生命情報学特別講義

-デジタル画像と定量化-
その5：領域抽出・ラベリング・細線化

第５回講義
2011年8月3日～4日
伊都新キャンパス

吉澤　信
shin@riken.jp, 非常勤講師
九州大学 大学院
独立行政法人 理化学研究所

よくある画像処理の流れ

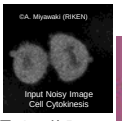
入力：画像データ

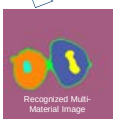
前処理：
e.g. フィルタリング、ノイズ除去、超解像度、多重解像度解析、空間変換等.

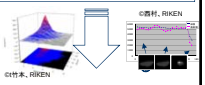
特徴抽出

認識・識別
e.g. 領域抽出

後処理：e.g. 統計・幾何処理





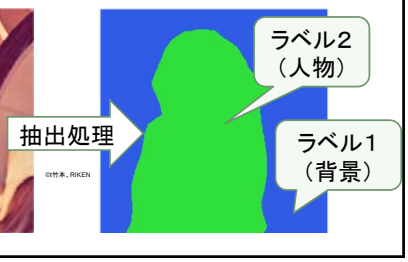


パターン認識では特徴量は形状記述子・画像記述子とも呼ばれる.

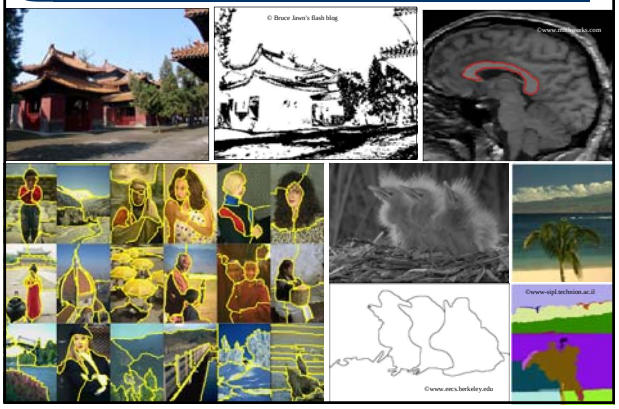
出力：解析結果

領域抽出とは？

- ✓ 領域抽出：画像の領域を分割する処理・対象の領域を切り出して他の領域と区別する事.
- ✓ 画像処理で最も重要な技術.
- ✓ 毎年何百！という新しい方法が提案されている.



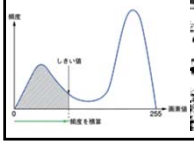
領域抽出の例

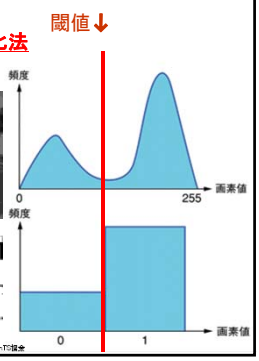


二値化

- ✓ 二値化：画像の画素値を二つに分ける事＝画像を二つの領域に分ける事.
  - 単純閾値、P-タイル法、大津の二値化法（判別分析法）等.

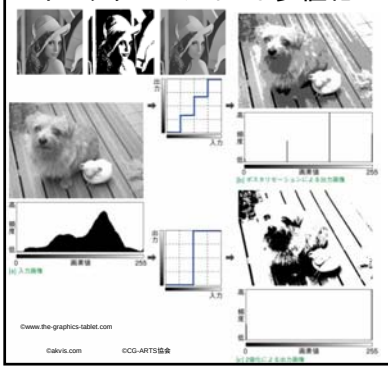
P-タイル法：対象の占める画素数が既知のとき、低いところから頻度値を積算. 予測される画素数付近を閾値とする方法.



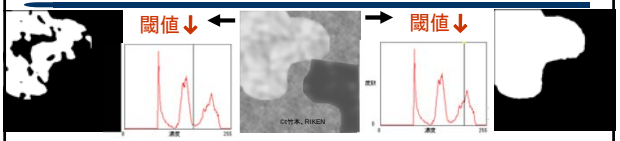


多値化と二値化

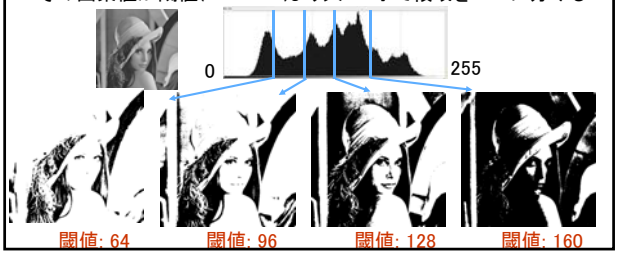
- ✓ ポスタリゼーションは多値化.



