アクティブ帳票用ドットテクスチャのコーディングとデコーディング

朱 碧蘭　　中川 正樹
東京農工大学情報コミュニケーション工学科 〒184－8588 東京都小金井市中町 2－24－16
E-mail: zhubilan@hands.ei.tuat.ac.jp

あらまし 本報告は，アクティブ帳票のためのドットテクスチャに情報を重畳するためのコーディング法と情報を読み出すデコーディング方法について述べる．本方式では，記入枠に微小なドットの集まりからなるドットテクスチャを用いることによって，手書きが記入枠に重なっても容易に分離することができる．また，記入枠に記入される手書きの属性，文字種類，見出し，指示情報をそのドットテクスチャに重畳することによって，属性と文字種類情報に従い切り出した手書き文字に対して認識率を高めることができ，さらに，その処理を帳票側で指示することができる．したがって，読取り装置は汎用機となり，多種の帳票を入力しても，それぞれの帳票に応じた処理が帳票主導で実行される．本報告では，種々のコード形状，そして，そのコーディングとデコーディング方法について比較し検討する．そして，記入枠と手書きの分離方法，記入枠と記入枠構造の検知方法についても述べる．

Information encoding and decoding in dot texture for active forms

Bilan ZHU　　Masaki NAKAGAWA
Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho, Koganei-shi, Tokyo, 184-8588 Japan
E-mail: zhubilan@hands.ei.tuat.ac.jp

Abstract This paper describes information encoding and decoding methods applied to dot texture for active forms. We employ dot texture made of tiny dots and looking as gray color to print various forms. It eases the separation of handwritings from their input frames even under monochrome printing/reading environments as well as makes the forms dictate how to process filled-in handwritings according to the information embedded in the dot texture. Due to the embedded information, the recognition rate of handwriting is improved, and the form processing is directed by the form itself rather than form reading machines. Thus, the form reading machine can be a general-purpose machine and various forms inputted into a single reader can be processed distinctly as specified by each form. This paper compares various dot shapes and information encoding/decoding methods for those shapes. Then, it presents how to locate input frames, separate handwritings from input frames.

1. はじめに

我々の日常生活では，様々な紙の帳票が使用されている．近年では，電子的な帳票も普及してきているが，紙帳票はいつでもどこでもペンだけで書けるし，また証拠として残るなどの利点があるので，依然として広範囲に利用されている．

光学帳票読取り装置や紙上の筆記から電子インク（時系列筆点情報）を読み取る e-pen [1]，anoto pen [2] などの装置が人々の注目を浴びているが，e-pen と anoto pen は特別なデバイスが必要である．

帳票の記入枠から手書きを抽出するために，記入枠にドロップアウトカラーがよく用いられている．しかし，カラーの使用はコストを高めるし，現在広く使われているモノクロコピー，モノクロファックスが利用できなくなる．一方，モノクロの光学読取り用帳票は安価で，普及してきているが，記入枠と手書きの重なりを分離するのが困難である．この問題を解決するためにいろいろな方法が試されてきたが[3~6]，全ての場合でうまく機能する訳ではない．

我々は，帳票の記入枠に微小なドットで構成されたドットテクスチャを用いることを提案する[7~9]．ドットと手書きの面積差を利用して，手書きが記入枠に重なっても容易に分離することができ，また，記入枠に記入される手書きの属性，文字種類，見出し，指示情報をそのドットテクスチャに重畳することによって，手書き記入に対して認識率を高めることができ，かつ，記入内容の処理を帳票側で指示できることになる．帳票の処理方法を指示するのは，帳票読取り装置に内蔵されたプログラムではなく，帳票自身である．したがって，帳票は受身ではなく能動（アクティブ）な媒体と見なすことができる．その結果，多種の帳票を入力しても，それぞれの帳票に応じた処理が帳票主導で実行される．つまり，読取り装置は汎用機となる．これは，事前に標準帳票を登録し入力帳票と標準帳票をマッチングすることにより帳票を特定する従来の OCR 帳票処理読取り装置[3, 4, 6, 10~13]とは異なる方式である．これらの装置では，いくつかのシステムにおいては手動の操作まで必要としている[4, 11]．

最近のレーザープリンタは高精度であり，ドットテクスチャを印刷するのに問題はない．ファックスの精度もますます高くなってきている．

我々は，一般の文書を対象に，元原稿をドットテクスチャで印刷し，そこに手書きされた校正記号を分離して実行する方式を文献[14]で報じた．微小のドットに情報を重畳する研究には文献[15,16]などがある．本方式は，これらと共通の考え方に基づいているが，記入枠にドットテクスチャを使用することにより，記入枠と手書きを容易に分離でき，かつ，情報のコーディングに余分なエリアを必要としないことを特徴とする．

我々は先にドットテクスチャに簡単の大小 2 種類のコードを試した[7, 8]．その結果，より効率的なコーディングとデコーディング方法の必要性が認識された．そこで，ドット形状とコーディング方法の改良，これに合わせたデコーディング方法の提案とノイズ対策を検討した．

