2022 年度　卒　業　論　文

回転剛体の衝突判定精度向上に関する研究

指導教員：渡辺 大地 教授

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0119310
山本 輝

2022 年 7 月

2022 年度　　卒　業　論　文　概　要

論文題目

回転剛体の衝突判定精度向上に関する研究

メディア学部 **学籍番号 : M0119310**	**氏名**	山本 輝	**指導教員**	渡辺 大地 教授

キーワード	リアルタイムグラフィックス、干渉判定、衝突判定、 扇形、境界ボリューム

今日, コンピュータゲームにおける剛体同士の衝突判定は, 判定にかかる処理時間が高速である境界ボリュームが利用されることが多い. しかし, コンピュータ内での剛体の移動が離散的であるため, 剛体の運動速度が非常に大きい場合は, 剛体同士が衝突せずにすり抜けてしまう場合がある.

この問題に対処するために様々な CCD(Continuous Collision Detection, 連続的衝突判定) 手法が考案され, 銃弾などの運動速度の大きい剛体を扱う FPS(FirstPersonShooting) ゲームなどで利用されてきた. 一例として Unity で採用されている Sweep-based CCD 手法と,Speculative CCD 手法があるが, これらの手法では回転運動の際に本来の運動軌跡によって起きる衝突の検知精度が低い.

本研究では, この問題を解決するために, 回転運動軌跡の近似図形である扇形の境界ボリュームを開発し, 利用することによって回転剛体の連続的衝突判定精度の向上を目的とした.

扇形境界ボリュームと Speculative CCD 手法の衝突判定精度を検証によって比較すると, 扇形境界ボリュームのほうが衝突判定精度が優れていることが判明した.

目 次

図 目 次

第 1 章

はじめに

1.1 背景と目的

今日のコンピュータゲームで実装されている剛体同士の衝突は, 様々な手法によって判定を行っている. その一つに, 境界ボリューム手法 [1] がある. 境界ボリュームは,AABB(Axis-Aligned Bounding Box, 軸並行境界ボックス)[2] や OBB(Oriented Bounding Box, 有向境界ボックス)[2], 球, カプセル形状 [3],k-DOPs[4] などがあり, その特徴は, 同一矩形同士の干渉判定が容易であり, かつ計算が高速であることにある. そのため,Unity などのゲームエンジンでも採用されている手法である.

また,Gilbert らにより提案された GJK アルゴリズム [5][6] は, 境界ボリュームよりも剛体の本来の干渉判定範囲に近い凸包矩形の干渉判定を高速に行うことができるため, コンピュータゲームや Bullet 等の物理エンジンライブラリなどに利用されてきた.

境界ボリューム手法はその高速性故に様々な手法の高速化に用いられてきた. コンピュータゲーム中には壁や床, プレイヤーキャラクターなどの様々な剛体が存在する. ただ, ゲーム空間内での剛体同士の距離が遠い場合など, 衝突する可能性の低い剛体同士は GJK アルゴリズム等の詳細な干渉判定を行う必要はない. そのため, 高速に干渉判定を行うことのできる境界ボリュームを利用して, 衝突可能性のない剛体同士は干渉判定を行わないようにする AABBtree[7] や OBBtree[8] などの手法が提案されてきた. これらの手法は,BVH(Bounding Volume Hierarchy)[4][9] とよばれ, レイトレーシングの高速化 [10] などにも用いられている.

境界ボリューム手法や GJK アルゴリズムのように, ある時間における剛体の位置をもとに干渉

判定を行う手法を DCD(Discrete Collision Detection, 離散的衝突判定) と呼ぶが, これらの手法では剛体同士がすり抜けてしまう場合がある. それは, 運動を行う剛体の運動速度が非常に大きい場合である. コンピュータゲーム内での剛体の運動は,1frame ごとに処理を行うため, 剛体の運動速度が大きい場合には本来衝突する予定の座標を通り過ぎてしまう可能性がある. そのため,DCD 手法では剛体同士の衝突を検知することができない場合がある.

剛体の運動速度が大きい場合には, 剛体の移動軌跡などから他剛体との衝突判定を行う手法があり,CCD(Continuous Collision Detection, 連続的衝突判定) と呼ぶ. コンピュータグラフィックスの分野の中でも特に医療シミュレーションやロボットシミュレーションなどの高精度なシミュレーションを必要とする分野で研究が行われてきた [11][12].

しかし,CCD 手法は DCD 手法よりも計算コストが高く, コンピュータゲームにおいては, 計算速度を優先するために衝突判定精度は低くしている場合が多い. コンピュータゲームにおける CCD 手法の一例として,NVIDIA 社の PhysX エンジンを使用する Unity では,Sweep-based CCD と Speculative CCD という手法 [13][14] がとられている.

Sweep-based CCD 手法は剛体の平行移動を想定した手法であり, 剛体の運動方向に沿って剛体の衝突判定形状をスイープさせることによって, 連続的な衝突判定を可能にしている. しかし, この手法は剛体の角運動を無視するため, 剛体が回転運動をする場合に対処できない.

Sweep-based CCD 手法の類似手法に,Swept Sphere Volume(以下,SSV) 手法 [15] がある. 運動前の剛体を包み込む球状境界ボリュームを運動後の位置までスイープすることによってカプセル形状を構成し, 衝突判定を行う手法である. そのため,FPS(First Person Shooting) ゲームの銃弾のように, 高速で動く剛体を扱うコンピュータゲーム [16] で使用する手法である.SSV 手法も, 剛体の角運動は取り扱わない.

