CK-001

統計的手法による画像からの点字分離アルゴリズム
Separation Algorithm for Braille from Image with Statistical Approach

伊藤 祥一 †
Shoichi Ito

藤澤 義範 †
Yoshinori Fujisawa

1 はじめに

点字は視覚障害者が使う文字である．日本で使われている点字は 3 行 ×2 列の計 6 点からなる 6 点式点字である．6 つの点は図 1 のように番号付けされており，それぞれ「1 の点」「2 の点」などと呼ぶ．点の大きさは JIS T 0923 [1] により規定されており，点の直径が 1.5[mm] 程度である．点と点の間隔は 2 種類あり，図 1 の 1 の点と 4 の点の間隔が 2.25[mm] 程度，2 文字並んだ点字の 1 文字目の 4 の点と 2 文字目の 1 の点の間隔が 3.45[mm] 程度である．

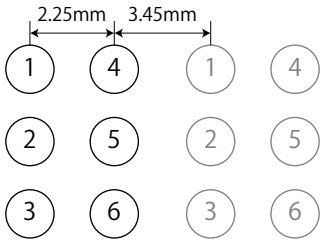


図 1 点字の点番号と点間隔

点字をコンピュータで読み取る方法としては，フィルム状圧力センサ [2] やフラットベッド型イメージスキャナ [3] 等のハードウェアを用いるものやデジタルカメラを用いるもの [4] 等が提案されている．

画像を用いる方法では，特定の画素に点が映るかどうかで判定を行うなど入力画像のサイズに制約がある．また，点のない部分はそもそも画像には写らないため一般的な画像解析手法では点字の空列を検出することができない．

本稿では，圧力センサやカメラから入力された点字の模様から点の有無と間隔を自動推定し，6 点式点字の 1 文字ごとに分離するアルゴリズムについて述べる．

2 点字 1 文字の符号化

点字の 1 文字は，Unicode においては “Braille Patterns” という領域に置かれており，U+2800 から U+28FF までの 256 文字に 8 点式の点字が割り当てられている [5]．6 点式点字は 8 点式点字の一番上あるいは一番下の 1 行が空白のもので代用する．この文字コードをそのまま使うのは冗長なため，本稿では図 1 の 1 の点を MSB，6 の点を LSB として点のあるところに 1，点のないところに 0 を並べた 6 ビットの数値で符号化する．図 2 は点字で「ながの」と書かれており，1 文字ごとにこのルールで符号化した 6 ビットの並びが割り当てられている．

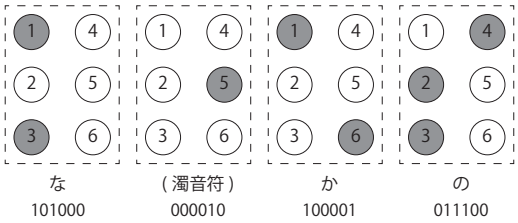


図 2 点字の符号化

3 ラベリングと重心座標の算出

図 3 は点字の模様をコンピュータに取り込み，角度補正や二値化などの一連の処理 [6] を行った例である．この例はカメラからの光学画像入力のため，光の当たり方に起因して点字の点の部分が真円ではなくゆがんだ形になっている．

図 3 点字の画像 (612×75px)

図 3 を入力としてラベリング [7] を行うと図 4 となる．ラベリングとは連続した領域に番号を割り振る処理のことである．図 4 では点字の凸部分の 1 つ 1 つが異なる色で塗り分けられており，それぞれが一つの連続した領域として認識されていることがわかる．

図 4 図 3 の画像をラベリングしたもの

図 4 で得られた個々の領域は点字の点としては不規則なゆがみ方をしているので，各領域の重心をその領域の代表点とする．図 4 のようにラベリングされた領域は 38 あり，個々の領域の重心座標を求めてプロットすると図 5 のようになる．個々の点のゆがみの影響をほとんど受けず整然とした点の並びが得られていることがわかる．

† 長野工業高等専門学校 電子情報工学科

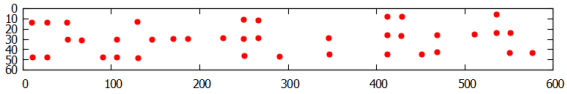


図 5　ラベリング + 重心計算により検出された点の場所

4　重心座標から点字への切り出し

本章では座標の集合を 1 文字ごとの点字として切り出すアルゴリズムについて述べる．

4.1　規格化

重心座標に通し番号を振り，(X_i, Y_i) とする．$i = 0, 1, 2, ..., N-1$ で，図 5 の重心座標であれば $N = 38$ である．この段階で N 個の (X_i, Y_i) はどのような順序で並んでいても構わない．

第一段階として，今後の処理が入力画像のサイズに依存しないようにするため，座標の規格化を行う．点の y 座標を $0 \leq y \leq 100$ の範囲に収まるように座標の並行移動とスケール変換を行う．単に $0 \leq y \leq 100$ の範囲に収まっているだけではなく最小の y 座標が 0，最大の y 座標が 100 に一致するようにする．同時に x 座標に対して左端の点の x 座標が 0 に一致するように座標の並行移動を行い，y 軸に対して行ったスケール変換をそのまま適用する．

