ITE4052 Computer Vision

Programming Assignment1

2020095178 최윤선

1. Projective Image Transformation

'projective2d'는 현재 matlab 버전에서 권장되지 않는다고 하여 'projtform2d'를 사용함.

<peppers.png>

(1) Scaling


```
% (1) Scaling
scaling = projtform2d([3,0,0; ...
                       0,2,0; ...
                       0,0,1]);
scaled_img = imwarp(img, scaling);
```

Matrix: $\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

x 축으로 3 배, y 축으로 2 배 scaling

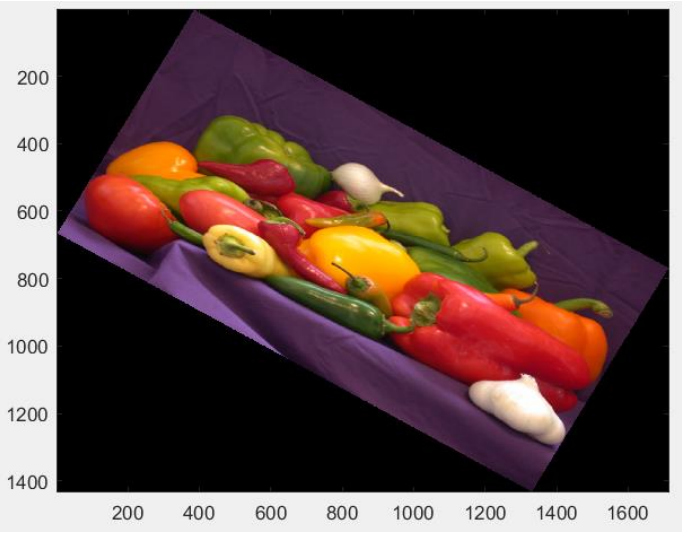
(2) Rotation


```
% (2) Rotation
rotation = projtform2d([cos(pi/6),-sin(pi/6),0; ...
                        sin(pi/6),cos(pi/6),0; ...
                        0,0,1]);
rotated_img = imwarp(img, rotation);
```

Matrix: $\begin{bmatrix} \cos\frac{\pi}{6} & -\sin\frac{\pi}{6} & 0 \\ \sin\frac{\pi}{6} & \cos\frac{\pi}{6} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

30° rotation

(3) Similarity transformation




```
% (3) Similarity transform
similarity = projtform2d([3*cos(pi/6),-2*sin(pi/6),50;
                          3*sin(pi/6),2*cos(pi/6),100;
                          0,0,1]);
similarity_img = imwarp(img, similarity);
```

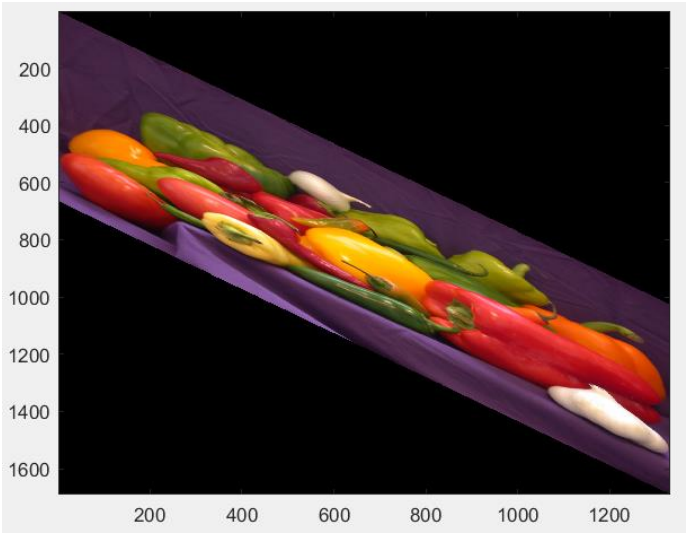
Matrix: $\begin{bmatrix} 3\cos\frac{\pi}{6} & -2\sin\frac{\pi}{6} & 50 \\ 3\sin\frac{\pi}{6} & 2\cos\frac{\pi}{6} & 100 \\ 0 & 0 & 1 \end{bmatrix}$

