凸包围多面体生成算法及应用

(申请清华大学工学硕士学位论文答辩报告)

学　　生：唐　　磊

指导教师：雍 俊 海 教授

计算机辅助设计图形学与可视化研究所

二〇一五年六月

目录

引言

凸包围体技术

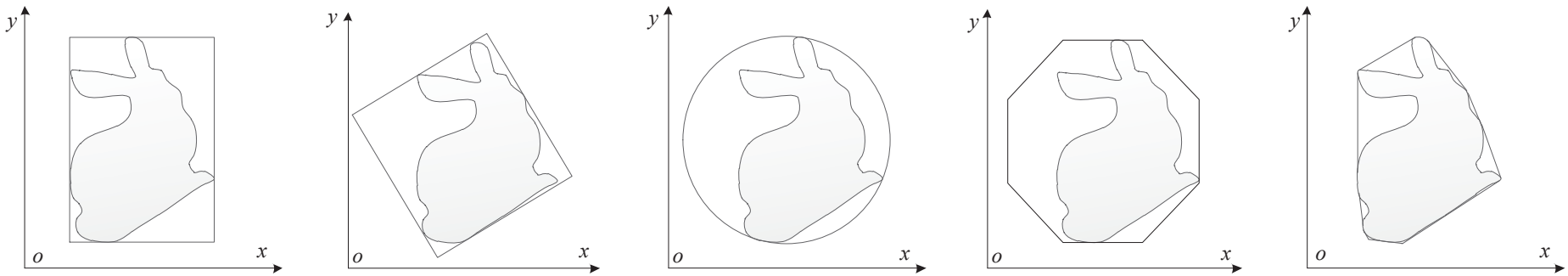
在计算机图形学领域里的各种算法中发挥着重要作用，如优化渲染和建模过程，加速求交、碰撞检测等算法。

碰撞检测问题

计算机图形学、虚拟现实等领域中的研究热点，是计算机模拟真实环境中不可或缺的技术，在物理仿真及游戏领域里应用十分广泛。

目录

凸包围体的种类



(a) AABB (b) OBB (c) Sphere (d) k-DOP (e) 凸包

Figure 1: 不同种类的包围体

其他：Tribox、Swept-sphere、Sphere-shell、Zonotopes、圆柱形、圆锥、椭球形等等。

本文目标

综合比较

k-DOP[1]: 方向固定且有限，不同模型其截面方向一致，不够紧致。

凸包: 很 (最) 紧致，但面片数量太多，复杂度 $O(n \log n)$。

本文凸包围体的目标

紧致: 能够自适应模型，根据模型形状特点有不同的方向；

快速: 生成凸包围体的速度要快，利用 GPU 加速；

灵活: 通过参数 k 调节凸包围体的简单性和紧致程度。

目录

碰撞检测算法

碰撞检测算法

许多应用的基础，例如在 3D 游戏，物理仿真，机器人，虚拟现实等领域中 [2]。

分类

加速结构： SPT（如四叉树、KD 树等） v.s BVH（OBB 树、k-DOP 树等）

表现形式： **刚体** v.s 可变形，凸体 v.s 凹体，CSG v.s 参数曲面 v.s **多边形网格**

碰撞环境： **成对** v.s **多体**，**静止** v.s **运动**，**离散** v.s 连续

基于 BVH 的碰撞检测算法

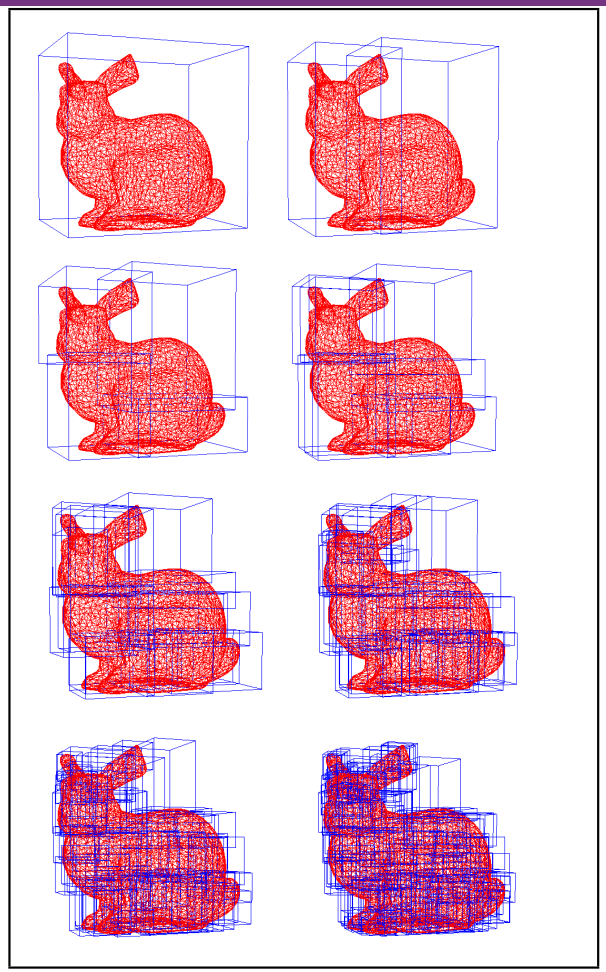


Figure 2: 八层 BVH 示例

算法 1 自顶向下层次遍历 BVH

输入: 两个 BVH 树的根节点 $node_1$，$node_2$
输出: 模型是否相交

```
1: function TraverseBVHTree(node1, node2)
2:     if node1.bv ∩ node2.bv = ∅ then
3:         return False  // 包围体重合测试, 包围体不相交直接返回
4:     else
5:         if node1.children = ∅ then
6:             if node2.children = ∅ then
7:                 // 最底层叶子节点原生几何相交测试
8:                 return CheckIntersection(node1.primitives, node2.primitives)
9:             else
10:                for all child ∈ node2.children do
11:                    TraverseBVHTree(node1, child)  // 递归调用
12:                end for
13:            end if
14:        else
15:            for all child ∈ node1.children do
16:                TraverseBVHTree(child, node2)  // 递归调用
17:            end for
18:        end if
19:    end if
20: end function
```

代价函数：$T_{cost} = n_v * C_v + n_p * C_p + (n_u * C_u)$(运动)

目录

问题的定义

凸包围 k 面体

k-Convex Bounding Polyhedron，简称 k-CBP，可通过 k 个半空间定义：

$$\begin{cases} \text{k-CBP} = \bigcap_{i=1}^{k} H_i \\ H_i = \left\{ p \in \mathbb{R}^3 \middle| n_i \cdot p \leq w_i, w_i \in \mathbb{R} \right\}, \end{cases} \tag{1}$$

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。

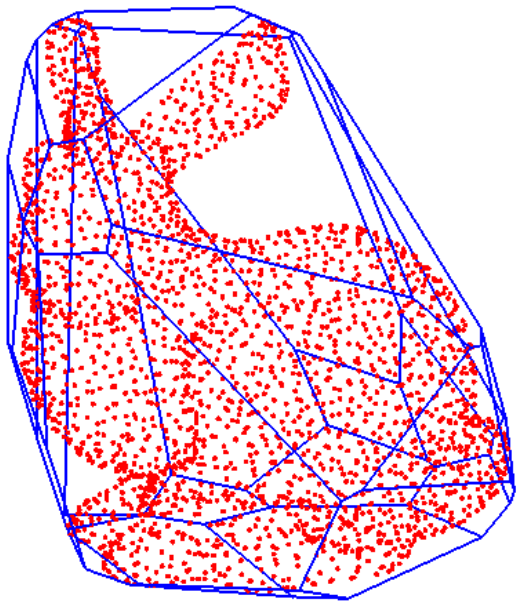
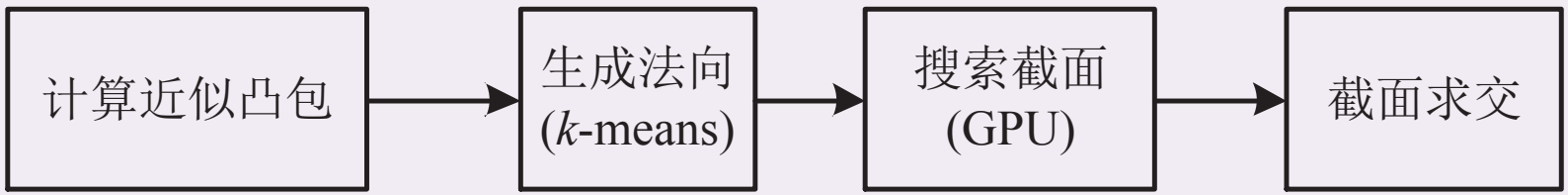


