Datenstrukturen für Bilder (in Java)

von Ralph Guderlei & Timo Tonn

Inhalt

1. Implementierte Operationen

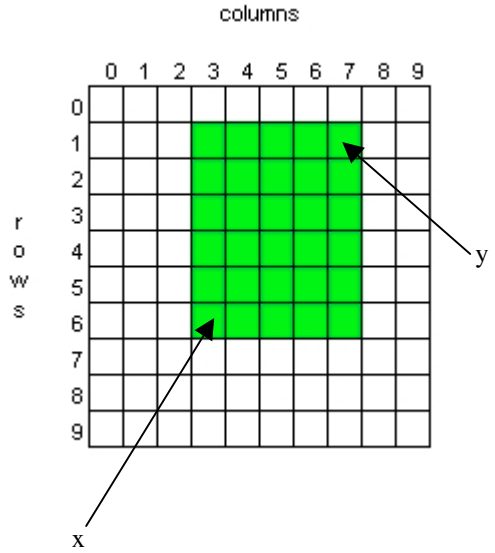
2. Implementierte Datenstrukturen

3. Literatur

1. Implementierte Operationen

1. DataStructure getSubrange(int[] x, int[] y)
2. DataStructure getSlice(int component, int value)
3. void flip(int component)
4. void transpose()
5. double getPixel(int[] x)

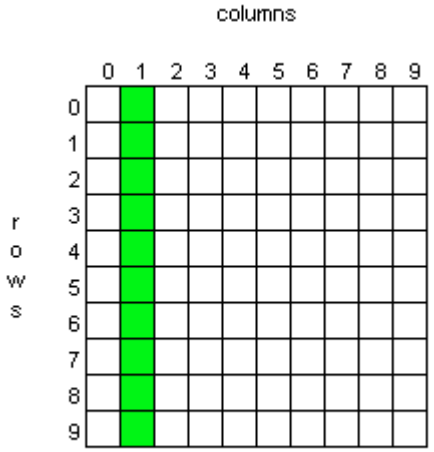
1. DataStructure getSubrange(int[] x, int[] y)



Liefert einen 2-dim. Ausschnitt des Bildes

Bsp.: x={3,6}, y={7,1}

2. DataStructure getSlice(int component, int value)



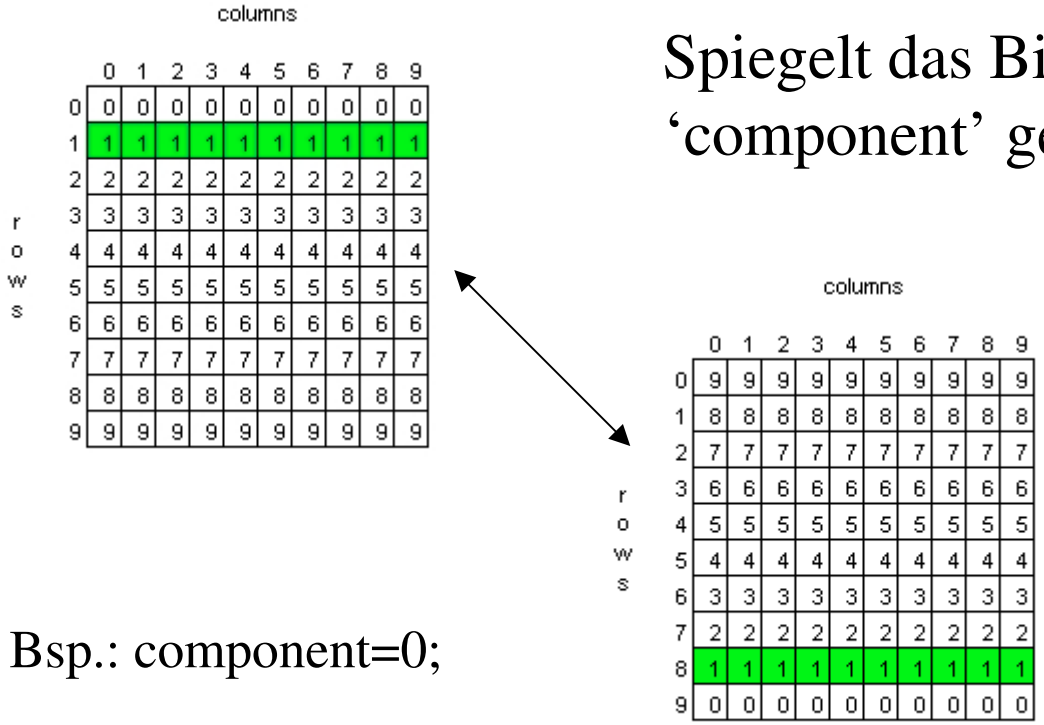
Liefert einen 1-dim. Ausschnitt des Bildes

