情報学群実験第 4i レポート

第 5 回

画像処理における画像成分の扱い

グループ 6

学籍番号 1190361

早川 晋矢

平成 29 年 7 月 26 日

1　p-タイル法

```
%p-タイル法
%読み込む画像の切り取る領域を 58%と仮定

%-----初期化処理-----
clear;
%-----画像読み出し-----
img = imread('usagi01.png');          %画像の読み込み
[y,x,z] = size(img);                  %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----RGB 値の取り出し-----
r = double(img(:,:,1));               %R 値取得
g = double(img(:,:,2));               %G 値取得
b = double(img(:,:,3));               %B 値取得
%-----グレースケール化-----
gray = 0.3*r+0.59*g+0.11*b;           %グレイスケール化
%-----降順ソート-----
sdata = sort(gray(:),'descend');      %降順にソート
%-----閾値設定-----
p = 0.58                              %p 値を 58%と指定する
t = sdata(round((x*y)*p));            %39150 番目の画素値を閾値にする
%-----2 値化処理-----
two_color = zeros(y,x);               %ゼロ配列を作成
two_color(gray>=t) = 255;             %閾値以下の画素値に 255(白) を格納
%-----画像表示-----
figure(4);
imshow(two_color);
```

2　モード法

```
%モード法

%-----初期化処理-----
clear;
%-----画像読み出し-----
img = imread('tutiusa.jpg');          %画像の読み込み
[y,x,z] = size(img);                  %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----RGB 値の取り出し-----
r = double(img(:,:,1));               %R 値取得
g = double(img(:,:,2));               %G 値取得
b = double(img(:,:,3));               %B 値取得
%-----グレースケール化-----
gray = 0.3*r+0.59*g+0.11*b;           %グレイスケール化
imwrite(uint8(gray),'guretuti.png');
```

```
%-----閾値設定-----
t = 140;
%-----2 値化処理-----
two_color = zeros(y,x);              %ゼロ配列を作成
two_color(gray<=t) = 255;            %閾値以下の画素値に 255(白) を格納
%-----画像表示-----
figure(4);
imshow(two_color);
imwrite(two_color,'mode.png');
```

3 判別分析法

```
%判別分析法

%-----初期化処理-----
clear;
%-----画像読み出し-----
img = imread('tutiusa.jpg');         %画像の読み込み
[y,x,z] = size(img);                 %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----RGB 値の取り出し-----
r = double(img(:,:,1));              %R 値取得
g = double(img(:,:,2));              %G 値取得
b = double(img(:,:,3));              %B 値取得
%-----グレースケール化-----
gray = 0.3*r+0.59*g+0.11*b;          %グレイスケール化
%-----ヒストグラム化------
gray8 = uint8(gray);                 %画素値を 0 から 255 の整数に
for k = 0:255                        %画素値を 0 から 255 まで探索
    data = (gray8 == k);             %探索値と同じ画素値を探索
    data = sum(sum(data));           %見つけた画素値の個数を数える
    histData(k+1) = data;            %配列に格納する
end
%-----閾値設定-----
max_t = 0;                           %クラス間分散が最大となる閾値の保存用
max_val = 0;                         %最大となるクラス間分散の保存用
for t = 0:255                        %閾値を 0 から 255 まで探索
    %---初期値設定---
    w1 = 0;                          %黒側クラスの画素数
    w2 = 0;                          %白側クラスの画素数
    sum1 = 0;                        %黒側クラス合計値
    sum2 = 0;                        %白側クラス合計値
    m1 = 0;                          %黒側クラスの平均
    m2 = 0;                          %白側クラスの平均
```

```
    %---黒側クラスの画素数，画素値---
    for k = 0:t                          %閾値までが黒側クラス
        w1 = w1 + histData(t+1);         %画素数を足し合わせる
        sum1 = sum1 + k*histData(t+1); %画素値を足し合わせる
    end
    %---白側クラスの画素数，画素値---
    for k = t:255                        %閾値以降が白側クラス
        w2 = w2 + histData(t+1);         %画素数を足し合わせる
        sum2 = sum2 + k*histData(t+1); %画素値を足し合わせる
    end
    %---ゼロ除算を防ぐ---
    if (w1==0 | w2==0)
        continue;                        %次の閾値判定へ
    end
    %---クラス別の平均値の算出---
    m1 = sum1/w1;                        %黒側クラスの平均算出
    m2 = sum2/w2;                        %白側クラスの平均算出
    %---クラス間の分散を算出---
    result = w1*w2*(m1-m2)*(m1-m2) / ((w1+w2)*(w1+w2)); %式 (9.2)
    %---クラス間分散値の更新---
    if (max_val < result)
        max_val = result;
        max_t = t;
    end
end
%-----2 値化処理-----
two_color = zeros(y,x);                  %ゼロ配列を作成
two_color(gray>=max_t) = 255;            %閾値以下の画素値に 255(白) を格納
%-----画像表示-----
figure(5);
imshow(two_color);
```

