EE 341 Summer 2018
Lab 1 Introduction to Image Processing

Professor: Tai-Chang Chen

July 6th, 2018

He Feng 1427841
Hanling Lei 1673094
Zikai Yang 1435493

Introduction

This lab introduces some basic concepts in image processing. We initially learn the way to transfer the image into grayscale format, then we used the grayscale image to find the convolution such that we can change the scale or other factors of an image. The lab also introduce the method to reverse the image based on x-axis, y-axis, and o-coordinate. Double type of image and uint8 type of image are also introduced through this lab. Generally this lab covers most of the basic skills in image processing, at the end of the lab we can learn the way to transfer the type of image, to reverse the image, and to change the scale of each images.

Results and Discussion

Exercise 1

The first exercise is the introduction of the whole image processing skills, and it covers most of important factors in this lab section. Initially, we set the image which we want to change by using the imread function. Then we used the code

```
image_gray = rgb2gray(image);
```

to change the image into grayscale format. This code will be useful in the following exercises, and it is especially important to detect the edges of each photo.

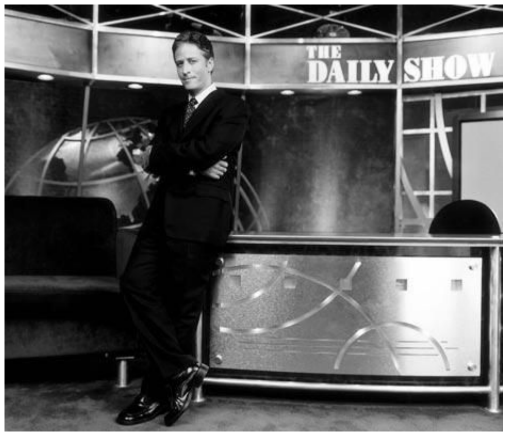
Then we use the imwrite function to display the grayscale photo such that we can compare it with the other photos we get later. After showing the initial grayscale graph, we used size function to determine the size of different variable, and we used the title function to add some words in the displayed photo above. Later we put the vertical convolution kernel and horizontal convolution kernel in the editor given by the spec, and we used double function to the grayscale image to convert the image to float point values.

After the steps above, we determined the row gradient magnitude, we can keep the vertical edges; and we determined the column gradient magnitude, which can keep the horizontal edges. The overall gradient can keep both the horizontal edges and the vertical edges. Then we used uint8 function to convert the values above into uint8 type, and the output we got are going to be displayed by using imshow function. Finally we used imwrite function to store the images.

After doing the first exercise, we got the grayscale image, the images which keep horizontal edges, vertical edges, and both of two edges above. This is the original image which also will be used in the following exercises. The name of this image is called “DailyShow.jpg”.

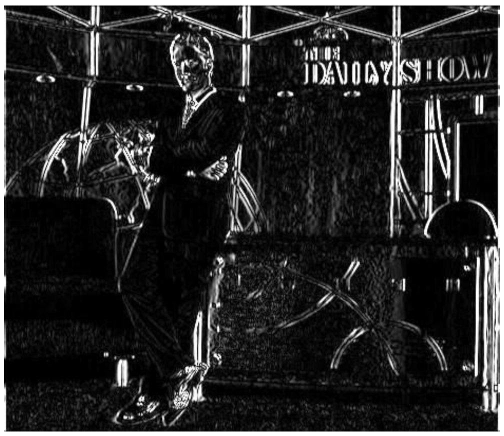
Original image:

The grayscale image:

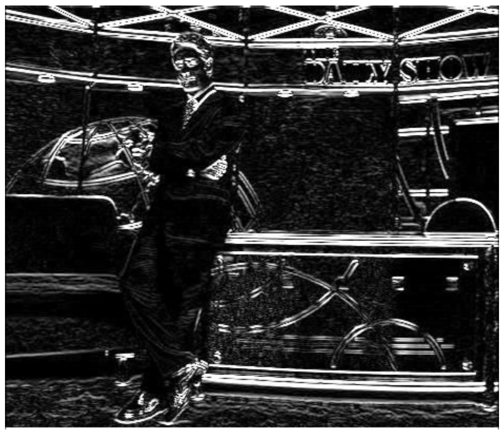


The dimensions of the original image is 400x468x3, and the dimensions of the grayscale format image is 400x468. The comment has been added on the grayscale format image by using the title function. Here are the images we got from this exercise:

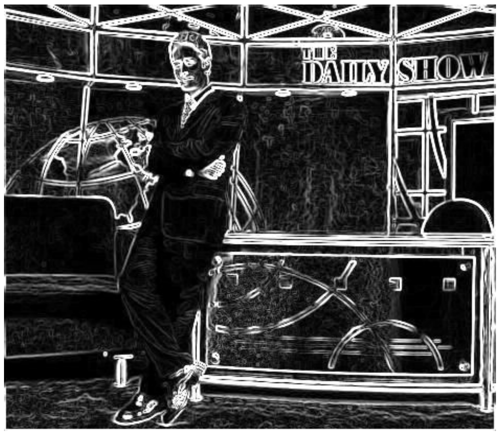
The one which keeps the vertical edges:



