画像処理および演習

2年次後期（第8&9回）

中島 克人

情報メディア学科

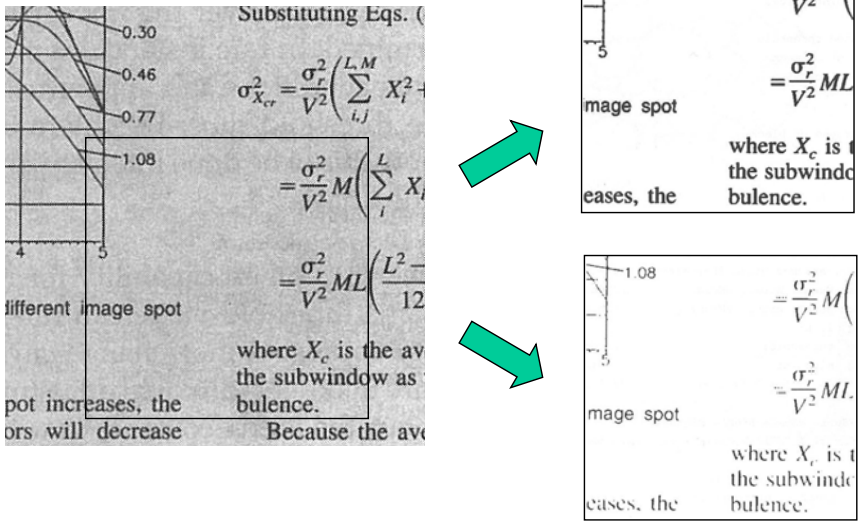
nakajima@mail.dendai.ac.jp

2値画像処理

- 2値化
  - ハーフトーニング，p-タイル法，モード法，判別分析法
- 2値画像の基本処理と計測
  - 連結性，輪郭追跡，収縮・膨張処理，ラベリング
  - 形状特徴パラメータ，距離
- 線画像のベクトル化
  - ベクトル化処理の流れ
  - 細線化手法，細線の特徴点抽出，ベクトル化

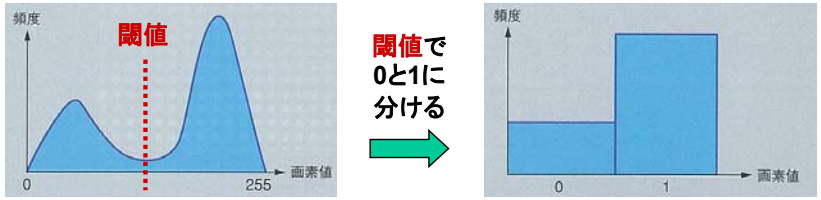
2値画像処理

- 2値化の例



2値化

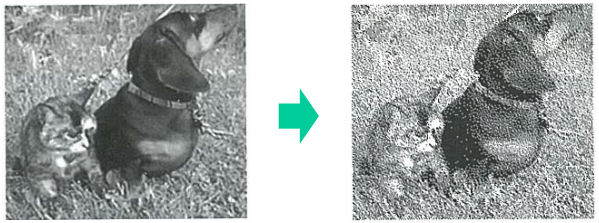
- 2値化（binarization）
  - 中間値を無くした完全な白黒の2階調化
  - 通常，白画素＝1，黒画素＝0
- 2値化の目的と方法
  - OCRやFAXなどで，白黒2色に見える文書も実はグレースケール画像のため，2値化により以後の処理を効率化
  - 閾値（threshold）を決めてグレースケールから2値に変換



●図9.1 文書画像のヒストグラム　●図9.2 2値化後のヒストグラム

ハーフトーニング

- 印刷時に画像の濃淡を連続的な階調で表現
  - 図a.28

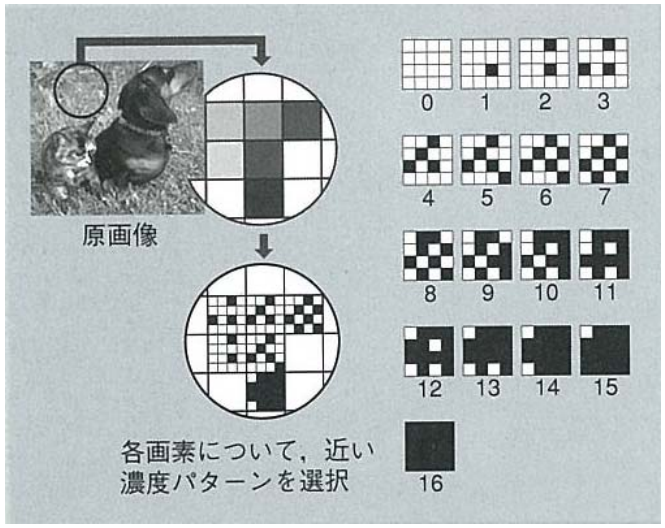


原画像　　ハーフトーン画像

- 代表的な3つの方法
  - 濃度パターン法
  - ディザ法
  - 誤差拡散法

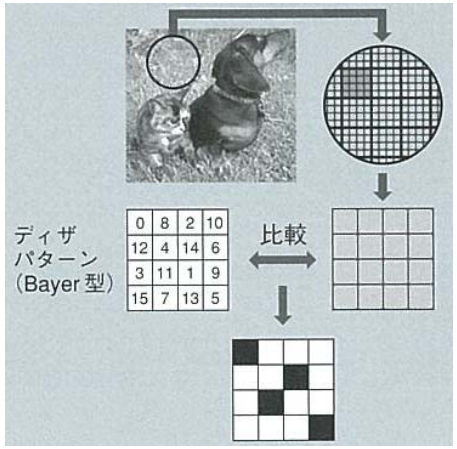
濃度パターン法

- 原画像1画素に対して数画素からなる2値画像パターンで表現
  - 4×4の例
    - 図a.29



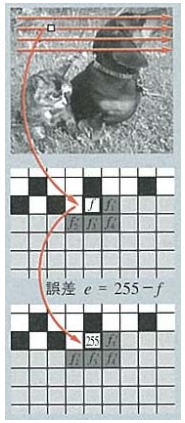
ディザ法（dither method）

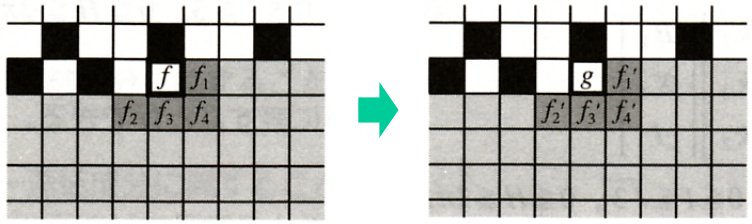
- 原画像と出力2値画像の画素数が同じ場合
- ディザパターン（通常4×4）と原画像の各画素値と比較し，その大小で白か黒を決定
  - 画像を4×4のブロックに分割
  - ブロック内の各画素がディザパターン内の対応画素の値×16＋8以上ならば白，そうでないならば黒に置き換える
  - この変換を全ブロックで行なう
    - 図a.30



