画像処理工学

２値画像に対する処理

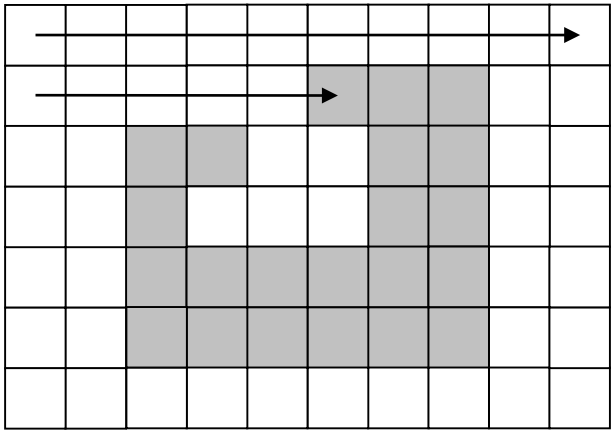
ラベリング

- ラベリング
  - 2値画像上に点在している図形成分（連結成分）のそれぞれに名前をつける処理
  - 図形成分の区別をしておけば，図形成分の個数やそれぞれの特徴（面積など）を計算できる

			A								B	B		
	A	A	A	A				B	B	B	B	B	B	
	A	A	A			B	B	B	B		B	B	B	
		A			B	B					B	B		
				B	B			C	C		B	B		
			B	B			C	C			B	B		
		B	B							B	B			D
	B	B	B	B	B	B	B	B	B	B			D	D
		B	B		B	B	B					D	D	D
												D	D	

ラベリング

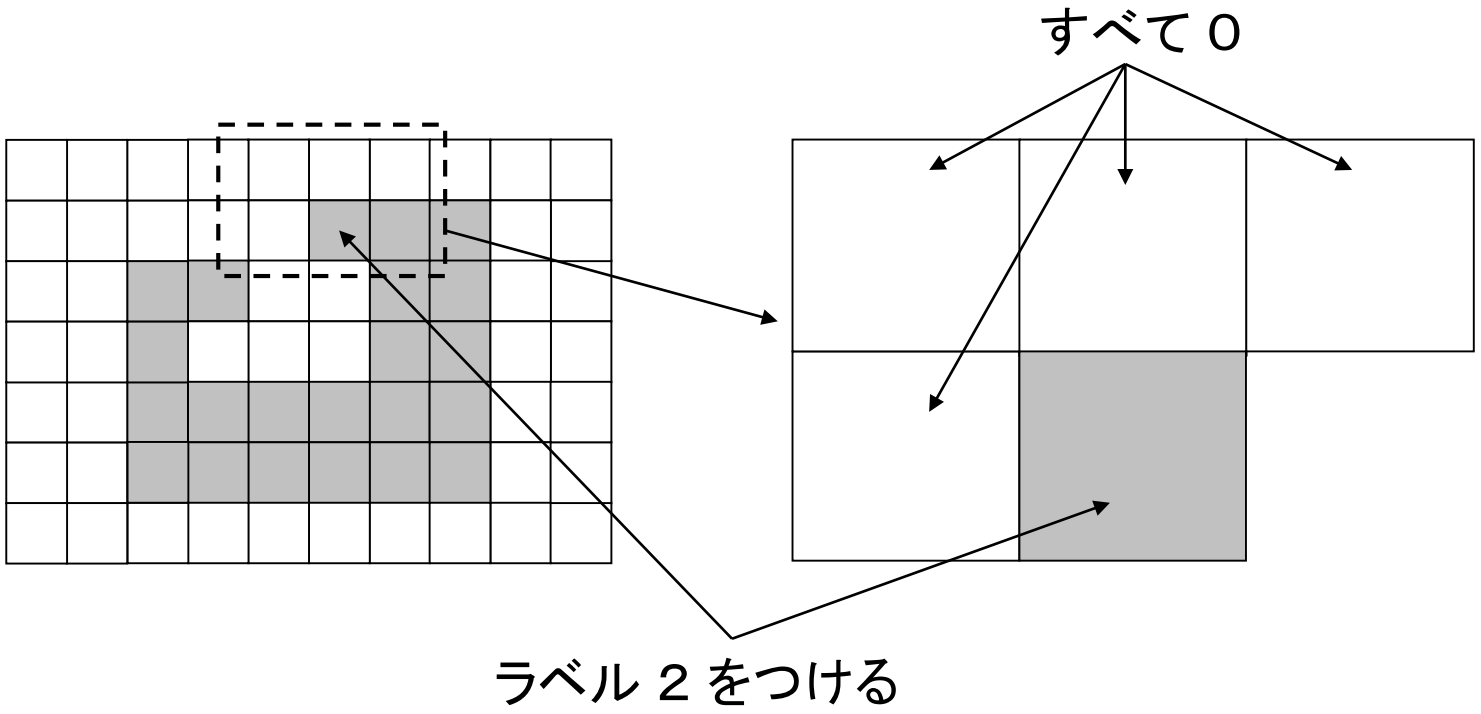
- ラベリングのアルゴリズムの例
  - ８連結で考え，２回の走査を行う
  - 左上の画素から走査しながら，１の画素を見つける
  - その画素 $f[i][j]$ に隣接する画素のうち
    $f[i-1][j-1]$ ， $f[i-1][j]$ ，$f[i-1][j+1]$，$f[i][j-1]$
    の値とラベルを調べる



