レイトレーシングによるコンピュータグラフィクス入門
- メタボール -

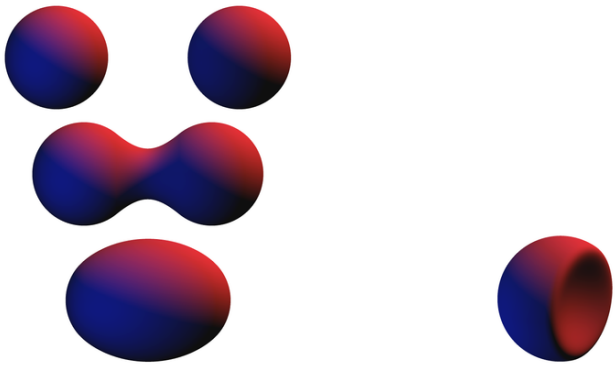
金森 由博*

2014 年 7 月 4 日

1　メタボール

CG で有機的で滑らかな形状を表現するには、**メタボール** (*metaball*) というものが使える。三角形メッシュに比べ少ない入力データで形状を表現できることから、メモリが貴重だった時代に人体や動物などの有機的な形状を表現するために用いられた。最近では、大量の粒子を使った物理シミュレーションの結果を表示する際にメタボールが用いられている。典型的な利用例としては、分子の挙動を計算したシミュレーション結果の電荷分布を可視化したものや、水などの液体の液面を表現したものなどがある[*1]。

メタボールについてもう少し詳しく説明する。3 次元空間中のある点について、その点から同心球状に、中心が濃く次第に薄くなるような密度分布を与える。この密度分布に対して、密度がある一定の値になっているところ、すなわち**等値面** (*isosurface*) を取り出すと、球が得られる。このような密度分布を持つ点を 2 つ、互いに近いところに置くと、それらの密度分布が足し合わさった密度分布が得られる。ここで同じように等値面を考えると、今度はひょうたんのような形が得られる (図 1(a))。密度分布は負の値を持っていてもよく、その場合、等値面はひょうたん型ではなく、つぶれたような形になる (図 1(b))。メタボールとは、このような球状の密度分布を持つ点、あるいはそれらの集合からなる密度分布の等値面のことをいう。



(a) 正の密度を持つメタボール　(b) 片方が負の密度を持つメタボール

図 1　メタボール. 画像は Wikipedia から転載.

* kanamori@cs.tsukuba.ac.jp

*1 なお液体に用いる場合には、液面を平坦にするために個々のメタボールを楕円体状に変形して用いることが多くなっている。

1.1 メタボールの定式化

では、メタボールの曲面が数式でどのように定義されるか説明する。メタボールは、数学的には陰関数曲面と呼ばれる曲面の一種である。例えばパラメトリック曲面では $x = x(u, v), y = y(u, v), \cdots$ のように曲面上の点 $\mathbf{x} = (x, y, z)^T$ が陽に与えられるのに対し、陰関数曲面上の点 $\mathbf{x}$ は、次の関係式を満たす点である、という風に陰に与えられる。

$$f(\mathbf{x}) = 0 \tag{1}$$

陰関数曲面を求めるには、上記の方程式 (1) を解くことになる。

メタボールの式を説明する。空間中にメタボールが N 個あるとする。メタボール i $(i = 1, 2, \ldots, N)$ の密度関数 $f_i(r)$ は、メタボール i の中心点 $\mathbf{c}_i$ から点 $\mathbf{x}$ までの距離 r の関数[*2]であり、r が増加するに従って単調に減少するようなものが用いられる。メタボール i の有効半径 R_i に対し、$r > R_i$ のとき $f_i(r) = 0$ となる[*3]。N 個のメタボールによって構成される等値面上の 3 次元の点 $\mathbf{x}$ は次の式を満たす:

$$f(\mathbf{x}) = \sum_{i=1}^{N} q_i \, f_i(r) - T = 0, \tag{2}$$

$$r = \frac{\|\mathbf{x} - \mathbf{c}_i\|}{R_i} \tag{3}$$

ここで T は閾値 (この実験では $T = 0.5$ を用いる)、$\{q_i\}$ は密度の係数である。等値面の法線ベクトル $\mathbf{n}$ は勾配ベクトル $-\nabla f(\mathbf{x})$ を正規化することで得られる。

$$-\nabla f(\mathbf{x}) = -\sum_{i=1}^{N} q_i \, \nabla\{f_i(r)\}, \tag{4}$$

$$\mathbf{n} = \frac{-\nabla f(\mathbf{x})}{\| -\nabla f(\mathbf{x})\|} \tag{5}$$