4　輪郭追跡

```
%配列格納順番
%searchX，searchY，entryCode はこの順番に基づいている
%[1,2,3
% 8,-,4
% 7,6,5]
%進入方向番号
%searchX，searchY，entryCode はこの順番に基づいている
%[0,1,2
% 7,-,3
```

```
% 6,5,4]

%-----初期化処理-----
clear;
%-----画像読み出し-----
im = imread('two_mohu.png');          %画像の読み込み
[y,x,z] = size(im);                   %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----画像周辺に画素追加-----
img = uint8(ones(y+2,x+2).*255);      %画像より一回り大きい白 (255) 配列作成
img(2:y+1,2:x+1) = im;                %読み込んだ画像に適用
%-----初期値設定-----
searchValue = 0;                      %探索で探すべき値の設定
searchX = [-1,0,1,1,1,0,-1,-1];       %3*3 探索の際の横座標調整用
searchY = [-1,-1,-1,0,1,1,1,0];       %3*3 探索の際の縦座標調整用
entryCode = [5,6,7,0,1,2,3,4];        %進入方向から座標算出用
tracked = zeros(y+2,x+2);             %追跡済み座標格納用
entryDirection = zeros(y+2,x+2);      %進入方向保存用 (更新は 1 本の輪郭追跡ごとの初回)
check = zeros(y+2,x+2);               %追跡済み座標格納用 (1 本の輪郭追跡ごとにリセット)
boxSize = 8;                          %探索を行う範囲のサイズ
modVal = boxSize;                     %余りを求める際に使用
loopbox = boxSize-1;                  %画素周辺の探索の繰り返し回数
exit = 0;                             %処理の終了用スイッチ
%-----目標画素値の探索-----
for m=2:y-1
    for n=2:x-1
    %-----目標画素値であり，左画素が反する画素であり，未追跡画素を見つけるまでループ-----
    if (img(m,n) ~= searchValue | img(m,n-1) == searchValue | tracked(m,n) ~= 0);
        continue;
    end
    %-----輪郭追跡で称する変数の初期化-----
    nextEntry = 1;                    %現在の画素にどの方向から進入したか
    entryDirection(m,n) = nextEntry;%右から進入したことを格納
    tracked(m,n) = 1;                 %探索済みと更新
    sx = n;                           %x 座標のコピー
    sy = m;                           %y 座標のコピー
    exit = 0;                         %処理の終了用スイッチ
    check = zeros(y+2,x+2);           %追跡済み座標格納用 (1 本の輪郭追跡ごとにリセット)
    %-----輪郭追跡-----
    while (exit==0)                   %終了条件を満たすまで続ける
        for k=0:loopbox               %画素周辺の探索
            %-----探索座標の算出-----
            boxX = sx+searchX(mod(k+nextEntry,modVal)+1);
            boxY = sy+searchY(mod(k+nextEntry,modVal)+1);
            %-----連続画素の探索-----
```

```
            if img(boxY,boxX) == searchValue;
                %-----現在の輪郭追跡で探索済みか判定-----
                if check(boxY,boxX) == 0;
                    %-----未探索なら進入方向、探索判定、次の進入方向の更新-----
                    tracked(boxY,boxX) = 1;
                    check(boxY,boxX) = 1;
                    entryDirection(boxY,boxX) = entryCode(mod(k+nextEntry,modVal)+1);
                    nextEntry = entryCode(mod(k+nextEntry,modVal)+1);
                elseif entryDirection(boxY,boxX) == entryCode(mod(k+nextEntry,modVal)+1)
                    %-----一周できたことを確認したら終了-----
                    exit=1;
                else
                    %-----次の進入方向のみ更新-----
                    nextEntry = entryCode(mod(k+nextEntry,modVal)+1);
                end
                %-----座標を更新して次の探索へ-----
                sx = boxX;
                sy = boxY;
                break;
            end
            %-----1 ドットが検出された際はここで終了する-----
            if(k ==loopbox)
                exit=1;
            end
        end
    end
end
end
%-----画像の出力-----
figure(5);
result=ones(y+2,x+2);
result=(result-tracked)*255;
resultImg=result(2:y-1,2:x-1);
imshow(resultImg);
imwrite(resultImg,'track2.png');
```