具体的には図 5 で示されている 38 個の重心座標を (X_i, Y_i) $(i = 0, 1, ..., 37)$ とする．X_i と Y_i の最小値はそれぞれ $X_{min} = 10.46$ と $Y_{min} = 5.74$ であり，最大値はそれぞれ $X_{max} = 576.20$ と $Y_{max} = 48.34$ である．拡大率を r とすると，

$$r = \frac{100}{Y_{max} - Y_{min}} \tag{1}$$

であるので，変換後の y 座標を

$$\hat{Y}_i = r(Y_i - Y_{min}) \tag{2}$$

と定義する．これですべての y 座標について $0 \leq \hat{Y}_i \leq 100$ の範囲に収まるような変換が行われる．同時に，$\hat{Y}_i$ の最小値は 0，最大値は 100 になっている．x 座標についても式 (1) から計算された r でスケール変換を行う．同時に，すべての X_i から X_{min} を引くことで左端の点の x 座標を 0 にする．変換後の x 座標を

$$\hat{X}_i = r(X_i - X_{min}) \tag{3}$$

と定義する．

図 5 の重心座標から式 (1) により計算した拡大率 r は 2.35 である．図 6 が規格化前の点の座標 (X_i, Y_i) で，(1)(2)(3) により規格化した重心座標 $(\hat{X}_i, \hat{Y}_i)$ をプロットしたものが図 7 である．図 7 では確かに y 座標が最小 0，最大 100 になるようにスケール変換と並行移動がされている．また，x 座標も左端の点が 0 をとり，y 軸と同じ拡大率でスケール変換がなされていることがわかる．

4.2　高さ方向への 3 分割

点字は 3 つの行を持つため，点がどの行に属しているかを振り分ける必要がある．点字の 1 文字は 3 行の点から構成されるので，0 から 100 までの範囲を単純に 3 分割して点の y 座標が

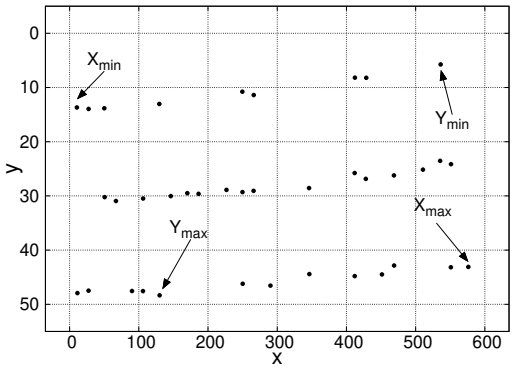


図 6　規格化前の点の座標

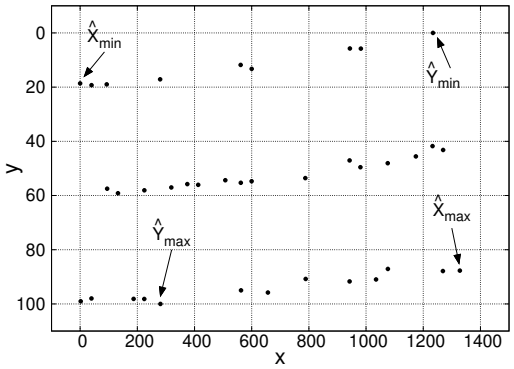


図 7　規格化後の点の座標

- $0 \leq y < 33.3$
- $33.3 \leq y < 66.7$
- $66.7 \leq y \leq 100$

の基準で 3 行のうちのどの行に入っているかを分類する．後の処理が簡単になるように点の 1 つ 1 つに座標以外に行番号という属性も持たせることにして，$0 \leq y < 33.3$ のものに行番号 0，$33.3 \leq y < 66.7$ のものに 1，$66.7 \leq y \leq 100$ のものに 2，をそれぞれ振っておく．

4.3　高さの平均とばらつきの計算

分割された行ごとに y 座標の値の平均値と標準偏差を求める．

一般に N 個のデータ t_i $(i = 0, 1, 2, ..., N-1)$ があったとき，平均値 $\bar{t}$ は式 (4) で与えられる．

$$\bar{t} = \frac{1}{N} \sum_{i=0}^{N-1} t_i \tag{4}$$

また，標準偏差を σ とするとその 2 乗の σ^2 は分散と呼ばれ，式 (5) で与えられる．データのばらつきが正規分布に従うと仮定すれば，$\bar{t} \pm 1\sigma$ の範囲に全体の 68.2%，$\bar{t} \pm 2\sigma$ の範囲に全体の 95.4%，$\bar{t} \pm 3\sigma$ の範囲に全体の 99.6% が含まれることが知られている．

$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N-1}(t_i - \bar{t}\,)^2 \tag{5}$$

図 7 には行番号 0 に分類された点が 9 個ある．この y 座標の平均値を求めると 12.29 となり，標準偏差は $\sigma_1 = 6.62$ となる．平均値と $\pm1\sigma_1, \pm2\sigma_1, \pm3\sigma_1$ の範囲を重ねてプロットしたものが図 8 である．

