

## < 목차 >

0. Issue .....	3
1. Survey related technologies .....	3
1-1. Nearest Neighbor .....	3
1-2. Bilinear .....	4
1-3. Bicubic .....	6
1-4. Zero Padding .....	8
1-5. Deep Learning(VDSR 신경망) .....	11
2. Matlab-based implementation .....	12
2-1. Nearest Neighbor .....	12
2-2. Bilinear .....	15
2-3. Bicubic .....	18
2-4. Zero Padding .....	21
2-5. Deep Learning(VDSR 신경망) .....	25
3. Result .....	28

## Issue

최근 Full HD (1920x1080), Ultra HD (3,840x2,160) display 보급으로 과거 방대한 SD(640x480)급 영상들의 크기를 해당 해상도로 변환해야 한다는 문제가 있음.

따라서 해당 프로젝트에서는 SD급 영상의 해상도를 변환하기 위한 Super-resolution 방법 5가지에 대한 조사과 구현 방법을 설명함으로써 수업 시간에 배운 내용들을 재정립하고 이론적인 측면과 구현의 측면에서 학습도를 더욱 높이는 활동을 함.

## Survey related technologies

### 1-1. Nearest Neighbor

Nearest neighbor interpolation은 최근접화소 보간법으로 그림1 처럼 현재 픽셀 위치에서 가장 가까운값을 채택하는 방식이다. 수식적으로는 그림2와 같이 표현된다. nn의 출력을 보면 다른 보간법에 비해 픽셀이 선명하게 보이는 경향이 있는데 nn의 가중치를 구하는 방식때문이다. 그래프의 기울기가 자연스럽게 변화하지 않고 계단 형태를 띠어 이미지를 봤을 때 부드럽지 않고 끊기는 느낌이 들게 되는 것이다.

과제는 각 배열의 위치값을 알아내기 위해 output배열 크기에서 scale factor를 나누어주는 방식으로 수행했다. 이 과정에서 코드 내부에서 round를 사용할 시 값이 0으로 떨어져 index error가 나는 경우가 생겨 매트랩의 round function인 round(반올림), floor(내림), ceil(올림) 중 ceil을 사용하여 인덱스로 사용할 값이 0이 나오는 경우를 없애주었다.

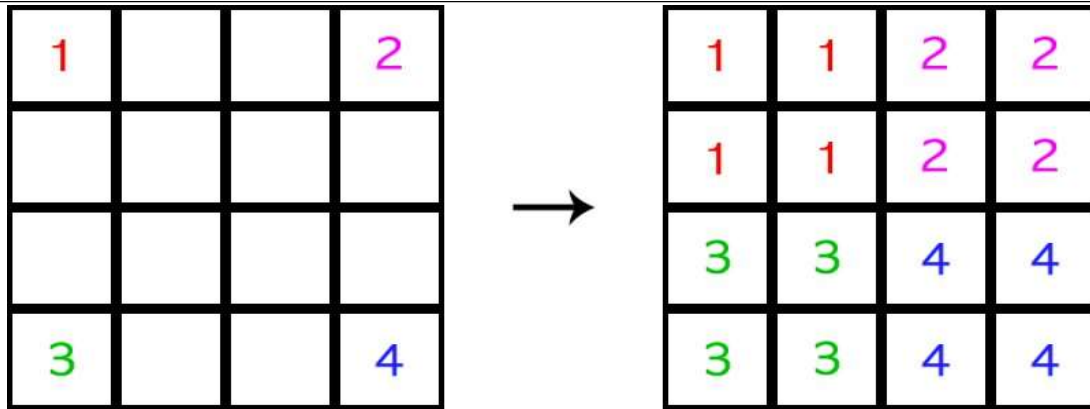


그림1

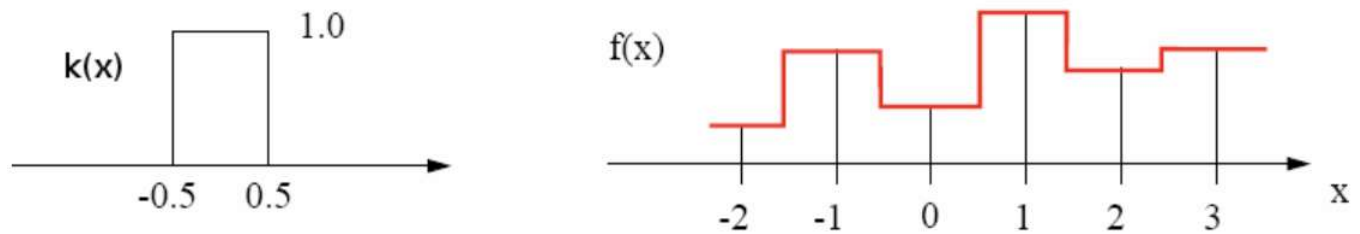


그림2

## Survey related technologies

### 1-2. Bilinear

bilinear 보간법은 인접한 4개의 픽셀의 거리를 측정해 가중치를 부여하는 방법이다. 1차원에 본 그림 1의 보간 대상은  $x$ 이고 인접한 점( $x_0, x_1$ )을 고려한다.  $y$ 값 추정을 위해 기울기를 고려하여  $\frac{f(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$ 로 정리되고  $f(x) = \frac{x - x_0}{x_1 - x_0}(f(x_1) - f(x_0)) + f(x_0)$ 와 같이 구할 수 있다. 이미지 측면에서 본다면 임의의 데이터값은 그림3과 같은 형태이고,  $f(x) = \frac{d_2}{d_1 + d_2}f(x_1) + \frac{d_1}{d_1 + d_2}f(x_2)$ 로 정리할 수 있다.

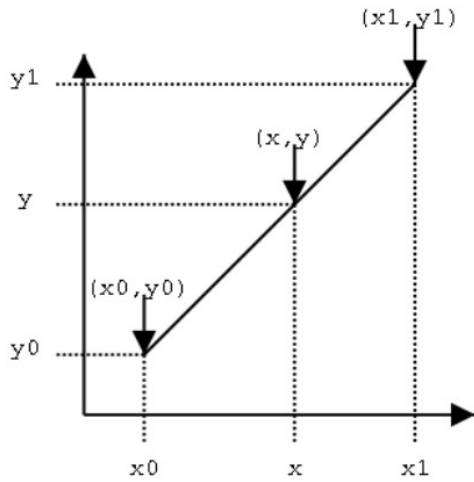


그림1

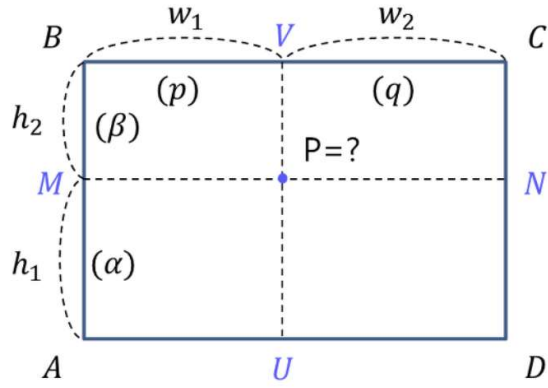


그림2

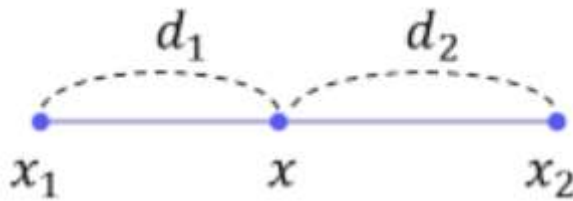


그림3

그림 2처럼 현재 픽셀의 위치에서 주변 화소 B, C, A, D의 좌표값을 안다고 가정하면 다음과 같은 수식으로 현재 픽셀의 값을 알 수 있다.