Figure 3: 34-CBP

算法流程

构造 k-CBP 算法流程图



关键步骤

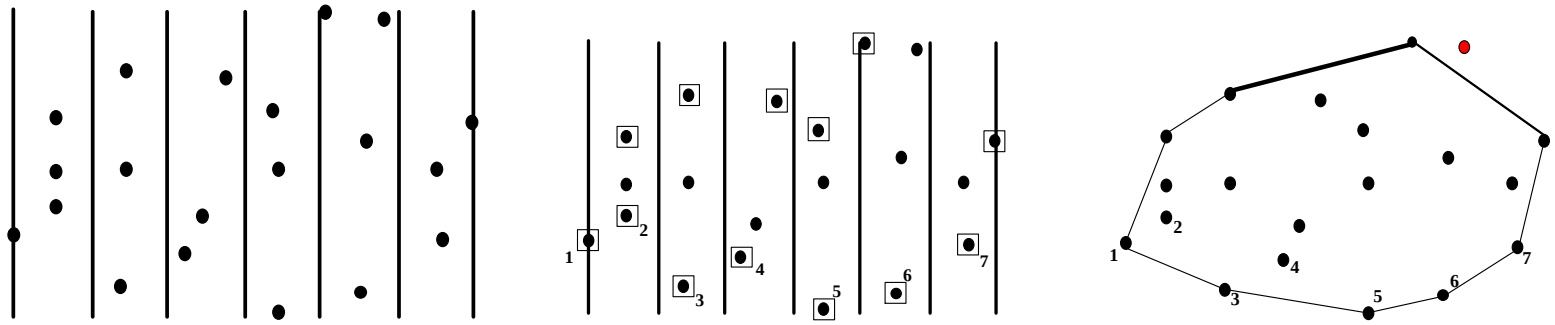
定法向： 结合近似内凸包和 k-means；

搜截面： GPU 中沿各法向搜索切点构造截面；

求交点： 截面对偶映射求得交点。

目录

近似凸包的构造

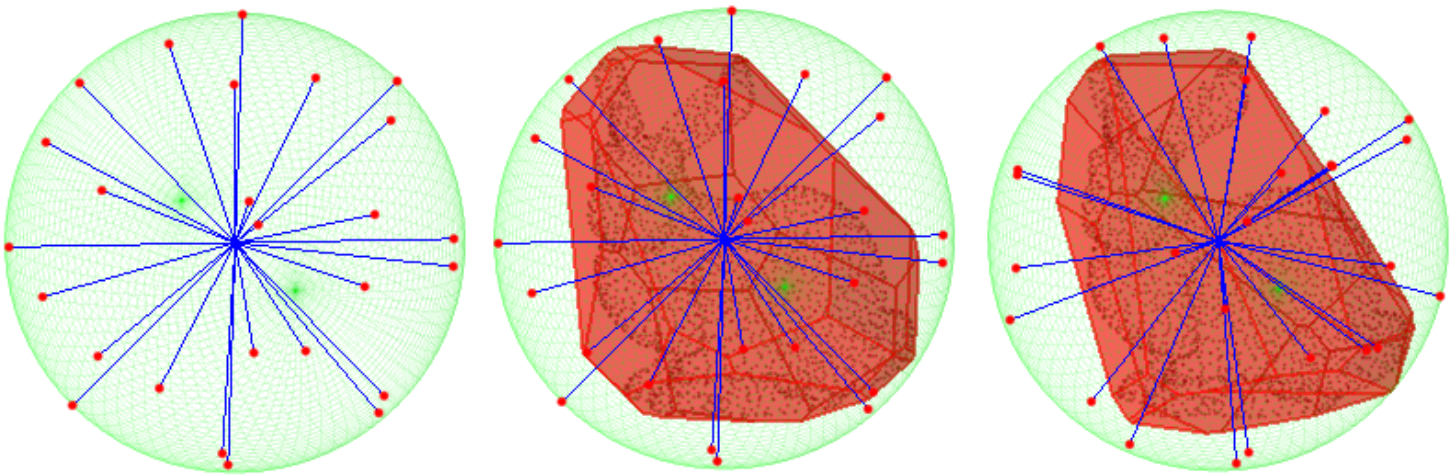


(a) 分组 (b) 选极值点 (c) 连线

Figure 4: 二维近似内凸包的构造

构造近似内凸包 [3]，算法复杂度为 $O(n+\xi)$，扩展到三维为 $O(n+\xi^2 \log \xi)$，然后利用 k-means 聚类。

k-means 聚类



(a) (b) (c)

Figure 5: 通过聚类确定法向

初始方向： 均匀分布；

聚类度量： 余弦；

中心更新： 中心点：$c_i = \frac{\sum_{i=1}^{i=n} \omega_i \cdot n_i}{\sum_{i=1}^{i=n} \omega_i}$，权重 ω_i 为面片面积。

目录

搜索截面

截面＝法向 + 点

法向已得，求投影点：对每个法向 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。时间复杂度为 $O(k \cdot n)$，其中 k 为法向数量，n 为模型所含点数。

并行可行性

各法向的计算相互独立，借助 GPU 并行加速。典型 GPU 并行平台：着色器（GLSL 为例）和基于 GPU 的通用计算框架（CUDA 为例）。

GLSL 实现

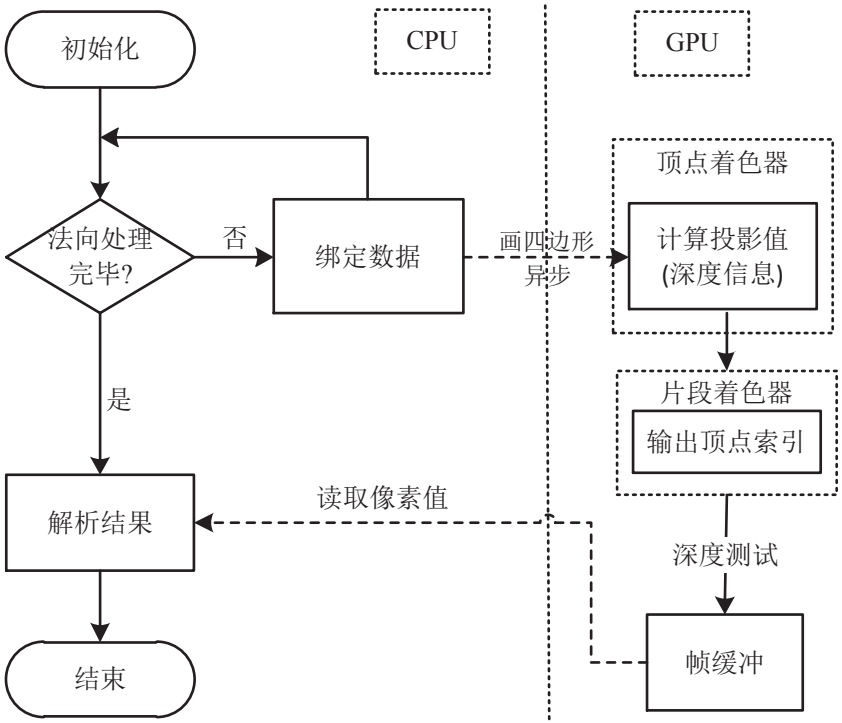


Figure 6: 基于 Z Buffer 算法流程图

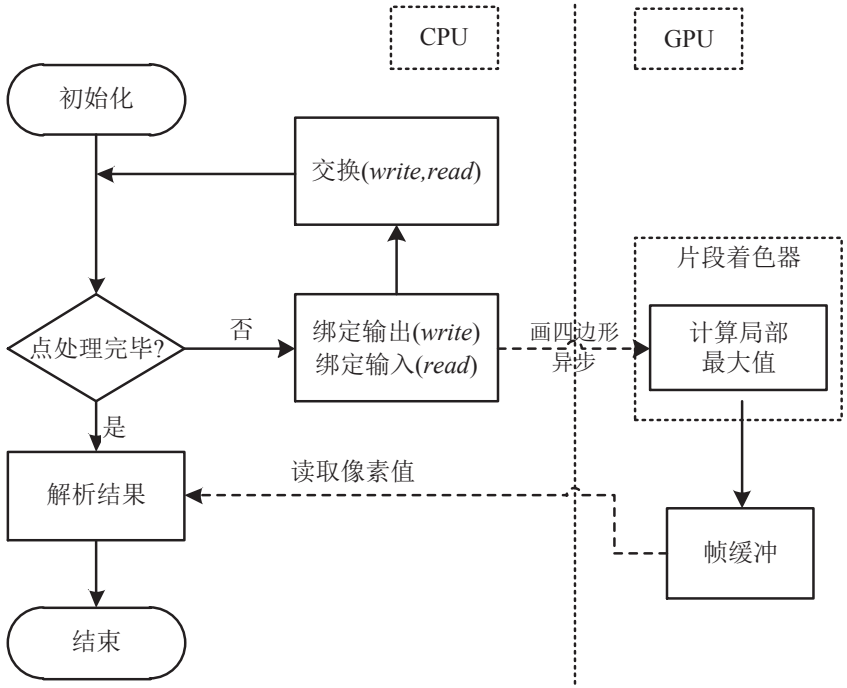


Figure 7: 基于乒乓技术算法流程图

CUDA 实现

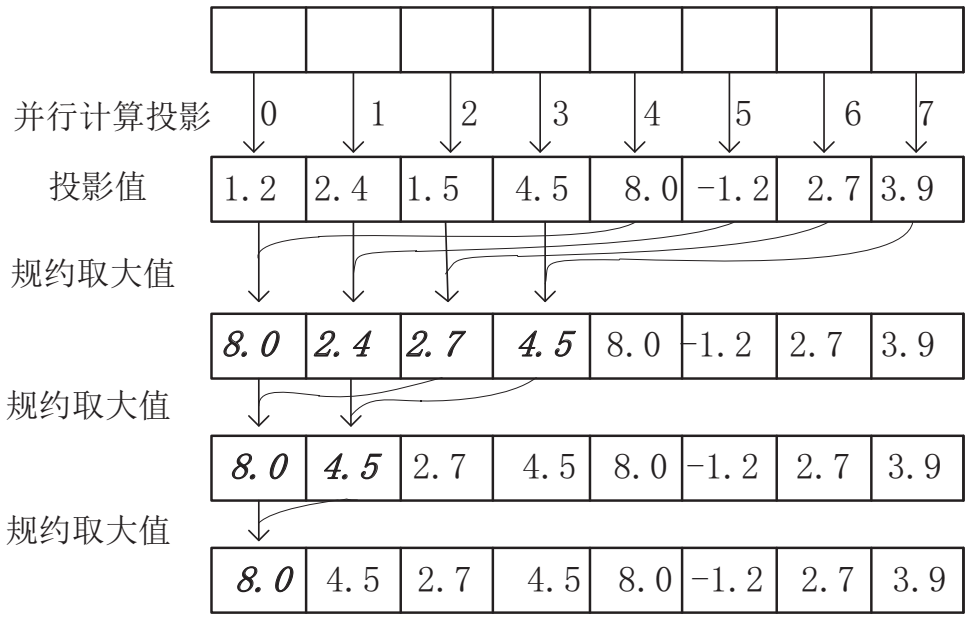


Figure 8: 并行规约求最大投影值

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。OpenCL 等并行计算框架类似。

求交算法

直接枚举

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，时间复杂度为 $O(k^3)$。

对偶映射

法向 $n(a, b, c)$ + 平面上点 $p(x_0, y_0, z_0) \Rightarrow$ 平面方程 $ax + by + cz = ax_0 + by_0 + cz_0 = d,\ d \neq 0$
对偶点为 $p'(a/d, b/d, c/d)$，对 k 个映射点求凸包，凸包平面映射回原来的交点，时间复杂度为 $O(k \log k)$[4]。

