

地図認識入力システム MARIS

4N-2

— 前処理 —

志沢雅彦、 鈴木 智、 山田豊通
(NTT 電気通信研究所)

1.はじめに

MARISの前処理について述べる。図1に示す流れで前処理が行われる。この中で、図郭抽出については投影計算を用いた方法をすでに提案した^[1]。本稿では、傾き補正のための画像のブロック分割を用いた高速画像微小回転アルゴリズムと、線図形分類のための線幅測定を線順次に行うアルゴリズムについて述べる。

2.傾き補正

ここでは、画像のブロック分割によって回転の際の座標計算量を大幅に削減できるアルゴリズムを提案する。図郭抽出により図面の傾きがわかっているため、その傾きだけ画像を回転させればよい。回転のアルゴリズムとしては通常はアフィン変換を用いた方法が使われる。すなわち、回転中心を (X_0, Y_0) 、回転角度を θ としたとき、

$$\begin{aligned} X' - X_0 &= (X - X_0) \cos \theta + (Y - Y_0) \sin \theta \\ Y' - Y_0 &= -(X - X_0) \sin \theta + (Y - Y_0) \cos \theta \end{aligned} \quad (1)$$

なる回転の式を用いて、出力画像バッファの各画素 (X, Y) に対して、入力画像バッファのどの画素 (X', Y') が対応するかを計算して転送を行う。この方法では、出力画像バッファ内の各画素に対して、式(1)の計算を行う必要がある。これは膨大な計算量を必要とするため、ソフトウェアで本システムで扱うような大きな画像を処理すると非常に計算時間を要する。また、入力画像と出力画像の各画素が1対1にならず、重複して転送されたり、まったく転送されない部分が生じる。

特に、本システムでは、図面内の線分の線幅情報を自動認識のための知識として用いているので、後者の問題を解決する必要がある。

従来、ハードウェア化を前提とした画像の高速回転手法が提案されている^[2]。しかし、いずれも、汎用計算機での処理には適さず、入力画像と出力画像の画素間の対応も1対1にはならない。

本システムでは回転角度が微小である。このような条件下では図2に示したように、座標移動量が等しい画素がなす領域に画像を分割し、それぞれの領域

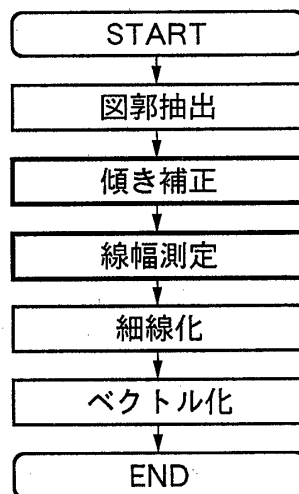
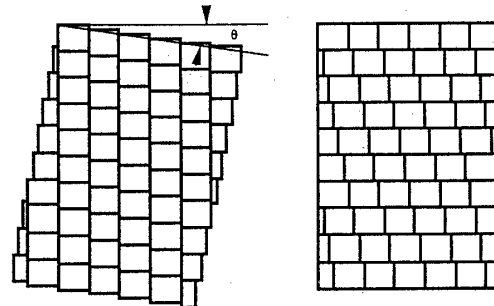


図1 前処理の流れ



(a) 入力画像 (b) 出力画像
図2 座標移動量が等しい領域への分割

ごとに画素の転送を行うことによって、座標計算のための計算量を大幅に削減できる。たとえば、傾きが1度の場合、ひとつの矩形領域の大きさは約 57×57

画素(画素数は約3300画素)となる。従来方法では、各画素ごとに座標計算が必要であるが、本アルゴリズムでは矩形領域の座標(たとえば、左上隅の画素と右下隅の座標)のみでよい。したがって、座標計算について約 $1/1000$ に計算量削減が期待できる。座標計算の計算量は傾き θ が小さいほど小さくなり、 $\tan^2 \theta \approx \theta^2$ に比例する。

本方法では、回転をX軸方向およびY軸方向の斜交軸変換の組み合わせで行っている。変換式を次に示す。

$$\begin{aligned} X' &= X + [(Y - [(X - X_0) \tan \theta'] - Y_0) \tan \theta] \\ Y' &= Y - [(X - X_0) \tan \theta'] \end{aligned} \quad (2)$$