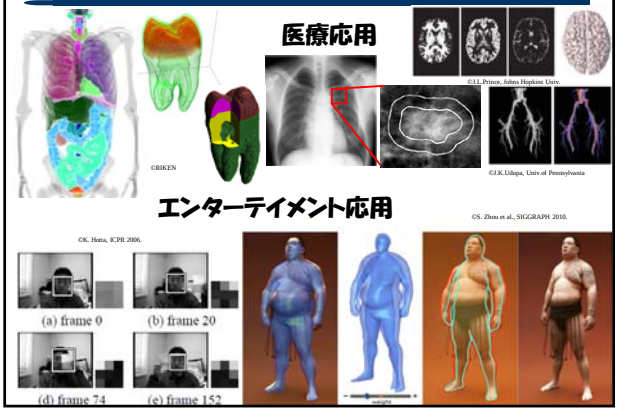
一番簡単な領域抽出：閾値による二値化



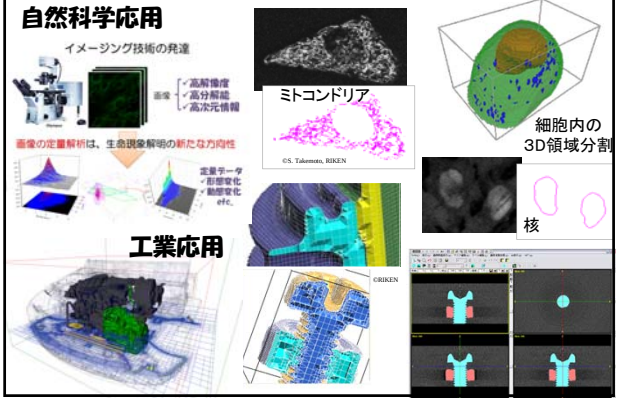
✓ その画素値が閾値(threshold)より大 or 小で領域を二つに分ける.



何の役に立つのか？

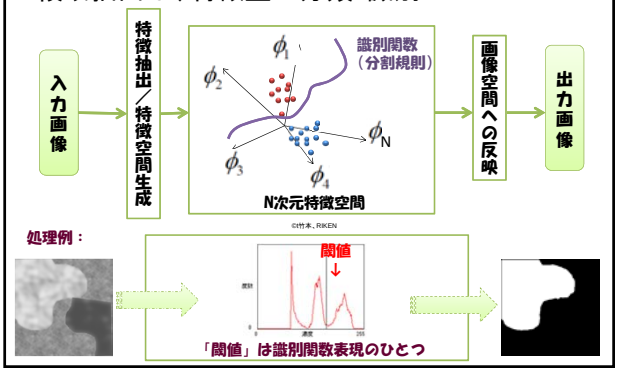


何の役に立つのか？2



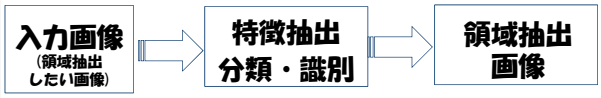
領域抽出処理の流れ

✓ 領域抽出は、特徴量の分類・識別.

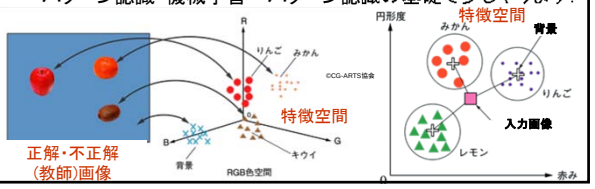


重要：領域抽出法の分類

✓ 教師なし(Unsupervised Segmentation):



✓ 教師あり(Supervised Segmentation):
- パターン認識・機械学習→パターン認識の基礎で少しやります.



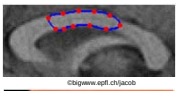
重要：領域抽出法の分類

✓ 教師なし(Unsupervised Segmentation):
- 領域の輝度値や抽出したい形状に関するエネルギー(目的関数)を最小化・最大化する事で特徴量の分布や滑らかさを基準.
- 領域抽出でよく用いられる方法は**大津の二値化法**, Snake (Active Contour), Graph Cuts, Mean Shift, Water Shed (Region Growing)等の方法が有名(目的関数の違いなど沢山の亜種).
- モデルを用いた検出：エッジ抽出、コーナー検出、テンプレートマッチング、線・円：→形状検出でやります.

Shin Yoshizawa: shin@riken.jp

Snake/Active Contour法

- ✓ 曲線と画像のエッジに基づくエネルギー関数の和を最小化する事で曲線を対象に収束させていく方法.
- ✓ エネルギーの種類:
  - 閉曲線の連続性や滑らかさ.
  - 画像のエッジ強度.
  - 閉曲線を縮ませる(曲率).


©bigwww.epfl.ch/jacob

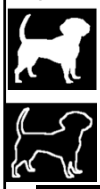
©www.cs.bris.ac.uk

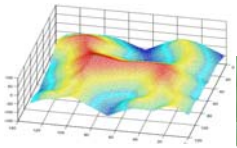
©math.berkeley.edu/~sethian

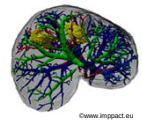
Shin Yoshizawa: shin@riken.jp

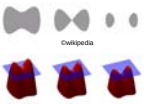
Snake/Active Contour法2

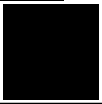
- ✓ Level Set法と呼ばれる方法と組み合わせる事で位相変化に対応し複数オブジェクトの領域抽出が可能.




