Elementare Bildverarbeitungsoperationen

- Lineare Filterung -

1 Einführung

- **Graustufenbild** $f\colon [0,M] \times [0,N] \cap \mathbb{Z}^2 \mapsto [0,255] \cap \mathbb{Z},\ (x,y) \to f(x,y)$

	0	1	2	3	4	5	6	M=7→x
0	111	100	121	168	189	255	165	143
1	88	78	82	190	99	201	111	167
2	76	56	65	153	132	167	156	159
3	72	53	60	186	156	108	132	107
4	101	97	178	160	122	123	170	85
5	124	146	190	205	188	179	154	97
6	143	156	253	238	176	148	167	109
N=7	167	189	205	176	144	175	199	154
↓ y								

- **Pixel** (x,y)
- **Nachbarschaft**
  $\Rightarrow$ Nachbarschaftsoperationen
- Anwendungen:
  1. Erkennung von Bilddetails
  2. Rekonstruktion von Bildern

2 Definition der Faltung

$$g(x,y) = (h * f)(x,y) := \sum_{k=-n}^{n} \sum_{j=-m}^{m} h(j,k)\, f(x-j, y-k),$$

falls $m \leq x \leq M - m$ und $n \leq y \leq N - n$

$g(x,y) = (h * f)(x,y) := 0,$ sonst

Beispiel:

..	..	..	..	..
..	65	153	132	..
..	60	186	156	..
..	178	160	122	..
..	..	..	..	..

3×3-Nachbarschaft von $(3,3)$

d.h. $m = n = 1$

	-1	0	+1→j
-1	-1	0	1
0	-2	0	2
+1 ↓ k	-1	0	1

Filter(matrix) $h(j,k)$

d.h. $g(x, y) =$ "gewichtete Summe der $f(x, y)$ in einer Nachbarschaft"

Merke: Die Faltung ist linear!

Motivation:

Was bedeutet eigentlich Filterung in der Bildverarbeitung?

- Zielsetzungen: Schärfen des Bildes oder auch Verwischung
- Beispiel: Hi-Fi Anlage mit Bass- und Höhenregelung
  $\Rightarrow$ 2 Arten linearer Filter:
  1. Low pass filter (Niederpassfilter)
  2. High pass filter (Hochpassfilter)
- Modifikation des High pass filters $\Rightarrow$ High frequency emphasis

3 Beispiele linearer Filter

	Low pass filtering	High pass filtering	High frequency emphasis
Aussehen	$h_{ij} > 0$	$h_{ij} \in \mathbb{R}$	$h_{ij} \in \mathbb{R}$
Wirkung	Filterung der hohen Frequenzen	Filterung der niedrigen Frequenzen	Verknüpfung des Originalbildes mit dem Ergebnis eines High pass filters

	Low pass filtering	High pass filtering	High frequency emphasis
Eigenschaften	Reduzierung von Bildrauschen, aber gleichzeitig Verwischung von Details	Hervorhebung der Hochfrequenzbereiche, aber u.U. auch Verstärkung von Bildrauschen	Betonung der hohen Frequenzen relativ zu den niedrigen
Anwendung	Glätten bzw. Verwischen des Bildes	Erkennung von Bildbereichen mit starken Änderungen	Schärfung von Bildern

4 Diskussion

Auftretende Probleme bei der Verwendung linearer Filter:

1. Erzeugung eines Bildes desselben Datentyps
   Lösung:
   - Normierung
   - Mapping

2. gleiche Operation auf (fast) sämtlichen Pixeln
   Lösung: **Adaptive Filter**
   $\Rightarrow$ Filter abhängig von lokalen Bildgegebenheiten

3. Vorgehen am Bildrand

   Lösung:

   - Kopie der Randpixel des Eingabebildes
   - Abschneiden des Bildes
   - Abschneiden des Filters
   - Spiegelung
   - Periodisierung

5 Ausblick: Nichtlineare Filterung

Motivation

- Faltung nur eine Möglichkeit für Filterung
- verschiedene nicht-lineare Vorgehensweisen
- weniger wichtig für praktische Anwendungen
- große Klasse: **Rang-Filter** $\curvearrowright$ Ordnungsstatistiken
- Mischungen von linearen und nicht-linearen Filtern: **Hybrid-Filter**

1. **Rang-Filter (Rank filtering)**
   - Median-Filter

     Wirkung: Beseitigung bestimmter Arten von Störungen
   - Minimum-Filter

     Wirkung: Verdunkelung des Bildes
   - Maximum-Filter

     Wirkung: Erhellung des Bildes
   - Range-Filter

     Wirkung: Erhellung von Gebieten mit hohen Änderungen in den Graustufen bzw. Verdunkelung von Gebieten mit geringen Variationen

2. **Hybrid-Filter**

- klassisches Beispiel: **eingeschränkter Mittelwertfilter**

$$\frac{1}{n^2 - 2\alpha} \sum_{i=\alpha+1}^{n^2-\alpha} f_i$$

wobei $0 \leq \alpha \leq \frac{n^2-1}{2}$ und $f_1 \leq f_2 \leq ... \leq f_{n^2}$

- $\alpha = 0$: reiner Mittelwert-Filter
- $\alpha = \frac{n^2-1}{2}$: Median-Filter
- $0 \leq \alpha \leq \frac{n^2-1}{2}$: Kombination von Mittelwert- und Median-Filter

  Wirkung: bessere Beseitigung von Rauschen

6 Literatur

- N. Efford, *Digital Image Processing: A Practical Introduction using Java*, Addison-Wesley, 2000
- B. Jähne, *Digital Image Processing: Concepts, Algorithms and Scientific Applications*, Springer-Verlag, 2002
- D. A. Lyon, *Image Processing in Java*, Prentice Hall, 1999
- P. Soille, *Morphological Image Analysis: Principles and Applications*, Springer-Verlag, 1998

