

1. Gegeben sei das Bild

1	1	1	1	10	10	10	10
1	1	6	1	8	10	2	10
1	3	1	1	9	10	7	10
1	1	1	2	8	9	10	10
1	1	1	1	10	10	10	10
1	4	1	2	9	10	2	10
1	2	1	8	10	10	10	10
1	1	1	1	10	10	10	10

Wenden Sie darauf

(a) den Gaußschen Tiefpass,

(b) den Medianfilter an!

Der Rand des Ausgabebildes sei jeweils mit Null vorbelegt.

(a)

Auf die Matrix wird ein entsprechender Filter gelegt:

0	0	0	0	0	0	0	0
0	2	3	7	9	8	7	0
0	1	3	7	9	10	7	0
0	1	3	7	9	9	7	0
0	2	4	8	9	8	7	0
0	2	5	9	9	9	7	0
0	1	3	6	8	8	6	0
0	0	0	0	0	0	0	0

(b)

Die Werte der Nachbarn werden in einer Reihe sortiert, der mittlere Wert ist der Median.

0	0	0	0	0	0	0	0
0	1	1	1	9	10	10	0
0	1	1	2	8	9	10	0
0	1	1	1	9	10	10	0
0	1	1	2	9	10	10	0
0	1	1	2	10	10	10	0
0	1	1	2	10	10	10	0
0	0	0	0	0	0	0	0

2. Matlab: Filter

Es gibt in Matlab eingebaute Filter und die Möglichkeit eigene zu benutzen. Laden Sie ein Graubild Ihrer Wahl.

(a) Filtern Sie das Bild mit einem 3 x 3-Mittelwertfilter!

```
function ergebnis = mittelwertfilter(matrixgroesse)
% Transformation der Grauwerte auf Gleichverteilung

% matrixgroesse = 3;
werte = zeros(matrixgroesse * matrixgroesse, 1);

von_bis = (matrixgroesse - 1) / 2;

% Bild laden
bild = imread('lena_std_sw.png');
zeilenspalten = size(bild); % Zeilen und Spalten des Bildes auslesen
vector = double(bild);
vector1 = double(bild);

zeilenspalten = size(vector);

for zeile=(1):zeilenspalten(1)
    for spalte=1:zeilenspalten(2)
        summe = 0;
        if (zeile < von_bis + 1 | zeile > zeilenspalten(1) - von_bis | spalte <
von_bis + 1 | spalte > zeilenspalten(2) - von_bis)
            vector1(zeile, spalte) = 0;
        else
            for bzeile=(zeile - von_bis):(zeile + von_bis)
                for bspalte=(spalte - von_bis):(spalte + von_bis)
                    summe = summe + vector(bzeile, bspalte);
                end
            end
            vector1(zeile, spalte) = double(summe / (matrixgroesse *
matrixgroesse));
        end
    end
end

figure('Name','Original'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(bild);

figure('Name','Ergebnis nach Mittelwertfilter'), set(gcf,'Color','white');
set(gca, 'Box','off');
imshow(uint8(vector1));
```

Aufruf des Programms über `mittelwertfilter(3)`;

(b) Wie verändert sich das Ergebnis, wenn Sie stattdessen einen Mittelwertfilter mit 5 x 5 oder t x 7 Maske oder noch größer wählen?

Aufruf des Programms über `mittelwertfilter(7)`;

Die Konturen werden weicher, dadurch erscheint das Bild "unscharf". Wegen des gewählten Algorithmus für den Rand ist ein dickerer schwarzer Rand zu sehen.

(c) Probieren Sie verschiedene Randooptionen! Bei welcher gefiel Ihnen das Ergebnis am besten und warum?

Natürlich bei der Spiegelung, weil hier das Ergebnis am realistischsten ist. Dafür ist der Algorithmus ein wenig komplizierter.

Der einfachste Algorithmus: Einfach die Randpixel mit einem beliebigen Wert belegen.

(d) Filtern Sie das Bild mit einem Gaußschen Tiefpass (siehe Vorlesung)!

```
function ergebnis = gausstiefpass()
% Transformation der Grauwerte auf Gleichverteilung

% alles loeschen
clear
close all

matrixgroesse = 3;

von_bis = (matrixgroesse - 1) / 2;

% Bild laden
bild = imread('lena_std_sw.png');
zeilenspalten = size(bild); % Zeilen und Spalten des Bildes auslesen
vector = double(bild);
vector1 = double(bild);

zeilenspalten = size(vector);

for zeile=(von_bis + 1):(zeilenspalten(1) - von_bis)
    for spalte=(von_bis + 1):(zeilenspalten(2) - von_bis)
        summe = 0;
        if (zeile < von_bis + 1 | zeile > zeilenspalten(1) - von_bis | spalte <
von_bis + 1 | spalte > zeilenspalten(2) - von_bis)
            vector1(zeile, spalte) = 0;
        else
            for bzeile=(zeile - von_bis):(zeile + von_bis)
                for bspalte=(spalte - von_bis):(spalte + von_bis)
                    if (bspalte == (spalte - von_bis) | bspalte == (spalte +
von_bis)) % 1.+3. Spalte
                        if (bzeile == zeile - von_bis | bzeile == zeile +
von_bis) % 1.+3. Zeile
                            summe = summe + vector(bzeile, bspalte);
                        else
% 2. Zeile
                            summe = summe + vector(bzeile, bspalte) * 2;
                        end
                    else
% 2. Spalte
                        if (bzeile == (zeile - von_bis) | bzeile == (zeile +
von_bis)) % 1.+3. Zeile
                            summe = summe + vector(bzeile, bspalte) * 2;
                        else
% 2. Zeile
                            summe = summe + vector(bzeile, bspalte) * 4;
                        end
                    end
                end
            end
        end
        vector1(zeile, spalte) = summe / 16; % Gauss
    end
end
end

figure('Name','Original'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(bild);

figure('Name','Ergebnis nach Gaussschem Tiefpass'), set(gcf,'Color','white');
```

```
set(gca, 'Box','off');
imshow(uint8(vector1));
```

