コンピュータビジョン 49－1
（１９８７．７．２３）

高速画像回転アルゴリズムT^2D^2分解法とその性能評価

宮沢 篤

日本アイ・ビー・エム株式会社 東京基礎研究所
102 東京都千代田区三番町 5-19

変換行列を4つの単純な基本変換の積に分解することによって，二値画像の回転を高速に実行する，空間（メモリ）効率の高いアルゴリズムについて報告する。変換行列の分解の仕方には，互いに同値な幾つかの組合せが存在するが，ある特定の分解順序の下では，単一のラスタ走査によって全ての変換が完了する，効率的なアルゴリズムを導くことができる。ところで，各々の基本変換を具体化するときに生じる標本化誤差のために，異なる分解によって得られる出力画像の品質は，必ずしも同一になるとは限らない。本研究では，画像回転アルゴリズム一般に適用可能な性能評価のための一手法を提案するとともに，分解順序の違いが画質に及ぼす影響についての，定量的な評価を試みた。

Performance Evaluation of the T^2D^2 Decomposition Method, a Fast Image-Rotation Algorithm

Atsushi Miyazawa

Tokyo Research Laboratory, IBM Japan Ltd.
5-19, Sanban-cho, Chiyoda-ku, Tokyo 102, JAPAN

This paper presents a fast and space-efficient algorithm for the rotation of binary images, obtained by decomposing any given transformation matrix into four fundamental matrices. Several mathematically equivalent decomposition orders are possible, however. By choosing a certain order, we can derive the algorithm capable of carrying out all four steps simultaneously within a single raster scanning. Different results are usually produced depending on which decomposition order is taken. To evaluate their performance in terms of image quality, a new method for general image-rotation schemes is proposed.

はじめに

デジタル画像の幾何学的な変換は，画像処理の分野における基本的な技術の一つであるが，最近のオフィスにおけるマルチメディア環境の普及にともない，ワークステーション上での実現を考慮した，高速かつ高品質な変換アルゴリズムの開発が重要になってきている。

周知のとおり，デジタル画像をその品質を保持したまま変換しようとすると，画素単位にかなりの計算量を必要とするのが普通である。すなわち，変換を定める一価正則な写像と，ある線型補間関数が与えられたときに，変換画像領域から原画像領域への逆写像を求めることよって，出力画素の座標を原画像の上に写していく。一般に，この逆像は，画素値の定義されている格子点上に乗らない。そのため，周辺の幾つかの画素値からの補間によって，変換画像の画素値を決定しなければならない。

もともと，この逆写像を用いた処理方法は，画像の幾何変換一般に（線型か非線型かを問わず）適用が可能なアルゴリズムであるが，変換のクラスをある程度実用的な範囲に限定することによって，より高速なアルゴリズムを導くことができる。本研究で対象にしているのは，例えば，ワークステーション上の対話型の環境で行なわれるオフィス文書処理であり，そこで必要とされる変換は，高々アファイン変換までであろう。

アファイン変換は，線型変換と平行移動との和として定義されるが，前者をより単純な幾つかの基本変換の積に分解することを考える。各々の変換は，整数演算とシフト操作のみからなる直線発生器を用いて実現できるため，これによって高速なアルゴリズムを得ることができる。対話型の環境では，出力の品質をある程度犠牲にしても，高速な処理が要求されることがしばしばあるので，こういった高速化のための方式について考察を加えることには意味がある。

デジタル画像の線型変換（もしくは回転）を，基本変換の積に分解して実行する方式それ自体については，かなり以前から提案がなされており [Casey71]，それ以降も現在に至るまで幾つかの研究がある [Weiman80] [田畑86]。[Weiman80] では，処理の並列化を考慮しながら，濃淡画像を滑かに拡大・縮小又は回転する手法が述べられている。また，主走査及び副走査両方向のアクセスが可能な画像メモリを持ったハードウェア構成上での効果的な実現方法についても，報告されている [田畑86]。

ところが，従来の研究においては，次の二つの点が考慮されていない。すなわち，線型変換の分解には互いに同値な24通りの組合せが存在し，アルゴリズムの実装時には，この分解順序の違いが出力画像の品質に影響を及ぼす。また，このとき特定の組合せを採ることによって，メモリの利用効率の高い単一ラスタ走査アルゴリズムが導かれる。

