Helligkeit und Kontrast (A207)

Aufgabenbereich Bildverarbeitung

Überblick

- Problemstellung
- Lösungsansätze und Optimierungen
- Performanzanalyse
- Korrektheit und Genauigkeit
- Zusammenfassung und Ausblick

Problemstellung

1. PPM-Bild lesen
2. Graustufenkonvertierung
3. Helligkeitsanpassung
   - $l \in [-255, 255]$
4. Kontrastanpassung
   - $k \in [-255.0, 255.0]$
5. PGM-Bild speichern

$$D = \frac{a \cdot R + b \cdot G + c \cdot B}{a + b + c}$$

$$Q_{(x,y)} = D$$

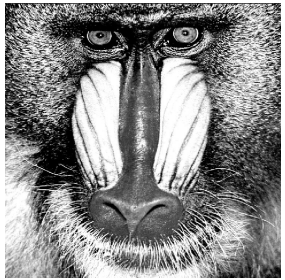
$$Q'_{(x,y)} = \text{clamp}_0^{255}\left(Q_{(x,y)} + l\right)$$

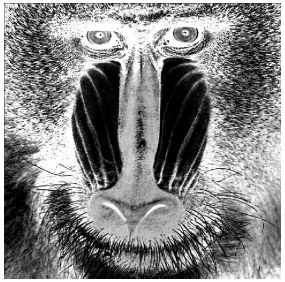
$$\sigma^2 = \text{Var}[Q'] = \frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \left(Q'_{(x,y)} - \mu\right)^2$$

$$Q''_{(x,y)} = \text{clamp}_0^{255}\left(\frac{k}{\sigma} \cdot Q'_{(x,y)} + (1 - \frac{k}{\sigma}) \cdot \mu\right)$$

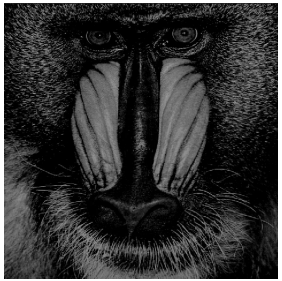
Helligkeit und Kontrast (Breithaupt, Sommersguter)

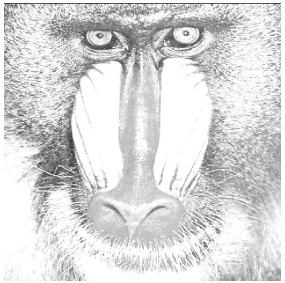
Beispielausgaben des Programms



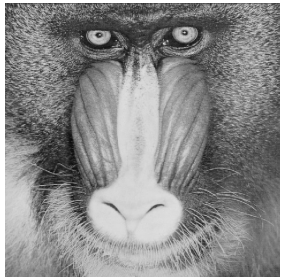


Kontrast

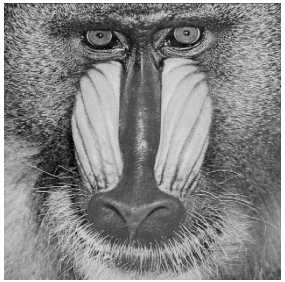




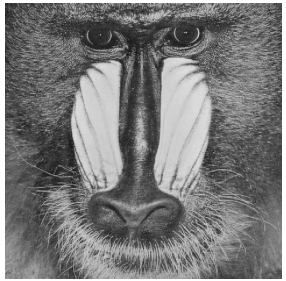
Helligkeit



Rgb



r**G**b



rg**B**

Optimierung Graustufenkonvertierung

- Ziel: Instruktionen innerhalb einer Schleifeniteration minimieren, teure Operationen (Division) vermeiden

$$D = \frac{a \cdot R + b \cdot G + c \cdot B}{a + b + c}$$

Optimierung Graustufenkonvertierung

- Ziel: Instruktionen innerhalb einer Schleifeniteration minimieren, teure Operationen (Division) vermeiden

$$D = (a \cdot R + b \cdot G + c \cdot B) \cdot \frac{1}{a + b + c}$$

Optimierung Graustufenkonvertierung

- Ziel: Instruktionen innerhalb einer Schleifeniteration minimieren, teure Operationen (Division) vermeiden

$$D = \boxed{\frac{a}{a+b+c}} \cdot R + \boxed{\frac{b}{a+b+c}} \cdot G + \boxed{\frac{c}{a+b+c}} \cdot B$$

- Nur drei Multiplikationen und drei Additionen pro Pixel

Lösung 1: C-SISD, ASM-SISD

- “Naiver” Ansatz
- Minimum an Instruktionen
- ASM:
  - Speicherzugriffe vermeiden
  - Software-Pipelining

