Universiteit van Amsterdam

Report

Beeldverwerken

Auteurs:
Philip Bouman, 10668667
Steven de Weille, 10606750

25 april 2016

1 Convolution

1.1 f = {1 $\underline{2}$ 1}
g = {0 0 0 0 $\underline{1}$ 1 1 1 1}

Applying convolution results in:

f*g = { 1 3 4}

The border cases are handled by cutting them off. A 1 x n input (f) will produce a 1 x n output (f*g).

1.2 f = {0 0 0 0 $\underline{1}$ 1 1 1 1}
g = {1 $\underline{2}$ 1}

Applying convolution results in:

f*g = {0 0 0 1 3 4 4 4 3}

The border cases are handled by adding a zero, when the index of (g) was out of bounds.

1.3 f = { 0 0 0 0 $\underline{1}$ 1 1 1 1}
g = {-1 $\underline{1}$}

Applying convolution results in:

f*g = {0 0 0 -1 0 0 0 0 1}

The border cases are handled by adding a zero, when the index of (g) was out of bounds.

1.4 Proof of commutative property of convolution:

$(f * g)(i) = \sum_{j=-\infty}^{\infty} f(i-j)g(j)$
Then substitute j for $(i-j)$
$= \sum_{j=-\infty}^{\infty} f(i-(i-j)g(i-j)$
$= \sum_{j=-\infty}^{\infty} f(i+(-i)+j)g(i-j)$
$= \sum_{j=-\infty}^{\infty} f(j)g(i-j)$
$= \sum_{j=-\infty}^{\infty} h(i-j)f(j)$
$= (g * f)(i)$

1.5 Proof of associative property of convolution:

$(f_1 * (f_2 * f_3))(n) = ((f_1 * f_2) * f_3)(n)$

$(f_1 * (f_2 * f_3))(n) = \sum_{-\infty}^{\infty} f_1(k_1)f_2(k_2)f_3((n-k_1)-k_2)$

$$= \sum_{-\infty}^{\infty} f_1(k_1) f_2(k_1 + k_2) - k_1) f_3((n - (k_1 + k_2))$$
$$= \sum_{-\infty}^{\infty} f_1(k_1) f_2(k_3 - k_1) f_3(n - k_3)$$
$$= ((f_1 * f_2) * f_3)$$

1.6 With the identity operation the image stays the same: $g = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \underline{1} & 0 \\ 0 & 0 & 0 \end{bmatrix}$

1.7 Multiplying the image intensity by 3:

In 3D: $g = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \underline{3} & 0 \\ 0 & 0 & 0 \end{bmatrix}$

in 1D: $g = \begin{bmatrix} \underline{3} \end{bmatrix}$

1.8 The image is shifted 3 pixels to the left:

$g = \begin{bmatrix} 0 & 0 & 0 & \underline{0} \\ 1 & 0 & 0 & 0 \end{bmatrix}$

1.9 An image can not be rotated trough a certain angly by using convolution, since the pixels do not move in the same direction while being rotated.

1.10 $g = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \underline{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$

1.11 It is not possible to sort the values in the matrix, since it is not a linear operation.

1.12 Computing the minimum value in a 5x5 neighbourhood is not possible, since it is not a linear operation. The convolution kernel is specific for each point.

1.13 Performing motion blur, results in:

$g = \frac{1}{5} \begin{bmatrix} 1 & 1 & 1 & 1 & \underline{1} \end{bmatrix}$

1.14 A Gaussian with a standard deviation of 3:
$\frac{1}{2 \cdot 3^2 \pi} \cdot e^{-\frac{x^2 y^2}{18}} = \frac{1}{18\pi} \cdot e^{-\frac{x^2 y^2}{18}}$

