Digital Image Processing

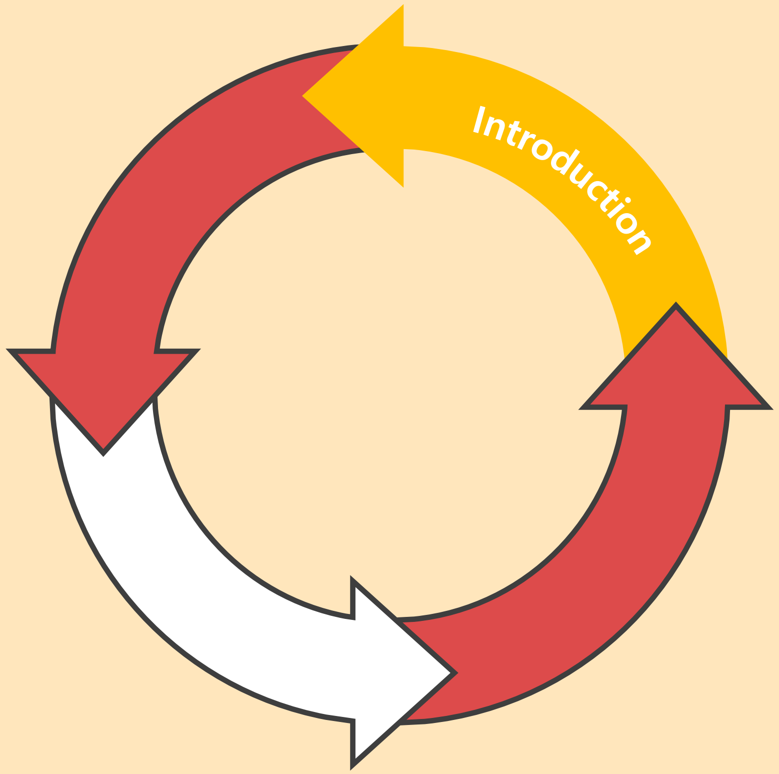
Term Project

지도교수 : 김성호

학과 : 전자공학과

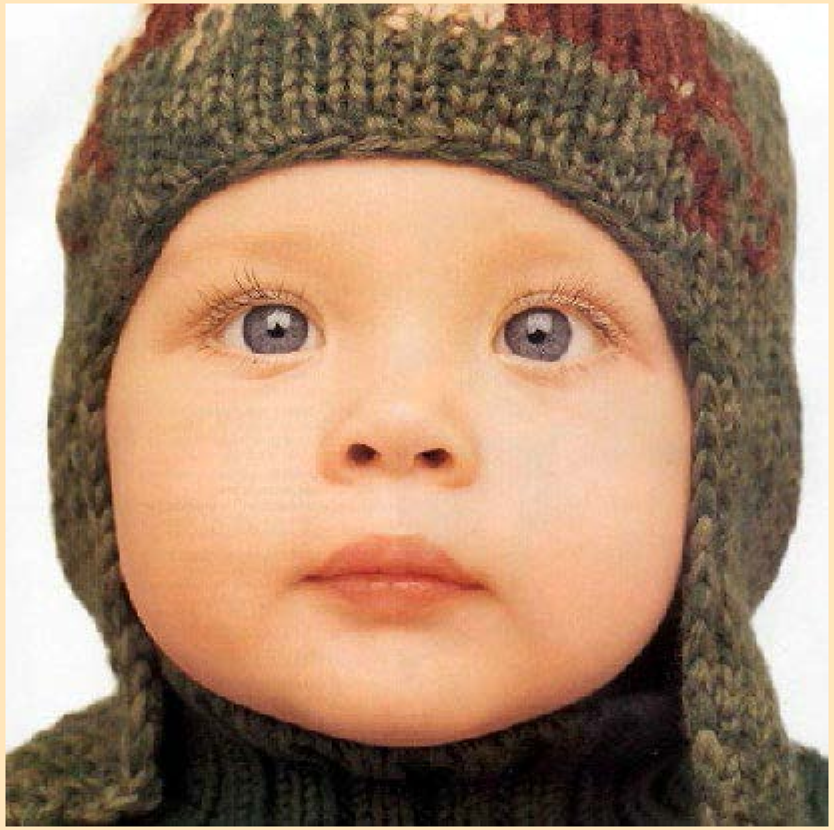
이름 : 유준상

Table of Contents



Introduction

SD 640x480



Full HD 1920x 1080

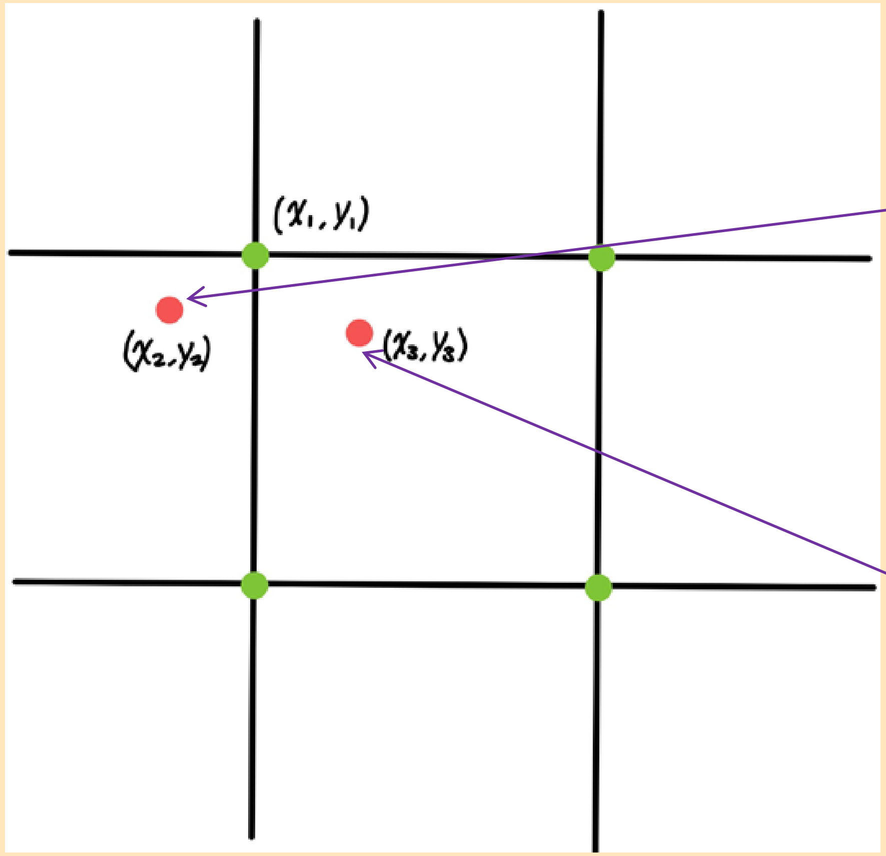
Image Super-Resolution

저해상도 이미지를 고해상도 이미지로 변환

- ◎ **Nearest Neighbor**
- ◎ **Bilinear interpolation**
- ◎ **Bicubic interpolation**
- ◎ **SRMDNF**

Survey related technologies

(1) NN(Nearest Neighbor)

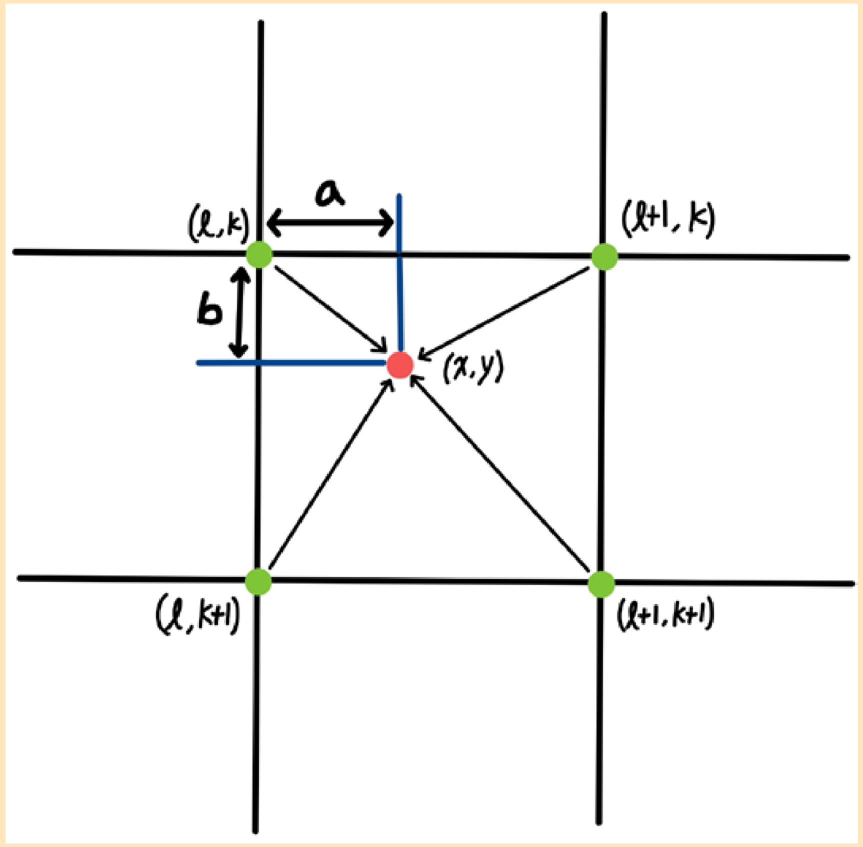


$$f_1(x_2, y_2) = f\big(round(x_2), round(y_2)\big) = f(x_1, y_1)$$

$$f_1(x_3, y_3) = f\big(round(x_3), round(y_3)\big) = f(x_1, y_1)$$

Survey related technologies

(2) Bilinear interpolation



$$l = floor(x), k = floor(y), a = x - l, b = y - k$$

$$f(x,y) = (1-a)*(1-b)*f(l,k) + a*(1-b)*f(l+1,k) \\ +(1-a)*b*f(l,k+1) + a*b*f(l+1,k+1)$$

Survey related technologies

(3) Bicubic interpolation

Cubic interpolation

1. $f(0,0) = p(0,0) = a_{00},$
2. $f(1,0) = p(1,0) = a_{00} + a_{10} + a_{20} + a_{30},$
3. $f(0,1) = p(0,1) = a_{00} + a_{01} + a_{02} + a_{03},$
4. $f(1,1) = p(1,1) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}.$

x축 cubic interpolation(1~4) / y축 cubic interpolation(5~8)

1. $f_x(0,0) = p_x(0,0) = a_{10},$
2. $f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30},$
3. $f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13},$
4. $f_x(1,1) = p_x(1,1) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i,$
5. $f_y(0,0) = p_y(0,0) = a_{01},$
6. $f_y(1,0) = p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31},$
7. $f_y(0,1) = p_y(0,1) = a_{01} + 2a_{02} + 3a_{03},$
8. $f_y(1,1) = p_y(1,1) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} j.$

xy 부분 미분

1. $f_{xy}(0,0) = p_{xy}(0,0) = a_{11},$
2. $f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31},$
3. $f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13},$
4. $f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} ij.$

Survey related technologies

(3) Bicubic interpolation

$$p(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}x^iy^j$$

$$p_x(x,y)=\sum_{i=1}^{3}\sum_{j=0}^{3}a_{ij}ix^{i-1}y^j,$$

$$p_y(x,y)=\sum_{i=0}^{3}\sum_{j=1}^{3}a_{ij}x^ijy^{j-1},$$

$$p_{xy}(x,y)=\sum_{i=1}^{3}\sum_{j=1}^{3}a_{ij}ix^{i-1}jy^{j-1}$$

$$A^{-1}=\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
-3&3&0&0&-2&-1&0&0&0&0&0&0&0&0&0&0\\
2&-2&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&-3&3&0&0&-2&-1&0&0\\
0&0&0&0&0&0&0&0&2&-2&0&0&1&1&0&0\\
-3&0&3&0&0&0&0&0&-2&0&-1&0&0&0&0&0\\
0&0&0&0&-3&0&3&0&0&0&0&0&-2&0&-1&0\\
9&-9&-9&9&6&3&-6&-3&6&-6&3&-3&4&2&2&1\\
-6&6&6&-6&-3&-3&3&3&-4&4&-2&2&-2&-2&-1&-1\\
2&0&-2&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&0&0&0&2&0&-2&0&0&0&0&0&1&0&1&0\\
-6&6&6&-6&-4&-2&4&2&-3&3&-3&3&-2&-1&-2&-1\\
4&-4&-4&4&2&2&-2&-2&2&-2&2&-2&1&1&1&1
\end{bmatrix}$$

$$A^{-1}x=a$$

$$p(x,y)=\begin{bmatrix}1 & x & x^2 & x^3\end{bmatrix}\begin{bmatrix}a_{00}&a_{01}&a_{02}&a_{03}\\a_{10}&a_{11}&a_{12}&a_{13}\\a_{20}&a_{21}&a_{22}&a_{23}\\a_{30}&a_{31}&a_{32}&a_{33}\end{bmatrix}\begin{bmatrix}1\\y\\y^2\\y^3\end{bmatrix}$$

