# Digital Image Processing HW4

電機四乙10828241 陳大荃

November 3, 2022

## 1. Problem Description

Show output images of Lena (Grayscale) using the three spatial filters below.

1. $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

2. $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

3. $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Command "filter2" is not allowed in this homework.

## 2. Program/Code

### 2.1 Filter 1

```
mask = ones(mask_size, mask_size) * 1 / (mask_size * mask_size);
t = fix(mask_size / 2);
[h, w] = size(img);
img_temp = img;
for i = 1+t:h-t
    for j = 1+t:w-t
        img_temp(i, j) = sum(sum(img(i-t:i+t, j-t:j+t) .* mask));
    end
end
img = img_temp;
```

### 2.2 Filter 2

```
mask = [0 -1 0; -1 4 -1; 0 -1 0];
[h, w] = size(img);
img_temp = zeros(h, w);
% masking - edge detection
for i = 2:h-1
     for j = 2:w-1
          img_temp(i, j) = sum(sum(img(i-1:i+1, j-1:j+1) .* mask));
     end
end
% scaling transform
if scaling_size ~= 0
     gh = max(max(img_temp)) * scaling_size;
     gl = min(min(img_temp)) * scaling_size;
     img = (img_temp - gl) / (gh - gl) * 255;
elseif scaling_size == -1
     img = abs(img_temp);
else
     img = img_temp;
end
```

### 2.3 Filter 3

```
mask = [0 -1 0; -1 5 -1; 0 -1 0];
[h, w] = size(img);
img_temp = img;
% masking - edge detection
for i = 2:h-1
     for j = 2:w-1
          img_temp(i, j) = sum(sum(img(i-1:i+1, j-1:j+1) .* mask));
     end
end
% scaling transform
if scaling_size ~= 0
     gh = max(max(img_temp)) * scaling_size;
     gl = min(min(img_temp)) * scaling_size;
     img = (img_temp - gl) / (gh - gl) * 255;
elseif scaling_size == -1
     img = abs(img_temp);
else
     img = img_temp;
end
```

Since this code will execute all three filters in a single run, there is no need to specify which filter to use. Following is the code to execute this function. (See this function in GitHub Repository)

```
>> imgsrc = 'lena.tif';
>> imgdst = 'output.jpg';
>> show = 1;
>> export = 0;
>> spatial_filter(imgsrc, imgdst, show, export)
```

## 3. Result and Discussion

### 3.1 Method

In this homework, different filters are placed on each pixel, and the edge pixels are ignored. "for" loop is used to do 2D convolution between the original image and filter.

One thing to be cautious of is that the result of processed pixels needs to be stored in a different matrix other than the original one. If not, the pixel values will be more and more extreme on both the positive and negative axes.

### 3.2 Result

(a) Original (b) Filter 1 (c) Filter 2 (d) Filter 3

Figure 1: Grayscale image Result.

### 3.3 Discussion

#### 3.3.1 Colored Conversion

Colored images can also be converted by separating R, G, and B, converting them one by one, and turning them back into one.

