论 文

模型适应的凸包围多面体并行生成算法

唐磊^{①②③}, 施侃乐^{①②③}, 雍俊海^{①②③}, 古和今^{④*}, 徐鑫^{①②③⑤}

① 清华大学软件学院, 北京 100084

② 信息系统安全教育国家重点实验室, 北京 100084

③ 清华信息科学与技术国家实验室, 北京 100084

④ 江西省科学院能源研究所, 南昌 330096

⑤ 清华大学计算机科学与技术系, 北京 100084

* 通信作者. E-mail: guhejin@vip.163.com

收稿日期: 2014-01-10; 接受日期: 2014-07-21; 在线日期: 2014-08-26

国家国际科技合作专项 (批准号: 2013DFE13120)、国家重点基础研究发展计划 (973 计划)(批准号: 2010CB328001) 和国家自然科学基金 (批准号: 61272235, 61173077) 资助项目

摘要 包围盒在计算机图形学和计算几何领域中应用广泛, 常用于加速几何求交、光线跟踪和碰撞检测等多种算法. 凸包围多面体是包围盒的推广, 对于一般不规则形体, 可达到比包围盒更好的紧致程度. 本文提出一种快速构造给定点集的紧致凸包围多面体的方法. 该方法首先根据点集的近似凸包, 通过 k -means 算法生成 k 个截面法向, 然后利用 GPU 沿各法向搜索切点构成截面, 最后求交构成多面体. 实验结果表明, 与同类算法相比, 该方法能够更快地构造给定点集更紧致的凸包围多面体, 并能有效加速碰撞检测算法.

关键词 凸包围多面体 近似凸包 包围体 并行计算 碰撞检测

1 引言

在计算机图形学、计算机辅助设计与制造等领域, 轴向包围盒 (axis-aligned bounding box, AABB) 和方向包围盒 (oriented bounding box, OBB) 常作为原始复杂模型的近似被广泛应用于光线跟踪、碰撞检测等算法, 如果两个模型的包围盒不相交则对应的原始模型一定不相交, 而包围盒的相交判断比原始模型之间更简单, 因此包围盒可用于提高几何算法的性能^[1~3]. 包围体作为包围盒的推广适用于不同场景, 如包围球^[4]适用于几何对象频繁旋转的场景. 包围体的紧致程度直接影响算法的效率, 对于不规则形体, 常见的包围体往往不够紧致, 凸包紧致但往往包含过多的面片数而增加算法的复杂性.

本文提出一种构造凸包围多面体的方法, 该方法首先利用近似内凸包和 k -means 聚类算法生成构成凸包围多面体 k 个截面的法向, 然后根据输入点集沿各法向搜索切点构成截面, 最后由这些截面通过对偶映射的方式求交构成凸包围多面体. 该方法的主要优势是: 1) 对给定点集可构造紧致的包围体; 2) 利用 GPU 加速, 能够快速构造包围体; 3) 通过参数 k 调节凸包围多面体的简单性和紧致性, 可适用于不同的应用场景.

引用格式: 唐磊, 施侃乐, 雍俊海, 等. 模型适应的凸包围多面体并行生成算法. 中国科学: 信息科学, 2014, 44: 1515–1526, doi: 10.1360/N112014-00019