Survey related technologies

(4) 평가 지표

PSNR Peak to Noise Ratio

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

ex) 8bit image : $MAX_I = 255 = 2^8 - 0$

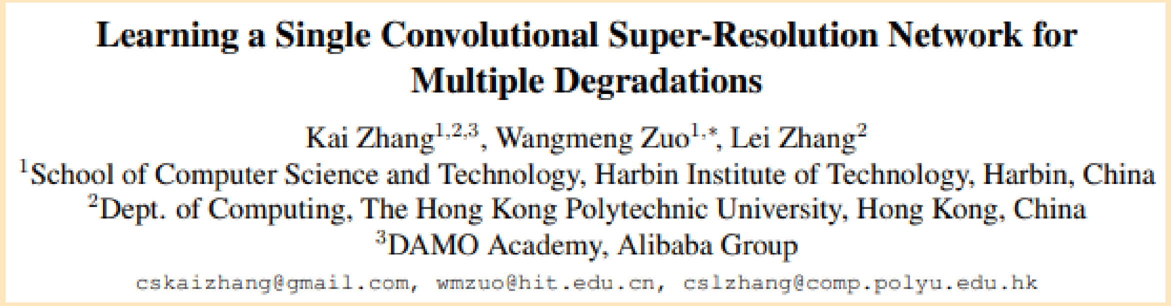
함수 처리 속도

프로그래밍을 할 때 성능과 함께 중요한 것.

Survey related technologies

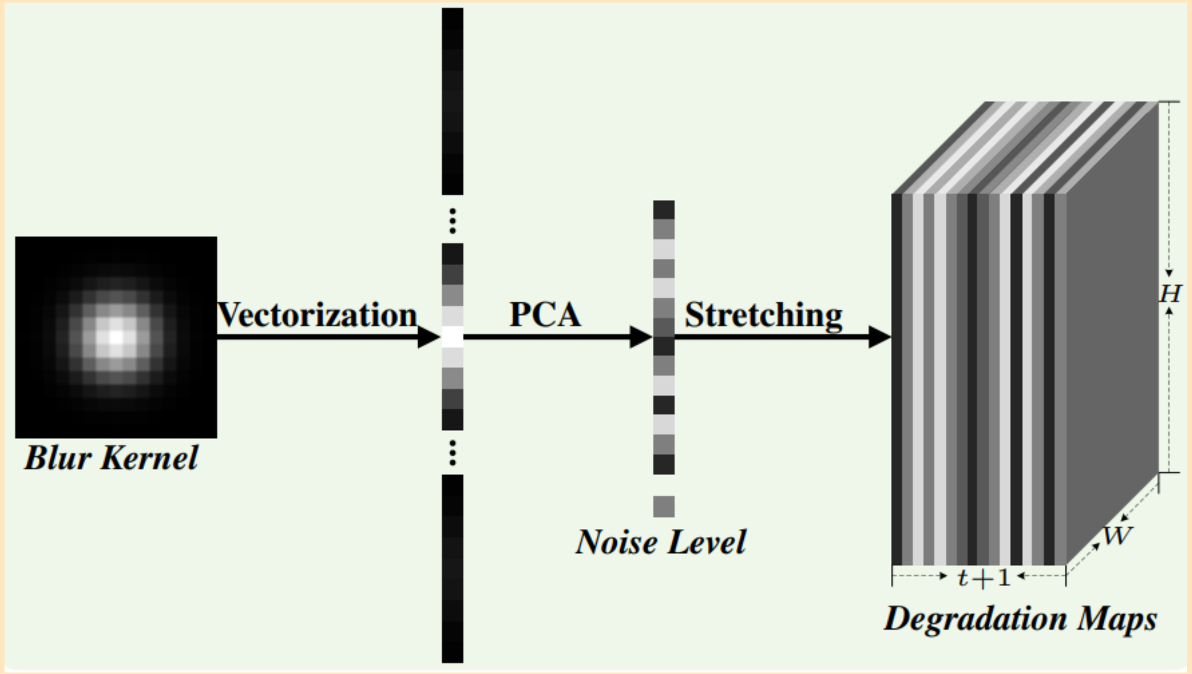
(5) 최신 기술 - SRMDNF

CVPR Conference on Computer Vision
and Pattern Recognition 2018



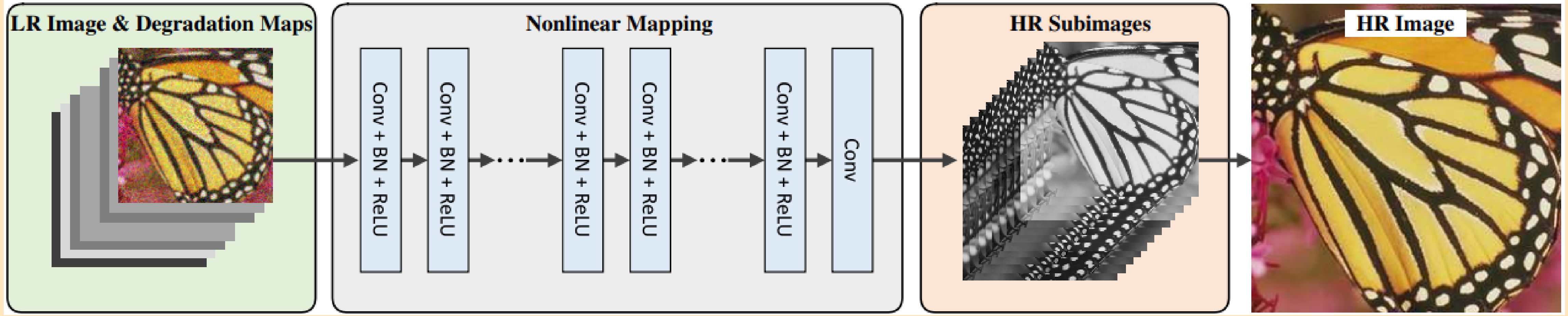
Learning a Single Convolutional Super-Resolution Network for Multiple Degradations

Kai Zhang[1,2,3], Wangmeng Zuo[1,*], Lei Zhang[2]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2]Dept. of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[3]DAMO Academy, Alibaba Group
cskaizhang@gmail.com, wmzuo@hit.edu.cn, cslzhang@comp.polyu.edu.hk



Survey related technologies

(5) 최신 기술 - SRMDNF



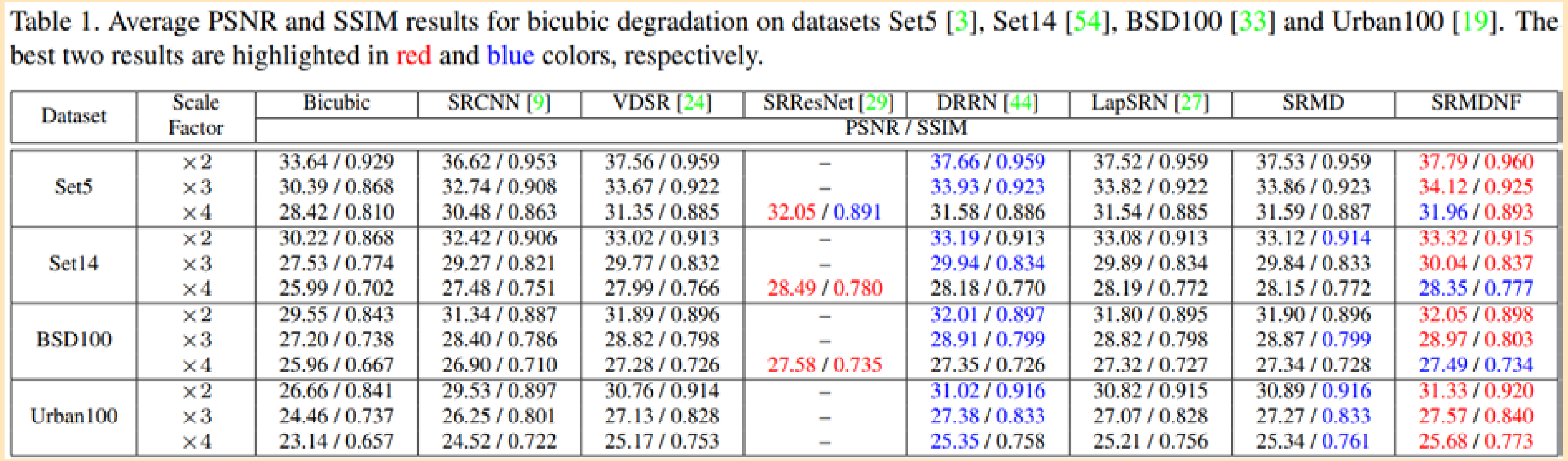


Table 1. Average PSNR and SSIM results for bicubic degradation on datasets Set5 [3], Set14 [54], BSD100 [33] and Urban100 [19]. The best two results are highlighted in red and blue colors, respectively.

Dataset	Scale Factor	Bicubic	SRCNN [9]	VDSR [24]	SRResNet [29]	DRRN [44]	LapSRN [27]	SRMD	SRMDNF
		PSNR / SSIM							
Set5	×2	33.64 / 0.929	36.62 / 0.953	37.56 / 0.959	–	37.66 / 0.959	37.52 / 0.959	37.53 / 0.959	37.79 / 0.960
	×3	30.39 / 0.868	32.74 / 0.908	33.67 / 0.922	–	33.93 / 0.923	33.82 / 0.922	33.86 / 0.923	34.12 / 0.925
	×4	28.42 / 0.810	30.48 / 0.863	31.35 / 0.885	32.05 / 0.891	31.58 / 0.886	31.54 / 0.885	31.59 / 0.887	31.96 / 0.893
Set14	×2	30.22 / 0.868	32.42 / 0.906	33.02 / 0.913	–	33.19 / 0.913	33.08 / 0.913	33.12 / 0.914	33.32 / 0.915
	×3	27.53 / 0.774	29.27 / 0.821	29.77 / 0.832	–	29.94 / 0.834	29.89 / 0.834	29.84 / 0.833	30.04 / 0.837
	×4	25.99 / 0.702	27.48 / 0.751	27.99 / 0.766	28.49 / 0.780	28.18 / 0.770	28.19 / 0.772	28.15 / 0.772	28.35 / 0.777
BSD100	×2	29.55 / 0.843	31.34 / 0.887	31.89 / 0.896	–	32.01 / 0.897	31.80 / 0.895	31.90 / 0.896	32.05 / 0.898
	×3	27.20 / 0.738	28.40 / 0.786	28.82 / 0.798	–	28.91 / 0.799	28.82 / 0.798	28.87 / 0.799	28.97 / 0.803
	×4	25.96 / 0.667	26.90 / 0.710	27.28 / 0.726	27.58 / 0.735	27.35 / 0.726	27.32 / 0.727	27.34 / 0.728	27.49 / 0.734
Urban100	×2	26.66 / 0.841	29.53 / 0.897	30.76 / 0.914	–	31.02 / 0.916	30.82 / 0.915	30.89 / 0.916	31.33 / 0.920
	×3	24.46 / 0.737	26.25 / 0.801	27.13 / 0.828	–	27.38 / 0.833	27.07 / 0.828	27.27 / 0.833	27.57 / 0.840
	×4	23.14 / 0.657	24.52 / 0.722	25.17 / 0.753	–	25.35 / 0.758	25.21 / 0.756	25.34 / 0.761	25.68 / 0.773

