Homework 7

Meta information	
Name	Savan Kiran
Program	Masters in Computer Science
Questions skipped	PART D, PART C
Questions substituted	N/A
Extra credit questions	N/A

PART A

1. The white patch in the Macbeth image is in the bottom left corner. We ignore the edges of the patch as they are unreliable and pick the top-left (start) pixel of the rectangle as (100,325) and bottom-right (end) pixel as (150,380). We calculate the average values of R, G and B for all the pixels inside the rectangle and then scale the result so that the max is 250. This gives us an estimate of the illuminant color of the light.
   Below is the code snippet to calculate the average R,G,B for the white patch.

```
function[r,g,b]=estimate_illuminant_color(image,x1,y1,x2,y2)
pixelCount=0;
rSum=0;
gSum=0;
bSum=0;
for i=x1:x2
    for j=y1:y2
        p=image(j,i,:);
        p=double(p);
        p=[p(1,1,1) p(1,1,2) p(1,1,3)];
        rSum=rSum+double(p(1,1));
        gSum=gSum+double(p(1,2));
        bSum=bSum+double(p(1,3));
        pixelCount=pixelCount+1;
    end
end
rAvg=double(rSum/pixelCount);
gAvg=double(gSum/pixelCount);
bAvg=double(bSum/pixelCount);
m=max([rAvg; gAvg; bAvg]);
r=double(rAvg*double((250/m)));
g=double(gAvg*double((250/m)));
b=double(bAvg*double((250/m)));
end
```

   Figure 1 shows the start and end pixels of the white patch in the Macbeth image.
   Illuminant color estimated for Macbeth image with ‘sy1-50MR16Q’ light is,

```
Illuminant color est for light -> syl-50MR16Q
  237.7861  220.3899  250.0000
```

Figure 1. The start and end pixel of the white patch in Macbeth image.

2. Using the same process as explained in step 1, we get the Illuminant color estimated for Macbeth image with 'solux-4100' light as,
```
Illuminant color est for light -> solux-4100
  132.1031  158.8935  250.0000
```
3. We calculate the angular error between the two illuminant colors in degrees by finding the dot product of both the color vectors divided by their magnitudes and using it as input in `acosd()` matlab function.
   Below is the code snippet that calculates angular error.
```
function[angularErr]=angular_error(l1,l2)
l1=double(l1);
l2=double(l2);
X=dot(l1,l2)/(norm(l1)*norm(l2));
angularErr=acosd(X);
end
```
   The angular error between 'sy1-50MR16Q' and 'solux-4100' is found to be,
```
Angular error between two light colors found above:
   13.8441
```
4. The diagonal model for the illuminant colors and image color is given by,

$$\begin{pmatrix} R_2 \\ G_2 \\ B_2 \end{pmatrix} = \left| \begin{matrix} \frac{R_{L2}}{R_{L1}} & & \\ & \frac{G_{L2}}{G_{L1}} & \\ & & \frac{B_{L2}}{B_{L1}} \end{matrix} \right| \begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix}$$

   Using the diagonal model with illuminant colors found earlier and the bluish image, we find the one under the canonical light. We remove the brightness factor from all 3 images – canonical, original and the corrected, by scaling up the images so that the max value in any channel is 250. Below is the code snippet for the same,

```
%Diagonal model
D=[lr1/lr2 0 0; 0 lg1/lg2 0; 0 0 lb1/lb2];
newImage=correct_image(D,image2);
newm=max(newImage);
newImage=newImage.*(250/newm);
figure;
imshow(newImage);

function[newImage]=correct_image(D,image)
newImage=zeros(size(image,1),size(image,2),3);
for i=1:size(image,1)
    for j=1:size(image,2)
        p=image(i,j,:);
        p=double(p);
        p=[p(1,1,1) p(1,1,2) p(1,1,3)];
        p=p';
        newP=D*p;
        newImage(i,j,1)=newP(1,1);
        newImage(i,j,2)=newP(2,1);
        newImage(i,j,3)=newP(3,1);
    end
end
newImage=uint8(newImage);
end
```

