芸術科学会論文誌 Vol. 23, No. 3, pp. 7:1 – 7:16 (2024)

リアルタイムグラフィックスにおける回転剛体の衝突判定精度向上に関する研究

山本輝 1)(学生会員)　阿部雅樹 2)(正会員)　渡辺大地 3)(正会員)

1) 東京工科大学大学院バイオ・情報メディア研究科　2) 3) 東京工科大学メディア学部

A Research on Improvement of Accuracy of Collision Detection for Rotaion Rigid Bodies In Real-Time Graphics

Hikaru Yamamoto[1)]　Masaki Abe[2)]　Taichi Watanabe[3)]

1) Graduate School of Bionics, Computer and Media Science, Tokyo University of Technology
2) 3) School of Media Science, Tokyo University of Technology

1) g3122034e1@edu.teu.ac.jp 2) abemsk@edu.teu.ac.jp 3) earth@gamescience.jp

概要

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，高速に判定を行える境界ボリュームが使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) 手法が提案されており，銃弾などの運動速度の大きい剛体を扱う FPS(First Person Shooting) 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，Sweep-based CCD 手法は角運動を無視しているため回転運動に対応しておらず，Speculative CCD 手法は回転運動の場合に衝突判定精度が低いという問題がある．本研究ではこの問題を解決するために，直方体の回転運動軌跡の近似形状である扇形によって運動軌跡を補間し，回転剛体の衝突判定精度の向上を目的とした．本論文では扇形とプリミティブ形状との干渉判定手法を提案し，扇形を用いた回転運動軌跡の補間によって衝突判定精度が向上したかどうかの検証結果と，提案手法の実行速度を示す．

Abstract

Today, collision detection between rigid bodies in computer games often uses Bounding Volumes that allow for fast decision making. However, because rigid bodies move discretely in the computer, they are tunneling each other without colliding if their motion velocity is very large. Various CCD (Continuous Collision Detection) methods have been proposed to address this problem and have been used in FPS (First Person Shooting) games that handle rigid bodies with large velocities such as bullets. One example is the Sweep-based CCD method and the Speculative CCD method used in Unity. However, the Sweep-based CCD method ignores angular motion and does not support rotational motion, while the Speculative CCD method has low collision detection accuracy for rotational motion. To solve this problem, this study interpolates the motion trajectory by using a fan shape, which is an approximate shape of the rotational motion trajectory of a rectangular body, to improve the accuracy of collision detection for rotating rigid bodies. This paper proposes a collision detection method between a fan shape and a primitive shape, and presents the verification results of whether the accuracy of collision detection is improved by interpolating the rotational motion trajectory using the fan shape, as well as the execution speed of the proposed method.

1 はじめに

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, 境界ボリュームを用いた手法 [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)[3]，球，カプセル [4] などがあり，その特徴は，同一形状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形状同士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場合があり，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，剛体の運動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)[7] と呼ぶ．CCD 手法は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 [8][9].

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，ゲームエンジンの Unity では，Sweep-based CCD と Speculative CCD(以降，S-CCD) という手法 [10] がある．

Sweep-based CCD 手法は剛体の並進運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．

S-CCD 手法は Sweep-based CCD 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，S-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衝突判定を行うため，本来の運動軌跡よりも広範囲に境界ボリュームを拡張してしまう．その為，本来の運動軌跡では衝突するはずのない剛体に対して衝突したと誤った判定を行う可能性がある．並進運動の場合には Sweep-based CCD 手法などを使用することで衝突判定精度を高精度に維持しながら衝突判定を行えるが，回転運動の場合には S-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形状である扇形によって運動軌跡の補間を行い，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．

扇形は非凸の形状になる場合があるため GJK アルゴリズムなどを使用できない．そこで，本論文では扇形とコンピュータゲームに頻出する球やカプセルといったプリミティブ形状との干渉判定を提案する．

本研究では回転運動を考慮した CCD 手法である S-CCD 手法と，扇形による運動軌跡補間の衝突判定精度を比較し，S-CCD 手法よりも衝突判定精度が向上したかを確認した．また，提案手法はリアルタイムグラフィックス上での使用を想定している．そこで，提案手法の実行速度を調査しリアルタイムでの実行に問題がないか確認した．

衝突判定精度の検証結果から，提案手法は剣や野球ゲームのバットのような細長い剛体が回転運動をする際に高精度になり，逆にハンマーなどの横幅が広い形状を持つ剛体が回転運動をする際に衝突判定精度が低くなることが判明した．そのため，提案手法は野球ゲームや剣などが登場するアクションゲーム，VR 剣戟ゲームにおいて有効に活用できる可能性がある．

我々は，本研究の初期成果を NICOGRAPH2023 にて発表し [11]，扇形と干渉判定を行える三次元形状の不足や，提案手法の実行速度の記述不足，回転運動を行う直方体の形状による衝突判定精度の違いに関する指摘を受

けた．本論文では，扇形と干渉判定を行うことのできる三次元形状としてカプセル形状を追加し，実行速度の調査結果と他手法との比較結果，考察を追記した．また，衝突判定精度についてより詳細に分析できるよう検証方法を修正し，指摘を受けた直方体の形状による衝突判定精度の違いに関する分析に加え，回転角度による衝突判定精度の違いについて追記した．

2　提案手法

2.1　本研究における扇形の定義

本節では，回転運動軌跡の近似形状である扇形の本研究における定義と，扇形とプリミティブ形状の干渉判定について述べる．

始めに，回転運動により生成される扇形の定義について述べる．細長い剛体であればあるほど回転運動時に衝突を無視する可能性も高くなるため，回転を行う剛体は二次元平面においては長方形とし，三次元空間では直方体とする．本研究で想定する回転運動は，剛体のローカル座標軸と回転軸，回転を行う剛体の中心点と回転中心点を通る直線が常に軸並行であるとし，回転中心点は回転を行う剛体の外部，または面上にあるとする．また，回転運動前後で回転中心点から回転を行う剛体の中心点までの距離は変わらないとし，回転角度は 0 度以上 360 度以下とする．図 1 に，本研究で想定する回転運動の一例を示す．

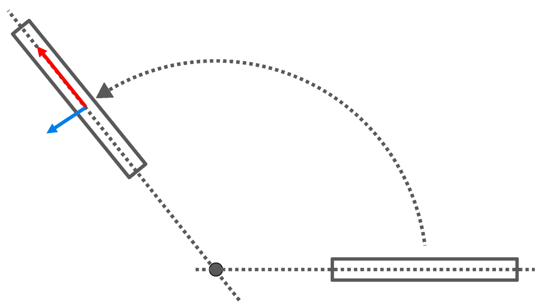


図 1　想定する回転運動の一例

また，これにより生成される扇形はピースケーキ型と，バウムクーヘン型を想定する．図 2 と図 3 にピースケーキ型とバウムクーヘン型の一例を示す．

続いて，扇形を構成する要素について考える．二次元平面における扇形は Krishnan らが提案した Spherical Shell[12] と同一の形状となる．しかし，三次元空間にお

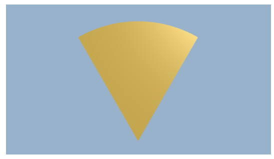


図 2　ピースケーキ型

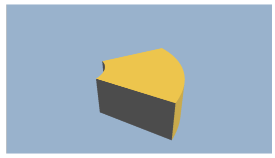


図 3　バウムクーヘン型

ける扇形は異なる形状であるため，本研究では Spherical Shell を参考に，扇形の構成要素を以下のように定義した．

- 回転中心点は点 O とする
- 回転軸を表す単位ベクトルを $\mathbf{U}$ とする
- 中心軸を表す単位ベクトルを $\mathbf{A}$ とする
- 範囲角度の半角を θ とする．
- 扇形を構成する 2 つの円のうち小さい円 (以降，内円) の半径を r，大きい円 (以降，外円) の半径を R とする
- 回転軸方向の厚みを $-h$ から h の $2h$ とする