誤差拡散法

- アルゴリズム
  - 処理対象画素値f，2値化後の値g
    - if（f ＞ 127）g＝255，if（f ≦127）g＝0
  - f と g との誤差 e＝f － g（g＝0 または 255）をキャンセルするように周囲の未処理画素を補正
    - $f_1' = f_1 + (5/16)\cdot e$，$f_3' = f_3 + (5/16)\cdot e$，
    - $f_2' = f_2 + (3/16)\cdot e$，$f_4' = f_4 + (3/16)\cdot e$，
    - 図a.31





誤差拡散法

●問題1：下図のf＝64 の時，2値化後のgと f_1'～f_4' を求めよ

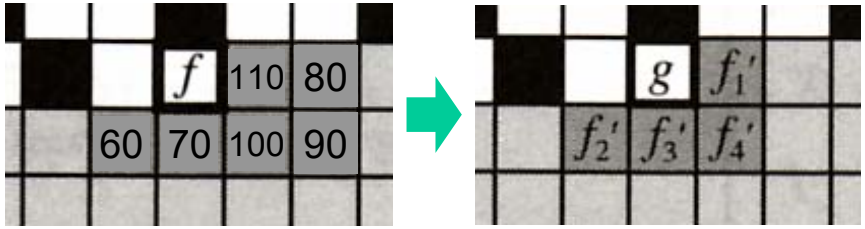
解答：g＝

f_1'＝

f_2'＝

f_3'＝

f_4'＝



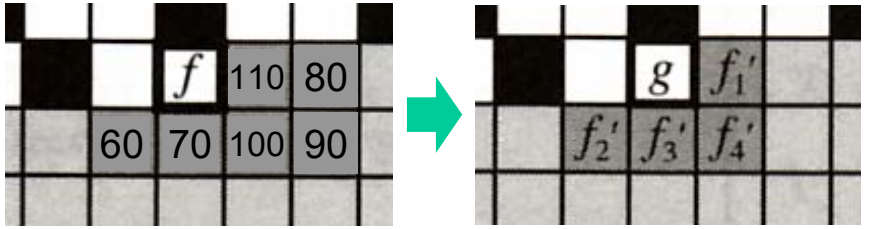
誤差拡散法

●問題2：続いて f_1' の2値化後の f_3'～f_4' の値を求めよ

解答：f_1'＝　　となるので，

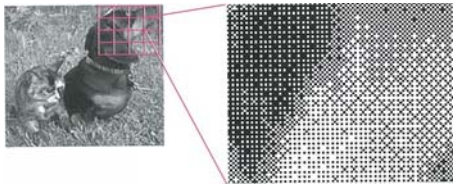
f_3'＝

f_4'＝

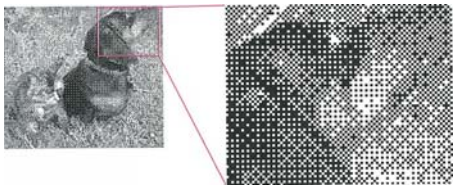


ハーフトーニングの結果比較

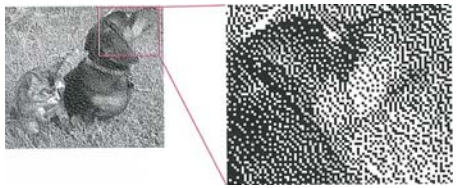
- 濃度パターン法/ディザ法/誤差拡散法
  - 図a.32



濃度パターン法



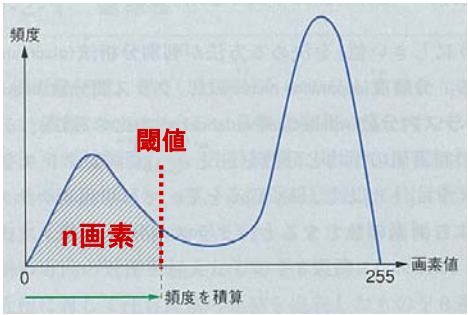
ディザ法



誤差拡散法

2値化

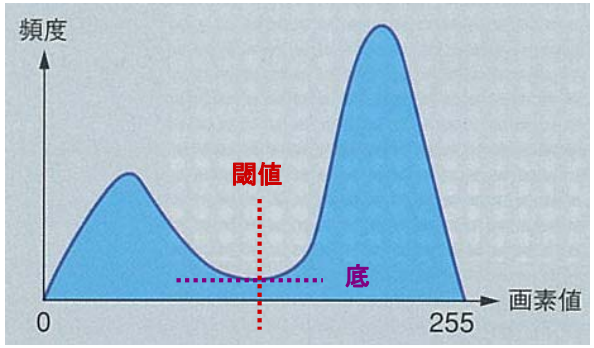
- p-タイル法（p-tile method／percentile method）
  - ◆文書画像では画像中の文字の占める画素数が予想可能
  - ◆この予想画素数（n）に応じて閾値を決定
    - 予想できない画像に対しては不適な方法
  - ◆画素値の低い所から頻度値（ヒストグラムの度数）を積算し，予想画素数（n）を超えた（あるいは超える直前）の画素値を閾値とする



●図9.3 p-タイル法による閾値の決め方

2値化

- モード法（mode method）
  - ヒストグラムの2つの山の間の谷の一番低い所を閾値とする
    - 画像の画素数が不十分であったり，2つの山とその間の谷がはっきりしない場合には，安定した2値化ができない
    - 谷の底が広いと，ノイズ等により「一番低い所」が揺らぎやすい



●図9.1 文書画像のヒストグラム

2値化

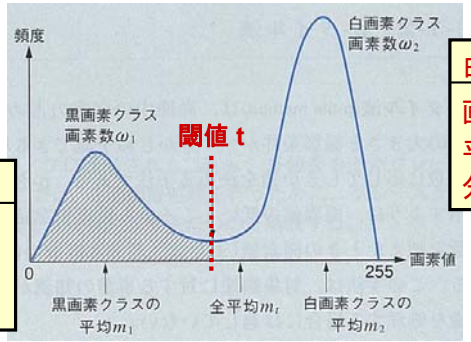
- 判別分析法（discriminant analysis method）
  - 黒山と白山の分離度が最大となる閾値t を求める

