修士学位論文

表形式文書要素の認識手法に関する研究

東北大学大学院工学研究科
電気及通信工学専攻

金津 知俊

目 次

第 1 章

序論

1.1 本研究の背景

マルチメディアデータベース及びミクストモード通信の普及により、図表、写真などの混在する印刷文書を計算機に自動的に読みとらせ、コード化するための技術が求められている。特に近年では、画像としてページ単位で入力された文書の記述内容を、単に文字単位で記号化するだけでなく、文書の構成要素として文字行領域、表領域、図・写真領域、その他の領域(ヘッダ、フッタ等)に分離抽出、これらの配置関係から文書の論理構造を把握して構造化する、いわゆる文書画像構造解析に関する研究が各所で盛んに行なわれている。現在では論文誌等を対象としてこれら実用化されつつある。

この構造解析技術を用いて、図表を含む文書のコード化をおこなう文書認識システムを構築しようと考える場合、抽出された文書構成要素に対して、それら各々を理解する認識部の確立が必須である。この構成要素の理解のうち、文字行領域の理解技術については既に印刷文字認識システムの技術としてほぼ完成されており、現在では様々な付加処理で100%の認識率を目指している段階と考えられる。また図や写真領域に関しては図形理解、データ圧縮等といった画像処理の分野で研究が盛んである。ここで残る構成要素は表領域である。特に論文誌や技術書を認識の対象とする場合、文書認識システムへ入力される文書中には表の現われる頻度が高く、しかもその表がもつ情報は重要であるため、表の論理構造を正しく理解してコード化する表認識システムへの要求は高い。しかし表のフォーマットは文書の種類、もしくは同一文書中にあっても筆記者の意図によって様々に変化しているため、求める表認識システムは多種多様のフォーマットにも柔軟に対応できるものでなければならない。例えば、文書中に登場する表には、図1.1のようにその各要素が各々一個ずつ罫線で区切られているものが多いが、図1.2のように罫線の一部あるいはほとんどが省略

	クローズ	オープン	処理時間
方式 a	98.50	88.66	1.00
方式 b	96.88	87.19	1.63
方式 c	99.28	89.12	1.12
方式 d	93.16	77.19	0.68

図 1.1: 一般的な表のスタイル

判定法	クローズ	オープン	処理時間
方式 a	98.50	88.66	1.00
方式 b	96.88	87.19	1.63
方式 c	99.28	89.12	1.12
方式 d	93.16	77.19	0.68

図 1.2: 罫線に省略のある表

されている表も決して少なくはない。また表要素が全て格子状に配置されている表ばかりではなく、包含や省略がある表の場合にもその対応を考えて正しく論理構造を抽出せねばならない。更に実用面からは、入力される表画像における品質の劣化や解像度の違いに対しても、常に安定した構造抽出がおこなわれることが要求されている。

1.2 表認識の概要

1.2.1 対象となる表の形式

文書認識システムが扱うような文書中において、表認識の対象とすべき「表形式の文書(以下、単に表と書く)」とはどのようなものを指すのであろうか。例えば辞書 [1] では表について「こみいった事がらを、見やすいように組織的に配列して書きあらわしたもの」と説明している。すなわち広義には単に物事を並べて書いたものをすべて表と呼ぶことができると考えられる。しかし一般に「表」という言葉から連想される文書の形式、或いは広く文書の中で「表」と特別視して扱う重要性が高いと思われる形式は図 1.3のようなものであろう。

このような表中の要素のひとつひとつを「項」と呼ぶことにする。「項目」とは、同じ属性を持つ、あるいは意味を共有する項の集合であり、縦あるいは横に並んでいるという関係によって分けられている。任意の項は列と行とに相当する縦と横との 2 つの項目に属しており、その相対的な位置関係からその 2 つの属性を直ちに知ることができるので、文書中においてはその表内に含まれる情報量も多い。実際文書中に登場する表の多くはこの形式であると考えてよく、本論文で述べる文書認識中の表構造認識はこの図 1.3のような項目の 2 次元的な広がりを持つ表が対象であると考える。

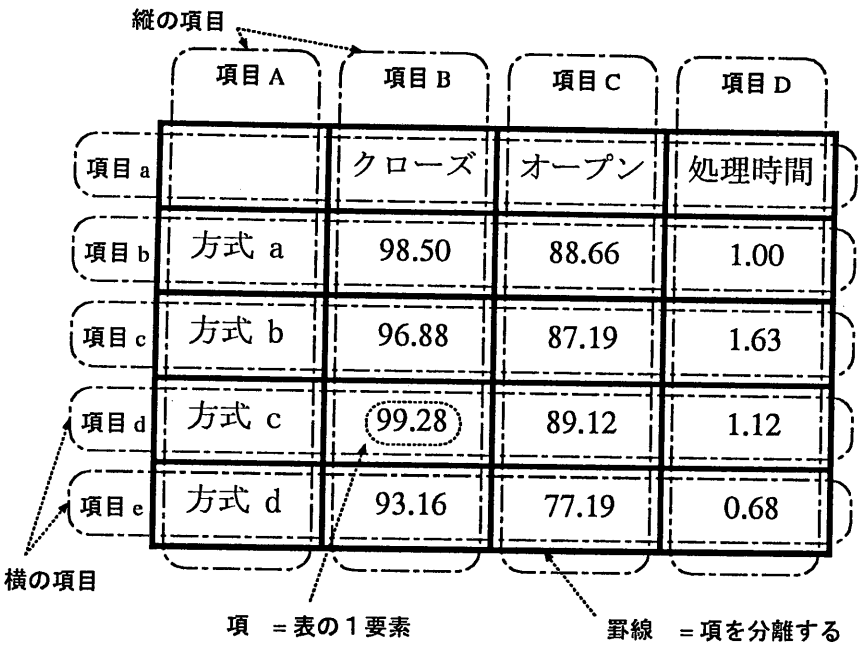

図 1.3: 認識対象となる表の形式

一方図 1.4 のような文書は図 1.3の表とは違い 2 次元的な項目の広がりは持たず、名称とデータという二項組で構成された項目が任意に配置される形になっている。このような文書の形式は帳票形式といい、特にデータ入力自動化の対象として古くから研究されていたもののひとつである。これを表と呼ぶことも多いが、前者の表とはあきらかに性格が異なるものであり、また文書中に登場することは稀であるため、本論文では表とは区別して考えることにする。

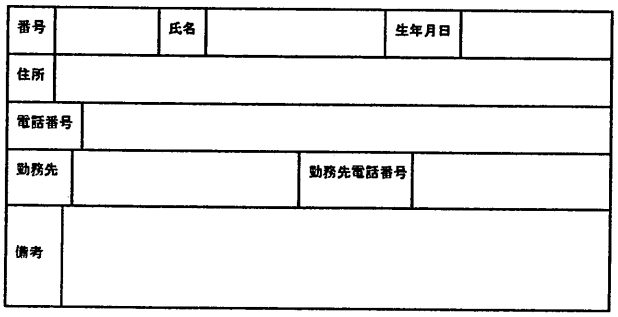

図 1.4: 帳票の例

1.2.2 表の認識処理

次に表を認識するとは具体的にどういう処理をさすのかを考える。文書認識システム中の表認識システムが目的としているのは、文書構成要素である表形式領域の認識である。

比較の為同クラスの構成要素の他の領域での認識処理を考える。例えば文字行領域認識では、まず領域として与えられる入力画像を文字行として縦に分け、次に文字行を個々の文字に分解し 1 つ 1 つ文字認識にかけて文字コードを得る。そして最終的に領域内の行ごとの文字コード列を得る。以上全部が認識の処理であると考えることができる。

これを表領域の場合で考えると、表領域の認識とは入力された領域画像から表の最小要素で、文字記号の集りである「項」のひとつひとつを抜き出してその配置をコード化し、更に各項に対しては文字認識をおこなって文字コードに変換する、という処理に相当する。ここで項の並び方が重要な情報をもつ表領域では、項の特定及びその配置をコード化する部分が特に重要であり、具体的には罫線がある場合はそれをセパレータにし、罫線がない場合は文字、記号の集群状況、上下、左右の項の相対関係から表の構造を判断することになる。このような構造のコード化処理を (表の) 構造理解と呼ぶことにする。

よって表の認識は次の 3 つのステップに分けることが出来る。

ステップ 1. 各項の空間的位置関係を得る
ステップ 2. 項の論理構造を理解する。
ステップ 3. 項の文字内容を得る。

ステップ 1 では、抽出された項の位置が絶対的な座標でのみ与えられる。つまりこの段階での出力は項の位置を示す座標とその内容である文字コード列の集合、そして画像上の線として記述される罫線の集合になる。適当な出力装置を用いた時に見た目が等しくなるように表が再構成されることのみが要求されているなら、この位置コードに、ステップ 2 を飛ばして直接ステップ 3 の文字認識の出力コードを付加することで十分に行なわれる。

しかし、コード化された表をデータベース等として活用するような場合、表中で注目する項がどの項目に属しているかということがデータより直ちに判ることが望まれる。このためには認識のときに表のもつ論理構造を理解し出力コードにそれを反映させなければならない。これが第 2 のステップであり、論理構造の理解と呼ぶことにする。表の特性を考慮すると、論理構造を理解するということは項の並びを明らかにするということに等しいと言える。

図 1.3のような表を対象にしているとき、すなわちすべての項がそれぞれ四方を罫線で囲まれているという仮定が成り立つときは、罫線によって項の並びが明らかである、すなわち同一罫線にはさまれた項が項目=ひとつの行や列、をなすと考えることができる。この行や列を図中の行列に対応させ、構造行列と呼ぶことにする。論理構造の抽出とはこの構造行列の抽出に等しいと言える。罫線を正しく抽出するだけで構造行列が書けるような表であればその論理構造を得ることは容易である。しかし罫線に省略があるような表では別

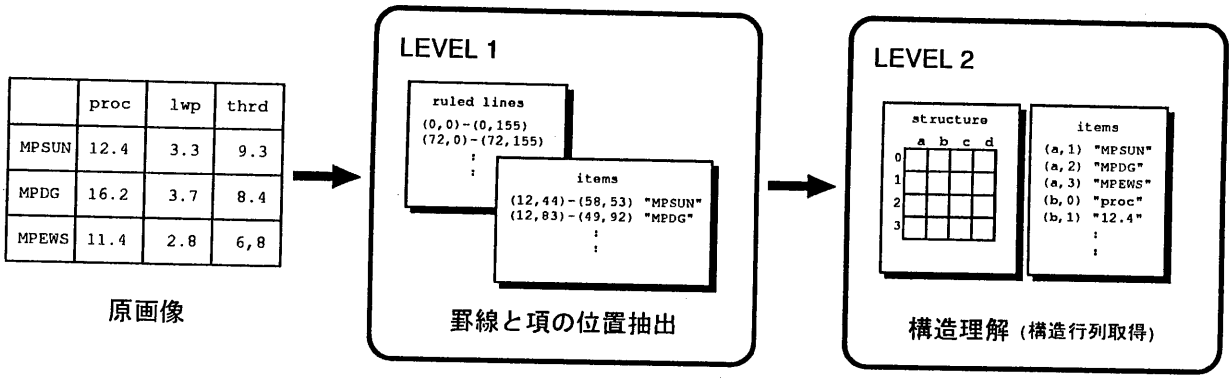

図 1.5: 表の構造理解

の手段、例えば項の相対位置などを利用して項の並びを得るなどしてから構造行列を構成しなければならない。

1.2.3 表認識の歴史

表形式の文書を認識するための研究は OCR 技術の開発当初から行なわれていた。帳票という定フォーマット文書に記載された内容を計算機に自動的に読みとらせて、大量データを高速に処理したいという要求は、文書情報の電子化が進み始めた当初からり非常に大きく、様々な研究成果が報告された。そのような処理を行なうのが帳票認識システムである。帳票認識では、読みとるべき文字の配置情報、すなわち表のモデルをオペレータが予めシステムに提示し、実際の読み取り時にこれを利用している。初期の帳票認識システムではこの読み取り位置は各帳票の種類毎に絶対座標で与えていた。帳票としては罫線がドロップアウトカラーで書かれた専用のものを用い、位置のずれは起らないように読み取りが行なわれ、技術的には文字認識の応用に過ぎなかった。

その後実際の印刷帳票からレイアウト情報を自動的に取得する研究がおこなわれ、いくつかの手法が提案されている。それらの手法は、帳票内のすべての項が枠に囲まれているとして、枠をなす罫線を抽出し相対関係を用いて表のレイアウトを認識するものであり、これは読み取り時の位置ずれの影響を軽減するために、罫線等の画像特徴を利用して読み取るべき項の位置を相対的に取得する技術、もしくは見本の帳票から「表の書式モデル」を自動的に取得するための技術として研究された。但し二値画像から罫線のような線特徴を抽出する処理は既に確立されていたので、それらの研究の主眼は構造の記述法 [2] や文字認識と協調しての表の意味の理解 [3] などに向けられていた。

だが近年、図表や写真の混在した一般的文書を対象にする「文書認識システム」の研究

が進み、各種文書の自動コード化がおこなわれるようになるにつれ、その文書認識システムの一部分として表構造を認識する必要性が生じてきた。この場合、入力として全く構造が未知である表画像をコード化するために、表の構造を理解しなければならない。

罫線によって各項が区切られた表に対しては、既存の方法によってその構造を自動的に抽出できた。しかし、読み取り対象が帳票認識における見本のように画像的に安定したものであるとは限らない。そこで、実用的なシステムとして主には様々な表を読み取り可能にする為の頑健性に対する対策が研究されていた。具体的には印刷状態の善し悪しによって生ずる罫線のかすれ、切れ [4]、あるいは潰れによる罫線と文字の接触への対策、さらにはフォーマット面での自由度を広げるために、周囲のみ罫線がない表への対応、点線、破線で表現された罫線の認識 [5] などである。特に [4] ではそのようなことを考慮しつつ、表のフォーマットには制限が大きいものの、前述の論理構造理解に相当する表の領域分けまでおこなっている。近年では主に論理構造に着目するものとして、構造の記述法に対する研究もおこなわれている。しかしこれまでに提案された手法では入力の解像度を固定しているなど、実際に文書認識システム中の表認識に使用するには頑健性の面で不十分である。

また、一般文書中の表を対象としたときの大きな問題として、罫線の一部あるいはほとんどが省略された表が多く存在するということがある。これまでの研究の多くは、罫線を抽出することで表の項がすべて分離できるという仮定を置き、罫線に省略のある表を無視してきた。実際、罫線のない表から項の一つ一つを確実に正しく抽出することは非常に難しい。確かに人間はそれを行えるが、これは各文字を認識して意味上から項を分離できることに因る所が大きく、現時点における表の文字内容を考慮しないシステムでは対応出来ない処理である。ただし、人間に文字部を単なる矩形とした表を提示した場合でも、多くの場合正しい項の領域分けをすることが出来る。これは一般の表に期待されるような、項内の文字は近接し、項間にはある程度のスペースがある、縦横の項目内で各項は並んでいるという、知識に基づくルールを適用しているからであると考えられる。つまりこのようなルールを与えれば計算機でもある程度の項分けが可能になるということ思われる。しかし、たとえ項を正しく分離できたとしても、罫線のような明確な区切りがあたえられてない部分において、大きさのまちまちな項の上下左右の並びからそれらの属す項目を正しくとらえ、論理構造を抽出することはたいへん困難である。

そのため、文書入力のための表構造認識手法に関する報告で、そのような表を対象としているものは少ない。糸乗は文字の集まりをひとつのブロックとして、その並びを利用し罫線の省略された表中の項を特定する手法を提案した [6]。しかしこの手法では、論理構造の抽出時には項目には省略がないことを仮定している。実際には罫線がない上に項に省略、包含のあるような表も多く存在するため、そのような表にも対応できる表認識システムを

構成する必要がある。

1.3 本研究の目的

本研究では、表認識システムを作成する際に必要な、画像から表の構造を自動的の認識するための技術として、二値化された入力画像から表の項を分離するための画像特徴抽出手法、及び抽出された項から表の論理構造を推定し構造行列を抽出手法について検討する。この時想定する表認識システムを、実用的な文書認識システム中のひとつのユニットとして位置づけ、提案する手法が様々なフォーマットの表に対応できること、入力画像の変動に対して安定した性能をもつ手法を開発することを目的としている。

具体的に以下のことが要求される。

- **様々なフォーマットへの対応:** 罫線主体の表、罫線に省略のある表のどちらからも表の項のひとつひとつを特定し、さらにその相対関係がなす論理構造を構造行列として正しく取得することが出来る。その際両者を入力段階で区別することはせず、システムが自動的に判断出来るようにする。
- **入力画像の変動に安定:** 解像度、サイズ、もしくは画像品質の劣化に対し広い範囲で対応できる。

1.4 本論文の構成

本論文の構成は以下のとおりである。

第 1 章 本研究の背景と目的について述べる。

第 2 章 表の構造の基本的な要素として項を分離するために、表中の罫線および文字ブロックを入力 2 値画像中から抽出する方法について、実用面を考慮し、特に入力画像の変動に左右されないための対策を中心に説明する。

第 3 章 論理構造理解のために表の構造行列を抽出する。罫線による区切りが充分であるかどうか判断の上で、罫線の省略された部分へ対応するための、行の対応と平均的行構造を考えた項目分け手法について説明する。

第 4 章 実験により、構造行列の抽出能力を評価する。

第 5 章 全体の結論と今後の課題について述べる。

第 2 章

表の罫線と文字ブロック抽出

2.1 まえがき

本章では、表認識の第 1 ステップである、入力された原画像から表の項を分離するための処理について説明する。多くの従来手法のように表の各項が罫線で仕切られているという前提のもとでは、線特徴として罫線を抽出することで表の項をすべて分離することができる。しかしここでは罫線の一部もしくはすべてが省略されている表に対応するために、矩形特徴として項中文字を抽出し、その集まりの度合から項を文字ブロックとして分離することもおこなう。

図 2.1 は表の項を分離する項分離抽出部の概略である。これは、原画像から線特徴を用いて罫線を抽出する部分と矩形特徴を用いて文字ブロックを求める部分との 2 つに分けることができる。本手法ではこの 2 つを完全に独立とせず、途中段階でデータを相互に参照することで相補的に性能を向上させられるような構成を取っている。

以下、罫線抽出部と文字ブロック抽出部の特徴抽出処理の内容を具体的に説明する。但し前述の線、矩形両特徴比較による相互補強処理のため、データが 2 つの処理間を行き来することになるので、説明は処理の順に従い両部分を交互に説明する。

2.2 入力画像

項の分離抽出部の入力となるのは、文書の二値画像中から文書認識システムが表領域と判断して切り出した矩形領域である。前段の文書認識システムによって余分な文字列 (註釈文字行など) は正しく分離され、また傾きもないように補正されているものと考えて以後の処理をおこなう。

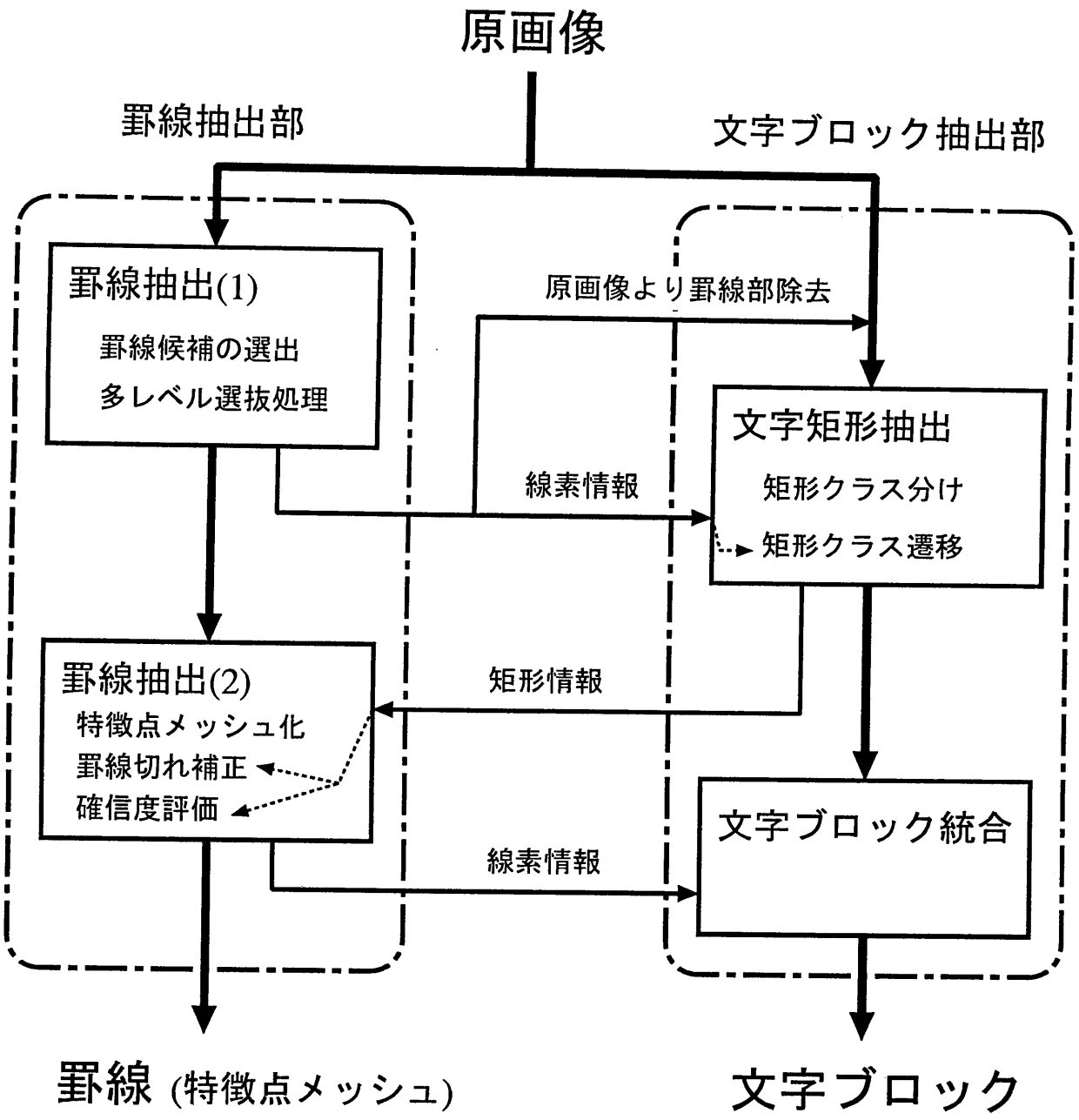

図 2.1: 罫線と文字ブロック抽出部

2.3　罫線抽出部 (1)