上述した定義の模式図を図 4 に示す．

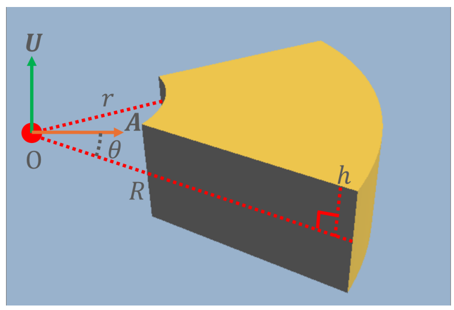


図 4　扇形の各点の定義図

続いて，直方体の回転運動から扇形を構成する方法を述べる．ここで，回転運動を行う前の直方体の中心点の位置ベクトルを $\mathbf{P}$，ローカル座標軸をそれぞれ $\mathbf{X}'$，$\mathbf{Y}'$，$\mathbf{Z}'$ とし，ローカル座標軸方向の直方体の厚みをそれぞれ W_x，W_y，W_z とする．ただし，前述した回転運動の制約から，回転軸 $\mathbf{U}$ は $\mathbf{Z}'$ と同一であり，回転中心点の位置ベクトルを $\mathbf{O}$ としたとき，$(\mathbf{P}-\mathbf{O})$ と $\mathbf{Y}'$ は同じ向きである．また，回転角度を α としたとき，扇形の各構成要素の値は式 (1) のようになる．また，中心軸ベクトル $\mathbf{A}$ はロドリゲスの回転公式 [13] から式 (2) のようになる．ただし，$\times$ は外積記号を表す．

$$
\begin{aligned}
\theta &= \frac{\alpha}{2} \,, \\
r &= |\mathbf{P} - \mathbf{O}| - \frac{W_y}{2} \,, \\
R &= |\mathbf{P} - \mathbf{O}| + \frac{W_y}{2} \,, \\
h &= \frac{W_z}{2} \,.
\end{aligned} \tag{1}
$$

$$
\mathbf{A} = (\cos\theta)\mathbf{Y}' + (1-\cos\theta)(\mathbf{Y}'\cdot\mathbf{U})\mathbf{U} + (\mathbf{U}\times\mathbf{Y}')\sin\theta \tag{2}
$$

次に，コンピュータゲームに頻出する幾つかのプリミティブ形状と扇形の干渉判定を定義する．判定数式を簡易化するため，本章では扇形を構成する要素の一部を次のように設定する．

- 回転中心点 O を原点とする
- 回転軸ベクトルを $\mathbf{U} = (0,0,1)$ とする
- 中心軸ベクトルを $\mathbf{A} = (1,0,0)$ とする

もし，扇形の回転中心点，回転軸ベクトル，中心軸ベクトルが上述の値ではない場合，座標変換を施すことで，本論文で提案する干渉判定手法を利用することが出来る．ここで，座標変換を施す点の位置ベクトルを Q とし，$\mathbf{U}$ と $\mathbf{A}$ に直行なベクトル $\mathbf{K}$ を $\mathbf{K} = \mathbf{U}\times\mathbf{A}$ としたとき，座標変換後の点の位置ベクトル $\mathbf{Q}'$ の各成分は式 (3) から算出できる．

$$
\begin{pmatrix} Q'_x \\ Q'_y \\ Q'_z \end{pmatrix} = \begin{pmatrix} A_x & A_y & A_z \\ K_x & K_y & K_z \\ U_x & U_y & U_z \end{pmatrix} \begin{pmatrix} Q_x - O_x \\ Q_y - O_y \\ Q_z - O_z \end{pmatrix} \tag{3}
$$

2.2 点との内外判定

本節では，任意の点 P が扇形の内部に存在するかどうか判定する方法について述べる．

点 P が三次元空間における扇形の厚み空間内部に存在する場合，始めに式 (4) を満たす．

$$
|P_z| \le h \tag{4}
$$

続いて，扇形の回転軸から点 P までの距離が扇形の内円半径以上，外円半径以下であるかを判定する．

$$
r^2 \le {P_x}^2 + {P_y}^2 \le R^2 \tag{5}
$$

最後に，xy 平面において点 P が扇形範囲角度以内に存在するかを判定する．まず，点 P を xy 平面に投影した点 P' の位置ベクトルと扇形の中心軸ベクトル $\mathbf{A}$ のなす角の余弦値 P_c を計算する．式 (6) から計算した P_c と扇形範囲角度の半角 θ の余弦値を比較し，点 P' が扇形範囲角度内に存在するかを判定する．

$$
P_c = \frac{\mathbf{A}\cdot\mathbf{P}'}{|\mathbf{P}'|} = \frac{P_x}{\sqrt{{P_x}^2 + {P_y}^2}} \,, \tag{6}
$$

$$
P_c \ge \cos\theta \,. \tag{7}
$$

式 (4)，式 (5)，式 (7) の条件式を満たすとき，点 P は扇形内部に存在する．

2.3 二次元平面における長方形との干渉判定

本節では，扇形と長方形の干渉判定手法について述べる．

長方形と扇形の干渉判定は，まず扇形を構成する線分，円と長方形を構成する線分の交差点を算出する．第 2.2 項で述べた扇形と点の内外判定に算出した交差点を代入し，算出した交差点のうち 1 つでも扇形上の点であることが判明したとき，扇形と長方形は干渉していることがわかる．交差点の算出には線分同士の交差点算出と，線分と円の交差点算出を利用する．

続いて，扇形が長方形に完全に内包されている場合や，長方形が扇形に完全に内包されている場合のように交差点が存在しない場合について述べる．これらの場合には，扇形内の任意の 1 点が長方形内部に存在するかどうかを第 2.2 節で述べた扇形と点の内外判定を用いて判定するか，または，長方形内の任意の 1 点が扇形内部に存在するかどうかを長方形と点の内外判定を用いて判定することにより，扇形と長方形が干渉しているかどうかを判定する．

2.4 二次元平面における円との干渉判定

本節では，中心点が点 P，半径が r_c の円と扇形の干渉判定について述べる．

まず，点 P が扇形の角度範囲内に存在するかどうかを判定する．これは，式 (7) を利用する．式 (7) の結果が真であるなら，式 (5) を円の半径 r_c 分拡張させた式 (8) を満たすかどうか判定することで，扇形と円が干渉しているかどうかがわかる．

$$(r-r_c)^2 \le {P_x}^2 + {P_y}^2 \le (R+r_c)^2 \tag{8}$$

式 (5) の結果が偽であるなら，扇形を構成する 2 つの線分と円の中心点 P の最小距離を計算し，扇形と円が干渉しているかどうかを判定する．ここで，扇形の 2 つの線分をそれぞれ線分 L と線分 M としたとき，線分 L の方向ベクトル $\mathbf{G}$ と，線分 M の方向ベクトル $\mathbf{H}$ は式 (9) のようになる．

$$\begin{cases} \mathbf{G} &= (\cos\theta, \sin\theta) \\ \mathbf{H} &= (\cos\theta, -\sin\theta) \end{cases} \tag{9}$$

線分 L の端点を点 G'，点 G" とし，線分 M の端点を点 H'，点 H" とすると，各端点の位置ベクトルの値は式 (10) のようになる．

$$\begin{cases} \mathbf{G}' &= r\mathbf{G} \\ \mathbf{G}'' &= R\mathbf{G} \\ \mathbf{H}' &= r\mathbf{H} \\ \mathbf{H}'' &= R\mathbf{H} \end{cases} \tag{10}$$

線分 L，線分 M と円の中心点 P との最短距離を点と線分の最短距離計算によって算出する．算出した最短距離の値が r_c 以下であるとき，扇形と円は干渉している．

2.5　三次元空間における球との干渉判定

本節では，中心点が点 P，半径が r_s の球と扇形の干渉判定について述べる．

始めに，球と扇形の厚み空間が干渉しているかを判定する．これは，式 (4) を球の半径分拡張させた式 (11) で判定を行う．

$$|P_z| \le h + r_s \tag{11}$$

式 (11) を満たすとき，次の処理を行う．球は半径の異なる円が層状に重なった形状であると考えられることから，扇形の厚み空間内における円の最大半径 R_m を計算する．

