数理解析研究所講究録
第872巻 1994年 1-15



# ディジタル画像における直線成分抽出のためのアルゴリズム

浅野哲夫
Tetsuo Asano
大阪電気通信大学

加藤直樹 Naoki Katoh
神戸商大

ディジタル画像から直線成分を検出する問題は、パターン認識やロボット・ビジョンなどにおける基本的な問題である。この目的のために双対平面における投票に基づくHough変換法が広く利用されているようであるが、計算複雑度と直線抽出能力の関係を理論的に解析した研究は少ない。本論文では、ディジタル画像に含まれる極大な直線成分をもれなく確実に抽出するアルゴリズムを提案する。整数論の定理に基づいて効率を改善しているのが特徴である。上記の直線抽出能力を達成するために最低限必要な時間で処理を終えることができる。

The problem of detecting lines components in a digital image is one of the most fundamental problems in pattern recognition and robot vision. Hough transform which is based on voting in the dual plane is widely used for this purpose. However, there have been few theoretical studies on the relationship between its computational complexity and ability of detecting straight lines. In this paper we present an algorithm for detecting every maximal line component contained in a digital image. This algorithm is designed to run efficiently based on Number Theory. It can complete the required task in least time needed to achieve the above-mentioned ability of line detection.

## 1 まえがき

ディジタル画像中に存在する直線成分を抽出するための手法としては、画像中のエッジ点を三角関数で表現される曲線に変換してその交点を検出するHough変換法（Hough Transform）[(1),(2)]が現在では最も広く用いられているようである。この方法の本質はエッジ点に対応する曲線の交点を列挙することにあるが、全ての交点を正確に列挙するのではなく、細かく分割されたパラメータ空間に「投票」を行うという形で実現されるのが普通である。直線成分の抽出精度を上げるためには分割を細かくしなければならないが、細分化は計算時間と記憶スペースの増大を招くので、分割のためのパラメータは慎重に選ぶ必要がある。しかしながら、どのような分割を行うのが望ましいかに関しては厳密に議論した研究はほとんどないのが実状である（たとえば、松山、輿水によるサーベイ論文[(3)]参照）。

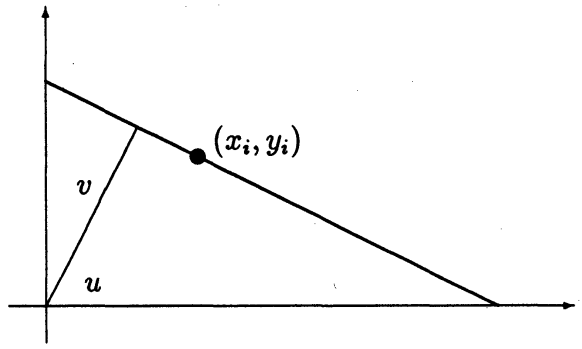


図 1: u-v transformation.

本論文では、$N \times N$のサイズをもつディジタル画像に含まれる極大な直線成分をもれなく確実に抽出するアルゴリズムを提案する。その計算時間は $O(N^4)$ である。整数論の定理を用いると、アルゴリズムが確実であることを証明できるが、さらにどのような確実なアルゴリズムも$\Omega(N^4)$ の時間が必要であることも示せるから、このアルゴリズムは計算時間のオーダーに関して最適である。

## 2 標準的な Hough 変換法

最初に標準的な Hough 変換法について説明しよう。

サイズ $N \times N$の入力のディジタル画像に対応する整数格子点の集合を

$G = \{(x_i, y_i) |\ x_i = 0, 1, \ldots, N-1,\ y_i = 0, 1, \ldots, N-1\}$

とし、整数格子点 $(x_i, y_i)$, $0 \leq x_i < N$, $0 \leq y_i < N$に点（ドット）が存在する場合（以下、**エッジ点**という）には $g(x_i, y_i) = 1$ と定め、存在しない場合には $g(x_i, y_i) = 0$ と定める。以下、本論文では関数 $g(x, y) = 1$ なる格子点（エッジ点）の集合で構成されるディジタル直線を抽出することを考える。また、通常は縦と横のサイズの異なる画像を想定するべきであるが、本論文では簡単のため縦横のサイズは等しいものと仮定する。

さて、原点からエッジ点 $(x_i, y_i)$ を通る角度 $u$ の直線までの距離を $v$とすると、図 1 より $v$は次式で表現できることがわかる。

$$v = x_i \cos u + y_i \sin u. \tag{1}$$

エッジ点 $(x_i, y_i)$ に対する曲線 $v = x_i \cos u + y_i \sin u$ と $(x_j, y_j)$ に対する曲線 $v = x_j \cos u + y_j \sin u$ が 1 点 $(u', v')$ で交差するとき、この 2 点を含む直線の式は

$$x \cos u' + y \sin u' = v'. \tag{2}$$

で与えられる。

一般に、このような $m$ 個の曲線が点 $(u', v')$ で交差するなら、対応する $m$ 個の点は (2) 式の直線上にあることになる。このような交点を求めるために、標準的な Hough 変換法ではパラメータ空間に対応する 2 次元の配列を用意し、この上で曲線 $v = x_i \cos u + y_i \sin u$ を辿りながら曲線上の要素に「投票」をしていき、最後に投票数がある定められたしきい値を超えた点に対応する直線を出力する。