$$P = Bw_2h_1 + Aw_2h_2 + Cw_1h_1 + Dw_1h_2$$

이 식을 매트랩 수식으로 구현하기 위해서는 ceil과 floor를 이용하여 가까운 4개 픽셀의 위치를 알 수 있게된다. 위에 나타난 식에서는 정규화 과정이 반영되어 있지만  $d_1 + d_2$  즉, 각 픽셀들 간의 거리가 무조건 1이기 때문에 따로 정규화 과정이 필요하지 않다. 따라서 구해진 가중치 값들과 픽셀의 화소값을 곱한 결과를 더해주면 P점의 값을 얻을 수 있다.

3차 보간법은 그림1과 같이 주변 점들의 값을 이용하여 3차함수를 만들고 그 함수를 통해  $x$ 의 값을 보간하는 방법이다. 3차 함수가 된다면  $f(x) = mx^3 + nx^2 + jx + k$ 의  $m, n, j, k$ 를 구해야한다.  $x$ 값에 0과 1을 대입하여 미지수가 3개인 방정식  $m + n + j = y_1 - y_0$ 를 얻게 되고 남은 미지수들을 구하기 위해 미분이 필요하다. 미분된 함수에 또 0과 1을 대입해주면  $f'(0) = j, f'(1) = 3m + 2n + j$ 가 식 세개를 연립하여 결과가 도출된다. 하지만 이미지는 이산적인 값들로 이루어져 미분치를 얻기 위해서는 이전 혹은 이후의 값을 필요로하게 된다. 따라서 3차 보간법은 그림1과 같이 -1, 2에 해당하는 값까지 포함하여 보간 시 4개의 점을 참조하게 된다.

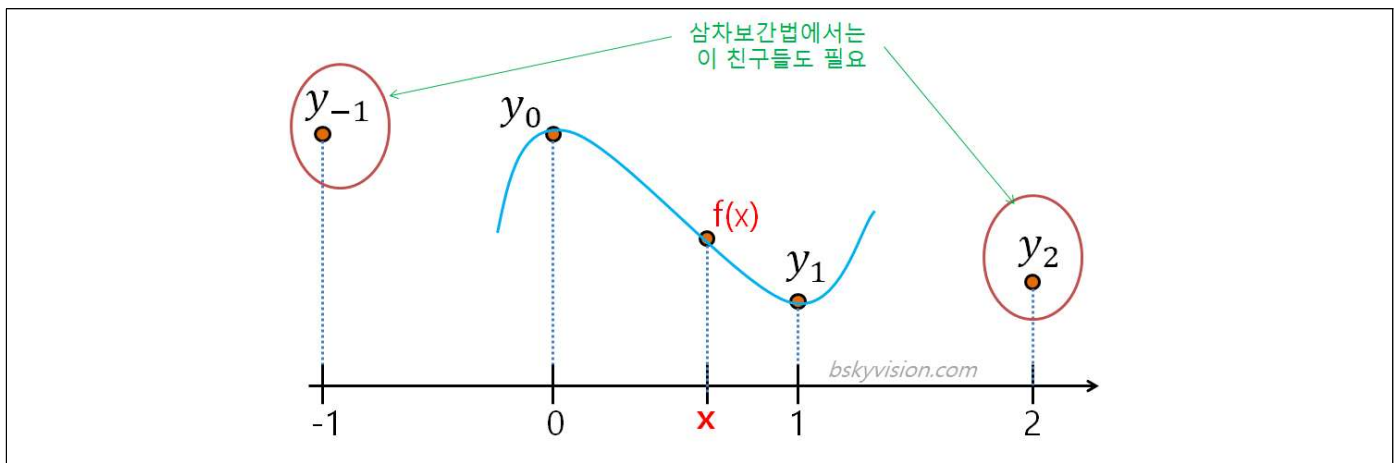


그림1

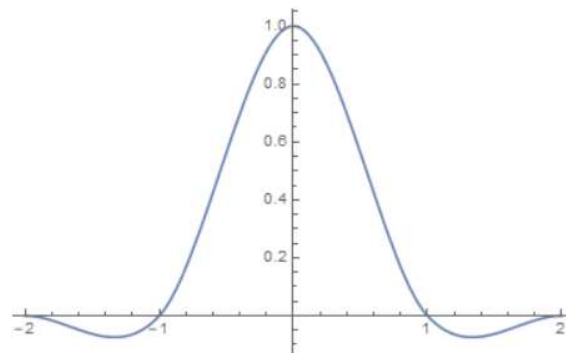
bicubic 보간법은 3차원 보간법으로 가중치 함수를 정의하고 2차원이기때문에 16개의 주변 화소를 이용하여 새로운 화소를 생성하는 방식이다. 과제 수행 코드에서는 각 픽셀간의 가중치를 주기위한 kernel이 존재한다. 그림2(b)에서 확인할 수 있 듯 bicubic은 완만한 형상의 그래프이다. 따라서 경사가 급한 nearest나 bilinear에 비해 이미지가 smoothing되는 효과가 나타난다. 그림2(a)에서  $x$ 는 거리  $a$ 값은 조정 파라미터로 음수값을 가지며 그래프에서 언더슈트의 정도를 나타낸다. 일반적으로 -0.5를 사용하지만 onpencv와 matlab에서는 -0.75를 사용하기때문에 과제 수행 파라미터 값은 -0.75로 선정하였다.

그림3(a),(b)은 보간 과정, (c)는 보간방정식을 나타낸 그림이다. 과제 수행 코드에서는 수직방향 보간을 진행 후 수평방향을 보간하여 수행하지만 수평 수직 순서를 바꿔서 진행했을 때도 결과가 동일하게 출력되었다. 그림3(a)와 같이 수직방향으로 4번 보간을 하여 ( $P_0, P_1, P_2, P_3$ )

가 생성된다. 생성된 4개의 화소를 이용하여 그림3(b)처럼 수평 방향으로 1번 보간을 해주면 해당 픽셀의 값을 얻을 수 있다.