本報告は，これらを反映して，再構築した方式全体を提示する．以下，2 章ではアクティブ帳票とその処理システムの流れを提示する．3 章ではコーディングの設計について，4 章では記入枠と手書きの分離方法について，5 章では記入枠と記入枠構造の検出方法について，6 章では情報のデコーディングについて，それぞれ述べる．そして，7 章はこれらの処理性能に関する予備的評価を報告する．最後に，8 章において結論を示す．

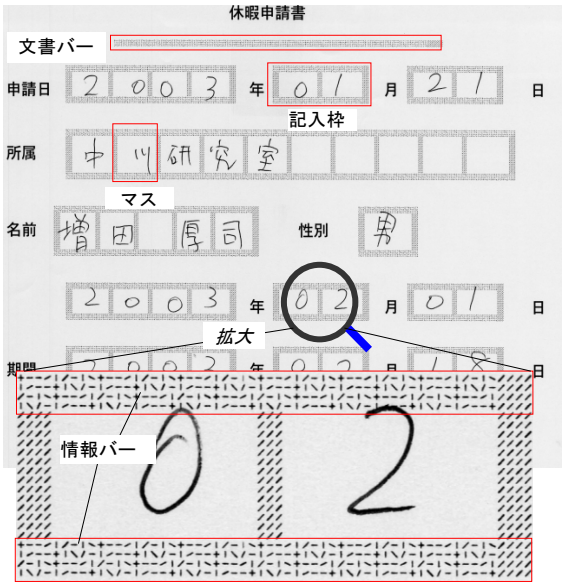


図 1. アクティブ帳票.

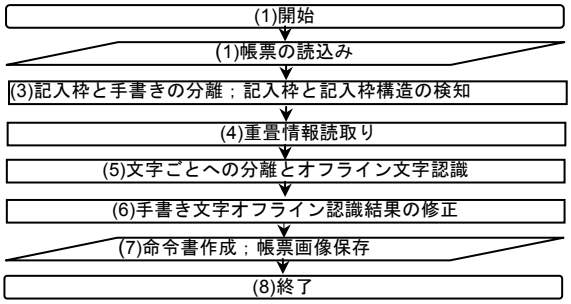


図 2. アクティブ帳票処理システム流れ図.

2. アクティブ帳票とその処理プロセス

アクティブ帳票の例を*図 1* に示す．帳票には 2 次元的に配置された微小なドットからなるドットテクスチャにより印刷されたいくつかの記入枠がある．記入枠のドットテクスチャには，記入される手書きの属性（人名，住所，メールアドレス，自由筆記など），文字種類（数字，漢字，英文など），見出しやその処理方法（手書き文字を認識する，記載内容をメールで送信する，あるいはデータベースに保存するなど）を指示する情報を記入枠上下側（情報バーと呼ぶ）に重畳する．重畳した属性と文字種類に従い，認識のカテゴリを限定することにより，手書きの認識率を高めることができる．また，帳票の全体に対する情報を帳票標題の下などに印刷される横線のドットテクスチャ(文書バーと呼ぶ)に重畳する．情報バーと文書バーに対して縦方向に同じ個数のドットを配置する．記入枠の中で文字を一文字ごとに区切って記入してもらうための分割をマスと呼ぶ．本方式においては帳票作成エディタを作成することにより，誰でも簡単に帳票をデザインすることができる．

アクティブ帳票処理システムはアクティブ帳票から記載内容や重畳情報を抽出し命令書に出力するシステムである．アクティブ帳票処理システムの流れ図を*図 2* に示す．

3. コーディングのデザイン

3.1. ドット形状

ドットテクスチャに情報を埋め込むためにいくつかのコード形状を考えられる．

(1) 大小 2 種類の図形を用いるコード

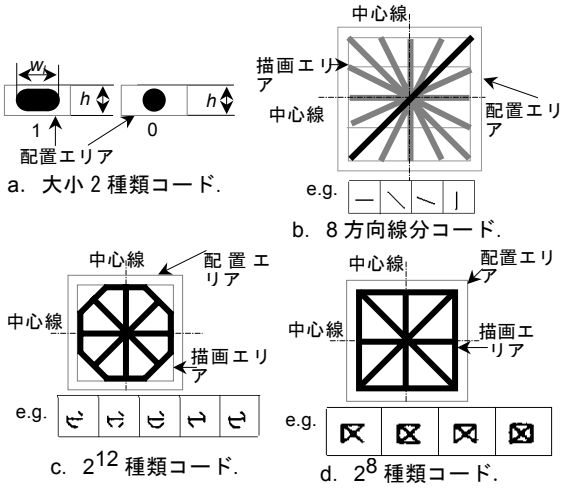


図 3. 情報重畳のためのコード形状.