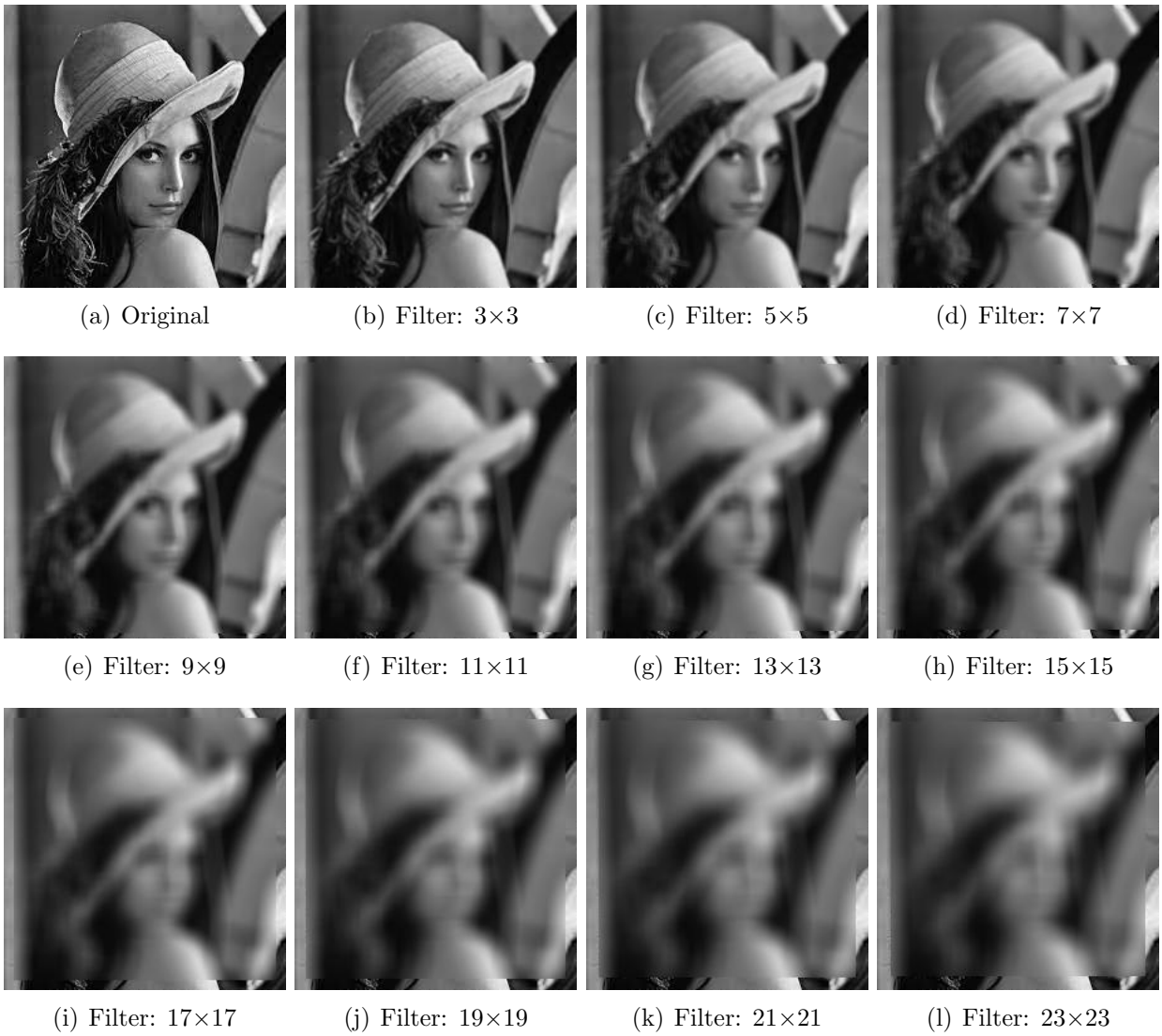
(a) Original (b) Filter 1 (c) Filter 2 (d) Filter 3

Figure 2: Colored image Result.

### 3.3.2 Filter 1 Adjustment

In this section, I adjust the size of this filter. Due to its nature, it is easily scalable. And also, due to its nature, the width and height of filters need to be odd numbers.

This is my code. and This is where the pictures are stored.



(a) Original (b) Filter: 3×3 (c) Filter: 5×5 (d) Filter: 7×7

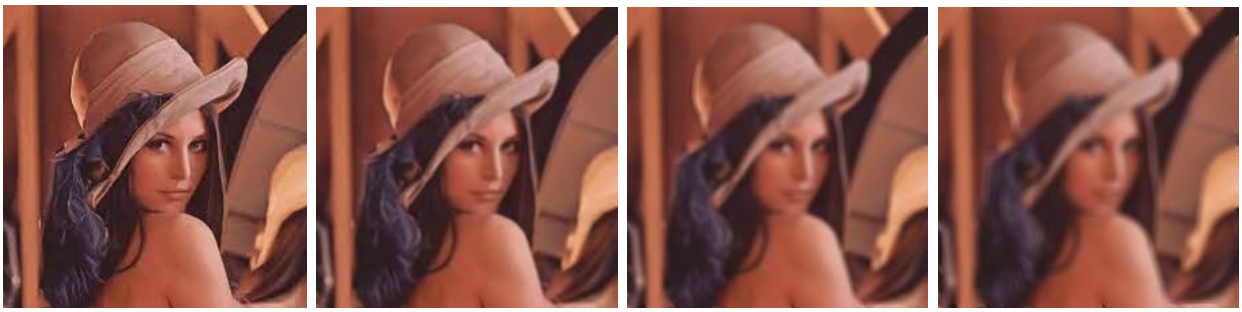
(e) Filter: 9×9 (f) Filter: 11×11 (g) Filter: 13×13 (h) Filter: 15×15