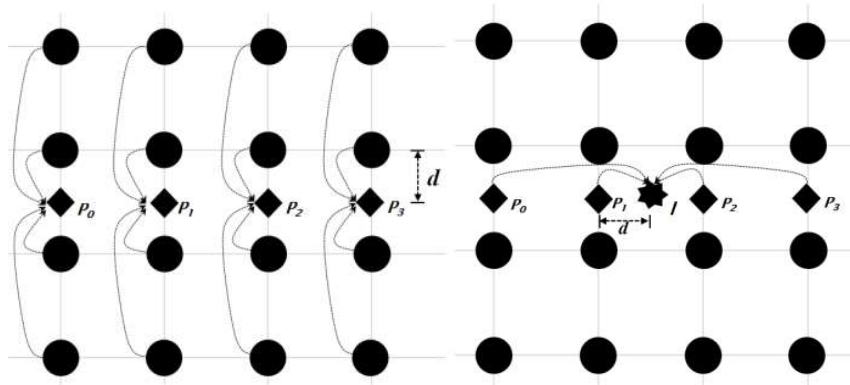
$$f(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}$$

(a) 가중치 커널 수식



(b) 커널의 그래프 형상

그림2



(a) 수직보간

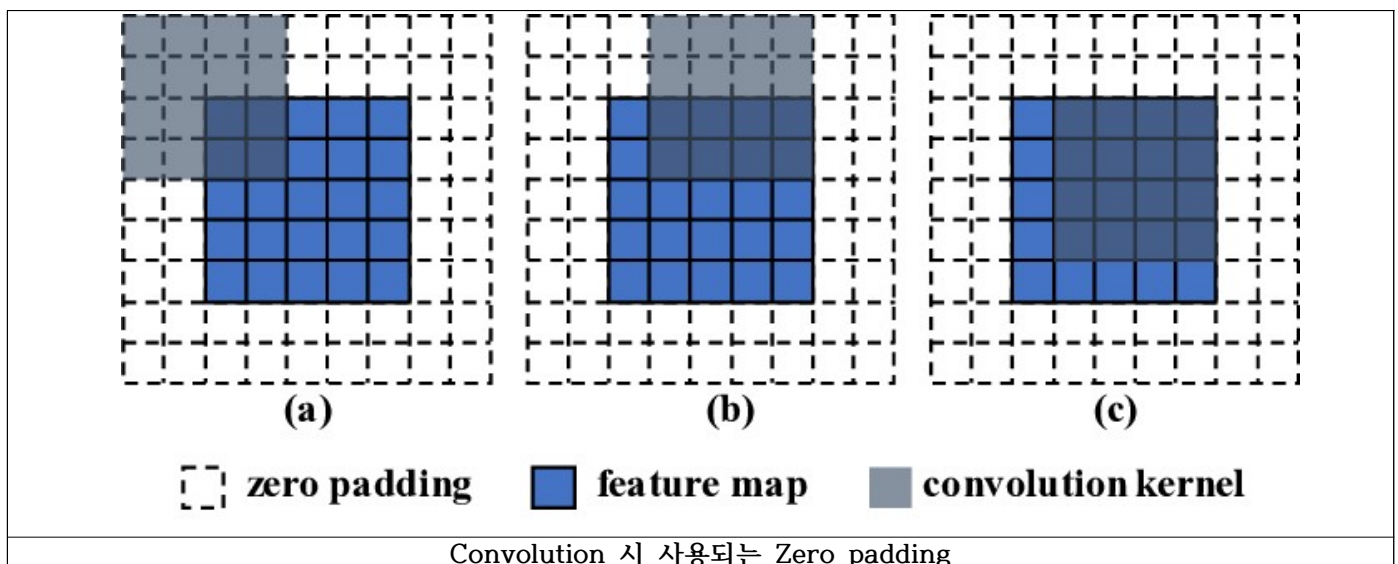
(b) 수평보간

$$\sum_{i=-1}^2 \sum_{j=-1}^2 f(x+i, y+j) C_a(dx+i) C_a(dy+j)$$

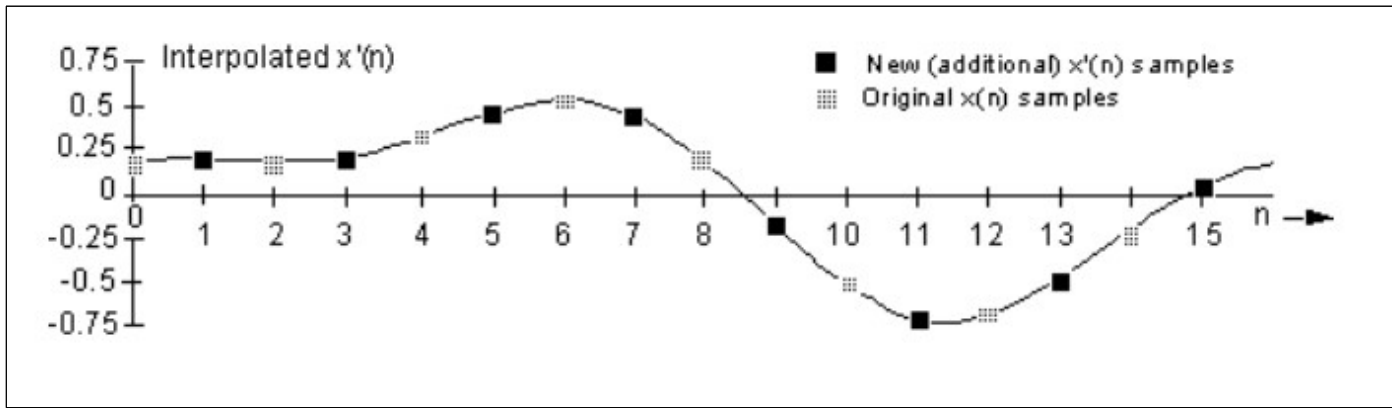
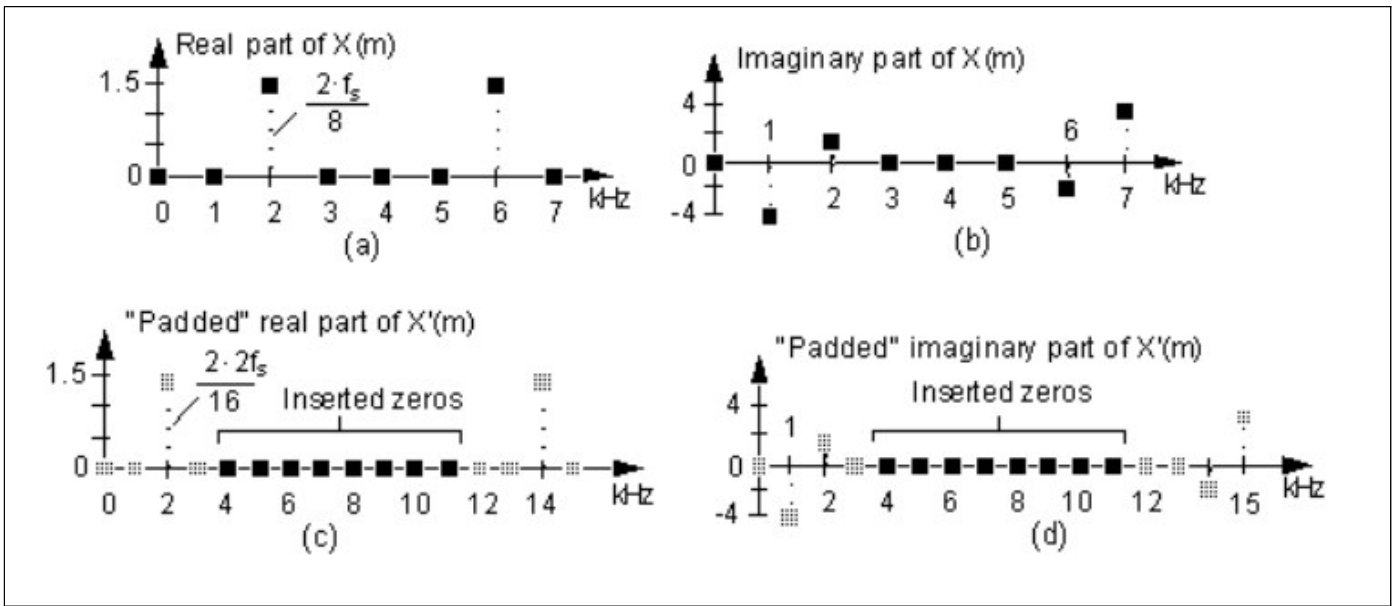
(c) bicubic 수식

