Signal and image processing: Assignment 4

Mariuta Nicolae 25-09-2014

Exercise 1.

For the purpose to show that $G(x,y,\sigma) * G(x,y,\tau) = G(x,y,\sqrt{(\sigma^2+\tau^2)})$ I have written the following code:

```
%1
%create I(x,y,s)
s = 10;
t = 6;
Is = fspecial('gaussian', [100 100], s);
It = fspecial('gaussian', [100 100], t);
Ist = fspecial('gaussian', [100 100], sqrt(s^2+t^2));

convfft = scale(fft2(Is),t,0,0);
conv = ifft2(convfft);
diff = Ist-conv;

subplot(1,5,1),imagesc(Is),title('Is'),colormap(gray);
subplot(1,5,2),imagesc(It),title('It'),colormap(gray);
subplot(1,5,3),imagesc(Ist),title('Ist'),colormap(gray);
subplot(1,5,4),imagesc(conv),title('conv'),colormap(gray);
subplot(1,5,4),imagesc(diff),title('diff'),colormap(gray);

disp(sum(diff));
```

Fig. 1 Code for show the result of convolution between 2 Gaussians having different deviations

I used the *fspecial* function of Matlab to generate 2 Gaussian images having the standard deviations 10 and 6 and also a Gaussian with standard deviation of $\sqrt{(\sigma^2+\tau^2}$ which should be the result of convolving the 2 images.

At next step I calculated the Fourier transform of the Gaussian image with deviation 10 and applied the function *scale* as provided on the course page in Absalon. This function calculates the convolution on the 2 images by using the Fourier transform. Parameter *t* is the second deviation, 6 and I have chosen not to derive the result (that is why *dx* and *dy* are 0).

To prove that I obtained the same image as *Ist*, the one using the equation *(2)* I calculated a difference image(*diff*) which should have values very close to 0.

I have plotted all the results and I also calculated sum of all pixels in the *diff* using the *sum* function in Matlab which calculates the sum of all pixels in a matrix.

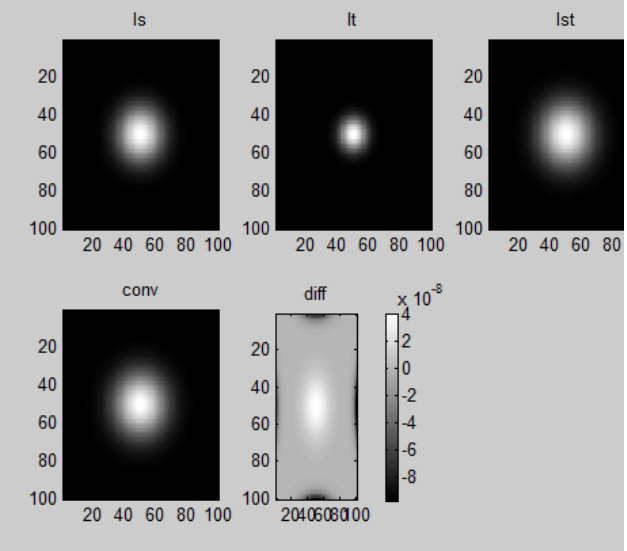

Fig. 2 The result for running the code in Fig. 1

- By running the code in Fig. 1 I obtained that by applying the convolution between the 2 Gaussians I obtained the same result as using the equation (*2*).
- Images *Ist* and *conv* look almost the same and as expecting, looking at the equation, the radius of the Gaussian is larger than the Gaussians in first 2 images.
- The difference image *diff* proves that the differences between *Ist* and *conv* are very small, with values that are close to 10^{-8} and the sum of all elements in *diff* is very small (aprox 10^{-5}).
- The difference is only because the calculation using both method use operations that generate double numbers with a lot of decimals (sqrt, euler number, divisions) and approximations but however, the differences are very small
- Equation *(2)* proves a very fast way to calculate convolution between 2 Gaussians which may prove to be very useful in edge detections and other areas where Gaussians kernels are used.

