Bildverarbeitung in R

Tobias Klinke

Proseminar R
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

Betreuer: Jakob Lüttgau

2016-07-13

Gliederung (Agenda)

1 Einleitung

2 Histogramme

3 Punktoperationen

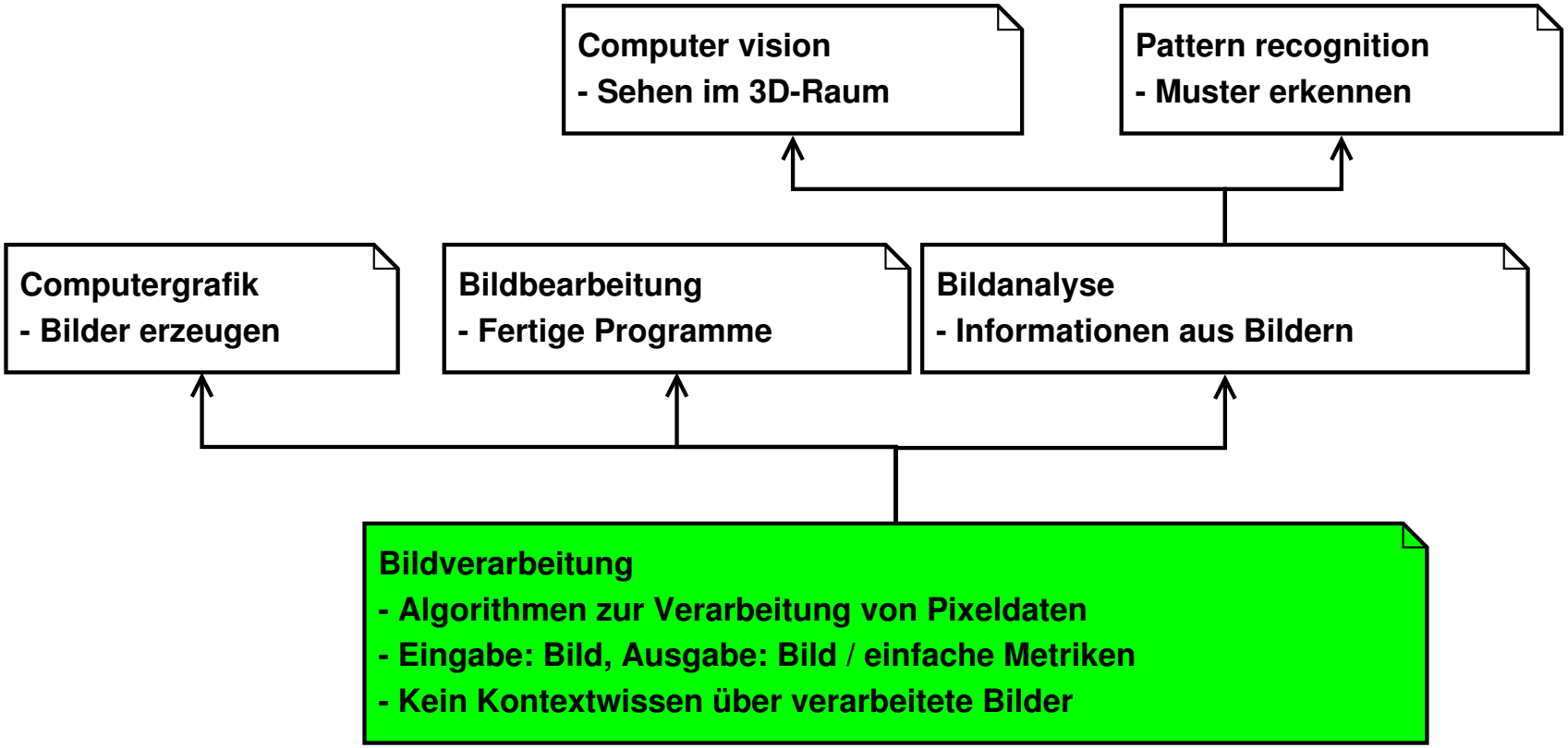
4 Filter

5 Zusammenfassung

6 Literatur

Was ist Bildverarbeitung?

- keine einheitliche Definition, sondern Abgrenzung:



eigene Grafik nach Ausführungen von [Burge, 2006]

Was ist ein Bild?

- zweidimensionale Abbildung: Quelle für (1) [Burge, 2006, S. 10]

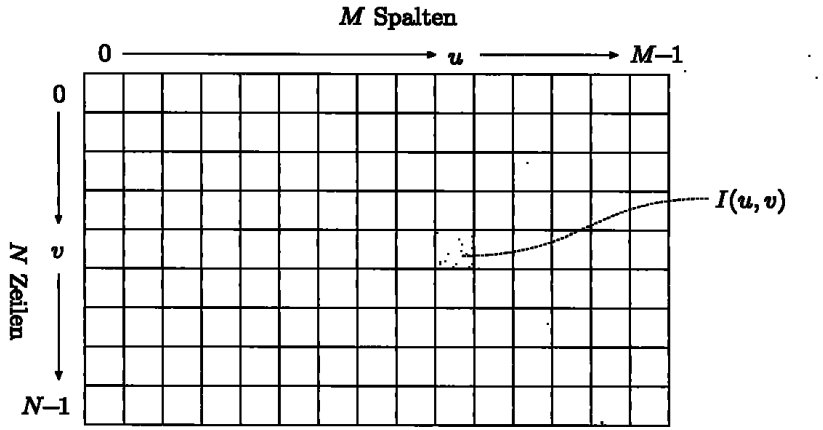
$$I(u, v) \in \mathbb{P} \text{ und } u, v \in \mathbb{N} \tag{1}$$

bzw.

$$I : \mathbb{N} \times \mathbb{N} \to \mathbb{P} \tag{2}$$

$\mathbb{P}$ ist Menge (Intervall) und bestimmt Farbtiefe

- Matrix von Zahlen



[Burge, 2006, S. 12]

Typen von Bildern

- Farbbild: Rot-, Grün- und Blaukomponente je 8 Bit: $\mathbb{P} = [0, 255]^3$
- Graustufenbild: Nur eine Farbkomponente 8 Bit: $\mathbb{P} = [0, 255]$
- Binärbild: Nur ein Bit pro Pixel: $\mathbb{P} = \{0, 1\}$

Farb-, Grau- und Binärbild [1]