Speculative CCD 手法はこれら 2 つの手法とは異なり, 並進運動のみではなく角運動にも対処した手法である. しかし,Speculative CCD は運動前の剛体と運動後の剛体を AABB で囲むことによって連続的衝突判定を行うため, 本来の運動軌跡よりもかなり広範囲に境界ボリュームを拡張

してしまうため, 本来衝突するはずのない剛体とも衝突可能性を算出してしまう. 並進運動の場合には Sweep-based CCD 手法や SSV 手法を使用することで衝突判定精度を維持しながら CCD を行うことができるが, 回転運動の場合には Speculative CCD 手法を使用するため, 衝突判定精度を維持しにくい.

本研究ではこの問題に対処するため, 回転運動軌跡の近似図形である扇形の境界ボリュームを開発し, 利用することによって, 回転運動を行う剛体の衝突判定精度が向上することを目的とした.

回転運動を考慮した CCD 手法である Speculative CCD 手法と, 本研究で提案する扇形境界ボリュームの連続的衝突判定精度を比較すると,Speculative CCD 手法は剛体の回転速度が上昇するにつれて精度が低くなっていくのに対して, 扇形境界ボリューム手法では回転速度にかかわらず高精度を維持していることを示した.

1.2 論文構成

本論文は、全 4 章にて構成する。第 2 章にて, 提案手法である扇形境界ボリュームの定義と, 他境界ボリュームとの干渉判定について述べる。第 3 章にて,Speculative CCD 手法と提案手法の精度向上の検証とその結果を述べる. そして第 4 章にてまとめを述べる.

第 2 章

提案手法

本章では, 衝突判定手法の向上を目的として開発した扇形ボリュームについて述べる.

2.1 扇形境界ボリューム

本論文では, 説明の簡単のため, 長方形が回転した際に生成する扇形について述べる. 本研究で想定する回転運動は以下の条件を満たす.

- 回転方向は正の方向とする.
- 回転角度は 0° から 180° とする.
- 回転の中心点は長方形の中心点から, 長方形のいずれかの線分と平行な方向に平行移動した, 長方形の外にある点である.

これらを満たす回転の一例を図 2.1 に示す.

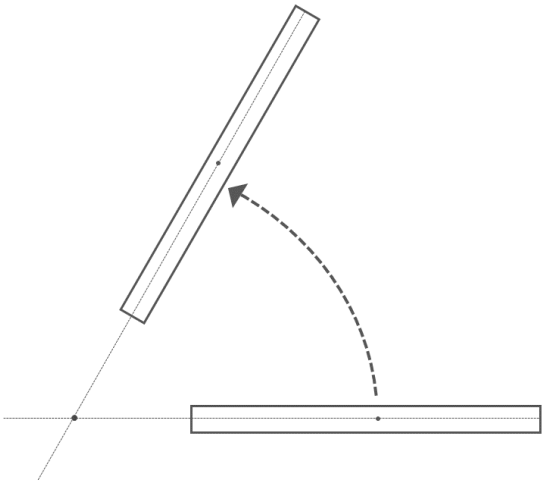

図 2.1　本研究における回転運動の一例

また, これにより生成される扇形はショートケーキ型と, バウムクーヘン型を想定する. 図 2.2 と図 2.3 にショートケーキ型とバウムクーヘン型の一例を示す. また, 現時点では 2 次元平面上での開発を進めているため,3 次元空間における扇形については言及しない.

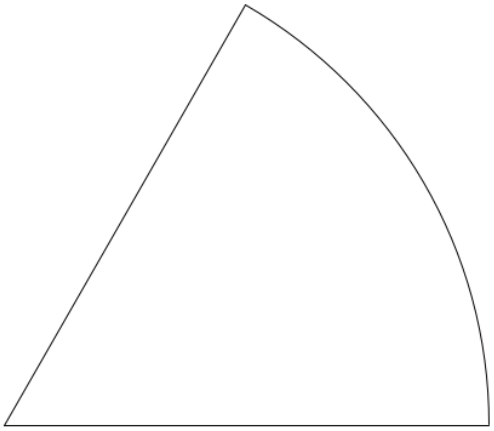

図 2.2　ショートケーキ型

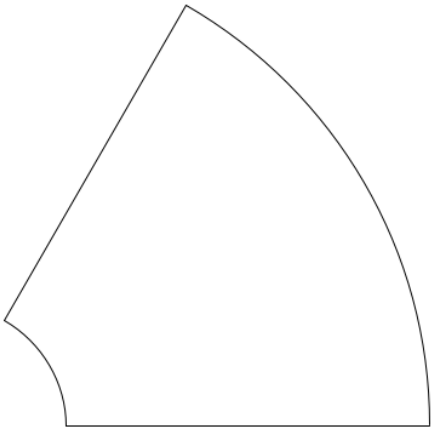

図 2.3　バウムクーヘン型

第 2.1 節と第 2.2 節では, 扇形境界ボリュームの各要素を以下のように定義する.

- 回転中心点は点 O とする.
- 線分の端点はそれぞれ点 A,B と点 A',B' とする.

図 2.4 にこの定義の模式図を示す.

