文字認識ソフトウェアの作成

４年Ｂ組　上田　　樹
指導教員　米田　隆恒

１．要約

私は、看板などで読めない漢字を見つけたとき、読みが分からなければ意味を調べるのも困難であると感じた。そこで、日本語に限らず、それらの文字を全て写真に撮るだけで認識することができれば、外国など読めない文字が使われている場所へも、もっと気楽に行くことができるようになって、便利であると考えた。

市販の文字認識ソフトもあるが、文字の状態によっては読めないことがある。そこで、Visual Basic を用いて、様々な状態の文字の写真に対応できる文字認識ソフトを自分で開発することにした。

キーワード　　文字認識、エッジ、HLS、細線化、輪郭追跡、学習機能

２．研究の背景と目的

看板などで読めない漢字や知らない言語の文字などを見つけたとき、そのたびに意味を調べるのは大変である。そこで、そのような場合でも全て、写真に撮るだけで文字を認識することができればよいと考えた。携帯電話などに付属している文字認識ソフトは、文字の並びや大きさによっては読めない、文字のバランスが悪いと読めない、斜めだと読めない、言語によっては読めないなど、読みとれる条件が限られている。また、私は昨年度までもこの研究を行っていたが、背景に模様等があると正しく文字を認識することができなかった。そこで、これらの問題を解消し、より多くの場合に対応できる文字認識ソフトの開発を行うことにした。

３．研究内容

文字の特徴点を抽出し、それらをつないでいくことができれば、文字を線の集まりとして読み取ることができると考えた。そのために、前処理として文字の領域の抽出、また文字の頂点の認識のために線の中心部の抽出を行った。

3.1 文字の抽出

昨年度の研究では、写真から文字を検出する手段として閾値を決め、その値を境に輝度情報を２つに分けて写真を白と黒のみの画像にする２値化を行っていた。しかし、文字部分だけを正確に２値化で抽出するためには、文字と背景で輝度の差があることが最低条件であり、図１の写真のように写っている影の色が、文字に使われている色より暗い場合、背景部分の画素の全てが背

景色に含まれるような閾値にすると文字部分の色も含まれてしまうため、図２のように正しく抽出することが不可能となってしまった。また、本来の目的である看板からの認識では、背景に複雑な模様が映っている場合も予想される。そこで、それらの場合でも正しく動作するような文字の抽出方法を考案した。

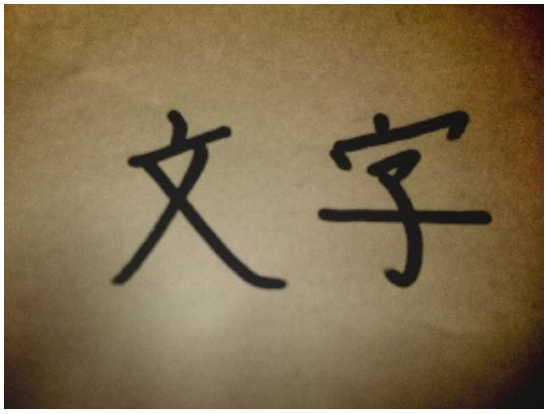

図１　影のほうが文字よりも暗い写真

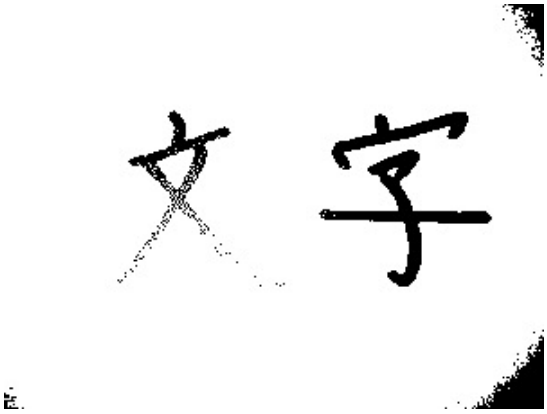

図２　図１の２値化結果

(1) エッジの検出

画像内で色が変わる部分の画素では、周辺画素との色の差が大きくなる。これを利用して、まず文字と背景との境目を求める。

画像内での色の変わり目の画素をエッジという。文字と背景では普通、輝度が異なるため、今回は各画素について周辺画素の輝度の値の差をエッジ値とし、その値からエッジの強さを検出する。

まず横方向のエッジ値は、調べたい画素の周辺画素の輝度の値に以下の値をかける。この値の和は、その画素の右側の方が明るければ正、暗ければ負になり、また輝度の差が大きいほど絶対値が大きくなる。そこで、この値を右向きのエッジ値とする。

