目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

従来の画像処理システムにおいては、受光系と処理系が別々でありその間は信号線で接続されデータ転送を行なっている。しかし画像の高解像度化や扱うフレーム数の増加に伴い扱う情報量が増加してきているが、リアルタイムでの画像処理を考えた場合、後段に続く処理系での処理時間があるため画像データを転送する時間は限られ、多量のデータを転送することは困難なものとなってくる。

また、半導体技術の発展により、処理系のデータ処理能力は飛躍的に向上してきた。しかし近年ムーアの法則に沿った発展にはかげりが見えはじめ、いずれは集積度やクロック速度の発展は飽和状態となることが予想される。そのため、さらなる性能向上のためにはこれまでの処理方法と異なる革新的な手法をとらざるを得ない。

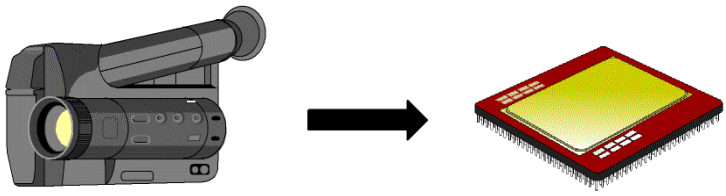


図 1.1: 従来の画像処理システム

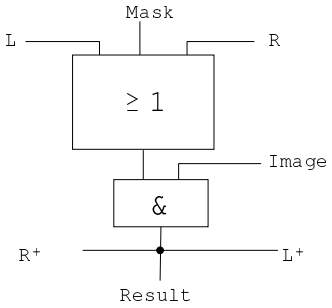


図 1.2: マスキング処理回路

1.1.1 スマートセンサ

この問題を解決する一手法として、集積回路技術の進歩により、受光系に処理系の一部を集積することが可能となり、いわゆるスマート・センサと呼ばれるものが現実のものとなってきた。[1] スマート・センサは受光系と処理系が同一チップに集積されているため、データ転送にかかる時間は従来のシステムと比べて短縮され、処理形式も従来のようにシーケンシャルな処理ではなく並列的に処理を行なうことにより高速な動作が可能であり、ノイズ除去やエッジ検出といった通常の画像処理のうちでの前処理部分を受光系とともに集積する例が多く報告されている。[2, 3] 以下にその一例を示す。

図 1.2 に示すような回路を一列に並べて、任意の領域をマスキングしその他の領域を消去する手法がある。[4] 図 1.3(a) に注目すると、 図 1.2 の上部の Mask が “種” となり上段から ‘1’ が出力される。そのため次段では Image がそのまま通り抜け Image が ‘1’ であれば Result も ‘1’ となり近傍へ伝わっていく。そして Image が ‘0’ となる領域まで伝搬して行き、結果マスクを ‘1’ とした Image‘1’ の領域がそのまま残ることになる。また Image が ‘1’ の領域でマスクを ‘1’ としなかった領域は “種” ができず Result は全て ‘0’ となり、マスキングができることになる。

これを 2 次元に拡張したものが図 1.2(b) であり、マスクを付けた領域のみが残っているのが見られる。

このようにして、比較的単純な機能のセルを複数並べて、従来の方法より高速

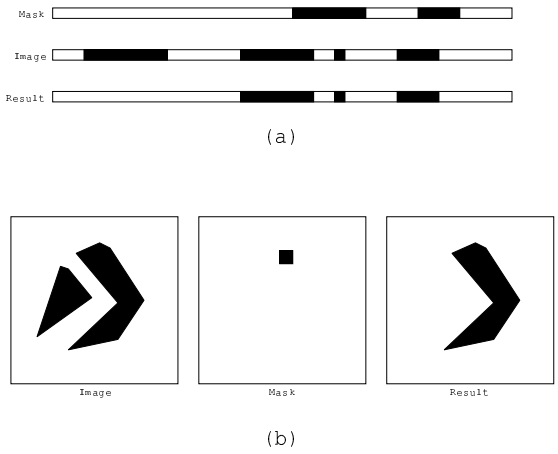


図 1.3: マスキングの一例

に画像処理の前処理部分を行なうことができる。

1.1.2 人間における視覚システム

しかし上に述べたような画像処理では、受光系から得られた画像を画素ごとの数字として見ており、人間が見るための画像をつくり出しているに過ぎず、人間の視覚システムが行っている画像の意味認識とは根本的に異なってくる。

図 1.4 に人間の視覚情報処理の流れを、表 1.1 に各部における処理機能について示す。[5] 受光系である網膜において細胞が互いに情報を伝達し重み付けをすることによってスムージングが行なわれ、脳内へ向かうにつれ、各部位でそれぞれの機能を持った細胞が反応し情報のやりとりを行なうことにより、形や色の構造抽出、動き検出、パターン認識へと高度な画像システムを構築している。

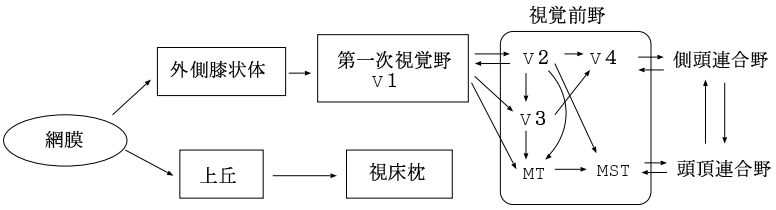


図 1.4: 人間の視覚システム

	機能
網膜	光電変換、順応レベルの適応、空間コントラスト、 時間変化の検出、三色過程からの反対色過程
V1	形の特徴抽出、動きの特徴抽出、両眼視差、色の処理
V2	立体情報、形の抽出、色の処理
V3	方位方向選択性、両眼視差選択性
V4	色の処理、色の恒常性、形の処理
MT	局所的な動きの統合
MST	オプティカルフロー

表 1.1: 各部の処理機能

1.2　本研究の目的

CCD カメラに代表される従来の撮像系や上に述べたようなスマートセンサでは、画像自身の意味を認識しようとするわけではなく、あくまでも人間が見るための画像を対象としており、画像の意味認識は人間の仕事としている。

本研究では、画像処理の一種である物体の位置、大きさの検出を行なう方法として、比較的単純な機能のセルが集まり、セルの集合として機能する人間の視覚システムの構造に着目し、従来のマトリクス状の数値列に対する逐次処理による画像処理とは異なり、比較的単純な機能を有するオートマトンを平面状に配置した構造によって、画素の空間的二次元相関を考慮しつつ、並列処理をすることによって物体のおよその位置と大きさを検出するアルゴリズムを提案する。またその回路構成を検討し、受光系と処理系を集積したスマート・センサとして実現する方法についても検討すると同時に、本手法が実時間で処理可能なアルゴリズムであることを示す。

本論文の構成は次のようになる。

2 章で正方形領域検出アルゴリズムを 1 次元の場合について考え、2 次元へと拡張する。また、その処理過程について考察し、実画像を用いて正方形領域の検出を行なう。3 章では検出した物体に内包される正方形領域の情報をもとに物体の位置と大きさを検出する 2 つの方法について考える。

以上の結果より、4 章では本手法の回路構成について考え、実際に設計した回路を示し測定結果について考察し、本研究で得られた結果について 5 章で述べる。

第2章　平面配置オートマトンによる矩形領域検出

図 2.1 のようにマトリクス状に配置された画素の間にオートマトンを配置した構造を考える。各オートマトンは、4 近傍の画素またはオートマトンからの情報を受け取り、クロックに同期して遷移するとする。以下にその手法を、まず 1 次元の場合で示し、つづいて 2 次元の場合に拡張する。また本手法を任意図形について用いた場合の処理過程を示したのち、実画像を用いた場合について述べる。

2.1　1 次元配列による検出

図 2.2 のような構造を考える。ここで最上段の長方形は画素であり、灰色は対象物体に属する (論理的に '1') ことを、白色は属さない (論理的に '0') ことを表してる。また 2 段目以下の丸印はノードを表し、以下の手順で接続する画素またはノードの論理積を取るとする。ただし d_j は左から j 番目の画素の値を表し、$S_{i,j}$ は i 段目の左から j 番目のノードの値を表す。

1. まず 1 段目のノードで接続する隣接画素の論理積をとる

$$S_{1,j} = d_j \cdot d_{j+1} \tag{2.1}$$

2. 次に、2 段目のノードで接続する上位の隣接ノードの値の論理積をとる

$$S_{2,j} = S_{1,j-1} \cdot S_{1,j+1} \tag{2.2}$$

3. 以下、同様の手順を全てのノードが 0 になるまで繰り返す。

$$S_{i+1,j} = S_{i,j-1} \cdot S_{i,j+1} \tag{2.3}$$

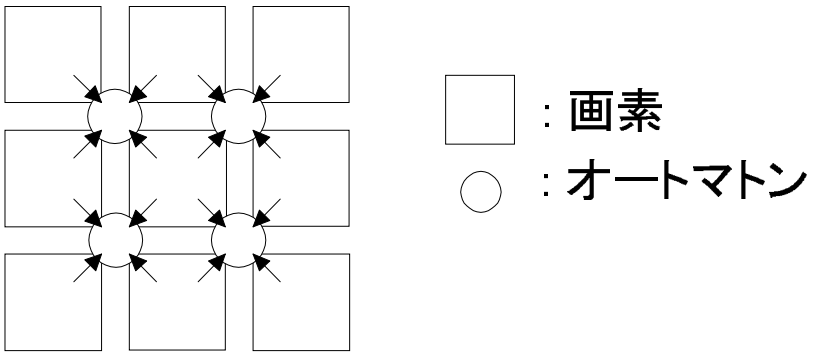


図 2.1: 画素とオートマトンの配置

ここで、各 i における状態について説明する。まず、$i = 1$ でノードの値が ‘1’ であることは、隣接画素の論理積をとった結果であるため大きさが 2 の画素 ‘1’ の領域に対応する。次の $i = 2$ では、$i = 1$ の結果のさらに論理積をとるため、ノードの値が ‘1’ であるということは、大きさ 2 の画素 ‘1’ の領域が重なることなく隣接して 2 つ存在する、つまり大きさ 4 の画素 ‘1’ の領域が存在していることを意味する。続く $i = 3$ でノードの値が ‘1’ であることは、大きさ 4 の画素 ‘1’ の領域が 2 つ存在していることを示すが、図 2.2 の $i = 1, 2$ の場合からわかるように、これらの領域は大きさ 2 だけ重なっているため、結果として大きさ 6 の画素 ‘1’ の領域が存在していることを意味している。以降同様に考えていくと、ステップ数 i が進むに従って、ノードの値が ‘1’ であることに対応する画素 ‘1’ の領域の大きさは、2 ずつ増えて行くことがわかる。つまり $i = n$ のとき、大きさ $2n$ の画素 ‘1’ の領域が存在していることを表している。

式 (2.3) および図 2.2 から、本手法では各ノードが隣接する画素あるいはノードの情報によってのみ動作が決定するため、集積回路として実現する場合には、局所配線のみでよいという特長がある。なお、各段のノード同士の接続が同一であ

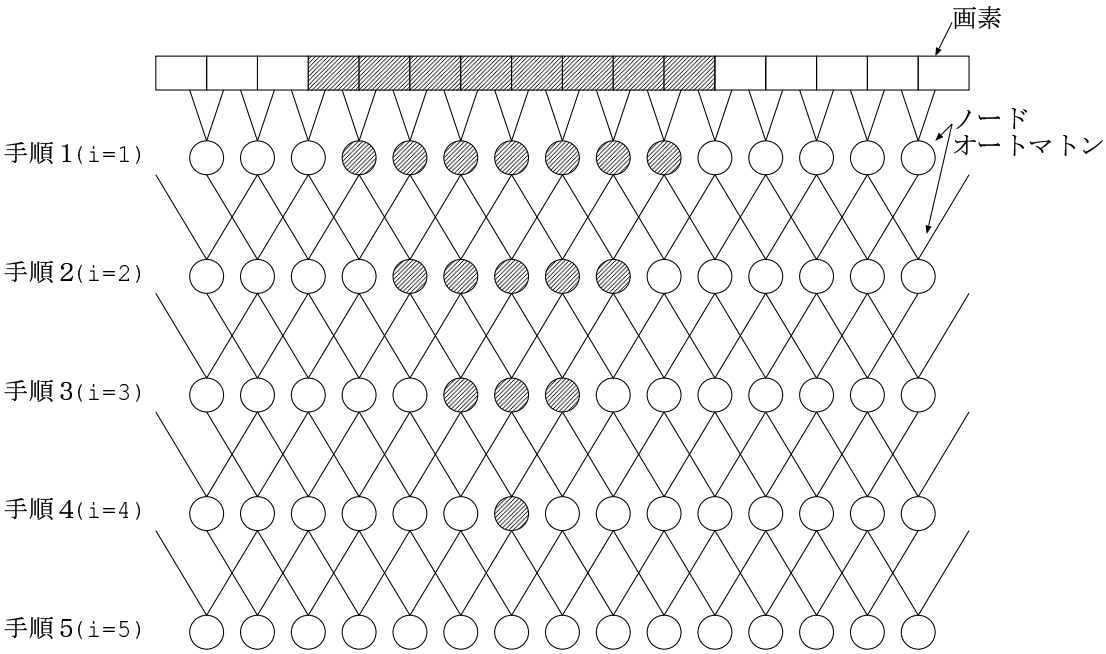


図 2.2: 1 次元配列による検出法

ることに注目すると、以上の手順はiを時間ステップとしたノードの遷移の時間変化を追っていると考えることができる。

2.2　２次元配列による検出

続いて、以上の手法を2次元に拡張することを考える。図2.3のように、マトリクス状に配置された画素とその間にノードを配置した構造を考える。1次元の場合は、各ノードは隣接する2つの画素またはノードの値の論理積をとったが、図2.3の2次元の場合においては隣接する4つの画素またはノードの値の論理積をとる。以下に2次元における2通りのアルゴリズムを示す。ただし、$d_{v,h}$は垂直方向v番目、水平方向h番目の画素を表し、$S_{i,v,h}$はi段目の垂直方向v番目、水平方向h番目のノードの値を表す。

2.2.1　ノードの対角線方向接続によるアルゴリズム

まず、二次元配列ノードの対角線方向接続によるアルゴリズムを以下に示す。

1.　まず、$i=1$で各ノードは隣接4近傍の画素の値の論理積をとる。

$$S_{1,v,h} = d_{v,h} \cdot d_{v,h+1} \cdot d_{v+1,h} \cdot d_{v+1,h+1} \tag{2.4}$$

2.　次に$i=2$では、$i=1$での各ノードの値を対角線方向の隣接4近傍のノードへ渡すと同時にそのノード自身も対角線方向の隣接4近傍のノードの値の論理積をとる。

$$\begin{aligned} S_{2,v,h} &= S_{1,v+1,h+1} \cdot S_{1,v+1,h-1} \\ &\quad \cdot S_{1,v-1,h+1} \cdot S_{1,v-1,h-1} \end{aligned} \tag{2.5}$$

3. 以下、全てのノードの値が0になるまで同様の手順を繰り返す

$$\begin{aligned} S_{i+1,v,h} &= S_{i,v+1,h+1} \cdot S_{i,v+1,h-1} \\ &\quad \cdot S_{i,v-1,h+1} \cdot S_{i,v-1,h-1} \end{aligned} \tag{2.6}$$

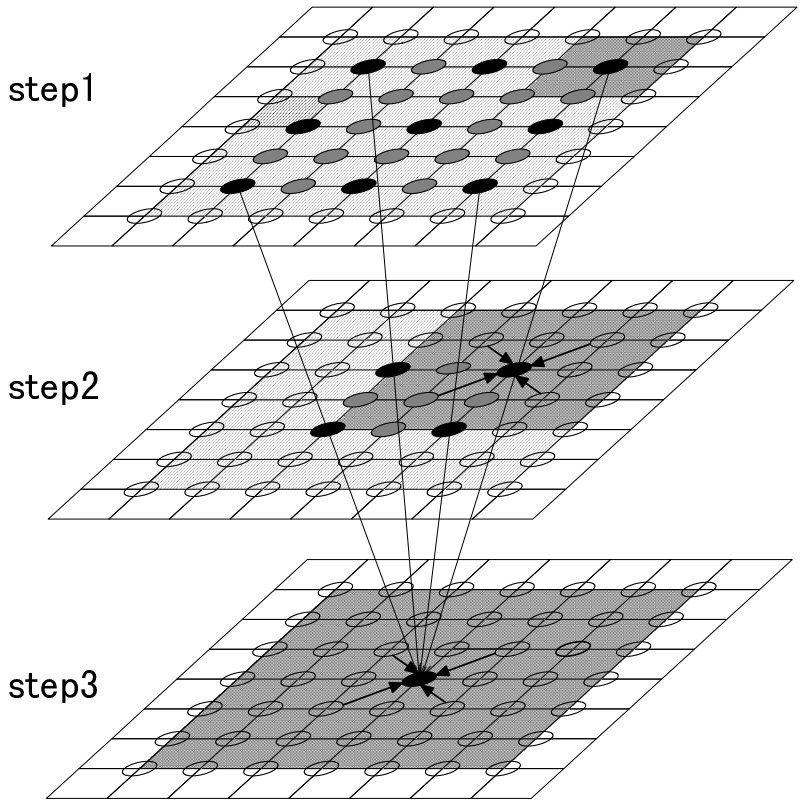


図 2.3: 二次元配列による検出法

$i = 1$ では、各ノードは隣接 4 画素の値の論理積をとるため、ノードが '1' であるということは図 2.4(a) のようにノードを中心とする 1 辺の大きさ 2 の画素 '1' の正方形領域に対応している。続く $i = 2$ では各ノードは隣接 4 ノードの値の論理積をとるため、値が '1' であるということは 1 辺の大きさ 4 の画素 '1' の領域に対応する図 2.4(b) を表している。1 次元の場合と同様に、$i = 3$ では値が '1' のノードは、1 辺の大きさ 6 の正方形の領域に対応し、図 2.4(c) のようになる。i とともに値が '1' のノードに対応する領域の大きさは増加し、ノードの値が '0' になったときの i に応じてそのノードを中心とする画素 '1' の最大正方形領域の大きさを知ることができる。

