順序付けによる点パターンの高速マッチング・アルゴリズム†

張 鴻賓†† 美濃導彦†† 池田克夫††

平面上の点集合の間のマッチング問題は，パターン認識において，最も重要な問題の一つである．この問題は，数学的に言えば，グラフのISOMORPHISM問題に属し，NP問題である．これを実用化してパターン認識に利用するためには，計算複雑度のオーダが高いので，高速なアルゴリズムが必要である．本論文では，点パターンを順序付けして高速化するアルゴリズムを提案する．まず，マッチングの度合いを評価するために，二つの点パターン間の誤差の尺度を定義する．次に，順序付けした点の概念を導入して，マッチングを高速化する．このマッチング・アルゴリズムは点パターンの幾何学的な位置関係などの情報を利用するもので，実験の結果，効率のよいマッチングが行えることを確認した．また実際の応用におけるいくつかの問題点についても考察する．

1. はじめに

パターン認識では，平面上の点の集合によって記述された点パターンのマッチングとして記述できる問題が数多く存在する．例えば，指紋画像の特徴点（分岐点，端点など）は，ある座標系に対する座標値で記述できる．二つの指紋画像の同定問題は平面上の点パターンのマッチング問題と考えられる．専門家の統計によると，人間の一本の指には，このような特徴点は数十から百個程度存在する．しかし，与えられた二つの指紋が同じであるかどうかを同定するためには，十個ぐらいあれば，十分であるという報告がある[1),2)]．

また，適当な数学的処理によって，パターン情報を点の集合に変換できる問題もある．例えば，物体の輪郭線を多角形で近似し，曲率の大きい点と円弧の端点を特徴点と考えれば，モデル物体と画像から抽出した物体の輪郭線を照合する問題は点パターンのマッチングとなる．

このように，パターン認識におけるマッチング問題において，点パターンのマッチングは重要な役割を果たしており，これまでにいろいろな研究が報告されている[3)~5),7)]．

点パターンは以下のように記述できる．

平面上の二つの点パターンを U, V とする．

$$U=\{u_1, u_2, \cdots, u_i, \cdots, u_m\} \quad 1\leqq i\leqq m$$

$$V=\{v_1, v_2, \cdots, v_j, \cdots, v_n\} \quad 1\leqq j\leqq n$$

ただし，$u_i=(x_{ui}, y_{ui})$, $v_j=(x_{vj}, y_{vj})$. u_i と v_j はそれぞれ点パターン U と点パターン V の点で，(x_{ui}, y_{ui}) と (x_{vj}, y_{vj}) はそれぞれの座標系に対する点の座標値である．m と n は点パターン U と点パターン V の点の数である．このような点パターン U と V との間に位相的な対応関係を見つける問題が点パターンのマッチング問題である．

ここでいう“点”とは，具体的には，画像中の物体の特徴点とか，円や楕円の中心点とか，領域の重心とか，曲率の大きい屈折点などが考えられる．したがって，普通は点が属性を持っているが，汎用性をもたせるため，ここでは対象とする点パターンの位置情報だけを利用する．点の属性情報の利用については，後で考察する．

点パターンのマッチングにおいては，組合せ問題の性質によって計算複雑度のオーダが非常に高くなる．数学的に言えば，これはグラフの ISOMORPHISM 問題であり，補助的な情報がなければ，NP 問題に属する．この問題を解くために，現在までに，いろいろな方法が報告されている．Simon らは点同士の間の相互距離を整合させることによって，点集合のマッチングを試みた[3)]．この方法では，二つの点集合の点の数が等しくなければならない．Zahn は最小生成木によるマッチング方法を提案した[4)]．この方法は点の挿入と脱落に対応する能力が弱い．Rosenfeld らは，弛緩法を点パターンのマッチングに適用した[6)]．弛緩法では，ある程度，点パターンの変形に対応することができるが，収束の速度が遅い．小川はファジ弛緩法を提案している[8)]．また最近は，DELAUNAY 三角形を用いて，点パターンを幾つかの小さい三角形に分割し，最大連係集合を見つけ出して，点パターンのマッチングを行うアルゴリズムを提案している[9)]．この方法は，

† Fast Algorithm for Point Pattern Matching by Using Ordered List by HONGBIN ZHANG, MICHIHIKO MINOH and KATSUO IKEDA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学科

局所領域をアフィン変換することによって，点パターンの局所的な線形変換に対応することができる．しかし，アルゴリズムを制御するキーとなる“顕著”な点の抽出を誤ると，後の DELAUNAY 三角形への分割とマッチングがうまくいかない可能性がある．

原理的にいえば，点パターンのマッチング問題はマッチング状態を表した木（マッチング状態木）を用いて解決できるが，どのような情報を利用して，枝刈りあるいはヒューリスティック型のサーチをするかが問題である[10]．従来から，枝刈りあるいはサーチの手法に関する議論は多くあるが，マッチング時の点の組合せ方に関してはあまり報告がない．そこで，本論文では，順序付けによる点パターンの組合せ方を工夫した高速マッチング・アルゴリズムを提案する．まず，2章ではマッチングの度合いを評価するために，二つの点集合間の誤差尺度を定義する．次に，従来のマッチング状態木型のアルゴリズムにおいて点の組合せ方に関する問題点を分析し，マッチングの速度を向上させるために，順序付けした点の概念を導入した高速マッチング・アルゴリズムを提案する．4章では，実験の結果を示す．点を順序付けしたアルゴリズムは，マッチングの精度を保証すると共に，計算量を大幅に減らすことができる．また，アルゴリズム自体は簡単で，ハードウェア化が可能である．5章では，アルゴリズムの複雑度を分析する．6章は論文のまとめで，実際の応用における点の属性の利用についても述べる．