[1] Quelle: Lenna-Testbild von `https://en.wikipedia.org/wiki/File:Lenna.png`

Packages zur Bildverarbeitung

- `readbitmap` - Sehr Low-level, nur lesen, Bilddaten als Array
- `jpg`, `png`, `bmp` - Lesen/Schreiben der jeweiligen Formate
- `EBImage` - Standardmethoden der Bildverarbeitung & Analyse, optimiert für Mikroskopie - **wir behandeln nur** `EBImage`
- `adimpro` - Glättung und einige weitere Filter
- (Spezialpakete...)

```
1 # Installation
2 source("https://bioconductor.org/biocLite.R")
3 biocLite("EBImage")
4
5 # Einbinden
6 library(EBImage)
```

Listing 1: Installation und Einbinden von EBImage

Bilder lesen / schreiben / anzeigen

```
> img <- readImage("inputimage.png")
> img
Image
  colorMode    : Color
  storage.mode : double
  dim          : 512 512 3
  frames.total : 3
  frames.render: 1

imageData(object)[1:5,1:6,1]
           [,1]      [,2]      ...
[1,] 0.8862745 0.8862745 ...
[2,] 0.8862745 0.8862745 ...
...
> # Process image ...
> writeImage(img, "outputimage.png") # display(img)
```

Listing 2: Grundgerüst für die weiteren Beispiele

Farbbild in Graustufen konvertieren

- `channel(x, mode)`
- `mode="gray"`
  - R, G, B gleichgewichtet im Durchschnitt
  - $gray = \frac{1}{3} \cdot (R + G + B)$
- `mode="luminance"`
  - R, G, B nach Wahrnehmung gewichtet
  - $gray = 0.2126 * R + 0.7152 * G + 0.0722 * B$ [2]

links: `gray`, rechts: `luminance` [3]

[2] [EBImage, 2016]

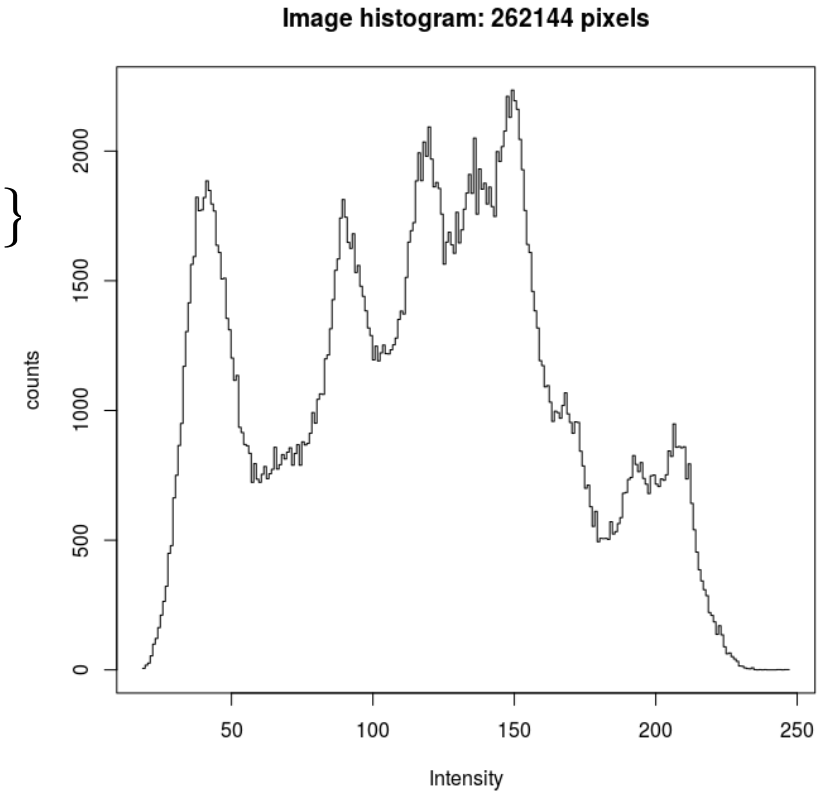
[3] Quelle: Farbbild Lenna-Testbild von `https://en.wikipedia.org/wiki/File:Lenna.png`

Histogramm: Definition

- Formel[a]

  $h(i) = \text{card}\{(u, v)|I(u, v) = i\}$

  für alle $0 \leq i < K$ mit $K = \text{card}\mathbb{P}$
- Häufigkeitsverteilung der einzelnen Intensitätswerte

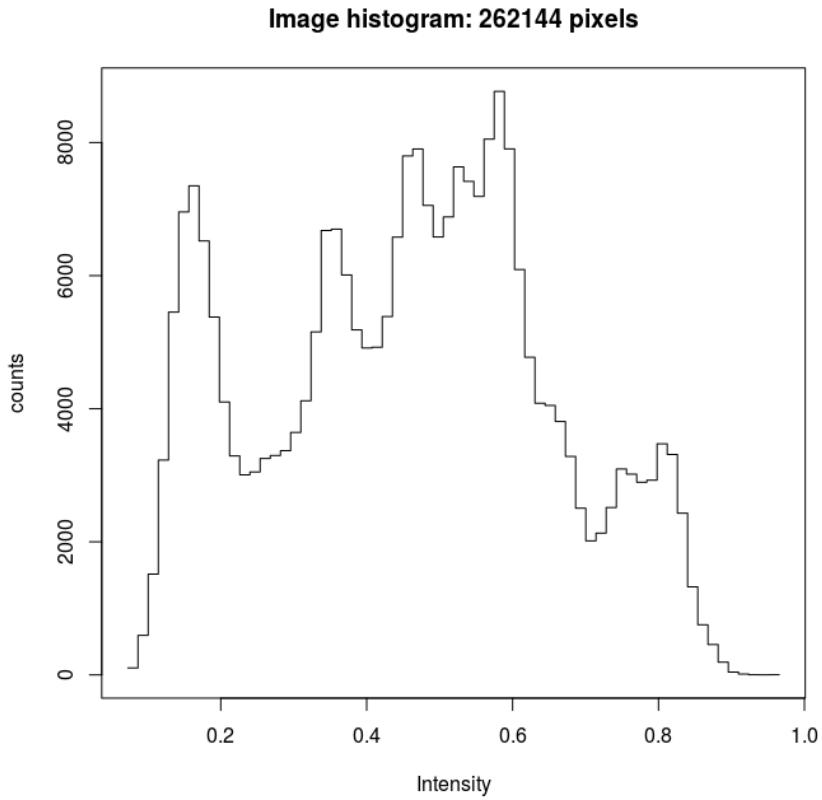


[a] nach [Burge, 2006, S. 40]

