ポリゴンと直交格子とのロバストな交差判定法について

徳永　健一*

A Robust Method for the Detection of Triangle-Cube Intersection

Ken-ichi Tokunaga*

埋め込み境界法などで必要になる 3 次元空間におけるポリゴンと直交格子との間の交差判定ルーチンは、単純な実装では通常は数値誤差については許容値を用いて処理している。本稿では、本質的に許容値を用いずにロバストに処理する方法を紹介する。

Keywords: Computational Geometry, Immersed Boundary, Intersection Testing

1. 問題設定と背景

3 次元空間内の三角形の集合で定義されるポリゴンデータが直交格子にどのように埋め込まれているかという問題は、流体解析における埋め込み境界法、カットセル法、ボクセルデータの生成、などでよく現れる。問題自体は単純であり、直交格子のセルに三角形が交差するかどうかの判定法は単純な実装も考えられる。

ただし複雑な形状に対して、目的の交差結果を得るのは意外と難しい。特にポリゴンの頂点がちょうど直交格子の面や辺上にある場合や、三角形の法線ベクトルが 3 次元空間の座標軸と平行になっている場合などは、交差判定の不等式が 0 に近い値の符号を評価することになり、わずかな数値誤差で判定結果が変わることになる。

交差判定を緩くする（交差していると判定されるセルを多めに見積もる）と、本来隙間があるところが埋まってしまうことがある。逆に厳しくする（交差していると判定されるセルを少なめに見積もる）と繋がっているべきところが孤立してしまったりする。

そこで多く用いられるのは、交差判定に現れる不等式について、許容値を与えて厳しさを調整し、目的に応じて許容値を変える方法である。しかしながら、この方法では、適切な許容値を探すのにコストがかかることになる。

ここではこれらの問題を解決するために新たに開発した本質的に許容値を与えずにロバストに処理する方法を紹介する。この方法は当社のプリポストプロセッサ Advance/REVOCAP で実際に用いられている方法である。

2. 入出力仕様

入力データは 3 次元空間における

- 三角形の集合であるポリゴンデータ
- 直交格子

である。直交格子は等間隔であることは仮定しない。

出力データは入力データ同じ直交格子のそれぞれのセルに

- ポリゴンが交差しているかどうか
- 交差している場合はそのポリゴンの識別子、交差面積

の情報を付加したものである。

実装は、例えばポリゴンデータは STL 形式、直交格子は VTK の Structured Points 形式で与えられることを想定している。

以下の議論では、三角形はポリゴンデータに属しており、セルは直交格子の立方体（等間隔でない場合は直方体）に属していることとする。

* アドバンスソフト株式会社　第 1 事業部
Computational Science and Engineering Division Ⅰ, AdvanceSoft Corporation

3. 既存のよくある実装方法

ここではいくつかの実装例（よくないものも含む）を紹介する。

3.1. 三角形の頂点で評価する方法

直交格子のセルに三角形の頂点が含まれているかで判定する。

この方法は、素人のソースコードによく見られる。 すぐ分かるようにセルの大きさに比べて十分大きな三角形を考えると、頂点から遠い三角形の内部しか含まれないセルは、交差しないと判定される。

この方法は交差判定を厳しくした判定法であり、 すべての三角形がセルの大きさよりも十分小さい場合にしか正しくない。

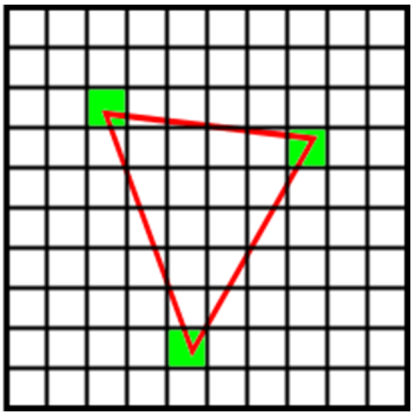

図 1 三角形の頂点で評価する方法

3.2. 三角形の Bounding Box で評価する方法

三角形の頂点をすべて含み、すべての辺が座標軸に平行な直方体で最小のものを三角形の Bounding Box と呼ぶ。Bounding Box と直交格子のセルが交わっている場合に交差すると判定する。