2. 誤差尺度の定義

画像から抽出される特徴点には，ノイズの影響あるいは画像処理手法に起因する各種要因により，誤って抽出された特徴点が含まれたり，あるべき特徴点が抽出されていなかったりすることがある．抽出された特徴点の位置にも誤差がある．したがって，画像から抽出された特徴点を点パターンと考え，それをあらかじめ与えられたモデルとしての点パターンとマッチングさせる問題においては，二つの点パターンが全く同一であると判定するのではなく，ある誤差尺度によって二つの点パターン間の誤差を求め，誤差値が最小となるものを見つけ出さなければならない．本章では，まず点パターンの基準線の概念を導入し，これによって，点パターン間の誤差尺度を定義する．

定義 1：[基準線 L^u_{ij} と L^v_{kl}]

点パターン U から任意の2点を選び出し，u_i, u_j とする（ただし，$i<j$）．この時，有向線分 $\overrightarrow{u_iu_j}$ を基準線 L^u_{ij} と定義する（**図 1**）．L^u_{ij} の長さを lu とする．点パターン V の基準線 L^v_{kl} も同様に定義する．また長さを lv とする．

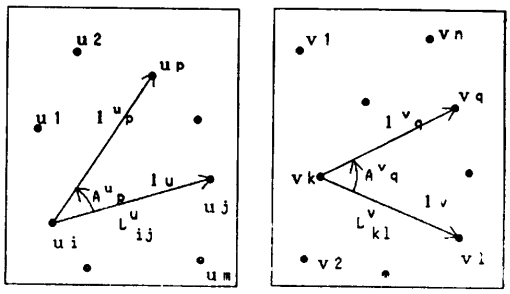


図 1 基準線，長さと角度の定義
Fig. 1 Definitions of base line, length and angle.

定義 2：[基準線対 (L^u_{ij}, L^v_{kl}) とマッチング誤差]

L^u_{ij} と L^v_{kl} とをマッチさせるとき，L^u_{ij} と L^v_{kl} を基準線対 (L^u_{ij}, L^v_{kl}) という．この基準線対 (L^u_{ij}, L^v_{kl}) のマッチングの長さに関する比例因子を $SF=lu/lv$ と定義する．その後，点パターン U の点 u_p と点パターン V の点 v_q をマッチさせる時，このペアのマッチングによる誤差を次のように計算する．

u_p と u_i との距離を l^u_p，線分 $\overrightarrow{u_iu_p}$ と $\overrightarrow{u_iu_j}$ 間の角度を A^u_p とする．同様に，v_q と v_k との間の長さを l^v_q，線分 $\overrightarrow{v_kv_q}$ と $\overrightarrow{v_kv_l}$ 間の角度を A^v_q とする．この時，基準線対 (L^u_{ij}, L^v_{kl}) に対して，v_q の u_p に対するマッチングの誤差を次式で定義する：

$$\omega(u_p, v_q)=KA\cdot|A^u_p-A^v_q| + KL\cdot|l^u_p-SF\cdot l^v_q|$$

ただし，KA と KL はそれぞれ角度と長さの荷重因子である．

比例因子を定義する理由は点パターンのマッチングを大きさに依存しないようにするためであり，相互角度と点同士の間の相互距離を使うのは，マッチングのアルゴリズムを点パターンの回転と移動に対応できるようにするためである．

3. 点パターン・マッチングの高速アルゴリズム

3.1 マッチング状態木探索アルゴリズムの問題点

原理的には，点パターンのマッチング問題は**図 2** に示すマッチング状態木を用いて記述できる．木のルート節点はマッチングの開始点である．葉節点は点パターン間の一種の可能なマッチング結果を表し，中間節点は，部分的なマッチング結果を表す．木の節点は，その親節点に一つの新しい点対を加えることによ

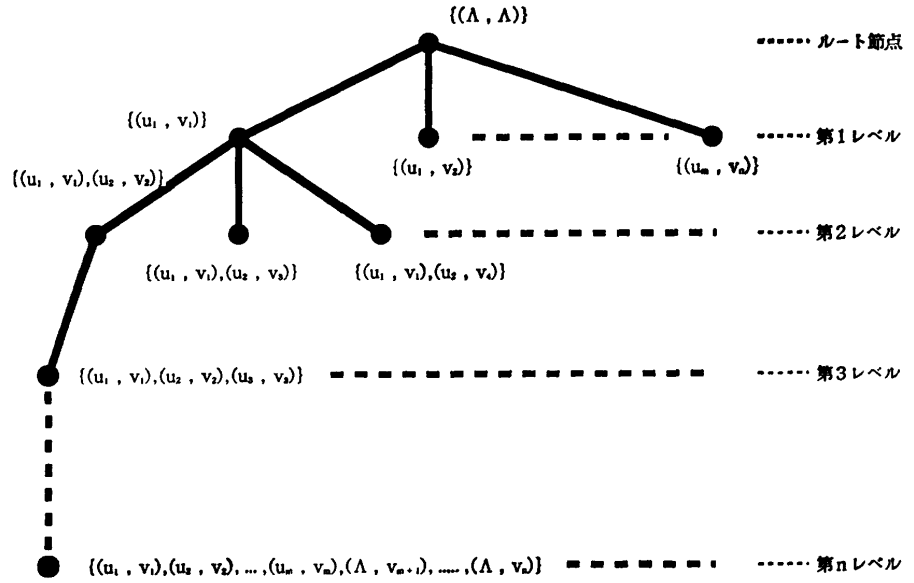


図 2　マッチング状態木
Fig. 2　Matching state tree.