$f[i-1][j-1]$	$f[i-1][j]$	$f[i-1][j+1]$
$f[i][j-1]$	$f[i][j]$	

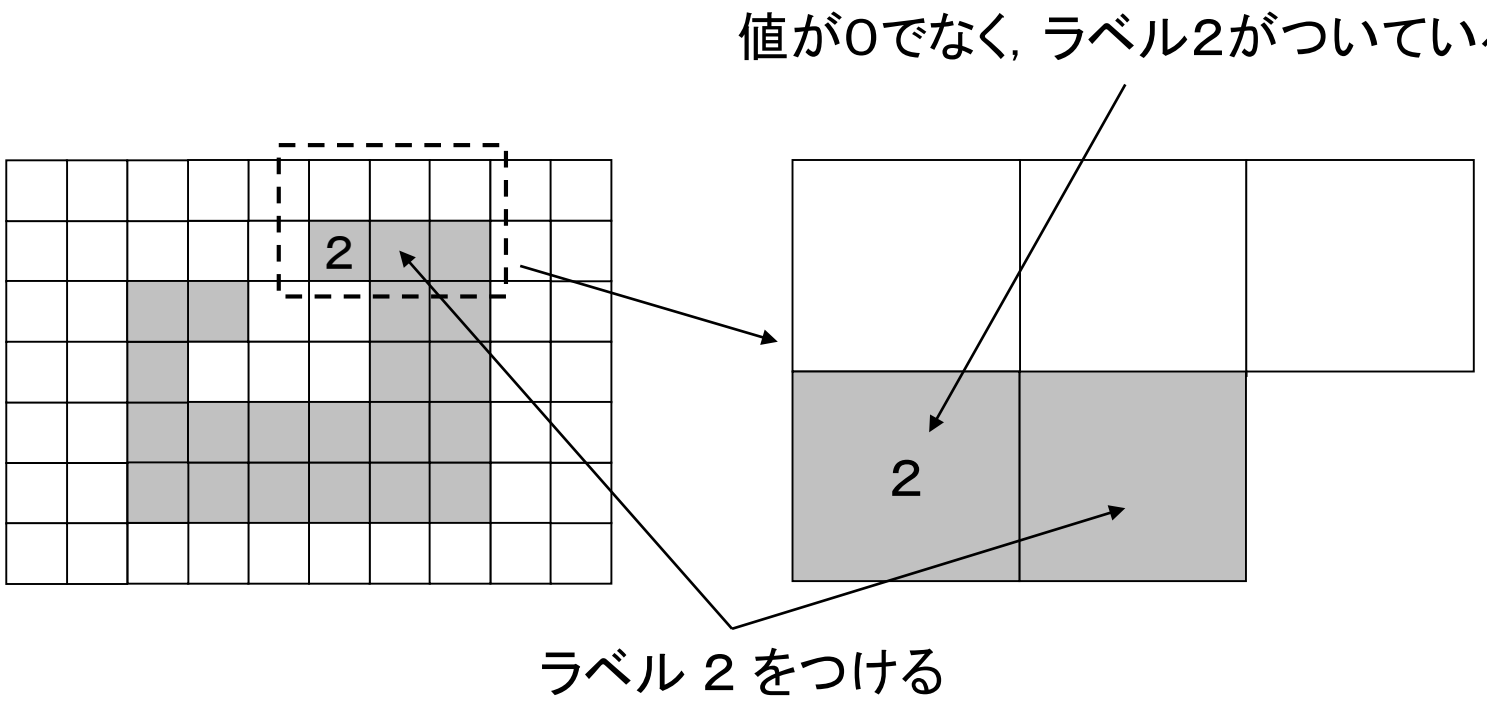
ラベリング

- ラベリングのアルゴリズムの例（続き）
  - ４つの画素すべてが0のとき，$f[i][j]$に新しいラベルをつけて，次の画素を走査する



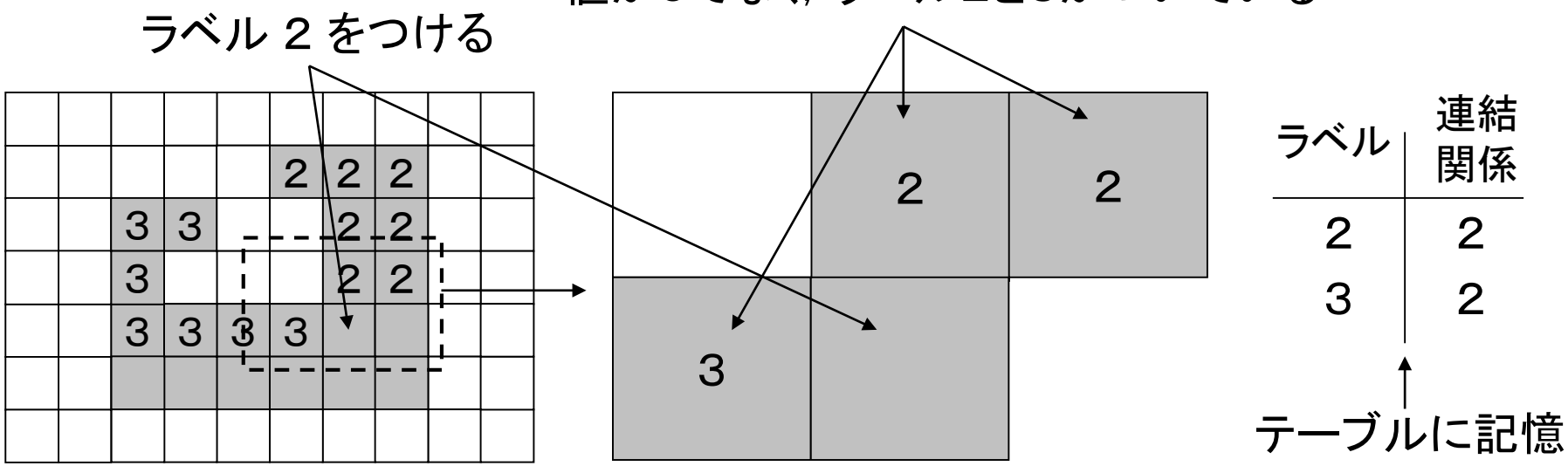
ラベリング

- ラベリングのアルゴリズムの例（続き）
  - ４つの画素のうち，値が0のもの以外に1種類のラベルがついている場合は，$f[i][j]$に同じラベルをつけて，次の画素を走査する



