

5 J-5

折れ線近似における線分分割による
高速ベクター変換

南方 博視 杉本 和敏

日本アイ・ビー・エム(株) 東京基礎研究所

1.はじめに

紙上に描かれた図面をオペレータにより会話型入力するには、入力に時間がかかりオペレータの人的コストも高く、熟練度も要し、またその作業も苦痛を伴うものである。そこで、図面自動入力システムが種々開発されてきており、輪郭線抽出後芯線化するものや、細線化後折れ線近似するものなどが考えられている。ところが、特殊なハードウェアを用いているシステムが多く、そのため価格も高価になり、ユーザーも限られていた。ここでは、ハードウェアに依存しない折れ線近似によるラスターベクター変換の高速アルゴリズムについて述べる。

2.分割線分による折れ線近似

2-1 折れ線近似

折れ線近似する場合の一般的なアルゴリズムを以下に示す。(図1参照)

- i) 線分 P_1P_i を直線で結ぶ。
- ii) 曲線上のすべての点に対し、その点から直線へ下ろした垂線の長さを計算する。
- iii) 垂線の長さが閾値以内であれば、 i をインクリメントして同様の処理を繰り返す。
- iv) 垂線の長さが閾値を超えていれば、 P_1P_i を直線で置き換え、 P_i を新たに、 P_1 とする。

上記のアルゴリズムは曲線の両側の最大許容誤差範囲だけ離れたところに点を置き、次にその点を通る平行線にはさまれる最長の曲線部分を見つけるというものである。すなわち、直線のあてはめがうまくいかなかったところで、分割点を作るようになっており、高速化のために1パスの融合法による折れ線近似となっている。¹⁾ ii) で用いる計算法、すなわち、図2において点 $B(X_b, Y_b)$ と点 $C(X_c, Y_c)$ を結ぶ線分から点 $A(X_a, Y_a)$ までの距離を求める計算法を以下に示す。

直線 BC の方程式は、

$$(X_c - X_b)Y + (Y_b - Y_c)X - (X_c - X_b)Y_b - (Y_b - Y_c)X_b = 0$$

である。これを、

$$aX + bY + c = 0$$

とおくと、

($X_b \neq X_c$) の時、

$$a = Y_b - Y_c \quad b = X_c - X_b \quad c = -aX_b - bY_b$$

($X_b = X_c$) の時、

$$a = 1 \quad b = 0 \quad c = -X_b$$

となる。よって、求める距離は L は、

$$L = |aX_a + bY_a + c| / \sqrt{a^2 + b^2} \quad \text{----- (1)}$$

となる。(図2参照)

従来方法では折れ線部分が無い場合、すなわち、線分からの距離が閾値を超える分割点がない場合、辿ってきた途中の点列をすべて持っていて、その各々の点で、始点と終点から成る線分からの距離が閾値を超えるかどうかを判断しなければならなかった。その結果、閾値を超えなければ持っている点列が増大し、その点の数に比例して計算の繰り返し回数が増大し、計算処理時間が増大した。これはベクター化する線分が長い直線を含む場合に顕著にあらわれる。

2-2 分割線分による折れ線近似

そこで、点列を一定数以上増大させずに、計算量を抑える手法を考案した。すなわち、対象とする線分が m 個の点から成っている時、その線分を幾つかのサブセグメントに分割する。(ここでは、1つのサブセグメントが n 個の点列から成っているとすると、) (図3参照) そして、1つのサブセグメント内で、閾値を超えなかったら、そのサブセグメント内の点は一直線上にのっているとみなし、途中の点列をサブセグメントの最後の1点に代表させる。これにより繰り返し回数を減らすことができる。なおかつサブセグメント内の計算は2バイト整数演算で桁あふれを生じないメリットもあるため、浮動小数点演算に比べ、計算速度が高くなる。

