영상처리 9주차 과제

학번: 201404376

이름: 정성욱

1. **과제 내용**

   connected를 recursive하게 구현

   opening ,closing,erosion,dilation을 구현

Connected

```
[x,y] = size(img);
%img = img > 108;
% recursive find 1 value
img = double(img);
for i =1:x
    for j = 1:y
        if(img(i,j)==1)
        img(i,j)=-1;
        end
    end
end
pad_img = zeros(x+1,y+1);
pad_img(2:x+1,2:y+1)= img;

res = my_recursive_label(pad_img,2,2,x,y,1);

img = uint8(res);
```

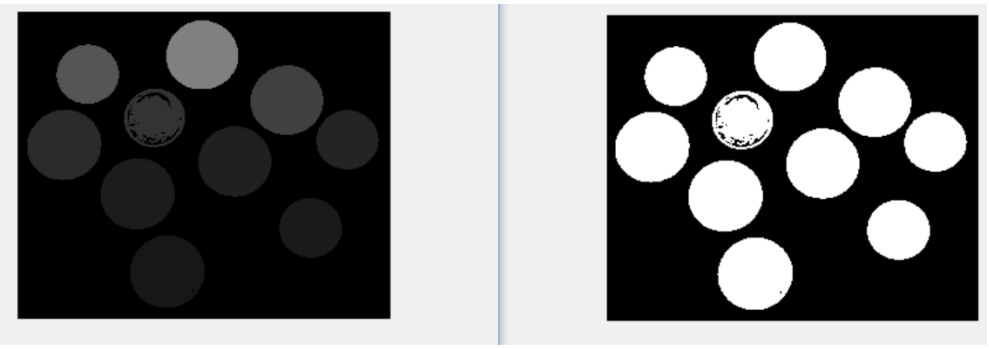
우선 이미지를 더블로 바꾼 후 1에 대응되는 픽셀은 -1로 모두 바꿔줍니다.

그후 recursive하게 돌아서 각 이미지별로 파트를 나누는 함수를 실행합니다.

```
for ix = x :width
    for iy = y: height
        if(img(ix,iy)==-1)
            if(ix<width&&iy<height&&img(ix-1,iy)==0&&img(ix,iy-1)==0)
            c=c+1;
            end
            img(ix,iy)=255/c;
            if(img(ix,iy+1)==-1)
                img = my_recursive_label(img, ix, iy+1, ix, iy+1, c); end
            if(img(ix,iy-1)==-1)
                img = my_recursive_label(img, ix, iy-1, ix, iy-1, c); end
            if(img(ix+1,iy)==-1)
                img = my_recursive_label(img, ix+1, iy,  ix+1, iy, c);end
            if(img(ix-1,iy)==-1)
                img = my_recursive_label(img, ix-1, iy, ix-1, iy, c); %end
            end

        end
    end
end
```

알고리즘은 이중 포 루프를 돌다가 -1인 픽셀을 만나게되면 우선 그 픽셀에 255/c(레이블)로 대체하고, 픽셀의 위 아래 양옆의 픽셀값을 확인한 뒤에 -1인 값이 있을 경우 재귀 호출을 하여 그 값을 255/c로 대체합니다. 하지만 만약 그 픽셀 이전에 있는 값 x축으로 -1위거나 y축으로 -1에 픽셀이 0이 아닐경우 하나의 레이블이므로 c값을 받은 그대로 사용하고 전부 0이면 새로운 객체에 온것이므로 레이블c 값을 하나 올려줍니다. 그렇게 첫 포문에서 주어진 width와 height값을 다돌면 종료됩니다.

실행결과 :



Dilation

```
% Apply dilation
[x,y] = size(img);
pad_size = floor(filter/2);
res_img = zeros(x,y);
zeros_filter =  zeros(filter,filter);
zeros_filter(pad_size+1,pad_size+1) = 1;
ones_filter =  ones(filter,filter);


for ix = 1+pad_size:x-pad_size
    for iy = 1+pad_size:y-pad_size

    if(double(img(ix-pad_size:ix+pad_size,iy-pad_size:iy+pad_size)).*double(ones_filter)==ones_filter)
        res_img(ix,iy) =  1;
    end

    end
end

dilation =res_img;
```

패드는 현재 인덱스에- +pad칸 만큼 가는 것을 가정했습니다. 예를 들어 filter가 3일 경우

Pad는 1 임

Pad	**pad**	**Pad**
pad	**X,y**	**Pad**
Pad	**Pad**	**Pad**

이므로 filter의 크기를 2로 나눈 몫의 floor가 사이즈가 됩니다.

Dilation은 정말 간단하게도 이미지를 돌면서 이미지의 필터가 전부 다 들어가는 구간의 인덱스만 표시하면되므로 저는 이미지의 크기와 동일한 zeros로 초기화한 검은 이미지 하나에 원본 이미지를 돌면서 dilation의 조건이 들어맞을 경우 인덱스값 가지고 zeros의 하나의 인덱스에 1을 넣었습니다.

Erotion

```
[x,y] = size(img);
pad_size = floor(filter/2);
res_img = zeros(x,y);
ones_filter =  ones(filter,filter);
```

우선 위처럼 초기변수를 잡아줍니다.