以下に具体的なアルゴリズムの記述を与える。

**[アルゴリズム 1 ：標準的な Hough 変換法]**

**必要な配列：**
(1) bit $g[N][N]$; /* 画像データ */
(2) int count$[N_u][N_v]$; /* 投票数のカウント用 */
/* ただし、画像データは逐次的にしか参照されないので、ファイルからデータを読みとる方式でもよい。 */
**予め決定しておくもの：**
(1) $(u_1, u_2, \ldots, u_M)$: /* 直線の傾きの変化のさせかた */
(2) $v$の値を量子化するための関数 $q(u, v)$.
(3) ディジタル直線を構成するエッジ点集合のサイズに関するしきい値 $t$.

**具体的な手続き：**

配列 count[.][.] の要素を全て 0 に初期化する;
for $x = 0$ to $N - 1$
  for $y = 0$ to $N - 1$
    if 場所 $(x, y)$ にエッジ点が存在する then
      for $i = 1$ to $M$
        $v = x\cos u_i + y\sin u_i$を計算;
        count$[i][q[u_i, v]]$ の値を 1 増やす;
count$[u_i][v_j] \geq t$ である $(u_i, v_j)$ を列挙し、対応する直線 $x\cos u_i + y\sin u_i = v_j$を出力する;

上に述べたアルゴリズム 1 では、(1) 直線の傾きの変化のさせ方を表わす傾きの系列 $(u_1, u_2, \ldots, u_M)$、(2) $v$の値を量子化するための関数 $q(u, v)$、および (3) ディジタル直線を構成するエッジ点集合のサイズに関するしきい値 $t$ を予め決定しておかなければならない。このうち、$v$に関する量子化関数 $q(u, v)$ については、藤井[4]らによって

$$\delta(u) = \max\{|\sin u|, |\cos u|\} \tag{3}$$

で定まるサンプリング間隔$\delta(u)$ で $v$軸を量子化すれば、すなわち、

$$q(u, v) = \lfloor v/\delta(u) \rfloor, \tag{4}$$

のように定めれば良好な結果が得られると報告されている。ただし、上式では簡単のため傾きが 0 ～ 1 の直線への当てはめだけを考えているので、$v$の値は非負である。実用的には上記の量子化関数で十分かも知れないが、量子化を行なう以上どこかに境界ができるので、どんな場合でも必ず所望のディジタル直線が抽出できるという保証はできない。

直線の傾きをどのように変化させるべきかについては、筆者の知る限り合理的な結果は得られていないようである。この問題については第 4 章において詳しく論じる。

## 3 v軸に関する量子化を用いない直線抽出法

前節で述べたように、$v$軸に関する量子化を用いる限り、量子化の境界の影響によって抽出不可能になるディジタル直線が生じることは避けられない。本章では、この問題を解決するために、$v$軸に関する量子化も投票も行なわない方法を提案する。

先のアルゴリズムとの違いの一つは、ディジタル画像に含まれるエッジ点を列挙しておいて、その座標データを別の配列によって蓄えておく点である。ディジタル画像に含まれるエッジ点の密度が高い場合には、このための配列は高価なものとなろうが、標準的なHough変換のアルゴリズムで使用する2次元の投票用の配列は不要になる。

第二の違いは、ループの順序が入れ替わっていることである。すなわち、アルゴリズム1では各点について角度$u$を順に変化させて投票を行い、投票がすべて終わった後、一定数以上の得票があったものを列挙して、それらに対応する直線を出力する。これに対して、今度は角度$u$を$u_1, u_2, \cdots$と順に変化させながら、それぞれの角度について全てのエッジ点に対する$v$の値を計算し、$v$の値に関してソートを行い、同じような$v$の値をもつエッジ点が一定個数以上あれば、それらに対応する直線を直ちに出力する。

**[アルゴリズム2]**

**必要な配列：**

(1) bit $g[N][N]$; /* 画像データ */
(2) struct point{ int x, y;} p[n]; /* エッジ点の座標 */
(3) double v[n]; /* $v$の値を蓄える配列 */
(4) int $\sigma[n]$; /* エッジ点の番号 */
/* $n$はエッジ点の総数 */

**予め決定しておくもの：**

(1) $(u_1, u_2, \ldots, u_M)$: /* 直線の傾きの変化のさせかた. */
(2) ディジタル直線を構成するエッジ点集合のサイズに関するしきい値$t$.
(3) ディジタル直線として許容する$v$の値の差に関するしきい値$\epsilon$.