5 収縮・膨張処理

```
%-----初期化処理-----
clear;
%-----画像読み出し-----
im = imread('inp.png');          %画像の読み込み
[y,x,z] = size(im);              %画像のサイズ (y=縦座標,x=横座標,z=RGB)
```

```
  %-----オープニング処理-----
  im = erosion(im);                %収縮処理
  im = dilation(im);               %膨張処理
  %-----クロージング処理-----
  im = dilation(im);               %膨張処理
  im = erosion(im);                %収縮処理
  %-----画像表示-----
  figure(5);
  imshow(im);
  imwrite(im,'opclo.png');

%----------収縮用関数 erosion---------
  function resultImg = erosion(im)
  %-----画像読み出し-----
  %im = imread(name);                    %画像の読み込み
  [y,x,z] = size(im);                    %画像のサイズ (y=縦座標,x=横座標,z=RGB)
  %-----画像周辺に画素追加-----
  img = uint8(ones(y+2,x+2).*255);       %画像より一回り大きい白 (255) 配列作成
  img(2:y+1,2:x+1) = im;                 %読み込んだ画像に適用
  %result = uint8(ones(y+2,x+2).*255);%結果を格納する配列を作成する
  result = uint8(zeros(y+2,x+2));
  boxSize = 8;                           %探索を行う範囲のサイズ
  %-----初期値設定-----
  searchValue = 0;                       %探索で探すべき値の設定
  checkValue = 255;                      %周囲で確認する値の設定
  searchX = [-1,0,1,1,1,0,-1,-1];        %3*3 探索の際の横座標調整用
  searchY = [-1,-1,-1,0,1,1,1,0];        %3*3 探索の際の縦座標調整用
  %-----目標画素値の探索-----
  for m=2:y-1
      for n=2:x-1
          %-----探索画素が見つかるまでループ-----
          if (img(m,n) ~= searchValue)
              continue;
          end
          %-----周辺画素探索用変数の初期化-----
          sx = n;                                  %x 座標のコピー
          sy = m;                                  %y 座標のコピー
          for k=1:boxSize                          %画素周辺の探索
              boxX = n+searchX(k);                 %座標の算出
              boxY = m+searchY(k);                 %座標の算出
              if img(boxY,boxX) == checkValue; %探索する値が見つかれば
                  result(m,n) = checkValue;    %更新する値を代入
                  break;
              end
```

```
            end
        end
    end
    %-----画像の出力-----
    %figure(5);
    result=uint8(result+img);
    resultImg=result(2:y-1,2:x-1);
    %imshow(resultImg);

%----------膨張用関数 dilation---------

    %function resultImg = dilation(name)
    function resultImg = dilation(im)
    %-----画像読み出し-----
    %im = imread(name);                  %画像の読み込み
    [y,x,z] = size(im);                 %画像のサイズ (y=縦座標,x=横座標,z=RGB)
    %-----画像周辺に画素追加-----
    img = uint8(ones(y+2,x+2).*255);    %画像より一回り大きい白 (255) 配列作成
    img(2:y+1,2:x+1) = im;              %読み込んだ画像に適用
    result = uint8(ones(y+2,x+2));      %結果を格納する配列を作成する
    boxSize = 8;                        %探索を行う範囲のサイズ
    %-----初期値設定-----
    searchValue = 255;                  %探索で探すべき値の設定
    checkValue = 0;                     %周囲で確認する値の設定
    searchX = [-1,0,1,1,1,0,-1,-1];     %3*3 探索の際の横座標調整用
    searchY = [-1,-1,-1,0,1,1,1,0];     %3*3 探索の際の縦座標調整用
    %-----目標画素値の探索-----
    for m=2:y-1
        for n=2:x-1
            %-----探索画素が見つかるまでループ-----
            if (img(m,n) ~= searchValue)
                continue;
            end
            %-----周辺画素探索用変数の初期化-----
            sx = n;                             %x 座標のコピー
            sy = m;                             %y 座標のコピー
            for k=1:boxSize                     %画素周辺の探索
                boxX = n+searchX(k);            %座標の算出
                boxY = m+searchY(k);            %座標の算出
                if img(boxY,boxX) == checkValue; %探索する値が見つかれば
                    result(m,n) = checkValue;   %更新する値を代入
                    break;
                end
            end
```

```
    end
end
%-----画像の出力-----
%figure(5);
result=uint8(result.*img);
resultImg=result(2:y-1,2:x-1);
%imshow(resultImg);
```