Figure 2. Macbeth bluish image

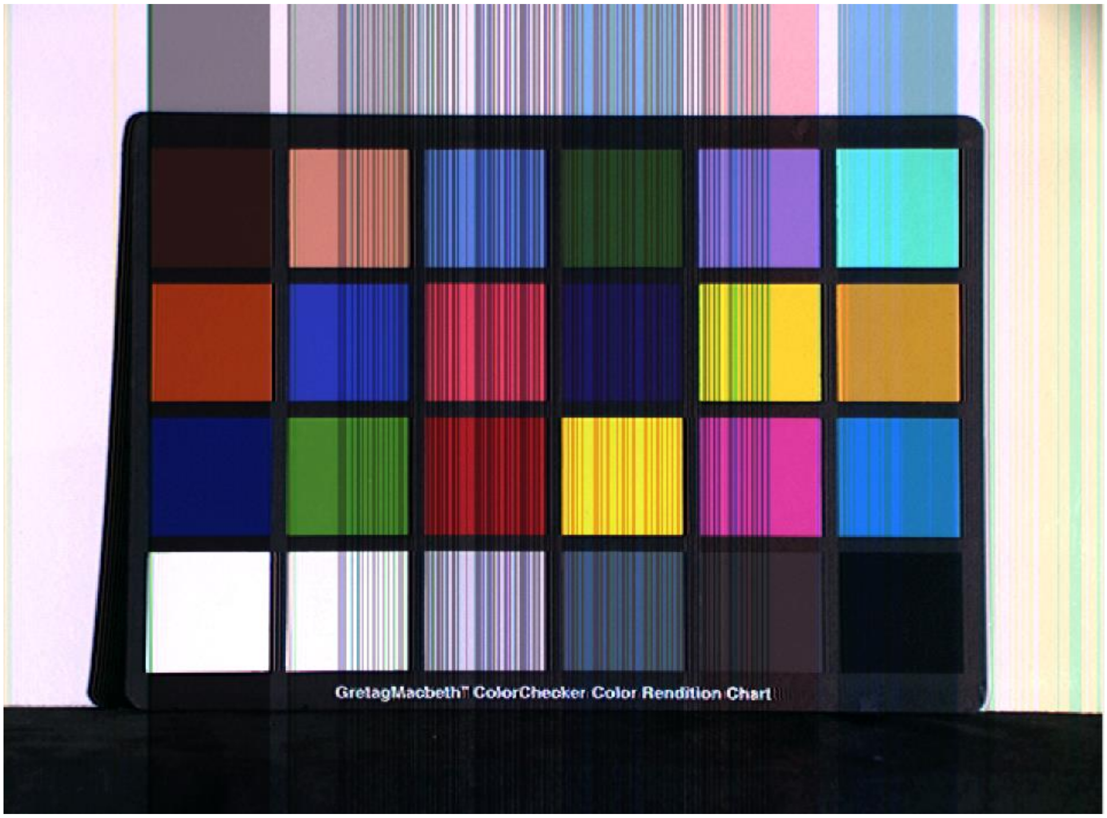


Figure 3. Macbeth canonical image

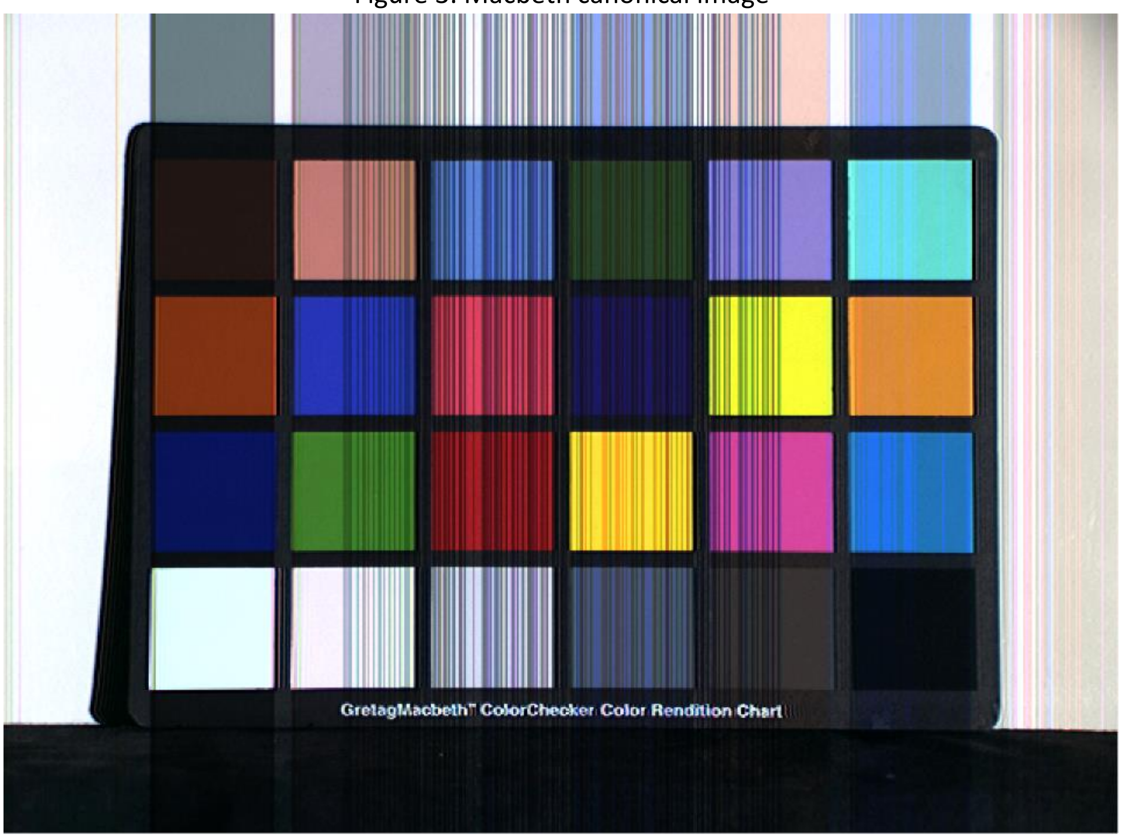


Figure 4. Macbeth corrected image

The figures 2, 3 & 4 show us the bluish, canonical and the correct image.

5. We now calculate the RMS error in (r,g) between
    a. The original image and the canonical image
       `RMS error(r,g) b/w original & canonical image: 0.25364`
    b. The improved corrected image and the canonical image
       `RMS error(r,g) b/w corrected & canonical image: 1.1819`

   The code snippet to calculate RMS error is given by,

```
function[rms]=rms_error(image,newImage)
SquaredErrSum=0;
pixelCount=0;
for i=1:size(image,1)
    for j=1:size(image,2)
        p=image(i,j,:);
        p=[p(1,1,1) p(1,1,2) p(1,1,3)];
        p2=newImage(i,j,:);
        p2=[p2(1,1,1) p2(1,1,2) p2(1,1,3)];
        if((p(1,1)+p(1,2)+p(1,3))>10 && (p2(1,1)+p2(1,2)+p2(1,3))>10)
            r=double(p(1,1)/(p(1,1)+p(1,2)+p(1,3)));
            g=double(p(1,2)/(p(1,1)+p(1,2)+p(1,3)));