次に、 n を m に対していくらにすれば全体の計算量が最小になるかを検討する。図1において、点 A の線分 BC からの距離をもとめる基本計算回数を1とする。(式(1)に相当) 1サブセグメントでは、 $n(n-1)/2$ 回の計算量となる。サブセグメントは、線分全体で $m/$

n 個ある。

また、各々のサブセグメントにつき1点が保持されるから、線分全体の計算量を S とすると、

$$S = \frac{m}{n} \times \frac{(n-1)n}{2} + \frac{m/n(m/n+1)}{2}$$

$$= \frac{m}{2} \left(n-1 + \frac{m}{n^2} + \frac{1}{n} \right)$$

ここで、

$$F(n) = n - 1 + \frac{m}{n^2} + \frac{1}{n}$$

とおくと、その導関数は、

$$F'(n) = 1 - 2 \frac{m}{n^3} - \frac{1}{n^2}$$

となる。

S が最小値になる n は、 $F'(n)$ が0になるときであり、ニュートン法等により容易に求めることができる。

この手法の有効性を示すため、以下の例を挙げる。

$m=200$ のとき総基本計算量は、

i) $n=199$ (サブセグメントに分けない時)

$$S = \frac{200}{2} \left(199 - 1 + \frac{1}{200} + \frac{1}{200} \right)$$

$$= 19800$$

ii) $n=7$ (S が最小値になる n の整数値)

$$S = \frac{200}{2} \left(7 - 1 + \frac{200}{7^2} + \frac{1}{7} \right)$$

$$= 1022$$

以上のように、サブセグメントに分けることにより計算量が激減したことが示された。なおかつ、この方法によればサブセグメント内の演算は整数演算でよいため、より一層の計算速度の向上が望める。図4に点列の数 m とその時の基本計算回数 S の関係を示す。

3. おわりに

折れ線近似における距離計算の際、対象とする線分を分割し繰り返し計算回数を減らした。その結果、整数演算でも桁あふれが生じないという長所も合わせ持つため、対象図形の点列が折れ線近似の閾値内に多く存在する場合、すなわち、長い直線部分が多い図形の場合に従来方法と比べて格段の処理速度の向上がみられる。

参考文献

1) 「コンピュータ・ビジョン」D.H.Ballard著、日本コンピュータ協会、1988

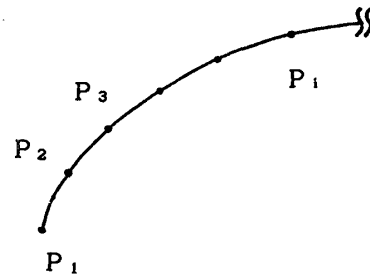


図1 折れ線近似

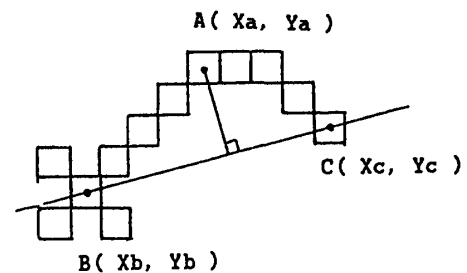


図2 折れ線近似の距離計算

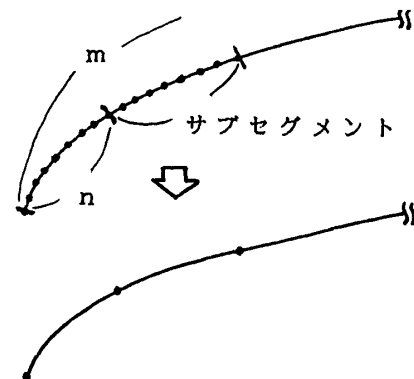


図3 線分の分割による処理

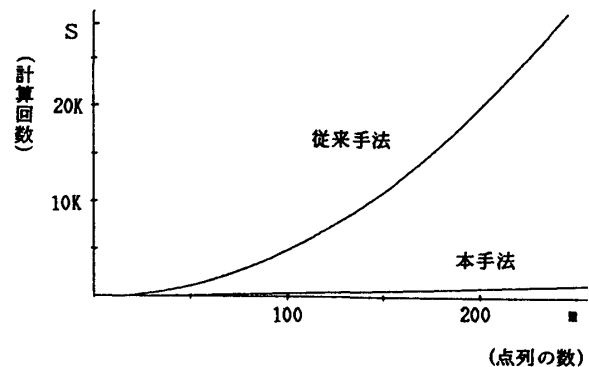


図4 点列の数と計算回数