2.2.2 ノードの水平・垂直方向接続によるアルゴリズム

もう一つのアルゴリズムは、二次元配列ノードの水平・垂直方向接続による方法である。構造的には前者とノードの接続方法が異なるのみであるが、ノードが表す意味は異なり、手順は以下のようになる。

1. まず、$i = 1$ で各ノードは隣接 4 近傍の画素の値の論理積をとる。

$$S_{1,v,h} = d_{v,h} \cdot d_{v,h+1} \cdot d_{v+1,h} \cdot d_{v+1,h+1} \tag{2.7}$$

2. 次に $i = 2$ では、$i = 1$ での各ノードの値を水平・垂直方向の隣接 4 近傍のノードへ渡すと同時にそのノード自身も水平・垂直方向の隣接 4 近傍のノードの値の論理積をとる。

$$\begin{aligned} S_{2,v,h} &= S_{1,v,h+1} \cdot S_{1,v,h-1} \\ &\quad \cdot S_{1,v+1,h} \cdot S_{1,v-1,h} \end{aligned} \tag{2.8}$$

3. 以下、全てのノードの値が 0 になるまで同様の手順を繰り返す

$$\begin{aligned} S_{i+1,v,h} &= S_{i,v,h+1} \cdot S_{i,v,h-1} \\ &\quad \cdot S_{i,v+1,h} \cdot S_{i,v-1,h} \end{aligned} \tag{2.9}$$

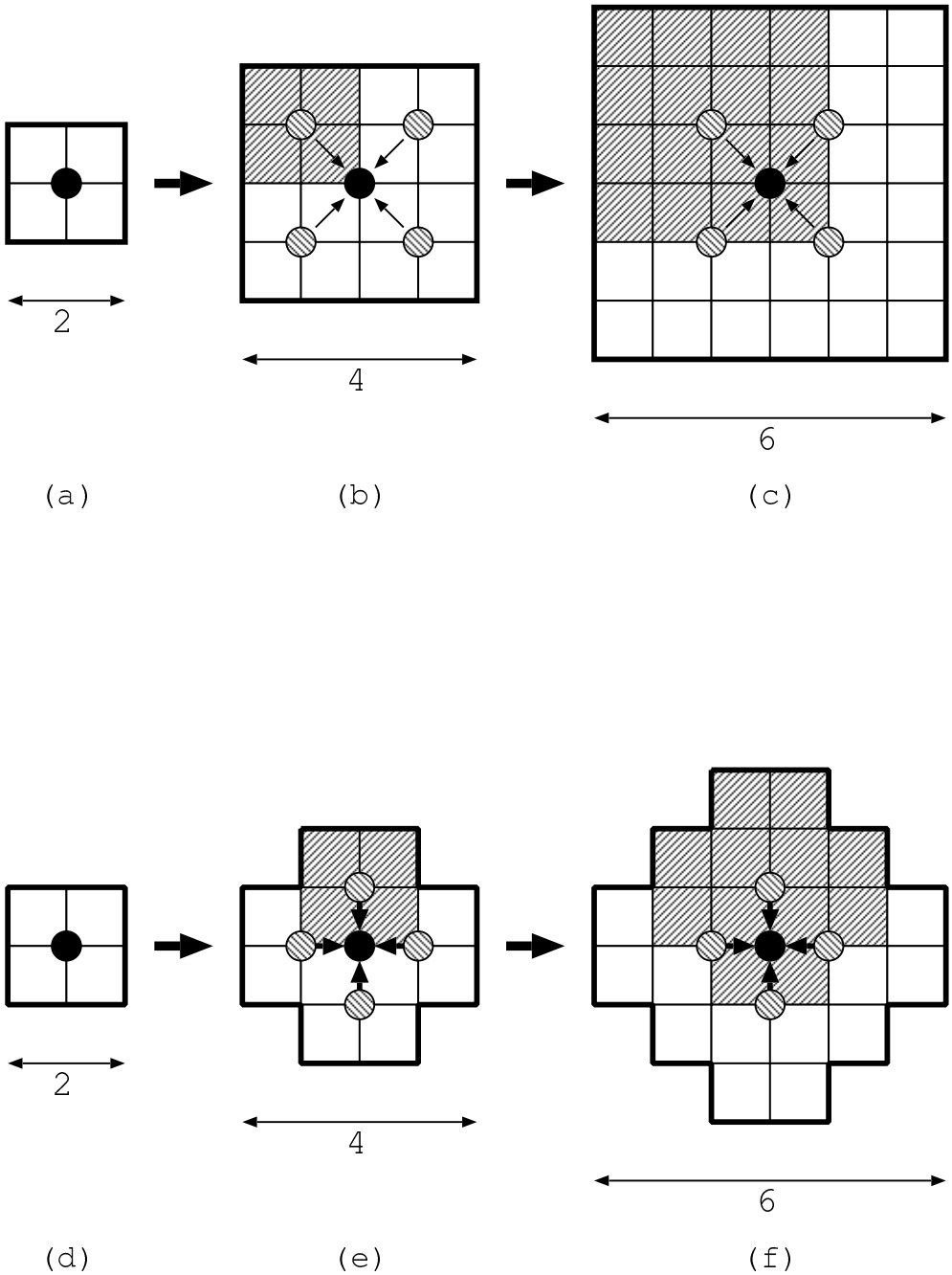


図 2.4: ノードが表す画素 '1' の領域の変化
(a)∼(c) 対角線ノード結合の場合と (d)∼(f) 水平・垂直ノード結合の場合

各ステップにおけるノードの ‘1’ が表す意味は、図 2.4(d)∼(f) のように画素 ‘1’ の領域を 45 度傾けた正方形を表し、各手順ごとに正方形の対角線方向の大きさが ‘2’,‘4’,‘6’ と大きくなっていく。

また、ここで正方形領域の検出に必要なステップ数について考えてみる。画素数 $m \times n$ $(m \geq n)$ の場合、ノードが ‘1’ である状態はステップが進むにつれ大きな正方形を表し、ステップ数 i において大きさ $2i$ の正方形を表している。つまり、最も検出にかかるステップが多いのは画素 ‘1’ の領域が $n \times n$ 以上の場合であり、$n/2$ 回のステップで検出が完了する。つまり画素数が増えた場合、処理にかかる時間は画素数や物体の面積ではなく、n の 1 乗に比例するため時間計算量は $O(n)$ となる。またノードの機能も単純であるため、回路で実現した場合にも高速動作が期待できる。

2.3 本手法ににおける処理過程

つぎに、図 2.5 の (a) のような任意の図形に対し 2 次元配置ノードの対角線方向接続によるアルゴリズムを用いたときの処理過程を図 2.5(b)∼(e) に示す。

図 2.5(a) は処理に用いた元画像で、正方形は画素を表す。図 2.5(b)∼(e) ではマトリクス状に配置した正方形はノードオートマトンを表し、黒色はノードオートマトンの状態が ‘1’ であることを表し、白色は ‘0’ を表す。(b) はノードオートマトンの隣接 4 近傍画素の論理積をとる手順 1 を行なった結果であり、(c) は対角線方向の隣接 4 近傍ノードオートマトンの論理積をとる手順 2 を行った結果である。また、(c) の灰色の部分は (b) に対し手順 2 を行ったときに ‘1’ から ‘0’ へと変わるノードオートマトンを表している。(c) と同様の手順を繰り返した結果が (d),(f) である。

ここで ‘1’ から ‘0’ へ変化するノードについて考えてみる。画素 ‘1’ の塊は物体を表し、ノードの ‘1’ はその物体に内包される正方形の中心を表しているため、正方形の大きさが大きくなるにつれ正方形の中心は物体の縁から遠くなる。そのため ‘1’ のノードの塊の縁に図 2.5 の灰色の領域として現われてくる。このアルゴリズムを別の観点からみると画素やノードの値の並びが、1 次元で考えたとき ‘00’, ‘01’, ‘10’ のいずれかのときにノードの値は ‘0’ となるが、‘11’ の場合のみ ‘1’ となることから、値が ‘1’ の領域の端の ‘10’, ‘01’ の状態を ‘0’ に変化させることになる。

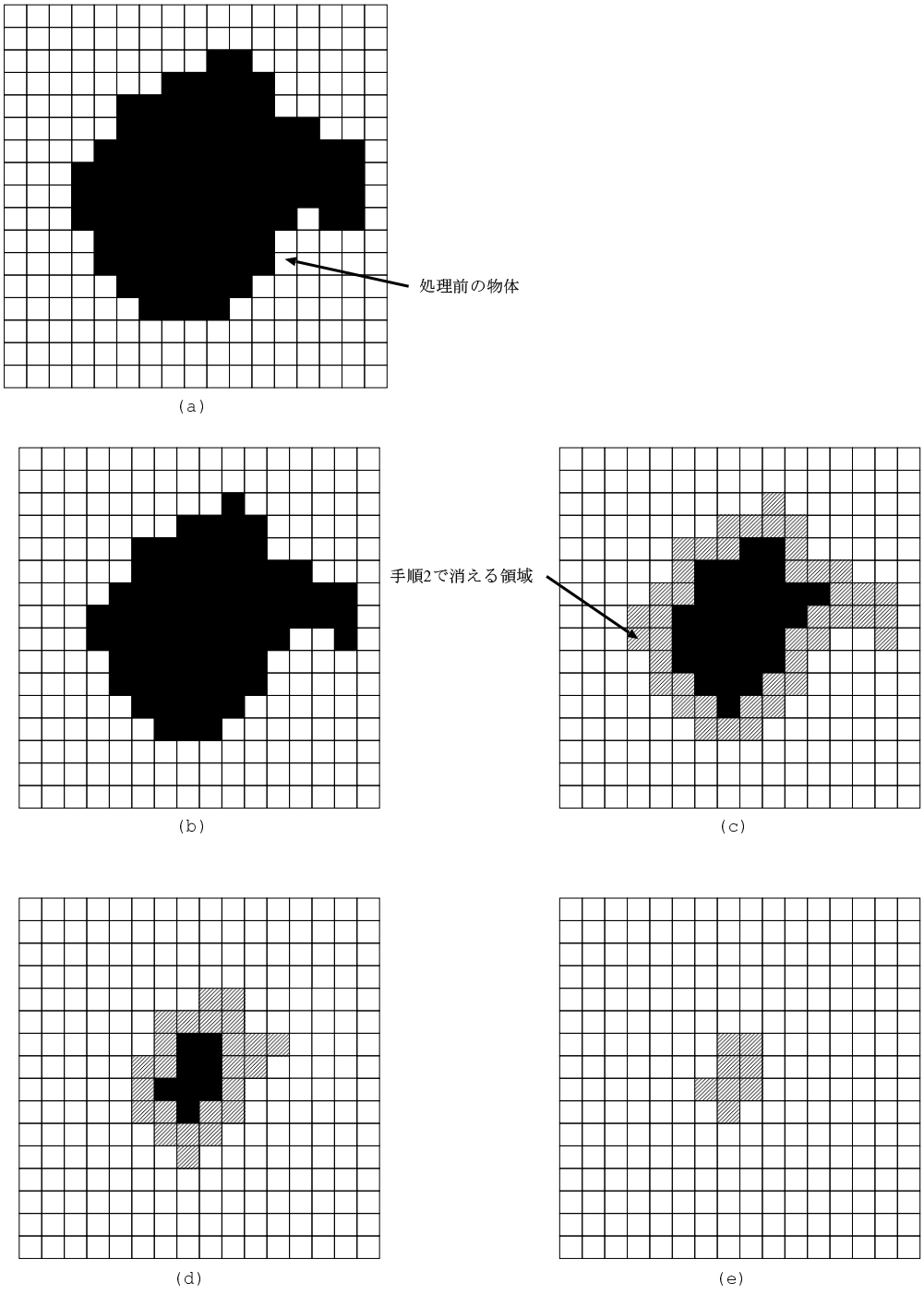


図 2.5: 任意図形 (a) における対角線ノード結合での処理過程 (b)∼(e)

これは 2 次元の場合でも同様で、物体を 1 皮づつむいていき、最後の「芯」を見つけ出しているといえる。

つぎにノードの水平・垂直方向接続によるアルゴリズムを用いたときの処理過程を図 2.6(a)∼(f) に示す。処理に用いた元画像は図 2.5(a) である。対角線方向接続によるアルゴリズムと似た動作をするが、ノードの ‘1’ から ‘0’ へ変化する仕方が異なる。ここでノードの ‘0’ の状態に注目すると、図 2.7 のように左下に ‘0’ を表す白色のノードがある場合、(a) ノードの対角線方向接続では図のノード A は左下のノードから見て対角線方向に位置するため ‘0’ が伝搬するが、(b) ノードの垂直・水平接続では、ノード B は左下のノードから ‘0’ が伝搬するところに位置しないため、‘1’ の黒色の状態が保持される。また画素 ‘0’ は物体でないところを表しているため連続して存在しており、それによりノードの ‘0’ も連続して存在する。このため、ノードの対角線方向接続では図 2.7(a) のように L 字形に曲がった領域 ‘1’ のノードの角では ‘1’ から ‘0’ へ変化する領域が水平・垂直方向へ連続していくのに対し、ノードの水平・垂直方向接続では ‘1’ から ‘0’ へ変化する領域は対角線方向へ連続していくことが図 2.6 に現われている。

2.4 実画像を用いた正方形領域検出

本手法は 2 値画像をターゲットとしているため、図 2.8 のような実画像を例として色情報を元に 2 値化し、図 2.9 のような 2 値画像へと変換した。図 2.9 を本手法のうちノードの対角線方向接続によるアルゴリズムを用いて処理を行った過程を図 2.10(a)∼(f) に示す。 図 2.10 の黒の 1 ドットはノードオートマトンが ‘1’ であることを示し、白の部分はノードオートマトンが ‘0’ であることを示している。図 2.10 の (a) から (f) へとステップが進むにつれ、ノードオートマトンが ‘1’ である領域が減少していく。例えば図 2.10 の (b) において、丸で囲んだ領域には ‘1’ の状態のノードオートマトンが存在するが、(c) の丸で囲んだ領域には ‘1’ の状態のノードオートマトンは無くなっている。つまり (c) の丸で囲んだ領域には、(b) におけるノードオートマトンが表す大きさ ‘4’ の正方形は存在するが、(c) における大きさ ‘6’ の正方形は存在しないことになる。同様にして、(e) の丸で囲んだ領域には大きさ ‘8’ までの正方形が存在することを表し、(f) には大きさ ‘10’ までの正方形が存在することになる。つまりステップ数 n において物体に内包される大き

