グラフィクスとCAD 15－2
（1984. 11. 6）

荒めに手書きされた論理回路図面の認識

Pattern Recognition for logical circuits diagrams written by freehand.

中嶋 正之　　安居院 猛　　飯塚 久登
東京工業大学像情報工学研究施設
Masayuki NAKAJIMA Takeshi AGUI Hisato IITSUKA
(Tokyo Institute of Technology)

A pattern recognition system for logical circuit diagrams using parallel vector tracers and argument differential functions is proposed .
An intersection tracer for the extraction of intersection of lines and an edge tracer for that of wiring lines are defined , and the argument differential function for the recognition of symbols are used .
Recognition results of logical circuits written by freehand and that written using a template are shown.
This system is also efficient to extract circuit structures from logical diagrams.

1．はじめに

筆者らは、現在各種の図面に対する計算機処理に関する研究を行なっている。地図図面の解析処理としては、山岳地図を対象として、等高線成分と文字記号成分の分離[1]、等高線パターンのグラフ表現[2,3]、効率的な高度付与[4]などの解析処理を試みた。また、市街化地図に対しては、ピラミッド構成を利用した建築物領域の抽出[5]、グラフ構造解析による道路情報の抽出[6]などの解析処理を行なった。また、簡単な３面図の自動認識の試行実験を行なった[7]。そこで本報告では、主として市街化地図の処理において使用したグラフ構造の解析の手法[8]を、手書きの論理回路図面に適用した実験結果について述べる。

手書きされた論理回路の自動的な認識は、これまでに、格子基準を設定する方式[9]、配置を水平、垂直に限定する方式[7]、補助マークを付与する方式[10]等が提案されている。これらの方式は、認識率を向上させるため記入の際に制約を与えたものであり、自由に描かれた図面を必ずしも対象とする方式ではなかった。そこで筆者らは、多少荒めに手書きされた論理回路図面に対しても認識を行う図面認識システムを設計した。

対象とする論理回路図は、図１に示すような各種の論理ゲートとその結線関係が表現されたものである。本システムは、この論理回路の図面を入力画像とし、入出力に対応する線にラベルを与えることにより、結果として入出力関係の論理式を決定することを目的としている。

2．対象とする論理回路図面について

ここで、まず対象となる入力画像としての論理回路図を明確に規定しておく。

(1) 入力画像は２値ディジタル画像である。

(2) 入力画像は図２に示す各種論理ゲート、即ち、AND,OR,NOT,NAND,NOR,および XORゲートとその結線関係を示す線、連結点、交差点だけで構成されているものとする。

ここでいう連結点とは、図３中のC1,C2,C3のような領域で実際に線が連結している点のことであり、交差点とは、図３中のX1のような領域で交差しているだけで実際には連結していない点のことである。

