SOBEL-FILTER (GRUPPE 57)

Leonhard Obkirchner, Julia Hahn, Henry Meyran

INHALT

Problemstellung und Vorgehensweise

Implementierung und Optimierungen

- SIMD Matrizenfaltung
- SIMD Ganzzahl-Wurzel Algorithmen

Output und Korrektheit

- Graustufenkonvertierung
- Sobel-Filter

Benchmarking

- Wurzel Funktionen
- Graustufenkonvertierung
- Implementation

Zusammenfassung

PROBLEMSTELLUNG

Bild 1

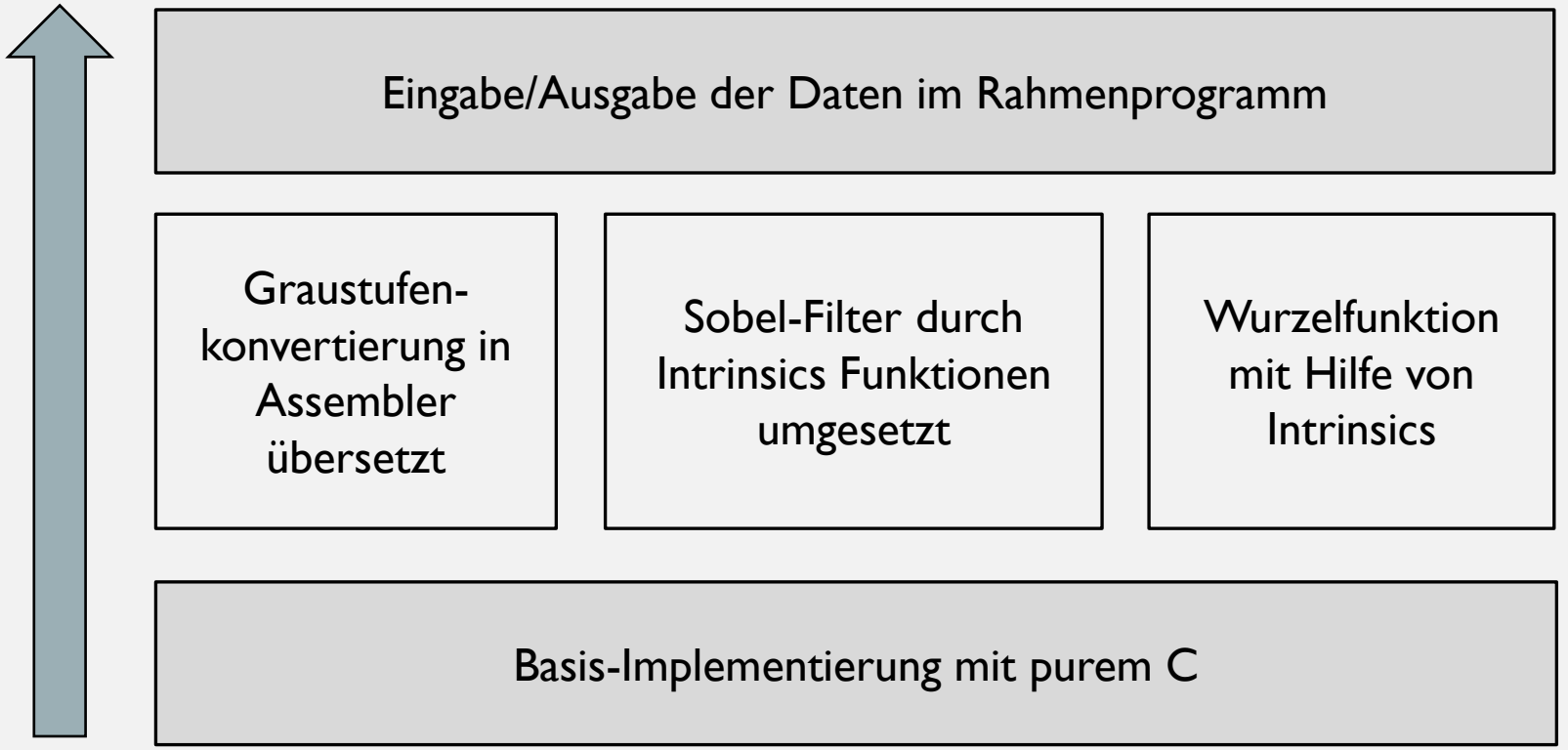