**具体的な手続き：**

$g[x][y] = 1$である座標$(x, y)$を構造体配列p[.]に蓄える;
配列$\sigma$[.]を次のように初期設定する;
for $i = 1$ to n $\sigma[i] = i$;
for $i = 1$ to $M$
  for $j = 1$ to $n$
    $v[j] = p[j].x \cdot \cos u_i + p[j].y \cdot \sin u_i$を計算;
  $v[\sigma[1]] \leq v[\sigma[2]] \leq \cdots \leq v[\sigma[n]]$ となるように
  ソートする;

上記のソート列をスキャンして、$v[\sigma[i+t]] - v[\sigma[i]] \leq \epsilon$である$i$が存在すれば、番号$\sigma[i] \sim \sigma[i+t]$のエッジ点を直線成分として出力する;

上記のアルゴリズム2において、ディジタル直線として許容する$v$の値の差に関するしきい値$\epsilon$をどのように定めるかについては後で論じるが、上記のアルゴリズム2が先の標準的なHough変換のアルゴリズム1よりも能力的に優れていることは明かであろう。すなわち、しきい値$\epsilon$を適切に設定することができれば、あるディジタル直線に対応するエッジ点集合が（単独で与えられたとき）アルゴリズム1で抽出できるならば、そのエッジ点

集合はアルゴリズム2でも必ず抽出できることを示すことができる。この点については次節において詳しく論じる。

# 4 直線抽出の能力評価

ディジタル画像から直線を抽出するためのアルゴリズムとして様々なものが考えられるが、それらのアルゴリズムの直線抽出の能力を評価するために、まず以下の定義を用意し、能力評価のための基準について述べよう。

先にも定義したように、ディジタル画像に対応するエッジ点の集合を

$G = \{(x,y) | x = 0,1,\ldots,N-1,\ y = 0,1,\ldots,N-1\}$

と定める。$x$ 座標が $x$ に等しい垂直線を $V_x$、$y$座標が $y$に等しい水平線を $H_y$と書くことにする。

**[定義1]** $P$をエッジ点の集合とし、それらのエッジ点の $x$ 座標の最小値と最大値をそれぞれ $x_{max}, x_{min}$とする。このとき、どの $x = x_{min},\ldots,x_{max}$に対しても $P \cap V_x \neq \phi$が成り立つとき、$P$は**水平方向に密である**という。また、どの $y = y_{min},\ldots,y_{max}$に対しても $P \cap H_y \neq \phi$であるとき、$P$は**垂直方向に密である**という。ただし、$y_{min}, y_{max}$は $P$のエッジ点の $y$座標の最小値と最大値である。

上の定義によると、画像の角の一点だけからなる集合も密であるということができるが、無用の些細な議論を避けるため、本論文では少なくとも3点を含む集合だけを想定している。

ここまではディジタル直線という用語を定義せずに用いてきたが、これを以下のように定義する。もちろん、これはディジタル直線の定義の一例に過ぎないが、定義を微修正しても同じ議論が成り立つものと考えられる。

**[定義2]** 直線 $y = ax + b$ ($a, b$は実数) に対応するエッジ点の集合 $G(a,b)$ を次のように定める。また、ある直線に対応するエッジ点の集合を**ディジタル直線**という。

$$G(a,b) = \begin{cases} \{(x_i,y_i) \in G |\ |y_i - ax_i - b| \leq 0.5\} & -1 \leq a \leq 1 \\ \{(x_i,y_i) \in G |\ |x_i - \frac{1}{a}(y_i - b)| \leq 0.5\} & |a| > 1 \end{cases}$$

このようにディジタル直線を定義するとき、明らかに次の補題が成り立つ。

**[補題1]** 水平方向に密なエッジ点の集合 $P$が

$\max_{(x_i,y_i)\in P}(y_i - a * x_i) - \min_{(x_j,y_j)\in P}(y_j - a * x_j) \leq 1$

を満たすならば、$P$は直線 $y = ax + b$ に対応するディジタル直線 $G(a,b)$ に等しい。ただし、

$b = [\max_{(x_i,y_i)\in P}(y_i - a * x_i) + \min_{(x_j,y_j)\in P}(y_j - a * x_j)]/2$

である。

**[補題2]** 垂直方向に密なエッジ点の集合 $P$が

$\max_{(x_i,y_i)\in P}(x_i - y_i/a) - \min_{(x_j,y_j)\in P}(x_j - y_j/a) \leq 1$

を満たすならば、$P$は直線 $y = ax + b$ に対応するディジタル直線 $G(a,b)$ に等しい。ただし、

$b = -a * [\max_{(x_i,y_i)\in P}(x_i - y_i/a) + \min_{(x_j,y_j)\in P}(x_j - y_j/a)]/2$

である。

本論文では、直線の式が与えられときに対応するエッジ点の集合を求めるというのではなく、直線に対応しそうなエッジ点の集合を検出して、その集合に最もよくあてはまる直線の式を求めることを目的としている。そこで、極大直線成分と呼ばれる性質を定義する。

[**定義３**] エッジ点の集合 $P$に対して下の (1) または (2) の条件を満たす実数 $u$ が存在するが、$P$を真部分集合として含むどのようなエッジ点の集合についてもそのような実数 $u$ は存在しないとき、$P$は**極大直線成分**という。

(1) $\max_{(x_i,y_i)\in P}(y_i - u * x_i) - \min_{(x_j,y_j)\in P}(y_j - u * x_j) \leq 1,\ |u| \leq 1.$

(2) $\max_{(x_i,y_i)\in P}(x_i - y_i/u) - \min_{(x_j,y_j)\in P}(x_j - y_j/u) \leq 1,\ |u| > 1.$

