1. 사진의 크기를 원래 사진의 0.83배와 1.47배 크기로 조정합니다.

1) 최단입점 보간(NN), 이중 선형 보간을 위한 크기 조정 기능(myResizeNN, myResizeBill) 만들기

2) 사진에 기능을 적용 후 matlab에서 imresize를 통해 얻은 결과와 비교하기

3) 보간 방법 및 이미지 품질 측면에서 결과를 분석하기

Lab.2-2)

NN보간법)

```
function out = myResizeNN(input, val)
[in_row in_col] = size(input);

out = zeros(ceil(in_row*val), ceil(in_col*val));
[out_row out_col] = size(out);

x = round((1:out_col)/val);
y = round((1:out_row)/val);

for i = 1:out_row
    for j = 1:out_col
       x_n = x(j);
       y_n = y(i);
       if x_n == 0
           x_n = 1;
       elseif x_n>in_col
           x_n = in_col;
       elseif y_n == 0
           y_n = 1;
       elseif y_n>in_row
           y_n = in_row;
       end

       out(i,j) = input(y_n, x_n);
    end
end
out = uint8(out);
```

resize value 값(0.38혹은 1.47)을 받아 zeros를 사용하여 결과가 들어갈 'out'행렬을 만든다. out행렬 크기를 val값만큼 나누어 각 픽셀의 정확한 위치를 구해준다. 그 값들을 round함수를 이용해 해당 픽셀이 가장 가까운 값이 어딘지를 x,y에 넣어 for문을 통해 out행렬 값을 변경 해 주었다.

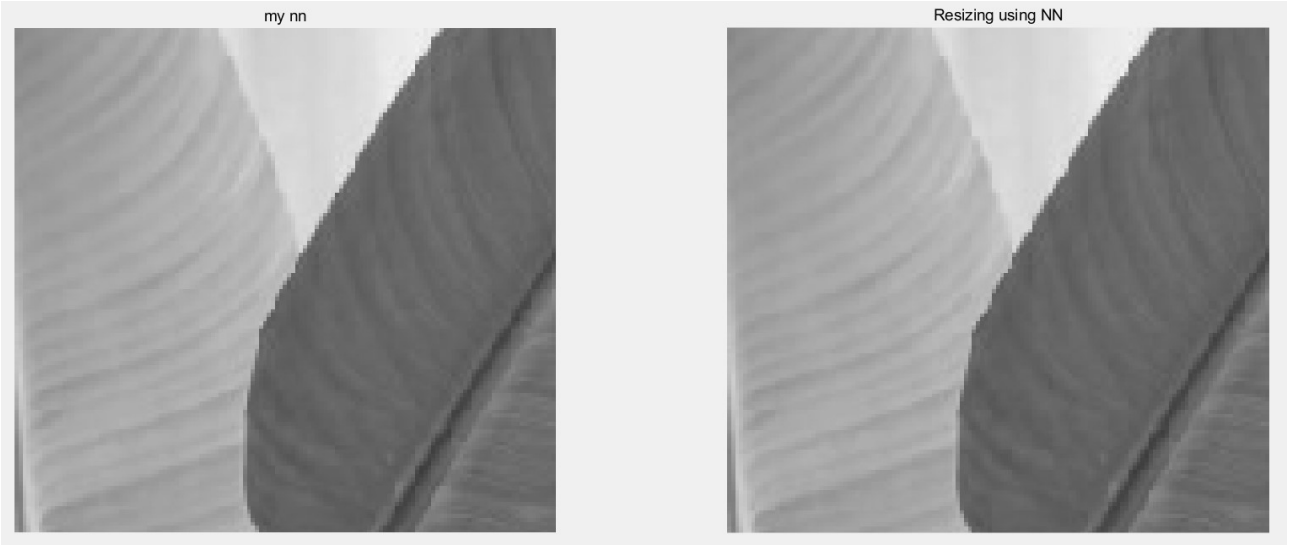
Bilinear보간법)

NN과 같은 방식으로 픽셀의 정확한 위치를 계산해준 후, ceil을 통해 값을 무조건 올려 원래값과 차를 계산하여 가중치(x_weight, y_weight)를 구하였다.