x 축으로 3 배, y 축으로 2 배 scaling

30° rotation

x 축으로 50, y 축으로 100 만큼 좌표 이동

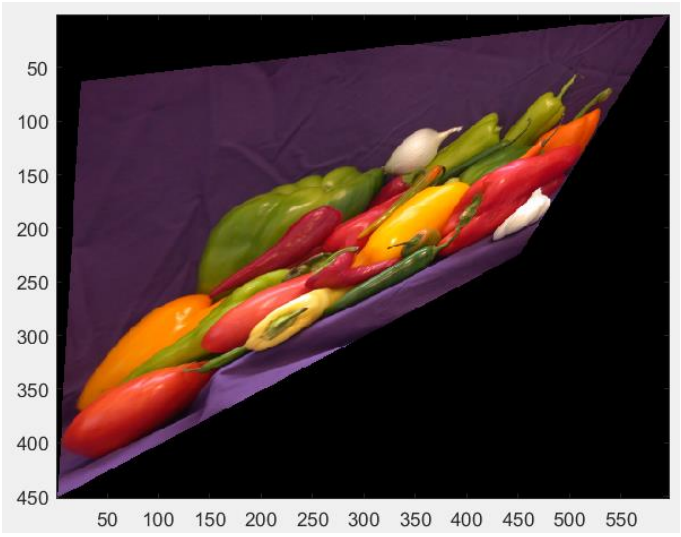
(4) Affine transformation




```
% (4) Affine transform
affine = projtform2d([3*cos(pi/6),-2*sin(pi/6)+1,50; ...
                      3*sin(pi/6)+0.5,2*cos(pi/6),100; ...
                      0,0,1]);
affine_img = imwarp(img, affine);
```

Matrix: $\begin{bmatrix} 3\cos\frac{\pi}{6} & -2\sin\frac{\pi}{6}+1 & 50 \\ 3\sin\frac{\pi}{6}+0.5 & 2\cos\frac{\pi}{6} & 100 \\ 0 & 0 & 1 \end{bmatrix}$

(5) Projective transformation

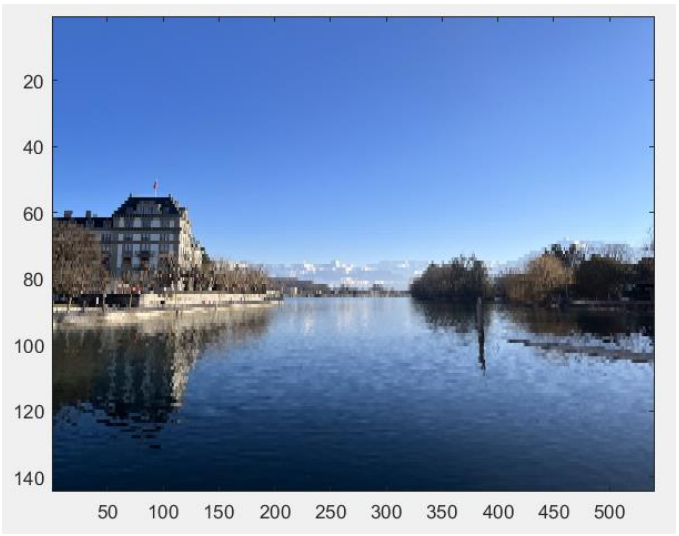



```
% (5) Projective transform
projective = projtform2d([3,0,50; ...
                          0,2,100; ...
                          0.003,0.002,1]);
projective_img = imwarp(img, projective);
```

Matrix: $\begin{bmatrix} 3 & 0 & 50 \\ 0 & 2 & 100 \\ 0.003 & 0.002 & 1 \end{bmatrix}$

<my own image>

(1) Scaling

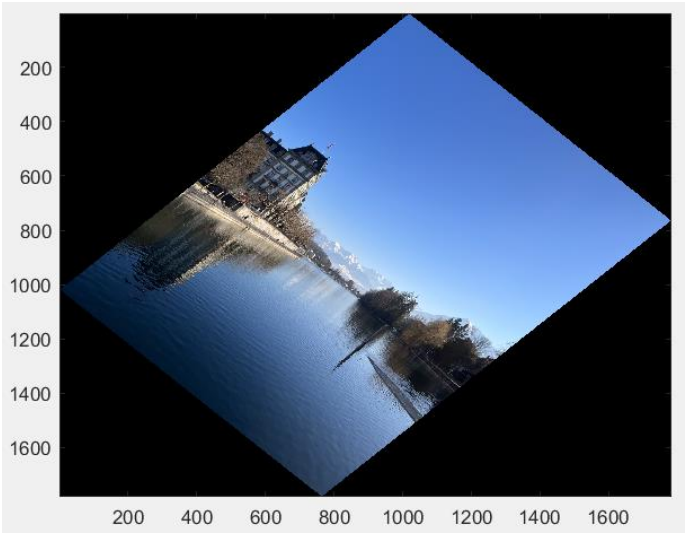



```
% (1) Scaling
my_scaling = projtform2d([0.5,0,0; ...
                          0,0.1,0; ...
                          0,0,1]);
my_scaled_img = imwarp(my_img, my_scaling);
```