Exercise 2.a

Considering the scale normalized image of the Laplacian,

$$H(x,y,\tau) = Ixx(x,y,\tau) + Iyy(x,y,\tau)$$

And doing the replacements using the equation (*5*)

$$H(x,y,\tau) = \tau^{1(1+1)}\frac{\partial^2 I(x,y,\tau)}{\partial x \partial x} + \tau^{1(1+1)}\frac{\partial^2 I(x,y,\tau)}{\partial y \partial y}$$
$$= \tau^2\frac{\partial^2 I(x,y) * G(x,y,\tau)}{\partial x \partial x} + \tau^2\frac{\partial^2 I(x,y) * G(x,y,\tau)}{\partial y \partial y}$$

I have done the previous calculations using the property from (*4*) in the assignment and if we consider that $I(x,y)$ is another Gaussian with standard deviation σ and considering the property that was proved in Exercise 1, I get that:

$$H(x,y,\tau) = \tau^2 \frac{\partial^2 G(x,y,\sqrt{\tau^2+\sigma^2})}{\partial x \partial x} + \tau^2 \frac{\partial^2 G(x,y,\sqrt{\tau^2+\sigma^2})}{\partial y \partial y}$$

$$= \tau^2 \frac{\partial^2 \frac{1}{2\pi(\tau^2+\sigma^2)} e^{-\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\partial x \partial x} + \tau^2 \frac{\partial^2 \frac{1}{2\pi(\tau^2+\sigma^2)} e^{-\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\partial y \partial y}$$

Because calculating those derivatives is too much time consuming and with high chance to do mistakes, I have used Maple to finish the calculations. I obtained this:

$$H(x,y,\tau) = \tau^2 \left(-\frac{e^{--\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\pi(\tau^2+\sigma^2)(2\tau^2+2\sigma^2)} + \frac{2x^2 e^{--\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\pi(\tau^2+\sigma^2)(2\tau^2+2\sigma^2)} \right)$$

$$+ \tau^2 \left(-\frac{e^{--\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\pi(\tau^2+\sigma^2)(2\tau^2+2\sigma^2)} + \frac{2y^2 e^{--\frac{x^2+y^2}{2(\tau^2+\sigma^2)}}}{\pi(\tau^2+\sigma^2)(2\tau^2+2\sigma^2)} \right)$$

Exercise 2.b

By using Maple, I have found that maximum and minimums are found for values of $\tau = 0, \sigma, -\sigma$. Unfortunately I could not save the script written in Maple and do not longer have the calculations.

Exercise 2.c

To prove results obtained at Exercise 2.b I have written a program in Matlab.

```
s = 10;

Is = fspecial('gaussian', [100 100], s);
H00 = zeros([300]);

for t = 1 : 300
    tau = t/10;
    Ixx = ifft2(scale(fft2(Is),tau,2,0)).*(tau^2);
    Iyy = ifft2(scale(fft2(Is),tau,0,2)).*(tau^2);
    H = Ixx + Iyy;
    H00(t) = H(size(H,1)/2,size(H,2)/2);
end

plot([0.1:0.1:30],H00);
```

Fig. 3 Code that plots the result obtained in 2.b

In my code, variable *s* is standard deviation for the *I(x,y)* (*Is)* Gaussian with the deviation σ.

In the *for* loop I take 300 values for standard deviation τ with values between 1 and 30 with step size of *0.1*. For each value of τ I create a pair of Ixx and Iyy just as in the equation (*6*) by using the scale function that was created for the previous assignment and sum them together to obtain $H(x, y, \tau)$. In order to get value for $H(0, 0, \tau)$in each image that was created, I consider the middle of $H(x, y, \tau)$.

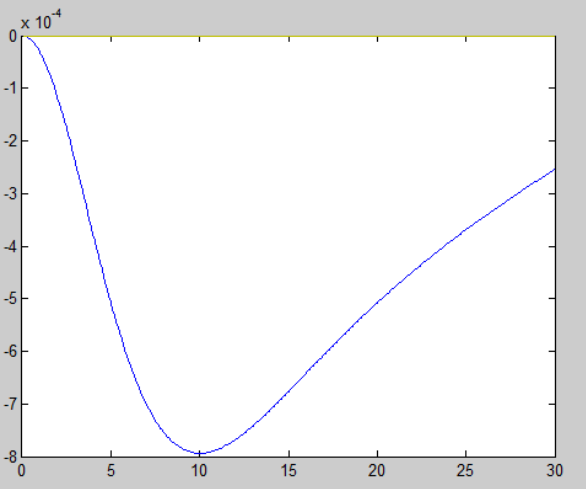

Fig. 4 Plotting the center of each scale normalized image of Laplacian

- As expected, in the resulting plot the maximum is for values $\tau = 0$ and $\sigma = 0$ and the minimum point is for value $\sigma = 10$