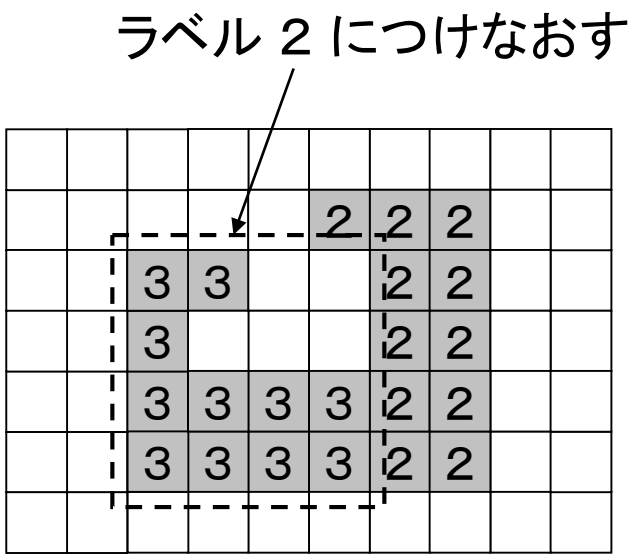
ラベリング

- ラベリングのアルゴリズムの例（続き）
  - ４つの画素のうち，値が0のもの以外に2種類以上のラベルがついている場合は，その中で最も小さいラベル番号を$f[i][j]$につけるとともに，それらが同じ連結成分であることを記憶しておき，次の画素を走査する



ラベリング

- ラベリングのアルゴリズムの例（続き）
  - 最後まで走査が終わったら，左上から2回目の走査を行い，ひとつの連結成分に対して同じラベルがつくように，ラベルをつけなおす



ラベル	連結関係
2	2
3	2

テーブルを見るとラベル 3 は ラベル 2 と同じ連結成分である

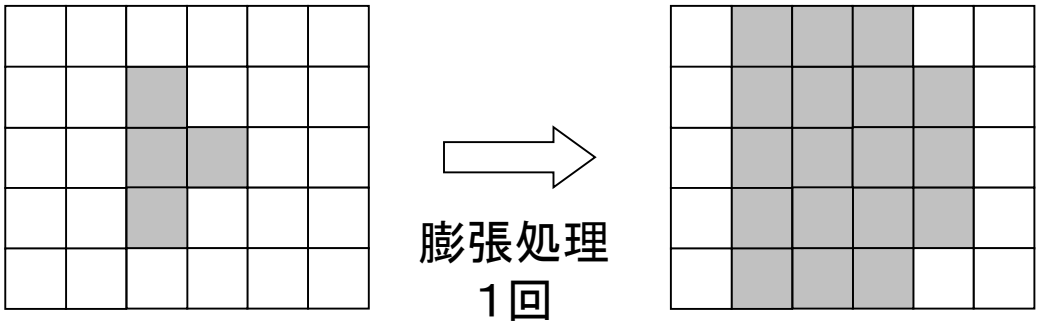
膨張・収縮処理

- 膨張
  - 図形を外側に1画素分広げる処理
  - 拡張，伝播とも呼ばれる
- 収縮
  - 図形を1画素分細くする処理
  - 侵食とも呼ばれる
  - 2値画像において，ノイズを除去する目的などに用いられる
  - 処理を何回行うかで結果が変わってくる
    （埋めてはいけない孔を埋めてしまうことがある）

膨張・収縮処理

- 膨張処理
  - ある画素とその8近傍（または4近傍）のいずれかに少なくともひとつの図形画素（1）がある場合，出力画素を1とする

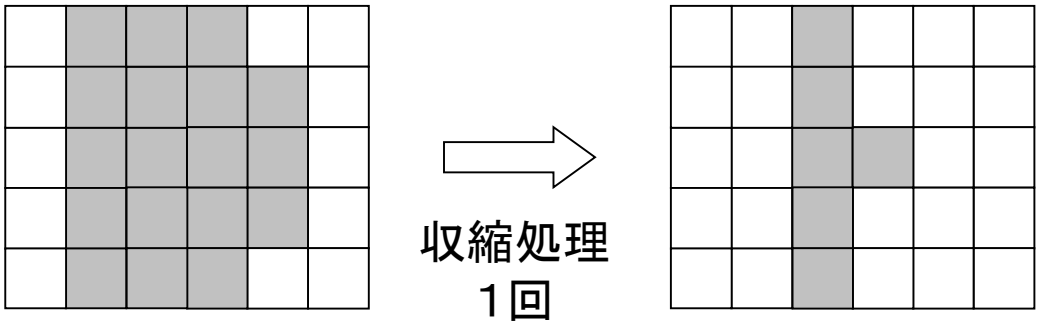
$$g[i][j]=\begin{cases} 1 & : \ f[i][j] \text{あるいはその8近傍（または4近傍）のいずれかが1のとき} \\ 0 & : \ \text{その他のとき} \end{cases}$$



膨張・収縮処理

- 収縮処理
  - ある画素とその8近傍（または4近傍）のいずれかに少なくともひとつの背景画素（0）がある場合，出力画素を0とする

$$g[i][j]=\begin{cases} 0 & : \quad f[i][j]\text{あるいはその8近傍（または4近傍）のいずれかが0のとき} \\ 1 & : \quad \text{その他のとき} \end{cases}$$



