ENG204 - Signals and Linear Systems - Assignment 1.3

Baley Eccles - 652137

[2024-09-12 Thu 14:12]

Contents

1 ENG204 - Signals and Linear Systems - Assignment 1.3

1.1 TODO a

1.2 b

```
clear
clc
close
pkg load symbolic
sigma=10; % Filter size

G = @(x, y) (1/(2*pi*sigma^2)) * exp(-1 * (x.^2 + y.^2) / (2 * sigma^2));
size=6*sigma;
GFilter=zeros(size);

for xCoord = -size/2:size/2-1
  for yCoord = -size/2:size/2-1
    Gval=G(xCoord,yCoord);
    GFilter(size/2+xCoord+1,size/2+yCoord+1)=double(Gval);
  end
end

% Normalise the matrix
GFilter=(GFilter - min(GFilter(:))) / (max(GFilter(:)) - min(GFilter(:)));
```

```
noise=imread("/home/Baley/UTAS/ENG204 - Signals And Linear Systems/Assignment
↪  1.3/Pic/image_5_noise.jpg");
```

```
output=conv2(noise,GFilter,'same');
imshow(output,[]);
```

1.3 c

```
clear
clc
close
pkg load symbolic

syms x y p phi sigma
G =(1/(2*pi*sigma)) * exp(-1 * (x^2 + y^2) / (2 * sigma^2));
% Sub in the cylindrical coordinates
xCyl=p*cos(phi);
yCyl=p*sin(phi);
G=subs(G,x,xCyl);
G=subs(G,y,yCyl);
latex(xCyl)
latex(yCyl)
latex(simplify(G))
```

To do this we can convert the function to cylindrical coordinates. using:

$$x = \rho \cos(\phi)$$

$$y = \rho \sin(\phi)$$

Which will give:

$$\frac{e^{-\frac{\rho^2}{2\sigma^2}}}{2\pi\sigma}$$

As we can see this does not depend on ϕ, which is the rotational aspect, so it is rotationaly symmetric.

1.4 d

1.4.1 TODO a

$$\begin{aligned} G(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ G(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \\ \Rightarrow G_x(x) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \\ \text{and } G_y(y) &= \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \end{aligned}$$

This shows that the Guassian kernal can be sperated into two individual components that can be acted seperately in the x and y direction.

1.4.2 b

We are taking the convolution of G and I resulting in O:

$$
\begin{aligned}
O &= I * G \\
O &= I * (G_x \cdot G_y) \\
I' &= I * G_x \\
I &= I' * G_y
\end{aligned}
$$

This shows that the Guassian kernal can be convoluted with the image in the x direction to get some intermediate image, which then can be convoluted in the y direction to get the final image.

1.4.3 TODO c

The output with $\sigma = 10$ is:

- The time to calculate the convolution of the single matrix is 0.240254 s
- The time to calculate the convolution of the two matrices is 0.025748 s

As we can see the convolution of the two matricies is about ten times as fast. And we can also see that this creates the exact same result.

Increasing the σ we see that the difference between the two times increases. For $\sigma = 50$:

- The time to calculate the convolution of the single matrix is 5.167703 s
- The time to calculate the convolution of the two matrices is 0.058540 s

Here we get a ≈ 100 times increase in speed. These resutls will vary based upon the hardware that it is being ran on. How ever we would still expect to see the increase in speed from one convolution to two.

1.5 e

$$\begin{aligned}
\nabla^2 f &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\
\text{subsitute in } \frac{\partial^2 f}{\partial x^2} &\approx f(x+1,y) - 2f(x,y) + f(x-1,y) \\
\text{and } \frac{\partial^2 f}{\partial y^2} &\approx f(x,y+1) - 2f(x,y) + f(x,y-1) \\
\text{gives } \nabla^2 f &\approx [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)
\end{aligned}$$

Reading the coefficents for the matrix:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

1.6 f

```
clear
clc
close
```

```
LFilter=[0, 1, 0;
         1,-4, 1;
         0, 1, 0];

% Normalise the matrix
LFilter=(LFilter - min(LFilter(:))) / (max(LFilter(:)) - min(LFilter(:)));
```

```
noise=imread("/home/Baley/UTAS/ENG204 - Signals And Linear Systems/Assignment
↪  1.3/Pic/image_1_noise.jpg");
```

```
output=conv2(noise,LFilter,'same');
imshow(output,[]);
```

```
Threshold = 10;
EdgeDetect = output > Threshold;
imshow(EdgeDetect,[]);
```

Noise in the image makes the derivative of the image contain a lot of larger values. The noise makes the difference between each pixel a larger result than without the noise. This resulst in the edge detect image having alot of large values, requiring the threshold to be larger and reducing the amount of true edges being detected.

