LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Gruppe 140 – Abgabe zu Aufgabe A208
Wintersemester 2023/24

Tianhao Gu Zhongfang Wang Julien Escaig

1 Einleitung

Das Ziel dieser Projektarbeit ist es, einen Algorithmus in C zu entwickeln, der ein farbiges Bild in ein Graustufenbild umwandelt und anschließend die Helligkeit des Graustufenbilds mithilfe der Gammakorrektur anpasst.

Als Eingabe akzeptiert Unser Programm nur PPM-Dateien des Typs P6. P6 bezieht sich dabei auf das Binärformat der Pixeldaten. Diese besteht aus einem Header mit Metadaten, worauf die Pixel-Informationen folgen. Im Datenteil der PPM Datei gibt es für jeden Pixel genau drei Werte, die jeweils die Stärke der Farben Rot, Grün und Blau speichern. Je größer der Wert desto stärker ist die Farbe in einem bestimmten Pixel vertreten. Unterhalb ist ein einfaches Beispiel eines solchen Bildes.

```
p6
4
2
255
0,0,0        255,0,0     0,255,0     0,0,255
255,255,0    255,0,255   0,255,255.  255,255,255
```

Abbildung 1: ein Beispiel für P6 PPM

Abbildung 2: erzeugt durch den Beispielcode

Die erste Phase des Projekts ist die Graustufenkodierung.Dafür verwenden wir die Formel (1) unten. Dabei wird der gewichtete Durchschnitt der Rot-, Grün- und Blau-Werte jedes Pixels ermittelt. Ein Beispiel ist in Abbildung 3 zu sehen.

$$D(x,y) = \frac{a * R + b * G + c * B}{a + b + c} \tag{1}$$

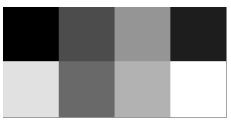

Abbildung 3: Graustufen Konvertierung von Abbildung2

Im zweiten Teil der Aufgabe befassen wir uns nun mit einem anderen Aspekt des menschlichen visuellen Systems (HVS) und zwar der Helligkeit. Diese hat einen großen Einfluss darauf, wie natürlich ein Bild auf uns Menschen wirkt. Die Gammakorrektur ändert die Helligkeit bzw. den Kontrast eines Bildes, und hängt von der Wahl des γ Parameters ab.Bei diesem Algorithmus wird die Gammakorrektur durch folgende mathematische Formel (2) bestimmt. Die Werte D'(x,y) ergeben die Intensität der Graustufenkodierung für alle Pixel (x,y). Ein kleinerer Gammawert führt zu einem helleren Bild, während ein größerer Gammawert zu einem dunkleren Bild führt.

$$D'(x,y) = \left(\frac{D(x,y)}{255}\right)^{\gamma} * 255 \tag{2}$$

Um den Effekt der Gammakorrektur zu visualisieren, betrachten wir nochmal die Abb.3. Unterhalb kann man 3 verschiedene „Helligkeits-Versionen“ von Abb.3 vergleichen.

Abbildung 4: Gamma=0.1

Abbildung 5: Gamma=1

Abbildung 6: Gamma=10

Die Hauptfunktion unsere Programms erhält 6 Parameter. Zwei davon sind `input_file_name` und `output_file_name`, welche die Pfade des Eingabebildes und des Ausgabebildes repräsentieren. Der Parameter `version` vom Datentyp `int` gibt die Version an. Der Parameter `benchmark_number` repräsentiert die Anzahl der Benchmark-Zyklen. Die Parameter `a`, `b`, und `c` vom Datentyp `float` stellen die RGB-Gewichte für die Graustufenumwandlung dar. Schließlich erhält die Funktion noch einen Parameter `_gamma` vom Datentyp `float`, der für den Gammawert steht.

Obwohl die oben genannten Parameter zusätzlich zu den Ein- und Ausgängen standardmäßig mit sinnvollen Defaultwerten besetzt sind, kann der Nutzer diese Parameter auch selbst mit spezifischen und sinnvollen Werten ersetzen. Wenn die gesetzten Werte nicht sinnvoll sind, wird eine Fehlermeldung ausgegeben und das Programm wird beendet.Hier ist eine Übersicht an Optionen, die der Nutzer beim Aufrufen des Programms setzrn kann:

- Option *-V<Zahl>* Die Option -V 0 ist die Standardimplementierung. -V 1 steht für die Implementierung V1 mit Taylorreihe. -V 2 steht für die Implementierung V2 mit SIMD. Andere Werte sind nicht erlaubt.
- Option *-B<Zahl>* Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das Argument gibt die Anzahl an Wiederholungen des Funktionsaufrufs an. Es darf nicht kleiner als 1000 sein.

