表示一体型タブレット上でのペンの囲みに対する筆跡パタンの包含判定アルゴリズム

2N−2

佐藤　俊，村瀬敦史，中川正樹
（東京農工大学　工学部　電子情報工学科）

1．はじめに

今日，計算機を用いた文字や図の作成では，入力方法としてキーボードやマウスの使用が主流である．しかし，人間が文字や図を作成する手段は，本来紙面上でのペンを用いた手書きによるものである．表示一体型タブレットは，紙の感覚で文字や図を入力する環境を提供する．

紙面上で文字や図を書く場合に，書き間違いによる訂正を行うように，表示一体型タブレットでも訂正は必要である．表示一体型タブレットでは，文字や図の部分消去は，文字や図を構成する一部のストロークの削除という編集を行う．さらに，表示一体型タブレットは，紙を上回る移動・複写などの操作編集機能も提供できる．このような文字や図の削除・移動・複写を行うためには，削除や移動を行うストロークをなんらかの方法で指定しなければならない．また，画面上に多数のストロークが存在する状態で，多数のストロークを指定しても，速度の低下が激しくないなるべく速い処理も必要である．

本稿では，あるストロークを領域で囲み，ストロークがその領域に入っているか否かを判定し，なおかつ，処理の速いアルゴリズムを報告する．

なお，このアルゴリズムの前提として，次のことが重要である．

(1) 人間の囲み動作は不安定で，閉曲線の作り方が多様になる．あるときは，何重にもなる．

(2) タブレットの解像度に比べて，液晶表示の解像度の方が粗い．

2．領域指定アルゴリズム

まず，領域の塗りつぶしアルゴリズムを用いて，本アルゴリズムの基本を述べる．

(1) 点の閉領域内判定の基本概念

紙面上にいくつかのストロークがあり，あるストロークを指定するために，そのストロークを囲む動作を考える．囲み線は仮に時計回りとし，囲み線の内側のことを領域という．このとき，囲んだ線の連続する筆点について考えると，その筆点を結ぶベクトルが上向きのとき，その右側が閉領域を構成し，ベクトルが下向きのとき，その左側が閉領域を構成する（図１）．

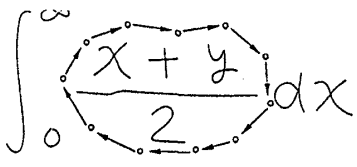

図１．閉領域

次に，ストローク中の一点が，その閉領域に囲まれるかどうかを判定する方法を述べる．

入力されたストロークの筆点座標はタブレット座標系であるが，そのストロークの描画は表示座標系で行うため，タブレットから入力された筆点座標を表示座標に変換する．タブレットの筆点座標（X_t，Y_t）に対する表示座標を（X_d，Y_d）とする対応を表す関係式を次に示す．

$$(X_d, Y_d) = f \times (X_t, Y_t)$$

$f = 1000/2048$（現在のハードウェア環境での値）

図２はある筆点座標（X_d，Y_d）に対する囲み線を表す．囲み線についても，筆点座標系列（X_{t1}，Y_{t1}）（i=1〜N）を表示座標系列（X_{d1}，Y_{d1}）に変換して，囲み線の Y_d_min から Y_d_max まで，各 Y_d 座標において X_d_min から X_d_max までの水平スキャンを作成しておく．その後，筆点（X_d，Y_d）が領域内かを判定するには，囲み線の Y_d について，X_d_min <= X_d <= X_d_max であるかを調べればよい．

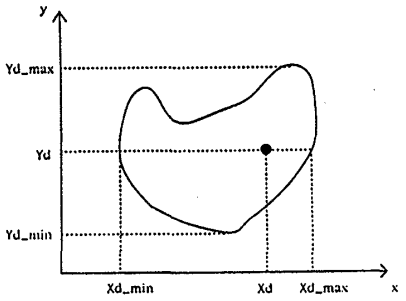

図２．閉領域内での筆点（X，Y）

(2) 塗りつぶしアルゴリズムの拡張

上記で示した囲み線は，時計回りで単純なものと限定していた．しかし，実際の手書きでは，囲み線の作り方は多様である．手書きでの囲み線の例をいくつか図３に示す．

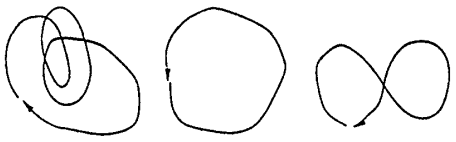

図３．手書きによる囲み線の例

塗りつぶしアルゴリズムでは，図３のような囲み線に対しては，判定を行うことができない．そこで，図３のような囲み線も判定できるアルゴリズムを次に述べる．

以下，すべての座標は表示座標系とする．

An algorithm to determine stroke inclusion within pen movement coutours on a display integrated tablet
Takashi SATOU , Atsushi MURASE and Masaki NAKAGAWA
Tokyo University of Agriculture and Technology

