Praxisorientierte Einführung in C++
Aufgabenblatt 9

Christof Elbrechter
celbrech@techfak.uni-bielefeld.de

8. Juni 2012

Aufgabe1: Vererbung (14 Punkte)

In der Bildverarbeitung versteht man unter einem Filter i.A. ein System, welches ein Bild *umwandelt* und dabei ein neues Bild ausgibt.
Verwenden Sie als Ausgangspunkt für diese Aufgabe folgendes Filter-Interface:

```
// filter.h
#ifndef FILTER_H
#define FILTER_H

#include "image.h"
#include <algorithm>
#include <iostream>
namespace image{

  class Filter{
    public:
    virtual ~Filter(){}
    virtual Image apply(const Image &image) const = 0;

    protected:
    static inline bool compatible(const Image &a, const Image &b){
      return (a.getWidth()==b.getWidth()) &&
             (a.getHeight()==b.getHeight());
    }
  };
}
```

a. (4 Punkte) Erweitern Sie die Klasse `Filter` durch `public` Vererbung zu einer Klasse `ThresholdFilter`, welche einen Schwellwertfilter implementiert (die Dateien sollen hiefür "threshold_filter.h" bzw. ".cpp" genannt werden). Überschreiben Sie in der abgeleiteten Klasse die Funktion

```
virtual Image apply(const Image &image) const;
```

. Der Konstruktor soll den zu verwendenden Schwellwert als Parameter übergeben bekommen (Default-Wert z.B.: 128) und diesen in einer entsprechenden Membervariable speichern.

b. (3 Punkte) Schreiben Sie ein Programm "threshold_main.cpp" welches ein Bild einliest und den von Ihnen implementierten Schwellwertfilter darauf anwendet. Das Ergebnisbild soll dann wieder gespeichert werden. Die Dateinnamen von Ein- und Ausgabebild sowie der zu verwendende Schwellwert sollen dem Programm folgendwerweise übergeben werden:

- -input=INPUTDATEINAME
- -output=OUTPUTDATEINAME
- -threshold=THRESHOLD

Erklärung:
Der sog. Schwellwertfilter ist ein besonders simples Beispiel; er findet allerdings in den meisten Bildverarbeitungssystemen Verwendung. Er vergleicht jeden Pixel des Eingabebildes mit einem Schwellwert, und weist, je nachdem ob der Pixelwert über oder unter dem Schwellwert liegt, dem ensprechenden Ausgabepixel z.B. 0 oder 255 zu.

$$\Theta(x) = \begin{cases} 0 & \text{x} \leq \text{Schwellwert} \\ 255 & \text{sonst} \end{cases} \tag{1}$$

c. (4 Punkte) Erweitern Sie analog zur vorherigen Teilaufgabe die Klasse `Filter` zu einer Klasse `Convolution3x3Filter` (Dateinamen analog). Der Konstruktor dieser Klasse soll folgende Signatur aufweisen:

```
Convolution3x3Filter(const float kernel[9]);
```

Für die Organisation der 3×3 Werte in dem Konstruktor-Argument `kernel`, soll die gleiche Logik wie für die Pixel in der `Image`-Klasse verwendet werden (Stichwort: Row-Major-Order).
Erklärung:
Es handelt sich hierbei um einen sog. Faltungsfilter, welcher ebenfalls zu den gebräuchlichsten Filtern gehört. Eine Faltung $f * g$ zweier diskreter zweidimensionaler Signale (z.B. Bilder) ist definiert als:

$$(f * g)(x, y) = \sum_{i=-W}^{W} \sum_{j=-H}^{H} f(x - i, y - j) \cdot g(i + W, j + H) \tag{2}$$

mit

- f Eingabebild
- g Faltungs-Maske (sog. "Kernel")
- W Halbe Kernelbreite, abgerundet ($W = \lfloor \frac{\text{width}(g)}{2} \rfloor$)
- H Halbe Kernelhöhe, abgerundet ($H = \lfloor \frac{\text{height}(g)}{2} \rfloor$)
- $(f * g)$ Ergebnisbild

Umgangsprachlich ausgedrückt, kann mann eine Faltung folgendermaßen beschreiben: Gegeben sei ein Bild I der Größe $I_w \times I_H$ und ein Faltungskernel K der Größe $K_W \times K_H$ (im Grunde ist der Faltungskernel auch ein kleines Bild). Nun wird der Faltungskernel sukzessive mit seinem Mittelpunkt an jede Position (x, y) des Bildes gelegt. Hierbei müssen solche Positionen, an denen der Rand des Kernel über den Rand des Bildes übersteht besonders behandelt werden (i.d.R. werden diese Positionen einfach ausgelassen; Ihre Implementation sollte dieses auch tun: das Ergebnisbild hat also einen ein Pixel breiten schwarzen Rand). Der Wert des Ergebnisbildes an der Position (x, y) ergibt sich dann, indem man Maske und Bild *lokal* elementweise multipliziert und die so erhaltenen Produkte aufaddiert. Um mathematisch korrekt zu bleiben, muss der Kernel zuvor sowohl an der X- also auch an der Y-Achse gespiegelt werden.

d. (3 Punkte) Erweitern Sie das Programm "threshold_main.cpp" zu einem neuen Programm "borderdetection_main.cpp", welches auf das eingelesene Bild zunächst eine Faltung und dann auf das Ergebnis **optional** noch den Schwellwertfilter anwendet. Die Faltungsmaske soll ebenfalls als Parameter übergeben werden können:

- -input=INPUTDATEINAME
- -output=OUTPUTDATEINAME
- -threshold=THRESHOLD
- -kernel=COMMA_SEPARATED_LIST_OF_KERNEL_VALUES

e. (0 Punkte) Diese Aufgabe ist optional! Was bewirkt die Verwendung folgender Kernel:

$$K_1 = \frac{1}{8} \cdot \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \tag{3}$$

$$K_2 = \frac{1}{16} \cdot \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \tag{4}$$

Aufgabe2: Qt (6+4 Punkte)

a. (6 Punkte) Machen Sie sich mit den Qt-Klassen `QImage` und `QWidget` vertraut und implementieren Sie eine Komponente, die Grauwertbilder vom Typ `Image` anzeigen kann.
Erweitern Sie die `QWidget` Klasse mit einer eigenen Klasse die Sie `ImageWidget` nennen. Die Klasse soll das anzuzeigende Bild im Konstruktor übergeben bekommen. Implementieren Sie `ImageWidget` in `ImageWidget.h` und `ImageWidget.cpp`. Überschreiben Sie in der `ImageWidget`-Klasse die virtuelle Funktion `paintEvent` aus der `QWidget`-Klasse.

```
virtual void paintEvent(QPaintEvent *);
```

Innerhalb der Methode müssen Sie eine `QPainter`-Instanz verwenden. Wie Sie das Bild dann anzeigen, bleibt Ihnen überlassen. I.d.R. ist es aber am effizientesten, die PixelDaten in ein `QImage` zu übertragen, welches dann mithilfe des `QPainter` effizient gerendert werden kann. Schreiben Sie ein Mainprogramm `imageview.cpp`, welches einen Dateinamen als Parameter erwartet, das Bild lädt und mithilfe der `ImageWidget` Komponente anzeigt.
Zum Übersetzen können Sie `qmake` verwenden.

b. (4 Bonuspunkte) Implementieren Sie ein Programm filter.cpp, welches ein Eingabebild und eine Filtermaske als Parameter erwartet, um das Eingabebild und Filterergebnis nebeneinander anzuzeigen.