‘component’ ist die Indexkomponente, die mit dem Wert ‘value’ konstant gehalten wird

Bsp.: component=0,

value=1

3. void flip(int component)

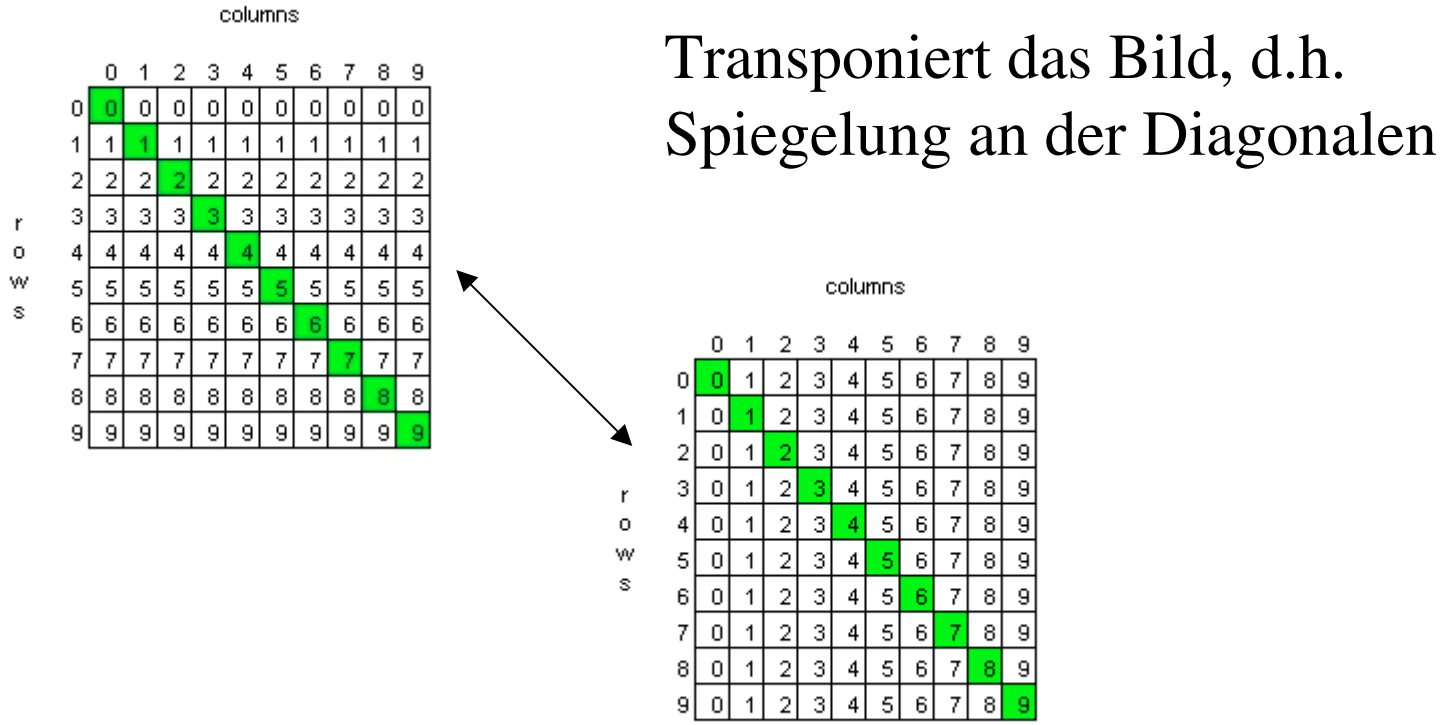


Spiegelt das Bild an der in ‘component’ gegebenen Achse

Bsp.: component=0;