-1	0	1
-2	0	2
-1	0	1

縦方向のエッジも同様に、周辺画素に以下の値をかけることで計算できる。

-1	-2	-1
0	0	0
1	2	1

この値から、縦横それぞれのエッジの向き、強さを計算できる。例として、図１の写真についてそれぞれの画素のエッジ値を求め、各画素の RGB 値にそれぞれエッジの強さをあてはめた画像を作成したところ、図３、図４のような画像が得られた。

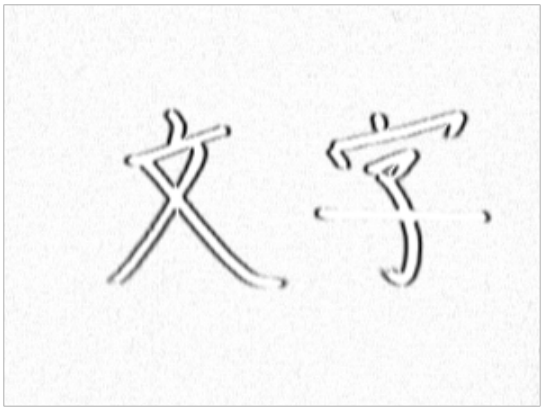

図３　横方向のエッジの強さ

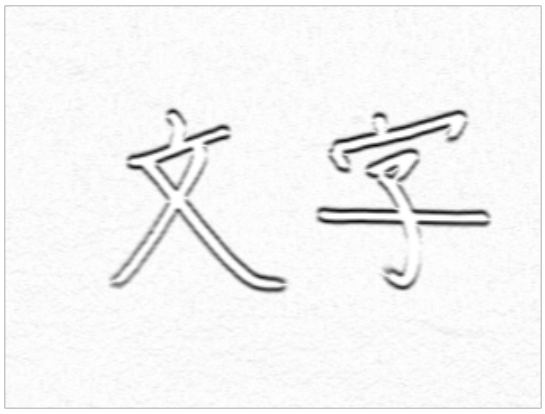

図４　縦方向のエッジの強さ

(2) エッジの方向の検出

先ほど得た値を横方向のベクトルと縦方向のベクトルと考えると、その合成ベクトルから各画素のエッジの向きと大きさが分かるのではないかと考えた。

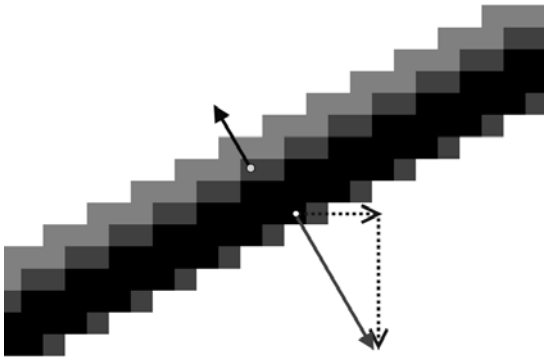

図５　エッジベクトル

まずこのベクトルの角度は、x 成分と y 成分が分かっているため、tan の逆関数によって求めることができる。またこのベクトルの大きさは３平方の定理から計算することができる。こうして得られるベクトルは、たとえば図５のような右上に向けてひかれた直線の画像の中では、図のように、その画素から見て明るい側に線の向きと垂直な角度をとる。また、得られるベクトルの大きさは、図のように、周辺画素の輝度の差が大きいほど大きくなる。

こうして得られるデータを、エッジの角度を色相、エッジの強さを彩度として可視化すると、図６のようになった。

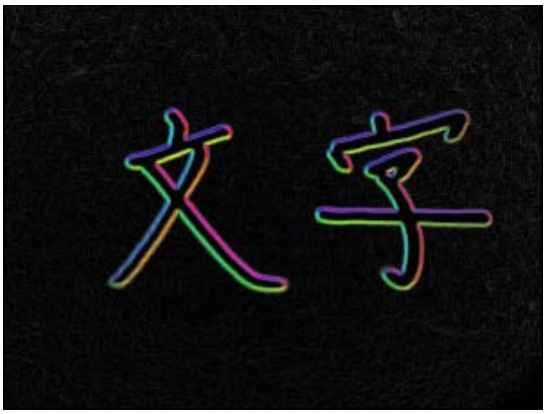

図６　エッジベクトルの角度と強さ

(3) エッジ間の距離検出

エッジの角度を計算したとき、文字の線の部分を挟む２つのエッジの向きはほぼ逆方向になるため、ある２つのエッジについて角度が真逆で、かつそのエッジの角度が２点を繋いだときの角度と一致するような位置関係のエッジ同士をつないで、その間の画素を記録すれば、線の部分を抽出できると考えた。