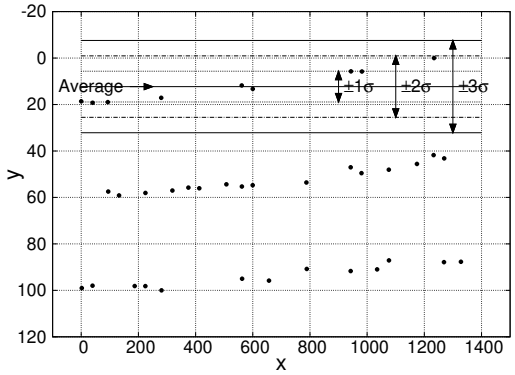


図 8　行番号 0 の y 座標平均値と $\pm1\sigma, \pm2\sigma, \pm3\sigma$ の範囲

図 7 の中央の行番号 1 と 2 についても同様に平均値と標準偏差を求めてプロットしたものが図 9 と図 10 である．

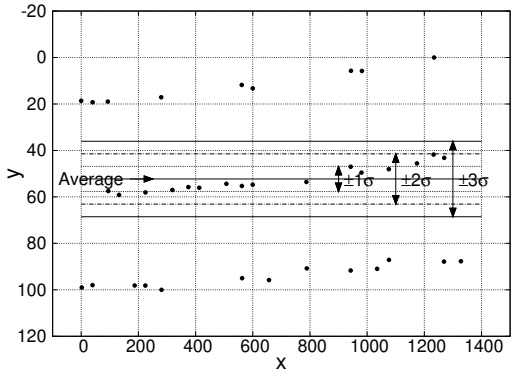


図 9　行番号 1 の y 座標平均値と $\pm1\sigma, \pm2\sigma, \pm3\sigma$ の範囲

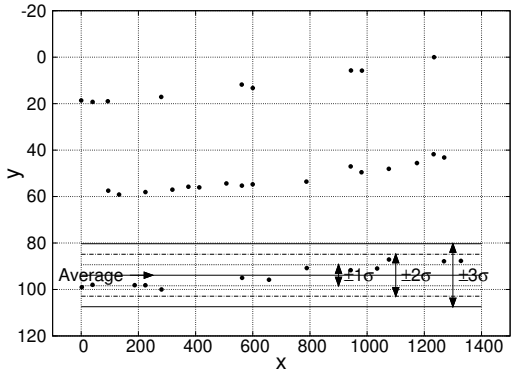


図 10　行番号 2 の y 座標平均値と $\pm1\sigma, \pm2\sigma, \pm3\sigma$ の範囲

図 8, 9, 10 から，規格化された y 座標 $\hat{Y}_i$ が各行の平均値を中心にして $\pm3\sigma$ の範囲に収まっていることがわかる．

ここで求めた標準偏差の平均値 $\bar{\sigma}$ を式 (6) により求めておく．標準偏差の平均をとることに物理的な意味があるわけではないが，列方向のゆらぎの指標としてこの $\bar{\sigma}$ を用いる．

$$\bar{\sigma} = \frac{1}{3}(\sigma_1 + \sigma_2 + \sigma_3) \tag{6}$$

4.4　列方向のグループ化

次に点の重心座標 $(\hat{X}_i, \hat{Y}_i)$ を x 座標に基づいてソートする．これで同じ列に属する点がお互いに隣接した場所に並ぶ．続いて，列方向に並んでいる点のグループ化を行い，同じ列に属するものには同じ列番号をつける．この手順を擬似コードで表すと次のようになる．

1: $i \leftarrow 0, j \leftarrow 0, \dot{X} \leftarrow \hat{X}_i$
2: **while** $i < N$ **do**
3:　　$A_i \leftarrow j$
4:　　$i \leftarrow i + 1$
5:　　**if** $|\hat{X}_i - \dot{X}| > 3\bar{\sigma}$ **then**
6:　　　　$j \leftarrow j + 1$
7:　　　　$\dot{X} \leftarrow \hat{X}_i$
8:　　**end if**
9: **end while**

ここまで終了した時点での j の値を C_{max} として保存しておく．図 5 の重心座標では C_{max} は 22 となった．

4.5　同じ列とみなす許容範囲について

4.4 章では同じ列に収まっているとみなす範囲を，基準となる点から $\pm3\bar{\sigma}$ の範囲とし，この範囲を実際にプロットしたのが図 11 である．1 つの列に点が複数ある場合のみ $\pm3\bar{\sigma}$ の範囲をプロットしてある．4.4 章の処理では，この範囲に収まっている点を同じ列とみなして同じ列番号を割り当てる．

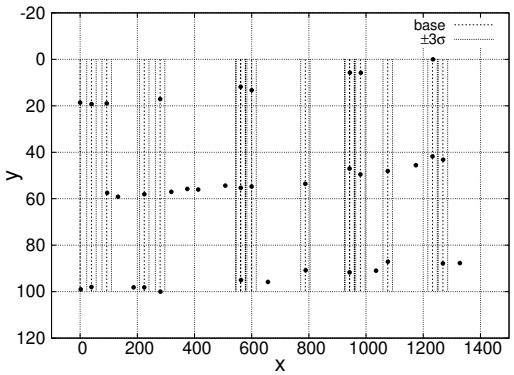


図 11　基準となる点から左右に $\pm3\bar{\sigma}$ の範囲

図 11 の $x = 540 \sim 620$ を拡大して $\pm1\bar{\sigma}, \pm2\bar{\sigma}, \pm3\bar{\sigma}$ の線も重ねたものを図 12 に示す．このデータに限っていえば，同じ列に収まっているとみなす許容範囲としては $\pm3\bar{\sigma}$ は少々広すぎで，$\pm1\bar{\sigma}$ で十分とも考えられる．

