題名：「英子文字認識の試行」

## 第１章　はじめに

　２年前に、シンボル認識のプログラミングに挑戦して、多くのことを学んだ。それは論文「パターン認識の一考察」にまとめたことである。自然言語の研究も一段落したので、今回はシンボル認識を一歩進めて、文字認識を目指すこととした。本当は連続文字認識に挑戦したかったのだが、まだ地歩ができていないということで、孤立文字、それも英子文字に絞って、ノウハウの蓄積を狙ってみた。

　基本は当初考えた通りにいったのであるが、外乱に強い強靱なものとするには考えが足りなかったことを痛切に感じることとなった。連続文字に入る前に孤立文字認識を目指して正解だった。今回の反省点を改善して、今度は連続文字の認識に挑んでいくつもりである。

　本論は、今回のシステムの構成を説明し、その限界を指摘する。改善したプログラム構成を提案し、将来の展望を言及して、本論を終えたい。

## 第２章　試行システムの構成

　本プログラムは３階層、実際にはその上に文字認識をする層があるから、全部で４層の構造をしている。

・（第１層）特異点抽出

　イメージデータをスキャンして端点、分岐点、穴、孤立点を抽出する。スキャンはテレビ走査法である。

・（第２層）線の抽出と、線と穴の属性の解析

　ライントレースを行い、線を抽出する。また、線の曲がり具合とか、穴の扁平率、方向などを計測していく。

・（第３層）文字を構成する要素の配置の解析

　穴や線の配置を解析し、テーブルに纏める。

・（第４層）文字の特徴にのっとり、１層から、３層までで得た情報をもとに、イメージデータの分類をする。

　本プログラムでは細線化処理はしていない。細線化で情報が落ちるのが気になってのことで、何処まで細線化しないでやれるか試行したかったからである。ただし、線幅の最大は４ピクセルと仮定した。今回の方法はこれをもっと太くしても適用できるものである。結論として、ライントレースと特異点抽出に苦労することになったので、細線化と今回の方法を併用で行うのがベストと判断している。

　以下に幾つかのアルゴリズムを紹介したい。

２．１　端点検出

検出用のボックスを用意し、イメージをスキャンしていく。輪郭線がボックス内で接触したとき、接触点と反対側に線画が伸びていると端点と判断する。

←ボックス

２．２　分岐点検出

イメージをスキャンし、ボックス内から線画が複数伸びていたら、精密な解析をして、分岐の中心を求めて、また分岐数も求めて、分岐点と判断する。

２．３　穴の検出

塗りつぶしアルゴリズムで、地とは異なる閉じた白地を抽出して、その大きさと、配置、輪郭トレースを求める。

２．４　ライントレース

分岐点検出アルゴリズムを流用した。つまり、新しい点はボックス幅だけ移動して、線画を切る回数が２回となる点を求める。トレースが逆戻りしないように、検出した線分にはマスクをした。また、トレース点をサンプリングしていくので、同じ値が現れない事を確認しつつ、新しい点を決定していく。

２．５　曲線の属性

始点と終点の作るベクトルと成す距離が最大となる点の距離と、それがベクトルの右か左かを解析し、曲線の属性とした。また、極座標で曲線の点の配置を分析、距離と角度の変異点の個数を求めて、複数個ならば複雑曲線として、滑らかな曲線とは異なるものとした。それは、文字"C"と"k"を考慮してのことであった。

滑らかな曲線

複雑な曲線

また、上下に膨らみがあるかどうかの解析も行っている。"n"とか"m"を考慮してである。

２．６　配置

線や穴の配置も解析した。基本的に２つのシンボルの配置関係をテーブルにしたのみである。各線や穴毎に、その近傍の線や穴の配置を求めたのである。

基本的に、横方向では、{"left"、"neutral"、"right"}の３つで、縦方向は{"over"、"up","middle"、"down"、"under}の５つの配置を求めた。配置は接点の位置である。

２．７　文字認識

線や穴の配置を求め、文字と一致するか調べる。さらに、曲線や穴の形状属性を観て、最終弁別をする。

## 第３章　試行システムの成果

予定していた文字の認識は成った。簡単な作りであるが、英子文字ならばなんとか認識してくれる。大文字、や数字はまた別のシステムを作り、それらを平行して走らせ、結果を弁別していくのが簡単でよい。このシステムを拡張するのは、処理を複雑にしてしまうばかりである。それに、想定の文字認識は上手く行くが、違う人の文字だと誤認識してしまう。まだまだ、ぬるい解析なのである。そこへ、大文字認識を取り入れるのは、問題を発散させてしまうおそれがある。