検出したエッジの強さが一定以上のとき、一般的に文字は背景より暗いため、そのエッジの向きと反対向きに直線を引き、重なった画素をたどって対応する線の端を探す。直線は（エッジの Y 成分）÷（エッジの X 成分）を比例定数 a とする１次関数と考え、$|a| \leqq 1$ のとき（傾きが 45 度以下のとき）は(X、a・X)となる点を X を１ずつ変化させて、$|a|>1$ のとき（傾きが 45 度以上のとき）は(Y/a、Y)となる点を Y を１ずつ変化させてその画素に注目していく。注目画

素について、(注目画素のエッジの強さ)×cos(注目画素と初めの画素のエッジの角度の差)を調べ、その値を足していく。この値の和について、本来注目画素が線から背景に移った時に０になるはずだが、判断基準がそれだけだと、線を挟んだ両側で影などによって背景色が異なっている場合正しく処理することができない。そこで、この和が始めのエッジの強さの半分より小さくなる、または(その値)≦ －１×(始めのエッジの強さ)のとき、初めの点からその点までを文字の線の内側であるとし、内側のそれぞれの画素についてエッジの強さを記録していくようにしたところ処理が安定した。すると文字である部分ほど値が大きくなる。そこで、各画素について、その値を最大値で割ったものをその画素が文字である割合とした。

しかし、この方法では文字のほうが明るい色である場合に対応できない。また、背景に模様がある場合、背景に文字よりも明るい色も暗い色も存在する場合がある。そこで、検出したエッジに対して逆の向きだけではなく同じ向きにも同様の処理を行うことにした。その結果を、文字である割合を輝度として可視化したものが図７である。

図７　ある画素が文字である割合

この得られた値にはノイズが多いため、平滑化を行う。平滑化の際、ただ単純に画像をぼかすだけでは、本当に線である部分の値も大きく変わってしまう。そこで、抽出したエッジの強さを利用して、本来の輪郭をできるだけ保持しながら平滑化を行う方法を考えた。

エッジが強い画素は元の画像では別の線の部分との境目になっていると考えられる。そこで、注目画素とその周辺画素について、エッジが強い画素ほど平滑化後の値への影響が小さくなるようにすれば、２つの線の間にエッジの強い部分が間にあればその両側の値が混ざらず、輪郭が保たれると考えた。そのため、注目画素とその周辺画素について、それぞれのエッジの強さをかけた重み付き平均を計算する。画素(x,y)の得られた文字である割合の値を f(x,y)、エッジの強さをその画像内でのエッジの強さの最大値で割り 0.0～1.0 までの値にしたものを g(x,y)とすると、平滑化後の(x,y)の値は

$$\frac{\sum_{i=-1}^{1}\sum_{j=-1}^{1} f(x+i,y+j)\{1.0-g(x+i,y+j)\}}{\sum_{i=-1}^{1}\sum_{j=-1}^{1}\{1.0-g(x+i,y+j)\}}$$

となる。この処理を先ほどの抽出結果で行ったところ、図８のようになった。

図８　平滑化後の画像

ただし、このままでは線同士の間も線とされてしまっている。そこで、エッジ同士を繋ぐとき、その距離を記録するようにした。

先ほどの画像でそれぞれのエッジ同士の距離を求め、疑似カラーで可視化すると、図９のようになった

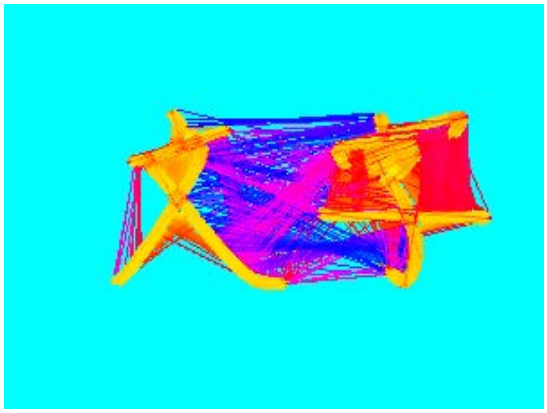

図９　それぞれの線の幅

このように、線の部分と線同士の間の部分に、明確に差がでた。

この記録した距離について、文字の部分と背景の部分では、文字の部分のほうが距離の値が小さいことが多い。そこで、文字である割合を計算する際、（1.0—記録した距離/距離の最大値/10）をかけ、線の距離が短いほど文字である割合が大きくなるようにした。この方法で、街中などの背景が複雑な写真で文字である割合を計算したところ、次の図 10 のようになった。