この方法は、実際には三角形は交差していなくても、Bounding Box と交差するような直交格子のセルが現れるので、交差判定を緩くした判定方法になる。

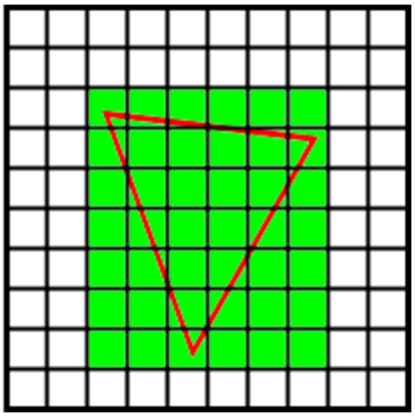

図 2 三角形の BoundingBox で評価する方法

3.3. 三角形とセル中心の間の距離で判定する方法

3 次元空間内の点と点との距離が定義されているとする。

線分（2 つの点に挟まれた直線の一部分）と点との距離は、直線への垂線の足が線分上にある場合は、その垂線の長さで、そうでないときは、両端の点との距離の小さいほうの値である。

三角形と点との間の距離は、三角形を含む平面への点の垂線の足が三角形の内部にあるときはその垂線の長さである。そうでないときは、三角形の 3 辺と点との距離の最小値である。

セル中心と三角形の間の距離が、セルの長さの半分以下の時に交差していると判定する場合は、セルの頂点の近傍でのみ交差している三角形がセル中心との距離はセルの長さの半分より大きくなって交差していないと判定されるため、交差判定を厳しくした判定法になる。

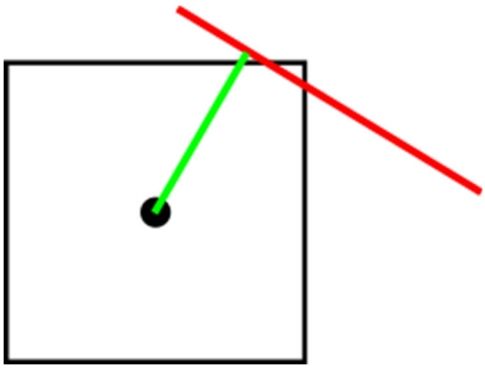

図 3 三角形とセル中心の距離で評価する方法（厳しくなる場合）

セル中心と三角形の間の距離が、セルの対角線の長さの半分以下の時に交差していると判定する場合は、面の近傍で平行している三角形は実際に交差していないが、交差していると判定されるので、交差判定を緩くした判定法になる。

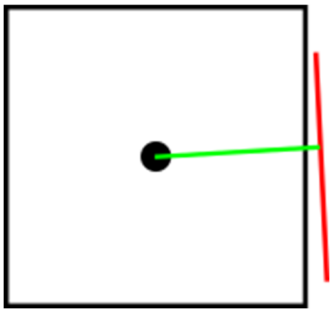

図 4 三角形とセル中心の距離で評価する方法（緩くなる場合）

3.4. 三角形とセルの距離の最小値を評価する方法

三角形上の点を 2 変数で表現し、セル上の点を 3 変数で表現して、三角形とセルの間の距離をこれらの 5 変数の関数として、定義域のなかに 0 を取るものがある場合は、交差していると判定する。

これは数値誤差を除けば、数学的に厳密な方法である。ただし三角形、セルを表すための定義域の条件が不等式になり、最小値問題を解くには煩雑な不等式に関する場合分けが必要となるため、実装は容易ではない。