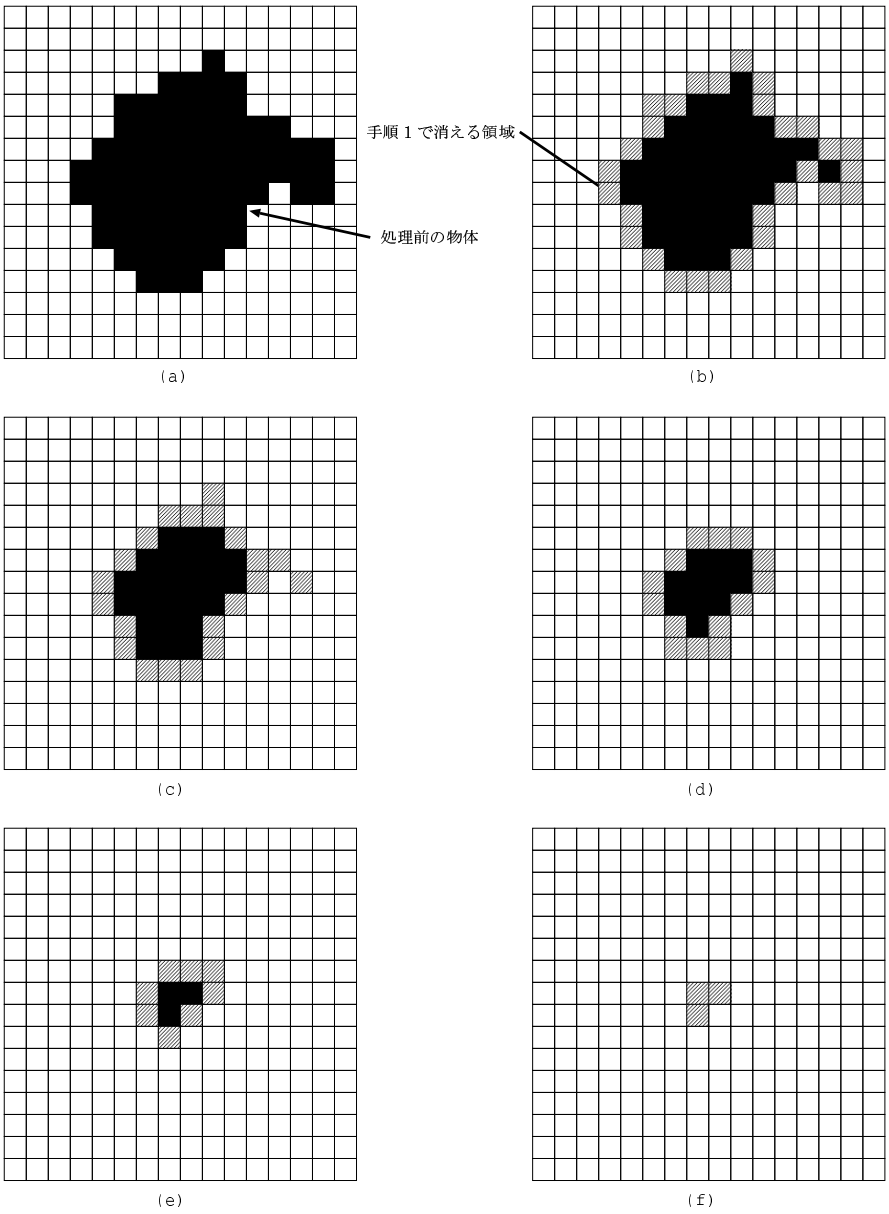


図 2.6: 任意図形における水平・垂直ノード結合での処理過程 (a)∼(f)

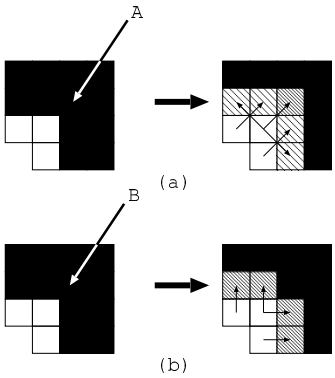


図 2.7: ノードの接続方法によるノードの ‘1’ から ‘0’ への違い (a) 対角線方向接続, (b) 水平・垂直方向接続

さ $2n$ の正方形領域の数が少なくなっていき、最終的にその物体に内包される最大もしくは極大正方形が現れていることがわかる。

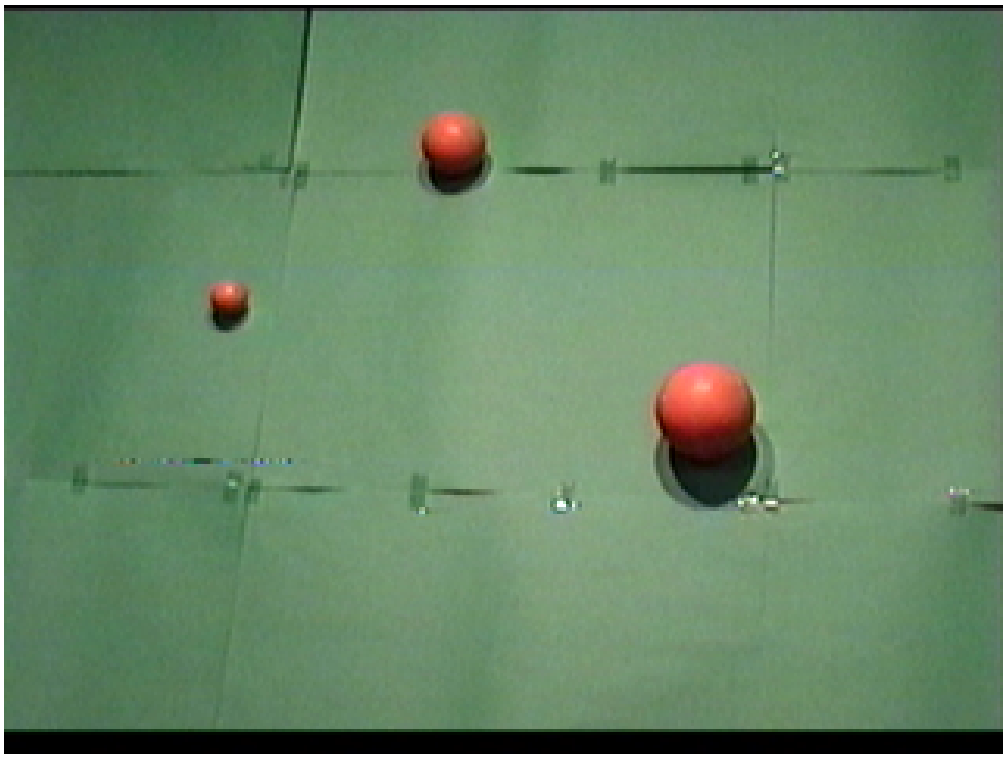


図 2.8: 正方形領域検出に用いた実画像

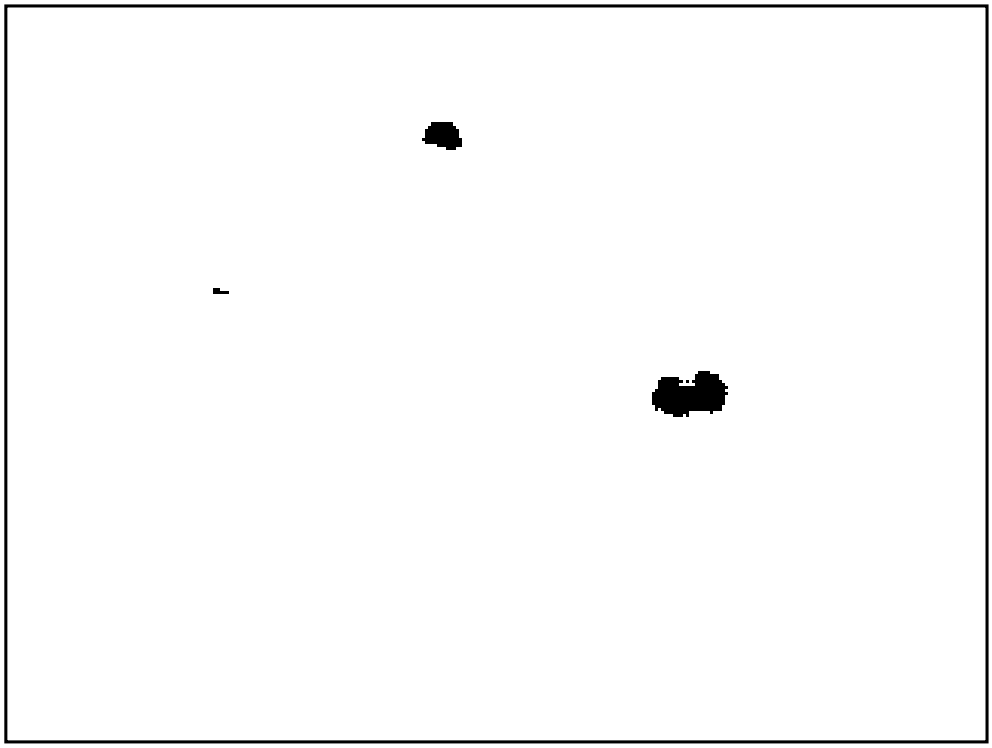


図 2.9: 色情報をもとに 2 値化した画像

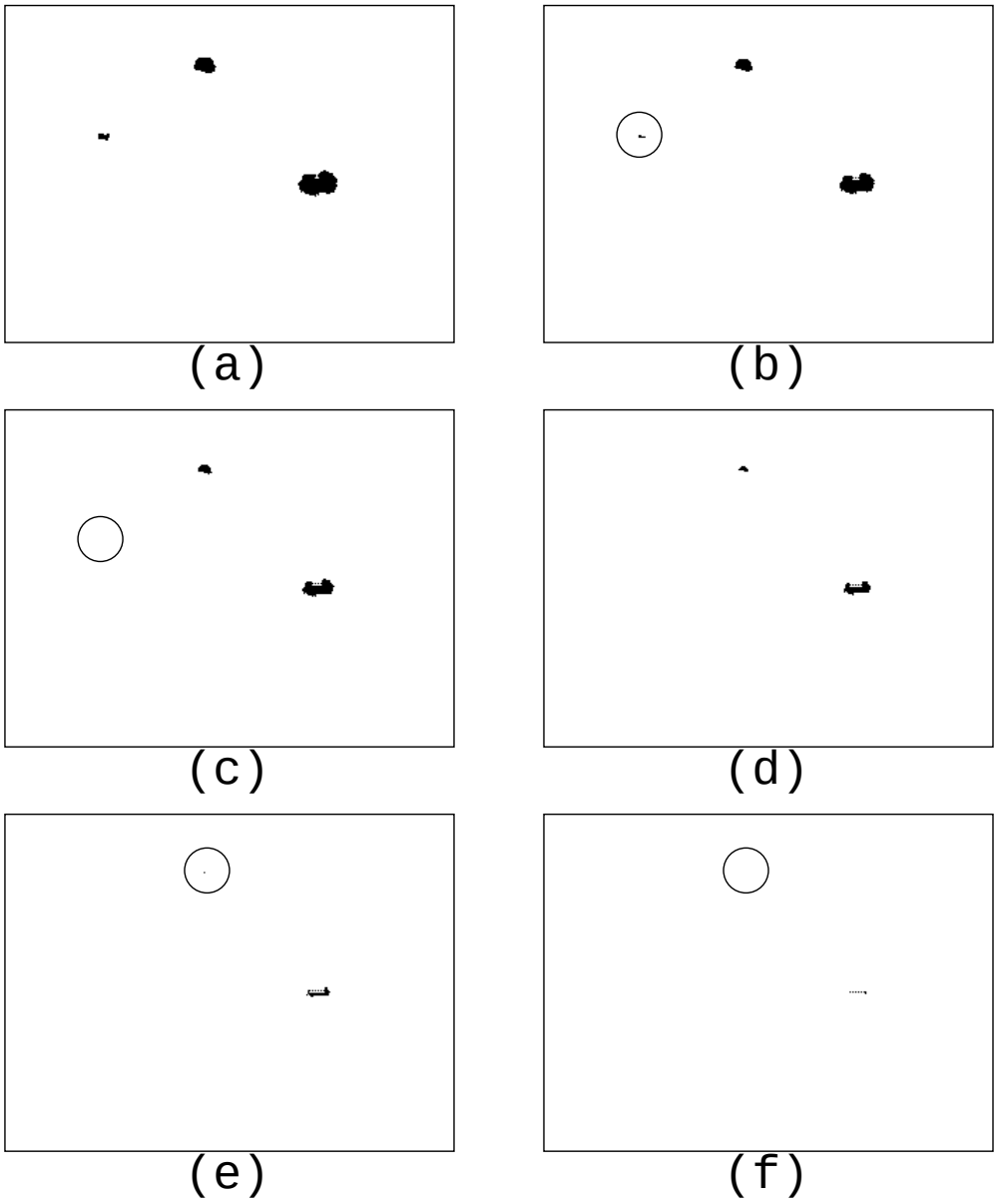


図 2.10: 正方形領域検出の過程 (a)∼(f)

第3章　物体の検出法

Chapter 2 で示した手法を用いることにより、物体に内包される各ステップに対応した大きさの正方形領域の存在を検出することができる。しかし、このままではマトリクス状のノード中に正方形領域の有無が示されているだけで、物体の位置と大きさを特定するためにはすべてのノードを検索しなければならない。つまり画素の増加に伴い検索にかかる時間が増加してしまう。

Chapter 2 において、本手法により処理を行う過程は、物体の「皮」をむいていき最後に「芯」が現れるようであることを示した。つまり最後に現れる「芯」は物体に内包される最大正方形を表しており、本手法による検出正方形のうちで、物体の大部分を占める正方形である。そこで、物体に内包される最大の正方形領域を物体を代表する点であると考え、最大正方形を検出することにより物体の位置と大きさを検出する方法の一例を考える。

3.1　水平・垂直方向への論理和出力による最大正方形検出

図 3.1 のように 各ノードの値を水平・垂直方向へ一列に論理和をとることで、ひとつでもノードの値が '1' であれば出力は '1' となり、各ステップで行または列に値が '1' のノードが存在しているかどうかを判断することができる。

ノードオートマトンの垂直方向論理和出力のステップ数 n における変化を図 3.2 に示す。ステップ数 n におけるノードオートマトンの出力を水平・垂直方向へ一列に論理和をとった結果を $H_{i,n}$, $V_{j,n}$ とする。ここで $H_{i,n}$ はステップ数 n における上から i 番目の水平方向へ一列の論理和をとった結果であり、$V_{j,n}$ はステップ数 n における左から j 番目の垂直方向へ一列の論理和をとった結果である。また図

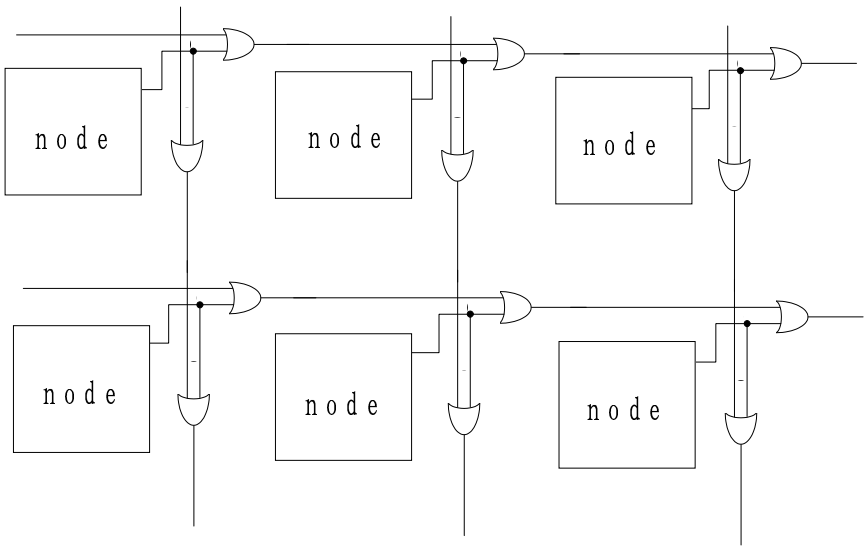


図 3.1: 正方形領域検出回路

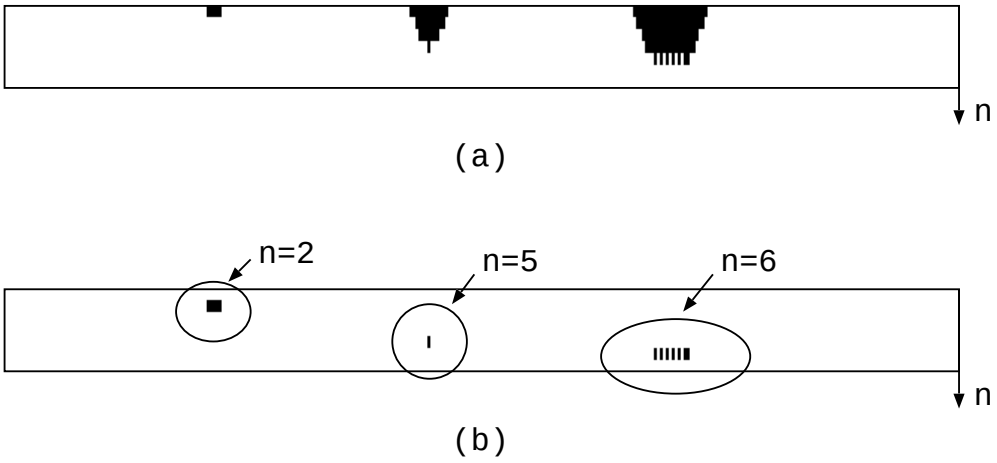


図 3.2: ノードの垂直方向論理和出力のステップ数 n における変化 (a) と頂点検出結果 (b)

3.2(a) では $V_{j,n} = 0$ のとき白のドットで示し、$V_{j,n} = 1$ のときは黒のドットで示している。

このとき $V_{j,n}$ が '0' であれば、ステップ数 n において大きさ $2n$ 以上の '1' の正方形領域の中心は、右から j 番目の垂直方向へ一列上には存在しないことを表している。また、$V_{j,n}$ が '1' であれば右から j 番目の垂直方向一列のいずれかに大きさ $2n$ の '1' の正方形領域が存在していることを表している。図 3.2(b) では (a) の '1' の塊のうち「山」の頂点を検出をした結果である。これは '1' の塊のうち最も大きな n の値と座標を検出しており、物体に内包される最大正方形の大きさ $2n$ と水平軸上の座標を検出している。図 3.2(b) では左から '4','10','12' の大きさの物体に内包される最大正方形が検出されていることがわかる。