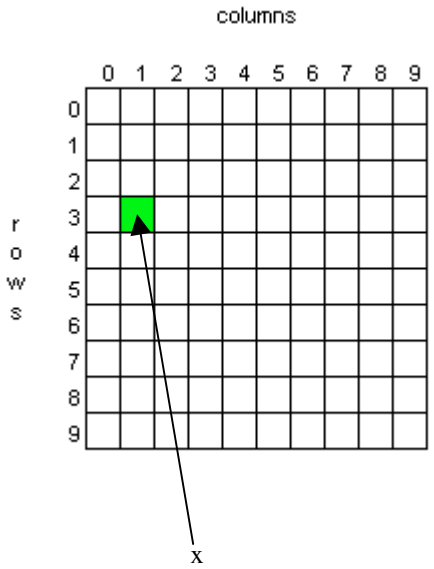
Spiegelung an x-Achse

4. void transpose()



Transponiert das Bild, d.h.
Spiegelung an der Diagonalen

5. double getPixel(int[] x)



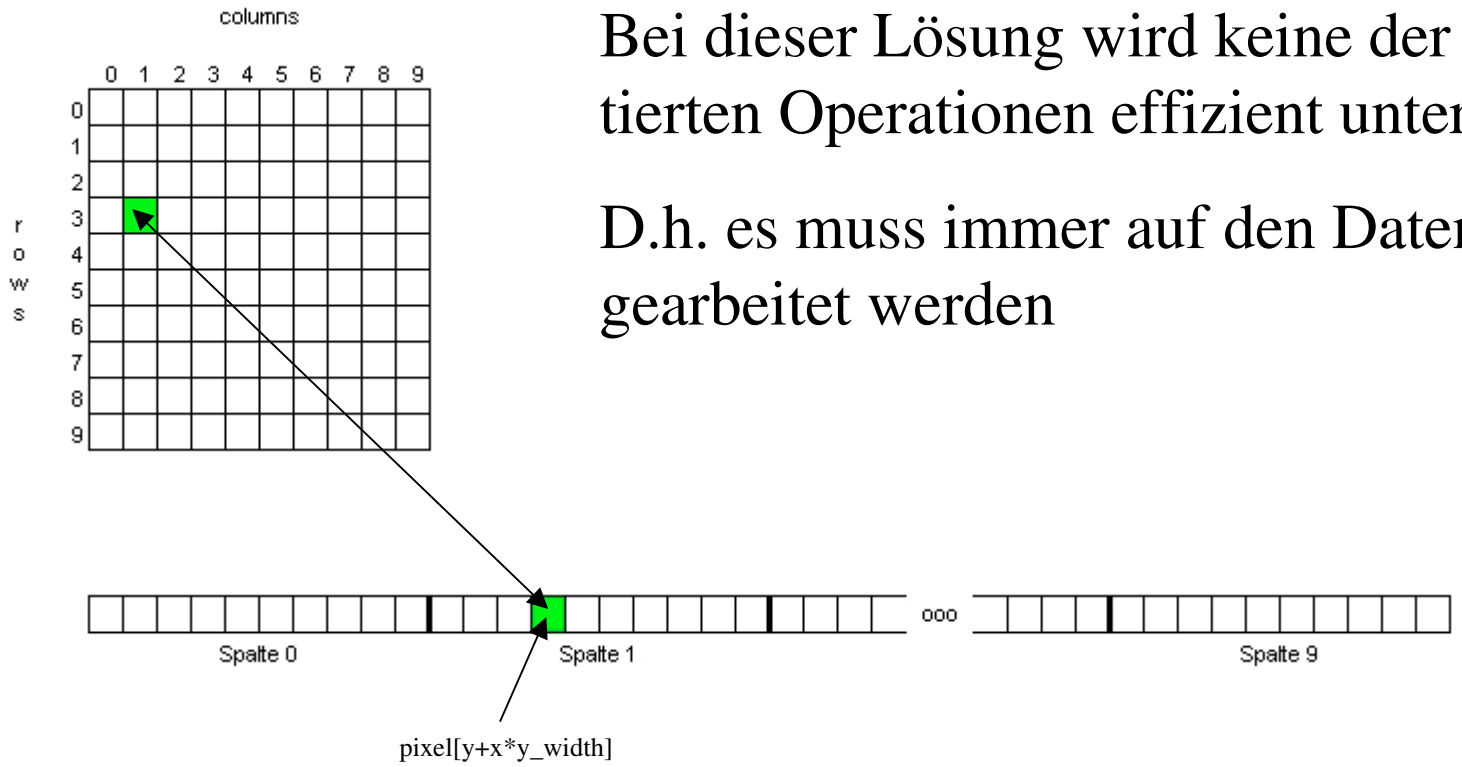
Liefert den Wert des Pixel x

Bsp.: x={1,3}

2. Implementierte Datenstrukturen

