Homework3 – Filters

Arya Araban

Abstract

In image processing, filters are mainly used to smoothen out the image, as well as enhancing or detecting edges throughout the image. We can apply filters in two domains: the frequency domain and the spatial domain. In this article we will explore filters throughout the spatial domain, how to use them, and their advantages / disadvantages.

Report Info

Date: 1399/09/06

Keywords:
Filters
Convolution
Sharpening
Edge detection
Noise reduction

1- Introduction

Filtering is a neighborhood operation, in which the value of any given pixel in the output image is determined by applying some algorithm to the values of the pixels in the neighborhood of the corresponding input pixel. A pixel's neighborhood is some set of pixels, defined by their locations relative to that pixel. In order to do this process in the spatial domain, we usually use what's known as a "Mask" (also known as the "Convolution Matrix" or "Kernel")

2- Technical Description

Problem 3.1.1:
Describe why box filters are bad smoothing filters. List all the reasons.

Solution:
The filter mask is calculated by inversing on the transfer function of the box filter, But by doing this, the coefficients decrease proportional to the distance from the center pixel. Which in turn makes the image blurry, however this does at least help in reducing the noise throughout the image.

Problem 3.1.2:
Do the bad features improve if we apply the filters several times?
Solution:
Yes, the bad features will improve if we Apply the filters several times, due to the center pixel getting closer to 0 with each iteration
Problem 3.1.3:
What is the resulting filter if apply the 3 × 3 box filter many times?
Solution:
In order to find a solution, we first need to understand convolution correctly, which we'll also make use of in the next questions.
Convolution is when a matrix is applied to a grayscale image using a mathematical operation comprised of integers. It works by determining the value of a central pixel by adding the weighted values of all its neighbors together. The output is a new modified filtered image

Solution:

Salt and pepper noise is a type of noise where only a few pixels are noisy, but their noise level is very high. The effect is similar to sprinkling white and black on the image. With the help of a filter known as the median filter, we can remove the S&P noise entirely.

The median filtering process is accomplished by sliding a matrix over the image. in each of the current states, we convert the matrix block containing the images pixel values into an array, then take the median of the array and place it in as the central pixel value. We keep doing this process until the matrix iterates over the entire image's pixel values.

Problem 3.2.2:

Add Gaussian noise with different variance, then apply filtering using the average and median filter.

Solution:

The Gaussian noise is a type of noise
Where the noise in the image has a Gaussian Distrubution, Meaning that If we were to acquire the image of the scene repeatedly,we would find that the intensity values of each pixel fluctuate so that a distribution of pixel values centred on the actual intensity value for that pixel is obtained.
With the help of both the box filter and the median filter, this can be aided to an extent. We'll describe the outecome of applying both filters in the next section.

Problem 3.3.1:

Take a blurry and noisy image, and try to improve it's appearance and legibility.

lets consider the mask below

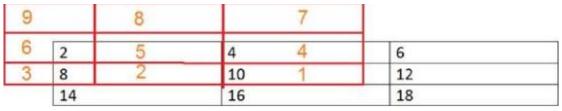
1	2	3
4	5	6
7	8	9

the steps to perform convolution involve:

step1: Flip the mask (horizontally and vertically) only once

9	8	7
6	5	4
3	2	1

step2: Slide the mask onto the image, multiply the corresponding elements and then add them



step3: slide the mask and repeat step 2, if all of the image units have been calculated, then end the procedure.

note: in areas where the mask units fall out of the image, we consider the image's unit values to be zero (we apply a zero padding to the image before the first step)

To answer the original question, when we apply the box filter by doing the above procedure many times, it will slowly reduce the pixel values based on their neighbours' values, in turn also blurring the entire image.

Problem 3.2.1:

add salt and-pepper noise to an image with a specified noise density. Then, perform median filtering with a square shape and specified window size

edges). The filters we used above, known as the Soble filters can be used as a way to approximate the 2-D gradients, here in the horizontal direction (note that by getting the transpose of them, will result in the vertical direction). Also, if we decide to increse the value of the coefficients, the impact of the Soble filter will also increase.

Problem 3.4.2:

$$a)\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}, \quad b)\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$$

In which directions do these filters detect edges? Compare The quality of the resulted image with the previous problems'

Solution:

If we applty filter "a", it will give us the image's edges in left diagonal direction, in contrast, filter "b" gives us the edges in right diagonal direction.

The quality of the image's found in this image will be slightly higher than the previous images,due to the coefficients being of a higher value, so in turn we can detect the edges more easily.