水平方向に対しても同様な処理を行うことにより、大きさ $2n$ の最大正方形領域の中心が水平・垂直軸上のどこに存在するを知ることができる。図 3.3 はその結果であり、中心の図は処理に用いた元画像で、その下に示す図は垂直方向論理和出力のステップ数 n における変化とその頂点を検出した結果である。また右の図は水平方向論理和出力に対しての同様な図である。例えば図 3.3 の右上の物体 A に

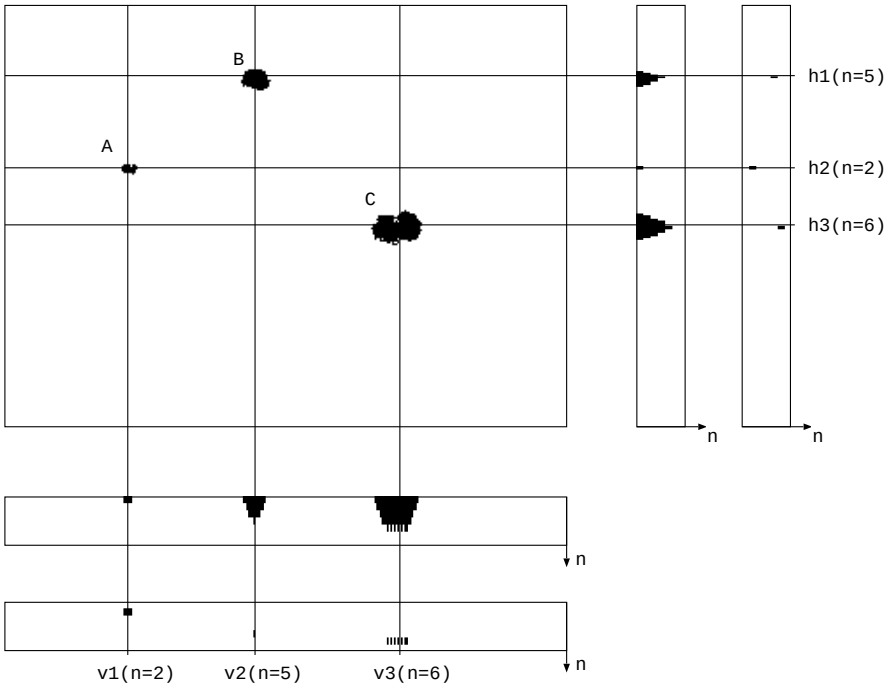


図 3.3: ノードの水平・垂直方向論理和出力のステップ数 n における変化と頂点検出結果

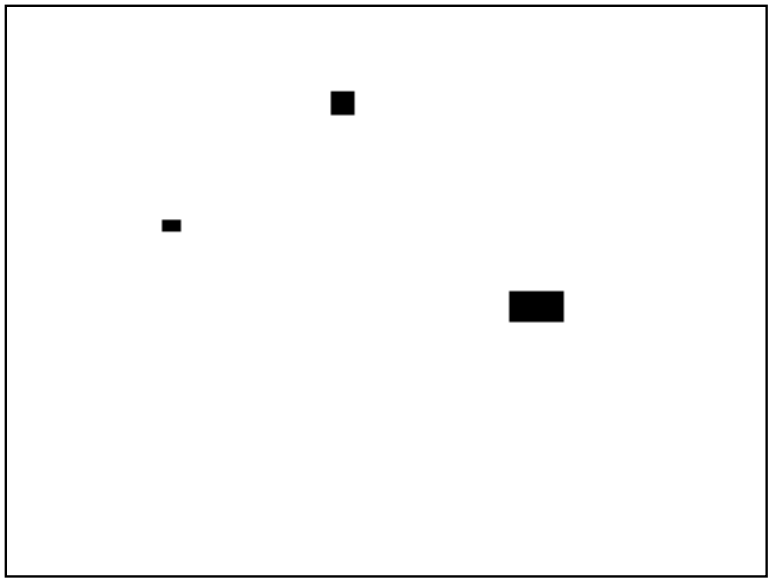


図 3.4: 各物体に内包される最大正方形の検出結果

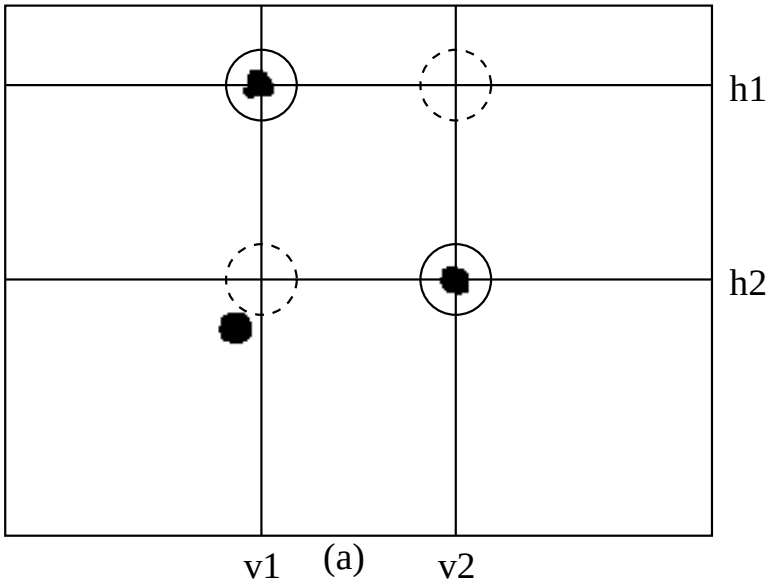
よって生ずる論理和出力より、v1,h2 の軸上で‘1’ の塊の頂点が $n = 2$ (大きさ‘4’) で検出される。つまり、座標 (v1,h2) で大きさ‘4’ の物体に内包される最大正方形があることがわかる。同様にして物体 B は座標 (v2,h1) で大きさ‘10’、物体 C は座標 (v3,h3) で大きさ‘12’ の物体に内包される最大正方形があることがわかる。

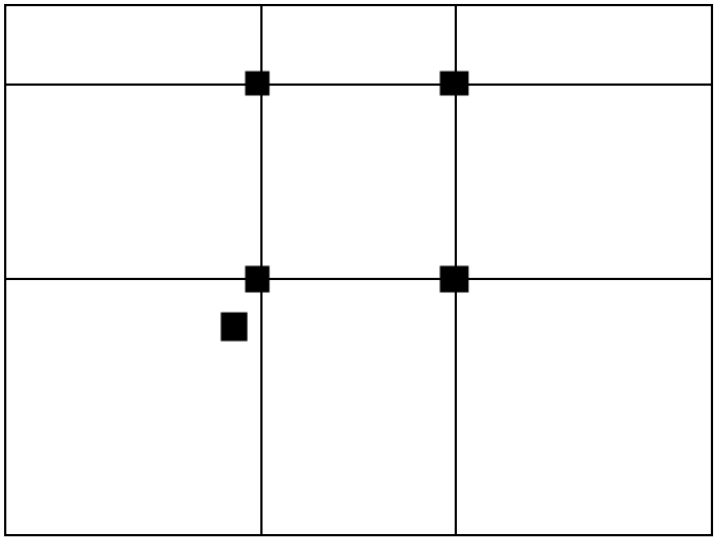
この結果をもとに物体を再現したものが図 3.4 である。v1 や v3 などは、1 点の座標ではなく複数の座標の集合であるため、再現物体は正方形ではなく矩形となっているが、物体のおよその大きさと位置を再現できていることがわかる。

この検出方は言い換えれば、水平・垂直方向それぞれから物体に光をあて、そのときにできる像をもとに水平方向の位置と垂直方向の位置を特定することと同じである。

ただしこの方法には 2 点ほど問題が生ずる。

- 図 3.5(a) のように同じ大きさの物体が 2 つ以上存在する場合
- 図 3.6(a) のように大きな物体と同軸上に小さな物体が存在する場合





(b)

図 3.5: 正方形領域検出時に「影」が出てしまう例の原画像 (a) と検出結果 (b)

まず同じ大きさの物体が 2 つ以上存在する場合について考える。図 3.5(a) は同じ大きさの物体に内包される最大正方形はその大きさが同じであるため、水平・垂直方向それぞれ同じステップ数 $n = i$ のときに水平・垂直方向論理和出力の「山」の頂点が v1,v2 と h1,h2 で検出される。これにより (v1,h1), (v1,h2), (v2,h1), (v2,h2) の 4 通りの組合せが考えられ、この情報をもとに物体の位置と個数を特定しようとした場合、図 3.5(b) のように実際と異なる虚像が現れてしまう。

つぎに大きな物体と同軸上に小さな物体が存在する場合について考える。図 3.6(a) の右側の物体のように、垂直もしくは水平方向に物体が重なっている場合には、図 3.6(b) のように大きい方の物体に小さい方の物体が隠れてしまう。そのため物体 A,B の水平方向の座標はそれぞれ h1,h2 であることを検出することはできるが、垂直方向の座標は大きな物体 C の影に隠れてしまうために特定することができない。そのため物体 A,B の座標は (?,h1),(?,h2) となり A 章に示す頂点検出時に現れるノイズと同じになってしまう。

しかし、これらのいずれの問題も軸方向に射影した像を用いることが原因であるので、物体の位置を特定する方法を工夫することにより解消されると考えられる。

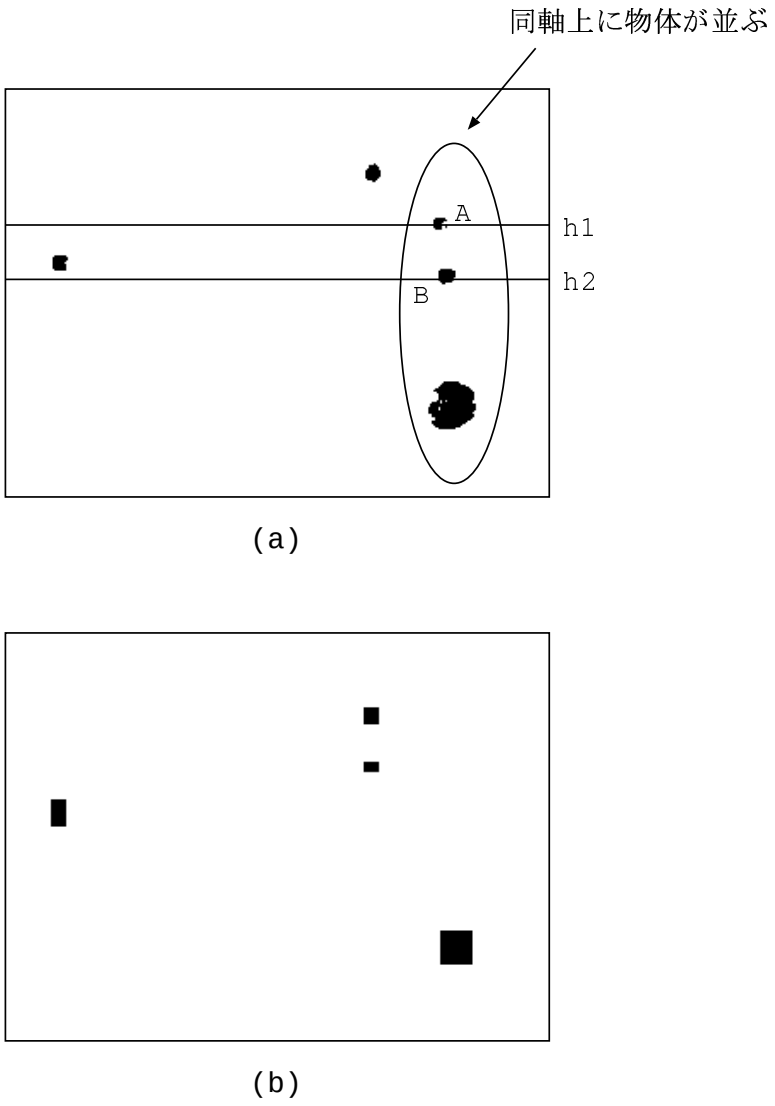


図 3.6: 正方形領域検出時に小さな物体が大きな物体に隠れてしまう例の原画像 (a) と検出結果 (b)

3.2 最大・極大正方形の4進木探索法を用いた検出

Chapter 3.1 では大きな物体に隠れて影が生ずる問題があった。大きな物体によって生ずるノードオートマトンの '1' の状態は、小さな物体によって生ずる '1' よりも長いステップの間残っている。そのため、ノードオートマトンの状態を一列に論理和をとると小さな物体は大きな物体の影に隠れてしまう。そこで物体に内包される最大・極大の正方形領域を表すノードオートマトンのみを検出し、それにフラグを立て4進木探索法により検出する方法を考える。

3.2.1 最大・極大正方形を表すノードオートマトンの検出

正方形領域検出アルゴリズムでは、対象物体の「皮」を1ドットずつむいていくのことと等価であることを Chapter 2.3 で示した。つまり1ドット「皮」をむいたとき、その「皮」が物体の「芯」であれば最後に残った正方形であるため、物体に内包される最大正方形を表していることになる。

これは、

- 「皮」である = ノードの値が '1' から '0' へ変化
- 「芯」である = ノードの値が '1' から '0' へ変化したとき、まわりのノードも '0' である

と考えることができる。図3.7のような 3×3 の中心ノードとそのまわりに注目した場合を考える。(a) は最後に残った1ドットであるため中心ノードは明らかに最大正方形を表している。(b) は物体の「芯」を表していると考えられる。しかし (c) のように 3×5 で考えてみると (b) の場合は物体の内包される極大正方形である場合も考えられる。また (d) のように中心ノードが '1' から '0' へ変化した場合にまわりのノードに '1' の状態が存在すればその中心ノードは「皮」であり「芯」ではないことになる。

以上をまとめると、(a),(b),(c) の状態変化の場合、中心ノードは最大もしくは極大正方形を表し、(d) の場合には最大・極大正方形でないと言える。これを式で表すと式 (3.1) のようになる。ここで $F_{i,v,h}$ は手順 i での垂直方向 v 番目、水平方向 h 番目のノードが最大もしくは極大正方形を表すフラグであり、$F_{i,v,h} = 1$ のと

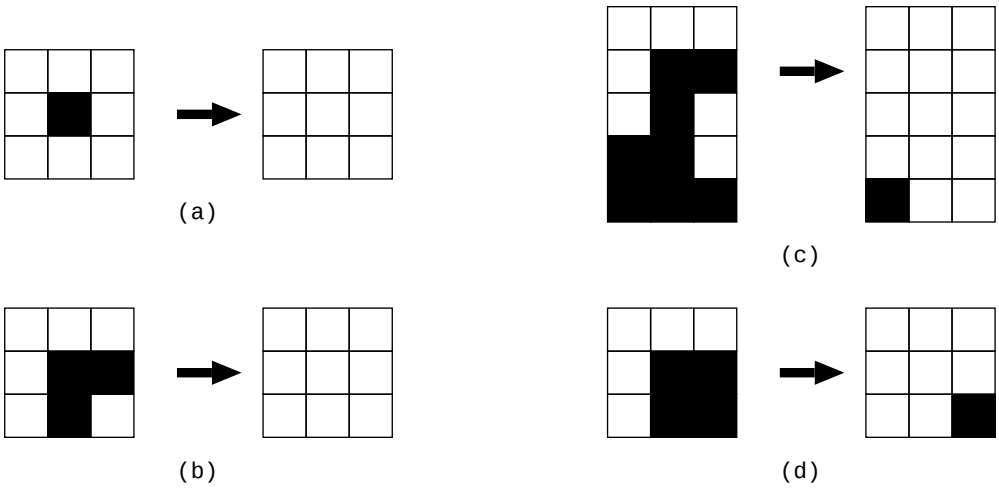


図 3.7: 中心ノードが (a),(b),(c) 最大・極大正方形である場合と (d) 最大正方形でない場合

きノードは最大・極大正方形であり、$F_{i,v,h} = 0$ のときは最大・極大正方形でないことを表す。また $S_{i,v,h}$ は手順 i での垂直方向 v 番目、水平方向 h 番目のノードの状態を表す。

$$\begin{aligned} F_{i,v,h} \quad = \quad & \left(S_{i-1,v,h} \cdot \overline{S_{i,v,h}}\right) \\ & \cdot \left(\overline{S_{i,v-1,h-1}} \cdot \overline{S_{i,v,h-1}} \cdot \overline{S_{i,v+1,h-1}}\right. \\ & \cdot \overline{S_{i,v-1,h}} \cdot \overline{S_{i,v+1,h}} \\ & \left. \cdot \overline{S_{i,v-1,h+1}} \cdot \overline{S_{i,v,h+1}} \cdot \overline{S_{i,v+1,h+1}}\right) \end{aligned} \tag{3.1}$$