膨張・収縮処理

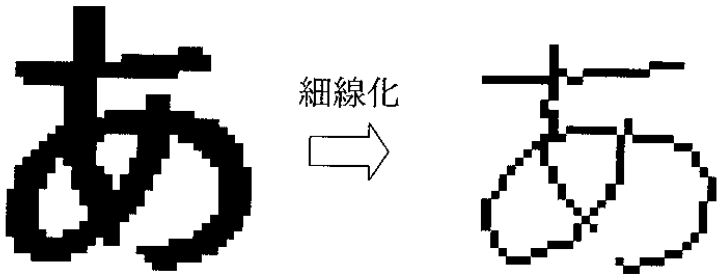
- 膨張処理と収縮処理の組み合わせ
  - 膨張→収縮処理
    - 連結成分の小さな孔や幅の狭いくぼみの部分（亀裂や分裂など）を埋める効果をもつ
  - 収縮→膨張処理
    - ２値画像の小成分や幅の狭い部分（ひげなどのノイズ）を取り除く効果をもつ

（ａ）　膨張と収縮の組み合わせ処理の例

（ｂ）　収縮と膨張の組み合わせ処理の例

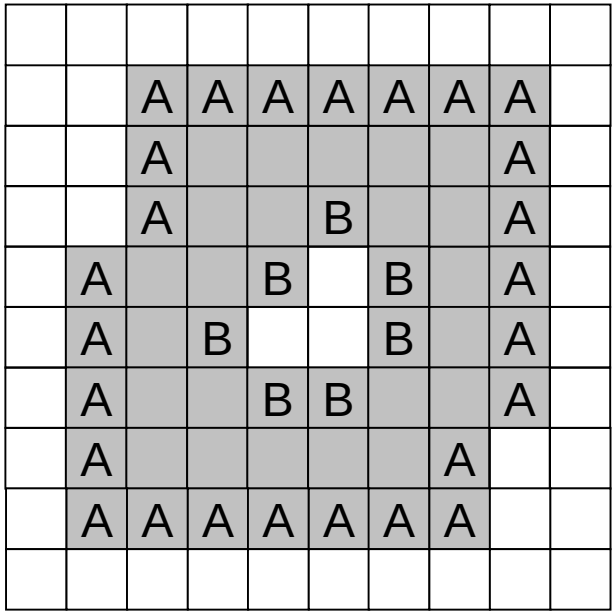
図形の細線化

- 細線化
  - 与えられた図形の線幅を細めて，幅1（画素）の中心線を抽出する
  - 縮退とは違い原図形の連結性が保持されている
- 細線化の方法
  - 図形画像の境界点の中から，消去可能であり，かつ線の端点でない画素を消去する
  - 消去可能→消去しても連結関係が変化しない



境界追跡

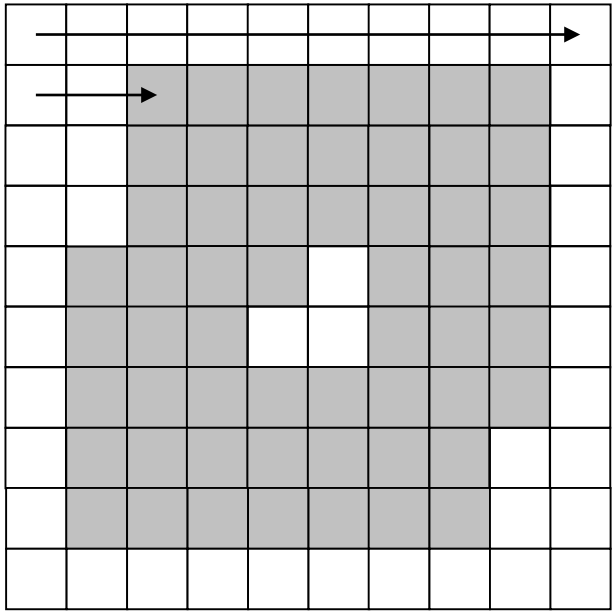
- 境界（輪郭線）追跡
  - 広がりのある塊状図形の境界画素を抽出する
  - 図形の形状を解析するのに利用
  - 単に境界画素の集合を抽出するだけでなく，順序付けられた画素の系列として抽出することができる



8連結で考えた場合の
境界追跡結果

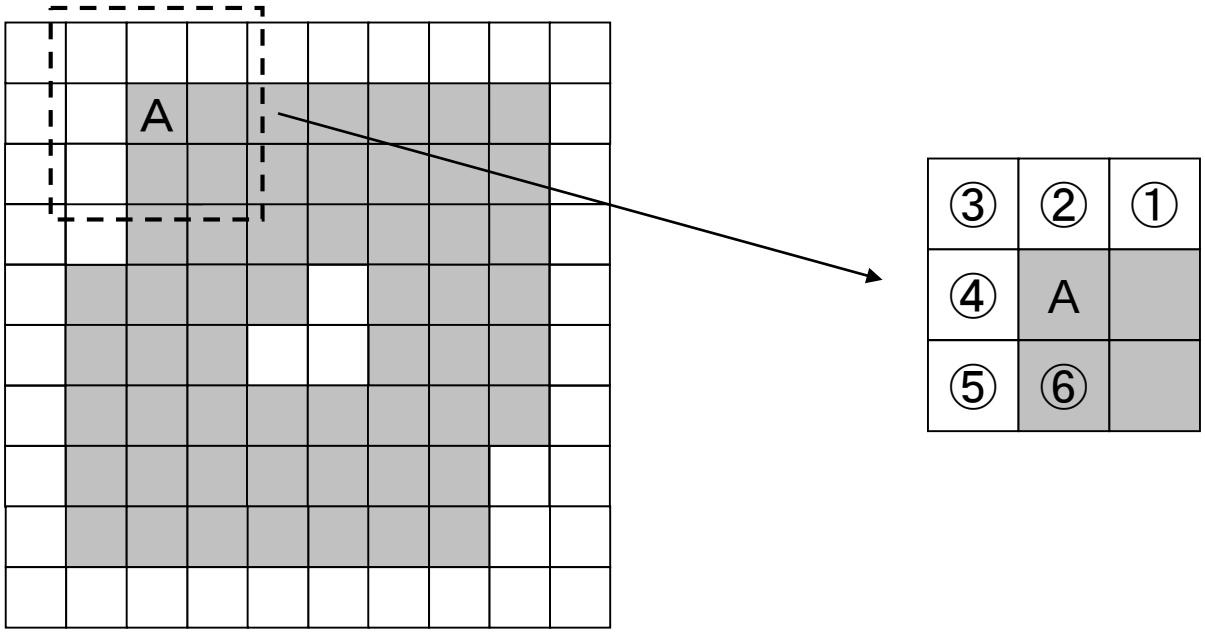
境界追跡

- 境界追跡のアルゴリズム（8連結）の例
  - 左上の画素から走査し，左側に0の画素が隣接する1の画素を見つけ，それを最初の境界画素とする
  - すでに境界画素として認識されていたものであれば無視する