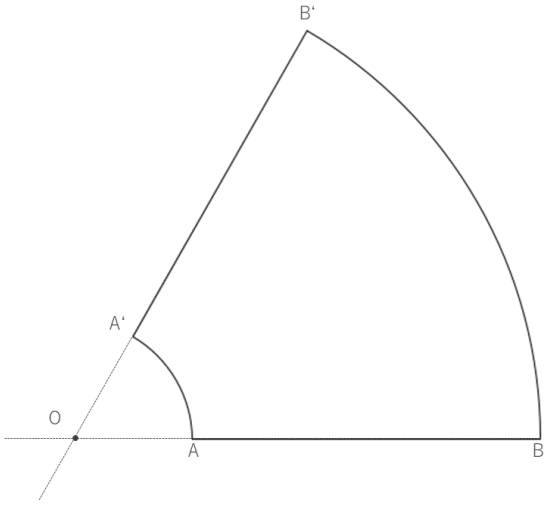

図 2.4　扇形の各点の定義図

2.2 点との内外判定

扇形境界ボリュームは 4 つの領域の組み合わせと考えることができる. それは以下の領域である.

- 点 O を中心点とする点 A,A' を通る円
- 点 O を中心点とする点 B,B' を通る円
- 線分 AB によってできる半空間
- 線分 A'B' によってできる半空間

ここでベクトル $\mathbf{R}, \mathbf{R}'$ を $\mathbf{R} = (\mathbf{P} - \mathbf{O}) \times (\mathbf{A} - \mathbf{O}), \mathbf{R}' = (\mathbf{P} - \mathbf{O}) \times (\mathbf{A}' - \mathbf{O})$ とすると, ある点 P が扇形の内側にある場合, 次の条件をすべて満たす. なお,XY 平面における条件としている.

$$|\mathbf{A} - \mathbf{O}| \leq |\mathbf{P} - \mathbf{O}| \leq |\mathbf{B} - \mathbf{O}| \tag{2.1}$$

$$R_z \leq 0 \tag{2.2}$$

$$R'_z \geq 0 \tag{2.3}$$

ただし, 式 (2.2) と式 (2.3) は剛体の回転方向が負の場合は不等式が反転する.

2.3 長方形との干渉判定

長方形と扇形の干渉判定では, まず剛体同士の交差点を算出する. その後, 第 2.2 節で述べた扇形と点の内外判定に交差点を用いることによって干渉判定を行う.

長方形は 4 つの線分で構成された図形であるため, 交差点の算出には「線分同士の交差点算出 [17]」と「円と線分の交差点算出 [18]」を利用する.

2.3.1 線分同士の交差点算出

本節では, 線分同士の交差点算出について説明する.

点 A,B を端点とする線分を線分 AB とし, 同様に点 C,D を端点とする線分を線分 CD とする. 線分 AB と線分 CD が交差するとき, 交差点を点 P とすると, 媒介変数 s, t を用いて点 P の位置ベクトルは式 (2.4) のように表せる.

$$\mathbf{P} = \mathbf{A} + s(\mathbf{B} - \mathbf{A}) = \mathbf{C} + t(\mathbf{D} - \mathbf{C}) \tag{2.4}$$

式 (2.4) を x,y 座標について整理すると式 (2.5) のようになる.

$$\begin{cases} A_x + s(B_x - A_x) = C_x + t(D_x - C_x) \\ A_y + s(B_y - A_y) = C_y + t(D_y - C_y) \end{cases} \tag{2.5}$$

式 (2.5) を s, t について解くと, 式 (2.6) のようになる.

$$\begin{aligned} s &= \frac{(C_x - A_x)(D_y - C_y) - (C_y - A_y)(D_x - C_x)}{(B_x - A_x)(D_y - C_y) - (C_y - B_y)(D_x - C_x)} \\ t &= \frac{(A_x - C_x)(B_y - A_y) - (A_y - C_y)(B_x - A_x)}{(D_x - C_x)(B_y - A_y) - (D_y - C_y)(B_x - A_x)} \end{aligned} \tag{2.6}$$

線分 AB,CD が平行でないとき, 式 (2.6) から s,t の値が決まるため,s, t いずれかの値を式 (2.5) に代入することで, 点 P の座標を求めることができる.

また,s, t が式 (2.7) の条件を満たさないとき, 線分 AB と線分 CD は交差していない.

$$0 \leq s, t \leq 1 \tag{2.7}$$

2.3.2 円と線分の交差点算出

本節では, 円と線分の交差点算出について説明する.

点 A,B を端点とする線分を線分 AB とし, 点 C を中心とする半径 r の円を円 C とする. 線分 AB と円 C が交差するとき, その交点を点 P,Q とする.

媒介変数 s を使うと, 点 A,B を通る直線上の点 L の位置ベクトルは式 (2.8) のようにあらわせる.

$$\mathbf{L} = \mathbf{A} + s(\mathbf{B} - \mathbf{A}) \tag{2.8}$$

線分 AB と円 C が交差しているとき, 円 C の方程式に点 L の x,y 成分を代入することで s の値が決定し, 交点を求めることができる. 点 L の x,y 成分を円 C の方程式に代入すると式 (2.9) のようになる.

$$(L_x - C_x)^2 + (L_y - C_y)^2 = r^2 \tag{2.9}$$

式の簡略化のため, ベクトル $\mathbf{V}$,$\mathbf{W}$ を $\mathbf{V} = \mathbf{B} - \mathbf{A}$,$\mathbf{W} = \mathbf{A} - \mathbf{C}$ のように定義する. 式 (2.9) を s についてまとめると, 式 (2.10) のようになる.

$$(V_x^2 + V_y^2)s^2 + 2(V_xW_x + V_yW_y)s + (W_x^2 + W_y^2 - r^2) = 0 \tag{2.10}$$

二次方程式の解の公式より, 式 (2.10) から s の値を求めることができる. 求めた s の値を $0 \leq s \leq 1$ に制限し, 式 (2.8) に代入することにより, 点 PQ の値を求めることができる. また, 二次方程式の解の公式の判別式から交差点の数を判別することができる. 判別式を D とすると, 交差点の数と点 P,Q の関係は次のようになる.