S1

$$D = \frac{a \cdot R + b \cdot G + c \cdot B}{a + b + c}$$

$$Q_{(x,y)} = D$$

$$Q'_{(x,y)} = \text{clamp}_0^{255}\left(Q_{(x,y)} + l\right)$$

S2

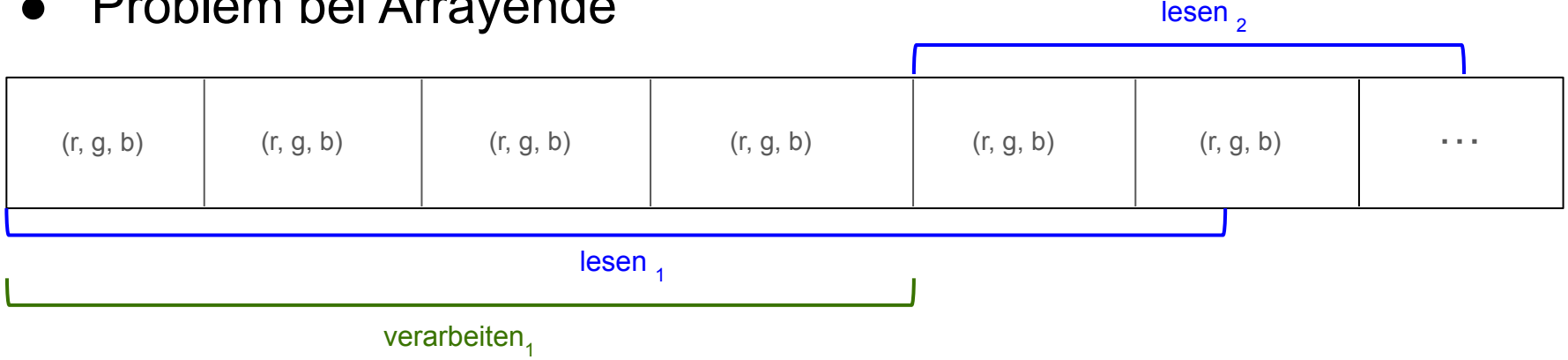
$$\sigma^2 = \text{Var}[Q'] = \frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \left(Q'_{(x,y)} - \mu\right)^2$$

S3

$$Q''_{(x,y)} = \text{clamp}_0^{255}\left(\frac{k}{\sigma} \cdot Q'_{(x,y)} + (1 - \frac{k}{\sigma}) \cdot \mu\right)$$

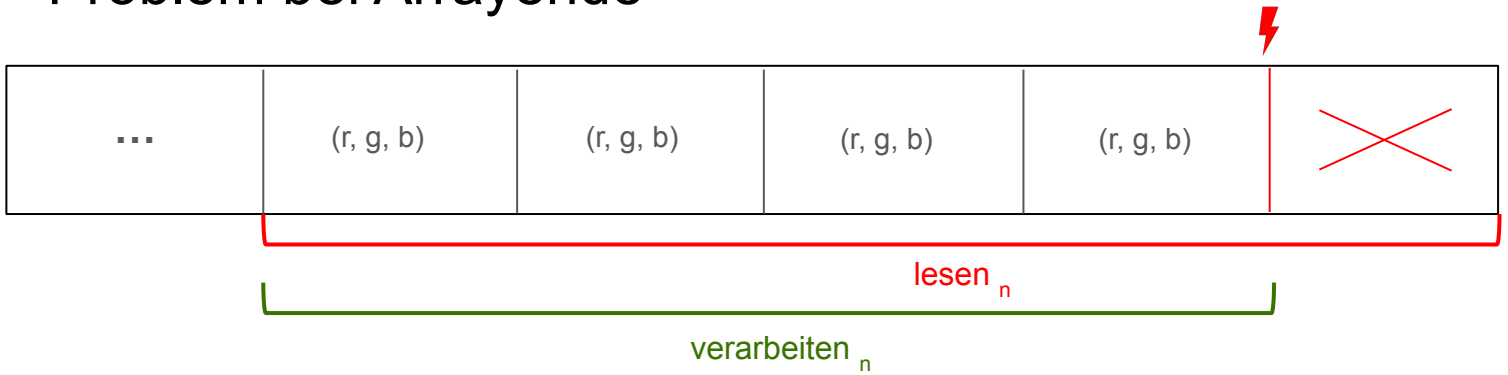
Lösung 2: C-SIMD

- Problem Graustufenkonvertierung
  - Ziel: [RRRR], [BBBB], [GGGG]
  - mit: [aaaa], [bbbb], [cccc]
- 16 Bytes lesen, zwölf verarbeiten
- Problem bei Arrayende