그림3

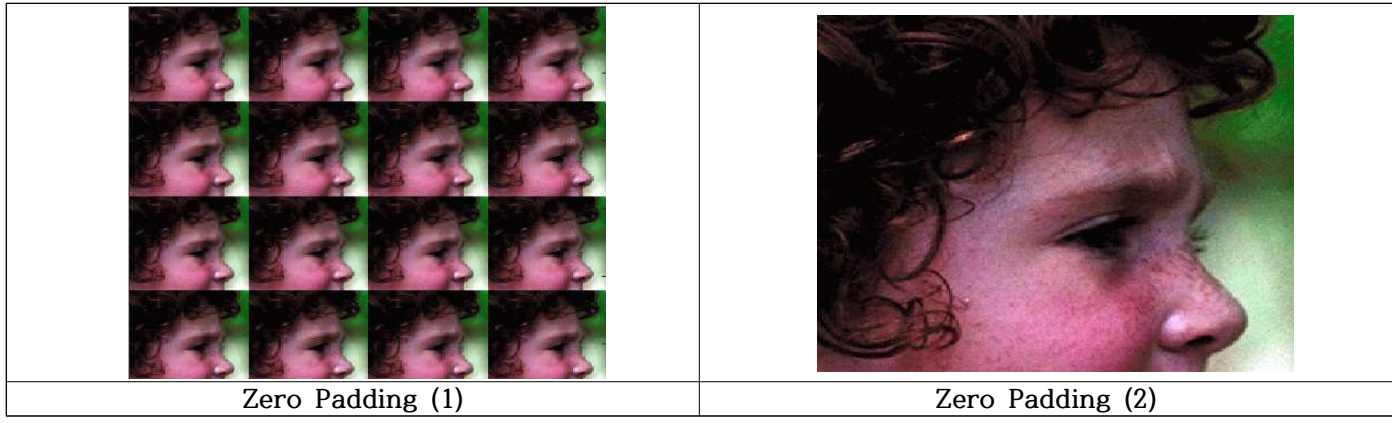
Zero padding은 padding에 대한 값을 0으로 하여 유익한 결과물을 내는 과정을 말한다. 이는 주로 딥러닝 혹은 필터링 과정에서 많이 사용되는데 Convolution 연산을 할 시, 필터 크기로 인해 원본 입력 matrix의 크기가 줄어드는 상태로 출력되는 것을 방지할 때 사용된다. 하지만 이를 Super-resolution을 위해 Frequency Domain에서 사용할 때는 원본 영상의 주파수 정보를 원하는 크기(확대된 크기)로 변환된 이미지에 적용할 때 사용된다. 이때 원본 영상의 주파수 정보 사이에 0을 끼워넣는 방식으로 주파수 영역에서의 이미지 확대 및 화질 개선을 이루어 낸다.



Frequency Domain에서의 Zero Padding은 아래 그림과 같이 Sampling 된 주파수 정보 사이에 0을 끼워넣음으로써 원래 가지고 있던 주파수 성질을 유지하되, 샘플링 주파수를 키울 수 있다.



아래 그림은 해당 실험에서 사용한 Zero Padding 방식이고, 이론상으로 구현한 방식 (1) 및 같은 원리로 구현된 방식 (2)가 매트랩 내장 함수에 의해 다른 방식으로 출력되는 것을 아래 그림과 같이 확인할 수 있다.





1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	0	2	0	3	0	4	0
0	0	0	0	0	0	0	0
5	0	6	0	7	0	8	0
0	0	0	0	0	0	0	0
9	0	10	0	11	0	12	0
0	0	0	0	0	0	0	0
13	0	14	0	15	0	16	0
0	0	0	0	0	0	0	0

본 프로젝트에서 사용한 Zero Padding (1)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	2	0	0	0	0	3	4
5	6	0	0	0	0	7	8
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
9	10	0	0	0	0	11	12
13	14	0	0	0	0	15	16