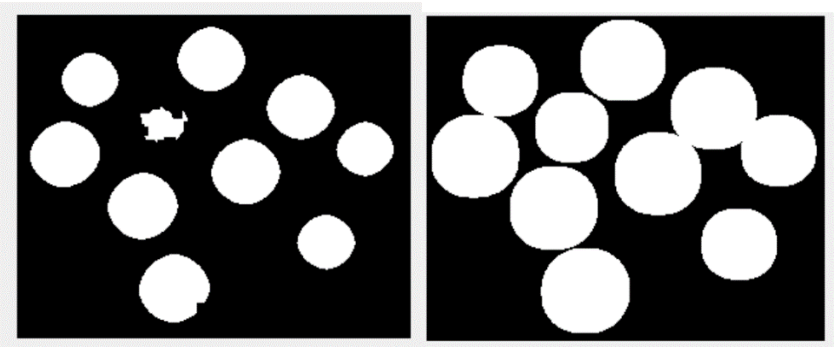
```
for ix = 1:x
    for iy = 1:y

        if(img(ix,iy) ==1)
            if((ix-pad_size<=0)&&(iy-pad_size<=0))
               res_img(1:ix+pad_size,1:iy+pad_size) =  ones_filter(1:ix+pad_size,1:iy+pad_size);

            elseif((ix-pad_size<=0)&&(iy+pad_size>y))
                res_img(1:ix+pad_size,iy-pad_size:y) = ones_filter(1:ix+pad_size,1:y-(iy-pad_size)+1);

            elseif((ix-pad_size<=0))
                res_img(1:ix+pad_size,iy-pad_size:iy+pad_size) = ones_filter(1:ix+pad_size,1:filter);

            elseif((ix+pad_size>x))
                res_img(ix-pad_size:x,iy-pad_size:iy+pad_size) = ones_filter(1:x-(ix-pad_size)+1,1:filter);


            elseif((ix+pad_size>x)&&(iy-pad_size<=0))
               res_img(ix-pad_size:x,1:iy+pad_size) =  ones_filter(1:1+pad_size,1:iy+pad_size);

            elseif((ix+pad_size>x)&&(iy+pad_size>y))
                res_img(ix-pad_size:x,iy-pad_size:y) = ones_filter(1:1+pad_size,1:1+pad_size);

            elseif((iy-pad_size<=0))
                res_img(ix-pad_size:ix+pad_size,1:iy+pad_size) = ones_filter(1:filter,1:iy+pad_size);

            elseif((iy+pad_size>y))
                res_img(ix-pad_size:ix+pad_size,iy-pad_size:y) = ones_filter(1:filter,1:y-(iy-pad_size)+1);




            else
                res_img(ix-pad_size:ix+pad_size,iy-pad_size:iy+pad_size) =  ones_filter;
            end
        end


    end
end
```

포문을 돌면서 이미지의 픽셀값이 1인 곳을 찾아다닙니다.

여기서 필터를 적용하기 전에 i-pad_size의 크기와 i+pad_size의 크기가 이미지의 인덱스를 넘지 않게 조심해야하는데 그 조건문들입니다. 만약 -쪽 패드를 뺀 것이 1보다 작은 값이 된다면 필터를 전부 씌우지않고 1까지 씌웁니다. 그에 따라 그때만큼은 씌우는 필터 크기도 넘는 이미지의 경계만큼 감소하여 슬라이싱을 ones를 씌웁니다.

마찬가지로 이미지의 +pad_size가 이미지의 최대 경계를 넘을 경우에 현재 필터의 중앙에있는 인덱스로부터 경계까지의 거리만 필터를 씌웁니다. 즉 이미지의 마지막 경계값 – 현재 인덱스의 크기로 필터값을 조정해주는 것입니다. 이것을 4방향 동서남북에 따라 적용한 것이 위 코드 입니다. 인덱스적으로만 했고 8개의 이유는 필터의 크기가 대각선 방향에도 영향을 끼칠수 있기때문에 그것을 염두해서 8방향을 전부 보았습니다.

Dilation filter_size =7 **Erotion filter_size =7**



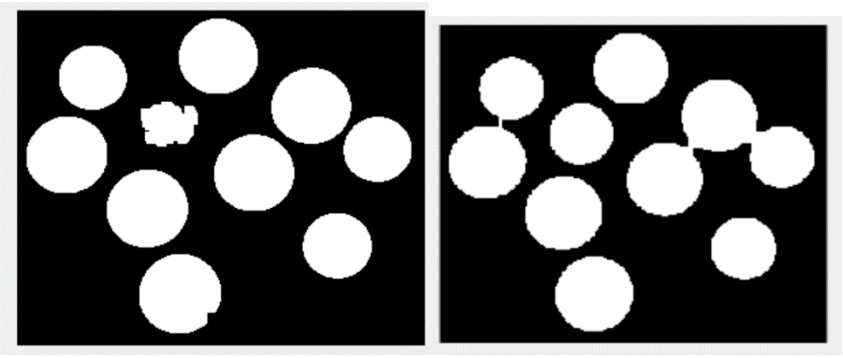
```
function opening = my_openning(img, filter)
% Apply opening of binary image
% img     : binary image
% filter  : filter for opening
% opening : result of opening
% Apply opening
opening =  my_erosion(my_dilation(img,filter),filter);
end
```

```
function closing = my_closing(img, filter)
% Calculate closing of binary image
 % img     : binary image
 % filter  : filter for closing

 % Apply closing
 closing =  my_dilation(my_erosion(img,filter),filter);
 end
```

Opening: 딜라이션에 이로션을 씌운것입니다. 리턴값 자체가 바이너리 이미지이므로 그대로 씌웠습니다.

Closing: 오프닝과 반대로 이로션에 딜라이션을 씌운 것인데 마찬가지로 같은 이로션이든 딜라이션이든 동일한 필터를 주고 이로션의 리턴값이 바이너리 이미지이므로 따로 작업은 필요없습니다.

Opening filter_size =7 **Closing filter_size =7**



2. 느낀 점

:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분