本研究では，主にこれらの点について検討を行なった結果として，高能率な画像回転アルゴリズムT^2D^2分解法を提案する（図1）。このアルゴリズムは，小型のワークステーションやパーソナルコンピュータ等，特別な画像処理用のハードウェアによる支援のない環境下での実装に適している。

T^2D^2分解法について

基本変換とその直線発生器による実現

まず，与えられた平面上の線型変換の分解に関して，以下の2つの変換を基本的なものであるとして，定義しておく [ディユドネ71]。

- 膨張 (dilatation)

$$\begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix} \qquad (2.1.1)$$

- 移換 (transvection)

$$\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ \mu & 1 \end{pmatrix} \qquad (2.1.2)$$

膨張（いわゆる拡大・縮小）は，平面上のある矩形領域を，いずれか一方の幅を変えずに他の矩形領域に写すのに対し（図2a），移換は，ある矩形領域を，一方の幅を変えずにある平行四辺形領域に写すような変換である（図2b）。この 2つの変換が基本的であるというのは，これらがいわゆる行列の基本変形，すなわち，

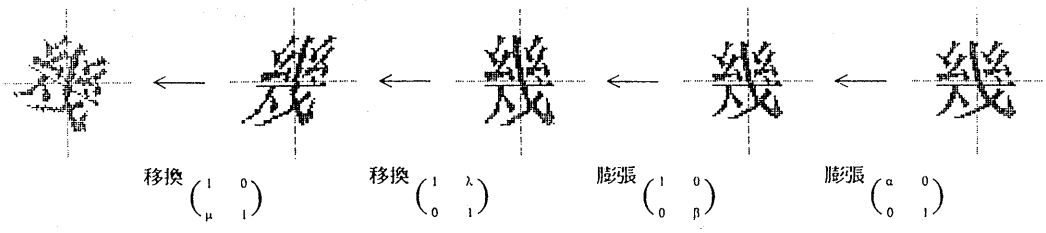


図1 T^2D^2分解法

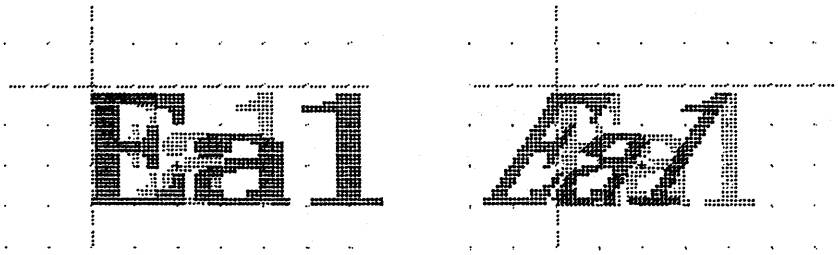


図2 基本変換[1]　　(a) 膨張　　(b) 移換

- ある行（または列）を α 倍する。
- ある行（または列）に，他の行（または列）の λ 倍を加える。

の表現行列になっていることによる（以降の文中では，式　(2.1.1)，　(2.1.2)　の代わりに記法 $D_{x,\alpha}$，$D_{y,\beta}$，および $T_{x,\lambda}$，$T_{y,\mu}$ を用いることにする）。重要なのは，これらの基本変換が直線発生アルゴリズムを用いて高速に実現できることである。以下，それについて詳しく説明する。

直線発生器は，もともとコンピュータグラフィックスの分野において，直交座標系の格子点上に誤差最小に離散化された直線を，微分解析器の原理を用いて高速に発生させるものである。加減算とシフトのみを含む直線発生のアルゴリズムは，傾きが n/m（ただし n<m）の直線に対して，

```
static int y, e;

initLine (m, n)
int m, n;
{
     y = 0;
     e = (n << 1) - m;
}

genLine()
{
     if (e >= 0) {
          y ++;
          e += (n - m) << 1;
     }
     else
          e += n << 1;
}
```