原画像から線素をすべて抽出して罫線の候補とし、その中から罫線らしいものを選抜してゆく。ここで、罫線抽出の処理は特に記載ない限り各方向 (縦、横) 同様におこなうものとする。

2.3.1　RUN の抽出、統合

画像を縦、又は横にライン走査し、黒画素が t_{minrun} 以上連続 (途中 2dot までの欠損を認める) している部分を抽出する。この集合を {RUN} とする。ここで t_{minrun} の値は小さくとっているので {RUN} には罫線をなす長い線素ばかりではなく、文字内の線なども含まれる。この {RUN} に対し、各 RUN どうし 8 連結で接触しているものに同一のラベルを与えていく。ここで同一のラベルを与えられた RUN の部分集合は図 2.2 のように、画像中で DIR (={'H'(横), 'V' (縦)}) 方向の線分をなす黒画素を抜きだしたものに相当する。

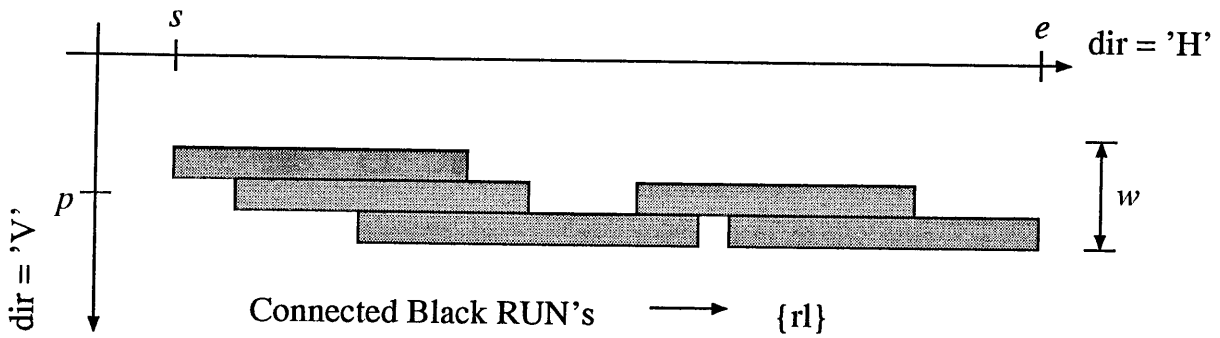

図 2.2: 同一ラベル (8 連結) の RUN 集合

同一ラベル RUN 集合に対しラベル毎画素追跡をおこなって、各々をひとつの線分とみなした

$$(d, s, e, p, w) = (DIR(\text{方向}), \text{開始座標}, \text{終了座標}, \text{位置座標}, \text{幅})$$

なる 5 つの値で表現する。開始、終了座標は長さ方向の座標値で、それぞれ RUN 集合中の画素の座標の最小、最大値に対応する。位置は長さに垂直な方向の座標値のことで、長さ方向に累積した画素ヒストクラムから重心として求めたものである。

この線分の集合から、長さ $(= e - s + 1)$ が t_{minrl} 未満のものを除き、残った長さ t_{minrl} 以上のものを前述の 5 項組

$$rl_i = (d, s, e, p, w)_i$$

の配列として記憶する。これを罫線候補集合 {rl} と呼ぶ。

値 t_{minrun}, t_{minrl} はそれぞれ実験的に $t_{minrun} = 8, t_{minrl} = 12$ に選んだ。

2.3.2　多段階罫線選抜処理

罫線候補集合 {rl} に対し、実際表中に存在する罫線に対応すると考えられるものを選び出してゆく。罫線候補を求めるときに用いた閾値 t_{minrl} もまた小さな値に設定してあるので罫線候補中には文字線の一部など、罫線ではないものが多く含まれている。その中から誤抽出を避けつつ罫線をすべて選び出すために、選出処理を多段階に分けておこなっている。

前処理

まず罫線候補から平均の罫線幅を推定する。この平均は罫線候補のうち長さの長いもの (表幅の 1/10 又は 32dot 以上) から算術平均で計算される。次に 2 つの罫線候補で幅方向に極めて接近している (平均罫線幅から判断)、あるいは長さ方向の延長上に存在し、空白距離が d_{wrlp} 以下のものを一本に統合する。統合された罫線候補はその座標値、幅などを再計算される。前者の場合、二重罫線の可能性を考慮するもので、それを示すフラグを追加して線の性質を後に参照できるようにしておく。後者は主に画像二値化の際に生じた数ドット程度の罫線の切れを修正することを目的としており、過度の統合を防止するため $d_{wrlp} =$ 8dot 程度 と小さく設定しておく。かすれなどによって生じた罫線の大きな欠けを補正するためのの罫線切れ補正処理は別処理として後におこなわれる。

主要罫線 (レベル 1 罫線) の選出

表中で複数の項にまたがっている長い罫線を主要罫線と呼ぶことにする。一般に原画像で大きな罫線切れが生じていなければ、

$$RL_1 \leftarrow \{\text{長さが表の幅 (高さ) の}\frac{1}{3}\text{以上のもの}\}$$

という条件で主要罫線を選出することができる。これをレベル 1 罫線と呼ぶ。

短い罫線の選出

罫線でありながら、レベル 1 罫線として選出出来ずに候補中に残っている罫線の状態としては次のような可能性が考えられる。

ケース 1　画像のかすれによって長い主要罫線が大きく切れ、レベル 1 罫線の下限に満たない長さの部分が残っている。

ケース 2　図 2.3中の太線で示される罫線のように、元々1〜数項に部分的に引かれている部分罫線で、長さがレベル 1 罫線の下限に満たない。

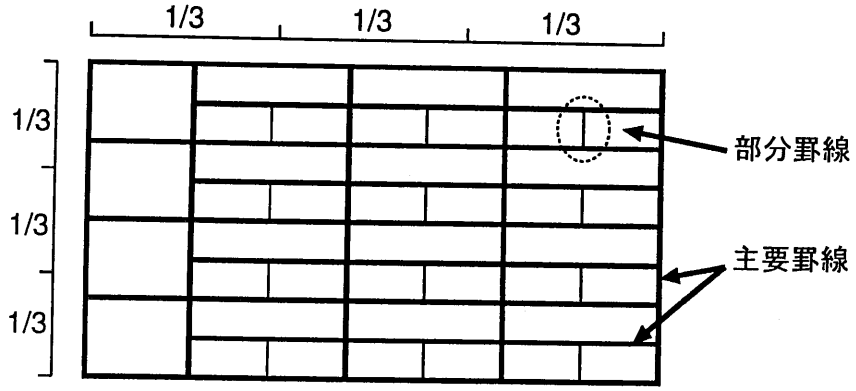

図 2.3: 主要罫線と部分罫線の例

レベル 2 罫線の選出

ケース 1 の罫線は、既に抽出したレベル 1 罫線の延長上に存在している罫線候補であると考え、これをレベル 2 の罫線として選出する。

$$RL_2 = \{RL_1 \text{の延長上に存在}\}$$

このレベル 2 罫線とレベル 1 罫線との結合は現段階ではおこなわず、後述の確信度再評価の後、罫線切れ補正処理によって妥当性を評価しながら結合をおこなう。

レベル 3 罫線 (部分罫線) の選出

ケース 2 の部分罫線のような短かい罫線を候補から選出する際には慎重を要す。従来的な手法では、長さの閾値のみを用いて短い罫線を選抜しており、しかもその閾値を、入力される表の原画像の解像度や項数の最大値などが一定であることを仮定し定数として組みこんでいた。そのため対象となる表の解像度や項数が表によって様々だと、短かい罫線が抽出できなかったり、逆に文字中の線を罫線と誤ったりという罫線の抽出誤りが生じた。

本手法ではこれを避けるために、レベル 1 で抽出された罫線を用いて表中項の最小幅 (高さ)v_{minsc}を動的パラメータとして推定し選出の際の閾値に用いる。この値 v_{minsc}の推定プロセスは以下のとおりである。

項最小幅 (高さ) の推定

1. 位置の異なるレベル 1 罫線が 3 本以上抽出されている。
2. それらの罫線の最小間隔が $t_{defaultminsc}$ 以上である。

という条件が両方満たされた場合に限り、その最小間隔 v_{minsc} を項の最小幅 (高さ) と推定する。また、上の条件が満たされなかった場合は v_{minsc} =$t_{defaultminsc}$ とする。この $t_{defaultminsc}$ は 200dpi の時の 6pt の文字高のおよそ 1.5 倍とした。

部分罫線選出のための閾値として、この v_{minsc} から求めた

$$v_{minrl3} = 0.8 \times v_{minsc}(\text{各方向別に})$$

なる値を用いることにする。上記のパラメータが推定出来なかった場合、デフォルトの固定閾値を用いねばならないが、ほとんどは罫線の省略が多くレベル 1 罫線の有効本数が元々存在しないような場合なので、この時にはケース 2 で抽出せねばならないような部分罫線も存在しないことが多く問題にはならないと考えられる。

さらに、部分罫線は既に選出した長い罫線と全く独立ではなく、交差やT字などといった接続関係をもつことに注目する。

図 2.4のように、各罫線 (と罫線候補) の周囲に各線幅から接触領域を定め、2 罫線間でこの領域が重なるとき 2 つの罫線は接続していると考える。

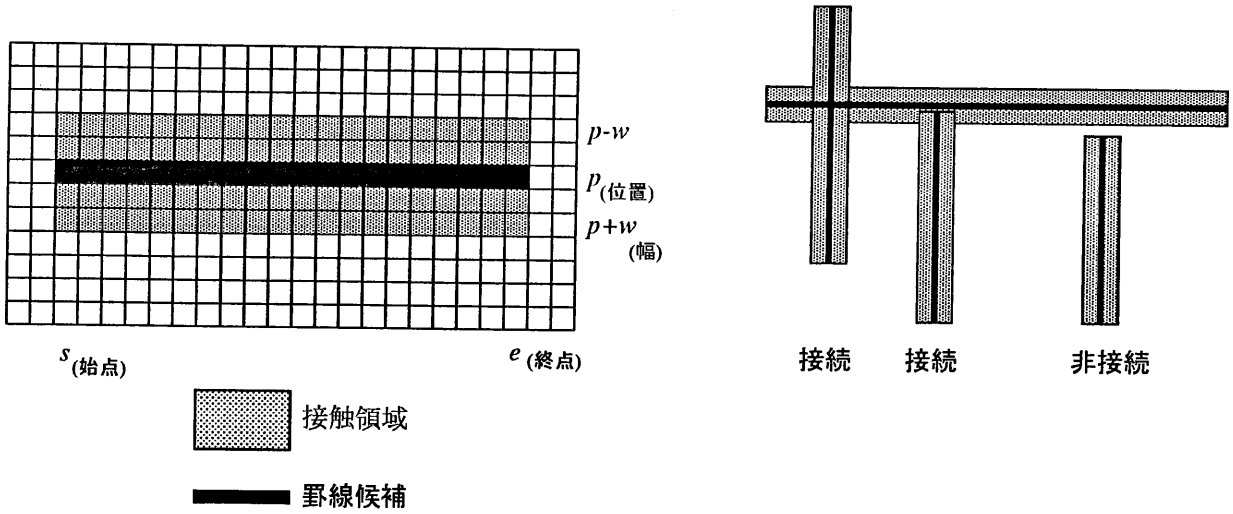

図 2.4: 接触領域と 2 罫線の接続

既に選出されているレベル 1 罫線とレベル 2 罫線の集合 ($\{RL_1\} + \{RL_2\}$) と、それらに垂直な罫線候補 {rl} 間の位置関係を調べ、接続関係をもつ罫線候補のみが選出の対象になる。

以上より、部分罫線をレベル 3 罫線

$$\{RL_3\} \leftarrow \{\ \text{垂直な}\ RL_1, RL_2\ \text{と接続し、長さ}\ v_{minrl3}\ \text{以上のもの}\ \}$$

として選抜する。

段階的罫線選出の効果と不十分な点

このように段階的な罫線抽出をおこなうことで、線素として多めに抽出された罫線候補から、広範囲の表に対して誤抽出をなるべく抑えて罫線を選出することが可能になる。

しかし罫線と文字中の線の区別を長さでしか行なっていない以上、選出された罫線の中に文字線の一部が混入するのを防ぐことは難しい。そのような誤選出をひきおこすの例として、レベル 1 罫線の延長上にたまたま線要素の大きいの文字があった場合、罫線と文字の間隔が狭い表に対してケース 2 の罫線と文字線との区別が難しい場合などがあげられる。このような文字線との詳細な区別は表中の文字の存在を明らかにしたうえでおこなう必要がある。

2.3.3　処理例

罫線抽出部 (1) の処理例を図 2.5〜図 2.8に示す。

	全文字数	リジェクト	誤認識		追加テンプレートパターン		認識率 (%)
			文字間違い	フォント間違い	初めての文字	初めてでない文字	
data1	3758	282	69		117		90.7
			44	25	80	37	
data2	3475	235	77		88		91.0
			64	13	49	39	
data3	3782	100	54		41		95.9
			42	12	18	23	
data4	3273	71	48		45		96.4
			44	4	11	34	
data5	3363	69	64		30		96.0
			50	14	14	16	
data6	2801	88	48		40		95.1
			47	1	19	21	

図 2.5: 原画像 (400dpi)

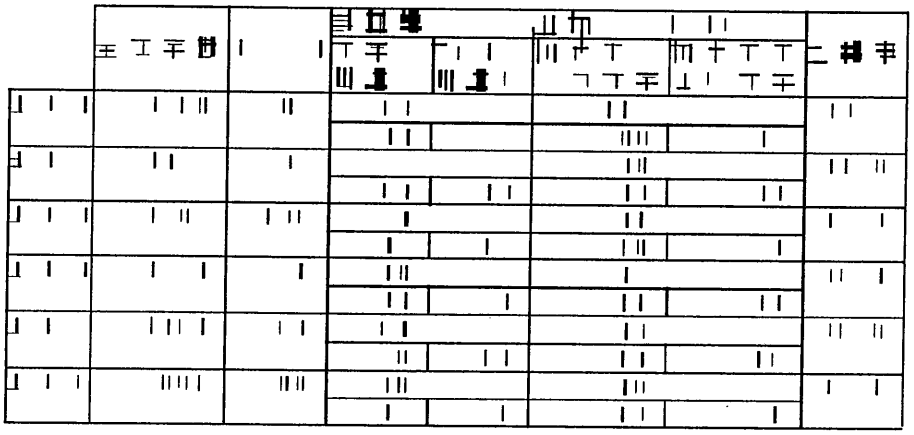

図 2.6: 罫線候補

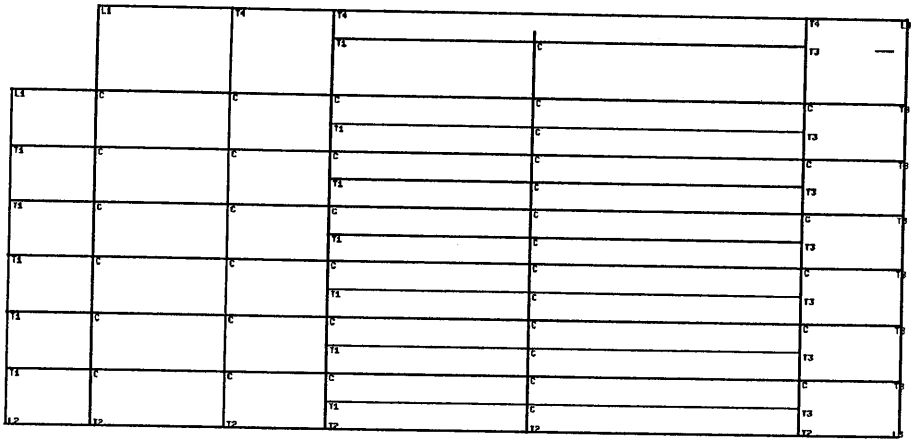

図 2.7: レベル 1、レベル 2 罫線

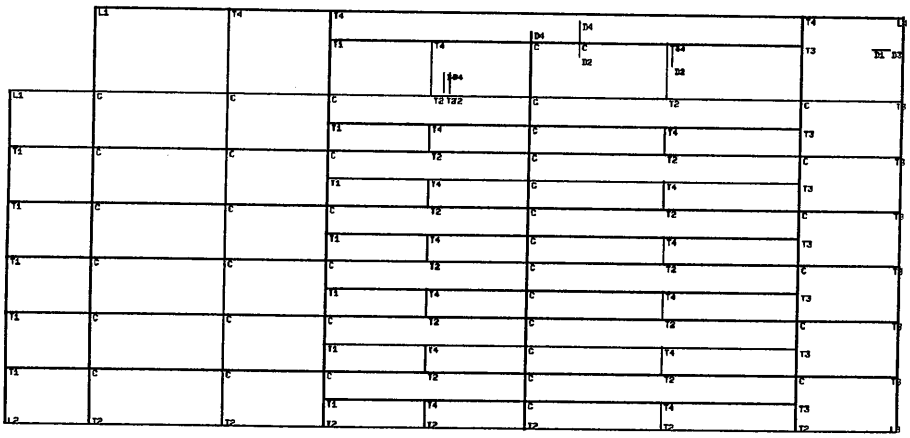

図 2.8: (レベル 1、レベル 2 罫線)+ レベル 3 罫線 (部分罫線)

2.4　文字要素矩形の抽出

表の原画像から矩形特徴を用いて文字または文字の一部となる文字要素矩形を抽出する。

2.4.1　矩形特徴を抽出する画像

前節の罫線抽出部 (1) で得られたレベル 1 の罫線、及びレベル 2 の罫線のうち長さが v_{minsc} 以上のものについて、その罫線部分の黒画素を表の原画像から取り除く。この処理は罫線に接触した文字が抽出不能になるのをさけるためにおこなう。レベル 3 の罫線及び、v_{minsc} 以下のレベル 2 罫線は罫線としての確信度が低く、これから抽出すべき文字の一部である可能性も高いので消去しない。

消去は、罫線 (d, s, e, p, w) に対し、その方向別に座標を (罫線に沿う方向, 罫線に垂直な

方向) で書くことにして、

$$(s, p - w/2), (s, p + w/2), (e, p + w/2), (e, p - w/2)$$

の 4 点で囲まれる矩形中の黒画素を白画素に変更することでおこなわれる。消去矩形端に細線状のごみが残る場合もあるが、それらは後の処理で対応する。

2.4.2　8 連結矩形の抽出

画像中 8 連結で接触している独立黒画素群を、その外接矩形 (すなわち、左上及び右下の座標) として抽出する。抽出された矩形集合を {cr} と書くことにする。

2.4.3　矩形クラスの分類

{cr} をその形状及び大きさで次のように 7 つのクラスに分類する。

矩形クラスの分類

$$\{\text{cr}\} \rightarrow \{\{\text{NOISE}\},\{\text{HLI}\},\{\text{VLI}\},\{\text{HLICH}\},\{\text{VLICH}\},\{\text{CHR}\},\{\text{LBOX}\}\}$$

各々は

{NOISE}:	小さな点図形。ノイズと考える。
{HLI},{VLI}:	扁平率、(短辺の) 幅から細長いと思われる矩形のなかで (長辺の) 長さの長いもの。罫線 (の一部) と考える。
{HLICH},{VLICH}:	{HLI},{VLI} 同様細長い矩形だが、長辺の短かいもの。罫線 (の一部) もしくは文字 (の一部) と考える。
{CHR}:	正方にちかい矩形、文字 (の一部) と考える。
{LBOX}:	偏平率は低いがその大きさが大きく、文字等が 2 個以上接触して生じたと考えられるもの。

これら 7 つのクラスのうち、{CHR},{HLICH},{VLICH} に属するもののみが文字を構成する要素と考えられ、後述文字ブロックの作成に用いられることになる。

次に {cr} から各クラスへの分類の具体的に手順を示す。

矩形クラス分類処理

それぞれの矩形に関して w_x, w_y = 幅, 高さ, r = 幅/高さとする。

$$\{\text{NOISE}\} \leftarrow \{\ \text{cr} \mid w_x < v_{noisew_x} \wedge w_y < v_{noisew_y}\ \}$$
$$\{\text{HLI}\} \leftarrow \{\ \text{cr} \mid r > t_{rate} \wedge w_y < v_{linewmax} \wedge w_x \geq v_{linelmin}\ \}$$
$$\{\text{HLICH}\} \leftarrow \{\ \text{cr} \mid r > t_{rate} \wedge w_y < v_{linewmax} \wedge w_x < v_{linelmin}\}$$
$$\{\text{VLI}\} \leftarrow \{\ \text{cr} \mid r < 1/t_{rate} \wedge w_x < v_{linewmax} \wedge w_y \geq v_{linelmin}\ \}$$
$$\{\text{VLICH}\} \leftarrow \{\ \text{cr} \mid r < 1/t_{rate} \wedge w_x < v_{linewmax} \wedge w_y < v_{linelmin}\ \}$$
$$\{\text{CHR}\} \leftarrow \text{others}$$