って，生成される．この点対の導入により増加した二つの点パターン間の誤差を，この二つの節点間の枝の重みとする．この時，ルート節点から葉節点に至る重み最小のパスが最良のマッチング結果を表す．

前に述べたように，画像から特徴点を抽出する時には，特徴点の誤抽出や脱落が起こる．よって，U パターンのある点が V パターンの点にマッチングできないことがある（V パターンの場合も同様）．これに対処するために，空の符号 Λ を導入する．すなわち，対応する点が存在しない時には，(u_i, Λ) や (Λ, v_j) などの符号対をマッチング状態木に導入する．この場合，誤差を $KA \cdot At + KL \cdot Le$ とする．ただし，At は最大許容角度閾値で，Le は u_i（あるいは v_j）と基準線の起点間の長さ（v_j に対しては，SF を掛けた後の長さ）である．

すべての点対の組合せに対応するマッチング状態木を作ると，節点の数が膨大になる．したがって，点パターン中の幾何情報を利用して，早期に不適切な点の組合せを除外することは状態木の大きさおよびマッチングの速度を向上させる上で極めて重要である．従来の手法は，枝刈りあるいはヒューリスティックサーチによってアルゴリズムの効率化を図っている．一般的には，マッチング状態木を作成する時に，まず，点の組合せが可能な候補を列挙し，次に長さと角度の構造情報を利用して，不適切な組合せを除去し，探索する節点の数を少なくする．しかし，この種のアルゴリズムでは，点の組合せ方はあまり検討されていない．点パターンに含まれる幾何学的な位置関係などの情報を利用して，点のマッチングを順序付け，探索をある範囲内に限定すれば効率の向上が期待できる．

3.2 アルゴリズムの原理

このアルゴリズムによるマッチングの原理は，U パターンと V パターンに対応する二つの順序付けしたリスト上でマッチングを行うことである．順序付けしたリストは次のように作成する．

(L^u_{ij}, L^v_{kl}) を U パターンと V パターンの一つの基準線対とする．基準線以外の各点に対して，基準線との角度を求め，**図 3** のように，角度の昇順に順序付けしたリストを構成する．ある基準線対に対して，順序付けした二つの点集合を U リストと V リストとする．

$$U \text{ リスト} = (u_1, u_2, \cdots, u_p, u_{p+1}, \cdots, u_{m-2})$$
$$1 \leqq p \leqq m-2$$
$$V \text{ リスト} = (v_1, v_2, \cdots, v_q, v_{q+1}, \cdots, v_{n-2})$$
$$1 \leqq q \leqq n-2$$

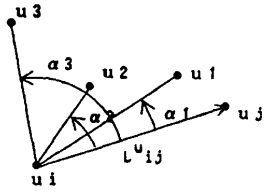


図 3　基準線 L^u_{ij} に対する角度による U 図要素の順序付け
Fig. 3　Sorting of U points by directional angle.

ただし，基準線の起点と終点は順序リストから除いてある．

今，u_p と v_q が対応したとすると，次のマッチング点対候補は，u_{p+1} と v_{q+1} あるいは v_{q+1} 以後のある範囲内にある．u_{p+1} と基準線の角度を Au，許容角度誤差を At とすると，V リスト内の点（角度を Av とする）の中で，$|Au-Av|<At$ を満足する点のみが，u_{p+1} と対応する可能性がある．したがって，探索の範囲がかなり限定できる．最大許容角度誤差 At に対して，U リストの各要素と対応可能な V リストの要素数は，現実的には，ある定数以下とみなせる．このとき，U リストと V リストのマッチングは，V リストに含まれている要素の数と無関係になるので，U リストの要素数の定数倍の時間（線形時間）で完了する．他に必要なコストは両リストの順序付けコストだけである．

3.3 アルゴリズム

以下，順序付けしたリストを利用するマッチング・アルゴリズムを具体的に述べる．アルゴリズムの流れ図を**図4**で示す．

アルゴリズム：

［Ⅰ］ Uパターンと Vパターンの間ですべての“可能な基準線対”を設定する．

［Ⅱ］ BEST_ERROR＝誤差の最大値．
BEST_MAP＝空リスト．

［Ⅲ］ FOR（各基準線対 (L^u_{ij}, L^v_{kl})）DO

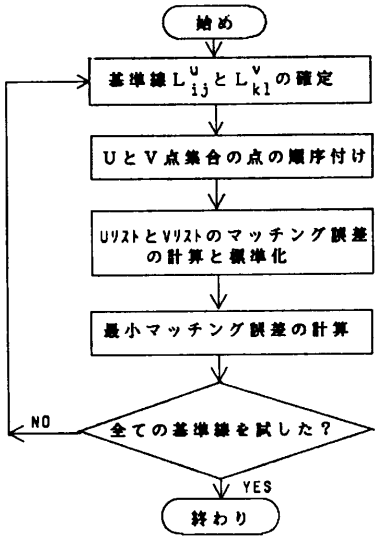


図4 アルゴリズムの流れ図
Fig. 4 Flow of the algorithm.