Lösung 2: C-SIMD

- Problem Graustufenkonvertierung
  - Ziel: [RRRR], [BBBB], [GGGG]
  - mit: [aaaa], [bbbb], [cccc]
- 16 Bytes lesen, zwölf verarbeiten
- Problem bei Arrayende



Lösung 3: ASM-SIMD

- Zusätzlich zu C-SIMD: verarbeiten von 16 Bytes parallel bei Sigma-Berechnung und Kontrastanpassung

- AVX-Optimierung bei Register-Shuffle:
  - Drei statt fünf (1) bzw. vier statt sieben (2, 3) Instruktionen
  - **vpshufb** xmm1, xmm2, xmm3 statt **mov** und **pshufb**

Lösung 4: C-SISD Multithreaded

- OpenMP-Library
- Schleifen als parallel markiert
  - **#pragma** omp parallel for
- “parallel for reduction”: keine Race-Condition-Behandlung notwendig

Wurzelberechnung

1. Math-Library

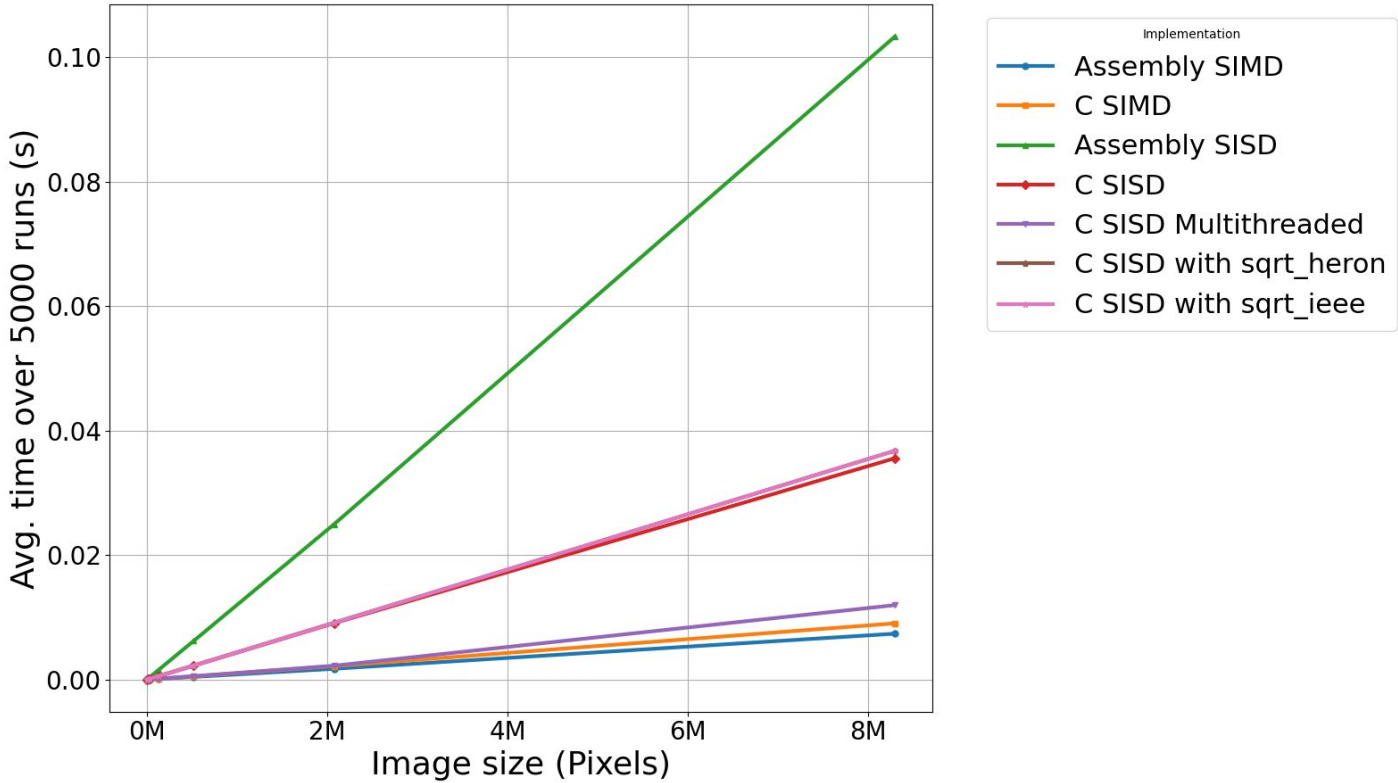
2. Heron-Verfahren
   - Sieben Iterationen