パラメータのうち $v_{noisew_x}, v_{noisew_y}, v_{linewmax}, v_{linelmin}$ 等は主要罫線の抽出結果から得られる表罫線の線幅から、解像度を予想して動的に決定する。ここまで分類したのち、{CHR} に属す {cr} の平均幅 ($\widetilde{CW}$) と平均高さ ($\widetilde{CH}$) を求め、

$$\{\text{LBOX}\} \leftarrow \{\text{CHR} \mid w_x > 4.0 \times \widetilde{CW} \wedge w_y > 2.5 \times \widetilde{CH}\}$$

として、標準以上の大きさのものを {CHR} から除外する。

2.4.4　罫線候補集合 {rl} との比較による {cr} クラスの変更

形状とサイズのみからおこなったクラス分けでは、罫線要素と文字要素との区別にあいまいな部分がある。ここでは罫線あるいは罫線の一部が文字矩形要素として抽出されるのを避けるために、罫線抽出部で得られる罫線候補集合 {rl} を利用する。

すべての {rl} と {cr} の間にある関係を求める。ひと組の {rl} と {cr} の間の関係は、位置 ({rl} の幅方向の距離) 関係と端点 (始点、終点) 状態の 3 つで表わされる (図 2.9参照)。

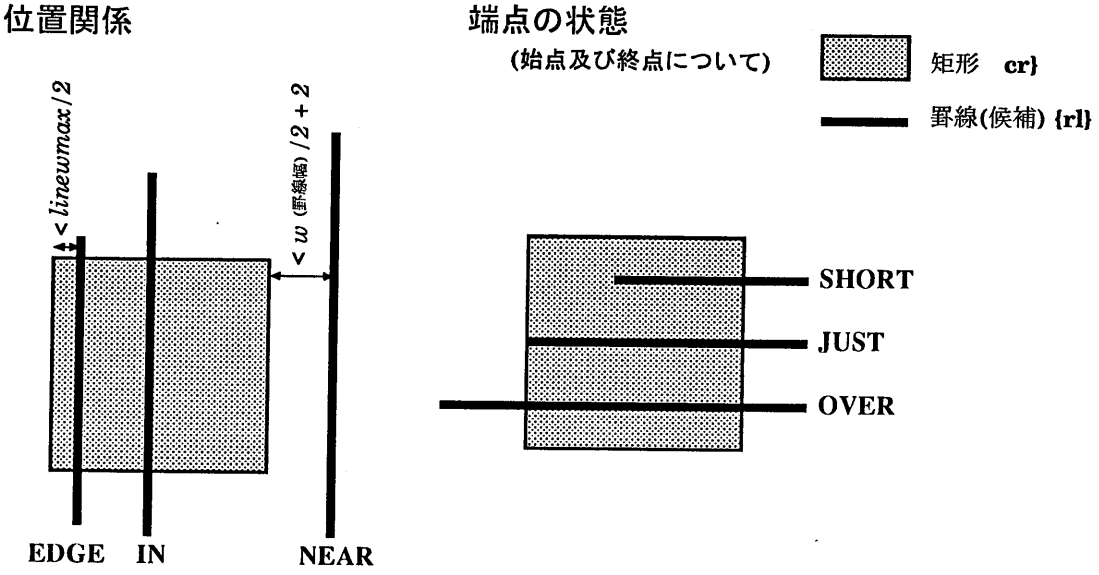

図 2.9: {cr} と {rl} の間の関係

{cr} 中の一部の矩形のクラスは、条件で示される関係をもつ {rl} の存在によって下表のように変化する。

変化前	条件	変化後
{LICH}	RL1 or minsc 以上の RL2 が NEAR	{LI}(変化 1)
{LICH}	SHORT 以外の RL3 が IN or EDGE	{LI} (変化 2)
{LI}	上記の 2 ついずれでもない	{LICH}(変化 3)

※ {LI}={{HLI},{VLI}}, {LICH}={{HLICH}, {VLICH}} であり、条件中の関係は {rl} と {LI}, {LICH} の向きが互いに等しい場合のものでなければならない。

変化 1 は {LICH} が原画像から除去された罫線の消しのこしであることを、変化 2 は {LICH} が部分罫線であることを判断して、それぞれを線要素と確定することを示す。逆に変化 3 では長さ的に文字ではなく線であると判断された矩形が対応する {rl} がない、すなわち線特徴から見た罫線としての裏付けがないために文字としての可能性を加えることを示している。

この処理ののちに {CHR},{HLICH},{VLICH} であったものだけが文字、あるいは文字を構成する要素であると考え、文字ブロックの生成にもちられる。3 つの集合の和を文字要素矩形集合 (={CHRC}) と呼ぶ。

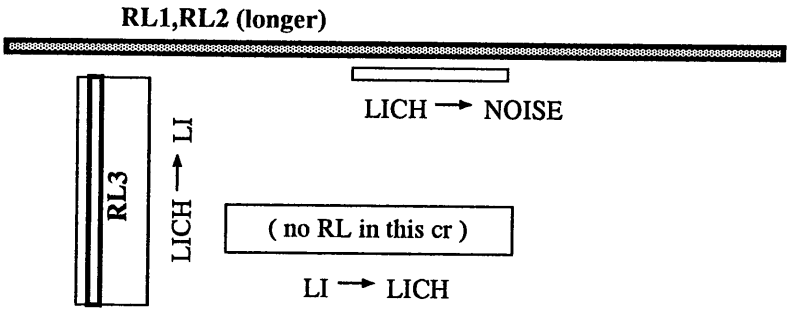

図 2.10: {rl} と {cr} の関係による {cr} のクラス変化の例

2.4.5　点線・破線の抽出

矩形 {cr} のうち幅の狭いもの {{NOISE}, {LICH}, {LI}} が一列に、ある間隔以内で並んでいるとき、これを点線・破線として抽出する。並びの判断はヒストグラムを用いておこなう。抽出された点線・破線はこれ以降罫線として扱い、長さおよび主要罫線との関係によって前段同様 3 レベルに分類される。

2.4.6　文字要素矩形抽出処理例

		ユークリッド距離		シティーブロック距離	
		甲	乙	甲	乙
方向線素特徴量	領域半径外(字)	14	16	21	29
	誤認識数(字)	23	26	25	34
	回復率(%)	60.9	61.5	84.0	85.3
	1位認識率(%)	99.922	99.912	99.916	99.885
方向密度ベクトル	領域半径外(字)	162	620	86	639
	誤認識数(字)	233	994	106	830
	回復率(%)	69.5	62.4	81.1	77.0
	1位認識率(%)	99.214	96.648	99.642	97.201

図 2.11: 原画像

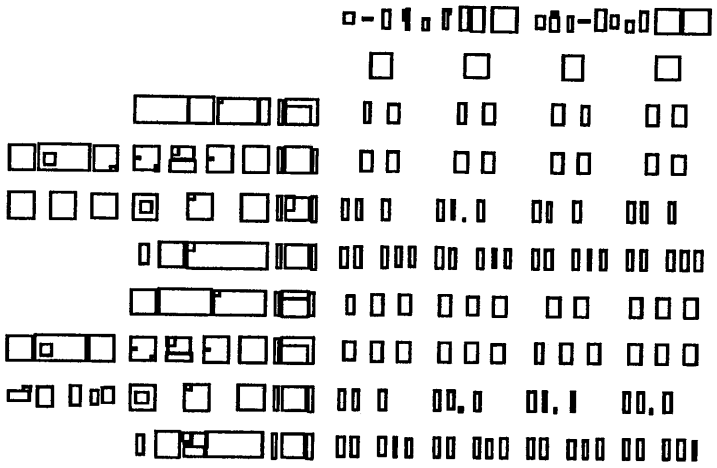

図 2.12: 文字要素矩形

2.5　罫線抽出部 (2)

前半の罫線抽出部 (1) によって線分として抽出された罫線を特徴点メッシュを用いて構造的に記述する。更に罫線切れの補正と確信度の低い単独罫線の除去処理をおこない、最終的な罫線抽出結果として出力する。

2.5.1　特徴点メッシュの作成

現時点では、選出されている罫線は各々始終点と位置座標が示すばらばらの線分にすぎない。例えば画像の状態によっては、角をなすべき互いに垂直な 2 罫線の端点座標が一致していなかったり、同一線上にあるべき複数の罫線の位置座標が異なるということもありうる。このような小さな座標のずれを正し、また罫線切れ補正処理などの見通しをよくするために、特徴点メッシュという 2 次元メッシュをつくってその上に罫線をマップして罫線を構造的に記述する。

特徴点 (={FP}) とは互いに垂直な 2 つの罫線が接続している点、および各罫線の始終点 (但し他の罫線と接続している点は除く) のことを指し、その方向、接続の仕方によって表 2.1のように分類される。

種類	罫線の状態	特徴点シンボル
交差点	┼	C
T 接続点	├	T1
	┴	T2
	┤	T3
	┬	T4
L 接続点	┌	L1
	└	L2
	┘	L3
	┐	L4
端点	─	D1
	│	D2
	─	D3
	│	D4

表 2.1: 特徴点の種類

スケール変換による特徴点メッシュへのマップ

原画像上の座標という絶対位置であらわされる特徴点に対して、

$$width \times height \rightarrow M_w \times M_h$$

($width, height$ は原画像の幅、高さ (dot), M_m, M_hは特徴点メッシュの幅、高さ)

というスケール変換をおこない、新しい空間上に各特徴点をマップする。このとき前述の座標上での小さな位置ずれは、変換の際に位置を量子化することで吸収することができる。

スケール変換のために、スケールという名前の一次元配列の変換テーブルを縦、横それぞれについて作成する。画像内での位置の実座標値 (pt) に対し、行列での位置である量子化値をスケール値 (mpt) と呼ぶことにする。

スケールの求めかた

縦、横それぞれについて次の手順で求める

1. 全特徴点の (現在注目している方向の) 座標値をある規則で並べる。
2. 並べられた座標値をスケール作成関数に順に与える。スケール値が戻り値として得られるので各特徴点に記録する。

スケール作成関数

入力された実座標値 pt に対し、既に作られている m 個 のスケール配列中の実座標値 $pt^i (i = 0, 1, 2, \ldots, m-1)$ と比較し、一番近接する座標値 pt^j を求める。このとき $|pt - pt^j| \leq w$ ならば、そのスケール値としてテーブルのインデックス j を返す。w より大きい時、もしくはスケール配列がまだ空のときはこの pt を実座標にもつ新しいスケールを配列に加えてその新しいインデックス m をスケール値として返す。

位置ずれ許容幅である w を 1 以上にとることによって、罫線の位置ずれをまとめることができる (wは平均線幅の 2 倍の値とした)。このとき変換処理では、入力する座標値の並び方によって結果が左右されるが、この座標値を

1. 求めるスケールの方向とは垂直な罫線のもつ特徴点の位置座標を、その罫線の長さの順
2. スケール方向に平行な罫線の端点を、罫線の長さの順

のように並べることによって、主要な罫線を核として座標がまとまる。

以上の処理を縦横それぞれについておこない、できた 2 つのスケール配列を実座標の小さい順に並べなおす。スケール値に従って行列上に各特徴点を置いてゆくと、図 2.13のような特徴点メッシュが完成する。

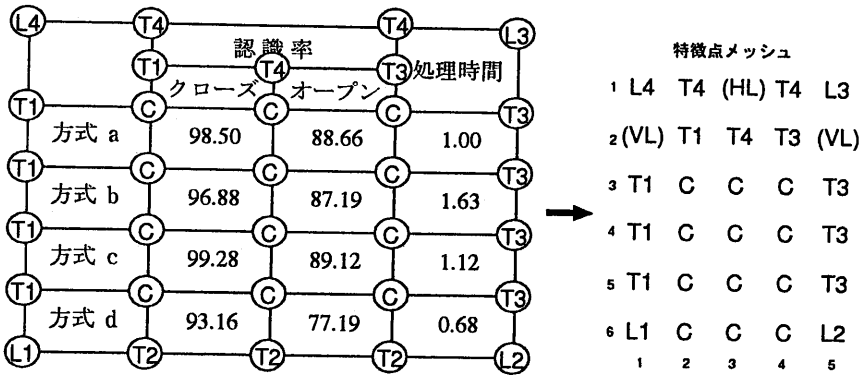

図 2.13: 特徴点メッシュの例

2.5.2　確信度の付与

レベル 3 罫線集合＋長さ minsc 以下のレベル 2 罫線集合 (以下これらをまとめて {RL'} と書く) に対し、{CHRC} (= {CHR},{HLICH},{VLICH}) との相互関係は既に求められている。この関係を用いて {RL'} に確信度というパラメータを付与する。ここで確信度を下げられた罫線は文字内の線部分である可能性が高いものである。

最初すべての {RL'} の確信度を 3 とし、{CHRC} との比較によって文字中の線である可能性が高い場合には確信度をマイナスする。減点基準は、

1. 位置関係
   - IN となる CHRC が存在 → -2
   - EDGE となる CHR が存在 → -1
2. 端点状態 (始、終点のそれぞれについて)
   - SHORT の場合 → -2
   - JUST の場合 → -1

減点を累積し、0 以下となった罫線の確信度は 0 とする。

2.5.3　罫線切れの補正

ここでは、原画像のかすれなどによって生じた大きな罫線の切れを補正する。罫線切れがあるとおもわれる部分の全幅 w に対する罫線存在部分の比

$$br = \frac{\sum_i b_i}{w}$$

が閾値以上のとき、注目部分内には罫線が連続しているものとする (図 2.15参照)。

補正する部分のパターンは図 2.15の 4 種に区別することが出来る。タイプ 1 では両側の交差部に罫線の断片があるので、罫線が存在するとして補間すべきである可能性が強い。しかし、断片が片方にしかないタイプ 2,3 両方ともないタイプ 4 となるにつれ補間すべきかどうかは検討を要するようになる。よって補正の閾値にはタイプ毎に異なる値 $th1$～$th4(th1 < th2 = th3 < th4)$ を定めている。

なお、部分的に罫線の確信度が低い場合はそこが文字線である可能性が高い。これを大きく考慮し比の計算の際には確信度の高い罫線のみを用いている。

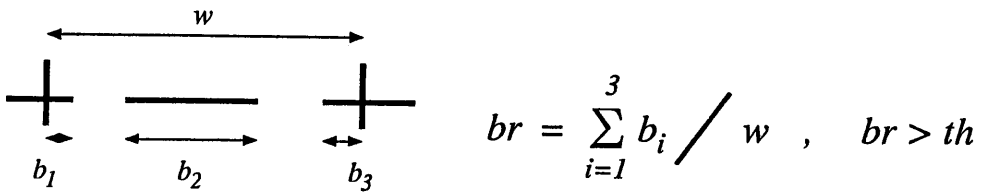

図 2.14: 全体に対する罫線の存在する部分の長さの比

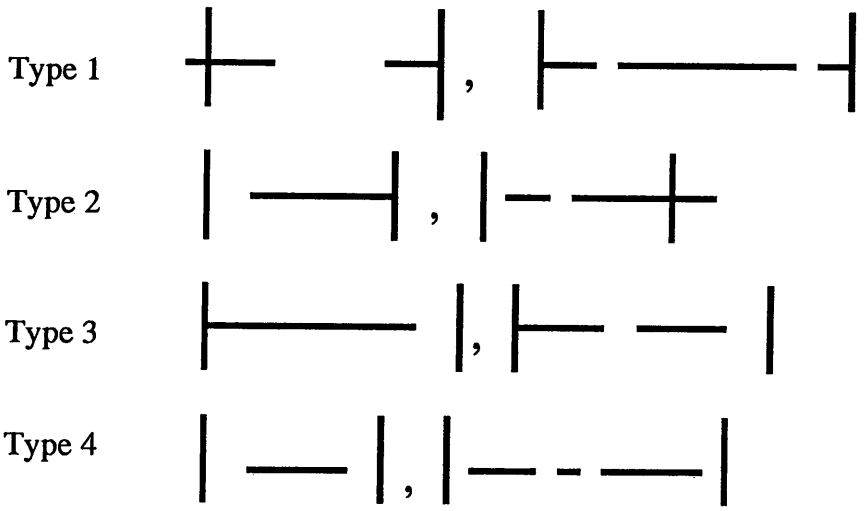

図 2.15: 罫線切れのタイプ 1～4

図 2.15中タイプ 1〜4 で示される罫線補正対象部分は特徴点メッシュ上ではある列 (行) における

Type 1: V E (S E)* S V

Type 2: V E (S E)* V

Type 3: V (S E)* S V

Type 4: V (S E)* V

なるシーケンスに対応している。ここで N* は N の 0 個以上の繰り返しを意味する。よって具体的罫線切れ補正処理プロセスは

1. 行列の行 (列) 中からタイプ 1〜4 のシーケンスを探し、補正をおこなう
2. 1 で補正がおこなわれたとき
   補正によって特徴点のなくなった行 (列) をとり除いて詰めた
   新しい特徴点メッシュを入力にして再び 1. へ
   おこなわれなかったとき
   終了

と書くことができる。

2.5.4　確信度の低い単独罫線の除去

一般に短い罫線、すなわち 1 つの項のみを仕切るような部分罫線は同一列上に複数個存在すると考えられる。これに反して同一列上に一本しかない部分罫線のことを単独罫線と呼ぶことにする。一方、原画像上での文字と罫線の接触等が原因で、文字内線が罫線として誤抽出されている場合、その誤抽出罫線は特徴点メッシュの行 (列) 上では孤立している可能性が高い。これを利用してメッシュ上の任意列の単独罫線に対し、罫線がもつ確信度が低い場合文字内線による誤抽出罫線であると考えて、その罫線を含む特徴点メッシュ上の 1 ラインを除去する。全ラインチェック後、消去されたラインを詰めるように特徴点メッシュを再構成して罫線抽出処理は終了である。

対象としている表では稀ではあるが、短い罫線がひとつの項を仕切るためだけに存在する場合も考えられなくはない (対象外の帳票形式ではむしろ縦線はそのような状態である場合が多い)。このことを考慮して確信度の高い罫線は除去の対象にしない。よって画像品質がよければそのような表の単独罫線でも正しく抽出される。逆に、実際に文字内線が罫線として誤抽出されていてその確信度が低くても、同列に他の罫線がある場合には除去することが出来ないので、罫線抽出誤りの原因となる。

2.5.5　罫線抽出部 (2) の処理例

	全文字数	リジェクト	誤認識 文字間違い	誤認識 フォント間違い	追加テンプレートパターン 初めての文字	追加テンプレートパターン 初めてでない文字	認識率 (%)
data1	3768	282	69		117		90.7
			44	25	80	37	
data2	3475	235	77		88		91.0
			64	13	49	39	
data3	3782	100	64		41		95.9
			42	12	18	23	
data4	3273	71	48		45		96.4
			44	4	11	34	
data5	3363	69	64		30		96.0
			50	14	14	16	
data6	2801	88	48		40		95.1
			47	1	19	21	

図 2.16: 原画像 (図 2.5と同)

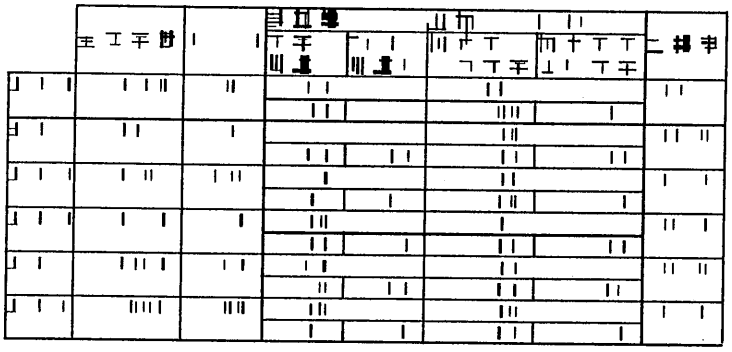

図 2.17: レベル 1〜3 罫線 (図 2.8と同)

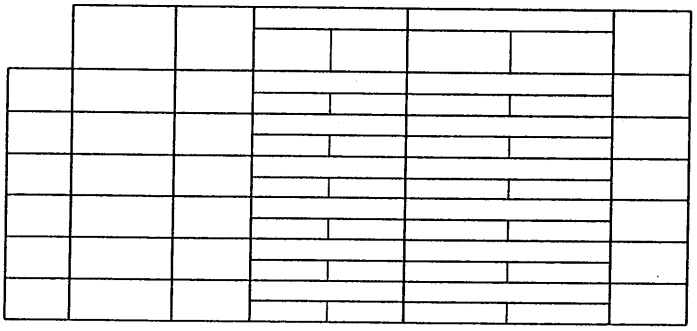

図 2.18: 罫線切れ補正、確信度評価 (単独罫線除去) 後

```
.  L1 T4 T4 HL T4 HL T4 L4
.  VL VL T1 T4 C  T4 T3 VL
L1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
T1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
T1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
T1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
T1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
T1 C  C  C  T2 C  T2 C  T3
VL VL VL T1 T4 C  T4 T3 VL
L2 T2 T2 T2 T2 T2 T2 T2 L3
```