Elementare Bildverarbeitungsoperationen

- Kantenerkennung -

Fabian Bohnert, Dieter Kiesenbauer

1 Einführung

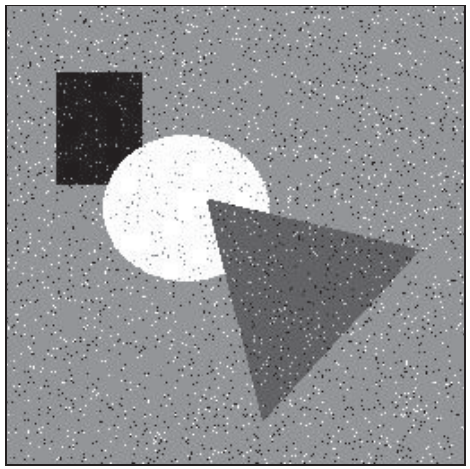
Kantenerkennung basiert auf *linearer Filterung*.

Definition

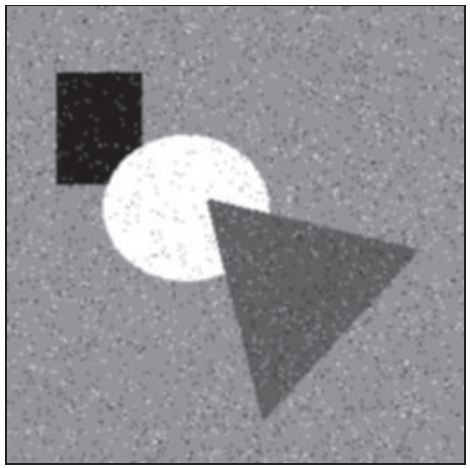
Ein Pixel nennt man *Kantenpixel*, wenn sich seine Nachbarschaft durch hohe Graustufenvariationen auszeichnet.
Eine Menge von Kantenpixeln nennt man *Kante*, wenn die Kantenpixel paarweise über einen Pfad in der Menge verbunden sind.

Den **Prozess der Kantenerkennung** kann man in drei Schritte untergliedern:

1. Reduktion des Bildrauschens
2. Kantenhervorhebung
3. Kantenlokalisation

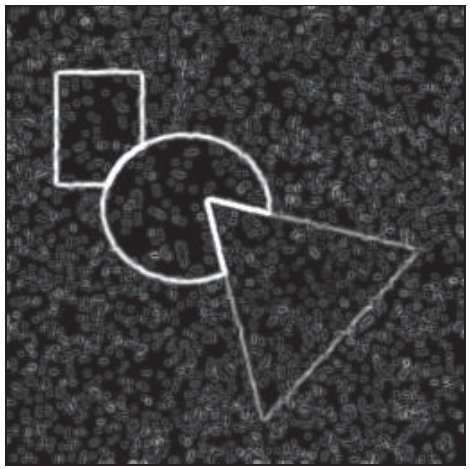


Wir betrachten ein verrauschtes Bild.



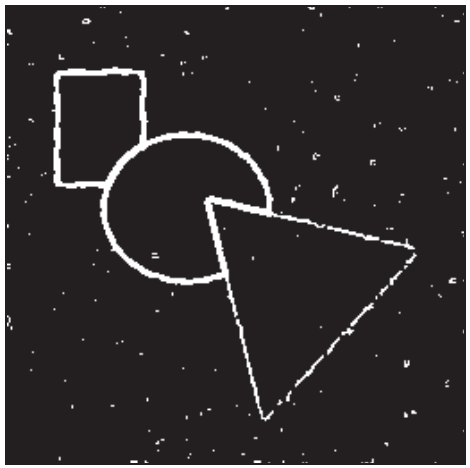
Reduktion des Bildrauschens

Das Bildrauschen sollte so stark wie möglich unterdrückt werden, ohne die eigentlichen Kanten stark zu glätten.



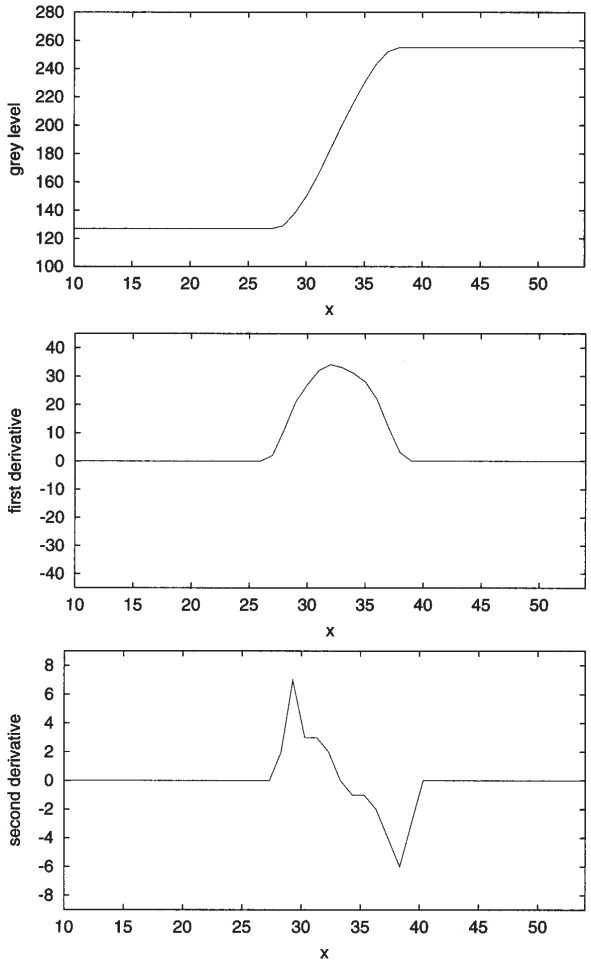
Kantenhervorhebung

In diesem Schritt wird ein (linearer) Filter ($\rightarrow$ *High Pass Filter*) auf das Bild angewendet, der Kanten verstärkt und andere Bildstrukturen abschwächt.