Matlab-based implementation

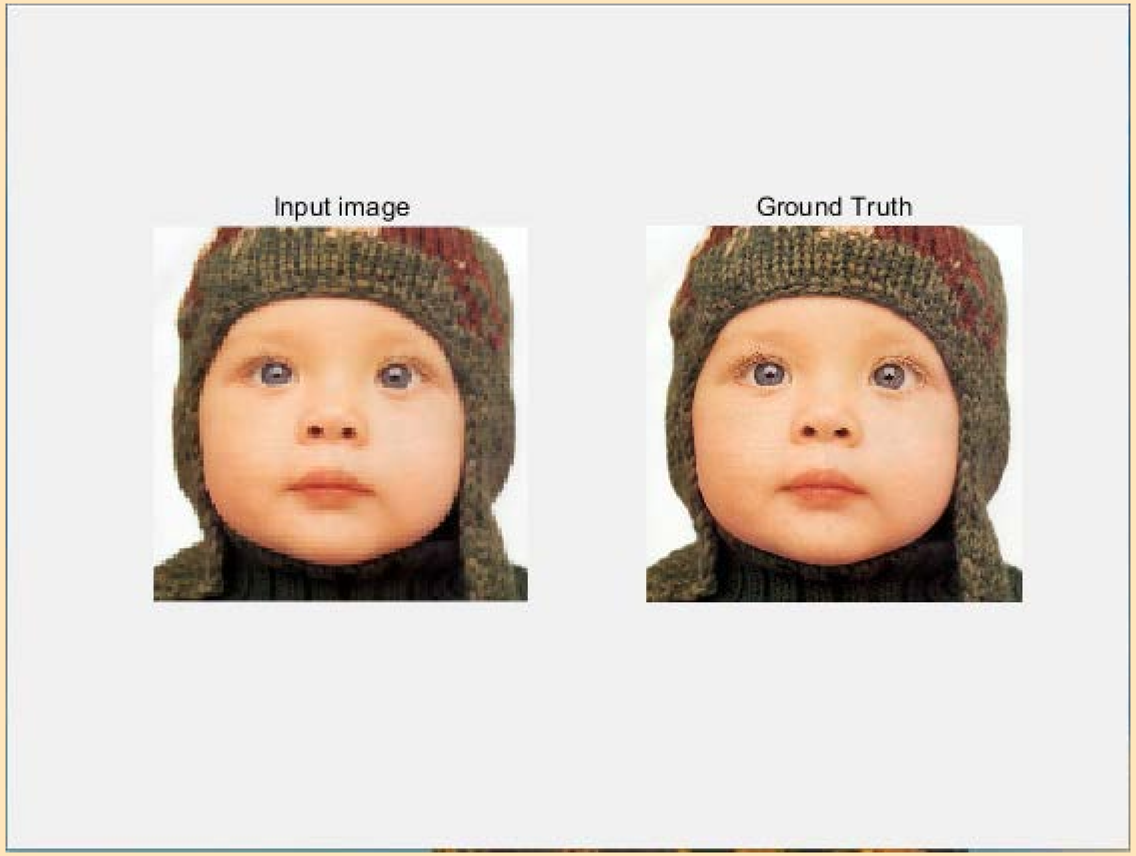
Nearest Neighbor,
Bilinear interpolation,
Bicubic interpolation

① Code

② Result image

③ PSNR 및 처리속도

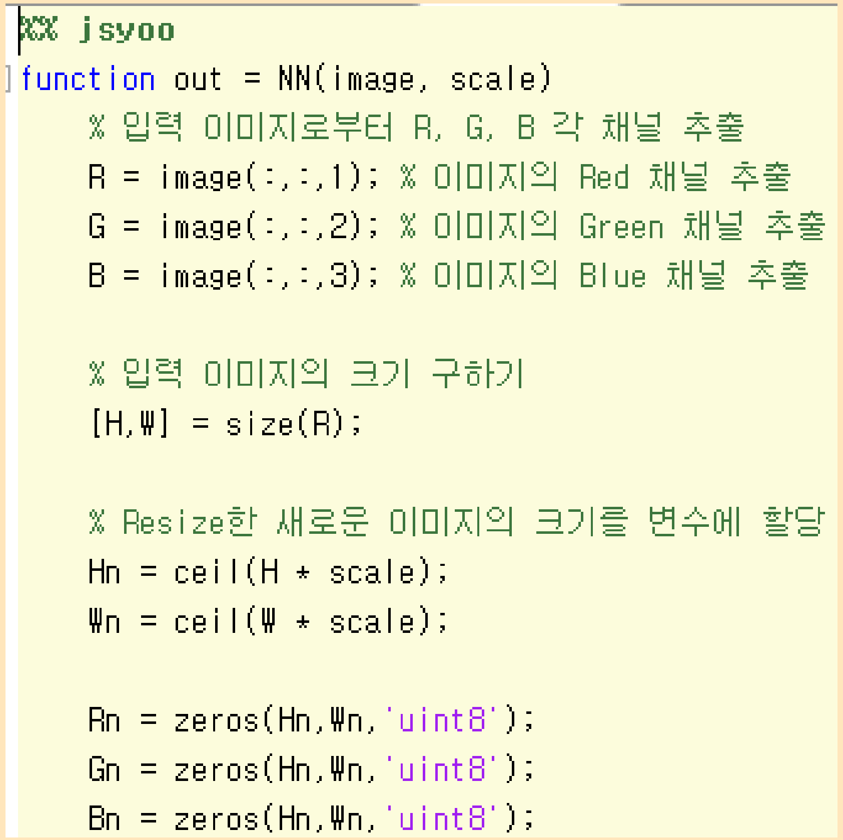
1. small(input) / Ground Truth



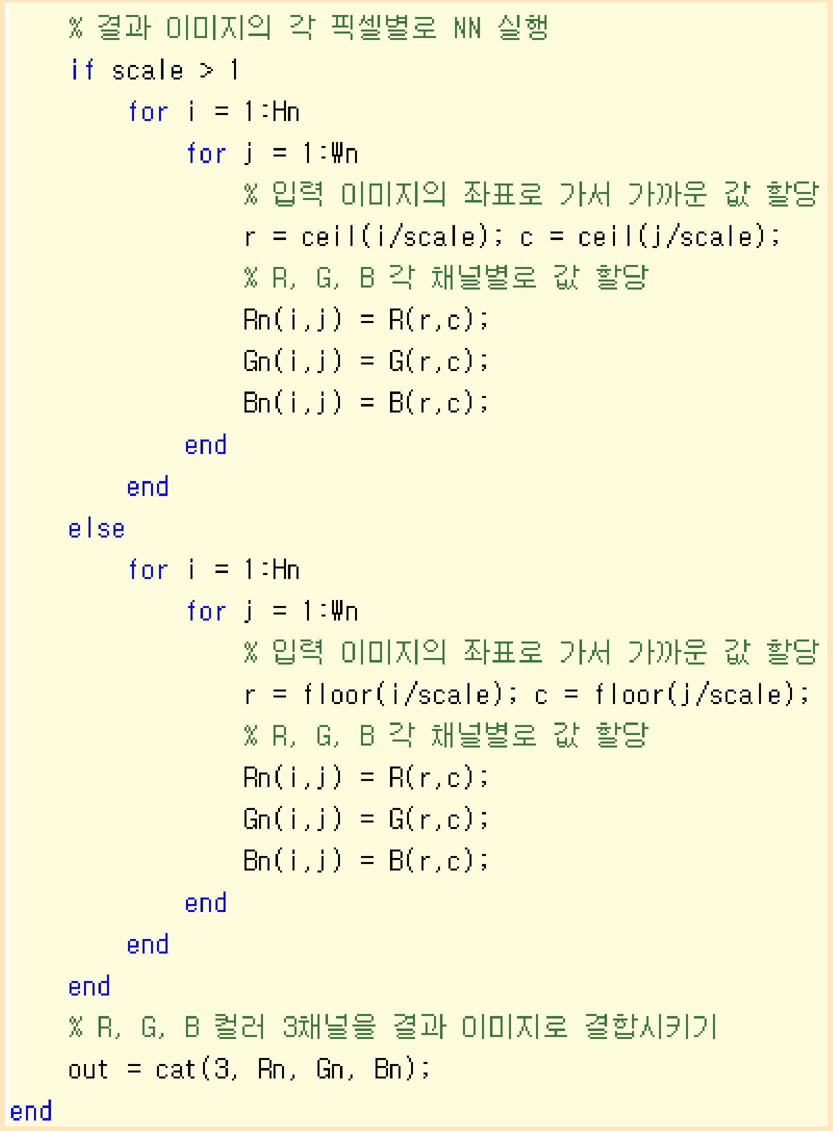
Matlab-based implementation

(1) Nearest Neighbor

① Code



```
%% jsyoo
function out = NN(image, scale)
    % 입력 이미지로부터 R, G, B 각 채널 추출
    R = image(:,:,1); % 이미지의 Red 채널 추출
    G = image(:,:,2); % 이미지의 Green 채널 추출
    B = image(:,:,3); % 이미지의 Blue 채널 추출

    % 입력 이미지의 크기 구하기
    [H,W] = size(R);

    % Resize한 새로운 이미지의 크기를 변수에 할당
    Hn = ceil(H * scale);
    Wn = ceil(W * scale);

    Rn = zeros(Hn,Wn,'uint8');
    Gn = zeros(Hn,Wn,'uint8');
    Bn = zeros(Hn,Wn,'uint8');
```



```
    % 결과 이미지의 각 픽셀별로 NN 실행
    if scale > 1
        for i = 1:Hn
            for j = 1:Wn
                % 입력 이미지의 좌표로 가서 가까운 값 할당
                r = ceil(i/scale); c = ceil(j/scale);
                % R, G, B 각 채널별로 값 할당
                Rn(i,j) = R(r,c);
                Gn(i,j) = G(r,c);
                Bn(i,j) = B(r,c);
            end
        end
    else
        for i = 1:Hn
            for j = 1:Wn
                % 입력 이미지의 좌표로 가서 가까운 값 할당
                r = floor(i/scale); c = floor(j/scale);
                % R, G, B 각 채널별로 값 할당
                Rn(i,j) = R(r,c);
                Gn(i,j) = G(r,c);
                Bn(i,j) = B(r,c);
            end
        end
    end
    % R, G, B 컬러 3채널을 결과 이미지로 결합시키기
    out = cat(3, Rn, Gn, Bn);
end
```