対角線方向ノード接続と水平・垂直方向ノード接続による正方形領域検出過程の図 2.5, 図 2.6 それぞれに対し最大・極大正方形検出の過程を図 3.8, 図 3.9 に示す。図の黒色は最大・極大正方形フラグが立っていることを表し、濃い灰色はノードが '1' から '0' へ変化したノードを表し、薄い灰色はノードが '1' であることを表している。また図 3.8(a)∼(d) は手順 1 から 4 を表し、図 3.9(a)∼(f) は手順 1 から 6 を表す。Chapter2.3 で、対角線方向ノード接続による正方形領域検出はノードの '1' から '0' への変化が水平・垂直方向に継っていくことを示した。そのため、図

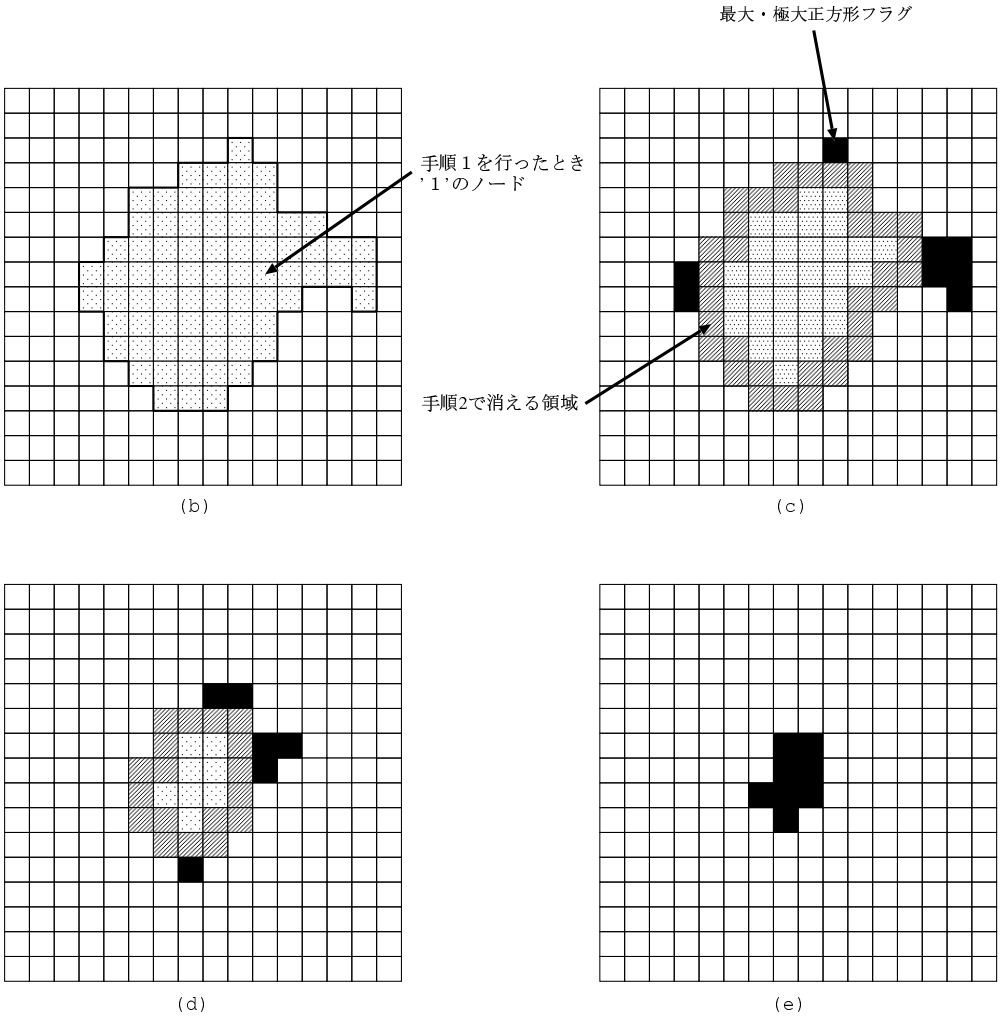


図 3.8: 対角線方向ノード接続による正方形領域検出過程に対する最大・極大正方形の検出結果

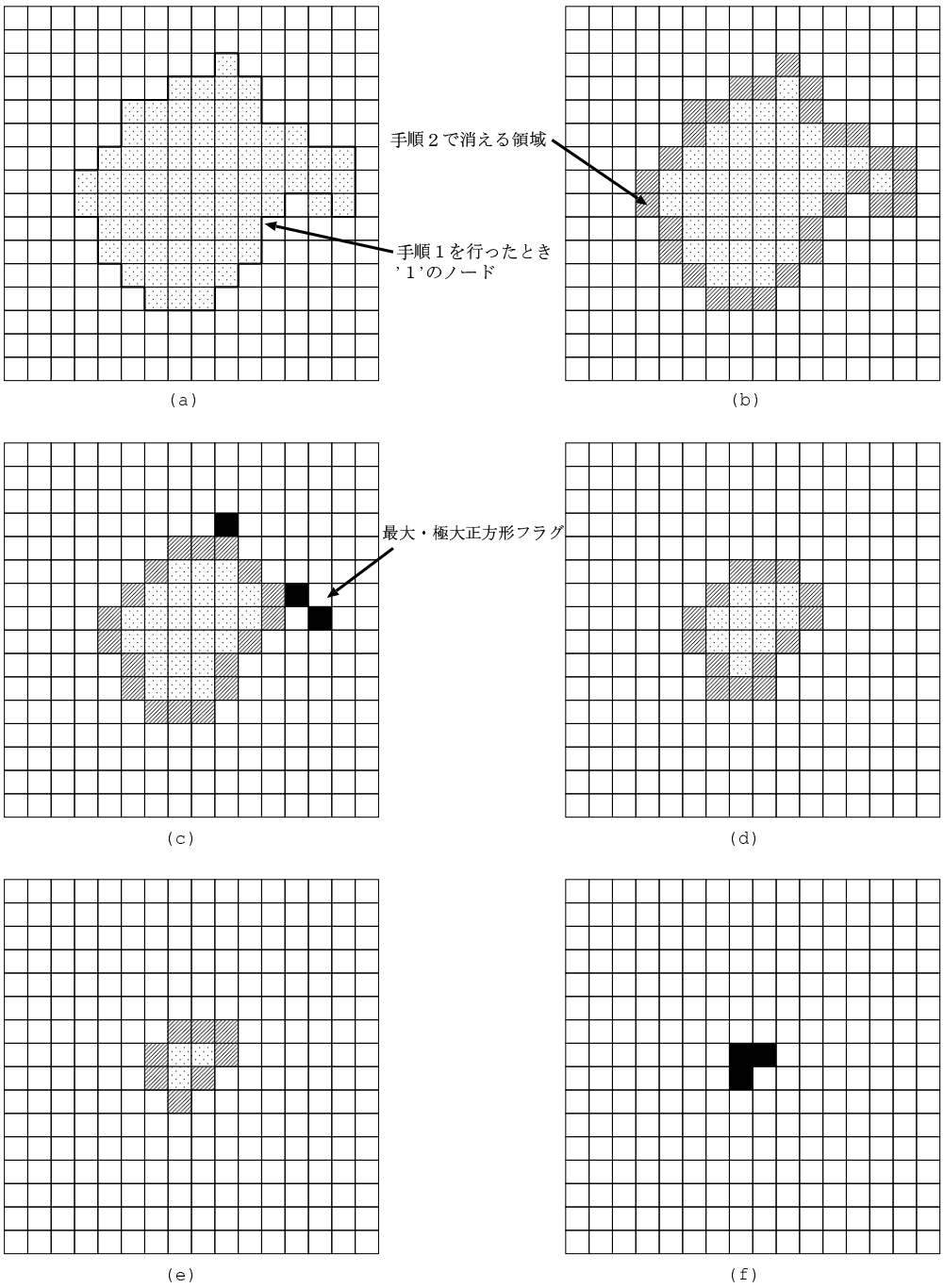


図 3.9: 水平・垂直方向ノード接続による正方形領域検出過程に対する最大・極大正方形の検出結果

3.8(b) の左側のように ‘1’ から ‘0’ へ変化する領域の塊ができる部分が現われてしまい、極大正方形フラグとして立つフラグが多くなってしまう。それに対し図 3.9 では ‘1’ から ‘0’ へ変化する領域は対角線方向へ継っていくため、塊となって現われにくい。そのため図 3.8 と比べてフラグの数が少なく、物体の突起部分と中心部分にフラグが立っている。

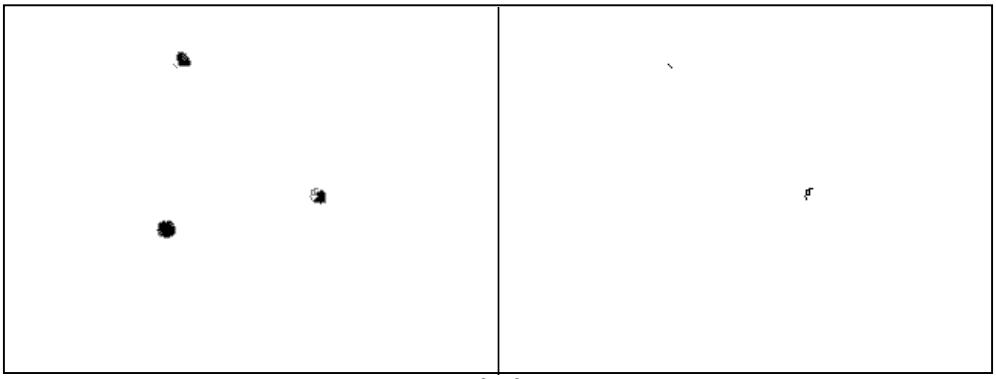
水平・垂直方向ノード接続では、1 回の手順で ‘1’ から ‘0’ へ変化するノードの数が少ない分、検出にかかるステップ数が多くなるが処理ステップ数のワーストケースで考えれば処理回数は変わらない。また最大・極大正方形フラグの数が少なくなるため、情報量が抑えられる利点がある。

以下に水平・垂直方向ノード接続による最大・極大正方形ノード検出のシミュレーションを行ったときのノードの状態変化と検出フラグを図 3.10, 図 3.11 に示す。 図 3.10, 図 3.11 の左側が正方形領域検出過程で黒の 1 ドットはノードの状態が ‘1’ であることを示し、右側が最大・極大正方形の検出結果であり、黒の 1 ドットはノードが最大・極大正方形でありフラグが立っていることを示している。また (a) は手順 1 を行った結果であり、以降 (b) ∼ (e) と手順 2 から 5 を行った結果である。(a) では元画像に斑模様の部分がある右側の物体でフラグが多数立ってしまったが、ある程度の ‘1’ の塊を物体と考え物体の大きさに閾値を設定し、処理の初めのほうに現われる小さな正方形を表すノードやそれによって現われるフラグを無視すれば、処理の完了までに立つフラグは各物体に対して 5,6 個程度であることがわかる。

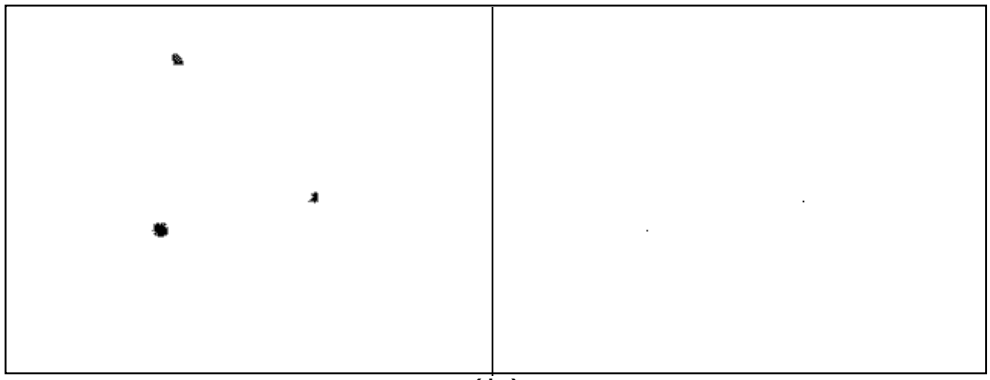
3.2.2 4 進木構造によるフラグの検出

フラグの位置の正確な検出方法として以下に示すツリー構造による探索法が考えられる。[6] 図 3.12 のように画素中にある黒の ‘1’ を探索する場合を考える。まず画素の 4 つの領域に分けたとき、各領域には自分の領域内に ‘1’ があるかどうかを示す機能を持たせる。これにより図 3.12 では右下の領域内に ‘1’ があることがわかる。つぎに右下の領域を 4 つに分け同様なことを行なった結果、分割した領域の右上の領域内に ‘1’ があることがわかり、以下これを画素の大きさになるまで繰り返すことによって、‘1’ がある位置を特定することができる方法である。

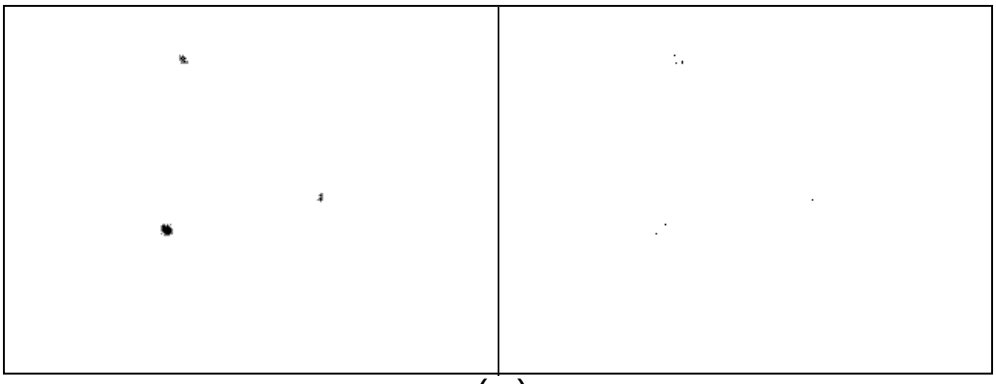
ここで 4 進木構造を用いたフラグの位置検出にかかるステップ数について考え



(a)

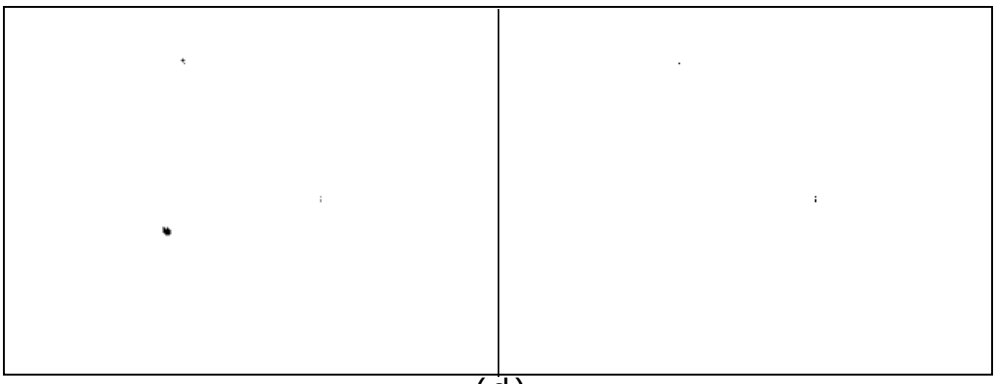


(b)

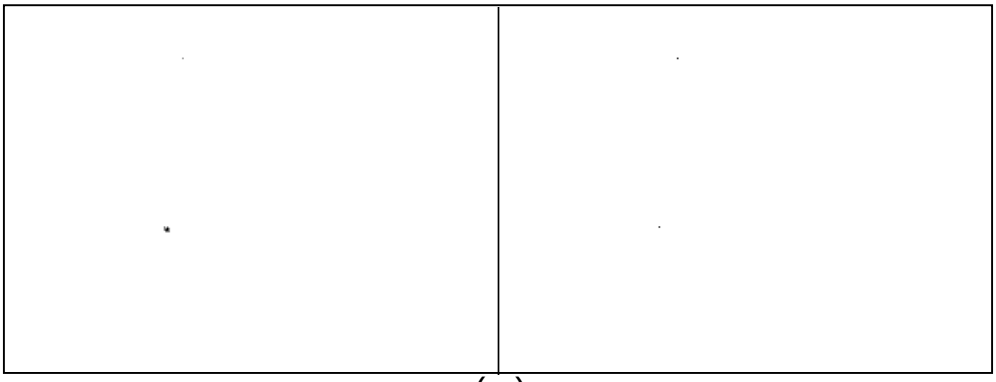


(c)

図 3.10: 正方形領域検出過程と最大・極大正方形の検出結果 (a) ～ (c)



(d)



(e)

図 3.11: 正方形領域検出過程と最大・極大正方形の検出結果 (d), (e)