以下に問題点を纏めて、成果とする。

３．１　問題点と原因

（１）端点抽出に失敗することがある。

端点でない凸凹を端点と誤認することがある。アルゴリズムがまだ粗いのだ。基本的に雑音除去の問題であり、文脈処理が必要なことがらということだ。今回は文脈処理は考えておらず、解析パラメータは固定だったので、上位層で、無意味な端点情報を除去する手法を用いて、この問題に対応した。

（２）　ライントレースがなかなか動かなかった。

やはり線に幅があるとビームがはずれてしまいやすい。細線化して、線幅を１ピクセルとして、その線と太さのある生な線を併用して、ライントレースするのがベストなやりかたであると判断する。であるが、今回は大方のところ、要請をクリアしたアルゴリズムを得ることができた。

（３）　曲線などの情報が不足した。

基本的に次のような情報が必要である。

・大きさ（絶対的な大きさ、相対的な大きさ）
・長さ（絶対的な長さ、相対的な長さ）
・位置（絶対的な位置、相対的な位置（配置））
・シンボルの配置順序

・円弧性（幅、長さ、向き、上下左右の湾曲特性、交差性）
・曲率
・平坦さ
・向き
・始点終点ベクトル
・ベクトルの原点
・特異点の配置
・分岐の数
・近傍情報（線（始点終点近接点、穴の形状とそれらの位置関係）
・イメージそのまま

今回は一部重要なもののみを実現しただけで、しかもパラメータは固定してしまった。できるだけ多くの情報を得て、その情報も固定パラメータでなくて、可変になっていて、文脈によって、判断を変えられるようにしていくべきである。

（４）雑音に弱い
髭があったり、接触すべきところが離れていたりすると、認識に失敗する。思わぬ文字に認識してしまうこともある。綿密な情報とともに、文脈処理の必要性を訴える根拠となる結果であった。

（５）遅い
ものすごい計算しているから、しかたないところである。分かりやすいコードを目指したというのも遅くなった理由である。

## 第４章　新しい構想

解析していく情報を充実させるのであるが、解析パラメータも情報として、集中して管理し、解析パラメータと解析結果とを対で評価していけるようにしたい。各層にこのような情報を持つデータエリアを持って、解析していくのである。

今回の結果を受けて、今度は連続文字認識に挑戦していくことになる。そこでは、まだ未解決の次の処理が待っている。
（１）特異点の連なりの解析
（２）錯視線生成
（３）見なし処理
・分かれている線を一本の線とみなす
・円を線と見なす
・曲線を円と見なす

・傾き線を垂直線あるいは水平線と見なす
・接続すべし、分離すべし
（４）重なり解決
（５）雑音除去

連続文字認識では、空白の解析が重要になる。文字幅とか文字高を求めるためである。また、ボトムラインとか トップライン、アンダーラインなんかの同定も重要になる。

## 第５章　発展

システムを作っていて気づいたことをあげたい。

### ５．１　曖昧性処理について

曖昧性処理には精密な機構が必要である。解析は精密にし、それを弁別するのは文脈機構によって、さらに精密に弁別していくようにしなくてはならない。解析の初期に情報を曖昧にしてしまうと、あとでバックアップのしようがないからである。

多段階で分類していくときでも、どこかで欠落した情報を補えるようにシステムを構成しておくこと、これが重要である。人間の場合、一つの見方だけでなく、いくつもの視点でものを見て、情報を精密に捕獲するようにしているようだ。一点からは曖昧でも、とにかく、システムとしては精密に情報を得ているのである。

ファジー化は曖昧性を持つデータを記号化するときに行うべきものである。初めから不用意にファジー化すべきではない。情報が落ちるからである。

### ５．２　パターン学習について

大局情報と文脈情報、解析結果をまとめて１つのデータとしてコーパスとして保持していく。

コーパスから変形規則を抽出していく。この過程は推論になる。つまり、先ず、コーパスは教師によって分類される。その分類が上手く行くような変形規則を既存のプリミティブな変形規則の組み合わせで作り出していくのである。これが自己学習部分になる。

未知の値を得たときには、特徴的な変形操作をもって、移行できる記憶内のシンボルに分類するようにする。

学習とは、この場合、変形規則と変形を行うトリガーを見いだす事であるといえる。

## 第６章　おわりに

２００６年５月から２ヶ月、文字認識に挑戦した。得られた結果は大きい。文字認識に必要な情報というものが洗い出せたと考える。また、パターン認識プログラムの構造はどうあるべきかも明確になった。簡単な構造でも、英子文字の認識は可能であることもわかり、連続文字の認識に向け弾みができた。

多くの有用なパターン認識システムができることを願う次第である。

以上