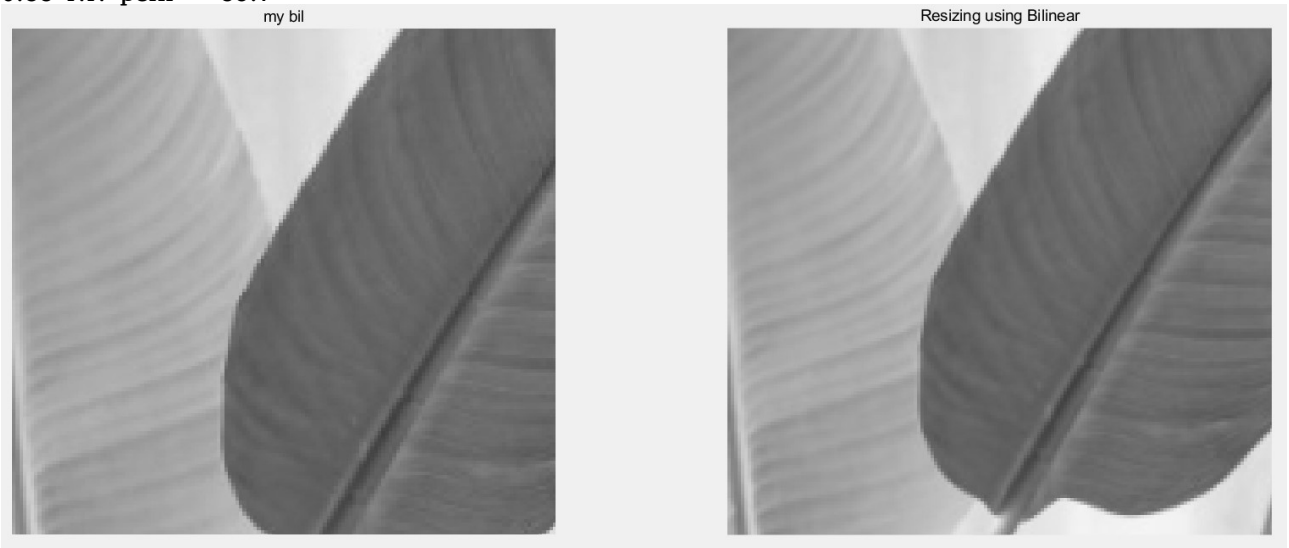
```
x_weight = [];
y_weight = [];

for i = 1:y
    for j = 1:x
        %weight
        x_w = ceil(x_val(j))-x_val(j);
        y_w = ceil(y_val(i))-y_val(i);
        x_weight(j) = x_w;
        y_weight(i) = y_w;
    end
end
```

그리고 a와 b 변수에 각 픽셀을 무조건 올림, 무조건 내림하고, for문을 사용하여 픽셀의 위치에 주변 값들을 가중치만큼 고려한 값으로 result행렬을 채워주었다.