［Ⅲ-Ⅰ］ Uパターンと Vパターンの基準線以外の各点に対し，基準線起点との長さと基準線との角度を計算する．角度によって，U パターンと V パターンの点を順序付けし，二つの順序付けしたリストUリストと V リストを作る．

［Ⅲ-Ⅱ］ $p=1$

［Ⅲ-Ⅲ］ U リストの第 p 番目の要素を取り出し，u_p とする．u_p の基準線に対する角度を U_ANGLE で表す．

① V リストの U_ANGLE との角度の差が許容角度閾値以内にある，まだマッチしていない，かつ角度が一番最小の要素を捜し出し，v_q と記す．v_q の角度を V_ANGLE で表す．

② WHILE (ABS (V_ANGLE
−U_ANGLE)$<At$)
DO
IF (u_p と v_q がマッチング条件を満足する)
THEN {u_p と v_q をマッチする；
マッチング誤差を計算する；
u_p と v_q にマッチ済フラグを付す；
$p=p+1$；
IF ($p<m-2$) THEN GOTO Ⅲ-Ⅲ；
ELSE GOTO Ⅲ-Ⅳ；
ENDIF}
ELSE {v_q=NEXT (V リスト, v_q)；
v_q の角度 V_ANGLE を取り出す}
ENDIF
ENDWHILE

③ もし最大許容角度誤差範囲内において，マッチング条件を満足する点がなければ，u_p を空とマッチさせる；
誤差を計算する；
$p=p+1$；
IF ($p<m-2$) THEN GOTO [Ⅲ-Ⅲ]；
ELSE GOTO [Ⅲ-Ⅳ]；
ENDIF

［Ⅲ-Ⅳ］ V リストのマッチしていない点の誤差を計算する．

［Ⅲ-Ⅴ］ マッチングの長さに関する誤差 L_ERROR と角度に関する誤差 A_ERROR を荷重平均して，正規化誤差 N_ERROR を計算する．この基準線対に対するマッチングの結果を MAP とする．

IF (N_ERROR<BEST_ERROR)

THEN {BEST_ERROR=N_ERROR;
BEST_MAP=MAP}
ENDIF

[Ⅳ] すべての基準線対の処理が終れば，終了．さもなければ，GOTO [Ⅲ].

上述のアルゴリズムの中で，BEST_ERROR はマッチングの最小誤差を記録するための変数，BEST_MAP は最小誤差のマッチング結果を記録するための配列である．関数 NEXT (V リスト, v_q) は V リストの v_q の次の要素を取り出す関数である．

u_p と v_q がマッチング条件を満足するためには，次の二つの条件が必要である．

条件 1：u_p の基準線 L^u_{ij} に対する角度を A_u, v_q の基準線 L^v_{kl} に対する角度を A_v とすれば，$A_\epsilon=|A_u-A_v|$ はある最大許容角度閾値 At より小さい．

条件 2：u_p と基準線 L^u_{ij} 起点間の長さを L_u, v_q と基準線 L^v_{kl} 起点間の長さに比例因子 SF を掛けて，その値を L_v とすれば，相対長さ誤差 $L_\epsilon=|L_u-L_v|/L_u$ はある最大許容長さ閾値 Lt より小さい．

U リストと V リスト中でマッチングしていない要素に対しては，誤差を計算し，総誤差に加える．これは，対応点が存在する方の誤差が対応点の存在しない方の誤差より大きくならないようにするためである．

正規化誤差 N_ERROR は次式のように定義する．

$$\text{N\_ERROR}=KL\cdot\text{L\_ERROR}/LN+KA\cdot\text{A\_ERROR}/AN$$

ただし，KL と KA は長さと角度の荷重因子とし，LN と AN はそれぞれ長さと角度の正規化因子とする．

長さの正規化因子 LN は次式で定義する．

$$LN=\sum_m(U\text{ リスト要素の長さ})+\sum_n(V\text{ リスト要素の長さ}\cdot SF)$$

角度の正規化因子 AN は $(m+n)\cdot At$ とする．ただし，m, n は U リストと V リストの要素の数である．

他に利用できる情報がなければ，“可能な基準線対”とは U パターンと V パターンそれぞれの二つの点の間の線分のすべての組合せである．実際の応用問題では，各種の拘束条件が利用できるので，可能な基準線対の数と計算量を大幅に減らすことができる．

4. 実験結果

NEC 製スーパミニコンピュータ MS-190 上でいろいろなデータを用いて，本アルゴリズムを実行した．以下，各々の実験および実験結果について述べる．

実験 1：

二次元乱数発生プログラムによって，ランダムに n 個の点を発生し，これらの点を V パターンのデータとする．n 点の中から任意に m 個の点を選び出し，一定角度回転させた後に，各点の動径を，方位は変化させずに伸縮させて，移動する．ただし，動径の伸縮率に一定のランダムノイズを加える．このようにして求めた m 個の点を U パターンとする．

すなわち，乱数発生プログラムによって作り出した V パターンの点 Vp の座標値を (X, Y) とする．U パターンにおける対応点 Up の座標値 (X', Y') は，次式によって求める．

$$X'=(X\cdot\cos\beta-Y\cdot\sin\beta)\cdot(f+c)$$
$$Y'=(X\cdot\sin\beta+Y\cdot\cos\beta)\cdot(f+c)$$