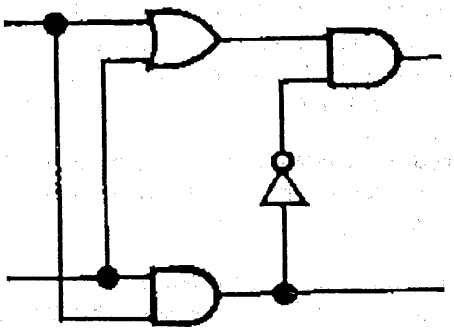


図１　対象となる論理回路図

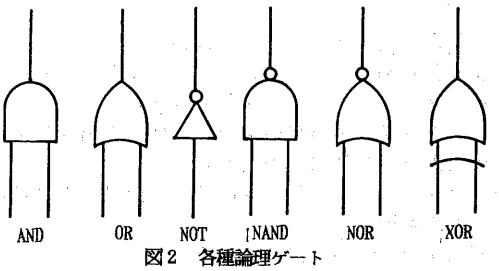


図２　各種論理ゲート

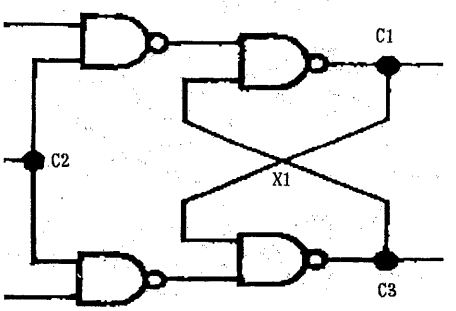


図３　連結点と交差点

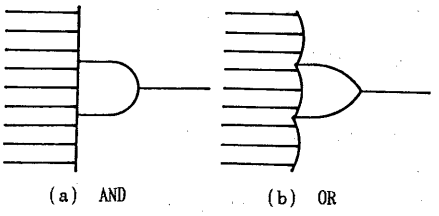


図4　多入力論理ゲート

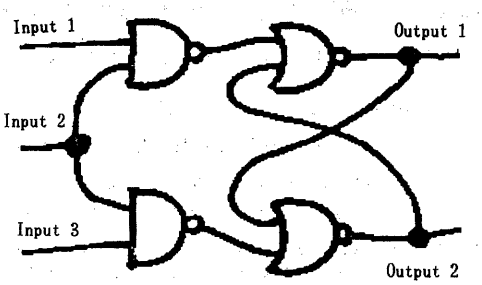


図5　対象とする論理回路図の例

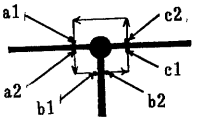


図6　連結点情報の抽出

(3) 入力画像に表現されている交差点は、枝数４以外のものは存在しないものとする。実際に論理回路を書く場合や現在用いられている論理回路図について考えてみると、交差点は交差する点は２本即ち枝数４のものが大部分である。これは３本以上の線が一点で交差する回路図は判読する際に誤解を招く恐れがあるためである。
(4) 連結点に流入する枝は何本でも良いものとする。
(5) 論理ゲートを示す領域は、２値化される際に４連結白画素領域の閉領域になっている必要がある。
(6) 各論理ゲートに流入する入力線は何本あっても良いが、図４に示すように、入力側の辺を延長して入力線を記入したゲートは対象としない。ただし、図４のようなゲートを許すようにアルゴリズムを変更するのは容易である。
(7) 図面は、論理ゲートが切断されないように切り出され、どの線がその図への入出力線が解かっているものとする。

なお、実験で用いた論理回路図は、テンプレートを用いて書いた比較的きれいな図面と、多少荒めに手書きされた論理回路図面の２種類である。

３．論理図面認識アルゴリズム

論理回路図認識システムのアルゴリズムは大別すると次の４つのステップに区分できる。

第一ステップ　連結点の抽出、連結点に出入りする線の識別および連結点の削除。
第二ステップ　論理ゲートの抽出、認識および論理ゲートの削除。
第三ステップ　結線関係の認識。
第四ステップ　各情報の統合および認識結果の出力。

つぎに、各ステップにおける処理および認識方法などについて詳しく述べる。

３．１　連結点の抽出

図５は入力画像として用いられた画像の一例であるが、連結点は他の部分に比べてまとまった広さをもった黒領域としている。そこで連結点の抽出は、明記されている連結点よりもやや小さめの正方形のマスクを用意し、このマスクによって画面全体を走査し、マスク内が全て黒画素である付近に連結点が存在するものとして行なう。

連結点が発見された場合には、図６に示すように、連結点を中心に連結点を含む正方形の縁を走査し、その連結点に流出入する線の端点を記憶する。ここでいう端点とは、実際には、図６のa1,a2,b1,b2,c1,c2 のように、端点付近の線を表現している領域をはさむ左右の輪郭画素である２点のことを意味する。将来、ここに記憶された端点に対応する枝には同じラベルを与える。

連結点の抽出が終了したならば、入力画像に対して連結点の部分を消去した画像を作成し、第二ステップの処理に移る。図７の画像は、図５の画像に対して、連結点を消去した画像である。

３．２　論理ゲートの認識

論理ゲートの認識を行なう場合、まず図面のどの領域が論理ゲートを表わしているかを知る必要がある。そこで、図８のように、連結点を消去した画像に対して４連結白領域のラベル付けを行なう。ラベル付けによって得られる情報は、図９に示すように、領域のもつラベル”Ｌ”、領域のもつ画素数”Ａrea[L]”、領域の輪郭の１点のＸ－Ｙ座標’(SX[L],SY[L])’、領域のおよぶ範囲’XMAX[L],XMIN[L],YMAX[L],YMIN[L]’である。これにより、図面内にどのように白領域が分

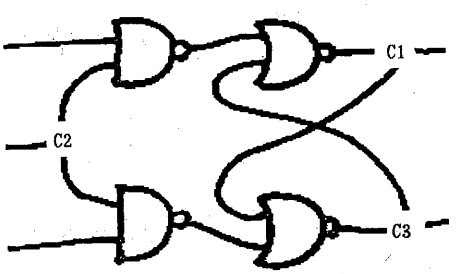


図7　連結点を消去した画像１

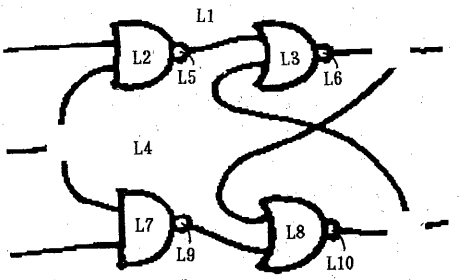


図8　４連結白領域のラベル付けをした画像１

布しているかを知ることができる。入力画像に対してラベル付けを行なわずに、連結点を消去した画像に対してラベル付けを実行する理由は、図10のL8,L10の領域のように、入力画像には論理ゲートのもつ領域、即ち、L2,L3,L12,L13 の領域に近い大きさの領域が存在する場合があり、連結点の消去により、そのような領域が他の領域と合併する可能性が大きいためである。実際に、図10のL1,L8,L9,L10,L15の領域は、図８のL1領域に統合されており、図10のL4,L5,L11 の領域は、図８のL4領域に統合されていることがわかる。

得られたラベル情報から、論理ゲートを表わしている領域を選び出す条件として、前章の (5)で規定したように閉領域になっていなくてはならない。そこで、ラベル情報のうち、領域のおよぶ範囲' XMAX[L],XMIN[L],YMAX[L],YMIN[L] ' を参照して、その範囲が画面の端におよぶ領域は、論理ゲート領域ではないとする。また、あまり大きな領域も論理ゲートとは考えにくい。そこで画素数を参照して、その値がある定められたいき値よりも大きい場合には、その領域も論理ゲート領域ではないとする。以上の条件により、選ばれた領域を論理ゲート候補領域とよぶ。