It's also worth noting that usually diagonal edges capture more than just using horizontal or just using vertical directions, due to it showing an impact to both of the main directions.

Problem 3.4.2:

$$(1-\alpha)I+\alpha I' = I+\alpha(I'-I),$$

The formula above is used as an unsharp masking filter. What impact does changing the value of "a" have on the image? How should we find the optimal value for "a"?

Solution:

Digital unsharp masking is a flexible and powerful way to increase the sharpness of an image. we need to choose a Threshold value "a" to establish how different in value

Solution:

We've achieved that one way to reduce noise in an image is to apply the Box Filter to an image, however this will result in a bit of a blur on the image. However, there is a way we are able to slightly aid this and make our image appear better, and that is by making the edges look better with the help of the below filter that sharpens the edges:

$$\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Problem 3.4.1:

$$a)\frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix}, \quad b)\frac{1}{6}\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix}, \quad c)\frac{1}{8}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

the above filters are used for edge detection. How efficient are they? Are they suitable for computing 2D-Gradients?

Solution:

If we apply the convolution steps using the above filters, the result will somewhat give us the edges. With the filter "a" we compure the difference between the right pixel and left pixel of each pixel, in turn this helps specify the edges of the image. In filter "b" rach of the pixels are calculated by taking into account the horizontal and diagonal neighbours each with the same weight. By doing this, we slightly get more details on the edges. Filter "c" is similar to b, except that the weights of the horizontal neighbours have a higher weight than the diagonal neighbours, in turn making the horizontal edges slightly more sharper.

In image processing when we take the 2-D gradient in the Y direction, it will give us the horizontal edges (and X will give vertical

If we apply the filter fifty times:

And finally, if we apply the filter one hundred times :

As it is obvious, the more times the box filter is applied to the image, the more blurry the image becomes.
Also we can notice that the more times we do this, a black edge is formed around the image. This is due to the zero padding I implemented, which is done before the convolution procedure happens.

Problem 3.2.1:
If we apply Salt and Pepper Noise to the Lena image with different noise densities, we get the following images:

an area of pixels must be from an adjacent area for it to be affected by the filter.
The optimal threshold value is different with each image, however with a certain image, In order to find the optimal value for “a” we can start with a Threshold value of 0 then gradually increase it, also taking note of how it affects the image. once the image reaches an “over sharped” state, we stop increasing the values.
Note that if we have a high contrast image, we need a higher threshold (a>0.5), in this case the high contrast areas will be sharpened and areas of lesser contrast will be sharpened much less.

3- Describing the Results

Problem 3.1.3:
The original Lena image is as follows.

If we do convolution with the box filter ten times we get the following results:

As we can see throughout the images, the higher the noise density is, by doing the median filter, we also get an overall higher contrast image. This is due to the noise being of higher density, and taking the median in the matrix blocks will result in a slightly higher pixel values.
It can also be seen that as we increase the Window size, the images slightly become more blurred. This is due to the fact that we take the median in a larger range, thus making the values the pixels may obtain have a wider variety.
Now let's look at the MSE's of the obtained images with the original Lena Image:

Report MSE				
	3×3	5×5	7×7	9×9
$\rho = 0.05$	260	443	612	774
$\rho = 0.1$	264	447	613	774
$\rho = 02$	272	450	615	778

We see as the window size increases, the difference between the obtained MSE's by using different noise densities slowly decreases. This is still due to the fact that the increasing the window size at the rate that we do, has a larger impact than increasing the noise density with the given rates.

Problem 3.2.2:
If we apply Guassian Noise to the Lena image with different variances, the result will be:

Now we'll apply the Median filter with different filter sizes to see the results:

~ Window Size = 3*3 ~

~ Window Size = 5*5 ~

~ Window Size = 7*7 ~

~ Window Size = 9*9 ~

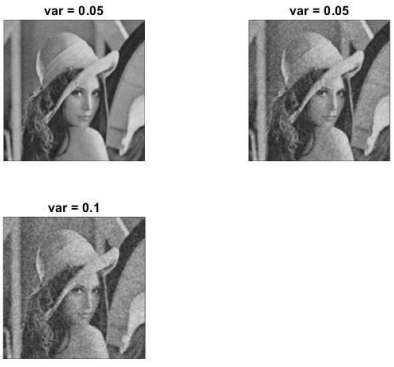
pixels involved in each step of applying the Median filter.
We can also see that if the noise variance isn't too high, by taking a window size larger than seven, we see the noise reduction works quiet well.