(i) Filter: 17×17 (j) Filter: 19×19 (k) Filter: 21×21 (l) Filter: 23×23

Figure 3: Grayscale image Result.

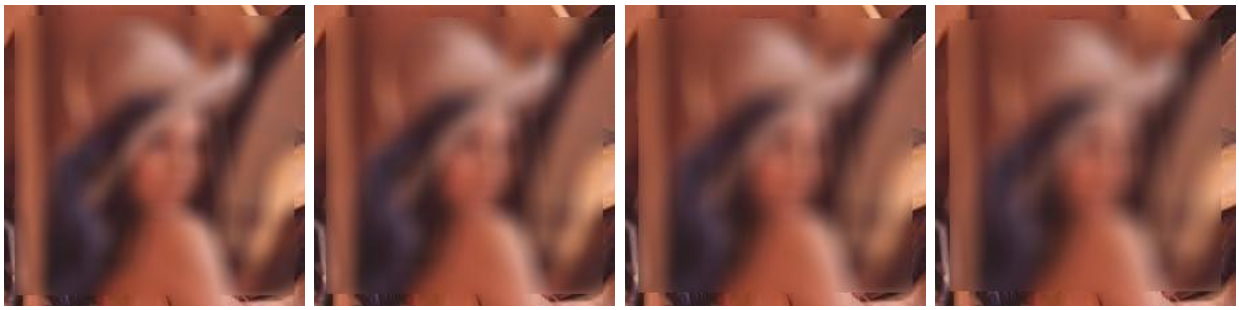
I code this filter not to expand the image matrix but use the available pixels to convolute with the filter. It is why as the filter size increases, the unprocessed frame on the edge of the photos grows wider and wider.

I made the same adjustment to the colored image as well.



(a) Original (b) Filter: 3×3 (c) Filter: 5×5 (d) Filter: 7×7

(e) Filter: 9×9 (f) Filter: 11×11 (g) Filter: 13×13 (h) Filter: 15×15



(i) Filter: 17×17 (j) Filter: 19×19 (k) Filter: 21×21 (l) Filter: 23×23

Figure 4: Colored image Result.

Based on the experiment result, as the filter's size increases, the photo becomes more blurry. The same effect applies to both grayscale and colored image.

### 3.3.3 Filter 2&3 Adjustment

After filter 2 is applied, the generated image usually has a large range of grayscale values. For the process Lena image, the range is approximately -500 to +500. The result available for implementing filter 2 is usually only used the numbers between 0 and 255 and clips out all the rest of the extreme values.

I'm going to experiment not on clipping ranges but to perform scaling transform. My code can do scaling transforms with different maximum and minimum values. In the following images, 70% in the caption means the maximum value is it's 70% value, the same goes to the minimum value.

This is my code. and This is where the pictures are stored.