- $D > 0$ の時, 交差点の数は 2 つで,$\mathbf{P} \neq \mathbf{Q}$
- $D = 0$ の時, 交差点の数は 1 つで,$\mathbf{P} = \mathbf{Q}$
- $D < 0$ の時, 交差点は存在しない

2.4 円との干渉判定

円と扇形の干渉判定は, 長方形との干渉判定と同様に, まず剛体同士の交差点を算出し, その後, 第 2.3 節同様に扇形と点の内外判定に交差点を用いることによって干渉判定を行う.

交差点の算出には第 2.3.2 節で述べた「円と線分の交差点算出 [18]」と「円同士の交差点算出 [19]」を利用する.

2.4.1 円同士の交差点算出

本節では, 円同士の交差点算出について説明する.

点 A を中心とする半径 r_a の円を円 A とし, 点 B を中心とする半径 r_b の円を円 B とする.

円 A の方程式を式 (2.11) に, 円 B の方程式を式 (2.12) に示す.

$$(x - A_x)^2 + (y - A_y)^2 = r_a^2 \tag{2.11}$$
$$(x - B_x)^2 + (y - B_y)^2 = r_b^2 \tag{2.12}$$

式 (2.11) と式 (2.12) を連立させ, 変数 y について解くと, 式 (2.4.1) のようになる.

$$y = -\frac{A_x - B_x}{A_y - B_y}x + \frac{A_x^2 - B_x^2 + A_y^2 - B_y^2 - r_a^2 + r_b^2}{2(A_y - B_y)} \tag{2.13}$$

式で求めた変数 y の値を式 (2.11) か式 (2.12) のいずれかに代入することで, 変数 x の二次方程式を求めることができる.

求めた二次方程式から, 二次方程式の解の公式より変数 x の値が決定するため, 交点座標も決定される. 第 2.3.2 節同様に, 二次方程式の解の公式の判別式より, 交差点の数がわかる. 判別式を D とすると,D の値と交差点の数は以下のような関係になる.

- $D > 0$ の時, 交差点の数は 2 つ
- $D = 0$ の時, 交差点の数は 1 つ

- $D < 0$ の時, 交差点は存在しない

ただし,(2.4.1) より,$A_y = B_y$ の時は 0 除算が発生してしまうことがわかるので,$A_y = B_y$ の場合は変数 x について解くことで対処する. もし,$A_y = B_y$ かつ $A_x = B_x$ の場合, 交差点の数は次のようになる.

- $r_a = r_b$ の場合: 交差点は無限の数存在する.
- $r_a \neq r_b$ の場合: 交差点は存在しない.

このため,$A_y = B_y$ かつ $A_x = B_x$ の場合は交差点を求めることができない.

第 3 章

提案手法の検証

本章では, 回転物体に既存手法と扇形境界ボリュームを適用し, それぞれの衝突判定精度を比較した結果を述べる.

3.1 対象手法と検証方法

本研究では, 回転運動に対処した CCD 手法である Speculative CCD 手法と, 提案手法である扇形境界ボリューム手法の衝突判定精度を比較する. 検証には, 次のようなプログラムを作成した.

1. 画面内に 100 × 100 個の XY 軸並行な長方形 (以下, 判定長方形) を配置
2. 回転長方形 R を 1frame に 1° ずつ回転させ,180° になるまで回転させる
3. 回転長方形 R の回転運動に Speculative CCD と扇形境界ボリュームを適用
4. 扇形境界ボリュームと判定長方形の衝突個数を 1frame ごとに記録
5. Speculative CCD により作成された AABB と判定長方形の衝突個数を 1frame ごとに記録
6. 回転長方形 R が 180° 回転するまでに「R と衝突した判定長方形の数」も 1frame ごとに記録
7. 手順 4 から手順 6 で記録した内容を CSV ファイルに出力

本プログラムの実装には processing-4.0 beta8[20] を使用した.

3.2 結果

扇形境界ボリュームと判定長方形の衝突数を「Sector」,Speculative CCD と判定長方形の衝突数を「SpeculativeCCD」, 回転長方形 R と判定長方形の衝突数を「RotatedBox」として, 折れ線グラフに衝突数をまとめた. 縦軸はそれぞれの手法と判定長方形の衝突数とし, 横軸は回転長方形 R の 1frame における回転速度とする. 図 3.1 にそのグラフを示す.