Now, lets try removing the Guassian noise by applying the Box Filter with different window sizes:

~ Window Size = 3*3 ~

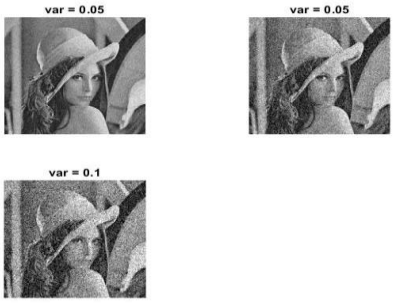
~ Window Size = 5*5 ~

~ Window Size = 7*7 ~

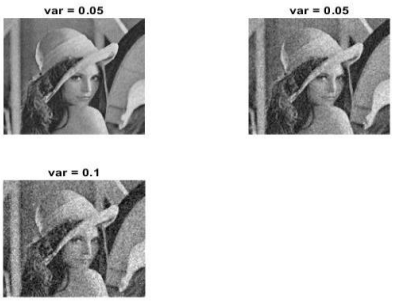


First, lets try removing the Guassian noise by applying the Median Filter with different window sizes:

~ Window Size = 3*3 ~

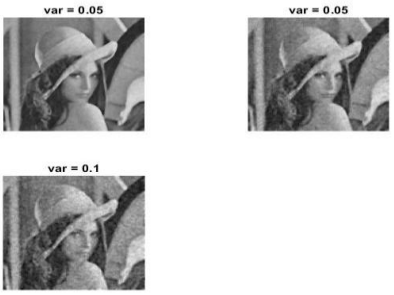


~ Window Size = 5*5 ~



~ Window Size = 7*7 ~

~ Window Size = 9*9 ~



As we see, the larger the window size gets, the better result we see using the median filter, due to there being a more variety of

filter, by increasing the window size, we're mostly getting better results until we reach a threshold for the window size. Here It seems that the threshold is 7.

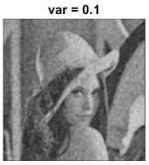
Problem 3.3.1:

The above image is a rather noisy and blury picture I took of my keyboard, alongside it there's a chair which may be hard to notice. The upper half of the image seems to have more noise than the lower half of the image

In order to reduce the noise, I did convolution using a box filter with the window size of 7. The result is as follows:

The noise is somewhat reduced, however the image has become more blury.

~ Window Size = 9*9 ~



As we see, by applying the box filter, the larger the window size, the better result we get, due to considering a larger scale of pixels that is averaged over. If the variance is increased, the results seem to look a bit better than the Median Filter.

If we compare the results obtained from the Median filter to the Box filter, we notice that the resulting image filtered with the median filter has a much lower contrast compared to the resulted image filtered by the Box filter, so in conclusion, applying the Box filter seems to give us a smoother and better quality image.

Now let's take a look at the MSE's of each of the images found above, with the original Lena image

Report MSE

	Median				Box Filter			
	3×3	5×5	7×7	9×9	3×3	5×5	7×7	9×9
$\sigma = 0.01$	299	434	576	711	174	182	222	269
$\sigma = 0.05$	524	528	617	717	430	286	286	315
$\sigma = 0.1$	815	664	693	768	693	419	381	394

As we can see, our guess that the box filter is removing the noise better than the Median filter seems to be true.

We can also see that by doing the median filter when our noise variance isn't too high, by increasing the window size we get farther away from the original image's quality. However, if we consider the Box

If we apply

$$b)\frac{1}{6}\begin{bmatrix}1 & 0 & -1\\ 1 & 0 & -1\\ 1 & 0 & -1\end{bmatrix},$$

the result will be:

If we compare this to the image obtained previously, we see that this obtained image shows slightly less details. This is due to giving the same weight with the diagonal neighbours, as the horizontal neighbours.

If we apply

$$c)\frac{1}{8}\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

The result will be:

In order to try to reduce the blur from the picture, I performed convolution with a sharpening filter.

This is the resulting Image I obtained, which isn't perfect, but looks slightly more appealing than the original image.

Problem 3.4.1:

These filters also known as the Sobel filters are widely used for edge detection.