目录

生成 k-CBP 的效率：GLSL 实验结果

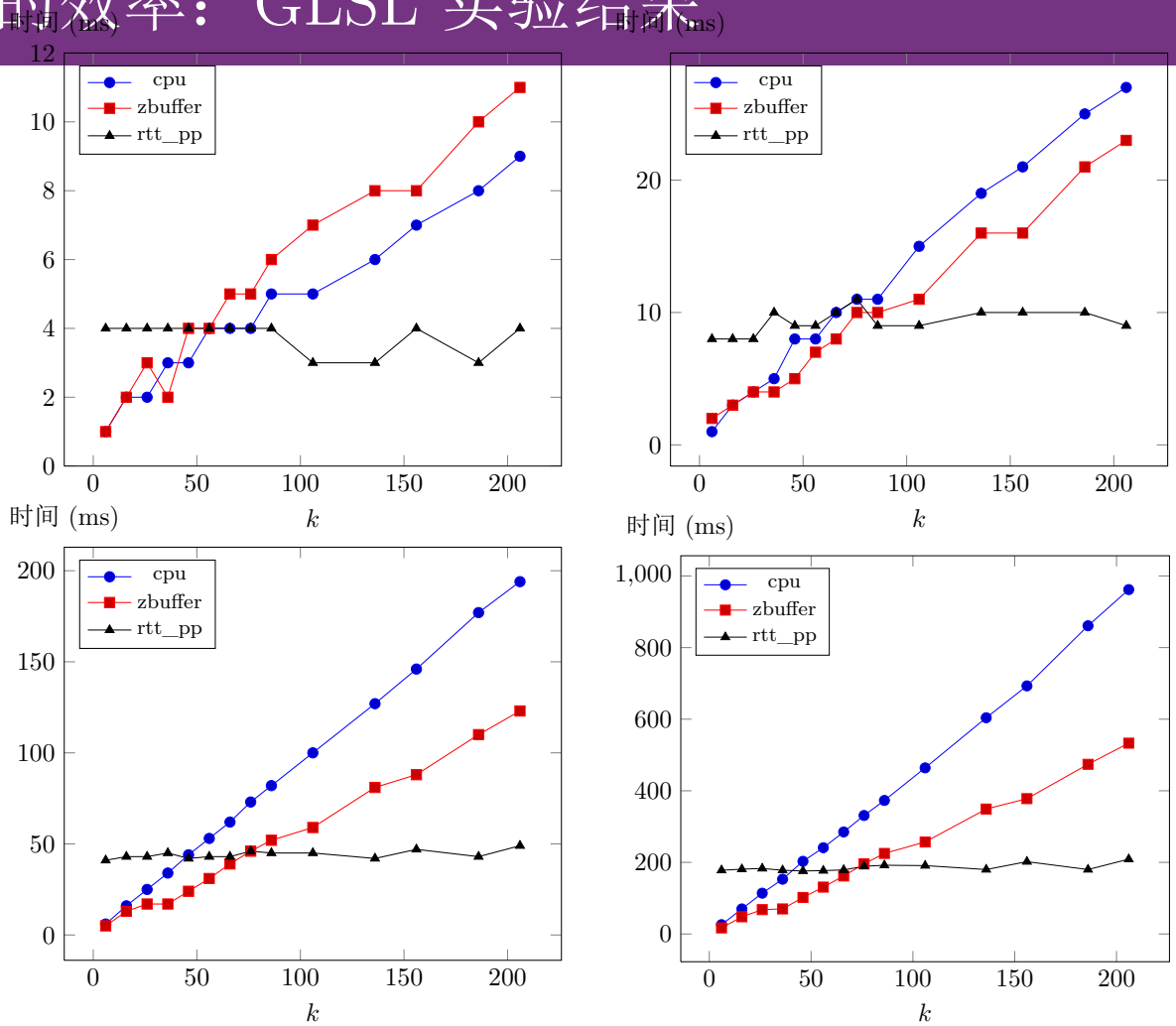


Figure 9: Apple(8k), Buddha(31k), Alice(224k), Bugatti(1011k)

生成 k-CBP 的效率：CUDA 实验结果

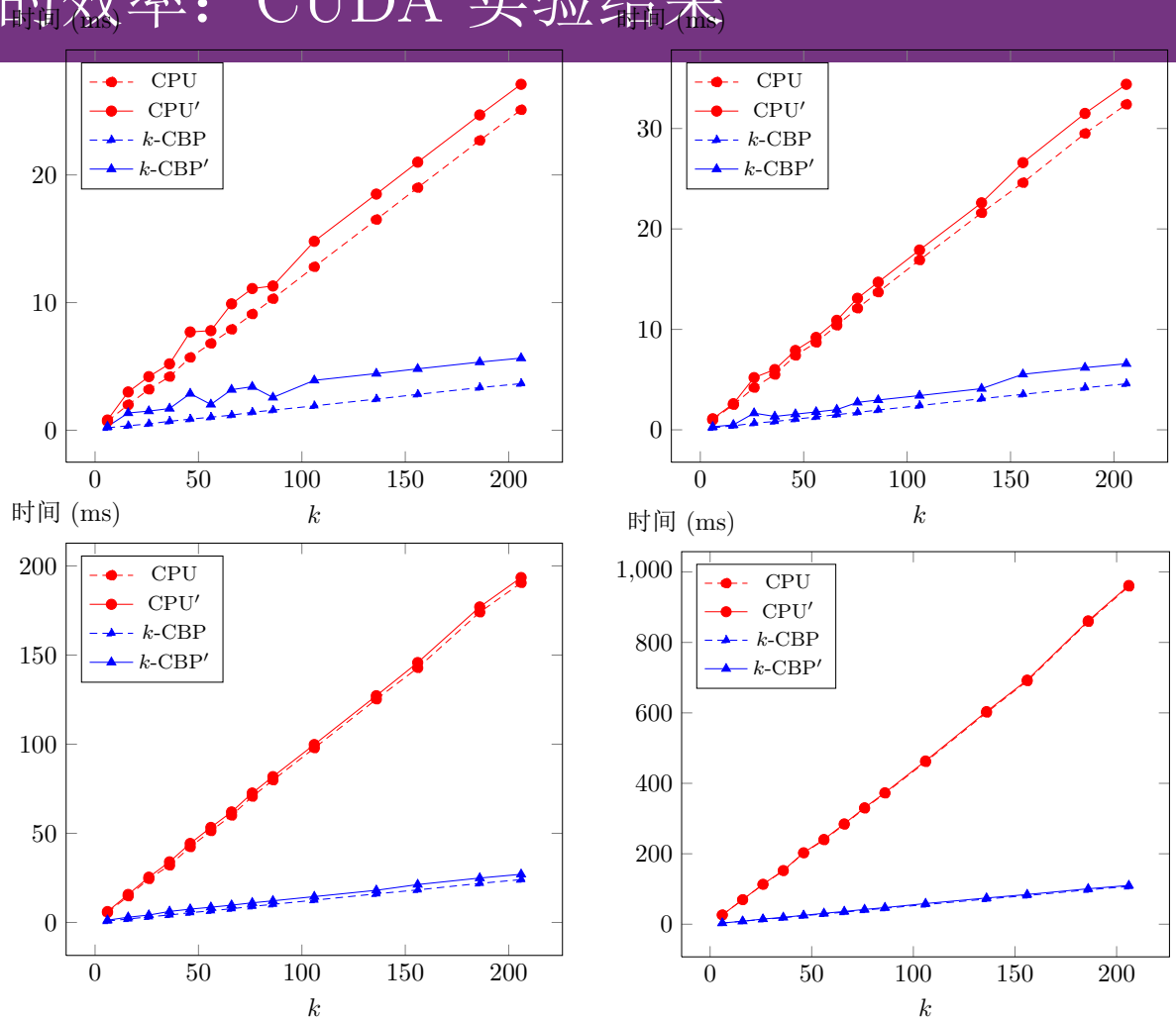


Figure 10: Budda(31k),Dinosaur(40k),Alice(224k), Bugatti(1011k)

生成 k-CBP 的效率：与文献 [5] 算法对比

Table 1: 本文算法与文献 [5] 算法对比

k	Apple(8118 points)			Bugatti(1010815 points)		
	SSE(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