Kantenlokalisation

Abschließend entscheidet man, welche hervorgehobenen Pixel echte Kantenpixel sind und erstellt ein *Binärbild*, in dem Kantenpixel auf 1 (weiß) gesetzt sind.



Was zeichnet eine Kante im zweidimensionalen Bild aus?

Betrachtung der 1. und 2. partiellen Ableitungen

2 Gradientenverfahren

Für $f : \mathbb{R}^2 \to \mathbb{R}, f \in \mathcal{C}^1$ gilt $\forall\ (x, y) \in \mathbb{R}^2$:

$$\frac{\partial f}{\partial x}(x,y) = \lim_{h\to 0} \frac{f(x+h,y)-f(x-h,y)}{2h} \overset{h=1}{\approx} \frac{1}{2}[f(x+1,y) - f(x-1,y)]$$

Entsprechendes gilt für $\frac{\partial f}{\partial y}(x, y)$ und es ergibt sich für den Gradienten:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x,y) \\ \frac{\partial f}{\partial y}(x,y) \end{bmatrix} \approx -\frac{1}{2} \begin{bmatrix} f(x-1,y) - f(x+1,y) \\ f(x,y-1) - f(x,y+1) \end{bmatrix}$$

Um das Bildrauschen zu reduzieren, berechnen wir den Gradienten über einer 3×3-Nachbarschaft. Dies führt zu folgender Darstellung mit Hilfe der Faltung:

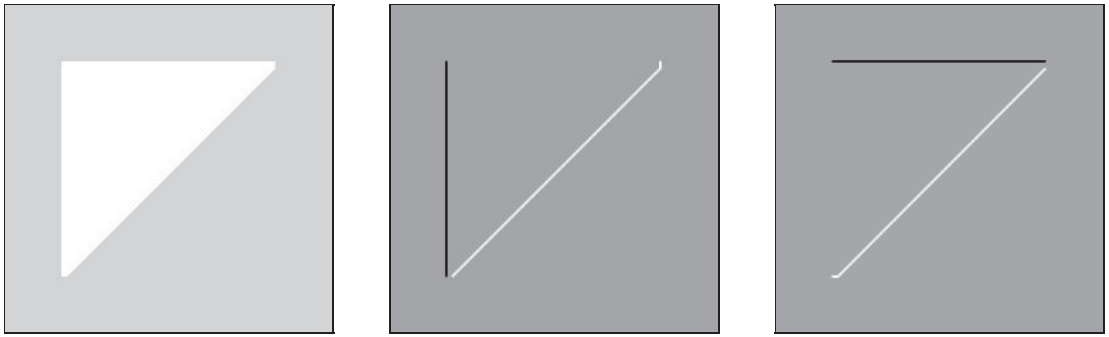
$$\nabla f(x,y) \approx \begin{bmatrix} (h_x * f)(x,y) \\ (h_y * f)(x,y) \end{bmatrix}.$$

Dabei sind h_x und h_y z.B. gegeben durch die sogenannten *Prewitt-Filter*

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \; h_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Durch zweifache Gewichtung der Pixel auf den Achsen ergeben sich die sogenannten *Sobel-Filter*.

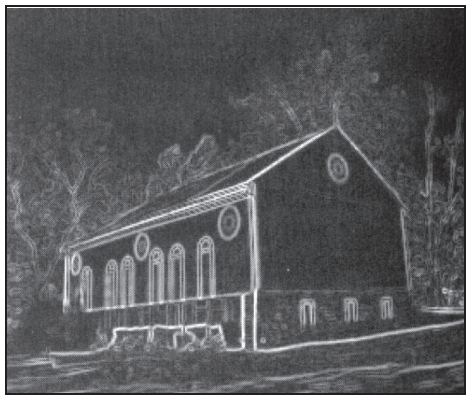
Beispiel



f $\qquad$ $h_x * f$ $\qquad$ $h_y * f$

Der Gradientenbetrag $|\nabla f| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \approx |\frac{\partial f}{\partial x}| + |\frac{\partial f}{\partial y}|$ ist ein Maß für die Graustufenvariation und hebt somit Kanten hervor.

Originalbild



Reduktion des Bildrauschens und Kantenhervorhebung

z.B. durch Anwendung der Prewitt-Filter und anschließende Bildung des Gradientenbetrags

Kantenlokalisation durch einfaches Thresholding

Über einfaches Thresholding (threshold (engl.): Schwellenwert) entscheidet man, ob das Pixel Kantenpixel ist oder nicht.

Betrachte dazu

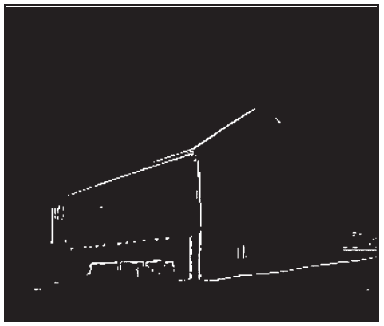
$$g(x,y) = \begin{cases} 0, & |\nabla f(x,y)| < T, \\ 1, & |\nabla f(x,y)| \geq T, \end{cases}$$

wobei T ein fest vorgebener Schwellenwert sei.