Matlab-based implementation

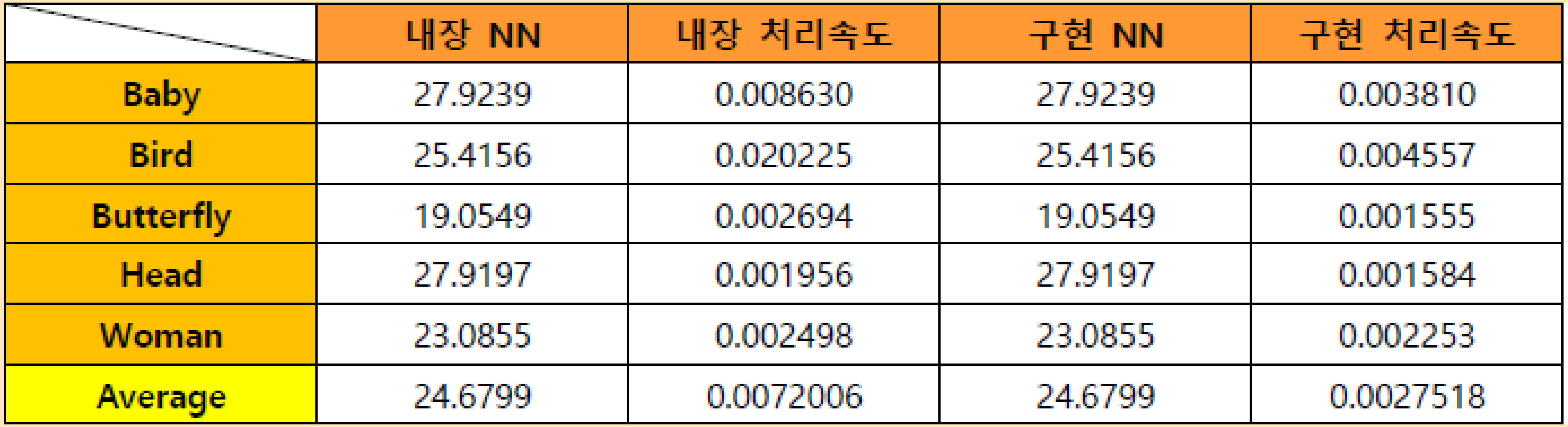
(1) Nearest Neighbor

② Result image

Matlab-based implementation

(1) Nearest Neighbor

③ PSNR 및 처리속도



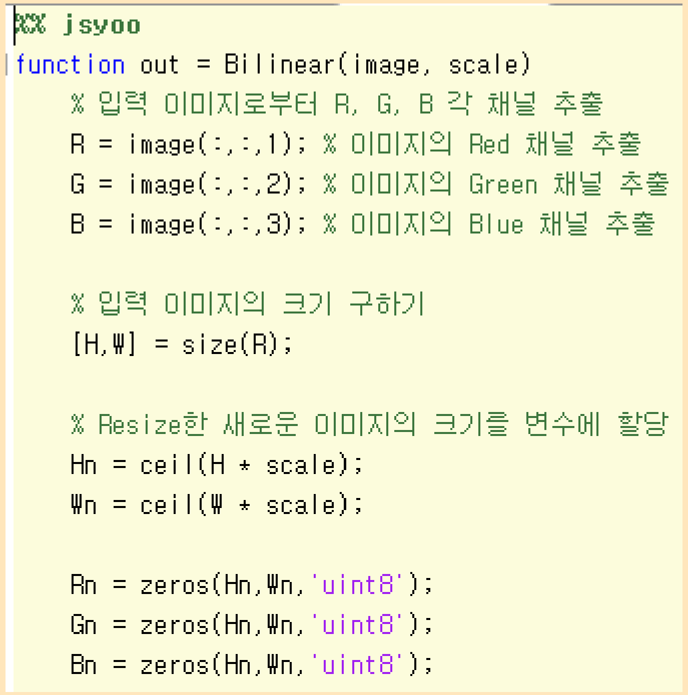
	내장 NN	내장 처리속도	구현 NN	구현 처리속도
Baby	27.9239	0.008630	27.9239	0.003810
Bird	25.4156	0.020225	25.4156	0.004557
Butterfly	19.0549	0.002694	19.0549	0.001555
Head	27.9197	0.001956	27.9197	0.001584
Woman	23.0855	0.002498	23.0855	0.002253
Average	24.6799	0.0072006	24.6799	0.0027518

➔ PSNR : 내장 = 구현 // 처리속도 : 내장 < 구현

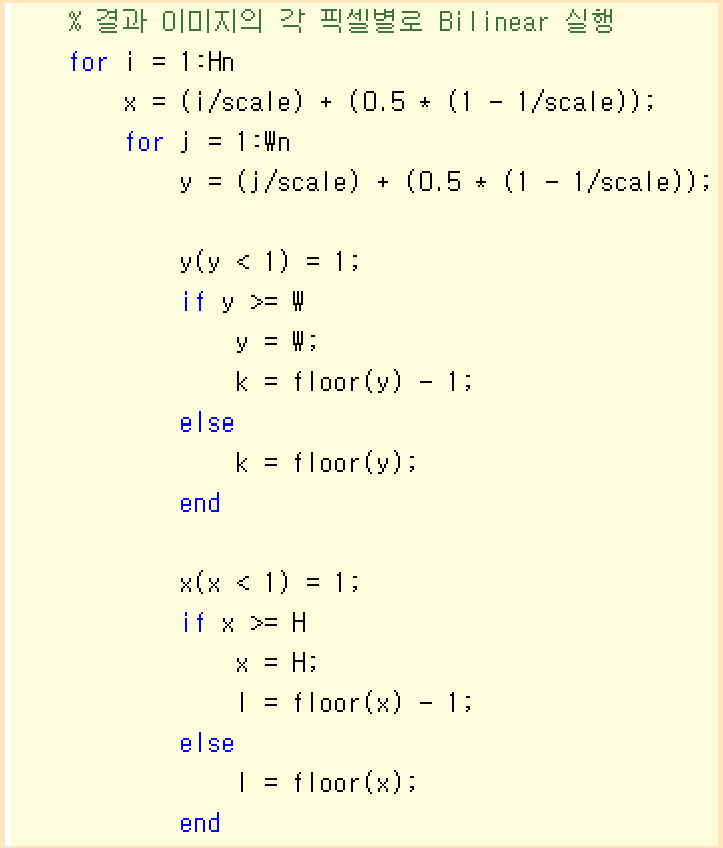
Matlab-based implementation

(2) Bilinear interpolation

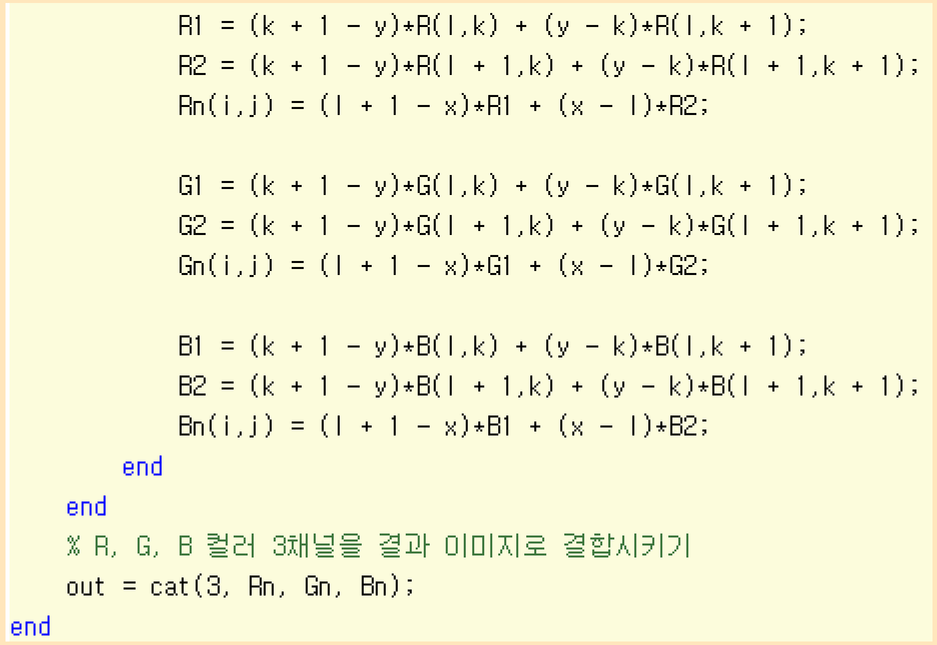
① Code



```matlab
%% jsyoo
function out = Bilinear(image, scale)
    % 입력 이미지로부터 R, G, B 각 채널 추출
    R = image(:,:,1); % 이미지의 Red 채널 추출
    G = image(:,:,2); % 이미지의 Green 채널 추출
    B = image(:,:,3); % 이미지의 Blue 채널 추출

    % 입력 이미지의 크기 구하기
    [H,W] = size(R);

    % Resize한 새로운 이미지의 크기를 변수에 할당
    Hn = ceil(H * scale);
    Wn = ceil(W * scale);

    Rn = zeros(Hn,Wn,'uint8');
    Gn = zeros(Hn,Wn,'uint8');
    Bn = zeros(Hn,Wn,'uint8');
```



```matlab
    % 결과 이미지의 각 픽셀별로 Bilinear 실행
    for i = 1:Hn
        x = (i/scale) + (0.5 * (1 - 1/scale));
        for j = 1:Wn
            y = (j/scale) + (0.5 * (1 - 1/scale));

            y(y < 1) = 1;
            if y >= W
                y = W;
                k = floor(y) - 1;
            else
                k = floor(y);
            end

            x(x < 1) = 1;
            if x >= H
                x = H;
                l = floor(x) - 1;
            else
                l = floor(x);
            end
```



```matlab
            R1 = (k + 1 - y)*R(l,k) + (y - k)*R(l,k + 1);
            R2 = (k + 1 - y)*R(l + 1,k) + (y - k)*R(l + 1,k + 1);
            Rn(i,j) = (l + 1 - x)*R1 + (x - l)*R2;

            G1 = (k + 1 - y)*G(l,k) + (y - k)*G(l,k + 1);
            G2 = (k + 1 - y)*G(l + 1,k) + (y - k)*G(l + 1,k + 1);
            Gn(i,j) = (l + 1 - x)*G1 + (x - l)*G2;

            B1 = (k + 1 - y)*B(l,k) + (y - k)*B(l,k + 1);
            B2 = (k + 1 - y)*B(l + 1,k) + (y - k)*B(l + 1,k + 1);
            Bn(i,j) = (l + 1 - x)*B1 + (x - l)*B2;
        end
    end
    % R, G, B 컬러 3채널을 결과 이미지로 결합시키기
    out = cat(3, Rn, Gn, Bn);
end
```

Matlab-based implementation

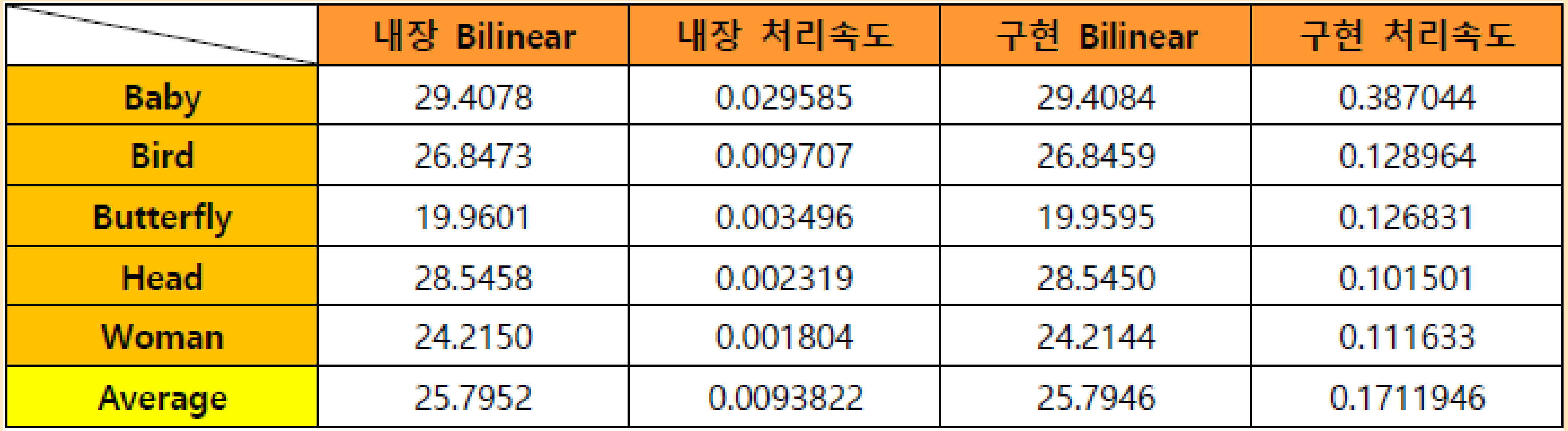
(2) Bilinear interpolation

② Result image

Matlab-based implementation

(2) Bilinear interpolation

③ PSNR 및 처리속도



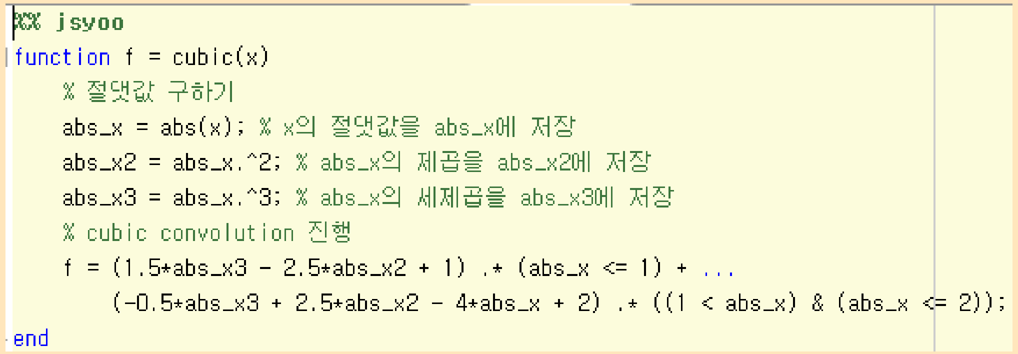
	내장 Bilinear	내장 처리속도	구현 Bilinear	구현 처리속도
Baby	29.4078	0.029585	29.4084	0.387044
Bird	26.8473	0.009707	26.8459	0.128964
Butterfly	19.9601	0.003496	19.9595	0.126831
Head	28.5458	0.002319	28.5450	0.101501
Woman	24.2150	0.001804	24.2144	0.111633
Average	25.7952	0.0093822	25.7946	0.1711946