([] は Gauss 記号)

ここで、 X, Y, X', Y' は整数値をとる。この変換は、X軸方向への角度 θ の斜交軸変換とY軸方向への角度 θ' の斜交軸変換の合成変換である。 θ と θ' の間には次の関係がある。

$$\tan \theta' = \tan \theta / (1 + \tan^2 \theta) \quad (3)$$

従来、斜交軸変換の組み合わせによって画像の回転を行う例はある^[3]が、それらはいずれも、X軸方向の斜交軸変換の傾き θ と、Y軸方向の斜交軸変換の傾き θ' の区別をしていない。この区別をしないと、特にサイズの大きな画像を処理した時に画像の斜めのゆがみが現われる。式(3)の関係を用いると、このゆがみを0にすることができる。式(2)の変換によって出力画像の水平・垂直方向の長さが多少伸縮するが、この補正は、図面の入力時の伸縮と合せて自動認識処理のあとに行うことができる。従って、本方式では、入力された画像情報をほとんど失うことがない。

この変換(2)には明らかに逆変換が存在するので入力画像と出力画像の各画素の間には必ず1対1の対応が保たれることがわかる。

具体的には以下の3段階のステップで処理する。

(1) 座標移動量変化範囲の計算

$\Delta X = X' - X$ 、 $\Delta Y = Y' - Y$ を座標移動量と呼ぶ。入力画像バッファ、出力画像バッファ内でこの ΔX 、 ΔY がとる値の範囲を計算する。具体的には、各画像バッファの4隅の画素における ΔX 、 ΔY の値の最大値、最小値を求めればよい。

(2) 各矩形領域の座標範囲の計算

座標移動量の組($\Delta X, \Delta Y$)に対して矩形領域は1対1に対応する。各矩形領域の座標範囲を ΔX 、 ΔY から計算する。具体的には、出力画像バッファにおける各矩形領域のX座標の最小値 $X_{\min}$ 、最大値 $X_{\max}$ 、および、Y座標の最小値 $Y_{\min}$ 、最大値 $Y_{\max}$ を求める。これらは、 ΔX 、 ΔY 、 θ から解析的に求めることができる。このとき、この座標範囲について入力画像バッファ、出力画像バッファの枠におけるクリッピング処理を同時に行う。

(3) 画素の転送

各矩形領域内の画素を入力画像バッファから出力画像バッファに転送する。すなわち、入力画像バッファ内の領域(左上隅($X_{\min} + \Delta X, X_{\max} + \Delta X$), 右下隅($Y_{\min} + \Delta Y, Y_{\max} + \Delta Y$))から、出力画像バッファ内の領域(左上隅($X_{\min}, X_{\max}$), 右下隅($Y_{\min}, Y_{\max}$))へ部分画像の転送を行う。

本システムで扱う画像は1枚当たり約10数MBの容量があるので、主記憶上に全画像を置いて処理を行うことはできない。そこで、短冊型に画像を分割して処理を行っている。出力画像バッファの横幅をH、ライン数をVとすると、入力画像バッファのライン数は少なくとも $(V + H \cdot \tan \theta_{\max})$ ライン必要である($\theta_{\max}$ は最大傾き角度)。画像バッファが大きいほど、座標移動量が同一の矩形領域が画像バッファからはみ出る確率が小さくなるため処理が高速になる。

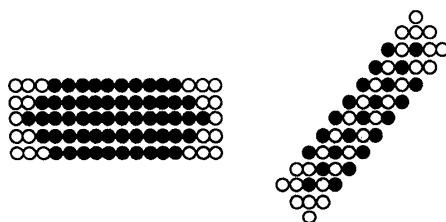
本方法によって、傾き角度が $0 \sim 2^\circ$ の場合、アフィン変換に比べて15倍から2倍の処理速度で傾き補正が可能であることが確かめられた。