図 2.19: 特徴点メッシュ

2.6　文字ブロックの抽出

表の各項に一対一対応するように、隣接する文字要素矩形 {CHRC} を統合して文字ブロックという文字矩形の集合を作成する。すべての項が罫線で区切られていることが保証されている表であれば、罫線を越えない範囲で 4 方の文字矩形をすべて統合すれば直ちに正しく項に一対一対応する文字ブロックが得られる。しかし現在はそのような保証はなく、しかも現時点では入力された表の画像からそれを判断することも不可能であるため、項間で罫線が省略されている可能性を考えながら文字ブロックの生成をおこなわなければならない。ここでは項内での文字間隔は項間の間隔よりもずっと狭いという、一般的な表に期待されている事実を仮定し、文字要素矩形間距離の近い矩形の統合をおこなう。

2.6.1　縦方向の統合

縦に並んだ {CHRC} のうち以下の条件をみたすものを統合する。

1. 罫線を越えない
2. 距離が v_{vdlim} 以内
3. 統合後の高さが v_{vcmax} を越えない
4. 統合によって近傍の射影における連続値が v_{vcmax} を越えるように変化しない (図 2.20 参照)

ここで

$$v_{vdlim} = 0.5 \times \{\text{CHRC}\} \text{ の平均高さ}$$

$$v_{cmax} = 1.4 \times \{\text{CHRC}\} \text{ の平均高さ}$$

4. における近傍とは、左右に罫線を越えない範囲で ± 4.0× {CHRC} の平均高さ の範囲である。

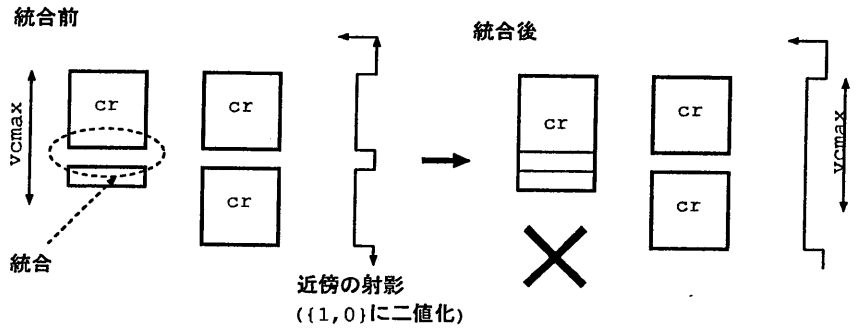

図 2.20: 文字矩形縦方向統合における近傍高さ制限の例

2.6.2　横方向の統合

1. 罫線を越えない
2. 距離が v_{hdlim} 以内

ここで

$$v_{hdlim} = 1.4 \times \{\text{CHRC}\} \text{ の平均幅}$$

2.6.3　形状による統合ルールの追加

項目が小数点を含んだ数値であるとき、その小数点は小さくノイズと見なされやすいので {CHRC} に入ることができず、その部分も文字間隔が閾値を越えてしまうという例が多数見られた。これに対しては次のようなヒューリスティックなルールを追加することによって対処することが出来た。

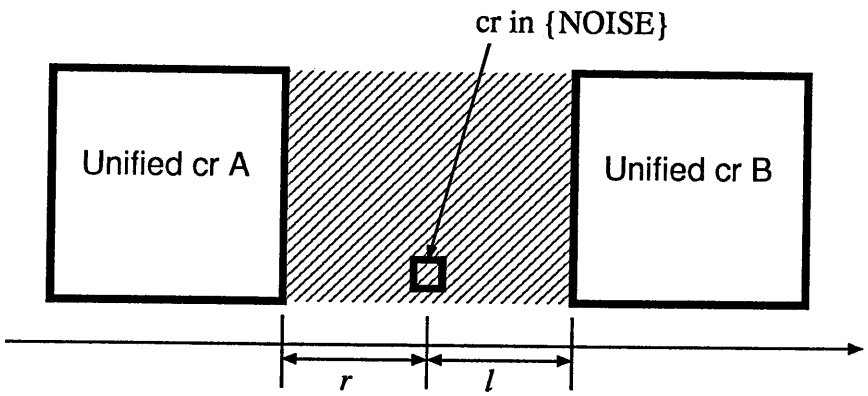

図 2.21: 小数点とみなされる NOISE による文字ブロック統合

図 2.21の斜線領域中に cr ∈ {NOISE} が存在し、

$$r < 2 \times \{\text{CHR}\} \text{ の平均幅}$$

$$l < 2 \times \{\text{CHR}\} \text{ の平均幅}$$

$$0.2 < \frac{r}{l} < 5.0$$

のすべてを満たすとき、これは項中の小数点であるとみなし両側の文字ブロックを統合する。

他にも形状に基づく統合ミスの原因が発見された場合には同様のヒューリスティックを用いて対処していくことが可能である。

2.6.4　統合処理の問題点

統合処理によって得られた文字ブロックは、表の各項に一対一対応していることが期待されている。しかし、項内での文字間隔は項間の間隔よりもずっと狭い、という最初の仮定は筆記者の意図やレイアウト上の都合によって成りたたないことが多く、実際の処理で一対一の対応を実現するのは大変難しい。

特に、項内の文字数によって著しく幅の変化する横方向の統合を正確におこなうのは非常に困難である。ここで、過統合してしまった場合に後の処理で再分離するのは容易ではないので、統合限界距離 v_{hdlim} は小さめに設定せざるをえない。そのためレイアウト上の理由から項内で文字が平均幅の数倍の間隔をとるような項があった場合、1 つの文字ブロックとして統合することが出来なくなる。このように文字が疎な項に対しては、局所的処理よりも上下の項あるいは表全体を考慮した判断が必要であるので、これについて次章の構造理解時に併せて検討することにする。

また、縦方向の統合に関して、本手法では統合後の高さの最大値を文字高さ以下に定義しているので、表の項として複数の文字行でひとつの項をなしているものがある時にこれらをひとつの文字ブロック統合にしようとはしない。この統合判断を正しくおこなうためには文字内容の理解が必要である。

2.6.5　文字ブロック抽出処理例

文字ブロックの抽出処理例を図 2.22～図 2.27に示す。図 2.22～図 2.24 では項に 1 対 1 対応する文字ブロックが得られているが、図 2.25～図 2.27 では文字間隔の広い部分で 1 項が 2 つの文字ブロックに分離してしまっている。

文字ブロック抽出処理例

	A	B	C	D	E	F
試料 A	1.0	0.32	0.38	1.6	0.99	0.50
試料 B	0.36	1.0	1.4	0.64	0.85	0.47
試料 C	0.19	0.69	1.0	0.78	0.56	0.82
試料 D	0.88	0.93	0.67	1.0	2.2	0.27
試料 E	0.44	0.38	1.4	0.38	1.0	0.99
試料 F	1.8	0.92	0.38	0.77	0.53	1.0

図 2.22: 原画像 (200dpi)

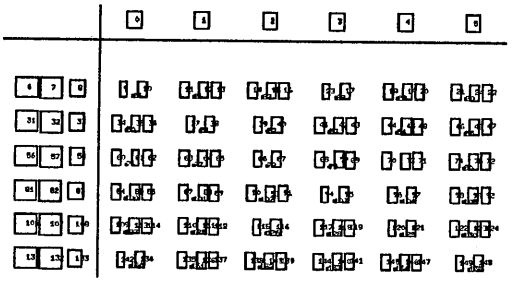

図 2.23: 8 連結矩形抽出

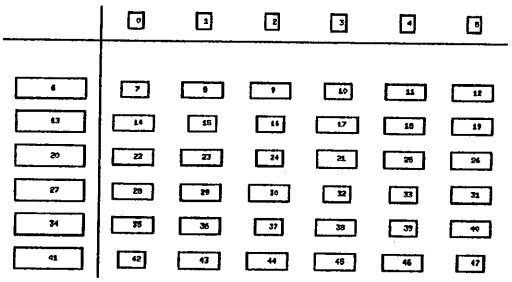

図 2.24: 文字ブロック生成 (成功例)

HMM	対数尤度
バックボレー (正解)	-29.8715
バックストローク	-431.7681
フォアボレー	-233 9752
フォアストローク	-442.1949
スマッシュ	-221.7908
サービス	-466.5597

図 2.25: 原画像 (200dpi)

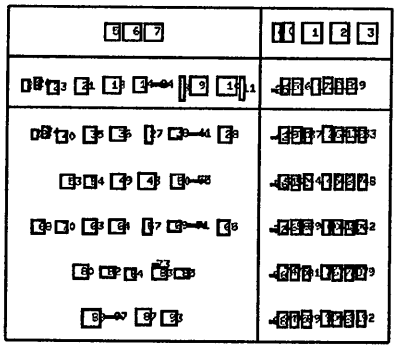

図 2.26: 8 連結矩形抽出

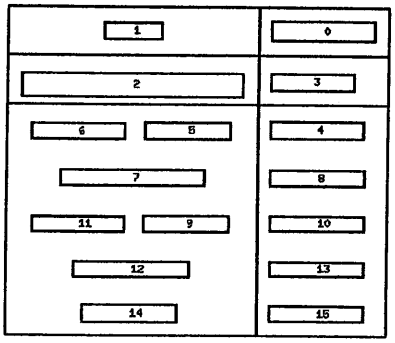

図 2.27: 文字ブロック (統合不十分あり)

2.7　まとめ

本章では画像から表の項を分離する処理として、項のセパレータである罫線と、項そのものを表す文字ブロックとを抽出する手法を具体的に説明した。

罫線抽出部分では、文字内の線なども含んで抽出された罫線候補から、長さと位置関係によるレベル分けを用いた多段階選出処理によって罫線を段階的に選出した。また、表全体の構造から判断される抽出罫線の補正処理として、周囲状態を考慮した罫線切れ補正、および単独罫線の除去処理を、文字矩形との比較で得られる確信度を利用しておこなった。これらの処理により、解像度、及び画像上実際起こりうる多種の例外的変動にも従来手法に比べより広く対応する、正確な罫線抽出をおこなうことができた。

現時点でも誤りを生ずるのは、誤りの要因が二重三重に重なった場合、例えば短い罫線が一行おきに存在する表における罫線のない行で、ちょうど文字中の線が罫線と同列にあってしかも上下の罫線との間隔が狭く罫線として誤抽出されてしまった場合などが考えられる。

文字ブロック抽出部では、文字矩形の抽出の際に線素情報を利用することで、あいまいさを排して正しく文字部分のみを抽出することができた。また文字ブロックへの統合の際には閾値を動的に定めることによって解像度によらない安定した統合性能を実現した。

項内文字間隔が項間隔よりも広かったり、1 つの項が複数の行にわたっている場合など、正しく項と 1 対 1 対応する文字ブロックが常に得られるわけではない。しかしこれは矩形情報だけではなく、文字内容等を見なければ正確におこなえないものであり、後の処理で解決すべき問題である。

第 3 章

表の構造行列の抽出

3.1 まえがき

前章の処理により文字ブロックとして抽出された表の各項は、それぞれ左上と右下の座標という絶対的な位置情報を持っているにすぎない。本章ではこれらの文字ブロックが項として縦横どの項目に属すかという相対的関係を明らかにし、表の論理構造を一意に表わす構造行列を生成する処理について説明する

本処理において入力される表は

(a) すべての項が罫線によって区切られている表

(b) 罫線の省略のある表

の 2 種に分けられる。タイプ (a) の表における項目分けは容易である。項目をなす項の並びは罫線によってあきらかであり、同一の項目となるのは同一の平行 2 罫線に狭まれた項にほかならない。これは前処理で罫線の構造が正しく抽出されていれば直ちに表の正しい論理構造が得られることを意味している。しかし入力された表がタイプ (a) の表である、すなわち「すべての項は罫線で区切られている」という仮定が成り立つかどうかはわからない。そこでが入力表からこの仮定が成り立つ表であるか否かを自動的に判定する処理を認識システムに与えなければならない。

一方この判定処理によってタイプ (b) の表、すなわち罫線に省略があると判定された表に対しては、文字ブロック間の相対関係のみから項を特定し、項目分けをおこなわなければならない。本論文ではその手法として、文字ブロックのヒストグラムを利用する従来的手法の問題点を補った、行対応と平均行構造を用いた項目分け手法を提案している。この提案手法は、対応の最適性を考慮することで、表中の項の省略や包含に対してより柔軟に対応するものである。本章処理の概略図を図 3.1に示す。

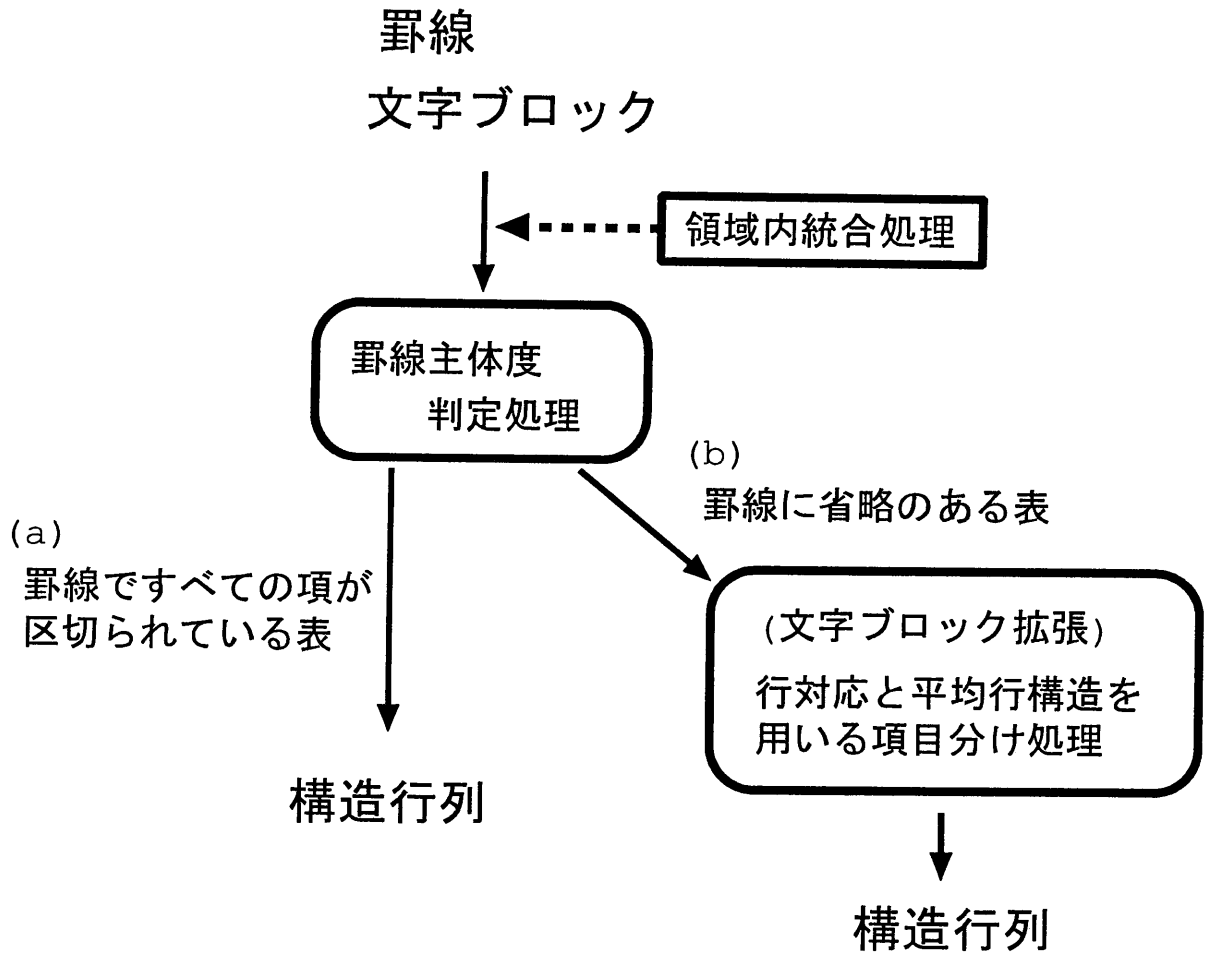

図 3.1: 構造行列抽出部の構成

3.2　罫線ですべての項が区切られている表の項目分け

3.2.1　特徴点メッシュから構造行列への変換

仮定 (a) すべての項は罫線で区切られている

上の仮定 (a) が成り立つことが分かっている表では、まえがきで述べたように、罫線から即その項目分けをおこなうことが可能である。この場合表の構造行列も罫線のなす特徴点メッシュから容易に得ることができる。この変換を簡単に述べる。

特徴点メッシュから構造行列への変換処理

特徴点メッシュを $M_m \times M_n$ とする。項目は罫線で狭まれた領域であるから、縦横の項目分割数はそれぞれ $M_m - 1, M_n - 1$ となり、構造行列は $(M_m - 1) \times (M_n - 1)$ の行列で書くことができる。表の各項は特徴点メッシュ上で罫線の囲む小領域に対応し、この小領域を R_k の左上、右下の座標をメッシュ上でそれぞれ (x_{k1}, y_{k1}), (x_{k2}, y_{k2}) とすると、構造行列上では要素 $\{(i,j)\}$ $x_{k1} \leq i < x_{k2}, y_{k1} \leq j < y_{k2}$ に対応する。小領域 R_k 内の文字ブロック (小領域の実座標範囲内に収まる文字ブロック) をそれに対応構造行列上の要素値 (文字列へのポインタ等) として、表の論理構造を表す構造行列が完成する (図 3.2参照)。

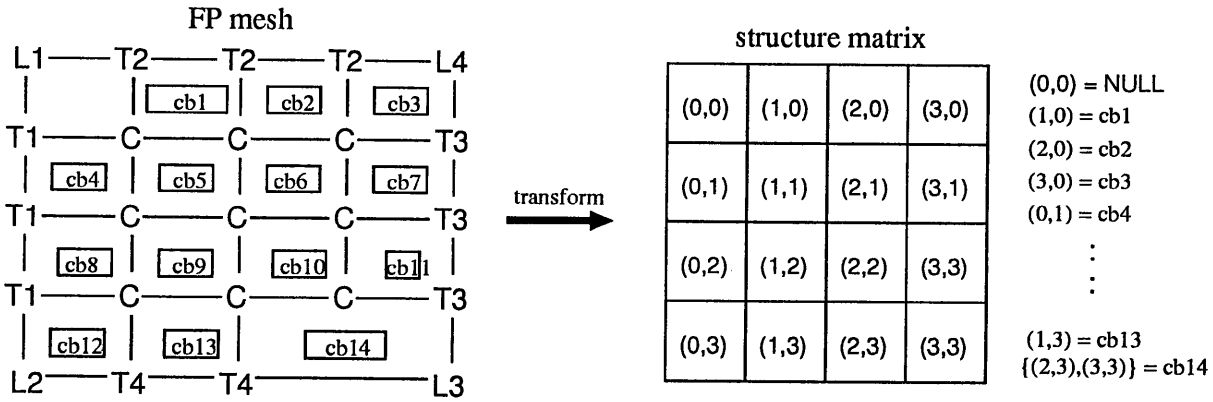

図 3.2: 特徴点メッシュから構造行列

3.2.2　罫線主体度

しかし入力表において仮定 (a) が成り立つかどうかは未知であるから、システムはこれを抽出された文字ブロックと罫線の情報のみから自動的に判断しなければならない。そこで下のような数値を求める。

$$罫線主体度\ f_r = \frac{(全小領域数)\text{ - }(文字ブロックが2個以上ある小領域数)}{(全小領域数)}$$

罫線主体度 f_r は、$0 \leq f_r \leq 1$ の値をとり、$f_r = 1$ のときあきらかに仮定 (a) が成り立つ。よってシステムは $f_r = 1$ のとき入力表をタイプ (a) の表と判断し、構造行列を特徴点メッシュから求めて出力することにする。

3.2.3　領域内統合処理

前節では $f_r = 1$ のとき仮定 (a) が成り立つこととしたが、その逆は成り立たない。すなわち入力表がタイプ (a) の表であっても、抽出された文字ブロックと罫線を用いて計算された f_r は 1 に満たないことがある。これは領域内で

- 1 つの項が複数の行にわたっている
- 文字ブロックの (横方向) 統合ミスがある

ような小領域が存在する場合で、この小領域内の文字ブロックは項としてひとつの文字ブロックに統合することが望まれる。この統合を領域内統合と呼ぶことにする。

しかし入力がタイプ (b) の表であった場合、罫線の省略された部分においては、小領域に複数の文字ブロックがそれぞれひとつの項として存在しているので、この部分は領域内統合してはいけない。現時点では罫線の省略の有無は未知であるので、一般にこの統合が可能かどうかの判断は本来ならば人間のように文字内容を理解してのみ正しくおこなうことができる問題とも考えられる。しかし表全体を見れば、その文字内容を見ずとも文字ブロックの配置からあきらかに 1 つの項であることが妥当と判断できる領域が存在する。本処理はそういった領域の文字ブロックを統合することでタイプ (a) の表として出力できる表を増やそうというものである。

以下に領域内統合処理の具体的手順を示す。

領域内統合処理

注目小領域 R_i 内で (図 3.3中領域 A)