(e) Wenden Sie die beiden Sobelfilter an und erzeugen Sie ein Bild mit dem Betrag der Richtungsdifferenz! Welche Schwierigkeiten treten auf? Wie haben Sie sie behoben?

Das Problem äußert sich dadurch, daß auf das erstmals gefilterte Ergebnis kein anderer Sobelfilter angewendet werden kann. Behoben wird das Problem, indem beide Ergebnisse aus einem gemeinsamen Bild hervorgehen und per Mittelwertberechnung zu einem Bild vereint werden. Da das Bild auch noch dunkel ist, werden die Werte mit 127 in einen mittleren Grauwert "gehoben". Durch Spielereien mit den Formeln für H_x und H_y kann der Kontrast weiter hervorgehoben werden: Die Herabsetzung des Quotienten (9) bewirkt einen stärkeren Kontrast, die Veränderung des Addierers (127) bewirkt eine Aufhellung oder Abdunklung des gesamten Bildes.

```
function ergebnis = sobel()
% Transformation der Grauwerte auf Gleichverteilung

% alles loeschen
clear
close all

matrixgroesse = 3;

von_bis = (matrixgroesse - 1) / 2;

% Bild laden
bild = imread('lena_std_sw.png');
zeilenspalten = size(bild); % Zeilen und Spalten des Bildes auslesen
vector = double(bild);
vector1 = double(bild);
H_x = double(bild);
H_y = double(bild);

zeilenspalten = size(vector);

for zeile=(von_bis + 1):(zeilenspalten(1) - von_bis) %
    for spalte=(von_bis + 1):(zeilenspalten(2) - von_bis)%
        summe_x = 0;
        summe_y = 0;
        if (zeile < von_bis + 1 | zeile > zeilenspalten(1) - von_bis | spalte <
von_bis + 1 | spalte > zeilenspalten(2) - von_bis)
            H_x(zeile, spalte) = 0;
            H_y(spalte, zeile) = 0;
        else % Sobelfilter H_x
            for bzeile=(zeile - von_bis):(zeile + von_bis)
                for bspalte=(spalte - von_bis):(spalte + von_bis)
                    if (bzeile == (zeile - von_bis)) % 1. Zeile
                        if (bspalte == spalte - von_bis)
                            summe_x = summe_x + vector(bzeile, bspalte);
                            summe_y = summe_y + vector(bspalte, bzeile);
                        end
                    if (bspalte > spalte - von_bis & bspalte < spalte -
von_bis) % 2. Spalte
                        summe_x = summe_x + vector(bzeile, bspalte) * 2;
                        summe_y = summe_y + vector(bspalte, bzeile) * 2;
                    end
                    if (bspalte == spalte - von_bis)
                        summe_x = summe_x + vector(bzeile, bspalte);
                    end
                end
            end
        end
    end
end
```

```

        summe_y = summe_y + vector(bspalte, bzeile);
    end
end
if (bzeile == (zeile + von_bis)) % 3. Zeile
    if (bspalte == spalte - von_bis)
% 1. Spalte
        summe_x = summe_x - vector(bzeile, bspalte);
        summe_y = summe_y - vector(bspalte, bzeile);
    end
    if (bspalte > spalte - von_bis & bspalte < spalte -
von_bis) % 2. Spalte
        summe_x = summe_x - vector(bzeile, bspalte) * 2;
        summe_y = summe_y - vector(bspalte, bzeile) * 2;
    end
    if (bspalte == spalte - von_bis)
% 3. Spalte
        summe_x = summe_x - vector(bzeile, bspalte);
        summe_y = summe_y - vector(bspalte, bzeile);
    end
end
end
end
H_x(zeile, spalte) = summe_x / 9 + 127;
H_y(spalte, zeile) = summe_y / 9 + 127;
end
end
end
end

% Mittelwerte aus beiden Sobel-Bildern
for zeile=1:zeilenspalten(1)
    for spalte=1:zeilenspalten(2)
        vector1(zeile, spalte) = (H_x(zeile, spalte) + H_y(zeile, spalte)) / 2;
    end
end

figure('Name','Original'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(bild);

figure('Name','H_x'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(uint8(H_x));

figure('Name','H_y'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(uint8(H_y));

figure('Name','Sobel x und y'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(uint8(vector1));

```

3. Matlab: Denoising

Laden Sie das Bild 'eight.tif' !