3. 線幅測定のための線順次アルゴリズム

MARISでは、図面認識処理において地図内の線幅情報を有効に使っている。したがって、図面上の線を線幅によって分類しておくことが必要である。このとき、線幅そのものが必ずしも必要なのではなく、各画素が図面によってあらかじめ決まっている線幅種類のいずれに属するかが得られれば十分である。ここでは、線幅を線幅種類に対応した区間に分けて、その区間のラベルを画素の値として求める線幅測定を線順次アルゴリズムで実現したものを提案する。

背景の画素

が0、線上の画素が1で表された2値画像から線幅によってラベル付けされた画像をつくる。N種類の線幅 $W(1), \dots, W(N)$ を持つ線を抽出する。



●: 正しい線幅が得られる線上の画素
○: 実際よりも小さい線幅として処理される線上の画素

図3 従来の方法による結果例

ただし、 W

$(n) < W(n+1)$ ($n=0, \dots, N$)を満足する。また、 $W(0) = 0$ 、 $W(N+1) = \infty$ とする。 W

$(n) \leq w < W(n+1)$ を満たす線幅 w にラベル $n+1$ を対応させる。本アルゴリズムでは、線の中心部にある画素とそれ以外の線上の画素を区別して扱うことにより、従来方法^[4]では線の端部や周辺部において実際よりも小さい線幅として処理されてしまう問題(図3参照)を避けている。線順次処理に必要なバッファのライン数 L は最大線幅 $W(N)$ を奇数に換算した大きさ $L = 2 \lfloor W(N)/2 \rfloor + 1$ だけである。これは線の中心部にある画素がバッファの中心ライン($L_0 = \lfloor L/2 \rfloor + 1$)上にあるとき、その線幅はバッファ内で測定可能で、線の中心部以外の線上にある画素はバッファ内にあるという事実にもとづいている。線順次処理は、入力画像を順次1ラインずつバッファのライン L へ転送し、ライン L_0 の線幅測定を行い、出力画像は順次1ラインずつバッファのライン1から取り出し、1ライン処理するごとにバッファの内容を1ライン上へシフトするというサイクルで行う。1サイクルの間に行われる線幅測定処理は次の2段階である。

(1) 線の中心部にある画素の処理

ライン L_0 上の画素で線の中心部にあるものをみつけて、線幅に対応したラベルに付けかえる。具体的には、まず、注目している線上の画素に対して図4に示した8方向の距離 $L1 \sim L8$ を計算する。そして、 k を N から注目画素の値まで変化させ、次の2つの条件を共に満たす最大の k をみつける。

$$\min \{L1+L5, L2+L6, L3+L7, L4+L8\} \geq W(k-1)$$

$$\min \{L1, L2, L3, L4, L5, L6, L7, L8\} \geq \lfloor W(k-1)/2 \rfloor$$

このような k が存在すれば、注目画素は線の中心にあるので注目画素の値を k に付けかえる。

(2) 周辺にある画素の処理

(1)で抽出された画素に連結し、かつ、距離が $\lfloor W(k-1)/2 \rfloor$ 以下のところにある画素の値が、 k よりも小さい場合、画素の値を k に付けかえる。

4. おわりに

本報告では、高速画像微小回転アルゴリズムと線幅測定のための線順次アルゴリズムを提案した。

<<参考文献>>

- [1] 志沢、鈴木:「図面の図郭抽出法」、昭61信学会総合全大、1201、(昭61)。
- [2] 武田ほか:「2次元ブロック転送による高速画像回転方式」、情報学会26回全大、5B-1、(昭58)。
- [3] 田畑ほか:「2次元ブロック転送によるメモリ・アドレス制御方式の提案と文書画像処理への応用」、情報論、Vol.24, No.4, pp.462-473。
- [4] 宮武、松島、江尻:「一様画像演算に基づく平行線抽出手法とその地図への応用」、昭58信学会情報・システム部門全大、56、(昭58)。

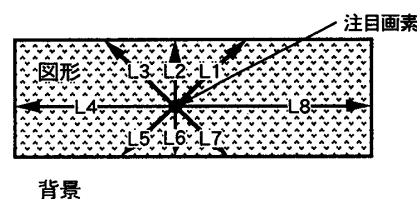


図4 L1~L8の定義