論理ゲート候補領域は白領域であるから、図11に示すように、NOT,NAND,NOR,XORの各論理ゲートは、複数の論理ゲート候補領域で構成されることになる。そこであらかじめ、同一の論理ゲートを構成している論理ゲート候補領域どうしをグループ分けしておく。これにもラベル情報が用いられる。

まず、全ての論理ゲート候補領域を、負論理を意味する丸' Negative circle ' とその他の領域に分類する。負論理の丸とその他の領域では明らかに大きさが異なるため、適当ないき値を設定しておき、画素数がそのいき値よりも小さな領域を全て負論理の丸とする。負論理の丸は、論理ゲートの入出力部に出現し、必ず他の領域と共に論理ゲートを構成するため、近傍に必ずグループを組むべき領域が存在する。そこで、領域のおよぶ範囲を参照して、ある定められたいき値Thn よりも近くに存在する領域とグループを組ませる。

次にXOR のOR形と方形を組ませる必要がある。本システムではゲートレベルの論理式の決定を目的とするため、XOR ゲートには負論理の丸を付けることを禁止する。そこで負論理の丸とグループを組んだ領域を除く論理ゲート候補領域に対して、領域のおよぶ範囲を参照して、いき値Thn より近くにある領域どうしにグループを組ませる。

論理ゲート候補領域のグループ分けが終了したならそのゲートへの入出力線の端点を認識する。これは連結点への流出入線の認識の場合と同様に、図12に示すように、ラベル情報の領域のおよぶ範囲 ' XMAX[L],XMIN[L],YMAX[L],YMIN[L] ' の値から、ゲートを含むような長方形を求めて、その長方形の縁を走査することにより得る。この時に、どの線が入力線で、どの線が出力線かを知る必要がある。これは、対象ゲートが AND,OR,NAND,NOR ,XOR の場合には、入出力線が複数