            r2=double(p2(1,1)/(p2(1,1)+p2(1,2)+p2(1,3)));
            g2=double(p2(1,2)/(p2(1,1)+p2(1,2)+p2(1,3)));
            e=abs(r2-r)+abs(g2-g);
            SquaredErrSum=SquaredErrSum+(e*e);
            pixelCount=pixelCount+1;
        end
    end
end
rms=sqrt(SquaredErrSum/pixelCount);
end
```

6. We will now use the MaxRGB method to estimate light color for apples, ball and blocks. Below is the code snippet for MaxRGB algorithm.

```
function[r,g,b]=max_RGB(image)
ra=image(:,:,1);
ra=ra(:);
r=double(max(ra));
ga=image(:,:,2);
ga=ga(:);
g=double(max(ga));
ba=image(:,:,3);
ba=ba(:);
b=double(max(ba));
end
```

   We also calculate the angular errors between the 3 estimates. The angular errors were as follows,

```
Angular error for apple: 10.348
Angular error for ball: 19.3247
Angular error for blocks: 7.897
```

The code snippet for the same is below,

```
function report_maxRGB_ae(i1,i2,text)
[lr1,lg1,lb1]=max_RGB(i1);
l1=[lr1 lg1 lb1];
[lr2,lg2,lb2]=max_RGB(i2);
l2=[lr2 lg2 lb2];
ae=angular_error(l1,l2);
end
```

