Gammakorrektur

Effiziente Erzeugung eines gammakorrigierten Graustufenbildes

1. Umwandlung in Graustufen mit gewichtetem Durchschnitt

2. Gammakorrektur des Graustufenbildes zur Anpassung der Helligkeit des Bildes

1. Umwandlung in Graustufen

- Berechnen des Graustufenfilters
- Formel: $D = \frac{a \cdot R + b \cdot G + c \cdot B}{a + b + c}$ $\qquad p_{x,y} = \begin{pmatrix} D \\ D \\ D \end{pmatrix}$
- Vorgabe für Optimierung: a + b + c = 256
- Auge nimmt **grün** heller war als **rot** und **rot** heller als **blau**
- Optimale Gewichtung: a = 77, b = 151, c = 28

Beispielbilder mit unterschiedlichen Gewichtungen des Graustufenfilters

Farbbild

a=20, b=6, c=230

a = 77, b = 151, c = 28

Optimierung mit SIMD

1. Einlesen aus dem Speicher

Stelle im Xmm-Register:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte Werte im Speicher	12	144	222	44	66	124	185	129	244	255	9	37	131	181	179	0

2. Nach Farbwerten in Xmm-Register aufteilen

erstes Xmm-Register			12			44		185				255			131	
zweites Xmm-Register			144			66		129				9			181	
drittes Xmm-Register			222			124		244				37			179	

Xmm-Register mit den Koeffizienten

Xmm-Register für a				77				77				77				77
Xmm-Register für b				151				151				151				151
Xmm-Register für c				28				28				28				28

Multiplikation und Speichern des Ergebnisses

3. Multiplikation mit den Koeffizienten

Multiplikation mit a		3	156		13	60	55	165			76	179		39	103	
Multiplikation mit b		84	240		38	238	76	23			5	79		106	195	
Multiplikation mit c		24	72		13	144	26	176			4	12		19	148	

4. Addition der Xmm-Register

Ergebnis der Addition		112	212		65	186	158	108			86	14		165	190	

5. Extrahieren und Duplizieren der Ergebnisse

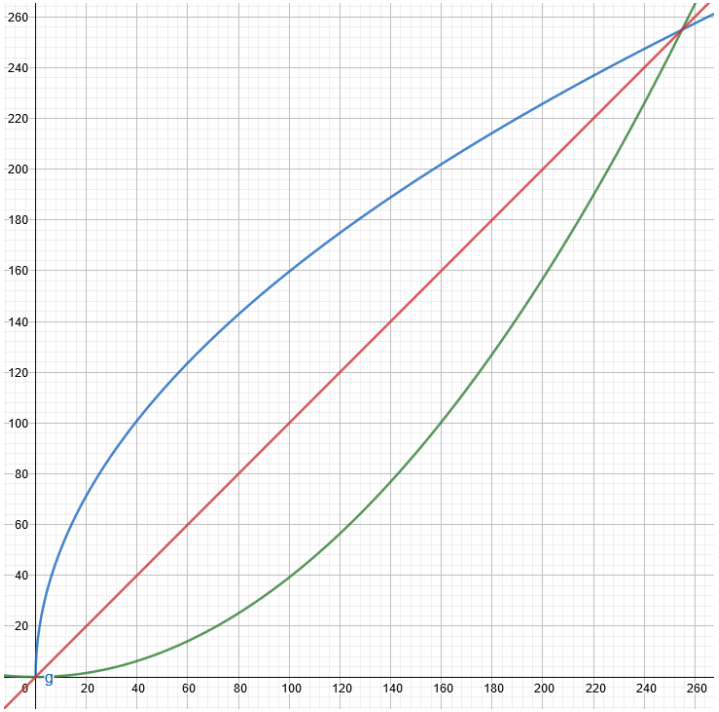
Ergebnis Xmm-Register	112	112	112	65	65	65	158	158	158	86	86	86	165	165	165	0

Performance der SIMD-Optimierung

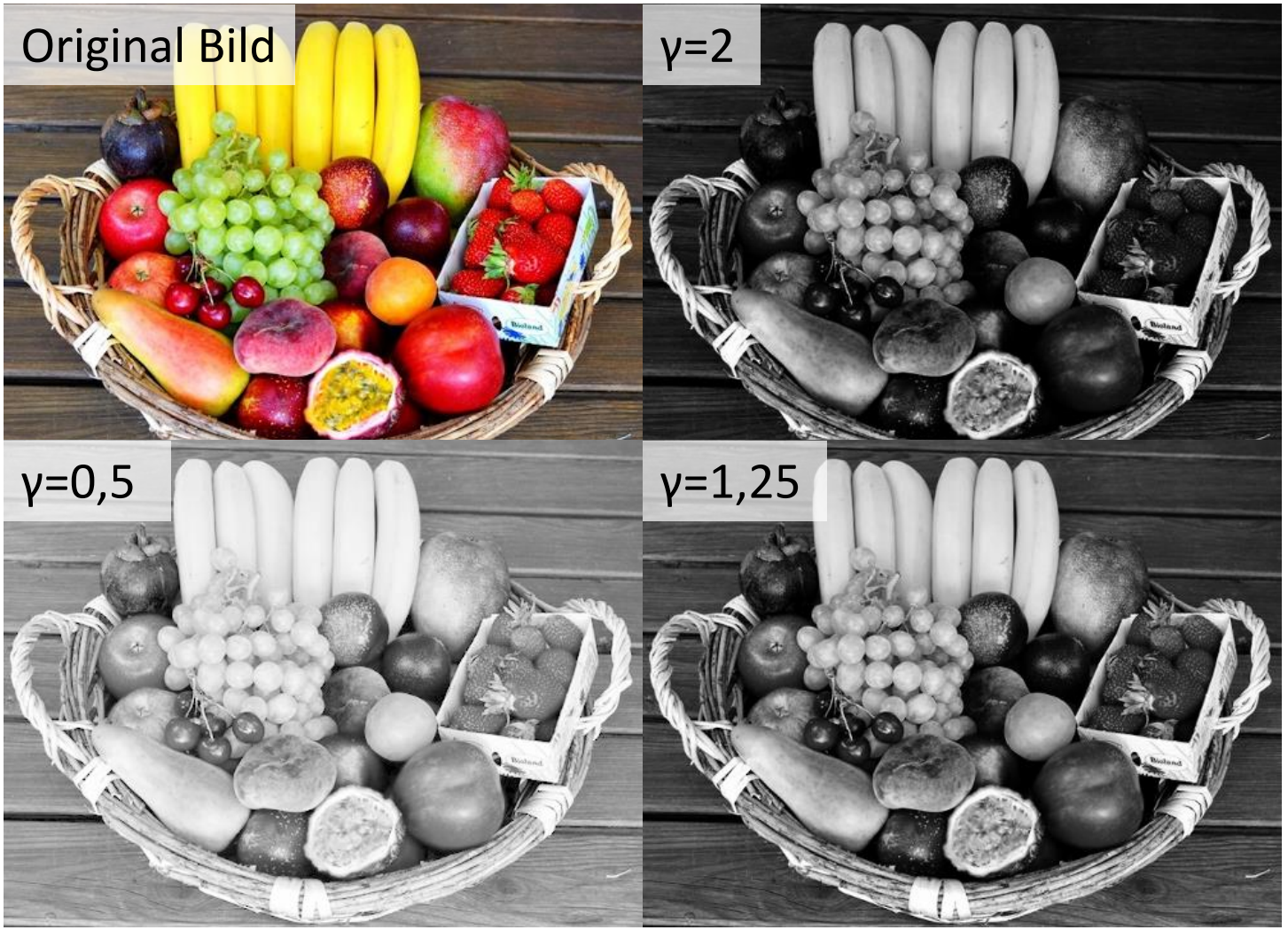
- Ohne Alignement 48% langsamer als die Standardimplementierung (bei 2000x2000 Pixeln)
- Mit 16 Byte Alignment durch Hinzufügen eines 16. null Bytes 66% schneller als die Standardimplementierung
- Grund: Reduktion von Shiftbefehlen und schnellerer Speicherzugriff