©www.cim.mcgill.ca/~friggi

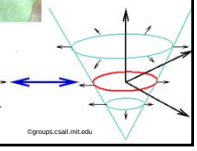

©www.impact.eu


©wikipedia



©math.berkeley.edu/~sethian

©www.math.ucla.edu


©groups.csail.mit.edu

Shin Yoshizawa: shin@riken.jp

Snake/Active Contour法3

- ✓ 3次元曲面への拡張もある.

©A. Sharf et al. EG'06.

Shin Yoshizawa: shin@riken.jp

Snake/Active Contour法4

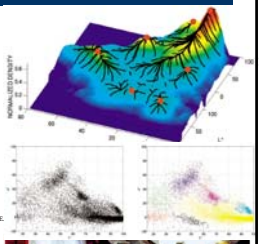
- ✓ 物理方程式の境界面を計算する事でのシミュレーション.

©physbam.stanford.edu/~fedkiw

Shin Yoshizawa: shin@riken.jp

Mean Shift法

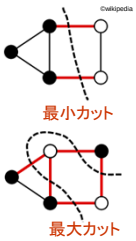
- ✓ 画素の座標値＋色やその他の特徴を組み合わせた特徴空間で(ガウス関数等の)重み付平均を繰り返し適用し、(特徴空間の)同じ場所に集まってきた(収束した)画素を同じ領域とする方法.


©D. Comaniciu and P. Meer, IEEE.

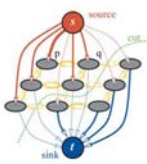
Shin Yoshizawa: shin@riken.jp

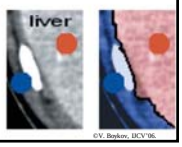
Graph Cuts法

- ✓ 画素の格子や近傍の画素への辺をグラフの辺として画素中心をグラフの頂点とし、エッジ強度等の重みを持ったグラフ構造を分離(カット)する方法.
  - 最小カット(Minimum Cut): 重みの和が最小.
  - 最大カット(Maximum Cut): 重みの和が最大.


©wikipedia
最小カット
最大カット

©T. Ijiri, RIKEN





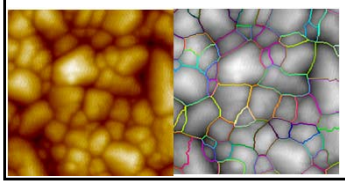
©V. Boykov, ICCV'06.

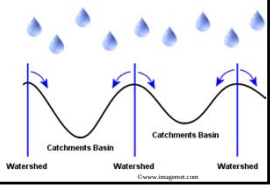
Shin Yoshizawa: shin@riken.jp

Region Growing法

✓ 複数のSeed画素からスタートし領域を拡張していく、拡張のルールはエッジ強度や形状モデルからの距離(例えば領域が平面に近いかどうか)等から構成されるエネルギー関数を最小化する様な近傍画素を随時Seed画素に加えて領域を大きくしていく:

- Watershed法, K-means Clustering, Lloyd Partitioning,重心ボロノイ図, etc.



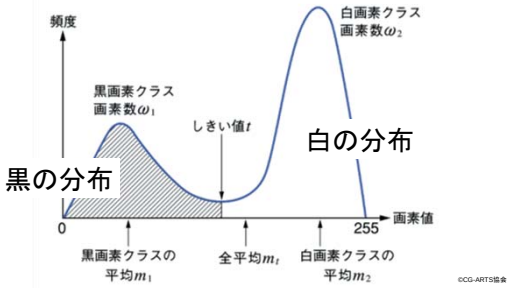


Shin Yoshizawa: shin@riken.jp

重要:大津の二値化法(判別分析)法

✓ 白の分布と黒の分布の「分離度」が大きくなるように閾値を自動的に決める.

✓ 分離度:クラス間分散÷クラス内分散.

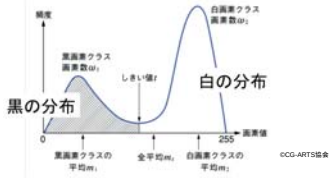


Shin Yoshizawa: shin@riken.jp

閾値によるクラス

✓ 閾値によるクラス分け＝閾値による二値化:

✓ 全体とそれぞれのクラスの平均と偏差:



平均

$$m = \frac{1}{\omega}\sum_{i=1}^{\omega} x_i$$

分散

$$\sigma^2 = \frac{1}{\omega}\sum_{i=1}^{\omega}(x_i - m)^2$$

$m_t,\ \sigma_t^2$ 全体の平均と分散

$m_1,\ \sigma_1^2,\ \omega_1$ 黒画素クラスの平均と分散, 画素数

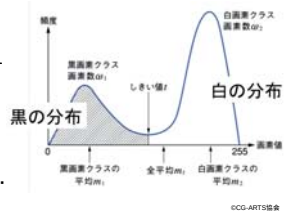
$m_2,\ \sigma_2^2,\ \omega_2$ 白画素クラスの平均と分散, 画素数

