**HW3: Flash/No Flash Photography**

Yiming Dai, YDF9097

**1. Write an android program to capture a flash/ no flash pair**

I changed the background of the screen and then capture a white background image and a black background image to capture a flash/no flash image pair. I set the foreground to be white with 200 ms delay and set the foreground to be transparent with 400 ms delay. I implemented captureJPEG() to take picture as jpg.

Captured no flash and flash image pair are as follows:

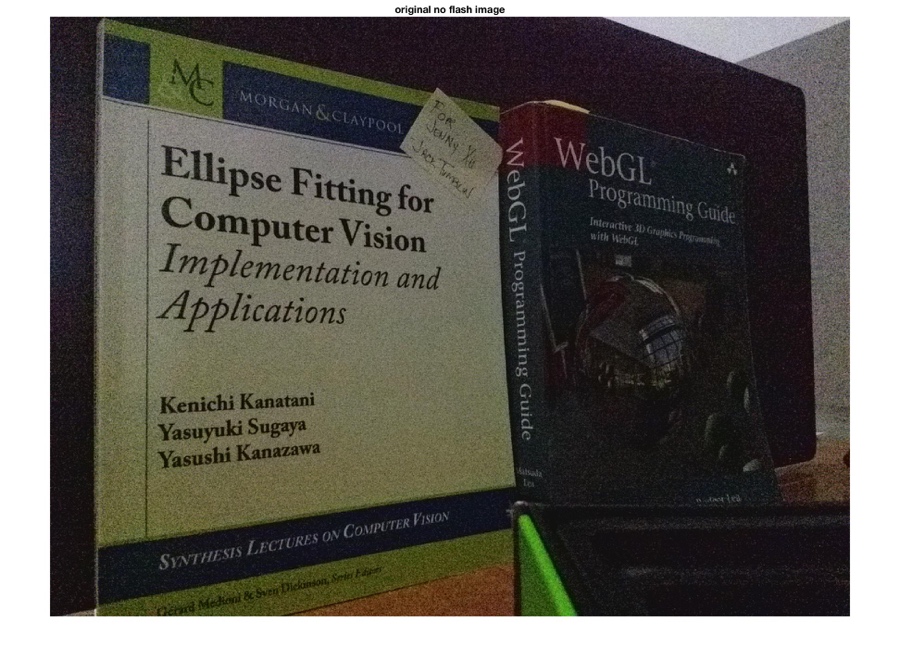
****

Figure 1 original no flash image

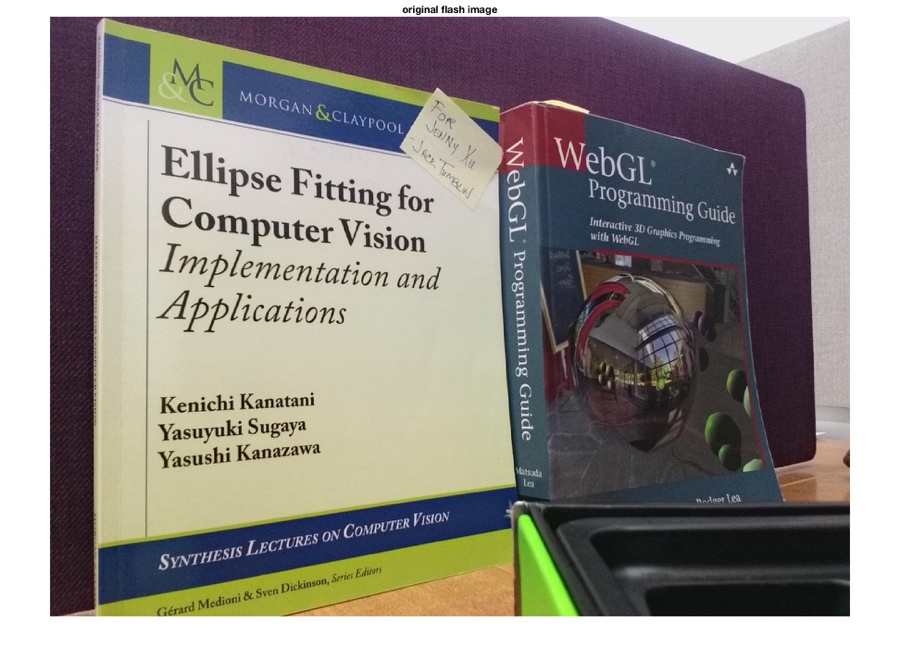
****

Figure 2 original flash image

**2. Denoise the no-flash image I captured in part 1**

(1) I cut the image size into 1024x768 to make the bilateral filter code run not too slowly. The cut images are as follows:

