.org/journal/csa.html)

Computing the Hausdorff Distance between Two Algebraic Surfaces*

Huahao Shou[1], Yongming Huang[1], Kaili Gu[1], Yongwei Miao[2], Liping Wang[3]

[1]College of Science, Zhejiang University of Technology, Hangzhou
[2]College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou
[3]College of Business and Administration, Zhejiang University of Technology, Hangzhou
Email: shh@zjut.edu.cn

Received: Sep. 27th, 2013; revised: Oct. 29th, 2013; accepted: Nov. 10th, 2013

Abstract: An algorithm for computing the approximate Hausdorff distance as well as its error value between two algebraic surfaces is proposed based on dividing and conquering subdivision technique and interval arithmetic. Theoretically, as long as the size of the voxels is small enough, the computed approximate Hausdorff distance can reach any precision, however, the CPU time used may be overwhelming.

Keywords: Hausdorff Distance; Algebraic Surface; Interval Arithmetic; Subdivision Algorithm

两张代数曲面之间 Hausdorff 距离的计算*

寿华好[1]，黄永明[1]，顾凯丽[1]，缪永伟[2]，王丽萍[3]

[1]浙江工业大学理学院，杭州
[2]浙江工业大学计算机科学与技术学院，杭州
[3]浙江工业大学经贸管理学院，杭州
Email: shh@zjut.edu.cn

收稿日期：2013 年 9 月 27 日；修回日期：2013 年 10 月 29 日；录用日期：2013 年 11 月 10 日

摘　要：基于细分算法和区间算术，本文提出一种计算代数曲面间的 Hausdorff 距离的新算法。该算法在计算出 Hausdorff 距离近似值的同时能给出误差值。在理论上讲，只要设置的体素大小足够小，就可以使得计算出的 Hausdorff 距离近似值与精确值之间的误差达到任意小。但具体计算的时候, 如果精度要求较高则时间成本会变得很高。

关键词：Hausdorff 距离；代数曲面；区间算术；细分算法

1. 引言

Hausdorff 距离是两组点集之间相似程度的一种度量，它度量了两个点集间的最大不匹配程度。Hausdorff 距离在计算机图形学、计算辅助几何设计、计算机视觉、图像处理等领域有十分重要的应用[1]。

已有的有关 Hausdorff 距离的工作一般都是针对点集(图像)、多边形网格、或者参数曲线曲面提出来的。早期 Rucklidge[2]针对 2 维图像提出了一种高效的 Hausdorff 距离计算方法，但该方法很难推广到 3 维。Atallah[3]针对非相交平面凸多边形提出了一种计算时间为线性函数的 Hausdorff 距离计算方法。Barton 等[4]

*项目基金：国家自然科学基金(No. 61272309, 61070135)。

针对多边形网格提出了一种计算精确 Hausdorff 距离的方法，但是速度很慢，达不到实时计算的目的。Tang 等[5]借助于 BVH 技术提出了一种多边形网格之间 Hausdorff 距离近似计算的方法，速度很快，可以达到实时计算的要求。Kim 等[6]借助于双圆弧和深度缓存技术提出了一种计算参数曲线之间 Hausdorff 距离的方法。Bai 等[7]用折线逼近的办法提出了一种计算参数曲线之间 Hausdorff 距离的方法。Chen 等[8]提出了一种计算 B 样条曲线之间 Hausdorff 距离的方法。近期，Hanniel 等[9]使用 GPU 加速技术提出了一种针对 NURBS 曲面的 Hausdorff 距离计算方法。以上这些算法都没有涉及到代数曲面之间的 Hausdorff 距离计算问题。Jüttler[10]对隐式曲线之间或者参数曲线之间的 Hausdorff 距离的上界进行了理论上的估计，但是关于隐式曲线之间 Hausdorff 距离没有给出具体的计算算法。

