

# Einführung in das 4. Aufgabenblatt: Bildverarbeitung und Klassifikation

**Thomas Deselaers, Oliver Bender**

`{deselaers,bender}@i6.informatik.rwth-aachen.de`

**Praktikum im Grundstudium SS 2006 – 18. Mai 2006**

**Human Language Technology and Pattern Recognition**

**Lehrstuhl für Informatik 6**

**Computer Science Department**

**RWTH Aachen University, Germany**

# Aufgabe 0: IO-Routinen

## Definitionen:

Ein Grauwert-Bild wird als Abbildung interpretiert:

$$g : \mathcal{I} \times \mathcal{J} \longrightarrow \mathcal{G}$$

$$(i, j) \longmapsto g(i, j) = g_{ij}$$

**hier:**  $g_{ij} \in \{0, \dots, 255\} = \mathcal{G}$

**.pgm-Format (ASCII) für Grauwertbilder:** ( $I = |\mathcal{I}|$ ,  $J = |\mathcal{J}|$ ,  $G = |\mathcal{G}|$ )

```
P2
# Kommentar
J I
G
g11 g12 ... g1J
g21 g22 ... g2J
.
.
.
gI1 gI2 ... gIJ
```

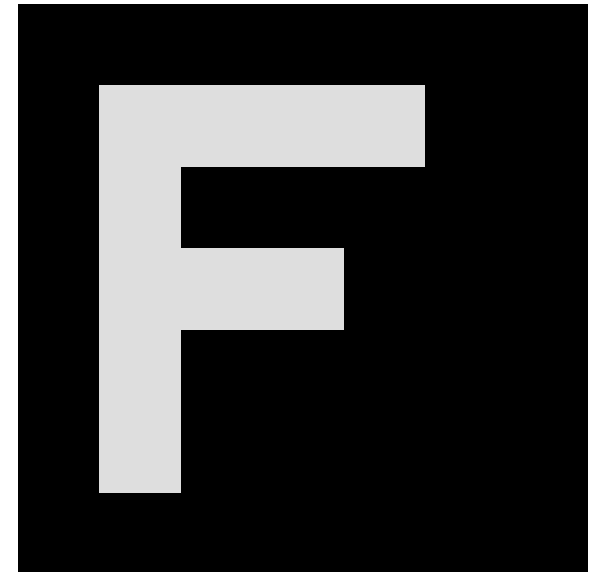
**Der Zeilenumbruch muss nicht nach  $J$  Einträgen erfolgen.**

**Anmerkung: “Output-Routine” bedeutet nur die Ausgabe von .pgm Dateien in ASCII. Sie sollen keinen Image-Viewer selbst programmieren! Benutzen Sie etwa xv zum Anschauen der Bilder**

# PGM-Beispiel

```

P2
7 7
255
0 0 0 0 0 0 0
0 222 222 222 222 0 0
0 222 0 0 0 0 0
0 222 222 222 0 0 0
0 222 0 0 0 0 0
0 222 0 0 0 0 0
0 0 0 0 0 0 0
    
```

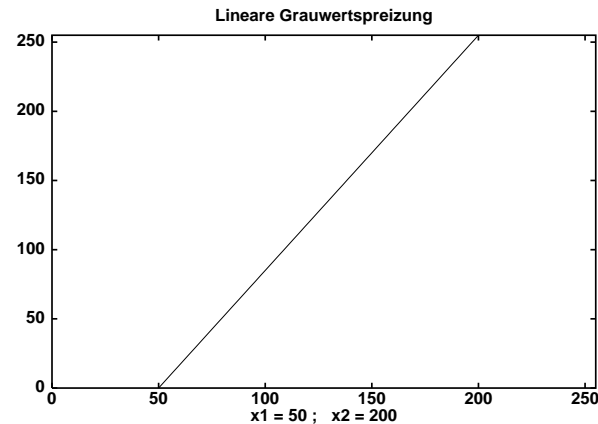


# Aufgabe 1: Punktoperatoren

## Lokale Modifikation der Grauwerte oder des Grauwertbereichs

$$\hat{g}_{ij} = h(g_{ij})$$

Zu a): lineare Grauwertspreizung:



MIN-MAX-Spreizung — Ausnutzung des vollen Grauwertbereiches:

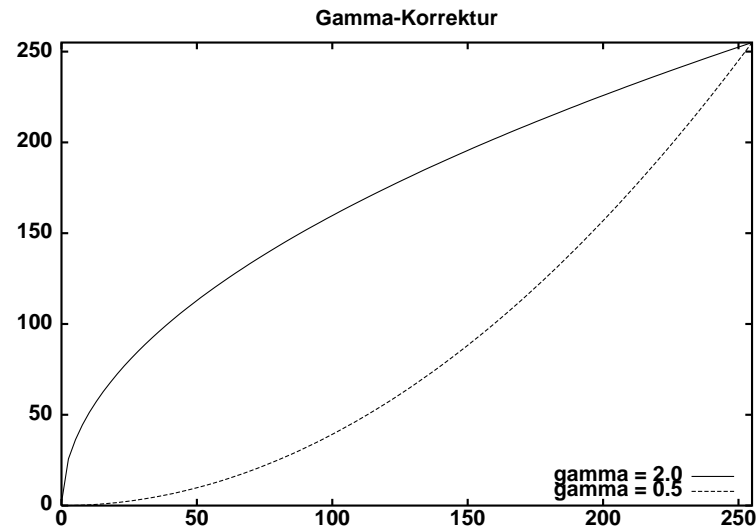
$$x_1 = \min_{ij} g_{ij}, \quad x_2 = \max_{ij} g_{ij}$$

Anwendung:

- Bildverbesserung
- Normalisierung (bei der Klassifikation)
- Elimination einfacher Beleuchtungsmodelle
- ...

# Aufgabe 1: Punktoperatoren

Zu b): Gamma-Korrektur:



**Besonderheit der Gamma-Korrektur:**

- Nichtlineare Grauwertmodifikation
- Einerseits: “logarithmische Kennlinie” des Auges
- Außerdem: nichtlineare Verzerrung von Monitoren (oft in Monitor “implementiert”,  $\gamma$  etwa 1.5 - 1.9)

## Aufgabe 2: Lineare Filter

$\hat{g} = f(g)$ , wobei  $f$  lineare Funktion.

**Anschaulich meist: Modifikation der Grauwerte in Abhängigkeit von der Umgebung des Pixels  $(i, j)$ .**

- **Bildverbesserung**
- **Normalisierung (insbes. Klassifikation)**
- **Tiefpassfilter: Rauschunterdrückung**
- **Hochpassfilter: Kantendetektion**
- ...

# Lineare Filter

## Vorgehensweise:

- **Lege Template mit “Aufpunkt” auf aktuelles Pixel  $(i, j)$ , führe Operationen aus, schreibe Ergebnis in  $(i, j)$  zurück. Dabei keine Werte überschreiben, die noch unmodifiziert benötigt werden!**
- **Sonderbehandlung des Randbereichs:**
  - Ignoriere Ränder (hier in Aufgabe 2)
  - Wrap–Around
  - Außerhalb des Bildes mit festem Wert auffüllen (Aufgabe 3)

## Sonderhinweis (für besonders interessierte Studierende):

Symmetrische lineare Filter (wie wir sie hier betrachten) können beschleunigt angewendet werden, wenn man sie in einen horizontalen und einen vertikalen filter zerlegt.

## Aufgabe 2: Gauß-Filter

Approximation des Gauß-Filters der Ordnung  $n$  über Binomialkoeffizienten:  
Pascal'sches Dreieck:

```
n=0:      1
n=1:     1 1
n=2:    1 2 1
n=3:    1 3 3 1
n=4:    1 4 6 4 1
...

```

Beispiel: Gaußfilter für  $n = 2$ :

$$\frac{1}{4}(1, 2, 1)^T \cdot \frac{1}{4}(1, 2, 1) = \frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$



Empfehlung:

**Aufgabe 1 und 2 zusammen in ein Kommandozeilenprogramm:**

```
filter_program filter_type parameters infile outfile
```

**Bei Aufruf ohne irgendwelche Parameter  $\Rightarrow$  Erklärung ausgegeben!**

**Bei falschen Filtertypen, Parametern oder Dateiformaten  $\Rightarrow$  sinnvolle Fehlermeldungen!**

# Aufgabe 3: Einfacher Klassifikator

**Nearest Neighbor-Klassifikator:**

**Klassifikation von handgeschriebenen Ziffern ( $16 \times 16$  Pixel) aus dem US-Postal Service Datensatz.**

**Vorgehensweise:**

- **Betrachte das zu klassifizierende Bild als Vektor:**

$$X = (x_1, \dots, x_D) = (g_{11}, g_{12}, \dots, g_{IJ}) \in \mathbb{R}^D, \quad D = I \cdot J$$

- **Berechne den euklidischen Abstand zu allen Trainingsdaten  $Y$ :**

$$d(X, Y) = \sum_{d=1}^D \frac{(x_d - y_d)^2}{\sigma_d^2}$$

**Hier mit  $\sigma_d = 1$ . Besser: geeignetes  $\sigma_d$  errechnen (freiwillig).**

- **Die Klasse des Trainingsvektors mit minimalem Abstand wird als Klasse des Testvektors  $X$  ausgegeben.**

## Aufgabe 3: Anmerkungen

- **Bei allen Aufgabenteilen die Fehlerrate =  $\frac{\text{\#Fehler}}{\text{\#Tests}}$  in % angeben. Hinweis: Bei Verwendung aller Test- und Trainingsdaten sollte die Fehlerrate etwa 5-6% sein.**
- **Zum Filtern in Teil b) die Umgebung des Bildes (gedanklich) mit Nullen auffüllen.**