1.15 Unsharp masking of an image:

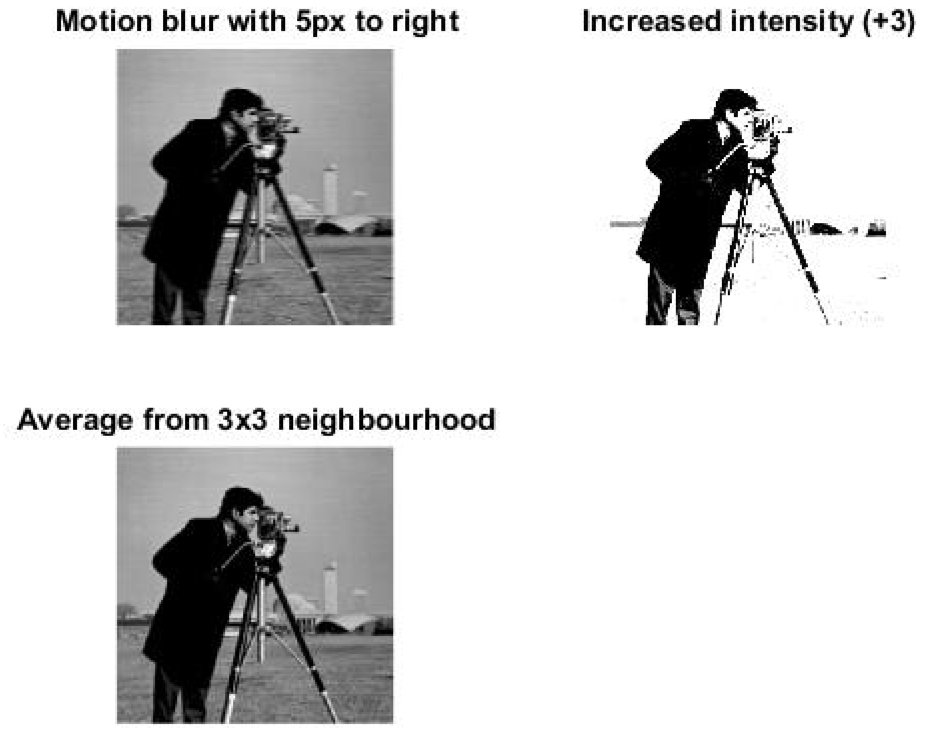
$g = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \underline{1} & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \underline{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$

1.16 Taking an approximation of the derivative in the x-direction:

$g = \frac{1}{2} \begin{bmatrix} -1 & \underline{0} & 1 \end{bmatrix}$

1.17 Taking the second derivative in the z-direction:

$g = \frac{1}{2} \begin{bmatrix} -1 & \underline{0} & 1 \end{bmatrix}$
$g * g = \frac{1}{4} \begin{bmatrix} 1 & 0 & \underline{-2} & 0 & 1 \end{bmatrix}$



Figuur 1: 3 filters applied

1.18 Zooming in on an image is not possible through a convolution kernel, since it requires operations in multiple different directions.

1.19 Thresholding an image is not possible through convolution, since it can not detect differences between points in the same image.

1.20 $G\sigma(x) = \frac{1}{(\sqrt{2\pi\sigma})} e^{-\frac{|x|^2}{2\sigma^2}}$
$\frac{\partial G\sigma}{\partial x} = -\frac{x}{\sigma^2} G\sigma x$
The Gaussian function can be derived infinitely, because the derivative of an e^n term is $n * e^n$, the higher order derivative will never be 0.

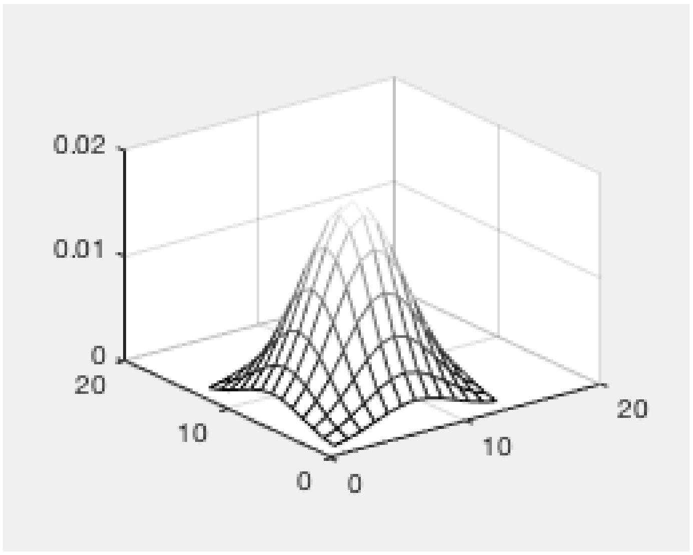
1.21 Derivative only uses local information, hence it can be written as a convolution.

1.22 Taking the derivative of an image and then apply a Gaussian filter results in the following formula: $G * (\delta * f)$. Because of the associative property of convolution this can be written as $G * \delta * f$. The commutative property allows us to rewrite this as $\delta * G * f$. Again applying associativity results in $(\delta * G) * f$. The final result represents how we apply a Gaussian derivative kernel to an image f.

1.23 σ is a constant, so we take the derivative of the e term $= \frac{-x^2}{2\sigma^2}$
$\frac{\partial x}{\partial \sigma} = \frac{-2x}{2\sigma^2} = \frac{-x}{\sigma^2}$ and that times the initial function $G\sigma x$

1.24 Because you have to take the product rule of $\frac{-x}{\sigma^2}$ and $e^{-\frac{|x|^2}{2\sigma^2}}$. The rsult is $\frac{-1}{\sigma^2} \cdot e^{-\frac{|x|^2}{2\sigma^2}} + \frac{-x}{\sigma^2} \cdot \frac{-x}{\sigma^2}$
$\frac{\partial^2 G}{\partial x^2} = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}) G_\sigma(x)$.