2 相关工作

沿坐标轴方向的包围盒 AABB 是最简单的凸包围多面体, 因其相交测试的简单性和易实现性被广泛应用于计算机图形学中^[1], AABB 包围盒对细长且斜置的几何对象往往紧密性较差. OBB 包围盒是沿任意方向的 AABB, 文献 [2, 3] 介绍了如何确定特定的方向使得 OBB 体积最小的算法, 与 AABB 相比, OBB 更紧致但难于构造且相交测试较复杂. 包围球存储简单应用广泛^[4], 但对于普通非球状形体的包围往往不够紧致, 文献 [5] 介绍了一种计算最小包围球的线性算法. 离散方向多面体 (k -discrete orientation polytope, k -DOP)^[6] 用 $k/2$ 对平行平面来包围物体, 也被称为固定方向凸包 (fixed directions hull, FDH)^[7], 最早由 Kay 等人^[8] 引入用以解决光线追踪问题. k -DOP 方向固定, 对不同输入的模型其包围体面的法向一致, 导致对一般不规则形体的包围体不够紧致. 此外, 还有其他形状的包围体, 如包围椭球^[9]、包围柱^[10]、Tribox^[11] 和 Zonotopose^[12] 等. 凸包紧致但面片数量过多且构造时间复杂度为 $O(n \log n)$, 不适用于大模型. 在一些应用中需用到近似凸包, 近似凸包可分为三类: 近似外凸包、近似内凸包和近似凸包^[13], 文献 [14, 15] 介绍了两种计算近似内凸包的算法, 文献 [16] 介绍了一种计算二维近似外凸包算法, 从某种意义上讲, 本文算法是计算一种三维近似外凸包. 文献 [17] 介绍了计算机图形学领域中用到的如空间划分树等更多的数据结构及相应算法. 随着 GPU 的产生, 研究人员充分利用其计算能力来提高程序性能, 例如用于加速碰撞检测^[18]、光线跟踪^[19] 等算法. Lauterbach 等人^[20] 提出了一种基于 CUDA 框架的 AABB 树构造方法并应用于光线跟踪, Karlsson 等人^[21] 利用 SSE 指令即每次操作可同时处理 4 个浮点数来加速计算 AABB、包围球和 k -DOP 等.

3 模型适应的凸包围多面体构造算法

由 k 个截面构成的凸包围多面体称为凸包围 k 面体 (k -convex bounding polytope, k -CBP), 可通过 k 个半空间定义:

$$\begin{cases} k\text{-CBP} = \bigcap_{i=1}^k H_i \\ H_i = \{p \in \mathbb{R}^3 \mid n_i \cdot p \leq w_i, w_i \in \mathbb{R}\}, \end{cases} \quad (1)$$

其中, n_i 是半空间 H_i 的法向, 方向指向包围体外部, w_i 是输入点集中沿 n_i 方向投影的最大值. 图 1 是 Bunny 模型的凸包围 34 面体 (34-CBP).

k -CBP 的紧致程度用精确凸包与其体积之比衡量, 该值越趋近于 1 表明这个包围体越紧致.

本文提出的算法主要流程如图 2 所示, 首先利用近似内凸包和 k -means 聚类算法生成构成凸包围多面体 k 个截面的法向, 然后根据输入点集在 GPU 中沿各法向搜索切点构成截面, 最后由这些截面通过对偶映射的方式求交构成凸包围多面体. 搜索截面需要多次扫描输入点集, 在 CPU 中计算耗时, 因此用 GPU 加速使得算法整体性能得以提升, 其他步骤在 CPU 计算即可.

3.1 截面法向的生成

截面法向的选定与多面体的紧致程度密切相关. k -DOP^[6] 预先定义 $k/2$ (k 为偶数) 对法向, 导致其生成的多面体不能自适应模型, 对于不规则形体不够紧致. 根据经验, 为了使凸包围多面体尽可能逼近凸包, 在多面体数量一定的情况下需尽量保留凸包中面积较大的面. 本文算法先通过一种线性算

图 1 Bunny 模型的 34-CBP

Figure 1 34-CBP of Bunny model

图 2 算法流程图

Figure 2 Flowchart of the algorithm

图 3 二维近似内凸包的构造

Figure 3 Approximate convex hull construction in 2D. (a) Place points in bins; (b) find extremes in bins; (c) choose extremes to build approximate hull

法^[14]构造近似内凸包, 然后从近似内凸包里选取面积最大的 a 个面的法向, 最后用聚类算法从剩下的面中生成 $k - a$ 个法向.