2. Gammakorrektur

- Gezielter Einfluss auf die Helligkeit des Bildes
- Formel: $p_{neu} = \left(\frac{p_{alt}}{255}\right)^{\gamma} \cdot 255$
- Monoton steigend
- Für γ < 1 oberhalb von f(x) = x → größere Werte (heller)
- Für γ > 1 unterhalb von f(x) = x → kleinere Werte (dunkler)



Bilder mit unterschiedlichen γ-Werten



Optimierung der Gammakorrektur

$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$

Optimierung der Gammakorrektur

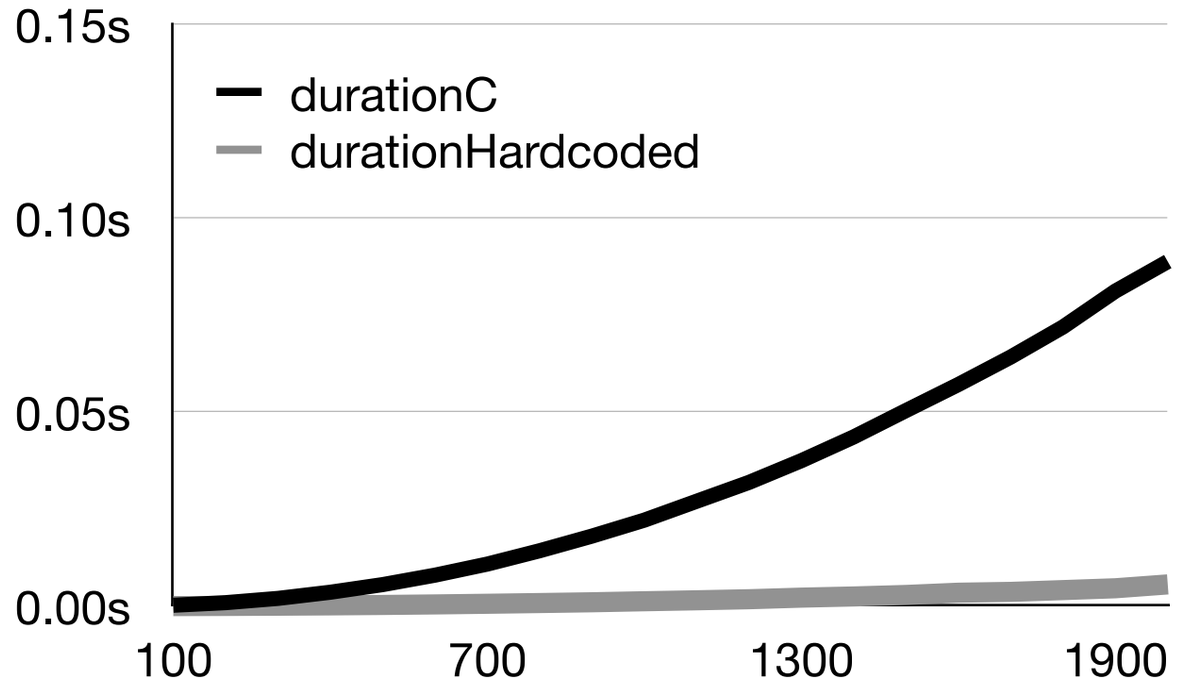
$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$