ただし，β は回転角で，f は伸縮係数，c は標準偏差値が σ，平均が 0 の正規乱数である．

実験では，二次元乱数発生プログラムにより，10組ずつの異なる点集合を 7 種類発生した．実験中で採用した長さのしきい値 Lt は相対長さ誤差 10% とし，角度誤差しきい値 At は 10^0 とした．ノイズに当たる正規乱数の標準偏差は 0.025 とした．これらのデータに対して，すべて正しい結果を得た．処理に要した CPU 時間の平均値を**表 1** に示す．

比較のために，総当り法によりマッチングを行い，長さと角度を利用して枝刈りを行う方法により，同一のデータを用いて，実験した結果を**表 2** に示す．

当然のことながら，表 1 と表 2 を比較すると，順序

表 1 アルゴリズムの平均 CPU 時間（ミリ秒）(実験 1)
Table 1 Average cpu time of the algorithm (ms).

U図の点数 / V図の点数	5 / 15	6 / 16	7 / 15	6 / 18	7 / 17	6 / 21	8 / 16
平均時間 (ms)	345	673	850	999	1,266	1,781	1,473

表 2 マッチング木のアルゴリズムの平均 CPU 時間
Table 2 Average cpu time of the matching-tree algorithm (ms).

U図の点数 / V図の点数	5 / 15	6 / 16	7 / 15	6 / 18	7 / 17	6 / 21	8 / 16
平均時間 (ms)	20,451	83,674	190,526	158,253	423,255	467,533	667,270

付けのアルゴリズムの方がかなり速い．これは，点集合を順序付けして，対応点を探索する範囲を限定し，ランダムな組合せを避けることによる効果である．

実験 2：

対応させる点の数を実験 1 に比べて倍以上に増加させて，点数と平均 CPU 時間の関係を調べた．データは実験 1 の時と同じように乱数により作成した．実験結果を**表 3** にまとめる．この表からわかるように，この程度の点数に対しては，このアルゴリズムは実用的に利用できる．**図 5** は，実験 1 と 2 の結果から求めた U パターンと V パターンの点数の積に対する平均の CPU 時間の曲線である．この曲線は後で述べるアルゴリズムの複雑度からの考察（5 章の場合 1）とほぼ一致する．

実験 3：

二次元形状のマッチングに適用する実験を行った．

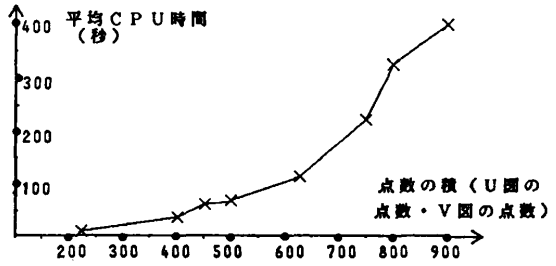
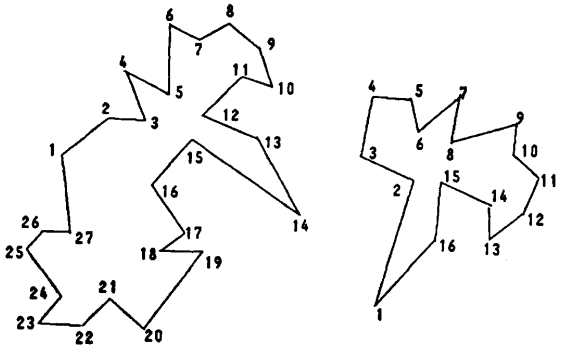


図 5 CPU 時間と点数の積の関係
Fig. 5 Curve of cpu time and point number.



(a) V-pattern (b) U-pattern

図 6 二次元形状のマッチング（実験 3）
Fig. 6 Matching of two dimensional shapes (experiment 3).

図 6 の (b) はモデルパターン，(a) は (b) を含むパターンで，点数はそれぞれ 16 点と 27 点である．(b) と (a) の対応付けの結果は**図 7** のようになった．対応付けられた点に丸のマークを，マッチングのできなかった点に三角形のマークを付けている（以下の実験の結果も同様）．マッチングの CPU 時間は 31,128 ミリ秒であった．

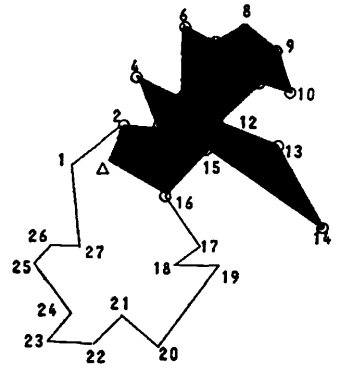


図 7 実験 3 のマッチング結果
Fig. 7 Matching result of experiment 3.

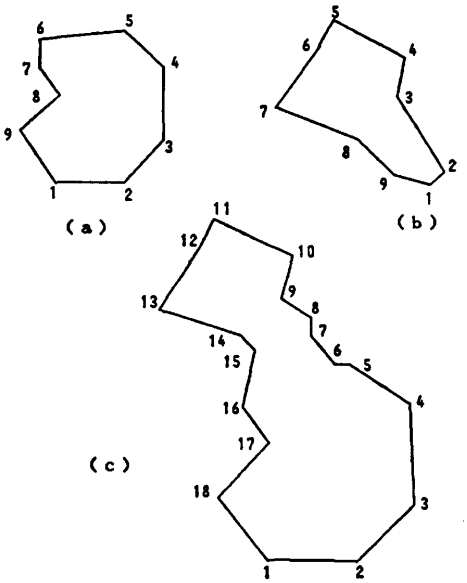


図 8 二次元形状のマッチング（実験 4）
Fig. 8 Matching of two dimensional shapes (experiment 4).