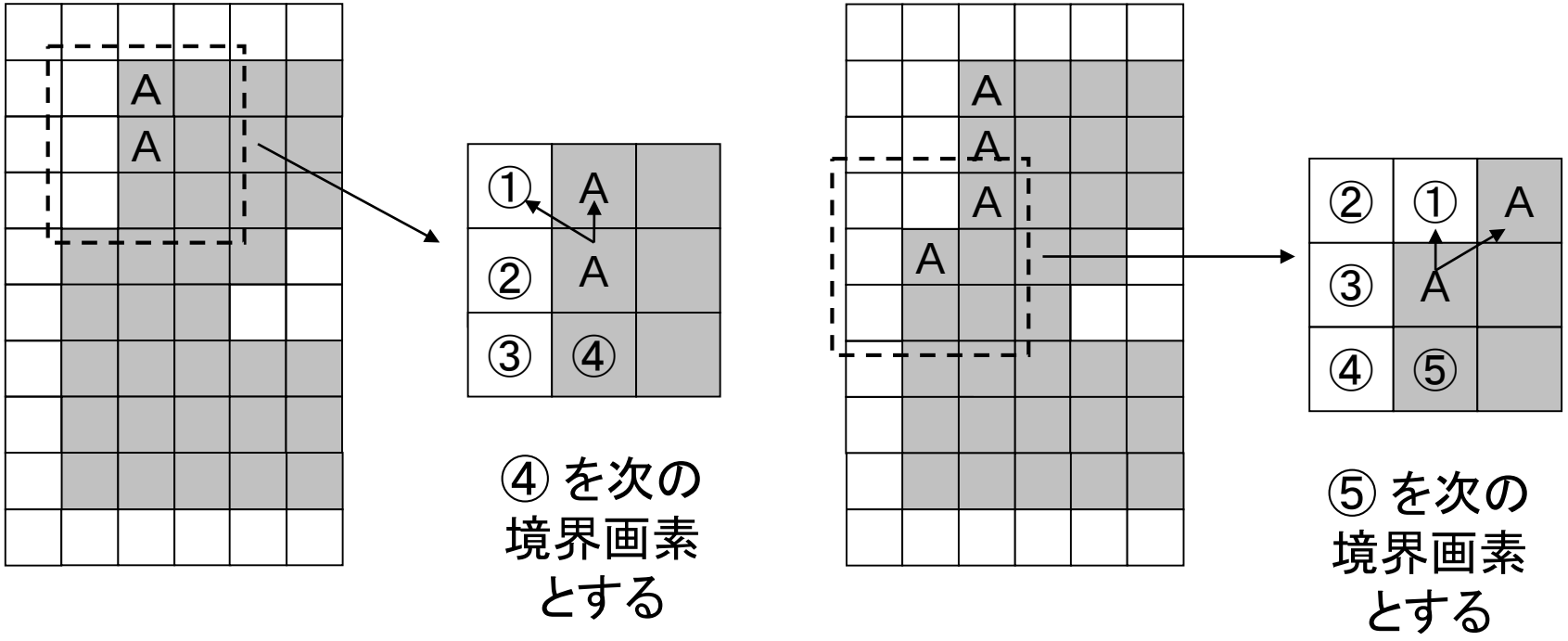
境界追跡

- 境界追跡のアルゴリズム（8連結）の例（続き）
  - 最初の境界画素を中心とする8近傍を，反時計回りに調べ，1の画素をみつける
  - その画素を次の境界画素とする（下図の例では⑥の画素）



