Projektdokumentation Bildfaltung in Csharp

Aufgabe 3 des Praktikums des Modules
Informatik 3

Gruppe B

Hochschule Reutlingen Fakultät Informatik
Studiengang: Medien-Kommunikations-Informatik

Martin Lauterbach

Amina Anzorova

Sissi Wu

Abstract (Wu): In dieser Dokumentation wird eine Bildverarbeitungssoftware implementiert, die verschiedene Filteroperationen auf PGM-Bildern durchführt. `Image.cs` bietet Methoden zum Einlesen und Speichern von PGM-Bildern sowie die Methode `convolve`, die eine Bildfaltung mit unterschiedlichen Filterkernen und Randbehandlungen ermöglicht. Zwei Randbehandlungsstrategien, ZeroPadding und Clamping, werden implementiert, um die Unterschiede in den Filterergebnissen zu verdeutlichen. Zusätzlich wird `KernelFactory.cs` bereitgestellt, die statische Methoden zur Erstellung von Prewitt-Filtern (horizontal und vertikal) und Box-Filtern in variablen Größen enthält. Die Hauptmethode in `Program.cs` filtert zwei gegebene Bilder mit verschiedenen Filtern und speichert die Ergebnisse in entsprechenden Verzeichnissen. Dabei wird die Auswirkung der Filterkernelgröße und der Randbehandlungsstrategie auf das Filterergebnis untersucht. Die Ergebnisse werden sowohl auf der Festplatte gespeichert als auch auf dem Bildschirm ausgegeben.

Link zur Hochschuloffenen Repository aller Praktikumsaufgaben:
`https://gitlab.reutlingen-university.de/inf3-theteam/aufgaben-repo-inf3-sose14`
Stand: 27. Juni 2024
SoSe 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Strukturelle Erklärung des Projekts (Anzorova)

Bei der Problemlösung gehen wir wie folgt vor: Zuerst werden die Bilder ausgesucht und geladen. Diese werden durch Prewitt-Filter verarbeitet. Prewitt-Filter werden in der Bildverarbeitung häufig zur Kantenerkennung verwendet. Die veränderten Bilder werden in einem neuen Ordner gespeichert und durch ein Frontend-Canvas dargestellt.

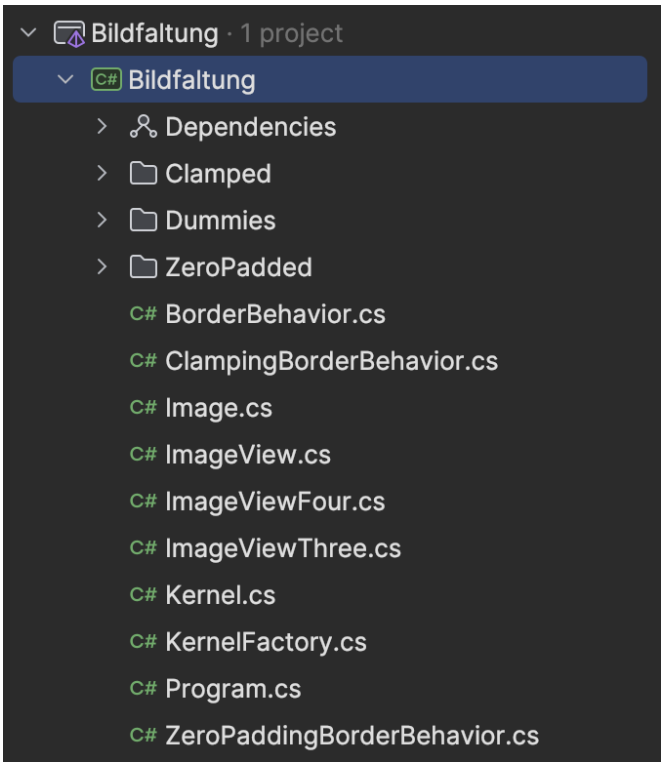

Abbildung 1: Projektstruktur, Screenshot aus Rider, Anzorova

Diese grundlegenden Methoden ermöglichen eine strukturierte und modulare Vorgehensweise zur Anwendung von Filtern auf Bilder, das Speichern der Ergebnisse und das Anzeigen der gefilterten Bilder:

Die Methode public void ReadFromFile() ermöglicht das Lesen eines Bildes oder einer Bilddatei von einer spezifischen Quelle. Sie übernimmt die Verantwortung für den Zugriff auf die Bilddaten und deren Darstellung als Bildobjekt in der Anwendung. Dies ist der erste Schritt, um ein Bild für nachfolgende Operationen wie Filteranwendung oder Analyse vorzubereiten.

private double ConvolvePixel() - diese Methode führt die Faltung für einen einzelnen Pixel des Bildes aus. Die Faltung ist eine grundlegende Operation in der Bildverarbeitung, bei der ein Filter über das Bild angewendet wird, um bestimmte Effekte wie Glättung, Kantenerkennung oder Schärfung zu erzielen. Die ConvolvePixel()-Methode berechnet das Ergebnis der Faltung für einen Pixel basierend auf den umliegenden Pixeln und den Gewichtungen des Filters.