点の列方向のゆらぎが，点字文字列全体の横方向の解像度と比較したときに比較的大きい場合，$\pm1\bar{\sigma}$ 基準よりも $\pm3\bar{\sigma}$ 基準の場合は隣の列の点を「同じ列である」とみなす可能性が高くなる．狭くしすぎると，点の列方向のゆらぎが大きいときに

同じ列とみなされない可能性が高くなる．5 章では 612×75, 712×132, 743×118, 336×24 ピクセルの 4 つの入力画像を元に評価を行っているが，$\pm 3\bar{\sigma}$ 基準で特に問題は生じなかった．将来的に，より多様な入力に対して安定した処理精度を保つために動的な補正などが必要になる可能性がある．

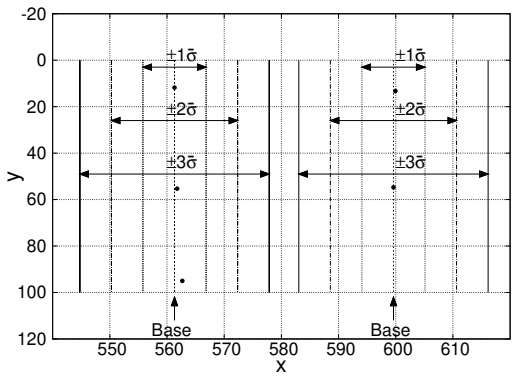


図 12　図 11 の $x = 540 \sim 620$ を拡大して $\pm 1\bar{\sigma}, \pm 2\bar{\sigma}, \pm 3\bar{\sigma}$ の線も重ねたもの

4.6　点の間隔の決定

4.4 章の方法で同じ列に属する点をグループ化することができた．ただしこれは 1 つでも点のある列に左から 0, 1, 2, ⋯ と番号を振っていることに対応しているので，列にまったく点がない場所は空列をパディングする必要がある．たとえば図 5 の座標 $(\hat{X}_i, \hat{Y}_i)$ を 4.1 章〜4.5 章の方法で処理すると最終的に C_{max} の値は 22 になるが，これは図 3 を左から見ていったときに 0〜22 というラベル番号がついた合計 23 の列が見えるということである．9 と 10 の間，13 と 14 の間，14 と 15 の間，18 と 19 の間の間隔は実際はそれぞれ空列が存在しているがカメラには写っていない部分である．点字は点のない部分も意味を持つので，ここを自動で認識して空列をパディングする．このパディングの前段階として，1 つの点字の中の 1 列目と 2 列目の間隔 d_1 と，2 つの点字の間隔 d_2 を求める．

まず列番号が同じ点 (最大 3 個) を見て，それらの x 座標の平均値を求める．ここでは便宜上，$\bar{p}_i$ とする．

$$\bar{p}_i = (\text{同じ列番号を持つ点の } x \text{ 座標の平均}) \tag{7}$$

この i は $0, 1, 2, \cdots, C_{max}$ の値をとる．次に隣接する $\bar{p}_i$ と $\bar{p}_{i+1}$ の差の絶対値を式 (8) により計算する．

$$\delta_{i,i+1} = |\bar{p}_i - \bar{p}_{i+1}| \tag{8}$$

図 13 は点字の 1 行目のみを示したものである．黒丸が点のある列を示し，白丸が点のない列を示している．点が空列を挟まずに並んでいる部分は似た値が交互に出現し，空列がある場所はこの値が大きく変化する．図 13 では，$\delta_{01} \fallingdotseq \delta_{23} \fallingdotseq \delta_{56}$ が 1 文字の中での 1 の点と 4 の点の間隔を表しており，$\delta_{12} \fallingdotseq \delta_{34}$ が 2 文字並んだ 4 の点と 1 の点の間隔を表している．空列がある部分は δ_{45} のように値が大きくなる．点字の規格は決まっているので，この値から何列の空列を挟んでいるかを推測することができる．

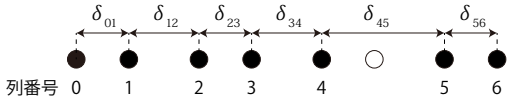


図 13　列間隔 (点字の 1 行目のみ図示)

次に δ_{ij} を昇順にソートする．すると，図 13 の $\delta_{01} \fallingdotseq \delta_{23} \fallingdotseq \delta_{56}$ に該当する値が先頭付近に集まる．この平均値を d_1 とする．続いて $\delta_{12} \fallingdotseq \delta_{34}$ に該当する値が集まる．この平均値を d_2 とする．これ以降は δ_{45} のように空列を 1 つ以上含む大きな値が集まるので無視する．d_1 のグループと d_2 のグループの境目は，ソートされた値で隣接する 2 つの値の比を計算し，10% 以上の差がある場合に境目であると判断した．