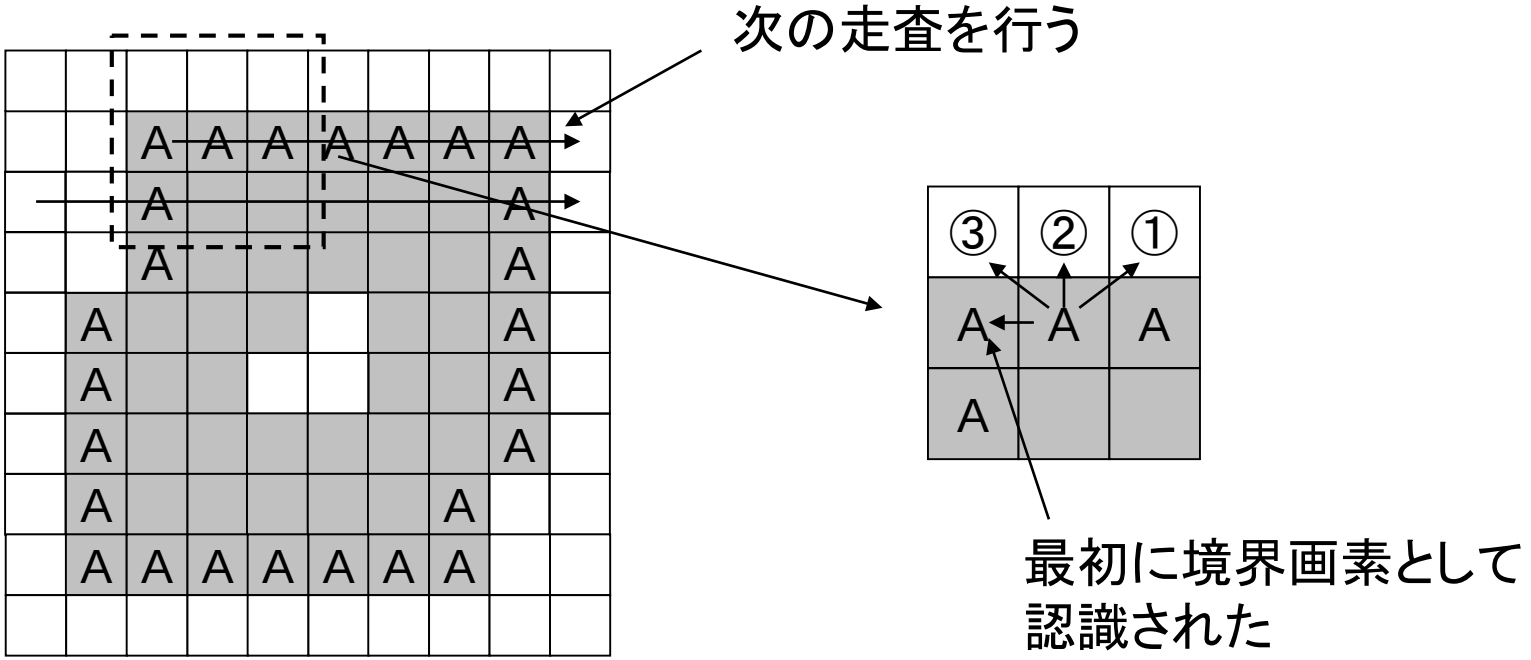
境界追跡

- 境界追跡のアルゴリズム（8連結）の例（続き）
  - 次の境界画素を見つけるときには，前の境界画素の（反時計回りに）隣りの画素から探索すればよい



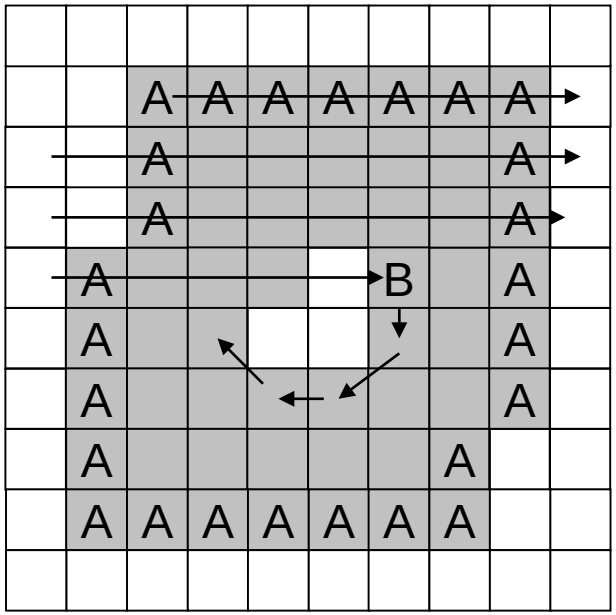
境界追跡

- 境界追跡のアルゴリズム（8連結）の例（続き）
  - 次の境界画素が，最初に認識された境界画素と一致したときには追跡処理をやめて，次の走査を行う
  - 走査を行う上で，すでに境界画素とされているものは無視していく



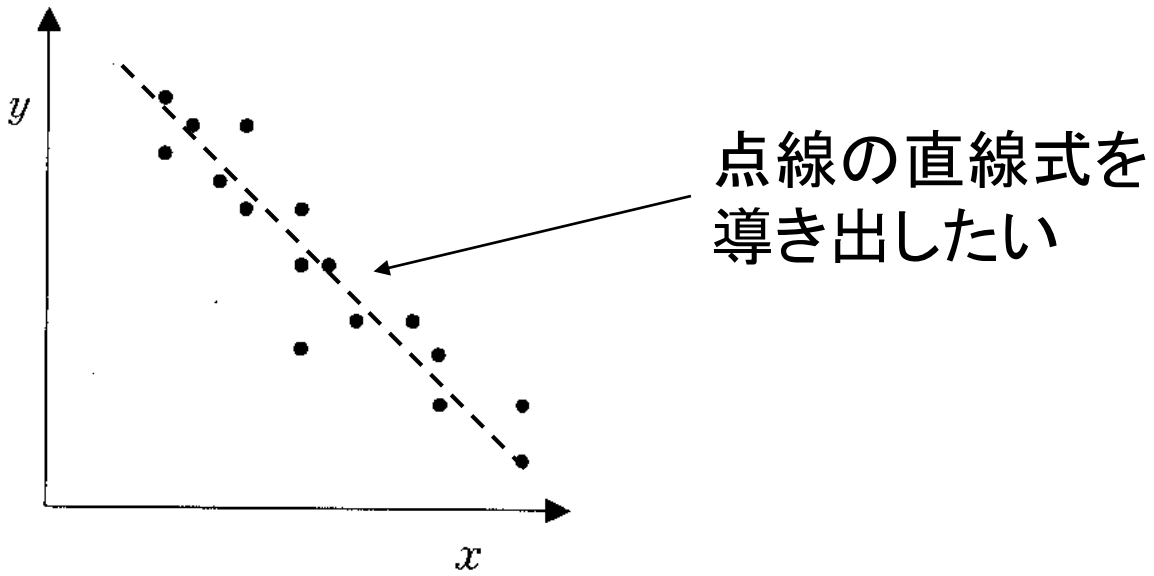
境界追跡

- 境界追跡のアルゴリズム（8連結）の例（続き）
  - 外側の境界は，図形に対して反時計回りに追跡される
  - 孔の部分の境界は，図形に対して時計回りに追跡される



