ショートノート

ビットローテーションを用いたハッチングパターン発生方式†

坪田信孝†† 田村純子†† 奥田久徳††

近年のLSIの進歩によりマイクロプロセッサをCPUとしてもつパーソナルコンピュータにおいても，実用レベルのCRTグラフィックディスプレ装置をもつようになった．これらの装置で多様なハッチングパターンを出力しようとするとき，多数の斜線を引くために，斜線の両端の座標を決定するためのソフト的負担が大きくなる．一方，多数の境界線で決定される多数の領域を一画面内で視覚的に認識させるためには多様なハッチングパターンが要求される．本論文では，任意のビットパターンをもつ1バイトを回転させながら積み重ねる方式によって，きわめて単純なアルゴリズムでしかも多様なハッチングパターンの発生が可能であることを述べる．また，この方式は複雑な境界線をもつ領域のぬりつぶし方式の拡張として利用できることを述べ，実現例を示す．

1. まえがき

グラフィックイメージを内部記憶装置に保存し，これを常にスキャンして表示する方式のCRTグラフィックディスプレ装置（以下GCRT）は，近年のメモリLSIの低価格化・高密度化によってパーソナルコンピュータなどでも実用化されるようになった．GCRTはXYプロッタや蓄積型CRTに比べ精度的にはやや劣るものが多いが，図形メモリに対する読み書きが可能であることから，応用範囲は広く，LSIの進歩とともに今後さらに広く利用されると考えられる．

GCRTに対する出力は点を単位として点のON・OFFの組合せで行われる．したがって，最小単位の線分の出力を基本としたXYプロッタとは異なった思考に基づくアルゴリズムが可能である．

一方，ハッチングを「平行な斜線を多数引くことによって行う」方法は，XYプロッタ向きの思考に基づくもので，GCRT上でも可能ではあるが，斜線の，両端座標を求めるための演算が必要となる．したがって多様なハッチングパターンを出力する場合や複雑な境界線をもつ領域に適用する場合には大きなソフトが必要となる．これに対し，GCRTの点を単位としているという特徴を生かし，「平行な斜線を水平方向に切ったパターンを積み重ねてハッチングパターンを構成する」方法を用いれば，斜線の両端座標を求めるための演算は水平な線分の両端座標を求めることに変化する．これによりハッチングのソフトはきわめて単純化され，しかも多様なハッチングパターンが可能となる．

ハッチングは一画面に表示される多数の領域を視覚的に認識させることを目的としたものである．したがって，たんに斜線間の間隔の大小や斜線の角度による変化だけでなく，視覚的な多様性を有することが期待される．本論文に述べる方式は，基本となるビットパターンを回転させながら用いる方式であるため基本パターンを変えるのみで，容易に視覚的多様性をもったパターンを発生させることが可能である．

以下では，8ビット・マイクロプロセッサ（以下MPU）によって制御することとして，上に述べた方式の詳細と実現例について述べる．

2. 矩形領域のハッチング

2.1 横8ドット×縦hドットの矩形

矩形の左下部の座標を(0, 0)とする．(0, 0)から(7, 0)の8点に，16進数01のイメージを表示すると(7, 0)のみONとなる（以下ではビットが1のときONを表示することとする）．次に16進数01を1ビット左方回転したイメージをy座標を1だけ増して，(0, 1)～(7, 1)に表示する．これをくりかえせば斜線のパターンが表示される．初めに用いた16進数01（以下では基準パターンと呼ぶ）を16進数11にすれば斜線間隔は1/2となる．このようにして，斜線のスタート位置の違いを含めると254種のパターンが得られ，斜線のスタート位置の違いのみのパターンを除いても33種のパターンが得られる（ただし，すべて白と

† Hatching Pattern Generation Using Bit Rotation by NOBUTAKA TSUBOTA, JUNKO TAMURA and HISANORI OKUDA (Department of Hygiene, Hiroshima University, School of Medicine).

†† 広島大学医学部衛生学教室

すべて黒のパターンを除く).