The one which keeps the horizontal edges:



The one which keeps both vertical edges and horizontal edges:

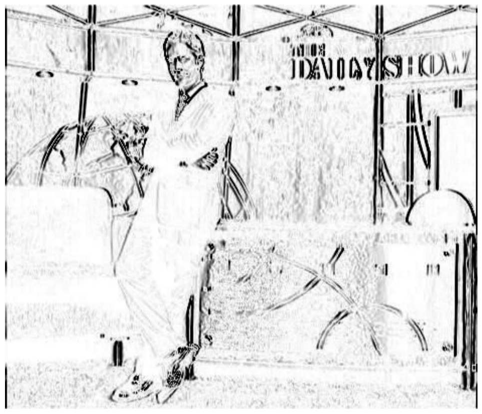


The optional problem ask us to map the original darkest area to the brightest one, and the original brightest area to the darkest one. Therefore we used the code

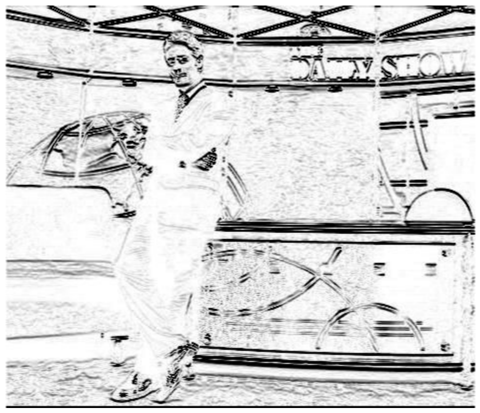
```
optional_image_1 = uint8(255) - new_row;
optional_image_2 = uint8(255) - new_column;
optional_image_3 = uint8(255) - new_overall;
```

to reverse brightness and darkness of the image which keeps the vertical edges, horizontal edges, and both horizontal edges and vertical edges. Here are the output:

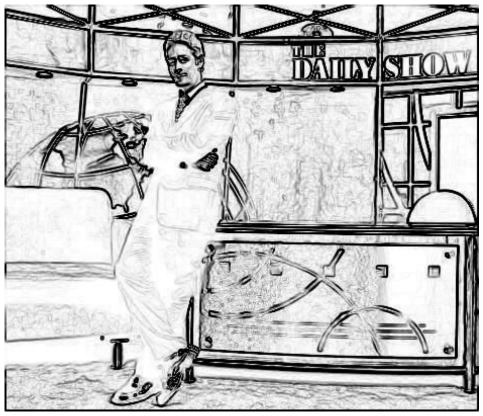
The one which keeps the vertical edges:



The one which keeps the horizontal edges:



The one which keeps both vertical edges and horizontal edges:

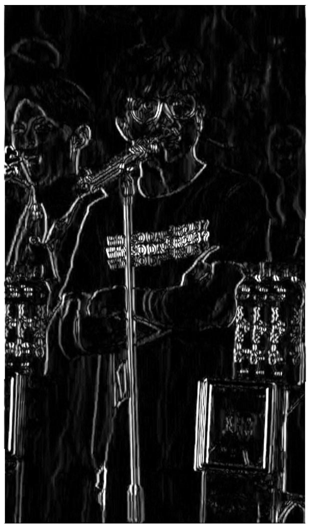


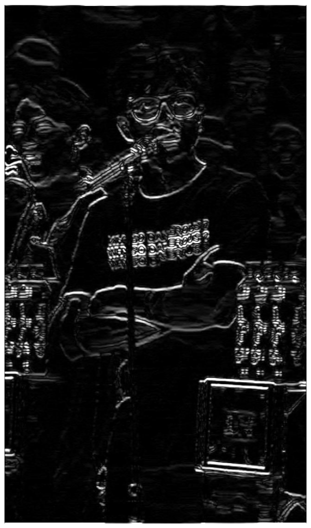
Exercise 2

This exercise is similar to the previous exercise, and we chose our own image for the processing. In this exercise we are going to display the edge gradient magnitudes as we did in Ex1. Here is the original image:

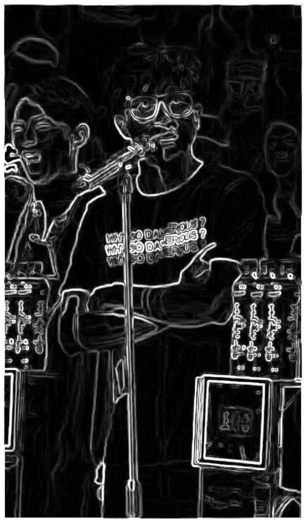
We used the same horizontal kernel and vertical kernel, and we used them to determine the convolution. Then we found the row gradient, column gradient and overall gradient magnitude based on the convolution. We will use the three magnitudes above to get the edge images. Here are the three edge images:

The one which keeps the vertical edges: The one which keeps the horizontal edges:





The one which keeps both vertical edges and horizontal edges:



Exercise 3

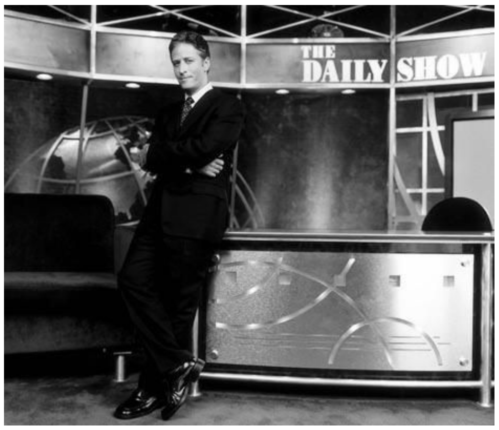
We will learn scaling in this exercise. There are two types of scaling introduced in the spec. The first one is the basic scaling, and the second one is average first and then scale the image. We are going to make one function for each of them such that we can scale the image by using the scale factor 2 and 5.

Initially we built the basic scaling function. We determine the relation between the scale factor and 2. If the scale factor is an even number, we are going to get the upper left number of the square of S^2, S in this case represent the scale factor. If the scale factor is an odd number, we keep the center-most number and build a new matrix.

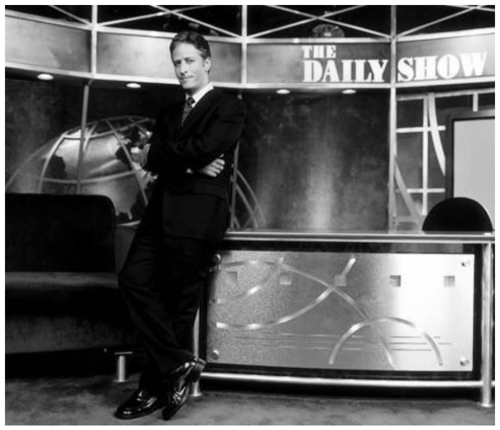
Then we built the second scaling function. We build a matrix full of one, and then we used the scale factor and the matrix described above to set a kernel, and we changed the image type to double and make the convolution by using the double type image and the kernel. Then we changed the image type to uint8, and we call the basic scaling function described above. The input is the new uint8 image and scale factor, and finally we can get the ideal image we determined.

In the Ex3 script, we initially transfer the image into grayscale format, then we use the two functions above and scale factors 2 and 5 to figure out the new images. Because it is hard to show the size difference in the report, so I put the result images in both report and the lab 2 file which will be uploaded with the report. Please check the folder for the accurate images. I will upload the two images which are the image from the basic scaling and average scaling by using the scale factor 2 such that we can compare the difference between the two scaling functions. The two images below can only show the image details difference, and their image size is not the real image size after scaling.

Basic scale with the scale factor 2:



Average scale with the scale factor 2:



By observing two images above, we think the image after average scaling looks better because we can see more details in the image, and the image looks clearer.

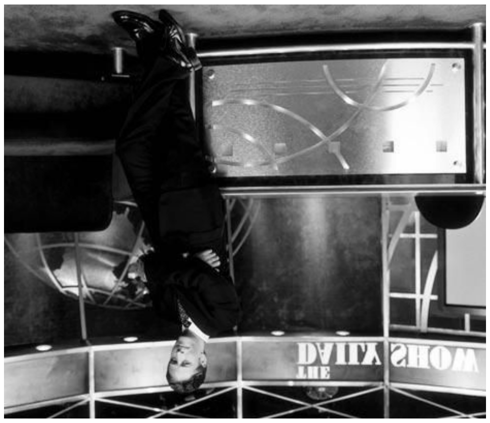
Exercise 4

This exercise is the most interesting exercise to us. We initially guessed the result. The first part should reverse the image by x-axis because it only changes the row position and the new row element is N-n+1; the second part should reverse the image by y-axis because it only changes the column position and the new column element is M-m+1; The third part should reverse the image by o-coordinate because it combines the changes in the previous two parts. Then we do the lab to verify the guessing. We initially transfer the original image into the grayscale format, then we changed the position of the pixels to reverse the image. To reverse the image by x-axis, we kept the columns the same and change the each row's position. Similarly, we kept the rows position and changed the column position to reverse by y-axis; and we changed both column and row position to reverse by o-coordinate. We used for loop to change the pixels position, and here is an example for the reversing by x-axis:

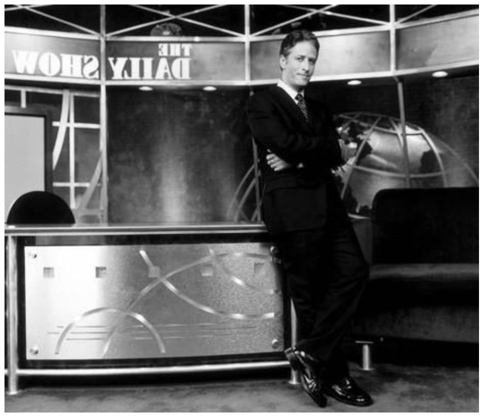
```
for m = 1:M
  for n = 1:N
    new_image_1(n, m) = image_gray(N-n+1,m);
  end
End
```

After using the for loop, we used imshow function and imwrite function to show and store the output images. We also need to transfer the image into grayscale format at first in this exercise. After printing out the result images, we verified that our guessing is correct. Here are the output images:

Part 1:



Part 2:



Part 3:

Exercise 5

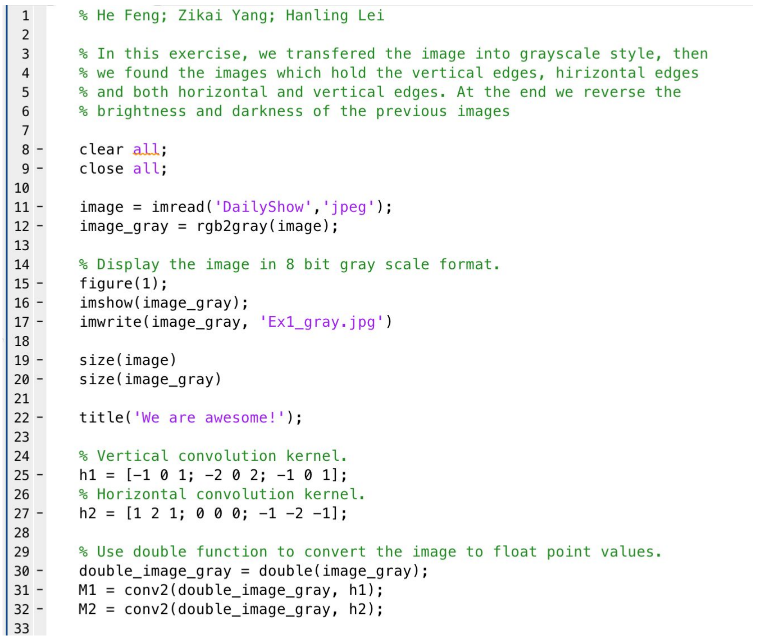
In this exercise we tried to make the image bigger by using linear interpolation. We increased the row length by 2 and the column length by 2 as well. As the spec said, we build a new function to such that we can make the image as big as we want which has be described above. In this function we change the image type to double, and then we used the interp2 function to make the size changes. In this case this step is called bilinear interpolation. After set up the function, we built the Ex5 script. We initially transfer the image to grayscale format, then we called this function on the grayscale format image to increase the size. Finally we change the image type back to uint8 and show and store the new images by using imshow and imwrite. Because the report cannot show the size difference accurately, so please check the folder if you want to see the size difference.

Conclusion:

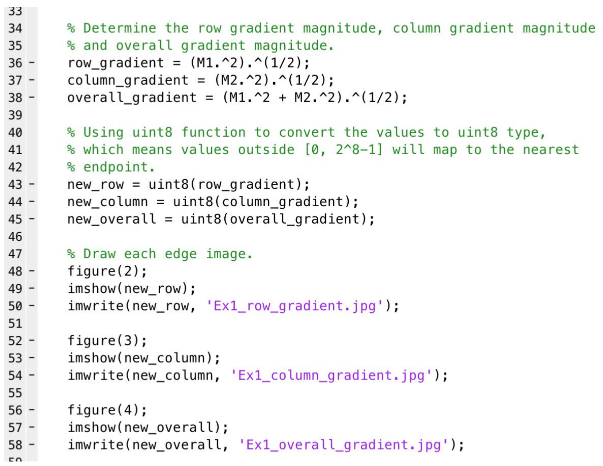
This lab is the introduction for the image processing, and we learnt a lot through this lab. We know the importance transfer the image into grayscale format, and we also learnt the difference between double type images and uint8 type images. We also learnt the way to scale the image and make the image larger. To change the size of the images, it is much easier to build a function instead of writing everything in one script. The hardest part of the lab is the fifth exercise. Even though the spec has already give us a function, but it would be really hard if the spec does not give us a function or the spec requires us to increase the image by a different factor instead of 2. The most interesting part is the fourth exercise, we can reverse the images and the whole exercise is easy to understand and to work on. To conclude, we know some basic steps in image processing after this lab, and we are looking forward for the upcoming labs to improve our image processing skills.