Histogramme in R anzeigen

```
1 hist(img)
```

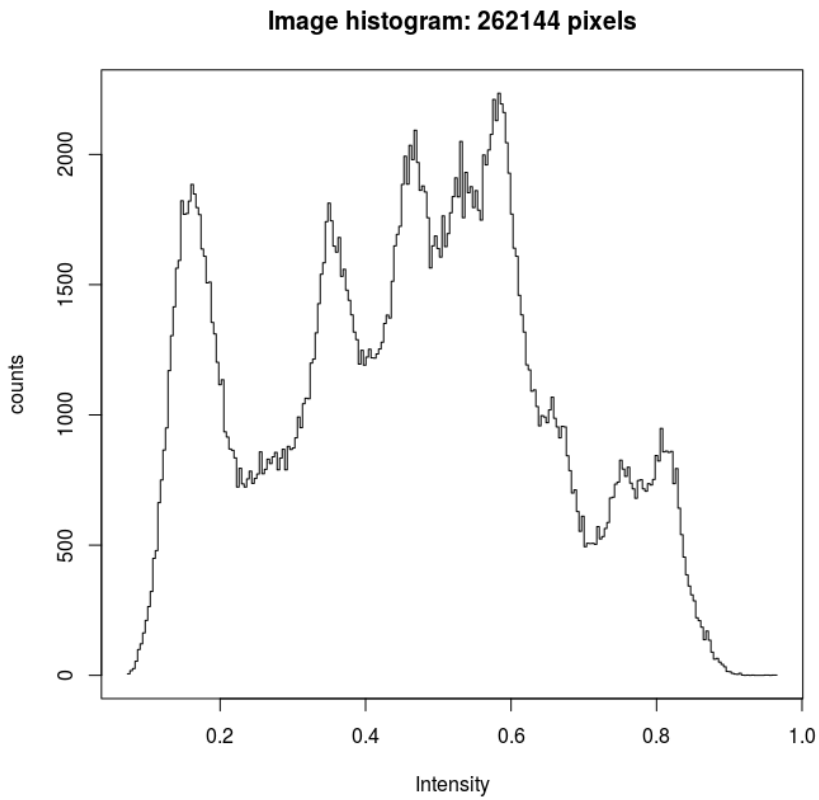
Listing 3: Histogramm anzeigen



Histogramm mit feinerer Einteilung

```
1 hist(img, breaks=256)
```

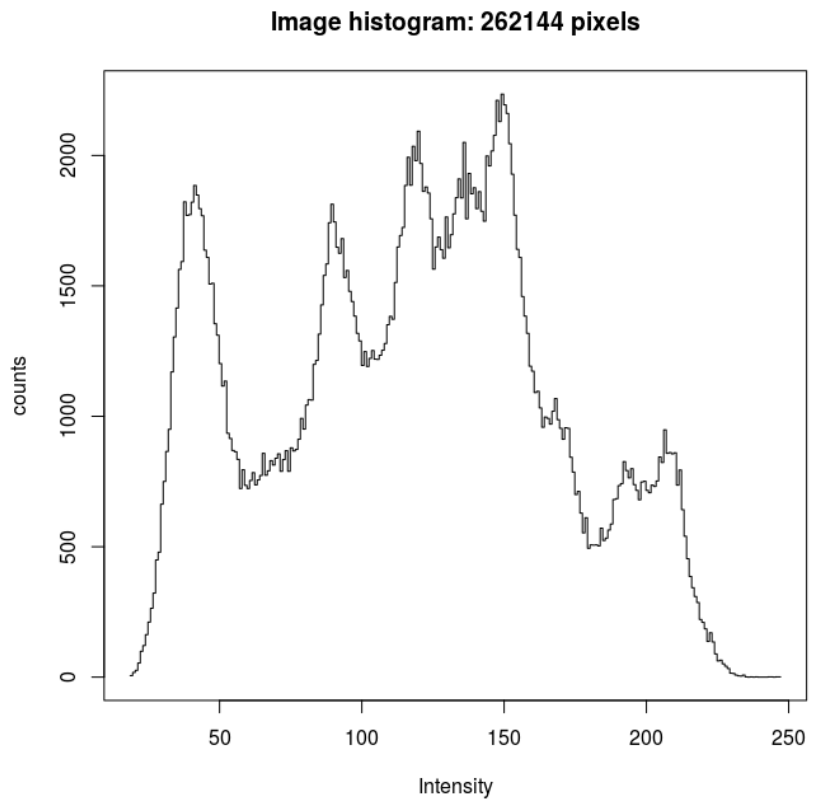
Listing 4: Histogramm mit feinerer Achseneinteilung



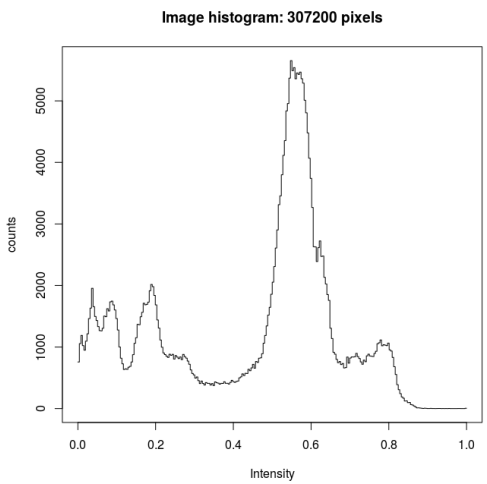
Histogramm Skalierung

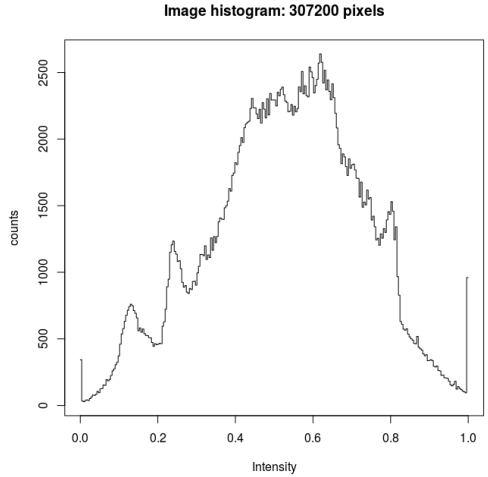
```
1 hist(img * 256, breaks=256)
```

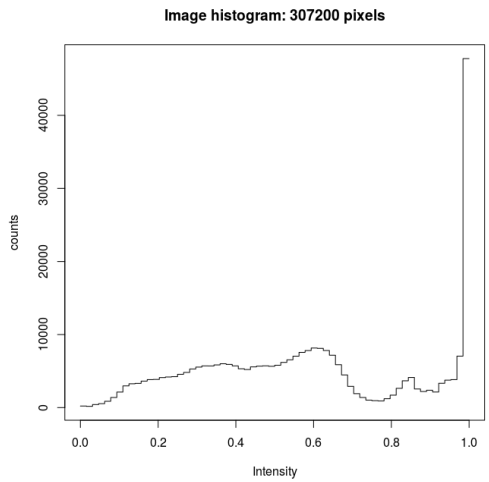
Listing 5: Histogramm mit x-Achse von 0 bis 255