Matrix: $\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

x 축으로 0.5 배, y 축으로 0.1 배 scaling

(2) Rotation

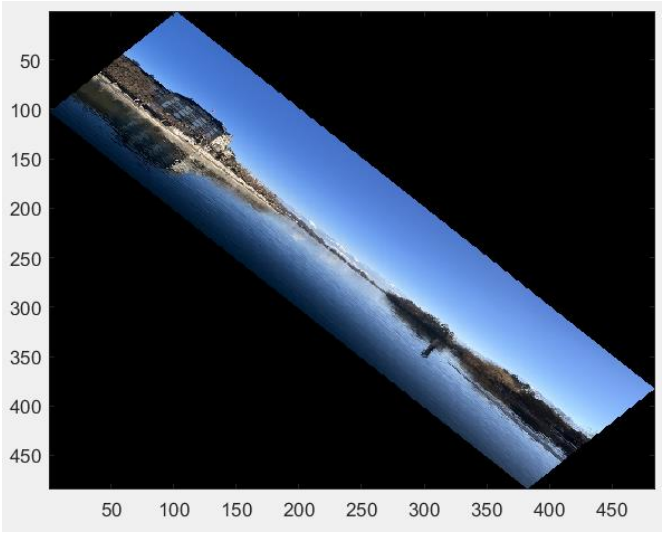



```
% (2) Rotation
my_rotation = projtform2d([cos(pi/4),-sin(pi/4),0; ...
                           sin(pi/4),cos(pi/4),0; ...
                           0,0,1]);
my_rotated_img = imwarp(my_img, my_rotation);
```

Matrix: $\begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix}$

45° rotation

(3) Similarity transformation




```
% (3) Similarity transform
my_similarity = projtform2d([0.5*cos(pi/4),-0.1*sin(pi/4),20; ...
                             0.5*sin(pi/4),0.1*cos(pi/4),40; ...
                             0,0,1]);
my_similarity_img = imwarp(my_img, my_similarity);
```

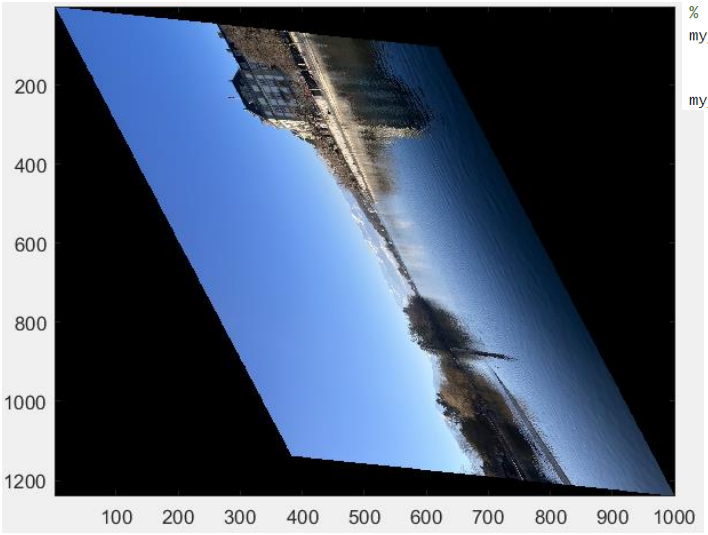
Matrix: $\begin{bmatrix} 0.5\cos\frac{\pi}{4} & -0.1\sin\frac{\pi}{4} & 20 \\ 0.5\sin\frac{\pi}{4} & 0.1\cos\frac{\pi}{4} & 40 \\ 0 & 0 & 1 \end{bmatrix}$