- **„Nur“ 65.000 Funktionen**

Optimierung der Gammakorrektur

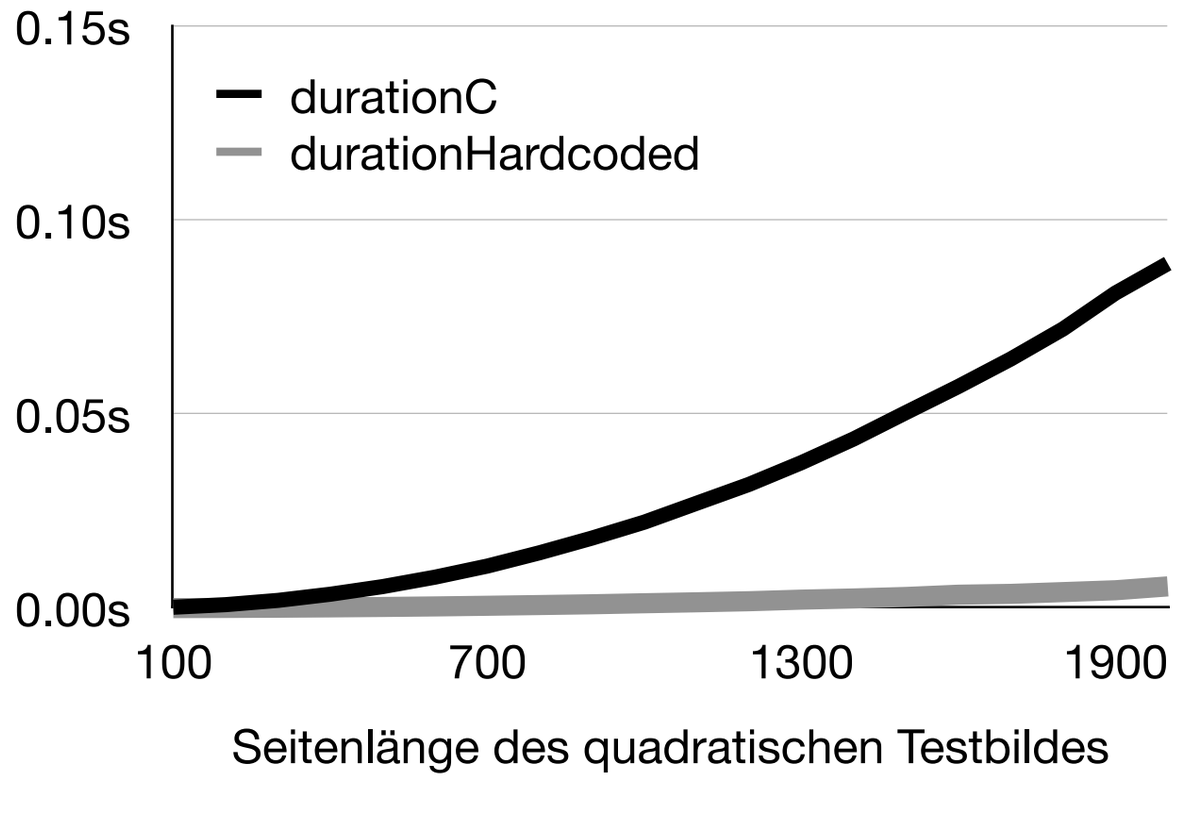
$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$