Attached Code:

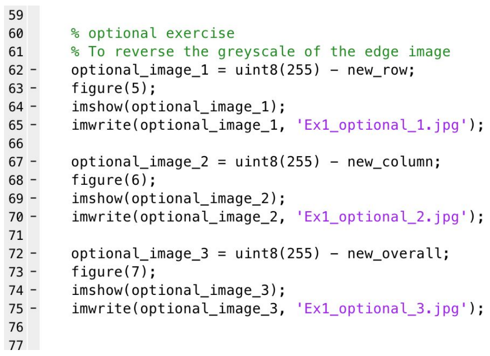
Ex1.m



```
% He Feng; Zikai Yang; Hanling Lei

% In this exercise, we transfered the image into grayscale style, then
% we found the images which hold the vertical edges, hirizontal edges
% and both horizontal and vertical edges. At the end we reverse the
% brightness and darkness of the previous images

clear all;
close all;

image = imread('DailyShow','jpeg');
image_gray = rgb2gray(image);

% Display the image in 8 bit gray scale format.
figure(1);
imshow(image_gray);
imwrite(image_gray, 'Ex1_gray.jpg')

size(image)
size(image_gray)

title('We are awesome!');

% Vertical convolution kernel.
h1 = [-1 0 1; -2 0 2; -1 0 1];
% Horizontal convolution kernel.
h2 = [1 2 1; 0 0 0; -1 -2 -1];

% Use double function to convert the image to float point values.
double_image_gray = double(image_gray);
M1 = conv2(double_image_gray, h1);
M2 = conv2(double_image_gray, h2);

```

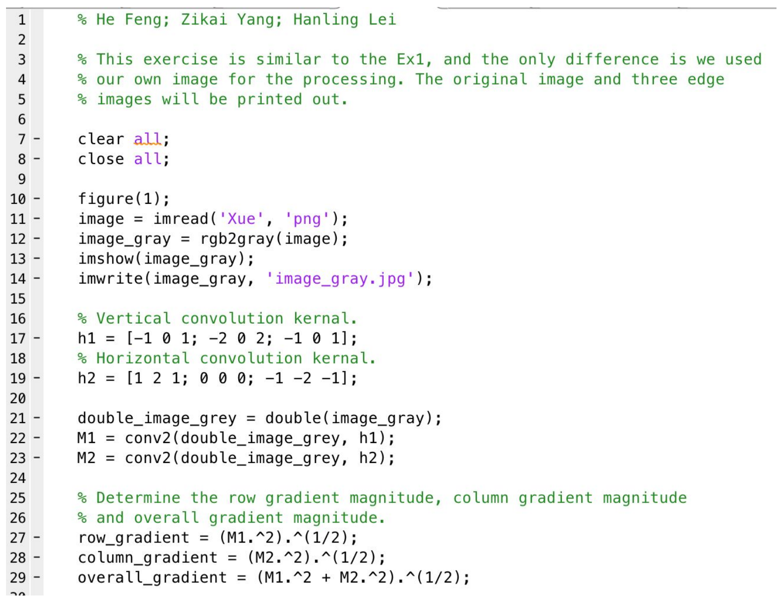


```

% Determine the row gradient magnitude, column gradient magnitude
% and overall gradient magnitude.
row_gradient = (M1.^2).^(1/2);
column_gradient = (M2.^2).^(1/2);
overall_gradient = (M1.^2 + M2.^2).^(1/2);

% Using uint8 function to convert the values to uint8 type,
% which means values outside [0, 2^8-1] will map to the nearest
% endpoint.
new_row = uint8(row_gradient);
new_column = uint8(column_gradient);
new_overall = uint8(overall_gradient);

% Draw each edge image.
figure(2);
imshow(new_row);
imwrite(new_row, 'Ex1_row_gradient.jpg');

figure(3);
imshow(new_column);
imwrite(new_column, 'Ex1_column_gradient.jpg');

figure(4);
imshow(new_overall);
imwrite(new_overall, 'Ex1_overall_gradient.jpg');
```

