7J-1 拡張Bresenhamアルゴリズムによる画像の高速アフィン変換

伊藤　雅晴
(株)日本アイ・ビー・エム　パーソナルシステム開発統括本部

1．はじめに

近年のユーザ・インターフェース技術の進歩によるグラフィック表示技術の普及に伴って，アフィン変換に代表されるデジタル画像の幾何学的な変換の技術は，より高速で，かつメモリなどの資源を必要としない方式が要求されている。

従来から画像のアフィン変換は、画像処理の技術の中で最もＣＰＵ時間やメモリを消費するものの一つとされており、小型のワークステーションを使ったオフィス環境では実現が難しかった。

従って従来から多くの研究が為されてきた、それらの中で最近のものとしてはアフィン(または線形)変換を基本変換の積に分解して高速化する技法がいくつか報告されている[田畑86][宮沢87]。

特に[宮沢87]では、任意の線形変換を２つの拡大行列と２つのシアー行列に分解する本質的な６つの組合せのうちの特定の組合せを用いた中間メモリを必要としないアルゴリズムが報告されている。 本研究でもこの分解と同じものが、各ラスタ・ラインの変換後の先頭位置を決定するのに用いられている。

2．Bresenhamの直線発生アルゴリズム

Bresenhamの直線発生アルゴリズム [Bres65]はよく知られた高速直線発生アルゴリズムで、発生した直線の誤差を最小にするように誤差変数ｄ及びその増加分incl, inc2は次のように初期化される。

```
d := 2 * dy - dx;          (1.1)
incl := 2 * dy;            (1.2)
inc2 := 2 * (dy - dx);     (1.3)
```

ここでdx，dyは直線の始点と終点のｘ方向及びｙ方向の距離をそれぞれ表す。

直線の傾きが１より小さい場合を考えるとアルゴリズムはｘ座標を始点から終点へ１つずつ増やしながら、もしｄが負ならｄをincl だけ増加し，正または０ならｙ座標を１つ増やしてｄをinc2だけ増加する操作を繰り返して画素を発生する。具体例としてdx = 9, dy = 3 (tangent = 1/3) の場合図１に示すように、A0からA9の画素が発生される。この直線を今後Bresenham直線、またはただ直線と呼ぶ。

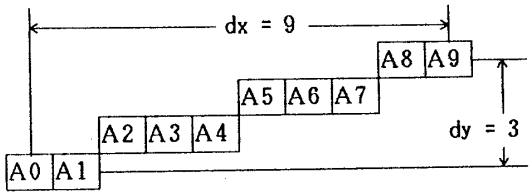

図１．Bresenham直線

Bresenham直線の性質として，傾きが０と１の間のときには各ｘ座標の列に画素が必ず１つしかない。これはアルゴリズムの繰り返しの中で毎回ｘを１つずつ増加していることからも明らかである。従ってこのBresenham直線をｙ方向に１つずつ平行移動して置いてゆくと空間を隙間なく充填する(図２)。これを画像の線形変換に応用して元画像のラスタ・ラインをこの各々の直線上に置くことにより、前後の直線と重複したり隙間を作ることなく変換された画像を発生することが可能である。

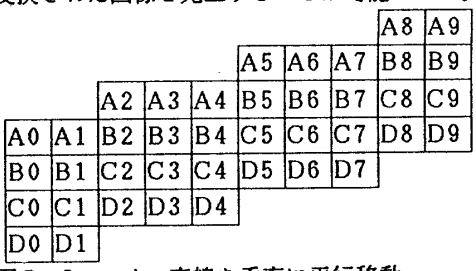

図２．Bresenham直線を垂直に平行移動

3．誤差変数ｄの初期値の決定

次に問題となるのは変換後の各直線の始点の決め方と，Bresenhamの誤差変数ｄの初期値の決め方である。まず後者の方から図３を用いて説明する。

d = -4　4　-12　-4　4　-12　-4　4　-12　-4　4

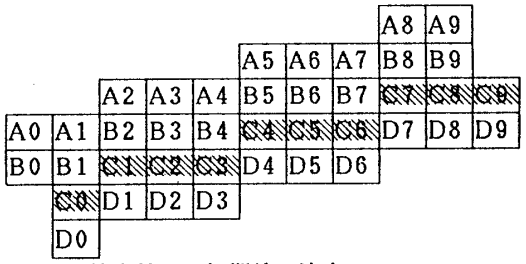

図３．誤差変数ｄの初期値の決定

まず最初の直線は通常のBresenhamと同様に(1.1)を用いて初期値ｄ=－４でA0からA9まで発生する。このときのｄの途中の値が図の上に示されている。２番目の直

High Speed Affine Transformation of Image using Extended Bresenham Algorithm
Masaharu ITOH
IBM Japan, Ltd.