で与えられる[Bresenham65]。ここで，initLine(n, m) によって初期化された直線発生器は，それ以降 genLine() が呼出されるたびに，直線の y 座標値を順次発生していくものとする。このとき得られる傾き n/m の直線を，n 個の画素列と m 個との画素列間の対応付けのための規則ともみることができる（図3）。

これは，前章において述べた，逆写像を求めて幾何変換を行なうアルゴリズムの一つである最近隣法を，高速に，しかも丸め誤差最小に実行していることに他ならない。すなわち，膨張 $D_{x,m/n}$ に対してその逆変換 $D_{x,n/m}$ は，変換画像領域内の x′ から原画像領域内の x への逆写像である。したがって，次の漸化式から数列 $\{x_i\}$ を決めることができる。

$$x_{i+1} = \frac{n}{m} x'_{i+1} = \frac{n}{m}(x'_i + 1)$$
$$= \frac{n}{m} x'_i + \frac{n}{m}$$
$$= x_i + \frac{n}{m}$$

これは，傾きが n/m の直線を発生することと等価である。（実際には，膨張 $D_{x,n/m}$ の倍率 n/m は実数で与えられることが多いので，例えば，連分数展開によるなどして，これの最良近似分数を求めてやる必要がある）

なお，膨張の倍率が1以下になる場合，この方法によれば発生される直線の傾きが 45° を越えるために，変換の過程で画素の間引きが起こり，原画像の情報が失われることになる。これを回避する意味で，ここでは縮小時に画素単位の論理和をとることにするが，そうすると今度は，倍率の値が小さくなるに従って黒画素のツブレを生じ，逆に白画素の情報が消失するという問題が出てくる。いま，変換の種類を回転に限るものとすれば，論理和をとるこの方式は，次のような理由によって正当化できる。

すなわち，回転においては，膨張の倍率の小さい方が $\cos\theta$ で与えられるが，これは $\theta = \pm 45°$ のとき最小で（$|\theta| \leqq 45°$），このとき，

$$\cos 45° = 0.707107 \ldots \fallingdotseq \frac{70}{99}$$

である。傾きが 70/99 の直線では，最も多いところでも2画素の論理和がとられることにしかならないため，実用上支障をきたす程の黒画素のツブレは起こり得ないことがわかる。