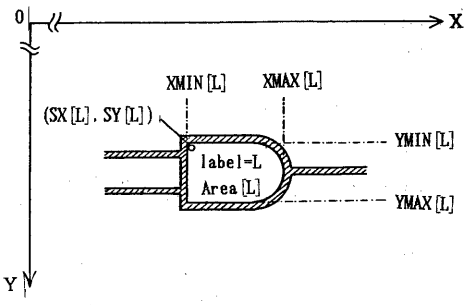


図9　４連結白領域ラベル付け情報

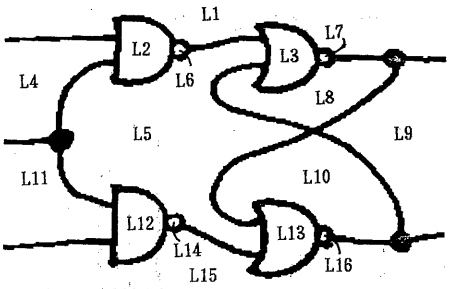


図10　４連結白領域のラベル付けをした画像２

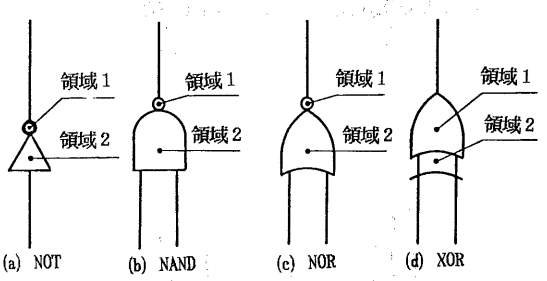


図11　複数の白領域で構成される論理ゲート

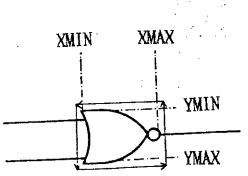


図12　論理ゲートへの入出力情報の抽出

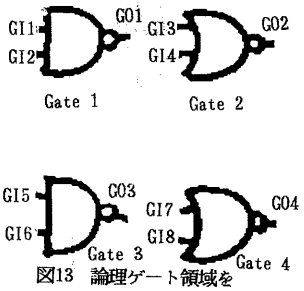


図13　論理ゲート領域を抽出した画像１

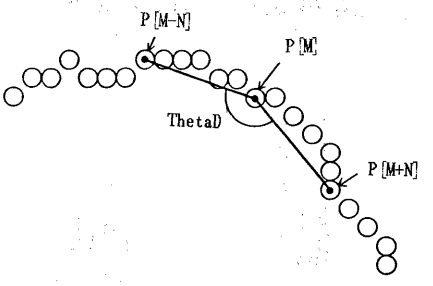


図14　偏角差分関数の定義

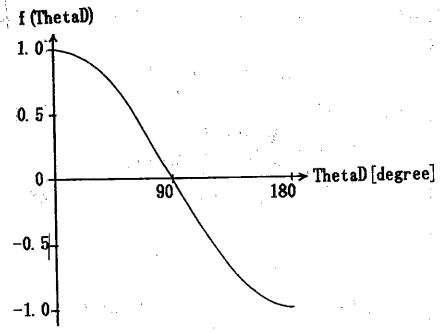


図15　偏角差分関数の特性

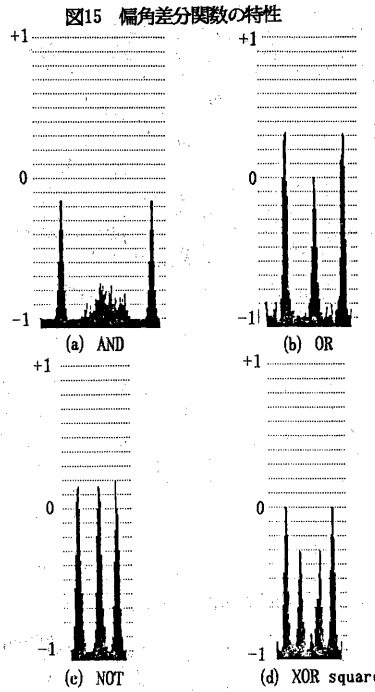


図16　各種論理ゲート領域の輪郭線に対する偏角差分関数1

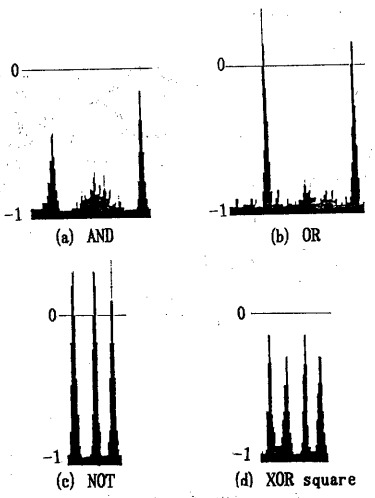


図17　各論理ゲート領域の輪郭線に対する偏角差分関数2

本あるため、互いに近くにある複数本と孤立している１本を区別し、前者を入力線、後者を出力線とすればよい。また、対象ゲートはNOT の場合には、入力線、出力線、共に１本であるが、目的が論理式の決定であるため、この２本の線が互いに否定の関係にあるという情報が得られれば、厳密に、どちらが入力線で、どちらが出力線かを決める必要はない。このため、NOT ゲートについては、入出力線の決定は行なわない。ただし、もし入出力線を決定する必要性が生じた場合には、後で、論理ゲートの形の認識の際に用いる偏角差分から、論理ゲートの向きが解かるため、決定することは可能である。図５の画像に対して論理ゲートを抽出したものが、図13である。

得られた各ゲートの形は、論理ゲート領域の輪郭線の偏角差分関数に対するパターンマッチングにより、AND 形、OR形,NOT形,XORの方形に区別される。ここで用いる偏角差分関数F[M]は、図14に示すように、ゲート領域の輪郭線のある画素P[M]に対し、その画素自身と前後Ｎ画素離れた点を結ぶ２本の直線のなす角度をThetaD(M) とすると、式１に示すように偏角差分関数の値f(ThetaD(M))を輪郭点列の順序に並べたものとする。

$$F[M] = f(ThetaD(M)) = \cos(ThetaD(M)) \quad (1)$$

但し、F［M］は、M番めの輪郭画素における偏角差分関数の値を意味する。

0[度]<=ThetaD<180[度] の範囲におけるf(ThetaD) の値を示したグラフが、図15である。f(ThetaD) はもともと三角関数の余弦をもとにしているため、90度付近が 0度あるいは 180度付近に比べて感度が高くなっている。これが論理ゲートに出現する形状の認識には、非常に都合がよい。図16、図17に各論理ゲートに対する偏角差分関数の形を示す。図16(a),(b),(c),(d) は、それぞれテンプレートを用いて書いた AND形、OR形、NOT形,XORの方形である。図17(a),(b),(c),(d) は手書きによる AND形、OR形,NOT形、 XORの方形である。

XORの方形は、他の図形のピーク数が３以下であるのに対し、テンプレート、手書きともに明確なピークが４つ出る。このため XORの方形の認識は容易である。

NOT形は、大きなピーク、すなわち関数値が0.0 をこえるようなピークが３つ出るので、これも容易に認識できる。

AND形とOR形の区別は、必ずしも容易ではない。テンプレートを用いて書いたOR形は、0.0 をこえるピークが２つ、-0.5をこえるピークが１つでる。テンプレートを用いて書いた AND形は-0.5をこえるピークが２つでる。このため、テンプレートを用いて書いた AND形、OR形の区別は容易である。微妙なのは、手書きによる AND形、OR形の区別である。手書きの場合、OR形の先端をとがらせて書かないことがよくあるため,AND形、OR形ともにピーク数が２つとなってしまうのである。こうなると、残された特徴は、入力側の角が鋭角か否か、入力側の辺が直線か曲線かという２つに限定され

る。ここで、前に述べた「偏角差分関数が９０度付近で高感度である。」という特徴が生かされる。残された特徴のうち入力側の角が鋭角か否かという情報がこの偏角差分関数では、重み付けされていない偏角差分関数に比べて明確に表われるためである。OR形のもつ角のほうが AND形のもつ角に比べてより鋭角であるためピークの頂点のしきい値を-0.2付近に設けることにより区別する。以上から、論理ゲートの形の認識が行なわれる。