近年来随着计算机计算能力的大幅提升，代数曲线曲面在计算机图形学和几何造型中的运用越来越多[11]，从而代数曲线曲面间的 Hausdorff 距离的计算也就显得十分重要。然而由于代数曲面的难操作性，一般情况下很难进行参数化，所以到目前为止代数曲面之间的 Hausdorff 距离计算还没有任何算法问世。本文在区间分析和细分算法的基础上针对代数曲面之间的 Hausdorff 距离计算问题首次提出了一种计算方法。该算法的基本思想是用修正仿射算术[12]先对代数曲面进行离散化，然后通过计算离散化后的一个个小立方体(体素)间的 Hausdorff 距离来近似代替代数曲面间的 Hausdorff 距离，在求解过程中借助了八叉树和区间算术进行加速。数值试验表明本文给出的算法能有效且稳定地计算出两张代数曲面之间的 Hausdorff 距离的近似值，并且能在计算出近似值的同时给出误差范围。但是当精度要求较高的时候，时间开销会变得很大。

2. 算法

两张代数曲面 S_1 和 S_2 间的 Hausdorff 距离定义为

$$H(S_1,S_2)=\max\left\{h(S_1,S_2),h(S_2,S_1)\right\},$$

其中 $h(S_1,S_2)=\max\limits_{p\in S_1}\min\limits_{q\in S_2}\|p-q\|$ 称为是从 S_1 到 S_2 的单向 Hausdorff 距离，$h(S_2,S_1)=\max\limits_{p\in S_2}\min\limits_{q\in S_1}\|p-q\|$ 称为是从 S_2 到 S_1 的单向 Hausdorff 距离，$\|p-q\|$ 是某种距离范式(通常取 Euclidean 距离)，取到 Hausdorff 距离的点对称为 Hausdorff 点对。

假设 $f_1(x,y,z)=0$，$f_2(x,y,z)=0$ 是给定的两张空间代数曲面，其中 $f_1(x,y,z)$，$f_2(x,y,z)$ 是三元多项式，所考虑的计算区域是 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$，终止条件给定为 ε。算法的第一个关键步骤是代数曲面离散化，也就是利用八叉树和区间算术分别找出包含这两条代数曲面的像素点的两个集合

$$A_1=\bigcup_{i=1}^{n_1}\left\{[\underline{a}_i,\overline{a}_i]\times[\underline{b}_i,\overline{b}_i]\times[\underline{c}_i,\overline{c}_i]\right\}$$

和

$$A_2=\bigcup_{j=1}^{n_2}\left\{[\underline{d}_j,\overline{d}_j]\times[\underline{e}_j,\overline{e}_j]\times[\underline{f}_j,\overline{f}_j]\right\}。$$

具体过程是：先通过修正仿射算术计算 $f_1(x,y,z)=0$ 在计算区域 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 上的取值范围 $[\underline{f_1},\overline{f_1}]$，然后判断 $[\underline{f_1},\overline{f_1}]$ 是否包含 0，如果 $[\underline{f_1},\overline{f_1}]$ 不包含 0，说明 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 内不包含代数曲面 $f_1(x,y,z)=0$，则抛弃该区域，否则如果 $[\underline{f_1},\overline{f_1}]$ 包含 0，说明 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 有可能包含代数曲面 $f_1(x,y,z)=0$，则将该空间区域在中点处一分为八，通过八叉树算法的递归过程使得细分后的区域逐渐减小，一直细分到区域的大小即区域的长、宽和高都小于等于给定的终止条件 ε 为止，如果还是排除不掉，则将该区域存入 A_1，同理可得 A_2。这里特别值得一提的是：如果不用八叉树数据结构，那么需要对每个小长方体进行判断，比较费时间。而使用八叉树数据结构，那么当 $[\underline{f_1},\overline{f_1}]$ 不包含 0 时，整个区域 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 可以抛掉，不需要对这个区域内的小的长方体作进一步的判断，而能不能把一个不包含代数曲面的区域成功的抛掉又取决于所使用的区间算术的精确度，这也是为什么我们选用相对精确的修正仿射算术的原因。总之八叉树数据结构和相对比较精确的区间算术可以起到计算加速的作用。