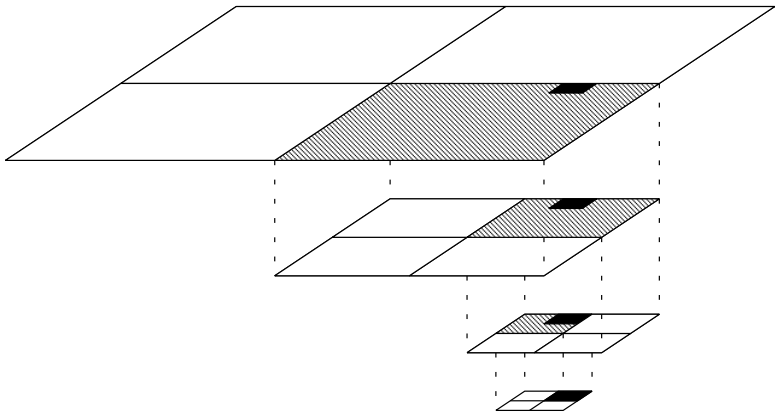


図 3.12: 木構造による最大・極大正方形フラグの検出方法

る。4 進木構造を用いたフラグの位置検出にかかるステップ数は、全フラグ数やフラグの分布状況に応じて大きく変化してしまう。そこで、実画像による正方形検出過程時にフラグ付けを行なうとしたとき、起こり得るケースを考え試算を行なう。

画素数 n^2 で、処理過程の 1 フレームに注目したとき全ノード中のフラグの占める割合を p_0 とすると、処理ステップ数の期待値 $\overline{L}$ は次式のようになる。[6]

$$\overline{L} = 1 + \sum_{l=1}^{\lfloor \log_4 n^2 \rfloor} 4^{\log_4 n^2 - l + 1} \{1 - (1 - p_0)^{4^l}\} \tag{3.2}$$

1 枚の画像中にあるフラグ '1' の数は、物体の個数とその物体の極大正方形として検出される部分の合計であるため、物体の個数の数倍程度であると考えられる。また式 (3.2) より処理過程の各フレーム中にフラグが分散し 1 フレーム中のフラグ数が少ない方が処理ステップ数が多くなる。

そこで画素数 1000×1000, 物体数 100 個, 物体 1 つに付く最大・極大フラグ数 5 個で、フラグは処理過程の各フレームに 1 つのみと仮定すると式 (3.2) よりフラグ 1 つの検出ステップ数の期待値は 38.825 となるため全フラグ 500 個の検出にかかるステップ数の期待値は約 20000 ステップとなる。また 1 フレームに全てのフラグ 500 個があるとすると検出にかかるステップの期待値は約 11000 ステップとな

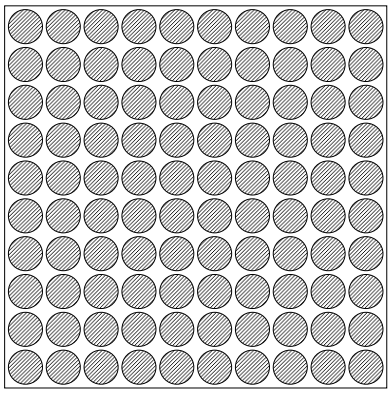


図 3.13: 画素数 1000 × 1000 において物体が 100 個ある状態の一例

る。しかし画素 1000 × 1000 中に 100 個の物体があるということは、平均 100 × 100 の大きさの物体であり図 3.13 のような状態であると考えられる。つまり正方形領域検出過程は平均 50 ステップで完了すると考えられるため、一度に検出される平均フラグ数は 10 個と考えることができ、式 (3.2) を用いて計算すると期待値は約 16000 ステップとなる。

図 3.14 は、このような条件下でフラグ数を横軸としてフラグの検出にかかるステップ回数を計算ものであり、図の worst case は 1 フレーム中に 1 フラグとした結果で、best case は全てのフラグが 1 フレームに集中して現われた場合である。上に述べたように、実際の画像では worst case や best case は稀で、2 本のグラフの間のどこかで実際の処理ステップ数が決まると考えら、図 3.13 のように多くの物体が存在するケースは少なく、これまでに用いてきた図 2.8 のような種類の画像ではフラグ数が数十個程度となり、検出ステップの期待値は 2000 ステップ以下となる。

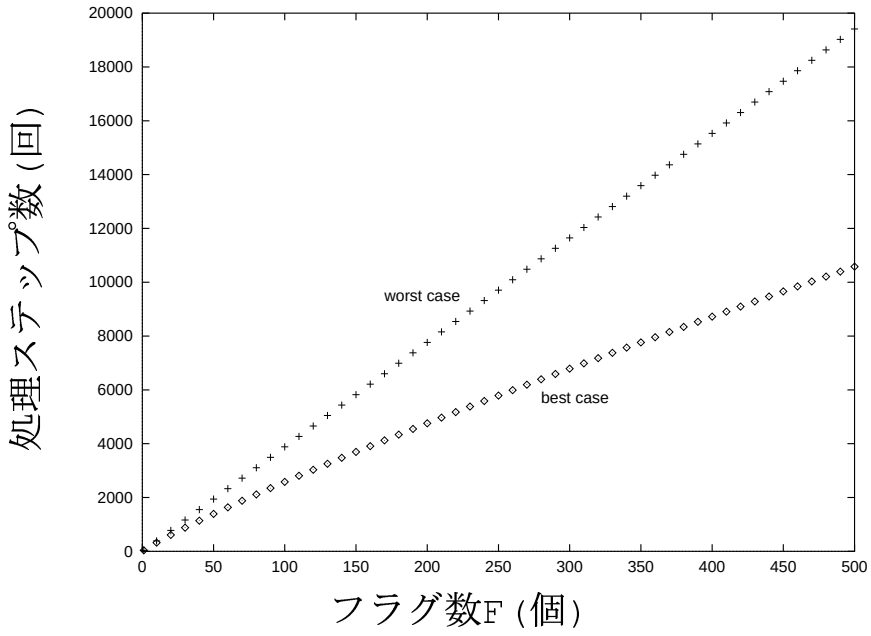


図 3.14: 4 進木構造による最大・極大正方形フラグの検出にかかるステップ数 (画素数 1000×1000 の場合)

第4章　平面配置オートマトンによる物体検出回路の構成

本アルゴリズムを受光素子である画素と状態遷移回路であるノードによって集積回路上で実現する方法を検討する。また実際に設計した回路やその測定結果についても述べる。

4.1　各機能ブロックの回路構成

4.1.1　ノードオートマトン

ノードオートマトンの回路は、次のような機能を有する。

- 4近傍の画素とノードからの2系統の4本のデータ入力線とそのセレクト信号、クロック信号、ノードの値の出力を持つ。
- セレクト信号で選択された4本のデータ入力線の論理積をノードの値とし、クロックに同期して保持し出力する。

これは図4.1に示す回路で構成でき、6個の論理セルから構成されるため、受光素子と同一チップ上に集積する場合でも高い開口率を確保できると考えられる。またゲートの段数が少ないため、高速動作が期待できる。

4.1.2　正方形検出回路の構成

正方形領域検出アルゴリズムは画素の手順2以降のノードの接続と処理が同じであることに注目すると、ノードの状態遷移はクロック信号に同期した時間遷移と

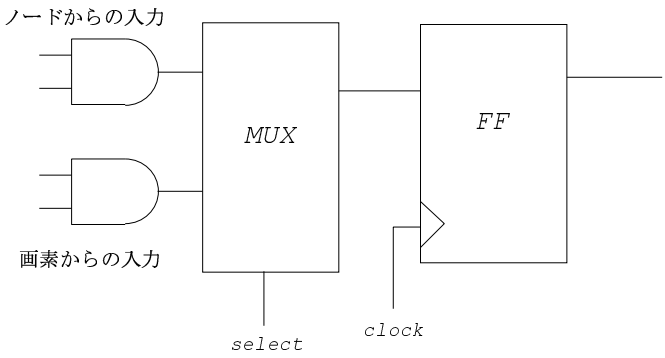


図 4.1: ノードオートマトンの回路構成

考えることができ、マトリクス状に配置した画素とその間を埋める平面配置ノードオートマトンで実現することができる。

まず 1 次元配列によるアルゴリズムの回路構成を考え、次に 2 次元へ拡張する。1 次元配列によるアルゴリズムの回路構成は図 4.2 のようになり、以下に示す動作を行なうことにより実現できる。

1. まず手順 1 の処理である画素からの入力の論理積の結果がノードの値となるようセレクト線で選択し、クロックに同期して値を保持する
2. 次に隣接ノードで保持されている値の論理積の結果がノードの値となるようセレクト線で選択し、クロックに同期して値を保持する
3. 全てのノードが '0' となるまで 2. を繰り返す

この構造は 1 次元におけるものであるが、ノードの回路を 2 入力 AND から 4 入力 AND へ変更し、図 4.3 や図 4.4 のように 2 次元配置のノード接続を行なえば本アルゴリズムの平面配置オートマトンによる正方形領域検出回路が実現できる。

4.1.3 最大・極大正方形を表すノードの検出回路構成

最大・極大正方形領域検出回路は式 (3.1) より、一つ前のノードの状態と現在のノードの状態の反転 ('1' のノードが「消える」ための条件)、廻りのノードの状態

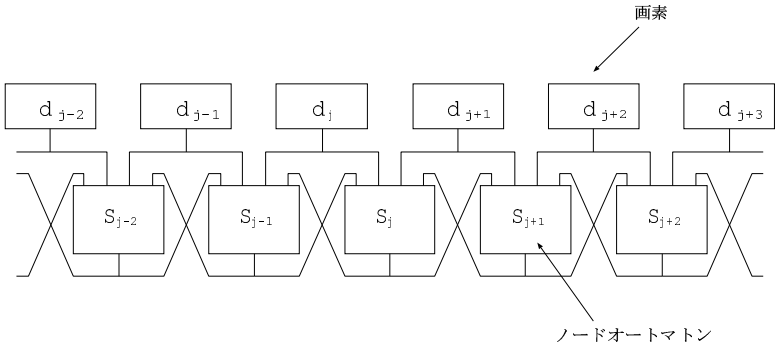


図 4.2: 1 次元における領域検出法の回路構成

の反転との論理積 (廻りのノードがすべて '0' である条件) をとる図 4.5 のような構成となる。

この最大・極大正方形検出回路とノードの回路を組み合わせると図 4.6 のようになるのだが、フリップフロップが 2 段連続して存在しその間に論理が含まれていないため、前段の状態が 1 クロック遅れて次段へ伝搬する回路となっている。つまり、最大・極大正方形検出回路とノードの回路を組み合わせて用いる場合、フリップフロップは 1 個省略でき、図 4.7 のような回路構成となる。このとき、最大・極大正方形検出方法の式 (3.1) は現在のノードの状態と次のノードの状態を用いた式 (4.1) となる。

$$
\begin{aligned}
F_{i,v,h} \quad = \quad & \left(S_{i,v,h} \cdot \overline{S_{i+1,v,h}}\right) \\
& \cdot \left(\overline{S_{i+1,v-1,h-1}} \cdot \overline{S_{i+1,v,h-1}} \cdot \overline{S_{i+1,v+1,h-1}}\right. \\
& \cdot \overline{S_{i+1,v-1,h}} \cdot \overline{S_{i+1,v+1,h}} \\
& \left. \cdot \overline{S_{i+1,v-1,h+1}} \cdot \overline{S_{i+1,v,h+1}} \cdot \overline{S_{i+1,v+1,h+1}}\right)
\end{aligned} \tag{4.1}
$$