球の中心点 P を式 (4) に代入した結果が真であるとき，R_m は球の半径 r_s と等しい．式 (4) の結果が偽であるとき，式 (12) により，R_m を求める．また，R_m についての模式図を図 5 と図 6 に表す．

$$R_m = \sqrt{{r_s}^2 - (|P_z| - h)^2} \tag{12}$$

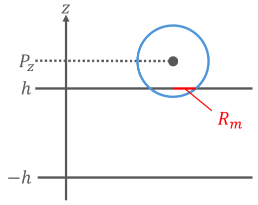


図 5　P_z が厚み空間外にある時

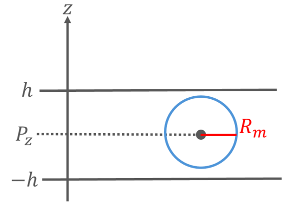


図 6　P_z が厚み空間内にあるとき

続いて，球の中心点 P を xy 平面に投影し点 P' を得る．これにより，中心点が点 P'，半径が R_m の円を第 2.4 節の干渉判定処理に代入することが出来る．代入した計算結果が，球と扇形の干渉判定の結果となる．

2.6　二次元平面におけるカプセルとの干渉判定

本節では，始点 S と終点 E，半径 r_c からなるカプセルと扇形の干渉判定について述べる．

始めに，カプセルの中心をとおる線分 (以降，カプセル線分) の方向ベクトル $\mathbf{V}$ を $\mathbf{V} = \mathbf{E} - \mathbf{S}$ とすると，線分上の点 P は媒介変数 t を用いて，式 (13) のように表せる．ここで t は式 (14) という値域を持つ．

$$\mathbf{P} = \mathbf{S} + t\mathbf{V}\ , \tag{13}$$
$$0 \le t \le 1\ . \tag{14}$$

カプセルは，カプセル線分に沿って半径 r_c の円がスイープしてできる形状である [14]．このことから，中心点を P とし，半径が r_c の円 C が扇形と干渉する条件を考える．

まず，円 C が扇形を構成する内円，外円の少なくとも片方と干渉する条件は，式 (15) のようになる．

$$(r-r_c)^2 \le {P_x}^2 + {P_y}^2 \le (R+r_c)^2 \tag{15}$$

式 (15) を展開してえられる媒介変数 t の値域は，次の 2 つの二次方程式の解から導き出される．

$${P_x}^2 + {P_y}^2 = (R+r_c)^2\ , \tag{16}$$
$${P_x}^2 + {P_y}^2 = (r-r_c)^2\ . \tag{17}$$

ここで，$P_x = S_x + tV_x$，$P_y = S_y + tV_y$ より，式 (16)，式 (17) は t の二次式となる．また，式 (16) の解を a_R，$b_R (a_R \le b_R)$ とし，式 (17) の解を a_r，$b_r (a_r \le b_r)$ とすると，式 (15) から得られる t の値域は，式 (16) の判別

式 D_R と式 (17) の判別式 D_r を用いて，次の 3 つの場合に分けられる．また，判別式 D_R と D_r から成る 3 つの場合分けは，図 7 に示す状況を表している．

- $D_R < 0$ のとき：値域なし
- $D_R \geq 0$ かつ $D_r \geq 0$ のとき：$a_R \leq t \leq a_r$，$b_r \leq t \leq b_R$
- $D_R \geq 0$ かつ $D_r < 0$ のとき：$a_R \leq t \leq b_R$

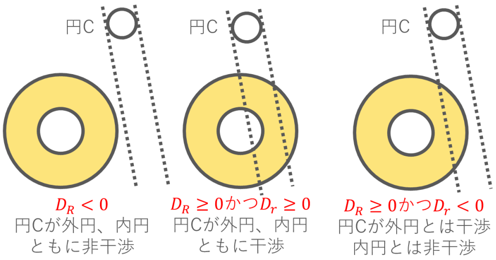


図 7　判別式からわかる 3 つの場合

続いて，点 P が扇形の範囲角度内に存在する t の範囲を求める．ここで，扇形を構成する 2 つの線分の方向ベクトル $\mathbf{G}$，$\mathbf{H}$ は次のようになる．

- $\mathbf{G} = (\cos\theta, \sin\theta)$
- $\mathbf{H} = (\cos\theta, -\sin\theta)$

点 P の位置ベクトルと線分の方向ベクトル $\mathbf{G}$，$\mathbf{H}$ の外積演算により算出されるベクトルをそれぞれ，$\mathbf{G}'$ と $\mathbf{H}'$ とすると，$\mathbf{G}'$ と $\mathbf{H}'$ の z 成分から，点 P が扇形の範囲角度内に存在するかどうかを判定できる．ここで，${G'}_z$ は式 (18)，${H'}_z$ は式 (19) のようになる．

$$\begin{aligned} {G'}_z &= P_x G_y - P_y G_x \\ &= S_x G_y - S_y G_x + (V_x G_y - V_y G_x)t \end{aligned} \tag{18}$$

$$\begin{aligned} {H'}_z &= P_x H_y - P_y H_x \\ &= S_x H_y - S_y H_x + (V_x H_y - V_y H_x)t \end{aligned} \tag{19}$$

点 P が扇形の範囲角度内に存在するかどうかの条件式は，扇形範囲角度の半角 θ の値から，式 (20) のように場合分けを行う．

$$\begin{cases} {G'}_z \geq 0 \text{ and } {H'}_z \leq 0 & \text{if } 0 \leq \theta < \frac{\pi}{2} \\ {G'}_z \geq 0 \text{ or } {H'}_z \leq 0 & \text{if } \frac{\pi}{2} \leq \theta \leq \pi \end{cases} \tag{20}$$

ただし，式 (20) において ${G'}_z$，${H'}_z$ から t の値域を求めるとき，式 (18) の $V_x G_y - V_y G_x$ と式 (19) の $V_x H_y - V_y H_x$ の正負によって不等式の符号を考慮する必要がある．

式 (14) と，式 (15)，式 (20) のすべてを満たす t の値域が存在するとき，扇形とカプセルは干渉している．

しかし，式 (14) と，式 (15)，式 (20) のすべてを満たす t の値域が存在しないとしても，扇形とカプセルが干渉している場合がある．図 8 にその一例を示す．そのため，式 (14) と，式 (15)，式 (20) のすべてを満たす t の値域が存在しないならば，線分と線分の最短距離計算を用いて，扇形を構成する 2 つの線分とカプセル線分の最小距離を計算する．こうして計算できた最小距離 d_G と d_H のどちらか，または両方が r_c 以下の時，扇形とカプセルは干渉している．

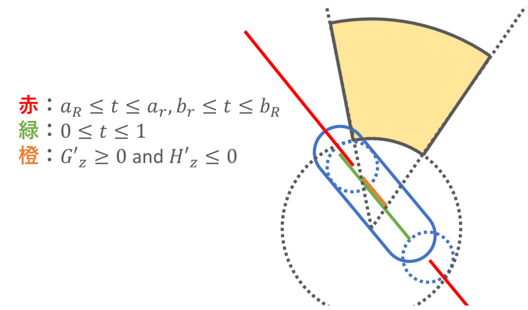


図 8　式 (14) と，式 (15)，式 (20) のすべてを満たす t の値域が存在しないが，扇形とカプセルが干渉している一例

2.7　三次元空間におけるカプセルとの干渉判定

本節では，始点 S と終点 E，半径 r_c からなるカプセル C と扇形の干渉判定について述べる．本節の干渉判定は近似解であるため，本来は干渉していないが干渉していると判定する様な誤った判定結果になる場合があるが，それについては後述する．

始めに，カプセル C が扇形の厚み空間と干渉しているかどうかを確認する．ここで $S_z > E_z$ であるとすると，式 (21) を満たすときカプセル C と厚み空間は干渉していないことがわかる．カプセル C と扇形の厚み空間が干渉していないなら，カプセル C と扇形が干渉している可能性はないため，以降の干渉判定処理を行う必要は無い．

$$S_z + r_c < -h \text{ or } E_z - r_c > h \tag{21}$$

三次元空間におけるカプセルと扇形の干渉判定は，第

2.5 節で述べた扇形と球の干渉判定同様，二次元平面の干渉判定に置き換えることを目指す．まず，カプセルと扇形の厚み空間からなる二次元形状（以降，疑似カプセル）について考える．カプセルは球が平行移動したときの移動軌跡であるため，第 2.5 節で述べた厚み空間内の最大半径の円から疑似カプセルの形がわかる．図 9 に疑似カプセルの模式図を示す．カプセルと扇形の厚み空間の位置関係が図 9 の左側のようになっているとき，疑似カプセルの形状は図右側の黒色の形状になる．ただし，カプセルの始点と終点がどちらも厚み空間内に存在するとき，疑似カプセルの形は二次元平面でのカプセル形状と完全に同一となる．また，カプセルの始点と終点を端点とする線分が z 軸と平行であるとき，疑似カプセルの形状は円と完全に同一となる．