Bemerkungen

- Der Erfolg der Kantenerkennung hängt entscheidend von der Wahl des Schwellenwertes T ab:

$T = 50$



$T = 150$

- Kanten zwischen Bildelementen sind nicht immer scharf. Unscharfe Kanten führen zu kleineren Gradientenbeträgen und können so herausgefiltert werden.
- Rauschen kann sehr hohe Gradientenbeträge hervorrufen und kann so fälschlicherweise als Kante erkannt werden.

3 Laplace-Verfahren

Der Laplace-Operator $\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$ ist ein Maß für die Rate der Graustufenvariation und hebt somit Kanten hervor.

1. **Kantenhervorhebung durch Bestimmung des Laplace-Bildes**
   ∇^2 kann über einer 3×3-Nachbarschaft angenähert werden durch

$$\nabla^2 f \approx h * f, \text{ wobei } h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

   Berechne also für jedes Pixel $\nabla^2 f(x, y)$ und führe die Informationen in einem Laplace-Bild zusammen.

2. **Kantenlokalisation**
   erfolgt durch anschließende Suche nach Nulldurchgängen (*zero crossings*).

Bemerkungen

- Zur Reduktion des Bildrauschens wird der Laplace-Filter oftmals mit einem Gauss-Filter kombiniert (*Laplacian of Gaussian - LoG*):

  Mit der Gauss-Funktion $h : \mathbb{R}^2 \to \mathbb{R}$, $h(x, y) = \exp(-\frac{1}{2}\frac{x^2+y^2}{\sigma^2})$ ergibt sich der LoG-Filter

  $$\nabla^2 h(x, y) = (\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}) \exp(-\frac{1}{2}\frac{x^2 + y^2}{\sigma^2}).$$

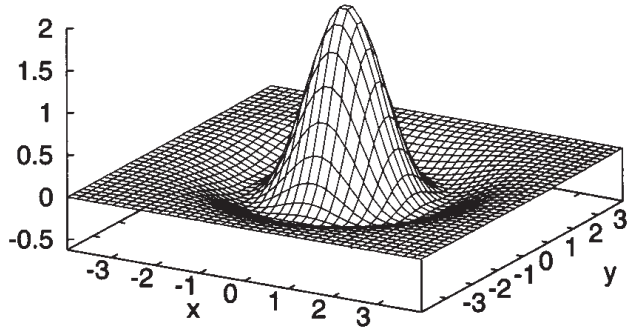
  Dabei bestimmt σ die Breite des Filters ($> 6\sigma$) sowie den Grad der Glättung (und damit die Anzahl erkannter Kanten).

  Aufgrund der Form des Graphen von $\nabla^2 h$ nennt man den LoG-Filter auch *Mexican-Hat-Filter*.

- Das Laplace-Verfahren liefert scharfe Kanten.

LoG-Filterung mit $\sigma = 3.0$

LoG-Filterung mit $\sigma = 5.0$



Graph von $-\nabla^2 h$

4 Canny-Verfahren

Originalbild

Reduktion des Bildrauschens mit Gauss-Filter

Kantenhervorhebung über den Gradientenbetrag

Diese beiden Schritte können auch als Einheit durchgeführt werden.

Kantenlokalisation (1. Teilschritt)

Nicht-Maximum-Unterdrückung

Bereiche um lokale Maxima im Gradientenbetrag (d.h. um Kanten) werden reduziert auf scharfe, ein Pixel breite Kanten.

Kantenlokalisation (2. Teilschritt)

Hysterese

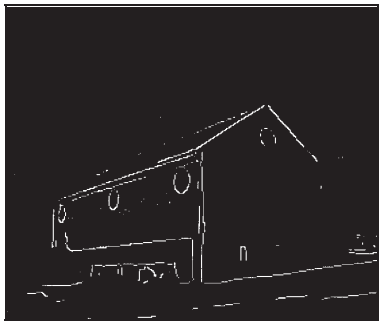
Hysterese bietet einen Lösungsansatz der bereits genannten Probleme beim einfachen Thresholding.

Hysterese arbeitet mit zwei Schwellenwerten T_{low} und T_{high}:

- Liegt ein lokales Maximum über T_{high}, wird es sofort als Kantenpixel erkannt.
- Liegt ein lokales Maximum unter T_{low}, wird es sofort als Kantenpixel abgelehnt.
- Liegt ein lokales Maximum zwischen T_{low} und T_{high}, so wird es akzeptiert, wenn es von ihm aus einen Pfad aus Pixeln mit $f(x_i, y_i) > T_{low}$ zu einem Pixel mit $f(x_0, y_0) > T_{high}$ gibt.

Bemerkungen

- Hysterese verringert das Auftreten von unterbrochenen Kanten dramatisch:



$T = 100$

$T = 50$

$T_{low} = 50,$

$T_{high} = 100$

- Das Canny-Verfahren bietet einen sehr guten Kompromiss zwischen Reduktion des Bildrauschens und Kantenlokalisation.
- Das Canny-Verfahren liefert scharfe Kanten.