基準パターンをy座標の1増加ごとに2ビット左方回転するとゆるやかな角度の斜線となり，y座標が2増すごとに1ビット左方回転すると急角度の斜線となる．このように回転量を変化させることによってハッチングパターンをさらに多様化することが可能である．

上述の左方回転を右方回転とすれば，右上りのハッチングパターンとなる．さらに，2バイトを用意し，y座標の1増加ごとに一方は左方に，他は右方に回転し，表示に際しては両者のORを実行してから表示することとすれば，クロス線によるハッチングが行える．

2.2 横wドット×縦hドットの矩形

横方向の幅を1バイト以上にする場合，2.1節のイメージを8ビット単位で横方向に並べることによってパターンは連続する．この場合wが8の倍数でない場合には端数の処理が必要となる．ところが，横方向の拡大にも回転が利用できる．すなわち，x座標の1増加ごとに基準パターンを1ビット左方回転し，第7ビットを表示する，これを(0, 0)から(w, 0)まで行う．次にy座標を1増加し，x方向の回転を行う以前の基準パターンを1ビット左方回転し，新たな基準パターンとしてx方向の処理を行い，y座標hまでくりかえす．以下では最初に与えた基準パターンを「原基準パターン」と呼び，x方向の回転に用いるパターンを「出力用基準パターン」と呼ぶ．

3. 複雑な境界をもつ領域のハッチング

複雑な境界をもつ領域では，斜線が境界線を切る点を求めて斜線を引く方法を用いると，斜線と境界線の交点を求めるためのソフト的負担が大きくなる．これに対し，上に述べた方法では，水平線を引くことに等しいから，境界線上の点を求めるためのソフトは簡単なものになる．すなわち，GCRTではすでに出力したイメージを再入力する（図形メモリを読む）ことが可能であるから，領域内の1点の座標を与えれば，水平線が境界線を切る点の座標は，与えられた点から左右に探索的にON・OFFを調べることによって得ることができる．この探索を上下に拡大して行えば，少なくとも凸部のみで構成される多角形ではすべての境界点の座標を得ることができる．この部分のアルゴリズムは，領域全体をぬりつぶす方式（すべてONにする方式，以下PAINT方式と呼ぶ）としてすでに多くのパーソナルコンピュータで実用化されている．

PAINT方式により領域を水平線で切った場合の両端のx座標を得ることが可能であることを上に述べた．また，y座標の等しい2点を出力用基準パターンの回転によって結ぶ方法を2.2節で述べた．しかしながら，複雑な形状の領域の場合，矩形の場合と異なりy座標の変化とともに水平な線分の端部のx座標も変化するため，2.2節のアルゴリズムではy方向への水平線の積み重ねに際してパターンのずれを生じる．以下ではこのずれを生じさせないための調整方法を述べる．

ずれを生じさせないためには，原基準パターンを常にGCRT上の特定の座標（以下では原点とする）におき，水平線のスタート点からすでにその座標だけ前もって回転を加えた出力パターンを用いることとすればよい．いま，PAINT方式によって求めた境界点が(x_1, y_0)と(x_2, y_0)で，$x_1<x_2$であったとすれば，原基準パターンをx_1回左方回転し，さらにy_0回左方回転したものを出力用基準パターンとする．出力用基準パターンを左方回転しながら第7ビットをx軸方向に出力して行き(x_1, y_0)から(x_2, y_0)まで実行する．このようにして，PAINT方式で得られるすべての(x_{1i}, y_i)から(x_{2i}, y_i)までをパターンで結べば，左上りの斜線によるハッチングが出力される．

上例と同様に，原基準パターンをx_1回左方回転した後，上例とは逆にy_0回右方回転したものを出力用基準パターンとして用いると，右上りの斜線によるハッチングとなる．2種の出力用基準パターンを用意し両者のORを実行したものを出力用基準パターンとして用いると，2.1節に述べたと同様にクロス線によるハッチングとなる．

以上で述べたx_1回およびy_0回の回転はもちろん$(x_1 \div 8)$および$(y_0 \div 8)$の余りの回数だけ回転するのみでよい．したがって，x_1およびy_0の下位3ビットで表される数値，すなわちx_1およびy_0の下位8ビットと16進数07とのANDを実行して得られる数値で示される回数だけ回転すればよい．このように出力用基準パターンの調整は比較的単純な処理で行うことができる（**図1**）．

4. 実現例について

用いたハードウェアはソード電算機製のM100ACE IIである[1]．GCRTは320×256ドットと比較的荒い密度であるが統計図表などに用いる場合には十分な実用性がある．MPUはザイログ製のZ-80を用いており，出力用基準パターンの出力には左方回転命令で第