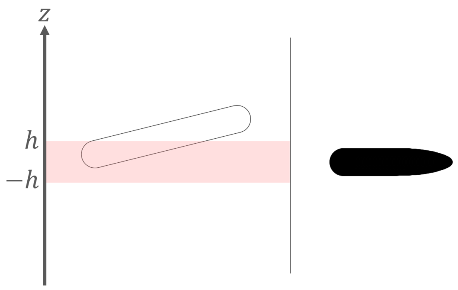


図 9　扇形の厚み空間とカプセルの位置関係からわかる疑似カプセルの形状

扇形の厚み空間とカプセル C からなる疑似カプセル α を二次元形状のカプセル C’ で囲うことで，三次元空間におけるカプセル C との干渉判定結果を近似する．そのため，カプセル C’ に囲われた空間の中で，疑似カプセル α が存在していない箇所が扇形と触れている場合，本来は干渉していないという判定結果になるはずが，干渉しているという誤った判定結果になってしまう．図 10 に疑似カプセルを二次元平面におけるカプセルで囲んだ様子を示す．

ここからは，扇形の厚み空間とカプセル C からなる疑似カプセル α をカプセル C’ で囲う方法を述べる．カプセル C’ が始点 S’ と終点 E’，半径 $r_c{}'$ からなるとしたとき，カプセル C’ が疑似カプセル α に外接するように点 S’，E’ の位置，半径 $r_c{}'$ の大きさを計算する．

始めに，半径 $r_c{}'$ の大きさを計算する．まず，点 S を中心点とする球 a と点 E を中心点とする球 b の厚み空間内での最大半径 r_s と r_e を第 2.5 節で述べた式 (12) を用いて計算する．ただし，球 a と球 b の半径はどちらも r_c である．このとき，$r_c{}'$ は式 (22) から求められる．

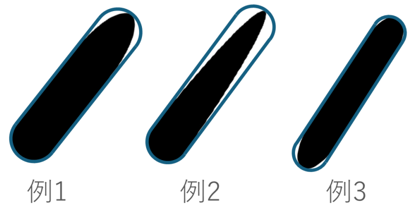


図 10　疑似カプセルを二次元平面におけるカプセルで囲んだ様子

$$r_c{}' = \begin{cases} r_c & \text{if } S_z \geq -h \text{ and } E_z \leq h \\ \max(r_s, r_e) & \text{if } S_z < -h \text{ or } E_z > h \end{cases} \tag{22}$$

次に, 点 S’, 点 E’ の位置を計算する．始めに, 点 S と点 E を端点とする線分 m の方向ベクトル $\mathbf{V}$ を $\mathbf{V} = \mathbf{E} - \mathbf{S}$ とおく．また，$\mathbf{V}$ を xy 平面に射影したベクトルを $\mathbf{V}'$ とおく．ここで，線分 m の属する直線 M 上の点を中心点とし，半径が r_c である球群からなる疑似カプセル β について考える．まず，直線 M 上の点で z 成分が h である点 Q を式 (23) から計算する．

$$\mathbf{Q} = \mathbf{S} + \frac{h - S_z}{V_z}\mathbf{V} \tag{23}$$

続いて，点 Q を xy 平面に投影した点 Q’ から，疑似カプセル β 上の点で $-\mathbf{V}'$ 方向に最も遠い点までの距離 d を計算する．点 Q’ から疑似カプセル β 上の点までの $-\mathbf{V}'$ 方向の距離は関数 $L(s)$ から求められる．式 (24) に関数 $L(s)$ を示す．ただし，変数 s は媒介変数であり，式 (25) を満たす．また，変数 w はベクトル $\mathbf{V}$ の z 成分と半径 r_c の比から式 (26) より求められる．図 11 に変数 w の模式図を示す．

$$\begin{aligned} L(s) &= sw + \sqrt{r_c{}^2 - (sr_c)^2} \\ &= sw + r_c\sqrt{1 - s^2} \end{aligned} \tag{24}$$

$$0 \leq s \leq 1\,, \tag{25}$$

$$w = \frac{r_c}{-V_z}\sqrt{V_x{}^2 + V_y{}^2}\,. \tag{26}$$

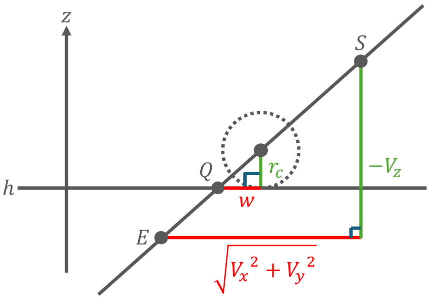


図 11　変数 w の模式図

ここで，$L(s) = d$ となるのは，関数 $L(s)$ の導関数 $L(s)'$ の値が 0 となるときである．式 (27) に導関数 $L(s)'$ を示す．$L(s)' = 0$ のとき，式 (27) は二次方程式となるが，媒介変数 s は式 (25) を満たすため，s の値は式 (28) により求められる．

$$L(s)' = w - \frac{rs}{\sqrt{1-s^2}} \tag{27}$$

$$s = \frac{w}{\sqrt{w^2 + {r_c}^2}} \tag{28}$$

求めた媒介変数 s から，直線 M 上の点 G，H の z 座標を式 (29) より算出する．ここで，点 G，H を中心点とする球 g，h は疑似カプセル β を構成する球群の中で，それぞれ $-\mathbf{V}'$ 方向，$\mathbf{V}$ 方向に最も遠い点を持つ球である．

$$\begin{aligned} G_z &= Q_z + sr_c\ , \\ H_z &= -G_z\ . \end{aligned} \tag{29}$$

求めた G_z，H_z を用いて，xy 平面に投影した際に点 S’，E’ になる可能性がある点 S”，E” を式 (30) より計算する．

$$\begin{aligned} \mathbf{S}'' &= \mathbf{S} + \frac{\min(S_z, G_z) - S_z}{V_z}\mathbf{V}\ , \\ \mathbf{E}'' &= \mathbf{S} + \frac{\max(E_z, H_z) - S_z}{V_z}\mathbf{V}\ . \end{aligned} \tag{30}$$

この時，点 S”，E” を xy 平面に投影した点を点 S’，E’ と設定すると，カプセル C’ が疑似カプセル α に外接していない場合が存在する．それは，点 S”，E” を中心点とし，半径 r_c の球 a” と球 b” の扇形の厚み空間内での最大半径 ${r_a}'$，${r_b}'$ がカプセル C’ の半径 ${r_c}'$ と等しくない時である．そこで，カプセル C’ が疑似カプセル α に外接するように，先ほど計算した点 S”，E” を $\mathbf{V}'$ 方向，$-\mathbf{V}'$ 方向に ${r_a}'$，${r_b}'$ と ${r_c}'$ の差分平行移動させてから，xy 平面に投影する．ここで，点 S”，E” を平行移動させた点をそれぞれ点 J，K とすると，点 J，K の位置ベクトルは式 (31) により求められる．

$$\begin{aligned} \mathbf{J} &= \mathbf{S}'' + \frac{{r_c}' - {r_a}'}{\sqrt{{V_x}^2 + {V_y}^2}}\mathbf{V} \\ \mathbf{K} &= \mathbf{E}'' - \frac{{r_c}' - {r_b}'}{\sqrt{{V_x}^2 + {V_y}^2}}\mathbf{V} \end{aligned} \tag{31}$$

最後に，点 J，K をそれぞれ xy 平面に投影し，点 S’，E’ を得ることで，三次元空間におけるカプセルと扇形の干渉判定を二次元平面の問題に置き換え，干渉判定結果を近似することが出来る．