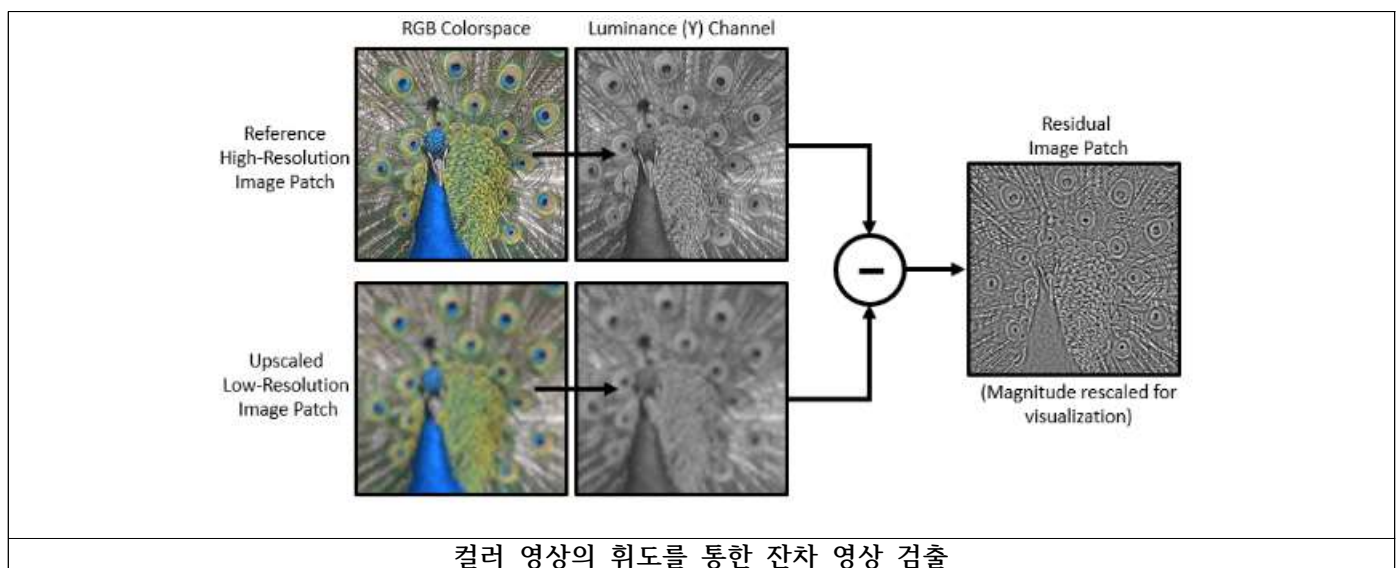
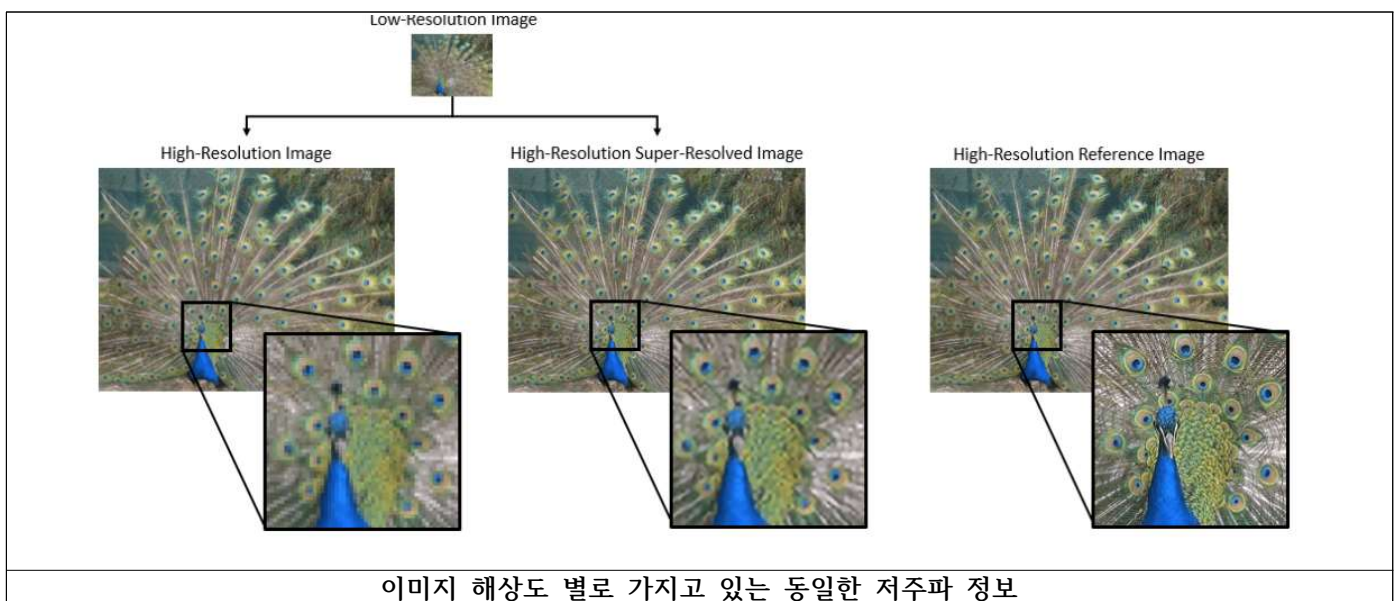
본 프로젝트에서 사용한 Zero Padding (2)

## Survey related technologies

### 1-5. Deep Learning (VDSR 신경망)

VDSR 신경망은 Very-Deep Super-Resolution의 약자로 단일 영상 초고해상도 복원을 딥러닝으로 수행하는 알고리즘이다.

VDSR은 저해상도 영상과 고해상도 영상 사이의 매핑을 학습한다. 이 매핑은 아래 그림과 같이 저해상도 영상과 고해상도 영상이 비슷한 영상 정보를 가지고 있고, 주로 고주파에 해당하는 세부 정보에서 차이가 있기 때문에 이를 이용하여 저화질의 이미지를 고화질로 변환할 수 있다.



위 그림과 같이 해당 신경망은 컬러 영상의 휘도로부터 잔차 영상을 검출하는데 이는 참조 영상의 크기와 일치하도록 쌍삼차 보간(Bicubic)을 사용하여 업스케일링 후 저해상도 영상과 고해상도 참조 영상의 차이로부터 구할 수 있다.

여러 영상으로부터 취득한 잔차 영상을 데이터로 하여 신경망을 학습시키고, Input 되는 영상의 잔차 영상을 추정할 수 있도록 구성한다. 이렇게 추정된 잔차 영상을 가지고 업샘플링된 저해상도 영상에 더해준 후 RGB 스페이스로 변환할 경우 고해상도 영상으로의 복원이 가능하다.

## Matlab-based implementation

### 2-1. Nearest Neighbor

```
clc
clear all
close all

input = imread("[크기변환]woman.png");
real_nn = imread("woman.png");
t = now;
[in_row, in_col, rgb] = size(input);
val = 4;

out_rgb = zeros(round(in_row*val), round(in_col*val), rgb); % 출력 matrix
for rgb_level = 1:3
    rgb_1D = input(:, :, rgb_level); % rgb를 순서대로 연산해준다.
    out = zeros(round(in_row*val), round(in_col*val));
    [out_row, out_col] = size(out);

    x = ceil((1:out_col)/val); % 올림 함수를 이용하여 0값 방지
    y = ceil((1:out_row)/val);

    for i = 1:out_row
        for j = 1:out_col
            x_n = x(j);
```

```

        y_n = y(i);
        out(i,j) = rgb_1D(y_n, x_n);
    end
end
out_rgb(:, :, rgb_level) = out;
end

my_nn = uint8(out_rgb);
% plot
fprintf("구현 함수 Wn")
t = now-t
nn_psnr = psnr(real_nn, my_nn)
fprintf("내장 함수 Wn")
t = now;
built_in_nn = imresize(input, val, 'nearest');
t = now-t
nn_psnr = psnr(real_nn, built_in_nn)
figure(1);
subplot(1,3,1);imshow(real_nn);title('원본','fontsize',20);
subplot(1,3,2);imshow(my_nn); title('구현 이미지','fontsize',20);
subplot(1,3,3);imshow(built_in_nn); title('imresize 이미지','fontsize',20);