- Positionalem Argument <*Dateiname*> Der Pfad zur Eingabedatei ist ein obligatorischer Parameter. Falls dieser ungültig ist, wird eine Fehlermeldung ausgegeben und das Programm wird beendet.
- Option *-o*<*Dateiname*> Der Pfad zur Ausgabedatei ist ein obligatorischer Parameter. Falls dieser ungültig ist, wird eine Fehlermeldung ausgegeben und das Programm wird beendet.
- Option *--coeffs*<*FP Zahl*>,<*FP Zahl*>,<*FP Zahl*> Die drei Argumente müssen stets nicht-negativ sein und dürfen nicht den Wert Infinity und NaN annehmen. Ihre Summe darf auch nicht 0 betragen.
- Option *--gamma*<*Floating Point Zahl*> Der Gammawert kann nicht negativ sein. Wenn diese Option nicht gesetzt wird, wird standardmäßig der Wert 1 verwendet.
- Option *-h* | *--help* Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.

2 Lösungsansatz

Wir betrachten nun die Implementierung des Programms. Beim Aufrufen des Programms muss der Benutzer die Eingabedatei und die Ausgabedatei angeben, sonst erhält er eine Fehlermeldung über unvollständige Parameter. Zusätzlich hat der Benutzer die Möglichkeit, beim Ausführen des Programms andere Optionen auszuwählen,Zunächst ruft die main-Funktion parse_options(argc, argv) auf, welche die Methode GETOPT_LONG(3) verwendet, um die Parameter von der Eingabe von dem Nutzer zu erhalten. Anschließend werden switch-case-Anweisungen zur Verarbeitung und Speicherung der jeweiligen Parameter verwendet. getopt_long ist eine Funktion in C, die zum Parsen von Befehlszeilenparametern verwendet wird. Sie kann kurze Optionen (z.B.-o) und lange Optionen (z.B.–coeffs) verarbeiten.

Gemäß der Netpbm-Dateibeschreibung ist die ppm-Datei in zwei Teile unterteilt: Metadaten und Pixelbereich. Für die Formatierung dieser beiden Teile gibt es entsprechende Regeln. Zu beachten ist auch, dass in den Metadaten ein Kommentar an beliebiger Stelle eingefügt werden kann. Daher wird beim Lesen der Metadaten in der ppm-Datei ein Zustandsautomat entworfen und verwendet, um mit möglichen Randfällen fertig zu werden.Der Zustandsautomat für das Lesen der Metadaten in der ppm-Datei ist in den folgenden beiden Diagrammen dargestellt.Abbildung 1: Lesen von P6 in den Metadaten und Verarbeitung möglicher Kommentare. state N zeigt an, dass ein Kommentar eingegeben wurde. state N kann nur durch das Lesen eines LF oder CR verlassen werden.Abbildung 2: Dieser Zustandsautomat wird zum Lesen von width, height und maxval und zur Verarbeitung möglicher Kommentare verwendet. state N zeigt an, dass ein Kommentar eingegeben wurde. state N kann nur durch das Lesen eines LF oder CR verlassen werden.

Bei der Versionsnummer kann der Benutzer zwischen 0, 1 oder 2 wählen. Bei Version 0 und 1 handelt es sich um Algorithmen mit sequentiellen Anweisungen, Version 2 mit