1. Multi-Dimensionale Arrays in C
2. Multi-Dimensionale Arrays in Java
3. Index Arrays
4. Arrays mit ‘Strides and Offset’

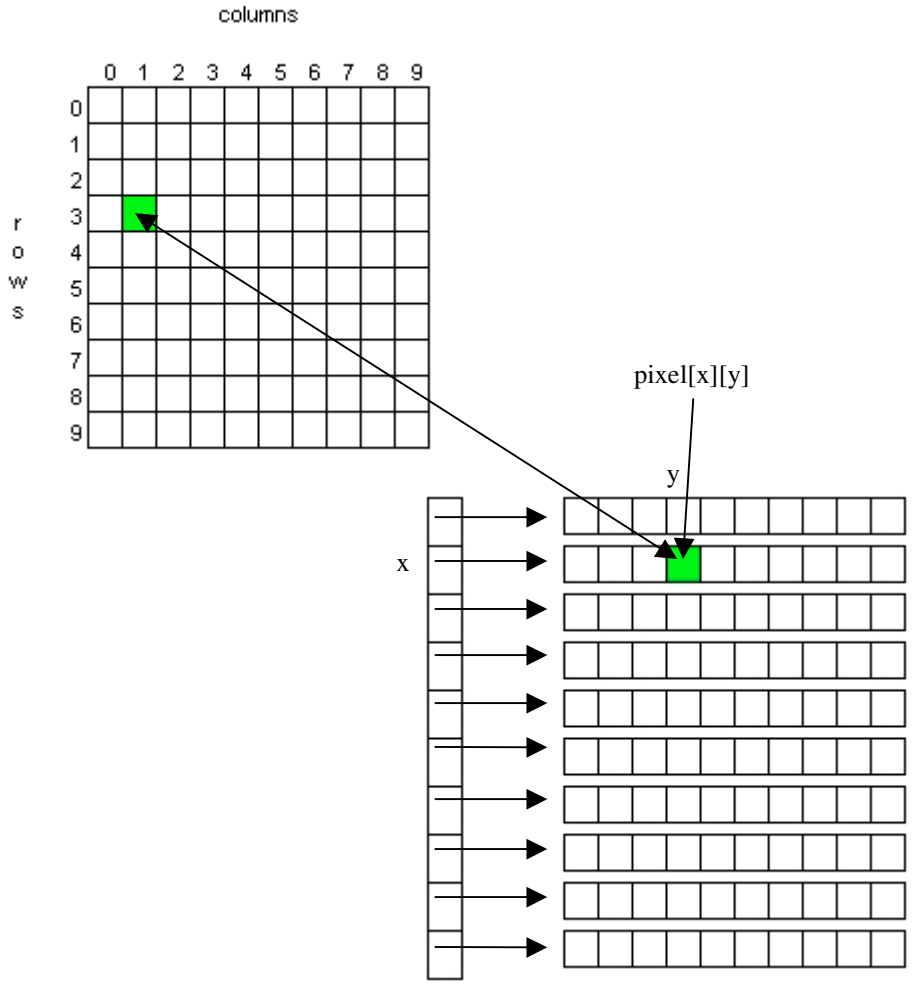
1. Multi-Dimensionale Arrays in C



Bei dieser Lösung wird keine der diskutierten Operationen effizient unterstützt.