以平面点集的近似内凸包为例, 如图 3 所示, 首先根据 x 轴将输入点均分 ξ 组, 然后在每组中找出沿 y 轴的最大最小值, 最后按条件^[14]连接每组的最值构成近似内凸包. 此算法复杂度为 $O(n + \xi)$. 在三维空间里, 可按照 x, y 轴最多划分成 $(\xi + 2) \times (\xi + 2)$ 个网格, 每个网格取 z 的最大最小值, 因此所有网格含最多有 $2(\xi + 2)^2$ 个极值点, 其凸包可以在 $O(\xi^2 \log \xi^2) = O(\xi^2 \log \xi)$ 内找出, 因此整个算法时间复杂度为 $O(n + \xi^2 \log \xi)$, 关于此算法更详细的细节可参考文献 [14]. 实验过程中, 为了更快地构造近似内凸包, ξ 值通常取得较小 (例如取 $\xi = 10$).

图 4 从左至右分别显示了 Bunny 模型的 26-CBP、精确凸包和近似内凸包. 近似凸包是精确凸包的一种近似, 与精确凸包外观相似但其构造复杂度降低, 不少研究者利用该性质解决计算机图形学中

图 4 Bunny 模型的 26-CBP、凸包和近似内凸包

Figure 4 26-CBP, convex hull and approximate convex hull of Bunny model

图 5 通过聚类确定法向 ($k = 26$)Figure 5 Normals determined using k -means ($k = 26$). (a) Normals generated by equalization area; (b) 26-CBP for bunny using initial normals; (c) 26-CBP for bunny using clustered normals

很多问题^[13]. 本文也利用近似内凸包与精确凸包的相似性, 从众多近似凸包面片的法向中通过聚类算法生成凸包围多面体的 k 个法向.

假设近似内凸包共含 m 个面, 保留面积最大的 a 个 (例如取面积最大的 $k \cdot 10\%$ 个¹⁾) 平面所在的法向, 利用 k -means 算法对剩下的 $m - a$ 个面的法向聚类, 即从原始 $m - a$ 个点聚成 $k - a$ 个类. 聚类算法的初始中心点采用如下策略生成: 给定一个单位球, 将其等面积均分 k 份²⁾, 连接球心到每份的中心点构成的方向作为该份的法向, 其效果如图 5(a) 所示. 另外亦可使用如文献 [22] 中的数学方法在球面上生成若干点使其满足均匀分布.

聚类算法使用余弦距离度量, 聚类更新中心点时将面片的面积作为权重即 $\mathbf{c}_i = \frac{\sum_{j=1}^n \omega_j \cdot \mathbf{n}_j}{\sum_{j=1}^n \omega_j}$, 其中 $\mathbf{c}_i$ 为第 i 类的中心点, ω_i 为法向 $\mathbf{n}_i$ 所在面片对应的面积, 这样使得生成的法向尽量靠近原始近似凸包面积较大的面片的方向.

如图 5 所示, 其中图 5(b) 为按照初始方向为 Bunny 模型生成的凸包围多面体 ($k = 26$), 聚类后其法向及其生成的凸包围多面体如图 5(c) 所示, 后者的紧致程度比前者高 14.98%.

3.2 搜索截面及求交

搜索多面体截面等效于寻找最大投影值, 即对每个法向 $\mathbf{n}_i$, 从输入模型的所有点中寻找最大投影值的点作为切点进而确定 $\mathbf{n}_i$ 对应的截面. 该过程的算法复杂度为 $O(k \cdot n)$, 其中 k 为法向数量, n 为

1) $k \cdot 10\%$ 是一个实验得出的经验值.

2) 算法可参考 <http://www.cmu.edu/biolphys/deserno/pdf/sphere.equi.pdf>.

图 6 并行规约求最大投影值

Figure 6 Calculation of maximum projection value using reduction

模型所含点数. 该搜索过程中各法向的计算相互独立, 因此可借助 GPU 并行加速.

如图 6 所示, 最大投影值的计算可采用如下规约方式: 将输入点交给数量为 t 的线程计算点积得到投影值, 线程 i 和 $i + t/2$ 比较选取较大者, 经 $\log_2 t$ 次比较可得最大值. 详细规约优化技术可参考文献 [23].