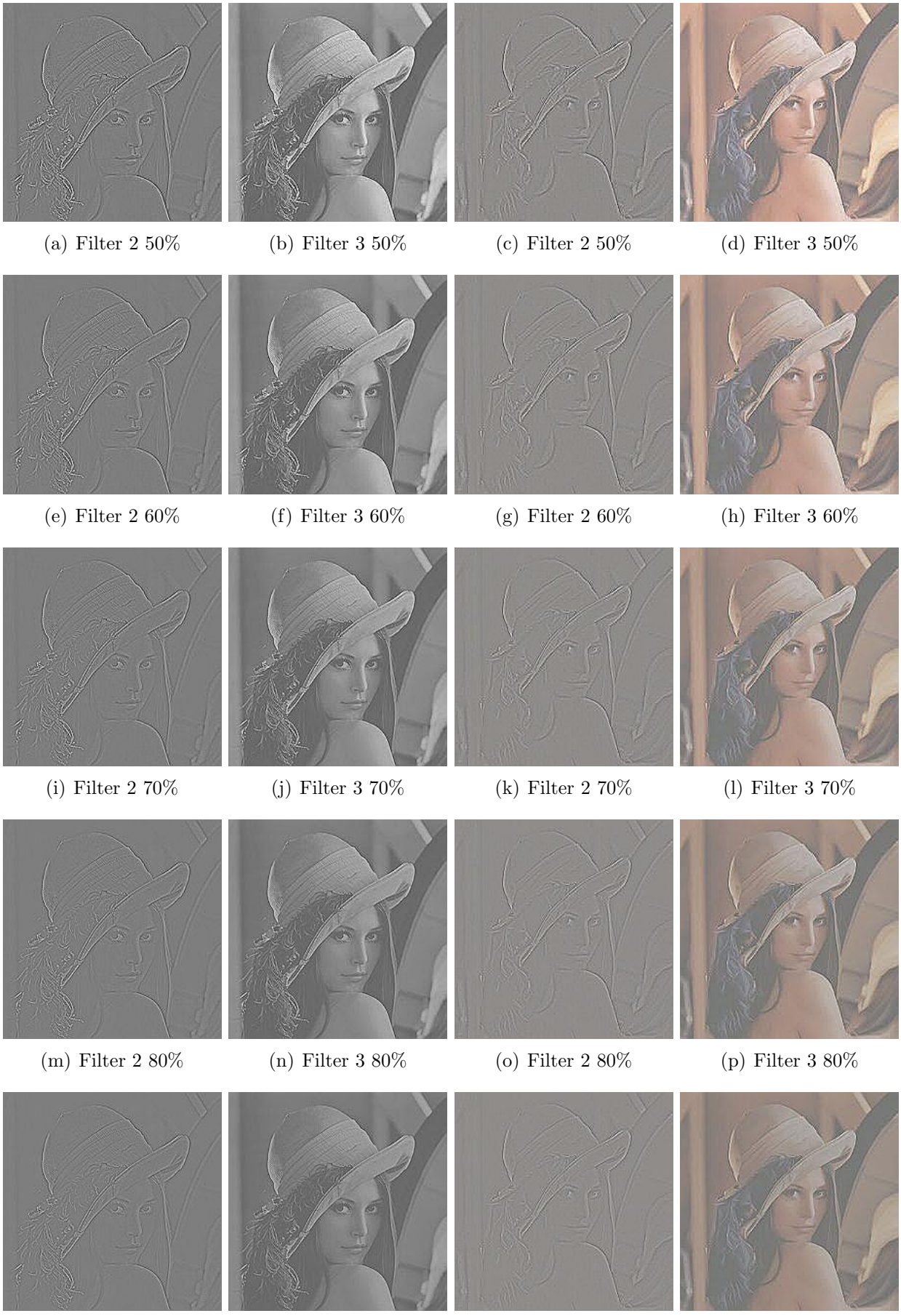
(a) Filter 2 10% (b) Filter 3 10% (c) Filter 2 10% (d) Filter 3 10%

(e) Filter 2 20% (f) Filter 3 20% (g) Filter 2 20% (h) Filter 3 20%

(i) Filter 2 30% (j) Filter 3 30% (k) Filter 2 30% (l) Filter 3 30%

(m) Filter 2 40% (n) Filter 3 40% (o) Filter 2 40% (p) Filter 3 40%

Figure 5: Result. (grayscale / colored)



