I-001

距離場と包含球を用いた 3 次元 B スプラインの編集

3D B-spline editing using distance fields and bounding spheres

西田友是
Tomoyuki Nishita
プロメテックＣＧリサーチ/デジタルハリウッド大学 /福山大学
nishita@shudo-u.ac.jp

1. はじめに

コンピュータグラフィックスや CAD では、形状表現や軌跡に曲線が多用される. 本稿では 3 次元曲線をスクリーン上でポインティングデバイス（マウス）を利用して、インタラクティブに簡易に編集するシステムを提案する. 本稿では、局所変形に有用なＢスプライン曲線を取り扱う. こうした曲線は制御点（及び重み）および節点（ノット）で定義されるが、制御点をマウスで移動する編集が一般的である. 本稿では制御点に限らず曲線上の任意の点（マウスからの最短点）を掴んで変位させる（あるいは節点挿入）により編集する. そのため、曲線と点との距離をベジエ関数で表現される距離場を定義し、効率的に曲線上のマウス近傍検出を行う方法を提案する. また、B スプライン曲線を構成する幾つかの区間(セグメント)の包含球を利用し、さらに効率化する. 距離場を表現するベジエ関数の制御点の凸包を利用し、Bezier Clipping 法[1]を適用して最短・最遠点を抽出できる方法である.

2. 従来法および基本的な方法

本稿では曲線上の任意の点をマウスで自由に変位させる、あるいはノットの挿入（曲線分割）する方法を提案する. パラメータから曲線上の点を計算するのは容易であるが、逆に曲線上の点（近傍）からパラメータを計算するのは容易ではない. 3 次元空間では曲線上の点を指定するのは特に難しい. この問題はスクリーン上においてマウスで指定した点（視線に相当）に対する曲線上の最近点を検出することで解決できる. 本稿ではＢスプライン曲線を構成するベジエ曲線の包含球とスクリーン上の曲線からの距離場を利用してマウス（レイ）の近傍の曲線上の点を検出する方法を提案する. 曲線上の任意の点を掴んで編集するが、曲線上の正確な点ではなく近傍点で処理する. そのため、本稿では点（および線分）と曲線の（2 乗）距離関数を利用する. まずスクリーン上のプローブ（マウスで指定）に近い曲線を選択し、プローブに最短の曲線上の点の座標およびパラメータ値を抽出する. 次にマウスドラッグに応じ曲線を変形する. あるいは、指定点での曲線分割（制御点の追加）もできる. 曲線としてはＢスプライン曲線を扱うが、Ｂスプライン曲線はベジエ曲線に変換できるので、ベジエ曲線に対応できれば十分である. 著者は 20 年前に有理ベジエ曲面に関してはレイトレーシング法で曲面との交点を求める方法を発表[1]（Bezier Clipping 法）しているが, 本稿ではこの方法を利用した最短・最遠点検出を用いる. また、筆者らは、点・曲線・曲面間の最短距離(衝突判定)の計算法についても発表している[3]. これらを応用した効率的な距離関数計算を提案する. 指定点に近い曲線は指定点を中心にした円（3 次元なら球）に交差する曲線および区間を抽出できる点に特徴がある. 提案法は基本的に曲線の再分割により、解に収束する方法であり、分割区間は線形計算のみで実現できる.

3. 提案法

著者は既に点とベジエ曲線の最短距離計算法を発表しているが[2]、それをＢスプライン曲線に適用するように改良する. Ｂスプライン曲線はベジエ曲線に変換できるので、各ベジエ曲線に関して、最短点・最遠点検出を行えばいい. 例えばｍ点の 3 次Ｂスプライン曲線の場合ｍ－3 個のベジエ曲線に分解できる. 簡単のため 2 次元で点との距離についてまず説明する. まず, 制御点P_k (x_k, y_k)をもつ n 次ベジェ曲線（パラメータ t）を考える.

$$\mathrm{P(t)} = \sum_{k=0}^{n} P_k B_k^n(t) \qquad (1)$$

ここで、$B_k^n(t)$はバーシュタイン多項式である. 点 Q および曲線上の点P(t)、2 点 PQ 間の距離の 2 乗距離関数Dを考える. 基本的にはベクトルの内積の形式であるが、実際に適用する際はすべてベジエ関数に変換される.

$$D(t) = \big(\mathrm{P}(t) - Q\big) \cdot \big(\mathrm{P}(t) - Q\big) \qquad (2)$$

曲線上の点P(x(t), y(t))の(x, y)座標は式(1)のベジエ関数で定義されているものとする. 点Pの(x, y)座標にベジエ曲線の式を代入すると、パラメータtに関するバーシュタイン多項式(有理の場合は分数形式)となる. 最近点は距離関数が最小になる点なので、距離関数の微分が0となる点として求まる. 3 次元の場合、スクリーン上の点はその点を通過する視線(レイ;線分)に近いかの判定になり、式 2 の Q は線分上の点である.