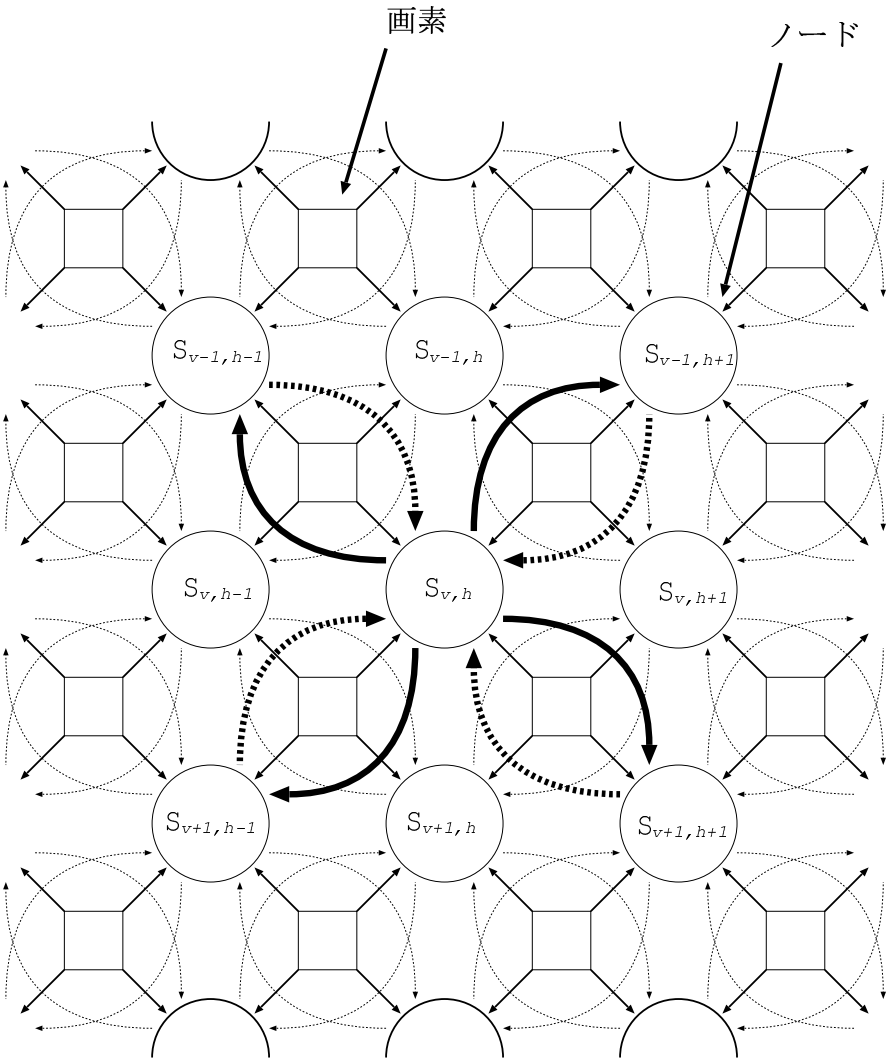


図 4.3: 対角線方向結線の平面配置オートマトンによる正方形領域検出法の回路構成

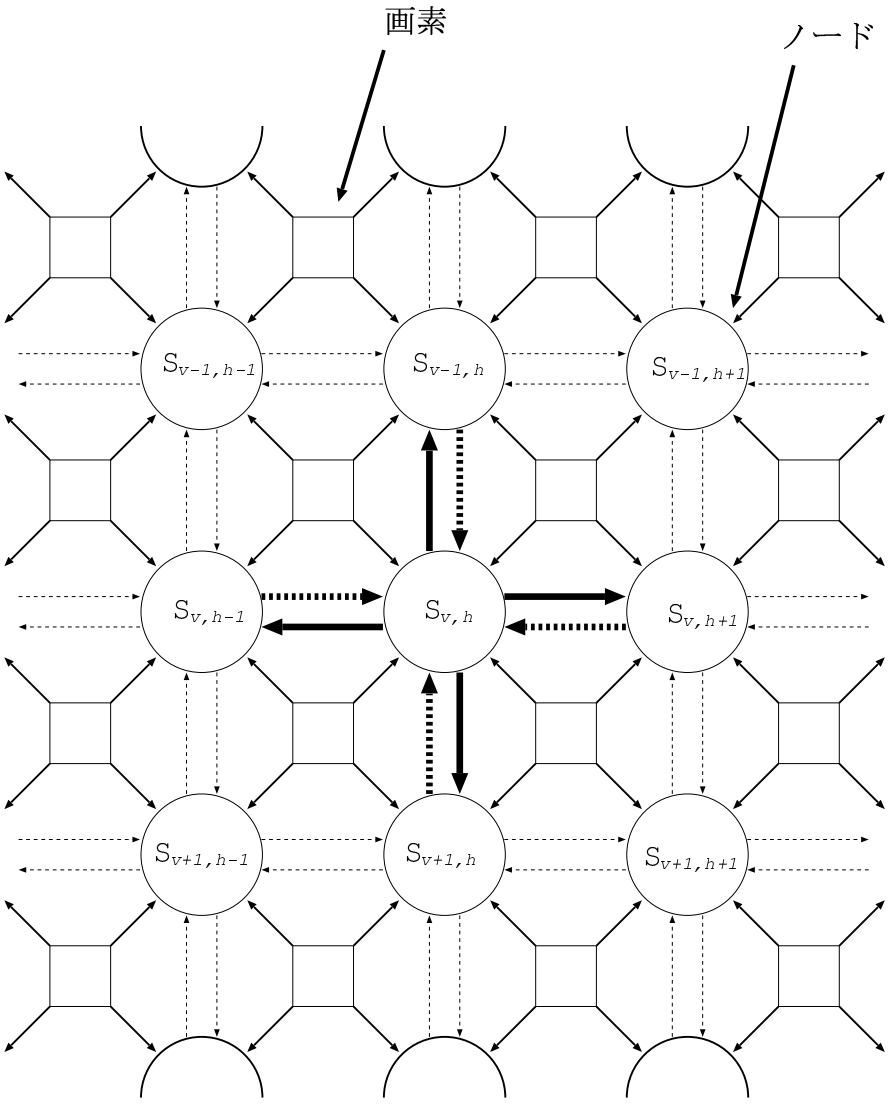


図 4.4: 水平・垂直方向結線の平面配置オートマトンによる正方形領域検出法の回路構成

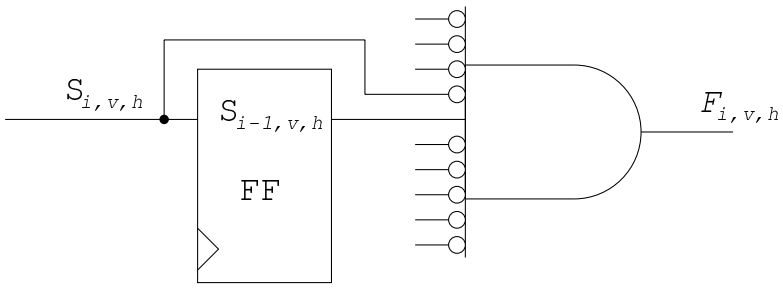


図 4.5: 最大・極大正方形検出回路

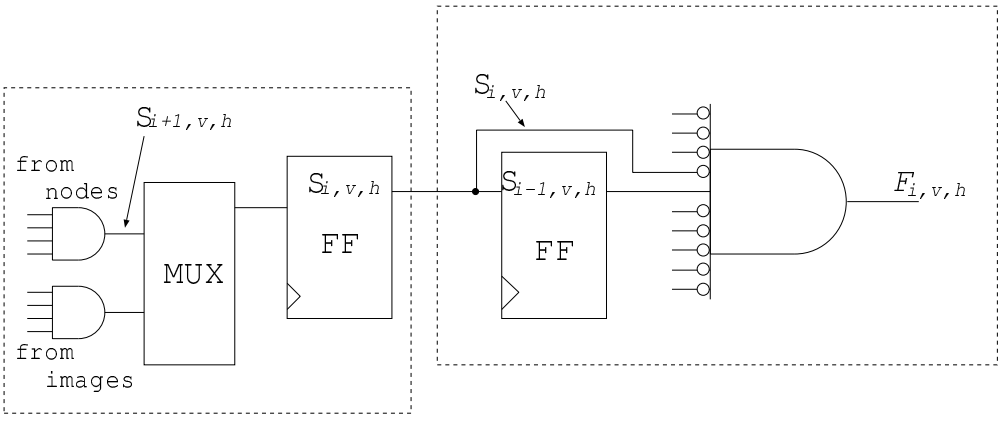


図 4.6: ノード回路と 最大・極大正方形検出回路

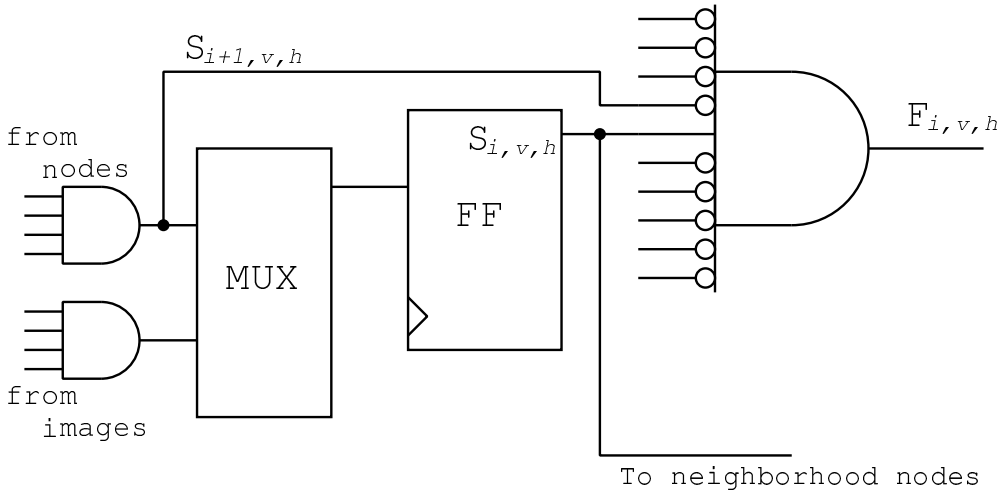


図 4.7: 最大・極大正方形検出機能付きノードの回路構成

4.2 矩形領域検出チップの試作

以上のような回路のうち、Chapter 4.1.2 の対角線方向ノード接続による正方形検出回路とノードの論理和出力部、画素を模擬するフリップフロップを同一チップ上に集積した検証回路をモトローラ CMOS 1.2μm ルールのプロセスを用いて設計し、試作を行った[1] 。そのチップ写真を図 4.8 に示す。ダイサイズ 2.3mm 角に 5×9 個の画素と 4×8 個のノードオートマトンをマトリクス状に配置した。画素を模擬するフリップフロップは、外部からデータをシリアル転送することで画像をつくり出し受光素子の機能を模擬している。

5×9 のすべての画素を '1' とした場合、図 4.9 のようにステップが進むにつれノードの値の '1' の領域が端から減っていき、クロック数 3 で水平方向の出力がすべて '0' となっていることから、大きさ 4 の物体を検出できていることがわかる。この過程を回路シミュレータ `spice` を用いて回路の動作確認を行った。図 4.10 はその結果であり、正しく動作していることがわかる。

[1] 本チップ試作は東京大学大規模集積システム設計教育研究センターを通し 日本モトローラ (株)、大日本印刷 (株)、および京セラ (株) の協力で行われたものである。

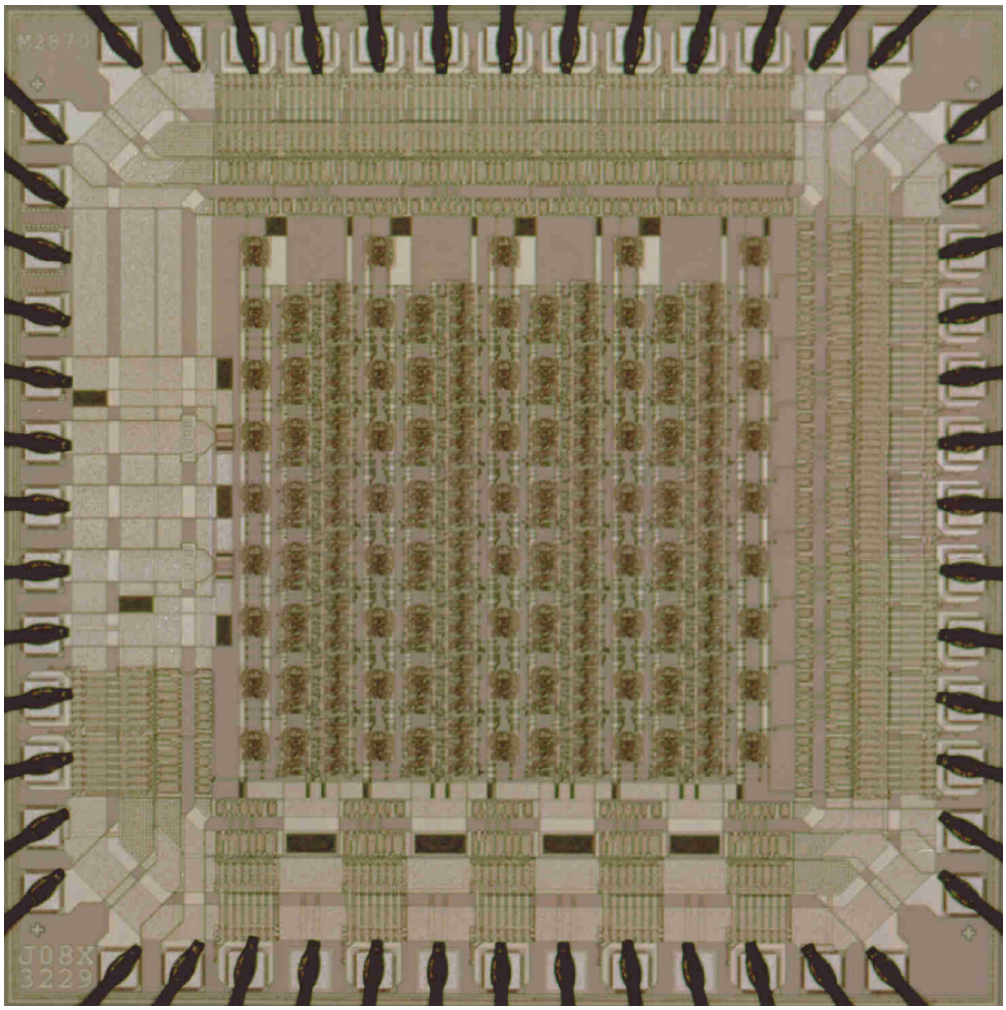


図 4.8: 正方形領域検出アルゴリズム検証用チップの写真 (CMOS1.2μm ルール、チップサイズ 2.3mm 角)

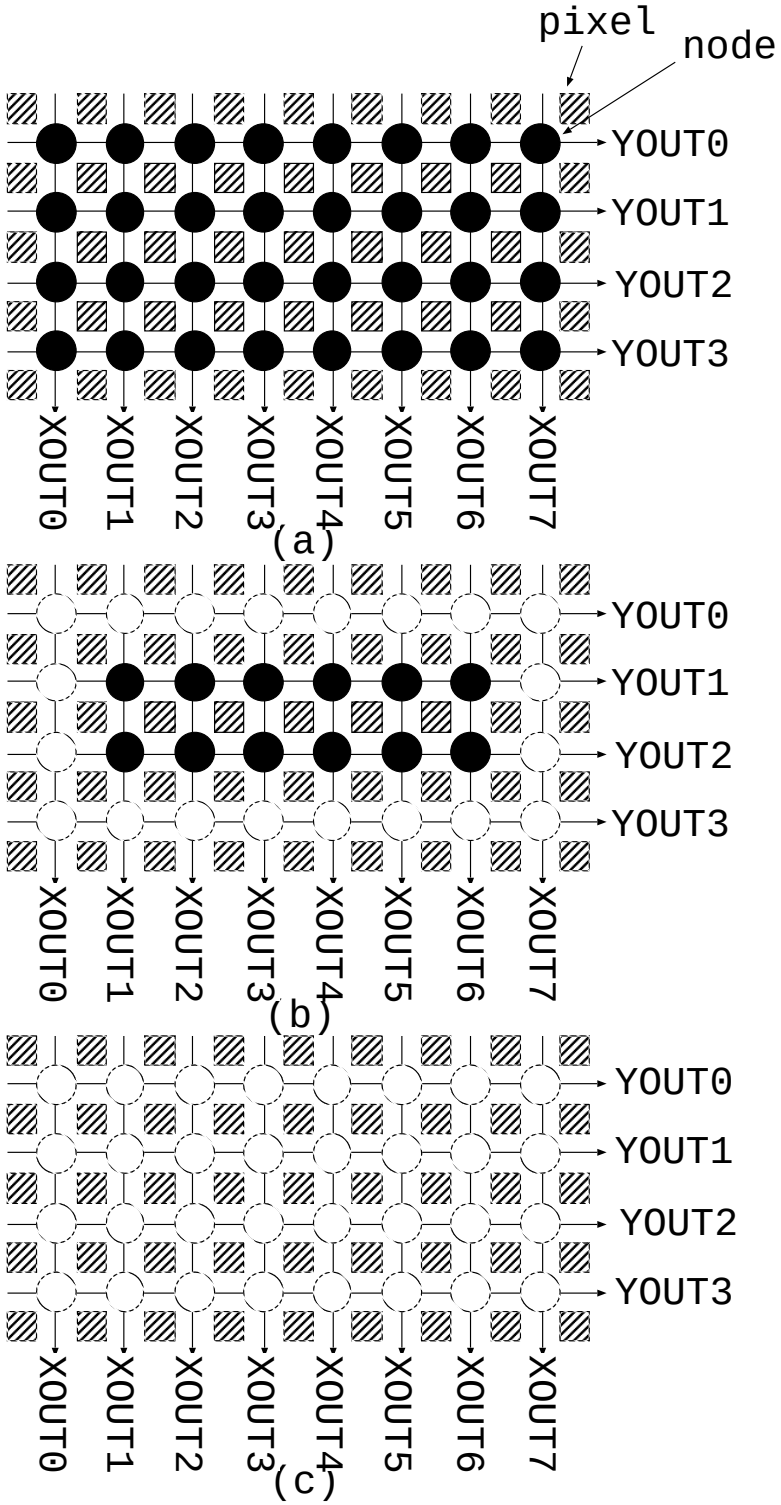


図 4.9: 画素 5×9 をすべて値 '1' にした場合の処理過程 (a) ステップ 1, (b) ステップ 2, (c) ステップ 3

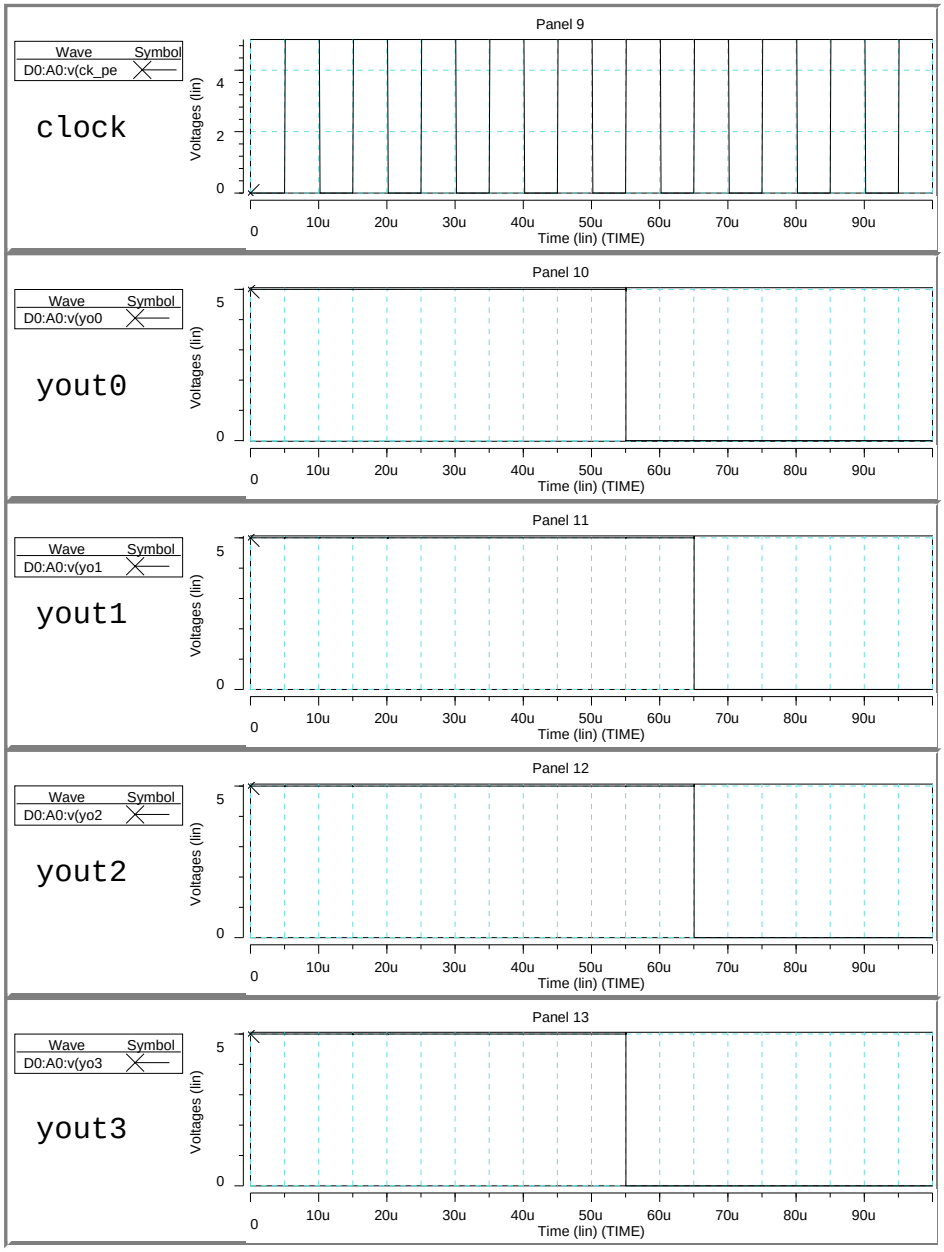


図 4.10: 設計した回路の spice によるシミュレーション結果

この検証回路の測定を行なった結果、最小クロック周期 7.8ns で動作確認をすることができた。本手法で最も多い処理クロック数となるケースは、すべての画素が '1' である状態のとき最も多いクロック数となる。例えば 1000×1000 の画素の場合を考えると、最大 500 クロックの処理ステップを必要とし、一列に並ぶノード数が約 100 倍になるためクロック周期も約 100 倍長くなる。よって、クロック周期を $1\mu s$(1MHz) で動作させると、ワーストケースの場合 $10(ns) \times 100$(倍) $\times 500($クロック$) = 500\mu s$ で処理が終わることになり、1 秒間に 2000 枚のフレームが処理可能であることを示している。