➔ PSNR : 내장 > 구현 // 처리속도 : 내장 > 구현

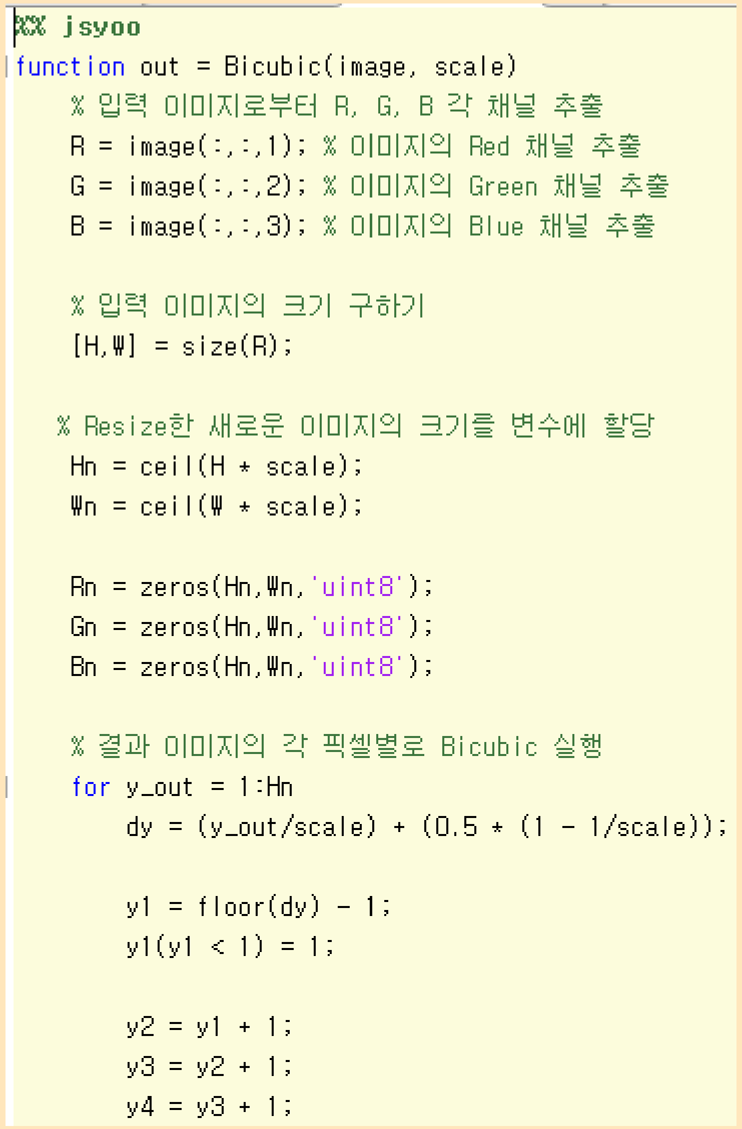
Matlab-based implementation

(3) Bicubic interpolation

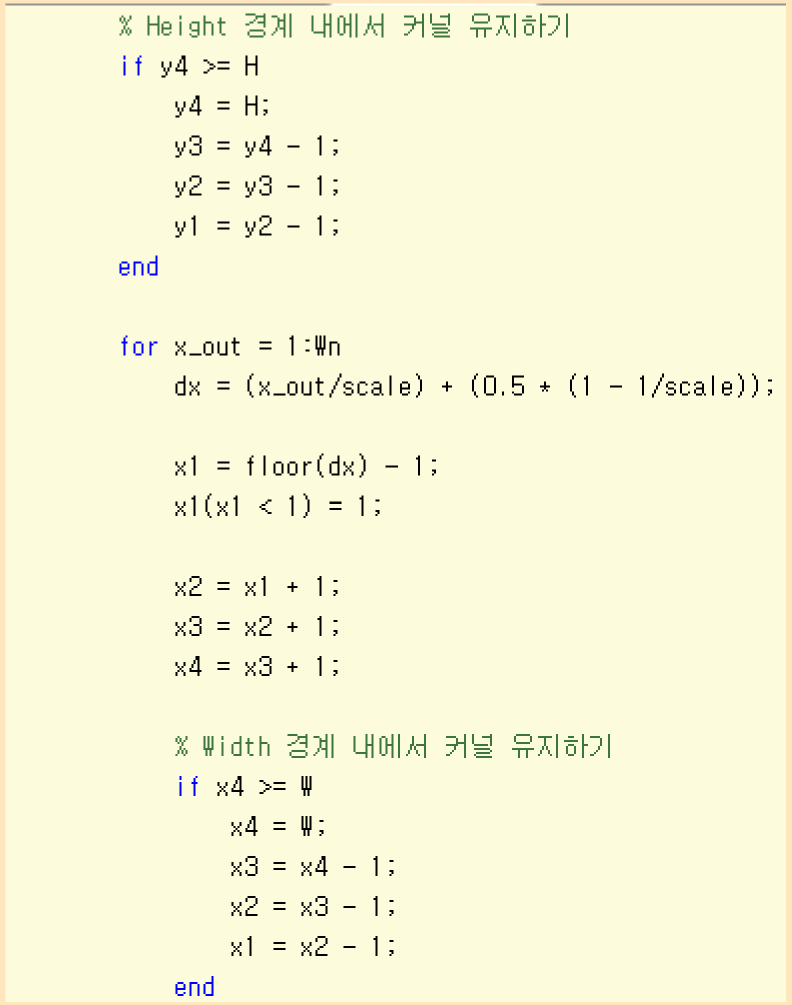
① Code



```matlab
%% jsyoo
function f = cubic(x)
    % 절댓값 구하기
    abs_x = abs(x); % x의 절댓값을 abs_x에 저장
    abs_x2 = abs_x.^2; % abs_x의 제곱을 abs_x2에 저장
    abs_x3 = abs_x.^3; % abs_x의 세제곱을 abs_x3에 저장
    % cubic convolution 진행
    f = (1.5*abs_x3 - 2.5*abs_x2 + 1) .* (abs_x <= 1) + ...
        (-0.5*abs_x3 + 2.5*abs_x2 - 4*abs_x + 2) .* ((1 < abs_x) & (abs_x <= 2));
end
```



```matlab
%% jsyoo
function out = Bicubic(image, scale)
    % 입력 이미지로부터 R, G, B 각 채널 추출
    R = image(:,:,1); % 이미지의 Red 채널 추출
    G = image(:,:,2); % 이미지의 Green 채널 추출
    B = image(:,:,3); % 이미지의 Blue 채널 추출

    % 입력 이미지의 크기 구하기
    [H,W] = size(R);

   % Resize한 새로운 이미지의 크기를 변수에 할당
    Hn = ceil(H * scale);
    Wn = ceil(W * scale);

    Rn = zeros(Hn,Wn,'uint8');
    Gn = zeros(Hn,Wn,'uint8');
    Bn = zeros(Hn,Wn,'uint8');

    % 결과 이미지의 각 픽셀별로 Bicubic 실행
    for y_out = 1:Hn
        dy = (y_out/scale) + (0.5 * (1 - 1/scale));

        y1 = floor(dy) - 1;
        y1(y1 < 1) = 1;

        y2 = y1 + 1;
        y3 = y2 + 1;
        y4 = y3 + 1;
```

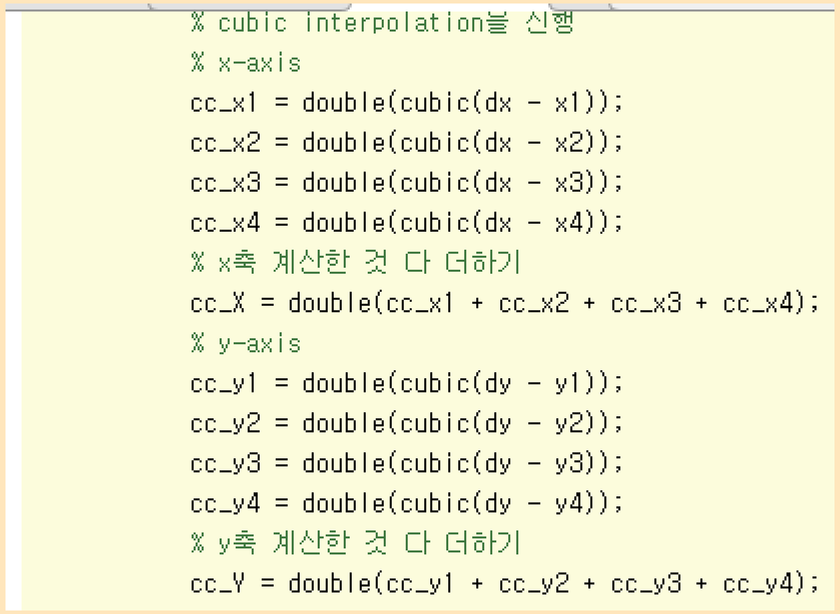


```matlab
        % Height 경계 내에서 커널 유지하기
        if y4 >= H
            y4 = H;
            y3 = y4 - 1;
            y2 = y3 - 1;
            y1 = y2 - 1;
        end

        for x_out = 1:Wn
            dx = (x_out/scale) + (0.5 * (1 - 1/scale));

            x1 = floor(dx) - 1;
            x1(x1 < 1) = 1;

            x2 = x1 + 1;
            x3 = x2 + 1;
            x4 = x3 + 1;

            % Width 경계 내에서 커널 유지하기
            if x4 >= W
                x4 = W;
                x3 = x4 - 1;
                x2 = x3 - 1;
                x1 = x2 - 1;
            end
```