1　本来、「膨張」「移換」という用語は、これらの行列に相似な表現行列を持つ写像まで含めた、もう少し広い変換のクラスを指す。

図3

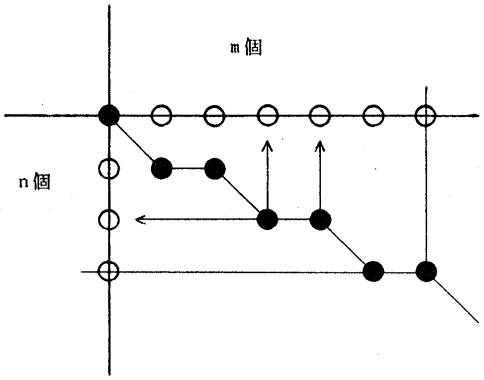


図4

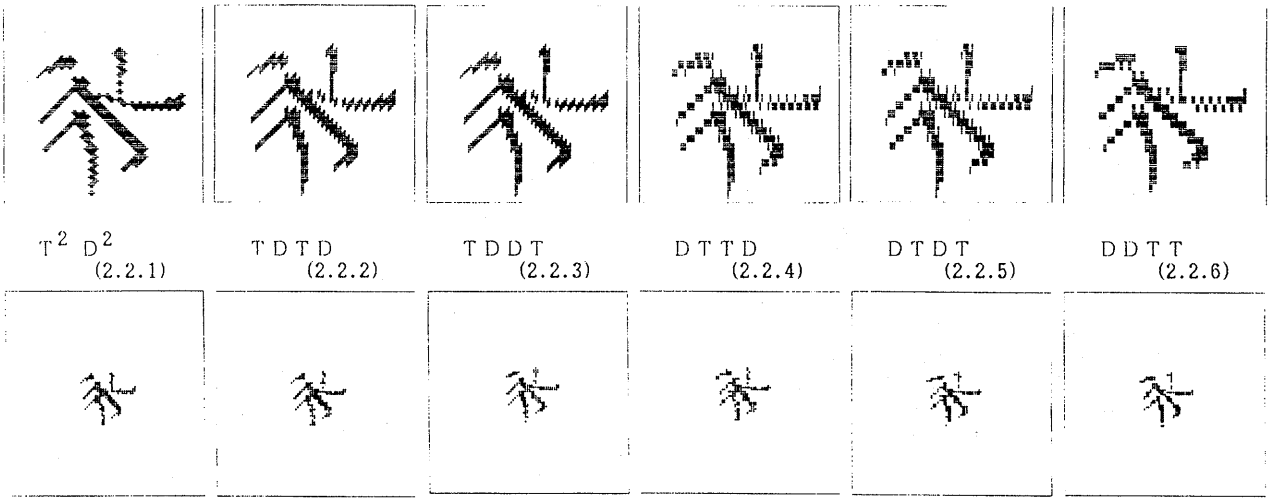
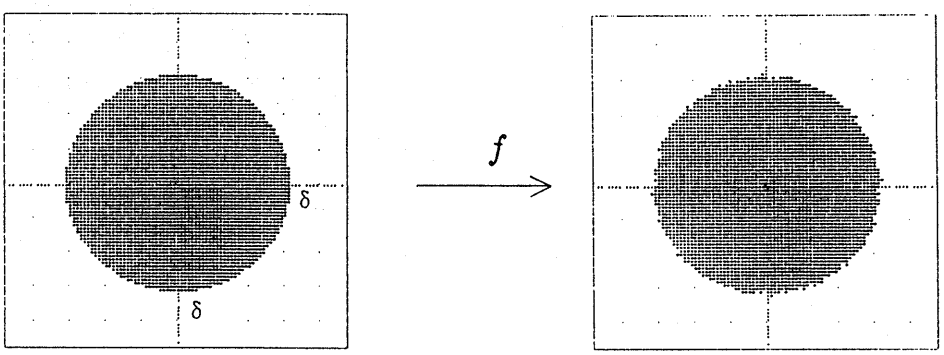


図5



$$h_\delta(x) = \begin{cases} 1, & \|x\|_2 \leq \delta, \\ 0, & \text{その他で} \end{cases}$$

図6　2変数ステップ関数と回転アルゴリズムのステップ応答

膨張に対して移換は，同じ直線発生器を用いてより直接的に実現することができる。図2b　に示すように，水平あるいは垂直方向の一連の画素は，移換によってその方向にそろってシフトされる。このときのシフト量は，変換の傾きに沿って発生された直線の座標値から得ることができる。

分解順序の相違による変換誤差の影響

平面上の任意の線型変換は，その表現行列が特異でない限り，前節において述べた4つの基本変換，すなわち2つの膨張と2つの移換との積に分解することができる。このときの分解の仕方は一意ではなく，積の順序から　4!＝24通りの組合せが存在する。ここで，連続する2つの膨張が（離散空間上においても）可換であることと，x と y について対称な組合せの重複とを考慮すると，以下に示した6通りの分解が本質的である。

$$T_{y,\frac{c}{a}} \times T_{x,\frac{ab}{ad-bc}} \times D_{y,\frac{ad-bc}{a}} \times D_{x,a} \qquad (2.2.1)$$

$$T_{y,\frac{c}{a}} \times D_{y,\frac{ad-bc}{a}} \times T_{x,b} \times D_{x,a} \qquad (2.2.2)$$

$$T_{y,\frac{c}{a}} \times D_{y,\frac{ad-bc}{a}} \times D_{x,a} \times T_{x,\frac{b}{a}} \qquad (2.2.3)$$

$$D_{y,\frac{ad-bc}{a}} \times T_{y,\frac{c}{ad-bc}} \times T_{x,b} \times D_{x,a} \qquad (2.2.4)$$

$$D_{y,\frac{ad-bc}{a}} \times T_{y,\frac{c}{ad-bc}} \times D_{x,a} \times T_{x,\frac{b}{a}} \qquad (2.2.5)$$

$$D_{y,\frac{ad-bc}{a}} \times D_{x,a} \times T_{y,\frac{ac}{ad-bc}} \times T_{x,\frac{b}{a}} \qquad (2.2.6)$$