2.8　干渉判定の様子

本節では提案手法をプログラム上で実装した様子を示す．扇形と対象のプリミティブ形状が干渉している場合には，対象のプリミティブ形状の色が赤色になり，非干渉の場合には青色になるようにしている．プログラムの実装には Fine Kernel Tool Kit System[15] を用いて作成した．バージョンは FK CLI 版 4.2.11.0 である．なお，各図における色のついた線分は次のような意味を持つ．

- 黒線の格子:xy 平面を表す
- 赤線:x 軸 (1, 0, 0) を表す
- 緑線:y 軸 (0, 1, 0) を表す
- 青線:z 軸 (0, 0, 1) を表す

図 12 と図 13 に第 2.2 節で述べた扇形と点の内外判定の様子を示す．また，扇形と点の内外判定も含め提案手法をプログラム上で実装した様子を以下の各 URL より閲覧できる．

- 扇形と点の内外判定:
  https://youtu.be/u_MTgTKJ6HQ
- 扇形と円の干渉判定:
  https://youtu.be/iwc4q_TEmec
- 扇形と球の干渉判定:
  https://youtu.be/n7HIlxgHcvc
- 2 次元平面における扇形とカプセルの干渉判定:
  https://youtu.be/YA2AG_lu0EY

- 3 次元空間における扇形とカプセルの干渉判定:
  `https://youtu.be/CsN7XAhJzhs`

これらの手法を活用し，提案手法を利用するコンピュータゲームを作成した．本ゲームは三次元空間中を飛び跳ねているボールに対し，野球のバットに見立てた棒を振り，バットとボールが衝突した際にボールを前方向に弾き飛ばすようになっている．提案手法をバットの回転運動に適用しており，バットを振る際には第 2.5 節で述べた球との干渉判定を行う．空間中を飛び跳ねるボールは 300 個あり，1frame 中にこれらすべてと衝突判定を行っているが，リアルタイムに実行することができている．実行の様子は `https://youtu.be/YpXgDbn2Xr4` より閲覧できる．

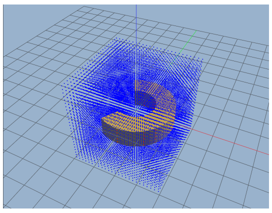


図 12　点群内に扇形が存在する様子

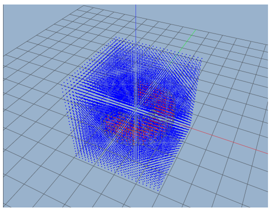


図 13　扇形内の点群の色が変化している様子

3　検証

本章では，既存手法である S-CCD 手法と扇形を用いた回転運動補間について衝突判定精度の比較結果と，提案手法の実行速度の調査結果を述べる．検証プログラムは第 2.8 節と同様の環境で作成した．

3.1　衝突判定精度の検証

本研究では，直方体が 2.1 節で述べた回転運動を行ったとき，直方体の回転運動軌跡と S-CCD 手法を適用した場合の干渉判定範囲，扇形による運動軌跡の補間を適用した場合の干渉判定範囲がどれだけ一致しているかから衝突判定精度を算出する．本検証を行うにあたり，直方体の回転運動の諸条件は次のように設定した．

- 以降で使用する疑似乱数は，C#言語の機能を用いた．
- 回転運動は原点を回転中心点とし，疑似乱数により決定される回転軸に沿って回転を行う．この時，疑似乱数により回転角度は 0 以上 2π 以下の値をとる．
- 直方体の中心点と回転中心点の距離は，疑似乱数により 12.5 以上 25 以下の値をとる．
- 直方体の横幅，高さ，奥行きは，疑似乱数により 0 以上 25 以下の値をとる．

続いて，衝突判定精度の算出方法について述べる．始めに，原点を中心に格子状に 100 万個の点を配置する．ここで，隣り合った点同士の距離が 1 になるように配置し，各点の座標の x，y，z 成分は-49.5 から 49.5 の値をとる．続いて，上述した回転運動から直方体の回転運動軌跡，S-CCD による干渉判定範囲，扇形の干渉判定範囲を計算する．直方体の回転運動軌跡は，回転運動を 100 回に分割して行うことで近似解を算出する．以降，直方体の回転運動軌跡の近似解を詳細 DCCD(Discrete CCD) 範囲，S-CCD による干渉判定範囲を S-CCD 範囲，扇形の干渉判定範囲を扇形範囲と呼称する． ここで，空間上に格子上に配置した点群全体を X とし，点群集合 P, Q, R を以下の様に規定する．

P ： X の元のうち，詳細 DCCD 範囲内にある点の集合
Q ： X の元のうち，S-CCD 範囲内にある点の集合
R ： X の元のうち，扇形範囲内にある点の集合

このとき，S-CCD 手法の衝突判定精度は式 (32) から，扇形による補間の衝突判定精度は式 (33) から求められる．絶対値記号は集合の要素数を表す．

$$\frac{|P \cap Q|}{|P \cup Q|} \tag{32}$$

$$\frac{|P \cap R|}{|P \cup R|} \tag{33}$$

また，それぞれの手法が誤判定を行う場合の違いについて分析を行う．誤判定とは，ある剛体 α と回転運動を行う剛体 β が存在するとき，剛体 α と剛体 β の回転運動軌跡の衝突判定結果と，剛体 β の回転運動に対して CCD 手法を適用した際の衝突判定結果が異なることである．誤判定には過剰判定と不足判定の 2 つの種類が存在する．以下に，2 つの誤判定の違いを示す．

- 過剰判定 : 剛体 α と剛体 β 回転運動軌跡は衝突していないが，CCD 手法を適用した場合に衝突していると判定してしまう場合
- 不足判定 : 剛体 α と剛体 β 回転運動軌跡は衝突しているが，CCD 手法を適用した場合に衝突していな

いと判定してしまう場合

衝突判定精度と同様に，S-CCD 手法と提案手法の過剰判定率，不足判定率を算出する．S-CCD 手法の過剰判定率は式 (34) から，不足判定率は式 (35) から算出できる．

$$\frac{|Q-P|}{|P\cup Q|} \tag{34}$$

$$\frac{|P-Q|}{|P\cup Q|} \tag{35}$$

また，提案手法の過剰判定率は式 (36) から，不足判定率は式 (37) から算出できる．

$$\frac{|R-P|}{|P\cup R|} \tag{36}$$

$$\frac{|P-R|}{|P\cup R|} \tag{37}$$

10 万件の直方体の回転運動データから，S-CCD 手法と提案手法の平均衝突判定精度，平均過剰判定率，平均不足判定率を算出する．ただし，式 (33) か式 (32) のどちらか一方でも分母の値が 0 になるようなデータは含まれない．

次に，回転角度による衝突判定精度の違いと，直方体の形状による衝突判定精度の違いについて調べる．回転角度による衝突判定精度の違いは，検証プログラムより得られた 10 万件のデータを回転角度を 10 度ずつ区切った 36 個の回転角度範囲に分類し，それぞれの回転角度範囲ごとに平均衝突判定精度を算出した．ただし，角度範囲ごとのデータ数はいずれも 2600 件から 2900 件の間であった．直方体の形状による衝突判定精度の違いは，まず，検証プログラムより得られた 10 万件のデータそれぞれに対し，直方体の縦幅から横幅を引いた縦横差を算出する．ここで直方体の縦幅とは，第 2.1 節で述べた直方体の厚みのうち $\mathbf{Y}'$ 方向の厚みである W_y のことを指し，横幅とは $\mathbf{X}'$ 方向の厚みである W_x のことを指す．算出した縦横差から 10 万件のデータを昇順に並べ，5000 件ごとに平均衝突判定精度を算出した．

3.2 衝突判定精度の比較

表 1 に各手法の衝突判定精度の検証結果を示す．

表 1 から，提案手法のほうが S-CCD 手法よりも 63.41% 高精度で衝突判定を行えていることが判明した．また，それぞれの手法が誤判定を行った際，S-CCD 手法は過剰判定を行う確率が高く，提案手法は不足判定を行う確率が高いことが判明した．