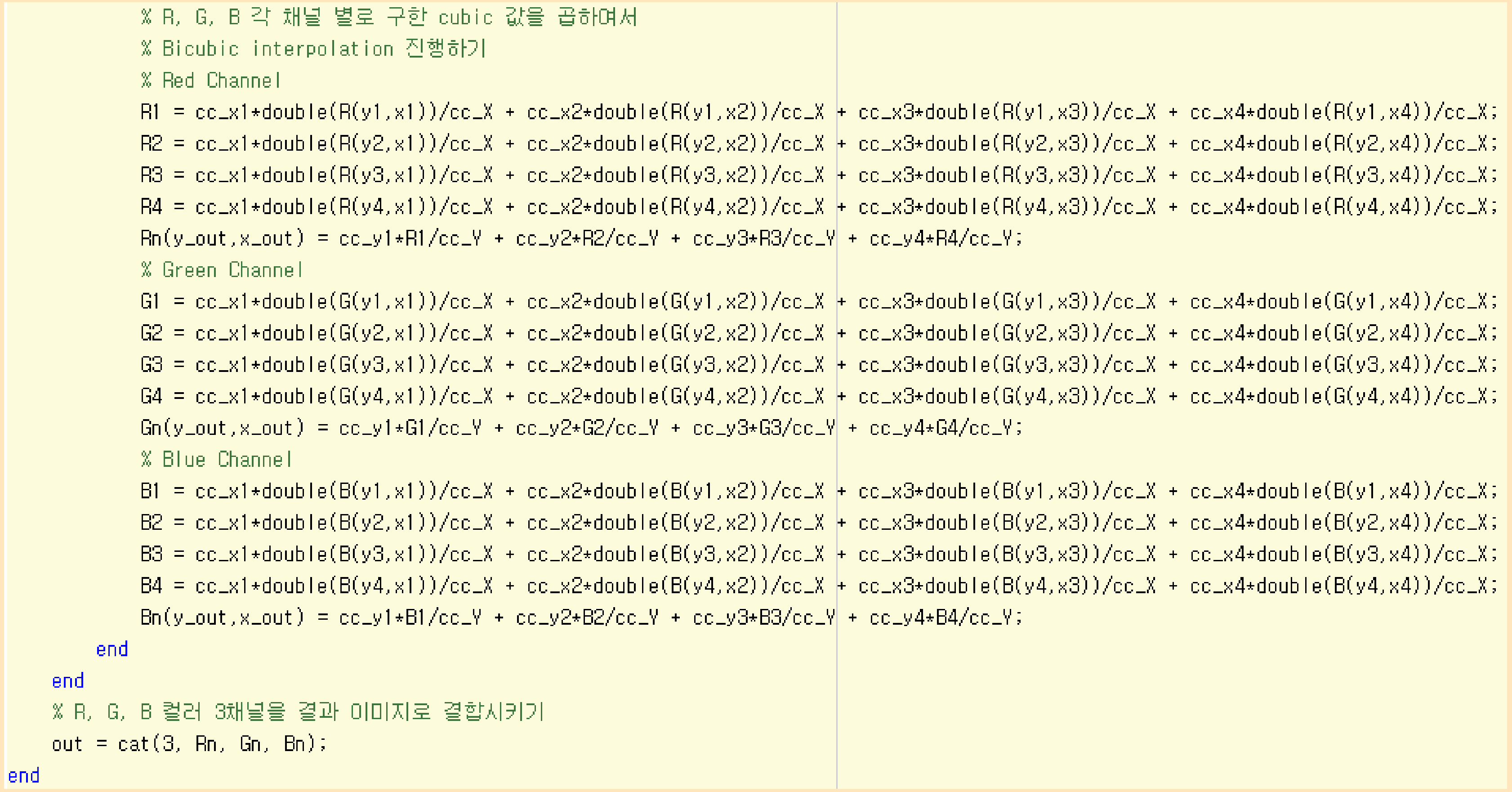
Matlab-based implementation

(3) Bicubic interpolation

① Code



```
% cubic interpolation을 진행
% x-axis
cc_x1 = double(cubic(dx - x1));
cc_x2 = double(cubic(dx - x2));
cc_x3 = double(cubic(dx - x3));
cc_x4 = double(cubic(dx - x4));
% x축 계산한 것 다 더하기
cc_X = double(cc_x1 + cc_x2 + cc_x3 + cc_x4);
% y-axis
cc_y1 = double(cubic(dy - y1));
cc_y2 = double(cubic(dy - y2));
cc_y3 = double(cubic(dy - y3));
cc_y4 = double(cubic(dy - y4));
% y축 계산한 것 다 더하기
cc_Y = double(cc_y1 + cc_y2 + cc_y3 + cc_y4);
```



```
            % R, G, B 각 채널 별로 구한 cubic 값을 곱하여서
            % Bicubic interpolation 진행하기
            % Red Channel
            R1 = cc_x1*double(R(y1,x1))/cc_X + cc_x2*double(R(y1,x2))/cc_X + cc_x3*double(R(y1,x3))/cc_X + cc_x4*double(R(y1,x4))/cc_X;
            R2 = cc_x1*double(R(y2,x1))/cc_X + cc_x2*double(R(y2,x2))/cc_X + cc_x3*double(R(y2,x3))/cc_X + cc_x4*double(R(y2,x4))/cc_X;
            R3 = cc_x1*double(R(y3,x1))/cc_X + cc_x2*double(R(y3,x2))/cc_X + cc_x3*double(R(y3,x3))/cc_X + cc_x4*double(R(y3,x4))/cc_X;
            R4 = cc_x1*double(R(y4,x1))/cc_X + cc_x2*double(R(y4,x2))/cc_X + cc_x3*double(R(y4,x3))/cc_X + cc_x4*double(R(y4,x4))/cc_X;
            Rn(y_out,x_out) = cc_y1*R1/cc_Y + cc_y2*R2/cc_Y + cc_y3*R3/cc_Y + cc_y4*R4/cc_Y;
            % Green Channel
            G1 = cc_x1*double(G(y1,x1))/cc_X + cc_x2*double(G(y1,x2))/cc_X + cc_x3*double(G(y1,x3))/cc_X + cc_x4*double(G(y1,x4))/cc_X;
            G2 = cc_x1*double(G(y2,x1))/cc_X + cc_x2*double(G(y2,x2))/cc_X + cc_x3*double(G(y2,x3))/cc_X + cc_x4*double(G(y2,x4))/cc_X;
            G3 = cc_x1*double(G(y3,x1))/cc_X + cc_x2*double(G(y3,x2))/cc_X + cc_x3*double(G(y3,x3))/cc_X + cc_x4*double(G(y3,x4))/cc_X;
            G4 = cc_x1*double(G(y4,x1))/cc_X + cc_x2*double(G(y4,x2))/cc_X + cc_x3*double(G(y4,x3))/cc_X + cc_x4*double(G(y4,x4))/cc_X;
            Gn(y_out,x_out) = cc_y1*G1/cc_Y + cc_y2*G2/cc_Y + cc_y3*G3/cc_Y + cc_y4*G4/cc_Y;
            % Blue Channel
            B1 = cc_x1*double(B(y1,x1))/cc_X + cc_x2*double(B(y1,x2))/cc_X + cc_x3*double(B(y1,x3))/cc_X + cc_x4*double(B(y1,x4))/cc_X;
            B2 = cc_x1*double(B(y2,x1))/cc_X + cc_x2*double(B(y2,x2))/cc_X + cc_x3*double(B(y2,x3))/cc_X + cc_x4*double(B(y2,x4))/cc_X;
            B3 = cc_x1*double(B(y3,x1))/cc_X + cc_x2*double(B(y3,x2))/cc_X + cc_x3*double(B(y3,x3))/cc_X + cc_x4*double(B(y3,x4))/cc_X;
            B4 = cc_x1*double(B(y4,x1))/cc_X + cc_x2*double(B(y4,x2))/cc_X + cc_x3*double(B(y4,x3))/cc_X + cc_x4*double(B(y4,x4))/cc_X;
            Bn(y_out,x_out) = cc_y1*B1/cc_Y + cc_y2*B2/cc_Y + cc_y3*B3/cc_Y + cc_y4*B4/cc_Y;
        end
    end
    % R, G, B 컬러 3채널을 결과 이미지로 결합시키기
    out = cat(3, Rn, Gn, Bn);
end
```

Matlab-based implementation

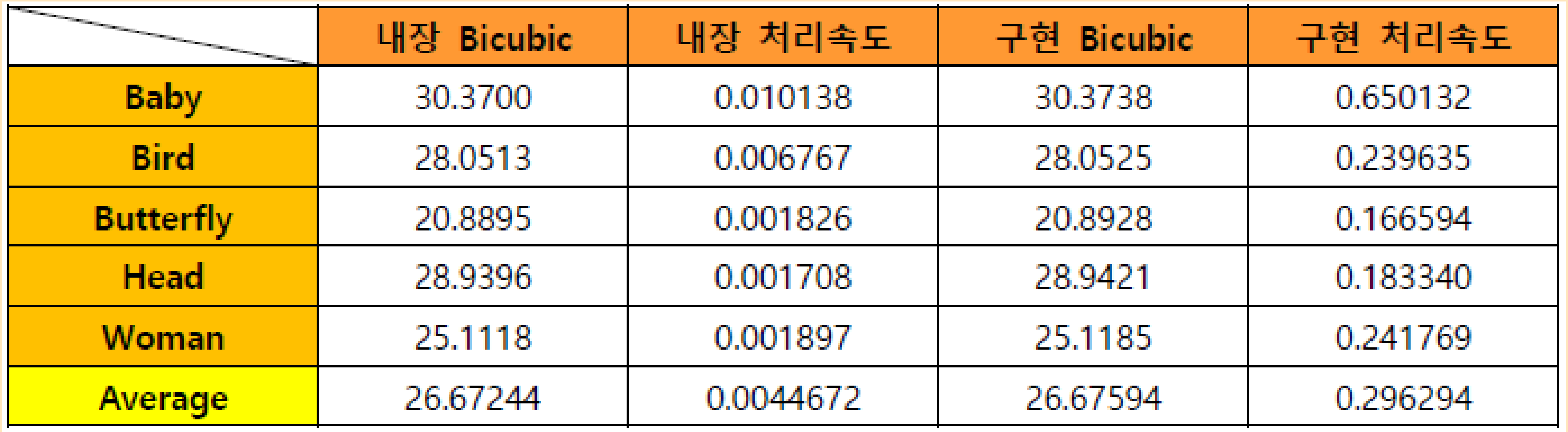
(3) Bicubic interpolation

② Result image

Matlab-based implementation

(3) Bicubic interpolation

③ PSNR 및 처리속도

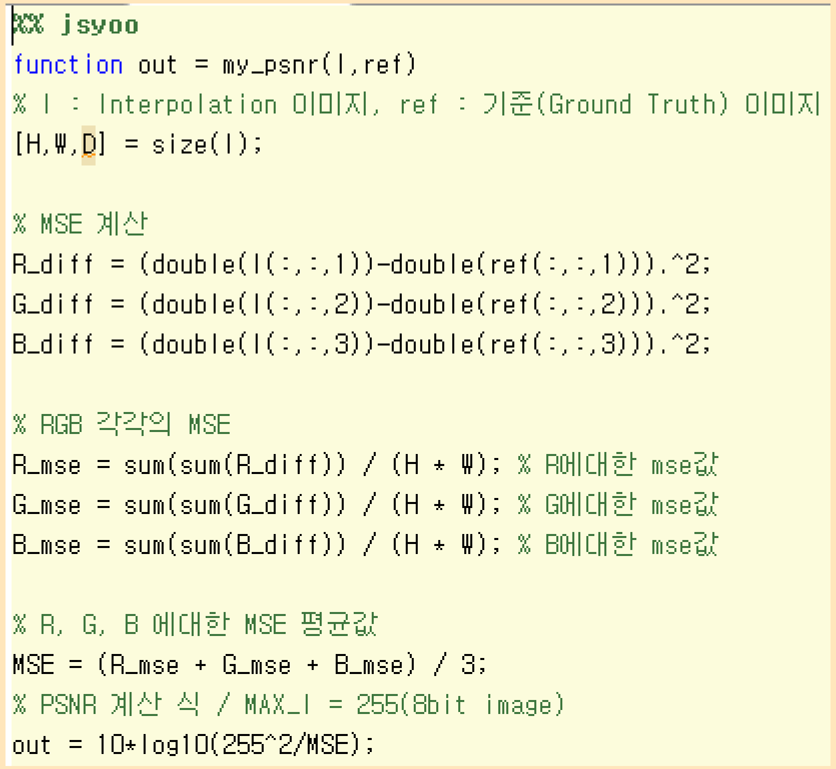


	내장 Bicubic	내장 처리속도	구현 Bicubic	구현 처리속도
Baby	30.3700	0.010138	30.3738	0.650132
Bird	28.0513	0.006767	28.0525	0.239635
Butterfly	20.8895	0.001826	20.8928	0.166594
Head	28.9396	0.001708	28.9421	0.183340
Woman	25.1118	0.001897	25.1185	0.241769
Average	26.67244	0.0044672	26.67594	0.296294

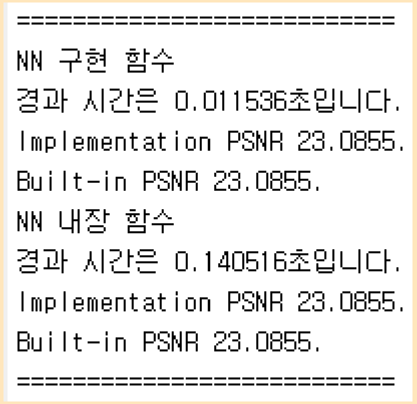
➔ PSNR : 내장 < 구현 // 처리속도 : 내장 > 구현

Matlab-based implementation

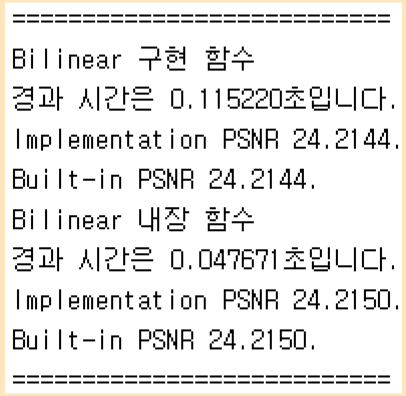
★ PSNR



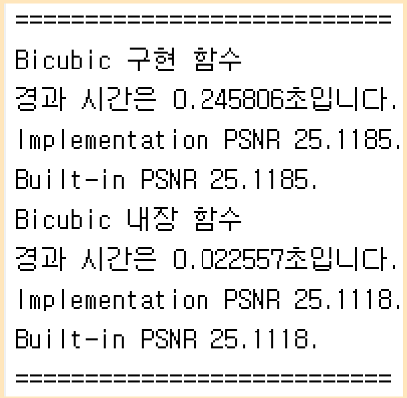
```
%% jsyoo
function out = my_psnr(I,ref)
% I : Interpolation 이미지, ref : 기준(Ground Truth) 이미지
[H,W,D] = size(I);

% MSE 계산
R_diff = (double(I(:,:,1))-double(ref(:,:,1))).^2;
G_diff = (double(I(:,:,2))-double(ref(:,:,2))).^2;
B_diff = (double(I(:,:,3))-double(ref(:,:,3))).^2;

% RGB 각각의 MSE
R_mse = sum(sum(R_diff)) / (H * W); % R에대한 mse값
G_mse = sum(sum(G_diff)) / (H * W); % G에대한 mse값
B_mse = sum(sum(B_diff)) / (H * W); % B에대한 mse값

% R, G, B 에대한 MSE 평균값
MSE = (R_mse + G_mse + B_mse) / 3;
% PSNR 계산 식 / MAX_I = 255(8bit image)
out = 10*log10(255^2/MSE);
```



```
============================
NN 구현 함수
경과 시간은 0.011536초입니다.
Implementation PSNR 23.0855.
Built-in PSNR 23.0855.
NN 내장 함수
경과 시간은 0.140516초입니다.
Implementation PSNR 23.0855.
Built-in PSNR 23.0855.
============================
```



```
============================
Bilinear 구현 함수
경과 시간은 0.115220초입니다.
Implementation PSNR 24.2144.
Built-in PSNR 24.2144.
Bilinear 내장 함수
경과 시간은 0.047671초입니다.
Implementation PSNR 24.2150.
Built-in PSNR 24.2150.
============================
```