当点数量较小时，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。

生成 k-CBP 的紧致程度：k-DOP v.s k-CBP

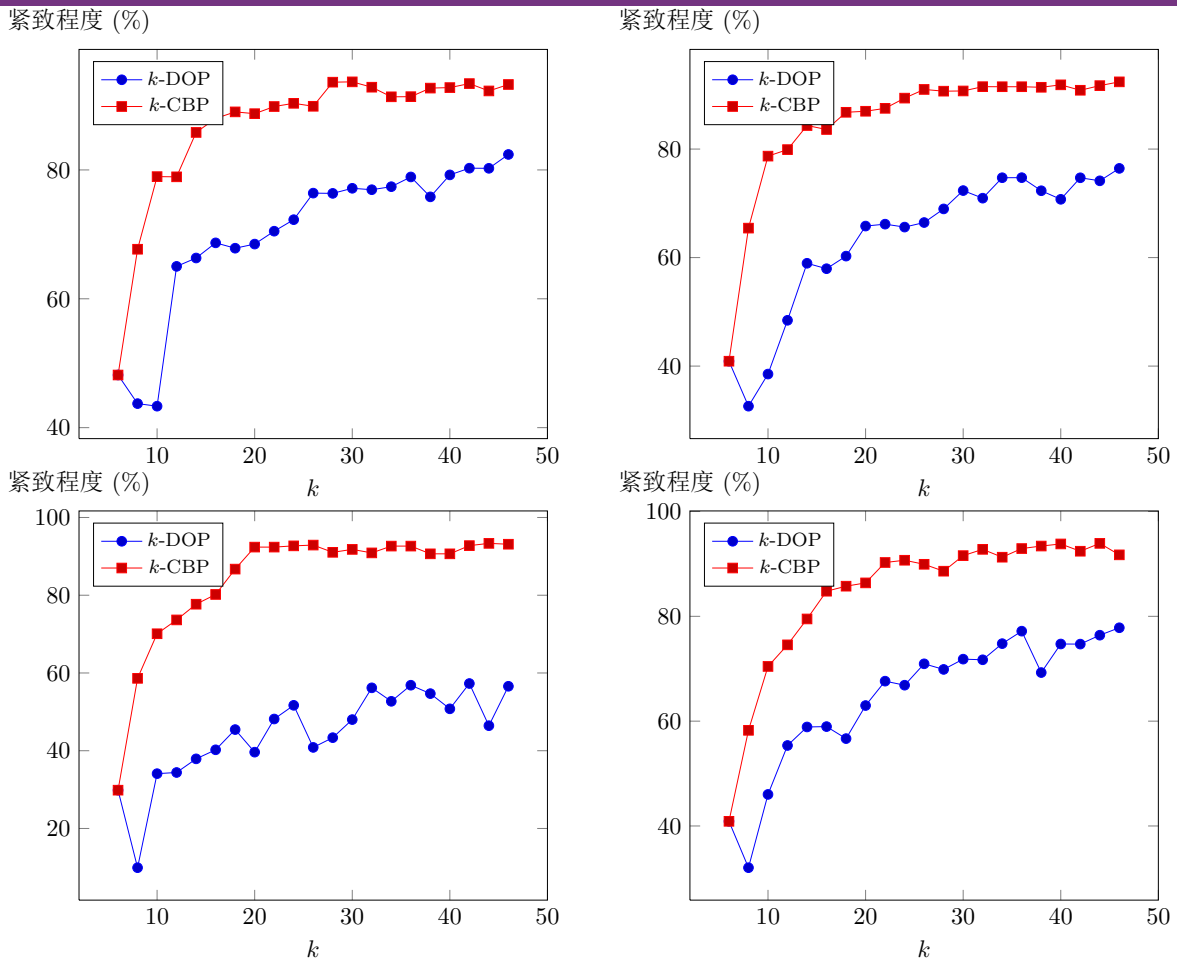


Figure 11: 紧致程度对比：Apple(8k), Budda(31k), Dinosaur(40k), Alice(224k)

生成 k-CBP 的紧致程度：k-DOP v.s k-CBP

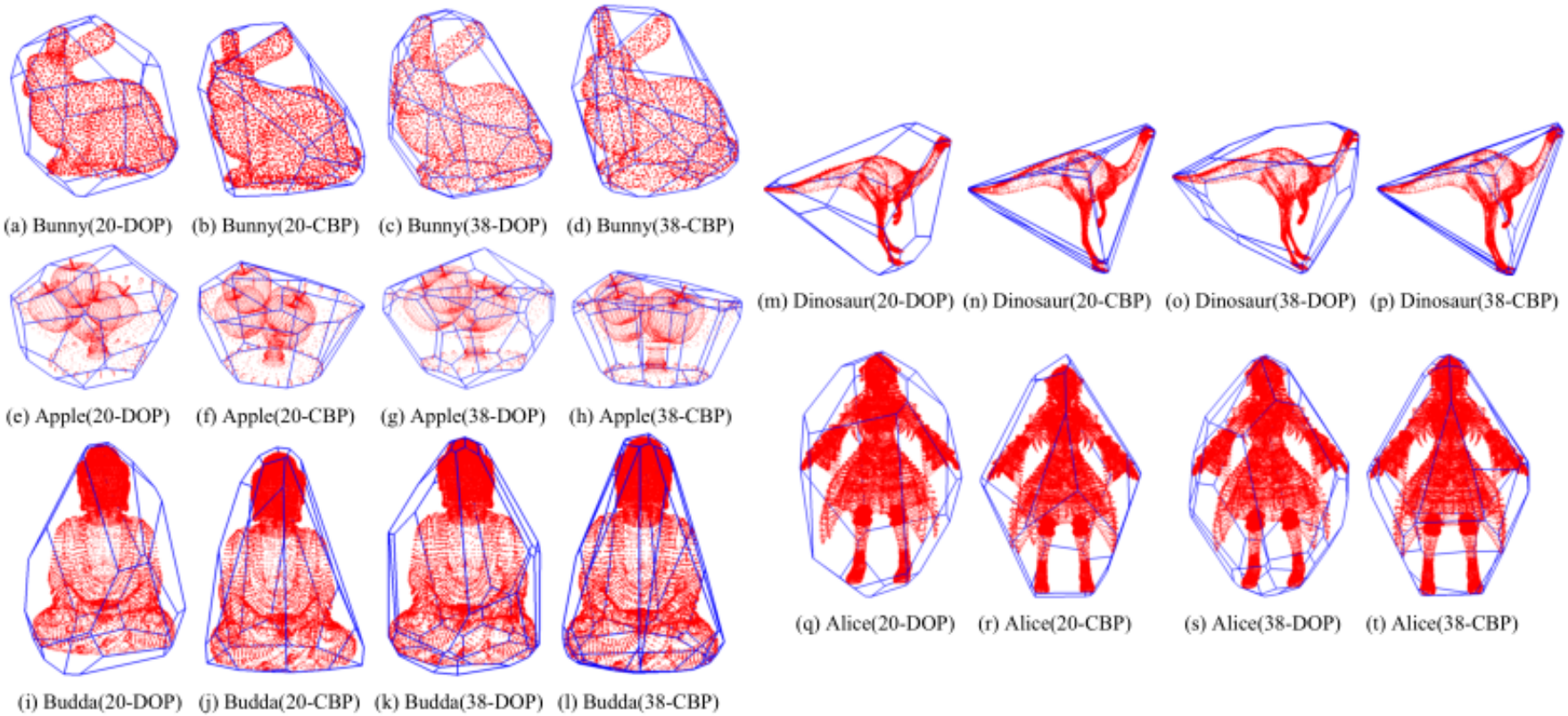


Figure 12: k-CBP 与 k-DOP 对比

生成 k-CBP 的紧致程度：k-DOP v.s 凸包

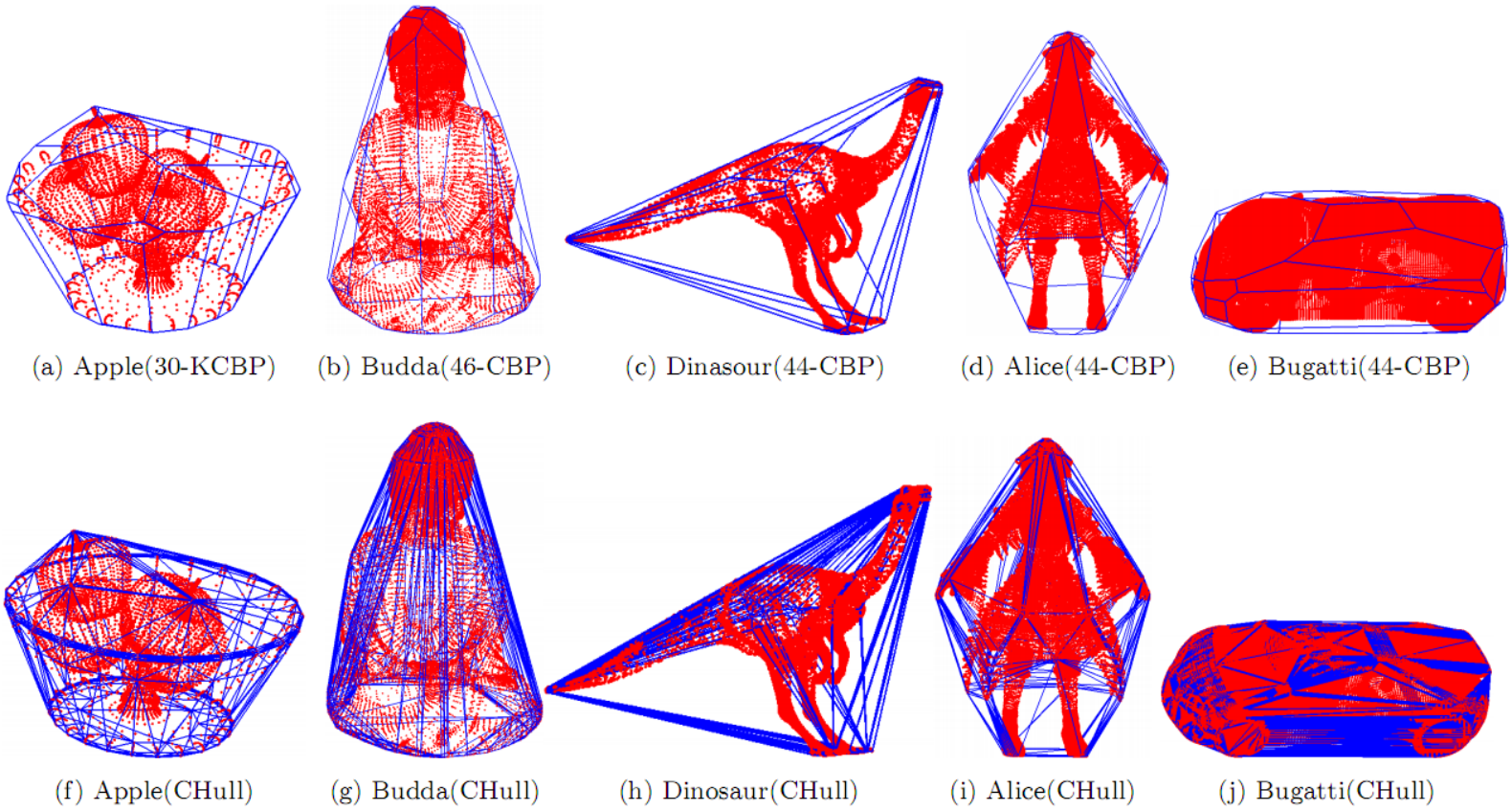
Table 2: k-CBP 与 QuickHull 凸包算法比较

Model	f(CHull)	f(k-CBP)	τ (k-CBP)	t(CHull(ms))	t(k-CBP(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	24654	44	95.06%	688.7	25.41