表 3 アルゴリズムの平均 CPU 時間（ミリ秒）（実験 2）
Table 3 Average cpu time of the algorithm (ms).

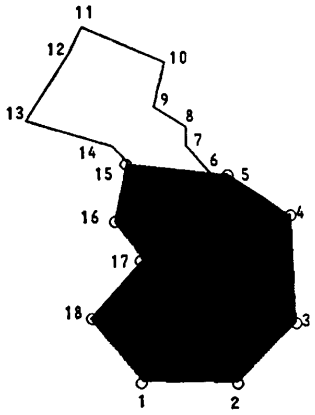
U図の点数	15	20	15	20	25	25	20	30
V図の点数	15	20	30	25	25	30	40	30
平均時間 (ms)	7,450	39,430	59,921	72,916	118,298	224,254	325,961	399,136

実験 4:

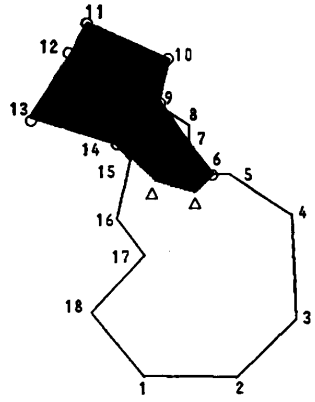
図 8 の(c)は(a)と(b)の合成パターンである．ただし，(c)では，(a)のパターンが拡大されている．実験の結果，**図 9** のようにマッチングできた．この実験のデータは文献 11) から引用した．実験環境が異なるので正確には比較できないが，われわれの方が効率がよいと考えられる．**表 4** は本アルゴリズムと文献 11) の所用 CPU 時間および実験の環境の比較表である．

実験 5:

図 10 は文献 11) に掲載されている問題である．



(a)　a と c のマッチング結果



(b)　b と c のマッチング結果

図 9　実験 4 の結果
Fig. 9　Result of experiment 4.

表 4　本アルゴリズムと文献 11) の比較 (単位：秒)
Table 4　Comparison with reference 11).

	マッチング CPU 時間		利用コンピュータ
	a と c	b と c	
本手法	3.67	2.40	MS-190, 4.0 MPS
文献 11)	52.79	42.90	PDP-10, 1.8 MPS

(a)と(b)と(c)の物体が重なって，(d)になっている．アルゴリズムは，パターン(d)の中から部品のパターン(a)，(b)，(c)を見つけ出した．**図 11** は(a)，(b)，(c)のそれぞれと(d)との間の対応付けの結果である．(a)と(d)のマッチング時間は 0.93 秒，(b)と(d)のは 0.84 秒，(c)と(d)のは 2.40 秒で，合計時間は 4.17 秒である．文献 11) のアルゴリズムでは，152.7 秒がかかっている．

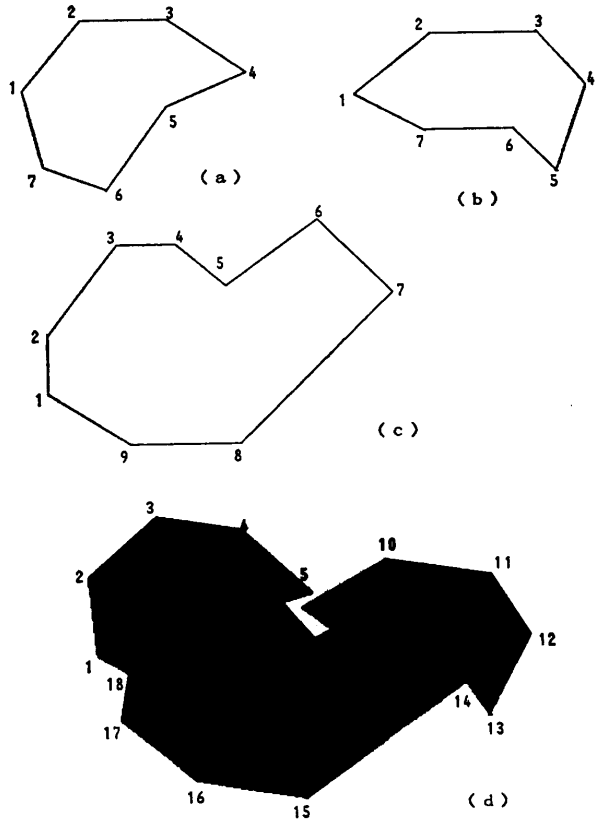


図 10　文献 11) の問題 (実験 5)
Fig. 10　The problem in the reference 11) (experiment 5).

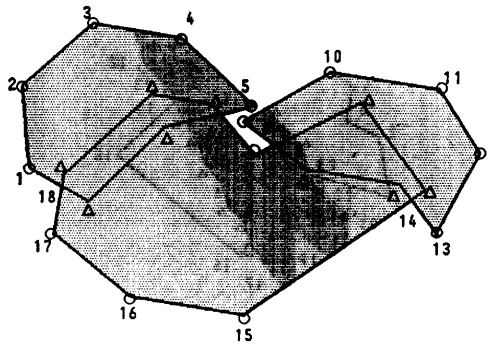


図 11　実験 5 の結果
Fig. 11　Result of experiment 5.