Die Methode public void WriteToFile() ermöglicht das Speichern eines Bildes oder einer Bildmatrix in einer Datei auf der Festplatte oder einem anderen Speichermedium. Nachdem Operationen wie Filteranwendung oder Bearbeitung abgeschlossen sind, kann diese Methode verwendet werden, um das bearbeitete Bild dauerhaft zu speichern. Sie ist entscheidend, um die Ergebnisse der Bildverarbeitung für die spätere Analyse oder Darstellung zu sichern.

Die BorderBehavior-Klasse beinhaltet die Umsetzung der Behandlung von Pixeln am Rand des Bildes. Damit wird das Verhalten der Pixel, die sich außerhalb der Bildgrenzen befinden, behandelt.

Die Clamping-Klasse ist eine konkrete Implementierung der BorderBehavior-Klasse, die das Clamping-Verhalten für die Randpixel definiert. Beim Clamping wird der Wert eines Randpixels auf den nächsten inneren Pixelwert des Bildes gesetzt. Dies bedeutet, dass die Farbe oder der Wert des Randpixels durch den nächsten inneren Pixelwert begrenzt wird, wodurch eine glatte Erweiterung des Bildes entsteht, ohne dass Artefakte wie helle Linien oder "Ghostingëntstehen.

2 UML-Diagram (Lauterbach)

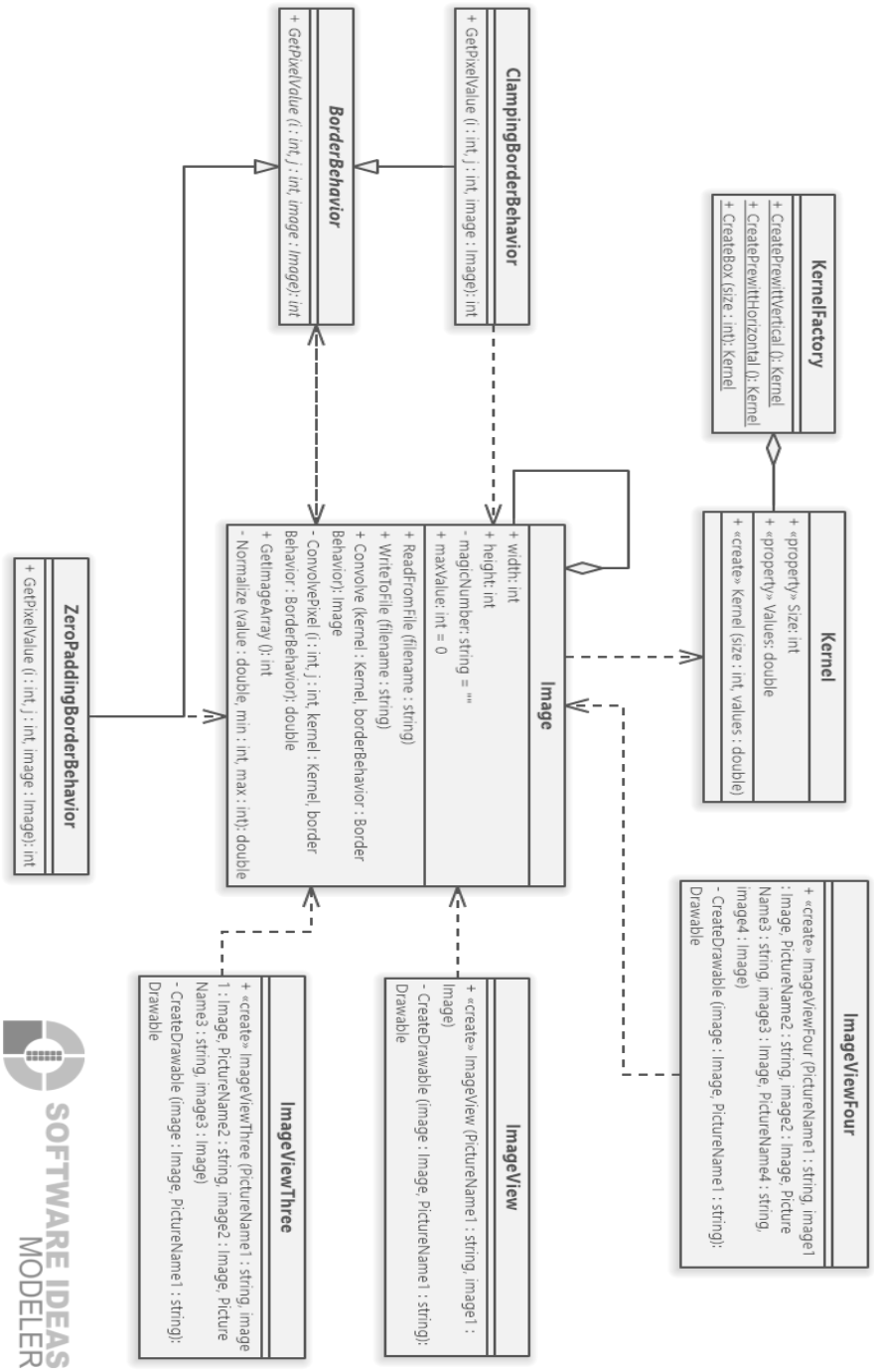

Abbildung 2: UML-Diagram, Software Ideas Modeler, Lauterbach Martin 20/06/2024

3 Projektmetriken (Anzorova)

Tabelle 1: Code-Metriken pro Klasse und hervorgehobenen Methoden; Anzorova; 27. Juni 2024

Entity	Lines of Code	Lines of Comments
Klassen:		
Image	118	45
ImageView	58	5
ImageViewThree	63	5
ImageViewFour	65	5
Kernel	8	0
KernelFactory	36	3
ZeroPaddingBorderBehavior	16	0
BorderBehavior	4	
ClampingBorderBehavior	11	0
Grundlegende Methoden:		
public void ReadFromFile()	43	6
private double ConvolvePixel()	25	5
public void WriteToFile()	26	6
Program.cs:		
gesamt	276	57

4 Antworten zu Aufgabenstellungen (Wu)

4.1 Implementieren Sie eine Klasse Image (1p)

Implementieren Sie eine Klasse `Image`, die folgende Methoden bietet:

- `Image.cs`

4.1.1 readFromFile(filename)

- Diese Methode liest ein PGM-Bild in ein zweidimensionales int-Array ein. Die Methode muss mindestens die Variante aus Listing 3 lesen können, d.h. eine Zahl pro Zeile. Die Auflösung des Bildes ist auf 1500x1500 begrenzt. Entspricht das Bild nicht einem der beiden vorgestellten Schemata, soll ein Fehler ausgegeben werden (2p).
- `Image_class` ReadFromFile_Methode Zeilen 25-74: Definiert zunächst alle Attribute wie das gegebene Format, liest die Datei und parst den Header. Initialisiert das Bild-Array und liest am Ende die Pixelwerte ein. Wenn keine Ausnahme auftritt, wird die nächste Zeile gelesen.

4.1.2 writeToFile(filename)

- Diese Methode speichert ein zweidimensionales int-Array in ein PGM-Bild. Es bietet sich an, das Schema aus Listing 2 zu verwenden, da dieses einfacher als Textdatei zu lesen ist (3p).
- `Image_class` WriteToFile_Methode Zeilen 78-103: Erstellt durch List_attribute eine Liste, die das Bild speichert. Fügt mit `lines.Add_attribute` den Header, die Dimensionen und den maximalen Wert hinzu. Mit einer for-Schleife werden die Pixelwerte hinzugefügt. Am Ende werden die Zeilen in den Ordner geschrieben.

4.1.3 convolve(kernel, borderBehavior)

- Diese Methode faltet das Bild mit einem Filterkernel. Das Verhalten am Rand des Bildes wird durch das Objekt `borderBehavior` festgelegt. Das Ergebnis der Faltung ist ein neues Bild, das als Rückgabewert übergeben wird. Da wir die Mitte des Filterkernels ermitteln müssen, muss ein Filterkernel immer eine ungerade Anzahl an Spalten und Zeilen haben (5p).
- `Image_class` Convolve_Methode Zeilen 105-195: Mit gegebenem Filterkernel und `BorderBehavior_Object` wird das Bild gefaltet. Erstellt ein neues Bild, um das Ergebnis zu speichern, durch das `Image result_Object` und `ConvolvePixel_Methode` (Zeilen 151-197).