PROBLEMSTELLUNG

Graustufen-konvertierung mittels gewichtetem Durchschnitt

Kantenerkennung mit Sobel-Filter

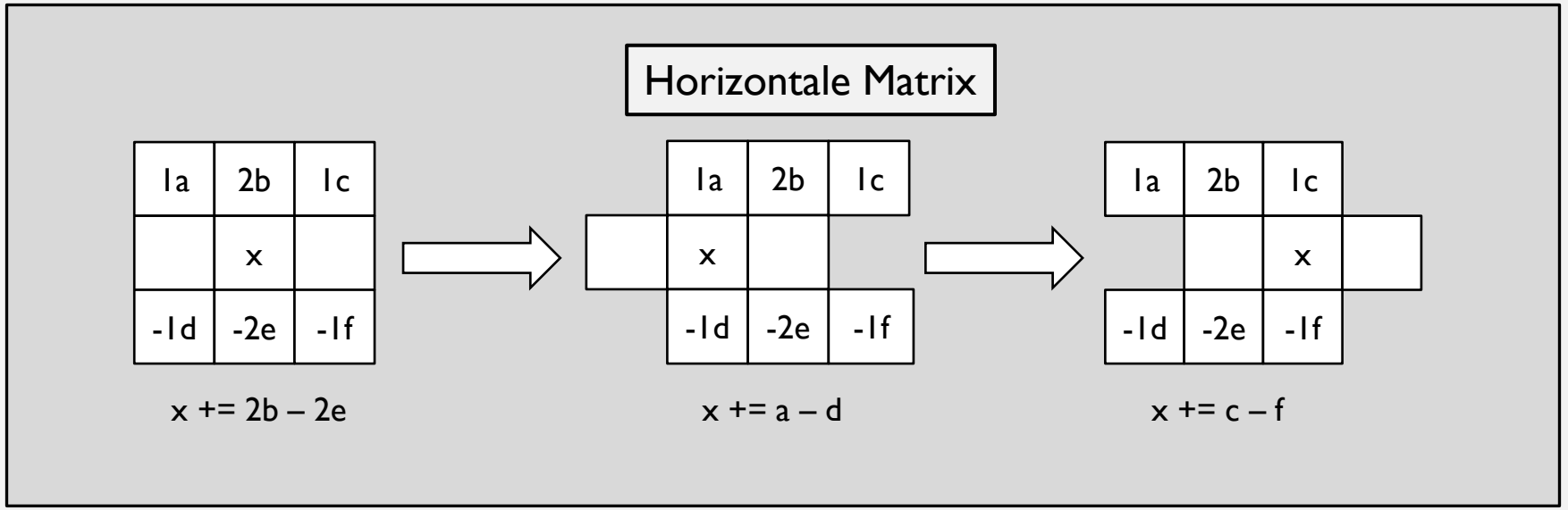
Verknüpfung vertikaler und horizontaler Kanten durch Wurzelfunktion

VORGEHENSWEISE



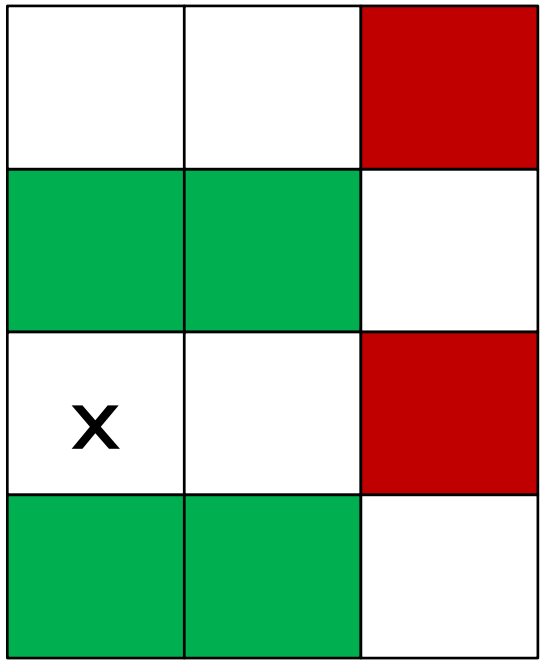
SIMD: MATRIZENFALTUNG

- Problem: Schwierig parallel auf die Daten zuzugreifen -> brauchen eigentlich mehrere am Stück
- Lösung: Mit Offset auf Daten zugreifen