结论

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（Bugatti 凸包面的 0.17% 的达到 95.06% 紧致程度，构造速度快 27 倍），下图为可视化结果。

生成 k-CBP 的紧致程度：k-DOP v.s 凸包



(a) Apple(30-KCBP) (b) Budda(46-CBP) (c) Dinasour(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)

Figure 13: k-CBP 与凸包对比

目录

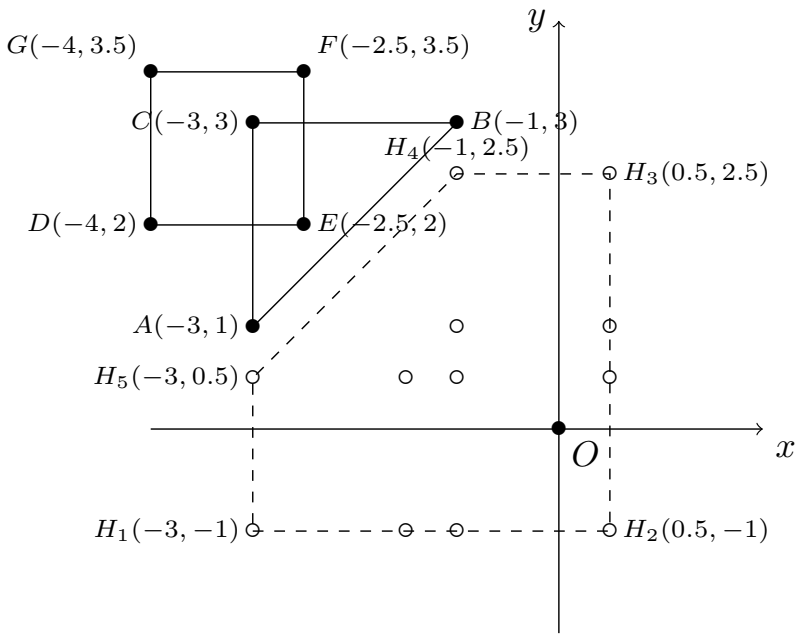
AABB 树法

将生成的 k-CBP 视为普通的三角形，实现简单，适用于模型较小的静止碰撞检测场景。

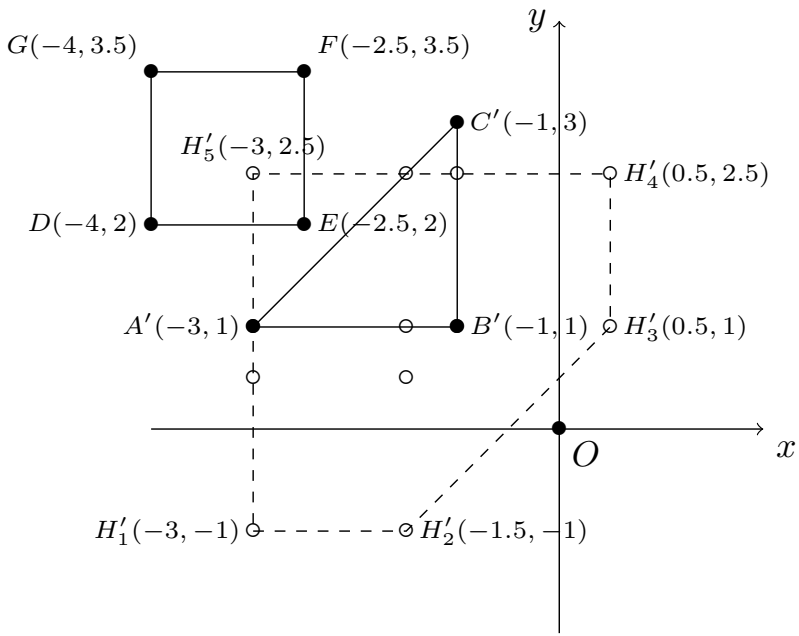
GJK 法

计算凸多面体之间的最近距离的 GJK 算法 [6]。
Minkowski 差，即 $\mathbb{A} - \mathbb{B} = \{a - b | a \in \mathbb{A}, b \in \mathbb{B}\}$。GJK 算法的核心基础在于若两个凸多边相交，则凸多边形顶点
的 Minkowski 差所围成的多边形必包含原点，因为若 $\mathbb{A}$ 和 $\mathbb{B}$ 相交即 $\mathbb{A}$ 和 $\mathbb{B}$ 必含有公共交集，即至少含有一点同时属于 $\mathbb{A}$ 和 $\mathbb{B}$ ，该点的 Minkowski 差即为原点 $O(0, 0)$。

二维 GJK 算法示例

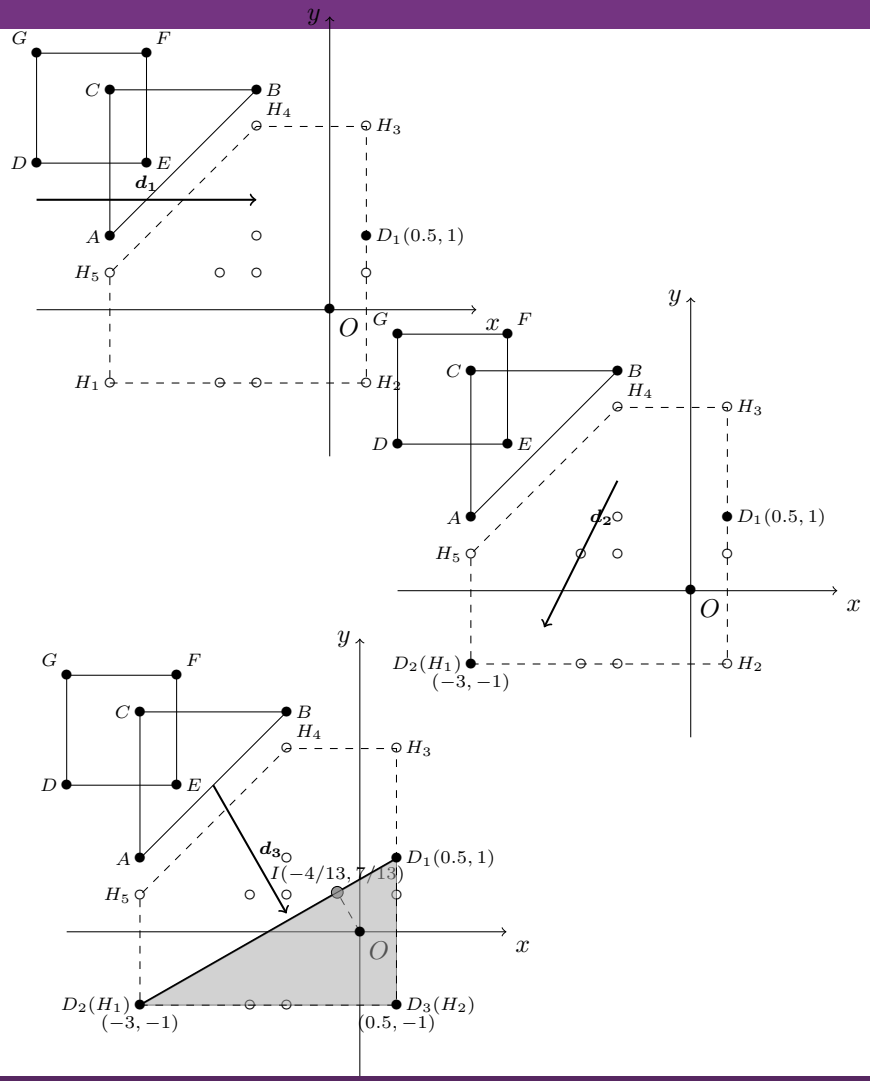


(a) 相交



(b) 不相交

GJK 算法



算法 2 基于 GJK 的 k-CBP 相交检测算法

输入: 两个 k-CBP $\text{k-CBP}_1, \text{k-CBP}_2$
输出: k-CBP 是否相交

```
function KCBPDetectionBasedOnGJK(k-CBP1, k-CBP2)
    d ← initNormal()
    D ← Support(k-CBP1, k-CBP2, d)
    S ← {p}
    iter ← 1, d ← −d
    while inter++ < MaxIter do
        D ← Support(k-CBP1, k-CBP2, d)
        if D · d < 0 then
            return False
        end if
        S ← S ∪ D
        contains ← CheckContainUpdate(S, d) // 检测是否包含原点，对集合 S 进行规约，并获取下一次迭代的方向 d
        if contains then
            return True // 包含原点，直接返回相交，否则继续迭代
        end if
    end while
    return False // 达到最大迭代次数，根据需求返回相交或者不相交
end function
```