确定截面后对其进行对偶映射求凸包, 该凸包面映射的点即为原始截面的交点即凸包围多面体的顶点, 方法如下: 每个平面经对偶映射成一个三维点, 例如平面由法向 $\mathbf{n}(a, b, c)$ 及平面上一点 $\mathbf{p}(x_0, y_0, z_0)$ 确定, 转化为平面方程 $ax + by + cz = ax_0 + by_0 + cz_0 = d$, $d \neq 0$. 对偶映射后的点为 $\mathbf{p}'(a/d, b/d, c/d)$, 然后通过这 k 个映射点求凸包, 最后将凸包的每个平面转换即可得到原始平面的交点, 此算法的时间复杂度为 $O(k \log k)$, 在利用对偶映射求交点时要注意原始模型需包含原点. 另外亦可直接通过枚举所有每 3 个平面交于 1 点的情况, 然后排除在平面外部的交点, 剩下的构成 k -CBP 的顶点 [24].

4 实验与分析

本文算法在 CUDA 框架下实现, 并针对不同点集规模的模型进行测试³⁾. 将本文算法与传统基于 CPU 的算法及文献 [21] 中的并行算法从计算凸包围多面体的速度上进行了对比, 与文献 [25] 中的 k -DOP 从凸包围多面体的紧致程度上进行了对比. 同时, 我们将生成的凸包围多面体应用于一个简单的碰撞检测场景, 并与传统的基于包围盒的方法进行比较, 实验验证了本文提出的凸包围多面体有助于提高碰撞检测算法的效率.

4.1 凸包围多面体的生成速度

与传统 CPU 算法进行对比. 如图 7 所示为本文算法与传统的 CPU 算法对比结果, 图中横纵坐标分别代表多面体面数和运行时间, 其中虚线代表搜索截面的过程, 实线为构造凸包围多面体总体耗时.

3) 运行时环境为: Intel(R) Core(TM) i5-2320 CPU @ 3.2GHz 8G RAM NVIDIA GeForce GTX 650.

图 7 本文算法与 CPU 算法对比

Figure 7 Comparison of result between CPU method and GPU's. (a) Budda (31232 points); (b) Dinosaur (40277 points); (c) Alice (224291 points); (d) Bugatti (1010815 points)

当模型点数量较大时, 搜索截面的过程占据了算法绝大多数时间, 且随着凸包围多面体的面数 k 值增加而线性增长, 这与搜索截面时间复杂度 ($O(k \cdot n)$) 一致, 截面求交过程的时间复杂度为 $O(k \log k)$, 当点数量极大时, 如图 7(d) 所示, 实线虚线几乎重合即求交等步骤耗时相比整体算法而言几乎可忽略. 输入模型的点数量越大, 本文算法的优势越明显. 如 Budda 模型点数量 3.1×10^4 左右, 加速比约为 3~6 倍, 而含有 2.24×10^5 个点的 Alice 模型和 1.01×10^6 个点的 Bugatti 模型, 能够加速 7~9 倍.

与文献 [21] 中的并行算法对比. 我们实现了文献 [21] 中的算法并在相同机器上进行实验, 表 1 为实验统计结果 (均为搜索截面耗时, 二者其他步骤相同), 其中 k 为多面体面数, 列 SSE 和列 k -CBP 分别为文献 [21] 中的算法和本文算法的运行时间. 与文献 [21] 中的算法相比, 本文算法优势明显. 点数量较小时, 能够提高 3~4 倍速度, 模型变大, 加速比更大, Bugatti 模型的提速达到 4~8 倍.

4.2 凸包围多面体的紧致程度

本文算法通过凸包与凸包围多面体的体积之比来量化包围体的紧致程度. 文献 [25] 是基于 k -DOP 实现的层次结构的包围体, 其顶层的 k -DOP 与本文算法生成的多面体的紧致程度对比如图 8 所示, 图中横坐标 k 为多面体面数, 纵坐标为紧致程度, 曲线 k -DOP 为文献 [25] 的方法, 曲线 k -CBP 为本文方法, 由图可知, 对于不同模型, 本文构造的凸包围多面体紧致程度均有提升, 如 Alice 模型提升了 12.08%, Dinosaur 模型提升了 34.00%.