いま例に挙げている図 5 では，$d_1 = 38.27$，$d_2 = 55.97$ となった．入力が画像データのため，単位はピクセルである．

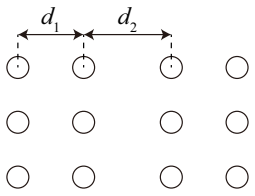


図 14　点の間隔 d_1, d_2 の定義

4.7　d_2 が直接決定できない場合への対応

ここで図 15 のように d_2 を直接決定することができない場合があり得るので，別の算出方法を追加する．図 15 では 4 文字の点字が並んでおり，これを適当な点字フォントで画像化して測定したところ，1 文字目の 2 列目の点と，3 文字目 1 列目の点の間が 183 ピクセル，3 文字目の 1 列目の点と 4 文字目 1 列目の点の間が 113 ピクセル，4 文字目の 1 列目の点と 2 列目の点の間が 44 ピクセルであった．d_1 は 44 と決まるが，その次が 113 で，44 との比率は 2.57 倍にもなる．図 15 では d_2 を測定できる場所 (距離 d_2 だけ隔てた 2 点) が存在しないので，113 をそのまま d_2 として採用してしまうのは適切でない．点字の JIS 規格 [1] によれば，1 の点と 4 の点の間隔の中央値は 2.25[mm] で，2 文字並んだ点字の 1 文字目の 4 の点と 2 文字目の 1 の点の間隔の中央値は 3.45[mm] である．この比率を求めると次のようになる．

$$\frac{d_2}{d_1} = \frac{3.45}{2.25} = 1.5333... \tag{9}$$

d_1 からどのくらい離れたところに次の列が存在すべきかという推定値としてこの比率を用いて，

$$d_2 = 1.533 d_1 \tag{10}$$

を d_2 の推定値として採用してその後の処理を進めるようにした．

4.8　列のパディング

点の間隔 d_1 と d_2 が求まったので，δ_{ij} と d_1, d_2 を比較して空列の数を決定し，適切な数の空列をパディングする．

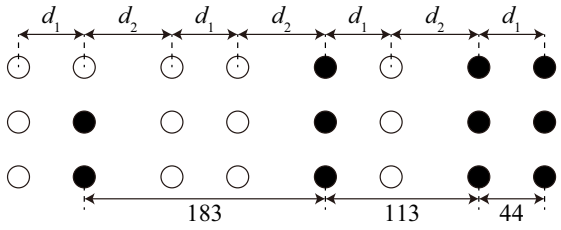


図 15　d_2 を決められない例

$\delta_{i,i+1}$ を計算し，これが式 (11) を満たしていればケース A とする．

$$0.9d_1 < \delta_{i,i+1} < 1.1d_1 \tag{11}$$

ケース A は隣接する 2 点が 1 つの点字の中の 1 列目と 2 列目にあると考えられる場合である．ここで 10% の誤差を許容しているがこの 10% という値は経験的に決めた値である．

$\delta_{i,i+1}$ を計算し，これが式 (12) を満たしていればケース B とする．

$$0.9d_2 < \delta_{i,i+1} < 1.1d_2 \tag{12}$$

ケース B は隣接する 2 点が 2 つの点字の 2 列目と 1 列目にあると考えられる場合である．

$\delta_{i,i+1}$ を計算し，これが式 (13) を満たしていればケース C とする．

$$0.9(d_1 + d_2) < \delta_{i,i+1} < 1.1(d_1 + d_2) \tag{13}$$

ケース C は隣接する点が 1 文字目の 1 列目と 2 文字目の 1 列目 (図 16)，あるいは 1 文字目の 2 列目と 2 文字目の 2 列目 (図 17) にあると考えられる場合である．どちらの場合も点と点の間に 1 列の空行を挿入するという同じ処理で済ませることができる．

$\delta_{i,i+1}$ を計算し，これが式 (14) を満たしていればケース D とする．

$$0.9(2d_1 + d_2) < \delta_{i,i+1} < 1.1(2d_1 + d_2) \tag{14}$$

ケース D は隣接する点が 1 文字目の 1 列目と，2 文字目の 2 列目にあると考えられる場合である (図 18)．この場合，空列を 2 列はさんでいることが推測できる．

$\delta_{i,i+1}$ を計算し，これが式 (15) を満たしていればケース E とする．

$$0.9(d_1 + 2d_2) < \delta_{i,i+1} < 1.1(d_1 + 2d_2) \tag{15}$$

ケース E は隣接する点が空白文字を挟んでいると考えられる場合である (図 19)．この場合もケース D と同様，空列を 2 列挟んでいることになる．

ケース C の場合は 1 列，ケース D と E の場合は既知の列の間に 2 列の空列を挿入すればよい．同様に考えて，ケース F(図 20) の場合は 3 列，ケース G(図 21) の場合は 4 列の空列をそれぞれ挿入すればよい．通常，点字の文章は空白 1 文字を挟んで分かち書きされているため，4 列以上の空列を挟むことはほとんどない．このため，空列の推定としてはケース G までを考えておけば十分である．4.4 章の処理終了時点で，点が存在する列の数が C_{max} に入っているので，ここで空列を挿入した場合，C_{max} にも挿入した列数を加えておく．これにより，終了時点での C_{max} は空列も含めた列の数になっている．