(a) Filter 2 50% (b) Filter 3 50% (c) Filter 2 50% (d) Filter 3 50%

(e) Filter 2 60% (f) Filter 3 60% (g) Filter 2 60% (h) Filter 3 60%

(i) Filter 2 70% (j) Filter 3 70% (k) Filter 2 70% (l) Filter 3 70%

(m) Filter 2 80% (n) Filter 3 80% (o) Filter 2 80% (p) Filter 3 80%

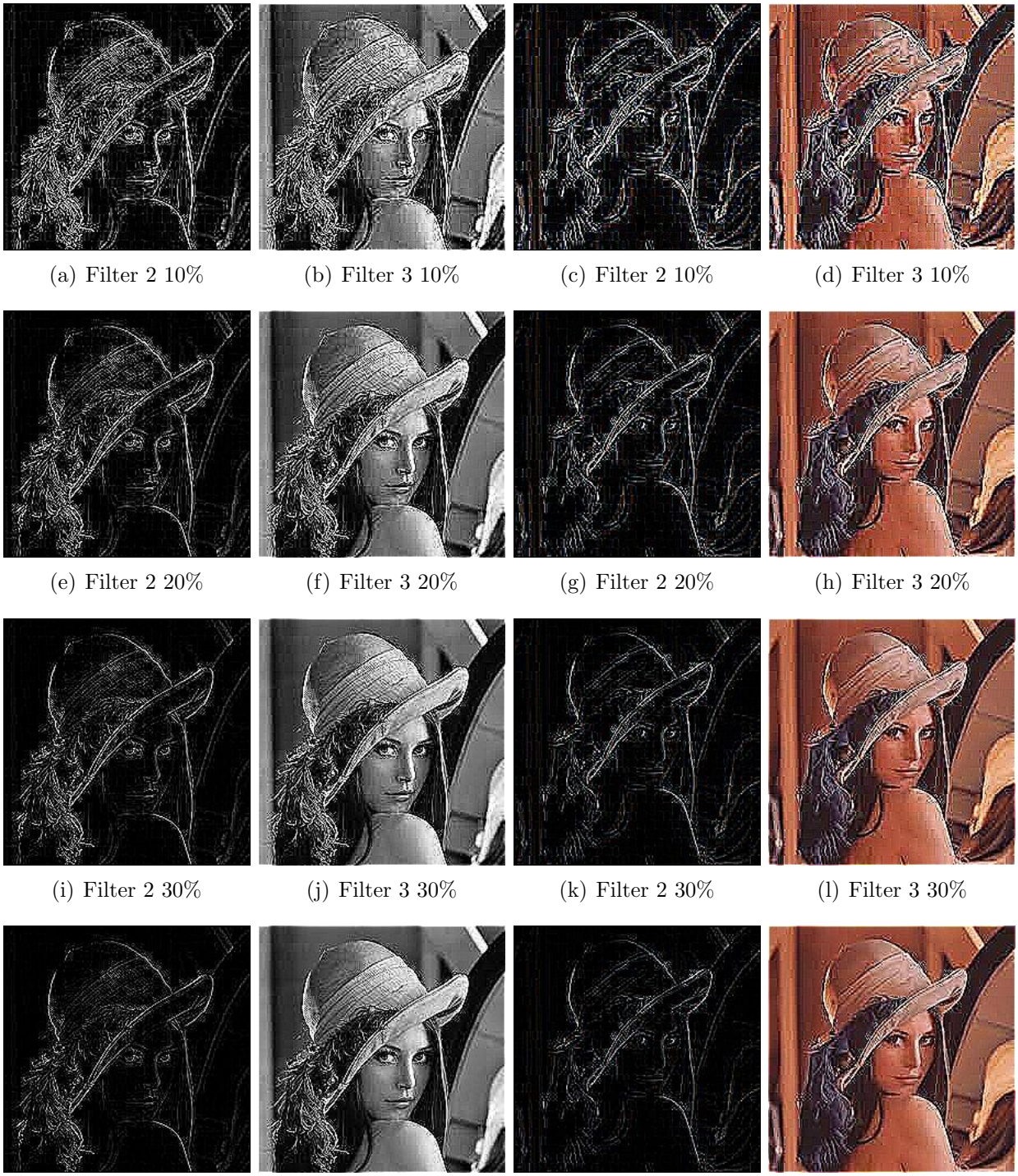
(q) Filter 2 90% (r) Filter 3 90% (s) Filter 2 90% (t) Filter 3 90%

Figure 6: Result. (grayscale / colored)

#### 3.3.3.1 Tweak scaling transform.

From the experiment result, if the negative values are not neglected or clipped, those values will affect most pixels instead of the edges we are looking for. Therefore the next section is to clip out the negative region and perform scaling transform on the positive range. Note that filter 3 is not directly applied since the original image is going to be transformed as well.