表 1 本文算法与文献 [21] 算法对比

Table 1 Comparison of result between [21] and ours

k	Apple(8118 points)			Bugatti(1010815 points)		
	SSE ^[21] (ms)	k -CBP (ms)	Speedup	SSE (ms)	k -CBP (ms)	Speedup
6	0.4	0.12	3.20	24.2	3.20	7.56
16	0.9	0.26	3.43	44.5	8.44	5.27
26	1.4	0.41	3.38	66.5	13.65	4.87
36	1.9	0.52	3.65	91.1	18.34	4.97
46	2.5	0.67	3.74	119.5	24.13	4.95
56	2.9	0.79	3.66	138.4	28.86	4.80
66	3.5	0.95	3.69	170.6	34.10	5.00
76	4.0	1.08	3.70	197.1	39.85	4.95
86	4.5	1.22	3.69	219.8	45.08	4.88
106	5.4	1.49	3.62	267.8	55.52	4.82
136	6.8	1.92	3.54	342.9	71.24	4.81
156	7.7	2.17	3.55	411.3	81.18	5.07
186	9.3	2.60	3.58	479.4	97.39	4.92
206	10.5	2.85	3.68	523.0	106.87	4.89

图 8 紧致程度对比: k -DOP 为文献 [25] 的算法, k -CBP 为本文算法

Figure 8 Comparison of result of tightness between k -DOP [25] and k -CBP. (a) Apple (8118 points); (b) Buddha (31232 points); (c) Dinosaur (40277 points); (d) Alice (224291 points)

表 2 k -CBP 与 QuickHull 凸包算法比较Table 2 Comparison of result between QuickHull and k -CBP

Model	$f(\text{CHull})$	$f(k\text{-CBP})$	$\tau(k\text{-CBP})(\%)$	$t(\text{CHull})(\text{ms})$	$t(k\text{-CBP})(\text{ms})$
Apple	499	30	93.67	5.5	1.30
Budda	1608	46	92.39	21.3	2.86
Dinosaur	1240	44	93.34	22.6	1.99
Alice	1332	44	93.92	85.8	8.47
Bugatti	24 654	44	95.06	688.7	25.41

图 9 k -CBP 与凸包对比

Figure 9 Comparison of result between k -CBP and convex hull. (a) Apple (30-CBP); (b) Budda (46-CBP); (c) Dinosaur (44-CBP); (d) Alice (44-CBP); (e) Bugatti (44-CBP); (f) Apple (CHull); (g) Budda (CHull); (h) Dinosaur (CHull); (i) Alice (CHull); (j) Bugatti (CHull)

本文算法与 CGAL⁴⁾库中利用 QuickHull 算法构造的凸包进行比较的结果如表 2 所示, 其中 $f(\text{CHull})$ 和 $f(k\text{-CBP})$ 分别表示凸包面数和凸包围多面体的面数, $\tau(k\text{-CBP})$ 为凸包围多面体的紧致程度, $t(\text{CHull})$ 和 $t(k\text{-CBP})$ 分别表示凸包和 k -CBP 构造所花费时间. 相应模型的可视化结果如图 9 所示, 与凸包相比, 本文算法在大大简化包围体平面数量的同时能保持较好的紧致程度, 例如 Apple 模型的凸包有 499 个平面, 本文算法仅用 30 个平面就能达到 93.67% 的紧致程度, 而对 Bugatti 模型, 仅用了其凸包平面数量的 0.17% 就达到 95.06% 的紧致程度, 且构造速度快了 27 倍.

本文算法利用近似凸包与精确凸包的相似性, 从近似凸包的众多面片对应的法向中通过 k -means 聚类算法生成 k 个法向. 表 3 为 Alice 模型利用近似凸包和精确凸包聚类生成法向构造 k -CBP 的紧致程度对比. 可以看出, 一方面, 利用精确凸包构造的 k -CBP 并不一定比近似凸包构造的结果更紧致, 但由于近似凸包与精确凸包的外观的近似性因此二者得到的结果相差并不大 (5% 以内). 另一方面, 近似凸包的时间复杂度为线性, 而精确凸包为 $O(n \log n)$, 因此本文利用近似凸包聚类生成法向. 示例中构造近似凸包耗费时间仅为 0.88 ms, 而精确凸包为 79.31 ms.