以上の準備の下に、本節の最初に述べたアルゴリズムの直線抽出能力に関する基準を説明しよう。

[**定義４**] エッジ点の集合 $S$としきい値 $t$ が与えられたとき、サイズが $t$ 以上の $S$のすべての極大直線成分を抽出することができるアルゴリズムは**確実**であるという。

このようにアルゴリズムの能力を定義するとき、次の補題は明かであろう。

[**補題３**] 標準的な Hough 変換のアルゴリズムは確実でない。

それでは $v$に関する量子化を行わないアルゴリズム２は確実であろうか？ディジタル直線の定義と補題１、２より、ディジタル直線として許容する $v$の値の差に関するしきい値 $\epsilon$を適切に選んでやれば、アルゴリズム２の出力は必ず何らかの直線に対応するエッジ点の集合にすることができる。すべての極大直線成分を抽出できるかどうかは、直線の傾きをどのように変化させていくかと、しきい値$\epsilon$の決め方に依存する。もちろん、直線の傾きをごく微小量ずつ変化させていけばよいが、計算時間は変化の回数の比例して増大することを忘れてはならない。問題は、アルゴリズムを確実なものにするためは、直線の傾きを変化させる回数が最低どの程度でなければならないかである。この問題に明快に答えるにはアルゴリズム２の形では難しいので、三角関数を用いない変形を考える。

## 5　新たな変換式

計算幾何学においてよく用いられる双対変換（duality transform）は、点 $(a, b)$ を直線 $y = ax + b$に、直線 $y = cx + d$を点 $(-c, d)$ に変換するものである。このとき、点 $(a_1, b_1)$ に対応する直線 $y = a_1x + b_1$と点 $(a_2, b_2)$ に対応する直線 $y = a_2x + b_2$の交点は、これらの２点を通る直線に対応する。したがって、多数のエッジ点が与えられたとき、これらの点を双対変換によって直線に変換し、$t$ 本以上の直線が交わる交点に対応する直線を求めればよいことになる。しかしながら、この方法ではパラメータ空間において交点が存在する範囲を限定することができないので、アルゴリズム１で用いたような「投票」に基づく方法が利用できず、「実用的でない」と言われてきた[(2)]。その点、前述の (1) 式 ($v = x_i \cos u + y_i \sin u$) による現在の標準的な Hough 変換の表現方法では、$u$ に関しては $0 \sim \pi$に限定することができ、$v$に関しても $|v| \leq 2N$の範囲に限定できる（厳密には、$v$に関してはさらに範囲を限定することができる）。実用的には、このようにパラメータ

空間を狭い範囲に限定できたことがHough変換法の成功の主たる要因ではないかと考えられる。しかしながら、前節で述べたように直線抽出の能力を厳密に考慮すると先に述べた標準的なHough変換法が最善ではない。本節では前節の定義に沿ってアルゴリズムを作り変える。

Houghが考案したHough変換の原形は、傾き$a$、切片$b$によってパラメータ表現された直線$y = ax + b$ を、$(a, b)$で張られるパラメータ空間上で抽出するという方式であった[(1),(3)]。本節で考える方法は、むしろHoughのこの考え方に近いものである。すなわち、標準的なHough変換法では、原点からエッジ点$(x_i, y_i)$を通る角度$u$の直線までの距離を$v$として計算したが、今度はエッジ点$(x_i, y_i)$から原点を通る傾き$u$の直線$y = ux$に至る垂直距離、厳密には

$$v = y_i - u \cdot x_i \tag{5}$$

を計算し、$v$の値に関して点をソートする。このとき、2点$(x_i, y_i)$と$(x_j, y_j)$に対して対応する$v$の値が等しいならば、すなわち、

$$y_i - u \cdot x_i = y_j - u \cdot x_j = v$$

という等式が成り立つならば、$y = ux + v$という直線はこれらの2点を通ることがわかる。したがって、与えられた点について(5)式で定まる$v$の値を計算して、その値でエッジ点をソートして、$v$の値の差が1以内であるような極大のエッジ点集合を求めれば、これがディジタル直線に対応することは明白である。したがって、以下のアルゴリズムを得る。

**[アルゴリズム3]**

**必要な配列：**

(1) bit $g[N][N]$; /* 画像データ */
(2) struct point{ int x, y;} p[n]; /* エッジ点の座標 */
(3) double v[n]; /* $v$の値を蓄える配列 */
(4) int $\sigma[n]$; /* エッジ点の番号 */
/* $n$はエッジ点の総数 */

**予め決定しておくもの：**

(1) $(u_1, u_2, \ldots, u_M)$: /* 直線の傾きの変化のさせかた. */
(2) ディジタル直線を構成するエッジ点集合のサイズに関するしきい値$t$.