算法的第二个关键步骤就是通过计算两个小立方体列表集合 A_1 和 A_2 的距离来近似两张代数曲面间的 Hausdorff 距离。即将计算点与点之间的距离替换为计算两个立方体之间的区间距离。即单向 Hausdorff 区间距离 $\mathbf{h}(A_1,A_2)=\max\limits_{R_1\in A_1}\min\limits_{R_2\in A_2}\|R_1-R_2\|$。其中 A_1 和 A_2

分别表示两张代数曲面离散化后的立方体列表，R_1 和 R_2 分别表示 A_1 和 A_2 中的立方体，$\|R_1-R_2\|$ 表示两个立方体之间的区间距离。假设空间立方体区域 $R_1=[\underline{x}_1,\overline{x}_1]\times[\underline{y}_1,\overline{y}_1]\times[\underline{z}_1,\overline{z}_1]$ 和 $R_2=[\underline{x}_2,\overline{x}_2]\times[\underline{y}_2,\overline{y}_2]\times[\underline{z}_2,\overline{z}_2]$ 那么根据区间算术它们之间的区间距离定义为

$$\|R_1-R_2\|=[\underline{g},\overline{g}]=\left\{([\underline{x}_1,\overline{x}_1]-[\underline{x}_2,\overline{x}_2])^2+([\underline{y}_1,\overline{y}_1]-[\underline{y}_2,\overline{y}_2])^2+([\underline{z}_1,\overline{z}_1]-[\underline{z}_2,\overline{z}_2])^2\right\}^{1/2}。$$

当给定某一立方体 $R_i\in A_1$，i 在 1 与 n_1 之间，因为 A_2 中有 n_2 个立方体，那么遍历 A_2 中所有 n_2 个立方体后将得到 n_2 个区间距离记为 $[\underline{g_{ij}},\overline{g_{ij}}]$，$j=1,2,\cdots,n_2$。令 $[\underline{g}_i,\overline{g}_i]=\left[\min_{j=1}^{n_2}\{\underline{g_{ij}}\},\min_{j=1}^{n_2}\{\overline{g_{ij}}\}\right]$，因为 A_1 中有 n_1 个立方体，若遍历 A_1 中所有的立方体将得到 n_1 个 $[\underline{g}_i,\overline{g}_i]$，再对这 n_1 个区间分别取左端点的最大值和右端点的最大值即可得到单向 Hausdorff 区间距离 $\mathbf{h}(A_1,A_2)$，即

$$\mathbf{h}(A_1,A_2)=\max_{R_1\in A_1}\min_{R_2\in A_2}\|R_1-R_2\|=\left[\max_{i=1}^{n_1}\{\underline{g}_i\},\max_{x}^{n_1}\{\overline{g}_i\}\right],$$

同理可求出 $\mathbf{h}(A_2,A_1)$。

若 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 二者没有交集，那么 $\mathbf{H}(A_1,A_2)$ 就取端点值较大的区间；若二者有交集，那么 $\mathbf{H}(A_1,A_2)$ 就取 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 二者中左端点的较大值以及 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 二者中右端点的较大值所组成的区间。具体算法如下：

1) 输入两张代数曲面的多项式函数表达式 $f_1(x,y,z)$，$f_2(x,y,z)$，以及所要计算的区域范围 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 和终止条件 ε。

2) 利用修正仿射算术计算 $f_1(x,y,z)$ 在区域 $[\underline{x},\overline{x}]\times[\underline{y},\overline{y}]\times[\underline{z},\overline{z}]$ 上的取值范围 $[\underline{f_1},\overline{f_1}]$，如果 $[\underline{f_1},\overline{f_1}]$ 不包含 0，则将该区域剔除，否则将该区域在其中点处一分为八个小区域，对每个小区域重复步骤(2)，一直细分到区域的大小为小于等于给定的终止条件 ε 为止，如果还是排除不掉，则将其存入 A_1，最后得

$$A_1=\bigcup_{i=1}^{n_1}\{[\underline{a}_i,\overline{a}_i]\times[\underline{b}_i,\overline{b}_i]\times[\underline{c}_i,\overline{c}_i]\}。$$