４．結線関係の認識

結線関係の認識は、パラレル・ベクトル・トレーサで論理回路図面の結線を示す領域を追跡し、そのグラフ構造を明らかにすることにより実行する。

４．１　パラレル・ベクトル・トレーサについて

パラレル・ベクトル・トレーサは、主に直線や曲線が交差・分岐して構成されている図形の構造を自動解析するトレーサであり、次の機能を有している。

(1) 枝の部分の追跡が行なえること。枝を示している細長い領域を追跡し、図面の結線情報を抽出する。
(2) 枝の追跡中、前方にある交差点あるいは端点が存在する場合には、それを認識できる。
(3) 交差点が存在する場合には、探索を行なうことにより、その構造を明らかにできる。

パラレル・ベクトル・トレーサは、２種類のトレーサからなる。１つは(1),(2) の機能を実現するためのトレーサであり、図18に示すように４本のベクトル PLV,PRV,FLV,FRV が２本ずつ連結し平行に配置されたもので、主に枝の部分を追跡するのに用いられる。これを枝追跡子とよぶことにする。もう１つは機能(3)を実現するためのトレーサであり、図19に示すように２本のベクトルPISV,FISV が連結されたものと交差状態を探索するベクトルBSSVから成る。これを交差点探索子とよぶことにする。

４．１．１　枝追跡子の動作

枝追跡子は、追跡すべき細長い領域をはさむ２点を与えることにより設定される。通常この２点は、交差点の探索の際に得られる枝追跡待ち行列リスト "Branch0"の先頭の２点を用いる。２点追跡法とは、細長い領域の左右の輪郭線に対して、ひとつずつ、計２つの追跡点を設定して、一方の輪郭線は時計回りに、他方の輪郭線は反時計回りに追跡していく方法である。図20に示すように、いったん枝追跡子が設定されると、PLV,PRV の先端の２点から２点追跡法によって次の輪郭点を決定し、各ベクトルが現在の輪郭点からひとつ先行する輪郭点に頭尾を移動させることにより１ステップの追跡が実行される。

[交差点が存在する場合]

枝追跡子は、前方に、交差点が存在する場合、それを確実に認知し対応する必要がある。そこで枝追跡子

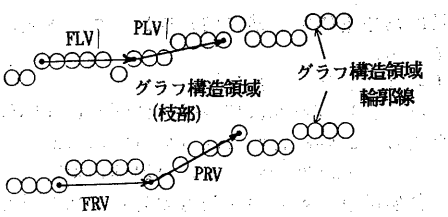


図18　枝追跡子の構造

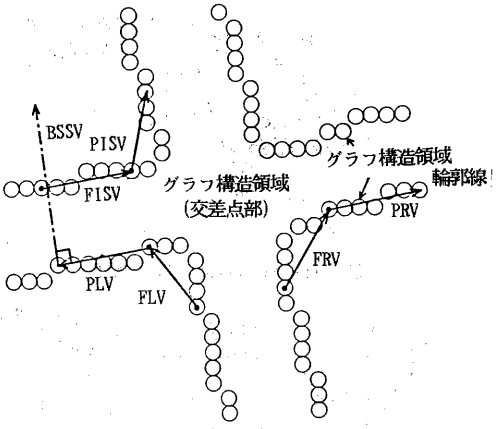


図19　交差点探索子の構造

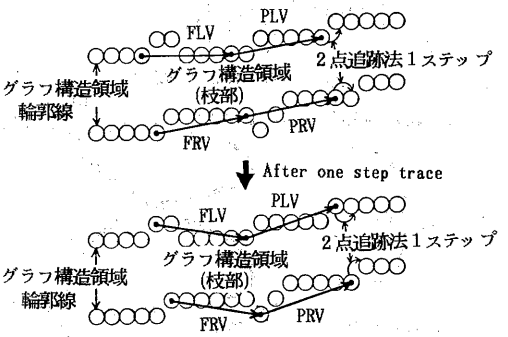


図20　枝追跡子の追跡動作（１ステップ）

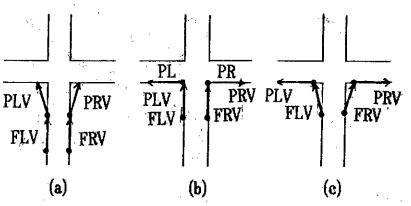


図21　枝追跡子の交差点における動作

は輪郭線を追跡中、常に(2)式に示す内積値DTを計算している。

$$DT=(PLV,PRV)=(PLVx*PRVx+PLVy*PRVy)/\{abs(PLV)*abs(PRV)\}$$

ただし $abs(PLV)=(PLVx**2+PLVy**2)$

$abs(PRV)=(PRVx**2+PRVy**2)$ (2)