Belichtung im Histogramm ablesen







Unterbelichtetes, normal belichtetes und überbelichtetes Bild[4]

[4] Quelle der Bilder: [Burge, 2006, S. 42]

Kontrastanpassung

- Kontrast: Differenz zwischen minimalem / maximalem vorkommenden Grauwert
- Kontrastanpassung Die einzelnen Grauwerte a werden wie folgt abgebildet[5]:

$$f_{ac}(a) = (a - a_{low}) \cdot \frac{a_{max} - a_{min}}{a_{high} - a_{low}}$$

$a_{min/max}$: minimal / maximal *mögliche* Grauwerte
$a_{low/high}$: minimal / maximal *vorkommende* Grauwerte

```
1 equimg <- equalize(img)
```

Listing 6: Kontrastanpassung mit EBImage

[5] Formel aus [Burge, 2006, S. 59]

Vorher / nacher

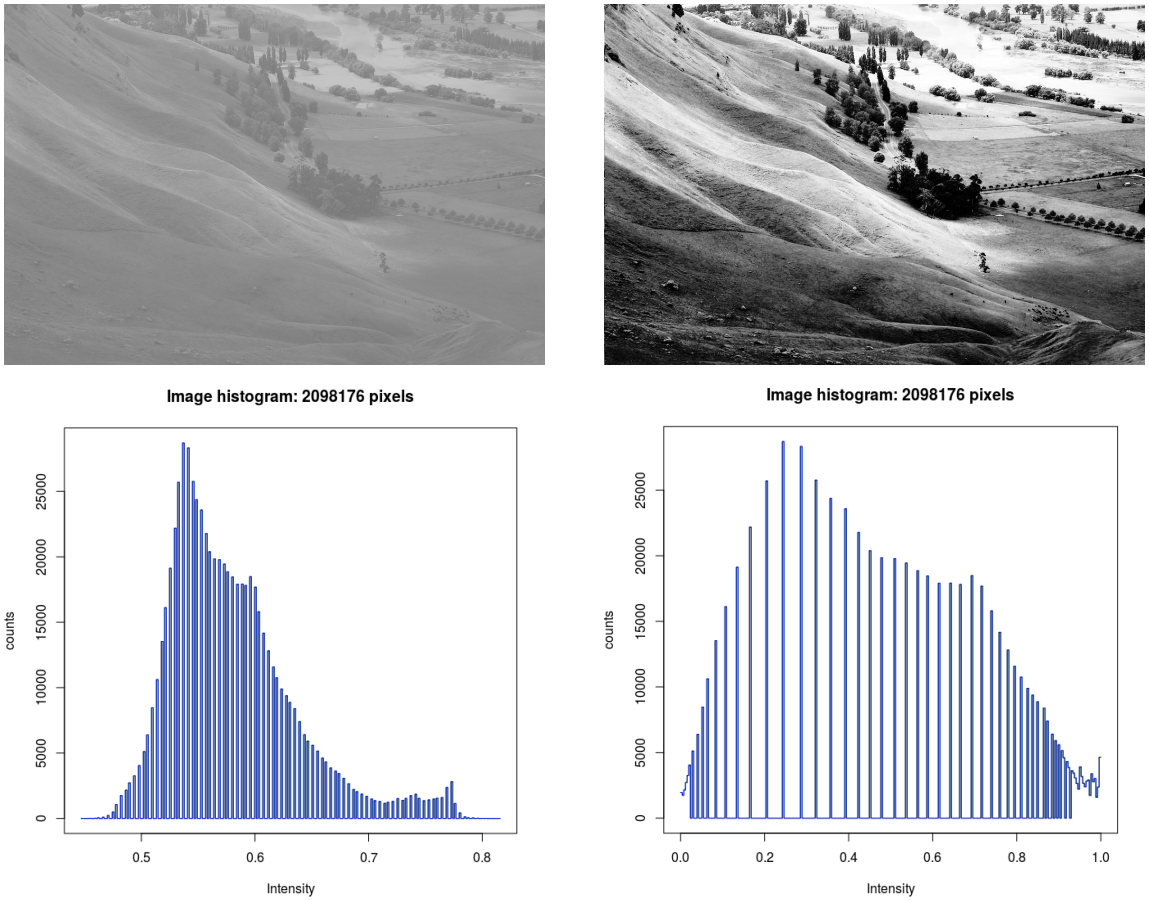


Bild und Histogramm vor und nach Kontrastanpassung[6]

[6] Quelle Originalbild: https://en.wikipedia.org/wiki/File:Unequalized_Hawkes_Bay_NZ.jpg

Grundlagen

$$I'(u, v) = f(I(u, v))$$

wobei $f : \mathbb{P} \to \mathbb{P}$ ist. [7]

- Jeder Pixel wird auf die gleiche Weise transformiert
- Unabhängig von Koordinaten
- in R mit `EBImage`
  - Rechnen wie mit Arrays z.B. `img + 0.5`
  - Kein automatisches Clamping
  - Clamping erst beim Speichern / Anzeigen

[7] Quelle Formel: [Burge, 2006, S. 55]

Invertieren

- $f(a) = a_{max} - a$ [8]
- Vertikale Spiegelung des Histogramms in der Mitte

```
1 img_inverted <- 1.0 - img
```

Listing 7: Invertieren eines Bildes

Bild und Negativ[9]

[8] Quelle Formel: [Burge, 2006, S. 57]

[9] Quelle: Lenna-Testbild von https://en.wikipedia.org/wiki/File:Lenna.png

Aufhellen / Verdunkeln / Kontrast

- Aufhellen = Addition von Konstante
  - $f(a) = a + c$ mit $c > 0$
- Verdunkeln = Subtraktion von Konstante
  - $f(a) = a - c$ mit $c > 0$
- Kontrast ändern = Multiplikation mit Konstante
  - $f(a) = a \cdot c$ mit $c > 1$ zum Erhöhen, $0 < c < 1$ verringern

`img + 0.5`

`img - 0.4`

`img * 1.5`

[10]

[10] Quelle: Lenna-Testbild von `https://en.wikipedia.org/wiki/File:Lenna.png`

Thresholding

$$f_{th}(a) = a_{min} \quad \text{für } a < a_{thr}$$
$$f_{th}(a) = a_{max} \quad \text{für } a \geq a_{thr}$$