ちなみに，[Casey71] [Weiman80] [田畑86] で扱われている分解の順序は，このうちの　(2.2.5), (2.2.6), (2.2.2) にそれぞれ相当する。

図4は，変換 $\begin{pmatrix} 2 & 2 \\ -2 & 2 \end{pmatrix}$ に関する上記の分解 (2.2.1) ～ (2.2.6) を，24ドットの文字フォントに対して適用したものである。これらから，互いに同値な，順序の異なる分解によって得られる出力結果は，必ずしも同一にはならないことがわかる。（分解 (2.2.1) において画質の劣化が最も少ないことは，この図からほとんど明らかである。したがって，拡大の含まれている回転処理を行なう場合には，分解 (2.2.1) 以外には選択の余地はないとも言える。）

このような分解順序による出力結果の違いは，移換によって離散的に変換された画素の位置に関する丸め誤差が，その後の膨張によって拡大（膨張の倍率が　1以下のこともあり得る）されることに起因する。図7に，このときのいくつかの状況を示す。

単一ラスタ走査アルゴリズムによる実現

前節までで，任意の線型変換が2つの膨張と2つの移換との積によって表わされること。各々の変換がよく知られた直線発生器を用いて高速に実現できること。また，分解の順序がアルゴリズムの性能に影響を及ぼし得ることについて見てきた。

このような変換行列を分解するという方式に従う限り，いずれの分解順序を採るにせよ，それぞれの基本変換に対応する4つの処理過程を1つずつカスケード式に実行しなければならず，そのための余分な処理時間と中間結果のための作業領域とを，常に必要とするように思われる。しかし，以下に説明するように，分解 (2.2.1) によれば，単一のラスタ走査によって全ての基本変換が同時に完了する，効率的なアルゴリズムを導くことができる。

個々の基本変換の特性についてもう一度考えてみると，図2a および 2b からも明らかなように，膨張がその変換の前後で画素の数を変え得るのに対して，移換は画素数を変えないことがわかる。言い換えれば，移換の入力画像と出力画像との間には，画素の1対1対応が存在する。図8は，単一ラスタ走査アルゴリズムによって，矩形領域の30°回転が行なわれる過程について示したものである。分解 (2.2.1) においては，　2つの膨張が　2つの移換の前に行なわれる。すなわち，膨張によってある画素の位置が決定された（a → a′）直後に，その画素の x, y 座標に対して，それぞれ水平，垂直方向の移換の値を加えることができる（a′ → a″）。あとは，この操作を膨張後の全画素について反復すればよい。

分解 (2.2.1) の順序に依存した，この単一ラスタ走査アルゴリズムは，　T^2D^2分解法に固有のものである。画素数を変化させる膨張が，移換の後にくるようなそれ以外の分解に対しては，このアルゴリズムを適用することはできない。

この単一ラスタ走査アルゴリズムは，膨張について2つ，移換について 2つ，合計 4つの独立した直線発生器の組合せることによって，プログラムとしては単一の二重ループ構造で実現することができる。この部分について，以下に示す（$\cos\theta \fallingdotseq p/q$, $\sin\theta \fallingdotseq r/s$ とする）。

```
 . . .

initLineII(p, q, 0);
initLineIV(r*q, s*p, 0);

for (yi = 0; yi < yn; ++yi) {
      . . .

     initLineI(q, p, 0);
     initLineIII(q*s, -p*r, yIV);
       /* 水平方向の移換の値で初期化 */

     for (xi = 0; xi < xn; ++xi) {
        . . .
```

