

6. Aufgabenblatt zum Softwarepraktikum: „Muster- und Bilderkennung“ im Grundstudium

Philippe Dreuw
dreuw@informatik.rwth-aachen.de

Hinweis: Schauen Sie sich noch einmal die Folien des Einführungsvortrages an, dort werden ergänzende Hinweise gegeben.

Aufgabe 0: IO-Routinen für das .pgm-Fileformat

Entwerfen Sie eine Einlese- und eine Ausgaberroutine für das in der Vorbesprechung vorgestellte .pgm-Format. Speichern Sie die eigentlichen Bilddaten in einem 2-dimensionalen Array mit Graustufenwerten $g_{ij} \in \{0, \dots, 255\}$. Sie sollen *kein* Programm zur Visualisierung der Daten implementieren. Benutzen Sie dazu z.B. das Programm xv. Die Ausgaberroutine muss sicherstellen, dass die ausgegebenen Werte im Bereich $\{0, \dots, 255\}$ liegen.

Aufgabe 1: Grauwert-Punktoperationen

In dieser Aufgabe sollen verschiedene Grauwert-Punktoperatoren implementiert werden:

- a) Schreiben Sie eine Routine zur linearen Grauwertspreizung, d.h.

$$\begin{aligned}\hat{g}_{ij} &= 0, \text{ für } 0 \leq g_{ij} \leq x_1 \\ \hat{g}_{ij} &= \left[255 \cdot \frac{g_{ij} - x_1}{x_2 - x_1} \right], \text{ für } x_1 \leq g_{ij} \leq x_2 \\ \hat{g}_{ij} &= 255, \text{ für } x_2 \leq g_{ij} \leq 255\end{aligned}$$

Realisieren Sie nun eine Min-Max-Spreizung der Testbilder (d.h. $x_1 = \min_{i,j}\{g_{ij}\}$, $x_2 = \max_{i,j}\{g_{ij}\}$) und untersuchen Sie andere (Ihnen geeignet erscheinende) Werte x_1, x_2 . Welche Extremfälle sind denkbar?

- b) Realisieren Sie eine Gamma-Korrektur für Grauwertbilder, also $\hat{g}_{ij} = 255 \cdot \left(\frac{g_{ij}}{255}\right)^{1/\gamma}$ und untersuchen Sie die Auswirkung unterschiedlicher Gamma-Werte auf die Beispielbilder.

Aufgabe 2: Lineare Filter

Implementieren Sie die folgenden linearen Filter und untersuchen Sie deren Anwendung auf die Beispielbilder.

- a) Implementieren Sie einen Rechteck-Filter (für beliebige ungerade Kantentlängen), der für die Kantentlänge 3 durch folgendes 3×3 -Template gegeben ist:

$$\frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{Untersuchen Sie die Auswirkungen anderer Template-Größen!}$$

- b) Implementieren Sie einen Gauß-Filter (für beliebige ungerade Kantenlängen), der für die Kantenlänge 3 durch folgendes Template gegeben ist:

$$\frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

- c) Lineare Filter finden in der Praxis zahlreiche Anwendungen. Eine der einfachsten Anwendungen in der Bildverbesserung wird als *unsharp masking* bezeichnet, bei dem das Bild geschärft wird. Beim unsharp masking wird in drei Schritten vorgegangen:

1. Das Bild wird mit einem Tiefpassfilter geglättet (z.B. mit einem Gauss Filter)
2. Die Differenz zwischen dem Bild und dem Geglätteten wird berechnet (dies zeigt die Kanten des Bildes, d.h. die hochfrequenten Anteile im Bild)
3. Diese Differenz wird auf Originalbild addiert.

Da es sich um lineare Filter handelt, kann man dies auch in einem Schritt machen (D.h. das Verfahren soll mit nur einer Filteroperation auskommen.) Dazu muss ein neuer Filter als Linearkombination aus den Filtern, die oben verwendet werden, definiert werden. (Tipp: Welcher Filter wird implizit auf ein Bild angewandt, das nicht verändert wurde?). Realisieren Sie eine One-Pass-Variante des Unsharp Masking, basierend auf Gaußfiltern der Ordnung $n = 2$ bzw. $n = 4$ und größer.

Beachten Sie bei diesen Aufgaben immer, dass die Summe aller Matrixeinträge 1 ergeben sollte, um eine Veränderung der Originalhelligkeit zu vermeiden!

Aufgabe 3: Einfacher Bildklassifikator

Im Folgenden sollen einfache Experimente auf dem Gebiet der Handschrifterkennung durchgeführt werden. Dazu erhalten Sie 7291 Trainingsbeispiele sowie 2007 Testbeispiele handgeschriebener Ziffern des US-Postal-Service Datensatzes.

- a) Führen Sie unter Verwendung der gegebenen Trainingsdaten eine Nearest-Neighbor Klassifikation der Testdaten durch. Benutzen Sie dazu das Quadrat der Euklidischen Distanz $\|x-y\|^2 = \sum_d (x_d - y_d)^2$. Wieso verändert die Verwendung des Quadrates das Ergebnis nicht? Schreiben Sie eine **separate Funktion**¹ zur Distanzberechnung! Behalten Sie alle Trainingsdaten im Hauptspeicher. Welche Fehlerrate erzielen Sie?
- b) Untersuchen Sie nun die Auswirkungen unterschiedlicher Vorverarbeitungen (Grauwert-Punktoperationen, Filter ...) auf die Fehlerrate.
- c) Geben Sie für einige Fehlklassifikationen die Dateinamen des falsch klassifizierten Bildes sowie des nächsten Nachbarn aus den Trainingsdaten aus und interpretieren Sie die Bilder.

Sollten Probleme mit dem Speicher oder mit der Rechenzeit auftreten so benutzen Sie jeweils entsprechende kleinere Datensätze zum Debugging.

Die benötigten Dateien finden sie auf der Webseite des Praktikums.

Abgabe:

Donnerstag, **12./16. Juli 2006**; bitte senden Sie **einen Tag vorher** Ihre Lösungen als Quellcode an dreuw@informatik.rwth-aachen.de. Achten Sie darauf, dass Ihre Programme unter Unix/Linux ohne Warnungen mit dem Parameter `-Wall` und einem entsprechenden `Makefile` kompilierbar sind und bei Aufruf mindestens eine kurze Hilfe ausgeben, falls das Programm ohne Parameter aufgerufen wird²

¹Erfahrungsgemäß benötigt die Distanzberechnung die meiste Rechenzeit. Dies können Sie mittels Profiling überprüfen, wenn Sie möchten. Achten Sie daher besonders auf die effiziente Implementierung der Distanzfunktion. Verwenden Sie z.B. *nicht* die Funktion `pow()` zur Berechnung des Quadrates. (Warum nicht?)

²Sie können z.B. `libargtable2 [src]` benutzen, um ihre Kommandozeilenparameter einzulesen.