3) 同理可得

$$A_2=\bigcup_{j=1}^{n_2}\{[\underline{d}_j,\overline{d}_j]\times[\underline{e}_j,\overline{e}_j]\times[\underline{f}_j,\overline{f}_j]\}。$$

4) 计算两个立方体之间的区间距离这里有两种方法可供选择，其中一种方法是直接采用区间运算即：

$$[\underline{g_{ij}},\overline{g_{ij}}]=\left\{\left([\underline{a_i},\overline{a_i}]-[\underline{d_j},\overline{d_j}]\right)^2+\left([\underline{b_i},\overline{b_i}]-[\underline{e_j},\overline{e_j}]\right)^2+\left([\underline{c_i},\overline{c_i}]-[\underline{f_j},\overline{f_j}]\right)^2\right\}^{1/2},$$

$i=1,2,\cdots,n_1$，$j=1,2,\cdots,n_2$；另一种方法是通过分别计算出立方体间的八个顶点间的距离，在计算出的 64 个距离值中以其中的最小值作为区间距离 $[\underline{g_{ij}},\overline{g_{ij}}]$ 的左端点，以其中的最大值作为区间距离 $[\underline{g_{ij}},\overline{g_{ij}}]$ 的右端点。令 $[\underline{g}_i,\overline{g}_i]=\left[\min_{j=1}^{n_2}\{\underline{g_{ij}}\},\min_{j=1}^{n_2}\{\overline{g_{ij}}\}\right],i=1,2,\cdots,n_1$，则 $\mathbf{h}(A_1,A_2)=\left[\max_{i=1}^{n_1}\{\underline{g}_i\},\max_{i=1}^{n_1}\{\overline{g}_i\}\right]$。同理可得 $\mathbf{h}(A_2,A_1)$。

5) 如果 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 不相交，取端点值较大的作为 $\mathbf{H}(A_1,A_2)$；如果二者有交集，那么 $\mathbf{H}(A_1,A_2)$ 就取 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 二者中左端点的较大值以及 $\mathbf{h}(A_1,A_2)$ 和 $\mathbf{h}(A_2,A_1)$ 二者中右端点的较大值所组成的区间，算法结束。

由于区间运算的保守性，两张代数曲面之间的精确 Hausdorff 距离 $H(A_1,A_2)$ 一定落在已经通过区间算术和细分算法计算得到的 Hausdorff 区间距离 $\mathbf{H}(A_1,A_2)$ 之中。所以我们可以取 $\mathbf{H}(A_1,A_2)$ 的中点作为 $H(A_1,A_2)$ 的近似值，而且误差一定不会超过区间 $\mathbf{H}(A_1,A_2)$ 长度的一半。也就是说我们在计算出 Hausdorff 距离的近似值的同时给出了误差值，这是很多其它有关 Hausdorff 距离计算的算法所不具备的优点。此外，从理论上讲只要 ε 足够地小，可以使得计算出的 Hausdorff 距离的误差达到任意小，但具体计算的时候如果要求误差很小那么计算时间的开销会变得非常大。