SIMD: MATRIZENFALTUNG

- Problem: konstante Überprüfung nach Anfang/Ende einer Zeile notwendig
  - Bestimmte Werte dürfen nicht einfließen
- Lösung: Auslagerung der Überprüfung in separate Funktion, Problem in Hauptfunktion ignorieren
- Hauptfunktion fängt erst in zweiter Zeile an und hört in vorletzter auf
  - Verhindert problematische Speicherzugriffe
- Separate Funktion läuft danach über die Ränder und korrigiert diese



WURZEL-ALGORITHMEN

- Basierend auf James Ulery‘s Algorithmus
- Stellt Ergebnis Bit für Bit zusammen
- Dank einfacher Funktionsweise, ohne große Arithmetik implementierbar
- 4 Implementierungen (mit/ohne Mul Operation)
  - Zwei SISD C Implementierungen
  - Zwei SIMD Implementierungen
    - Simple Übersetzung des Algorithmus

Bit: 1000.0000
Input: 0110.0000
Result: 0000.0000
Bit + Result = 1000.0000
(Bit + Result)^2 > Input
Neues Result: 0000.0000
Bit>>1
Neues Bit: 0100.0000
8mal wiederholen
-> Baut Result für jedes Bit auf

SIMD: 16 ZU 8 BIT KONVERTIERUNG

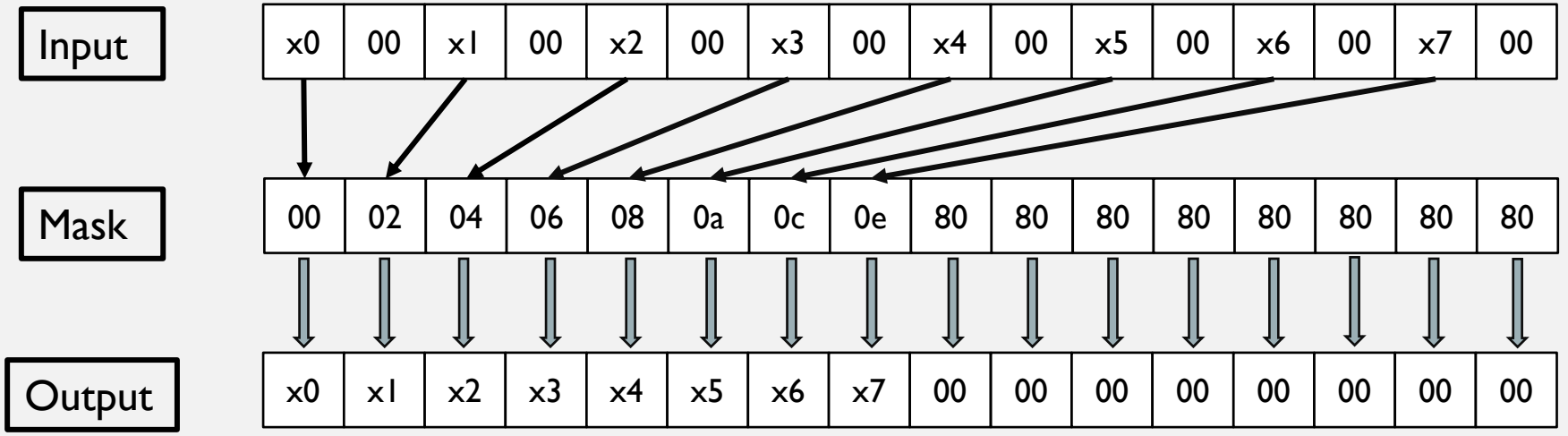


Bild 2

Input

Implementierung