1.2 メタボールの密度関数

メタボールに用いられる密度関数にはいくつかの種類がある。レンダリング方法としてレイトレーシングではなくメッシュ分割を用いる場合で、かつメタボールの数が少ない場合は、距離の二乗の逆数もよく使われる。

$$f_i(r) = \frac{1}{r^2 + \epsilon} \tag{6}$$

ここで ϵ は、$r = 0$ のときにゼロ除算が発生するのを防ぐための小さな正の定数である。また、単なる距離ではなく距離の二乗が好まれるのは、計算コストの高い `sqrt` 関数を呼ばなくて済むためである。密度関数としては、歴史的には Blinn が 1982 年にガウス関数を用いたのが最初である。

$$f_i(r) = \exp\left(-\beta_i \, r^2\right) \tag{7}$$

[*2] このようにある点からの距離だけで値が決まる関数を**放射基底関数** (*Radial Basis Function; RBF*) と呼ぶ。

[*3] 一般に、関数 f_i の値が非ゼロとなる部分を**サポート** (*support*; 台) と呼ぶ。有効半径 R_i をサポート半径とも呼ぶ。R_i が有限の値であれば関数 f_i はコンパクトサポート (compact support) を持つ、といい、R_i が無限となるような関数はグローバルサポート (global support) を持つ、という。つまり、コンパクトサポートを持つ関数 f_i なら、r が一定半径 R_i より大きくなれば値がゼロとなるが、グローバルサポートを持つ関数 f_i は、r がいくら大きくなっても値がゼロにならない。

ここで β_i は定数である。彼はガウス関数を用いた陰関数曲面を、メタボールではなく *blob* と呼んだ。

上記の距離の二乗の逆数やガウス関数は無限のサポートを持つため、等値面の計算にはすべての密度関数の影響を考慮する必要があり、計算コストが高い。そこで、有限のサポートを持つ密度関数として、大村らが 1983 年に区分的 2 次多項式を提案した。

$$f_i(r) = \begin{cases} 1 - 3r^2 & \left(0 \le r \le \frac{1}{3}\right) \\ \frac{3}{2}(1-r)^2 & \left(\frac{1}{3} < r \le 1\right) \\ 0 & (1 < r) \end{cases} \tag{8}$$

「メタボール」という用語は大村らが初めて用いた[*4]。さらに、1987 年に村上らが 4 次多項式

$$f_i(r) = \begin{cases} \left(1 - r^2\right)^2 & (0 \le r \le 1) \\ 0 & (1 < r) \end{cases} \tag{9}$$

そして 1986 年に Wyvill らが 6 次多項式を提案した。

$$f_i(r) = \begin{cases} -\frac{4}{9}r^6 + \frac{17}{9}r^4 - \frac{22}{9}r^2 + 1 & (0 \le r \le 1) \\ 0 & (1 < r) \end{cases} \tag{10}$$