If we apply $a)\frac{1}{2}[1 \quad 0 \quad -1].$

To the Lena image, the result will be:

It's also worth noting that if we increase the value of the coefficient from (1/2), to for example 2, we'll be able to see the horizontal edges more clear, however, there will be more noise:

$b) \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

we get the right diagonal edges, which the result will be as follows:

As it can be seen, applying any one of these filters gives us much clearer edges compared to all the ones we found from the previous problem, this is due to the fact that when applying diagonal edges, these edges take impact from both the diagonal and vertical directions, resulting in a stronger separation between the edges and non-edges.

Problem 3.5.1:

We will compare the impact of the unsharp masking filter of the filter sizes (3,5,7,9) alongside each other, and also by changing the threshold value of a.

~ threshold value a = 0.1~

Which it can be clearly seen that this filter shows the edges better than both the previous images. This is because for each pixel it takes both diagonal and horizontal neighbours into account, however, the horizontal neighbours have a weight twice as much as the diagonal neighbours.

Problem 3.4.2:

We figured out that applying the given filters gives us the diagonal edges of an image.

If we apply

$a) \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$

Which gives us the left diagonal edges, the result will be as follows:

Also if we apply the following

performing convolution on an image.

The Median function is as follows:

```
function out = do_median(img,WINDOW_SIZE)

[X, Y] = size(img);
matrix_rows = WINDOW_SIZE;
matrix_cols = WINDOW_SIZE;

out = uint8(zeros(X,Y));

for ii = 1 : X
    for jj = 1 : Y
        % Get the block
        row_begin = ii;
        row_end = min(X, ii + matrix_rows); %we put the min here so we don't go out of b
        col_begin = jj;
        col_end = min(Y, jj + matrix_cols); %we put the min here so we don't go out of b
        blk = img(row_begin : row_end, col_begin : col_end, :);
        O = median(blk(:)); %convert blk to a column, then find median of that
        % the output of this is in O, concatenate this with the other grids
        out(ii, jj) = O;
    end
end
end
```

The convolution function is as follows:

```
function out = do_conv(img, kernel)
    img = double(img);
    [ker_y, ker_x] = size(kernel);
    im_pad = padarray(img, [ker_x ker_y]);  % STEP 0 -> zero Pad original image
    [y, x] = size(im_pad);
    out = zeros(x,y);
    kr = rot90(kernel);          % STEP 1 -> Rotate kernel 180 deg for convolution
    kr = rot90(kr);
    %STEP 2 -> slide the kernel alongside the IMAGE
    for i=(1+ker_y):(y-ker_y)
        for j=(1+ker_x):(x-ker_x)     % index through each image pixel
            neigh = im_pad(i-floor(ker_y/2):i+floor(ker_y/2), j-floor(ker_x/2):j+floor(ker_x/2));   % Create local neighborhood of image
            accum = 0;
            for u=1:ker_y      % index through each kernel row
                for v=1:ker_x  % index through each kernel element
                    if(i>ker_y && i<y-ker_y && j>ker_x && j<y-ker_x)
                    temp = neigh(u,v)*kr(u,v);
                    accum = accum + temp;
                    end
                end
            end
            out(i,j) = accum;   %Set value of pixel in new image with convolution operation results%
            %STEP 3 -> REPEAT STEP 2 UNTIL WE'RE DONE OVER ALL THE PIXEL VALUES

        end
    end
    out=out(1+ker_y:y-ker_y,1+ker_x:x-ker_x);       % Remove image padding
    out = uint8(out);
end
```

Problem 3.1.3:

The code below is used to apply the box filter with a windows size of 3*3 many times:

```
figure(1);
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\Lena.bmp');
WINDOW_SIZE = 3;

out = rgb2gray(img);
TIMES_TO_FILTER = 10;
box_filter = ones(WINDOW_SIZE) * (1/((WINDOW_SIZE)^2));
for i = 1:TIMES_TO_FILTER
    out = do_conv(out,box_filter);
end
imshow(out);
```

Problem 3.2.1:

In the code below, I apply the salt and pepper noise with different densities, and then try to reduce the noise by

~ threshold value a = 0.5~

~ threshold value a = 1~

The original Lena image is more high contrast, and we see the higher the threshold value, the sharper the high contrast parts of the image become.

So as a result, the closer the threshold value is to 1, we can see that we get a sharper looking image.

Also we see as the filter size increases, the sharpness of the image also increases, as long as we pick a threshold value that is high enough.