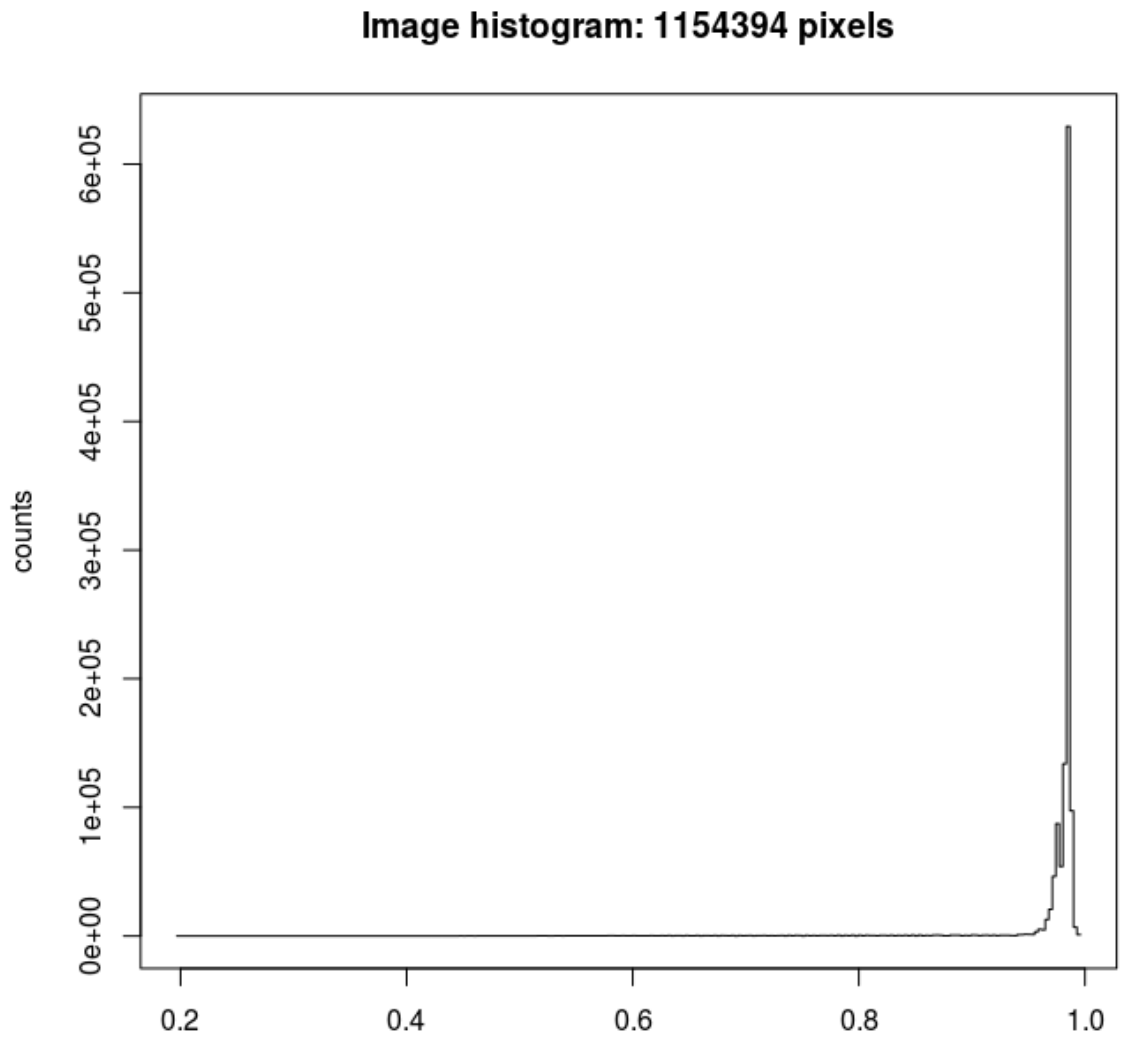
Formel nach [Burge, 2006, S. 57]

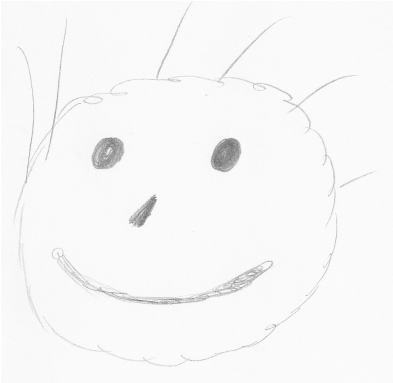
- Umwandlung Grau- nach Binärbild
- Alle Werte bis zu einem Schwellenwert werden schwarz
- Alle Werte ab einem Schwellenwert werden weiß
- Ergebnis stark abhängig von der Wahl des Schwellenwertes

```
1 thresholded_image <- img > 0.5
2 # alle Werte > 0.5 werden zu weiss, alle anderen zu
    ↪ schwarz
```

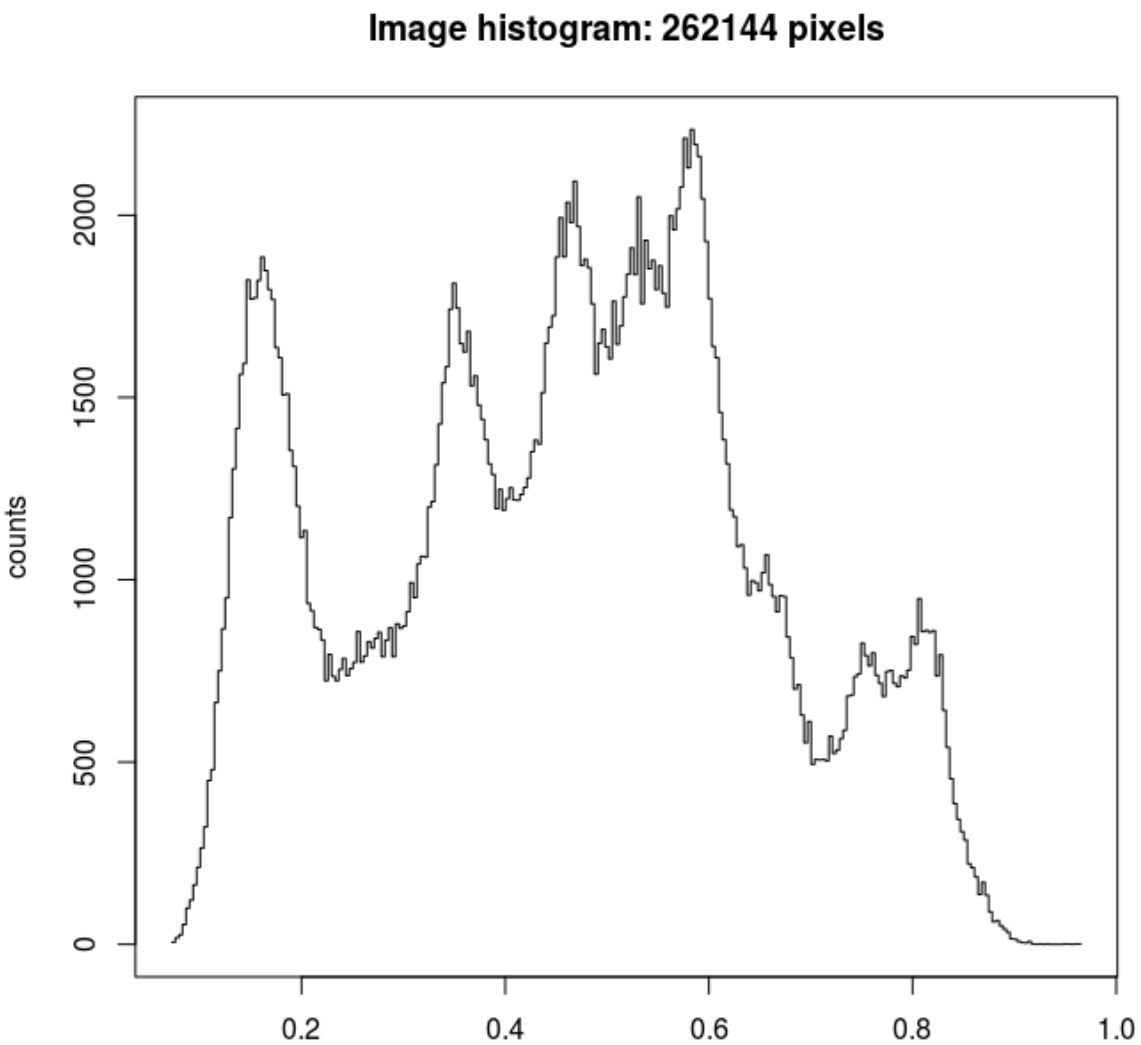
Listing 8: Thresholding mit Schwellenwert von 0.5