4.2 Eine abstrakte Klasse BorderBehavior mit einer abstrakten Methode getPixelValue(i, j, image) (1p), die den Pixelwert des übergebenen Bildes image an der Stelle (i, j) zurückgibt. Von dieser Klasse werden zwei Spezialisierungen abgeleitet:

4.2.1 ZeroPaddingBorderBehavior, welche für (i, j) außerhalb des Bildes, wie in Abbildung 2 oben dargestellt, den Wert 0 zurückgibt (2p).

- Durch `BorderBehavior.cs` wird eine abstrakte Klasse bereitgestellt. Durch `ZeroPaddingBorde`
  wird der Wert 0 zurückgegeben.

4.2.2 ClampingBorderBehavior, welche für (i, j) außerhalb des Bildes, wie in Abbildung 2 unten dargestellt, den Wert des am nächsten gelegenen Randpixels zurückgibt (2p).

- Durch `ClampingBorderBehavior.cs` wird der Wert des am nächsten gelegenen Randpixels zurückgegeben.

4.3 Eine Klasse KernelFactory (1p)

- `KernelFactory.cs`

die folgende statische Methoden bietet:

4.3.1 createVerticalPrewittKernel, um einen vertikalen 3×3 Prewitt-Filter zu erstellen (1p)

- `KernelFactory.cs` mit `CreatePrewittVertical()` _Methode, Zeilen 3-13

4.3.2 createHorizontalPrewittKernel, um einen horizontalen 3×3 Prewitt-Filter zu erstellen (1p)

- `KernelFactory.cs` mit `CreatePrewittHorizontal()` _Methode, Zeilen 15-25

4.3.3 createBoxFilter(size), um einen quadratischen Box-Filter der Größe `size` zu erstellen. Bei einem Box-Filter hat jedes Element den Wert 1/`AnzahlDerElemente`. Bei einem 3×3 Box-Filter hat also jedes Element den Wert 1/9 (3p).

- `KernelFactory.cs` mit `CreateBox()` _Methode, Zeilen 27-40

4.4 Implementieren Sie eine Main-Methode, die folgende Filteroperationen einmal mit ZeroPadding und einmal mit Clamping durchführt und das Ergebnis jeweils in einem Ordner ZeroPadded und Clamped speichert. Insgesamt erhält man so 10 gefilterte Bilder. (1p)

- `Program.cs`

4.4.1 Das gegebene Bild1.pgm (Benutzen Sie GIMP für die Erstellung der Testbilder) wird je einmal mit dem horizontalen und einmal mit dem vertikalen Prewitt-Filter gefiltert und das Ergebnis als PGM-Bilder in die Dateien Ergebnis1-horizontal.pgm und Ergebnis1-vertical.pgm gespeichert. Wieso liefert hier ZeroPadding ein minimal anderes Ergebnis als Clamping? (1p)

- ZeroPadding verwendet Nullwerte zur Füllung, was dazu führt, dass in den Randbereichen bei der Faltung viele Nullwerte eingeführt werden. Dies kann dazu führen, dass die Randpixel dunkler erscheinen.
- Clamping hingegen verwendet den Wert der nächstgelegenen Randpixel zur Füllung, wodurch die ursprünglichen Informationen der Randbereiche erhalten bleiben und das Faltungsergebnis glatter wird.

4.4.2 Das gegebene Bild2.pgm wird je einmal mit einem Box-Filter der Größe 3, 11 und 27 gefiltert und die Ergebnisse werden jeweils in die Dateien Ergebnis2-3.pgm, Ergebnis2-11.pgm und Ergebnis2-27.pgm gespeichert. Was bewirkt eine Vergrößerung des Filterkernels bei einem Boxfilter den Ergebnissen nach zu urteilen? Wie wirken sich ZeroPadding und Clamping hier aus? (1p)

- Eine Vergrößerung des Filterkerns bei einem Boxfilter bewirkt, dass das Bild zunehmend geglättet wird. Das bedeutet, dass kleine Details und Rauschen im Bild reduziert werden, während größere Filterkerne zu einer stärkeren Unschärfe führen. Je größer der Filterkern ist, desto gleichmäßiger werden die Pixelwerte innerhalb der gefilterten Region, was zu einer weicheren Darstellung des Bildes führt.
- ZeroPadding kann zu dunkleren Rändern führen, während Clamping die ursprünglichen Randwerte beibehält und somit ein gleichmäßigeres Glättungsergebnis erzielt.

4.4.3 Geben Sie sowohl die eingelesenen Bilder (Bild1.pgm und Bild2.pgm) als auch alle Ergebnisse der Faltung auf dem Bildschirm aus. (1p)

- `Program.cs` einlesen Bilder Zeilen 25-44
- Faltungsergebnisse Zeilen 45-181
- Ergebnisse auf dem Bildschirm anzeigen Zeilen 182-276