これらは次数が高いものほど滑らかな曲面を表現できる一方、交差判定のための方程式の求解のコストが増大する。図 2 にこれらの密度関数のグラフを示す。

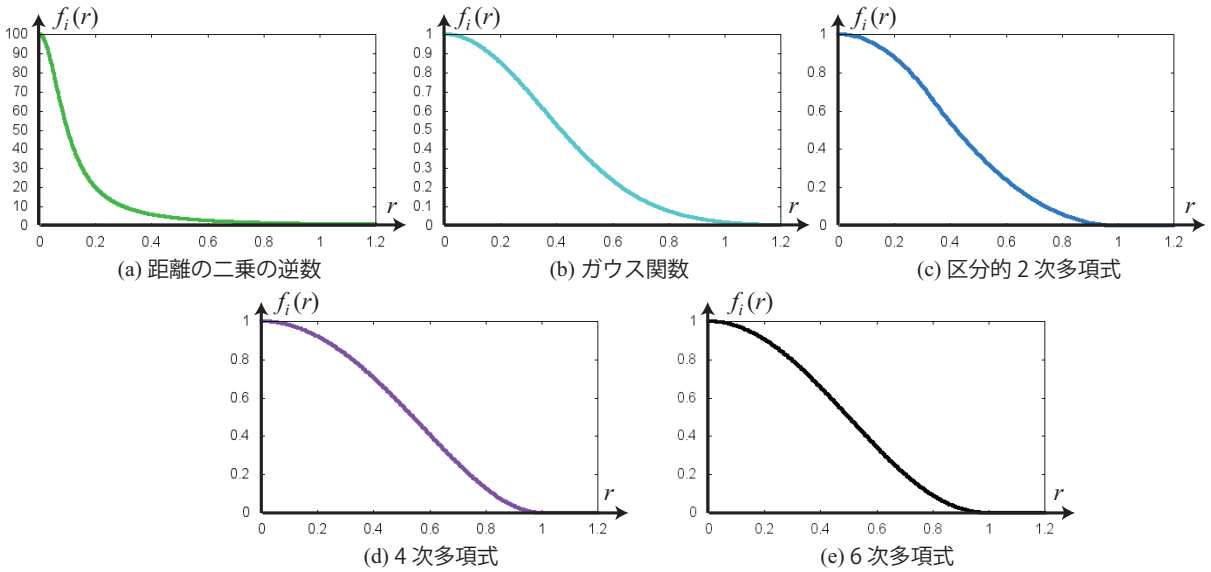


図 2　メタボールの密度関数のグラフ．ただし, (a) 距離の二乗の逆数では $\epsilon = 0.01$ とし, (b) ガウス関数では $\beta_i = 4$ とした.

[*4] 名前の由来は「メタボールを用いて形状をモデリングしようとしてもメタメタ (大阪弁) になる、つまり思ったような形をうまく作れない」から、ということらしい。

2　Bézier Clipping によるレイと等値面の交差判定

この実験では、上記の 6 次式の密度関数で定義されたメタボールについて、*Bézier Clipping* と呼ばれる手法を用いて、レイと等値面の交差判定を行う。この方法は西田らによって 1994 年に提案された。Bézier Clipping とは、1990 年に西田らによって提案された、多項式方程式の求解法である。他の求解法、例えば Newton 法と比べると、Newton 法では初期値を適切に与えなければ解が求まらないが、Bézier Clipping では初期値は必要ない。また、多項式関数のグラフが明らかに軸と交差しないような場合、方程式に解がないことを素早く判定できる。詳しくはこのレジュメの付録を参照してほしい。

レイと等値面との交差判定で解くべき方程式 (2) は、式 (6) や式 (7) のようなグローバルサポートを持つ密度関数を用いた場合、つねにすべてのメタボールを対象として計算する。一方、上記の 6 次式 (10) のようにローカルサポートを持つ密度関数を用いる場合、視線と交差しうるメタボールのみを対象に方程式を立てる。メタボールがレイと交差しうるかどうかは、メタボールのサポートを表す球 (つまり密度がゼロとなる境界を表す球。以降、サポート球と呼ぶ) がレイと交差するかどうかで判定する。レイに沿ってレイと交わるサポート球の交点を計算し、視点に近い順にソートする。そして、そして交点と交点の間の各区間において、視線と交差するサポート球の個数によって、以下のように異なる処理を行う (図 3):

レイと 1 つのサポート球が交差:　等値面は球面になるため、等値面を持つ球と視線の交差判定を行う。この球の半径は前計算できるので、実行時は高速に処理できる。

レイと複数のメタボールが交差:　視線と交差するサポート球を抽出し、交差判定のための Bézier 形式の方程式を立式し、Bézier Clipping により求解する。

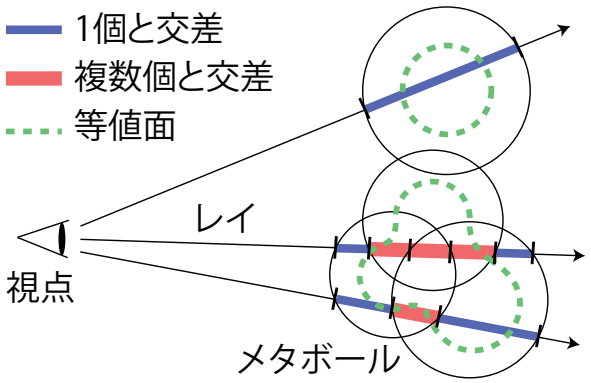


図 3　レイと交差するサポート球の数による場合分け.

以下、上記の場合分けのそれぞれについて説明する。

2.1　レイと 1 つのサポート球が交差する場合

レイと交差するのが 1 つのサポート球だけの場合、等値面は球になる。この球の半径は、密度分布の値と閾値 T とが一致する場所で決まる。球の半径がわかってしまえば、あとはレジュメ「レイと三角形の交差判定」に書いてあるような、レイと球の交差判定を行えばよい。