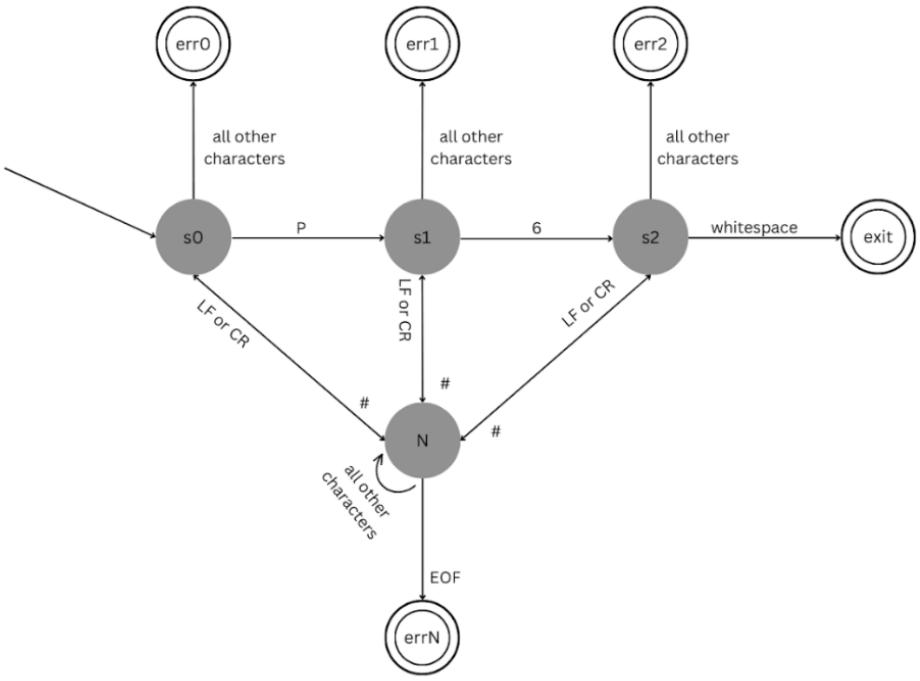

Abbildung 7: Zustandsautomat für das Lesen der Metadaten in der ppm-Datei

SIMD Instruktionen. Wenn der Nutzer keine Version angibt, wird Version 0 als default durchgeführt.

Bei der Implementation von V0 und V1 werden die drei Farben jedes Pixels beim Lesen der Eingabedatei in der originalen Reihenfolge gespeichert. Wenn V2 verwendet wird, werden beim Einlesen die drei Farben jedes Pixels getrennt gespeichert. Dadurch werden drei Speicherbereiche im Heap erstellt, die jeweils für eine bestimmte Farbe alle Pixelwerte in fortlaufender Reihenfolge speichern. In main wird eine Switch-Anweisung verwendet, um verschiedene Funktionen abhängig von der gewählten Versionen aufzurufen.

In allen aufgerufenen Funktionen wird zunächst die Eingabedatei gelesen und dann basierend auf den Metadaten der Eingabedatei Speicher für die Eingabe und Ausgabe allokiert. Die Allokation des Speichers wird durch den Aufruf einer neuen Funktion implementiert. In V2, der SIMD-Implementierung, wird aligned_alloc() verwendet, um Speicher zu reservieren (cite). Die Startadresse des Speichers wird auf ein Vielfaches von 16 ausgerichtet und erhöht damit die Geschwindigkeit des Ladens von Speicher in die xmm-Register. Anschließend werden verschiedene Versionen von gamma_correct()aufgerufen.

In V0 und V1 wird zunächst die Graustufenkonvertierung für jeden Pixel sequentiell berechnet. V2 verwendet SIMD-Anweisungen, welche es ermöglichen die Graustufenwerte von vier Pixeln gleichzeitig zu berechnen. Nachdem die Graustufenwerte

der Pixel berechnet wurden, verwenden V0 und V2 die pow-Funktion der math.h-Bibliothek,um die Graustufenwerte nach der Gammakorrektur für jedes Pixel sequenziell zu berechnen. V1 verwendet keine Bibliotheksfunktionen, sondern implementiert die Taylor-Entwicklung.

Die Taylor Entwicklung ist eine lokale Approximation eines Funktionswertes der Form $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + \ldots + \frac{f^n(a)(x-a)^n}{n!} + R_n(x)$ Wir verwenden den Fall $a = 1$ und erhalten die Formel:

$$f(x) = 1 + \frac{(x-1)^1}{1!}(\gamma) + \frac{(x-1)^2}{2!}(\gamma)(\gamma-1) + \cdots + \frac{(x-1)^n}{n!}\prod_{i=0}^{n-1}(\gamma - i) + R_n(x) \quad (3)$$

Bei Gamma-Werten größer als der magischen Zahl 67075968 ist das Ergebnis für einen anderen Basiswert als 1 immer 0 (Unterlauf). Daher wird darauf geachtet, dass die nachfolgenden Gammawerte kleiner als diese magische Zahl sind. Im Falle eines Basiswerts von 1 wird dies als Sonderfall behandelt, und das Ergebnis bleibt unabhängig vom Gamma-Wert immer 1.

Eine Beobachtung für Formel (1) ist, dass unser Base D(x,y) ist immer eine Zahl zwischen 0 und 1 ist. Gamma ist auf die Bedingung beschränkt, dass es größer als 0 und nicht Infinity oder NaN ist. Bei der Entwicklung haben wir festgestellt, dass die Fakultäten von Gamma bei größeren Gammawerten leicht zu einem Overflow führt, denn wie müssen die Fakultät von Gamma berechnen.Nachdem wir den Fall behandelt haben, dass Gamma zu groß ist, haben wir Gamma in eine ganze Zahl und eine Dezimalzahl zerlegt und separat berechnet,um immer noch mögliche Overflows zu vermeiden.