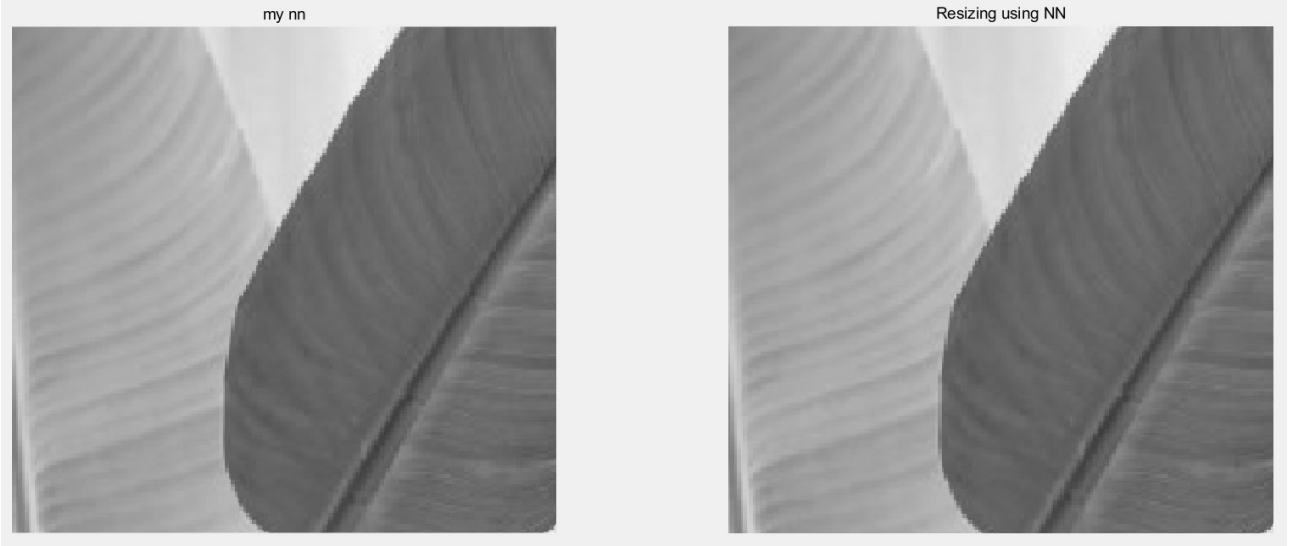
```
for i = 1:y
    for j = 1:x
        %weight
        a = floor(x_val(j));
        a_1 = ceil(x_val(j));

        if (a_1 > in_col)
            a_1 = in_col;
        end

        if (a == 0)
            a = 1;
        end

        b = floor(y_val(i));
        b_1 = ceil(y_val(i));

        if b == 0
            b = 1;
        elseif b>in_row
            b = in_row;
        end

        if b_1>in_row
            b_1 = in_row;
        end

        result(i,j) = input(b,a)*
    end
input(b,a)*(y_weight(i))*(x_weight(j)) + input(b,a_1)*(y_weight(i))*(1-x_weight(j))
+ input(b_1,a)*(1-y_weight(i))*(x_weight(j)) + input(b_1,a_1)*(1-y_weight(i))*(1-x_weight(j));
```