This is my code. and This is where the pictures are stored.



(a) Filter 2 10% (b) Filter 3 10% (c) Filter 2 10% (d) Filter 3 10%

(e) Filter 2 20% (f) Filter 3 20% (g) Filter 2 20% (h) Filter 3 20%

(i) Filter 2 30% (j) Filter 3 30% (k) Filter 2 30% (l) Filter 3 30%

(m) Filter 2 40% (n) Filter 3 40% (o) Filter 2 40% (p) Filter 3 40%

Figure 7: Result. (grayscale / colored)

(a) Filter 2 50% (b) Filter 3 50% (c) Filter 2 50% (d) Filter 3 50%

(e) Filter 2 60% (f) Filter 3 60% (g) Filter 2 60% (h) Filter 3 60%

(i) Filter 2 70% (j) Filter 3 70% (k) Filter 2 70% (l) Filter 3 70%

(m) Filter 2 80% (n) Filter 3 80% (o) Filter 2 80% (p) Filter 3 80%

(q) Filter 2 90% (r) Filter 3 90% (s) Filter 2 90% (t) Filter 3 90%

Figure 8: Result. (grayscale / colored)

#### 3.3.3.2 Replace scaling transform with clipping.

I also tried to use clipping to replace scaling transform. Still, negative values are all clipped, but adjust the positive limit with exponential numbers like $e^1$, $e^{1.5}$, $e^2$ .... Note that filter 3 is not directly applied since the original image is going to be transformed as well.

This is my code. and This is where the pictures are stored.

(a) Filter 2 $e^0$ (b) Filter 3 $e^0$ (c) Filter 2 $e^0$ (d) Filter 3 $e^0$

(e) Filter 2 $e^{0.5}$ (f) Filter 3 $e^{0.5}$ (g) Filter 2 $e^{0.5}$ (h) Filter 3 $e^{0.5}$

(i) Filter 2 $e^1$ (j) Filter 3 $e^1$ (k) Filter 2 $e^1$ (l) Filter 3 $e^1$

(m) Filter 2 $e^{1.5}$ (n) Filter 3 $e^{1.5}$ (o) Filter 2 $e^{1.5}$ (p) Filter 3 $e^{1.5}$

Figure 9: Result. (grayscale / colored)

(a) Filter 2 $e^2$ (b) Filter 3 $e^2$ (c) Filter 2 $e^2$ (d) Filter 3 $e^2$

(e) Filter 2 $e^{2.5}$ (f) Filter 3 $e^{2.5}$ (g) Filter 2 $e^{2.5}$ (h) Filter 3 $e^{2.5}$

(i) Filter 2 $e^3$ (j) Filter 3 $e^3$ (k) Filter 2 $e^3$ (l) Filter 3 $e^3$

(m) Filter 2 $e^{3.5}$ (n) Filter 3 $e^{3.5}$ (o) Filter 2 $e^{3.5}$ (p) Filter 3 $e^{3.5}$

(q) Filter 2 $e^4$ (r) Filter 3 $e^4$ (s) Filter 2 $e^4$ (t) Filter 3 $e^4$

Figure 10: Result. (grayscale / colored)

(a) Filter 2 $e^{4.5}$ (b) Filter 3 $e^{4.5}$ (c) Filter 2 $e^{4.5}$ (d) Filter 3 $e^{4.5}$

(e) Filter 2 $e^{5.0}$ (f) Filter 3 $e^{5.0}$ (g) Filter 2 $e^{5.0}$ (h) Filter 3 $e^{5.0}$

(i) Filter 2 $e^{5.5}$ (j) Filter 3 $e^{5.5}$ (k) Filter 2 $e^{5.5}$ (l) Filter 3 $e^{5.5}$

Figure 11: Result. (grayscale / colored)

Three things stand up from the pictures in sections 3.3.3.1 and 3.3.3.2. 1. Edge detection filter is less effective on colored images. 2. Despite that it is possible to give an even larger value to the clipping function, it isn't very meaningful because there isn't a pixel have a value that big. 3. Using scaling transform can extract more edges. Using clipping can perform roughly the same effect but less powerful.