```

원본



구현 이미지



imresize 이미지





원본



구현 이미지



imresize 이미지



원본



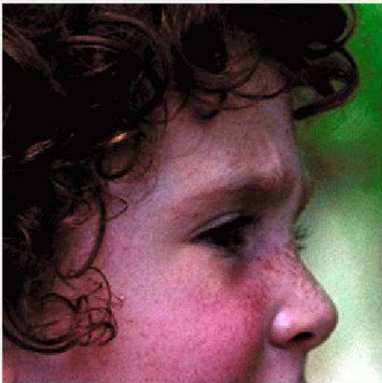
구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



## Matlab-based implementation

### 2-2. Bilinear

```
clc
clear all
close all

input = imread("[크기 변환]woman.png");
real_bil = imread("woman.png");
t = now;
[in_row, in_col, rgb] = size(input);
val = 4;

x = in_col*val;
y = in_row*val;

y_val = (1:y)/val+(0.5*(1-1/val));
x_val = (1:x)/val+(0.5*(1-1/val));

x_weight = [];
y_weight = [];

out_rgb = zeros(round(in_row*val), round(in_col*val), rgb);

for rgb_level = 1:3
    rgb_1D = double(input(:, :, rgb_level));
    for i = 1:y % 가중치 생성을 위한 for문
        for j = 1:x
            x_w(j) = ceil(x_val(j))-x_val(j); % 픽셀 간 거리값
            y_w(i) = ceil(y_val(i))-y_val(i);
        end
    end
    pad_image = double(padarray(rgb_1D,[1 1],'symmetric')); % index에러 방지를 위한 패딩
    result = [];
    for i = 1:y
        for j = 1:x
            %weight
            a = floor(x_val(j))+1;
            a_1 = ceil(x_val(j))+1;
            b = floor(y_val(i))+1;
```

```

        b_1 = ceil(y_val(i))+1;
        out(i,j) = pad_image(b,a)*(y_w(i))*(x_w(j)) + pad_image(b,a_1)*(y_w(i))*(1-x_w(j)) + ...
            pad_image(b_1,a)*(1-y_w(i))*(x_w(j)) + pad_image(b_1,a_1)*(1-y_w(i))*(1-x_w(j));
    end
end
out_rgb(:, :, rgb_level) = out;
end

my_bil = uint8(out_rgb);

fprintf("구현 함수 %n")
t = now-t
my_bil_psnr = psnr(real_bil, my_bil)

fprintf("내장 함수 %n")
t = now;
built_in_bil = imresize(input, val, 'bilinear');
t = now-t
psnr_bil_psnr = psnr(real_bil, built_in_bil)
figure(1);
subplot(1,3,1); imshow(real_bil);title('원본','fontsize',20);
subplot(1,3,2); imshow(my_bil); title('구현 이미지','fontsize',20);
subplot(1,3,3); imshow(built_in_bil); title('imresize 이미지','fontsize',20);

```

원본



구현 이미지



imresize 이미지





원본



구현 이미지



imresize 이미지



원본



구현 이미지



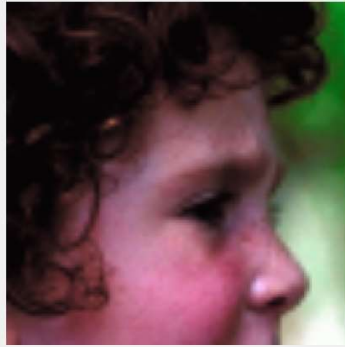
imresize 이미지



원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지





## Matlab-based implementation

### 2-3. Bicubic

```
clc
clear all
close all

input = imread("[크기변환]woman.png");
real_bic = imread("woman.png");
val = 4; % scale factor
t=now;
[in_row in_col rgb] = size(input);
out = zeros(ceil(in_row*val), ceil(in_col*val)); % 2차원 출력 배열
[out_row out_col] = size(out);
out_rgb = zeros(round(in_row*val), round(in_col*val), rgb); % 3차원 출력 배열
y = (1:out_row)/val+(0.5*(1-1/val)); % 확대된 좌표들의 y 위치
x = (1:out_col)/val+(0.5*(1-1/val)); % 확대된 좌표들의 x 위치

for rgb_level = 1:3
    rgb_1D = double(input(:, :, rgb_level)); %R->G->B 순서대로 연산
    % y값 정의
    for i = 1:out_row
        y1 = floor(y(i)) - 1;
        y1(y1 < 1) = 1;
        y1(y1 > in_row-3)=in_row - 3;
        dy = y(i)-y1-1; % -1번 기준
        % x값 정의
        for j = 1:out_col
            x1 = floor(x(j)) - 1;
            x1(x1 < 1) = 1;
            x1(x1 > in_col-3) = in_col-3;
            dx = x(j)-x1-1;
            for p = -1:2
                r_weight(p+2) = kernel1(dy-p); %y방향 가중치 -1번부터 2번까지
                c_weight(p+2) = kernel1(dx-p); %x방향 가중치
            end
            sum_x=sum(c_weight); sum_y=sum(r_weight);
            % 세로 방향부터 보간진행
            for k=0:3
```

```

        for p = 1:4
            yy(p)=r_weight(p)*rgb_1D(y1+p-1,x1+k)/sum_y; %가중치*밝기값/가중치 합
        end
        Y(k+1) = sum(yy);
    end
    % Y(k+1)의 밝기값으로 가로방향 bicubic진행
    for k = 1:4
        X(k) = c_weight(k)*Y(k)/sum_x;
    end
    out(i,j)=sum(X);
end
end
out_rgb(:, :,rgb_level) = out;
end
%% plot
my_bic=uint8(out_rgb);
fprintf("구현 함수 Wn")
t = now-t
bic_psnr = psnr(real_bic, my_bic)
fprintf("내장 함수 Wn")
t = now;
built_in_bic = imresize(input, val, 'bicubic');
t = now-t
bic_psnr = psnr(real_bic, built_in_bic)
figure(1);
subplot(1,3,1); imshow(real_bic);title('원본','fontsize',20);
subplot(1,3,2); imshow(my_bic); title('구현 이미지','fontsize',20);
subplot(1,3,3); imshow(built_in_bic); title('imresize 이미지','fontsize',20);

function h = kernel1(x)
x = abs(x);
a=-0.75; % matlab과 opencv는 -0.75를 사용
if x<=1
    h = (a+2)*x^3 -(a+3)*x^2 +1;
elseif 1<x && x<=2
    h = a*x^3 -5*a*x^2 +8*a*x -4*a;
else
    h = 0;
end
end
end