- I could also consider negative standard deviations for τ but that was going to be a symmetrical plot and in practice the negative values for standard deviation of Gaussian is not used
- The fact that results at Exercise 2.b have been confirmed in Matlab shows we can use these results to local maximums and minimums in images by convolving them with different standard deviations which can help with detecting edges

Exercise 2.d

Basically, I have started just as for the 2.c but instead of Gaussian image, I have loaded the image '*sunflower.tiff*' and I have tried to find maximum and minima in the image.

For each value for the standard deviation *t* I have stored the result in 3d matrixes, *Ht* and *Htabs* for the purpose to find the maximum values in *Htabs* and then search in *Ht* to see if those values were minimums or maximums.

```
%%
Is = imread('sunflower.tiff');

Ht = zeros(size(Is,1),size(Is,2),300);
Htabs = zeros(size(Is,1),size(Is,2),300);

for t = 1 : 300
    tau = t/10;
    Ixx = ifft2(scale(fft2(Is),tau,2,0)).*(tau^2);
    Iyy = ifft2(scale(fft2(Is),tau,0,2)).*(tau^2);
    H = Ixx + Iyy;
    Ht(:,:,t) = H;
    Habs = abs(H);
    Htabs(:,:,t) = Habs;
end

[sorted,sortingIndices] = sort(Habs(),'descend');
maxValues = sorted(1:20);
maxValueIndices = sortingIndices(1:20);

imshow(Is);
hold on;

for i = 1 : 20
    [x,y,t]=sub2ind(size(Habs), find(Habs==maxValues(i)));
    scatter()
end

hold off;
```

Fig. 5 Detecting maximum and minimums in 'sunflower.tiff'

Unfortunately, after sorting the values in the 3d array I did not manage to get the indexes of the first elements for the purpose to draw the circles on the image. If I could manage to get the coordinates, *x* and *y* were going to be coordinates in the image while *t* was going to be radius of the circles corresponding to the deviation of the Gaussian on which it was found.

Exercise 3.a

I have mostly used the calculation shown by Jon on Wednesday at the course.

$$J(x,y,\tau) = J(x,y) * G(x,y,\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} J(x',y')G(x-x',y-y',\tau)\,dx'dy'$$

$$J(x) = \frac{\partial}{\partial x}\big(J(x,y) * G(x,y,\tau)\big) = G(x,0,\tau) * G(x,y,\tau)$$

$J(y) = \frac{\partial}{\partial y}\big(J(x,y) * G(x,y,\tau)\big) = G(x,0,\tau) * G(x,y,\tau) = 0$ because the integral $J(x,y,\tau)$ contains only x.

$$\big||\nabla J(x,y,\tau)|\big| = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x,0,\tau)G(x-x',y-y',\tau)\,dx'dy'$$

$$G(x,y,\tau) = \frac{1}{2\pi\tau^2}\exp\left(\frac{-x^2-y^2}{2\tau^2}\right) = = \frac{1}{2\pi\tau^2}\exp\left(\frac{-x^2}{2\tau^2}\right)\exp\left(\frac{-y^2}{2\tau^2}\right) = G(x,\tau)G(y,\tau)$$

$$J(x,y,\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(x',0,\tau)G(x-x',\tau)\,dx'G(y-y',\tau)dy'$$

But: $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} kG(y-y',\tau) = k$ so $J(x,y,\tau) = \frac{1}{2\pi\tau^2}G(x,\sqrt{\sigma^2+\tau^2})$

$\big||\nabla J|\big|^2 = J_x^2 + J_y^2 = \tau\frac{1}{2\pi\sigma^2}\frac{1}{2\pi(\sigma^2+\tau^2)}e^{-\frac{x^2}{2(\sigma^2+\tau^2)}}$

Exercise 3.b

If *x=0* and *y=0* the analytical expression obtained at 3.a becomes:

$\big||\nabla J|\big|^2 = \tau\frac{1}{2\pi\sigma^2}\frac{1}{2\pi(\sigma^2+\tau^2)}$ So the maximum is obtained for $\tau = 0$.