4) CGAL. Computational Geometry Algorithms Library. <http://www.cgal.org>, 2013

表 3 用近似凸包与精确凸包构造 k -CBP 紧致程度对比

Table 3 Comparison of result of tightness using between approximate convex hull and exact convex hull

k	τ (ACHull)(%)	τ (CHull)(%)	k	τ (ACHull)(%)	τ (CHull)(%)
10	70.44	65.56	26	89.93	94.45
12	74.56	75.63	28	88.61	92.24
14	79.50	82.49	30	91.58	92.35
16	84.79	85.20	32	92.78	93.78
18	85.75	90.09	34	91.28	93.22
20	86.38	90.36	36	92.93	94.68
22	90.27	92.48	38	93.41	93.20
24	90.70	93.09	40	93.81	94.70

图 10 k -CBP 应用于碰撞检测示例

Figure 10 Example of collision detection using k -CBP

4.3 凸包围多面体的简单应用

凸包围多面体可应用于加速相关几何算法的整体效率, 图 10 为利用 Bunny 模型进行碰撞检测的示例, 图中模型 1 与 2, 2 与 3 的包围盒分别相交, 而其 16-CBP 仅 1 与 2 相交, 实际模型仅 1 与 2 相交. 用 16-CBP 可排除模型 2 与 3 之间的碰撞检测, 而仅用包围盒算法则无法排除, 显然检测模型 2 与 3 的 16-CBP 是否相交比直接通过检测模型 2 与 3 是否相交更省时间.

本文实验通过生成不同数量的模型 (模型位置和旋转角度随机生成), 碰撞检测时首先判断包围盒是否相交, 然后判断凸包围多面体是否相交, 最后再判断实际模型是否相交. 凸包围多面体之间的碰撞检测时可采用文献 [26] 中提到的方法, 本文案例中模型和凸包围多面体是否相交都采用了普通 AABB 树 [17] 的方式进行判断, 从实验结果 (表 4) 可看出含有凸包多围体的模型之间的碰撞检测算法能显著提高整体应用的效率.

如表 4 所示, 其中 n 表示场景中模型的数量, $c(\text{Box})$, $c(16\text{-CBP})$ 分别表示模型包围盒的构造时间和凸包围 16 面体的构造时间, $t(\text{Box})$, $t(16\text{-CBP})$ 分表表示用包围盒进行碰撞检测和利用凸包围 16 面体进行碰撞检测所耗费的时间, $r(\text{Box})$, $r(16\text{-CBP})$ 分别表示包围盒、16-CBP 的命中率 (即用实际模型相交的数量除以包围体检测出来相交的数量), $n(\text{Model})$ 模型实际相交的数量, 显然计算模型包围

表 4 k -CBP 和包围盒应用于碰撞检测结果对比Table 4 Comparison of result of collision detection between k -CBP and bounding box

n	$c(\text{Box})(\text{ms})$	$c(16\text{-CBP})(\text{ms})$	$t(\text{Box})(\text{ms})$	$t(16\text{-CBP})(\text{ms})$	$r(\text{Box})(\%)$	$r(k\text{-CBP})(\%)$	$n(\text{Model})$
10	0.1	1.8	26.0	0.1	0.00	100.00	0
30	0.2	2.9	134.0	70.0	45.45	83.33	5
50	0.5	4.8	506.0	255.2	46.34	86.36	19
70	0.4	4.8	901.1	492.5	44.16	80.95	34
90	0.7	5.7	1324.0	734.7	41.82	73.02	46
100	0.7	7.8	1481.0	870.7	43.31	75.34	55
150	1.0	9.8	4153.1	2473.0	42.98	70.75	150
200	1.6	12.8	8049.3	4430.9	41.02	71.32	281

盒所耗费的时间要明显少于计算凸包围多面体的时间, 但由于凸包围多面体比包围盒紧致, 因而命中率比包围盒高, 能排除更多本不相交的模型进而节省碰撞检测总时间, 提高算法效率.