5 Literatur

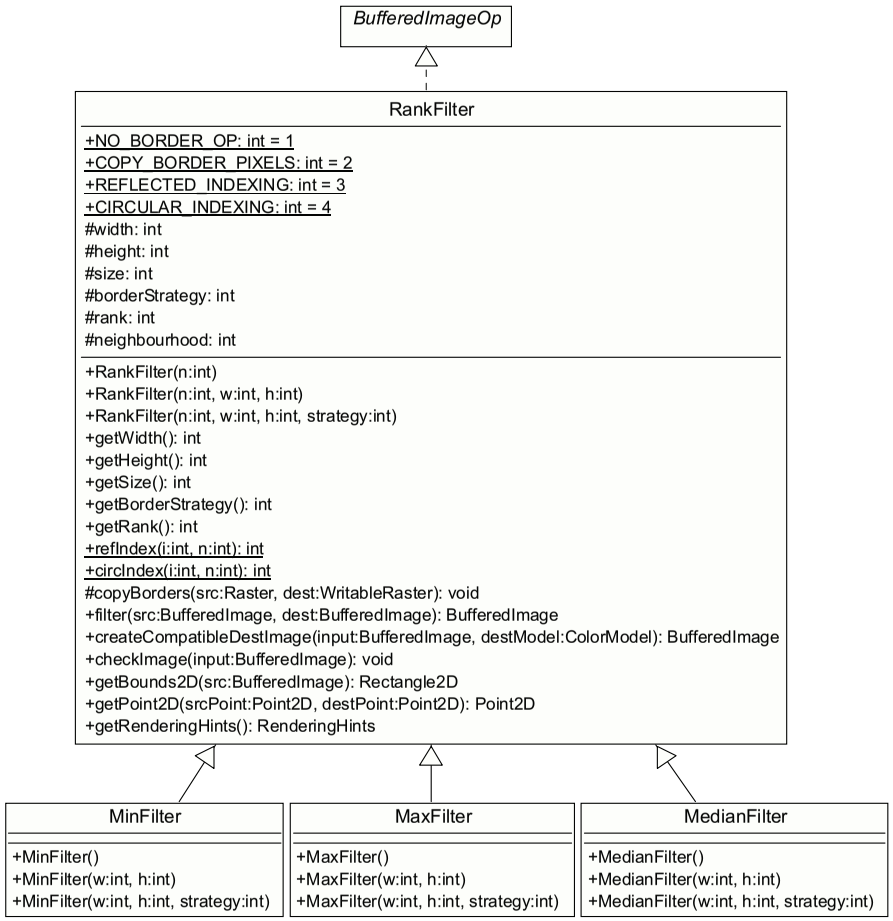
- J. Canny, *A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986
- N. Efford, *Digital Image Processing: A Practical Introduction using Java*, Addison-Wesley, 2000
- B. Jähne, *Digital Image Processing: Concepts, Algorithms and Scientific Applications*, Springer-Verlag, 2002
- D. A. Lyon, *Image Processing in Java*, Prentice Hall, 1999
- D. Marr und E. Hildreth, *Theory of Edge Detection*, 1980
- P. Soille, *Morphological Image Analysis: Principles and Applications*, Springer-Verlag, 1998
- Java-Applet auf *http://www-mm.informatik.uni-mannheim.de/veranstaltungen/animation/multimedia/segmentation/Applet/Segmentation.html*