GIMP

OUTPUT: GREYSCALE

Rot

Grün

Blau

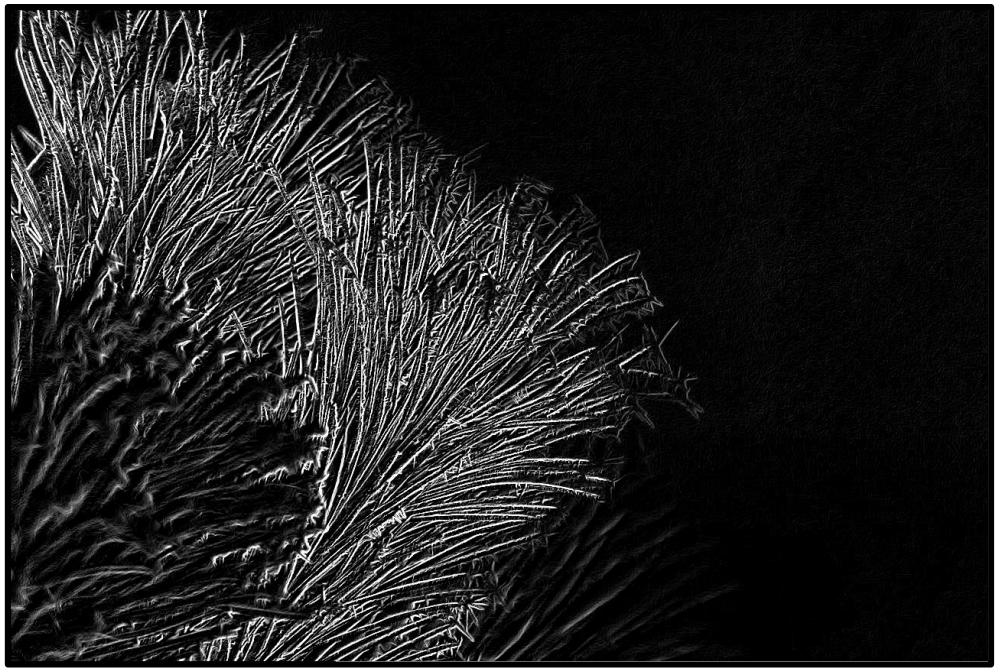
OUTPUT: GREYSCALE

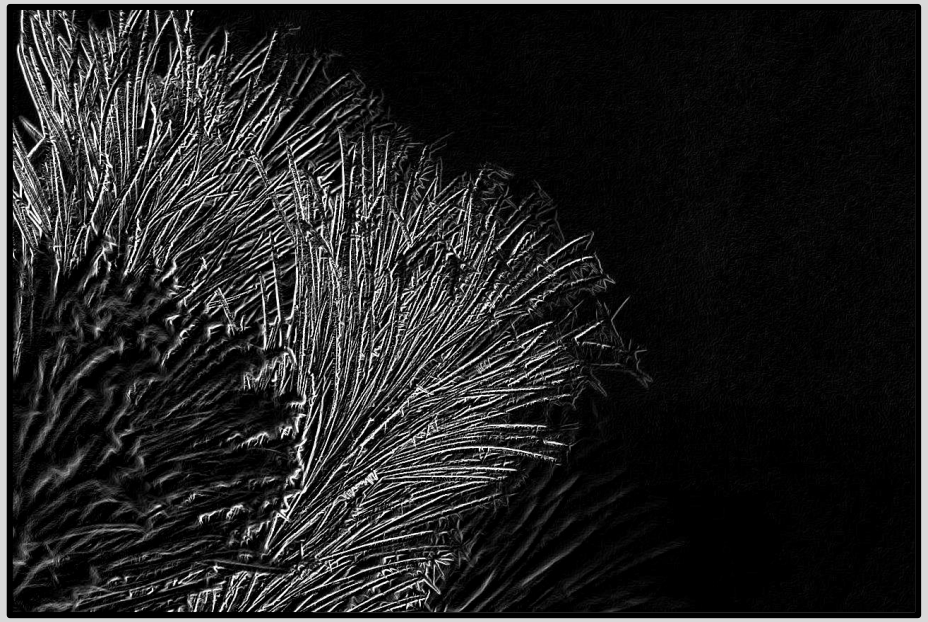
OUTPUT: SOBEL

Input

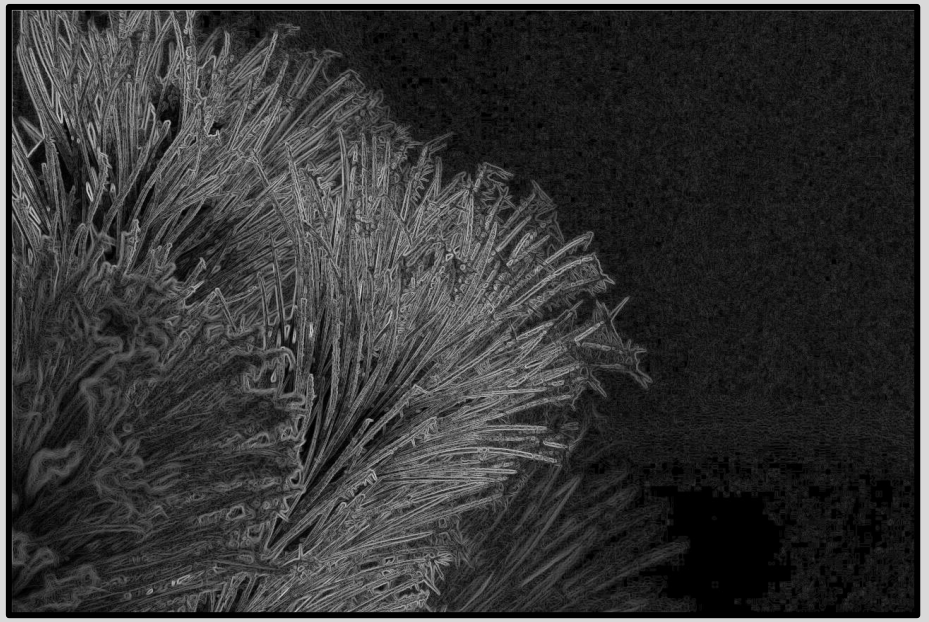
Implementierung

Bild 3





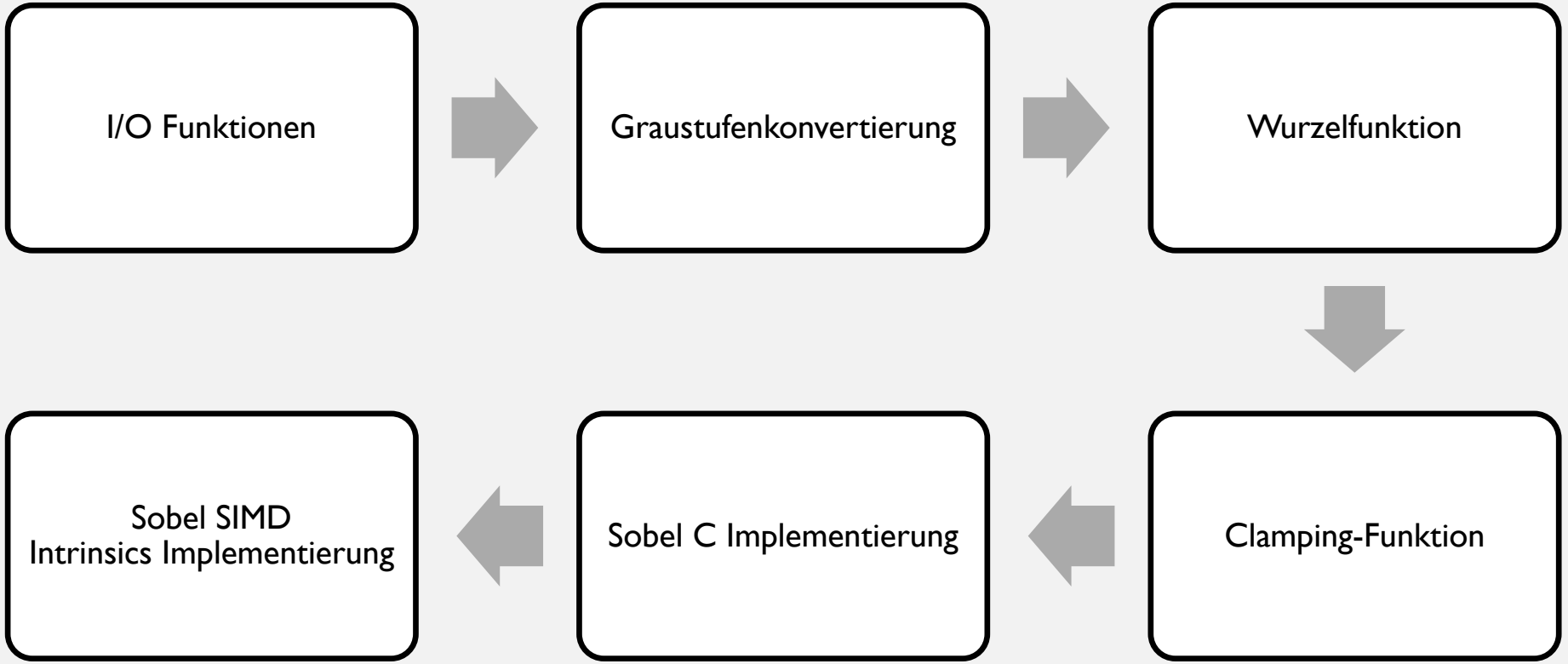
Implementierung



GIMP

KORREKTHEIT

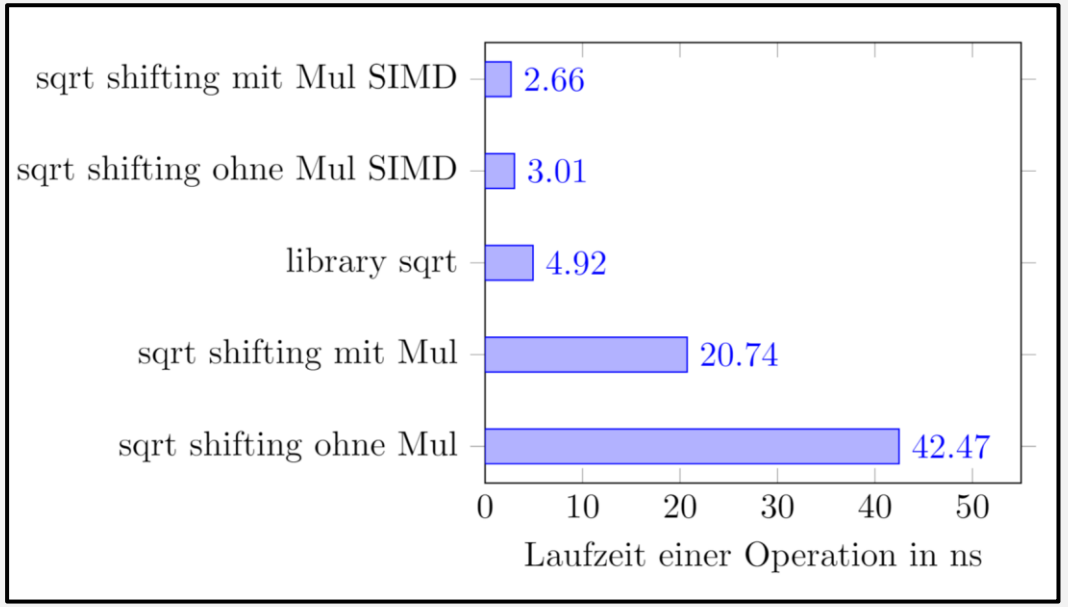