（1）点と曲線との近傍距離

近傍円(または球)と曲線の交差判定は式(2)を利用できる. 円や球でも曲線への距離は、$2n$ 次のベジエ関数であり、この関数が解があるかで交差判定ができる. 中心 Q(x_q, y_q)および半径 R の円とn次ベジエ曲線(パラメータt)の場合、式(2)は

$$e(t) = (\sum_{k=0}^{n}(x_k - x_q)B_k^n(t))^2 + (\sum_{k=0}^{n}(y_k - y_q)B_k^n(t))^2 \qquad (3)$$

整理すると（ベジエ関数間の積の公式利用）

$$e(t) = \sum_{k=0}^{2n} d_k\, B_k^{2n}(t) \qquad (4)$$

上式の最小値が最短距離であるが、$e(t) - R^2 = 0$で円との交差が判定できる. この高次式を精度よく解く必要があるのは距離が 0 に近い時のみである. 制御点の凸包の性質から, （2n+1）個の制御点の最小値d_{min}で距離を代表できる(図 1 参照). 指定距離より小さい時は、この関数を微分した関数$e'(t) = \sum_{k=0}^{2n-1}(d_{k+1} - d_k)\, B_k^{2n-1}(t)$（隣接した制御点の差分で構成）が 0 になる区間を求めこの区間で曲線をクリップし、距離の最小値を計算できる. この分割された区間の関数は、区間が微小なら制御点と関数は誤差がなくなり、分割区間の制御点の最小値で代表できる. 図 1 に曲線からの距離関数$e(t,\ R)$および距離関数の最小値を求める関数$e'(t)$ (区間はクリップされている)を示す. この距離関数の制御点の値も点 Q と制御点P_iへのベクトル($p_i = P_i - Q$)の

内積の線形和で簡易に求まる. 次数 n=3 なら例えば関数の制御点 1 はd_1=$(p_0 \cdot p_1)$ ． なお、図 1 は点 Q からの距離関数$e(t)$を示し、距離の最小値は関数の制御点d_k（図中の青 x）の最小値d_{min}で近似できる、距離関数を微分した関数の制御点の凸包から、極値の存在区間を抽出し再帰計算で解に収束する、$d_{min} > R^2$ （図中青の水平線）なら近傍でない曲線とみなせる. 近傍の場合、距離関数の制御点は 2n＋ 1 個、すなわちパラメータ区間は 2 n 区間あるので、まずは解の無い区間がわかればいい. 単に$(d_k > R^2,\ d_{k+1} > R^2)$ である区間 k は切り捨てることができる. 一般に距離関数は下に凸であるので、両端を切り捨てできる. クリップした区間をもとに、再帰的にクリッピングし、近傍円との交差の可能性、さらに最短距離を算出する.

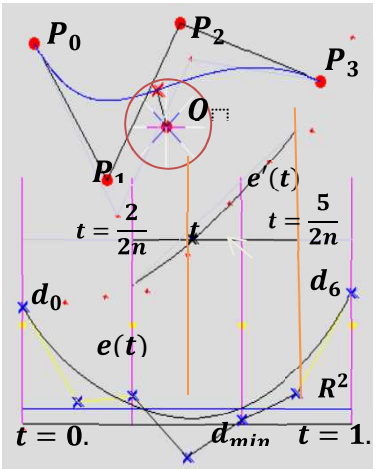

図 1 点と曲線の距離関数

（2） 線分と 3 次元曲線との距離

視線（レイ：スクリーン上のマウスと視点を結ぶ）に近い曲線の検出を示す. これはレイを中心とする円柱と曲線の交差判定と言える. レイの単位ベクトルを$\boldsymbol{v}=(v_x,\ v_y,\ v_z)$とし、レイ（線分）とベジエ曲線$P(u)$との距離は下記で表現される. $\boldsymbol{Q}$は線分上の点（例えば視点）である. $\mathbf{p}(u)=(\boldsymbol{P}(u)-\boldsymbol{Q})$とし、

$$e(u)=(\boldsymbol{P}(u)-\boldsymbol{Q})\cdot(\boldsymbol{P}(u)-\boldsymbol{Q})-(\boldsymbol{P}(u)-\boldsymbol{Q})\cdot v)^2$$
$$=\mathbf{p}(u)\cdot\mathbf{p}(u)-(\mathbf{p}(u)\cdot v)^2=\sum_{k=0}^{2n} d_k B_k^{2n}(u) \quad (5)$$

ただし$d_k=\mathrm{d}_k^x(1-x_v^2)+\ \mathrm{d}_k^y(1-y_v^2)\ +\mathrm{d}_k^z(1-z_v^2)\ -2(\mathrm{d}_k^{xy}+\mathrm{d}_k^{yz}+\mathrm{d}_k^{zx})$である. なお、視線が z 軸になるように回転させると、$d_k=\mathrm{d}_k^x+\mathrm{d}_k^y$ $(\mathrm{d}_k^x,\ \mathrm{d}_k^y$ は式(3)の x 、 y の内積成分に相当)と簡単化することも可能である. いずれにしても 2n 次のベジエ関数であり、点と曲線との判定と同じ操作で線分と最近距離が計算できる.