6　ラベリング

```
%-----初期化処理-----
clear;
%-----画像読み出し-----
im = imread('two_mohu.png');                %画像の読み込み
[y,x,z] = size(im);                         %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----画像周辺に画素追加-----
img = uint8(ones(y+2,x+2).*255);            %画像より一回り大きい白 (255) 配列作成
img(2:y+1,2:x+1) = im;                      %読み込んだ画像に適用
%-----初期値設定-----
searchValue = 0;                            %探索で探すべき値の設定
white = 255;
searchX = [-1,0,1,
           -1,0,1,
           -1,0,1];
searchY = [-1,-1,-1,
            0, 0, 0,
            1, 1, 1];
lutSize = x*y;
label = zeros(y+2,x+2);                     %追跡済み座標格納用
lookupTable = [];          %同一連結成分の設定
%lookupTable = updataLUT(lookupTable,7,12)
lutpoint = 1;
nextlabel = 1;
%-----目標画素値の探索-----
c = 'check';
for m=2:y+1
    for n=2:x+1
        %-----探索画素が見つかるまでループ-----
        if img(m,n) ~= searchValue
            continue;
        end
        %-----ラベルの更新-----
        for k = [1,2]%左上と上のラベル確認
```

```
            if label(m+searchY(k),n+searchX(k)) ~= 0
                label(m,n) = label(m+searchY(k),n+searchX(k));
                break;
            end
        end
        if label(m,n) ~= 0%注目ラベルがゼロでなければ
            for k = [2,3,4]%上と右上と左
                if label(m+searchY(k),n+searchX(k)) ~= 0 & label(m+searchY(k),n+searchX(k)) ~= lab
                %-----lookpuTable 更新-----
                lookupTable = updataLUT(lookupTable,label(m,n),label(m+searchY(k),n+searchX(k)));
                end
            end
            if label(m+searchY(3),n+searchX(3)) ~= 0 & img(m+searchY(6),n+searchX(6)) == white & l
                %-----lookpuTable 更新-----
                lookupTable = updataLUT(lookupTable,label(m,n),label(m+searchY(3),n+searchX(3)));
            end
            pretrue = 1;
            for k = [1,2]%左上と上
                if img(m+searchY(k),n+searchX(k)) ~= white
                    pretrue = 0;
                    break;
                end
            end
            if pretrue == 1 & label(m+searchY(4),n+searchX(4)) ~= 0
            %lookpuTable 更新---------------------------------
            lookupTable = updataLUT(lookupTable,label(m,n),label(m+searchY(4),n+searchX(4)));
            end
        end
        pretrue = 1;
        for k = [1,2]%左上と上と左
            if img(m+searchY(k),n+searchX(k)) ~= white
                pretrue = 0;
                break;
            end
        end
        if pretrue == 1
            label(m,n) = nextlabel;
            nextlabel = 1+nextlabel;
        end
    end
end
%-----ルックアップテーブル更新作業-----
renVal = zeros(nextlabel-1,nextlabel-1);
for k = 1 : nextlabel-1
```

```
    rvc = 1;
    search = find(lookupTable == k);
    for l = search'; %'
        if l <= length(lookupTable)
            l = l + length(lookupTable);
        else
            l = l - length(lookupTable);
        end
        save = 1;
        for m = 1:rvc-1
            if renVal(m) == 0
                break;
            end
            if renVal(k,m) == lookupTable(l)
                save = 0;
                break;
            end
        end
        if save == 1
            renVal(k,rvc) = lookupTable(l);
            rvc = rvc + 1;
        end
    end
end

lookupTable2 = zeros(nextlabel-1,nextlabel-1);
valSet = [1:nextlabel-1];
exit = 1;
while true
    for k = 1 : nextlabel-1
            min = valSet(k);
        for m = 1 : nextlabel-1
            if renVal(k,m) == 0
                break;
            end
            if renVal(k,m) < min
                min = renVal(k,m);
                exit = 0;
            end
            if valSet(renVal(k,m)) < min
                min = valSet(renVal(k,m));
                exit = 0;
            end
        end
```

```
        if min ~= valSet(k)
            valSet(k) = min;
        end
    end
    if exit == 1
        break
    end
    exit = 1;
end

type = unique(valSet);
for m = 1:length(type)
    for n = 1:length(valSet)
        if valSet(n) == type(m)
            valSet(n) = m;
        end
    end
end

%
for m=2:y+1
    for n=2:x+1
        if label(m,n) ~= 0
            label(m,n) = valSet(label(m,n));
        end
    end
end

maxLabel = max(valSet);

result = zeros(y,x,3);
for m=2:y+1
    for n=2:x+1
        if label(m,n) ~= 0
            result(m,n,1) = label(m,n)/maxLabel;
            result(m,n,2) = 1;
            result(m,n,3) = 1;
        else

            result(m,n,3) = 1;
        end
    end
end
```

```
resultImg = hsv2rgb(result);
imshow(resultImg);
imwrite(resultImg,'labelinged.png');

function lookupTable = updataLUT(lookupTable,valA,valB)
[ty, tx] = size(lookupTable);

if ty == 0
    lookupTable = [lookupTable;valA, valB];
else
    for t = 1:ty
        if lookupTable(t,1) == valA
            if lookupTable(t,2) == valB
                break;
            end
        elseif lookupTable(t,2) == valA
            if lookupTable(t,1) == valB
                break;
            end
        end
        if t == ty
            lookupTable = [lookupTable;valA, valB];
        end
    end
end
```