KORREKTHEIT



KORREKTHEIT: RÄNDER

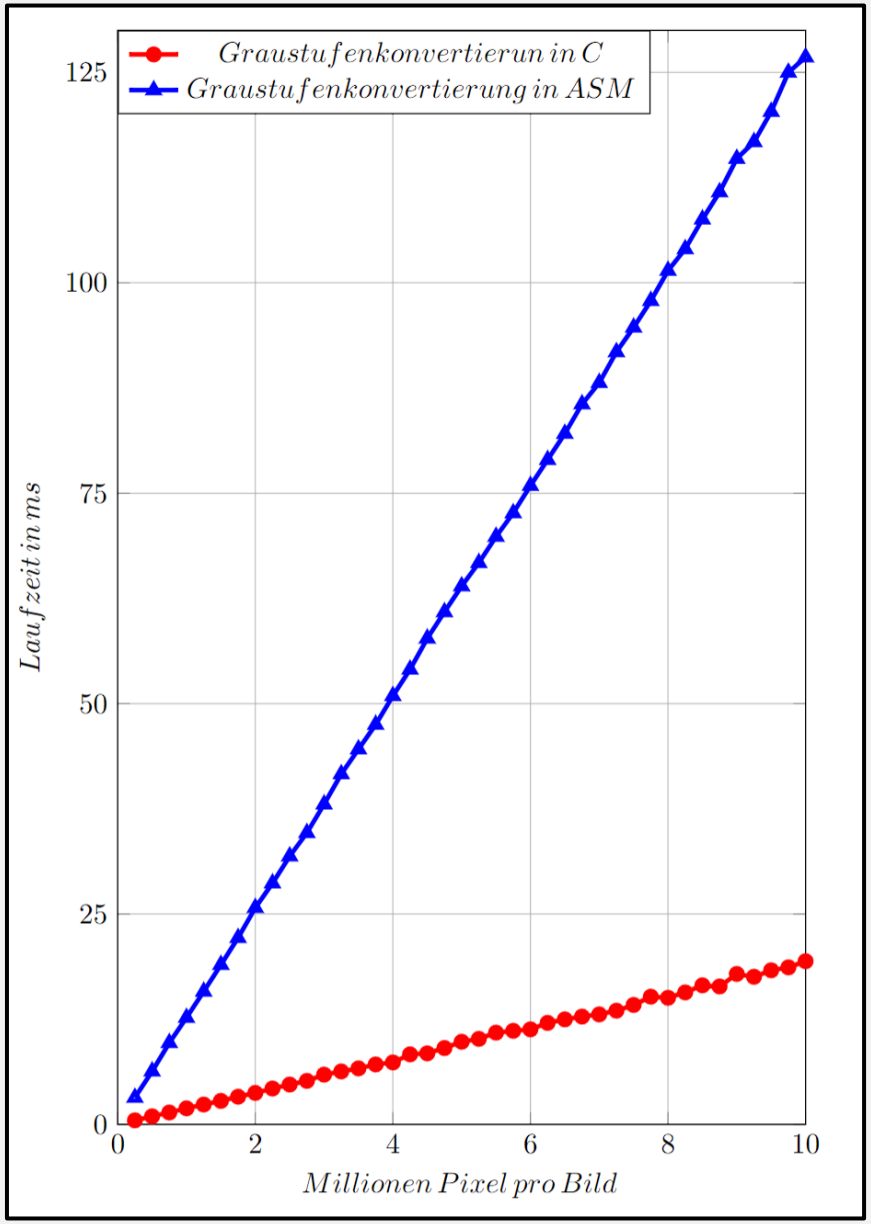
BENCHMARKING: WURZELFUNKTIONEN

- Durchschnitt von jeweils 10^9 Wurzeloperationen
  - ohne SIMD: 10^9 Ausführungen
  - mit SIMD: eine Ausführung mit Array der Länge 10^9
- sqrt()-Funktion der C-Standard-Bibliothek als Vergleichswert
- SIMD realisiert durch C-Intrinsics