表 1　各手法の衝突判定精度と誤判定率

	S-CCD	提案手法
平均衝突判定精度	4.27%	67.68%
平均過剰判定率	93.90%	1.25%
平均不足判定率	1.83%	31.07%

次に，図 14 に S-CCD 手法，提案手法の回転角度範囲ごとの衝突判定精度を折れ線グラフにまとめた図を示す．

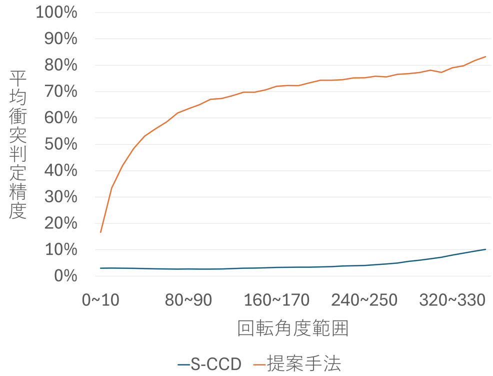


図 14　回転角度範囲ごとの衝突判定精度

図 14 から，どの回転角度範囲においても提案手法は S-CCD 手法よりも平均衝突判定精度が高いことが判明した．提案手法の衝突判定精度のみに注目すると，回転角度が大きくなるほど衝突判定精度も高くなる傾向にあることが判明した．

最後に，図 15 に S-CCD 手法，提案手法の直方体の形状の違いによる衝突判定精度を折れ線グラフにまとめた図を示す．

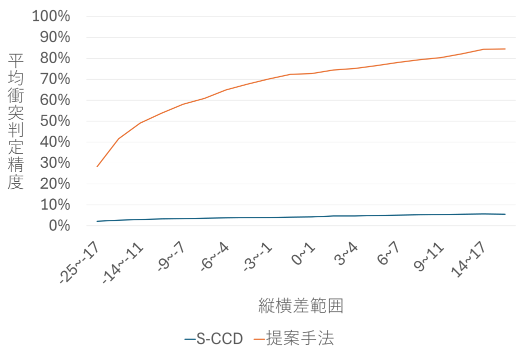


図 15　直方体の縦横差ごとの衝突判定精度

図 14 から，直方体の形状に関わらず提案手法は S-CCD 手法よりも平均衝突判定精度が高いことが判明した．提案手法の衝突判定精度のみに注目すると，縦幅のほうが横幅よりも大きくなるにつれ平均衝突判定精度も高くなっていることが判明した．

3.3　実行速度の検証

本研究では，第 2 章で提案した扇形とプリミティブ形状との干渉判定の実行速度を調査し，DCD 手法の一例である境界ボリューム手法と，CCD 手法の一例である S-CCD 手法との比較を行った．また，提案手法は衝突判定システムにおける Broad Phase と Narrow Phase[6] のうち，Narrow Phase にて使用することを想定している．そこで，Narrow Phase における CCD 手法であり，S-CCD 手法同様に角運動に対処している DCCD 手法とも実行速度の比較を行った．

始めに，DCD 手法の一例である境界ボリューム手法との実行速度の比較方法を述べる．境界ボリュームは次の 4 つの形状の干渉判定手法と提案手法の実行速度の比較を行った．

- 球
- カプセル
- AABB
- OBB

実行速度の計測では，各手法を 10 万回連続で実行したときの実行速度を計測した．各形状の形や大きさ，向きは C#言語の疑似乱数機能を用いて決定した．ただし，扇形は第 2.1 節で述べた，判定数式を簡易化するための条件を必ず満たすように扇形の回転中心点 O，回転軸 **U**，中心軸 **A** の値は設定した．そのため，座標変換は行っていない．各手法の実行速度データを 1 万件ずつ集計し比較を行った．

次に，CCD 手法の一例である S-CCD 手法との実行速度の比較方法を述べる．CCD 手法の実行速度計測では，第 3.1 節で述べた方法により決定される直方体の回転運動に対して CCD 手法を適用したときに，C#言語の疑似乱数機能によって決定された中心点の位置，半径を持つ 10 万個の球と衝突判定を行ったときの実行速度を計測する．ただし，DCD 手法の実行速度の計測とは違い，直方体の回転運動から各手法によって運動軌跡の補間範囲を算出する時間も含み，第 3.1 節で述べた直方体の回転運動では回転軸は疑似乱数機能により決定されるため，提案手法の実行速度の計測では球の中心点に対する座標変換にかかる時間も含む．そこで，実際に衝突したかどうかを計算する座標変換と干渉判定のみを行ったときの実行速度と，運動軌跡の補間形状の計算のみを行ったときの実行速度，どちらも同時に行ったときの実行速度の 3 つの実行速度を 1 万件ずつ集計し比較を行った．

最後に，Narrow Phase における CCD 手法であり，角運動に対処している DCCD 手法との実行速度の比較方法を述べる．DCCD 手法は第 3.1 節にて運動軌跡の近似解として使用した詳細 DCCD と同じく，1frame での運動を複数回に分割して行うことで高精度の衝突判定を行う手法である．DCCD 手法は運動の分割回数が大きければ大きいほど衝突判定精度を高めることが出来るが，同時に実行速度も遅くなっていく．そこで，まず，分割回数がいくらの時に DCCD 手法の平均衝突判定精度が提案手法と同等になるかを調査した．DCCD 手法の平均衝突判定精度は，第 3.1 節で述べた S-CCD 手法と提案手法の衝突判定精度の算出方法と同様に，詳細 DCCD 範囲と DCCD 手法の干渉判定範囲がどれだけ一致しているかから算出した．ただし，平均衝突判定精度は各分割回数ごとに 1 万件のデータから算出を行った．また，調査した DCCD 手法の分割回数は，2 回から 10 回の 9 件である．DCCD 手法の分割数と平均衝突判定精度の関係を図 16 に示す．

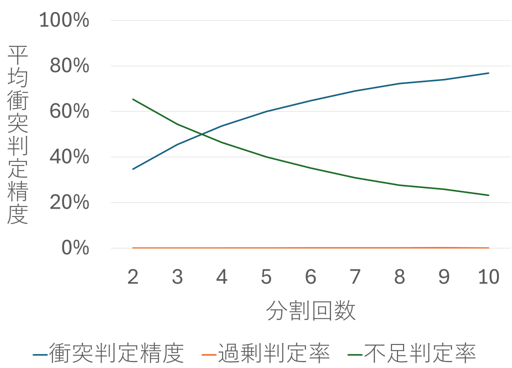


図 16　DCCD 手法の分割数と衝突判定精度の関係

第 3.2 節の結果から提案手法の平均衝突判定精度は 67.68% であるため，分割回数が 7 回の DCCD 手法が提案手法と同等の平均衝突判定精度であることがわかる．そこで，分割回数が 6，7，8，回の DCCD 手法と提案手法の実行速度を比較する．DCCD 手法の実行速度計測

では，前述した S-CCD 手法と提案手法の実行速度計測と同じ条件をとる．ただし，定めた分割回数よりも前の時点で衝突を検知した場合，DCCD 手法ではその後の処理を行わない．例えば，分割回数が 8 回で，4 回目の分割の時点で衝突を検知した場合には，5，6，7，8 回目の分割と干渉判定は行わない．また，球との干渉判定には，球と OBB の干渉判定を使用した．

実行速度の検証プログラムは，以下の環境で実行した．

- OS : Windows 11
- CPU : Intel(R) Core(TM) i9-10900K CPU @ 3.70 GHz
- メモリ : 32GB

3.4 実行速度の比較

本節に記載されている実行速度はすべて，単位は ms(ミリセカンド) である．

まず，各 DCD 手法の実行速度の箱ひげ図を図 17 と図 18 に示す．

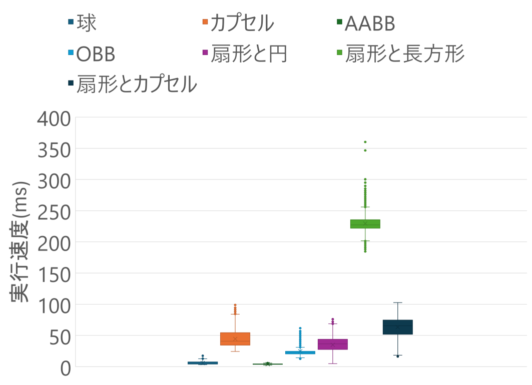


図 17 二次元平面における DCD 手法の実行速度 (ms) の箱ひげ図

まず，二次元の手法に注目する．扇形と円は平均速度ではカプセル同士の干渉判定手法よりは高速であることが判明した．ただし，最低速度を見ると，カプセルのほかに OBB 同士の干渉判定よりも実行速度は優れていることが判明し，リアルタイムでの実行に十分に利用できることが判明した．続いて，扇形と長方形に注目すると，どの手法よりも非常に実行速度が遅いことが判明した．少ない数の剛体同士の判定であれば，リアルタイムでの実行は問題ないが，非常に多くの剛体同士の干渉判定として利用する事は難しい可能性があることが判明した．