大きさの違う 2 種類のドットでバイナリビットの 1 と 0 を表し，これに情報を重畳する．*図 3a* に示す．2 つコードの高さは等しく h とする．大きいコードの幅を w_1 とする．2 つコードとも，コードを中心にしてその周りの余白を持つ同じ大きさの配置エリアを持つ．各コードの配置エリアを碁盤の目のように縦横に隙間を空けずに配置すれば，コードとしては隙間を確保したドットテクスチャになる．他の種別のコードも同じで，配置エリアの中心に置いて並べることにより，記入枠のドットテクスチャを生成する．

(2) 8 方向線分を用いるコード

8 方向の線分により，0 から 7 までの*図 3b* に示す 8 種類のコードが表される．コードを描画するための同じ大きさの正方形領域を描画エリアとする．描画エリアは配置アリアの中心に位置する．

(3) 2^{12} 種類の図形を用いるコード

図 3c に示す 12 種類の線分の組合せからなる．大きさ a×a の描画エリアを基準とし，これら 12 種類の線分の有無により 2^{12}=4096 種類のコードを作る．このコードの配置エリアと描画エリアの配置は前のコードと同じである．

(4) 2^8 種類の図形を用いるコード

図 3d に示す 8 種類の線分の組合せからなる．大きさ a×a の描画エリアを基準とし，これら 8 種類の線分の有無により 2^8=256 種類のコードを作る．このコードの配置エリアと描画エリアの配置は前のコードと同じである．

3.2. 重畳情報

属性，文字種類，命令，見出しなどの情報を記入枠の中に重畳する．属性は，その記入枠の中に記入される手書きの属性で，我々の大学で使用されている帳票の見出し文字を調査し，14 種類に分類した．文字種とは，その記入枠の中に記入された手書き文字の字種のことである（数字，漢字，英文など）．文字種類に従い手書き文字の認識のカテゴリを限定することができる．命令とは，その記入枠の中に記入された手書きに対する読取り後の処理方法のことである．記入枠の見出し文字は，最長で 10 文字までの Shift-JIS コードで表す．この列を情報レコードと呼ぶ．

属　性 (9bit)	文字種類 (9bit)	命　令 (9bit)	見出し文字(全角1～10文字，図形6～60個の見出しのShift-JISコード)

図 4. 重畳する情報レコード.

図 5. 8 方向線分コードの重畳方式.

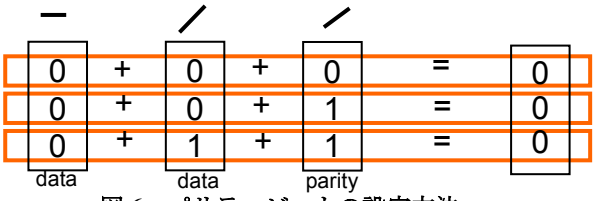


図 6. パリティビットの設定方法.

図 4 に示す情報レコードをコードシステムのサイズによりセクションに分割し，一つのセクションをこのコードシステムの一つのコードで表す．8 方向線分コードシステムを使用するとすると，情報レコードは 3bit ずつのセクションに分割し，一つのセクションを 8 方向線分コードの一つで表す．最後に，セクションごとの対応する位のビッドの値を偶数になるように，パリティ符号を付加する．パリティ符号は使用するコードシステムの一つのコードにより表す．8 方向線分コードシステムのパリティ符号の設定方法を*図 6* に示す．そして，情報レコードの始めと終りには“+“の形状の区切り記号を付ける．

情報レコードは縦方向の上から下へ，そして，横方向の左から右へドットテクスチャに配置する．たとえば，8 方向線分のコードシステムの 4 行からなるドットテクスチャを考えると，列ごとに 4 つのコードを上から下に配置し，それを左から右へコーディングしていく（*図 5* の 1，2，3，4，5 の方向でコードを配置している）． そして，可能な限り，2 番目，3 番目のコピーを繰り返す．さらに，情報レコードを記入枠の上側と下側に重畳する．

情報レコードが手書きの重なりやノイズなどにより破損した場合，パリティでまずエラー混入があるかどうかを検査し，さらに，複数の情報レコードの多数決をとって，損失した情報を復元する．

ドットテクスチャの微小なドット形状を保証するため，PostScript 言語で帳票を作り，PostScript プリンタで帳票を印刷する．

3.3. コード図形の認識方法

コードを認識するには，以下のいくつかの方法が考えられる．

(1) 四方向への射影ヒストグラムによる認識方法

図 7a に示すように四方向への射影ヒストグラムをとることによって，その強度分布からコードを識別することができる．

(2) マッチングによる認識方法

図 7b に示すように，普通の統計パタンマッチング方法を使用することにより，コードの形状を認識する．

十分なサンプルパタンを用意し辞書を学習させれば良い, 認識エンジンには, 文字認識エンジンを利用する.

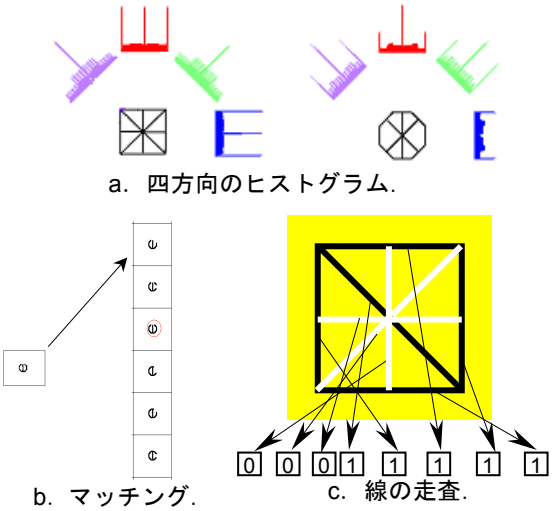
(3) 部分的な走査による認識方法

部分的にコード画像の線分を走査することによるコード図形を認識する．コード図形中の線分があるべき位置ごとに順番に走査し，線分があれば対応の情報ビットを 1 とし，線分がなければ 0 とする．2^8 種類コードを例として，この方法を*図 7c* に示す．白い部分は線分のない部分, 黒い部分は線分のある部分とする．各線はバイナリコードの桁に対応する．これらの線分を順番に走査し，線分があれば対応の桁を 1 とし，線分がなければ 0 とする．こうすることによって，バイナリコード 00011111 が求まる．