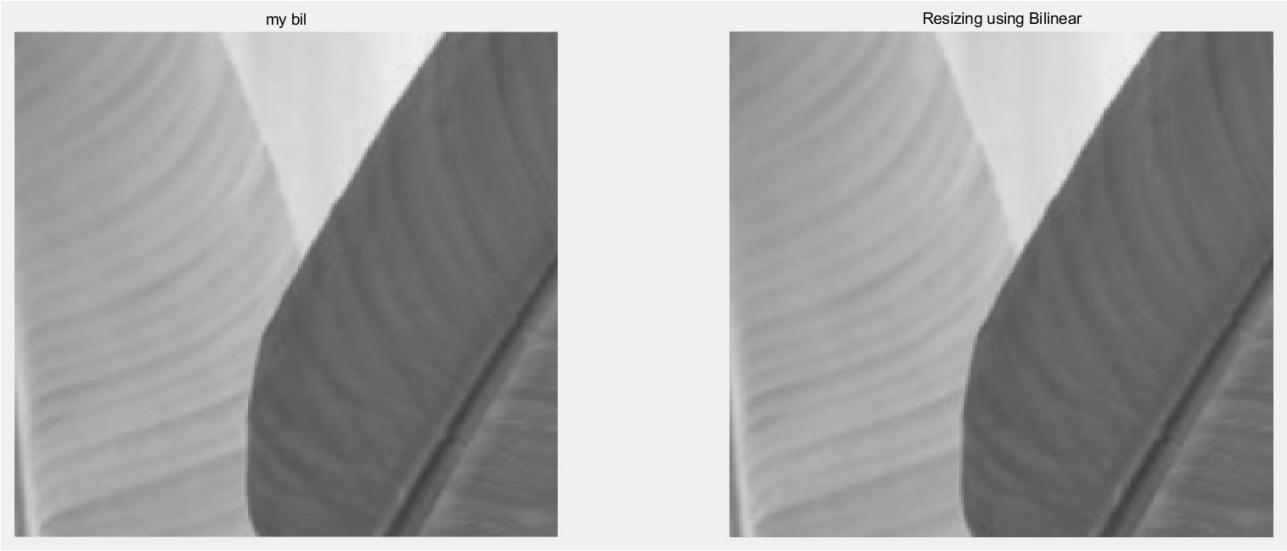
0.83 NN psnr = 35.7



0.83 Bil psnr = 43.5



1.47 NN psnr = 33.7



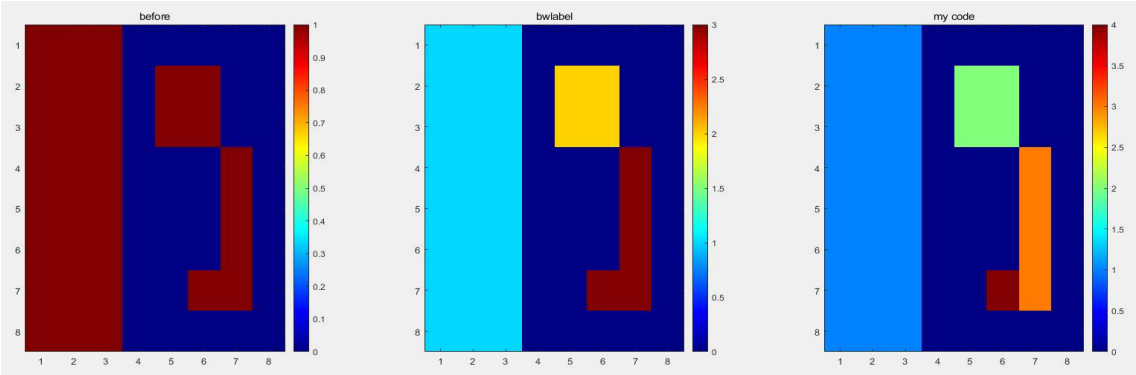
1.47 Bil psnr = 41.5

육안으로 봤을 때 차이가 확실히 구분되지 않아 figure창에서 나뭇잎 부분을 확대한 결과이다. resize 결과 bilinear 보간법이 nearest neighbor 보간법 보다 화질이 좋은 상태로 이미지가 변형됐다는 것을 알 수 있었다. 두 보간법의 품질 차이가 느껴지는 이유는 NN의 경우 변형된 좌표에서 해당값을 반올림하여 가장 거리가 가장 가까운 점을 선택한 값이다. biliear는 하나의 값만 선택하는 것이 아닌 주변 픽셀들과 얼마나 가까운지를 모두 고려한 가중치 값이 들어가기 때문에 NN처럼 뚝뚝 끊기는 느낌이 들지 않고 이미지가 변형된다.

2. 알고리즘을 구현하여 Lab.2-3,2-4와 동일한 작업을 수행

Lab.2-3)

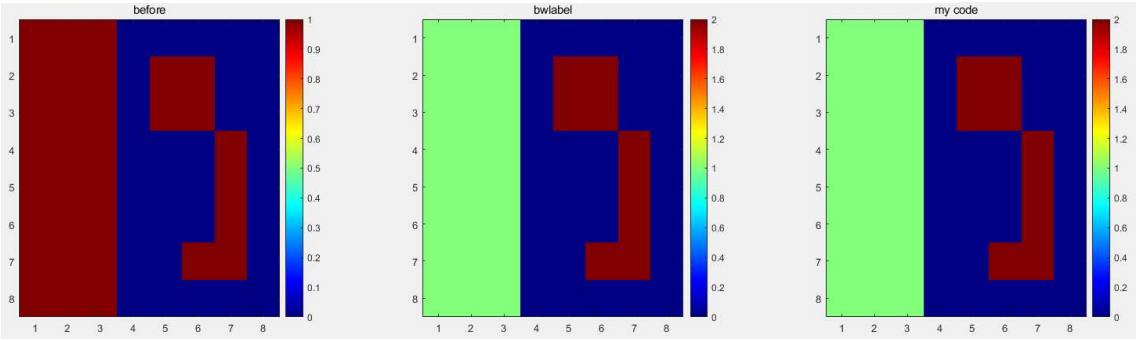
수업 때 배운 방식(start from top left to bottom right)대로 A,B,C,D(4adjacency의 경우 A,B)를 만들었으나 4adjacency는 bwlabel과 다르게 나오는 것을 확인할 수 있었다. 직접 짠 코드의 경우 같은 영역에 있어도 모서리 부분이 이상해지는 경우가 생기는 데에 반해 bwlabel의 출력은 모서리 부분이 다른 색을 유지하는 것이 아닌 동일한 색으로 라벨링되었다.