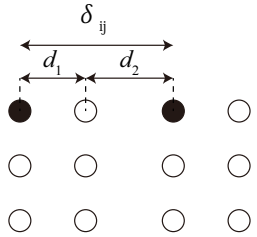


図 16　ケース C (その 1)

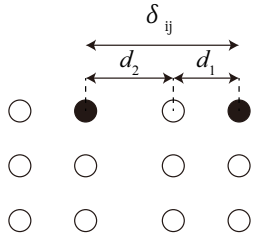


図 17　ケース C (その 2)

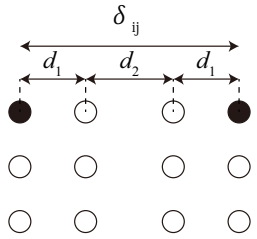


図 18　ケース D

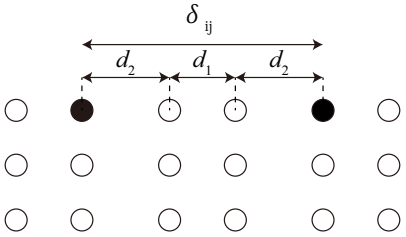


図 19　ケース E

図 22 のように，1 文字目の 1 列目が空列で始まる場合の処理を考える．図 22 のように 1 文字目と 2 文字目の距離 δ_{01} が式 (12) を満たしている場合，先頭に 1 列の空列を挿入する．

同様に，ケース C (その 2)，ケース E，ケース F が 1 文字目で発生した場合，先頭に 1 列の空列を挿入したうえで通常通りの列挿入操作を行う必要がある．難しいのは，ケース C が 1 文字目で発生した場合，ケース C (その 1) なのかケース C

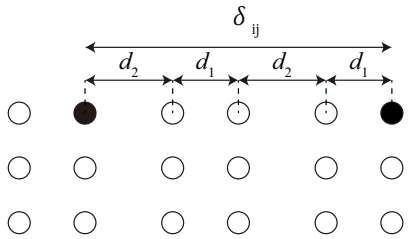


図 20　ケース F

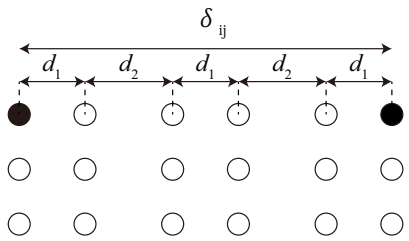


図 21　ケース G

(その 2) なのかは「1 文字目かつ $\delta_{01} = d_1 + d_2$」という条件だけでは区別がつかないという点である．同様に，ケース F が 1 文字目で発生した場合，「1 文字目かつ $\delta_{01} = 2d_1 + 2d_2$」という条件だけでは図 23 と区別がつかないという点である*．現段階では，ケース C (その 2)，ケース E，ケース F が 1 文字目で発生した場合のうち，ケース E のみに対応している．

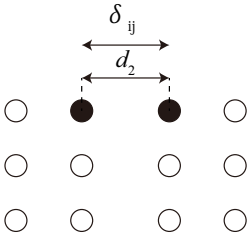


図 22　例外処理 1

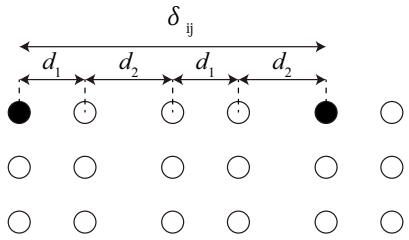


図 23　例外処理 2

4.9　0/1 の並びとして出力

最後に内部的な情報を 0 と 1 の並びとして点字 1 文字を 6 ビット値として表した形で出力する．

列番号 $i \leftarrow 0$ とする．点字の左の列の縦 3 点を順に走査していく．まず一番上の点に注目し，すべての点の列番号と行番号をチェックし列番号が i で，かつ行番号が 0 のものを探し，見つかれば 1 を出力し，見つからなければ 0 を出力する．次に注目点を中央の行にずらし，列番号 i かつ行番号 1 の点を探して見つかれば 1 を出力し，見つからなければ 0 を出力する．一番下の点についても同様にする．これで点字の 1 列の 0/1 の出力ができた．次に i に 1 を足して同じことを繰り返す．$i = C_{max}$ のところまで繰り返したら処理を終了する．

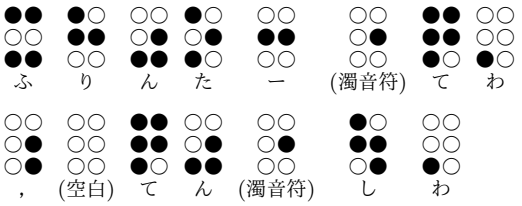
C_{max} は 0 から数え始めているので，点字 1 文字=2 列が整数個並んでいれば C_{max} は奇数で終わっているはずである．もし C_{max} が偶数の場合は C_{max} に 1 を足してから上記の処理を開始する．これにより右端の点字の 2 列目が空列の場合でもその点字の 2 列目に 000 がパディングされ 6 ビット単位の出力が得られる．