3.4. 可能性実験

実験用 PC は CPU Pentium(R)4 の 2.26GHz，メモリ 1.00GB（OS Microsoft Windows XP）である．プリント精度は安価なレーザビームプリンタでも印刷できる 1200dpi を採用した．実験の結果，この精度でプリントできる丸いドットの最小直径と線の太さは 0.1mm である．スキャンの精度は普通のスキャナで達成できる 600dpi である．ドットテクスチャの小さいドットがスキャンにより連結しないため，ドット同士の間に適切な間隔をあける必要がある．プリント，及び帳票が傾いた状態でのスキャンや二値化等による誤差を含め，実験により 600dpi のスキャンでは 0.2mm のドット間の最小間隔が必要である（このサイズは，プリント精度 1200dpi で印刷可能である）．この間隔は帳票の見た目を損なわない．

各コードシステムについて，一枚の帳票においてのコードの抽出と認識のスピード，及びコードの情報量を比較し評価した．各コードシステムにおいては，最適な認識方法を使用した．*図 3a* に示す大小 2 種類コードの高さ h，及び 8 方向線分コードと 2^{12} 種類，2^8 種類コードの線の太さは，プリントできる最小サイズである 0.1mm とした．結果を*表 1* に示す．各コードの描画エリアの大きさは，各コードシステムの最適な認識エンジンがある程度高い認識率でコードを認識できる限界のサイズに設定した．コード間最小間隔は 0.2mm とする．1 ビットあたりのデコード時間は，一定量のコードを抽出して認識する時間を，コード量とコードあたりの情報量で割った時間である．



a. 四方向のヒストグラム.
b. マッチング.
c. 線の走査.

図 7. コード形状の認識方法.

表 1. 各コーディングシステムの比較.

性能＼コード	大小 2 種類	8 方向線分	2^{12} 種類	2^8 種類
最適な認識方法	ヒストグラム	ヒストグラム	部分的走査	部分的走査
描画エリアサイズ (mm^2)	0.025	0.16	0.81	0.6084
1 cm^2 あたりの情報量(bit)	740.74	833.33	991.74	832.99
1 ビットあたりのデコード時間(μs)	5.469	2.376	2.025	2.820
抽出と認識	easy	easy	difficult	difficult

3.5. 各コードシステムについての結論

2^{12} 種類と 2^8 種類のコードは，現実的な大きさではコードの切出しと認識に課題が残る．*3.4 節*で使用したサイズでも，コードはドットテクスチャではなくシンボルのように見えてしまう．さらに，後の節で述べる記入枠と手書きの分離方法のうち最高速であるラベリングの方法により，この二つのコードシステムにより作った帳票を処理すると，小さい文字（句読点など）と分離できなくなる．文字と重なった場合，コードが手書きに付いてしまい，大きいコードは手書き文字の認識に対して大きいノイズになる．他の方法で記入枠と手書きを分離することも考えられるが，分離の処理時間が長くなる．

8 方向線分コードは大小 2 種類コードより情報量とデコード速度ともに良い．我々は，前のプロトタイプに大小 2 種類コードシステムを使用したが，将来の発展として 8 方向線分コードを選んだ．

4. 記入枠と手書きの分離方法

記入枠と手書きを分離するには，いろいろな方法がある．検討した結果，連結している黒画素の連結成分に同じラベルをつけるラベリングによる分離方法が，一番高速で, かつ, 手書きの太さに制限がかからない．したがって，ラベリングの方法を採択した．帳票画像をラベリングし，閾値より大きさの小さい連結成分を削除することにより，手書きを記入枠から分離する．記入枠と手書きが重なった場合に対する分離処理例を*図 8* に示す．記入枠と手書きが重なっていても，ドットテクスチャが手書きを抽出するには有効であることが確認できる．

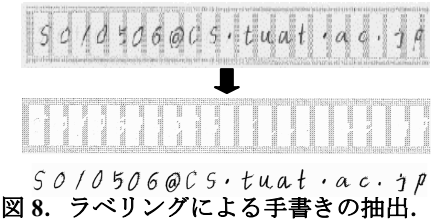


図 8. ラベリングによる手書きの抽出.