分離度（separation metrics）＝ クラス間分散（between- class variance） / クラス内分散（within-class variance）

全画素
画素数：ω_t
平均：m_t
分散：σ_t^2

黒画素クラス
画素数：ω_1
平均：m_1
分散：σ_1^2

白画素クラス
画素数：ω_2
平均：m_2
分散：σ_2^2



●図9.4 判別分析法による閾値tの決め方

2値化

ω_1，ω_2：画素数
m_1，m_2：平均
σ_1^2，σ_2^2：分散

- 判別分析法
  - 以下に定義される分離度が最大となる閾値t を，画素値の値域内（ex. 0～255）で探索して求める

クラス間分散 $$\sigma_b^2 = \frac{\omega_1(m_1-m_t)^2+\omega_2(m_2-m_t)^2}{\omega_1+\omega_2} = \frac{2\omega_1\omega_2(m_1-m_2)^2}{(\omega_1+\omega_2)^2}$$ ●式9.2

クラス内分散 $$\sigma_w^2 = \frac{\omega_1\sigma_1^2+\omega_2\sigma_2^2}{\omega_1+\omega_2}$$ ●式9.1

全分散 $\sigma_t^2 = \sigma_b^2 + \sigma_w^2$ ●式9.3

分離度＝ クラス間分散 / クラス内分散 $$\frac{\sigma_b^2}{\sigma_w^2} = \frac{\sigma_b^2}{\sigma_t^2-\sigma_b^2}$$ ●式9.4

  - 全分散は閾値に関わらず一定のため，クラス間分散の分子 $\omega_1\omega_2(m_1-m_2)^2$ を最大化すればよい

演習（2値化）

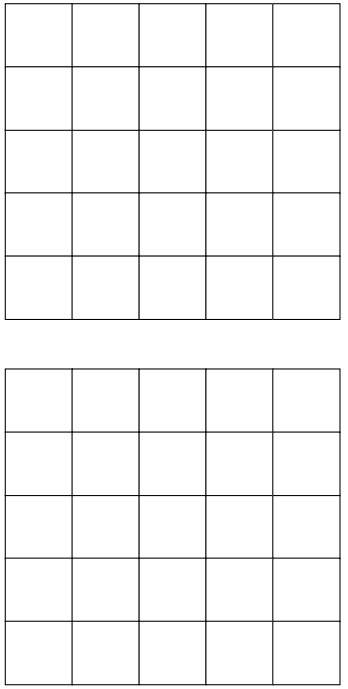
- 下記の5×5画素のグレースケール画像（256階調）を下記の方法でそれぞれ2値化（0と1）せよ
  - pータイル法
    （画素値の低い方から60%を閾値とする）
  - モード法
    （ヒストグラムは8ビンとする）

20	35	150	185	140
45	130	75	165	180
65	70	55	170	205
190	170	145	70	210
160	55	45	60	55

グレースケール原画像

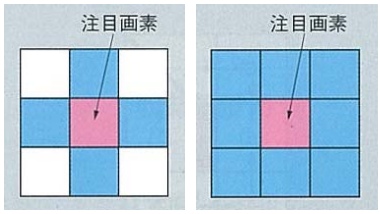
◆pータイル法での2値化

◆モード法での2値化



2値画像の基本処理と計測

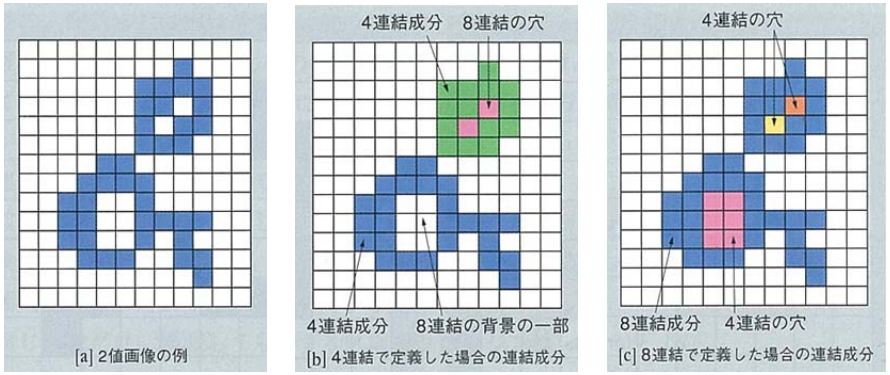
- 連結性
  - ◆画素値1（以降，黒（または白色以外の）画素で説明）を持つひとまとまりの領域を定義するため，連結（connection）という概念を導入
  - ◆4近傍（4-connected neighbor）
    - 注目画素に対して上下左右の4 画素
  - ◆4連結（4-connection）
    - 4近傍の画素も同じ画素値
  - ◆8近傍（8-connected neighbor）
    - 4近傍の4画素＋斜め方向4画素
  - ◆8連結（8-connection）
    - 8近傍の画素も同じ画素値



●図9.5 4近傍/4連結（左），8近傍/8連結（右）

2値画像の基本処理と計測

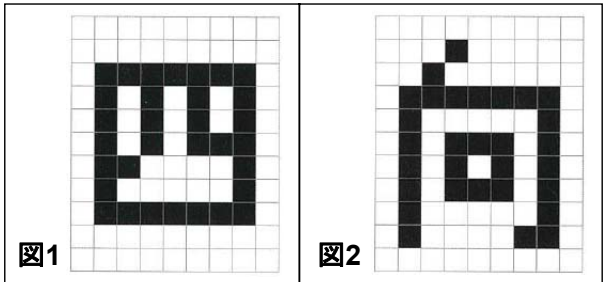
- ◆連結成分（connected component）
  - 連結している画素の集合
- ◆穴（hole）
  - 連結成分に囲まれ，背景に連結していない白画素（画素値0）集合
- ◆4連結で見るか8連結で見るかで連結性は異なる



●図9.6 2値の連結成分

演習（連結性）

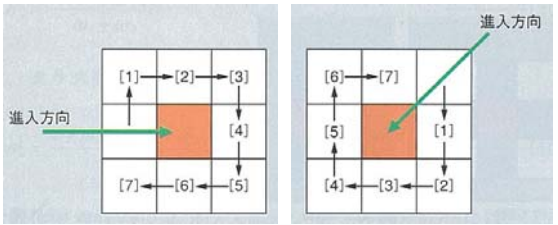
- 2値画像中の図形が持つ連結成分の数や穴の数などの形状特徴は，画素の連結性を4連結で定義するか，8連結で定義するかによって異なる．
- 図1と図2に関して，4連結で定義した場合の連結成分および穴の数はそれぞれ幾つかを答えよ．ただし，白画素を背景，黒画素を対象の領域とする．