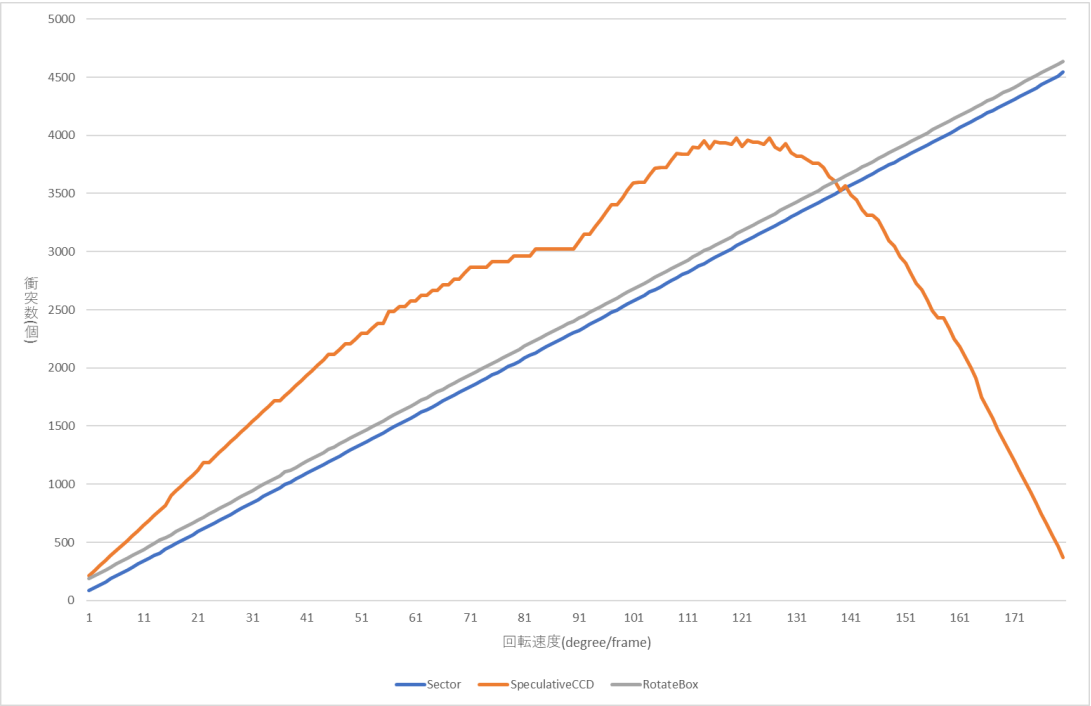

図 3.1　回転速度と手法ごとの衝突数の関係

図 3.1 から, 扇形境界ボリュームと回転長方形 R の判定長方形との衝突数にはそれほど差がないことがわかる.Speculative CCD 手法と回転長方形 R の判定長方形との衝突数には明らかに差があることがわかる. また, 扇形境界ボリュームと回転長方形 R の衝突数に存在する差は, 第 3 章で述べた本研究において想定する回転運動の定義から, 図 3.2 に示す薄青色の範囲に相当する.

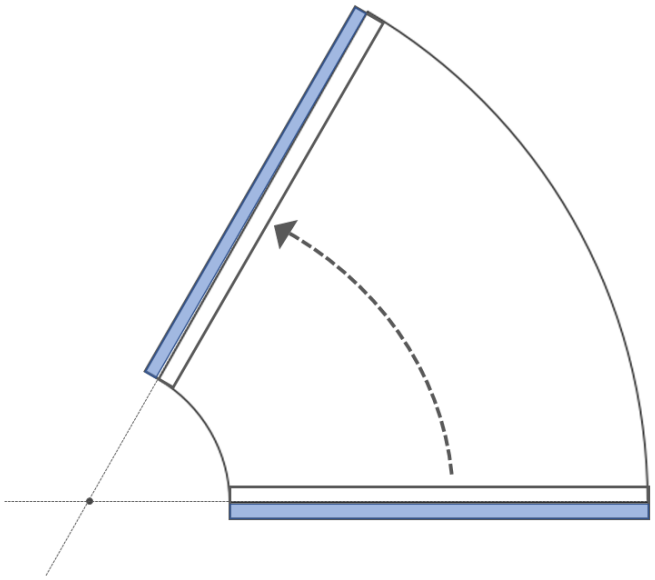

図 3.2　扇形境界ボリュームと回転長方形 R の衝突数の差部分

次に, 衝突判定精度について考察を行うため, 次の 4 つのデータを扇形境界ボリュームと Speculative CCD に対して算出した.

- 同時判定数: 回転長方形 R と連続的衝突判定手法が共に衝突したと判定した判定長方形の数
- 過剰判定数: 回転長方形 R は衝突したと判定していないが, 連続的衝突判定手法は衝突したと判定した判定長方形の数
- 不足判定数: 回転長方形 R は衝突したと判定したが, 連続的衝突判定手法は衝突していないと判定した判定長方形の数
- 誤判定数: 過剰判定数と不足判定数の和

図 3.3, 図 3.4, 図 3.5, 図 3.6 にそれぞれのグラフを示す.