```

원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



원본



구현 이미지



imresize 이미지



## Matlab-based implementation

### 2-4. Zero Padding

```
clc
clear all
close all

testImage = "[크기 변환]baby.png";
compareImage = "baby.png";
Ireference = imread(testImage);
Ireference = im2double(Ireference);
Icompare = imread(compareImage);
Icompare = im2double(Icompare);
figure(1); imshow(Ireference)

[m n refCh] = size(Ireference);
[p q comCh] = size(Icompare);

refF = fft2(Ireference);
paddingSize = 4;

upsampleF = zeros(p,q,comCh);
```

```

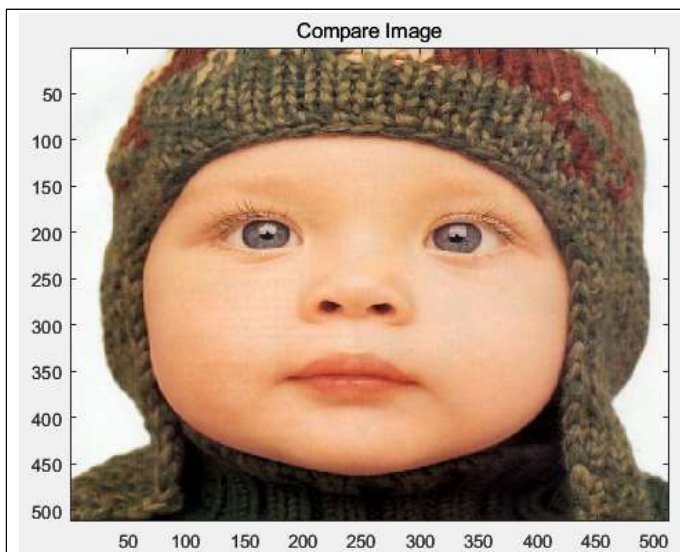
for i=1:comCh
    upsampleF(1:p/8, 1:q/8,i) = refF(1:m/2, 1:n/2,i);
    upsampleF(1:p/8, (q*7/8)+1:q,i) = refF(1:m/2, (n/2)+1:n,i);
    upsampleF((p*7/8)+1:p, 1:q/8,i) = refF((m/2)+1:m, 1:n/2,i);
    upsampleF((p*7/8)+1:p, (q*7/8)+1:q,i) = refF((m/2)+1:m, (n/2)+1:n,i);
end

upsampleF = upsampleF*(paddingSize^2);
lupsample = real(ifft2(upsampleF));
a = upsampleF(:,1);
figure(2); imagesc(lupsample);
title("Zero Padding Implementation")

figure(3); imagesc(lcompare);
title("Compare Image")

[psr nr] = psnr(lupsample, lcompare)

```






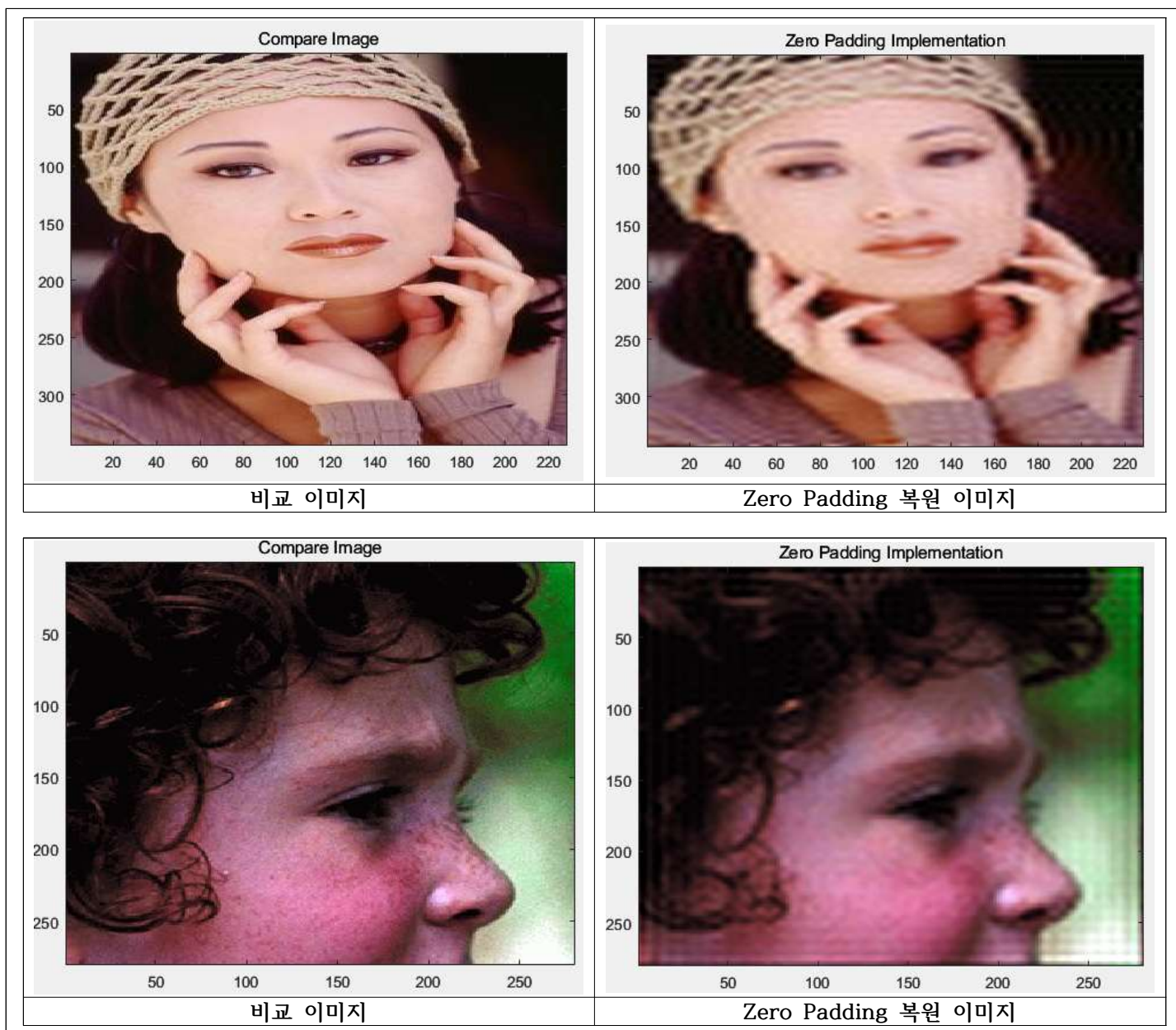
비교 이미지



Zero Padding 복원 이미지



<p>Compare Image</p> 	<p>Zero Padding Implementation</p> 
<p>비교 이미지</p>	<p>Zero Padding 복원 이미지</p>
<p>Compare Image</p> 	<p>Zero Padding Implementation</p> 
<p>비교 이미지</p>	<p>Zero Padding 복원 이미지</p>