3. 实例

我们用 Mathematica 8.0 编程实现上面的算法，并

且在配置为 Intel Core Duo CPU E7500 @2.93 GHz 处理器内存为 2 GB 的计算机里运行该程序进行了相应的实例计算。

例：计算以下代数曲面在 $[-2,2]\times[-2,2]\times[-2,2]$ 区域范围内的 Hausdorff 距离：

$$f_1(x,y,z)=\frac{1}{4}x^2+\frac{1}{4}y^2-\frac{1}{9}z^2,$$

$$f_2(x,y,z)=\frac{1}{16}x^2+\frac{1}{4}y^2-\frac{1}{9}z^2$$

在计算过程中我们取 $\varepsilon=\frac{1}{64}=0.015625$，计算的结果得到 Hausdorff 区间距离为

$$\left[\frac{\sqrt{\frac{17}{2}}}{4},\frac{\sqrt{\frac{103}{2}}}{8}\right]\approx[0.7288689868,0.8970437559],$$

我们取该区间的中点值 0.81295637135 作为这两个代数曲面间 Hausdorff 距离的近似值，那么误差不超过该区间长度的一半即 0.08408738455。计算花费总的 CPU 时间为 789694 秒。如果需要进一步提高精度，那么 ε 应该取更小的数值，但计算时间会变得无法忍受。如何对算法进行优化，或者利用计算机硬件设备比如 GPU 等进行加速是我们下一步需要做的工作。

4. 结论

从以上算法描述和实例可以看出，本文提出的算法可以有效地计算出两张代数曲面之间 Hausdorff 距离的近似值，并且在计算出 Hausdorff 距离近似值的同时给出了误差范围。本文的意义在于首次针对代数曲面之间的 Hausdorff 距离给出了一种计算方法，由于到目前为止还没有其它针对代数曲面之间的 Hausdorff 距离的算法出现，我们无法进行比较研究。本文算法的缺点是当精度要求较高的时候耗时过多，这个问题有待将来通过算法优化或者利用 GPU 加速进行改进。

参考文献 (References)

[1] Kim, Y.-J., Oh, Y.-T., Yoon, S.-H., et al. (2013) Efficient Hausdorff distance computation for freeform geometric models in close proximity. *Computer Aided Design*, **45**, 270-276.

[2] Rucklidge, W. (1996) Efficient visual recognition using the Hausdorff distance. *Lecture Notes in Computer Science*, **1173**, 1-161.

[3] Atallah, M. (1983) A linear time algorithm for the Hausdorff distance between convex polygons. *Information Processing Letters*, **17**, 207-209.

[4] Barton, M., Hanniel, I., Elber, G., et al. (2010) Precise Hausdorff distance computation between polygonal meshes. *Computer Aided Geometric Design*, **27**, 580-591.

[5] Tang, M., Lee, M. and Kim, Y.-J. (2009) Interactive Hausdorff distance computation for general polygonal models. *ACM Transactions on Graphics*, **28**, 1-9.

[6] Kim, Y.-J., Oh, Y.-T., Yoon, S.-H., et al. (2010) Precise Hausdorff distance computation for planar freeform curves using biarcs and depth buffer. *The Visual Computer*, **26**, 1007-1016.

[7] Bai, Y.-B., Yong, J.-H., Liu, C.-Y., et al. (2011) Polyline approach for approximating the Hausdorff distance between two planar free-form curves. *Computer Aided Design*, **43**, 687-698.

[8] Chen, X.-D., Ma, W., Xu, G., et al. (2010) Computing the Hausdorff distance between two B-spline curves. *Computer Aided Design*, **42**, 1197-1206.

[9] Hanniel, I., Krishnamurthy, A. and McMains, S. (2012) Computing the Hausdorff distance between NURBS surfaces using numerical iteration on the GPU. *Graphical Models*, **74**, 255-264.

[10] Jüttler, B. (2000) Bounding the Hausdorff distance of implicitly defined and/or parametric curves. Mathematical Methods in CAGD, 1-10.

[11] Li, Q.-D. and Tian, J. (2009) 2D piecewise algebraic splines for implicit modeling. *ACM Transactions on Graphics*, **28**, 1-13.

[12] Shou, H.-H., Lin, H.-W., Ralph, M., et al. (2003) Modified affine arithmetic is more accurate than centered interval arithmetic or affine arithmetic. *Lecture Notes in Computer Science*, **2768**, 355-365.