x 축으로 0.5 배, y 축으로 0.1 배 scaling

45° rotation

x 축으로 20, y 축으로 40 만큼 좌표 이동

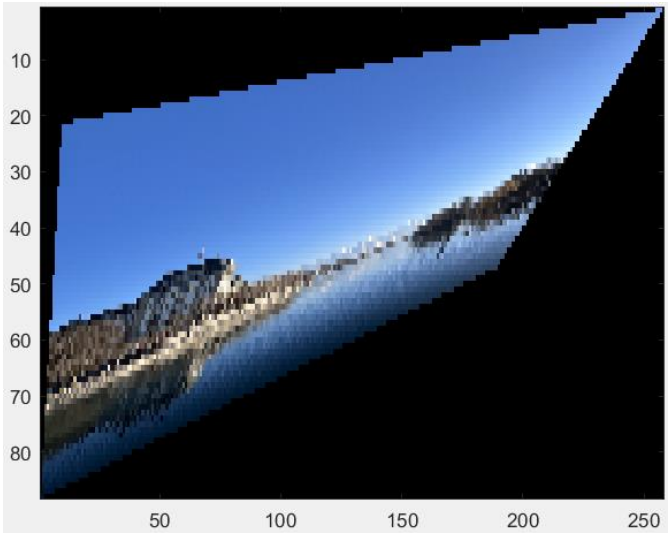
(4) Affine transformation




```
% (4) Affine transform
my_affine = projtform2d([0.5*cos(pi/4),-0.1*sin(pi/4)+0.5,20; ...
                         0.5*sin(pi/4)+0.7,0.1*cos(pi/4),40; ...
                         0,0,1]);
my_affine_img = imwarp(my_img, my_affine);
```

Matrix: $\begin{bmatrix} 0.5\cos\frac{\pi}{4} & -0.1\sin\frac{\pi}{4}+0.5 & 20 \\ 0.5\sin\frac{\pi}{4}+0.7 & 0.1\cos\frac{\pi}{4} & 40 \\ 0 & 0 & 1 \end{bmatrix}$

(5) Projective transformation



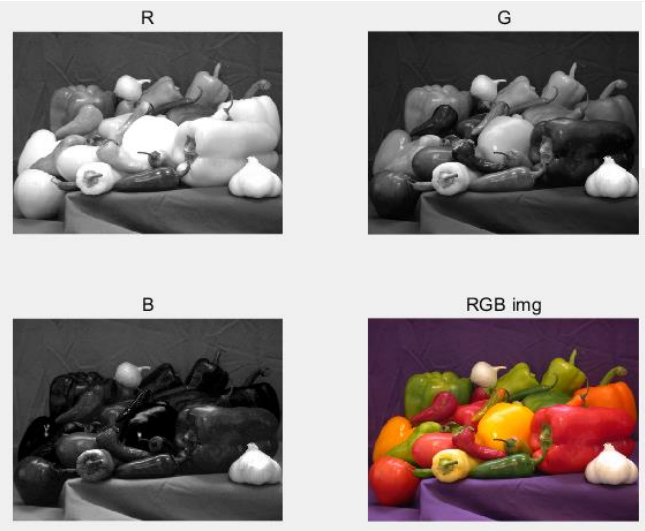

```
% (5) Projective transform
my_projective = projtform2d([0.5,0,20; ...
                             0,0.1,40; ...
                             0.001,0.0005,1]);
my_projective_img = imwarp(my_img, my_projective);
```

Matrix: $\begin{bmatrix} 0.5 & 0 & 20 \\ 0 & 0.1 & 40 \\ 0.001 & 0.0005 & 1 \end{bmatrix}$

2. Color Space

<pepper.png>

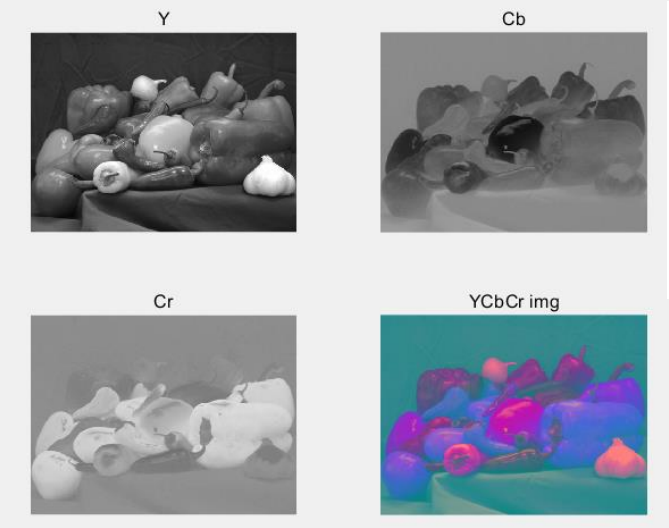
(1) RGB




```
% (1) Display RGB components
subplot(2,2,1); imshow(r); title('R');
subplot(2,2,2); imshow(g); title('G');
subplot(2,2,3); imshow(b); title('B');
subplot(2,2,4); imshow(img); title('RGB img');
```

img 에서 R, G, B 값을 각각 추출하여 RGB components 를 나타내었다.