メタボール 1 個の場合の、等値面がなす球の半径を求める。6 次式 (10) は、$r' = r^2$ とおくと、r' に関する 3 次式になっている。

$$f_i(r') = -\frac{4}{9}r'^3 + \frac{17}{9}r'^2 - \frac{22}{9}r' + 1 \tag{11}$$

この式を、Bernstein 基底関数 $B_k^n(u) = \binom{n}{k}u^k(1-u)^{n-k}$ (ここで n は次数、$k = 0, 1, \cdots, n$ であり、$u \in [0, 1]$, $\binom{n}{k}$ は二項係数) の線形和で表すことができる。

$$f_i(r') = \sum_{k=0}^{3} d_k B_k^3(r') \tag{12}$$

上式における d_k $(k = 0, 1, 2, 3)$ に対し、4 つの座標 $(\frac{k}{3}, d_k)$ を制御点として持つような 3 次 Bézier 曲線が、$f_i(r')$ のグラフを表す (図 4)。ただし、$d_0 = 1$, $d_1 = \frac{5}{27}$, $d_2 = d_3 = 0$ である。この式を、メタボールの等値面の方程式 (2) に代入し、Bézier Clipping を用いて解を求める。

$$f(r') = q_1\, f_i(r') - T \tag{13}$$

$$= q_1 \sum_{k=0}^{3} d_k\, B_k^3(r') - T = 0 \tag{14}$$

なお、閾値 $T = 0.5$ かつ係数 $q_1 = 1$ であれば、この方程式の解は $r' = 0.25$、すなわち $r = 0.5$ となる。6 次式 (10) は、閾値 T が 0.5 であれば、等値面の球の半径がサポート半径 R_i のちょうど半分の $\frac{R_i}{2}$ になるよう設計されている。

法線ベクトルを計算するには、球の中心から交点に向かうベクトルを正規化すればよい。法線ベクトルを $\mathbf{n}$, レイと等値面の球の交点を $\mathbf{p}$, 球の中心を $\mathbf{c}_i$ とすると

$$\mathbf{n} = \frac{\mathbf{p} - \mathbf{c}_i}{\|\mathbf{p} - \mathbf{c}_i\|} \tag{15}$$

となる。

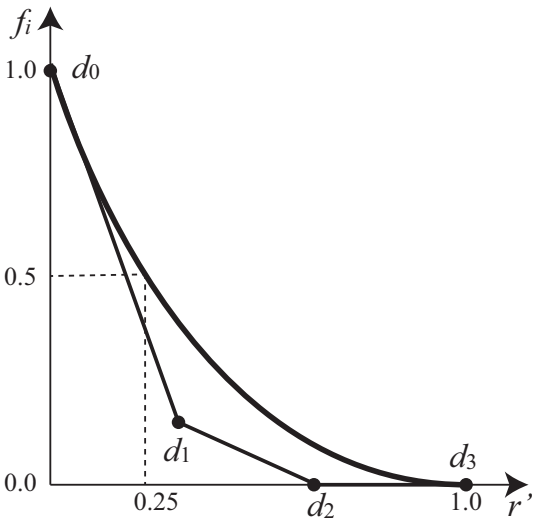


図 4　r' に関する 3 次式 $f_i(r')$ のグラフ.

2.2 レイと複数のサポート球が交差する場合

複数のメタボールを考慮してレイと等値面の交点を求める場合、まず個々のメタボールの密度関数を Bézier 曲線として表す。そしてそれらの Bézier 曲線の制御点の横軸の座標を揃え、制御点の高さを合計し、その結果得られる新しい Bézier 曲線と、閾値 T の直線との交点を求める。

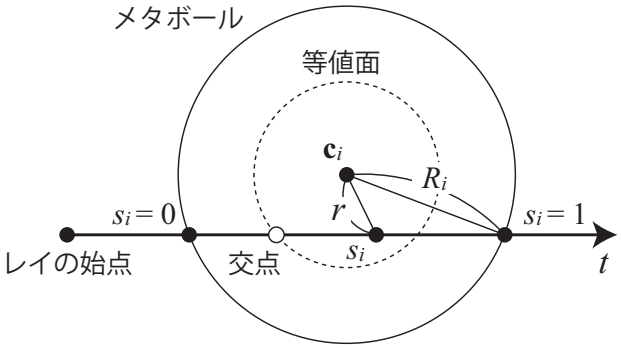


図 5 メタボールの中心点からレイ上の点までの距離 r を s_i でパラメータ化した図.

レイとサポート球の交差判定で出てくる、2 次方程式の判別式を D とすると、レイがサポート球と交差する区間の長さは $2\sqrt{D}$ である。この区間をパラメータ $s_i \in [0, 1]$ で表し (図 5)、s_i を用いて r' $(= r^2)$ をパラメータ化する。$a_i = \frac{D}{R_i}$ とおくと、

$$r'(s_i) = 4a_i\, s_i^2 - 4a_i\, s_i + 1 \tag{16}$$

となる。この式を r' の 3 次式 (11) に代入する。

$$f_i(s_i) = -\frac{256}{9}a_i^3 s_i^6 + \frac{768}{9}a_i^3 s_i^5 + \frac{16}{9}(5 - 48a_i)a_i^2 s_i^4 + \frac{32}{9}(8a_i - 5)a_i^2 s_i^3 + \frac{80}{9}a_i^2 s_i^2 \tag{17}$$