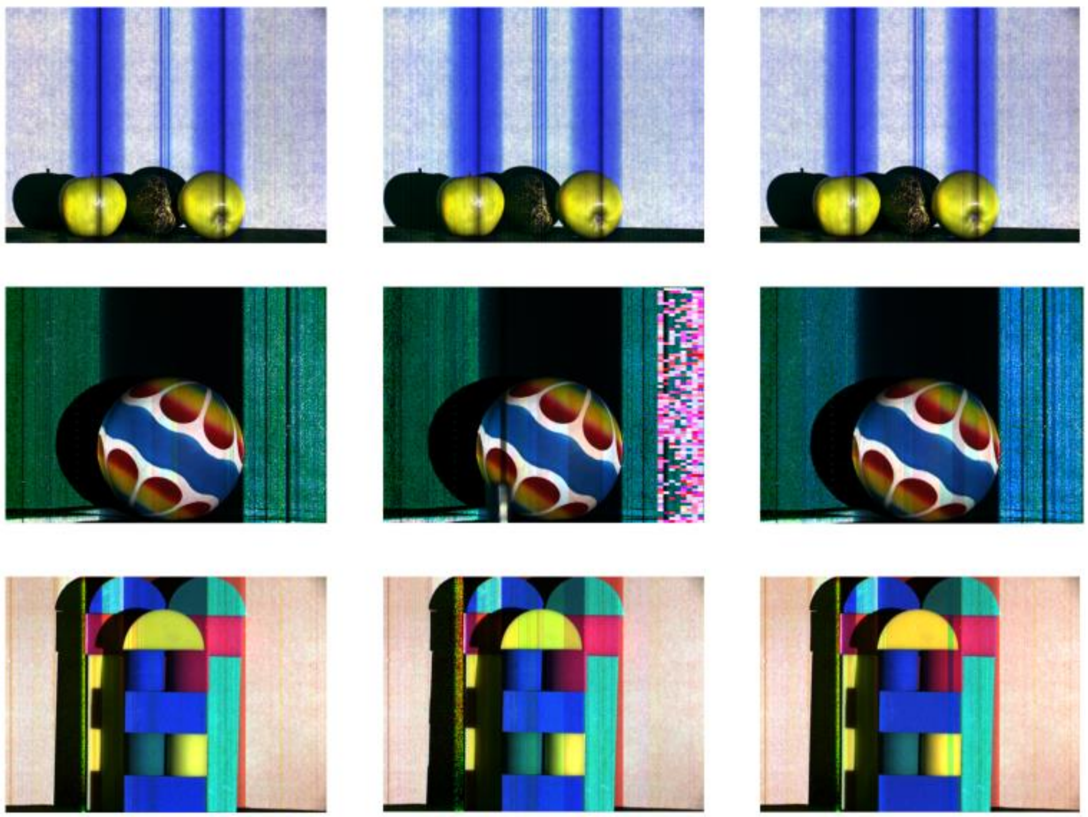
We also estimate the illuminant color for solus-4100 light on Macbeth image as,

```
Macbeth solus-4100 light Max RGB est. light:
   109   121   176
```

7. We now calculate the corrected image for all 3, apples, ball and blocks, using the MaxRGB illumination found in previous step.
   Below is the code snippet for the same.

```
function
scale_and_display_image(origImagePath,canonImagePath,text,maxOrGray)
origImage=imread(origImagePath);
canonImage=imread(canonImagePath);
if maxOrGray==1
    [lr1,lg1,lb1]=max_RGB(origImage);
    [lr2,lg2,lb2]=max_RGB(canonImage);
else
    [lr1,lg1,lb1]=gray_world(origImage);
    [lr2,lg2,lb2]=gray_world(canonImage);
end
D=[lr1/lr2 0 0; 0 lg1/lg2 0; 0 0 lb1/lb2];
newImage=correct_image(D,origImage);
newm=max(newImage);
newImage=newImage.*(250/newm);
mo=max(origImage);
mc=max(canonImage);
origImageS=origImage.*(250/mo);
canonImageS=canonImage.*(250/mc);
figure;
imshow(origImageS);
figure;
imshow(canonImageS);
figure;
imshow(newImage);
rms=rms_error(canonImage,newImage);
disp(['RMS error(r,g) for ' text ': ' num2str(rms)]);
end
%MaxRGB correction
scale_and_display_image('apples2_syl-50MR16Q.tif','apples2_solux-
4100.tif','apple',1);
scale_and_display_image('ball_syl-50MR16Q.tif','ball_solux-
4100.tif','ball',1);
scale_and_display_image('blocks1_syl-50MR16Q.tif','blocks1_solux-
4100.tif','blocks',1);
```

```
function report_maxRGB_ae(i1_path,i2_path,text)
i1=imread(i1_path);
i2=imread(i2_path);
[lr1,lg1,lb1]=max_RGB(i1);
l1=[lr1 lg1 lb1];
[lr2,lg2,lb2]=max_RGB(i2);
l2=[lr2 lg2 lb2];
ae=angular_error(l1,l2);
disp(['Angular error for ' text ': ' num2str(ae)]);
end
```



original canonical corrected

Figure 5. The original and canonical images for apple, ball and blocks are shown in first two columns. The third column is the image corrected using MaxRGB illumination estimate