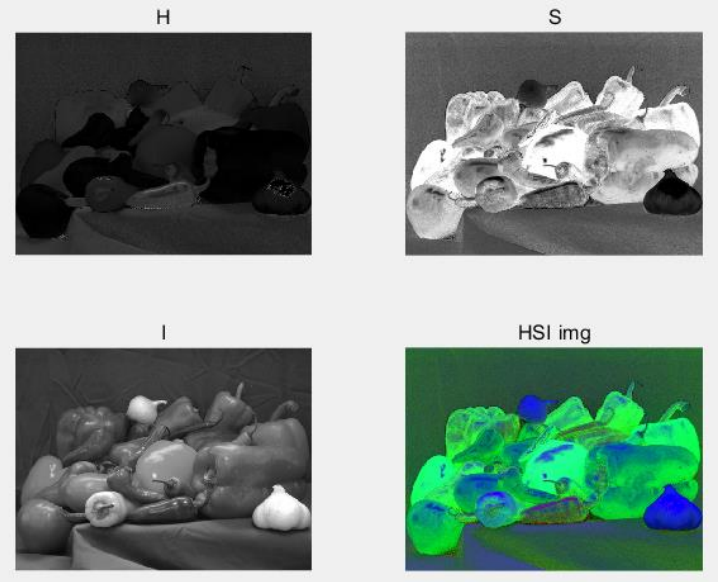
(2) YCbCr




```
% (2) Display YCbCr
y = 0.299*r_n + 0.587*g_n + 0.114*b_n;
cb = -0.169*r_n - 0.331*g_n + 0.5*b_n + 0.5;
cr = 0.5*r_n - 0.419*g_n - 0.081*b_n + 0.5;
ycbcr = cat(3,y,cb,cr);

subplot(2,2,1); imshow(y); title('Y');
subplot(2,2,2); imshow(cb); title('Cb');
subplot(2,2,3); imshow(cr); title('Cr');
subplot(2,2,4); imshow(ycbcr); title('YCbCr img');
```

RGB 를 YCbCr 공간으로 바꿔주는 행렬을 사용하였고, 이때 RGB 값을 0~1 사이의 값으로 normalize 한 후 계산하였다.

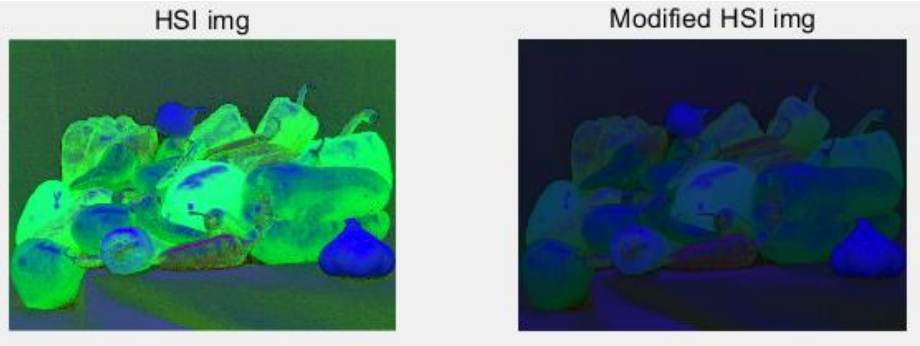
(3) HSI




```
% (3) Display HSI
i = (r_n+g_n+b_n)/3;
s = 1-(3./(r_n+g_n+b_n)).*min(min(r_n,g_n),b_n);
h = acos((2*r_n-g_n-b_n)./(2.*sqrt((r_n-g_n).^2+(r_n-b_n).*(g_n-b_n))))/(2*pi);
hsi = cat(3,h,s,i);

subplot(2,2,1); imshow(h); title('H');
subplot(2,2,2); imshow(s); title('S');
subplot(2,2,3); imshow(i); title('I');
subplot(2,2,4); imshow(hsi); title('HSI img');
```

RGB 값을 HSI 공간으로 바꿔 주는 공식을 사용하였고, 이때 RGB 값을 0~1 사이의 값으로 normalize 한 후 계산하였다.