目录

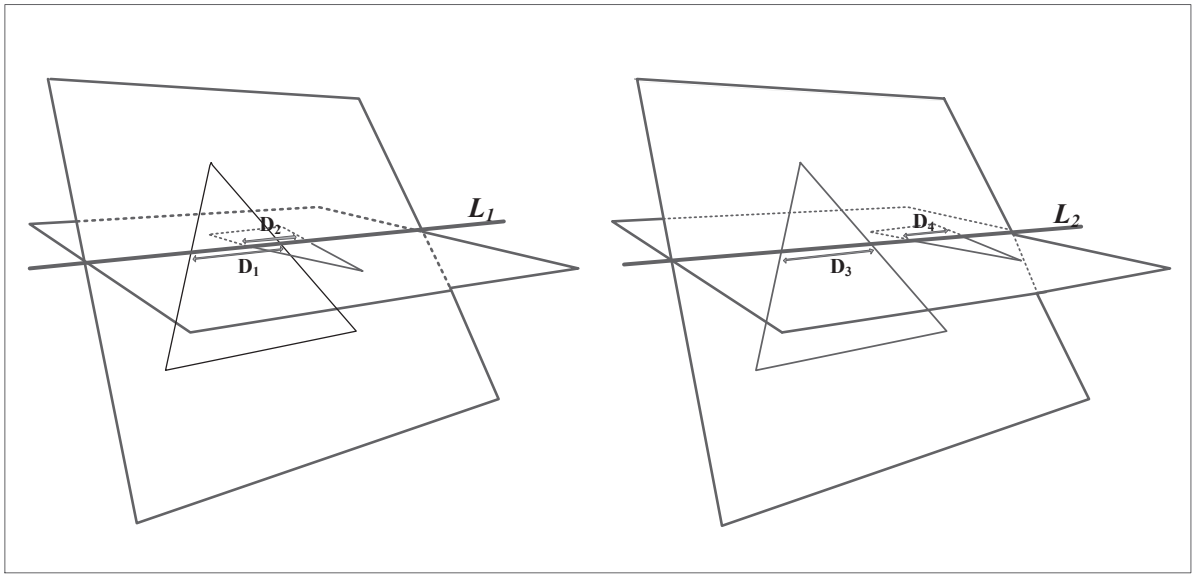


Figure 14: 两个非共面三角形的位置关系

三角形 T_1, T_2 坐标 $\Rightarrow$ 平面方程 $\Pi_1, \Pi_2 \Rightarrow$ T_2 到 Π_1 的有向距离 $l_{1i}, i \in \{1,2,3\}$[7]:

(1) 若 $\forall i \in \{1,2,3\}, l_{1i} = 0$，即三角形 T_2 的三个顶点到三角形 T_1 所在 Π_1 的距离都为 0，则两个三角形共面；$\Rightarrow$ 共面三角形求交。

(2) 若 $\forall i \in \{1,2,3\}, l_{1i} > 0$ 或 $\forall i \in \{1,2,3\}, l_{1i} < 0$，即三角形 T_2 的三个顶点到三角形 T_1 所在 Π_1 的有向距离同号，则 T_2 在 Π_1 的同一侧，可立即排除相交；

(3) 其他情况，三角形 T_2 必交 Π_1 于一条线段。$\Rightarrow$ 判断两个区间线段 D_1, D_2 是否相交。

目录

k-CBP 的有效性

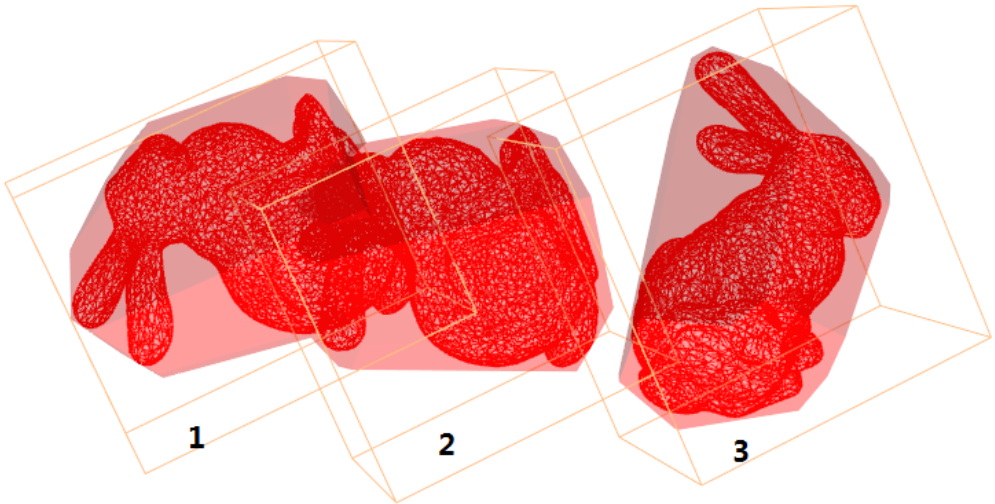


Figure 15: k-CBP 应用于碰撞检测示例

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.