Elementare Bildverarbeitungsoperationen

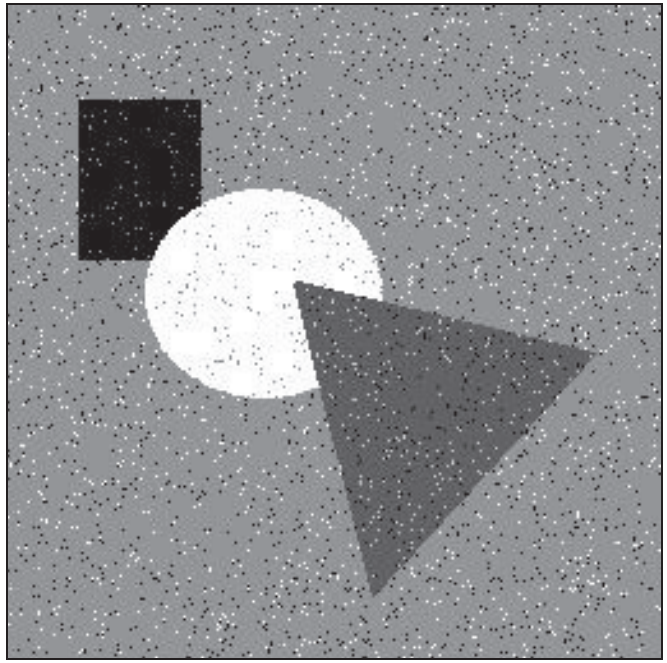
- Implementierung von Rang-Filtern -

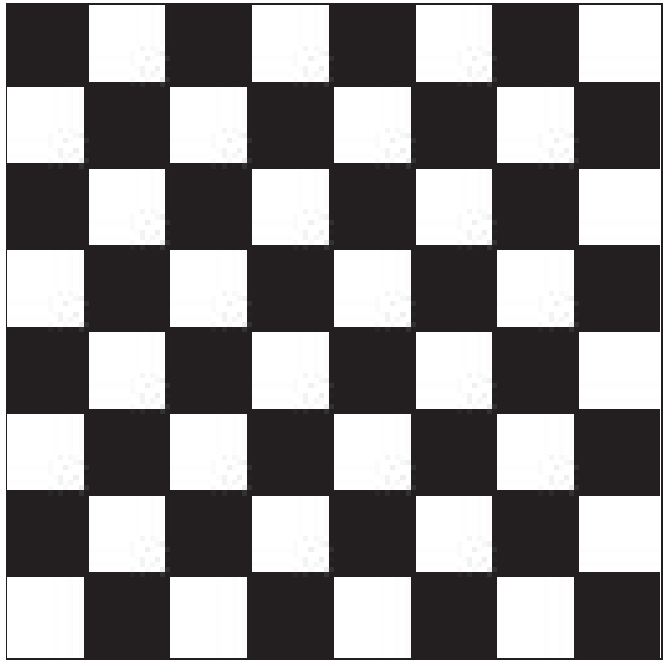
Fabian Bohnert, Dieter Kiesenbauer

1 UML-Diagramm



2 Betrachtete Beispielbilder

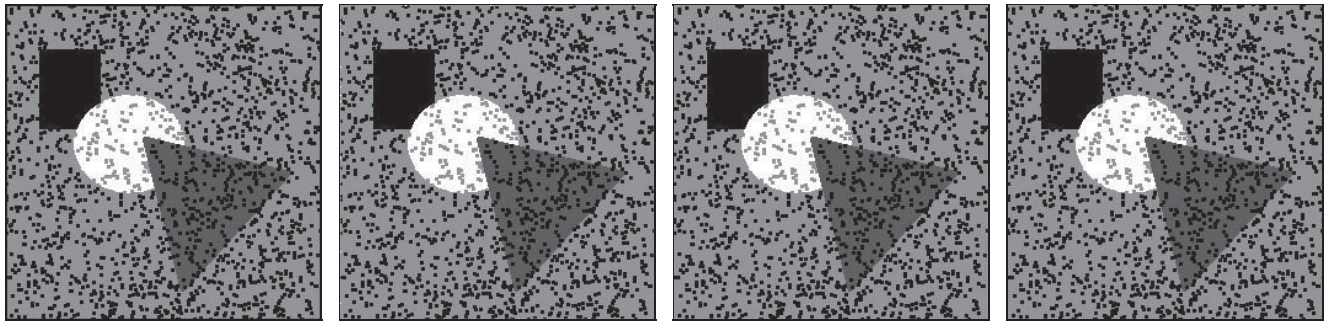




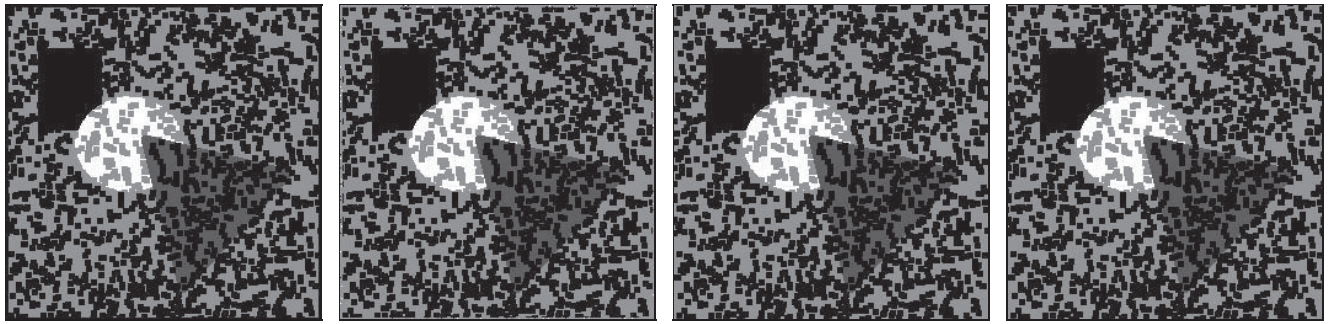
3 Beispiel-Bild 1

3.1 Minimum-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

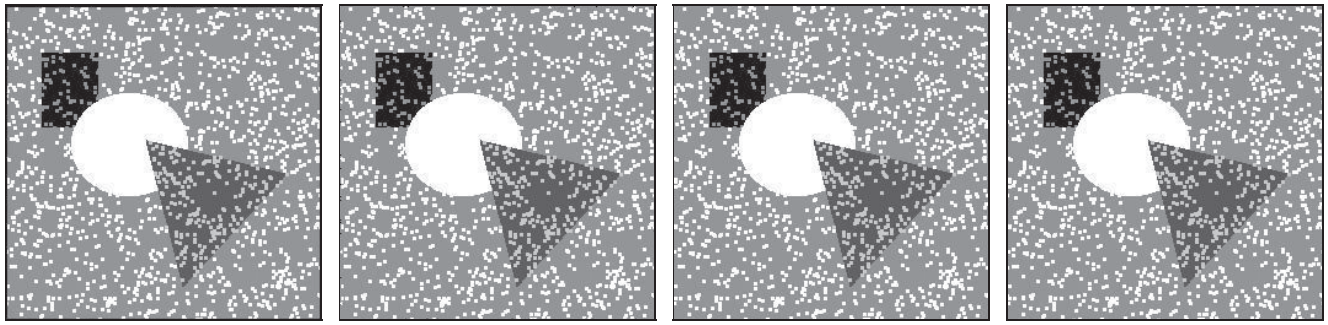