実験 6:

重なっている部品の識別はこれまでにいろいろな報告がなされている．われわれは本アルゴリズムをいろいろな例に適用した．**図 12** はその一例で，(a)，(b)はそれぞれ部品の輪郭線の折れ線近似，(c)は(a)と(b)が重なっている輪郭線の折れ線近似である．実験の結果，(a)と(c)は**図 13** のようにマッチングされた．(a)の頂点数は 21 点，(c)の頂点は 23 点，CPU 時間は 47.70 秒であった．

二次元形状のマッチングや重なった物体の識別問題では，例えば，輪郭線の点の順序や，隣接点との関係などの情報が利用できる．今後は，これらの情報を利用し，さらにマッチング速度の向上を計る予定である．

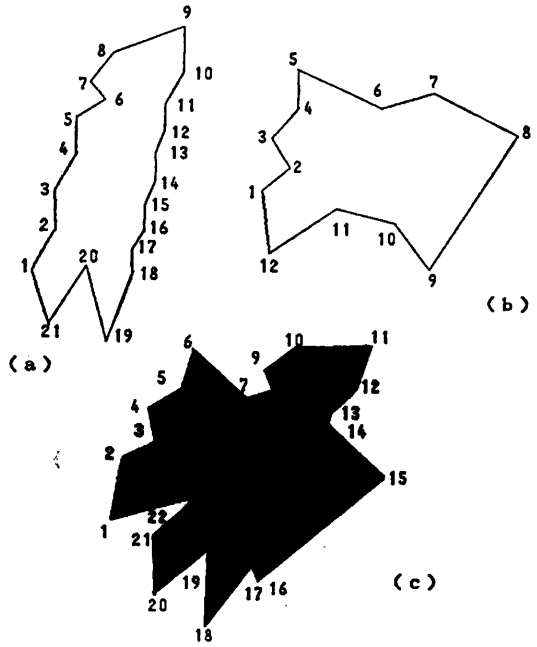


図 12 重なっている物体の認識（実験 6）
Fig. 12 Recognition of overlapping objects (experiment 6).

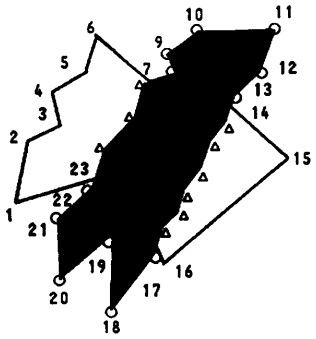


図 13 実験 6 の結果
Fig. 13 Result of experiment 6.

5. アルゴリズムの複雑度の分析

本章では，アルゴリズムの複雑度を簡単に分析する．ここでは，アルゴリズムの複雑度という術語を時間計算量という意味で用いる．アルゴリズムの複雑度は主として，基準線対の数と一つの基準線対の中で実行される操作の計算量で決まる．本アルゴリズムでは，一つの基準線対に対して，ソーティングとマッチングを行っている．マッチングは U リストの要素数の線形時間内で終了する．ソーティングは U，V リストそれぞれ $m\cdot\log m$ と $n\cdot\log n$ の時間でできる．ただし，m と n は U パターンと V パターンの点の数である．したがって，一つの基準線対に対する処理時間は $p\cdot\log p$（ただし，$p=\mathrm{Max}(m,n)$）のオーダとなる．基準線対の数は，利用できる情報がない場合では，主に点の信頼性によって大きく変化する．“信頼性の高い点”とは U パターンと V パターン中で，互いに確実に対応できる点をいう．以下，四つの場合に分けて分析する．

(1) U 点集合の m 個の点と V 点集合の n 個の点の信頼性が不明の時，すべての基準線対をマッチングのために計算する必要がある．この場合は，基準線対の数は $O(m^2n^2)$ ので，アルゴリズムの複雑度は $O(m^2n^2\cdot p\cdot\log p)$ となる．ただし，$p=\mathrm{Max}(m,n)$，m,n はそれぞれ U パターン，V パターンの点の数である．これはアルゴリズムの複雑度が最も高くなる場合で，複雑度の上限である．実際の問題では，ほかの情報が利用できるので，一般的にはこれよりも低い複雑度で計算できる．

(2) U 点集合の一つの点 u' と V 点集合の一つの点 v' が対応するとあらかじめわかっている場合：アルゴリズムの複雑度の上限は $m\cdot n\cdot p\cdot\log p$ となる．

(3) U 点集合の一つの基準線 L^u_{ij} と V 点集合の一つの基準線 L^v_{kl} が対応するとあらかじめわかっている場合：アルゴリズムの複雑度は $p\cdot\log p$ となる．

(4) U 点集合の点と V 点集合の点がすべて確実な特徴点で，挿入点や脱落点がなく，誤差も許容誤差の範囲内の場合：U 点集合の一つの基準線に対して V 点集合の中に必ず対応基準線が存在するので，V 点集合の対応基準線は，最大 n^2 のオーダで探索できる．よって，この時の複雑度の上限は $O(n^2\cdot p\log p)$ である．

このアルゴリズムは，可能な点パターンの組合せのなかで，総誤差 N_ERROR が最小のものを探している．この条件をゆるめて，総誤差 N_ERROR がある誤差しきい値より小さい時にアルゴリズムを終了させると，上述の理論分析の上限よりもっと低い複雑度で計算できる．