4- Appendix(code)

Throughout the codes that I've written, I use two functions that I've wrote. One for performing the median filtering on an image, and one for

Problem 3.4.1:

The code used to apply the sobel filters:

```
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\Lena.bmp');
out = rgb2gray(img);
a = [1,0,-1] * (2);
b = [1,0,-1;1,0,-1;1,0,-1]* (1/6);
c = [1,0,-1;2,0,-2;1,0,-1]* (1/8);

out = do_conv(out,c);

imshow(out);
```

Problem 3.4.2:

The code used to apply Robert's filters:

```
clear;
figure(1);
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\Lena.bmp')
out = rgb2gray(img);
a = [1,0;0,-1];
b = [0,1;-1,0];

out = do_conv(out,b);

imshow(out);
```

Problem 3.5.1:

The code that I used to apply the unmasking filter, with different filter sizes:

```
img = rgb2gray(img);
img = double(img);
a = 1; % a is between [0,1]
sigm1 = 0.5; %filter size is 2*ceil(2*sigma)+1
            %sigma is either 0.5(FS ->3), or 1 (FS -> 5),or 1.5 (FS -> 7), or 2 (FS -> 9)
sigm2 = 1;
sigm3 = 1.5;
sigm4 = 2;
%converting to double to perform arithmatic operations
img_guass1 = double(imgaussfilt(uint8(img), sigm1));
out1 = uint8(img + a*(img_guass1 - img));
subplot(2,2,1);imshow(out1);title("FS ->3");

img_guass2 = double(imgaussfilt(uint8(img), sigm2));
out2 = uint8(img + a*(img_guass2 - img));
subplot(2,2,2);imshow(out2);title("FS ->5");

img_guass3 = double(imgaussfilt(uint8(img), sigm3));
out3 = uint8(img + a*(img_guass3 - img));
subplot(2,2,3);imshow(out3);title("FS ->7");

img_guass4 = double(imgaussfilt(uint8(img), sigm4));
out4 = uint8(img + a*(img_guass4 - img));
subplot(2,2,4);imshow(out4);title("FS ->9");
```

performing the Median filter on them:

```
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\Lena.bmp');
img = rgb2gray(img);
out1 = imnoise(img,'salt & pepper',0.05);
out2 = imnoise(img,'salt & pepper',0.1);
out3 = imnoise(img,'salt & pepper',0.2);
[X, Y] = size(img);
WINDOW_SIZE = 9;
out1 = do_median(out1,WINDOW_SIZE);
out2 = do_median(out2,WINDOW_SIZE);
out3 = do_median(out3,WINDOW_SIZE);

subplot(2,2,1);
imshow(out1);
title ('p = 0.05')
immse(img,out1)

subplot(2,2,2)
imshow(out2);
title ('p = 0.1')
immse(img,out2)

subplot(2,2,3)
imshow(out3);
title ('p = 0.2')
immse(img,out3)
```

Problem 3.2.2:

In the code below, I apply the Guassian Noise with different variances, and then try to reduce the noise by applying the Median or the Box filter:

```
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\Lena.bmp');
img = rgb2gray(img);
WINDOW_SIZE = 7;
box_filter = ones(WINDOW_SIZE)*(1/((WINDOW_SIZE)^2));

out1 = imnoise(img, 'gaussian', 0, 0.01);%add noise
out2 = imnoise(img, 'gaussian', 0, 0.05);
out3 = imnoise(img, 'gaussian', 0, 0.1);

%out1 = do_median(out1,WINDOW_SIZE);
out1 = do_conv(out1,box_filter);
subplot(2,2,1);
imshow(out1);
title ('var = 0.05')
immse(img,out1)

%out2 = do_median(out2,WINDOW_SIZE);
out2 = do_conv(out2,box_filter);
subplot(2,2,2);
imshow(out2);
title ('var = 0.05')
immse(img,out2)

%out3 = do_median(out3,WINDOW_SIZE);
out3 = do_conv(out3,box_filter);
subplot(2,2,3);
imshow(out3);
title ('var = 0.1')
immse(img,out3)
```

Problem 3.3.1:

This is the code I used to try to enhance the picture I took:

```
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\3\piano.jpg');
WINDOW_SIZE = 7;
out = rgb2gray(img);
box_filter = ones(WINDOW_SIZE) * (1/((WINDOW_SIZE)^2));
sharpen_filter = [0,-1,0;-1,5,-1;0,-1,0];
sobel_filter =[1,0,-1] * (2);


TIMES_TO_FILTER = 3;

for i = 1:TIMES_TO_FILTER
    out = do_conv(out,box_filter);
end
out = do_conv(out,sharpen_filter);
imshow(out);
```