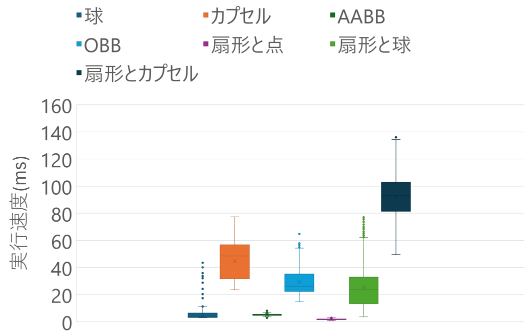


図 18 三次元空間における DCD 手法の実行速度 (ms) の箱ひげ図

最後に，扇形とカプセルに注目すると，概ね扇形と円と同じ評価を下すことが出来ることが判明した．ただし，扇形と円よりはどの値においても遅い実行速度であることが判明した．

次に，三次元の手法に注目する．まず扇形と点に注目すると，どの値においても非常に高速であり，球同士の干渉判定や AABB 同士の干渉判定よりも高速に内外判定を行えていることが判明した．次に扇形と球を見ると，扇形と円の時とは違いカプセル同士の干渉判定手法のほかに，OBB 同士の干渉判定手法よりも高速であるということが判明した．最後に扇形とカプセルに注目すると，どの手法よりも実行速度が遅いことが判明した．ただし，二次元の手法である扇形と長方形ほど遅いわけではなく，リアルタイムでの実行には問題ないことが判明した．

続いて，S-CCD 手法と提案手法の座標変換と干渉判定のみを行ったときの実行速度を箱ひげ図 19 に，運動軌跡の補間形状の計算のみを行ったときの実行速度を箱ひげ図 20 に，どちらも同時に行ったときの実行速度を箱ひげ図 21 に示す．

始めに，座標変換と干渉判定のみを行ったときの実行速度に注目すると，S-CCD 手法に比べ提案手法の平均速度は約 100ms ほど遅い実行速度であることが判明した．次に，運動軌跡の補間形状の計算のみを行ったときの実行速度に注目すると，座標変換と干渉判定のみを行ったときの実行速度とは逆に，提案手法のほうが S-CCD 手法よりも非常に高速であることが判明した．最後に，どちらも同時に行ったときの実行速度に注目すると，S-CCD 手法よりも提案手法のほうが高速である

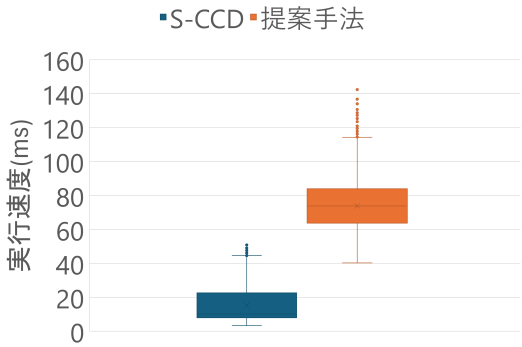


図 19　座標変換と干渉判定のみを行ったときの提案手法と S-CCD 手法の実行速度 (ms) の箱ひげ図

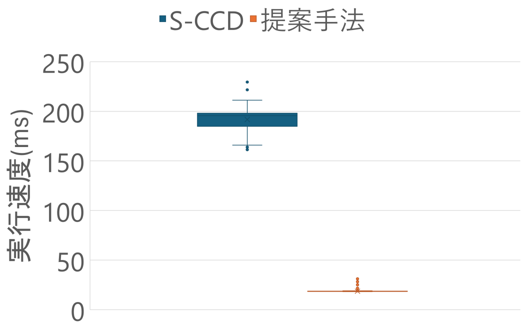


図 20　運動軌跡の補間形状の計算のみを行ったときの提案手法と S-CCD 手法の実行速度 (ms) の箱ひげ図

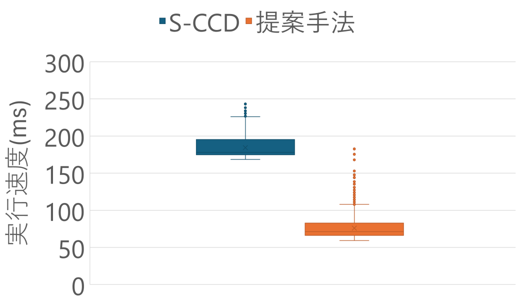


図 21　提案手法と S-CCD 手法の実行速度 (ms) の箱ひげ図

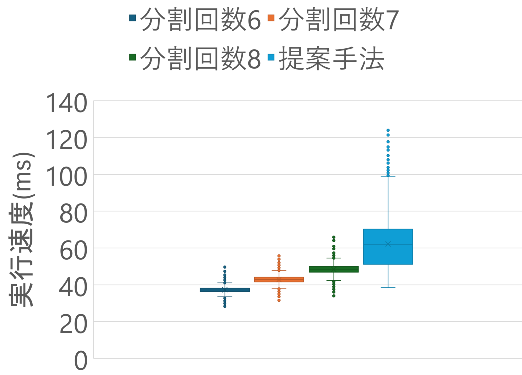


図 22　座標変換と干渉判定のみを行ったときの提案手法と DCCD 手法の実行速度 (ms) の箱ひげ図

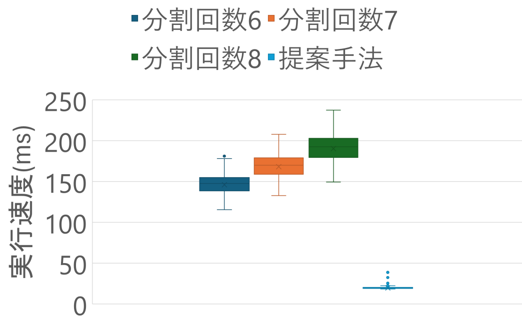


図 23　運動軌跡の補間形状の計算のみを行ったときの提案手法と DCCD 手法の実行速度 (ms) の箱ひげ図

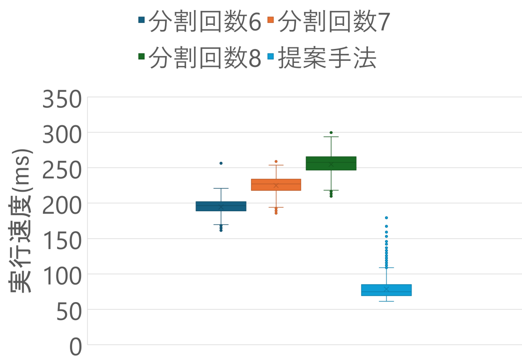


図 24　提案手法と DCCD 手法の実行速度 (ms) の箱ひげ図

ことが判明した．このことから，S-CCD 手法の実行速度の大部分が運動軌跡の補間形状であるのに対し，提案手法の実行速度の大部分は座標変換と干渉判定であることが判明した．

最後に，DCCD 手法と提案手法の座標変換と干渉判定のみを行ったときの実行速度を箱ひげ図 22 に，運動軌跡の補間形状の計算のみを行ったときの実行速度を箱ひげ図 23 に，どちらも同時に行ったときの実行速度を箱ひげ図 24 に示す．

始めに，座標変換と干渉判定のみを行ったときの実行速度に注目すると，提案手法の平均速度はどの分割回数の DCCD 手法と比較しても遅いことが判明した．また，平均衝突判定精度が同等である分割回数 7 回の DCCD 手法とは，約 20ms ほどの実行速度の差があることが判明した．次に，運動軌跡の補間形状の計算のみを行ったとき，どちらも同時に行ったときの実行速度に注目する