- **„Nur“ 65.000 Funktionen**



Optimierung der Gammakorrektur

$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$

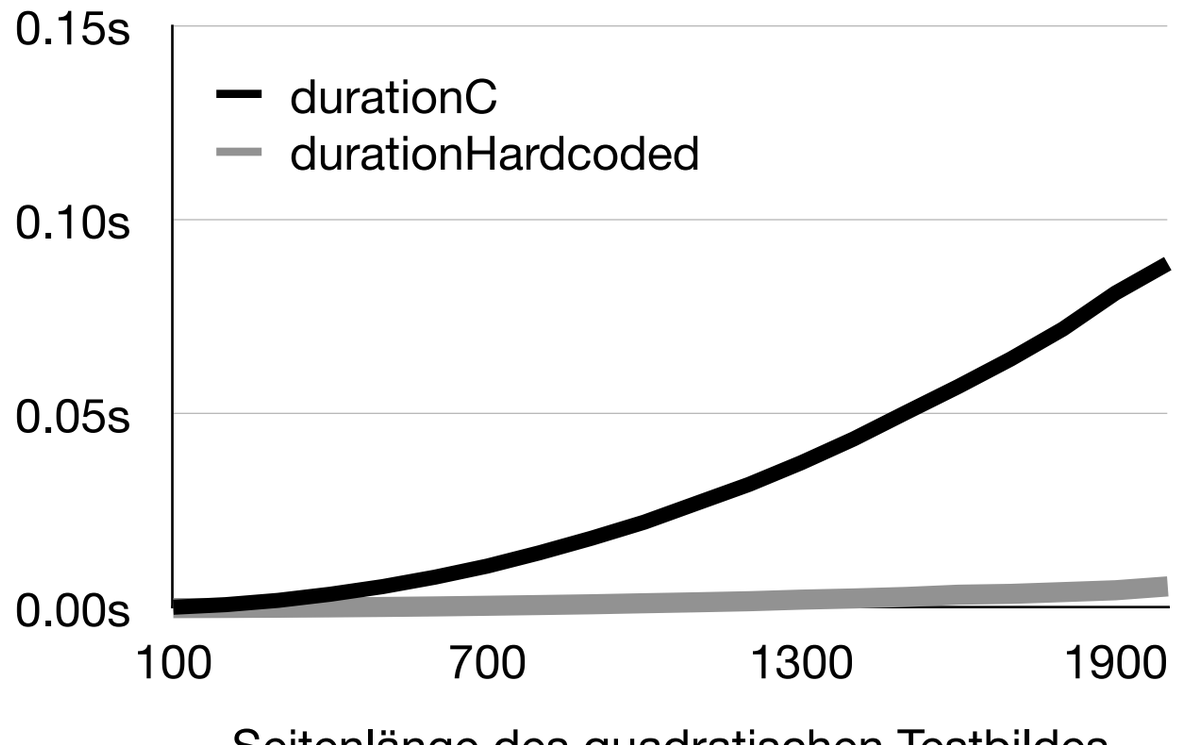


- **„Nur“ 65.000 Funktionen**

Ziel: Schöner und Schneller

Optimierung der Gammakorrektur

$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$



- **„Nur“ 65.000 Funktionen**

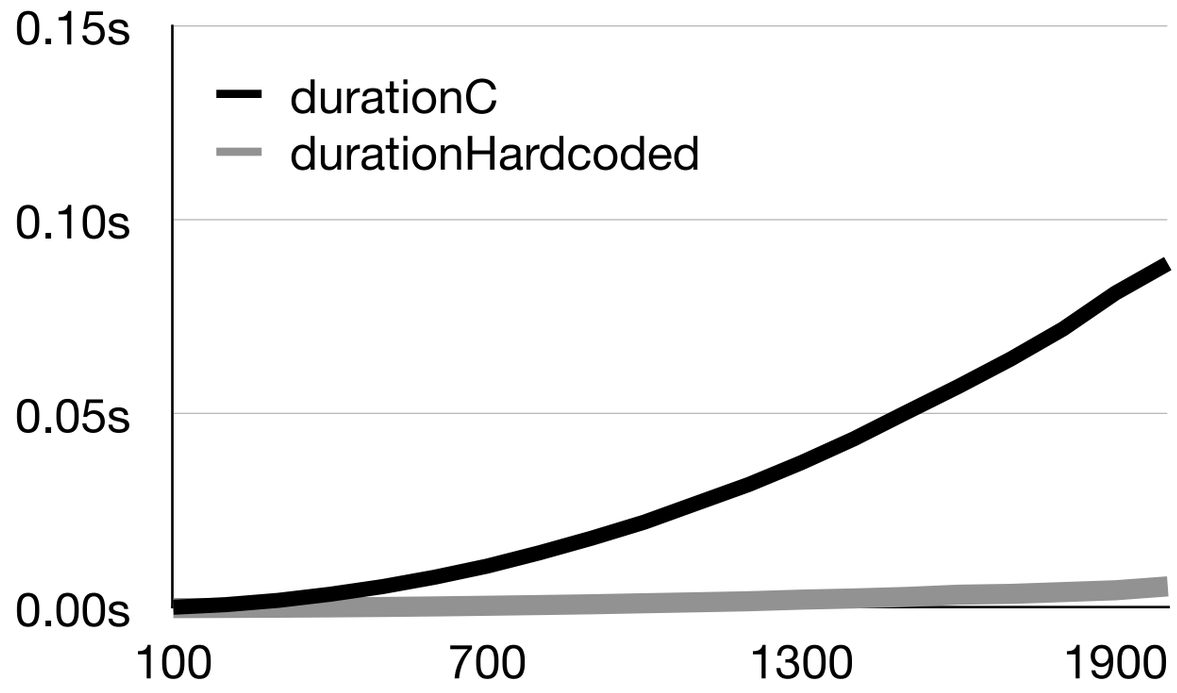
Ziel: Schöner und Schneller

wähle $z, n \in \mathbb{N} : \frac{z}{n} = \gamma$

$$\left(\frac{x}{255}\right)^{\gamma} = \sqrt[n]{\left(\frac{x}{255}\right)^{z}}$$

Optimierung der Gammakorrektur

$$f(x) = \left(\frac{x}{255}\right)^{\gamma} \cdot 255$$



- **„Nur“ 65.000 Funktionen**

Ziel: Schöner und Schneller

wähle $z, n \in \mathbb{N} : \frac{z}{n} = \gamma$

$$\left(\frac{x}{255}\right)^{\gamma} = \sqrt[n]{\left(\frac{x}{255}\right)^{z}}$$

Auf Exponenten rechnen, sei $l(z) = \ln\left(\frac{z}{255}\right)$**:**

Optimierung der Gammakorrektur

$$f(x) = \max\{n \in \mathbb{N} : l(n) \leq l(x) \cdot \gamma\}$$

Optimierung der Gammakorrektur

$$f(x) = \max\{n \in \mathbb{N} : l(n) \leq l(x) \cdot \gamma\}$$

Seien Werte von $l(n)$ für $0 < n \leq 255$ gespeichert.

Optimierung der Gammakorrektur

$$f(x) = \max\{n \in \mathbb{N} : l(n) \leq l(x) \cdot \gamma\}$$

Seien Werte von $l(n)$ für $0 < n \leq 255$ gespeichert.