線もB0のｘ座標はA0のと同じであるので初期値ｄ=－４で始める。３番目の直線の先頭がC0だとすると初期値ｄを同じにすると正しく引けない。このときにはA1のときのｄの値、即ち４をｄの初期値として直線を発生すると画素の重複や隙間を生じない。４番目の直線も同様で、以下各直線の始点がずれるに従って、その直線を発生する時のｄの初期値をやはりBresenhamのアルゴリズムに従って変えて行く。

ただし、ここで注意する必要があるのは、図３の場合は画像の回転の方向が反時計回りであったので、先頭の画素のずれる方向が直線を発生する方向と同じであったが、これが逆の時にはｄを直線の順序とは逆に発生する必要がある。

４．各ラインの先頭の画素の位置の決定

［宮沢87］に任意の線形変換が２つの拡大行列と２つのシアー行列に分解できる本質的な６つの組合せが存在することが述べられているが、本研究でもこの分解のうちの次のものを用いる。

$$\begin{matrix} & & & & \text{yシアー} & \text{xシアー} & \text{y拡大} & \text{x拡大} \\ \begin{bmatrix} a & b \\ c & d \end{bmatrix} & = & \begin{bmatrix} 1 & 0 \\ c/a & 1 \end{bmatrix} & \begin{bmatrix} 1 & ab/D \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & D/a \end{bmatrix} & \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \end{matrix} \quad (2.1)$$

ここで　D = ad-bc.

この分解では右から順番にｘ方向の拡大、ｙ方向の拡大、ｘ方向のシアーそしてｙ方向のシアーの順に実行される。この順序は非常に重要で本方式では最後に直線発生器を用いて元画像のラスタ・ラインを直接直線上に転送したいので、その傾きを決めるｙ方向のシアーが最後に一つだけある分解でなければならない。

最初のｘ，ｙ方向の拡大はやはりBresenhamのアルゴリズムを用いて実現でき、シアーの処理と同時に行うのは容易であるので今後の説明では省略する。

図４では各直線の始点がｘ方向のシアーそしてｙ方向のシアーによってどのように動くかを示す。(1)は元画像の左端の画素である。まず (2.1)式のｘシアー値ab/Dに従ってｘシアーを行ない(2)、次にｙシアー値b/aに従ってｙシアーを行う(3)。

これらの処理で各直線の始点の位置が決定されるが、これはメモリ上で行なうのではなく各直線への画像の転送と平行して行なう。従ってこの処理用の独立した直線発生器がｙ方向の拡大を含めて３つ必要である。

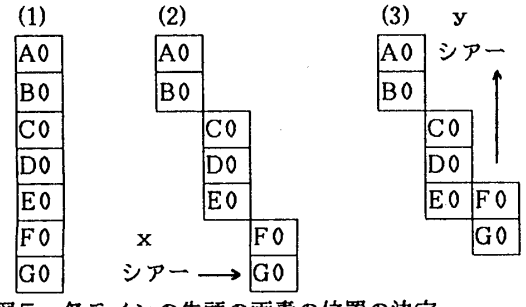

図５．各ラインの先頭の画素の位置の決定

５．全画素の処理

４.の処理で決定された位置からBresenham直線上に画像の各ラスタ・ラインを転送する。このときに３で述べた誤差変数ｄの初期値を求める必要があるが、新たにこの計算をする必要はなく４.の操作の中のｙシアーを行う直線発生器のそのラインにおけるｄの値をそのまま利用できる。図６に変換された全画素の位置を示す。画像の各ラスタ・ラインを転送するときにｘ方向の拡大も同時に行うので２個の直線発生器を用いる。４.と合わせると全部で５個の直線発生器が必要である。

							A8	A9		
				A5	A6	A7	B8	B9		
		A2	A3	A4	B5	B6	B7	C7	C8	C9
A0	A1	B2	B3	B4	C4	C5	C6	D7	D8	D9
B0	B1	C1	C2	C3	D4	D5	D6	E7	E8	E9
	C0	D1	D2	D3	E4	E5	E6	F6	F7	F8	F9
	D0	E1	E2	E3	F3	F4	F5	G6	G7	G8	G9
	E0	F0	F1	F2	G3	G4	G5				
		G0	G1	G2							

図６．変換された全画素

６．おわりに

本研究ではBresenhamの直線発生アルゴリズムを拡張した、高速で中間メモリを必要としない、画像のアフィン変換アルゴリズムについて述べた。このアルゴリズムは単に画像の回転などだけではなく、グラフィックの分野の基礎技術としても応用可能である。

７．文献

[Bres65]Bresenham,J.E.,"Algorithm for Computer Control of Digital Plotter," IBM Syst. J.,4(1) 1965,pp.25-30

[田畑86]田畑,武田,町田: ラスタ走査とテーブル参照による画像回転の高速処理. 信学論, Vol.J69-D,No.1,pp.80-90(Jan.1986)

[宮沢87]宮沢: 高速画像回転アルゴリズムT²D²分解法とその性能評価. コンピュータ・ビジョン49-1 (1987/7/23)