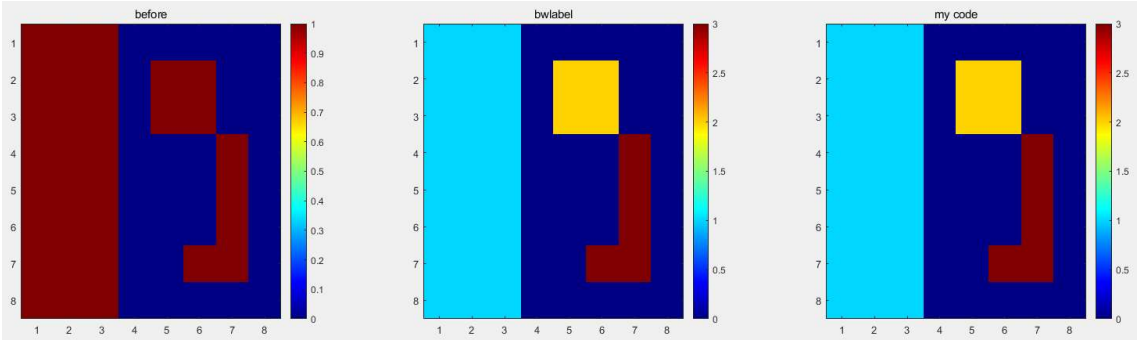
모서리 부분의 라벨링 이상

따라서 매트랩 bwlabel과 똑같이 구현하기 위해서 주변 색의 값을 행렬로 만들어 0을 제외하고 2개 이상이면 더 높은 컬러를 가진 픽셀들을 낮은 컬러로 바꾸는 코드를 추가했다.

```
sur = unique([0,A,B]); %duplication remove and sorting
[sur_row, sur_col]= size(sur);

if sur_col == 1
    c = c+1; %color count + 1
    result(i,j) = c;
elseif sur_col == 2
    result(i,j) = sur(2);
else
    result(i,j) = sur(2);
    result(i,j-1) = sur(2);
end
```



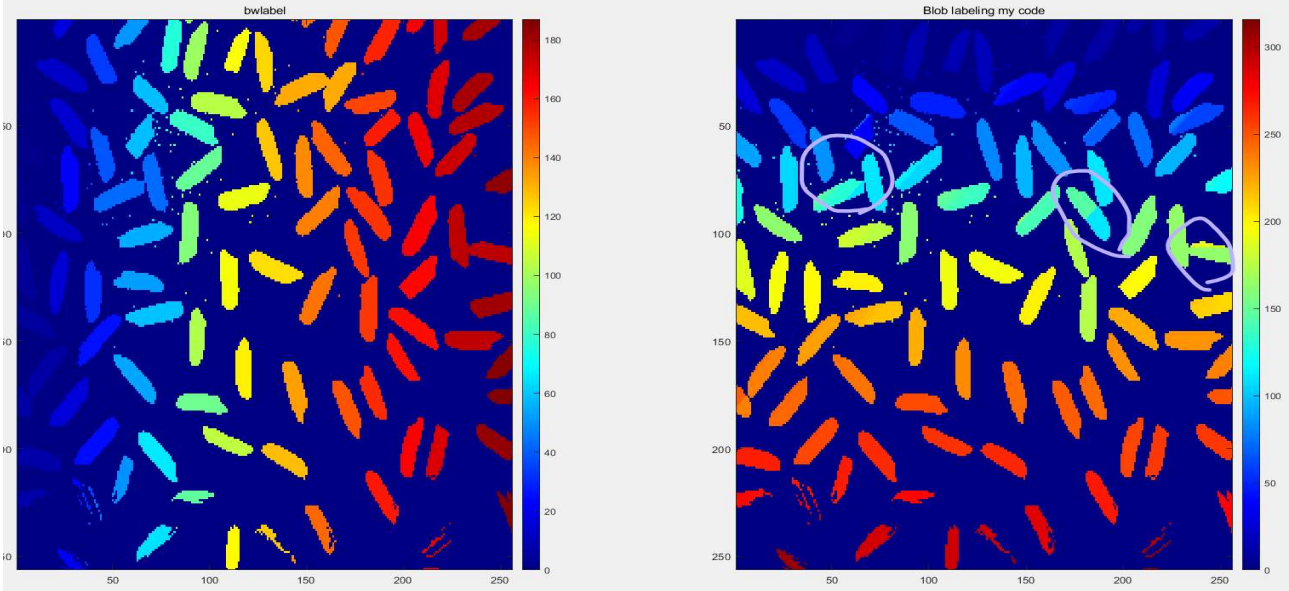
8adjacency-왼쪽부터 원본, bwlabel, 구현코드



4adjacency-왼쪽부터 원본, bwlabel, 구현코드

Lab.2-4)

2-3과 마찬가지로 영역의 왼쪽 끝부분에서 값이 이상한 경우가 많아서 이전까지 라벨링 했던 값들을 읽어 값을 교체해주는 코드를 추가해 구역마다 색을 통일시켰다.