D.h. es muss immer auf den Daten gearbeitet werden

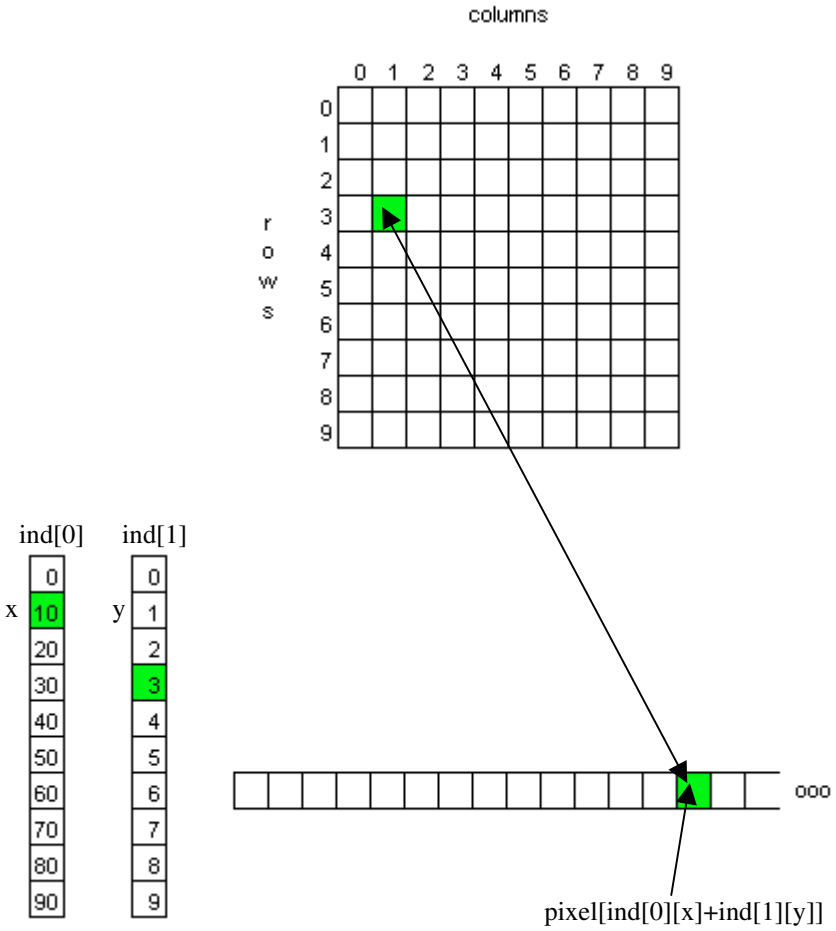
2. Multi-Dimensionale Arrays in Java



Effizient unterstütze Operationen:

- flip(1)
- getSlice(0, val)