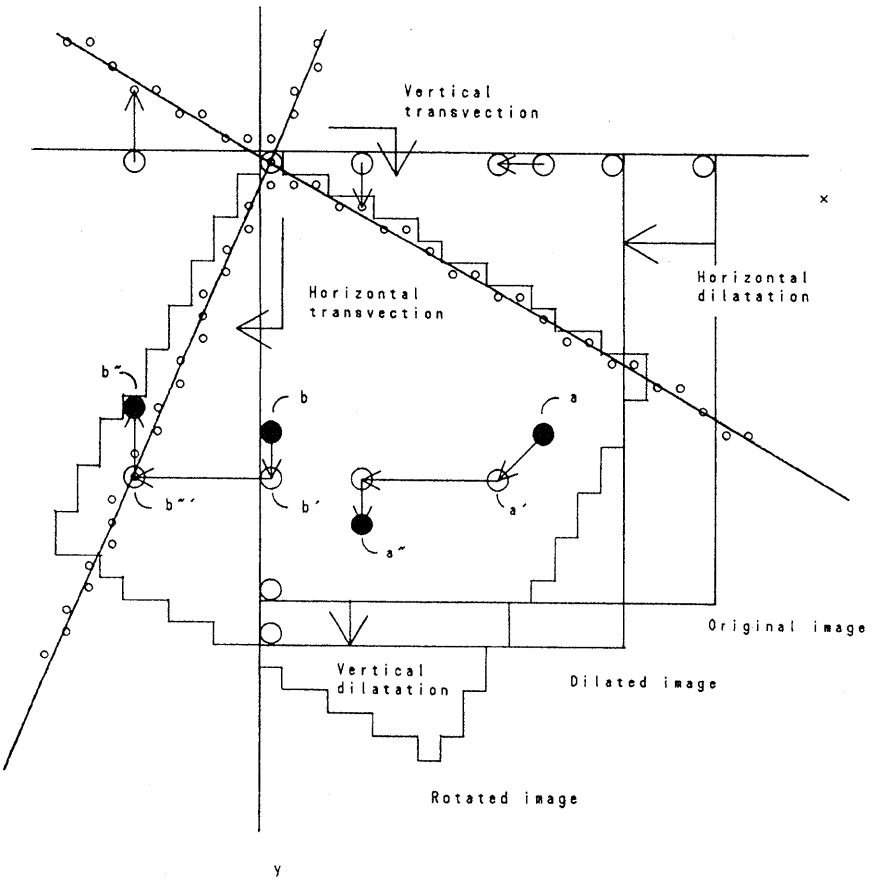


図8 単一ラスタ走査アルゴリズム

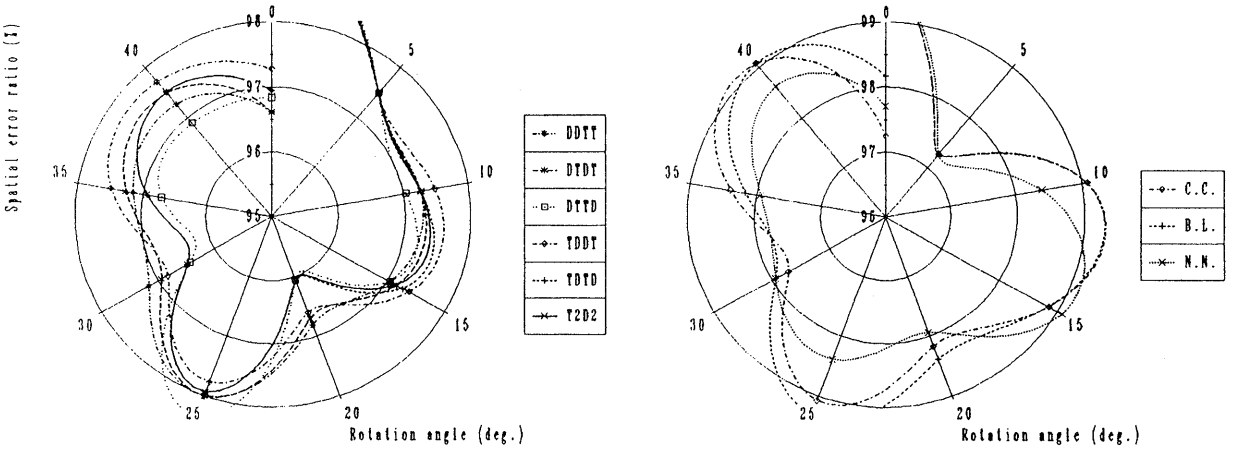


図9 評価結果 (a) (b)

```
        if (getPixel(yII, yI))
          setPixel(xi+yIV, yi+yIII);
        . . .

        genLineI();
        genLineIII();
      }

      genLineII();
      genLineIV();
}

 . . .
```