	連結成分の数	穴の数	連結成分の数	穴の数
4連結で定義した場合				
8連結で定義した場合				

2値画像の基本処理と計測

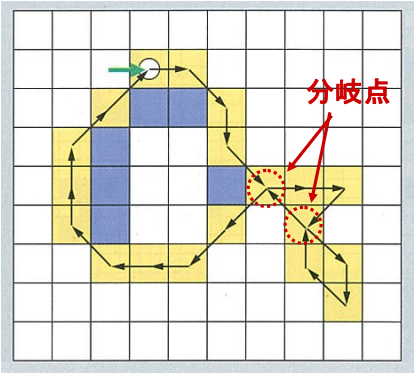
- 輪郭追跡（contour tracking）
  - ◆8連結でのアルゴリズム（4連結でもほぼ同じ）
  - （1）ラスタスキャン（raster scan／左上から右下への走査）により，白画素から黒画素への変わる画素を発見
  - （2）探索した方向（進入方向）を起点に右回りで黒画素を探索
  - （3）見つかった黒画素に移動し，それが開始点でかつ，移動点が追跡済みの場合は処理を終了し結果を書き出す．そうでない場合は（2）に戻る．



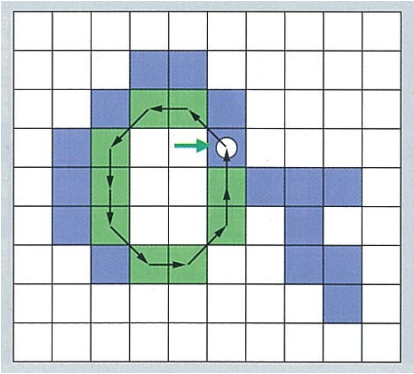
●図9.7 黒画素の探索

2値画像の基本処理と計測

- 輪郭追跡
  - (3)において「移動点が追跡済みの場合は処理を終了」とすると，分岐点等の2回通るべき点で追跡が止まってしまう
  - 内輪郭の追跡は同じアルゴリズムで自動的に求められる



●図9.8 外輪郭の追跡



●図9.9 内輪郭の追跡

2値画像の基本処理と計測

- 収縮(erosion)
  - 背景または穴に接する黒(値1)画素をひとまわり剥ぎ取る
  - 収縮を繰り返すと黒画素領域がやせ細り，やがて消滅する
- 膨張(dilation)
  - 背景または穴に接する黒画素にひとまわり付け加える
  - 膨張を繰り返すと穴は小さくなり，やがて塞がる

●図9.12 収縮と膨張

演習(膨張)

- 下記の7×7画素の2値画像(2階調，ここでは0を背景，1を前景と称する)の前景に4近傍の膨張(dilation)を1回施した結果画像を図示し，8近傍の膨張ならば前景となる画素も追加図示せよ。

0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0

2値原画像

膨張

0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0

→ 膨張結果

演習(輪郭追跡と収縮)

- 下記の7×7画素の2値画像(2階調，ここでは0を背景，1を前景と称する)の前景の8連結の外輪郭をその図の上で塗りつぶし，その輪郭のみを剥がした収縮結果を右に図示せよ

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	1	0
0	1	0	1	1	1	0
0	1	0	1	1	0	0
0	0	1	1	1	1	1
0	0	0	1	1	0	0

2値原画像 → 外輪郭追跡結果

収縮

→ 収縮結果

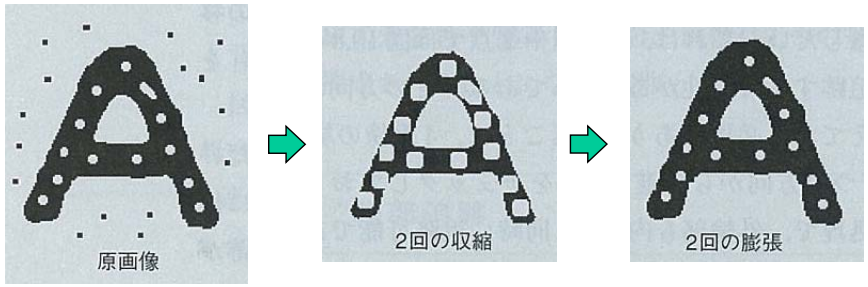
2値画像の基本処理と計測

- クロージング（closing）
  - 同じ回数だけ膨張して収縮する
  - 画像の小さな穴を塞ぐことが出来る

●図9.13　小さな穴と小さな連結成分を含む2値画像のクロージング

2値画像の基本処理と計測

- オープニング（opening）
  - 同じ回数だけ収縮して膨張する
  - 画像の小さな連結成分を除くことが出来る



●図9.12　小さな穴と小さな連結成分を含む2値画像のクロージング

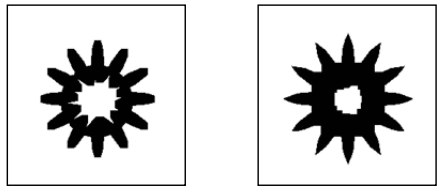
2値画像の基本処理と計測

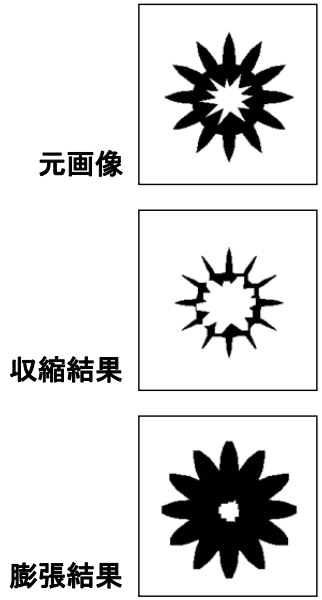
- クロージングとオープニングによるノイズ除去例
  - クロージングでまず穴を塞ぎ，次にオープニングで背景のノイズを除去

●図9.13　ノイズを含む原画像にクロージングとオープニングを適用

収縮・膨張フィルタのまとめ

- 収縮（erosion）
  - 対象図形を1画素分だけ縮める
- 膨張（dilation）
  - 対象図形を1画素分だけ膨らませる
- opening（ノイズ除去に利用可）
  - erosion の後に dilation
- closing （ノイズ除去に利用可）
  - dilation の後に erosion





演習（2値画像のノイズ除去）

- 図1の2値画像に対して，連結成分中の白画素の領域を取り除くためにある処理を行ったところ，図2の画像を得た．
  この処理は（a）～（d）のいずれか？
  ただし，白画素を背景，黒画素を対象の領域とする．