1. Berechne $l(x) \cdot \gamma$

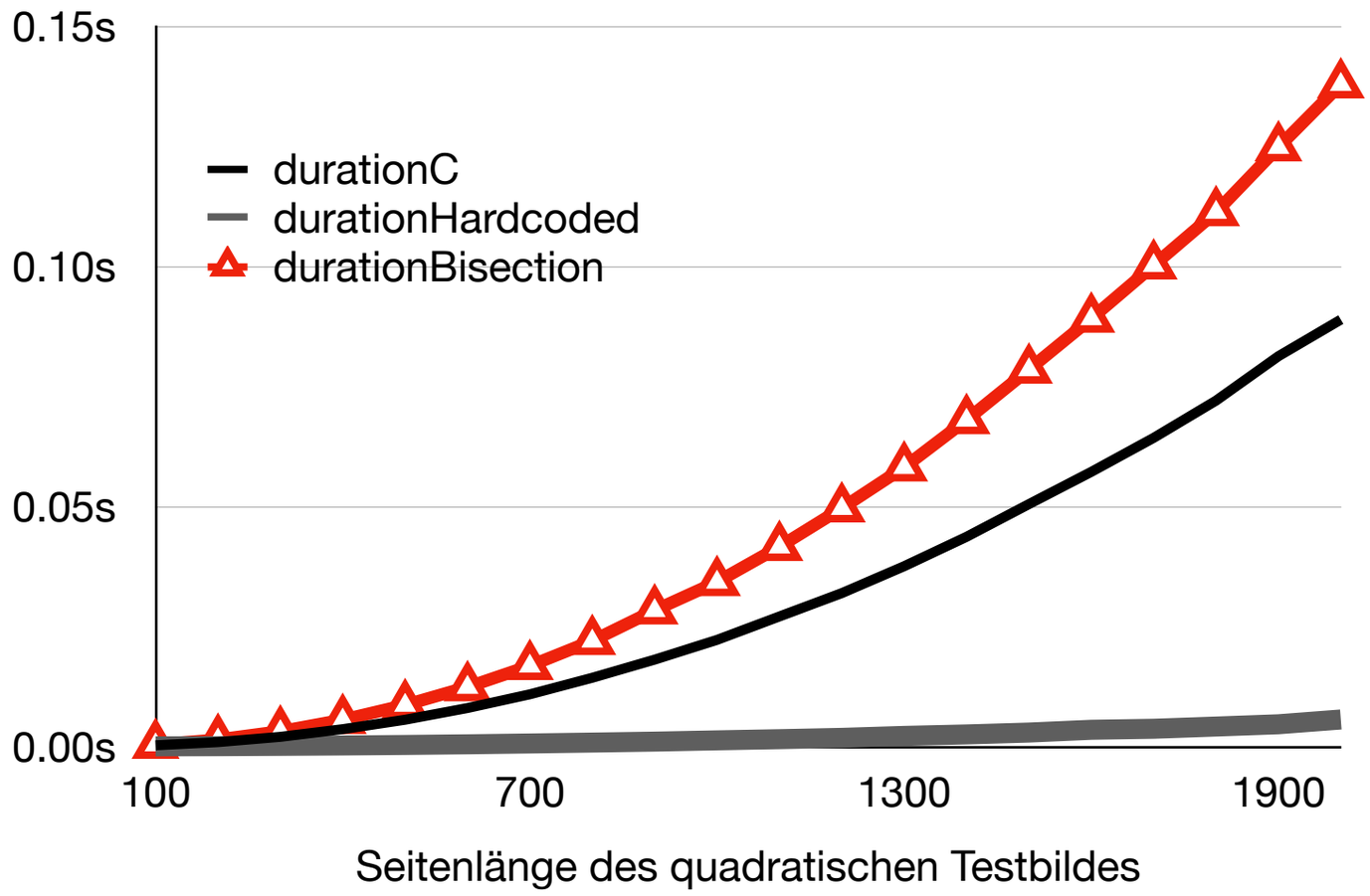
2. Suche binär $f(x)$

(Max. $\log_2(256) = 8$ Vergleiche.)

Optimierung der Gammakorrektur

$$f(x) = \max\{n \in \mathbb{N} : l(n) \leq l(x) \cdot \gamma\}$$

Seien Werte von $l(n)$ für $0 < n \leq 255$ gespeichert.



1. Berechne $l(x) \cdot \gamma$

2. Suche binär $f(x)$

(Max. $\log_2(256) = 8$ Vergleiche.)

Optimierung der Gammakorrektur

Schneller: vorher alle Werte für $f(x)$ berechnen.

Optimierung der Gammakorrektur

Schneller: vorher alle Werte für $f(x)$ berechnen.

Wir wissen: $f(x)$ steigt monoton. -> $f(x) = f(x-1) + d$

Optimierung der Gammakorrektur

Schneller: vorher alle Werte für $f(x)$ berechnen.

Wir wissen: $f(x)$ steigt monoton. -> $f(x) = f(x-1) + d$

mit $d = |\{n \in \mathbb{N} : n > f(x-1) \ \wedge \ l(n) \leq l(x) \cdot \gamma\}|$

Optimierung der Gammakorrektur

Schneller: vorher alle Werte für $f(x)$ berechnen.

Wir wissen: $f(x)$ steigt monoton. -> $f(x) = f(x-1) + d$

mit $d = |\{n \in \mathbb{N} : n > f(x-1) \ \wedge \ l(n) \leq l(x) \cdot \gamma\}|$

Beispiel mit $\gamma = 0.9$

Stets gilt f(0) = 0

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5	-4.9	
2	-4.8	-4.4	
3	-4.4	-4	
4	-4.2	-3.7	
5	-3.9	-3.5	
6	-3.7	…	
7	-3.6	…	
…	-3.4	…	
…	…	…	

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5 ≤	-4.9	f(1) = f(0)+|{1}| = 1
2	-4.8	-4.4	
3	-4.4	-4	
4	-4.2	-3.7	
5	-3.9	-3.5	
6	-3.7	…	
7	-3.6	…	
…	-3.4	…	
…	…	…	

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5	-4.9	f(1) = f(0)+|{1}| = 1
2	-4.8	-4.4	f(2) = f(1)+|{2,3}| = 3
3	-4.4	-4	
4	-4.2	-3.7	
5	-3.9	-3.5	
6	-3.7	…	
7	-3.6	…	
…	-3.4	…	
…	…	…	

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5	-4.9	f(1) = f(0)+|{1}| = 1
2	-4.8	-4.4	f(2) = f(1)+|{2,3}| = 3
3	-4.4	-4	f(3) = f(2) + |{4}| = 4
4	-4.2	-3.7	
5	-3.9	-3.5	
6	-3.7	…	
7	-3.6	…	
…	-3.4	…	
…	…	…	

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5	-4.9	f(1) = f(0)+|{1}| = 1
2	-4.8	-4.4	f(2) = f(1)+|{2,3}| = 3
3	-4.4	-4	f(3) = f(2) + |{4}| = 4
4	-4.2	-3.7	f(4) = f(3) + |{5,6}| = 6
5	-3.9	-3.5	
6	-3.7	…	
7	-3.6	…	
…	-3.4	…	
…	…	…	

Optimierung der Gammakorrektur

x	I(x)	I(x)*ɣ	f(0)=0
1	-5.5	-4.9	f(1) = f(0)+|{1}| = 1
2	-4.8	-4.4	f(2) = f(1)+|{2,3}| = 3
3	-4.4	-4	f(3) = f(2) + |{4}| = 4
4	-4.2	-3.7	f(4) = f(3) + |{5,6}| = 6
5	-3.9	-3.5	f(5) = f(4) + |{7}| = 7
6	-3.7	…	…
7	-3.6	…	…
…	-3.4	…	…
…	…	…	…

Optimierung der Gammakorrektur

Algorithmus:

Spalten als Arrays

Sortiertes Durchlaufen beider Arrays mit Counter für linke Werte

510 Lesezugriffe und Vergleiche
255 Schreibzugriffe und Multiplikationen

Optimierung der Gammakorrektur

counter = 0

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	
-4.8	-4.4	
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 1

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	
-4.8	-4.4	
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 1

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = counter = 1
-4.8	-4.4	
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 2

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 3

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 3

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = counter = 3
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 4

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 4

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = counter = 4
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 5

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = 4
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 6

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = 4
-4.2	-3.7	
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 6

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = 4
-4.2	-3.7	f(4) = counter = 6
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 7