5 结论

本文提出了一种快速构造紧致凸包围多面体的算法, 实验证明本文算法比传统 CPU 算法快 7~9 倍, 比文献 [21] 中的并行算法快 3~7 倍, 且生成的包围体比传统的 k -DOP^[25] 紧致 15%~25%. 本文算法经简单改动能移植到 DirectX, OpenCL 等平台. 实验验证了本文算法应用于碰撞检测有助于提高应用的整体效率.

参考文献

- 1 Bergen G V D. Efficient collision detection of complex deformable models using aabb trees. J Graph Tools, 1997, 2: 1-13
- 2 Joseph O R. Finding minimal enclosing boxes. Int J Comput Inf Sci, 1985, 14: 183-199
- 3 Chan C K, Tan S T. Determination of the minimum bounding box of an arbitrary solid: an iterative approach. Comput Struct, 2001, 79: 1433-1449
- 4 Larsson T. Fast and tight fitting bounding spheres. In: Proceedings of The Annual Swedish Computer Graphics Association Conference. Sweden: Linkoping University Electronic Press, 2008. 27-30
- 5 Welzl E. Smallest enclosing disks (balls and ellipsoids). In: Maurer H, ed. New Results and New Trends in Computer Science, LNCS 555. Berlin: Springer Berlin Heidelberg, 1991. 359-370
- 6 Klosowski J T, Held M, Mitchell J S B, et al. Efficient collision detection using bounding volume hierarchies of k-dops. IEEE Trans Vis Comput Graph, 1998, 4: 21-36
- 7 Wei Y M, Wang Y, Wu Q Y, et al. Research on fixed direction hull bounding volume in collision detection. J Softw, 2001, 12: 1056-1063 [魏迎梅, 王涌, 吴泉源, 等. 碰撞检测中的固定方向凸包围盒的研究. 软件学报, 2001, 12: 1056-1063]
- 8 Kay T L, Kajiya J T. Ray tracing complex scenes. In: Proceedings of ACM SIGGRAPH Computer Graphics. New York: ACM Press, 1986. 269-278
- 9 Wang W P, Choi Y K, Chan B, et al. Efficient collision detection for moving ellipsoids using separating planes. Computing, 2004, 72: 235-246
- 10 Schomer E, Sellen J, Teichmann M, et al. Smallest enclosing cylinders. Algorithmica, 2000, 27: 170-186
- 11 Crosnier A, Rossignac J. Tribox bounds for three-dimensional objects. Comput Graph, 1999, 23: 429-437