**具体的な手続き：**

$g[x][y] = 1$である座標を構造体配列 p[.] に蓄える;
配列$\sigma$[.] を次のように初期設定する;
for $i = 1$ to $n$
　$\sigma[i] = i$;
for $i = 1$ to $M$
　for $j = 1$ to $n$
　　$v[j] = p[j].y - u_i \times p[j].x$ を計算;

$v[\sigma[1]] \leq v[\sigma[2]] \leq \cdots \leq v[\sigma[n]]$ となるようにソートし、このソート列に基づいて直線成分を出力する。

アルゴリズム3は先に与えたアルゴリズム2とほぼ同じである。違いを列挙すると、
(1)$v[j]$ の計算式が異なる。
(2)$u_i$は角度ではなく直線の傾きである。
(3) ディジタル直線として許容する$v$の値に関するしきい値が1に固定されている。
(4) ソート列の処理の仕方が異なる。

アルゴリズム2ではエッジ点を$v$の値によってソートしたあと、自分より$t$個先の$v$の値との差が1以内であれば直線成分が見つかったとして出力していたが、$v$の値にタイが多数存在するような場合には同じ直線が何度も出力されることになってしまう。欲しいのは、要素数が$t$以上の直線成分ではなく、要素数が$t$以上の極大直線成分であるから、極大性の判定をどこかで行なう必要がある。

求めたいのは$v$の値が高々1しか違わない範囲にある極大なエッジ点の集合である。これを求めるために、エッジ点を$v$の値に応じてソートしておいて、幅1の区間$[a-1,a]$をスライドさせながら、その区間に含まれるエッジ点の集合を管理する。一つのエッジ点が区間$[a-1,a]$に入るのは、その$v$の値が$a$に等しくなるときであり、この区間から出るのは$v$の値が$a+1$に等しくなるときである。したがって、各エッジ点$\sigma[i]$に対して$(v[\sigma[i]],\sigma[i],\text{"ins"})$と$(v[\sigma[i]]+1,\sigma[i],\text{"del"})$という組を作って、それらの組をソートしておくと、上記のスキャンを効率よく実行することができる。すなわち、第一のキーの順に上記の組を取り出し、ラベルが"ins"なら対応するエッジ番号を集合$P$に加え、"del"なら集合$P$から削除することにより、幅1の区間に含まれるエッジ点の集合を管理することができる。このとき、ラベル"ins"の組と"del"の組が同じ$v$の値をもつ場合には、"ins"の処理を優先しなければならないのは極大集合を求めるためには当然である。この集合が極大になるのは削除が行なわれる直前であるから、削除を行なう前に集合のサイズがしきい値$t$以上であるかどうかを確かめて、もし$t$以上なら対応する直線を出力する。

次にアルゴリズム3ですべての極大直線成分を抽出できるようにするには直線の傾きをどのように変化させていくべきかについて考察しよう。

**[補題4]** アルゴリズム3を確実にするためには$M=\Omega(N^2)$とすることが必要である。

**証明：** 4個の格子点$A(0,0),B(0,1),C(b,a),D(b,a+1)$は明らかに極大直線成分であるが、これは$y=(a/b)x+0.5$という直線にしか対応していないから、この成分を抽出するためには傾きの系列に$a/b$が含まれていなければならない。$a/b,\ a=0,1,\ldots,N-1,\ b=1,2,\ldots,N-1,\ a\leq b$という形式の有理数は全部で$\Omega(N^2)$個存在するから、補題を得る。□

分母、分子とも$N-1$以下の正整数であり、値が1以下の既約分数の増加列を$u_1,u_2,\ldots,u_M$とし、$u_0=0$とする。$u_0,u_1,\ldots,u_M$をアルゴリズム3で用いる直線の傾きの変化のさせかたとする。しきい値は1と定める。このとき、次の補題が成り立つ。

**[補題5]** 傾きの系列$(u_0,u_1,\ldots,n_M)$を、上で定めたものとすると、すべての極大直線成分をアルゴリズム3により確実に抽出することができる。

**[観察1]** $N\times N$の画像内の2つの格子点を通る直線の傾きは全部で$\Theta(N^2)$通り存在する。

**証明：** 文献(5)に$N$が十分大きいとき$M/N^2\to 6/\pi^2$となることが書かれている。□

**[観察2]** $N\times N$の画像内の2つの格子点を通る直線の傾きをソートしたとき、連続する

傾きを $b/a$ および $b'/a'$ とすると（この二つの分数とも既約であるとする）、$ab'-a'b=1$ である。したがってその差の最小値は $1/(N-1)(N-2)$、最大値は $1/(N-1)$ である。
**証明：** 分母、分子ともに $N-1$ 以下で値が 1 以下の既約分数の増加列は Farey 数列と言って、$\Theta(N^2)$ で生成できる。その方法は次の通りである。はじめに、$0/1, 1/1$ からスタートして、次は $0/1, 1/2, 1/1$, それから、$0/1, 1/3, 1/2, 2/3, 1/1$、$0/1, 1/4, 1/3, 1/2, 2/3, 3/4, 1/1$ というように、つねに隣り合う二つの分数 $a/b, c/d$ から $a+c/b+d$ を作り、$a/b, c/d$ の間に挿入して $a/b$, $(a+c)/(b+d)$, $c/d$ というふうに追加していく。このとき、生成される分数はすべて既約であることが知られている（上の文献に書かれている）。またこのプロセスを、どの相続く二つの分数に対しても、そこから新しく作られる分数の分母が $N-1$ を超えるところで終了すれば、得られる分数列が求める列であることも知られている。したがってその差の最小値は $1/(N-1)(N-2)$ であることは $1/(N-1), 1/(N-2)$ がこの列に含まれることからほぼ明かである。最大値についても $0/1, 1/(N-1)$ と $(N-2)/(N-1), 1/1$ がこの列に含まれることからほぼ明らかである。