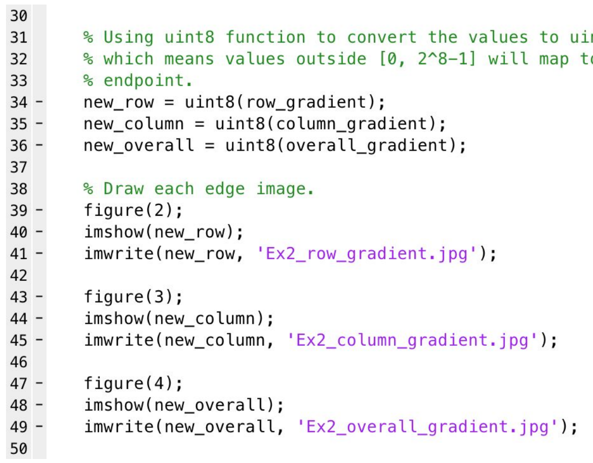


```

% optional exercise
% To reverse the greyscale of the edge image
optional_image_1 = uint8(255) - new_row;
figure(5);
imshow(optional_image_1);
imwrite(optional_image_1, 'Ex1_optional_1.jpg');

optional_image_2 = uint8(255) - new_column;
figure(6);
imshow(optional_image_2);
imwrite(optional_image_2, 'Ex1_optional_2.jpg');

optional_image_3 = uint8(255) - new_overall;
figure(7);
imshow(optional_image_3);
imwrite(optional_image_3, 'Ex1_optional_3.jpg');


```

Ex2.m

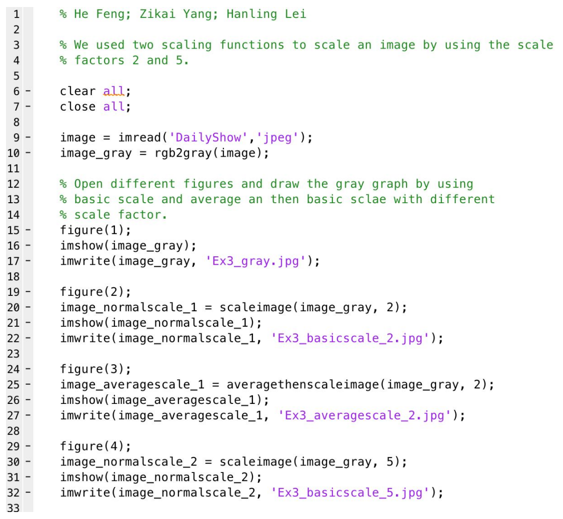


```
% He Feng; Zikai Yang; Hanling Lei

% This exercise is similar to the Ex1, and the only difference is we used
% our own image for the processing. The original image and three edge
% images will be printed out.

clear all;
close all;

figure(1);
image = imread('Xue', 'png');
image_gray = rgb2gray(image);
imshow(image_gray);
imwrite(image_gray, 'image_gray.jpg');

% Vertical convolution kernal.
h1 = [-1 0 1; -2 0 2; -1 0 1];
% Horizontal convolution kernal.
h2 = [1 2 1; 0 0 0; -1 -2 -1];

double_image_grey = double(image_gray);
M1 = conv2(double_image_grey, h1);
M2 = conv2(double_image_grey, h2);

% Determine the row gradient magnitude, column gradient magnitude
% and overall gradient magnitude.
row_gradient = (M1.^2).^(1/2);
column_gradient = (M2.^2).^(1/2);
overall_gradient = (M1.^2 + M2.^2).^(1/2);
```

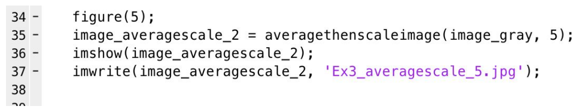


```

% Using uint8 function to convert the values to uint8 type,
% which means values outside [0, 2^8-1] will map to the nearest
% endpoint.
new_row = uint8(row_gradient);
new_column = uint8(column_gradient);
new_overall = uint8(overall_gradient);

% Draw each edge image.
figure(2);
imshow(new_row);
imwrite(new_row, 'Ex2_row_gradient.jpg');

figure(3);
imshow(new_column);
imwrite(new_column, 'Ex2_column_gradient.jpg');

figure(4);
imshow(new_overall);
imwrite(new_overall, 'Ex2_overall_gradient.jpg');