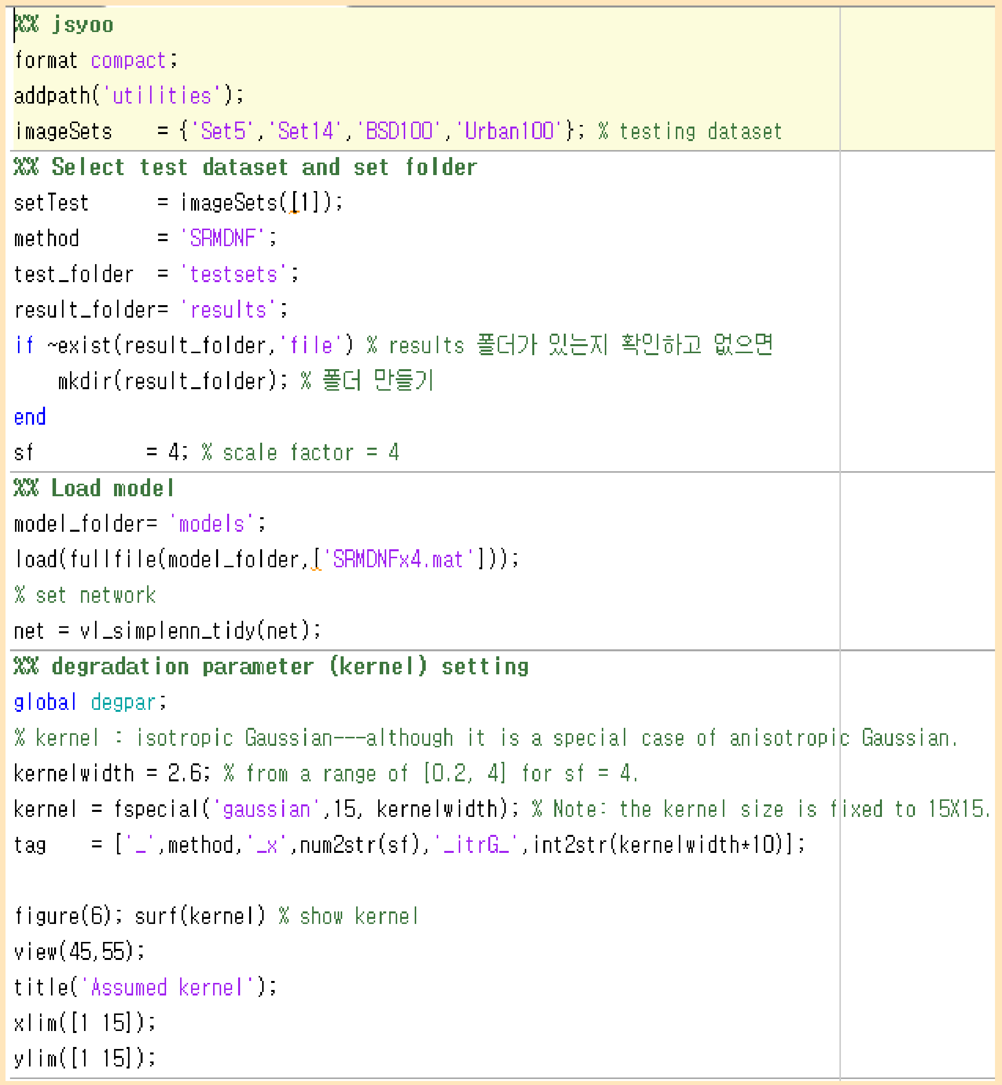
```
============================
Bicubic 구현 함수
경과 시간은 0.245806초입니다.
Implementation PSNR 25.1185.
Built-in PSNR 25.1185.
Bicubic 내장 함수
경과 시간은 0.022557초입니다.
Implementation PSNR 25.1118.
Built-in PSNR 25.1118.
============================
```

➔ 구현한 PSNR 계산 함수와 매트랩 내장 함수인 psnr의 결과가 같다

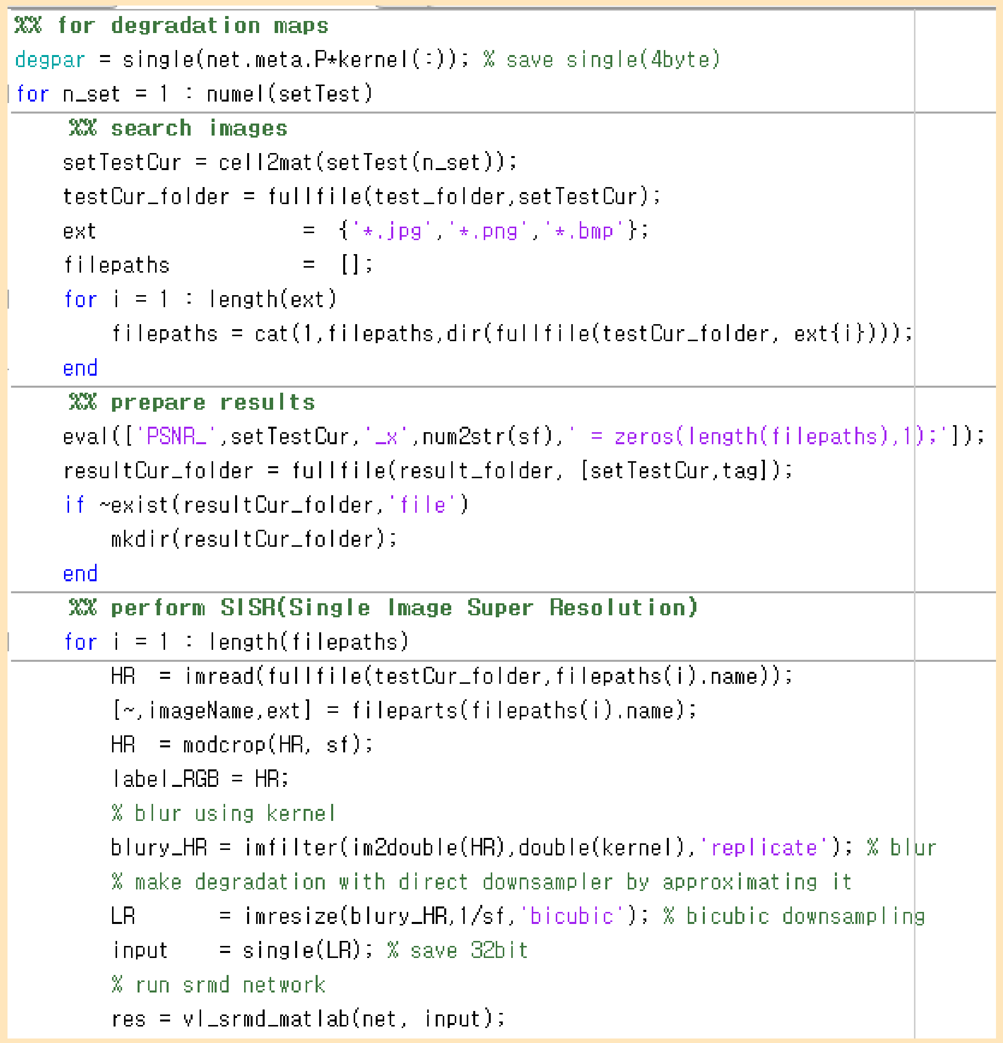
Matlab-based implementation

(4) SRMDNF

① Code



```
%% jsyoo
format compact;
addpath('utilities');
imageSets    = {'Set5','Set14','BSD100','Urban100'}; % testing dataset
%% Select test dataset and set folder
setTest      = imageSets([1]);
method       = 'SRMDNF';
test_folder  = 'testsets';
result_folder= 'results';
if ~exist(result_folder,'file') % results 폴더가 있는지 확인하고 없으면
    mkdir(result_folder); % 폴더 만들기
end
sf          = 4; % scale factor = 4
%% Load model
model_folder= 'models';
load(fullfile(model_folder,['SRMDNFx4.mat']));
% set network
net = vl_simplenn_tidy(net);
%% degradation parameter (kernel) setting
global degpar;
% kernel : isotropic Gaussian---although it is a special case of anisotropic Gaussian.
kernelwidth = 2.6; % from a range of [0.2, 4] for sf = 4.
kernel = fspecial('gaussian',15, kernelwidth); % Note: the kernel size is fixed to 15X15.
tag    = ['_',method,'_x',num2str(sf),'_itrG_',int2str(kernelwidth*10)];

figure(6); surf(kernel) % show kernel
view(45,55);
title('Assumed kernel');
xlim([1 15]);
ylim([1 15]);
```

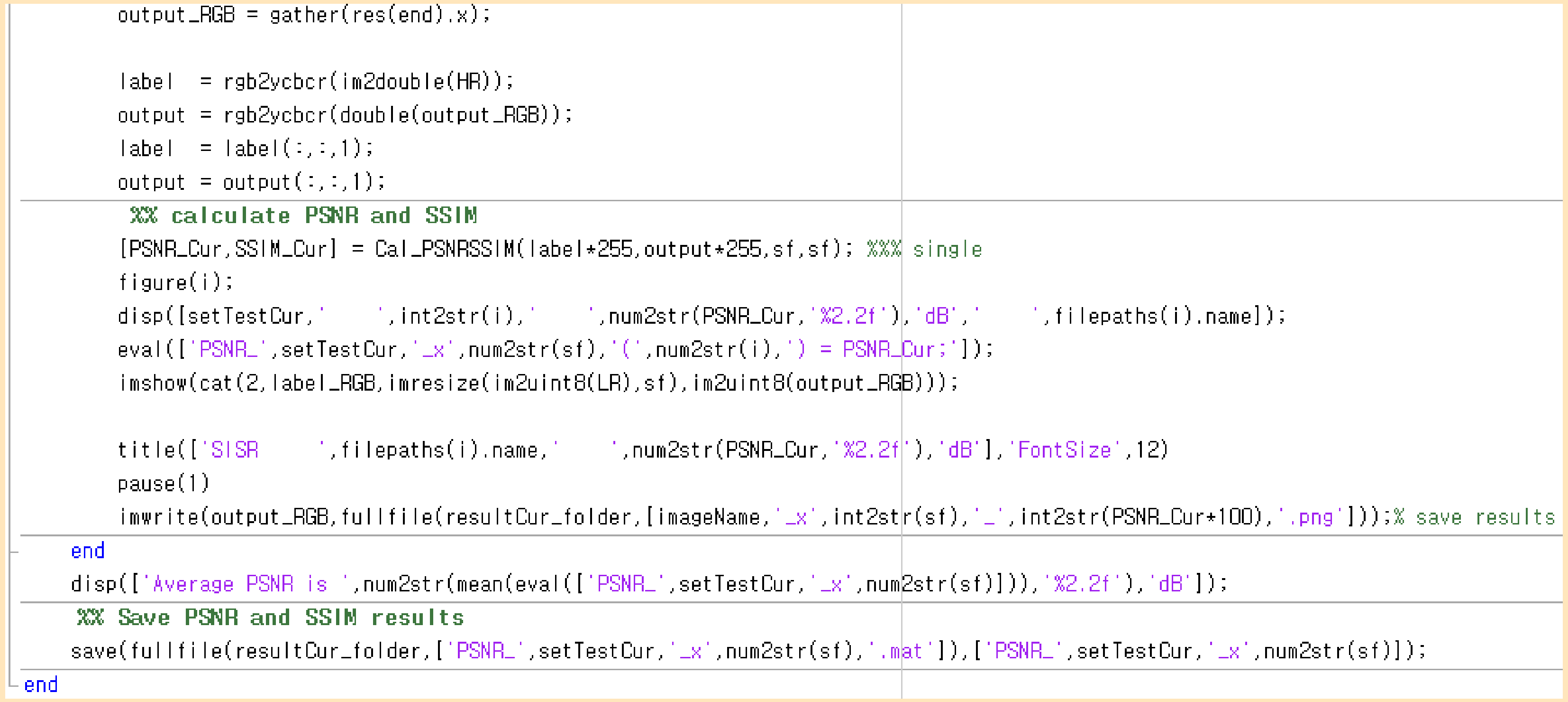


```
%% for degradation maps
degpar = single(net.meta.P*kernel(:)); % save single(4byte)
for n_set = 1 : numel(setTest)
    %% search images
    setTestCur = cell2mat(setTest(n_set));
    testCur_folder = fullfile(test_folder,setTestCur);
    ext              =  {'*.jpg','*.png','*.bmp'};
    filepaths        =  [];
    for i = 1 : length(ext)
        filepaths = cat(1,filepaths,dir(fullfile(testCur_folder, ext{i})));
    end
    %% prepare results
    eval(['PSNR_',setTestCur,'_x',num2str(sf),' = zeros(length(filepaths),1);']);
    resultCur_folder = fullfile(result_folder, [setTestCur,tag]);
    if ~exist(resultCur_folder,'file')
        mkdir(resultCur_folder);
    end
    %% perform SISR(Single Image Super Resolution)
    for i = 1 : length(filepaths)
        HR  = imread(fullfile(testCur_folder,filepaths(i).name));
        [~,imageName,ext] = fileparts(filepaths(i).name);
        HR  = modcrop(HR, sf);
        label_RGB = HR;
        % blur using kernel
        blury_HR = imfilter(im2double(HR),double(kernel),'replicate'); % blur
        % make degradation with direct downsampler by approximating it
        LR       = imresize(blury_HR,1/sf,'bicubic'); % bicubic downsampling
        input    = single(LR); % save 32bit
        % run srmd network
        res = vl_srmd_matlab(net, input);
```