Shin Yoshizawa: shin@riken.jp

重要:クラス内分散とクラス間分散

✓ クラス内分散:クラスの散らばりの大きさ.

$$\sigma_w^2 = \frac{\omega_1\sigma_1^2 + \omega_2\sigma_2^2}{\omega_1 + \omega_2}$$



✓ クラス間分散:
二クラス間の散らばり度合.

$$\sigma_b^2 = \frac{\omega_1(m_1 - m_t)^2 + \omega_2(m_2 - m_t)^2}{\omega_1 + \omega_2}$$

$$= \frac{\omega_1\omega_2(m_1 - m_2)^2}{(\omega_1 + \omega_2)^2}$$

Shin Yoshizawa: shin@riken.jp

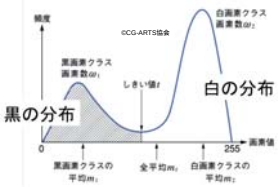
重要:分離度

✓ 分離度:クラス間分散÷クラス内分散.

$$\text{分離度} = \frac{\sigma_b^2}{\sigma_w^2}$$

クラス間分散

クラス内分散



- 二つのクラスができるだけ分離しているためには,
- クラス内分散＝クラスの分布の広がり
  →なるべく小さいほうがよい
- クラス間分散＝クラスの隔たり
  →なるべく大きいほうがよい
- 分離度＝クラス間分散÷クラス内分散を最大にする.

Shin Yoshizawa: shin@riken.jp

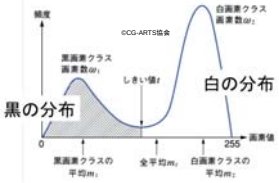
分離度2

✓ 分離度:クラス間分散÷クラス内分散.

$$\text{分離度} = \frac{\sigma_b^2}{\sigma_w^2}$$

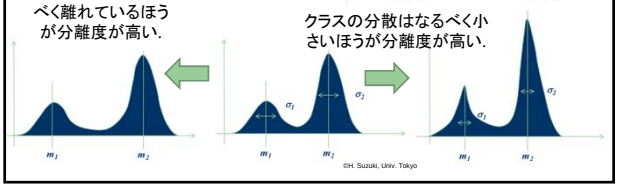
クラス間分散

クラス内分散



クラスの平均はなるべく離れているほうが分離度が高い.

クラスの分散はなるべく小さいほうが分離度が高い.



Shin Yoshizawa: shin@riken.jp

分離度3

✓ 分離度の最大化.

$$\text{分離度} = \frac{\sigma_b^2}{\sigma_w^2}$$ クラス間分散 / クラス内分散

$$= \frac{\sigma_b^2}{\sigma_t^2 - \sigma_b^2} \quad (\because \sigma_t^2 = \sigma_w^2 + \sigma_b^2 \leftarrow \text{証明してみよう})$$

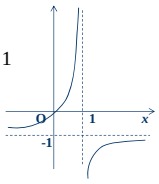
$x = \frac{\sigma_b^2}{\sigma_t^2}$とおくと

$$= \frac{x}{1-x} \quad (0 < x < 1) \qquad x = \frac{\sigma_b^2}{\sigma_t^2} < \frac{\sigma_b^2 + \sigma_w^2}{\sigma_t^2} = \frac{\sigma_t^2}{\sigma_t^2} = 1$$

グラフから、この値は$0 < x < 1$で単調増加。

σ_t^2は、閾値の選び方によらないので、

σ_b^2が最大くなるように閾値を選べばよい。



©H. Suzuki, Univ. Tokyo

Shin Yoshizawa: shin@riken.jp

重要：大津の方法アルゴリズム

1. 画像からヒストグラムを作成. ビンの数をNとする.
2. 閾値が0のときのクラス間分散を計算しその値をSmax, そのときの閾値をTmaxとする.
3. for(i=1;i<N;i++){
   1. 閾値がiのときのクラス間分散を計算しSとする.

$$\sigma_b^2 = \frac{\omega_1(m_1 - m_t)^2 + \omega_2(m_2 - m_t)^2}{\omega_1 + \omega_2} = \frac{\omega_1\omega_2(m_1 - m_2)^2}{(\omega_1 + \omega_2)^2}$$

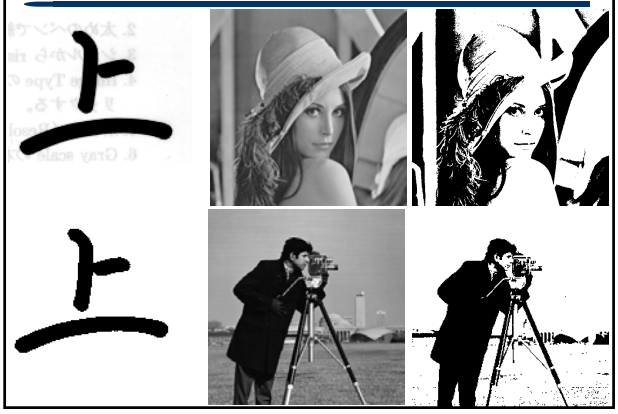
平均 $m = \frac{1}{\omega}\sum_{i=1}^{\omega} x_i$

分散 $\sigma^2 = \frac{1}{\omega}\sum_{i=1}^{\omega}(x_i - m)^2$

   2. もしもS>SmaxならばSmax=S, Tmax=Tとする.
4. }
5. Tmaxが大津の閾値となる.