写真の引用元　街画ガイド
(http://komekami.sakura.ne.jp/archives/567/p8236268)

図 10　街中の写真からの文字部分の検出

このように、複雑な背景である写真の中からでも、文字の大まかな位置を認識することができるようになった。

(4) HLS モデル上の座標の計算

写真の中で、文字と背景では色に偏りがあることが多い。エッジからの線の抽出だけでは「線」と「線と同じ幅の模様」を区別することができない。そこで、各画素の色の情報についても考慮することにした。色の違いを正確に数値化するために、各画素の色の HLS モデル上の座標を計算する。HLS とは、色情報を双六角錐にモデル化

されたもので、H が色相、L が明度、S が彩度を表している。

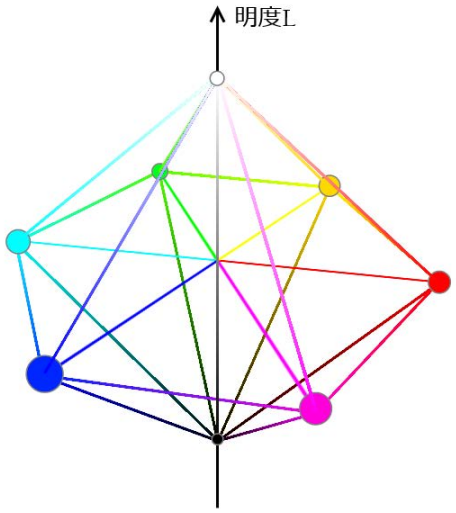

図 11　HLS モデル

調べたい画素の赤の値を R、緑の値を G、青の値を B とおき、それぞれの値を 255 で割って範囲を 0～1.0 に変換する。
X、Y 座標について、図 12 のように双 6 角錐を上から見て考える。

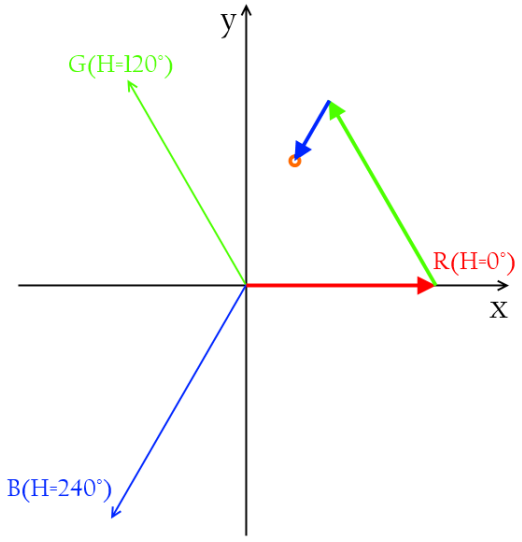

図 12　真上から見た HLS モデル

R の方向を X 軸とすると G の方向は正の X 軸から反時計回りに 120°、B の方向は正の X 軸から反時計回りに 240°であるから、

$$x = R - \frac{G}{2} - \frac{B}{2}$$

$$y = \frac{\sqrt{3}}{2}G - \frac{\sqrt{3}}{2}B$$

と計算できる。また、モデル内の高さ Z について、R,G,B の最大値を Imax、最小値を Imin とすると、

$$Z = L = \frac{Imax + Imin}{2}$$

である。

２つの色の違いは、このモデル上での距離と考えられるので、数値化することができるようになった。

(5) クラスタリング

データを自動的に分類する手法をクラスタリング、同じものと分類された部分をクラスタ、それぞれのクラスタの中心をクラスタ中心という。検出したデータを基に、どの部分が背景でどの部分が文字かを分けるため、k-means++法を参考にクラスタリングを行った。

まず、文字と背景の２つのクラスタ中心を作る。クラスタ中心の初期値は、1 つ目をデータからランダムに選ぶ。それぞれのデータ点 x に関して、最も近いクラスタ中心との距離$D(x)$（この場合は１つめのクラスタ中心からの距離）を計算し、その値が最も大きいものを２つめのクラスタ中心とした。クラスタ数を３つ以上にする場合も同様に、距離$D(x)$が最大である点を新しいクラスタ中心としていく。

次に、各データ点に、クラスタ中心が最も近いクラスタを割り振っていき、割り振ったデータを基に各クラスタの中心を計算

する。そして、もう一度各データ点にクラスタを割り振っていく。これを全てのクラスタの割り当てが変化しなくなるまで行う。先ほどの写真についてクラスタ数を２個としてクラスタリングを行ったところ、図13のようになった。