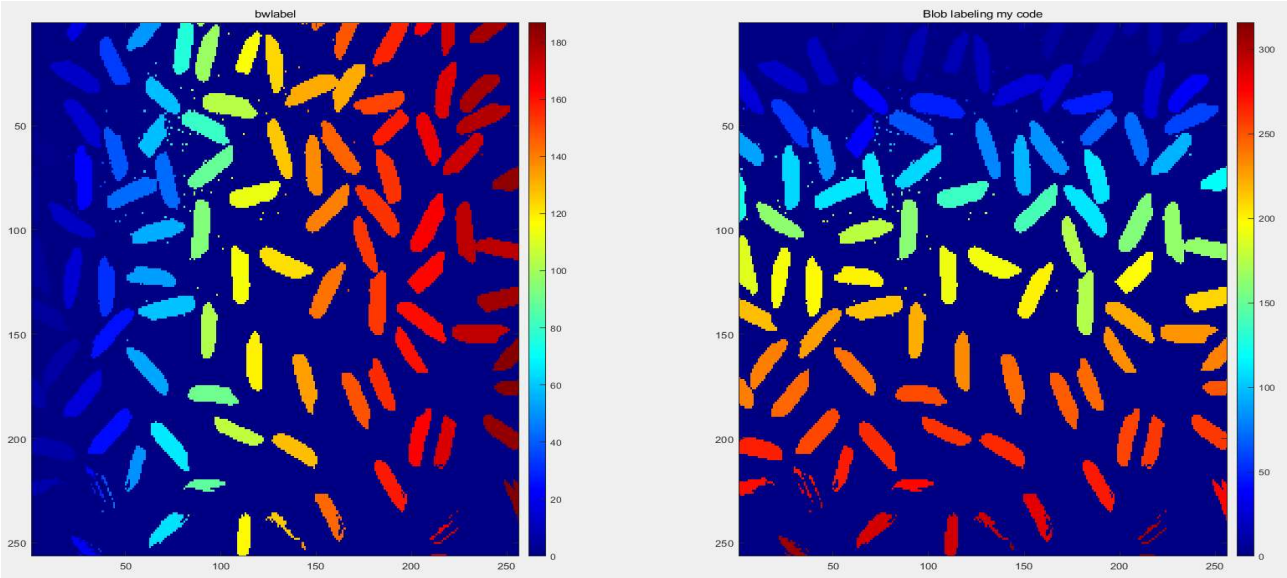


수정 전 코드 : 일부분 끊김 발생

해당 코드 설명 - 주변 값의 개수(변수 sur)가 하나라면 현재 주변값 행렬에 0만 있는 것이다. 즉, 배경이고 새로운 라벨링 값이 된다. sur 행렬의 길이가 2가 된다면 주변에 라벨링 값이 있다는 것이고 그 값이 해당 영역의 라벨링값이 된다. 만약 sur행렬의 길이가 3이라면 주변에 값이 2개가 존재한다는 것이고 가장 작은 값을 채택한다. 그리고 같은 영역은 색이 같아야한다는 조건을 맞춰주기 위해 큰값을 가진 영역의 값을 작은 값으로 바꾸어 준다.

```
if sur_col == 1
    c = c+1;
    result(i,j) = c;
elseif sur_col == 2
    result(i,j) = sur(2);
else
    result(i,j) = sur(2);
    low = sur(2);
    for m = 1:j
        for n = 1:i
            if (result(n,m) == sur(3))
                result(n,m) = sur(2);
            end
        end
    end
end
```

bwlabel 함수와 그라데이션 방향이 달라진 이유는 코드를 왼쪽 상단부터 오른쪽 아래로 가는 방식으로 구현하였기 때문인 것같다.



수정 후 코드 : 영역이 확실하게 나누어짐