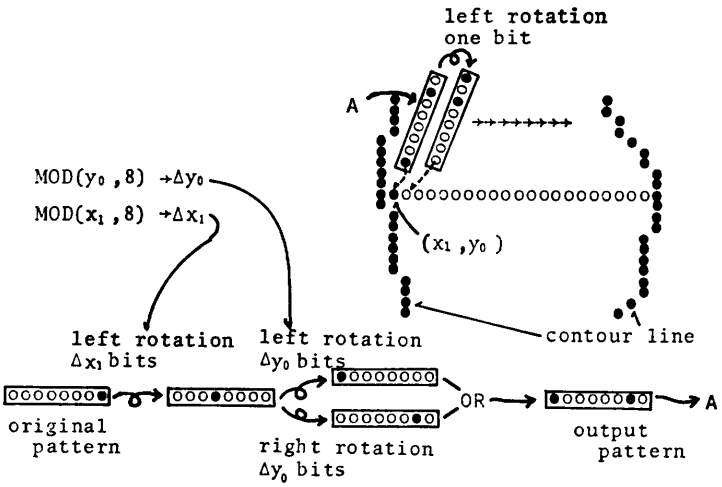


図 1　複雑な領域へのハッチング（クロス線の場合）

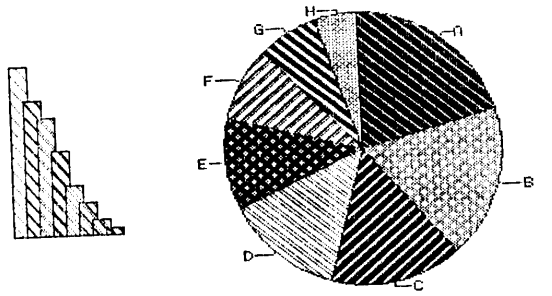
Fig. 1　Hatching with crossed lines on a complicated area.

7 ビットがキャリーフラッグに入ることを用いることができる．GCRT のコントロール用の命令としては，絶対座標を与えてその点を ON にする命令と同じく絶対座標を与えてその点の ON・OFF を調べる命令の 2 種を用いた．以上により矩形専用のサブルーチンと PAINT 方式を応用したサブルーチンをソード電算機製のアセンブラを用いて開発し，同社製の BASIC にリンクした．BASIC プログラム内で CALL 文を用いて利用することができる．PAINT 方式を応用したサブルーチンのうち，3 章で述べた処理のために要したメモリ容量はわずか 126 バイトであった．

図 2-a に矩形専用サブルーチンを用いた例を，図 2-b に PAINT 方式を応用したサブルーチンを用いた例を示した．前者は通常後者で代用できるが，矩形専用ルーチンは図 2 のように重なった矩形の出力には都合がよい．また，面積の広い矩形では PAINT 方式の応用サブルーチンよりも速度が速い，領域内の 1 点の座標を与えなくてもよいなどのメリットがある．

PAINT 方式の応用サブルーチンでは，幅の広い領域に適用すると実行速度は低下する．用いたシステムの図形メモリへのアクセス（境界点の探索）が遅いことが主因と考えられるが，ON 点以外では図形メモリへのアクセス（点の表示命令）をスキップできるので，全面ぬりつぶしに比べるとむしろ実行速度は速い．したがって，3 章で述べた部分の処理は十分速く実行されると考えられる．

実現例では GCRT の密度が比較的荒いため，ほぼ 2 章および 3 章に記述したとおりにプログラムを開発した．しかしながら，より高密度の GCRT ではハッチングパターンの線の間隔が狭くなりすぎる可能性がある．このような場合には GCRT の複数の点を出力用基準パターンの 1 ビットに対応させ，複数の点の 1 点のみを ON する（第 7 ビット 1 のとき）などの工夫によって容易に改善できる．また，16 ビット，32 ビットを単位として回転を行う方式としても原理はまったく変わるものではなく，用いる MPU や GCRT に適した応用が可能である．



a. Using rectangular routine　b. Using expanded PAINT routine

図 2 実　行　例

Fig. 2　Example.

5. むすび

本論文では，ビット回転命令を用いることによって多様なハッチングパターンを出力する方式について述べた．ビット回転命令はきわめて基本的な MPU の命令であり，また必要な GCRT コントロール用の命令も GCRT が基本的にもつべき命令のみである．したがって，多くの GCRT で実現可能であり，そのソフトウェアもきわめて単純である．しかも，基準パターン（実現例では 1 バイト）の変化により多様なハッチングが可能であり実用性も高いと考えられる．

参考文献

1) ソード電算機：M 200/M 100 ACE シリーズ・アセンブラ言語マニュアル，p. 430，ソード電算機，東京 (1980)．

(昭和 57 年 8 月 17 日受付)
(昭和 57 年 10 月 4 日採録)