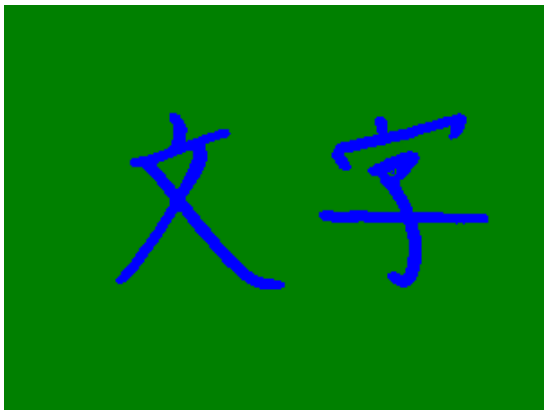

図 13　クラスタリング結果

(6) 文字のクラスタの特定

割り振ったクラスタについて、どのクラスタが文字であるかを特定する。

一般的に文字の形は図形より複雑である。その為、そのクラスタと他のクラスタとの境目に当たる画素数を調べれば、文字と背景で違いが現れ、見分けることができるのではないかと考えた。

そこで、簡単な図形と文字とで実際に比較を行った。その結果が図 14 である。

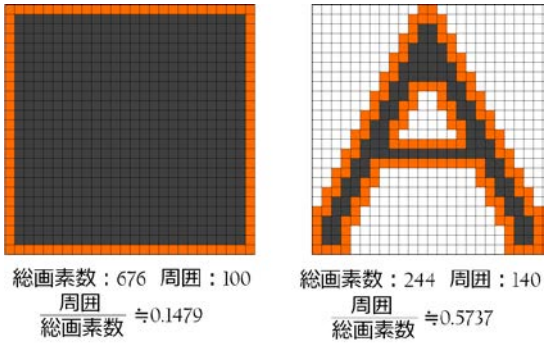

図 14　端の画素の数の違い

このように、四角形と文字とでそれぞれ

$$\frac{\textit{クラスタの端にあたる画素の数}}{\textit{クラスタ内の総画素数}}$$

を求めたとき、図形と文字とで明確な差が生じた。そこで、各クラスタについてこの値を求め、その平均よりも値が大きいクラスタを「文字のクラスタ」とした。

3.2 細線化

文字の形を認識する際、線に太さがあると認識しづらい。そこで、文字の中心１画素以外を除去する細線化を行った。今回は、田村の方法（参考文献４）を基に、文字の認識に最適な細線化方法を考案した。

文字の画素を１、背景の画素を０とおく。この時、例えば注目画素が１でその右が０の時、その画素は線の右の端であると考えられる。上、左、下も同様に線の端であるので、そのような画素を除去(０に)していくと、線が細くなっていく。しかし、それだけでは線が途切れてしまう。そこで、図15、図 16 のように、除去しないパターンに相当する場合は除去を行わないようにした。その結果が図 17 である。

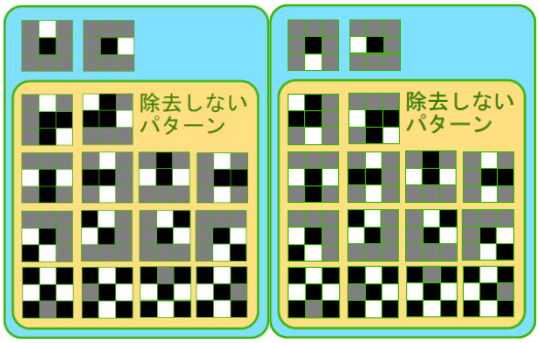

図 15（左）　除去パターン１

図 16（右）　除去パターン２

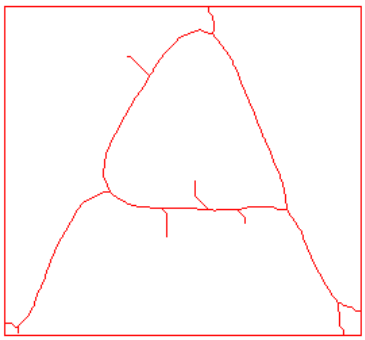

図 17　細線化結果

図 17 をみると、線が本来無いはずの場所に飛び出てしまっているところがある。また、線の交点付近が大きく歪んでおり、このままでは処理が困難である。

線が飛び出ているところを処理前の画像データで確認すると、飛び出ている線の端の部分が 1 画素分の突起になっていて、そこが線の端であるとされてしまっていることが分かった。そこで、図 18 のような 4 パターンに相当する場合は、除去しないパターンにあてはまっていても強制的に除去するようにした。その結果が図 19 である。このように、飛び出ている線を大幅に減らすことができた。