(4) Modified RGB



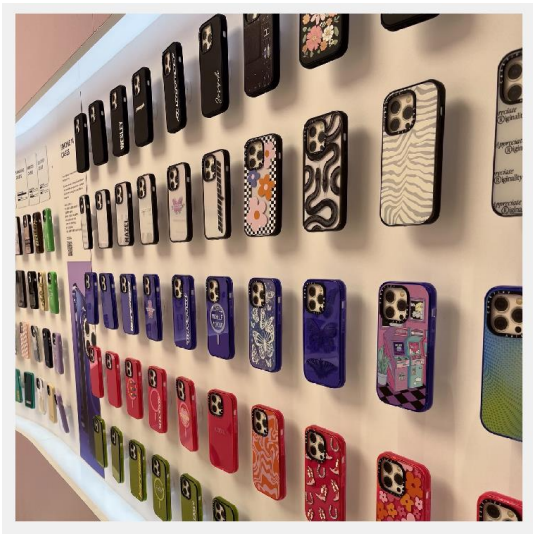

```
% (4) Modified Image
new_h = 0.5.*h;
new_s = 0.25.*s;
new_i = 0.75.*i;
```

HSI 공간에서 우선 Hue 를 0.5 배, Saturation 을 0.25 배, Intensity 를 0.75 배로 조정하였다.

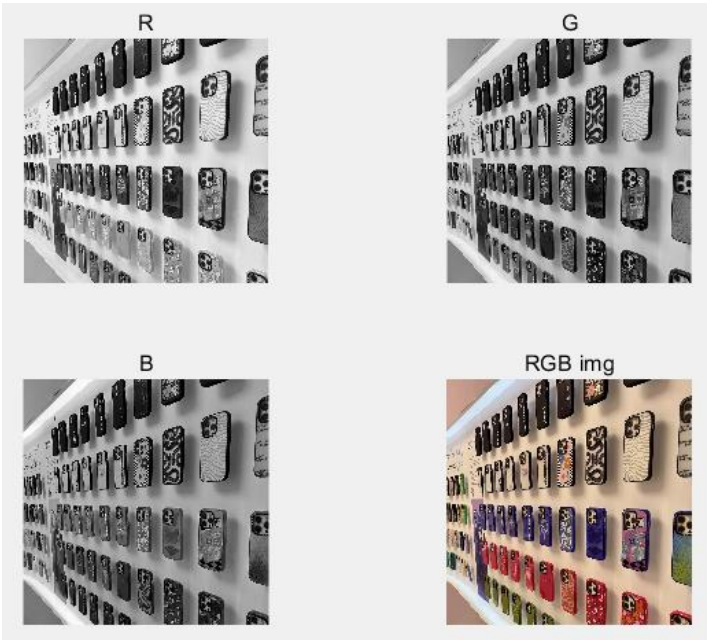

```
new_h = new_h*360;
new_r = zeros(size(new_h));
new_g = zeros(size(new_h));
new_b = zeros(size(new_h));

for i = 1:numel(new_h)
    % 0 <= new_h < 120
    if (new_h(i) >= 0) && (new_h(i) < 120)
        new_b(i) = new_i(i).*(1-new_s(i));
        new_r(i) = new_i(i).*(1+(new_s(i).*cosd(new_h(i))./cosd(60-new_h(i)));
        new_g(i) = 3.*new_i(i)-(new_r(i)+new_b(i));
        temp_h(i) = new_h(i)-120;
    % 120 <= new_h < 240
    elseif (new_h(i) >= 120) && (new_h(i) < 240)
        new_r(i) = new_i(i).*(1-new_s(i));
        new_g(i) = new_i(i)*(1+((new_s(i).*cosd(temp_h(i)))./cosd(60-temp_h(i))));
        new_b(i) = 3.*new_i(i)-(new_r(i)+new_g(i));
        temp_h(i) = new_h(i)-240;
    % 240 <= new_h < 360
    elseif (new_h(i) >= 240) && (new_h(i) < 360)
        new_g(i) = new_i(i).*(1-new_s(i));
        new_b(i) = new_i(i).*(1+((new_s(i).*cosd(temp_h(i)))./cosd(60-temp_h(i))));
        new_r(i) = 2.*new_i(i)-(new_g(i)+new_b(i));
    end
end

new_rgb = cat(3,new_r,new_g,new_b);
```

조정한 H, S, I 값을 new_h, new_s, new_i 로 새로 정의한 후, HSI 공간을 RGB 공간으로 바꾸어 주는 공식을 사용하였다. Hue 값이 [0, 120), [120, 240), [240, 360)인 구간에 따라 새로운 r, g, b 값을 구하는 방법이 달라, if 문을 사용하여 코드를 작성하였다.