```

Ex3.m

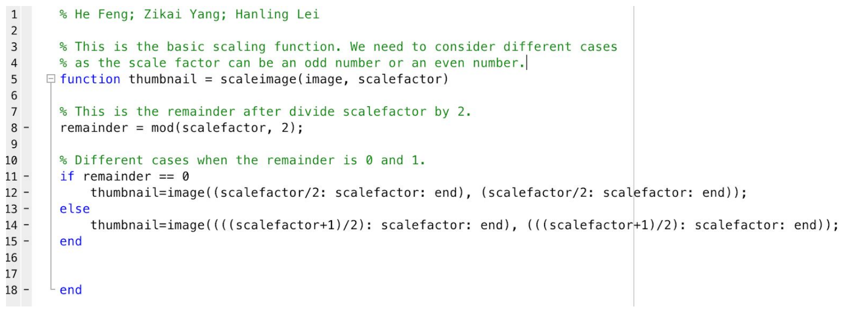


```matlab
% He Feng; Zikai Yang; Hanling Lei

% We used two scaling functions to scale an image by using the scale
% factors 2 and 5.

clear all;
close all;

image = imread('DailyShow','jpeg');
image_gray = rgb2gray(image);

% Open different figures and draw the gray graph by using
% basic scale and average an then basic sclae with different
% scale factor.
figure(1);
imshow(image_gray);
imwrite(image_gray, 'Ex3_gray.jpg');

figure(2);
image_normalscale_1 = scaleimage(image_gray, 2);
imshow(image_normalscale_1);
imwrite(image_normalscale_1, 'Ex3_basicscale_2.jpg');

figure(3);
image_averagescale_1 = averagethenscaleimage(image_gray, 2);
imshow(image_averagescale_1);
imwrite(image_averagescale_1, 'Ex3_averagescale_2.jpg');

figure(4);
image_normalscale_2 = scaleimage(image_gray, 5);
imshow(image_normalscale_2);
imwrite(image_normalscale_2, 'Ex3_basicscale_5.jpg');

```



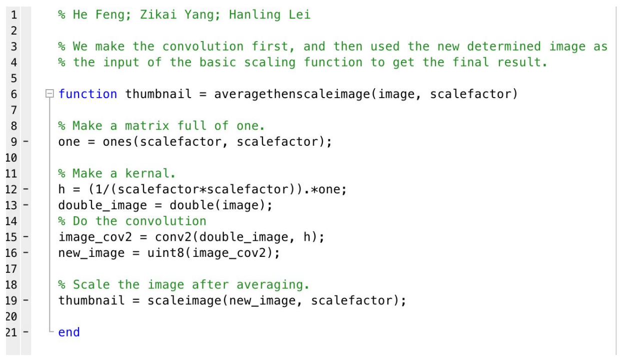
```matlab
figure(5);
image_averagescale_2 = averagethenscaleimage(image_gray, 5);
imshow(image_averagescale_2);
imwrite(image_averagescale_2, 'Ex3_averagescale_5.jpg');

```

scaleimage.m



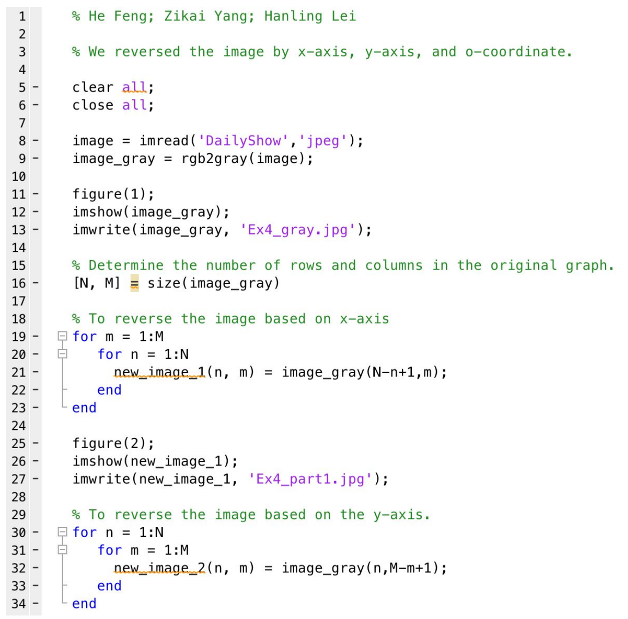
```matlab
% He Feng; Zikai Yang; Hanling Lei

% This is the basic scaling function. We need to consider different cases
% as the scale factor can be an odd number or an even number.
function thumbnail = scaleimage(image, scalefactor)

% This is the remainder after divide scalefactor by 2.
remainder = mod(scalefactor, 2);

% Different cases when the remainder is 0 and 1.
if remainder == 0
    thumbnail=image((scalefactor/2: scalefactor: end), (scalefactor/2: scalefactor: end));
else
    thumbnail=image((((scalefactor+1)/2): scalefactor: end), (((scalefactor+1)/2): scalefactor: end));
end


end
```

averagethenscaleimage.m

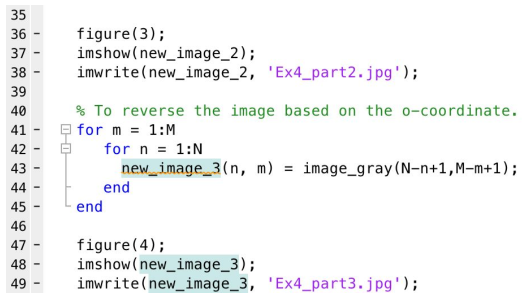


```matlab
% He Feng; Zikai Yang; Hanling Lei

% We make the convolution first, and then used the new determined image as
% the input of the basic scaling function to get the final result.

function thumbnail = averagethenscaleimage(image, scalefactor)

% Make a matrix full of one.
one = ones(scalefactor, scalefactor);

% Make a kernal.
h = (1/(scalefactor*scalefactor)).*one;
double_image = double(image);
% Do the convolution
image_cov2 = conv2(double_image, h);
new_image = uint8(image_cov2);

% Scale the image after averaging.
thumbnail = scaleimage(new_image, scalefactor);

end
```