Für Den Ganzzahlanteil ist unser Design Gedanke, den Exponenten in eine binäre Zahl umzuwandeln und dann zu zerlegen. Im folgenden Beispiel wird die 29. Potenz von 0,79 berechnet

$$29 = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4$$
$$0.79^{29} = 0.79^{2^0} \cdot 0.79^{2^2} \cdot 0.79^{2^3} \cdot 0.79^{2^4}$$

Da Gamma eine Fließkommazahl ist, ist der Ganzzahlanteil maximal 67075968, also zwischen 2^{25} und 2^{26}, so dass wir unabhängig vom Exponenten höchstens 26 Schleifen benötigen.

!!!!Die Graustufenwerte nach der Gammakorrektur werden über einen weiteren Funktionsaufruf in die Ausgabedatei geschrieben. Bevor alle Ressourcen freigegeben werden, wird gamma_correct() wiederholt ausgeführt, wenn der Benutzer die Option B in den Optionen aktiviert hat. Die Zeit wird zu Beginn und am Ende des Zyklus aufgezeichnet. Diese Laufzeit des Zyklus wird später für die Leistungsanalyse verwendet.

!!!!Am Ende der mainFunktion müssen alle Ressourcen freigegeben werden. Um Speicherlecks zu vermeiden, verwenden wir die freeFunktion und fclose, um den Speicherbereich freizugeben. Zusätzlich wird an sämtlichen Stellen in mainFunktion, an denen eine Allocation oder eine IOOperation durchgeführt wird, überprüft, ob sie erfolgreich

war. Wenn der entsprechende Vorgang fehlschlägt, werden alle Ressourcen des aktuellen Programms freigegeben und eine Fehlermeldung in stderr ausgegeben. Danach wird das Programm mit dem Status EXIT_FAILURE beendet.

Das Programm verteilt alle Funktionen sinnvoll auf die verschiedenen Funktionen. Jede Funktion ist angemessen groß, logisch korrekt, leicht zu warten und vermeidet Blob Muster.

3 Genauigkeit

Schauen wir uns die Ungenauigkeit an, die entstanden wäre, wenn das Programm in Version 2 anstelle von pow Formel (3) verwendet hätte. Dies liegt daran, dass wir lediglich eine beschränkte Anzahl von Termen der Taylorreihe berechnen können. Dadurch werden Terme ab dem (n+1)-ten vernachlässigt, was zu Genauigkeitsverlusten führt. Zur Schätzung der Präzision ziehen wir das Peano'sche Restglied heran.

$$o\left((x-a)^n\right)$$

Es zeigt sich, dass das Restglied bei einem Wert von x nahe null langsam und bei x nahe eins schneller konvergiert. Eine präzisere Beschreibung ermöglicht das Lagrange'sche Restglied, insbesondere da der Exponent ein Bruchteil des Gammas ist und wir uns auf Werte zwischen 0 und 1 konzentrieren.

Wir analysieren, wie sich der Kosinusterm einer als FMIN bezeichneten kleinsten durch float darstellbaren Zahl annähert. Hier sind die Ergebnisse bei festem Gamma:

Daraus folgt, dass die Auswirkung von Gamma linear und die von der Basis krummlinig ist. Wenn also die Basis nahe null liegt, konvergiert sie langsam, was ebenfalls zu größeren Ungenauigkeiten führt.

Ergänzend sind folgende Punkte zu beachten:

1. Die Wahrscheinlichkeit liegt bei einem sehr geringen Wert von 0,01.
2. Unter der Annahme einer mittleren Gamma-Verteilung führt eine kleine Basis und ein großes Gamma zu besseren Ergebnissen.
3. Der Fehler wird hierdurch minimiert.

4 Performanzanalyse

5 Zusammenfassung und Ausblick

Bei[1] diesem Projekt ging es darum, eine Graustufen-Kodierung und Gammakorrektur selber zu programmieren. Das Umsetzen der Graustufen-Kodierung war relativ einfach, genau wie die Implementierung mit der Pow-Funktion. Die Aufgabe erhielt eine neue Stufe an Komplexität, sobald es zur Implementierung mit der Taylorreihe und der SIMD Implementierung kommt. Die Taylorreihe führt zu einer geringeren Genauigkeit. Die

SIMD-Funktion hat zu einer höhere Performanz von etwa ... Prozent geführt. Abschließend kann man sagen, dass dieses Projekt einen interessanten Einblick in verschiedene Bildformate, das Approximieren mit Hilfe der Taylorreihe und das Optimieren mit Hilfe von SIMD-Intrinsics geliefert hat.

Literatur

[1] `https://www.cs.miami.edu/home/burt/learning/Csc521.101/notes/virtual-memory-notes.html`, 10 2009.