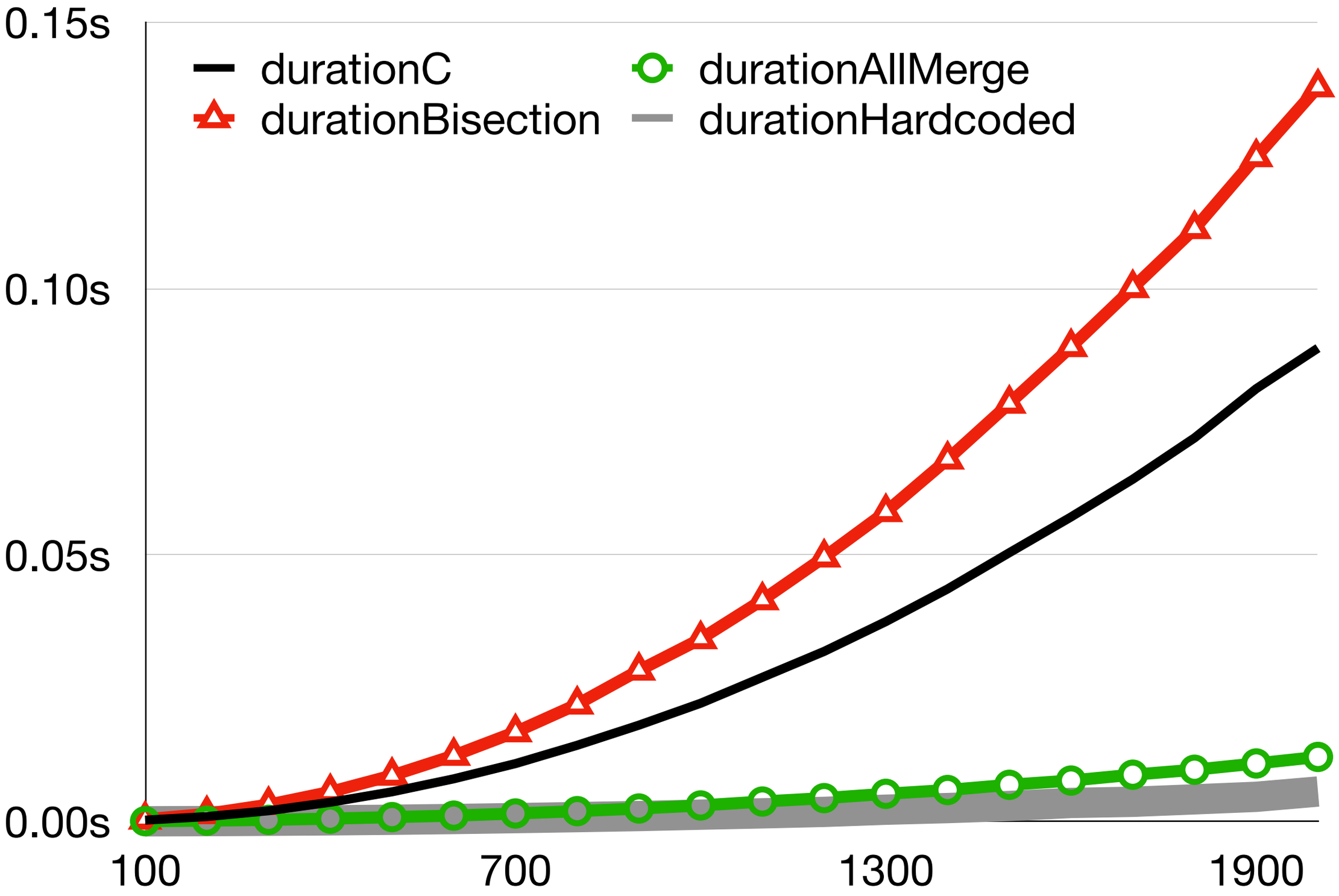
I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = 4
-4.2	-3.7	f(4) = 6
-3.9	-3.5	
-3.7	…	
-3.6	…	
-3.4	…	
…	…	

Optimierung der Gammakorrektur

counter = 7

I(x) =	I(x)*ɣ =	f(0)=0
-5.5	-4.9	f(1) = 1
-4.8	-4.4	f(2) = 3
-4.4	-4	f(3) = 4
-4.2	-3.7	f(4) = 6
-3.9	-3.5	f(5) = counter = 7
-3.7	…	…
-3.6	…	…
-3.4	…	…
…	…	…

Optimierung der Gammakorrektur



Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ **ist Anzahl Nullen vor** x**-ter 1.**

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ ist Anzahl Nullen vor x-ter 1.

f(1) = 1

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ ist Anzahl Nullen vor x-ter 1.

f(1) = 1
f(2) = 3

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ ist Anzahl Nullen vor x-ter 1.

f(1) = 1
f(2) = 3
f(3) = 4

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ ist Anzahl Nullen vor x-ter 1.

f(1) = 1
f(2) = 3
f(3) = 4
f(4) = 6

Optimierung der Gammakorrektur

Flaschenhals Hauptspeicher: Funktion in Registern?

Kodierung: counter erhöhen -> 0
Funktionswerte speichern -> 1

0 1 0 0 1 0 1 0 0 1 0 1 …

$f(x)$ ist Anzahl Nullen vor x-ter 1.

f(1) = 1
f(2) = 3
f(3) = 4
f(4) = 6
f(5) = 7

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

	f(x1)	x1	f(x2)	x2	f(x3)	x3	f(x4)	x4
Bytes		1		3		4		5

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Beispiel: Nach vierter Eins (zCounter = 4, nCounter = 6)

	f(x1)	x1	f(x2)	x2	f(x3)	x3	f(x4)	x4
Bytes	1	1	4	3		4		5
Words	257		1027		4		5	

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Beispiel: Nach vierter Eins (zCounter = 4, nCounter = 6)

	f(x1)	x1	f(x2)	x2	f(x3)	x3	f(x4)	x4
Bytes	1	1	4	3		4		5
Words	257		1027		4		5	
Mask	0	0	0	0	0xFF	0xFF	0	0

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Beispiel: Nach vierter Eins (zCounter = 4, nCounter = 6)

	f(x1)	x1	f(x2)	x2	f(x3)	x3	f(x4)	x4
Bytes	1	1	4	3		4		5
Words	257		1027		4		5	
Mask	0	0	0	0	0xFF	0xFF	0	0
&256*nCounter	0	0	0	0	6	0	0	0