5　評価

実際に web カメラで撮影した画像を入力として本アルゴリズムの評価を行う．

5.1　素直に検出できる場合 (その 1)

図 24，図 25 はそれぞれ図 3，図 5 と同じものである．図 24 の画像サイズは 612 × 75 ピクセルである．空白の点まで含めると次のような並びになる (実際の画像では 1 行である)．



5 文字目の長音府 (ー) と 6 文字目の濁音府の間がケース C(その 2)(図 17) に，9 文字目の読点 (，) と 10 文字目の空白の間がケース E(図 19) に該当している．

ラベリングと重心計算で得られた点の座標を本アルゴリズムに入力して得られた点の並びは

101101, 110010, 001011, 101010, 010010, 000010,
111110, 001000, 000011, 000000, 111110, 001011,
000010, 110011, 001000

である．右端の点字の右の列が画像から欠けている場合や，途中に空白文字 (000000) がある場合も正確に認識できている．

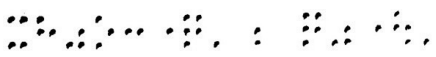


図 24　元画像 A

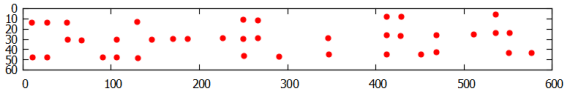
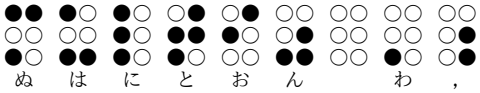


図 25　元画像 A からラベリング + 重心計算により検出された点の場所

* ケース E が 1 文字目で発生した場合には図 19 の状況であることはユニークに決まる．

5.2 素直に検出できる場合 (その 2)

別の例を図 26 に示す．画像サイズは 743 × 118 ピクセルである．空白の点まで含めると次のような並びになる．



ラベリングと重心計算で得られた座標は図 27 のようになる．これを本アルゴリズムに入力して得られた点の並びは

101100, 101001, 111000, 011110, 010100,
001011, 000000, 001000, 000011

である．こちらも正確に認識できている．

図 26　元画像 B

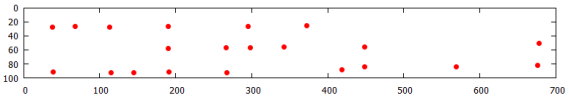
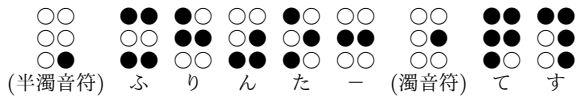


図 27　元画像 B からラベリング + 重心計算により検出された点の場所

5.3 左端 1 列に点がない場合

図 28 は左端の文字の左 1 列に点がない例である．画像サイズは 717 × 132 ピクセルである．空白の点まで含めると次のような並びになる．



ラベリングと重心計算で得られた座標は図 29 のようになる．これを本アルゴリズムに入力して得られた点の並びは

000001, 101101, 110010, 001011, 101010,
010010, 000010, 111110, 100111

である．左端の文字の左 1 列に点がない場合でもその他の点の間隔から 1 列目に点がないことを正確に認識できている．

図 28　元画像 C

5.4 点字フォントを入力画像として用いた場合

別の例を図 30 に示す．画像サイズは 336 × 24 ピクセルである．これは点字フォント [8] を使って作成したものである．点字フォントを使用して作成した画像なので点の重心座標に揺れがなく，画像サイズもほかのものと比べて小さめである．

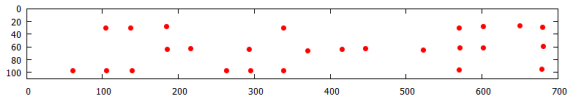
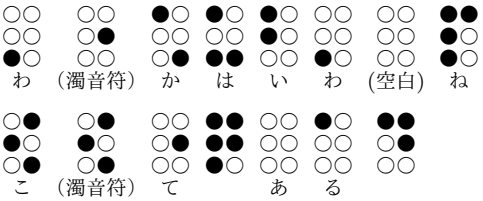


図 29　元画像 C からラベリング + 重心計算により検出された点の場所

また，点字フォントが JIS 規格に沿ってデザインされているわけではないので点間距離が実際の点字と異なり，$d_2/d_1 = 2.0$ となっている．空白の点まで含めると次のような並びになる．



ラベリングと重心計算で得られた座標は図 31 のようになる．これを本アルゴリズムに入力して得られた点の並びは

001000, 000010, 100001, 101001, 110000,
001000, 000000, 111100, 010101, 000010,
111110, 000000, 100000, 100110

である．こちらも正確に認識できている．

図 30　元画像 D

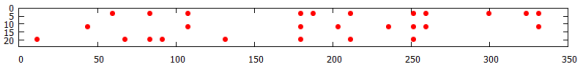


図 31　元画像 D からラベリング + 重心計算により検出された点の場所

5.5 d_2 が直接決められない場合への対応

4.7 章の対応についての評価を行う．図 15 を再現した画像が図 32 である．これは d_2 を直接決められないので 4.7 章の方針で d_1 を 1.533 倍して d_2 を推定する．