式 (17) を Bézier 曲線の式に変換する。

$$f_i(s_i) = \sum_{k=0}^{6} d_k B_k^6(s_i) \tag{18}$$

ここで 7 つの制御点 $(\frac{k}{6}, d_k)$ $(k = 0, 1, \cdots 6)$ の座標について、d_k は次のような値となる。

$$d_0 = d_1 = d_5 = d_6 = 0,\ \ d_2 = d_4 = \frac{16}{27}a_i^2,\ \ d_3 = \frac{8(8a_i + 5)a_i^2}{45} \tag{19}$$

これらの制御点からなる Bézier 曲線は左右対称であり、$s_i = 0.5$ のとき最大値 $f_i(0.5) = \frac{(4a_i+5)a_i^2}{9}$ となる。このとき $r'(0.5) = 1 - a_i$ である。なお、密度関数として 4 次式 (9) を用いた場合、d_k $(k = 0, 1, 2, 3, 4)$ は次のようになる。

$$d_0 = d_1 = d_3 = d_4 = 0,\ \ d_2 = \frac{8}{3}a_i^2 \tag{20}$$

複数のサポート球がレイと交差する場合、サポート球とレイの 2 つの交点の間の区間の分だけ、密度関数を表す Bézier 曲線を切り取る (*clip*; クリップする) 必要がある。Bézier 曲線のクリップには de Casteljau の細

分割アルゴリズムを使う。いま 2 つのサポート球がレイと交差していて、クリップ後の 2 つの Bézier 曲線を f_1, f_2 とし、さらにそれらの制御点を d_k^1, d_k^2 とする $(k = 0, 1, \cdots 6)$。すると、2 つのメタボールの密度関数を合計したグラフは、f_1 および f_2 の制御点の高さを足して得られる Bézier 曲線となる (図 6)。すなわち、得られた Bézier 曲線を f_{12}, その制御点を d_k^{12} とすると $d_k^{12} = d_k^1 + d_k^2$ である。この Bézier 曲線 f_{12} に対して Bézier Clipping を適用することで、方程式の解が得られる。レイの始点に近い順に、区間ごとに Bézier Clipping によって解を求め、得られた最小の解 s がレイと等値面との交点である。

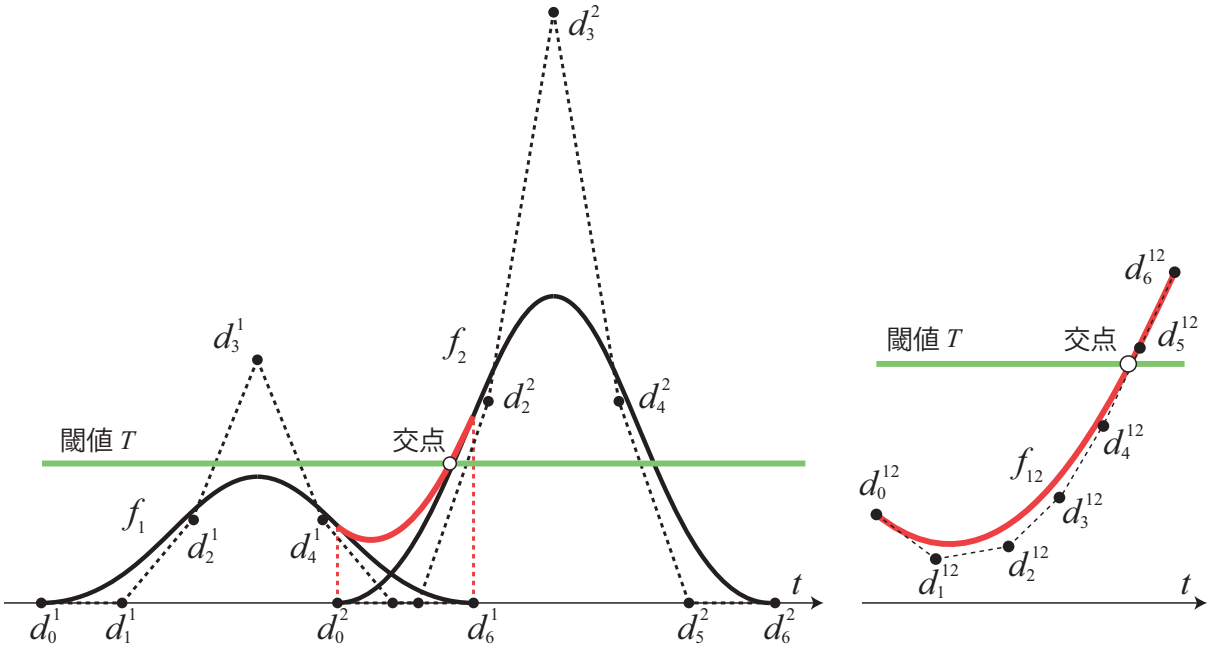


図 6　複数のメタボールの密度分布の和を Bézier 曲線で表した図.