3.5. Möller の方法

これは CG の分野のレイトレーシングでよく用いられている定評のある方法である。これも数値誤差を除けば、数学的に厳密な方法である。

3 次元空間内の 2 つの交差しない凸図形には、2 つを分離する面（すなわち、2 つの図形が面の異なる側にある）が存在する。

2 つの凸多面体が交差しないことは、次のベクトルを法線に持つ面のいずれかで分離することが必要十分である。

1. それぞれの多面体の法線ベクトル
2. 互いの多面体の辺同士の外積

三角形とセル（六面体）について当てはめると、以下の 13 個のベクトルについて、それらと直交する面で分離するかどうかを判定すればよい。

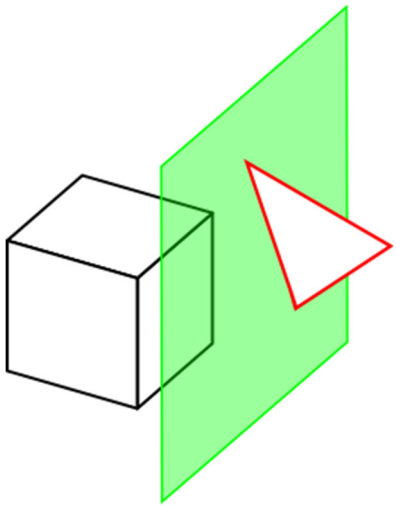

図 5 三角形と六面体が平面で分離する場合

1. セルの法線ベクトルとして、座標軸に平行な単位ベクトル。（3 通りのテスト）
2. 三角形の法線ベクトル。（1 通りのテスト）
3. セルの辺ベクトルとしての座標軸に平行な単位ベクトルと、三角形の辺のベクトルの外積ベクトル。（$3 \times 3 = 9$通りのテスト）

面で分離するかどうかのチェックは、法線ベクトル方向に射影して得られる三角形とセルの区間に交わりがないかをチェックすることでなされる。

実装では、射影の計算（内積）と交わりがないかの不等式の判定で、数値誤差の影響があることに注意しておく。

4. 本稿で紹介する方法

4.1. アルゴリズム

三角形ごとに交差しているセルを判定する。

1. 三角形の Bounding Box を計算する。
2. 三角形の Bounding Box と交わりを持つような直交格子の部分格子を計算する。
3. 三角形の Bounding Box が 1 つのセルに含まれている場合（上記の部分格子が 1 つのセルからなる場合）は、そのセルを交差セルとしてマークしてこの三角形に対する処理は終了する。
4. 複数のセルに含まれる場合（部分格子が複数のセルからなる場合）、座標軸方向に関して最も長い方向で部分格子をセル単位で 2

分割する。分割する方向のセル数が奇数の場合はどちらかを 1 つ大きくする。

5. セルを分割する面で元の三角形を分割する。分割した片方が四角形になる場合は、対角線で分割して三角形にする。
6. 分割した三角形に対して、再び交差判定処理をする。

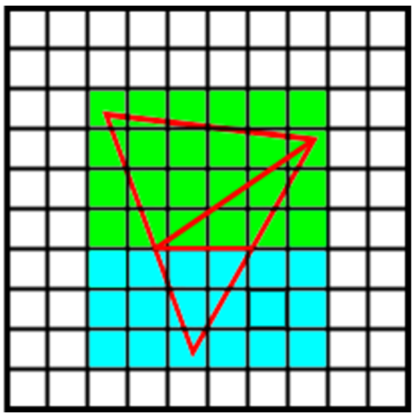

図 6 部分格子を分割した場合

三角形を分割していけば、最後には必ず 1 つのセルに含まれる大きさになるまで小さくなるので、この繰り返しは必ず有限回で終了する。

4.2. ロバスト性

上記の処理に不等式は

- 三角形の Bounding Box が含まれる直交格子のセルを計算する

部分にしか現れない。これは内積などの数値演算した値に対する比較ではなく、三角形の頂点の座標値の最大値または最小値に対する比較である。これには数値誤差による許容値を与える必要はない。また、三角形の Bounding Box は空集合にはならないので必ずいずれかのセルと交わりを持つため、それぞれの三角形は必ず 1 つ以上の直交格子のセルと交差する。

3 節で紹介した手法のうち、数値誤差のため許容値を調整するもの（3.3、3.4、3.5）では、交差判定条件を厳しくすると、三角形に対して交差するセルが現れない可能性を排除できない。

以上をまとめると、

- 判定条件に数値誤差を考慮する許容値を入れる必要がない
- 三角形に対して必ず 1 つ以上のセルが交差する

性質を持つことが分かり、この意味でロバストである。

4.3. 計算時間のコスト

数学的に厳密な手法として Möller の方法を対象として比較して計算コストを概算する。

Möller の方法も本稿の方法も 1 つの三角形に対して、本質的に判定条件を計算しなければならないのは、三角形の Bounding Box を含むようなセルである。このセル数は同じである。