内積値DTは、前方の交差点の有無を調べるために計算される。図21(a)に示すように、枝追跡子が交差点にさしかかると、PLV,PRV,それぞれのベクトルの示す方向がずれてくるため、内積値DTは減少しはじめる。そこで、ある一定のいき値Thを設定しておき,DT<Thとなった時点で(3)式,(4)式,(5)式に示す内積値DL,DR,DFを計算し、その値を記憶しつつ追跡をつづける。

$$DL=(PLV,FLV)=(PLVx*FLVx+PLVy*FLVy)/\{abs(PLV)*abs(FLV)\}$$

ただし $abs(PLV)=\sqrt{(PLVx**2+PLVy**2)}$

$abs(FLV)=\sqrt{(FLVx**2+FLVy**2)}$ (3)

$$DR=(PRV,FRV)=(PRVx*FRVx+PRVy*FRVy)/\{abs(PRV)*abs(FRV)\}$$

ただし $abs(PRV)=\sqrt{(PRVx**2+PRVy**2)}$

$abs(FRV)=\sqrt{(FRVx**2+FRVy**2)}$ (4)

$$DF=(FLV,FRV)=(FLVx*FRVx+FLVy*FRVy)/\{zbs(FRV)\}$$

ただし $abs(FLV)=\sqrt{(FLVx**2+FLVy**2)}$

$abs(FRV)=\sqrt{(FRVx**2+FRVy**2)}$ (5)

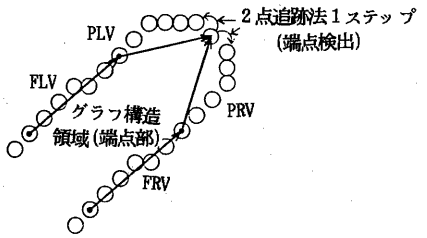


図22 枝追跡子の端点における動作

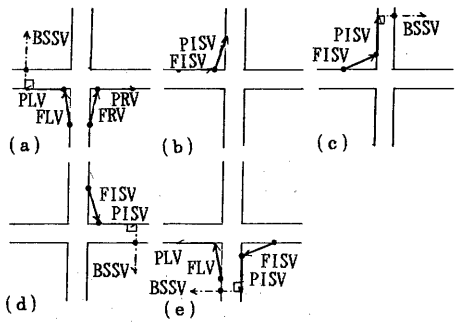


図23 交差点探索子の動作

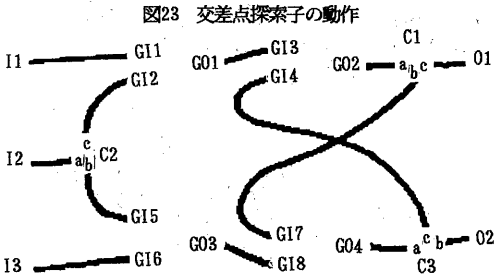


図24 初期設定時に枝追跡待ち行列に記憶される節

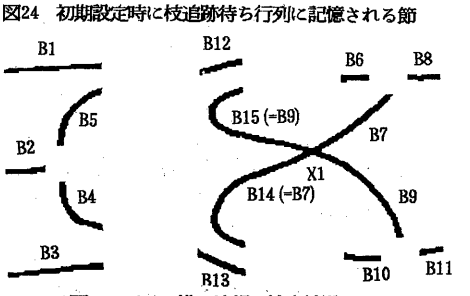


図25 グラフ構造情報の抽出結果1

そのまま追跡をつづけると図21(c)に示すように、DF<Thとなる時点がくる。この時点で記憶されている内積値DL,DRの最小値を見つけ、その時点におけるベクトルの連結点、即ち図21(b)の点PL,PRを得る。このことにより、現在追跡中の得だの枝の先端で発見された交差点において、その枝が両隣の枝とどのくらいの角度で交わっているかが、内積DLとDRから分かる。枝追跡子は、ここで追跡を終了する。

[端点の場合]

現在追跡中の枝の先端が、図22のように交差点ではなく、端点になっている場合には、2点追跡法により確認できるため、端点であるとして追跡を終了する。

4.1.2 交差点探索子の動作

枝追跡子が追跡を終了時に、前方に交差点があると判断され、かつその交差点が未探索であった場合には交差点探索子が設定される。図23を例に、交差点探索子の動作を説明する。

交差点探索子は、まず図23(a)に示すように、PLVの先端から反時計回りに90度方向にBSSVを設定し、反対側の輪郭点を見つける。反対側の輪郭点が見つからない場合には、交差点探索不可能として処理される。輪郭点が見つかった場合には、同図(b)に示すように、その輪郭点から時計回り8連結で、輪郭線を2N+1ステップ追跡し、PISV,FISVを設定し角点を検出する。さらに今度は、図23(c)に示すように、PISVの先端から時計回り90度方向にBSSVを設定し、反対側の輪郭点を見つける。以下同様に、この操作を繰り返し、図23(e)に示すようにFLVが見つかるまで行う。

4.2 回路結線の認識

図24に示すように、先に求めた連結点Dataの点対とゲート認識DataのInputLine, OutputLineの点対、および画面への入出力線の点対を、枝追跡待ち行列"BranchiQ"の初期データとして設定しておき、原画像に対して、procedure Graphsearchを動作することにより、グラフ構造を抽出する。ここで注意しなくてはならないことがある。それは、交差点探索子が動作中に、"BranchQ"に記憶されている節点対が発見された場合の処理の問題である。4.1.2の交差点探索子の動作では、角を決定した後、反対側の輪郭点を発見すべく、BSSVが設定される。しかし、連結点の抽出

表1　枝抽出情報

Branch label	Intersection label of both end points	
B1	I1	GI1
B2	I2	C2a
B3	I3	GI6
B4	C2b	GI5
B5	C2c	GI2
B6	C1a	GO2
B7	C1b	X1
B8	C1c	O1
B9	C3c	X1
B10	C3a	GO4
B11	C3b	O2
B12	GO1	GI3
B13	GO3	GI8
B14	GI7	X1
B15	GI4	X1