法線ベクトルを求めるには勾配ベクトルを計算する。6 次式の密度関数を式 (11) のように $r'(= r^2)$ の 3 次式に置き換えた式を使う。

$$\frac{\partial f_i(r')}{\partial r'} = -\frac{4}{3}r'^2 + \frac{34}{9}r' - \frac{22}{9}, \tag{21}$$

$$\nabla r' = \nabla\left\{\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{R_i^2}\right\} = \frac{2}{R_i^2}(\mathbf{x} - \mathbf{c}_i) \tag{22}$$

であるから、勾配ベクトルは

$$-\nabla f(\mathbf{x}) = -\sum_{i=1}^{N} q_i \nabla\{f_i(r)\} \tag{23}$$

$$= -\sum_{i=1}^{N} q_i \frac{\partial f_i(r')}{\partial r'} \nabla r' \tag{24}$$

$$= -2\sum_{i=1}^{N} \frac{q_i}{R_i^2}\left(-\frac{4}{3}r'^2 + \frac{34}{9}r' - \frac{22}{9}\right)(\mathbf{x} - \mathbf{c}_i) \tag{25}$$

となる。なお多項式の計算では、変数の累乗と係数の積を項ごとに行うよりも、次のように一次式を順次計算

していくようにした方が、乗算と加算の回数を減らせる。このような計算方法を**ホーナー法**と呼ぶ。

$$-\frac{4}{3}r'^2+\frac{34}{9}r'-\frac{22}{9}=\left(-\frac{4}{3}r'+\frac{34}{9}\right)r'-\frac{22}{9} \tag{26}$$

2.3 レイとメタボールの等値面の交差判定の擬似コード

ここまでの手順をまとめると、レイとメタボールの等値面の交差判定は次の擬似コードのようになる。

レイとメタボールの等値面の交差判定の擬似コード

```
レイが交差しうるパラメータの範囲を [t_min, t_max] とする;
レイとサポート球の交点のリスト L を空集合で初期化;
現在レイと交差しているサポート球のリスト M を空集合で初期化;

for (シーン中の各メタボール m について) {  // レイと交差するサポート球の列挙
    if (m のサポート球とレイが交差しない) continue;
    レイとサポート球 m の交点のパラメータ t_1, t_2 を計算;
    if (t_min ≤ t_1 ≤ t_max)  L に (t_1, m) を追加;
    if (t_min ≤ t_2 ≤ t_max)  L に (t_2, m) を追加;
    if (レイの始点が m のサポート球の内部)  M に m を追加;
}

if (リスト L が空) return false;  // レイがどのサポート球とも交差しなかった
リスト L をレイのパラメータ t に関して昇順にソート;
交点の区間のうち始点を表すパラメータ t_current を t_min で初期化;

for (リスト L の各要素 (t_new, m) について) {  // t_new は区間の終端. 区間 [t_current, t_new] を検査
    if (リスト M の要素数が 1) {  // 等値面は球
        メタボール m の等値面の半径 r を計算;
        if (レイが等値面の球と交差) {
            レイと等値面の球との交点のパラメータ t_intersect を計算;
            if (t_current ≤ t_intersect ≤ t_new)
                必要な情報を計算したのち return true;
        }
    } else if (リスト M の要素数が 2 以上) {
        Bernstein 基底関数の係数 d_0, d_1, ··· , d_6 を −T で初期化;  // T は閾値
        for (リスト M 中の要素 k について) {
            メタボール k について Bernstein 基底関数の係数 d_0^k, d_1^k, ··· , d_6^k を計算;
            d_0, d_1, ··· , d_6 にそれぞれ d_0^k, d_1^k, ··· , d_6^k を加算;
        }
        d_0, d_1, ··· , d_6 を引数に Bézier Clipping を適用して解 α ∈ [0, 1] を求める;
        if (解 α が見つかった) {
```

$t_{intersect} = (t_{new} - t_{current})\,\alpha + t_{current}$;
if $(t_{current} \leq t_{intersect} \leq t_{new})$
必要な情報を計算したのち **return true**;
}
}

// レイと等値面の交点が見つからなかったので次の区間に進む
$t_{current} = t_{new}$;
if (メタボール m がリスト M に含まれる) リスト M から m を除外;
else リスト M に m を追加;
}
return false;