The angular error for the mapped images is given below.

```
Angular error for apple: 10.348
Angular error for ball: 19.3247
Angular error for blocks: 7.897
```

The (r,g) RMS error for the mapped images is given below.

```
RMS error(r,g) b/n canonical and original image apple: 0.18592
RMS error(r,g) b/n canonical and corrected image apple: 1.635
RMS error(r,g) b/n canonical and original image ball: 0.35792
RMS error(r,g) b/n canonical and corrected image ball: 0.99758
RMS error(r,g) b/n canonical and original image blocks: 0.34908
RMS error(r,g) b/n canonical and corrected image blocks: 0.96665
```

There is a good agreement between RMS (r,g) error (b/n canonical & original) and angular error for apple and ball. But it is a bit odd for blocks where it has low angular error but high RMS error.

8. We would now compute angular and RMS (r,g) error for gray-world illumination estimates. The code is similar as last step, except we now introduce gray-world estimation,

```
function[r,g,b]=gray_world(image)
ra=image(:,:,1);
ra=ra(:);
r=double(2*mean(ra));
ga=image(:,:,2);
ga=ga(:);
g=double(2*mean(ga));
ba=image(:,:,3);
ba=ba(:);
b=double(2*mean(ba));
end
```

The angular error for the gray-world is given below.

```
Angular error for apple: 9.952
Angular error for ball: 13.6535
Angular error for blocks: 13.8635
```

The (r,g) RMS error for the gray-world is given below.

```
RMS error(r,g) b/n canonical and original image apple: 0.18592
RMS error(r,g) b/n canonical and corrected image apple: 1.6322
RMS error(r,g) b/n canonical and original image ball: 0.35792
RMS error(r,g) b/n canonical and corrected image ball: 0.99297
RMS error(r,g) b/n canonical and original image blocks: 0.34908
RMS error(r,g) b/n canonical and corrected image blocks: 0.96836
```

From the RMS (r,g) errors, we can observe that gray-world has lower error for apple and ball, but slightly higher for blocks. Thus, we can conclude that the gray-world works better for this data.

PART B

9. From the diagonal model we know that the diagonal matrix is composed of ratio of the illumination lights in each channel.

$$\begin{pmatrix} R_2 \\ G_2 \\ B_2 \end{pmatrix} = \begin{vmatrix} \frac{R_{L2}}{R_{L1}} & & \\ & \frac{G_{L2}}{G_{L1}} & \\ & & \frac{B_{L2}}{B_{L1}} \end{vmatrix} \begin{pmatrix} R_1 \\ G_1 \\ B_1 \end{pmatrix}$$

However, from above equation, we can see that the ratio of lights in each channel is equivalent to ratio of pixels in images for corresponding channel. Since we know the RGB composition of both the images, we can calculate the diagonal matrix. We will consider that minimum sum of the squared errors is our definition of best. Now, to have overall sum of squared errors to be minimum across all pixels for each channel, we can pick median of ratio of colors as the diagonal matrix elements. However, taking the median (or mean) of the ratios as diagonal matrix elements is only an approximation and hence cannot guarantee a better answer using (r,g) measure. Below is the code snippet that shows the calculations for the same.

```
function best_diagonal(origImagePath,canonImagePath,text)
origImage=imread(origImagePath);
canonImage=imread(canonImagePath);
Ro=origImage(:,:,1);
Ro=Ro(:);
Go=origImage(:,:,2);
Go=Go(:);
Bo=origImage(:,:,3);
Bo=Bo(:);

Rc=canonImage(:,:,1);
Rc=Rc(:);
Gc=canonImage(:,:,2);
Gc=Gc(:);
Bc=canonImage(:,:,3);
Bc=Bc(:);

alpha=Ro./Rc;
alpha=median(alpha);
beta=Go./Gc;
beta=median(beta);
gamma=Bo./Bc;
gamma=median(gamma);
D=[alpha 0 0; 0 beta 0; 0 0 gamma];
D=double(D);
newImage=correct_image(D,origImage);
newm=max(newImage);
newImage=newImage.*(250/newm);
mo=max(origImage);
mc=max(canonImage);
origImageS=origImage.*(250/mo);
canonImageS=canonImage.*(250/mc);
figure;
imshow(origImageS);
figure;
imshow(canonImageS);
figure;
imshow(newImage);
rms=rms_error(canonImage,newImage);
disp(['[Best] RMS error(r,g) for ' text ': ' num2str(rms)]);
end

best_diagonal('apples2_syl-50MR16Q.tif','apples2_solux-
4100.tif','apple');
best_diagonal('ball_syl-50MR16Q.tif','ball_solux-4100.tif','ball');
best_diagonal('blocks1_syl-50MR16Q.tif','blocks1_solux-
4100.tif','blocks');
```

10. We now calculate the corrected image for all 3, apples, ball and blocks, using the best diagonal map we derived in previous step. Figure 6 shows the same.