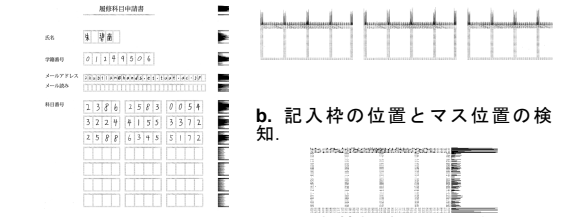
5. 記入枠と記入枠構造の検知

5.1. ヒストグラムでの検出方法

図 9 に示すとおり，横方向と縦方向にヒストグラムを取る方法により記入枠の位置と記入枠の構造を検出する．この方法では帳票画像が多少傾いてスキャンされても，スムーズに記入枠と記入枠構造を検出することができる．

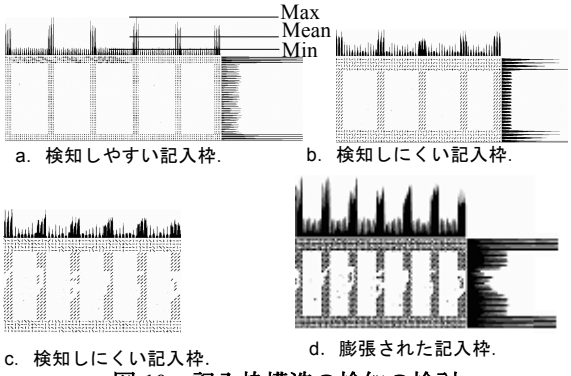
5.2. 記入枠構造検知方法についての検討

図 10a に示すように，記入枠の高さと幅が十分に大きいとき，縦方向と横方向へのヒストグラムにより，記入枠の構造がはっきり区別できる．ヒストグラムの最大値と最小値の中間値を求め，この中間値を閾値とし，記入枠のマス位置や情報バーの位置を検出することができる．しかし，*図 10b* に示すように，記入枠の幅や高さが小さい場合，中間値を安定に求めることができない．さらに，*図 10c* に示すような手書きとの重なりにより消された記入枠は，そのままの状態で処理すると誤った結果になることがある．このことから，記入枠の構造を識別する時，まず，*図 10d* に示すように記入枠のドットを膨張する．記入枠のドットが完全に接触するまで膨張するのは時間がかかるため，ドットのサイズに従いある程度膨張すればよい．膨張した記入枠に対して，ヒストグラムの中間値を求め，簡易に記入枠の構造を識別することができる．



a. 一行ずつの記入枠の検知.　b. 記入枠の位置とマス位置の検知.　c. 記入枠情報バー位置の検知.

図 9. 縦横方向へのヒストグラム.



a. 検知しやすい記入枠.　b. 検知しにくい記入枠.　c. 検知しにくい記入枠.　d. 膨張された記入枠.

図 10. 記入枠構造の検知の検討.

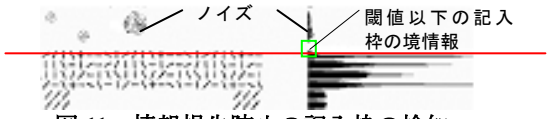


図 11. 情報損失防止の記入枠の検知.

5.3. 空白の処理時間を省く方法

帳票内の記入枠の間には空白があるので，ラベリングでの空白の処理時間を省けば帳票の処理時間が早くなる．そして，行内の位置検出には手書きが邪魔になるが，行の検出には邪魔にならないことに注意されたい．したがって，手書きを分離する前に行検出を行い，一行ずつの画像だけに対してラベリングをすれば，行間でのラベリング処理が省ける．処理手順は次の通りである．

(1) 全帳票画像に対し横方向へのヒストグラムをとり，行ごとの画像を検出する．
(2) 行ごと画像に対しラベリング処理を適用し，記入枠と手書きを分離する．
(3) 行ごとの記入枠画像に対し縦方向へのヒストグラムで記入枠の位置とマス位置を検出する．
(4) 記入枠ごとの画像に対し横方向へのヒストグラムで記入枠の情報バーの位置を検出する．

5.4. 記入枠検出時の情報損失防止方法

ヒストグラムにより記入枠の位置を検出するとき，ノイズの妨げを避けるために，ヒストグラムの結果に対してある敷値以下の値を 0 と見なして，ノイズのある場所でも空白として無視する．例を*図 11* に示す．記入枠の上外側にいくつかのノイズがある．こうすることにより，ノイズの影響を受けず，記入枠の位置 f を検出できる．しかし，記入枠の境のところで閾値以下の情報も無視されてしまう．したがって，記入枠の位置を検出する時に，このような情報損失を防止するために，ヒストグラムで閾値以下のところで記入枠の位置 f を検出してから，さらに，この位置 f から記入枠の外側の方向にヒストグラムの値を辿って行き，ヒストグラムの値が 0 より大きい値から 0 になったか，始めの位置 f から辿った長さがある閾値になっていれば，辿り着いた位置を記入枠の位置とする．

6. デコーディング

6.1. コードサイズとコード行数の検出

8 方向線分コードシステムにおいては，描画エリア (L_d x L_d)は 0.4×0.4mm にセットされる．描画エリアの L_d が 0.35mm，0.45mm，0.5mm，0.6mm でも，デコーディングが可能である．コードの大きさは大きければ大きいほど認識率が高いが，小さければ小さいほど帳票の見た目がよい．帳票エディタでは，ドット同士間

の必要な間隔 0.2mm をあけ，任意のコードサイズで帳票を作れるようにしたので，配置エリア(L_a x L_a) は(L_d+0.2 x L_d+0.2)になる．デコーディングでは，自動的に文書バーからコードの大きさを検出しこのコードの大きさを以下のコードに適応する．そして，文書バーのコード行数を検出することで帳票全体の情報バーのコード行数が分かる．