縦に並んだ文字ブロックの数: v_i

横に並んだ文字ブロックの数: h_i

a) $v_i > 1 \wedge h_i > 1 \Rightarrow$ なにもしない

b) $v_i > 1 \wedge h_i = 1 \Rightarrow$

表中で小領域 R_i と丁度同じ高さの帯部分 (図 3.3中領域 B) に注目
帯内に含まれる小領域数を a, うち R_i と同様な (同方向に同数の文字ブロックが並んだ) 領域のを $\{R_{i_1}, R_{i_2}, \ldots, R_{i_k}\}$ とすると

$$a \geq 3 \quad \text{かつ} \quad k/a < 1/3 \qquad \leftarrow \text{判定式 (i)}$$

のとき、$R_{i_j}(j = 1, 2, \ldots)$ 内の文字ブロックを統合

c) $v_i = 1 \wedge h_i > 1 \Rightarrow$

縦横転置して同様

上処理では、領域内統合が可能かどうかの判定を以下のように考えておこなっている。

領域内統合可能判定

注目領域内で分離した文字ブロックひとつひとつが項であるならば、それぞれが項目をなす筈であるから、分離方向と垂直な方向には同様な (同方向に同数分離した文字ブロックを持つ) 小領域が十分数存在する筈である。しかし、同様な小領域が無い、あるいは表の項目数に比べて十分に少ないならば、注目小領域内の分離文字ブロックはそれぞれ単独で項をなすものではなく、ひとつの項が分離したものだと考えて統合することが妥当なので領域内統合を実行する。

分離している文字ブロックのそれぞれが項であるかどうかの判定は (i) 式でおこなっており、項目をなすための領域数は帯状領域内全領域数の 1/3 以上とした。

全小領域について本処理をおこなうことでタイプ (a) の表として処理可能となった例を図 3.4 に示す。

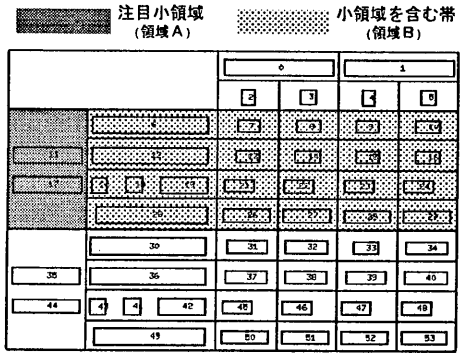

図 3.3: 領域内統合処理過程

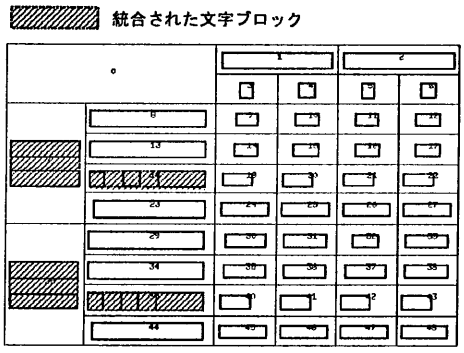

図 3.4: 領域内統合処理結果

3.3　罫線に区切られない領域の項目分け

罫線で仕切られない項に対して、その相対関係を正しく判断し項目分けをおこなうのはたいへんむずかしい。項の幅が文字数やレイアウトよって著しく変化する上に、しばしばみられる項の省略や包含、さらに位置ずれも考慮しなければならないためである。また前段処理の都合上、文字ブロックが正しく項と一対一に対応しているという保証もない。本節ではそのように罫線で仕切られていない文字ブロックに対して構造理解をすすめるため上下左右の項の対応づけをおこなう、すなわち項目分けの手法について検討する。

3.3.1　文字ブロックヒストグラムを用いた文字ブロック拡張

文献 [6] では罫線の省略された表の画像に対して

1. 文字ブロックの並びが表の構造を反映している
2. 罫線は文字ブロックの並びを明確にする

という特徴をとらえて表構造を取得する手法を提案している。

この報告の中では、抽出された文字ブロックに対し、「文字ブロックの拡張」という処理をおこなうことで構造の抽出を可能にしている。処理の概要は以下のとおりである。

文字ブロックの拡張

文字ブロックの拡張を、以下に述べる 2 段階でおこなう。(図 3.5〜3.8参照)

1. **罫線をもとにした拡張 (exp1)**
   次の 2 つのルールにより、文字ブロックの拡張をおこなう。

   ルール 1: 文字ブロックの各辺に平行で最も距離の近い罫線まで、文字ブロックを拡張する。

   ルール 2: ルール 1 で拡張の対象となった罫線の端点まで、文字ブロックを拡張する。

2. **文字ブロックの分布による拡張 (exp2)**
   この処理は縦／横の方向において重なりのある文字ブロックの内で、最も大きい文字ブロックに縁を揃えようよとするものである。

   (a) 図 3.7に示すように文字ブロックの累積分布を求める。

   (b) 分布値を調べ、文字ブロックの縁が揃うように拡張する。

ただし、他の文字ブロックと重複が発生する場合はその方向の拡張をおこなわない。結果を図 3.8に示す。

上記処理による文字ブロック拡張は簡単な処理でありながら、罫線が省略された多くの表に対してたいへんうまく働く。拡張された文字ブロックからは、文字ブロックの並びを座標値の一致として直ちに得られる。よって図 3.9のように、仮想的な罫線を拡張された文字ブロック間に引き実罫線と併せたものからあらたに特徴点メッシュを作り直せば、前節の変換によって表の構造行列を得ることができる。

このヒストグラム手法の優れたところは、項目という帯の内の文字ブロックを得るために、射影という形で帯内全体を巨視的にとらえ、かつこの射影が隣りあう方向に近接する文字ブロックの情報も重畳されているところである。

しかしこの処理はトップダウン的な巨視的処理であるために、対応できない文字ブロックの分布パターンもある。項の位置ずれが激しかったり、包含をなすべき項がその項目と射影が重なっていたりするとヒストグラムから正しく文字ブロックを拡張することが出来ない。また項目の省略や文字ブロックに統合不十分があった場合については対応できない。

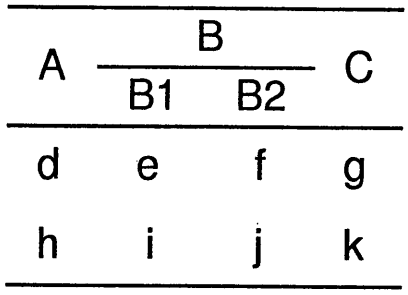

図 3.5: 表の例

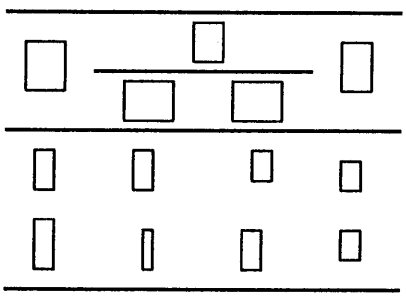

図 3.6: 文字ブロック抽出

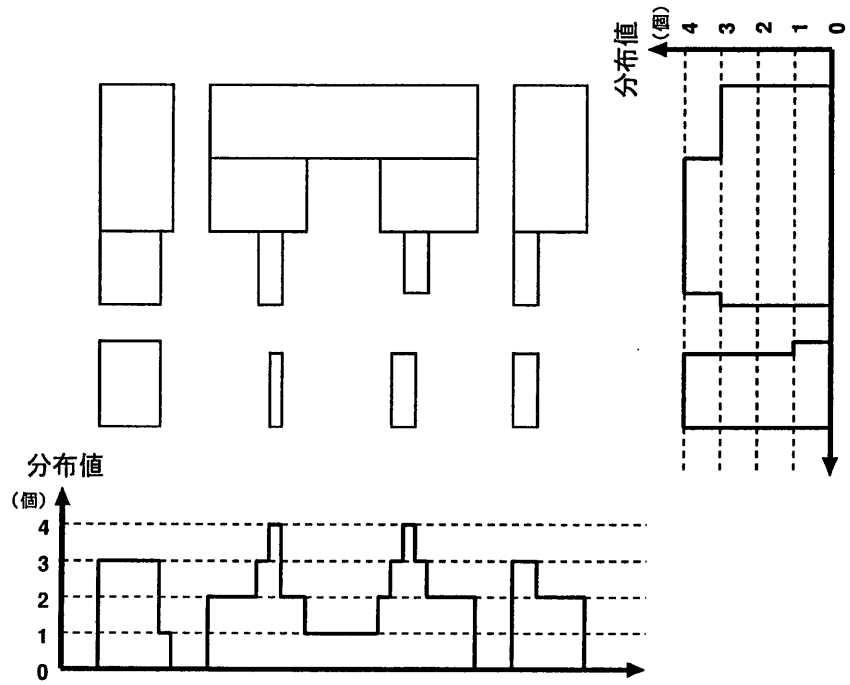

図 3.7: 文字ブロックの分布 (after exp1)

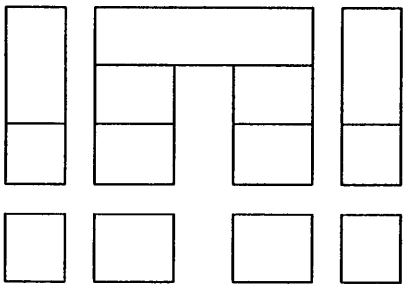

図 3.8: 文字ブロック拡張結果 (after exp2)

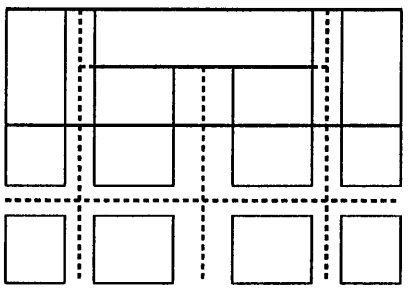

図 3.9: 仮想罫線 (点線) による項分割

3.3.2　行対応と平均行構造を用いた項目分け手法

前述手法によっては対処できない項の位置ずれ、包含、省略といった例外には、トップダウン的な処理だけではなく、その特殊な項に注目した処理が必要である。そのために各文字ブロックの四方の位置関係から相対的に構造を記述してゆくような、極めてボトムアップな処理方法も考えられるが、様々な例外には個々に対応していかねばならないので、組み合わせ数が増大する等して見通しがたちにくい。特に、微妙な変動で結果が大きく変ってくるという欠点もあり、実現は困難である。

そこで、特殊な例外項に対しても個々の項へ注目することによって柔軟に対応しつつ、かつ全体的な視野から項の並びを取得するために、処理単位を行とする手法を提案した。以下にその概要を述べる。

行対応と平均行構造を用いた項目分け手法

表を図 3.10のように、横方向に並んだ項の集合を帯状に「行」とみなしてして分割する。システムが対象としている表は縦方向にも項目をなす項の並びがあるので、分割された任意の 2 つの行では、大見出しなど一部を除いて、双方に含まれる項の間に同一項目としての対応関係 (図 3.11) があると考えられる。但し実際の処理では、項に相当する文字ブロック間の対応関係を取り扱う。

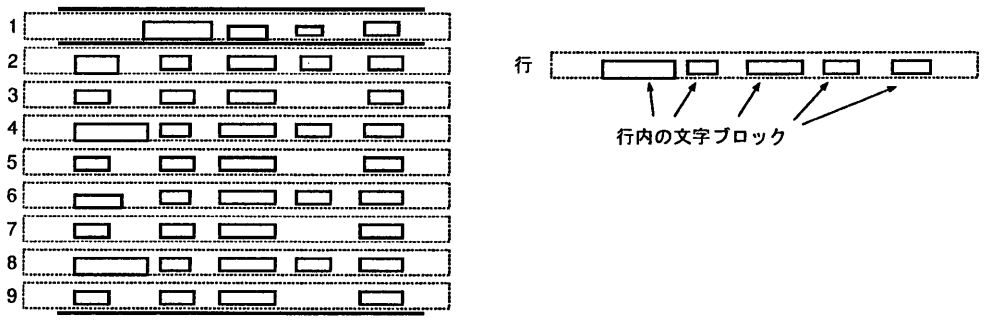

図 3.10: 行分割の例

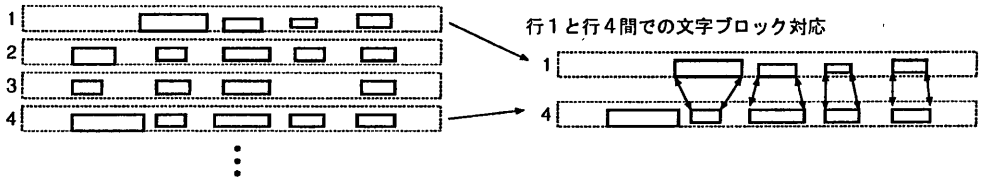

図 3.11: 2 行間の対応の例

統合ミスによる複数の文字ブロックが他行の項である 1 つの文字ブロックに対応するという場合や、項に省略や包含がある場合も想定しているので、文字ブロック間の対応は必

ずしも 1 対 1 である必要はなく、それぞれ 1 対 2、1 対 0 など様々な組み合わせを考えることもできる (図 3.12)。これらを用いて 2 つの行内の文字ブロックどうしの最も妥当な対応付けが求められれば、その対応をもって 2 行間での項目分けとすることが出来るであろう。これを行間の文字ブロック対応、もしくは行対応と呼ぶことにする。

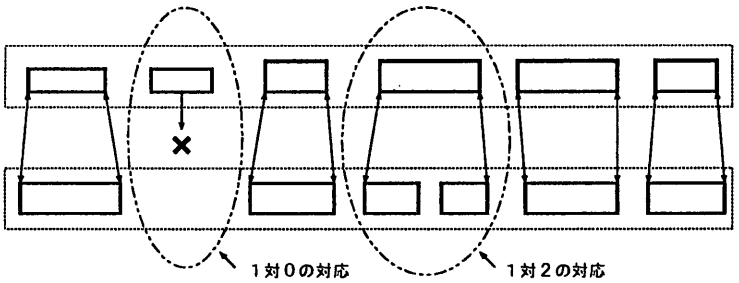

図 3.12: 2 行間の特殊対応の例

さらに任意の 2 行の対応づけのみならず、この対応を利用して表中で矛盾しない対応づけの出来る行をすべて集め、それらがもつ文字ブロックの構成を平均化することにとよって、その表における平均的な行構造 (＝平均行構造) というものを定めることが出来る。

その上で任意の行の構造を表の平均行構造との対応づけによって決定することにする。このときに前述の最適対応を求める仕組みを利用すれば、問題となっていいた特殊な項に対しても局所的に最適な対応が得られる筈である。また平均構造と比較しているので、巨視的な視点での判断を加えることが出来る。

なお、現在は横書きの表を対象にしているので、各項の幅はその内容文字数、文字種類によって大きく変わり、表内で一定ということは少ない。文字ブロックの横方向開始、終了位置が揃っている可能性も同様の理由で低い。ところが項をなす文字ブロックの高さは表で使用されているフォントによってほとんど決まってしまう上、印刷の都合上項目内の位置が高さ方向に変化することもほとんどない。この理由から、一般に表を縦方向に分割することは難しく、横に分割するのは比較的容易であると考え、具体的な構成でも横方向に簡略な処理 (文字ブロック拡張処理) で分割した行を用いて上記手法で縦方向の項目分けを詳細におこなうことにした。

以上が提案手法の概要である。これからその具体的な適用方法について

(1) 表の行分割

(2) 行間の文字ブロック対応

(3) グループ化と平均行構造の作成

(4) 平均行との対応による各行の修正

の順に説明する。

(1) 表の行分割

入力は、拡張された文字ブロックの集合 (元の文字ブロックの大きさも記憶してある) と罫線の構造をあらわす特徴点メッシュである。拡張された文字ブロックすべてについて、隣り合う文字ブロック間のちょうど中央に境界線をひく。但し文字ブロック間に罫線が存在する場合にはそこを境界とする。各文字ブロックの四方に引かれた境界線 (のうち実罫線ではないもの) を「補助罫線」と呼ひ、特徴点メッシュに追加する。

補助罫線を追加した特徴点行列の例を図 3.13に示す。ただし見易さのためにここでは行列上の特徴点は元画像での座標の位置に示してあり、実罫線を細い実線、補助罫線を 2 点鎖線で書き足している。参照のため文字ブロックと拡張文字ブロックが位置に応じて書き入れてある。

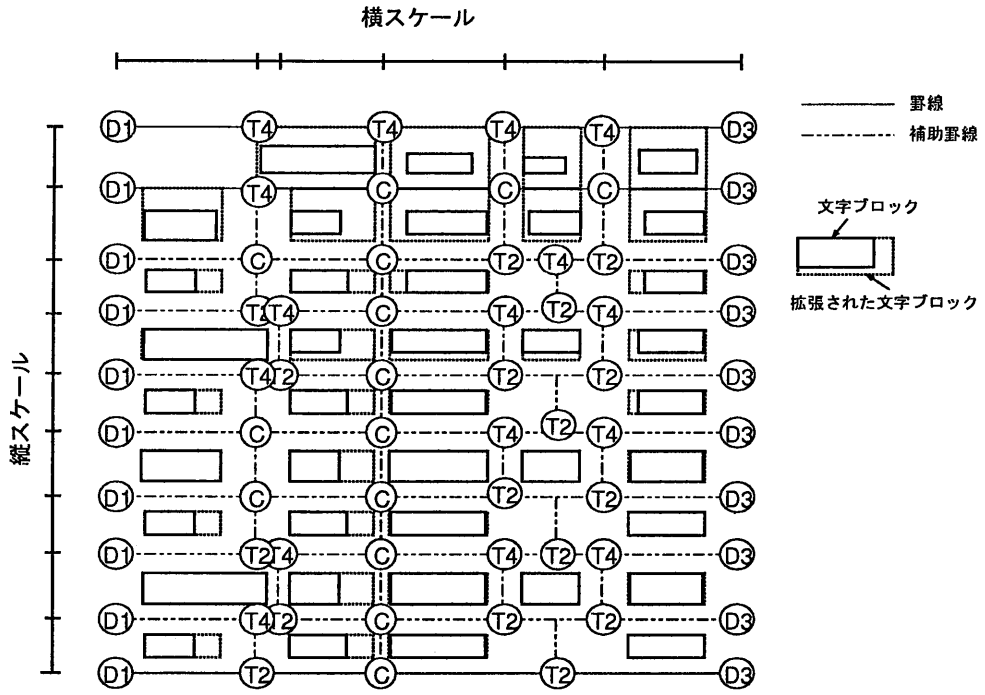

図 3.13: 補助罫線を追加した特徴点行列

文字ブロック拡張処理によって行内の文字ブロックが高さ方向に正しく並んでいると考え、このメッシュから横方向の帯 (＝行) の分割をおこなう。この分割処理は

- 必ず 1 個以上の文字ブロックを完全に含む。
- 縦に 2 つ以上の文字ブロックを含まない。

という規則にもとづいておこない、実際の分割の様子は図 3.10のようになる。但し実際には横方向のメッシュに従って切るだけでは、下図 3.14のようにひとつの文字ブロックの射

影が縦2つ以上の文字ブロックに重なっている場合、上の条件を満すことが出来ない。これについては図 3.14のように特別な扱いを施して強制的に分割し、共有している文字ブロックに関しては対応づけのときに優先対応するようにしている。

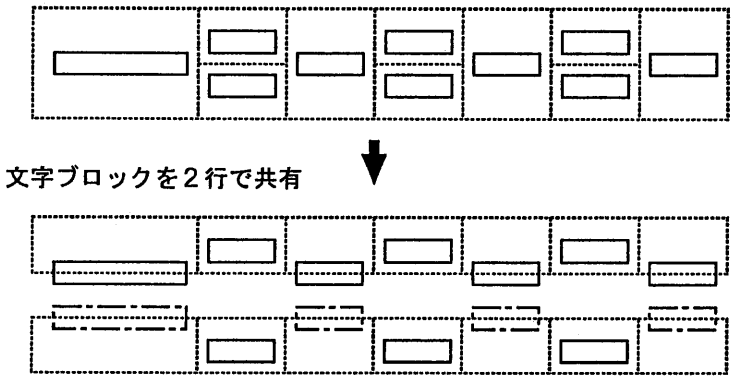

図 3.14: 文字ブロックを共有する行の強制分割

(2) 行間の文字ブロック対応

行間の文字ブロック対応は、2行内すべての文字ブロックの対応づけであり、2行の項の項目分けに相当する。このとき文字ブロックの対応づけが大きさや左右の要素も考慮して最適に行なわれれば、項の位置ずれ、省略、包含に正しく対応する項目分けができる筈である。この最適な対応を実現するために、行間の類似度を次のように定める。

行間の類似度

2行の内の文字ブロックを任意の組み合わせで対応を行なって個々の文字ブロック間の対応の得点の総和を求めたとき、その最大値を行間の類似度と定義する。

また最大の類似度を得るときの文字ブロック対応を最適対応という。

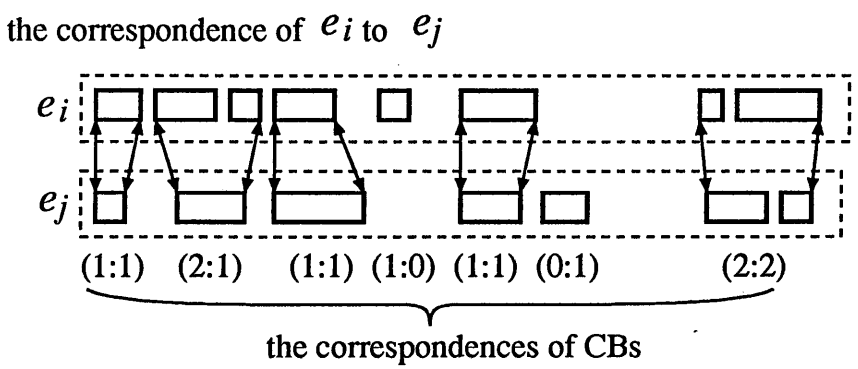

図 3.15: 行対応と文字ブロック (集合) 間対応

次に個々の文字ブロック間の対応の得点を定義する。ここで文字ブロックの対応づけは図 3.15のように 1 対 1 に限らず 1 対 0〜n、0〜n 対 1、2 対 2 等の種類を考えることによって項の省略、包含や文字ブロック統合ミス等に対処している。複数対 1 の対応を含める意味で、文字ブロック集合間の対応の得点と言い直し、以下のように定める。