- 12 Guibas L J, Nguyen A, Zhang L. Zonotopes as bounding volumes. In: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms. New York: ACM Press, 2003. 803–812
- 13 Hossain M Z, Amin M A. On constructing approximate convex hull. Am J Comput Math, 2013, 3: 11–17
- 14 Bentley J L, Preparata F P, Faust M G. Approximation algorithms for convex hulls. Commun ACM, 1982, 25: 64–68
- 15 Kavan L, Kolingerova I, Zara J. Fast approximation of convex hull. In: Proceedings of the IASTED International Conference on Advances in Computer Science and Technology. Canada: ACTA Press, 2006. 101–104
- 16 Zunié J. An outer approximation of the convex hull for finite grid point sets. Novi Sad J Math, 1992, 22: 177–185
- 17 Zachmann G, Langetepe E. Geometric data structures for computer graphics. Tutorial ACM SIGGRAPH, 2003, 16: 1–54
- 18 Tang M, Lin J, Tong R F. Graphics hardware accelerated continuous collision detection between deformable objects. Chin J Comput, 2010, 33: 2022–2030 [唐敏, 林江, 童若锋. 图形硬件加速的柔性物体连续碰撞检测. 计算机学报, 2010, 33: 2022–2030]
- 19 Lu H Q, Bao P, Feng J Q. OpenCL-based real-time KD-Tree and raytracing for dynamic scene. J Comput Aided Des Comput Graph, 2013, 25: 963–973 [卢贺齐, 鲍鹏, 冯结青. 基于 OpenCL 的实时 KD-Tree 与动态场景光线跟踪. 计算机辅助设计与图形学学报, 2013, 25: 963–973]
- 20 Lauterbach C, Garland M, Sengupta S, et al. Fast bvh construction on gpus. In: Proceedings of Eurographics, Computer Graphics Forum, Wiley Online Library, 2009, 28: 375–384
- 21 Karlsson M, Winberg O, Larsson T. Parallel construction of bounding volumes. In: Proceedings of the Annual Swedish Computer Graphics Association Conference. Sweden: Linköping University Electronic Press, 2010. 65–69
- 22 Wong T T, Luk W S, Heng P A. Sampling with Hammersley and Halton points. J Graph Tools, 1997, 2: 9–24
- 23 Harris M. Optimizing parallel reduction in CUDA. NVIDIA Developer Technology, 2007. 1–37
- 24 Ericson C. Real-Time Collision Detection. San Francisco: Morgan Kaufmann Publishers, 2005. 114–121
- 25 Zachmann G. Rapid collision detection by dynamically aligned DOP-trees. In: Proceedings of Virtual Reality Annual International Symposium. Piscataway: IEEE, 1998. 90–97
- 26 Ehmann S A, Lin M C. Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching. In: Proceedings of Intelligent Robots and Systems. Piscataway: IEEE, 2000. 2101–2106

Parallel construction for model-adaptive convex bounding polyhedron

TANG Lei^{1,2,3}, SHI KanLe^{1,2,3}, YONG JunHai^{1,2,3}, GU HeJin^{4*} & XU Xin^{1,2,3,5}

1 School of Software, Tsinghua University, Beijing 100084, China;

2 Key Laboratory for Information System Security, Ministry of Education, Beijing 100084, China;

3 Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China;

4 Energy Institute of Jiangxi Academy of Science, Nanchang 330096, China;

5 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

*E-mail: guhejin@vip.163.com

Abstract Bounding box is extensively used to accelerate algorithms such as geometric intersection, ray tracing and collision detection in computer graphics and computational geometry. Convex bounding polyhedron, which is a generalization of bounding box, can approximate a point set with higher tightness compared to the corresponding bounding box in usual cases. This paper proposes a method to construct the convex bounding polyhedron from a point set. Based on the coarsely approximate convex hull, the algorithm generates k normals by k -means algorithm. The algorithm searches the tangent point to generate a cutting plane along each normal on GPU and then constructs the convex bounding polyhedron by intersection of the cutting planes. Experiments illustrate

that the algorithm can construct tighter convex bounding polyhedron efficiently and this advantage accelerates collision detection algorithms.

Keywords convex bounding polyhedron, approximate convex hull, bounding volume, parallel computing, collision detection

TANG Lei was born in 1989. He received B.S. degree in software engineering from Central South University, Changsha, in 2012. Currently, he is an M.S. candidate in Tsinghua University. His research interests include computer graphics, computer aided design, data mining and software engineering.

SHI KanLe was born in 1985. He received Ph.D. degree in the Department of Computer Science and Technology and also in the School of Software, Tsinghua University, Beijing, in 2011. Currently he is an assistant professor at School of Software, Tsinghua University. His research interests are geometric modeling, computer-aided design and computer graphics.

YONG JunHai was born in 1973. He received Ph.D. degree in computer science from Tsinghua University, Beijing, in 2000. Currently he is a professor at the School of Software at Tsinghua University. His research interests include computer graphics and computer aided design, computer animation and software engineering.

GU HeJin was born in 1964. He received the Ph.D. degree in industrial economics from Wuhan University of Technology, Wuhan, in 2009. Currently he is a senior professor at Jiangxi Academy of Sciences. His research interests include computer graphics, computer aided design and software engineering.