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Beispiel: Nach vierter Eins (zCounter = 4, nCounter = 6)

	f(x1)	x1	f(x2)	x2	f(x3)	x3	f(x4)	x4
Bytes	1	1	4	3		4		5
Words	257		1027		4		5	
Mask	0	0	0	0	0xFF	0xFF	0	0
&256*nCounter	0	0	0	0	6	0	0	0
Mask | Bytes	1	1	4	3	6	4	0	5

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 1: SIMD

Durchlaufen der Bits:
-nCounter zählt Nullen
-zCounter zählt Einsen

Annahme: Byte vor x-Wert = 0

Bei Eins:
Words <= zCounter -> ByteMaske
Verunden mit nCounter*256
Verodern mit SIMD-Register

Leider deutlich langsamer als der Cache…

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 2: Binäre Suche

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 2: Binäre Suche

Suche für jeden x-Wert die Position, wo die x-te 1 steht.
Zähle Anzahl der Nullen bis zu dieser Position

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 2: Binäre Suche

Suche für jeden x-Wert die Position, wo die x-te 1 steht.
Zähle Anzahl der Nullen bis zu dieser Position

Dafür nötig:
Anzahl an Einsen (und Nullen) bis zum x-ten Bit.

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 2: Binäre Suche

Suche für jeden x-Wert die Position, wo die x-te 1 steht.
Zähle Anzahl der Nullen bis zu dieser Position

Dafür nötig:
Anzahl an Einsen (und Nullen) bis zum x-ten Bit.

Möglichkeit 1:
Register kopieren, verschieben, popcnt -> Zu langsam

Optimierung der Gammakorrektur

Funktion in 510 Bits darstellbar. Dekodierung?

Ansatz 2: Binäre Suche

Suche für jeden x-Wert die Position, wo die x-te 1 steht.
Zähle Anzahl der Nullen bis zu dieser Position

Dafür nötig:
Anzahl an Einsen (und Nullen) bis zum x-ten Bit.

Möglichkeit 1:
Register kopieren, verschieben, popcnt -> Zu langsam

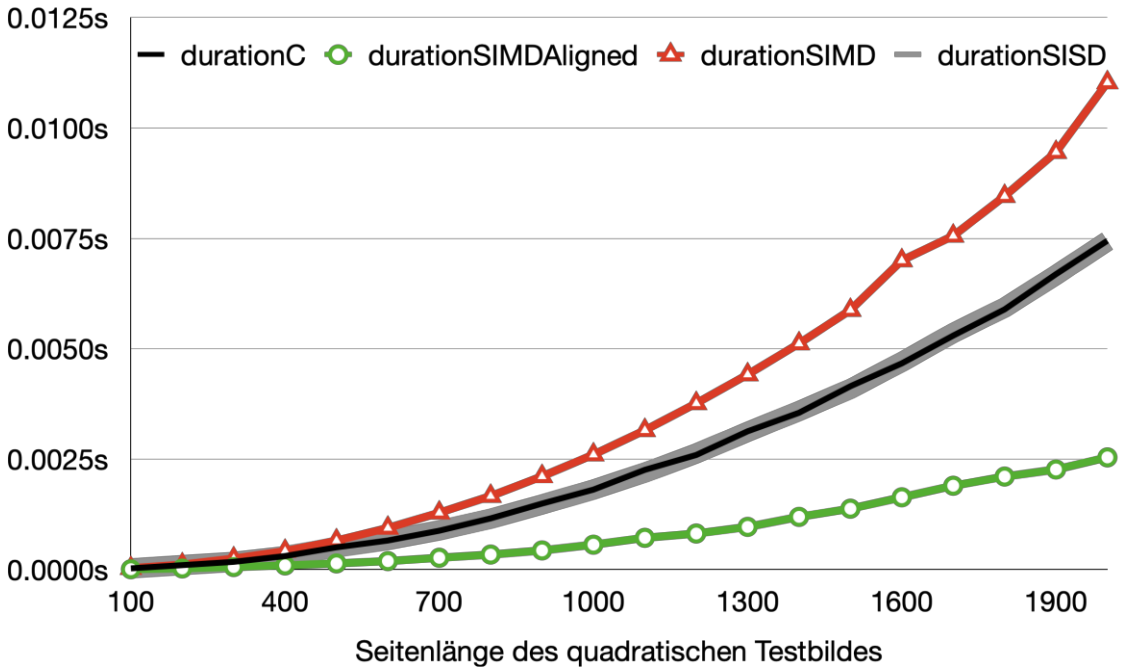
Möglichkeit 2:
Befehl, welcher gesetzte Bits bis zu einer bestimmten Position zählt -> Gibt es (noch) nicht

Funktionen unseres Programms

- Gammakorrektur auf Farbbild anwenden
- Benchmarking mit beliebigem Bild
- Erstellen von Laufzeitdiagramm im CSV-Format
- Tests mit verschiedenen Gamma Werten und Bildern
- Ausführliche Hilfsausgabe