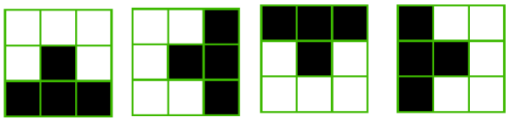

図 18　強制除去するパターン

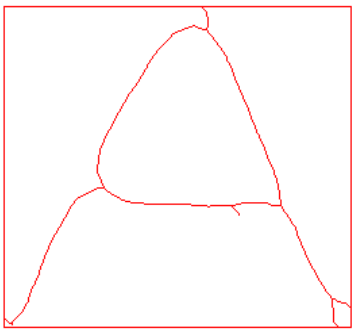

図 19　細線化結果

また、歪みについて、細線化のパターンが「上または右」が 0 の場合と「下または左」が 0 の場合という 2 パターンのみであったが、「上または左」「下または右」が 0 であるというパターンも考えられる。そこで、その 2 パターンを追加すれば、除去していく向きの偏りが減り、歪みが軽減されると考えた。そこで、除去パターン 1 、 2 を参考に、図 20、図 21 の除去パターン 3 , 4 を作成した。この処理の結果が図 22 である。

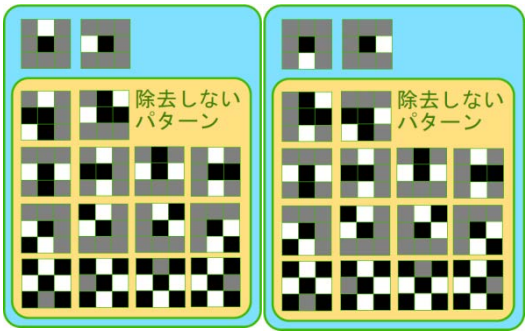

図 20（左）　除去パターン 3

図 21（右）　除去パターン 4

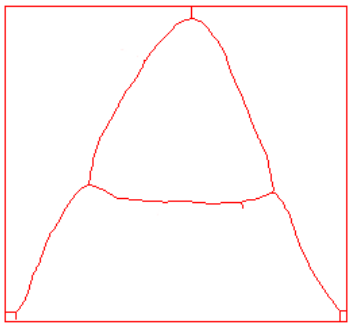

図 22　細線化結果

このように、線の歪みを大幅に軽減することができた。

3.3 線の座標抽出

細線化した画像データから、どこからどこに線が引かれているかを抽出する。そのためには、線の端、交点、角を認識すればよいと考えた。

細線化したデータにおいて、ある画素が１で、隣り合う１の画素が１つだけの場合、その画素は線の画素と考えられる。また、ある画素が１で、隣り合う１の画素が３つ以上ある場合、その画素で線が交わっていると考えられる。そこで、まずこの２種類の特徴点を抽出する。

次に、抽出した特徴点を結んでいく。この際、線の角の抽出も行いたいので、輪郭追跡を行った。輪郭追跡では、文字のクラスタと他のクラスタとの境界部分を、反時計回りに一周する。

輪郭追跡の開始位置は、特徴点を結んでいきやすいよう、線の端の中の一つから行う。線の端が存在しない場合はそのクラスタ内でもっとも上にある画素から行う。

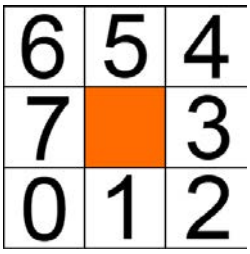

図 23　進む方向

進む方向に、図 23 のように番号をふる。輪郭を辿る際、例えばひとつ前に下に進んでいた場合、次の画素は今の注目画素の左上や左になることはない。なぜなら、下に進んだ時点で、すでに左と左下（今の注目画素から見て左上と左）はすでにチェックされ除外されているということなので、下に進んだ後も選ばれることはない。一般化すると、ひとつ前の向きから時計回りに３つ分の向きに進むことはない。この性質を利用し、輪郭を検索するときの１つ前の向きを Vold、次の向きの検索の初めの向きを Vnew とすると、