直線の検出

- 直線の検出
  - ノイズや対象物の重なりなどの影響で散在した点（画素）から，本来抽出したい直線を導き出す



（a） 線の候補となる画素

直線の検出

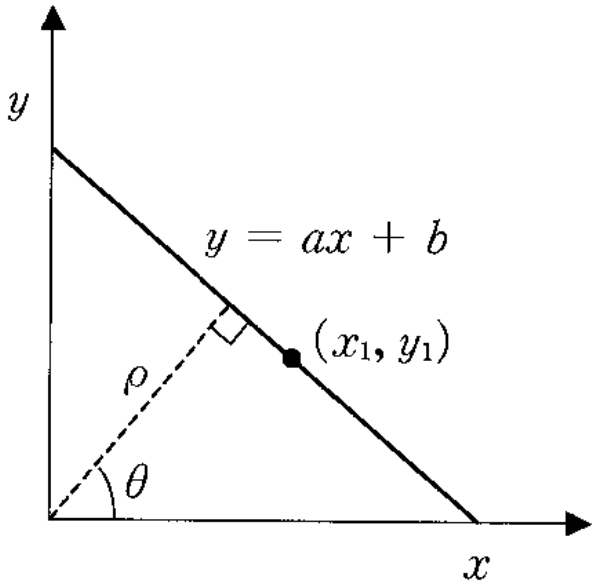
- ハフ(Hough)変換を用いた直線の検出
  - ある1点(x_1, y_1)が与えられたとき，これを通る直線は，傾きaと切片bのパラメータを用いて