以上の分析から，基準線を決めるための利用できる情報がない場合では，本アルゴリズムの複雑度は主に点の信頼性に依存している．

6. おわりに

本論文では，点パターンのマッチング問題に対して，まず二つの点集合の間の誤差の尺度を定義し，次に，この誤差の尺度をマッチングの評価基準として，点の組合せの方法を検討し，高速マッチング・アルゴリズムを提案した．本アルゴリズムでは，点パターンを順序付けしたリストで表現し，これによって，マッチング過程に一定の順序を与えた．この結果，マッチング速度が大幅に改善できることを確かめた．また，点の間の相互距離と相互角度および比例因子を利用しているので，点の脱落と挿入，座標系の回転，移動と大きさの変化などに対応することができる．

本論文では，一般的な点パターンを対象にし，幾何学的な構造情報などを利用した高速なマッチング手法を提案した．具体的な問題に応用する場合は，さらにほかの有益な情報，たとえば，点の属性などを使って，さらに高速化しなければならない．あるいは，あらかじめある点対またはある線対は必ず対応するという基準点対または基準線対を何らかの手段で発見し，アルゴリズムの複雑度のオーダを下げてゆくことが重要である．

今後は，犯罪現場に残った遺留指紋から犯人を割り出す機能を持つ検索システム[12]や，二次元形状のマッチングを利用した実用システムなどの場合に，本文で提案したアルゴリズムを応用してゆくことが課題である．

謝辞 本研究を進める上でご指導頂いた坂井利之教授とご協力頂いた元坂井研の諸氏に感謝いたします．

参考文献

1) Moayer, B. and Fu, K. S.: A Syntactic Approach to Fingerprint Pattern Recognition, *Pattern Recogn.*, Vol. 7, pp. 1-23 (1975).
2) The Science of Fingerprints, FBI Manual, U. S. Gov. Printing Office (1963).
3) Simon, J. C., Checroun, A. and Roche, C.: A Method of Comparing Two Patterns Independent of Possible Transformations and Small Distortions, *Pattern Recogn.*, Vol. 4, No. 1, pp. 73-81 (1972).
4) Zahn, C. T.: Graph-Theoretical Methods for Detecting and Describing Gestalt Cluster, *IEEE Trans. Comput.*, Vol. C-20, No. 1, pp. 68-86 (1971).
5) Kahl, D. J., Rosenfeld, A. and Danker, A.: Some Experiments in Point Matching, *IEEE Trans. Syst. Man Cybernet.*, Vol. SMC-10, No. 2, pp. 105-116 (1980).
6) Ranade, S. and Rosenfeld, A.: Point Pattern Matching by Relaxation, *Pattern Recogn.*, Vol. 12, No. 4, pp. 269-275 (1980).
7) Lavine, D., Lambird, B. A. and Kanal, L. N.: Recognition of Spatial Point Patterns, *Pattern Recogn.*, Vol. 16, No. 3, pp. 289-295 (1983).
8) Ogawa, H.: Labeled Point Pattern Matching by Fuzzy Relaxation, *Pattern Recogn.*, Vol. 17, No. 5, pp. 569-573 (1984).
9) Ogawa, H.: Labeled Point Pattern Matching by Delaunay Triangulation and Maximal Clique, *Pattern Recogn.*, Vol. 19, No. 1, pp. 35-40 (1986).
10) Eric, W., Grimson, L. et al.: Localizing Overlapping Parts by Searching the Interpretation Tree, *IEEE Trans. PAMI*, Vol. PAMI-9, No. 4, pp. 469-482 (1987).
11) Bhanu, B. and Faugeras, O. D.: Shape Matching of Two Dimensional Objects, *IEEE Trans. PAMI*, Vol. PAMI-6, No. 2, pp. 137-156 (1984).
12) 瀬戸，星野：指紋照合の自動化技術，画像電子学会誌，Vol. 15, No. 3, pp. 184-191 (1986).

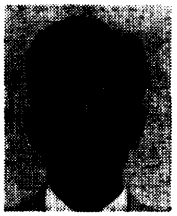
(平成元年1月17日受付)
(平成2年4月17日採録)

張 鴻賓（正会員）

1968年中国・清華大学制御工学科卒業．1970年北京自動車製作工場電気技師．1981年清華大学大学院修士課程パターン認識と知能制御専攻修了．1982年北京計算機学院大学講師．1986年～1989年京都大学工学部客員研究員．現在北京計算機学院大学副教授．パターン認識，画像処理，グラフィックス，および，それらの応用に関する研究に従事．

美濃 導彦（正会員）

昭和53年京都大学工学部情報工学科卒業．昭和58年同大学院博士課程修了．同年，工学部助手．平成元年助教授．画像処理，人工知能，知的コミュニケーション関係の研究に従事．工学博士．電子情報通信学会，画像電子学会，ロボット学会各会員．



池田 克夫（正会員）

昭和12年生．昭和35年京都大学工学部電子工学科卒業．昭和37年同大学大学院修士課程修了．昭和40年同博士課程学修退学．同年京都大学助手．昭和46年同助教授，工学部．昭和46年9月より1年間文部省在外研究員として，米国ユタ大学および MIT に留学．昭和53年筑波大学教授，電子・情報工学系．昭和63年8月京都大学教授，工学部情報工学教室．コンピュータ組織法，LAN，画像処理／理解に興味を持つ．著書に「オペレーティングシステム論」（電子情報通信学会）などがある．工学博士．電子情報通信学会，人工知能学会，IEEE，ACM 各会員．