行 i 中の文字ブロック集合 C_x、行 jの文字ブロック集合 C_y間の対応の得点

$$\{C_x\} = \{c_{i_1}, \ldots, c_{i_k}\}$$
$$\{C_y\} = \{c_{j_1}, \ldots, c_{j_l}\}$$
$$(k, l) = (0, \{1, 2, \ldots\}),\ (\{1, 2, \ldots\}, 0),\ (2, 2)$$

1. 制限領域による対応のペナルティ

対応が重複したり、罫線をはみだすような対応を避けるために (あるブロックが罫線をはさんだ2ブロックと対応するのは包含として認める)、制限領域によるペナルティを設け、ペナルティが課される対応には得点を与えないようにする。この制限領域は

1:0 の対応のとき

行 j 内で $(xmin(C_y), xmax(C_y))$

0:1 の対応のとき

行 i 内で $(xmin(C_x), xmax(C_x))$

それ以外のとき

行 i 内で

$(xmin(C_y) + w, xmin(C_x))$ 但し $xmin(C_y) < xmin(C_x)$ のとき

$(xmax(C_x), xmax(C_y) - w)$ 但し $xmax(C_x) < xmax(C_y)$ のとき

行 j 内で

$(xmin(C_x) + w, xmin(C_y))$ 但し $xmin(C_x) < xmin(C_y)$ のとき

$(xmax(C_y), xmax(C_x) - w)$ 但し $xmax(C_y) < xmax(C_x)$ のとき

の領域であり、行内での高さ方向の座標は考慮しない。各行でこの領域に文字ブロックあるいは罫線が存在するときペナルティを +1 する。実際には図のように平均文字幅の 1/4 の許容幅 wを設けている (図 3.16)。

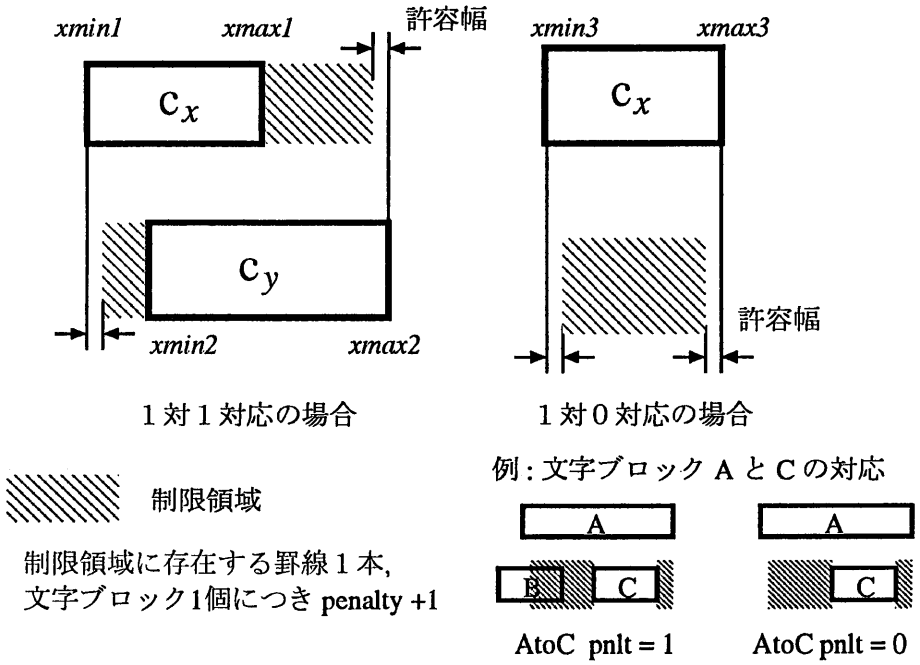

図 3.16: 対応に対する制限領域とペナルティ

このペナルティを用いて、文字ブロック (集合) 間の対応の得点を

$$simirality(\{C_x\} \leftrightarrow \{C_y\}) = \begin{cases} q \times (\underbrace{|\{C_x\}| + |\{C_y\}|}_{\text{対応内文字ブロック総数}}) + \underbrace{p}_{\text{最適性のための加点項}} & (pnlt = 0) \\ 0 & (pnlt > 0) \end{cases}$$

と定める。但し p は後述する最適性のための加点項であり、q は全体の得点において文字ブロック数が加点項に対して支配的であるために十分大きい係数である。

$p = 0.0$ の時、対応の得点の行全体での総和が最高になるように (同点で多種の対応が存在する場合は対応の数が大きくなるように) 対応を決定することで、重複などの矛盾のない対応が得られる。しかし、ペナルティのない複数の対応 (1 対 2 or 1 対 1 と 1 対 0 など) に対する最適性はない。

2. 位置関係の導入による最適性の強化

最適な対応を得るためにはその最適性の評価を得点に導入する必要がある 。図 3.17は最適性の 1 例であり、Pattern1〜3 は 1 個対 2 個の文字ブロック間における望ましい対応が、上の行の文字ブロックの位置によって変わる様子を示している。

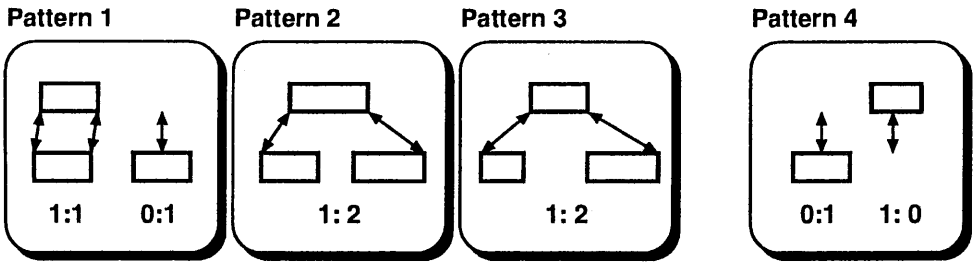

図 3.17: 最適な対応の選択

この図 3.17では

ルール 1:「項目をなす項はその中央の座標 (重心) が接近している」

が成り立つような項の対応の最適性を説明していると言える。そこでこのルール 1 を採用し、かつ図の Pattern4 のような場合も正しく対応を選択できるように、以下のような加点項を設定し、対応の得点へ加えた。

最適性のための加点項 $p \rightarrow [0.0 \sim 1.0]$

$$p = \begin{cases} 1.0 & (d_g < -\gamma) \\ (d_g + \gamma)/(\gamma - \delta) + 1 & (-\gamma \leq d_g \leq -\delta) \\ 2.0 & (-\delta < d_g < +\delta) \\ -(d_g - \gamma)/(\gamma - \delta) + 1 & (\delta \leq d_g \leq -\gamma) \\ 1.0 & (\gamma < d_g) \end{cases}$$

$d_g = g_x - g_y$

$\gamma = min(w_{C_x}/4, w_{C_y}/4) \in [2 \times \widetilde{CW}, 3 \times \widetilde{CW}]$

$\delta = \widetilde{CW}/4$

w_{C_x}, w_{C_y}: C_x, C_yの幅, g_x, g_y: C_x, C_yの重心座標

$\widetilde{CW}$: 文字の平均幅

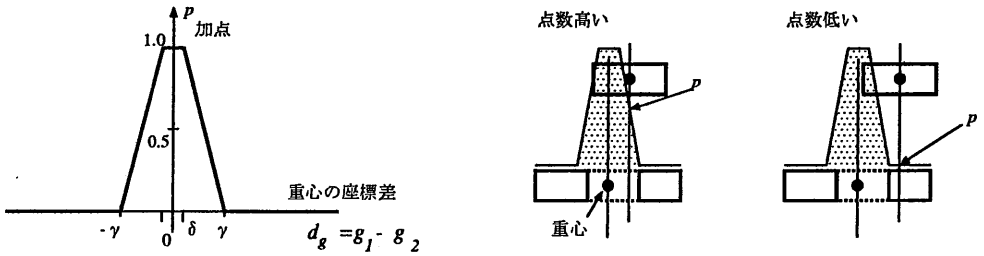

図 3.18: 重心接近優先の最適性を示す加点

(3) グループ化と平均構造行の作成

平均的な行構造を求めるために表中の全行をグループ化し、そのグループごとの代表的な構造を決定する。まず、お互い矛盾しない構造をもつ行どおしをグループとしてまとめるグループ化処理のアルゴリズムを示す

行のグループ化

input:　行の集合 $U^0 = \{L_1, L_2, \ldots, L_m\}$

　　行間類似度の集合 $D = \{(L_1, L_2), \ldots, (L_j, L_k), \ldots\}$

output: 同一構造の行グループ $G^0, G^1, \ldots,$

　　$G^i = \{L_{i_1}, \ldots, L_{i_p}\}$

Process

1. U^iを行内文字ブロックの多い順に、
   Dを得点の高い順にソート
2. if $U^i = \emptyset$　end.
   else
   $U^i \leftarrow U^i - L_j$ (L_j is a head of U^i)
   $G^i = \{L_j\}$
3. while $D \neq \emptyset$
   $d = (L_a, L_b)$ (= head of D), $D \leftarrow D - d$
   cond
   1) $L_a \in G^i$ *and* $L_b \in U^i$
   $\{G^i + L_b\}$ 内で矛盾する対応があるか？
   yes:　do nothing.
   no:　$G^i \leftarrow G^i + L_b$, $U^i \leftarrow U^i - L_b$
   2) $L_b \in G^i$ *and* $L_a \in U^i$
   1) 同様
4. $U^{i+1} \leftarrow U^i - G^i, i++$
   go to 2.

ここで構造が矛盾しないとは、行 a の i 番目の文字ブロックを c_{ai} と書くとき、

$$c_{ai} \leftrightarrow c_{bj}, \quad c_{ai} \leftrightarrow c_{ck} \quad \text{ならば} \quad c_{bj} \leftrightarrow c_{ck} \qquad (\leftrightarrow \text{は対応を示す})$$

がすべてのグループ要素行に対して成りたつことを言う。

アルゴリズムから明らかなように、グループ化する行間の文字ブロック対応はすべて 1 対 1 である必要はなく、省略や包含を含んですべての矛盾しない対応をつくれる行どうしが 1 つのフループにまとめられる。

グループ内での代表的構造行の作成

次にグループの平均的な構造をもつ行 (グループ代表行) を作成する。グループ内で、各行が持つ文字ブロックの行間対応によって始点、終点の座標の組ができるので、これらの組の中で座標の平均値を算出し、これをグループ代表行における文字ブロックの始点、終点と考える。グループ内の文字ブロックが行間ですべて一対一対応しているならば、各始終点の組の要素数はグループ数に等しく、すべての始終点からつくる文字ブロックを代表行の文字ブロックとできる。しかし項の省略、包含、統合ミス等により 1 対 2、1 対 0 等の対応が生じていると、始終点の組の要素数にグループ数に満たないものがでてくる。代表行はグループ内の主要構造をあらわす行であると考えることができるからこのような始終点を採用するかどうかは、その組が作る項がグループ内で普遍的であるかどうか判断して決めるべきである。この考えから、グループ内で全行と 1 対 1 対応していない項については、グループ要素数を k としたとき 1 対 1 対応する項をもつ行の数が $max(k/3,\ 2)$ 以上のものをグループ内に普遍的な項と認め平均構造に採用することにした。

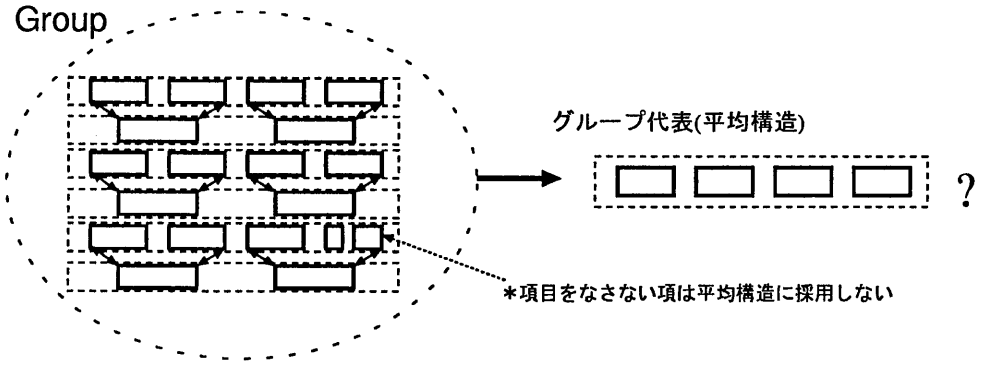

図 3.19: グループ化とグループ代表

表の平均的行構造の作成

1 回のグループ化では位置ずれ等によって 1 つにグループ化されるべき行が 2 つ以上のグループとなってしまうことが多い。よって 1 回目でグループ化された行の代表行に対して再度グループ化、代表行作成をおこなう。これによって得られたグループのうち、要素数 (行数)3 以上のグループの代表行を表の平均的行構造とする。

平均的行構造の例を図 3.20に示す。文字ブロック間に罫線がない部分について両ブロックの中央に仮想的な補助罫線をひき、実罫線と併せて項の境界としている。

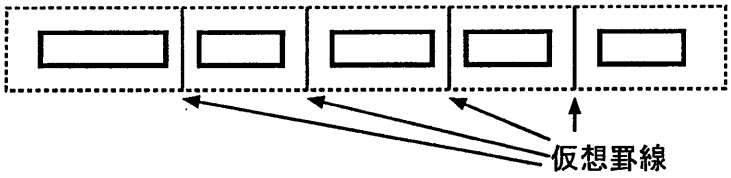

図 3.20: 平均的行構造の例

(4) 平均行との対応による各行の修正

実際の各行に対し平均構造との対応づけをおこなってゆく。各行の文字ブロックで平均構造の文字ブロックとの対応がとれたものは、その左右の仮想罫線の位置を平均構造のものと一致するように修正する。平均行構造の 1 個の文字ブロックに対し、注目行の 2 個以上の文字ブロックが対応する時はこの 2 個の文字ブロックが統合ミスによるものとし、1 つの項に統合する (但し罫線を狭む場合は除く)

但し、この対応づけは各行においてその行が平均構造をなすグループのどれかの要素であるような場合はその平均構造に対しておこなう。どの平均構造の要素でもない場合はすべての平均構造と類似度を計算して最大の相違度を持ちかつその値が閾値をうわまわる場合のみその平均構造に対しておこなう。閾値を下まわる類似度の場合は孤立行 (大見出しなど) の可能性が強いので修正はおこなわない。

この修正によって得られた仮想罫線を実罫線に加えて、再び特徴点メッシュを書き直す。全行が正確に修正されれば、項目分けが罫線のつくる領域分割によって正しく行えるような特徴点メッシュが生成される。前の図 3.13の特徴点メッシュに対し図 3.20 の平均構造を用いて各行ごと修正をおこない仮想罫線引き直した後の特徴点行列の様子を図 3.21に示す。省略や位置ずれに対しても正しい項目分けがおこなわれたのがわかる。

最終的にこの特徴点行列から作成される構造行列が、タイプ (b) の表に対する認識結果として出力される。

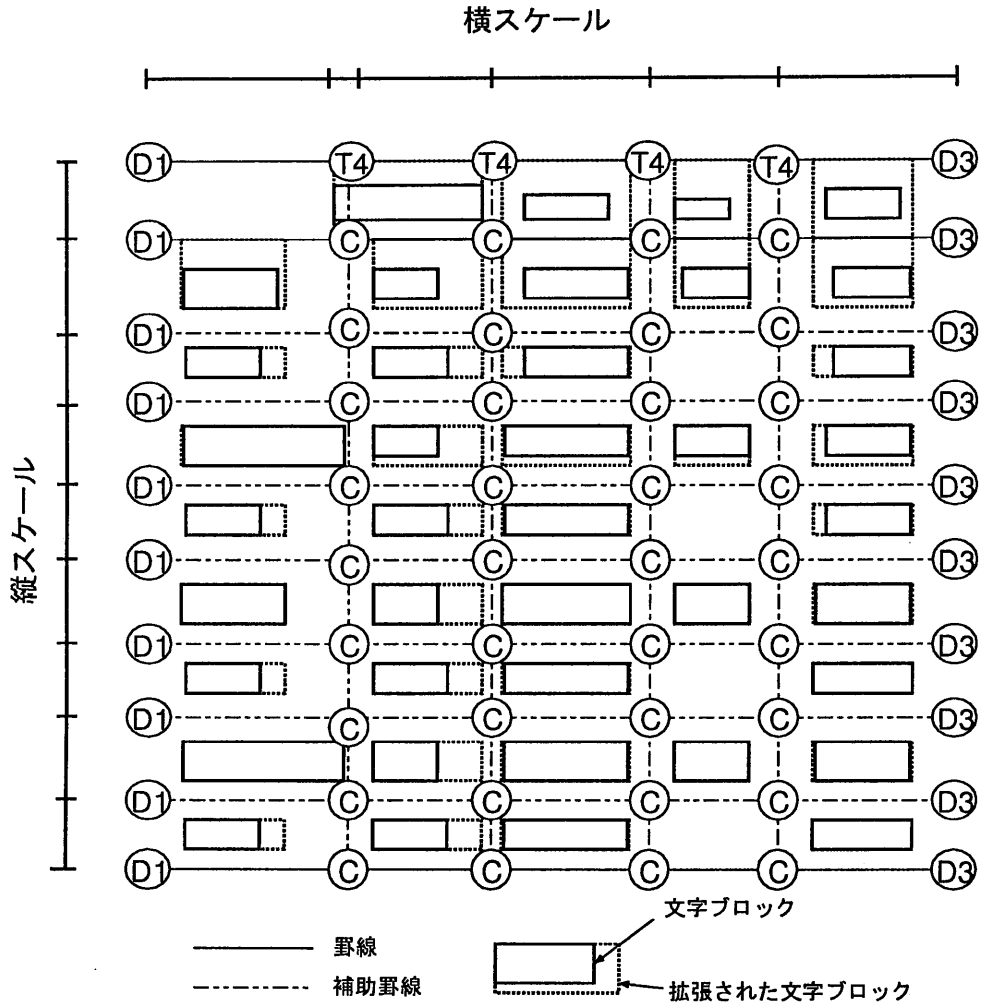

図 3.21: 平均的行構造によって修正された特徴点行列の例

3.4　処理例と考察

本章の処理の実例を示し、問題点などの考察をおこなう。

タイプ (a) すべての項が罫線によって区切られた表

入力表がタイプ (a) の表と判断されて正しく構造行列が出力された例を図 3.22〜図 3.25 に示す。このうち図 3.22→図 3.23は罫線と文字ブロックが抽出段階で実際の項と 1 対 1 で得られた例、 図 3.22→図 3.23は 領域内統合処理が複数の文字ブロックからなる項を正しく統合しタイプ (a) の表として出力に成功した例である。

一方図 3.26→図 3.27は領域内統合ができなかったためにタイプ (a) の表として出力できなかった例である。このような部分の統合は文字内容の意味を見てそのまとまりを理解せ

ねばおこなうことができない。

タイプ (b) 罫線に省略のある表

罫線の省略された部分に仮想罫線を補って正しく構造行列が出力された例を図 3.28～図 3.36に示す。3.28→図 3.31では項の省略、位置ずれ、図 3.32→図 3.36では項の包含に正しく対処している様子が示される。

一方図 3.38～図 3.40は本手法が対応出来ない表の例である。

3 項目以上にまたがる大見出し　(図 3.38→図 3.37)

本手法では 1 対 2 の対応に最適性が得られるようなルールで類似度点数を定めており、1 対 3 以上の対応では最適性が得られない。特に類似度得点が等しいときは包含よりも省略が優先される為、図 3.38のような場合、3 つを包含すべき見出しが 2 つの省略と 1 つの 1 対 1 に対応してしまう。これをすべて包含に処理しようとすれば今度は省略に対応できなくなってしまう。

この解決法としては、例のような 1 対多の対応は見出し部に多いということを利用し、見出し部とそれ以外では対応の最適性のルールを変えることが有効であろう。しかしそれには入力表から自動的に見出し部を特定するための高レベルな処理が必要になる。

縦方向の包含　(図 3.39→図 3.40)

本章で説明した手法は項の縦方向の特殊な対応に対し、文字ブロック拡張で処理できるような射影の包含以外は未処理である。これは横書き表の性質上、特に罫線の省略された表では縦の項の包含などはおこりにくいと仮定したためである。しかし見出し部に関してはこの仮定があてはまらず、例のような表も多数存在する。縦方向の項目分けに横と同様な処理を用いて対処することもできるが、縦の包含のほとんど見出し部にみられることから、前ケース同様見出し専用のルールを適用するほうが効果的であろう。そのためには見出し部の特定をあらかじめおこなわなければならない。

罫線に省略のない表からの構造抽出処理例

タイプ (a) 表、成功例

from	to			
	1	2	3	0
1	0.595	0.076	0.101	0.228
2	0.121	0.742	0.136	0.000
3	0.005	0.141	0.552	0.302

図 3.22: 原画像 (1)