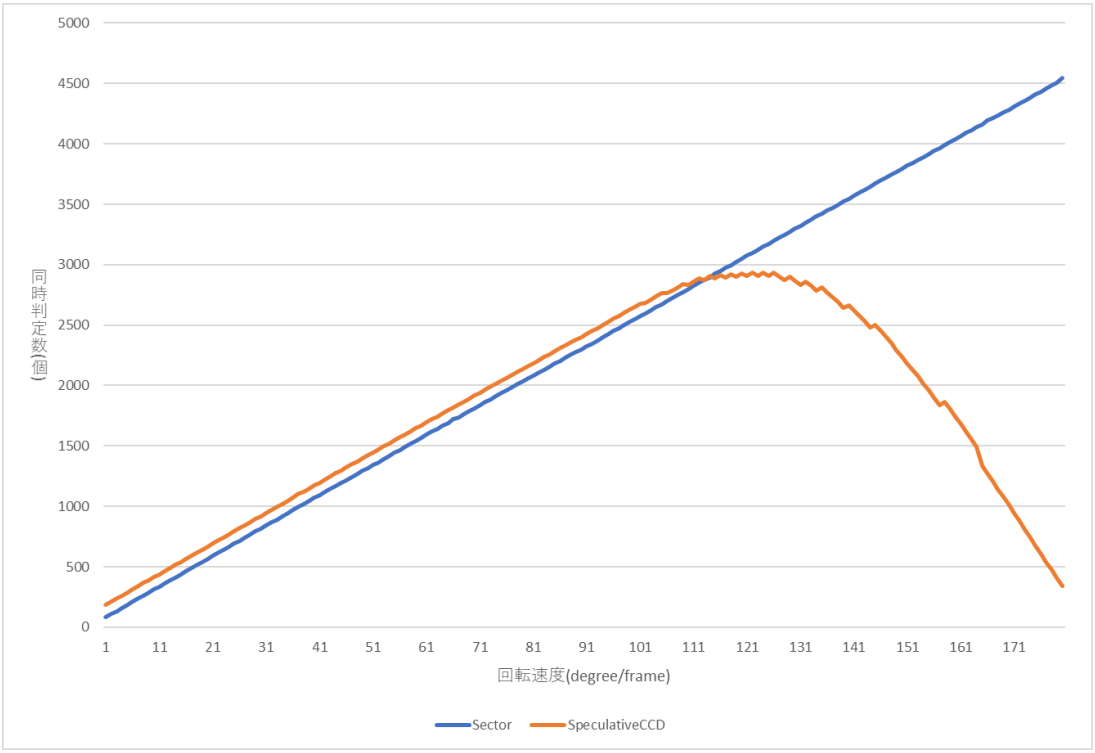

図 3.3　回転速度と手法ごとの同時判定数の関係

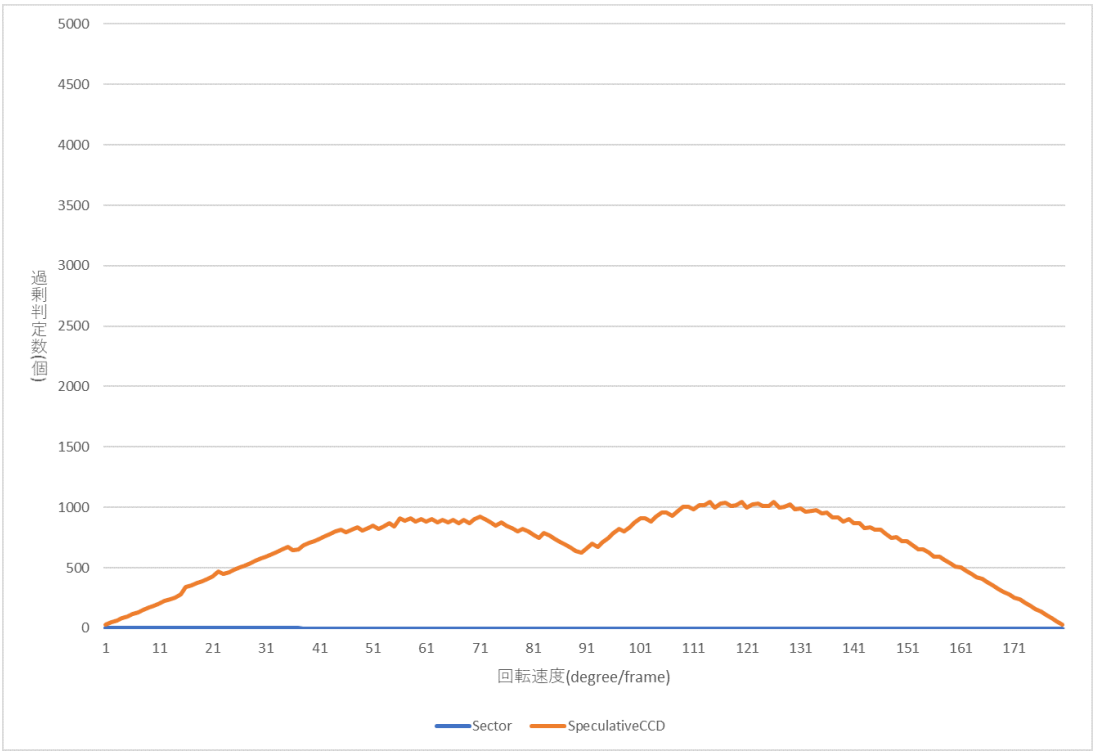

図 3.4　回転速度と手法ごとの過剰判定数の関係

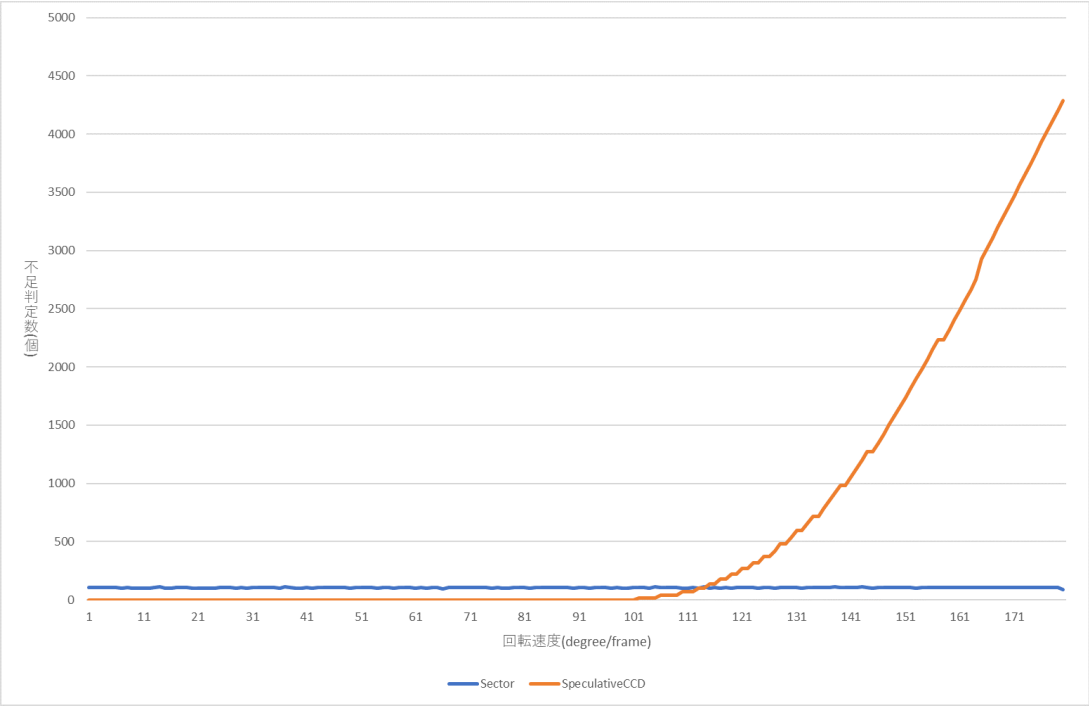

図 3.5　回転速度と手法ごとの不足判定数の関係

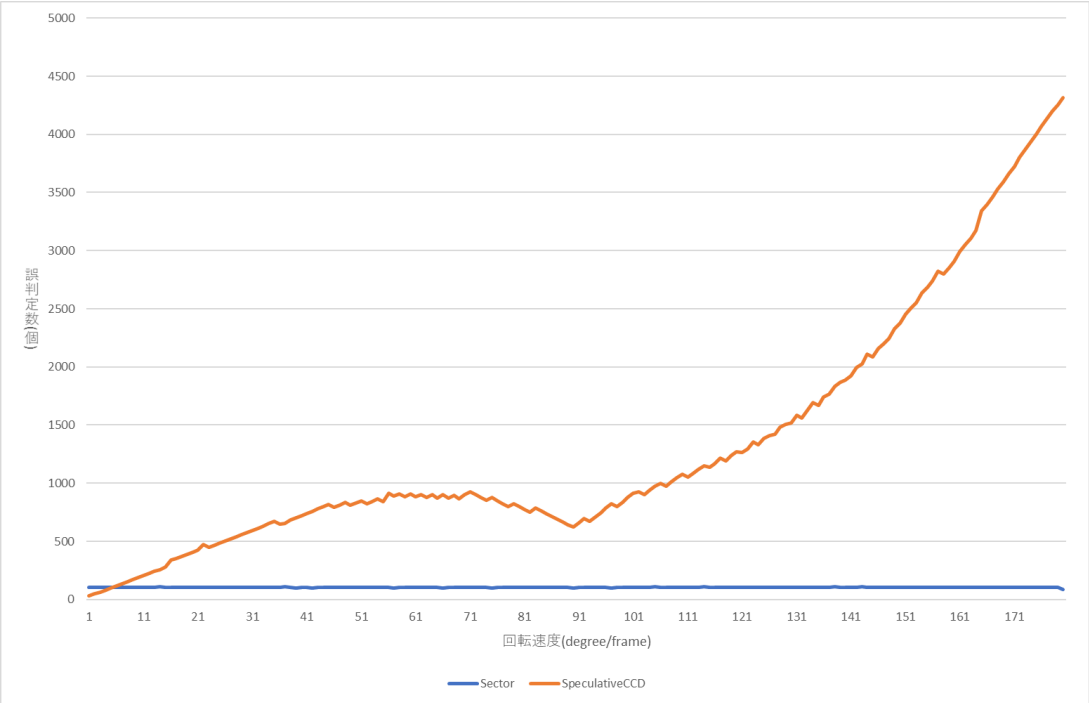

図 3.6　回転速度と手法ごとの誤判定数の関係

図 3.3, 図 3.4, 図 3.5, 図 3.6 から, 回転長方形 R の回転速度が上昇するにつれて Speculative

CCD 手法の衝突判定精度は低くなっていくことがわかる. 対して, 扇形境界ボリュームの衝突判定精度は回転速度が上昇しても高精度な状態を維持していることがわかる. このことから, 回転剛体に対する連続的衝突判定手法として,Speculative CCD 手法よりも, 扇形境界ボリュームの方が衝突判定精度は優れていることが判明した.

第 4 章

まとめ

本研究では, 本手法と既存の連続的衝突判定手法の一つである Speculative CCD 手法を, 回転剛体における衝突判定の制度について比較した. その結果,Speculative CCD は回転剛体の 1frame における回転速度が上昇するにつれて衝突判定精度が減少するのに対して, 本手法こと扇形境界ボリュームでは回転速度が上昇しようともその衝突判定精度は高精度を維持しており, 既存手法に対して衝突判定精度が向上したことがわかった.

しかし, 本研究では回転剛体の連続的衝突判定の精度向上を目的に扇形境界ボリュームを開発し回転運動長方形に適用したが, ゲームなどのリアルタイムグラフィックスにおいては衝突判定にかかる実行時間も考慮しなければならない. その点において Speculative CCD 手法は非常に高速に作成, 干渉判定を行うことができる AABB を採用しているため, 衝突判定の実行時間については扇形境界ボリュームよりも利便性が高い可能性がある.