Label X1 means new intersection

表2　節連結情報

Node label	Number of branches	Branch label of the node
I1	1	B1
I2	1	B2
I3	1	B3
C1a	1	B6
C1b	1	B7
C1c	1	B8
C2a	1	B2
C2b	1	B4
C2c	1	B5
C3a	1	B10
C3b	1	B11
C3c	1	B9
GI1	1	B1
GI2	1	B5
GI3	1	B12
GI4	1	B15
GI5	1	B4
GI6	1	B3
GI7	1	B14
GI8	1	B13
GO1	1	B12
GO2	1	B6
GO3	1	B13
GO4	1	B10
O1	1	B8
O2	1	B11
X1	4	B7 B9 B14 B15

を行なった場合には "Branch0 " に記憶されている節点対により、対応点が分かっているため、交差点探索子は、BSSVを設定せずに、発見された節点対を結ぶベクトルを、BSSVの代わりに用いることにより、探索を続けることが可能となる。

図24の初期データに対して、グラフ構造を抽出したものが表1、表2、および、図25である。

第2章の(3) で述べたように、結線を示す領域に出現する交差点の枝数は4つである。そこで、グラフ構造情報から、交差点では、第1枝と第3枝、第2枝と第4枝がそれぞれ同じ線を示していることがわかる。そこで、第1枝と第3枝、第2枝と第4枝にそれぞれ同じ枝ラベルを与える。表2の交差点ラベルX1には、時計回りにB7,B9,B14,B15 の順に、枝が流れ込んでいるが、B7とB14,B9とB15 が同じ線だと認識できる。

4．3　論理式の決定

表1、表2のグラフ構造から、与えられた図面の論理式が決定される。図面に対する入出力線には、あらかじめラベルを与えておく。そこで、図面への入出力ラベルが与えられている枝と同一の結線を表わしている節・枝には、入出力ラベルを与える。また、各論理ゲートの出力線に対応している節・枝で、かつ、入出力ラベルが付かなかったものには、最も古い枝ラベルを、結線のラベルとする。以上の手続きに従って、入力した論理回路図に対応する論理式を決定する。

図5に示す論理回路図に対応する出力された論理式を、式6 に示す。

```
Output1 = B12 NOR Output2
Output2 = B13 NOR Output1
  B12   = Input1 NAND Input2
  B13   = Input2 NAND Input3          (6)
```

式6 から、B12,B13 を消去した式7 のような論理式の導出は、リスト処理を行なうことにより、可能であるが、論理回路が複雑になると、かなり複雑なリスト処理が必要になるため、本実験では、このような処理は行なっていない。

```
Output1 =(Input1 NAND Input2) NOR Output2
Output2 =(Input2 NAND Input3) NOR Output1 (7)
```

5．論理回路図の認識実験

以上述べたアルゴリズムに従って論理図面の認識実験を行なった。始めにテンプレートを用いて描かれた図26の図面に対して適用した結果が図27～29である。すなわち、連結点を消去した画像が図27、論理ゲートを抽出した画像が図28、結線関係の認識を行なった画像が図29である。これらの情報を統合した結果を式8に示す。

```
Output1 = B8 NAND Output2
Output2 = B9 NAND Output1
  B8    = Input1 NAND Input2
  B9    = Input2 NAND Input3          (8)
```

図26はテンプレートを用いて描かれた論理図面であるが、荒めに描かれた図面に対しても、図30の画像に対して、連結点を消去した画像が図31、論理ゲートを抽出した画像が図32、結線関係の認識を行なった画像が図33である。これらの情報を統合した結果を式9 に示す。

```
Output1 = B11 AND B12
Output2 = Input1 AND Input2
  B11   = Input1 OR Input2
  B12   = NOT Output2                 (9)
```

この認識実験の結果より、本システムがきれいに書かれた図面のみならず多少荒めに手書きされた図面に対しても有効であることが分かる。

6．　おわりに

本報告は、パラレル・ベクトル・トレーサによるグラフ構造の抽出アルゴリズムを適用した簡単な論理図面の認識について述べた。

ここで提案した方式は、枝追跡子と交差点探索子という2つのトレーサを使用し、連結関係を忠実に抽出する手法を用いたことにより、多少荒めに手書きされた論理図面のパターン認識へも応用可能であることは明らかとなった。