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$$

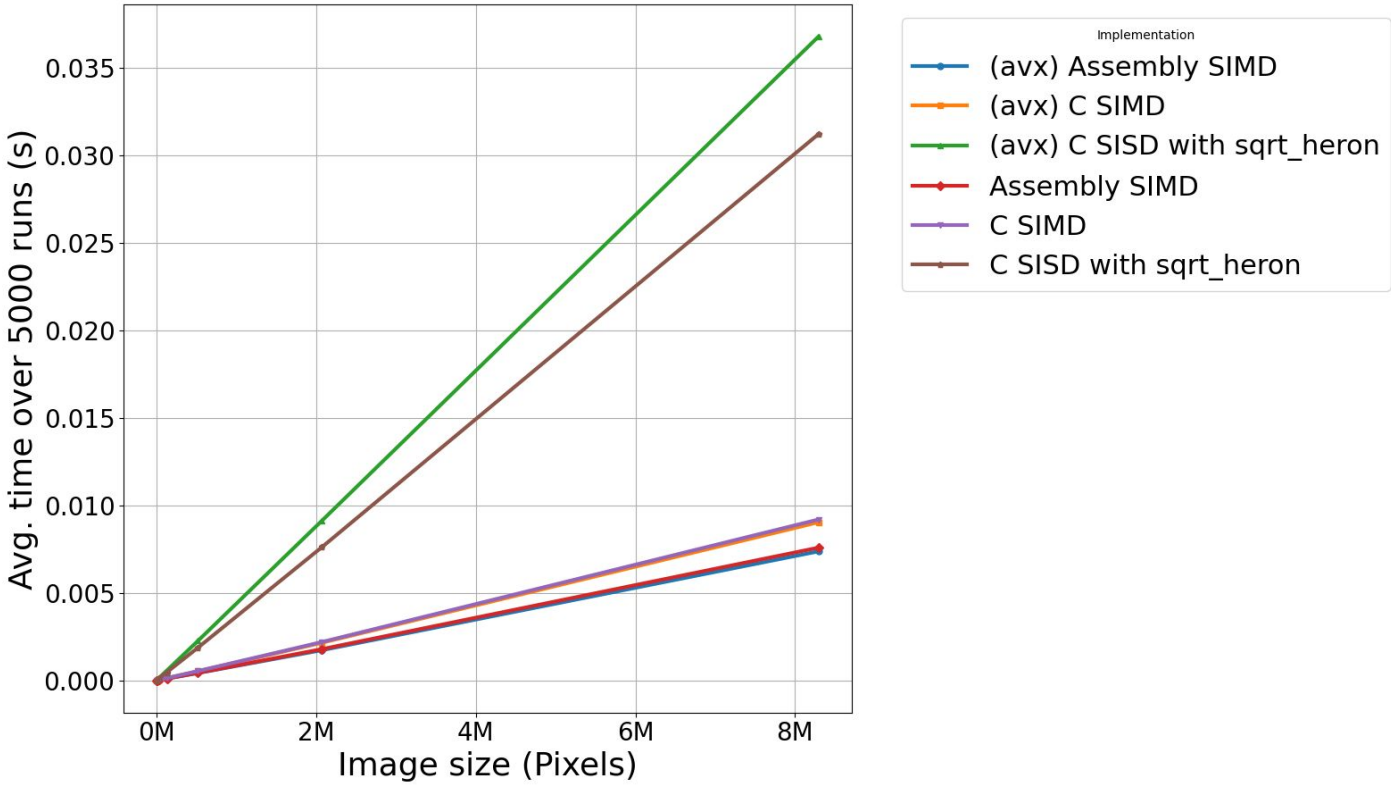
   - AVX-Optimierung:
     vdivss xmm1, xmm2, xmm3 spart ein **movss**

3. IEEE 754-basierend
   - float als int interpretieren, eine Subtraktion, eine Addition, drei Bitshifts

Performanzanalyse (-O3, -fno-unroll-loops)



Performanzanalyse (-O3, -fno-unroll-loops)



Korrektheit und Genauigkeit

- Tests

```
./BrightnessAndContrast.out: Running tests with Assembly SIMD as reference implementation...
[Test passed] C SIMD (max. delta: 0, diff. pixels: 0)
[Test passed] Assembly SISD (max. delta: 1, diff. pixels: 4333)
[Test passed] C SISD (max. delta: 1, diff. pixels: 4333)
[Test passed] C SISD Multithreaded (max. delta: 0, max. diff. pixels: 0)
[Test passed] C SISD with sqrt_heron (max. delta: 1, diff. pixels: 15469)
[Test passed] C SISD with sqrt_ieee (max. delta: 1, diff. pixels: 35538)
```