## Aufgabe 3: Einfacher Klassifikator

### Empfehlung: Kommandozeilen Programm

```
recog_numbers filt_type params training_data test_data log_file
```

**Das Logfile sollte Angaben über den benutzten Filter, Trainings- und Testdateien und die Fehlerrate in % enthalten. Dabei soll auch die Entscheidung des Klassifikators für jedes Testbild ausgegeben werden. Optional zusätzlich eine Verwechslungsmatrix der Ziffern.**

**Die Parameter, die die Bilddaten betreffen, können dabei Verzeichnisnamen oder Dateien mit den Dateinamen der Bilder sein.**

**Die Bilddaten liegen als .pgm Dateien vor, bezeichnet mit <Klasse>\_<Nummer>.pgm.**

### Hinweise:

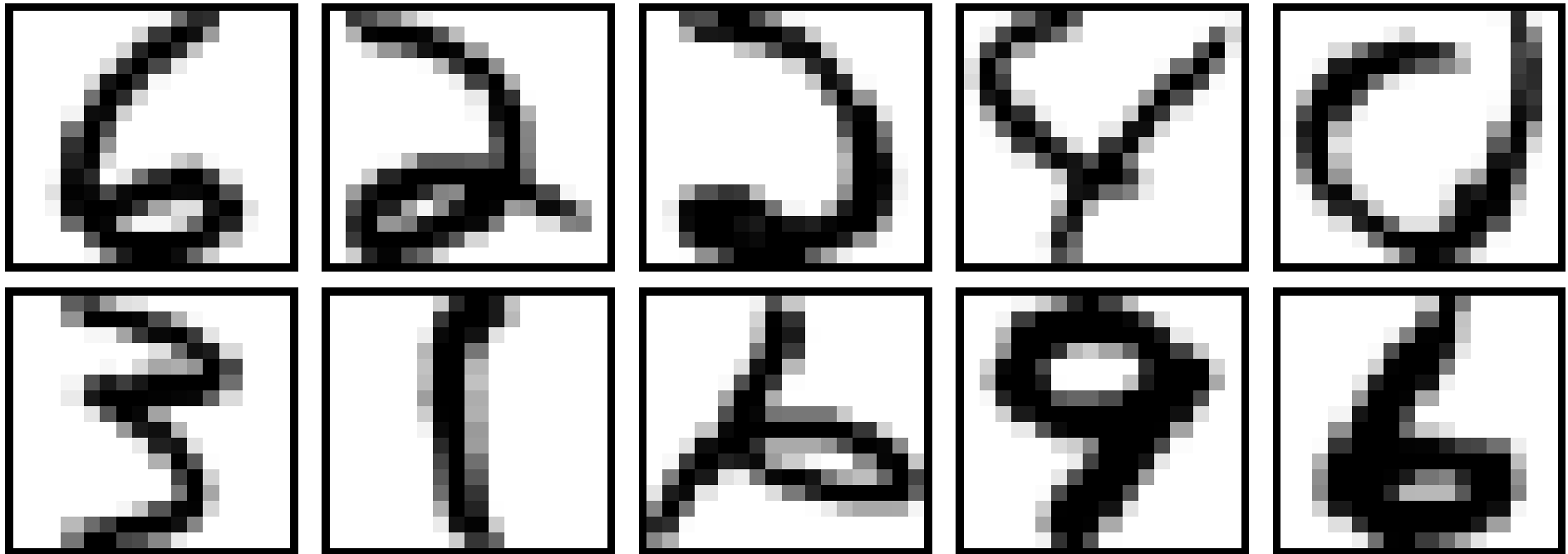
**Alle benötigten Datenfiles finden Sie auf der Webseite des Praktikums.**

**Die Daten sollen vollständig in den Hauptspeicher eingelesen werden, um langsame Plattenzugriffe zu minimieren.**

**Achten Sie darauf, dass die Programme unter Unix/Linux (ohne Warnungen!) kompilierbar sind und bei Aufruf mindestens eine kurze Hilfe ausgeben.**

**Benutzen Sie Funktionen und Kommentare.**

# Beispielbilder aus dem USPS Datensatz



# Anhang: Definition der Filter-Operation (Faltung)

## Definition der Filter-Operation (Faltung):

**Bild**  $g(i, j)$ ,  $i = 1, \dots, I$ ,  $j = 1, \dots, J$

**Filter**  $f(k, l)$ ,  $k = 1, \dots, K$ ,  $l = 1, \dots, L$

$$\hat{g}(i, j) = \sum_{k=1}^K \sum_{l=1}^L f(k, l) \cdot g\left(i + k - \frac{K+1}{2}, j + l - \frac{L+1}{2}\right)$$

**Anmerkung:** Dies ist die angemessene Definition hier. Sonst wird die Faltung oft ohne Offset und mit negativem Vorzeichen für  $k$  und  $l$  definiert.