Matlab-based implementation

(4) SRMDNF

① Code



```
        output_RGB = gather(res(end).x);

        label  = rgb2ycbcr(im2double(HR));
        output = rgb2ycbcr(double(output_RGB));
        label  = label(:,:,1);
        output = output(:,:,1);
        %% calculate PSNR and SSIM
        [PSNR_Cur,SSIM_Cur] = Cal_PSNRSSIM(label*255,output*255,sf,sf); %%% single
        figure(i);
        disp([setTestCur,'    ',int2str(i),'    ',num2str(PSNR_Cur,'%2.2f'),'dB','    ',filepaths(i).name]);
        eval(['PSNR_',setTestCur,'_x',num2str(sf),'(',num2str(i),') = PSNR_Cur;']);
        imshow(cat(2,label_RGB,imresize(im2uint8(LR),sf),im2uint8(output_RGB)));

        title(['SISR     ',filepaths(i).name,'    ',num2str(PSNR_Cur,'%2.2f'),'dB'],'FontSize',12)
        pause(1)
        imwrite(output_RGB,fullfile(resultCur_folder,[imageName,'_x',int2str(sf),'_',int2str(PSNR_Cur*100),'.png']));% save results
    end
    disp(['Average PSNR is ',num2str(mean(eval(['PSNR_',setTestCur,'_x',num2str(sf)])),'%2.2f'),'dB']);
    %% Save PSNR and SSIM results
    save(fullfile(resultCur_folder,['PSNR_',setTestCur,'_x',num2str(sf),'.mat']),['PSNR_',setTestCur,'_x',num2str(sf)]);
end
```

Matlab-based implementation

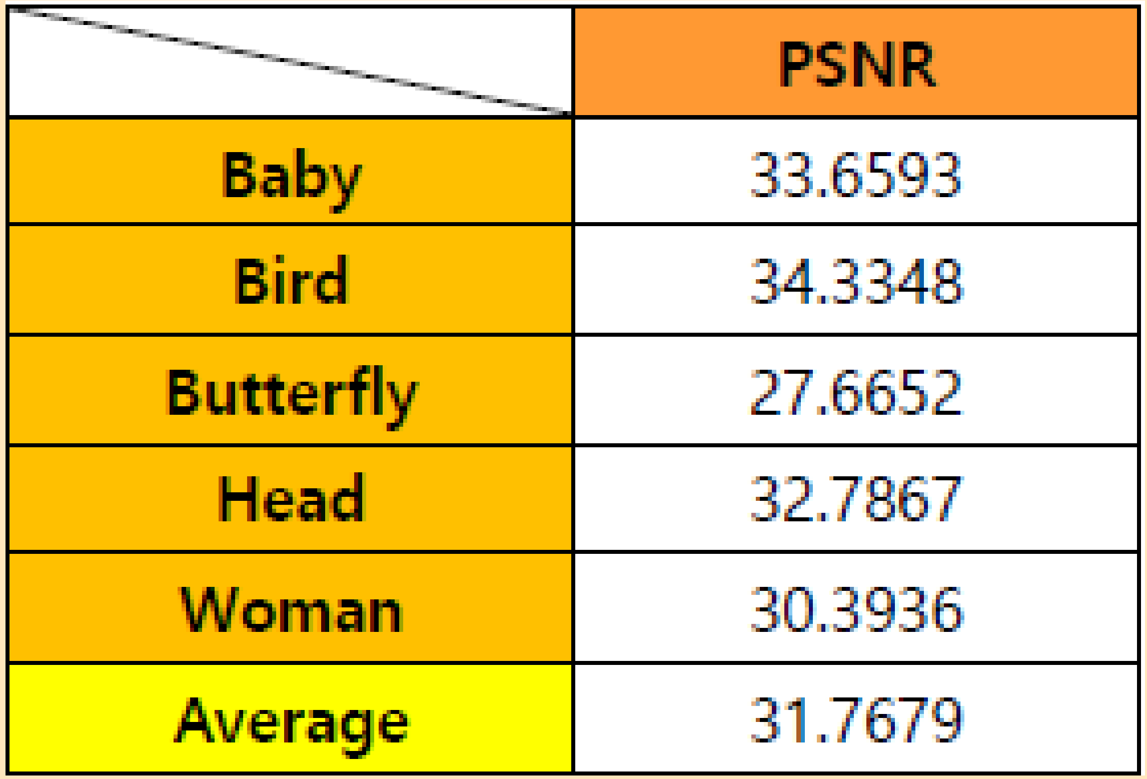
(4) SRMDNF

② Result image

Matlab-based implementation

(4) SRMDNF

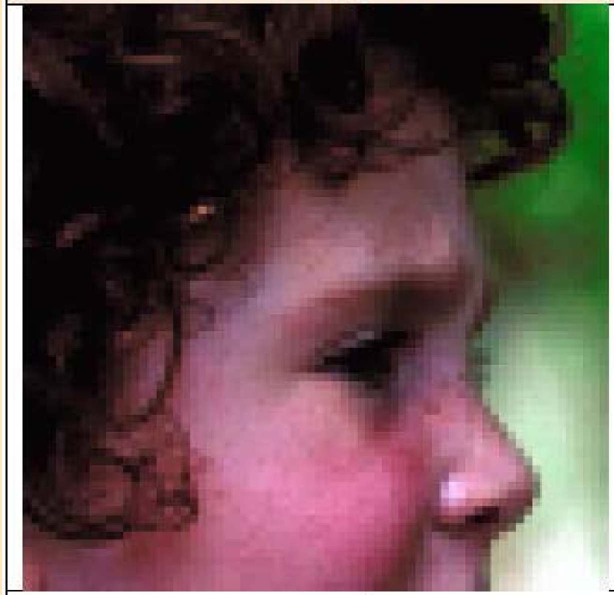
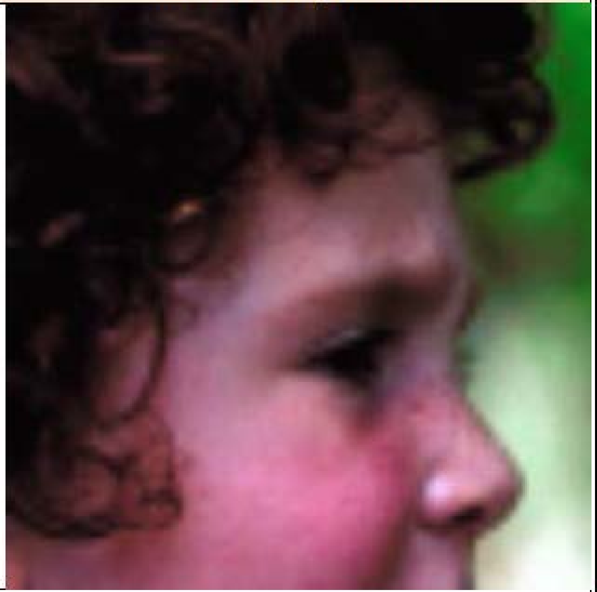
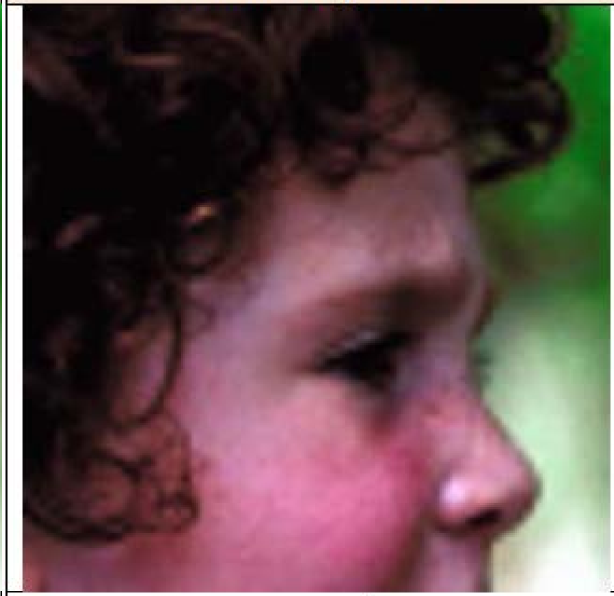
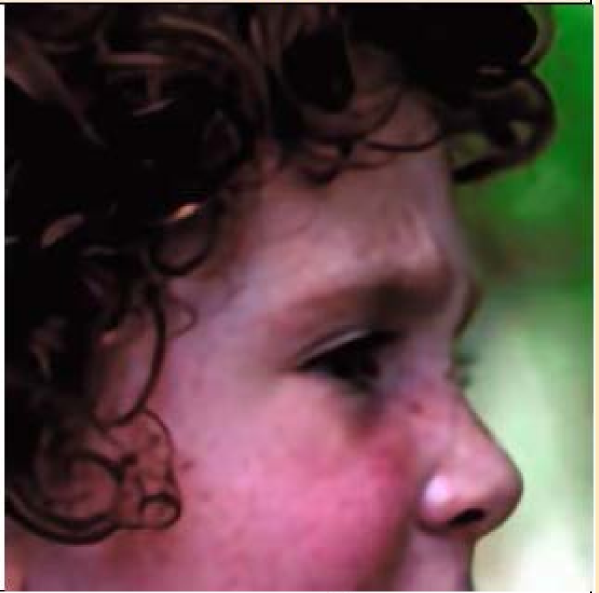
③ PSNR



	PSNR
Baby	33.6593
Bird	34.3348
Butterfly	27.6652
Head	32.7867
Woman	30.3936
Average	31.7679

Conclusion

★ **Nearest Neighbor, Bilinear interpolation, Bicubic interpolation(구현), SRMDNF 비교**

Nearest Neighbor	Bilinear interpolation	Bicubic interpolation	SRMDNF
			
27.9197 db	28.5450	28.9421 db	32.7867 db

Reference

https://en.wikipedia.org/wiki/Bicubic_interpolation

https://ko.wikipedia.org/wiki/%EC%B5%9C%EB%8C%80_%EC%8B%A0%ED%98%B8_%EB%8C%80_%EC%9E%A1%EC%9D%8C%EB%B9%84

https://github.com/thoste/matlab-scaler

https://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_Learning_a_Single_CVPR_2018_paper.html

https://github.com/cszn/SRMD

Thank you