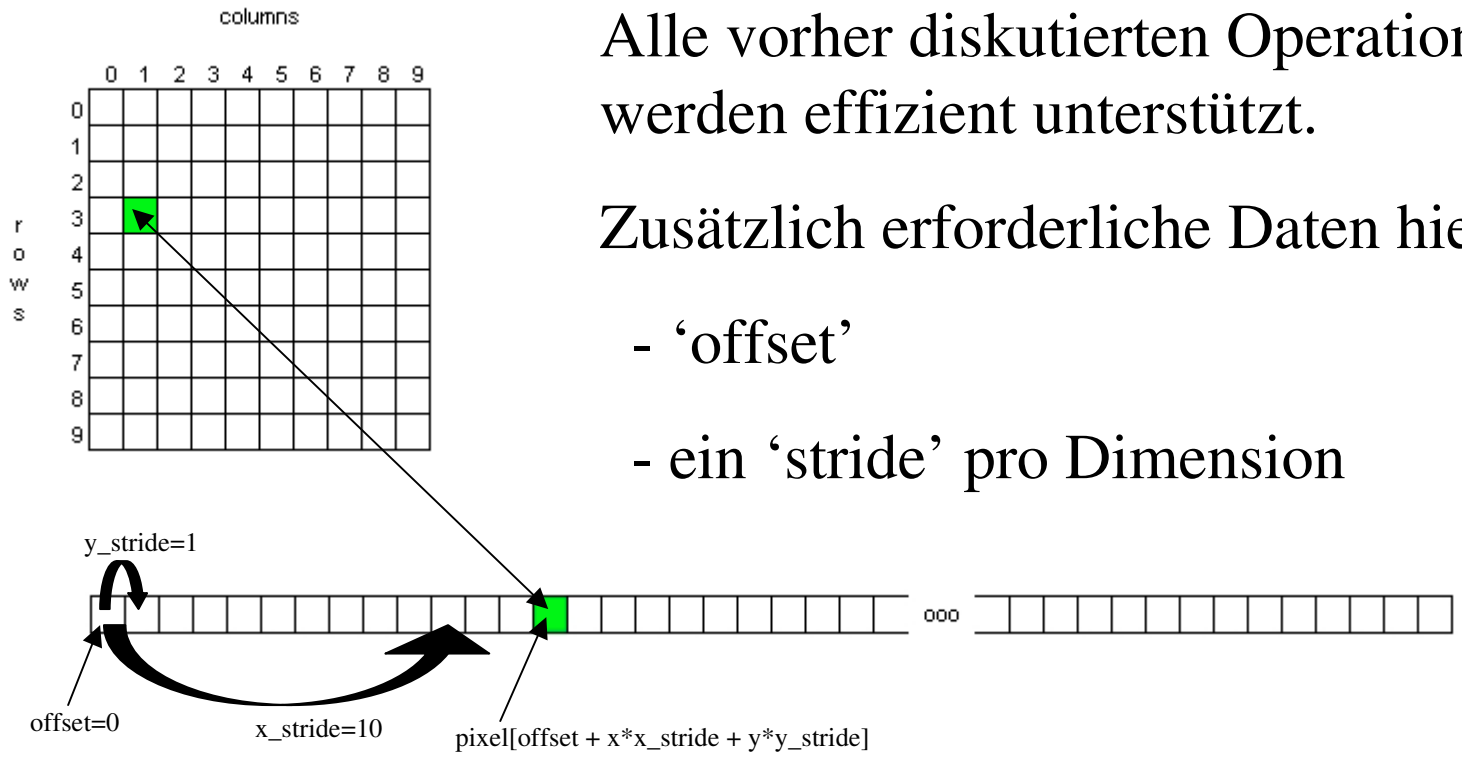
3. Index Arrays



Alle vorher diskutierten Operationen werden effizient unterstützt.

Nachteil: relativ großer ‘Overhead’

4. Arrays mit ‘Strides and Offset’

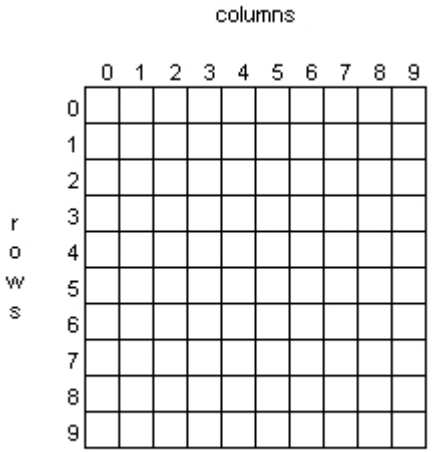


Alle vorher diskutierten Operationen werden effizient unterstützt.

Zusätzlich erforderliche Daten hierbei:

- ‘offset’
- ein ‘stride’ pro Dimension

Beispiel warum bei Arrays mit ‘Strides and Offset’ die Operationen effizient unterstützt werden



Routine zum Aufbau des Bildes:

```
for (int y=0, i=offset; y<y_dim; y++, i+=y_stride)
    for (int x=0, j=i; x<x_dim; x++, j+=x_stride)
        pixel[j];
```

Nun:

offset=9, x_stride=10, y_stride=-1 ⇔ flip(0)

offset=90, x_stride=-10, y_stride=1 ⇔ flip(1)

offset=0, x_stride=1, y_stride=10 ⇔ transpose()

(desweitern müssen hier x_dim und y_dim vertauscht werden)

x_dim=5, y_dim=6, offset=31, x_stride=10, y_stride=1 ⇔

getsubrange(x, y) mit x={3,6}, y={7,1}

3. Literatur

- H. Braxmeier, Data Structures for Images on Java, In: Proceedings of the Net.ObjectDays 2002
- J. Mayer, On Quality Improvement of Scientific Software: Design and Development of the GeoStoch Library and Graphical User Interface., Dissertation, Universität Ulm, 2003