1.25



Figuur 2: Gaussian kernel

1.26 2D Gaussian is seperable:

$$G_{\sigma,x} = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{x^2}{2\sigma^2})$$
$$G_{\sigma,x} = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{y^2}{2\sigma^2})$$
$$G_{\sigma,x}G_{\sigma,y} = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{x^2}{2\sigma^2}) \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{y^2}{2\sigma^2})$$
$$= \frac{1}{\sqrt{2\pi\sigma}} exp(\frac{-x^2-y^2}{2\sigma^2})$$
$$= G_\sigma$$

1.27 Show all derivatives of the 2D Gaussian functions are seperable:

$$\frac{\delta(\frac{\delta G_\sigma(x)}{\delta x})}{\delta y}(y) = \frac{\delta G_\sigma(x,y)}{\delta x \delta y}$$

2 Implementation of Gaussian derivatives

2.1 Looking at the normal distribution we see that $2 \cdot \sigma$ is appropriate.

2.2 The expected value was 1, yet we did not get this result in the matlab calculation. As the value for sigma increases, the value of the sum of the kernel decreases. Take the sum of the e term of the gaussian and divide the gaussian by it to avoid this effect.

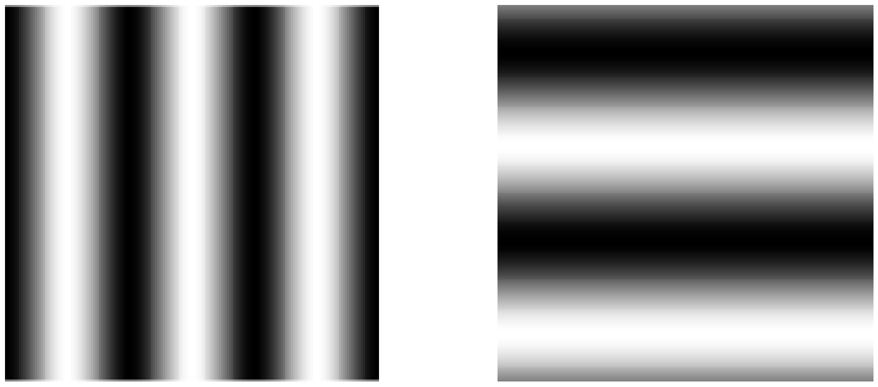
2.3 Plot of the Gaussian kernel: (figure 2)

2.4 The physical unit of the scale parameter σ in this case influences how far the pixels are smeared or how intense the blur is. The physical unit of σ is therefore the distance pixels are smeared across an image.

2.5

2.6 Written down in big-O notation, the order of the computational complexity is O(sigma).

2.7 The complexity is now O(n).



Figuur 3: f_x and f_y

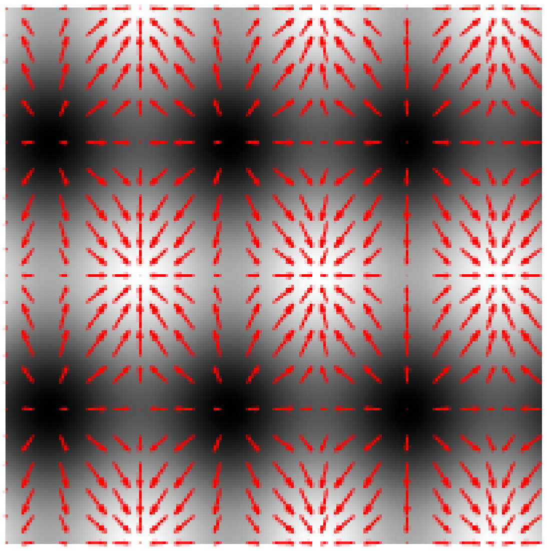
3 The Canny Edge Detector

3.1 The derivatives $(f_x, f_y, f_{xx}, f_{yy}, f_{xy})$ of $f(x,y) = Asin(Vx) + Bcos(Wy)$:

$$f_x = A \cdot V \cdot cos(V \cdot x)$$
$$f_y = -B \cdot W \cdot sin(W \cdot y)$$
$$f_{xx} = -A \cdot V^2 \cdot sin(V \cdot x)$$
$$f_{yy} = -B \cdot W^2 \cdot cos(W \cdot y)$$
$$f_{xy} = 0$$

3.2 See figure 3.

3.3 See figure 4.



Figuur 4: Image with quiver