静止场景碰撞检测算法

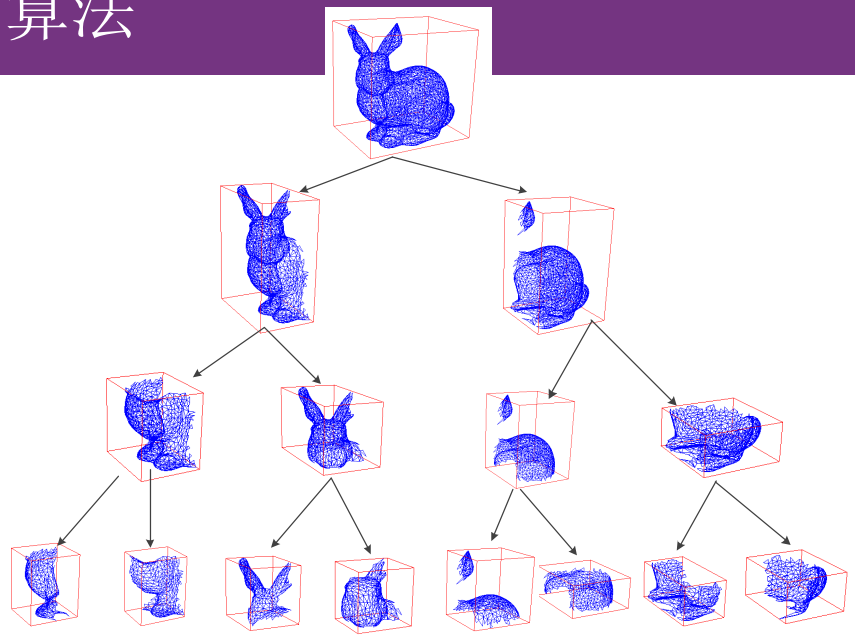


Figure 16: Bunny 模型的 AABB 树形结构 (部分) ▸ 动态图

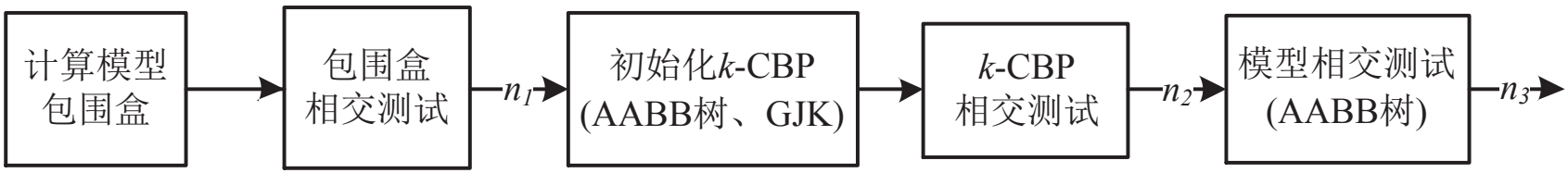


Figure 17: 基于 k-CBP 的碰撞检测算法流程图

运动场景碰撞检测算法

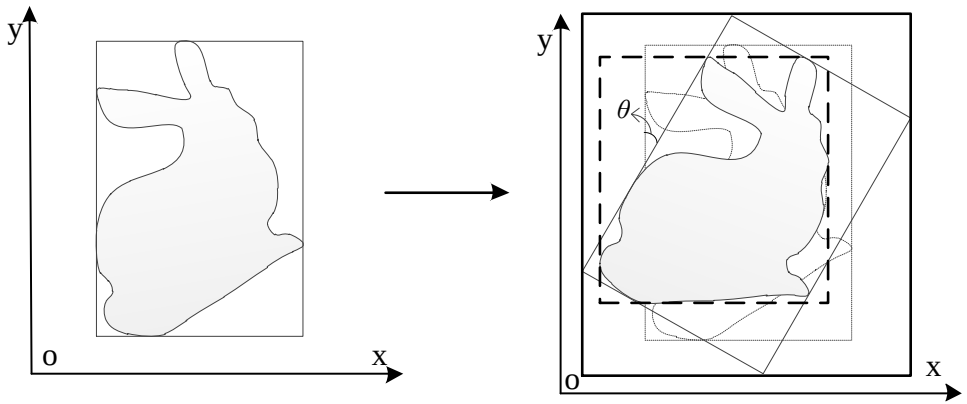


Figure 18: AABB 更新策略图

将变换矩阵
$M = R(n, \theta) \cdot T(t)$
应用于 GJK 顶点、AABB 顶点。

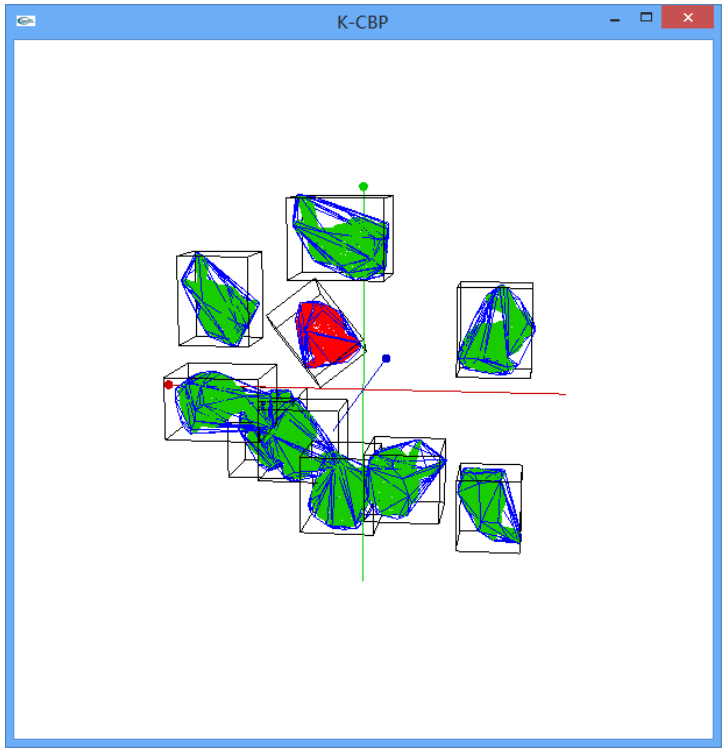


Figure 19: 运动场景碰撞检测示例 ▸ 动态图

目录

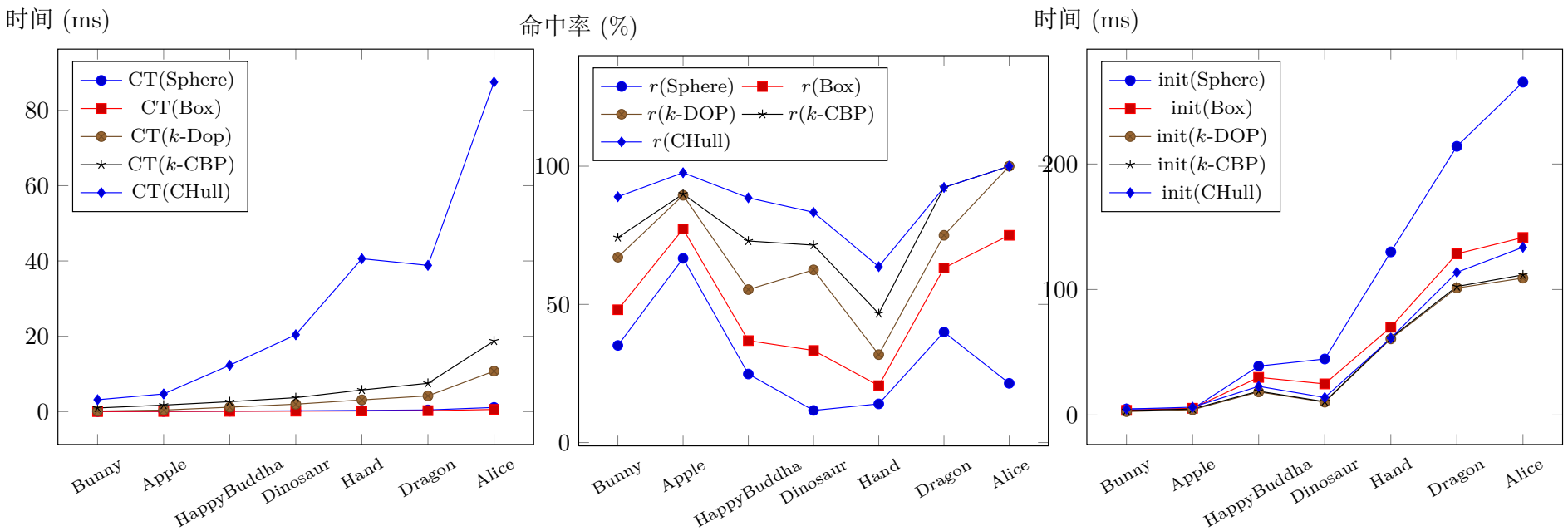
实验结果：k-CBP 用于碰撞检测的有效性

Table 3: k-CBP 和包围盒应用于碰撞检测结果对比

n	CT(Box) (ms)	CT(16-CBP) (ms)	DT(Box) (ms)	DT(16-CBP) (ms)	r(Box) (%)	r(k-CBP) (%)	DP(Model) (对)
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

其中模型和凸包围多面体是否相交都采用了 AABB 树的方式进行判断。

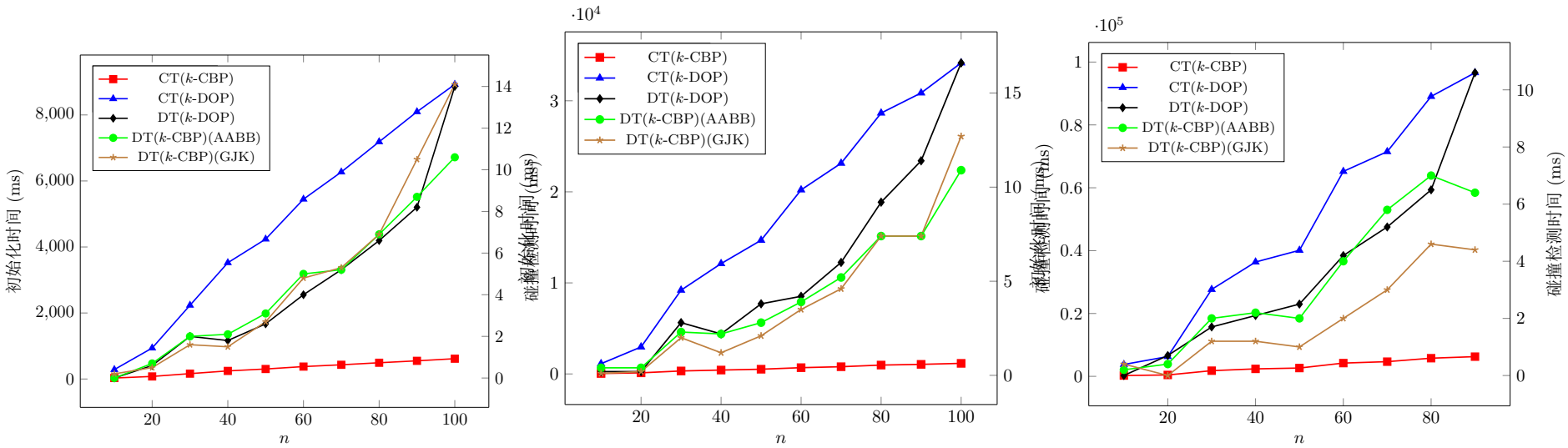
实验结果：不同包围体对比



(a) 包围体构造时间　　(b) 包围体命中率　　(c) 初始化时间

构造时间上基本满足：凸包 > k-CBP > k-DOP > Sphere ≈ Box，包围体命中率基本满足：凸包 > k-CBP > k-DOP > Box > Sphere。紧致程度和包围体的命中率满足正相关关系。

实验结果：静止场景与 k-DOP 树对比

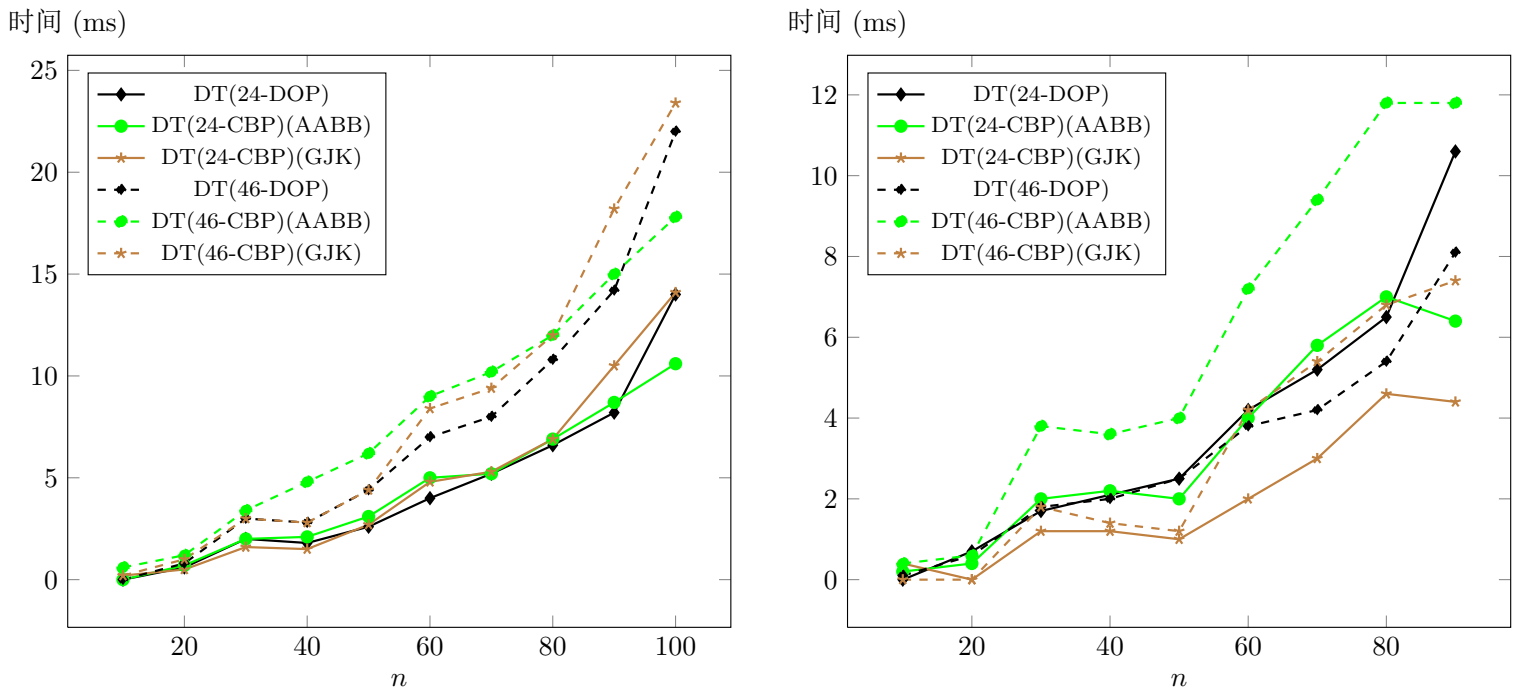


(d) Bunny(4968 个三角形) (e) Apple(8040 个三角形) (f) HappyBuddha(50000 个三角形)