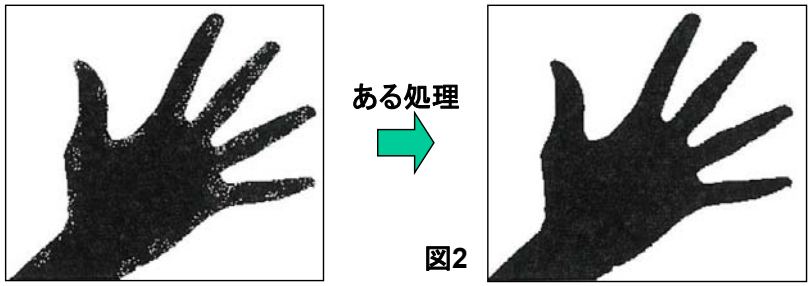
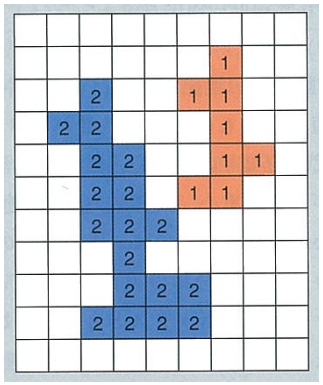


図1　　図2

（a）クロージング　（b）膨張　（c）収縮　（d）オープニング

2値画像の基本処理と計測

- ラベリング（labeling）
  - ◆各連結成分にユニークなラベル（通し番号等）を付与する
  - ◆連結成分ごとに色分けしたり，面積（画素数）や形状を計測したりする際に前処理として行なう



●図9.16　4連結を用いたラベリングの例

2値画像の基本処理と計測

- ラベリングのアルゴリズム（4連結[および8連結]）
  - ◆ラベル付けフェーズ

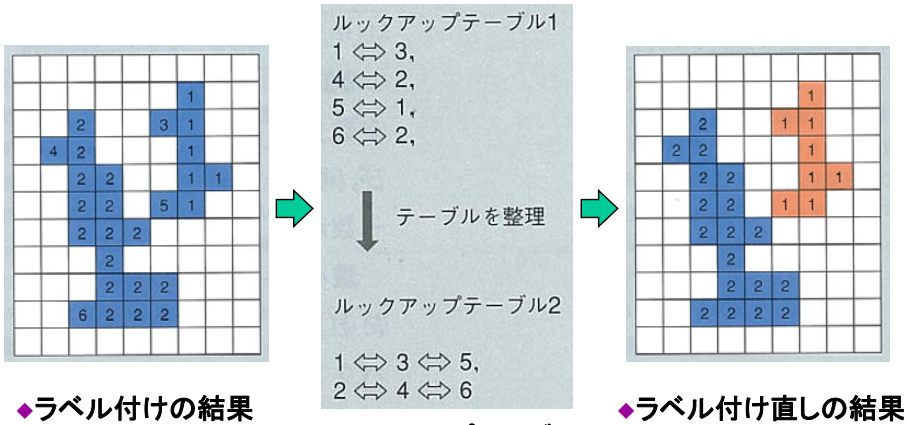
（1）ラスタスキャンにより，ラベルの付与されていない画素を見つけて，それを注目画素とする．
（2）注目画素の上の画素[または左上の画素]がラベルを持つとき，上の画素[または左上の画素]の[どちらかの]ラベルを注目画素に付与する．左の画素ラベルを持ち，注目画素のラベルと異なる時，ルックアップテーブルに2つのラベルが同一連結成分に属することを記録する．
（3）注目画素の上の画素[および左上の画素]が白（0）画素で，左の画素がラベルを持つとき，そのラベルを注目画素に付与する．
（4）注目画素の上も左も[左上の画素も]白画素の時，新しいラベルを注目画素に付与する．
（5）走査する画素がなくなるまで（1）から繰り返す．

  - ◆ラベル付け直しフェーズ

再度ラスタスキャンを行い，ルックアップテーブルを参照しながら，同一の連結成分に属するラベル群から，最も小さいラベルを選んでラベルを付け直す

2値画像の基本処理と計測

- ラベリングの例

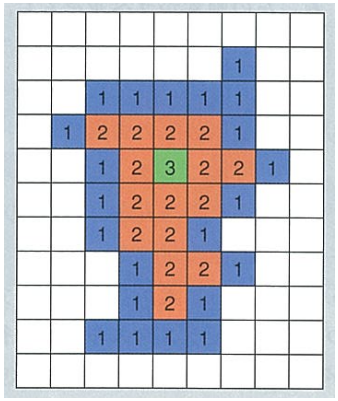


◆ラベル付けの結果　◆ルックアップテーブルの内容　◆ラベル付け直しの結果

●図9.16　4連結を用いたラベリングの例

2値画像の基本処理と計測

- 輪郭追跡によるラベリング
  - ◆輪郭追跡のアルゴリズムにより外輪郭にラベルを付与（ 1 ）
  - ◆外輪郭の画素から近傍の画素にラベルを伝播（ 2 → 3 ）



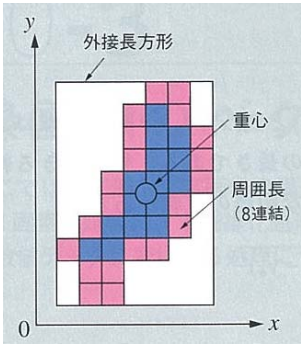
説明のために異なるラベル（番号）が付与されているが，実際には 1 2 3 は同じラベルを付与

- ◆ **距離変換画像**（distance transform image）
  - 異なるラベル（番号）を敢えて付与した画像
  - 番号が周辺（白（0）画素）からの距離に相当

●図9.17 8連結を用いたラベリングの例

2値画像の基本処理と計測

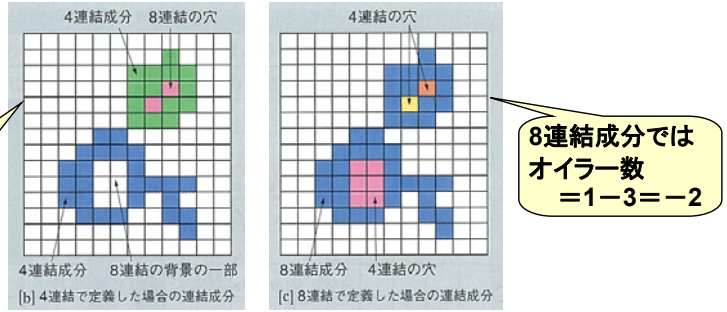
- 形状特徴パラメータ（geometric feature parameter）
  - ◆それぞれの連結成分の特徴を表すためのパラメータ
  - ◆面積（area） … 対象の総画素数
  - ◆重心（centroid） … 画素に等しい重さがあると仮定
  - ◆外接長方形（bounding box） … 外接する最小の長方形
  - ◆周囲長（perimeter） … 輪郭追跡し，一周する移動量
    - 8隣接で追跡したのが右図
    - 上下左右への移動量をC_1，斜めへ移動量をC_2とする時，周囲長＝$C_1+\sqrt{2}\cdot C_2$ とする