今後、各種の図面に対する計算機自動解析処理手法として適用することを行う予定である。

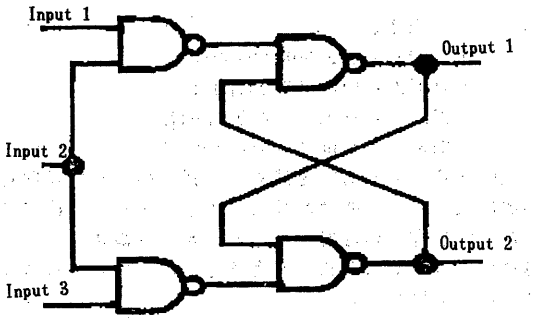


図26　実験で用いた論理回路図２

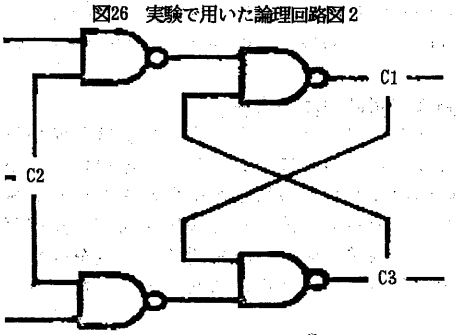


図27　連結点を消去した画像２

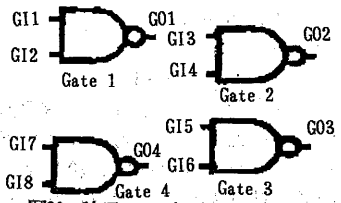


図28　論理ゲート領域を抽出した画像２

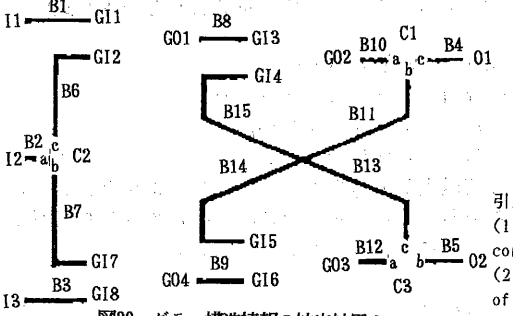


図29　グラフ構造情報の抽出結果２

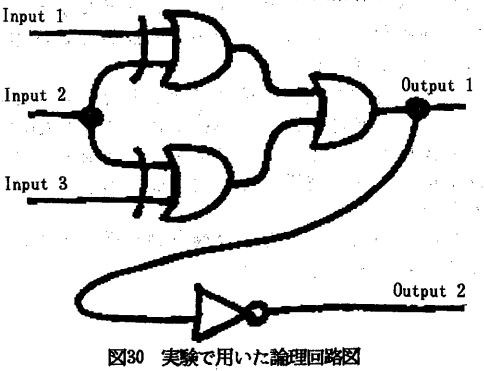


図30　実験で用いた論理回路図

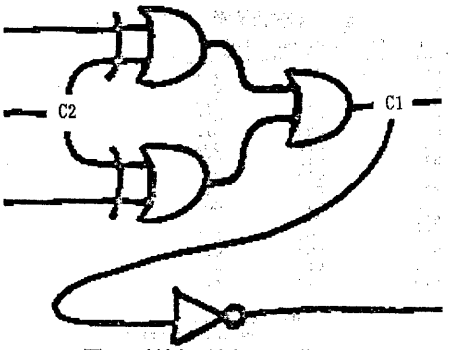
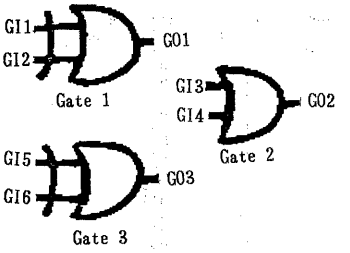


図31　連結点を消去した画像



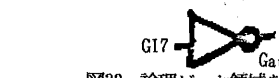


図32　論理ゲート領域を抽出した画像

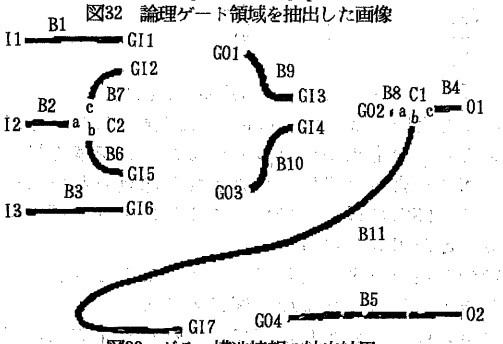


図33　グラフ構造情報の抽出結果

引用文献

(1) T.Agui, T.furukawa : A method for extractingand restoring of contour lines on a contour map,IECE of Japan ; E63, 8, p.581(1980)

(2) T.AGUI,K.Matsubara and M.Nakajima:Sequentialcomputer processing of a collection of closedcurves and its application to pattern recognition,IECE of Japan, E64, 10,p.661(1981)

(3) T.Agui , K.Matsubara and M.Nakajima : Anautomatic generation algorighm of colsedcurvesin contoure map processing, ECE of Japan, E64,9,p.610 (1981)

(4) 安居院、伊藤、中嶋：等高線群に対する効率的な高度付与方法、信学論、J64-D,12, P.1507(1982)

(5) 安居院、飯塚、中嶋：ピラミッド階層構造データの位相変化情報を利用した市街化地図の処理、信学論、J64-D,10, P.1243(1982)

(6) 中嶋、安居院、飯塚：市街化地図情報処理に関する研究、第２報、信学技報、PAL84-20

(7) S.Kimura ,T.Agui and N.Hoshina : An algorithm for extracting a solid object from three views,IECEof Japan, J66-E, 1, p.51 (1983)

(8) 吉田他：手書き図面の自動認識、ＴＶ誌、vol35,No11

(9) 久保田他：線順次アルゴリズムを用いた論理回路図面入力法、信学技報PAL82-17(10)名倉他：文字認識機能を導入したファクシミリベースの手書き図面清書法、信学技報、IE82-12