Wahl des Schwellenwerts





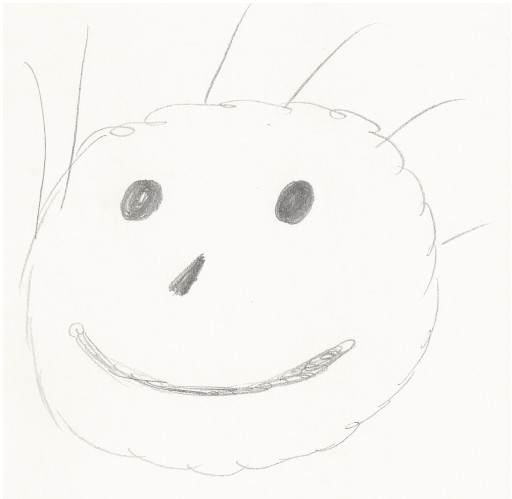
Wahl des Schwellenwerts

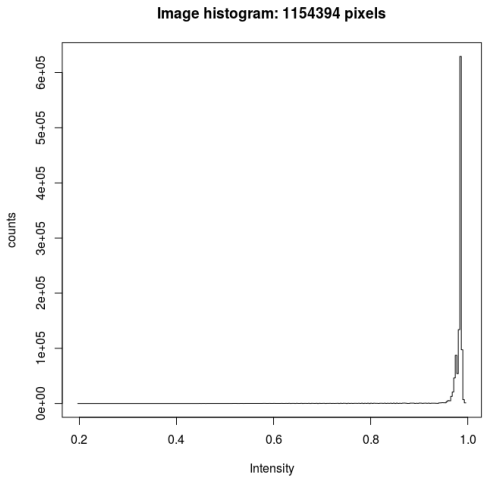


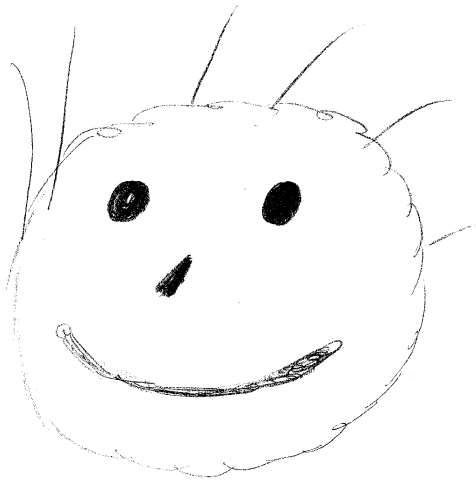
Wahl des Schwellenwerts

- Schätzen im Histogramm:
  - in der Mitte im „Tal“ zwischen zwei Spitzen
  - am „Fuß“ eines Berges
- Berechnung des Optimums (nach Otsu) [wikipedia, 2016]
  - Aufteilung in zwei Klassen
  - geringste Varianz innerhalb der Klassen
  - maximale Varianz zwischen den Klassen

Beispiel







```
1 t <- img > otsu(img)
```

Graubild mit Histogramm und Binärbild nach Thresholding mit $a = otsu(img) \approx 0.82$

Weitere Punktoperationen

- Verknüpfen von Bildern
  - $I'(u,v) = f(I_1(u,v), I_2(u,v), ..., I_n(u,v))$ [11]
  - Beispiel: Alpha-Blending mit zwei Bildern:

```
img <- a * img1 + (1.0 - a)* img2
```

- Farbraumkonvertierungen (z.B. Farbe nach Grau)
- Kontrastanpassung mittels Histogramm

[11] Formel nach [Burge, 2006, S. 83]

Definition linearer Filter

- Filterregion: $R \subseteq \mathbb{Z} \times \mathbb{Z}$
- Filterfunktion $H : R \to \mathbb{R}$
- $I'(u, v) = \sum_{(i,j) \in R} I(u + i, v + j) \cdot H(i, j)$
- Darstellung des Filters als Matrix:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Einfacher Box-Glättungsfilter (Ursprung)

Formeln nach [Burge, 2006, S. 92f]

Berechnung der Anwendung eines linearen Filters

```
1 filter(img, filtermatrix):
2 erstelle kopie von img in img2
3 fuer jeden Pixel:
4     Setze Ursprung der Filtermatrix auf diesen Pixel
5     Multipliziere die umgebenden Pixel mit den
      ↪ Gewichten aus der Matrix
6     Summiere alle gewichteten Pixelwerte
7     Schreibe Ergebnis an dieselbe Stelle in img2
8 gib img2 aus
```

Listing 9: Algorithmus zur Anwendung eines Filters

Box-Filter zur Glättung

Box-Filter:

- Matrix:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

- Eigenschaften:
  - Bildet den Durchschnitt der benachbarten Pixel
  - erzeugt einfachen „Weichzeichnen“-Effekt
  - Unterdrückt Bildrauschen

lineare Filter in `EBImage`

- `makeBrush(size, shape, [...])`
  - Erstellt Filtermatrix für gängige Filter
  - `size` Größe des Filters in Pixeln
  - `shape` Filter-Form: z.B. „box“ oder „Gaussian“
- `filter2(img, filter, [...])`
  - Wendet eine Filter-Matrix auf ein Bild an
  - `img` Das zu filternde Bild
  - `filter` Der Filter (von `makeBrush()`)

```
1 # box braucht manuelle Skalierung
2 brush <- makeBrush(5, "box") * (1/25)
3 smooth <- filter2(img, brush)
```

Listing 10: Box-Filter auf Bild anwenden

Vor / nach Box-Filter

Links: vor, rechts: nach Box-Filter mit 5x5 Pixeln[12]

[12] Quelle Original: Lenna-Testbild von `https://en.wikipedia.org/wiki/File:Lenna.png`

Weitere lineare Filter

- Gauß-Filter als anderer Glättungsfilter

```
1 brush <- makeBrush(5, "Gaussian")
```

Listing 11: Anwendung des Gauß-Filters

- Laplace-Filter zur Kantenerkennung/-verstärkung

Definition

- Anwendung auf Binärbilder: $I(u, v) \in \{0, 1\}$
- Strukturelement (= Filter): $H(i, j) \in \{0, 1\}$
- Bilder in Mengenschreibweise: Alle Koordinaten-Paare der Vordergrund (weißen) Pixel: $Q_I = \{(u, v) | I(u, v) = 1\}$
  - Oftmals ist aber Schwarz auf Weiß, daher Bild negieren oder jeweils duale Operation (Erosion statt Dilation, Opening statt Closing) wählen! Bei den folgenden Beispielen ist jeweils Schwarz der Vordergrund!
- Verändern im Gegensatz zu linearen Filtern die *Struktur* des Bildinhalts

Formeln nach [Burge, 2006, S. 174f]

Erosion

- $I \ominus H = \{(u', v') | \forall (i, j) \in Q_H((u' + i, v' + j) \in Q_I)\}$

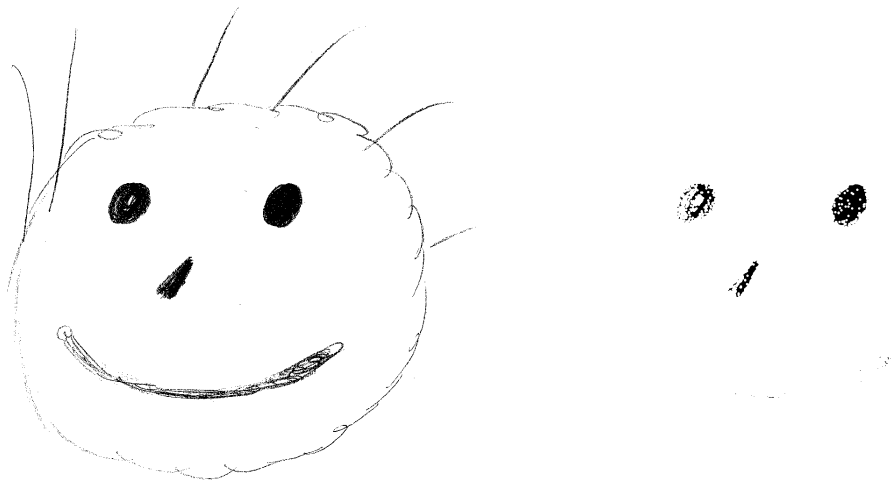
  [Burge, 2006, S. 176]

```
1  Fuer jeden Pixel (u, v) in I:
2      positioniere Ursprung von H ueber (u, v)
3      Wenn unter jedem Vordergrundpixel von H ein
           ↪ Vordergrundpixel von I liegt:
4          I'(u, v) = Vordergrundpixel
5      Sonst:
6          I'(u, v) = Hintergrundpixel
```

- Lässt Strukturen schrumpfen

Beispiel: Erosion

```
1 img <- 1.0 - img # Schwarz ist Vordergrund
2 struct <- makeBrush(5, "disc") # 5x5 Kreis
3 eroded <- erode(img, struct)
4 eroded <- 1.0 - eroded
```



links: Original, rechts: mit 5x5 Kreis erodiert (schwarz = Vordergrund)

Dilation

- $I \oplus H = \{(u', v') = (u + i, v + j) | (u', v') \in Q_I, (i, j) \in Q_H\}$

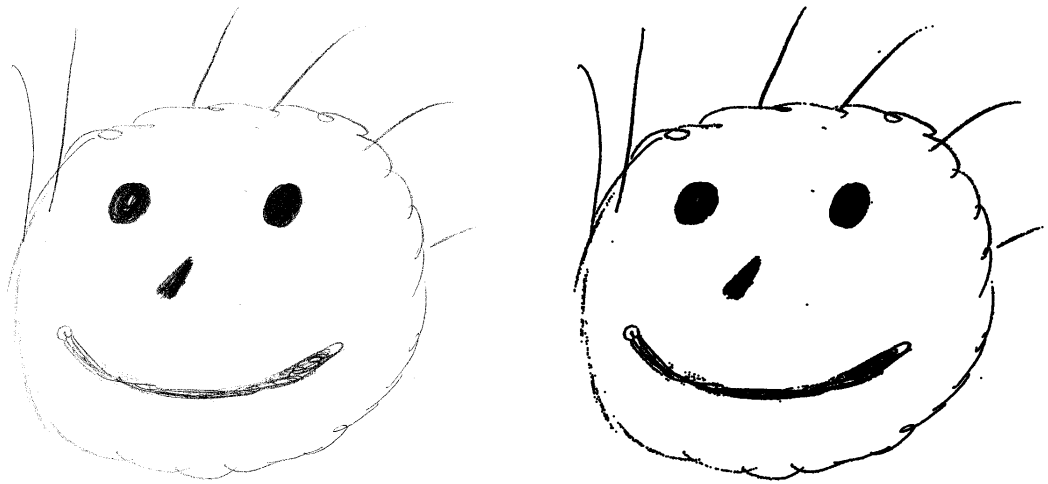
  [Burge, 2006, S. 175]

```
1 Fuer jeden Vordergrund-Pixel (u, v) in I:
2     positioniere Ursprung von H ueber (u, v)
3     Fuer alle Vordergrundpixel von H:
4         Mache darunterliegende Pixel von I zu
              ↪ Vordergrundpixeln
```