1.7 g

$$LoG(x,y) = \nabla^2 G(x,y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

$$\begin{aligned}
\frac{\partial G}{\partial x} &= -\frac{x e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^3} \\
\Rightarrow \frac{\partial^2 G}{\partial x^2} &= -\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^3} + \frac{x^2 e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^5}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial G}{\partial y} &= -\frac{y e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^4} \\
\Rightarrow \frac{\partial^2 G}{\partial y^2} &= -\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^4} + \frac{y^2 e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^6} \\
\Rightarrow LoG(x,y) &= -\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\pi\sigma^4} + \frac{x^2 e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^6} + \frac{y^2 e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^6} \\
\Rightarrow LoG(x,y) &= -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}
\end{aligned}$$

1.8 h

Focusing on $1 - \frac{x^2+y^2}{2\sigma^2}$ in the kernal. We can see that it contains $x^2 + y^2$, which is not separable, so the entire kernal is not separable.
The second derivatives of the Gaussian kernal can be expressed as a product of an individual

varible and the Gaussian kernal. That is:

$$\frac{\partial^2 G}{\partial x^2} = -\frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^3} + \frac{x^2 e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^5}$$
$$\frac{\partial^2 G}{\partial x^2} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(\frac{x^2}{\sigma^3} - \frac{1}{\sigma}\right)$$
$$\frac{\partial^2 G}{\partial x^2} = G(x,y)\left(\frac{x^2}{\sigma^3} - \frac{1}{\sigma}\right)$$

$$\text{Similarly for } \frac{\partial^2 G}{\partial y^2}$$
$$\frac{\partial^2 G}{\partial y^2} = \frac{1}{2\pi\sigma^2} e^{-\frac{y^2+x^2}{2\sigma^2}} \left(\frac{y^2}{\sigma^3} - \frac{1}{\sigma}\right)$$
$$\frac{\partial^2 G}{\partial y^2} = G(x,y)\left(\frac{y^2}{\sigma^3} - \frac{1}{\sigma}\right)$$

We know that the Gaussian kernal is separable, and that is being multiplied by a function of the respective varible. So, the derivatives of the Guassian kernal are separable.
To speed up the computation of the LoG kernal we can use:

$$\nabla^2 G \approx \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

Where we can calculate the first and second derivatives from their separable forms.

1.9 i

```
clear
clc
close
pkg load symbolic
sigma=100; % Filter size

LoG = @(x, y) (-1/(pi*sigma))*(1-
↪  ((x.^2+y.^2)/(2*sigma^2)))*e^(-1*(x^2+y^2)/(2*sigma^2));
size=6.*sigma;
LoGFilter=zeros(size);

for xCoord = -size/2:size/2-1
  for yCoord = -size/2:size/2-1
    LoGval=LoG(xCoord,yCoord);
    LoGFilter(size/2+xCoord+1,size/2+yCoord+1)=double(LoGval);
  end
end

% Normalise the matrix
LoGFilter=(LoGFilter - min(LoGFilter(:))) / (max(LoGFilter(:)) - min(LoGFilter(:)));
```

```
noise=imread("/home/Baley/UTAS/ENG204 - Signals And Linear Systems/Assignment
↪  1.3/Pic/image_1_noise.jpg");
```

```
output=conv2(noise,LoGFilter,'same');
imshow(output,[]);
```

```
Threshold = 50;
EdgeDetect = output > Threshold;
imshow(EdgeDetect,[]);
```

1.10 TODO j

To do this we will get an edge detect of the image and then add it back onto the original image. How ever, as mentioned before the noise in the image will make it look bad, so first we are going to apply the Gaussian filter and then the edge detect.

```
clear
clc
close
pkg load symbolic

sigma=50; % Filter size

G = @(x, y) (1/(2*pi*sigma^2)) * exp(-1 * (x.^2 + y.^2) / (2 * sigma^2));
size=6*sigma;
GFilter=zeros(size);

for xCoord = -size/2:size/2-1
  for yCoord = -size/2:size/2-1
    Gval=G(xCoord,yCoord);
    GFilter(size/2+xCoord+1,size/2+yCoord+1)=double(Gval);
  end
end

% Normalise the matrix
GFilter=(GFilter - min(GFilter(:))) / (max(GFilter(:)) - min(GFilter(:)));

LFilter=[0, 1, 0;
         1,-4, 1;
         0, 1, 0];
```

```
noise=imread("/home/Baley/UTAS/ENG204 - Signals And Linear Systems/Assignment
↪  1.3/Pic/image_1_noise.jpg");
```

```
Blur=conv2(noise,GFilter,'same');

Edge=conv2(Blur,LFilter,'same');
Threshold = 0;
EdgeDetect = Edge > Threshold;
output=noise+2*EdgeDetect;

subplot(1, 2, 1);
imshow(output, []);
title('Output');
subplot(1, 2, 2);
imshow(noise, []);
title('Input');
%
%subplot(2, 2, 2);
%imshow(Blur, []);
%title('Blurred Image');
```

```
%
%subplot(2, 2, 3);
%imshow(Edge, []);
%title('Edge Detected Image');
%
%subplot(2, 2, 4);
%imshow(EdgeDetect, []);
%title('Edge Detection Result');
```