第5章　まとめ

本研究では、画素とノードを平面マトリクス状に配置した構造を用い、隣接 4 近傍画素または隣接 4 近傍ノードの対角線方向ノード接続もしくは水平・垂直方向ノード接続によって論理積をとることにより、大きさ $2n$ の正方形領域の中心をノードの n 回の状態遷移で表現する手法を提案した。この手法はノードが並列に動作することによりシーケンシャルな処理と比べて少ないクロック数で正方形領域の検出が可能であり、構造がシンプルなため高速動作が可能なであることを示した。

物体のおよその位置と大きさの検出法として、物体に内包される最大・極大正方形が物体を代表する領域であると考え、ノードの水平・垂直方向論理和出力を用いる方法と最大正方形ノードにフラグを立て 1:4 ツリー構造を用いた探索法によりフラグの位置を特定する方法を提案した。前者は、正方形領域検出過程において水平・垂直方向から物体に光をあて、その投影像から物体の位置と大きさを検出することと等価であり、位置を特定できない場合や「影」が現われる問題があったが、検出にかかる処理回数は物体の大きさの半分の回数で処理が完了できる利点がある。

後者を用いる場合、隣接ノードの状態からノードが最大・極大正方形を表しているかを判断し、最大・極大正方形の条件に当てはまるノードにフラグを立て、そのフラグを 1:4 ツリー構造による検出法を用いて位置を特定することにより最大・極大正方形の位置を正確に特定した。また最大・極大正方形フラグを検出する過程では、極大正方形フラグの数を抑えるために平面配置ノードの接続を対角線方向に接続する場合に比べて水平・垂直方向接続を行なった方が有利であることを示した。

後者の処理回数は、画素数が 1000×1000 であるとき物体数が十数個程度であればフラグ数が 100 個程度となり最悪値でも 5000 回程度の処理回数で処理が完了

し、十分にリアルタイム処理が可能なアルゴリズムであることを示した。

さらに、本論で示した手法をスマート・センサとして実現する回路構成を行った。ノードオートマトンを状態遷移回路で構成し、ノードの水平・垂直方向論理和出力による物体検出回路をノードが 8×4 個、画素が 9×5 個を集積したチップを Motorola 1.2μm ルールのプロセスにより設計した。回路を測定したところ、最小クロック周期 7.8ns で動作確認をすることができた。例えば 1000×1000 の画素の場合を考えると、最大で 500 クロックの処理ステップを必要とし、一列に並ぶノード数が約 100 倍になるためクロック周期も約 100 倍長くなる。よって、ワーストケースの場合 $10(ns) \times 100(\text{倍}) \times 500(\text{クロック}) = 500\mu s$ で処理が終わると予想でき、1 秒間に 2000 枚のフレームが処理可能であることを示している。

以上で得られた結論から、従来の画像処理システムと比べて高速な物体の位置と大きさの検出を行なえる可能性を示すことができた。

付 録A　論理和出力の頂点検出のための回路構成

論理和出力の状態変化は図 A.1(a) のような例が考えられる。ここでステップ n における右から j 番目の垂直方向論理和出力の状態 $V_{j,n}$ に注目して考える。

- $V_{j,n}$ が'0' であるとき、前の状態 $V_{j,n-1}$ は'1' である (論理和出力が「消える」)
- $V_{j,n}$ が'0' であるとき、隣接近傍の出力も'0' である

以上のような条件が満たされるとき、その出力は頂点を表していると言える。

これは凸型の上部が'1' で下部が'0' となっているパターンを考え、A.1(b) のように論理和出力の「山」の縁にそって凸型パターンと論理和出力を照合していくとき、頂点でなければ凸型の左もしくは右の部分が「山」と重なり不一致となる。また頂点であれば隣接近傍に'1' が存在しないため、パターンと一致し頂点であることが検出できる。

つまり隣接近傍を左右 1 ドットとすると、

$$F_{j,n} \quad = \quad V_{j,n-1} \cdot \overline{V_{j,n}} \cdot \overline{V_{j+1,n}} \cdot \overline{V_{j-1,n}} \tag{A.1}$$

となる。$F_{j,n}$ は頂点を表すフラグである。

図 A.1(c) は式 (A.1) による頂点検出結果であるが、左右 1 ドットのみを調べる場合、頂点以外でも一致する領域が現われてくる。

隣接近傍として左右 1 ドットのみを調べる場合、図 A.1(c) のように頂点以外でフラグがたつ場合がでてしまうが、あまり離れた出力まで調べた場合、隣の「山」を含めて調べる場合が考えられ、その場合にはフラグがたたなくなるという問題がある。

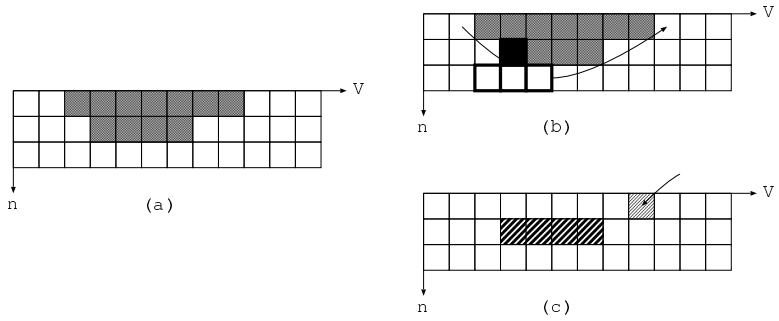


図 A.1: 論理和出力のよる頂点検出の (a) ステップ n における検出前の元状態, (b) 頂点検出の概念図, (c) 頂点検出結果

付 録B　物体検出回路のチップデータ

ピン情報							
1	VDD	21	VDD	41	VDD	61	VDD
2	NC	22	NC	42	NC	62	NC
3	GND	23	GND	43	GND	63	GND
4	NC	24	NC	44	NC	64	NC
5	NC	25	NC	45	NC	65	NC
6	NC	26	sen_out4	46	select	66	sen_in0
7	NC	27	NC	47	Xout0	67	Yout0
8	NC	28	sen_out3	48	Xout1	68	sen_in1
9	NC	29	NC	49	Xout2	69	Yout1
10	NC	30	sen_out2	50	Xout3	70	sen_in2
11	NC	31	NC	51	Xout4	71	Yout2
12	NC	32	sen_out1	52	Xout5	72	sen_in3
13	CK_sen	33	NC	53	Xout6	73	Yout3
14	CK_node	34	sen_out0	54	Xout7	74	sen_in4
15	NC	35	NC	55	NC	75	NC
16	NC	36	NC	56	NC	76	NC
17	GND0	37	GND0	57	GND0	77	GND0
18	NC	38	NC	58	NC	78	NC
19	VDD0	39	VDD0	59	VDD0	79	VDD0
20	NC	40	NC	60	NC	80	NC

ピン情報			
VDD	内部回路用電源	CK_sen	センサ用クロック
GND	内部回路用グランド	CK_node	ノード用クロック
VDD0	IO バッファ用電源	select	ノード入力セレクト
GND0	IO バッファ用グランド	sen_in	画像データ入力バス
Xout	X 方向ノード論理和出力バス	sen_out	画像データ出力バス
Yout	Y 方向ノード論理和出力バス	NC	未接続

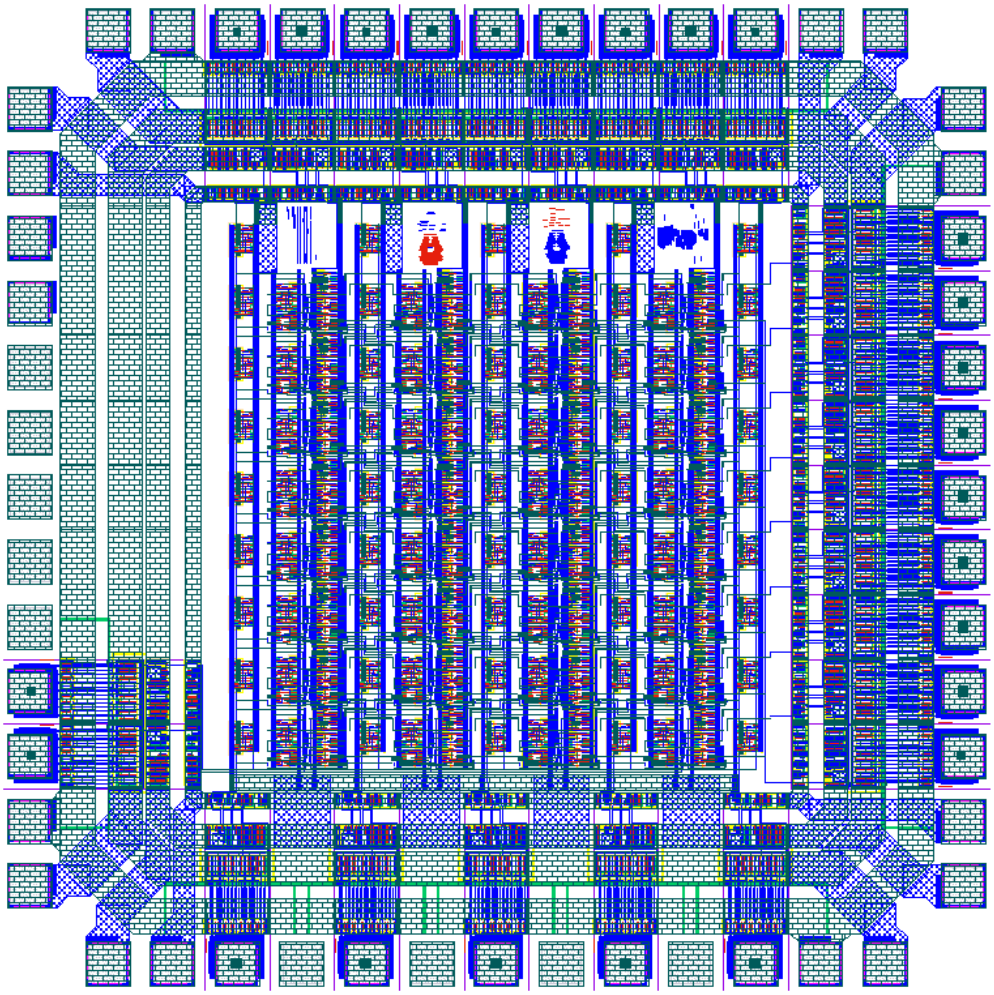


図 B.1: 設計した回路のレイアウト (CMOS1.2μm ルール、チップサイズ 2.3mm 角)

図 B.2: 物体検出チップの外形写真 (CMOS1.2μm ルール, チップサイズ 2.3mm 角, QFP 80 ピン)

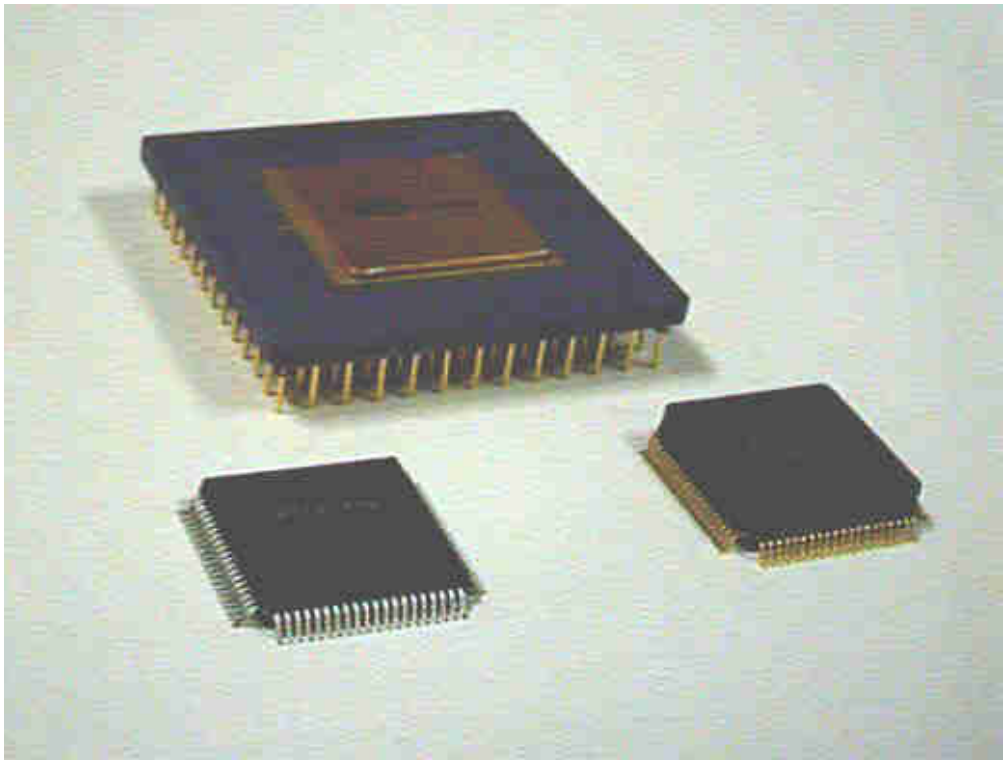


図 B.3: 著者が設計した歴代のチップたち, 手前左 : 加算器と DA コンバータ (Motorola, 1997 年 8 月), 手前右 : 物体検出回路 (Motorola, 1999 年 8 月), 奥 : 8bit CPU(NEL, 1998 年 6 月)

関連図書

[1] A. Gruss *et al.*, "A VLSI Smart Sensor for Fast Range Imaging," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems,* 1992.

[2] J. E. Eklund *et al.*, "VLSI Implementation of a Focal Plane Image Processor - A Realization of the Near-Sensor Image Processing Concept," *IEEE Trans. VLSI SYSTEMS.* Vol.4, No.3, pp. 322-335, Sep. 1996.

[3] G. A. Horridge and P. Sobey, "An artificial seeing system copying insect vision," *INTERNATIONAL JOURNAL OF OPTOELECTRONICS.* Vol.6, NOS.1/2 177-193.

[4] A. Åström *et al.*, "Global Feature Extraction Operations for Near-Sensor Image Processing," *IEEE Trans. Image Processing.* Vol.5, No.1, Jan. 1996.

[5] 平井有三, "視覚と記憶の情報処理," 培風館, 1995.

[6] J. Akita and K. Asada, "An Image Scanning Method with Selective Activation of Tree Structure," *IEICE Trans. on Electronics.* Vol.E80-C, No.7, 1997.

口頭発表

- 前多和洋, 秋田純一, 北川章夫, 鈴木正國, “平面配置オートマトンによる矩形領域検出回路,” 電子情報通信学会技術研究報告, Vol.98, No.516, CAS98-79 pp.31-38, 1999 年 1 月, 鳥取大学
- 前多和洋, 北川章夫, 鈴木正國, “HDL を用いた CPU 設計とチップ試作,” 電気関係学会北陸支部連合大会, p.144, 1997 年 11 月, 金沢工業大学

謝辞

本研究を行なうにあたり多くの方々に御助言、御協力を頂きました。この場を借りて感謝の意を表したいと思います。

様々な面で御助言、御指導をして頂きました鈴木正國教授に心から感謝致します。また研究、生活面においてお世話になりました北川章夫助教授に深く感謝致します。研究の御指導、御助言、生活面でもお世話になりました秋田純一助手に深く感謝致します。機器の発注、工作や生活面で御指導頂きました柿本芳雄技官に深く感謝致します。

VLSI 設計システムの初期設定、UNIX の操作指導をして頂いた集積回路工学研究室卒の中山和也助手、池田真俊氏に感謝致します。3 年間の研究生活を有意義なものにし、良きアドバイスや相談にのって頂いた夏目雅弘氏、田口和彦氏、水橋蜂嘉章氏に感謝致します。また、生活面でも研究面でも良きアドバイスをして頂いた高瀬信二氏、早川史人氏、小川明宏氏、中橋憲彦氏に感謝致します。また同じ研究室生として研究を行ってきた卒研生の今井豊氏、数馬晋悟氏、佐藤堅氏、房川実氏、藤田隼人氏、水野浩樹氏、村上崇氏、渡辺晃氏に感謝致します。

大学での LSI 設計を可能にしてくれた大規模集積システム設計教育研究センターに感謝致します。最期に、大学生活をとても有意義な時間とし、音を奏でる楽しさを教えてくれた金沢大学吹奏楽団とその団員の皆様に心から感謝致します。