(3) 囲み線を登録するアルゴリズム

囲み線を示す筆点列（X_1, Y_1），（X_2, Y_2），（X_3, Y_3），・・・，（X_n, Y_n）が与えられたとする．（X_n, Y_n）は（X_1, Y_1）に連結し，このとき（X_0, Y_0）＝（X_n, Y_n）とする．この筆点列の隣合う筆点間において，DDA で補間していく座標を（x, y）とする．その y のスキャンラインに，x の値をソートしながら登録し，その際，点（x, y）の上向きか下向きかの情報も登録する．この手順を説明するプログラムを図４に示す．

```
for ( i = 1; i <= n; i++ ) {
   Vi = Yi - Yi-1;
   if ( Vi > 0 ) {                /* 上向き */
      for ( Y = Yi-1; Y <= Yi; Y++ ) {
         DDA で補間し x, y を計算
         スキャンラインに登録（上向き, x, y)
      }
   }
   if ( Vi < 0 ) {                /* 下向き */
      for ( Y = Yi-1; Y <= Yi; Y++ ) {
         DDA で補間し x, y を計算
         スキャンラインに登録（下向き, x, y)
      }
   }
}
スキャンラインに登録（ベクトル, x, y)
      /* x の値の順序を乱さずに登録 */
```

図４．囲み線を登録するプログラム

(4) 点の領域内外の判定

上記のように，囲み線の各 Y のスキャンラインの点情報として，X の値と上向きか下向きかを登録した．これらの情報からスタックを用いて，判定される点が領域内かどうかを判定する方法を次に述べる．

点（X, Y）に対し領域判定を行うとき，Y のスキャンラインの点情報で，X の値が小さい方から順に一つづつスタックに push する．連続して（上向き，X_i）と（下向き，X_j）または（下向き，X_i）と（上向き，X_j）が push されたなら，X_i <= X <= X_j を調べて入る場合，領域内と判定する．入らない場合，それら二つを pop up して，同様の処理を繰り返す．スタックに push する情報がなくなったなら処理を終了し，領域外と判定する．以上の処理について，例を挙げて図５に説明を示す．

この領域指定アルゴリズムの特徴として，処理の高速性以外に，囲み線を何本描いても判定ができるということがある．

３．処理速度の測定実験

多数のストロークがあり，その中のいくつかを囲んで判定するという処理を，本アルゴリズムと別のアルゴリズムの二つの方法で処理速度について比較したい．

今回の実験では，総ストローク数 1434 本に対して，囲まれるストロークの数が 0，1，4，14，23，33，35，66，103，123 本の十通りの場合について，それぞれのアルゴリズムの処理速度を測定した．ただし，ストロークが，囲んだ線の領域内かを判定するには，ストロークのある一点だけで判定することにする．この測定結果を図６に示す．

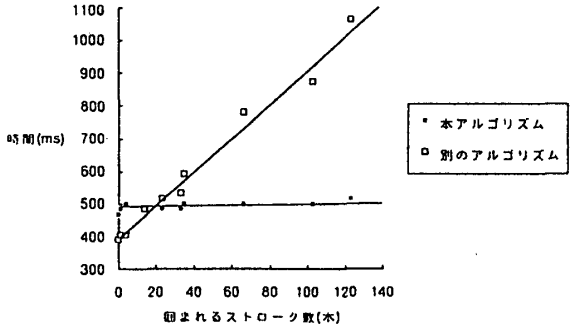

図６．各領域判定における処理速度の測定結果

図６の結果から，囲まれるストロークが少数の場合は，本アルゴリズムよりも別のアルゴリズムの処理の方が若干速かった．これは，スキャンライン作成によるコストがかかるためである．しかし，囲まれるストロークの数が増えるほど，本アルゴリズムの処理の方が次第に速くなることが確認できた．さらに，本アルゴリズムは，囲まれるストロークの数の増加による処理速度の劣化が，ほとんど起らないことに注目したい．

４．おわりに

上記のように，表示一体型タブレット上でのペンの囲みに対する筆跡パタンの包含判定アルゴリズムについて述べた．また，本アルゴリズムと別のアルゴリズムの処理速度を測定する実験を行い，測定結果について述べた．そこで，囲まれるストロークの数が多数に及んでも，いかに本アルゴリズムが速いかを確認できた．

本実験では，ストロークの包含関係をストロークの代表点一点で判定を行ったが，この点を複数にした場合，本アルゴリズムの効果はさらに増大する．

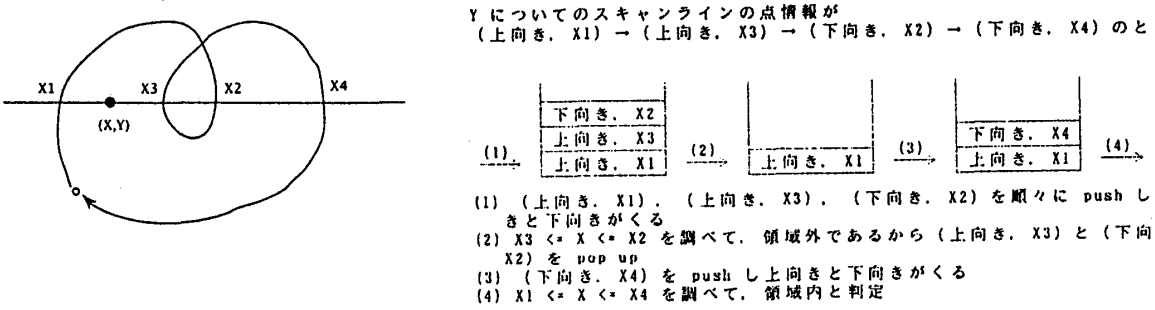

図５．点（X, Y）の領域判定