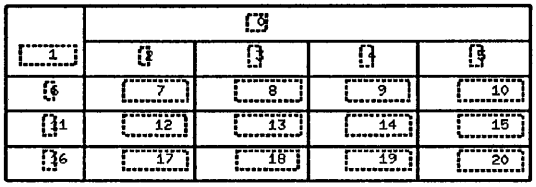

図 3.23: 出力構造行列

		ユークリッド距離		シティーブロック距離	
		甲	乙	甲	乙
方向線素特徴量	領域半径外 (字)	14	16	21	29
	誤認識数 (字)	23	26	25	34
	回復率 (%)	60.9	61.5	84.0	85.3
	1 位認識率 (%)	99.922	99.912	99.916	99.885
方向密度ベクトル	領域半径外 (字)	162	620	86	639
	誤認識数 (字)	233	994	106	830
	回復率 (%)	69.5	62.4	81.1	77.0
	1 位認識率 (%)	99.214	96.648	99.642	97.201

図 3.24: 原画像 (2)

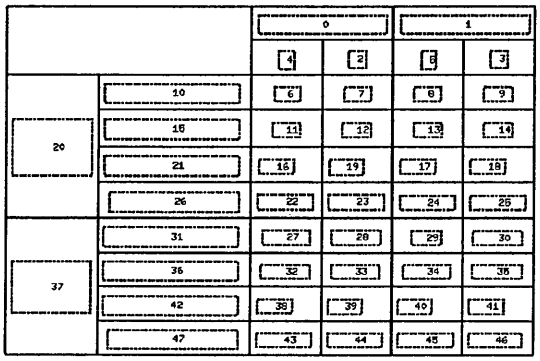

図 3.25: 出力構造行列

タイプ (a) 表、失敗例

使用する情報		結合処理 (957 文字)	優先切り出し＋結合処理 (3060文字)
単独	矩形評価値	87.5%	95.6%
	類似度	64.6%	88.4%
	差分類似度	88.9%	96.0%
併用	矩形評価値＋差分類似度	94.4%	97.7%

図 3.26: 原画像 (3)

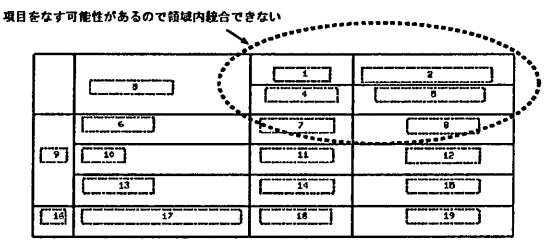

図 3.27: 出力構造行列 (誤)

罫線の省略がある表の項目分け処理例 (1)

	Indexing	steps	rate	time
prime	ON	95267	1.02	1690
ime	OFF	93324		1680
prime_dd	ON	153791	0.93	3570
imd	OFF	165750		3740
queen	ON	574499	0.95	12330
ued	OFF	603677		12420
lbvpascal	ON	433201	0.65	11360
bce	OFF	664138		12590

図 3.28: 原画像

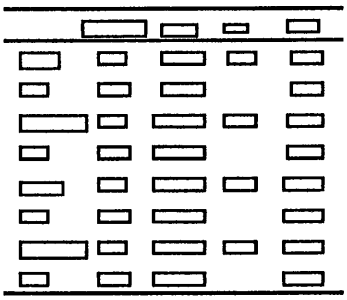

図 3.29: 抽出文字ブロックと罫線

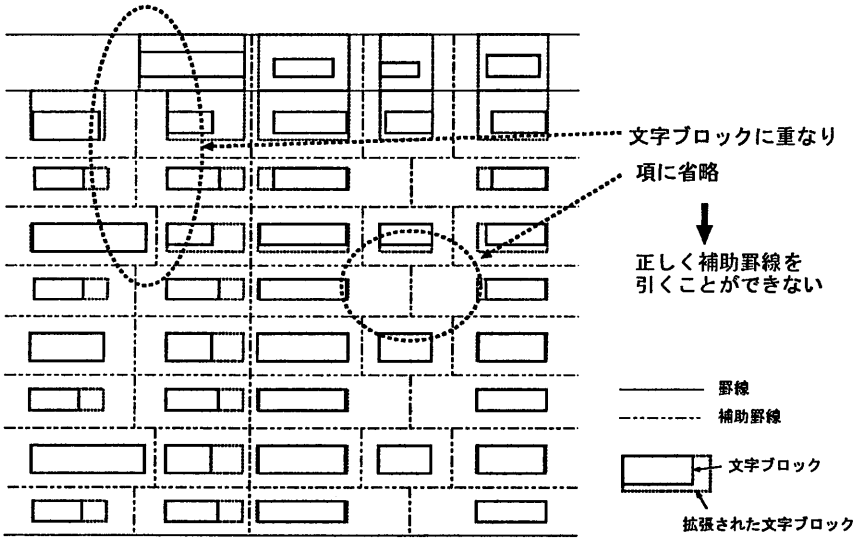

図 3.30: ヒストグラムによる文字ブロック拡張

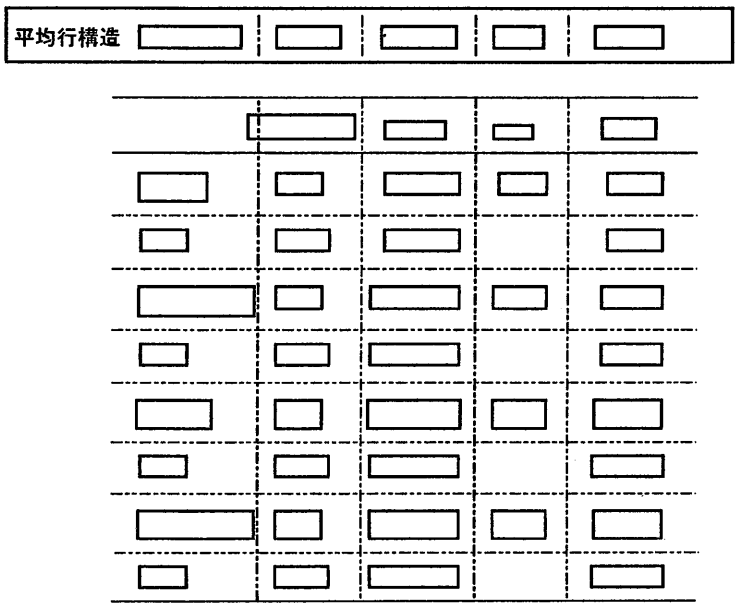

図 3.31: 平均行構造 (上) を用いて正しく引かれた補助罫線

罫線の省略がある表の項目分け処理例 (2)

	Indexing		Coding		Total	
	SX	DX	SX	DX	SX	DX
pre1	36,800		12,400		49,200	
post1	31,000	4,700	8,800	2,900	39,800	7,600
pre2	22,400		3,800		26,200	
post2	13,200	5,500	18,800	2,900	32,000	9,400
pre3	100,800		22,400		122,200	
post3	80,000	16,700	14,800	3,900	94,800	20,600

図 3.32: 原画像

図 3.33: 文字ブロック拡張のみでは誤りあり

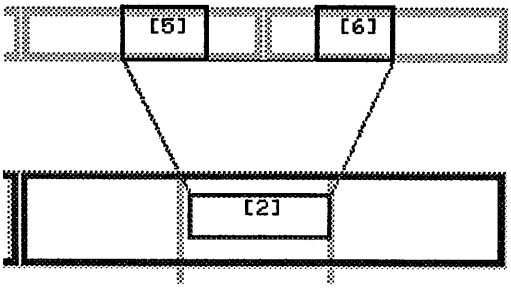

図 3.34: 平均構造 (上) と行 1 の対応

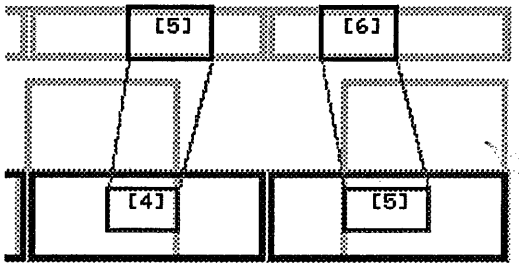

図 3.35: 平均構造 (上) と行 2 の対応

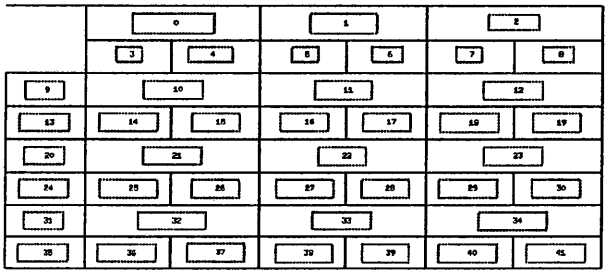

図 3.36: 正しく得られた補助罫線

罫線の省略がある表の項目分け処理例 (3),(4)

サーバ	CASE1		
	厳密	近似	誤差
1	1.17	1.15	+1.7%
2	1.15	1.15	+0.3%
3-1	1.15	1.15	+0.3
3-2		1.15	-0.0%
4	1.14	1.15	-0.9%

図 3.37: 原画像

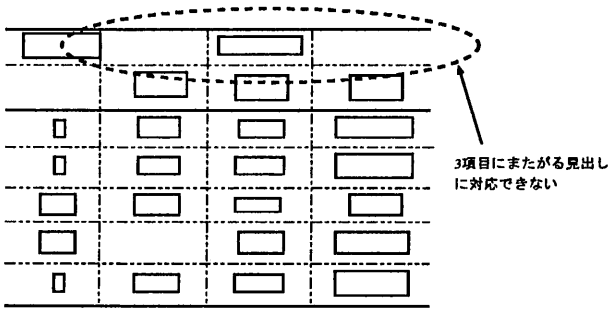

図 3.38: 出力構造行列 (誤)

クラスタ/カテゴリ	順位	Ward	k-means
	1位	75.09	74.58
2	2位	84.47	84.24
	3位	88.24	87.83
	1位	75.33	75.33
5	2位	85.13	84.81
	3位	88.52	88.76

図 3.39: 原画像

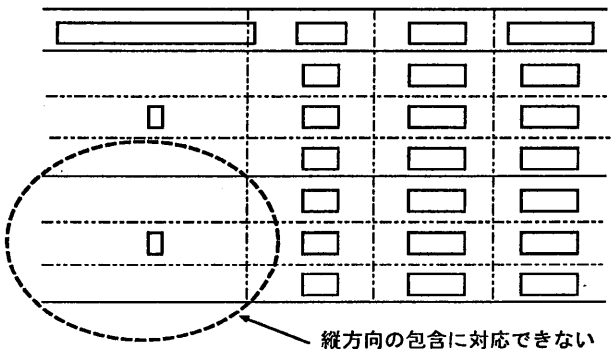

図 3.40: 出力構造行列 (誤)

3.5　まとめ

本章では 2 章で抽出された項の情報である、罫線と文字ブロックを用いて表の構造行列を作成する処理について検討をおこなった。まず形態が未知である入力表に対して、罫線主体度という評価値から自動的に罫線省略の有無を判定した。この判定により罫線の省略のない表と判断された場合は、罫線の構造からただちにその構造行列が得られる。また、罫線に省略はないが、項が複数行からなっていたり、文字ブロック統合ミスがあったりして罫線が囲む領域内に複数文字ブロックが存在している表もある。そのような表の一部は、構造から 1 項であることがあきらかな複数文字ブロックを領域内統合処理によって統合することで、罫線に省略のない表として正しく構造行列を得ることができる。しかし、文字内容を理解できない限り統合すべきかどうか判断できないような文字ブロックには、この領域内統合処理は対応できず、1 項の分離が未統合のまま罫線の省略として誤処理されてしまう。

一方、罫線に省略のある表に対しては、文字ブロックの並びから仮想的な項間の罫線を求めて実罫線に加えることで、構造行列を出力する処理をおこなった。この文字ブロックの並びを得るための手法として提案した、行対応と平均行構造を用いた項目分け手法は、項の省略や包含、位置ずれなどに柔軟であり、従来手法によりも正しい項目分け処理をすることが出来る。しかし、横方向の大見出しのように、項間の対応のルールが異なる部分には対応できていない。また、縦方向の包含が未処理である。

第 4 章

評価実験

4.1 まえがき

提案手法の能力評価のため、本手法を用いて実験的に表認識システムを構成し、実際の表サンプルからの構造行列抽出実験をおこなった。サンプルとする表は、電子情報通信学会論文誌及び情報処理学会論文誌の一定ページから表形式の部分を残さず抽出するという形で採取する。この部分が表であるという判断、及び表部分の切りとりは人間がおこない、附属するキャプションなどの文字列が入らない、純粋な表領域の画像のみが入力となるようにしている。

4.2 項分離抽出部の性能評価

項分離抽出部における、画像からの罫線及び文字ブロックの抽出能力の評価をおこなう。サンプルは電子情報通信学会論文誌、情報処理学会論文誌各 100 ページから抜きだした計 41 種の様々なフォーマットの表で、罫線の一部省略された表、点線を罫線とする表なども含んでいる。また、二値画像へ変換する際のスキャナの解像度を 200dpi,400dpi の 2 種類とし、解像度の変化に対する影響の評価もおこなった。結果は表毎に抽出成功か失敗かで表わされ、その判定は人間が出力と原画像を比較しておこなう。抽出誤りが 1 個でも生じればその表を失敗、それ以外を成功として係数している。

4.2.1 罫線抽出実験

結果を表 4.1に示す。

解像度	表数	抽出成功数 (%)
200dpi	41	40 (97.6%)
400dpi	41	40 (97.6%)
合計	92	80 (97.6%)

表 4.1: 罫線抽出実験結果

結果が示すように、罫線抽出部は解像度によらない高い抽出能力を持っている。誤りの原因について考察すると、200dpi での 1 件の誤りはつぶれによる文字線の罫線接触が他の罫線と同列上でおこってしまったケースで、本手法の多段階の選抜すべてで排除できないワーストケースのひとつであった。一方 400dpi の実験での失敗は大きなかすれが原因になっており、極低品質文書への対策を強化する必要があると考えられる。

4.2.2　文字ブロック抽出実験

文字ブロック抽出部において、すべて項と一対一に対応する文字ブロックを得るのは難しく、統合不十分を多く含むことが予想される。しかし本研究では、横方向の統合ミスに関して、その多くは構造解析部で修正可能と考えている。また縦方向に複数の文字行からなる項については、文字内容を見ずにはその統合は正しくおこなえないと考えられるので、その統合は文字ブロック抽出処理に組みいれていない。よってここでは以降の処理上に影響を与えるものとして、横方向の統合不十分と、縦横方向含めた過統合に限り計数をおこなった。結果を表??に示す。

解像度	表数	横統合不十分発生表数	過統合発生表数
200dpi	41	17	1
400dpi	41	18	1
合計	92	35	2

表 4.2: 文字ブロック抽出実験結果

横方向の統合処理では両解像度で各 1 件の過統合が生じた。本システムでは統合不十分は回復可能な誤りであるが、過統合は回復不能な誤りであると考えて、過統合を防止すべく統合の判定を厳しくした。この判定条件が、表によっては不十分であったと考えられる。しかしこれ以上判定を厳しくして統合不十分を増すことは、後段の処理の負担を増し精度も下げることになるので、別の面からの検討が必要である。解像度の変化に対しては、文字ブロック抽出部は解像度によらずほぼ同じふるまいを見せることがわかり、本手法の解像度対応能力が確かめられた。

4.3　構造行列抽出能力の評価

抽出された罫線と文字ブロックを用いて、入力表の論理構造を表現するように作られた構造行列の評価をおこなう。サンプルは電通学会論文誌 (以下 J76DII) と情報処理学会誌 (以下 IPS91) から各 300 ページ中の表 (合計 95 種) を抜きだし、すべて 200dpi で二値化している。これらは様々なフォーマットの表を含んでいるが、あらかじめ人間が見て

a) 罫線ですべての項が区切られている表

b) 罫線に省略のある表

に分類しておく (表 4.3)。但し、これは実験結果の検討に使用するのみで、システムに提示はしない。

文書	タイプ (a)	タイプ (b)	総数
J76	35	25	60
IPS91	11	24	35
計	46	49	95

表 4.3: 表のタイプ別分類

実験結果である構造行列の評価は、出力された構造行列が入力表の構造と正しく一致しているときに成功、一箇所でも違いがあれば失敗とした。この条件はたいへん厳しく、項の包含、省略などを正しく区別して構造化することを要求し、ひとつの項内に文字行が複数あった場合も、構造行列はこれをひとつの項としてなければならないことを意味している。

4.3.1　タイプ (a) の表の構造抽出能力の評価

実験システムにタイプ (a)、タイプ (b) の表を区別なく入力したとき, システムがタイプ (a) であると判断して出力したものを、前もってタイプ (a) に分類した表と比較して評価した。これを表 4.4に示す。括弧内の比は表総数に対する前段の罫線抽出が正しくおこなわれた表の数であり、項分離抽出部の性能を示している。

文書	(罫線抽出/表総数)	領域内統合前	領域内統合後
J76	(35/35)	8	25
IPS91	(10/11)	3	7
合計	(45/46)	11(23.9%)	32(69.6%)

表 4.4: タイプ (a) 表の構造抽出実験

入力がタイプ (a) 表であることをシステムが既知であれば、構造抽出の能力は前段の罫線の抽出能力に等しく、実験では表 4.4 中の括弧部分で示すように 95%以上の精度がえられる筈である。しかし本手法においてシステムがタイプ (a) と判断して結果を出力するのは、抽出された罫線で作った特徴点メッシュの小領域内全てで、文字ブロックが 1 個以下の状態になったときに限られる。抽出された罫線と文字ブロックをそのまま用いてこの判定をし、正しく出力されたものを認識成功としたとき、入力中のタイプ (a) 表総数に対する成功率は 11/46 = 約 23% と大きく低下してしまう (表 4.4の領域内統合前)。

ここでシステムがタイプ (a) と判定できないのは、文字ブロックの統合ミスにより項内の文字ブロックが分離している場合、もしくは元々の表が複数の文字行からなる項を含んでいる場合である。特に後者に注目し、サンプル中で調べたところ、1 項が複数行にわたる項をもつ表 (タイプ (a') と書くことにする) は 31 表あり、タイプ (a) 中の 6 割以上にものぼることがわかる。

本来、そういった特殊な項や前者の統合ミスを罫線の省略と区別して、正しく 1 項に統合するためには、文字内容の意味のまとまりを認識して統合しなければならない。しかし、表の一般的なルールを用いると、一部には表構造から統合すべきであることを判断できるものもある。本研究ではこの判断と統合をおこなうアルゴリズムを領域内統合処理として組みいれることで、表の特殊な項を統合し 1 領域 1 文字ブロックの状態を達成した。これによりタイプ (a) 表に対する総合的な構造抽出成功率は 32/46=約 70%まで回復できた (表 4.4 の領域内統合後)。実験では、文字を認識しなければ統合の難しいものもあるタイプ (a') の表のうち、17 の表を正しくタイプ (a) と判断して出力することができており、しかも誤統合は生じていない。このことから領域内統合処理の有効性が確かめられた。

更に統合をおこなってタイプ (a) の表を 100%正しく判定できるような機構が望まれるが、これ以上の強制的な統合では、現段階では生じていない誤統合を抑えることが難しくなるため、別の面からの検討が必要である。

4.3.2　タイプ (b) の表の構造抽出能力の評価

実験システムがタイプ (b) として出力した結果を考察する。この結果には本来タイプ (a) であった表のものの抽出失敗を含むが、それらはすべて本来ひとつ項内の文字ブロックを罫線省略とみなして分離してしまった誤りの例なので排除し、本来タイプ (b) に分類される表に対する処理結果のみ評価する。特に、文字ブロックの拡張のみでおこなう従来手法で作成した構造行列と、本研究の提案手法である行対応と平均行構造を用いた項目分け法で作成した構造行列とで、構造抽出成功率の比較をおこなう。

文書	(罫線抽出/表総数)	従来手法	提案手法
J76	(25/25)	10	13
IPS91	(23/24)	7	17
合計	(48/49)	17(34.7%)	30(61.2%)

表 4.5: タイプ (b) 表の構造抽出実験

罫線の省略された表に対し文字ブロックを用いておこなわれる構造解析法として、提案手法が従来手法よりも高い能力をもつことが本結果より示された。完全に正しい構造行列を抽出できたのはタイプ (b) の表の 6 割弱であり、この誤りのほとんどが、縦方向の見出しに包含項のある表、および複数行を持つ項を含む表に対するものであった。

誤りの主要因である縦方向の見出し部における包含は本手法では未処理の部分で、これには見出しという知識を利用した処理を付加するのが最も効果的と思われる。また罫線主体の表よりは少ないが、このタイプの表でも複数行にわかれる特殊な項が存在し、誤りの原因となっている。しかし、人間がみてわかりやすい表、という観点からは、罫線の省略された表でそのような複数行にわたる項を作ることは望ましくなく、実際にもそのような項は見出し部や罫線の補助のある部分にのみ存在する場合が多い。よってこれらに対してもまた見出し部の抽出のような知識処理が有効であると考えられる。

罫線抽出例

出力確率	非 left - right	left - right	
分布数	単一分布	単一分布	混合分布
6	77.6	73.4	76.2 (3×2)
7	78.2	74.3	
8	78.5	75.4	78.5 (4×2)
9	78.5	76.2	78.8 (3×3)
10	78.9	76.2	77.9 (5×2)
11	79.1	76.8	
12	79.9	75 9	79.0 (3×4) 79.2 (4×3) 77 6 (6×2)
13	79.8	76 2	
14	79.6	76.5	76 8 (7×2)

図 4.1: 原画像

```
D1 T4 T4    D3
   T1 C  T4 D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 C  C  C  D3
         T1 D3
         T1 D3
D1 C  C  C  D3
D1 C  C  C  D3
D1 T2 T2 T2 D3
```