3 プログラムについて

今回実装するのは、MetaballSurface.cpp ファイルの hit 関数および shadowHit 関数の 2 つである。より詳しくは、それぞれの関数において上記の擬似コードのうち、「レイと交差するサポート球の列挙」のところと、「リスト M の要素数が 1」のところの 2 つを実装する。

「レイと交差するサポート球の列挙」については、まず Metaball クラスが MetaballSurface.h に定義されているので、それを参照しつつ、サポート球の中心点の座標および半径を取得する。そしてレイと球の交差判定を行い、擬似コードを参考に必要な情報を登録する。可変長配列 sphereIntersections に、レイのパラメータ t とメタボールの ID 番号 m を登録するには、次のように書けばよい。

可変長配列 sphereIntersections への値の登録

```
sphereIntersections.push_back( SphereIntersectionInfo(t,m) );
```

続いて「リスト M の要素数が 1」について、こちらもレイと球の交差判定を行う。この実験のプログラムでは、等値面の式 (2) において、密度の係数 $q_i = 1$, 閾値 $T = 0.5$ に限定している。この場合、等値面の球の半径は $\frac{R_i}{2}$ とすればよい。

テスト用のシーンファイルは metaball.cfg である。下記にメタボールに関する部分のみ抜粋する。

シーンファルにおけるメタボールの指定方法

```
metaball_surface
{
  metaball { (0,0,0), 2.0 }; // (0,0,0) が中心点の座標、2.0 がサポート半径
  metaball { (0,4,0), 2.0 };

  /* ... */
}
```

コメントにも書いてあるように、各メタボールのサポート球の中心点と半径を対にして登録する。

付録: Bézier Clipping について

Bézier Clipping は、多項式方程式を Bézier 曲線として捉え、方程式の解を Bézier 曲線と横軸との交点として素早く求める手法である。任意の多項式は Bézier 形式、つまり Bernstein 基底関数の線形和に変換でき[*5]、そのグラフを Bézier 曲線として描くことができる。Bézier 曲線が軸と交わる点 (つまり方程式の解となる点) が存在する範囲を徐々に狭めていくことで解を求める。

Bézier Clipping の処理の様子を図 7 に示す。一般に、Bézier 曲線のすべての制御点をぴったり囲むような凸多角形 (*convex hull*; **凸包**) を考えると、Bézier 曲線は必ず凸包の内側に収まる。もし Bézier 曲線が横軸と交差するなら、凸包も横軸と交差する[*6]。さらに、Bézier 曲線が横軸と交差するのは、凸包が横軸と交差する最小の値 t^0_{min} と最大の値 t^0_{max} との間となる。そこで交点の範囲を狭めるために、Bézier 曲線を $[t^0_{min}, t^0_{max}]$ の範囲で切り取る。これには de Casteljau の細分割アルゴリズムを使う。切り取って得られる曲線も、同じ次数の Bézier 曲線となる。さらにこの Bézier 曲線の凸包を考えると、Bézier 曲線が横軸と交差するのは凸包が横軸と交差する範囲 $[t^1_{min}, t^1_{max}]$ の中である。そこでまた Bézier 曲線を $[t^1_{min}, t^1_{max}]$ の範囲で切り取る。これを k 回反復して $[t^k_{min}, t^k_{max}]$ の範囲が十分狭まったら (ある閾値 ϵ に対し $t^k_{max} - t^k_{min} < \epsilon$ となったら)、$t = \frac{1}{2}(t^k_{min} + t^k_{max})$ を解とする。