図 12 に示すように，文書バーに対して縦方向のヒストグラムをとり，各コード図形列の中心位置を検知する．次に，各コード図形列の中心間の距離を求め，これらの距離に対して平均を求める．この結果はコード図形の配置エリア正方形の辺長である．配置エリアの辺長からドット間隔のサイズ 0.2mm（600dpi スキャンで 4 画素）を引くと，描画エリア正方形の辺長が求まる．

図 13 に情報バーのコード行数の検出方法を示す．帳票に傾きがあると，ヒストグラムのピークがぼやけるので，限られた列数のコード図形だけに対して，横方向のヒストグラムを取り，記入枠の情報バーのコード行数を検出する．

6.2. コード図形の切出し

6.2.1. ヒストグラムによる切出し方法

コードの大きさが検出できれば，コードの大きさを一定長としてコードを切り出す方法が考えられる．しかし，*図 14* に示すように，スキャナとプリンタのスキャンや紙送りのズレなどにより，コード列の本当の位置と仮定の位置が情報バーの左のエッジから右へ行くほど大きくなる．このことから，ヒストグラムによる切出し方法を行う必要がある．

6.2.2. コード図形の切出し方向

情報バーに 2 次元的に配置されたコードは行から列，あるいは列から行への方向で抽出することができる．以下のような効率的な切出し方向が考えられる．

(1) 情報バーの画像に対して横方向のヒストグラムにとり，行ごとを切り出す．

(2) 行ごとの画像に対して縦方向のヒストグラムをとり，コードごとの図形を切り出す．

図 15a に示すように，傾きのない情報バーの場合，この切出し方向で容易にコード図形を切り出すことができる．しかし，*図 15b* に示すように，傾いてスキャンされた情報バーの場合，この切出し方向では容易にコード図形を切り出すことができない．しかし，このような場合でも，列数にくらべ行数が少なく縦方向へ

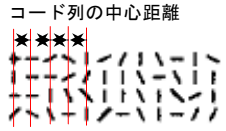


図 12．コード列間の中心距離．

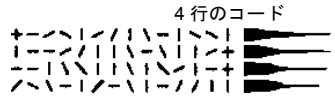
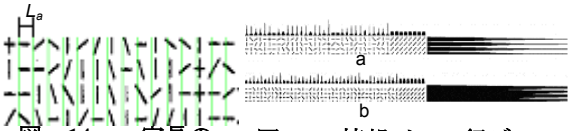


図 13．情報の何行かコードのヒストグラム．



図．14．一定長の切出し．

図 15．情報バーの行ごとのコード抽出．

のヒストグラムの重なりが小さいことは重要である．この性質を利用して，情報バーが多少傾いていても正しくコードごとの図形を切り出せるために，以下のような切出し方向をとる．

(1) 情報バーの画像に対して縦方向のヒストグラムにとり，情報バーの左から右へ列ごとを切り出す．

(2) 列ごとの画像に対して横方向のヒストグラムをとり，コードごとの図形を切り出す．

6.2.3. 手書き重なりエラーの検出と処理

手書きと記入枠の重なりにより，分離した記入枠画像では手書きと重なる部分が消される．これを効率よく検出し，エラーのある情報レコードに対して，処理を行わない方法をとる必要がある．情報バーの画像において，左から右へ列ごとのコード図形を切り出していき，前の列と次の列の間隔が想定される幅より大きいときにコード図形と手書きが重なって手書きの分離処理によりコード図形が欠落したと判断する．

図 16 を例として，この方法を示す．*図 16* は記入枠情報バーの最左の部分であり，仮想的に小さい正方形でコードの描画エリアを表す．描画エリア正方形の辺長 L_d は *6.1 節*の方法により求められる．今後，述べる“位置”は水平方向の位置を意味する．列の中心位置 S+描画エリア辺長 L_d / 2 の位置は列の最大右位置 R_2 とし，列の実際の右位置を R_1 とする．

基準位置 P_a の初期位置は情報バーの最左の位置とする．列ごとのコード図形を切り出していく度に，基準位置 P_a を R_2 に更新する．

新しいコード列を切り出すごとに，この列の右位置 R_1 と P_a の距離 d を求める．d が L_d より 6 画素（描画エリアの間隔 4 画素+マージンとしての 2 画素）以上大きい場合，手書きの重なりエラーと見なして，この後のコード図形に対して，区切り記号が見つかるまで無視する．

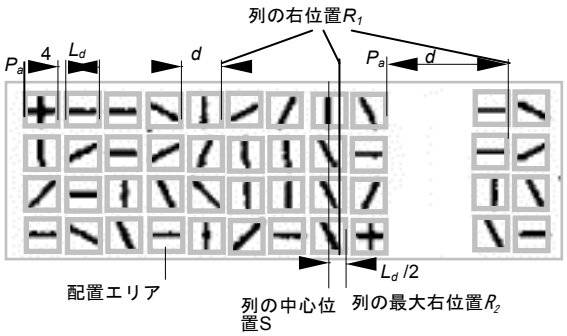


図 16．手書き重なりエラーの検出．

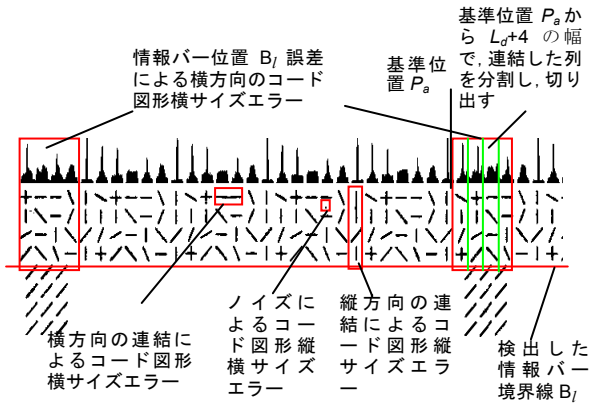


図 17. コード図形サイズの各種エラー.