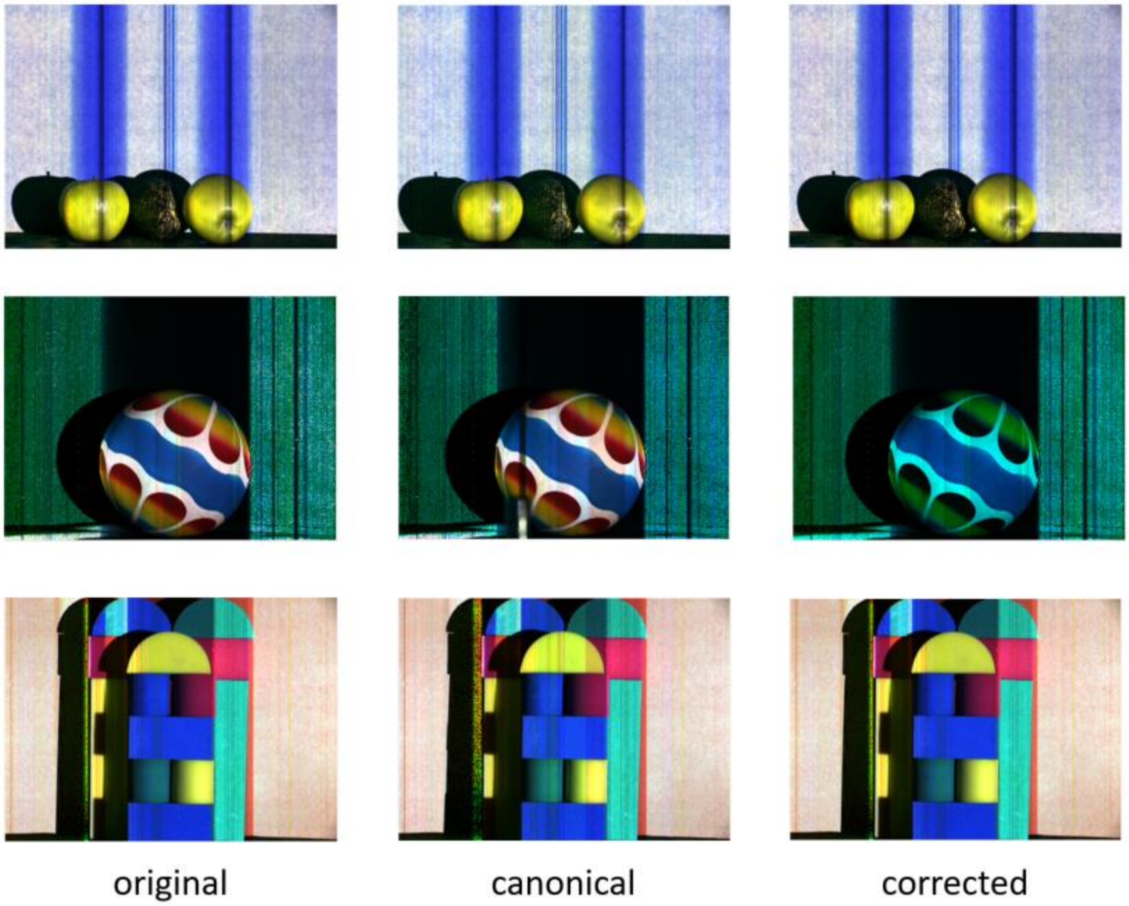


Figure 6. The original and canonical images for apple, ball and blocks are shown in first two columns. The third column is the image corrected using best diagonal map derived earlier

RMS error (r,g) is found to be,

```
[Best] RMS error(r,g) for apple: 1.6356
[Best] RMS error(r,g) for ball: 1.0465
[Best] RMS error(r,g) for blocks: 0.96377
```

Which is very much comparable to previous two methods (maxRGB and gray-world). The error however increases in case of the ball which is also evident in Figure 6 where corrected ball image is a lot bluer than it should have been.