Figure 20: 静止场景下本文算法与基于 k-DOP 树算法实验结果对比（k = 24）

基于 k-DOP 树 [8] 算法的碰撞检测库 CollDet 是 Gabriel Zachmann 等人实现的。

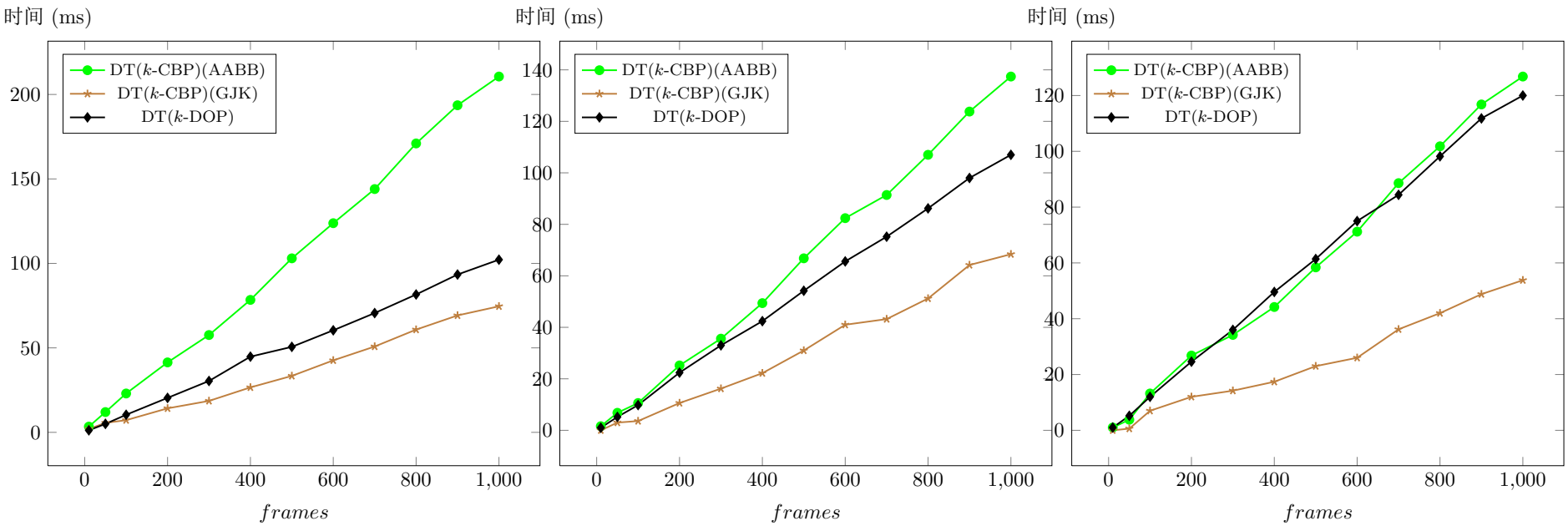
实验结果：静止场景与 k-DOP 树对比



(a) Bunny(4968 个三角形) (b) HappyBuddha(50000 个三角形)

Figure 21: 静止场景下不同 k 值实验结果对比

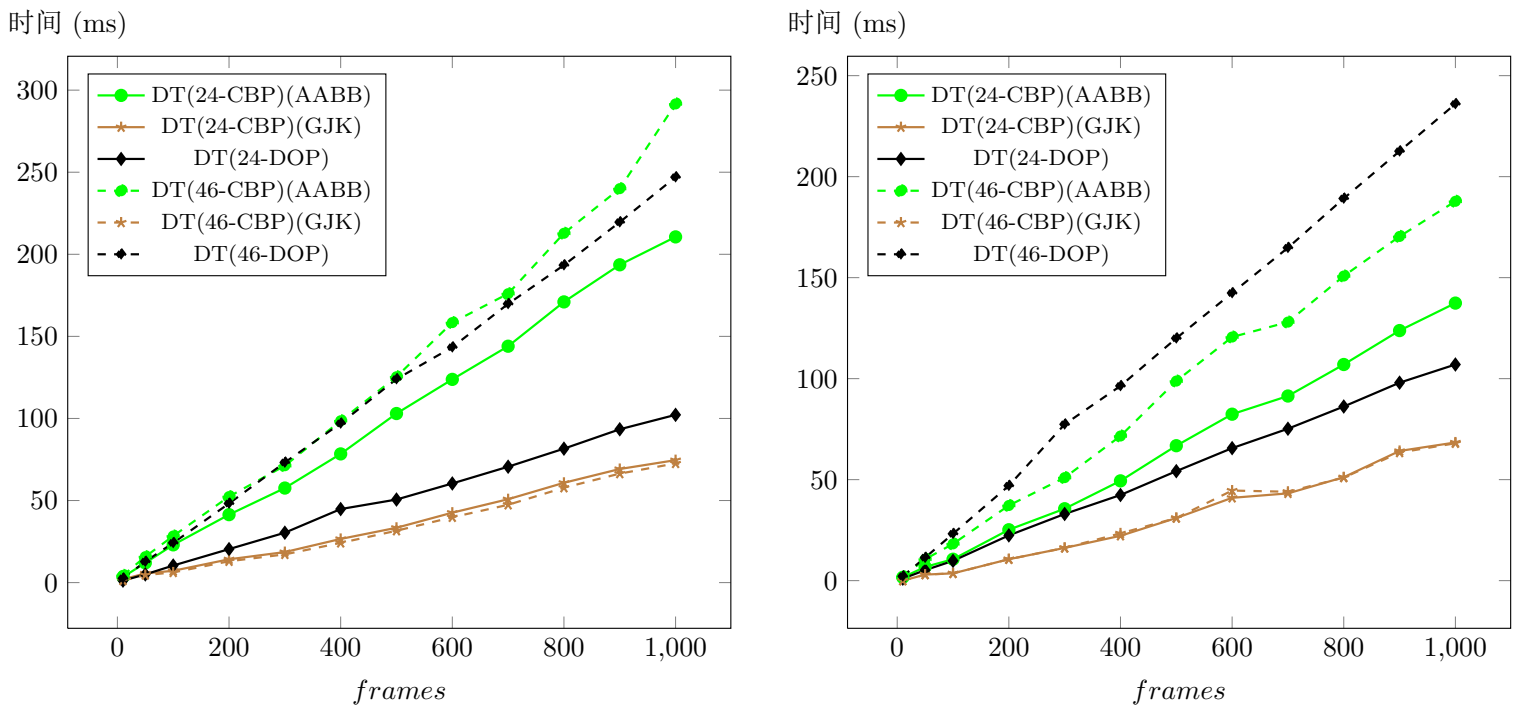
实验结果：运动场景与 k-DOP 树对比



(a) Bunny(4968 个三角形) (b) HappyBuddha(50000 个三角形) (c) Hand(128314 个三角形)

Figure 22: 运动场景下本文算法与基于 k-DOP 树算法实验结果对比（k = 24, n = 10）

实验结果：运动场景与 k-DOP 树对比



(a) Bunny(4968 个三角形)　　(b) HappyBuddha(50000 个三角形)

Figure 23: 运动场景下不同 k 值实验结果对比

总结与展望

总结

(1) 提出了一种构造紧致凸包围多面体–k-CBP 的算法；

(2) 构造 k-CBP 速度上比现有算法快 3~8 倍；

(3) 构造的 k-CBP 紧致程度比现有的 k-DOP 紧致 10% ~ 40%；

(4) 提出了一种基于 k-CBP 的碰撞检测算法，该算法较 k-DOP 树算法初始化时间快 8 倍以上，静止场景快 0.8 ~ 3.2 倍，运动场景快 0.8 ~ 5.6 倍。

展望

(1) 碰撞检测算法如何摆脱对 AABB 树的依赖；应用于近似碰撞检测算法；应用于可变形的模型连续碰撞检测，如何快速更新 k-CBP ；

(2) 如何将 k-CBP 应用于如机器人抓取、路径规划等其他应用领域中；

主要参考文献 I

[1] James T Klosowski et al. “Efficient collision detection using bounding volume hierarchies of k-DOPs”. In: IEEE Transactions on Visualization and Computer Graphics 4.1 (1998), pp. 21–36.

[2] Christer Ericson. Real-time collision detection. San Francisco, CA: Morgan Kaufmann Publishers, 2005.

[3] Jon Louis Bentley, Franco P Preparata, and Mark G Faust. “Approximation algorithms for convex hulls”. In: Communications of the ACM 25.1 (1982), pp. 64–68.

[4] 邓俊辉. 计算几何-算法与应用. 北京: 清华大学出版社, 2005.

[5] Mattias Karlsson, Olov Winberg, and Thomas Larsson. “Parallel Construction of Bounding Volumes”. In: The Annual Swedish Computer Graphics Association Conference(SIGRAD). 2010, pp. 65–69.

主要参考文献 II

[6] Gino van den Bergen. “A fast and robust GJK implementation for collision detection of convex objects”. In: Journal of Graphics Tools 4.2 (1999), pp. 7–25.

[7] Tomas Moller. “A Fast Triangle-Triangle Intersection Test”. In: Journal of Graphics Tools 2.2 (1997), pp. 25–30. issn: 0346718X. doi: 10.1080/10867651.1997.10487472.

[8] Sven Trenkel, René Weller, and Gabriel Zachmann. “A benchmarking suite for static collision detection algorithms”. In: International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG). Plzen, Czech Republic: Union Agency, 2007.

感谢

致谢

(1) 导师雍俊海老师的精心指导；
(2) 施侃乐老师帮助；
(3) 研究所各种项目的历练；
(4) 王斌老师、陈莉老师的评审及意见，答辩委员会老师们的指导。

Q & A

Questions?

Thank you!