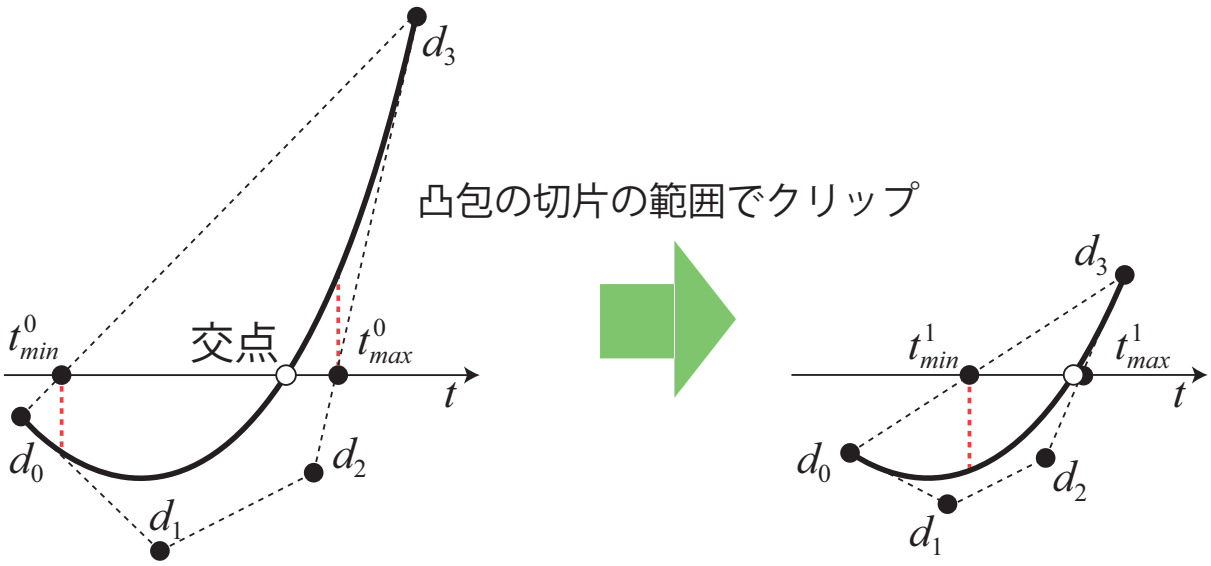


図 7　Bézier Clipping.

Bézier 曲線の形によっては、k 回目の反復で解が存在する範囲 $[t^k_{min}, t^k_{max}]$ から、$k+1$ 回反復したときの範囲 $[t^{k+1}_{min}, t^{k+1}_{max}]$ が、あまり狭まらない場合がある。そこで例えば 70% より狭まらなかったときは、Bézier 曲線を半分に分け、前半分と後半分に分けて、それぞれ再帰的に解を求める (図 8)。分割の際に前半分から先に

[*5] ただし変域 [0, 1] の範囲に限る。それ以外の範囲を扱う場合は、予め変数変換によって変域が [0, 1] の範囲になるよう式を変換しておく必要がある。

[*6] 逆に、凸包が横軸と交差しても Bézier 曲線が交差するとは限らない。凸包が横軸と交差しないなら Bézier 曲線も交差しない。明らかに凸包が横軸と交差しない場合、例えば制御点の高さがすべて正またはすべて負の場合、解が存在しないことを素早く判定できる。

解を求めるようにすれば、Bézier 曲線が横軸と複数回交差する場合に、小さい順に解を求めることができる。

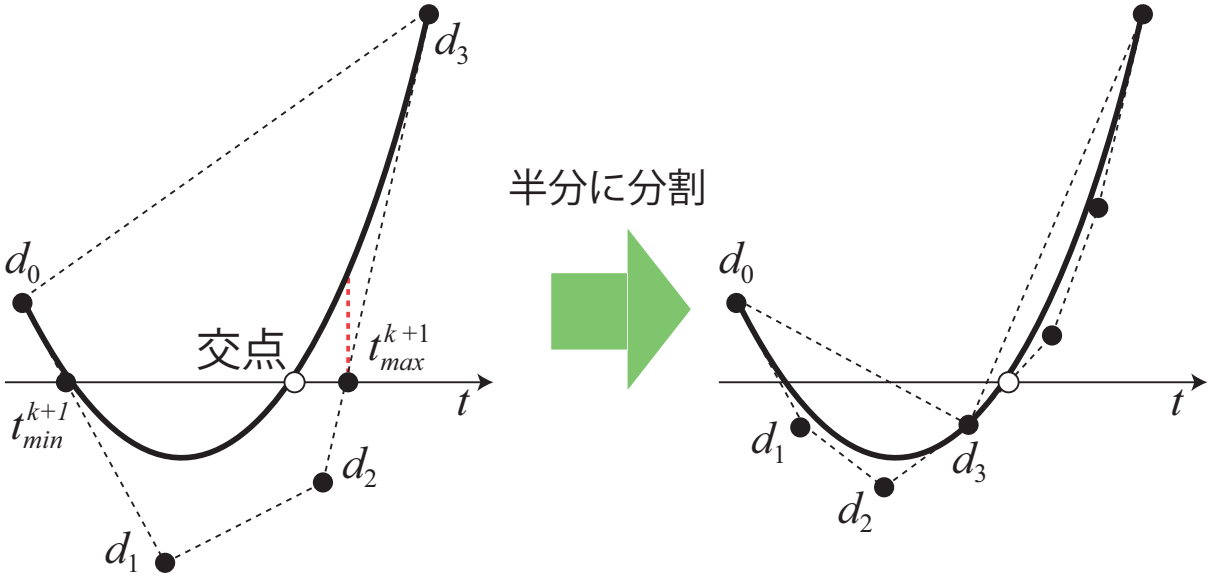


図 8　範囲が十分に狭まらない場合は Bézier 曲線を半分に分割する.