5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

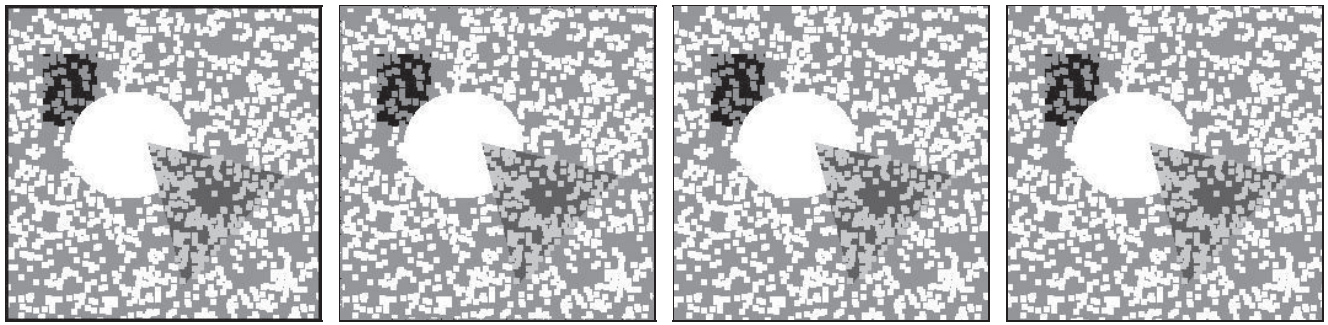


3.2 Maximum-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

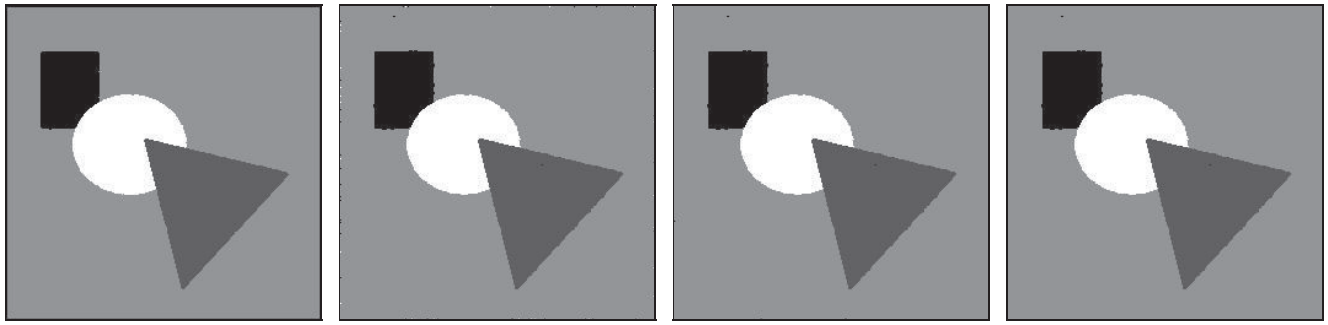


5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

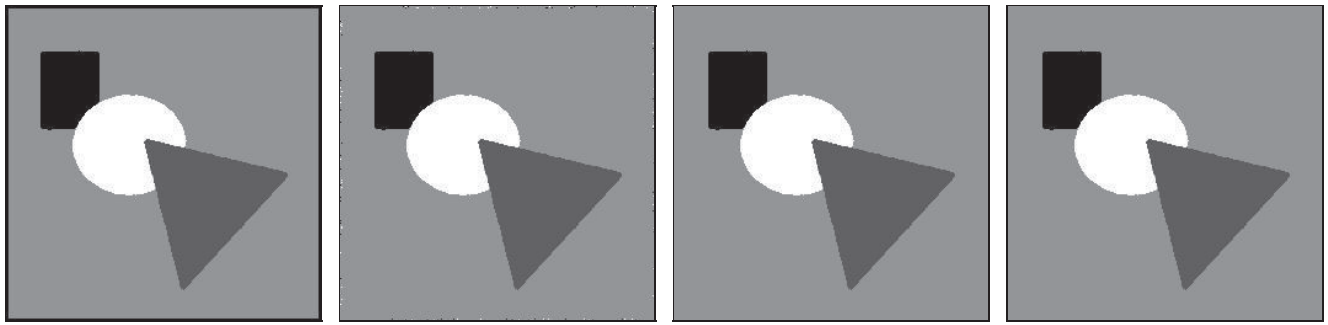


3.3 Median-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)



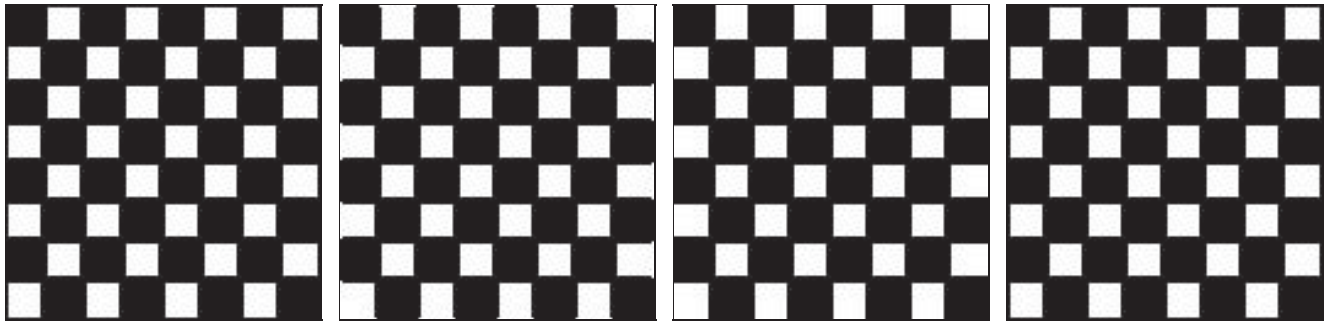
5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)



