

Bildverarbeitung: Übung 03

von
Naja v. Schmude (4127652)

Aufgabe 1

Die 2D-Konvolution funktioniert (nach dem Satz) wie folgt:

$$konvolution = F^{-1}(FKF) \otimes (FBF)F^{-1}$$

Wobei \otimes die komponentenweise Multiplikation darstellt.

In Octave ist dies in zwei Zeilen zu bewerkstelligen (aufgrund von Performance habe ich die schon existierenden FFT-Funktionen verwendet).

```
% konvolution of the grayscale image with the kernel
function convolution = convolution2D(img, kernel)
    convolutionFreq = fft2(kernel) .* fft2(img);
    convolution = ifft2(convolutionFreq);
end
```

Nun sollten mit verschiedenen Kernen (Gauß mit $\sigma = 3$, Gauß mit $\sigma = 10$, Sobel-X, Sobel-Y, Uniform) das gegebenen Bild konvolviert (sagt man das so??) und dann die Ergebnisse verglichen werden.

Da für Konvolution die Dimension der Kernel genauso groß sein muss wie die des Bildes, müssen die Leerstellen mit Nullen aufgefüllt werden, so dass der Sobel-X Kernel dann letztendlich folgende Form hat:

$$Sobel_X = \begin{pmatrix} 1 & 0 & -1 & 0 & \dots & 0 \\ 2 & 0 & -2 & 0 & \dots & 0 \\ 1 & 0 & -1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Die Gauß-Kernel habe ich auch nur als 6×6 -Matrix benutzt, wobei die Werte entsprechend über folgende Funktion berechnet wurden:

```
% gauss 2D
function g = gauss2D(sigma, x,y)
    g = exp(-( (x^2/ (2*sigma^2)) + (y^2 / (2*sigma^2) ) ));
end
```

Anschließend wurden die Gauß-Kernel dann noch normiert, so dass die mittlere Helligkeit des Bildes bestehen bleibt. Ebenso wurde auch der Uniform-Kernel dann normiert (er hat ja die Form, dass in allen Zellen der 5×5 -Matrix der selbe Wert steht).

Das Ergebnis sieht dann wie folgt aus:



Es ist zu erkennen, dass die beiden Sobel-Filter sehr schön die Kanten in die jeweilige Richtung finden. Die drei restlichen Kernel haben das Bild weichgezeichnet, wobei ich nicht wirklich Unterschiede in der Weichzeichnung erkennen kann. Wenn man ganz genau hinschaut könnte man meinen, dass die Stärke der Weichzeichnung von Uniform zu Gauß mit $\sigma = 3$ und dann zu Gauß mit $\sigma = 5$ zunimmt. Aber ich finde die Unterschiede eher marginal.

Aufgabe 2

Hier waren nun die konvoluierten Bilder aus Aufgabe 1 zu dekonvolutieren. Die Dekonvolution funktioniert allgemein so:

$$\text{dekonvolution} = F^{-1}(F B F) \div (F K F) F^{-1}$$

Wobei \div die komponentenweise Division darstellt, K der Kernel ist und B das mit K konvoluierte Bild.

In Octave habe ich das wie folgt umgesetzt.

```
% deconvolution of the convoluted image with the given kernel
function deconvolution = deconvolution2D(img, kernel)
    deconvolutionFreq = fft2(img) ./ fft2(kernel);
    deconvolution = ifft2(deconvolutionFreq);
end
```

Das Problem bei der Division ist, dass man z.B. beim Dekonvolutieren der Sobel-Filter gefalteten Bilder Nullen im Nenner bekommt und dadurch die Dekonvolution nicht richtig funktioniert. Zumindest habe ich dann nur schwarze Bilder erhalten. In den anderen Fällen hat die Dekonvolution super geklappt und ich habe keine Unterschiede zum Ausgangsbild feststellen können.

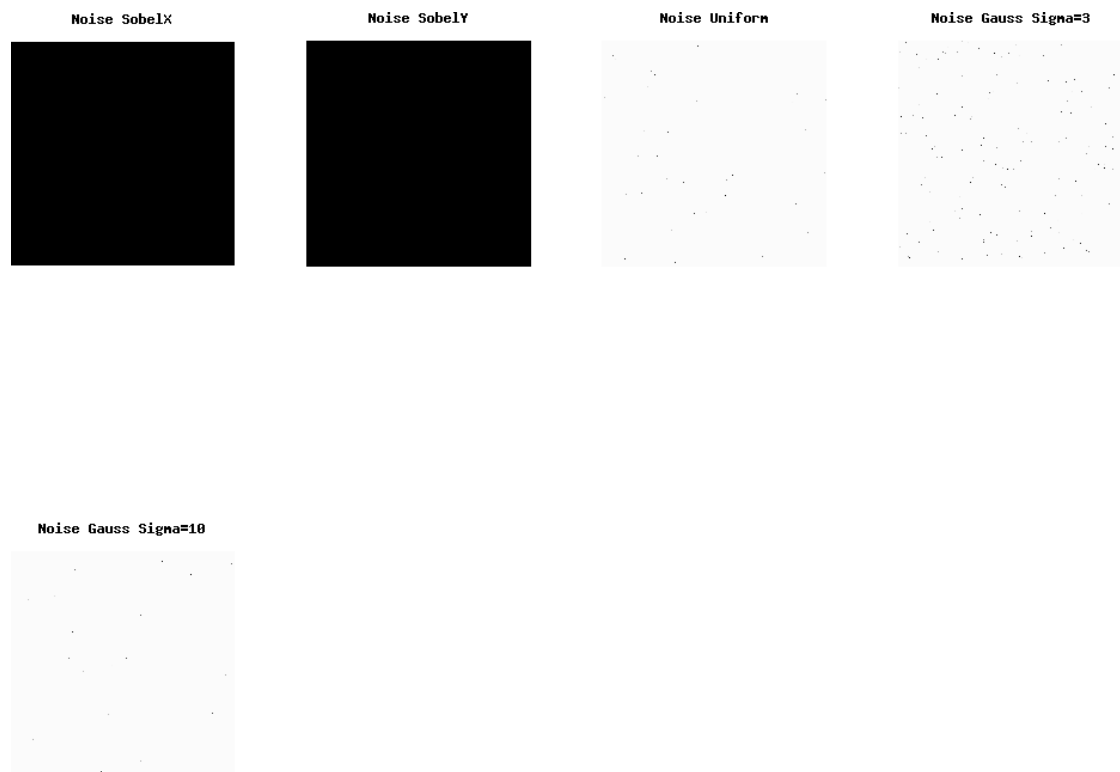


Aufgabe 3

In dieser Aufgabe sollten den in Aufgabe 1 konvolvierten Bildern noch gauß'sches Rauschen hinzugefügt werden und dann die verrauschten Bilder dekonvolviert. Die Bilder wurden wie folgt verrauscht und dann entsprechend wie in Aufgabe 2 dekonvolviert:

```
% add's gaussian noise to the image
function noisy = gaussianNoise(img)
    [m,n] = size(img);
    noisy = image + normrnd(0,1,m,n);
end
```

Das Ergebnis war erstaunlich:



Mit also schon geringen Änderungen funktioniert die Dekonvolution überhaupt nicht mehr. Die Ursprungsbilder sind in keinsterweise wieder zu erkennen.