今後は, 上述したようなリアルタイムにおける実行時間を考慮し, 扇形境界ボリュームの干渉判定実行時間の短縮や, 扇形境界ボリュームと他プリミティブ図形との干渉判定を定義することで扇形境界ボリュームの利便性を向上させることが課題として挙げられる. また, 本研究では二次元平面における連続的衝突判定について想定したが, 三次元空間における判定も可能にすることで, さらに扇形境界ボリュームの利便性向上につながると考えている.

謝辞

本論文を執筆するにあたりご指導いただいた渡辺先生, 阿部先生に心より感謝いたします.

本研究の肝となった扇形で回転運動の軌跡を補間するというアイディアは渡辺先生からのご指導の中から生まれたものであり, 私一人では発想に至ることはできませんでした. 扇形と矩形の干渉判定を思考する際にも, 自身の当時の発想で至らない点などを指摘していただき, 非常に感謝しております.

阿部先生には研究を進めるにあたって行わなければいけない点や考慮しなければいけない点を指摘していただき, 自身の研究の状態を振り返ることができました.

また, 自身の研究について意見をくれた先輩方や友人たちにも心より感謝を申し上げます.

皆様, 本当にありがとうございました.

参考文献

[1] Lazaros Lazaridis, Maria Papatsimouli, Konstantinos-Filippos Kollias, Panagiotis Sarigiannidis, and George F.Fragulis. Hitboxes: A Survey About Collision Detection in Video Games. In *HCI in Games: Experience Design and Game Mechanics*, pp. 314–326. Springer, 2021.

[2] Chaoqiang Tu and Lizhen Yu. Research on Collision Detection Algorithm Based on AABB-OBB Bounding Volume. In *2009 First International Workshop on Education Technology and Computer Science*, Vol. 1, pp. 331–333, 2009.

[3] Dehan Kong, Yongshan Liu, and Na Cui. Collosion Detection Research Based on Capsule Bounding Volume ☒. *Journal of Computational Information Systems*, Vol. 10(7), p. 2743 – 2750, 2014.

[4] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 4, No. 1, pp. 21–36, 1998.

[5] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, Vol. 4, No. 2, pp. 193–203, 1988.

[6] ゲームつくろー！衝突判定編. `http://marupeke296.com/COL_main.html`. 参照: 2022.7.1.

[7] Gino van den Bergen. Efficient Collision Detection of Complex Deformable Models Using AABB Trees. *J. Graph. Tools*, Vol. 2, No. 4, p. 1 – 13, 1998.

[8] S.Gottschalk, M.C.Lin, and D.Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. *Computer Graphics*, Vol. 30, p. 171 – 180, 1997.

[9] ChristerEricson 著, 中村達也訳. ゲームプログラミングのためのリアルタイム衝突判定. 株式会社ボーンデジタル, 2005.

[10] Daniel Meister, Shinji Ogaki, Carsten Benthin, Michael J.Doyle, Michael Guthe, and Jiri Bittner. A Survey on Bounding Volume Hierarchies for Ray Tracing. *Computer Graphics Forum*, Vol. 40, pp. 683–712, 2021.

[11] Hamzah Asyrani Sulaiman, Mohd Azlishah Othman, Mohd Muzafar Ismail, Maizatul Alice Meor Said, Azri Ramlee, Mohamad Harris Misran, Abdullah Bade, and Mohd Harun Abdullah. Distance computation using axis aligned bounding box (AABB) parallel distribution of dynamic origin point. In *2013 Annual International Conference on Emerging Research Areas and 2013 International Conference on Microelectronics, Communications and Renewable Energy*, pp. 1–6, 2013.

[12] Edvin Åblad, Domenico Spensieri, Robert Bohlin, and Ann-Brith Strömberg. Continuous Collision Detection of Pairs of Robot Motions Under Velocity Uncertainty. *IEEE Transactions on Robotics*, Vol. 37, No. 5, pp. 1780–1791, 2021.

[13] Unity Technologies. CCD (連続的衝突判定). `https://docs.unity3d.com/ja/2019.4/Manual/ContinuousCollisionDetection.html`. 参照: 2022.6.9.

[14] NVIDIA Corporation. Advanced Collision Detection. `https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/AdvancedCollisionDetection.html`. 参照: 2022.7.12.

[15] Eric Larsen, Stefan Gottschalk, Ming Lin, and Dinesh Manocha. Fast Proximity Queries with Swept Sphere Volumes. *Technical Report TR99-018, Department of Computer Science, Universityl of North Carolina, Chapel Hill*, 1999.

[16] Respawn Entertainment. Extreme SIMD: Optimized Collision Detection in 'Titanfall'. `https://www.gdcvault.com/play/1025126/`

Extreme-SIMD-Optimized-Collision-Detection. 参照: 2022.6.27.

[17] zu_rin. 2 線分の交点座標 (2 次元). https://qiita.com/zu_rin/items/09876d2c7ec12974bc0f. 参照: 2022.6.17.

[18] yaketake08. 直線 (線分) と円の交点. https://tjkendev.github.io/procon-library/python/geometry/circle_line_cross_point.html. 参照: 2022.6.17.

[19] 具体例で学ぶ数学. 2 つの円の交点の座標を求める＋答えの確認方法. https://mathwords.net/ennokoten. 参照: 2022.6.17.

[20] Foundation@processing.org. Reference. https://processing.org/reference/. 参照: 2022.6.28.