図 32　元画像 E1

得られた $d_1 = 44.64$ で，d_1 から推定した $d_2 = 68.43$ である．これを使って得られた点の並びは

000011, 000000, 111000, 111111

である．正確に認識ができている．

図 32 の右端に 1 つ点を付け加えた画像を生成し，d_2 を決定できるようにしたものが図 33 である．この場合，d_2 を直

接決定することができて，$d_1 = 44.64$ と $d_2 = 69.94$ が得られる．これを使って得られた点の並びは

000011, 000000, 111000, 111111, 100000

である．こちらは通常通りの動作であるが，正確に認識ができている．

図 33　元画像 E2

5.6　入力画像の解像度への依存性

ここでは入力画像の解像度に検出精度が依存するかを検証する．図 34, 35, 36, 37, 38 は，612 × 75 ピクセルの現画像 (図 3) をそれぞれ，75%, 60%, 45%, 30%, 15% に縮小したものである．これらを入力画像としてラベリング + 重心計算を行い，4 章で述べた方法で点字を認識させた結果，図 34, 35, 36, 37 では次のように図 5 と同一の認識結果が得られた．

101101, 110010, 001011, 101010, 010010,
000010, 111110, 001000, 000011, 000000,
111110, 001011, 000010, 110011, 001000

それぞれの入力での d_1 や d_2 は表 1 に示した．

図 38 のみ，認識結果が次のように誤った結果となった．点の並びはもとより，点字の数も誤認識されている．

101010, 011101, 010010, 010101, 110001,
000000, 010101, 011010, 110011, 001000

あまり小さな画像を入力として用いると，規格化したときに点と点が引き離されてしまい，結果として縦横の $\pm 3\sigma$ の範囲からほとんどの点がはずれて孤立した点と見なされてしまう可能性がある．このことが図 38 で正確に点字の検出ができていない原因の一つとして考えられる．

表 1　入力画像サイズへの依存性

画像	画像サイズ [px]	d_1	d_2	結果
元画像	612 × 75	38.27	55.97	○
元画像 F1	457 × 56	38.51	56.44	○
元画像 F2	368 × 45	38.53	56.88	○
元画像 F3	278 × 34	38.27	55.33	○
元画像 F4	180 × 22	38.31	55.66	○
元画像 F5	91 × 11	38.40	51.59	×

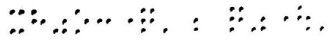


図 34　元画像 F1 (75% = 457 × 56px)

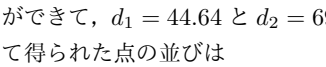


図 35　元画像 F2 (60% = 368 × 45px)

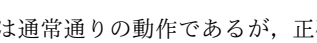


図 36　元画像 F3 (45% = 278 × 34px)

図 37　元画像 F4 (30% = 180 × 22px)

図 38　元画像 F5 (15% = 91 × 11px)

6　まとめ

感圧センサやカメラからの画像として点字の模様が入力されたときに点字の 1 文字ずつを切り出すアルゴリズムについて述べた．統計的な手法を用いることで，空列を挟む場合でもほぼ正確に点字の 1 文字ずつを切り出すことができている．また，既存の手法では入力画像のサイズに強い制約があったが，本アルゴリズムでは最初に座標の規格化を行っているため入力データのサイズに依らずに安定した出力を得ることが可能となっている．今後は本アルゴリズムで現時点では対応できていない限定的なパターンへの対応を行い，より高精度な認識結果が得られるようにする．

参考文献

[1] JIS T 0923 高齢者・障害者配慮設計指針—点字の表示原則及び点字表示方法—消費生活製品の操作部: 財団法人日本規格協会, 平成 21 年 3 月 20 日.

[2] ウェアラブル点字認識センサシステムの開発: 宮田薫, 田中真美, 西澤達夫, 長南征二, 日本機械学会 [No.05-6] IIP2005 情報・知能・精密機器部門講演会講演論文集, 2005.

[3] 光学的読み取り装置による点字認識: 宮本修, 長岡英司, 大武伸之, 電子情報通信学会技術研究報告. ET, 教育工学 108(470), 89-92, 2009-02-28.

[4] ディジタルカメラによる手すりの点字認識: 宮本修, 電子情報通信学会論文誌 2010/10 Vol.J93-D No.10, pp.2281-2291.

[5] Braille Patterns: http://www.unicode.org/charts/PDF/U2800.pdf

[6] 画像を入力とした点字の認識アルゴリズム: 伊藤祥一, 藤澤義範, 2017 年電子情報通信学会総合大会, H-4-11, 基礎・境界/NOLTA 講演論文集, p.277, 2017.

[7] ラベリング処理による点字の認識: 伊藤祥一, 藤澤義範, 越渓拓, 第 15 回情報科学技術フォーラム講演論文集, K-051, pp.557-558, 2016.

[8] 社会福祉法人日本ライトハウス 墨点字フォント: http://www.lighthouse.or.jp/tecti/tecti/br-font.html