実際、分母が $N$未満であるような Farey 数を$\Theta(N^2)$ で生成するには、次のように定義される（C言語の）関数を

generate(1, 2, 0, 1, 1, 1);

として呼び出せばよい。

```
generate(y, x, dly, dlx, dry, drx)
int y, x, dly, dlx, dry, drx;
{
     if(y+dly < N && x + dlx < N)
          generate(y+dly, x+dlx, dly, dlx, y, x);
     printf(" %d/%d", y, x);
     if(y+dry < N && x + drx < N)
          generate(y+dry, x+drx, y, x, dry, drx);
}
```

以上の議論より、次の定理を得る。

**[定理 1]** 傾き $u$ を長さ$\Theta(N^2)$ の Farey 数列に従って変化させていくとき、アルゴリズム 3 は確実である。

## 6 アルゴリズムの最適性

アルゴリズム 3 は、すべての極大直線成分を $O(nN^2)$ 時間で列挙することができるが、エッジ点の密度が高い場合、すなわち $n=O(N^2)$ である場合には、これが計算のオーダーとして最善であることを示そう。$n=O(N^2)$ の場合、計算時間は $O(N^4)$ ということになるから、極大直線成分が全部で最悪の場合$\Omega(N^4)$ 通り存在することを示せばよい。

**[観察 3]** $P$を任意の極大直線成分とする。このとき、$u=B/A$, $1\le A<N$, $0\le B<N$ の形の有理数 $u$ と有理数 $v$が存在して、$P\subseteq G(u,v)$ が成り立つ。

[補題6] $N \times N$の画像全体にエッジ点がある場合には以下の事柄が成立する。

1. 絶対値が1以内の傾きの直線に対応する任意の極大直線成分の垂直幅はちょうど1に等しく、絶対値が1以上の傾きの直線に対応する任意の極大直線成分の水平幅はちょうど1に等しい。

2. 任意の極大直線成分に対して上の観察の意味で傾きを表わす一つの有理数$u$を対応づけることができるが、そのような有理数はそれぞれの極大直線成分に対して唯一通りに定まる。

3. 異なる傾きに対応する極大直線成分が等しくなることはない。

[定理2] サイズ$N \times N$の画像の全ての格子点がエッジ点である場合には、$\Theta(N^4)$通りの異なる極大直線成分が存在する。

**証明：** 傾きが0から1までの直線に対応する極大直線成分の個数に注目する。$P$を任意の極大直線成分とする。補題6より、任意の極大直線成分を構成する点集合に対する凸包の垂直幅は丁度1に等しい。しかも、垂直幅1を与える$x$座標が少なくとも2か所ある。このような$x$座標の内、最小のものと2番目に小さいものを$x_1, x_2$とする。また、そのような$x$座標に$P$は2点をもつから、それらの座標を$(x_1, y_1), (x_1, y_1+1), (x_2, y_2), (x_2, y_2+1)$とする。このとき、2点$(x_1, y_1), (x_2, y_2)$を通る直線と2点$(x_1, y_1+1), (x_2, y_2+1)$を通る直線に囲まれた領域に存在する格子点の集合は、明らかに集合$P$に一致する。したがって、$P$は2点$(x_1, y_1), (x_2, y_2)$だけで完全に指定することができる。では、このような2点の取り方は全部で何通りあるだろうか?$(x_1, y_1)$は垂直幅1を与える最も左の点であり、その次に垂直幅1を与える点が$(x_2, y_2)$でなければならないから、$a = x_2 - x_1, b = y_2 - y_1$とすると、$y_1 < b$の場合は$x_1 \leq N - a$であれば両方の点が画像内に存在し、$b \leq y_1 \leq N - b$の場合は$x_1 < a$でなければ$(x_1, y_1)$が垂直幅1を与える最も左の点にはならない。したがって、$(x_1, y_1)$の取り方は、$N \times N$の格子平面において、全部で

$$b \times (N - a) + a \times (N - 2b)$$
$$= (a + b)N - 3ab$$

通り以上、

$$b \times (N - a) + a \times (N - b)$$
$$= (a + b)N - 2ab$$

通り以下である。

$P$に対応する直線の傾きは、$b/a$ すなわち、$(y_2-y_1)/(x_2-x_1)$ で与えられる。この分数は、$x1, x2, y1, y2$ の決め方より、既約分数である。逆に考えると、上の分数が既約になるように上に述べた性質をもつ2点 $(x1, y1), (x2, y2)$ を選ぶ方法が何通りあるかを考えれば、極大直線成分の個数が評価できる。これは、Farey 数列に含まれるすべての分数 $b/a$ について $(a+b)N-2ab$ および $(a+b)N-3ab$ の総和を計算することに等しい。これらの総和を正確に評価するのは困難であるが、$\gcd(a,b)=1,\ b<a$ である確率が $3/\pi^2$であることを考えると、Farey 数列に含まれる分数だけに限らず、既約でない分数も含めて上の和を求めれば、真の和はその高々定数倍以内に収まっている。