Shin Yoshizawa: shin@riken.jp

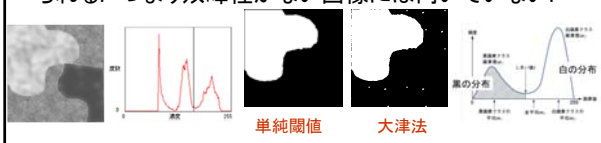
大津法の結果



Shin Yoshizawa: shin@riken.jp

大津の方法の問題点

✓ ヒストグラムが双峰性を持つ場合に非常に良い結果が得られる. つまり双峰性がない画像には向いていない.



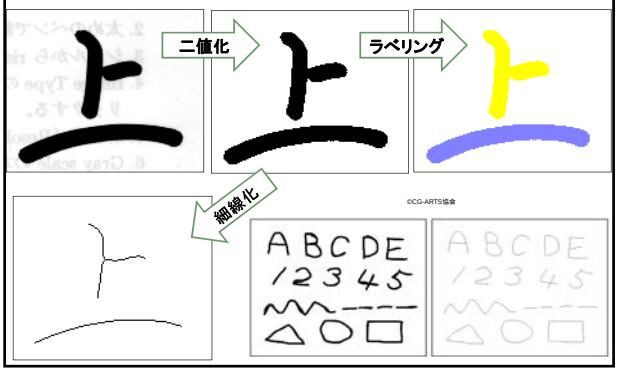
単純閾値　大津法

✓ 画像全体のヒストグラムを使っているため背景の明るさ変化に弱い.

Shin Yoshizawa: shin@riken.jp

細線化＆ラベリング

✓ 二値化後の典型的処理として細線化とラベリングがある.



©CG-ARTS協会

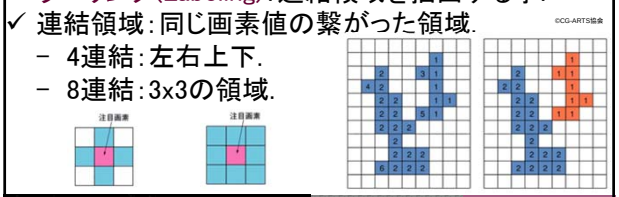
Shin Yoshizawa: shin@riken.jp

ラベリングとは？

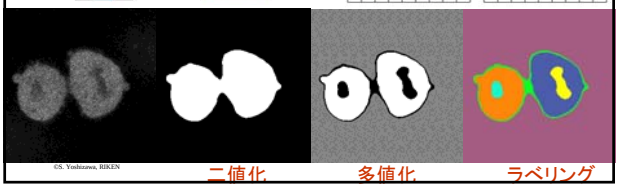
✓ ラベリング(Labeling)：連結領域を抽出する事.

✓ 連結領域：同じ画素値の繋がった領域.

- 4連結：左右上下.
- 8連結：3x3の領域.



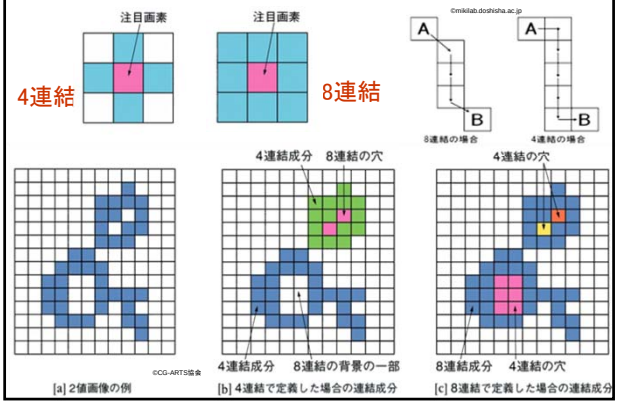
©CG-ARTS協会



©S. Yoshizawa, RIKEN

二値化　多値化　ラベリング

4連結 VS 8連結



ラベリングのアルゴリズム(再帰)

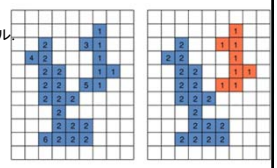
✓ 再帰関数で書くと超簡単！

©CG-ARTS協会

多値へも簡単に拡張可能.

```
for(y=0;y<sy;y++)
  for(x=0;x<sx;x++)
    if(bin[y][x] == BLACK){
      Recursive(x,y);
      label += L_BASE;
    }
```

bin[i][j]:黒 or 白.
out[i][j]:出力のラベル.
sx,sy:画像サイズ.