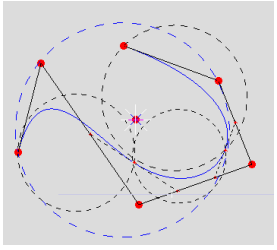
図 2 Ｂスプラインの3セグメントの包含円

（3） 包含球の抽出

Ｂスプライン曲線の各セグメントに対する包含円・球の検出法を示す. 2 次元で説明すると、包含円とは円の中心から曲線の最遠点に接する外接円である. すなわち、曲線を囲む中心Ｑと半径Ｒを抽出する問題である. 少なくとも曲線の 2 端点を通過する円より大きな包含円となる. したがって 2 端点の中点 Q と 2 端点の距離の半分を半径Ｒとする円を初期値とし探索する. ここで包含円（または球）と曲線の最遠点抽出には式(4)を利用できる.この距離関数は 2n 次のベジエ関数であり、この関数の制御点の最大値で判定ができる.図 2 にＢスプラインの包含円(青)と 3 つのセグメントの各包含円（黒）の例を示す.

（4） 3 次元曲線の編集

編集としては、①局所変形、②曲線の分割・切断、③平行移動を考える. ①は、指定した点がマウスの移動に応じて変形させる. これは指定点に近い制御点(端点は除く)を動かす. Ｂスプライン曲線の制御点Ｐ$_k$の重み係数の大きい制御点を選択し、マウスの移動ベクトルに比例して移動させる. ②は、指定点のパラメータを用いて、Ｂスプライン曲線の場合 De Boor-Cox の方法で曲線を分割する. これはノット挿入と等しい. 分割した際の新たな制御点が指定点の前後に生じる.

（5） 3 次元曲線の処理

3 次元表示の場合、球とのレイトレーシングによりマウスで制御点を選択できるようにした. 球のみでなくレイとの最短点も検出可能とした. またマウスの動きに応じスクリーン方向の移動、あるいはx、y、z成分を指定して移動させられる.

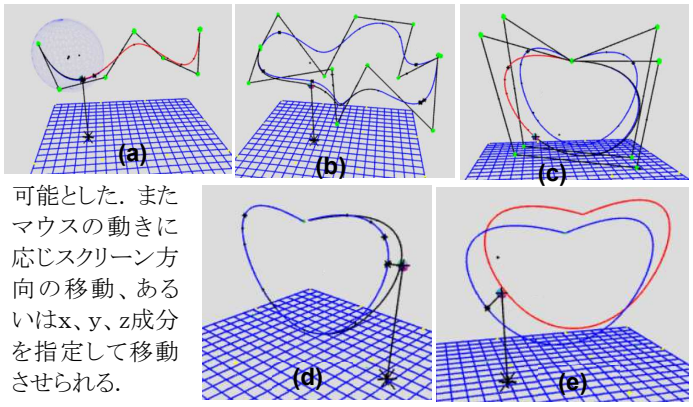

図 3 3 次Ｂスプライン曲線の変形・分割

4. 計算例

基本的には携帯端末など種々のプラットホームで動作できるように JavaScript で開発した. 図3に 3 次Bスプライン曲線の変形、分割の計算例を示す. (a)は 6 点の 3 次Bスプライン曲線をマウスで分割(赤曲線と黒曲線)した例で、マウスの近くセグメントの包含球も表示. 図(b)は 7 点の2曲線をマウスで一部変位させた例. (c)は 6 点で構成されるハート曲線の一部をマウスで分割した例、(e)はマウスで掴み平行移動した例. (d)、(e)のように、制御点が非表示の場合、特にマウスの曲線指定は有効である. 制御点の球の大きさで遠近感がわかり、かつプローブ球の底面メッシュへの最近点の表示で奥行きが把握できる.

5. おわりに

本稿では、3 次元のＢスプライン曲線を制御点に限らず曲線上をマウスで掴み、変形・分割を行う方法を提案した. 点・線分と曲線との距離関数を提案し、この関数がベジエ関数であることを利用し、制御点および Bezier Clipping 法を用い簡易に距離計算できる． n 次ベジェ曲線の場合、点（あるいは視線）との距離関数は、2n 次のベジエ関数で、最短点・最遠点を求めるには（2n－1）次の微分関数を利用する． 高次多項式を解くようであるが、幾何学的性質を用いた線形計算のみで実現できる特徴がある.

参考文献

[1] T. Nishita, T. Sederberg, M. Kakimoto, "Ray Tracing Trimmed Rational Surface Patches, " Computer Graphics, Vol.24, No.4, pp.337-345, 1990-8.
[2] T. Sederberg, T. Nishita, "Curve Intersection using Bezier Clipping, " CAD, Vol.22, No.9, pp.337-345, 1990-11
[3] T.Nishita, Y.Nakamura "Collision Detection between Spheres and B-spline Surfaces using Distance Functions from Curves", Nicograph International 2023, 2023-6