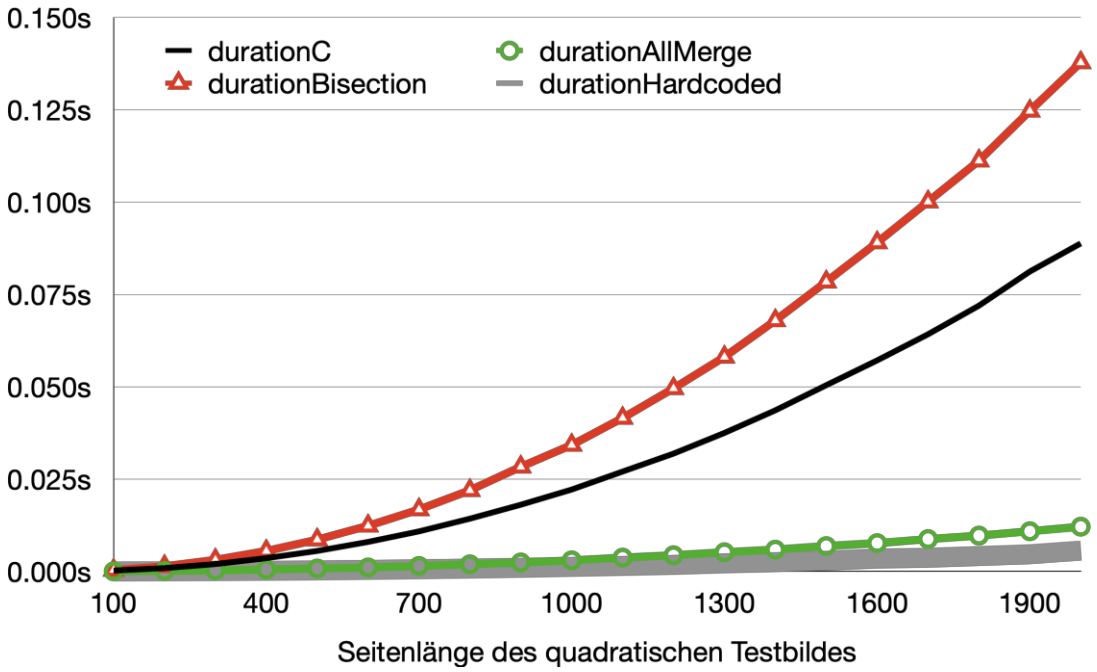
Performance

- Optimierung des Graustufenfilters:
- SIMD Berechnung
- Alignment



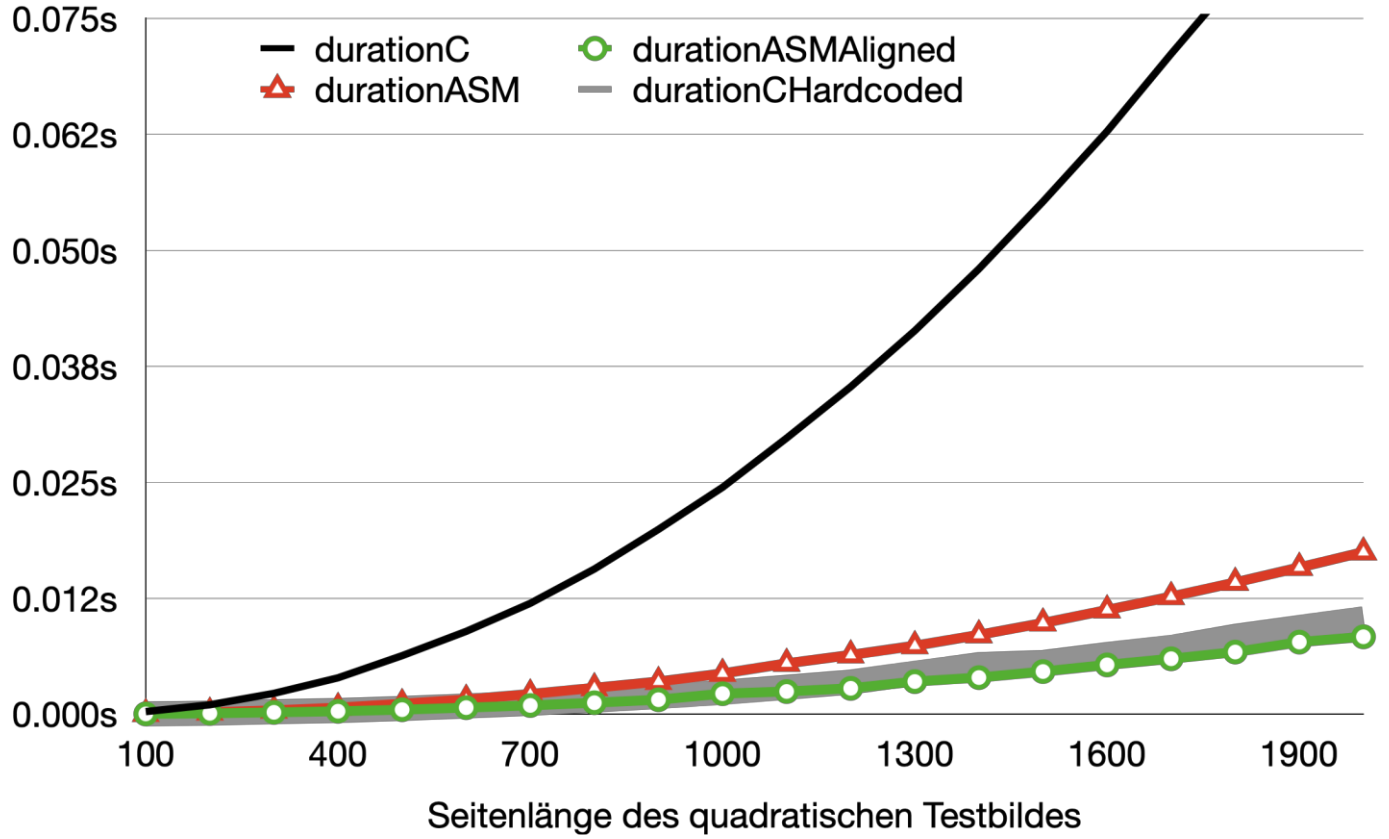
Performance

- Optimierung der Gammakorrektur:
- Hardcoded Functions
- Bisection
- Merge



Performance

- Gesamt:



Optimierungsschritte

- Graustufenfilter mit SIMD und Alignment
- Alle Funktionen mit Gamma Intervallen berechnet
- Binäre Suche zum Finden einer Funktion
- Gamma als Bruch -> Newtonverfahren
- Bisektion mit Zähler und Nenner als Exponenten
- Rechnen auf Exponenten
- Berechnen der gesamten Funktion durch Bisektion
- Berechnung durch eine einzige Schleife
- Ausblick: Zugriff auf Kodierte Funktion im Register

Zusammenfassung

- Mathematisch korrekte Berechnung für jeden Gamma-Wert
- Schneller und schöner als hardgecodete Funktionen in C
- 10 mal schneller als C-Standardimplementierung

Effizientes und robustes Programm für die Gammakorrektur mit Graustufenfilter