1. main関数の中で黒なら再帰関数を呼び出す.
2. 再帰が帰ってきたらラベルを変えて繰り返し.

```
int Recursive(int x,int y){
  if(out[y][x]!=BLACK)||(x>=sx)||(x<0)||(y>=sy)||(y<0))return 1;
  out[y][x] = label;
  bin[y][x] = WHITE;
  return (Recursive((x),(y+1))+Recursive((x),(y-1))+
          Recursive((x+1),(y+1))+Recursive((x+1),(y-1))+
          Recursive((x-1),(y+1))+Recursive((x-1),(y-1))+
          Recursive((x+1),(y)+Recursive((x-1),(y)));
}
```

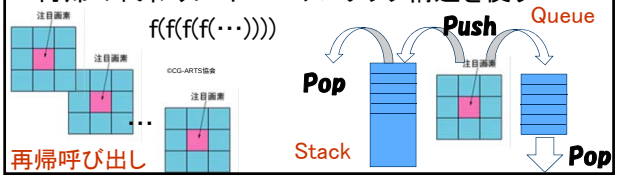
1. 再帰関数で8連結の周りを呼び出しながら同じ値ならラベルを付けていく.
2. 同時に黒→白.

重要:アルゴリズム(キュー or スタック)

✓ 残念ながら再帰関数は入れ子(階層的な呼び出し)の回数がOS毎に制限(高々10-20程度).

✓ 定理:再帰アルゴリズムは繰り返しアルゴリズムに常に書き換える事が可能.

✓ 再帰の代わりにキューやスタック構造を使う.



ラベリング(キュー or スタック)2

```
int doLabeling(Image *in){
  int i,j;
  sx=in->sx;
  sy=in->sy;

  bnum = sx*sy;
  bool *mycheck = new bool[bnum];
  for(i=0;i<bnum;i++)mycheck[i]=true;
  int index=0;
  No0;
  LabelInt= LABELBASEINT;
  for(i=0;i<sy;i++){
    for(j=0;j<sx;j++){
      if(mycheck[INDEX2D(i,j)]){
        Labelset(in,i,j,LabelInt,mycheck);
        LabelInt++;
        N++;
      }
    }
  }
  delete [] mycheck;mycheck=NULL;
  return N;
}
```

```
//8 connected in 2D
int Labelset(Image *in,int sy,int sx,int dLabel,bool *mycheck){

  std::queue(int> pqueue;
  mycheck[INDEX2D(sy,sx)]=false;
  int currentLabel=in->img[sy][sx];
  in->img[sy][sx]=dLabel;
  int i,j,k;
  Put8(sy,sx,in,currentLabel,&pqueue,mycheck);

  int popID=-1;

  while(pqueue.empty() == false){

    popID = pqueue.front();
    pqueue.pop();

    j = popID % sx;
    i = (popID-j) / sx;
    if((i>=0&&i<sy)&&
       (j>=0&&j<sx)){
      in->img[i][j]=dLabel;
      Put8(i,j,in,currentLabel,&pqueue,mycheck);
    }else{
      printf("Error ! impossible !! %d %d\n",i,j);
    }
  }
  return 0;
}
```

初期Push

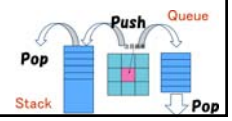
Popのループ

再帰のmainとほぼ同じ.

Put関数

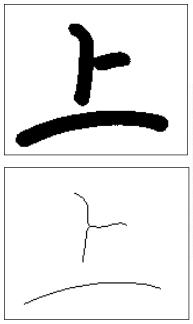
```
void Put8(int sy,int sx,Image *in,int currentLabel,std::queue(int> *pqueue,bool *mycheck){
  int i,j;
  for(i=sy-1;i<=sy+1;i++)
    for(j=sx-1;j<=sx+1;j++)
    {
      if((i>=0&&i<sy)&&
         (j>=0&&j<sx)){
        if(mycheck[INDEX2D(i,j)]&&in->img[i][j]==currentLabel){
          mycheck[INDEX2D(i,j)]=false;
          (*pqueue).push(INDEX2D(i,j));
        }
      }
    }
}
```

8方向へPush.

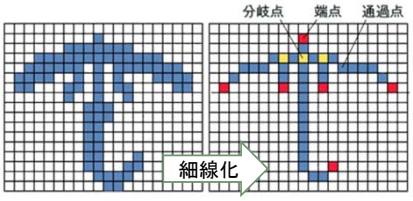


細線化(thinning, 骨格化:skeletonization)

✓ 領域抽出後(二値化)に領域を線状に簡略化する事、ただし通常は入力の二値画像と**同位相**の形状.



✓ 出来るだけ中心に細く、端点でない境界画素を削除していく.



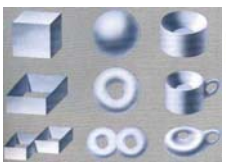
©CG-ARTS協会

文字認識等で非常によく用いられる！

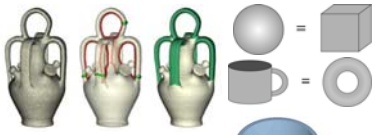
細線化その2

✓ 同位相:連続変形で変換可能である事:
- ✓ 球、平面、トーラス等はそれぞれ異なる位相.
- ✓ 穴(境界)の数、ハンドル(トーラス)の数等で分類.