$\sum_{a=1}^{N-1}\sum_{b=1}^{N-1}(a+b)N-3ab=\Theta(N^4)$

であるから、定理を得る。

## 7　計算複雑度に関する評価

本節ではこれまでに述べた3種類のアルゴリズムの計算複雑度を時間と空間の両面から評価する。最初に問題を規定するパラメータを整理しておく。

(1)$N$： 画像のサイズ（画素数は $N^2$)

(2)$n$： エッジ点の個数

(3)$M$： 傾きを変化させる回数

最初に、標準的な Hough 変換のアルゴリズム 1 について考えよう。このアルゴリズムでは、各点 $(x_i, y_i)$ に対して曲線 $v=x_i\cos u+y_i\sin u$ に従って「投票」を行なう。投票は傾き $u$ を $u_1, u_2, \ldots, u_M$の順に変化させながら、計算された $v$の値を $v$軸に関する量子化関数 $q(u,v)$ で整数値に変換しながら行なう。量子化関数を

$q(u,v)=\lfloor v/\delta(u)\rfloor,\quad \delta(u)=\max\{|\sin u|, |\cos u|\}$

とすれば、$\delta(u)\geq\sqrt{2}/2$ であるから、関数 $q(u,v)$ の値は $O(N)$ である。結局、カウント用の配列のサイズは $O(NM)$ となる。したがって、初期化に $O(NM)$ の時間が必要である。

投票に要する時間は、1点あたり $O(M)$ であるから、全体で $O(nM)$ 時間となる。これ以外に画像のラスター走査に $O(N^2)$ の時間が必要である。まとめると、次の補題を得る。

**[補題7]** アルゴリズム 1 の時間複雑度は $O(N^2+NM+nM)$ であり、画像以外に必要な記憶スペースは $O(NM)$ である。

次に、アルゴリズム2について調べよう。アルゴリズム1との異なる点は、エッジ点の座標データを配列に入れて蓄えることである。したがって、$O(n)$の配列が必要である。

アルゴリズムとしては、それぞれの傾き$u_i$に対して各点について$v$の値を計算して、$v$の値に関してソートし、ソート列を順に走査するというものである。したがって、ソートに$O(n\log n)$時間がかかるとすると、全体では$O(nM\log n)$時間となる。

しかしながら、傾き$u_i$の変化量は小さいから、$u_i$に対するソート列と次回の$u_{i+1}$に対するソート列はほぼ同じであると予想される。この性質を利用すると、次のような方法も考えられる。

$u_i$に対するソート順を点の番号の列の形式で配列$\sigma[.]$に入れて蓄えるものとする。次に、$u_{i+1}$に対するソート順を求めるには、とりあえず配列$\sigma[.]$の順に点を並べておき、この列に対して挿入法 (insertion sort) を用いてソートを実行し、結果を再び配列$\sigma[.]$に蓄えておく。

上のようにして毎回のソートを実行するとき、$u_i$に対するソート列と$u_{i+1}$に対するソート列がほぼ同じであるという性質より、挿入法は最悪の$O(n^2)$ではなく$O(n)$の時間でソートを実行するであろう。厳密には、$k$個のインバージョン (inversion) を含む長さ$n$の数列を挿入法でソートするのに要する時間は$O(n+k)$である。傾きを順に変化させていくとき、ソート順序が入れ替わる回数は全体で$O(n^2)$である。したがって、全体の計算時間は$O(nM+n^2)$と評価できる。

両者の方法を併用することにより、$O(\min(NM+n^2, nM\log n))$という計算複雑度を実現することができる。

**[補題8]** アルゴリズム2の時間複雑度は$O(\min(nM\log n, nM+n^2))$であり、画像以外に必要な記憶スペースは$O(n)$である。

残るのはアルゴリズム3である。これはアルゴリズム2とほぼ同じであるから、理論的な複雑度のオーダーはほぼ同じであると思われるが、座標値が整数であることを利用すると、後に述べるように、毎回のソートを最適な$O(n)$時間で済ませることができる。また、アルゴリズム2では三角関数の計算が多数必要であったのに対して、アルゴリズム3では一切必要がない。これも実用的にはある程度の改善と考えられよう。

アルゴリズム3ではソーティングについて詳細な記述をしていないが、以下のようにすれば毎回のソーティングを$O(n)$時間で実行することができる。この事実に基づいて、サイズ$N\times N$の整数格子点上の$n$点の集合$P$が入力として与えられたとき、$P$から、し

きい値 $t$ 以上のディジタル直線成分をすべて抽出する $O(nM+kN)$ のアルゴリズムを提案する。ここで、$k$は出力される成分の個数である。各成分の点集合を出力するならば、それだけで、$O(kN)$ かかることに注意しておく。$S$を $N \times N$の整数格子点上の二点から得られる（0 以上 1 以下の傾きの）直線のすべての異なる傾きの増加順のリストとする。$M=|S|$ とする。$S$の $i$ 番目の傾きを $u_i$とする。$u_0=0$ とする。

アルゴリズムは傾き $u_i$で原点を通る直線への垂直距離に関して、各点を非減少順にソートした、長さ $n$ の一次元配列$\sigma[n]$ を管理する。$\sigma[n]$ には垂直距離に関して $j$番目に小さい点の番号が格納されているものとする。アルゴリズムのアイデアは次の通りである。