(a) Verrauschen Sie es mit Einzelstörungen (Salz & Pfeffer) bzw. mit weißem Rauschen!

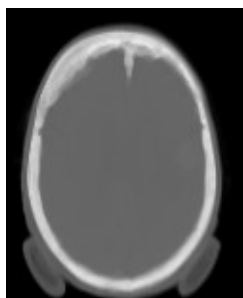
Durch Verwendung der entsprechenden Funktion:

```
bild1 = imnoise(bild, 'salt & pepper', 0.02);
```

(b) Entrauschen Sie es mit dem Mittelwert-, dem Median- und den Wiener Filter! Welcher Filter ist wofür besonders gut geeignet? Evtl. hilft es die Filter mehrfach anzuwenden.

Mittelwertfilter 5x auf 3x3-Matrix angewendet	Medianfilter 1x auf 3x3-Matrix angewendet	Wiener Filter 5x auf 3x3-Matrix angewendet
		

4. Matlab: Morphologische Operationen
Laden Sie das Bild 'maus.png'!

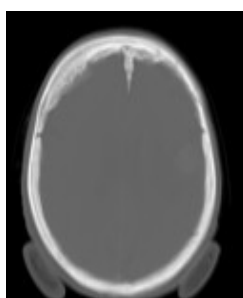


(a) Welche Morphologischen Filter kennen Sie?

Dilatation und Erosion: Helle Punkte/dunkle Punkte dehnen sich zu Lasten der anderen Punkte aus

Opening und Closing: Erosion mit anschließender Dilatation oder umgekehrt sorgen für die Unterdrückung von Pixelstörungen oder das Schließen von Lücken in Objekträndern

(b) Wenden Sie einen 3 x 3-Median an!



(c) Binarisieren Sie das Bild mit dem Schwellwert 105!

```
function ergebnis = binarisieren(grenzwert)
% Transformation der Grauwerte in Binaer nach Grenzwert

% Bild laden
bild = imread('maus.png');
zeilenspalten = size(bild); % Zeilen und Spalten des Bildes auslesen
vector = double(bild);

for zeile=1:zeilenspalten(1)
    for spalte=1:zeilenspalten(2)
        if (vector(zeile,spalte) <= grenzwert)
            vector(zeile,spalte) = 0;
        else
            vector(zeile,spalte) = 255;
        end
    end
end

figure('Name','Original'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(bild);

figure('Name','binarisiert'), set(gcf,'Color','white'); set(gca, 'Box','off');
imshow(uint8(vector));
```



(d) Laden Sie ein Strukturelement (am einfachsten 3 x 3), welche Strukturelemente gibt es noch!

- mit 4er-Nachbarn (wird verwendet beim City-Block-Abstand)
- mit schrägem Element

(e) Wenden Sie mehrfach eine Erosion auf das SW-Bild an!

```
for wiederholung=1:5
for zeile=1:zeilenspalten(1)
    for spalte=1:zeilenspalten(2)
        if (zeile < 2 | zeile == zeilenspalten(1) | spalte < 2 | spalte ==
zeilenspalten(2))
            vector_ero(zeile, spalte) = vector_ero_alt(zeile, spalte);
            vector_dil(zeile, spalte) = vector_dil_alt(zeile, spalte);
        else
            gefunden_ero = 0;
            gefunden_dil = 0;
            for bzeile=zeile - 1:zeile + 1
                for bspalte=spalte - 1:spalte + 1
                    if (vector_ero_alt(bzeile, bspalte) == 0)
```

```

        gefunden_ero = 1;
    end
    if (vector_dil_alt(bzeile, bspalte) == 255)
        gefunden_dil = 1;
    end
end
end
if (gefunden_ero == 1)
    vector_ero(zeile, spalte) = 0;
else
    vector_ero(zeile, spalte) = 255;
end
if (gefunden_dil == 1)
    vector_dil(zeile, spalte) = 255;
else
    vector_dil(zeile, spalte) = 0;
end
end
end
end
end

```



(f) Wenden Sie mehrfach eine Dilation auf das SW-Bild an!



(g) Probieren Sie Opening und Closing aus!

Opening: Erosion, anschließend Dilatation



Original (binarisiert)



Erosion



Dilatation

Closing: Dilatation, anschließend Erosion



Original (binarisiert)



Dilatation



Erosion

5. Zusatz: Filteroperationen mit OpenGL:

(a) Bauen Sie in Ihr Beispielprogramm oder in `myconvolution.c` die Möglichkeit der Filterung ein!

(b) Realisieren Sie den 3x3-Mittelwertfilter, den 5x5-Mittelwertfilter, den Gaußfilter,

(c) die Differenzenfilter

$$\frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad \text{und} \quad \frac{1}{3} \begin{pmatrix} 10 & -1 \\ 10 & -1 \\ 10 & -1 \end{pmatrix}$$

(d) Probieren Sie auch die verschiedenen Randoptionen aus!