ここで注意すべきことは，垂直方向の移換を行なう直線発生器は，必ずしも原点において初期化されるとは限らない，ということである（上のプログラムでは，関数 initLine???() の新たな三番目の引数が，初期化される座標を与えている）。再び図8 で，ラスタ方向の先頭に位置している画素，例えば b をとると，連続する 2つの膨張によって b の x 座標は変化しないが（ b → b′），水平方向の移換によってこれが変えられてしまう（ b′ → b″）。したがって，垂直方向の移換を実行する直線発生器は，画素 b″ の持つ x 座標で初期化されなければならない。この問題を解決するために，任意の初期値をとれるような直線発生器を，整数演算の範囲内で構成することができるが，ここではその方法については述べない。

なお，各方式による 256 × 256 画素の画像回転に要する処理時間の比較は，次のとおり（IBM 5560 の画面メモリ上での実測値の平均。使用プロセッサーは I80286-8MHz で，I80287 は使用せず）。

T^2D^2分解法	11秒
最近隣法	93秒
双一次補間法	198秒
三次畳み込み法	507秒

画像回転への適用

これまでは，主に平面上の線型変換全体，すなわち一般の 2次行列の分解について考えてきた。しかし，変換行列の行列式の値が 1より大きくなるに従って，基本変換の標本化誤差とともに，最近隣法（0 次補間）による補間誤差の影響が無視できなくなってくる。いま分解の対象を，線型変換の特別な場合である回転に限るものとすれば，この影響は比較的小さいままである。したがって，変換行列の分解による高速なアルゴリズムが，品質の面においても，実用的な意味をもつ可能性がある。

図5 は，図4 と同じ文字フォントに対して，分解 (2.2.1) ～ (2.2.6) による 45 ° の回転を行なったものである。分解順序による出力結果の差異は認められるが，図4 の場合ほど顕著ではなく，アルゴリズムの性能を評価するまでには至らない。

したがって，あるアルゴリズムの出力画像における品質を，入力画像の内容に依らずに，定量的に評価することが可能なのかどうかが，新たな問題点として出てくる。

T^2D^2分解法の性能評価

画像評価のための手法

計算機上に実装されたアルゴリズムの性能は，その処理速度，資源の利用効率，入力の容易さ及び出力の品質等によって決定される。したがって，これらの要因を客観的に評価するための手法の確立が，アルゴリズムの開発の次に重要である。一般に，あるアルゴリズムが高速かつ高能率であることの定量的な検証は比較的容易に行なえるのに対して，そのアルゴリズムによって得られた出力が高品質であるか否かの評価は，特に対象となる情報媒体が画像の場合には，困難であることが多い。

二値画像に限らず，一般の濃淡画像に対する幾何変換（補間）方式の性能評価については，これまでにも種々の方法が提案され使用されてきている。これらの方法は，実際の評価を画像の周波数領域で行なうもの [Schafer73]と，空間領域で行なうもの [Stucki79] [Abdou82]とに大別することができる。

[Schafer73] では，補間方式をデジタルフィルタとしてモデル化し，帯域制限された入力信号に対する周波数応答から，幾つかの線型補間関数の性能評価を行なっている。 [Stucki79] では，帯域制限された濃淡画像に対して，粗い標本化を行ない，補間によって再構成した画像の平均二乗誤差から，画素密度変換方式としての性能を評価している。また [Abdou82] では，二値画像に対する画素密度変換方式の性能について，入力された基本信号（ステップ関数）から元の信号を再構成したときの空間的な誤差を算出することで，標本化や量子化誤差を考慮に入れた解析的な評価を試みている。

しかし，これまでの研究では，画像回転アルゴリズム一般に適用可能な，客観的な性能評価の手法については言及されていない。

画像回転アルゴリズムに対する評価手法

対象画像が二値の場合には，信号として帯域制限されていないことなどから，空間領域で評価が行なわれるのが普通である。人間が知覚する二値画像の品質も，やはり，その画像を構成している要素の空間的な誤差に依存していると考えられる。そのため，このような評価手法は，人間の視覚系まで含めた評価尺度としても意味がある。