図 4.2: 特徴点メッシュ

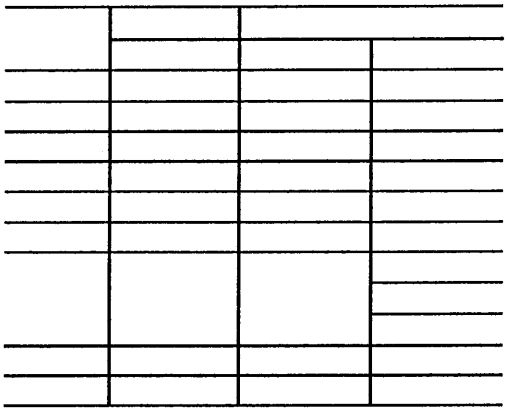

図 4.3: 抽出罫線 (左メッシュを実座標上で表現)

誤読回数	テスト字種	第 1 候補字種
1 5	烏	鳥
1 3	鳥	烏 息
1 0	問	間
9	采	采 菜
	突	実 安 宍 麦
	釆	采 禾 米 来

図 4.4: 原画像

```
D1 T4 T4 D3
D1 C  C  D3
D1 C  C  D3
D1 C  C  D3
D1 C  C  D3
   T1 C  D3
   T1 C  D3
D1 T2 T2 D3
```

図 4.5: 特徴点メッシュ

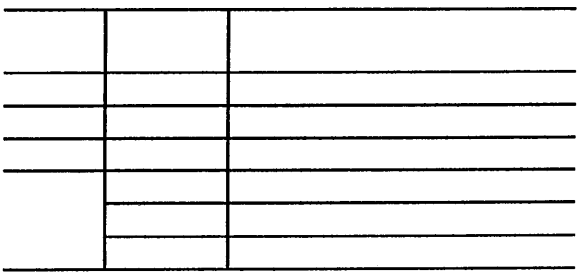

図 4.6: 抽出罫線 (左メッシュを実座標上で表現)

罫線抽出 (失敗) 例

(かすれによる罫線切れが補正できなかった例)

代表的な応用例			要求性能	
応用分野	条　件	文　字 フォント	画面(ページ)当り	1文字当り
CRT	640×400ドット/画面, 16色	16×16	0.1秒/画面	100 μs/文字
	1120×750ドット/画面, 256色	24×24		
プリンタ	300 DPI	40×40 ～48×48	3秒/ページ	1.5 ms/文字
	400 DPI	54×54 ～64×64		

図 4.7: 原画像

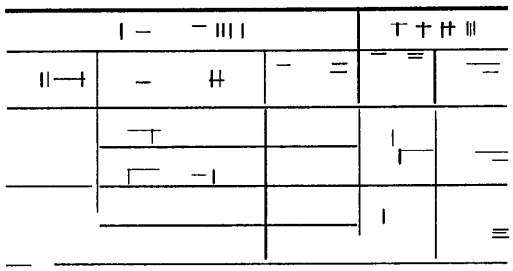

図 4.8: 罫線候補

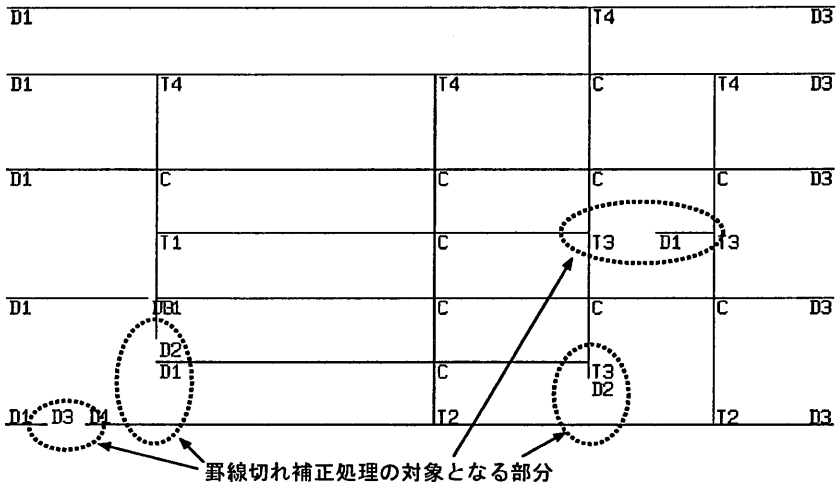

図 4.9: レベル 1～3 の罫線 (特徴点メッシュ上)

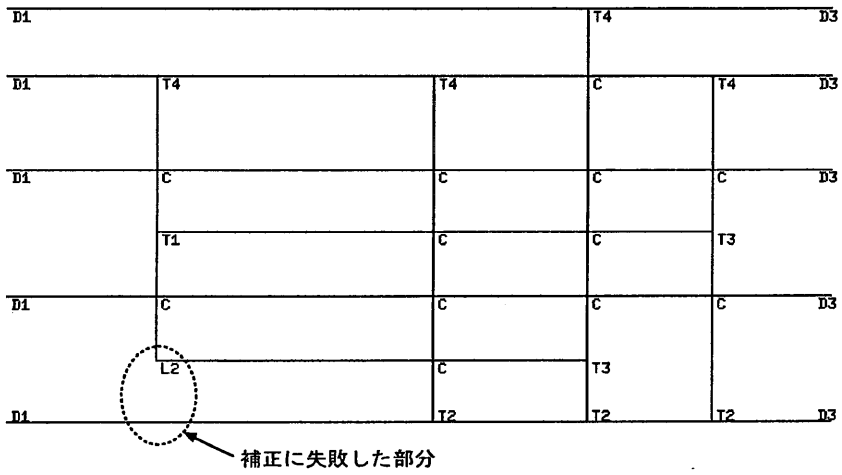

図 4.10: 罫線切れ補正処理後

構造行列抽出例

タイプ (a) 表、成功例

適応単語数	5	25	50	100	216
音声長 (sec)	3.0	16.7	37.3	75.8	102.6
認識率	50.7%	63.1%	71.8%	76.8%	82.1%
音素 HMM の出現率	22.5%	77.0%	87.8%	93.0%	100%
適応用学習資料中に現れた音素数	32	156	319	640	1359

図 4.11: 原画像

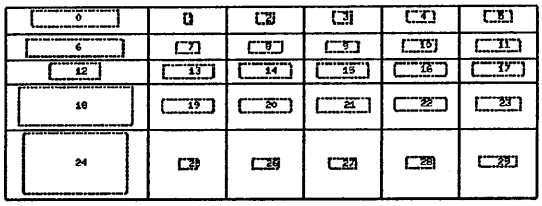

図 4.12: 出力構造行列

タイプ (a) 表、失敗例

原因:　複数の文字行からなる項が同じ項目内に多数あるため、領域内統合できなかった。

認識系	第 1 候補認識率 P_1(%)	文字認識承定数 a	実験データから求めた b	式 (3) から求めた b
1	95.8	1.71	4.92	4.91
1	87.4	1.71	3.76	3.64
1	61.8	1.71	2.64	2.55
2	96.0	2.36	4.78	4.85
2	79.9	2.36	2.87	2.87
2	46.6	2.36	1.86	1.84
3	94.0	2.74	4.45	4.30
3	72.1	2.74	2.48	2.43
3	46.5	2.74	1.74	1.72

図 4.13: 原画像

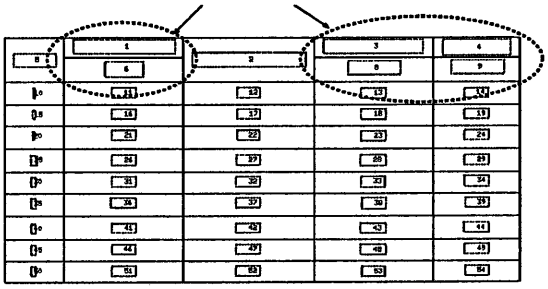

図 4.14: 出力構造行列 (誤)

タイプ (b) 表、成功例

状態	s	t	k	n	k'	時　間	時間比
1	2	1	2	3	1	17	1
2	3	2	4	5	2	31	1.8
3	3	3	6	6	3	50	2.9
4	4	4	8	7	4	133	7.8
5	4	5	10	8	5	205	12
6	5	6	12	10	6	383	22
7	5	7	14	11	7	1100	64
8	6	8	16	13	8	2850	170

図 4.15: 原画像

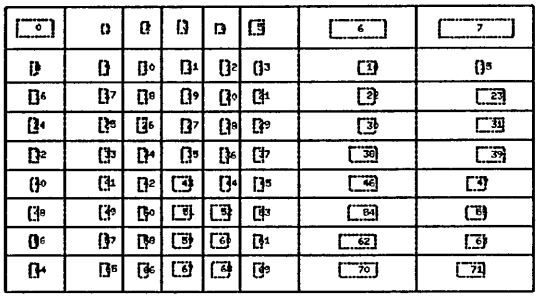

図 4.16: 出力構造行列

Circuit	(A)		(B)		(C)		(D)	
	#Node	CPU	#Node	CPU	#Node	CPU	#Node	CPU
dec 8	40	0.3	41	0.3	390	0.4	57	0.4
enc 8	33	0.3	31	0.3	30	0.3	37	0.4
add 8	49	0.4	120	0.4	452	0.4	1183	0.6
add 16	97	0.7	248	0.5	1700	0.9	94814	24.1
mult 4	330	0.5	358	0.4	304	0.5	394	0.5
mult 8	46594	18.3	38187	14.5	31026	14.0	77517	26.1
c 432	89338	34.1	11348	7.4	6205	5.6	479711	278.6
c 499	36862	21.5	68816	39.1	32577	21.0	112815	78.0
c 880	30548	11.5	(>500 k)		(>500 k)		(>500 k)	
c 1355	119201	51.4	246937	102.9	103301	46.9	373974	179.0
c 1908	39373	22.5	47990	22.7	65895	63.3	91082	47.4
c 5315	40306	29.8	105200	32.5	(>500 k)		(>500 k)	

図 4.17: 原画像

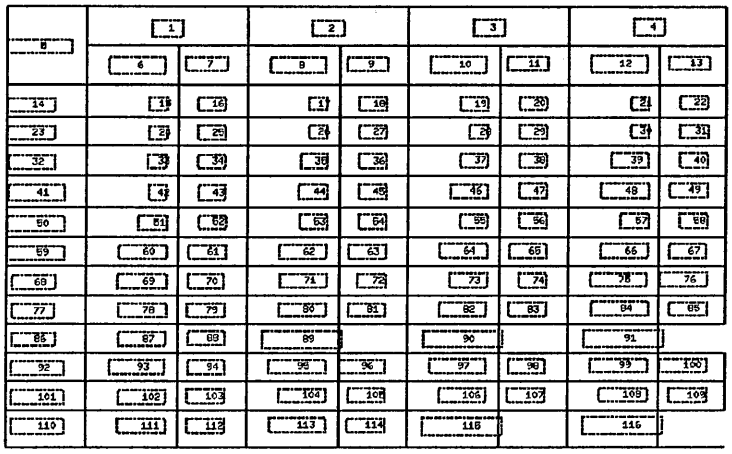

図 4.18: 出力構造行列

タイプ (b) 表、失敗例

原因:　項内の分離が 3 行同じようにおこり、誤った平均行構造を作成した

投票値	中心座標 (x_c, y_c)	(α, β, θ)
98	(54.074, 53.082)	(25, 15, 0)
98	(53.082, 54.074)	(25, 15, 0)
98	(52.090, 53.082)	(25, 15, 0)
...		

図 4.19: 原画像

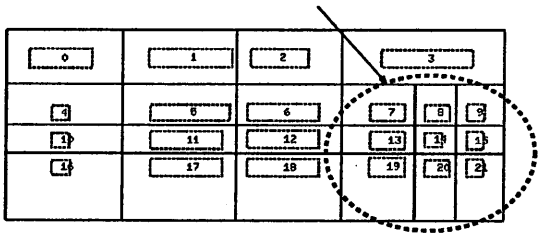

図 4.20: 出力構造行列 (誤)

原因:　縦方向の包含をなす見出し

	r_a	2.0	4.0	6.0	8.0
(a)	E_M	2.1E-05	2.5E-06	6.2E-06	7.5E-06
	E_A	2.7E-05	2.6E-06	6.1E-06	7.5E-06
	E_F	2.7E-05	2.6E-06	6.2E-06	7.5E-06
	E_F'	1.6E-04	2.3E-05	7.5E-05	1.1E-04
(b)	E_M	4.4E-07	1.0E-10	1.2E-13	8.7E-12
	E_A	3.1E-07	3.7E-11	1.3E-13	8.6E-12
	E_F	4.4E-07	1.0E-10	1.3E-13	8.7E-12
	E_F'	5.7E-05	1.3E-08	1.5E-11	7.0E-10

図 4.21: 原画像

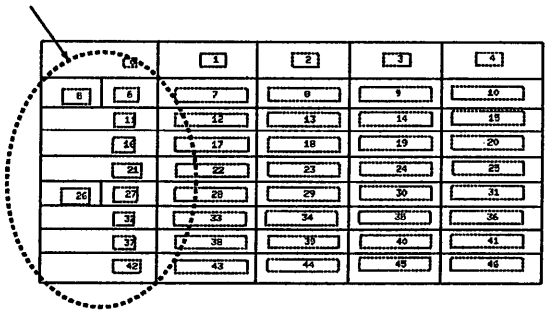

図 4.22: 出力構造行列 (誤)

第 5 章

結論

5.1 本研究のまとめ

本研究では、表の構造を自動的に認識するための表構造認識処理の実現について検討し、様々な形式の表に対応できる汎用性の高い表認識手法を提案した。この手法を用いて実験的に表認識システムを構築し、実験により性能の評価をおこなった。

まず 2 章では画像から表構造の最小要素である表の項を分離するための画像特徴として、罫線及び文字ブロックを抽出する処理について検討し、実用面で生じる入力の変動、すなわち画像の劣化および表のサイズや解像度の変化に広く対応できる処理を実現した。

特に罫線は表の構造をあきらかにする特徴として最も重要であるため、誤った罫線選択やもれがないようにすることを目的として多段階選抜法を提案した。多段階選抜法では、入力から動的に定める閾値を用いたり、文字矩形の情報から求まる罫線としての確信度を付加することによって精度を高めている。実験では 2 種の解像度を表を入力し、そのどちらにも本手法が高い抽出能力を示すことを確認した。

また、本研究では入力表が必ずしも罫線ですべての項が区切られていないことも想定し、罫線のない部分の項を分離するために「一般に項内の文字間隔は項の間隔よりせまい」ことを利用し、文字の集群を文字ブロックと呼んで項に対応させて抽出する手法を採用した。この文字ブロック作成は画像から抽出された矩形を統合する処理であるが、このとき罫線抽出部の線素情報を用いて矩形の文字らしさを強化したり、統合の閾値に動的な値を用いることによって文字ブロック作成においても解像度の影響を受けにくい処理を実現することができた。

3 章では項分離抽出部で抽出された罫線と文字ブロックの情報を用いて、表の論理構造表現である構造行列を作成する為の手法について検討した。従来の多くの表構造認識手法

のように表の各項が罫線で区切られているという仮定を用いれば、罫線の構造をそのまま表の構造行列として出力することが可能である。しかし実際の文書入力を想定した場合はそのような仮定を用いることはできないので、表認識システムが自動的に判断しなければならない。本研究では罫線のなす領域分割から得られる小領域数と、その領域内に存在する文字ブロックの数から罫線主体度という評価値を定義して罫線に省略があるかどうかの判断をおこなった。更に小領域で文字ブロックが複数存在した場合でも、その項の項目内での特殊性に注目してこれを一部統合することによって、文字ブロック統合不十分や、本来 1 つの項であるべき複数行を統合し罫線に省略のない表として出力できる表を大幅に増やすことができた。本手法によってもうまく処理ができなかった例として、複数行をもつ項が横に並んでいる表などが挙げられる。しかしこのような表は文字内容を見なければ正しい判断ができないものであり、文字認識の結果を利用しなければ処理が困難であると考えられる。

3 章後半では、罫線の省略された表や、処理上では罫線主体と判断されなかった表に対する構造抽出法について検討した。項分離抽出部では罫線に加え、項を文字らしい矩形の集合である文字ブロックとして出力している。本研究では文字ブロックの並びから表の構造を抽出する手法に関する研究をおこなった。このとき、文字ブロックのヒストグラムを用いて文字ブロックの重なりを表全体でとらえて項目分けをおこなう従来的手法は、項が整然と並んでいる表に対してはよく働くのに対し、項の位置にずれがあったり項の省略や包含があるような表には対応できない。ここで文字ブロックの重なりの乱すのは、文字内容によって表内で著しく変化する項の幅や位置のずれであることに注目し、項目分けを行間の問題に広げることで特殊な項の状態にも対応し、さらに表から平均的な行を求めて巨視的な視点から各行の項目わけを平均行との対応でおこなうという、行対応と平均行構造を用いた項目分け手法を提案した。従来手法を用いて表を行へ分割をしたのち、提案手法によって列方向の項目分けをおこなった場合の構造抽出結果は従来手法のみの場合の性能を上回っている。しかし実験によると罫線省略タイプ表全体に対する構造抽出率は 61% とまだまだ低く、現在未処理の縦方向の包含等を検討してさらに多くの表に対応していくことが必要であろう。

最後に、提案手法を用いて実験的に構成した表認識システムを用いておこなった認識実験の結果から本研究の総合的な成果を考察する。この実験的な表認識システムは、表部分の二値画像にの解像度、品質等について広い範囲に対応できること、更に罫線で区切られた表、省略のある表を自動的に判断し、その表構造と全く等価な構造行列の出力が期待できるという点で、実用的なシステムに望まれる仕様の多くを満たしている。

実験結果では、罫線の抽出性能に関しては実用に充分な抽出率を達成することが出来た。

また、文字ブロックの抽出についても、解像度の影響を受けない抽出性能が得られている。罫線及び文字ブロックの抽出については、入力画像の変動に安定であるという本研究の目的のひとつを、ほぼ達成していると考えられる。

また、構造行列抽出能力を評価すると、

タイプ (a)	罫線ですべての項が区切られた表	70%
タイプ (b)	罫線の省略がある表	61%

という成功率であった。特にタイプ (b) の表に対する構造抽出成功率を従来手法の 24%から大きく向上することができた。入力された表全体では 65%と低くまだまだ実用に充分であるとは言えないが、様々なフォーマットの表に対応するという目的に沿って行われた本研究の構造解析手法は、特に罫線の省略された表に関して対応する範囲を広げているという点で有効であるものと考えられる。

構造抽出の失敗の主原因は、表の 1 要素である項が複数行にわたっている場合に、罫線の省略された 2 つ以上の項と判断してしまうことである。文字内容を見なければ正しい判断が出来ない場合も多いこの特殊な項に対し、いかに文字内容以外の情報で正しくこれを統合できるかが今後の構造抽出性能を大きく左右すると言えよう。それ以外の誤り、例えばタイプ (b) の表で多数発生した見出し部に関する誤りに関しては、表をモデル化した知識を用いて入力表から見出し部を特定することで、そこに最適なルールを適用し処理するのが効果的と考えられる。

5.2　今後の課題

最後に今後の課題を述べる。

罫線抽出部分の問題点として、極低品質文書、とくにかすれのひどいものについては誤りを生ずることがあった。その対処法として、罫線の延長部にある小さな画素を矩形情報を利用して積極的に拾う方法が考えられる。

構造抽出の際に問題となった見出し部の項目分けは、見出し部特有のルールにしたがっておこなうことが望ましい。そのためには表の見出し部、データといった構造のモデルを記述した高レベルの知識を用い、このモデルを現手法で得られた構造行列と比較して見出しの部分を入力表から自動的に特定する処理が必要である。特定された見出し部に対して専用ルールを用い、項目分けをやりなおすことによって正しい構造行列を得られる表の種類をさらに増やすことができるであろう。

謝辞

本研究を進めるにあたり、全般的な御指導を賜わりました東北大学工学部 阿曽弘具教授に心より感謝いたします。また, 本論文をまとめるにあたり、貴重な御意見をいただいた東北大学工学部 伊藤貴康教授、阿部健一教授に深く感謝いたします。

東北大学情報処理教育センター 大町真一郎助手、東北大学大学院博士課程 後藤英昭氏には文書認識の専門分野においてさまざまな御指導、御助言をいただきました。ここに深く感謝いたします。

さらに、東北大学工学部 成富敬助手、東北大学大学院博士課程 黒岩丈介氏には研究全般にわたる有益な御教示をいただきました。厚く感謝いたします。

最後に日頃よりお世話になりました阿曽研究室の皆様に心より感謝致します。

参考文献

[1] 西尾、岩渕、水谷: 「岩波国語辞典 第三版」
岩波書店, (1979).

[2] 成瀬、駱、金井、渡邊、杉江:「帳票における枠罫線構造の記述と認識」
信学技法 PRU-90-150, (1990).

[3] 中野、藤沢、国崎、岡田、花野井:「文字認識と協調した表形式文書の理解」
電子情報通信学会論文誌 Vol.J69-D, No.3 (1986).

[4] 田端、鶴岡、木村、三宅:「表の構造理解のための罫線抽出と領域分け」
信学技法 PRU-90-73, (1990).

[5] 佐藤、井上、鳥生:「文書入力のための表構造の認識」
1988 年信学春全大, D-232 (1988).

[6] 糸乗:「文字ブロックの並びを考慮した表構造の認識」
1993 年信学春全大, D-410 (1988).