と，提案手法の実行速度については S-CCD 手法との比較結果と概ね同様の評価を下せることが判明した．

4　考察

4.1　衝突判定精度について

本論文で提案する扇形を用いた回転運動の補間は，第 3.2 節で述べた通り，S-CCD 手法よりも高い平均衝突判定精度である．また，回転角度が大きくなるにつれ，平均衝突判定精度も上昇する手法である．ただし，前述の結果は言い換えると回転角度が小さい時は平均衝突判定精度が低いということでもある．また，直方体の横幅が縦幅よりも大きい時に平均衝突判定精度が低下することが判明した．

これらの原因として，直方体の回転運動から扇形を構築する際に扇形の大きさが不適切であることが挙げられる．提案手法では回転前の直方体の中心から回転後の直方体の中心を補間するように扇形を構築する．そのため，回転角度が小さい時には構築される扇形も細長いものとなり，扇形によって補間される回転運動軌跡も少ないものとなってしまう．また，横幅が縦幅よりも大きい直方体が回転運動を行う際，回転運動軌跡は扇形との差異が大きくなる．図 25 に提案手法で衝突判定精度が低下する際の回転運動と構築される扇形の様子を示す．

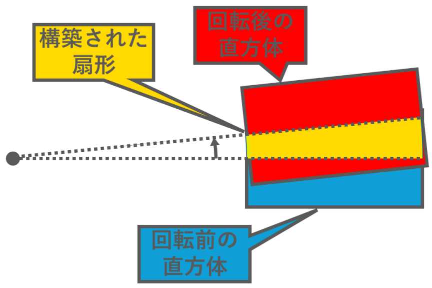


図 25　提案手法で衝突判定精度が低下する際の回転運動と構築される扇形の様子

この問題を解決するためには，直方体の回転運動から扇形を構築する方法を変える必要がある．提案手法では，扇形を構築する際に直方体の横幅を考慮していない．そこで，扇形を構築する際に直方体の横幅も考慮して扇形の形状を決めることが出来れば，前述の場合でも衝突判定精度を高くできる可能性がある．

ただし，横幅が縦幅よりも大きい直方体が高速に回転運動を行った場合，本研究で想定している縦幅が横幅よりも大きい直方体が高速に回転運動を行った場合に比べて剛体同士が衝突を無視してすり抜ける確率は低いと考える．そのため，本手法は第 2.1 節で述べた想定通り，縦幅が横幅よりも大きい直方体の回転運動に対して使用するのが望ましいと考える．縦幅が横幅よりも大きい直方体が高速に回転運動をするコンピュータゲームの例としてはバットや剣などの細長い剛体が登場するものが挙げられ，野球ゲームや，アクションゲーム，VR(Virtual Reality) 剣戟ゲームなどがある．

4.2　実行速度について

第 3.4 節の検証結果より，扇形と長方形の干渉判定は他手法と比較して非常に遅い実行速度であることが判明した．その原因として線分と線分の交差点算出など基本幾何計算を多用していることが挙げられる．提案手法の他の手法ではプリミティブ形状特有の数学的性質を利用している部分が多いのに対し，長方形との干渉判定では線分という幾何学的要素に対する処理を元に干渉判定を行っている．幾何学的要素に対する処理は簡易な処理ではあるものの，剛体同士の干渉判定などに利用すると何度も処理を行う必要があり，結果として実行速度が低下しやすい．そのため，長方形の数学的性質をうまく活用することが出来れば，より高速に干渉判定を行うことが出来る可能性がある．

続いて，S-CCD 手法との実行速度の比較について考察を行う．実行速度の比較結果から，提案手法の実行速度が高速だったのは運動軌跡の補間形状である扇形を構築するのが高速であったためだとわかる．ただし，第 2.1 節で述べた通り，直方体の回転運動には制約を課しており，さらに，第 4.1 節で述べた通り，直方体の回転運動から扇形を構築する際には直方体の横幅は考慮していない．そのため，第 2.1 節で述べた制約を満たさない場合でも扇形が回転運動を補間できるような扇形の構築方法などの場合には，扇形を構築する際により時間がかかる可能性がある．また，一般的な CCD 手法では，衝突していることが判明した後に，衝突点の計算や干渉解決，剛体の運動速度を考慮した撃力計算などの衝突応答と呼ばれる処理を行うが，現在の提案手法では衝突判定のみで，衝突応答を行うことはできない．よって，提案手法を用いて衝突判定を行った後に衝突応答を行う場合，本研究で比較を行った S-CCD 手法や分割回数が 6，7，8

回の時の DCCD 手法よりも一連の処理の合計では遅くなる可能性がある.

5 まとめ

本論文では回転運動軌跡の近似形状である扇形を定義し，扇形を用いて回転運動を補間する CCD 手法を提案した．また，コンピュータゲームで頻出するプリミティブ形状と扇形との干渉判定を提案した．検証から提案手法は回転運動に対応した CCD 手法の一例である S-CCD 手法よりも高い衝突判定精度を示し，リアルタイムでの実行に概ね問題ない実行速度であることが判明した．提案手法は剣や野球ゲームのバットのような細長い剛体が回転運動をする際に衝突判定精度が高精度になるため，野球ゲームや剣などが登場するアクションゲーム，VR 剣戟ゲームにおいて有効に活用できると考えられる．ただし，本研究では回転運動に対して制約を課しているため，より多くの場面で提案手法を活用できるようにするためには，多種多様な剛体の回転運動から扇形を構築できるように改善する必要がある.

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] Zhou xiangning. Research of collision detection based on obb in skinned mesh. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Vol. 6, pp. V6–643–V6–645, 2010.

[4] 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.

[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] 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.

[7] Quan Nie, Yingfeng Zhao, Li Xu, and Bin Li. A survey of continuous collision detection. In *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pp. 252–257, 2020.

[8] Stephane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum*, Vol. 21, , 05 2002.

[9] 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.

[10] 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.

[11] 山本輝, 阿部雅樹, 渡辺大地. リアルタイムグラフィックスにおける回転剛体の衝突判定精度向上に関する研究. In *NICOGRAPH2023*, pp. F–6:1 – F–6:8, 2023.

[12] Shankar Krishnan, Amol Pattekar, Ming C. Lin, and Dinesh Manocha. Spherical Shell: A Higher Order Bounding Volume for Fast Proximity Queries. In *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics : The Algorithmic Perspective: The Algorithmic Perspective*, WAFR ’98, pp. 177–190, USA, 1998. A. K. Peters, Ltd.

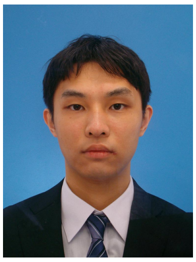
[13] KIT 物理ナビゲーション. ロドリゲスの

回転公式 (Rodrigues' rotation formula). `https://w3e.kanazawa-it.ac.jp/math/physics/category/physical_math/linear_algebra/henkan-tex.cgi?target=/math/physics/category/physical_math/linear_algebra/rodrigues_rotation_formula.html`. 参照: 2023.2.6.

[14] 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.

[15] Fine Kernel Project. Fine kernel toolkit system. `https://gamescience.jp/FK/`. 参照: 2023.8.20.

山本 輝



2022 年東京工科大学メディア学部卒業．同年より同大学大学院修士課程バイオ・情報メディア研究科メディアサイエンス専攻在籍．芸術科学会会員．

阿部　雅樹

2008 年東京工科大学メディア学部卒業．2010 年同大学大学院修士課程バイオ・情報メディア研究科メディアサイエンス専攻修了．2016 年より同大学メディア学部実験助手，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会会員．

渡辺　大地

1994 年慶応義塾大学環境情報学部卒業．1996 年慶応義塾大学政策・メディア研究科修士課程修了．2016 年岩手大学工学研究科デザイン・メディア工学専攻博士後期課程修了．博士 (工学)．1999 年より東京工科大学メディア学部講師．2017 年より同准教授，2020 年より同教授，現在に至る．コンピュータグラフィックスやゲーム制作に関する研究に従事．芸術科学会，情報処理学会，画像電子学会，人工知能学会会員．