T^2D^2分解法の性能評価を行なうための前提として，

回転も含めた一般的な画像の変換系をモデル化することから考える。

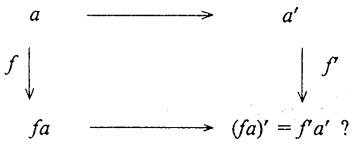


図10 画像変換系のモデル

ある画像 a が，標本化及び量子化の過程を経て入力され（a′），評価すべきある変換アルゴリズム f′ によって離散的に変換されたものを f′a′ とする。一方，原画像に対する誤差のない仮想的な変換系の出力を fa として，これを系に入力したものを (fa)′ で表わせば，この (fa)′ に対する f′a′ の評価がなされることになる。

変換行列の分解によって回転を行なう系は，大域的にみれば，線型かつ位置不変とみなすことができるから，ある基本的な信号，例えばステップ関数に対する系の応答をみれば，原理的にはその系に対する十分な評価が得られるはずである。ところで，回転は平面内の変換であるので，一次元の入力信号を用いたモデルで系を簡単化して記述することができない。したがって，2変数のステップ関数を新たに定義しなければならないが，ここではこれを，理想的な回転に対して形状が不変（空間的な誤差が零）になるように選び，これに対する系の応答を調べることにする（図6）。

上記の変換系のモデルに関する説明では，理想的な系の出力である (fa)′ の存在が不明であった。しかし，入力信号をこのように選ぶことによって，常に a′＝(fa)′ の関係が成立するので，我々は a′ と f′a′ との間で評価を行なえばよく，入力時の標本化及び量子化（二値化）の際に発生する誤差の問題を，ひとまず切り放して考えることができる。

評価結果と考察

実際の性能評価は，ステップ関数の半径 δ と回転角 θ について，各回転アルゴリズム (2.2.1) ～ (2.2.6) のステップ応答の平均二乗誤差から決定するものとする。これは，入力のステップ関数と出力のステップ応答との対称差をとった画像の画素数を，ステップ関数の画素数で正規化したもの，

$$E_{\delta,\theta} = \frac{\#(f_\theta(H_\delta) \triangle H_\delta)}{\#H_\delta} \times 100$$

を求めることによる。

図9a は分解 (2.2.1) ～ (2.2.6) について，図9b は最近隣法，双一次補間法と三次畳み込みに対する，δ = 23 のときの評価結果である。

この結果から，6通りの分解のうちでは (2.2.3)，(2.2.5) が良く，(2.2.4) は悪い。T^2D^2分解法 (2.2.1) は，平均的な中の上の性能を示している。

おわりに

空間効率の高い画像回転アルゴリズム，T^2D^2分解法の提案と，その実装のための詳細について報告した。また，画像変換系をモデル化し，回転アルゴリズムの性能評価を行なうための一手法を提案した。種々の回転アルゴリズムに適用した評価結果から，T^2D^2分解法が，画質の面においても，実用的な性能を示すことが判った。

謝辞

日頃よりご指導頂く当研究所松下武史担当に感謝致します。（なお，本稿の作成には，当研究所の山内長承研究員による JANUS の漢字タグを利用した。）

文献

[Casey 71] R.G. Casey and M.A. Wesley: Parallel linear transformationson two-dimensional binary images. IBM Tech. Disclosure Bull., Vol. 13, No. 11, pp. 3267-3268 (Apr. 1971).

[Weiman 80] C.F.R. Weiman: Continuous anti-aliased rotation and zoom of raster images. SIGGRAPH'80 Conf. Proc., Vol. 14, No. 3, pp. 286-293 (July 1980).

[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).

[Schafer 73] R.W. Schafer and L.R. Rabiner: A Digital Signal Processing Approach to Interpolation. Proc. IEEE, Vol. 61, No. 6, pp. 692-702 (Jun. 1973).

[Stucki 79] P. Stucki: Image Processing for Document Reproduction. Advances in Digital Image Processing, P. Stucki, Ed., Plenum Press, New York, pp. 177-201 (1979).

[Abdou 82] I.E. Abdou and K.Y. Wong: Analysis of Linear Interpolation Schemes for Bi-Level Image Applications. IBM J. Res. Develop., Vol. 26, No. 6, pp. 667-680 (Nov. 1982).