●図9.18 連結成分の重心，外接長方形，周囲長

2値画像の基本処理と計測

- 形状特徴パラメータ（続）
  - ◆円形度（roundness） … どれだけ丸いかを表す尺度
    - 面積S，周囲長Lとしたとき，円形度＝$4\pi S/L^2$
    - 円の時1（最大値）で複雑な（周囲長の長い）図形ほど小さくなる
  - ◆オイラー数（Euler number） … 連結成分の数から穴の数を引いたもの（複数の連結成分からなる対象にも適用可能）



●図9.6 2値の連結成分

2値画像の基本処理と計測

- 形状特徴パラメータ（続）
  - ◆モーメント特徴（moment feature）
    - 画素の位置に重み付けをして合計した数値
    - 対象の大きさ，位置，どちらの方向に長いか 等を表す
    - 面積と重心等から主軸（principal axis）方向を求めることも出来る

*pq*次モーメント ●式9.5

$$M(p,q)=\sum_{i,j} i^p j^q$$

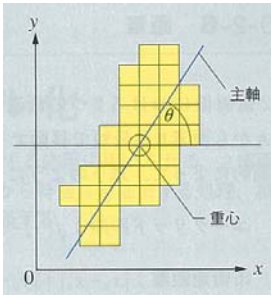
（*i*, *j*）：対象画素の位置

$M(0,0)=\Sigma i^0\cdot j^0=\Sigma 1$＝対象の面積
$M(1,0)=\Sigma i^1\cdot j^0=\Sigma i$ ＝x軸での1次モーメント
$M(1,0)/M(0,0)$＝x軸上での対象の重心

- 主軸は下記の式をθについて求めたもの（詳細略）

●式9.6

$$\tan^2\theta+\frac{M(2,0)-M(0,2)}{M(1,1)}\tan\theta-1=0$$



●図9.19 モーメント特徴を用いた主軸（傾き）

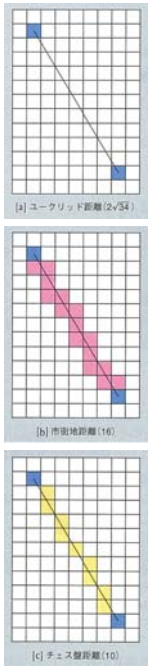
2値画像の基本処理と計測

- 距離（distance）
  - x,y軸の2次元空間での距離

ユークリッド距離：$\sqrt{(x_a-x_b)^2+(y_a-y_b)^2}$

市街地距離（マンハッタン距離）：$|x_a-x_b|+|y_a-y_b|$

チェス盤距離：$\max(|x_a-x_b|,|y_a-y_b|)$



◆2つの画素A,Bの色がRGB色空間上でどの程度異なるか，等の尺度にも用いる
  - 全ての軸が距離的に対等である事が前提

●図9.20 2画素間の各種距離

演習（距離と類似度）

- 画素A, B, CのRGB値がそれぞれ（50,40,30），（60,90,50），（80,70,60）である.
- 画素Aと比べて，画素Bと画素Cはどちらが色が近いか調べたい
  - ◆RGB空間上でのマンハッタン距離ではどうか？

マンハッタン距離をM（A,B），M（A,C）とすると，

M（A,B）＝

M（A,C）＝

解答	

  - ◆RGB空間上でのユークリッド距離ではどうか？

ユークリッド距離をD（A,B）,D（A,C）とすると，

$D(A,B)^2$＝

$D(A,C)^2$＝

解答	

演習（距離と類似度と正規化）

- Aさん, Bさん, Cさんの数学（100点満点）と英語（50点満点）の成績が（70,40），（80,38），（73,34）である.
- Aさんと比べて，BさんとCさんはどちらが成績が似ているか？
  - ◆マンハッタン距離ではどうか？

マンハッタン距離をM（A,B），M（A,C）とすると，

M（A,B）＝ | 70－80 | ＋ | 40－38 | ＝ 12

M（A,C）＝ | 70－73 | ＋ | 40－34 | ＝ 9

  - ◆ユークリッド距離ではどうか？

ユークリッド距離をD（A,B）,D（A,C）とすると，

$D(A,B)^2=(70-80)^2+(40-38)^2=100+4=104$

$D(A,C)^2=(70-74)^2+(40-34)^2=16+36=52$

解答	

演習（距離と類似度と正規化）

- Aさん, Bさん, Cさんの身長（m）と体重（Kg）がそれぞれ（1.70,60），（1.90,62），（1.67,63）である.
- Aさんと比べて，BさんとCさんはどちらが体型が近いか？
  - ◆マンハッタン距離ではどうか？

マンハッタン距離をM（A,B），M（A,C）とすると，

M（A,B）＝ | 1.70－1.90 | ＋ | 60－62 | ＝ 2.2

M（A,C）＝ | 1.70－1.67 | ＋ | 60－63 | ＝ 3.03

  - ◆ユークリッド距離ではどうか？

ユークリッド距離をD（A,B）,D（A,C）とすると，

$D(A,B)^2=(1.70-1.90)^2+(60-62)^2=4.04$

$D(A,C)^2=(1.70-1.67)^2+(60-63)^2=9.0009$

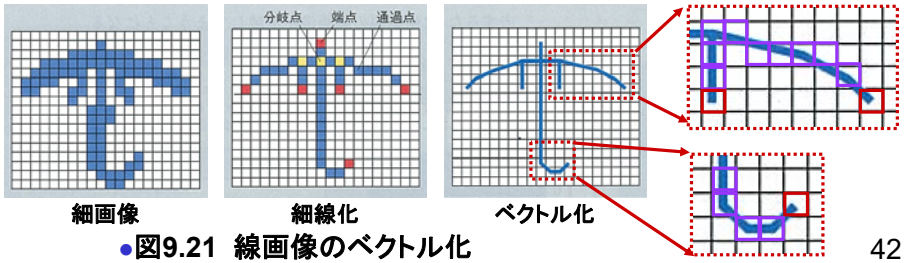
解答	

線画像のベクトル化

- ベクトル化処理の流れ（細線化とベクトル化）
- 細線化手法
- 細線の特徴点抽出
- ベクトル化

線画像のベクトル化

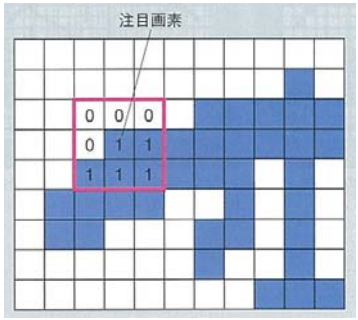
- 細線化
  - 連結性を保持しつつ画素を（1画素幅まで）削る処理
  - 端点（end point），分岐点（branch point），通過点（passing point）に分けられる
- ベクトル化
  - 図形や文字画像の構造を解析するため
  - 端点ー端点，端点ー分岐点，分岐点ー分岐点等のそれぞれの画素列に分割し，これらの始点から終点まで線分で近似