<my own image>



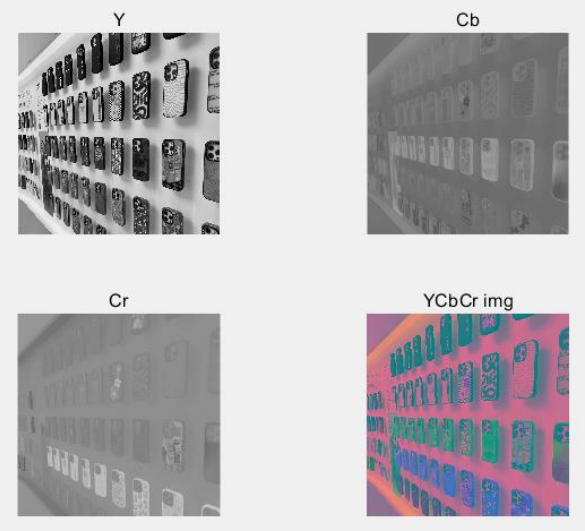
(1) RGB




```
% (1) Display RGB components
subplot(2,2,1); imshow(my_r); title('R');
subplot(2,2,2); imshow(my_g); title('G');
subplot(2,2,3); imshow(my_b); title('B');
subplot(2,2,4); imshow(my_img); title('RGB img');
```

my_img 에서 R, G, B 값을 각각 추출하여 RGB components 를 나타내었다.

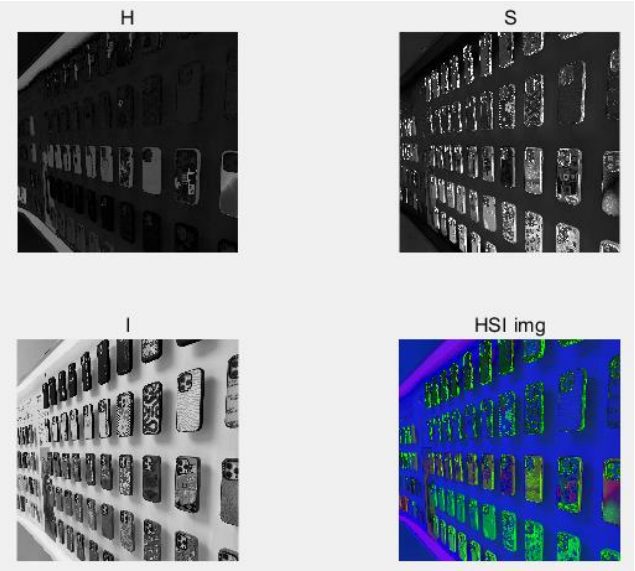
(2) YCbCr




```
% (2) Display YCbCr
my_y = 0.299*my_r_n + 0.587*my_g_n + 0.114*my_b_n;
my_cb = -0.169*my_r_n - 0.331*my_g_n + 0.5*my_b_n + 0.5;
my_cr = 0.5*my_r_n - 0.419*my_g_n - 0.081*my_b_n + 0.5;
my_ycbcr = cat(3,my_y,my_cb,my_cr);

subplot(2,2,1); imshow(my_y); title('Y');
subplot(2,2,2); imshow(my_cb); title('Cb');
subplot(2,2,3); imshow(my_cr); title('Cr');
subplot(2,2,4); imshow(my_ycbcr); title('YCbCr img');
```

RGB 를 YCbCr 공간으로 바꿔주는 행렬을 사용하였고, 이때 RGB 값을 0~1 사이의 값으로 normalize 한 후 계산하였다.