6.2.4. コード図形におけるエラー

図 17 に示すように，ノイズによってコード図形が接触し，形や大きさが異常になる場合がある．切り出した図形の大きさが想定されるコード図形より小さい場合はノイズとして無視し，大きすぎる場合には，想定される大きさで分離する．列ごとのコード図形を切り出す時，列の幅をチェックし，もし列の幅が描画エリア幅（たとえば，L_d+4）より大きければ，コード図形の横サイズエラーと判断し，基準位置 P_a （*6.2.3* 節で説明）から L_d+4 の幅でこの列を分割して切り出す．また，コード図形の長さか幅が描画エリア幅 L_d+4 より大きければ，コード図形のサイズエラーと判断し，この図形の最上か最左から L_d+4 の幅で分割して切り出す．

6.2.5. 情報バーの位置検出誤差によるコード図形損傷への対応

記入枠の情報バーの位置を検出するとき，*図 18* に示すように，検出した境界線 B_l は正しい情報バーの境界線と幾分ずれる場合がある．この情報バー境界線に従い，コード図形を切り出してデコーディングを行うと，情報バー境界線 B_l からはみ出したコード図形の一部を除外して認識することになり，誤認識を招く危険がある．このことを避けるために，以下の方法を考えた．

(1) *図 19* に示すように，検出した情報バー境界線に従い，情報バーの画像に対して，縦方向のヒストグラムにより，左から右へコードの列ごとの位置を検出する．

(2) *図 20* に示すように，列ごとのコード画像に対し，横方向のヒストグラムを走査し，コードごとを切り出す．*図 18* に示すように，記入枠の上側情報バーの切出し方向は上から下へ，下側情報バーの切出し方向は下から上へである．この段階以降は，境界線 B_l の情報を利用しない．

(3) 一つのコードを切り出した後，この列において切り出したコード数と走査した長さ L_s（*図 20*）を調べる．*6.1 節*で求めた情報バーのコード行数を N とする．もし，この列のコード数がすでに N に達しているか，あるいは，走査した長さ L_s がある長さまで達していれば，コードの切出しを終了する．ここで，検出された境界線の位置は使用しない．また，走査した長さで終了を検出する目的は，コード図形が手書きと重なったりして欠落した場合に，情報バーの領域を越えてコードを探しに行かないようにするためである．

(4) 列ごとに対して，コードの切り出しが終了した時点で，そのコード数をチェックする．もし N に達してなければ，コード数のエラーと見なし，この後のコード図形に対して，区切り記号が見つかるまで無視する．

コード図形の切出し段階において境界線 B_l を用いないことで，検出した情報バー境界線からはみ出したコード図形を正しく認識することができる．

6.3. コード図形の認識

コードが抽出されたら，コード図形に対して四方向のヒストグラムをとる．*図 21* に示すように，四方向へのヒストグラムの幅を w_0， w_1， w_2， w_3 とする．簡単な決定木により，8 方向線分コードを識別することができる．

6.4. 情報レコード長エラーの検出と処理

情報レコードの長さは*図 4* に示すように 16 から 70 までの間にある．情報レコードの長さがこの範囲にない場合，これらの情報レコードを削除する．

このようにして，一つの情報レコードの複数のコピーが求まるが，これらのコピーの長さがすべて同じでない場合は多数決をとり最多のレコード長を正しい長さとし，それらに対して次のパリティチェックを行う．

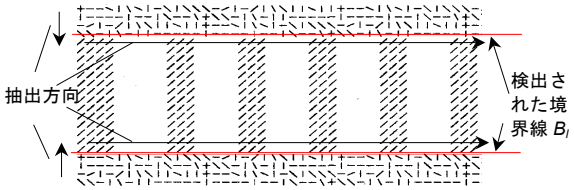


図 18. 情報バーの境界線検出エラーとコードの抽出方向.

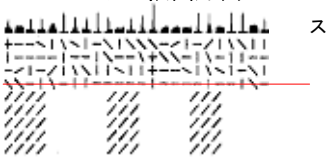


図 19. 水平方向のヒストグラム.

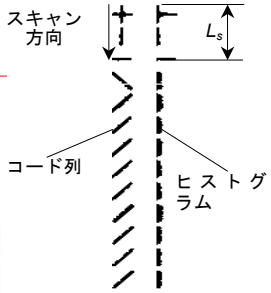


図 20. コードごとの切出し.

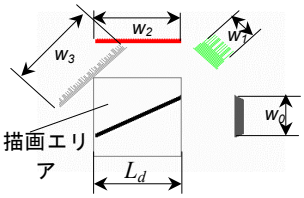


図 21. 8 方向線分コードの四方向ヒストグラム.