Exercise 4.

```
I = imread('lena.tiff');
subplot(2,4,1);imshow(I);colormap(gray);title('original');
%convolve with gaussian kernel
H = fspecial('gaussian', size(I), 3);
Ic = mat2gray(abs(ifft2(scale(fft2(I),3,0,0))));
subplot(2,4,2);imagesc(Ic);colormap(gray);title('original**kernel');
%add noise
Icn =imnoise(Ic,'gaussian',0,0.002);
subplot(2,4,3);imagesc(Icn);colormap(gray);title('original**kernel+noise');
```

Fig. 6 Convolve lena.tiff with a Gaussian and then add noise to image

I used same *scale* function to convolve the image easier with the Gaussian and obtain a blurred image; *H* saves the Gaussian that was convolved over the image, in Fourier domain. In the next stage, I have added some noise (Gaussian) to the image using Matlab *imnoise* function. I have displayed all the image in each stage.

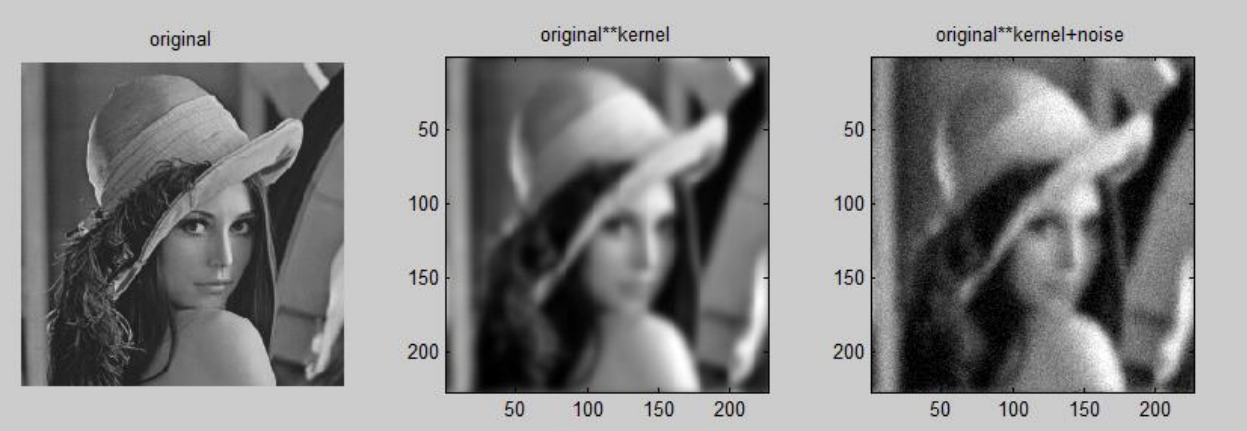

Fig. 7 Result for running the code in Fig 3

The code is basically simulating what happens when a camera is used for taking a photo. The kernel is the known error that camera makes when taking the photo but it is something known and can be easily removed. The noise is something random that cannot be predicted and that is added to the resulting image, basically the error that happens because of the environment (temperature, light influence).

Exercise 5.

I have tried to remove the noisy images obtained at Exercise 4 by writing the following code:

```
%5
Icfft = fftshift(fft2(Ic));
Imfft = Icfft./H;
Im = abs(ifft2(Imfft));

subplot(2,4,4);imagesc(Im);colormap(gray);title('inverse filter');
```

Fig. 8 Removing the noisy image obtained at Exercise 4 using inverse filtering

I have applied the formula for inverse filtering from the course book. I have taken the image without the noise because according to the book, it should not work for noisy image.

I turned the image into Fourier domain and divided it by the Gaussian kernel that was applied in Fourier domain but unfortunately I get a completely black image and I did not understand why it happens.