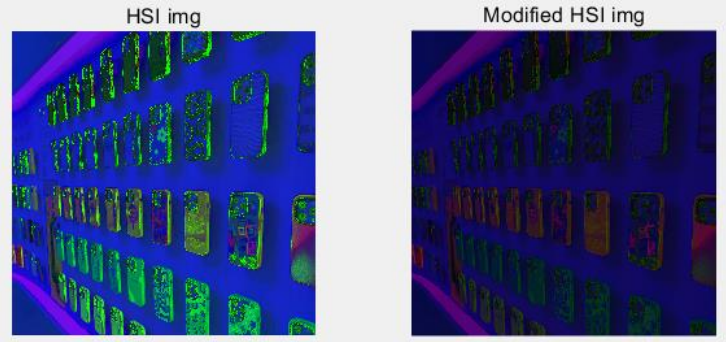
(3) HSI




```
% (3) Display HSI
my_i = (my_r_n+my_g_n+my_b_n)/3;
my_s = 1-(3./(my_r_n+my_g_n+my_b_n)).*min(min(my_r_n,my_g_n),my_b_n);
my_h = acos((2*my_r_n-my_g_n-my_b_n)./(2.*sqrt((my_r_n-my_g_n).^2+(my_r_n-my_b_n).*(my_g_n-my_b_n))))/(2*pi);
my_hsi = cat(3,my_h,my_s,my_i);

subplot(2,2,1); imshow(my_h); title('H');
subplot(2,2,2); imshow(my_s); title('S');
subplot(2,2,3); imshow(my_i); title('I');
subplot(2,2,4); imshow(my_hsi); title('HSI img');
```

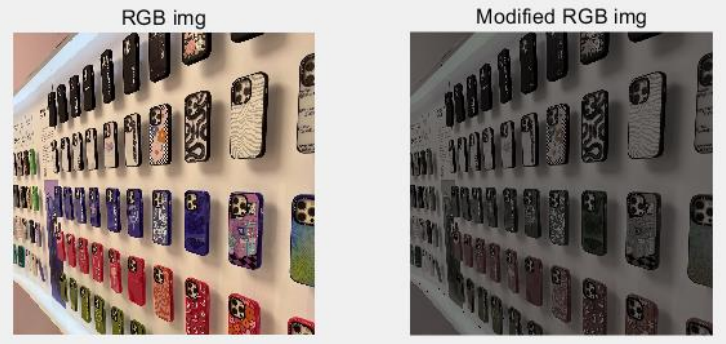
RGB 값을 HSI 공간으로 바꿔 주는 공식을 사용하였고, 이때 RGB 값을 0~1 사이의 값으로 normalize 한 후 계산하였다.

(4) Modified RGB




```
% (4) Modified Image
my_new_h = 0.75.*my_h;
my_new_s = 0.25.*my_s;
my_new_i = 0.5.*my_i;
```

HSI 공간에서 우선 Hue 를 0.75 배, Saturation 을 0.25 배, Intensity 를 0.5 배로 조정하였다.




```
my_new_h = my_new_h*360;
my_new_r = zeros(size(my_new_h));
my_new_g = zeros(size(my_new_h));
my_new_b = zeros(size(my_new_h));

for j = 1:numel(my_new_h)
    % 0 <= my_new_h < 120
    if (my_new_h(j) >= 0) && (my_new_h(j) < 120)
        my_new_b(j) = my_new_i(j).*(1-my_new_s(j));
        my_new_r(j) = my_new_i(j).*(1+(my_new_s(j).*cosd(my_new_h(j))./cosd(60-my_new_h(j)));
        my_new_g(j) = 3.*my_new_i(j)-(my_new_r(j)+my_new_b(j));
    % 120 <= my_new_h < 240
    elseif (my_new_h(j) >= 120) && (my_new_h(j) < 240)
        my_new_r(j) = my_new_i(j).*(1-my_new_s(j));
        my_new_g(j) = my_new_i(j)*(1+((my_new_s(j).*cosd(my_new_h(j)-120))./cosd(180-my_new_h(j))));
        my_new_b(j) = 3.*my_new_i(j)-(my_new_r(j)+my_new_g(j));
    % 240 <= my_new_h < 360
    elseif (my_new_h(j) >= 240) && (my_new_h(j) < 360)
        my_new_g(j) = my_new_i(j).*(1-my_new_s(j));
        my_new_b(j) = my_new_i(j).*(1+((my_new_s(j).*cosd(my_new_h(j)-240))./cosd(300-my_new_h(j))));
        my_new_r(j) = 2.*my_new_i(j)-(my_new_g(j)+my_new_b(j));
    end
end

my_new_rgb = cat(3,my_new_r,my_new_g,my_new_b);
```

조정한 H, S, I 값을 my_new_h, my_new_s, my_new_i 로 새로 정의한 후, HSI 공간을 RGB 공간으로 바꾸어 주는 공식을 사용하였다. Hue 값이 [0, 120), [120, 240), [240, 360)인 구간에 따라 새로운 r, g, b 값을 구하는 방법이 달라, if 문을 사용하여 코드를 작성하였다.