7 細線化処理

```
%[P9][P2][P3]
%[P8][P1][P4]
%[P7][P6][P5]
%

%-----初期化処理-----
clear;
%-----画像読み出し-----
im = imread('two_mohu.png');              %画像の読み込み
[y,x,z] = size(im);                       %画像のサイズ (y=縦座標,x=横座標,z=RGB)
%-----画像周辺に画素追加-----
img = uint8(ones(y+2,x+2).*255);          %画像より一回り大きい白 (255) 配列作成
img(2:y+1,2:x+1) = im;                    %読み込んだ画像に適用
searchValue = 0;
white = 255;
```

```
black = 0;
searchX = [0,0,1,1,1,0,-1,-1,-1];
searchY = [0,-1,-1,0,1,1,1,-1,-1];
searchData = img;
changeData = img;
changed = 0;
while true
    %-----ステップ 1-----
    for m=2:y+1
        for n=2:x+1
            %-----探索画素が見つかるまでループ-----
            if searchData(m,n) ~= black
                continue;
            end
            if f1(searchData,m,n) ~= 1
                continue;
            end
            if f2(searchData,m,n) < 2 | f2(searchData,m,n) > 6
                continue;
            end
            if searchData(m+searchY(2),n+searchX(2))==black & searchData(m+searchY(4),n+searchX(4)
                continue;
            end
            if searchData(m+searchY(4),n+searchX(4))==black & searchData(m+searchY(6),n+searchX(6)
                continue;
            end
            changeData(m,n) = white;
            changed = 1;
        end
    end
    searchData = changeData;

    %-----ステップ 2-----
    for m=2:y+1
        for n=2:x+1
            %-----探索画素が見つかるまでループ-----
            if searchData(m,n) ~= black
                continue;
            end
            if f1(searchData,m,n) ~= 1
                continue;
            end
            if f2(searchData,m,n) < 2 | f2(searchData,m,n) > 6
                continue;
```

```
                end
                if searchData(m+searchY(2),n+searchX(2))==black & searchData(m+searchY(4),n+searchX(4)
                    continue;
                end
                if searchData(m+searchY(2),n+searchX(2))==black & searchData(m+searchY(6),n+searchX(6)
                    continue;
                end
                changeData(m,n) = white;
                changed = 1;
            end
        end
        searchData = changeData;
        if changed == 0;
            break;
        end
        changed = 0;
    end

    imshow(searchData);
    imwrite(searchData,'thinninged.png')

    function result = f1(searchData,m,n)
    searchX = [0,0,1,1,1,0,-1,-1,-1];
    searchY = [0,-1,-1,0,1,1,1,-1,-1];

%-----f1 関数-----
    white = 255;
    black = 0;
    count = 0;
    beforeVal = black;
    for k = [2,3,4,5,6,7,8,9,2]
        if searchData(m+searchY(k),n+searchX(k)) == black
            if beforeVal == white
                count = count + 1;
            end
            beforeVal = black;
        else
            beforeVal = white;
        end
    end
    result = count;

%-----f2 関数-----
    function result = f2(searchData,m,n)
```

```
searchX = [0,0,1,1,1,0,-1,-1,-1];
searchY = [0,-1,-1,0,1,1,1,-1,-1];

white = 255;
black = 0;
count = 0;
beforeVal = black;
for k = [2,3,4,5,6,7,8,9]
    if searchData(m+searchY(k),n+searchX(k)) == black
        count = count + 1;
    end
end
result = count;
```