Ex4.m

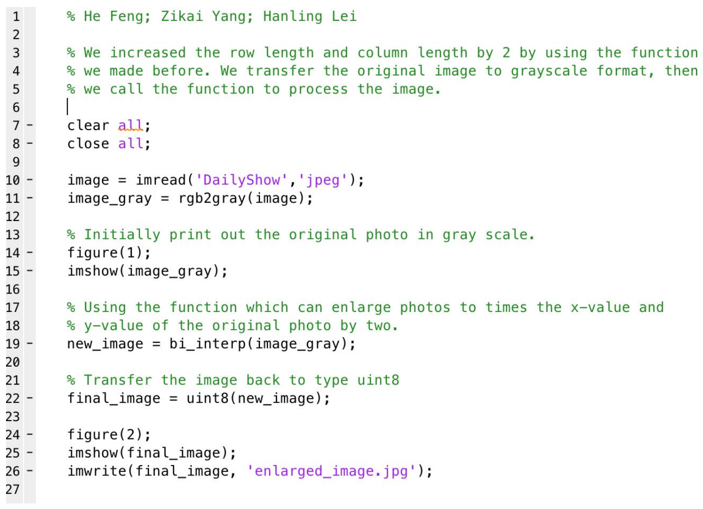


```matlab
% He Feng; Zikai Yang; Hanling Lei

% We reversed the image by x-axis, y-axis, and o-coordinate.

clear all;
close all;

image = imread('DailyShow','jpeg');
image_gray = rgb2gray(image);

figure(1);
imshow(image_gray);
imwrite(image_gray, 'Ex4_gray.jpg');

% Determine the number of rows and columns in the original graph.
[N, M] = size(image_gray)

% To reverse the image based on x-axis
for m = 1:M
   for n = 1:N
     new_image_1(n, m) = image_gray(N-n+1,m);
   end
end

figure(2);
imshow(new_image_1);
imwrite(new_image_1, 'Ex4_part1.jpg');

% To reverse the image based on the y-axis.
for n = 1:N
   for m = 1:M
     new_image_2(n, m) = image_gray(n,M-m+1);
   end
end
```



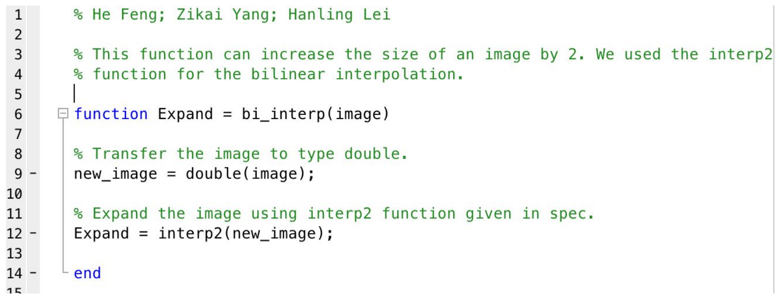
```matlab

figure(3);
imshow(new_image_2);
imwrite(new_image_2, 'Ex4_part2.jpg');

% To reverse the image based on the o-coordinate.
for m = 1:M
   for n = 1:N
     new_image_3(n, m) = image_gray(N-n+1,M-m+1);
   end
end

figure(4);
imshow(new_image_3);
imwrite(new_image_3, 'Ex4_part3.jpg');
```

Ex5.m



```
% He Feng; Zikai Yang; Hanling Lei

% We increased the row length and column length by 2 by using the function
% we made before. We transfer the original image to grayscale format, then
% we call the function to process the image.

clear all;
close all;

image = imread('DailyShow','jpeg');
image_gray = rgb2gray(image);

% Initially print out the original photo in gray scale.
figure(1);
imshow(image_gray);

% Using the function which can enlarge photos to times the x-value and
% y-value of the original photo by two.
new_image = bi_interp(image_gray);

% Transfer the image back to type uint8
final_image = uint8(new_image);

figure(2);
imshow(final_image);
imwrite(final_image, 'enlarged_image.jpg');
```

bi_interp.m



```
% He Feng; Zikai Yang; Hanling Lei

% This function can increase the size of an image by 2. We used the interp2
% function for the bilinear interpolation.

function Expand = bi_interp(image)

% Transfer the image to type double.
new_image = double(image);

% Expand the image using interp2 function given in spec.
Expand = interp2(new_image);

end
```