$$\mathrm{Vnew} = (\mathrm{Vold} + 6) mod\ 8$$

となる Vnew から向きの検索を始めれば効率が良い。このように開始地点に戻るまで輪郭を辿る。

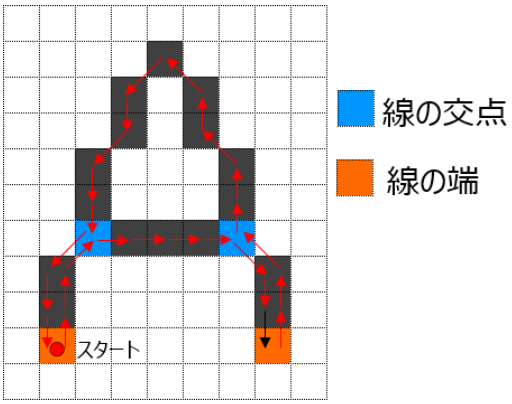

図 24　輪郭追跡

辿っていく際、最後に通った特徴点からひとつ前までの角度を計算し、次に進む向き

との差が一定値以上のとき、その画素を線の角とする。角も特徴点であるため、その画素から次の特徴点までで計算する角度は、その画素からの角度となる。

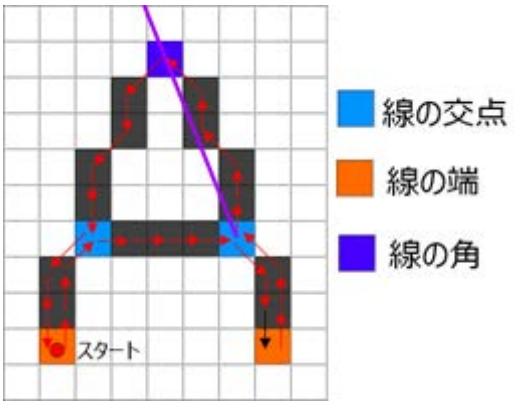

図 25　線の角の認識

こうして抽出した３種類の特徴点を、輪郭追跡の際に通った順に結んでいく。これで、画像から線を抽出できた。

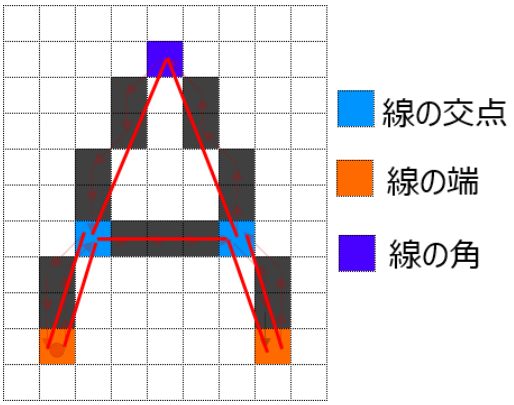

図 26　線の抽出

しかし、この段階では、一周しているため２回通る場所ができ、図のように抽出した線が２重になってしまうことがある。また、交点によって本来１つの線であるべき線が２つの線に分けられてしまっている部分がある。

そこで、これらの線の調整を行う。

線が２重になっている部分を除去する際、線の角の位置は通る向きによって変わることがあり、完全に同じ線になるとは限らない。そこで、線一本一本の座標を確認していき、類似した直線がないかを探す。２本の直線について、角度がほぼ同じで、かつそれぞれの対応する点との距離の和が一定値以下の場合、その２本の線を同じ線だとみなし、対応する点同士の中央を結んだ線を新しい線とし、元の２本の線を除去するこの際に、線の座標が変わった場合、もともと繋がっていた他の線と離れてしまう場合がある。そこで、座標を変更する際、ほかの線がその座標を使用していれば、その座標も新しい線に合わせるようにした。

また、分かれている線について、２つの線が頂点で接しており、かつ角度の差が一定値以下の場合は、接していないほうの２つの点から新しい線をつくり、もとの２本の線を除去することで１本にまとめた。

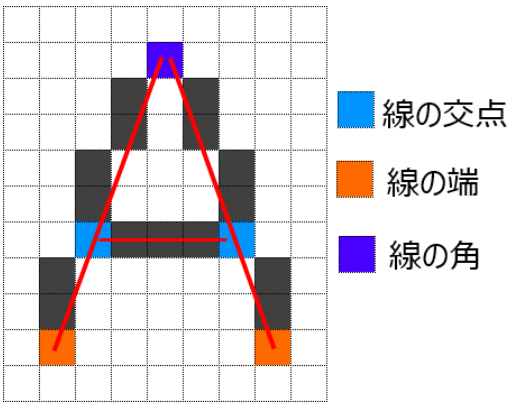

図 27　線情報の調整

3.4 文字の判定

登録された文字データと照合するために、読み取った文字の大きさ、縦横比を、次のようにして登録されたデータのものに合わせる。読み取った文字の大きさについ

て、抽出した線の頂点のなかから最も上にある点、右にある点、左にある点、下にある点をもとめる。すると、左上の頂点が（最も左の点の x、最も上の点の y)、右下の頂点が（最も右の点の x、最も下の点の y）である長方形に、文字がちょうどおさまる。この長方形の横幅を文字の幅、この長方形の高さを文字の高さとする。読み取った線の頂点の x 座標にそれぞれ$\frac{登録された文字データの幅}{読み取った文字の幅}$をかけ、y 座標にそれぞれ$\frac{登録された文字データの高さ}{読み取った文字の高さ}$をかければ、大きさ、縦横比が同じになる。