- Möller の方法：

  セル数 × 三角形数 × 13 × 分離チェック

  （分離チェック = 三角形と六面体の頂点の射影計算 + 不等式）
- 本稿の方法：

  三角形数 × 三角形の細分 ×（Bounding Box の計算 + セル座標との比較）

  （セルのサイズになるまでの三角形の細分 ≤ 2 ×セル数）

必要な三角形の細分の回数は三角形とセルの大きさに依存するが、少なくともセル数の 2 倍（四角形になったときに対角線で三角形化する可能性があるため）以下である。三角形の細分は辺の内分点の座標を計算するだけである。

本質的な計算量（セル数と三角形数のオーダー）で考えると、Möller の方法の計算量と本稿の方法はほぼ同等（どちらもセル数と三角形数に対して線形）である。

5. 計算例

5.1. 実装

C++ 言語で実装し、Visual Studio 2017 でコンパイルしたものを用いた。

5.2. Stanford Bunny

CG の分野で標準的に用いられる Stanford

Bunny のモデルについて、背景格子を 200×200×200 として、交差したセルを可視化した。

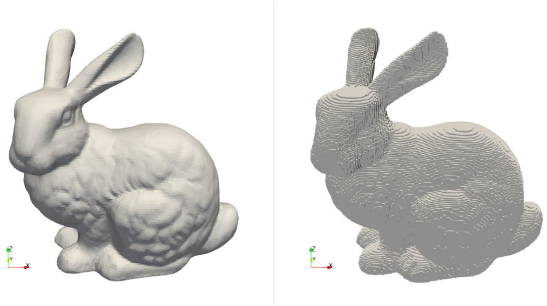

図 7 Stanford Bunny の交差判定結果

表 1 Stanford Bunny の処理時間

項目	値	備考
モデルポリゴン数	69664	
背景格子セル数	800 万	
延べ交差格子数	1294150	異なる三角形が同一のセルと交差する場合は重複して数えた
処理時間[s]	0.434	Core i5 2.8GHz 並列処理なし

5.3. Happy Budda

CG の分野で標準的に用いられる Happy Budda のモデルについて、背景格子を 200×400×200 として、交差したセルを可視化した。

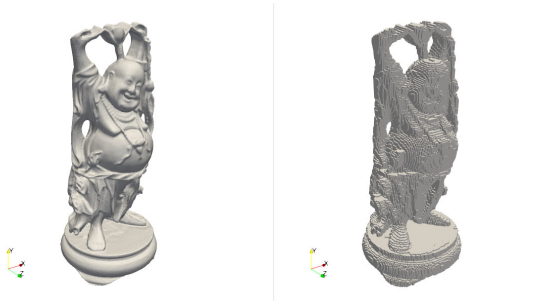

図 8 Happy Budda の交差判定結果

表 2 Happy Budda の処理時間

項目	値	備考
モデルポリゴン数	1087716	
背景格子セル数	1600 万	
延べ交差格子数	6332530	異なる三角形が同一のセルと交差する場合は重複して数えた
処理時間[s]	2.83	Core i5 2.8GHz 並列処理なし

6. まとめ

三角形が直交格子に交差するかどうかの判定方法について、ロバストな方法を提案した。処理は十分高速であることを確認した。

参考文献

[1] Tomas Möller and Ben Trumbore, Fast Minimum Storage Ray-Triangle Intersection, Journal of Graphics Tools Volume 2 Issue 1, 1997 pp.21-28

[2] Tomas Akenine-Möller, Fast 3D Triangle-Box Overlap Testing, Journal of Graphics Tools Volume 6, 2001, pp.29-33, https://fileadmin.cs.lth.se/cs/Personal/Tomas_Akenine-Moller/pubs/tribox.pdf

[3] Doug Voorhies, Triangle-Cube Intersection, Graphics Gems III, pp.236-239, http://www.realtimerendering.com/intersections.html

※ 技術情報誌アドバンスシミュレーションは、アドバンスソフト株式会社 ホームページのシミュレーション図書館から、PDF ファイルがダウンロードできます。（ダウンロードしていただくには、アドバンス/シミュレーションフォーラム会員登録が必要です。）