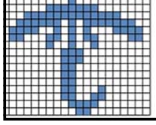
©skullsinthestars.com



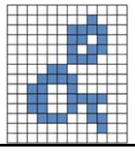
©danilnagy.wordpress.com



©T. Day et al., SIGGRAPH'08.



←異なる位相→



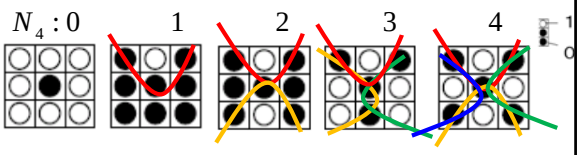
©CG-ARTS協会

©Wikipedia

↑のコップとトーラスは同位相

連結数

✓ 連結数：境界線追跡をしたとき、その画素を通過する回数：消去で連結数が変わらない＝同位相.



$$N_4 = \sum_{i \in C} (f(x_i) - f(x_i) f(x_{i+1}) f(x_{i+2}))$$

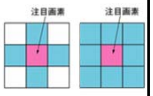
©CG-ARTS協会

連結数	画素の特徴
0	孤立点または内部点
1	端点または境界点
2	連結点
3	分岐点
4	交差点

X4	X3	X2
X5	X0	X1
X6	X7	X8

$C = \{1,3,5,7\}, \quad f = \begin{cases} 1 \\ 0 \end{cases}$

$\overline{f} = 1 - f, \quad x_9 = x_1.$

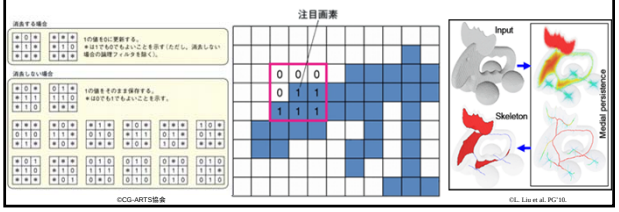


$$N_8 = \sum_{i \in C} (\overline{f}(x_i) - \overline{f}(x_i) \overline{f}(x_{i+1}) \overline{f}(x_{i+2}))$$

4連結 8連結

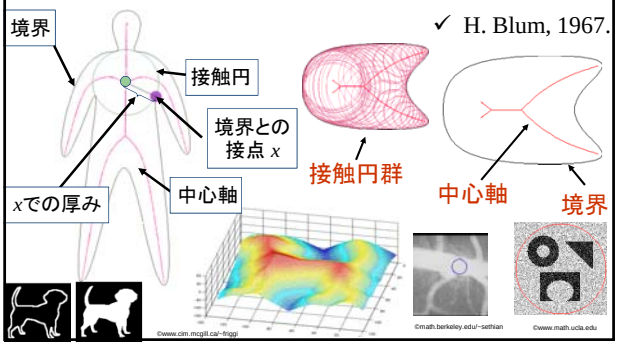
細線化その3

✓ 中心軸(Medial Axis)の近似である事が多い.
✓ 細線化後は線分の幾何特徴(長さや円形度等)を計算.
✓ 様々な方法：境界・連結数を変えない・端点を消去.
  ✓ テンプレートを用いた繰り返し法：Stentiford法、Hilditch法(連結数を使う)、Zhang-Suen法.
  ✓ 中心軸を用いる方法、etc.



中心軸(Medial Axis)

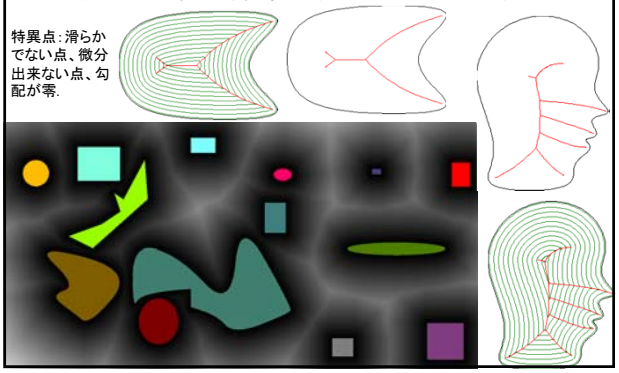
✓定義: 接触円の中心の軌跡.
✓接触円：二点以上で境界に接している境界内の円.
✓ H. Blum, 1967.



中心軸と距離場

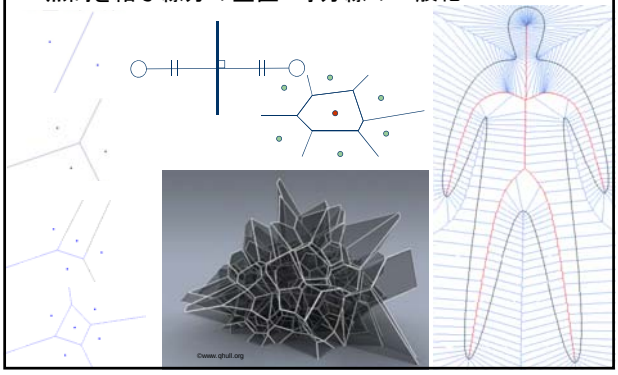
✓中心軸は距離場の等高線が特異点となる点の集合.