(1) $u_0(=0)$ のとき、各点の垂直距離は各点の $y$座標で決まるから、$N$通りある。タイが生じたときは、$x$ 座標の大きい順に並べておく。その理由は、傾きを増加させたとき、現在同じ $y$座標の点は $x$ 座標の大きい点が垂直距離が小さくなるからである。$u_0$における配列 v$[n]$ は $O(n\log n)$ 時間で構成できる。

(2) $u_i$のときの配列$\sigma[n]$ が与えられたとき、$u_{i+1}$のときの配列を求めるための更新処理は次のようにしておこなわれる。

ただし、仮定として、現在の配列$\sigma[n]$ には、傾き $u_i$で原点を通る直線への垂直距離に関して非減少順に$\sigma[1]$ から順に$\sigma[n]$ に格納されているものとする。ただし、タイが生じているときは $x$ 座標の大きい順に並べられているとする。点 $p_{\sigma[j]}$から $y=u_{i+1}x$ への垂直距離を $d_{\sigma[j]}$ とする。

**Case 1:** 点 $p_{\sigma[j]}$, $p_{\sigma[j+1]}$から $y=u_ix$ への垂直距離が等しいとき。この場合、仮定より、$p_{\sigma[j]}$の $x$ 座標の方が $p_{\sigma[j+1]}$ のそれより大きいので、$d_{\sigma[j]}<d_{\sigma[j]}$が成り立つ。したがって、変化はない。

**Case 2:** 点 $p_{\sigma[j]}$から $y=u_ix$ への垂直距離が $p_{\sigma[j+1]}$から $y=u_ix$ への垂直距離より小さいとき。$u_i,u_{i+1}$の間に二つの格子点を通る直線の傾きは無いから、$d_{\sigma[j+1]}\geq d_{\sigma[j]}$が成り立つ。$d_{\sigma[j+1]}>d_{\sigma[j]}$ならば、何もしない。$d_{\sigma[j+1]}=d_{\sigma[j]}$ならこれに等しい垂直距離を持つ点をすべて取り出す。これは$\sigma[j+2]$ から順に点を取り出して、$y=u_{i+1}x$ への垂直距離を調べていけばよい。その結果、$\sigma[j]$ から$\sigma[j']$ まで $y=u_{i+1}x$ への垂直距離が同じであったとしよう。このとき、これらの点の $x$ 座標は$\sigma[j]$ から$\sigma[j']$ の順に増加する。というのは、$y=u_ix$ への垂直距離に関しては、$\sigma[j]$ から$\sigma[j']$ の順に単調増加であるからである。$y=u_{i+2}x$ への垂直距離に関しては、逆に$\sigma[j']$ から$\sigma[j]$ の順に単調増加であるから、次に備えて、$\sigma[j]$ から$\sigma[j']$ の順序を逆転させる。

各傾き $u_i$に関して、この更新の計算量は $O(n)$ である。

(3) 傾き $u_i$で原点を通る直線への垂直距離が幅 1 以内に $t$ 個以上の点があるような成分を抽出するには、各傾き $u_i$に関して、求められた垂直距離に関するソート列$\sigma[n]$ を$\sigma[1]$ から順にスキャンすればよい。この手間は $O(n+k_iN)$ である。ここで、$k_i$は各傾き $u_i$に関して出力される成分数である。

(4) 上の議論より、全体の手間は $O(nM+kN)$ である。上で述べた手続きは以下のようになる。

[補題９] アルゴリズム３の時間複雑度は $O(nM+kN))$ であり、画像以外に必要な記憶スペースは $O(n)$ である。ただし、$k$は出力される成分の個数である。

## 結論

本論文では、ディジタル画像において直線成分を検出する問題について考察し、極大な直線成分を残さず抽出することのできる効率の良いアルゴリズムを提案した。従来の直線検出アルゴリズムに比べて効率を格別改善したわけではないが、直線検出能力が保証できる点が従来の方法と決定的に異なる。本論文では直線検出に限定したが、実用的には円弧や楕円の検出も重要である。そのような複雑な幾何図形を検出する問題の計算複雑度についても現在考察を行なっており、近い将来に発表を予定している。

## 謝辞

本研究の一部は文部省の科学研究費の援助の下に行なわれた。貴重なコメントを頂いた Xerox Palo Alto Research Center の Dr. Dan Green および McGill 大学の Prof. David Avis に感謝する。

## 参考文献

(1) P. V. C. Hough: "Method and Means for Recognizing Complex Patterns", U.S. Patent 3069654, December 18, 1962.

(2) R. O. Duda and P.E. Hart: "Use of the Hough Transformation to Detect Lines and Curves in Pictures", Comm. of the ACM, 15, January 1972, pp.11-15.

(3) 松山隆司、奥水大和: "Hough 変換とパターンマッチング", 情報処理、20, 9, 1989, pp.1035-1046.

(4) 藤井、和田、松山： "Hough 変換における歪のないパラメータ空間の構成法"、1991年度電子情報通信学会春季大会、D-562 (1991).

(5) ヴィノグラードフ著（三瓶与右衛門、山中健　訳）："整数論入門"、共立出版、1959 第2章.