- Lässt Strukturen wachsen

Beispiel: Dilation

```
1 img <- 1.0 - img # Schwarz ist Vordergrund
2 struct <- makeBrush(5, "disc") # 5x5 Kreis
3 dilated <- dilate(img, struct)
4 dilated <- 1.0 - dilated
```



links: Original, rechts: Dilation mit 5x5 Kreis (schwarz = Vordergrund)

Opening / Closing

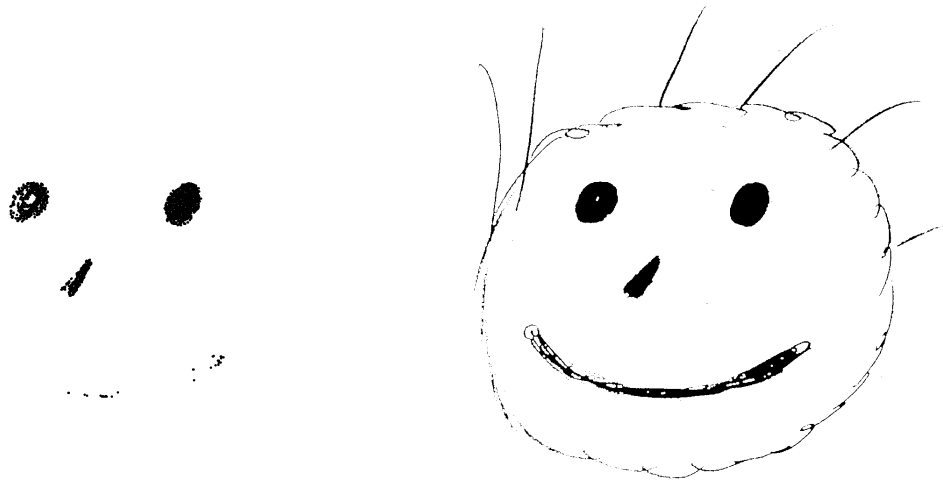
- Opening
  - $I \circ H = (I \ominus H) \oplus H$

    [Burge, 2006, S. 179]
  - Erosion gefolgt von Dilation
  - Lässt kleine Strukturen verschwinden
  - Funktion `opening(img, filter)`
- Closing
  - $I \cdot H = (I \oplus H) \ominus H$

    [Burge, 2006, S. 182]
  - Dilation gefolgt von Erosion
  - Schließt Lücken in Strukturen
  - Funktion `closing(img, filter)`

Beispiel: Opening / Closing

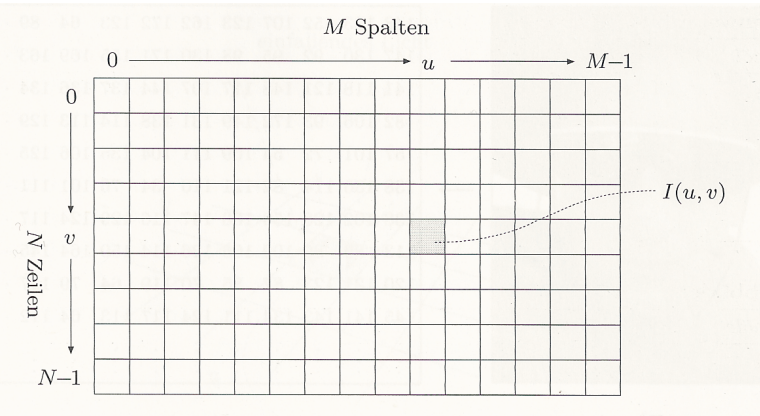
```
1 img <- 1.0 - img # Schwarz ist Vordergrund
2 struct <- makeBrush(5, "disc") # 5x5 Kreis
3 opened <- opening(img, struct) # bzw. closing(img,
      ↪ struct)
4 opened <- 1.0 - opened
```

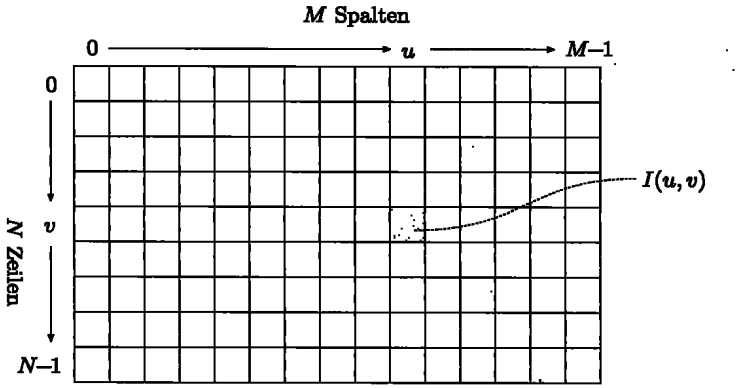


links: Opening, rechts: Closing je mit 5x5 Kreis (schwarz = Vordergrund)

Praxisbeispiel: Gescannte Abbildungen verschönern

```
1 scan <- readImage("scan.png")
2 grey <- channel(scan, "luminance")
3 binary <- grey > otsu(grey)
4 inv <- 1.0 - binary
5 brush <- makeBrush(3, "disc")
6 inv_dilated <- dilate(inv, brush)
7 dilated <- 1.0 - inv_dilated
8 display(dilated)
```





links: Original-Scan [Burge, 2006, S. 12] , rechts: Bearbeitetes Bild

Zusammenfassung

- Bilder sind im Prinzip 2D-Arrays
- Histogramme anzeigen mit `hist(img)`
- Punktoperationen
  - neuer Wert des Pixels nur vom alten Wert abhängig
  - In R rechnen wie mit Arrays
  - z.B. Invertieren, Heller/Dunkler/Kontrast, Thresholding
- lineare Filter
  - neuer Wert auch von umgebenden Pixeln abhängig
  - vor allem Glättungsfilter
  - `filter2()`, `makeBrush`
- morphologische Filter
  - Strukturelemente als Masken
  - Strukturen in Binärbildern wachsen / schrumpfen lassen
  - `erode()`, `dilate()`, `opening()`, `closing()`, `makeBrush()`

Literatur

[Burge, 2006] Burge, W. B. . M. J. (2006). *Digitale Bildverarbeitung : eine Einführung mit Java und ImageJ*. Springer, Berlin, 2., überarb. edition.

[EBImage, 2016] EBImage (2016). EBImage documentation `http://www.bioconductor.org/packages/release/bioc/manuals/EBImage/man/EBImage.pdf`, 26.06.2016.

[Parker, 2011] Parker, J. (2011). *Algorithms for Image Processing and Computer Vision*. Wiley Publishing, Indianapolis, 2nd edition.

[wikipedia, 2016] wikipedia (2016). Otsu's method `https://en.wikipedia.org/wiki/Otsu%27s_method`, 26.06.2016 .