**A screenshot of a cell phone

Description automatically generated**

Figure 3 original no flash cut image

**A screenshot of a cell phone

Description automatically generated**

Figure 4 original flash cut image

(2) I denoise each color channel separately by using the bilateralFilter function.

The 9 pairs of parameters’ values I chose are:

sigmaSpatial1 = 16; sigmaRange1 = 0.05;

sigmaSpatial2 = 16; sigmaRange2 = 0.15;

sigmaSpatial3 = 16; sigmaRange3 = 0.25;

sigmaSpatial4 = 32; sigmaRange4 = 0.05;

sigmaSpatial5 = 32; sigmaRange5 = 0.15;

sigmaSpatial6 = 32; sigmaRange6 = 0.25;

sigmaSpatial7 = 64; sigmaRange7 = 0.05;

sigmaSpatial8 = 64; sigmaRange8 = 0.15;

sigmaSpatial9 = 64; sigmaRange9 = 0.25;

In case that each channel’s optimal values are not the same, I imshowed each channel’s denoised results. The results are as follows:

A screenshot of a cell phone

Description automatically generated

Figure 5 channel red

A screenshot of a cell phone

Description automatically generated

Figure 6 channel blue

A screenshot of a cell phone

Description automatically generated

Figure 7 channel green

According to the three channels results above, I found out that all the eighth results were best. So, the optimal filter settings I chose are:

sigmaSpatial8 = 64, sigmaRange8 = 0.15.

And the denoised RGB results are as follows:

A screenshot of a cell phone

Description automatically generated

Figure 8 RGB

Thus, the optimal denoised no flash cut image is as follows:

A screenshot of a cell phone

Description automatically generated

Figure 9 denoised no flash cut image

**Problems I encountered:**

The bilateral filter MATLAB code can only denoise greyscale image, so at first I denoised each color channel separately. I did not know how to implement this step, and I did not know how to generate the colored picture like the exampled picture on the website. So I referred to some websites to find ways. I found out that I can use Matlab code below to separate each RGB channel:

*red = img(:,:,1); % Red channel*

*green = img(:,:,2); % Green channel*

*blue = img(:,:,3); % Blue channel*

and combine each RGB channel back to original image by using:

*back\_to\_original\_img = cat(3, red, green, blue);*

And this method worked out.

**3. Extract the details from the flash image and fuse the images**

I applied the bilateral filter to the no flash image and the flash image.

The denoised images are as follows:

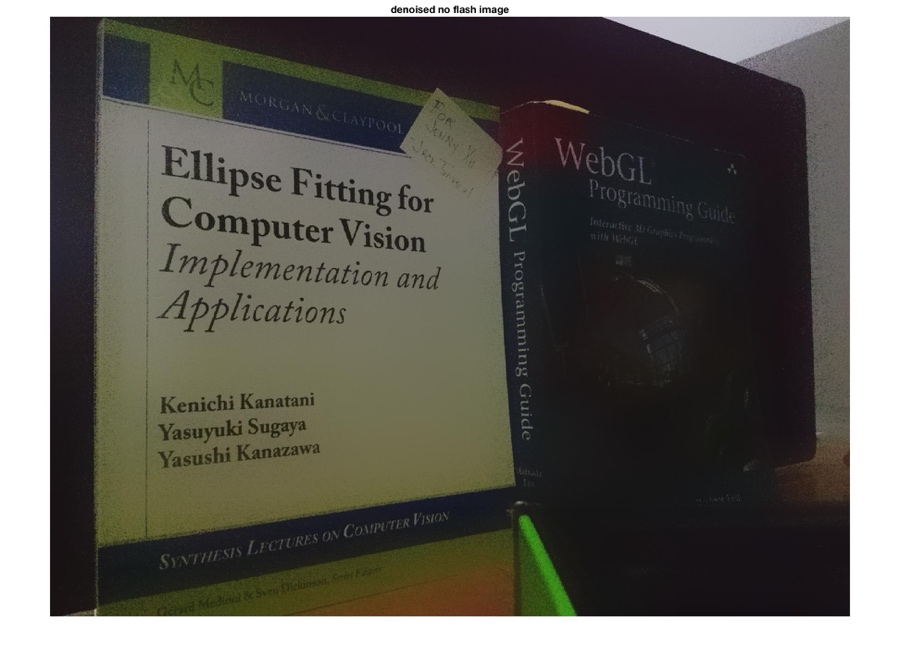


Figure 10 denoised no flash image

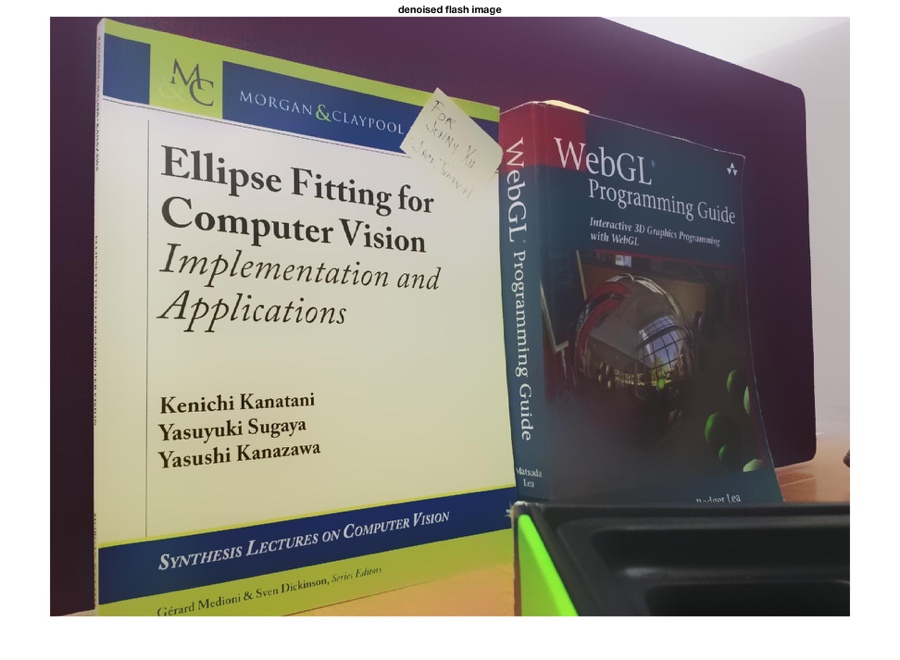


Figure 11 denoised flash image

Then I fused two denoised results by using the following equation:

which is the denoised no flash image, is the denoised flash image, and is the flash image.

The fused result is as follows:

A picture containing wall, indoor

Description automatically generated

Figure 12 fused image

Comparing the fused image with original no flash image, I found that the original image is denoised quite a lot.

**Problem I encountered:**

When I implemented the equation

I did not consider the situation that the image is three-dimensional matrix, so the previous Matlab code just did not work out. After using (:,:,:) I got a fused image.

However, at first, I imshowed the fused image and got the result below:

A screenshot of a cell phone

Description automatically generated

Figure 13 wrong image

It was so white, obviously it was the wrong result. I referred to several websites and found out that when I implemented imshow() to display an image, the double type is considered to be in the range of 0~1, that is, when a value is greater than 1, it is displayed as white. However, imshow() shows unit8, which is in the range of 0~255. Therefore, when display a double type image, I need to normalize the value to 0~1, or convert the double type value 0~255 to unit8 type. To implement it, I can use imshow(img/255). Thus, I modified my code and the result turned out normally.