- Abweichung von eins “gültig” aufgrund von float-Ungenauigkeiten

Genauigkeit

- Alle Berechnungen in floats, Ergebnisse in Integer runden
- Runden in ASM: cvtss2si
  - Rundet nach CPU rounding mode - default: “Round to Nearest, Ties to Even”
  - 68.5 → 68, 69.5 → 70
  - Gleiches Verhalten in C: rintf() (statt round())

- Mittelwert aufsummieren: großer Wert + kleiner Wert, ungenau
  - Verschiedene Ergebnisse aufgrund der verschiedenen Summationsreihenfolgen mit SIMD, SISD und Multithreaded

Genauigkeit

- Größte Genauigkeit: C-SISD Multithreaded
  - Teilsummen “mittelgroß”, somit Summanden gleicher Größenordnung
  - Auswirkung bei Extremwerten (z.B. Helligkeit = Kontrast = 255):

```
./BrightnessAndContrast.out: Running tests with Assembly SIMD as reference implementation...
[Test passed] C SIMD (max. delta: 0, diff. pixels: 0)
[Test passed] Assembly SISD (max. delta: 1, diff. pixels: 262144)
[Test passed] C SISD (max. delta: 1, diff. pixels: 262144)
[Test failed] C SISD Multithreaded: Value mismatch at index 0 - expected: 0, actual: 255
[Test passed] C SISD with sqrt_heron (max. delta: 1, diff. pixels: 262144)
[Test failed] C SISD with sqrt_ieee: Value mismatch at index 0 - expected: 0, actual: 11
```

Genauigkeit (C-SISD Multithreaded)

- Helligkeit 255, Kontrast > 0
  - Durchschnitt = 255, Sigma = 0

$$\sigma^2 = \mathrm{Var}[Q'] = \frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \left(Q'_{(x,y)} - \mu\right)^2$$

$$Q''_{(x,y)} = \mathrm{clamp}_0^{255}\left(\frac{k}{\sigma} \cdot Q'_{(x,y)} + (1 - \frac{k}{\sigma}) \cdot \mu\right)$$

- float k_over_sigma = contrast / sqrt(sigma);
- float adjusted_avg = ((1.0f - k_over_sigma) * avg);

Helligkeit und Kontrast (Breithaupt, Sommersguter)

Genauigkeit (C-SISD Multithreaded)

- Helligkeit 255, Kontrast > 0
  - Durchschnitt = 255, Sigma = 0

$$\sigma^2 = \mathrm{Var}[Q'] = \frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \left(Q'_{(x,y)} - \mu\right)^2$$

$$Q''_{(x,y)} = \mathrm{clamp}_0^{255}\left(\frac{k}{\sigma} \cdot Q'_{(x,y)} + (1 - \frac{k}{\sigma}) \cdot \mu\right)$$

- float k_over_sigma = contrast / sqrt(sigma);
  → Infinity
- float adjusted_avg = ((1.0f - k_over_sigma) * avg);
  → neg. Infinity

- Q'' = NaN; cast zu int → 255

Helligkeit und Kontrast (Breithaupt, Sommersguter)

Genauigkeit (SISD, SIMD)

- Helligkeit 255, Kontrast > 0
  - Durchschnitt ~ 255.7, Sigma ~ 0.5

$$\sigma^2 = \mathrm{Var}[Q'] = \frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \left(Q'_{(x,y)} - \mu\right)^2$$

$$Q''_{(x,y)} = \mathrm{clamp}_0^{255}\left(\frac{k}{\sigma} \cdot Q'_{(x,y)} + (1 - \frac{k}{\sigma}) \cdot \mu\right)$$

- float k_over_sigma = contrast / sqrt(sigma);
  → ~ 340
- float adjusted_avg = ((1.0f - k_over_sigma) * avg);
  → ~ -86730

- Q'' ~ 0.x → 0/1

Helligkeit und Kontrast (Breithaupt, Sommersguter)

Lösungsansätze

- Mit doubles rechnen
  - “verschiebt” Problem nur
  - SIMD in aktueller Form nicht möglich (benötigt 2x so viele Register)
- Überlegungen hinsichtlich Extremwerten
  - Kontrast bei extremen Helligkeitswerten sinnvoll?

Zusammenfassung und Ausblick

- Weitere Optimierungsmöglichkeiten:
  - Loop-Unrolling inkl. Optimierungen für Software-Pipelining
  - Mit 265-Bit ymm- oder 512-Bit zmm-Registern Parallelität u. Performance erhöhen

Vielen Dank für die Aufmerksamkeit!