●図9.21　線画像のベクトル化

線画像のベクトル化

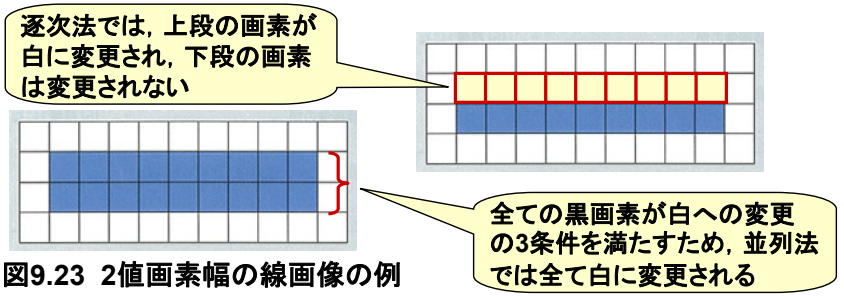
- 細線化手法（対象は黒（図では青）画素）
  - 画像全体を走査（輪郭のみを追跡しても良い）
  - 注目画素を中心とした3×3画素値のパターンが次の条件を満たすとき，その値を白画素（0）にする
    1. 注目画素が境界上にある黒画素
    2. 白画素に変更しても連結性が保たれる
    3. 端点ではない



●図9.22　3×3画素値のパターン例

線画像のベクトル化

- 細線化における画素値の更新
  - 逐次法
    - 原画像に直ちに更新をかける
    - 後から判定する方に（左上から右下に走査ならその方向に）線画像がずれる傾向有り
  - 並列法
    - 更新用（出力用）の画像を別に設ける
    - ちょうど2画素幅の線画像が消滅するという欠点有り

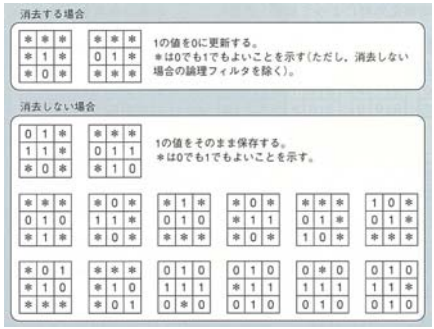


●図9.23　2値画素幅の線画像の例

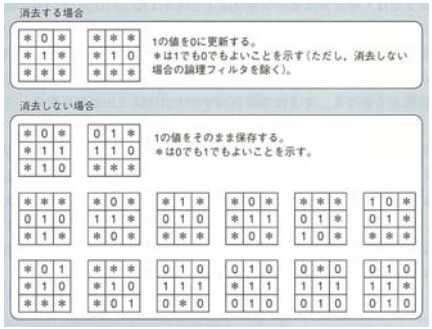
線画像のベクトル化

- 細線化手法例
  - ◆右記条件2と3を緩和
  - ◆2画素幅の線画消失を防止
  - ◆2つのサブサイクルからなる並列法
    - 第1サブサイクルは上と右から，第2サブサイクルは下と左から
    - 各サブサイクルで更新がなければ処理を終了し，あれば繰り返す

1. 注目画素が境界上にある黒(1)画素
2. 白(0)画素に変更しても連結性が保たれる
3. 端点ではない



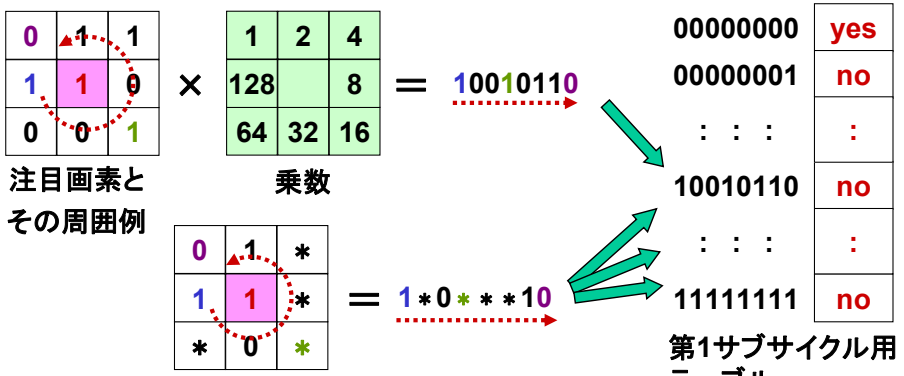
●第1サブサイクル用



●第2サブサイクル用

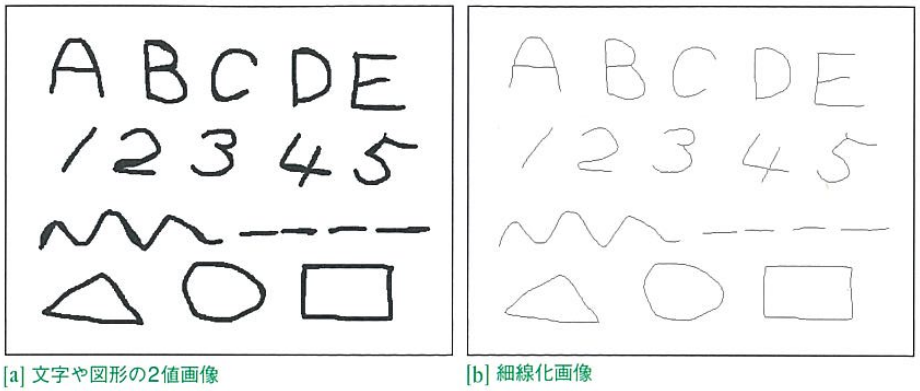
線画像のベクトル化

- 細線化のプログラミング法
  - ◆テーブル検索(table lookup)の利用
    - 3×3(＝9ビット)のビットパターンにより処理を切り替える
      ⇒ 中央画素が1の場合のみに限定し，2^8＝256通りの場合分け
      ⇒ 256エントリのテーブル検索で更新可否を判定