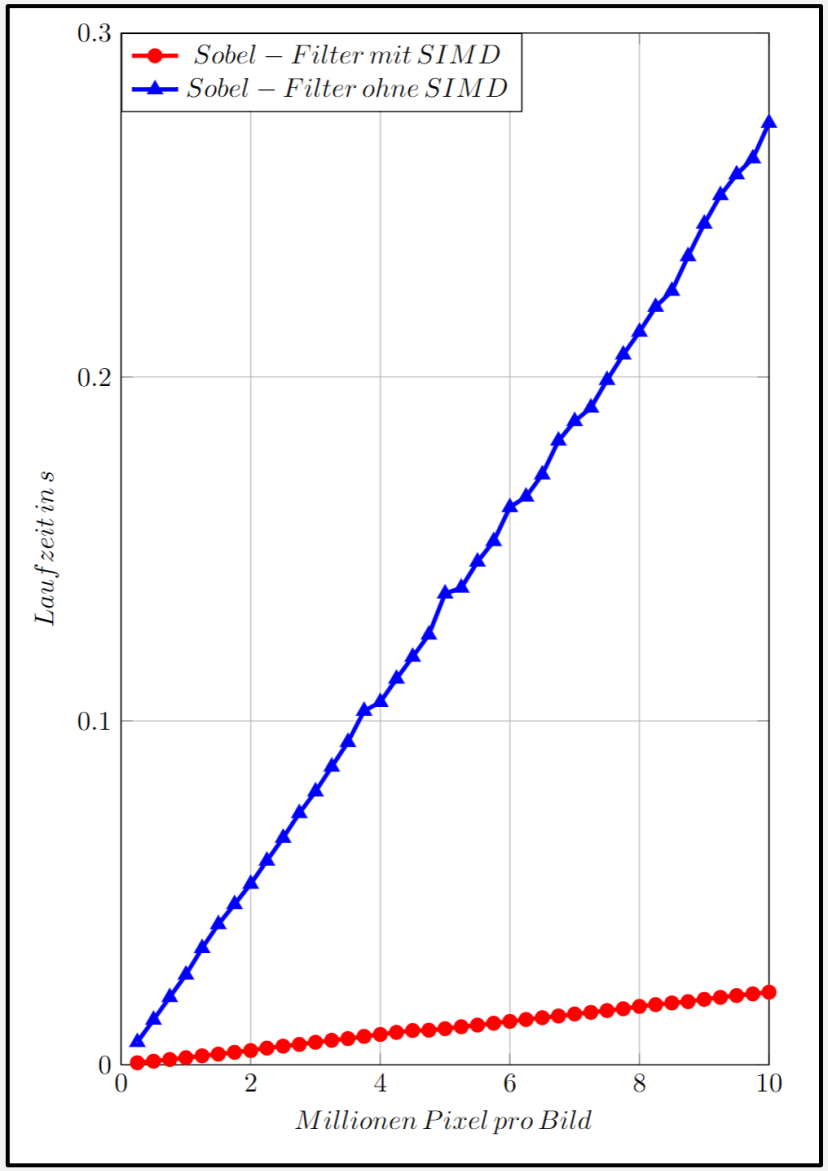
BENCHMARKING: GRAUSTUFENKONVERTIERUNG

- lineares Wachstum beider Implementationen
  - Komplexitätsklasse O(n)
- einfache C Implementation deutlich schneller als unsere Assembler Version
  - ca. um Faktor 6,5 schneller
  - aufgrund von Compileroptimierungen mit –O3



BENCHMARKING: IMPLEMENTIERUNG

- beide Versionen mit selber Graustufenkonvertierung
- Unterschied im eigentlichen Sobel-Filter
  - ➢ auch hier lineares Wachstum
  - ➢ SIMD-Version ca. um Faktor 13 schneller
  - ➢ SIMD realisiert durch C-Intrinsics
- Verarbeitung von 8 Pixeln parallel sowohl in horizontalen als auch in vertikalen Kanten



ZUSAMMENFASSUNG

Erreichte Ziele

- Basis Funktionalität
- Gute Performance

Mögliche Verbesserung

- Dynamische Bestimmung der Graustufenwerte
- Option für weicheren Filter

QUELLEN

- Bild 1: https://images.pexels.com/photos/439391/pexels-photo-439391.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2 01.01.2014
- Bild 2: https://www.cs.cornell.edu/courses/cs664/2003fa/images/, 20.07.2024, Prof. Dan Huttenlocher
- Bild 3: https://pixabay.com/photos/plant-ice-frost-cold-winter-8612513/, 20.07.2024