6.5. 信頼性の向上

情報レコードごとに，長さの正しい全てコピーに対して，パリティ符号のチェックを行う．パリティ符号は，*図 6* に示すように付加されている．そして，パリティチェックで認可されたコピーから多数決を取って，情報レコードを復元する．

7. 評価実験

A4 判大の記入枠数が異なる 2 種類の帳票を 1200dpi でプリントし，10 人に 1 枚ずつ書いてもらい，全部で 20 枚の帳票サンプルを収集した．これら帳票を 600dpi, 8 ビットグレーレベルでスキャンし，アクティブ帳票処理システムの予備評価を行った．評価用 PC は CPU Pentium(R)4 の 2.26GHz，1.00GB メモリである．

実験結果を *表 2* に示す．平均処理時間は帳票を認識し修正インタフェースを呼び出すまでの平均処理時間である．帳票サンプルの中で，書き間違った文字（20 枚の帳票サンプル中で 2 文字）は評価の中に入れない．

表 2 に示すように，処理時間は事務のドキュメント処理にとっては現実的である．重畳した情報の抽出率が高く，抽出された属性が手書き文字の認識には有効に機能している．

表 2. 評価実験結果.

普通	文字認識率（1 位決定率）	属性を利用した	98.85%
		属性を利用しない	86.77%
	文字候補含有率（10 位累積認識率）	属性を利用した	99.77%
		属性を利用しない	94.82%
メールアドレス	文字認識率（1 位決定率）	文字ごとの種類を利用した	97.17%
		属性を利用した	80.19%
		何も利用しない	62.26%
	文字候補含有率（10 位累積認識率）	文字ごとの種類を利用した	99.53%
		属性を利用した	99.06%
		何も利用しない	94.81%
平均処理時間(ms)		帳票 A	4003.0
		帳票 B	2987.2
記入枠位置検知率			100%
重畳データデコード成功率			100%

8. まとめ

本報告は，アクティブ帳票のためのドットテクスチャに情報を重畳するためのコーディング法とデコーディング方法について述べた．種々のコード形状を検討し，それらに相応しい識別方法を試作して，その性能を評価した．さらに，記入枠と手書きの高速で安定な分離方法，手書きと記入枠との重なりやノイズに頑健な記入枠と記入枠構造の検知方法を提示した．記入枠のドットテクスチャから抽出した属性を利用することで，分離された手書き文字の認識率が著しく高められた．実験的な結果は我々のアクティブ帳票の方法が有効であることを示した．

謝辞

本研究開発は，経済産業省の平成 13 年度即効型地域新生コンソーシアム研究開発事業による．

文　献

[1] http://www.deviceinmotion.com/

[2] http://www.anoto.com/

[3] 長谷川史裕，別所吾朗，山形秀明，アフィン変換の係数を用いた定型帳票内の文字抽出，電子情報通信学会進学技報，PRMU97-186，pp.7-14，1997.

[4] Y. T. Lin, C. C. Rung, “Recognition and data extraction of form documents based on three types of line segments”, Pattern Recognition, Vol.31, No.10, pp.1525-1540, 1998.

[5] 西脇大輔，上谷昌昭，田中直哉，山田敬嗣，黒色罫線枠帳票読み取りのための文字枠検出除去と接触文字の切り出し認識，電子情報通信学会進学技報，PRMU97-218，pp.9-16，1998.

[6] B. Yu, A. K. Jain, “A generic system for form dropout,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.18, No.11, pp.1127-1134, 1996.

[7] 朱 碧蘭，島村 太郎，中川 正樹: アクティブ帳票処理技術の試作，信学技報，PRMU2002－133, Vol.102, No.531, pp.25-30, 2002.

[8] B. Zhu, T. Shimamura, M. Nakagawa, “Document Image processing methods for active forms,” to appear in Proc. 3rd IASTED International Conference on Visualization, Imaging, and Image Processing, Spain, 2003.

[9] B. Zhu, M. Nakagawa, “Information encoding into and decoding from dot texture for active forms,” to appear in Proc. ACM Symposium on Document Engineering, France, 2003.

[10] 浅野三恵子，下辻成佳，セル構造を用いた帳票識別，電子情報通信学会論文誌，Vol.J80-D-Ⅱ，No.1，pp.131-138，1997.

[11] F. Cesarini, M. Gori, S. Marinai, G. soda, “INFORMys: A Flexible Invoice-Like Form-Reader System,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20, No.7, pp.730-745, 1998.

[12] M. D. Garris and D. L. Dimmick, “Form Design for High Accuracy Optical Character Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 6, p653-656, 1996.

[13] 平野敬，岡田康裕，依田文夫，ロバストなモデル照合に基づく FAX 送信された一般帳票の読取り，電子情報通信学会論文誌，Vol.J85-D-Ⅱ，No.9，pp.1371-1381，2002.

[14] 前田陽二，中川正樹，ペーパインタフェースによる文書編集方式，情報処理学会論文誌，Vol.41，No.5，pp.1308-1316，2000.

[15] K. Kise, Y. Miki, K. Matsumoto, “Backgrounds as Information Carriers for Printed Documents”, Proc.15thICPR’2000, Vol.4, pp380-384, 2000.9.

[16] http://www.parc.com/solutions/dataglyphs/