## Matlab-based implementation

### 2-5. Deep Learning (VDSR 신경망)

```
clc
clear all
close all

load("trainedVDSRNet.mat");

testImage = "[크기변환]woman.png";
compareImage = "woman.png";
Ireference = imread(testImage);
Ireference = im2double(Ireference);
Icompare = imread(compareImage);
Icompare = im2double(Icompare);
figure(1); imshow(Icompare)
title("Compare Image")

[nrows,ncols,np] = size(Icompare);
Iycbcr = rgb2ycbcr(Ireference);
Iy = Iycbcr(:, :, 1);
Icb = Iycbcr(:, :, 2);
Icr = Iycbcr(:, :, 3);






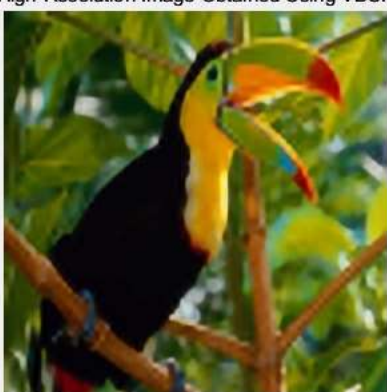

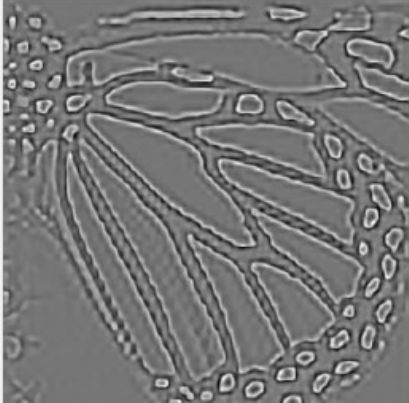

Iy_bicubic = imresize(Iy,[nrows ncols],"bicubic");
Icb_bicubic = imresize(Icb,[nrows ncols],"bicubic");
Icr_bicubic = imresize(Icr,[nrows ncols],"bicubic");

Iresidual = activations(net,Iy_bicubic,41);
Iresidual = double(Iresidual);
figure(2); imshow(Iresidual,[])
title("Residual Image from VDSR")

Isr = Iy_bicubic + Iresidual;
Ivdsr = ycbcr2rgb(cat(3,Isr,Icb_bicubic,Icr_bicubic));
figure(3); imshow(Ivdsr)
title("High-Resolution Image Obtained Using VDSR")

[psr nr] = psnr(Ivdsr, Icompare)
```



<p>Compare Image</p> 	<p>Residual Image from VDSR</p> 	<p>High-Resolution Image Obtained Using VDSR</p> 
<p>비교 이미지</p>	<p>잔차 영상</p>	<p>VDSR 결과</p>
<p>Compare Image</p> 	<p>Residual Image from VDSR</p> 	<p>High-Resolution Image Obtained Using VDSR</p> 
<p>비교 이미지</p>	<p>잔차 영상</p>	<p>VDSR 결과</p>
<p>Compare Image</p> 	<p>Residual Image from VDSR</p> 	<p>High-Resolution Image Obtained Using VDSR</p> 
<p>비교 이미지</p>	<p>잔차 영상</p>	<p>VDSR 결과</p>

<p>Compare Image</p>  <p>비교 이미지</p>	<p>Residual Image from VDSR</p>  <p>잔차 영상</p>	<p>High-Resolution Image Obtained Using VDSR</p>  <p>VDSR 결과</p>
<p>Compare Image</p>  <p>비교 이미지</p>	<p>Residual Image from VDSR</p>  <p>잔차 영상</p>	<p>High-Resolution Image Obtained Using VDSR</p>  <p>VDSR 결과</p>

## Result

	내장 NN PSNR	구현 NN PSNR	내장 처리 속도	구현 처리 속도
Baby	27.9239	27.9239	3.0082e-07	2.1991e-07
Bird	25.4156	25.4156	2.7777e-07	1.1583e-07
Butterfly	19.0549	19.0549	3.0093e-07	1.2736e-07
Head	27.9197	27.9197	2.6624e-07	1.0408e-07
Woman	23.0855	23.0855	2.6624e-07	1.1583e-07
평균	24.6799	24.6799	2.8240e-07	1.3660e-07

	내장 bilinear PSNR	구현 bilinear PSNR	내장 처리 속도	구현 처리 속도
Baby	29.4078	29.4040	3.1246e-07	1.2384e-06
Bird	26.8473	26.8449	4.6298e-07	6.2503e-07
Butterfly	19.9601	19.9594	4.5134e-07	5.0920e-07
Head	28.5458	28.5410	3.2410e-07	7.4075e-07
Woman	24.2150	24.2136	3.1258e-07	6.8278e-07
평균	25.7952	25.7926	3.7269e-07	7.5923e-07

	내장 bicubic PSNR	구현 bicubic PSNR	내장 처리 속도	구현 처리 속도
Baby	30.3700	30.5941	3.2398e-07	1.6748e-05
Bird	28.0513	28.3306	2.8941e-07	5.9027e-06
Butterfly	20.8895	21.0990	3.5879e-07	5.1736e-06
Head	28.9396	29.0054	3.7032e-07	5.5903e-06
Woman	25.1118	25.3306	3.0093e-07	5.9028e-06
평균	26.6724	26.8719	3.2869e-07	7.8635e-06

위 표는 공간영역에서 기존의 기법들을 사용하여 각 이미지를 확대한 결과를 나타낸다. NN은 구현이 가장 간단하였고 처리 속도도 빨랐지만 scale factor를 2로 해도 영상의 block 현상이 나타났고, factor값이 커질수록 block 현상도 심해졌다. 4개의 픽셀을 참조하는 bilinear는 nn보다 참조 픽셀의 수가 많은만큼 화질이 보다 깨끗한 결과가 나왔고 block현상이 두드러지지않는 결과가 나왔다. 16개의 픽셀을 참조하는 bicubic은 출력결과가 가장 좋았지만 참조 픽셀이 많아 연산시간이 오래걸리는 것을 확인할 수 있었다.

화질 Bicubic > Bilinear > NN  
연산시간 Bicubic > Bilinear > NN

	Zero Padding PSNR	처리 속도
Baby	24.9566	1.368367
Bird	22.7428	1.119336
Butterfly	17.8800	1.175267
Head	24.2303	1.083501
Woman	21.1322	1.205370
평균	22.1884	1.1904

	VDSR PSNR	처리 속도
Baby	31.2599	16.357200
Bird	29.1198	6.555174
Butterfly	22.8421	7.322813
Head	29.2564	6.238874
Woman	26.4978	6.435459
평균	27.7952	8.5819

위 표는 주파수 영역 및 딥러닝으로 구현한 Super-Resolution 결과이다. 주파수 Zero-Padding의 경우 컨볼루션 연산에 대한 처리 속도가 굉장히 빠르기 때문에 앞선 방식보다 매우 빨리 결과를 확인할 수 있었다. 하지만 화질 개선에 대한 품질은 조금 떨어지는 것을 확인할 수 있었다.

또한 VDSR 신경망을 이용한 딥러닝 방식은 조금 느린 처리 속도를 보여주지만 PSNR값이 1~3정도 더 높게 나오는 것을 확인할 수 있다.

화질 VDSR > Bicubic > Bilinear > NN > Zero Padding  
 연산시간 VDSR > Bicubic > Bilinear > NN > Zero Padding