特異点：滑らかでない点、微分出来ない点、勾配が零.



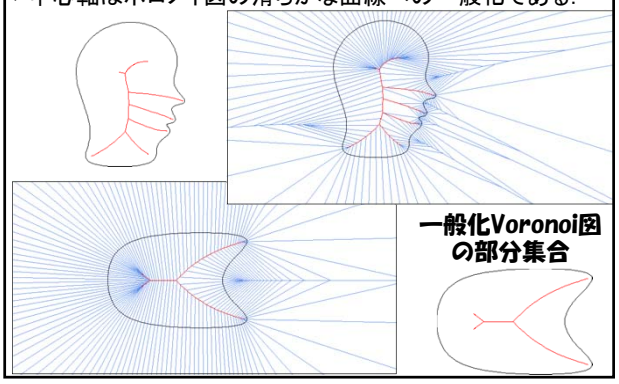
ボロノイ図(Voronoi Diagram)

✓2点間を結ぶ線分の垂直2等分線の一般化.



ボロノイ図と中心軸

✓中心軸はボロノイ図の滑らかな曲線への一般化である.



一般化Voronoi図の部分集合

Shin Yoshizawa: shin@riken.jp

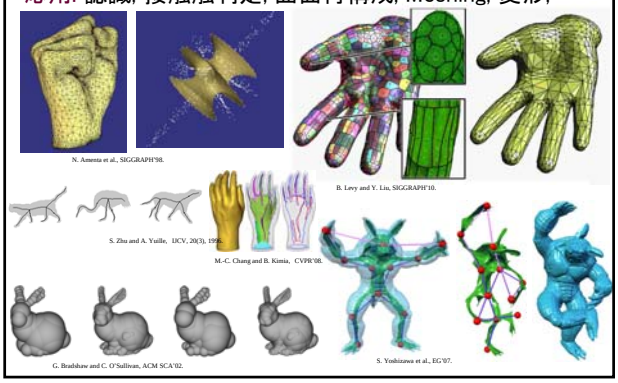
3D中心軸

✓多次元の中心軸もあり、CGやCAD等で応用されている.

Shin Yoshizawa: shin@riken.jp

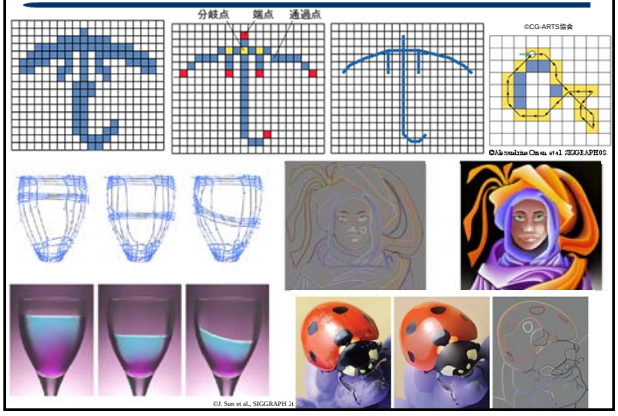
中心軸の応用

✓応用: 認識, 接触触判定, 曲面再構成, Meshing, 変形, …



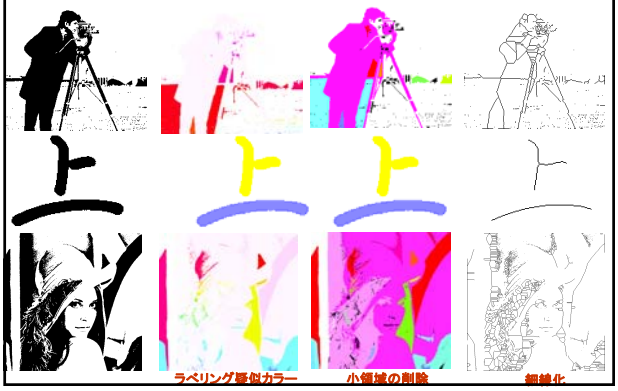
Shin Yoshizawa: shin@riken.jp

細線化の応用例：ベクトル化



Shin Yoshizawa: shin@riken.jp

ラベリング・細線化結果



Shin Yoshizawa: shin@riken.jp

まとめ

✓ 領域抽出は画像処理で最も重要で最も研究・開発が進んでいる＆盛んな分野.
- 教師なし: **大津の二値化法**, Snake (Active Contour), Graph Cuts, Mean Shift, Water Shed (Region Growing), モデルベースの方法等.
- 教師あり: パターン認識・機械学習.

✓ ラベリング＝連結領域の抽出: 4・8連結.

✓ 細線化=Skeletonization/中心軸抽出：
- 境界・連結数を変えない・端点を消去.
- テンプレートを用いた繰り返し法：
Stentiford法、Hilditch法(連結数を使う)、Zhang-Suen法.
- 中心軸は一般化されたボロノイ図.