次に、文字の判定のために文字の形の一致度を求める。文字の形の一致度は、各座標の絶対位置だけではなく、線の角度や相対的な位置の違いなどを含め、字に癖があったり字のバランスがある程度崩れていても認識できるようにした。読み取った線について、登録された文字データのすべての線に対して｛1.0—(角度の差)/360｝×（対応する頂点間の距離の合計）/(判定用データを囲む長方形の対角線の長さ)×(その線の長さ)を求める。全ての線でこの値を計算し、その和を、読み取った線の長さの和で割れば、線の一致度の合計が出せる。また、頂点同士の位置関係について、ある点より右か左か、上か下か、という関係が同じである点を数え、(一致した点の情報の数)/(全ての組み合わせ)を求める。これを先ほどの線の一致度にかけると、完全に同じ形のときを１とする一致度が求まる。

3.5. 推測

人は文中に読めない文字があっても、前後の文字から推測して読むことができる。もし同じことが機械でもできれば、読み取れない文字を減らし、また読み間違いも減り精度を上げることができると考えた。推測のために、まず単語のデータベースを作り、言葉を大量に登録する。読み取った文字を、その登録された全ての単語と比較し、一致度を調べた。読み取った単語には、間違った文字が含まれている可能性があるため、登録された単語の側から相当する文字を探していく。また、言葉の最初の文字と最後の文字は言葉を判別するうえで重要であると考えた。そこで、単語の初めの文字と最後の文字は一致しているときに４pt、それ以外の文字は場所も同じで２pt、何文字目かがずれていてもその文字が存在すれば 1 pt とし、文字の一致度は「$\frac{ポイントの合計}{全てあっていたときの合計}$」とした。さらに文字数の違いについて、文字数の一致度を文字数の差に反比例するように、$\frac{1}{(文字数の差)+1}$とした。単語の一致度を文字の一致度と文字数の一致度の積とすると、単語の一致度は完全に同じ単語のときに 1 となる割合となる。この値が最大になる単語が、本来そこに書かれていた単語であると考え校正を行うことで、正しく読み取れていない文字が混じっていても、自然な読み取りができるようになった。

3.6. 学習機能

文字を認識していく中で、自動的に精度を上げていく「学習機能」が実現すれば、文字の登録効率が非常によくなると考えた。判定時、形の一致度が８割以上の場合や、読み取りに失敗した文字を推測したときに文字数が完全一致かつ単語の一致度が ８割

以上の場合は、読み取った文字を「その文字」としてデータベースに追加するようにした。また、形の一致度が８割以上でかつ使われている線の数が同じである場合、その登録された文字データの各座標を、読み取った文字の座標に、一致度に応じて近づけるようにした。これで、認識の際にある程度の精度で読み取れていれば、使っているだけで自動的に精度が上がっていくようになった。

４．まとめと今後の課題

今回の研究で、文字を線として認識し、判定することができた。しかし、今の段階では文字が斜めになっている場合に正しく判定することができない。そこで、今後は判定時に、それぞれの最も長い線の向きが同じになるように回転させてから判定するなどして、写真が斜めに撮られていても認識できるようにしたい。

また、背景に影や模様が入っているような場合でも、文字の大体の場所の抽出ができるようになった。しかし、実際に背景が複雑な写真のクラスタリングを行おうとすると、様々な模様と区別するため多くのクラスタ中心が必要になる。そこで今後、最適なクラスタ数を計算する方法を考案し、より精度のよいクラスタリングを行えるようにしたいと考えている。

５．参考文献

(1) Visual　Basic　中学校
「http://homepage1.nifty.com/rucio/main/main.htm」
(2) DOBON.NET
「http://dobon.net/index.html」
(3) C#と VB.NET の入門サイト
「http://jeanne.wankuma.com/」
(4) 画像処理ソリューション
「http://imagingsolution.blog107.fc2.com/blog-entry-151.html」
(5) 村上研究室　画像処理
「http://ipr20.cs.ehime-u.ac.jp/column/gazo_syori/index.html」
(6) イメージングソリューション
「http://imagingsolution.net/」
(7) 無作為研究所「http://www.faicha.com/」

６．謝辞

今回の研究を行うにあたり、顧問の米田隆恒先生には多大なご指導を賜りました。また、サイエンス研究会の先輩方にも多くの助言をいただきました。この場を借りて、深く御礼申し上げます。