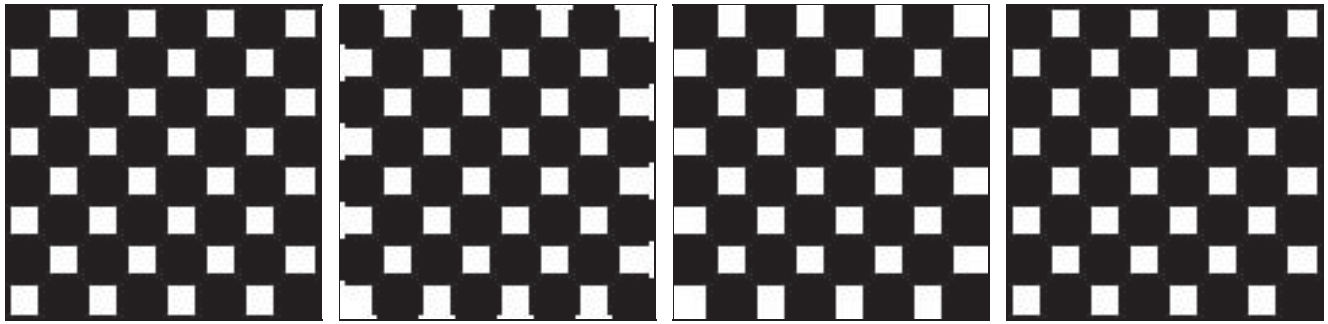
4 Beispiel-Bild 2

4.1 Minimum-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

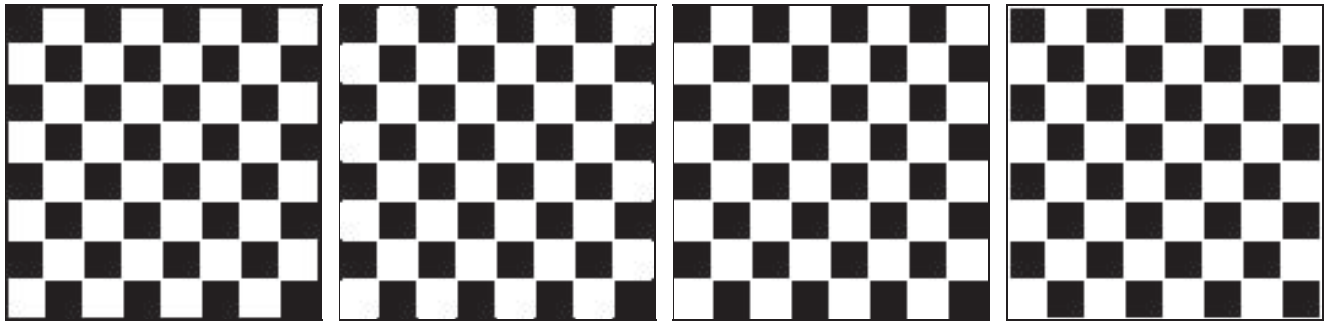


5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

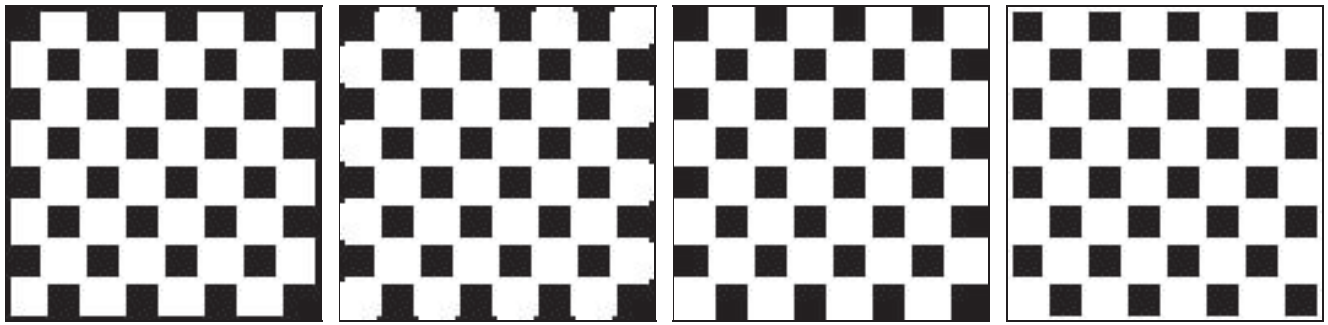


4.2 Maximum-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)

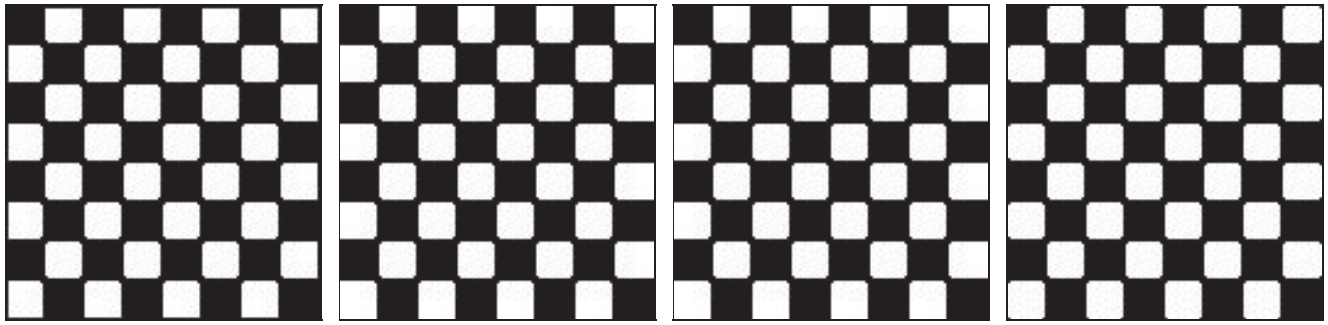


5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)



4.3 Median-Filter

3×3-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)



5×5-Nachbarschaft (NO_BORDER_OP, COPY_BORDER_PIXELS, REFLECTED_INDEXING, CIRCULAR_INDEXING)