    $$y = ax + b$$

    と表される
  - 原点から直線に下ろした垂線の長さρと，垂線がx軸となす角度θのパラメータを用いると，直線式は

    $$\rho = x\cos\theta + y\sin\theta$$

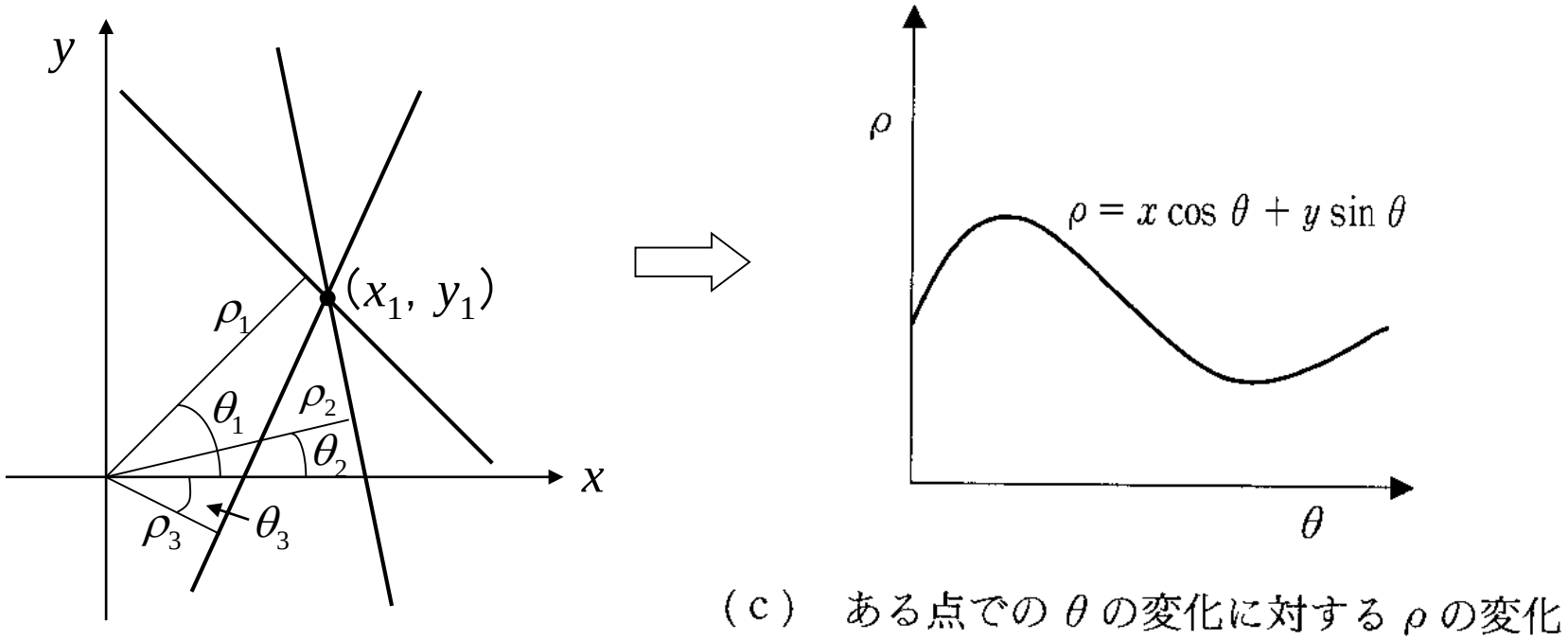
    とも表すことができる



（b） 点（x_1，y_1）を通る直線

直線の検出

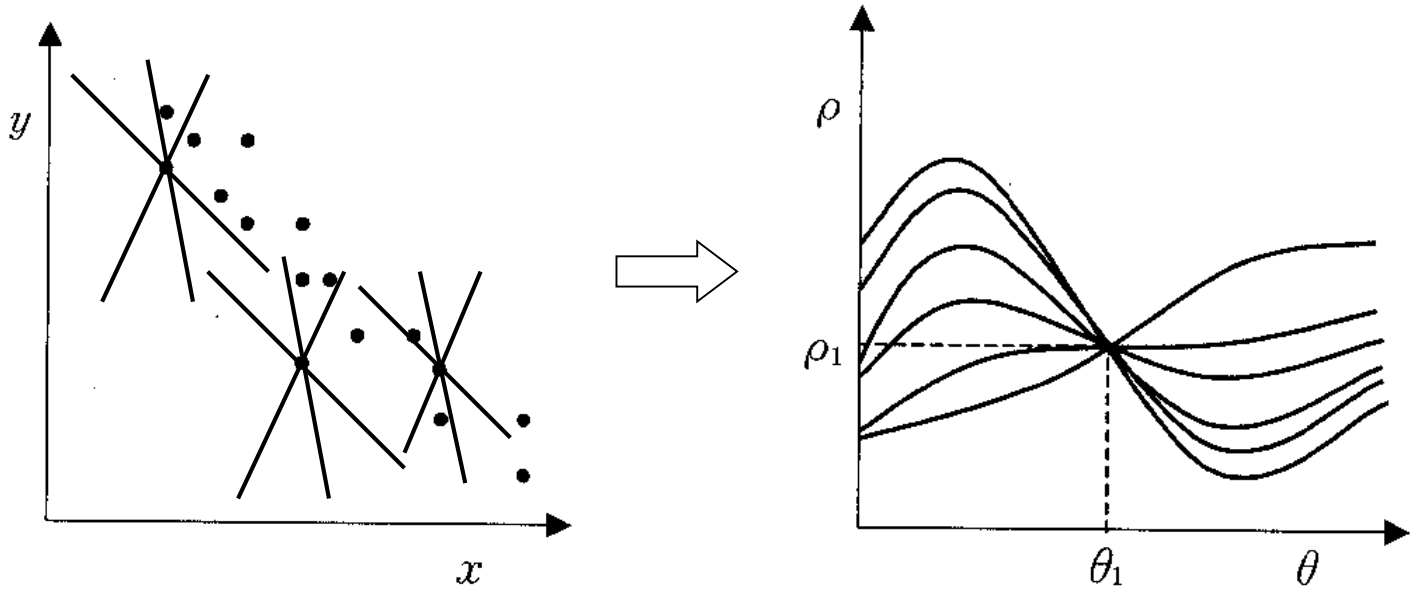
- ハフ(Hough)変換を用いた直線の検出
  - 点(x_1, y_1)を通るすべての直線をパラメータ(θ, ρ)で表したとき，θの変化に対するρの変化をグラフ化すると下図のようになる



（c） ある点での θ の変化に対する ρ の変化

直線の検出

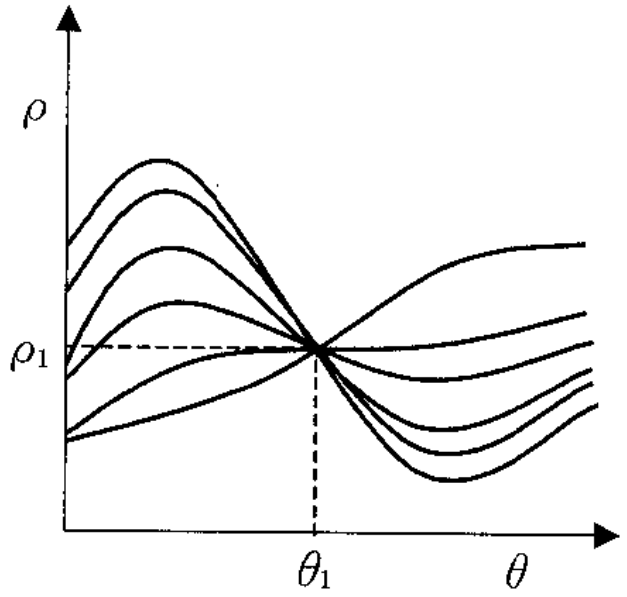
- ハフ(Hough)変換を用いた直線の検出
  - 与えられたすべての点についても，同様に曲線を求めると，下図のようになる



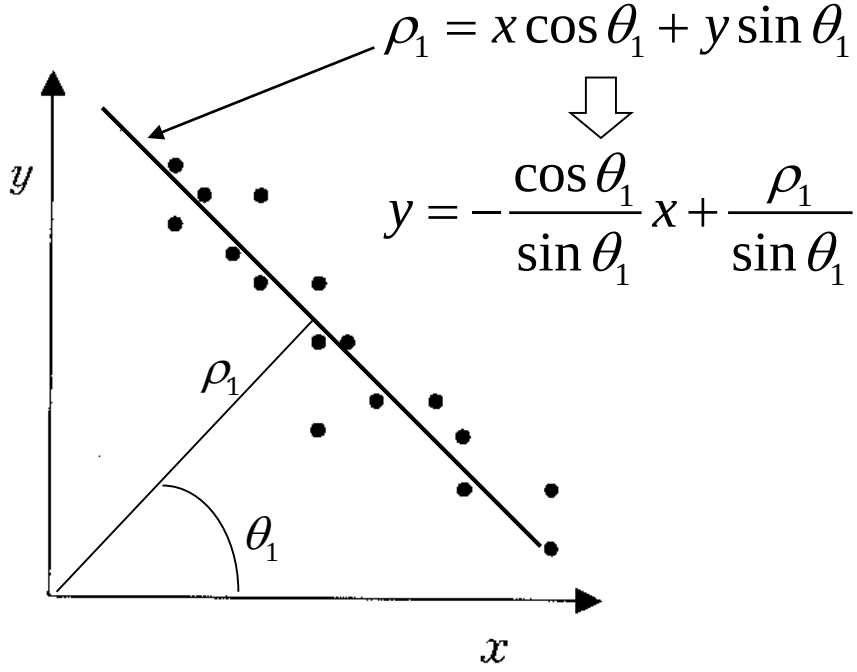
（a） 線の候補となる画素　　（d） 直線の候補の決定

直線の検出

- ハフ(Hough)変換を用いた直線の検出
  - 曲線が最も多く交わる点(θ_1, ρ_1)を見つければ1本の直線を検出することができる



(d) 直線の候補の決定



(a) 線の候補となる画素

直線の検出

- ハフ変換の利点
  - 同時に複数の直線を検出することができる
  - ノイズを含む2値画像からも直線を検出できる
- ハフ変換の欠点
  - 計算時間がかかる
  - 短い線分を多数含む図形には不向き