線画像のベクトル化

- 細線化結果例

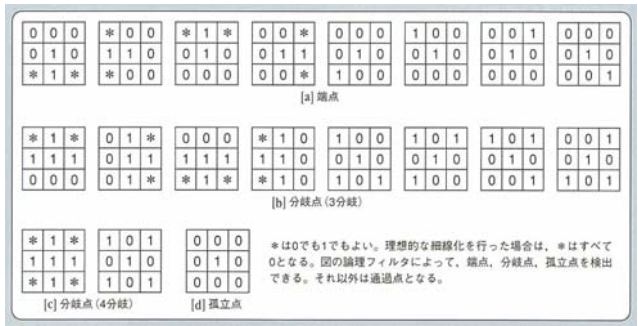


[a] 文字や図形の2値画像　[b] 細線化画像

●図9.24 文書画像の細線化処理例

線画像のベクトル化

- 細線の特徴点抽出
  - ◆細線化後の各黒(1)画素が，端点・分岐点・孤立点などの特徴点は，下記パターンと照合することにより検出可
    - 3×3の入力画素とAND演算(ただし，＊位置は無視)して論理値1となると照合が成立するため，論理フィルタと称する
    - 実際のプログラミングでは，テーブル検索が速くて簡単



問題：
端点・分岐点・孤立点のそれぞれのために3つのテーブルを検索せずに済ますにはどうすれば良いか？

●図9.25 特徴点を検出するためのパターン

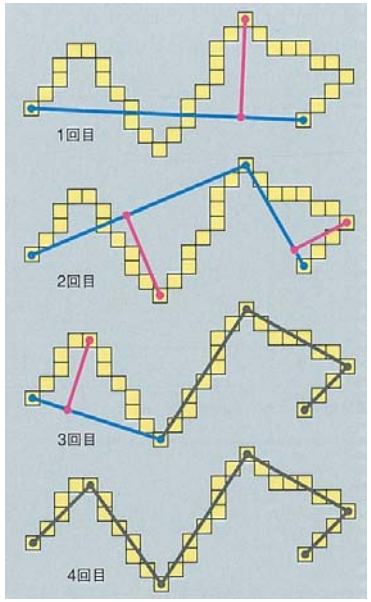
線画像のベクトル化

- ベクトル化（vectorization）
  - ◆細線化結果をある程度の長さの線分で近似表現
- ベクトル化アルゴリズム（1）
  - ◆2分割法

（1）画素列の始点と終点を繋いだ直線に対して，各画素の距離を計算

（2）最大距離が定めた閾値以下ならその直線を近似線分として採用し，そうでないならその最大距離の画素で画素列を分割し，分割した画素列でそれぞれ(1)を再帰的に処理する

（3）画素列の分割がなければ終了

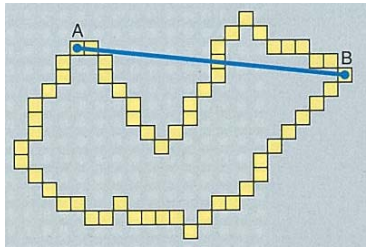


●図9.26 2分割法による近似線分

線画像のベクトル化

- ◆2分割法（続）
  - 画素列が環状につながっている場合

    最初に任意の点Aを始点とし，その点から最も遠い点Bを終点とする

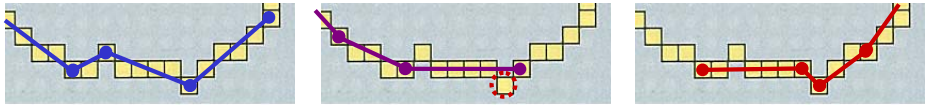


●図9.27 環状になった画素列

- 角点（コーナー／corner／corner point）が始点・終点になることが望ましいが，2分割法では保証されない

⇒角点を通らない近似線分が生成され得る

⇒近似線分の数が多くなるかも知れない



理想　　角点を通らない　　線分が多くなる

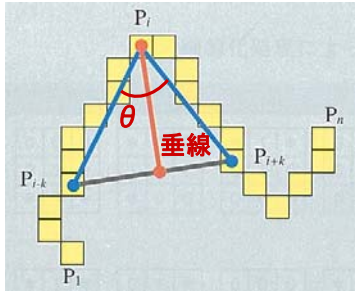
線画像のベクトル化

- ベクトル化アルゴリズム（2）
  - ◆角点（コーナー）検出（corner detection）
    - 積極的に角点を求めて分岐点（始点・終点）とする
    - 角点を始点・終点とする線分近似は，最大誤差を閾値以下に抑えつつ自然なベクトル化を可能にする

（1）画素列の始点P_1から終点P_nに至る途中の任意の点P_iに対して，両側にK画素離れた点P_{i-k}，P_{i+k}と結ぶ2つの線分の角度θが閾値以下なら角点とし，そこで画素列を分割する

（1）’角度θを求める代わりに点P_{i-k}とP_{i+k}を結ぶ線分に点P_iから下ろした垂線の長さが閾値以下かどうかで判定

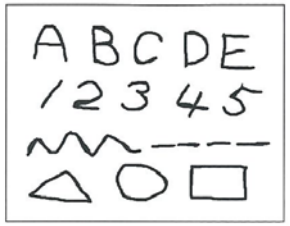
- kの値は応用ごとに決定



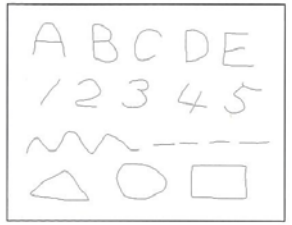
●図9.28 角点を検出し分岐点とする方法

線画像のベクトル化

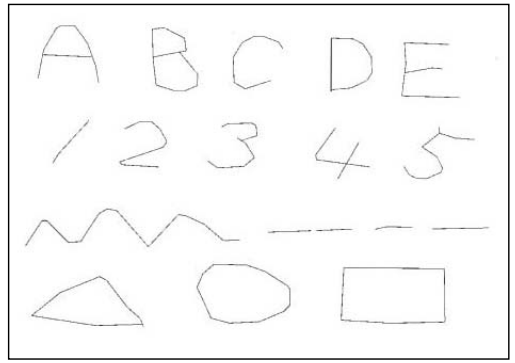
- ベクトル化結果例



2値化（線画）画像



細線化画像



ベクトル化（線分近似）画像

●図9.29 ベクトル化の例

●図9.24 文書画像の細線化処理例