LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Aspekte der systemnahen Programmierung bei der Spieleentwicklung

Laplace-Filter (A200)
Projektaufgabe – Aufgabenbereich Bildverarbeitung

1 Organisatorisches

Auf den folgenden Seiten finden Sie die Aufgabenstellung zu Ihrer Projektaufgabe für das Praktikum. Die Rahmenbedingungen für die Bearbeitung werden in der Praktikumsordnung festgesetzt, die Sie über die Praktikumshomepage[1] aufrufen können.

Wie in der Praktikumsordnung beschrieben, sind die Aufgaben relativ offen gestellt. Besprechen Sie diese innerhalb Ihrer Gruppe und konkretisieren Sie die Aufgabenstellung. Die Teile der Aufgabe, in denen Assembler-Code anzufertigen ist, sind für die 64-Bit ARMv8-Architektur (AArch64) zu schreiben.

Der **Abgabetermin** ist der **29. Januar 2020, 11:59 Uhr (MEZ)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der Praktikumsordnung angebene Liste von abzugebenden Dateien.

Der **erste Teil Ihrer Projektpräsentation** ist eine kurze Vorstellung Ihrer Aufgabe im Umfang von ca. 2 Minuten in einer Tutorübung in der Woche **09.12.2019 – 13.12.2019**. Erscheinen Sie bitte **mit allen Team-Mitgliedern** und wählen Sie bitte eine Übung, in der mindestens ein Team-Mitglied angemeldet ist.

Die **Abschlusspräsentationen** finden in der Zeit vom **02.03.2020 – 06.03.2020** statt. Weitere Informationen zum Ablauf und die Zuteilung der einzelnen Präsentationstermine werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind.

Bei Fragen/Unklarheiten in Bezug auf den Ablauf und die Aufgabenstellung wenden Sie sich bitte an Ihren Tutor.

Wir wünschen Ihnen viel Erfolg und Freude bei der Bearbeitung Ihrer Aufgabe!

Mit freundlichen Grüßen
Die Praktikumsleitung

PS: Vergessen Sie nicht, sich rechtzeitig in TUMonline zur Prüfung anzumelden. Dies ist Voraussetzung für eine erfolgreiche Teilnahme am Praktikum im laufenden Semester.

[1] `https://www.caps.in.tum.de/lehre/ws19/praktika/asp/`

2 Laplace-Filter

2.1 Überblick

Im Zuge Ihrer Projektaufgabe werden Sie theoretisches Wissen aus der Mathematik im Anwendungszusammenhang verwenden, um einen Algorithmus in Assembler zu implementieren. Sie konzentrieren sich dabei auf das Feld des *Image Processing*, in welchem Pixelbilder, wie sie typischerweise Digitalkameras produzieren, als Eingabe für bestimmte Algorithmen verwendet werden und mathematische Überlegungen dadurch sichtbar gemacht werden.

2.2 Funktionsweise

Der zu implementierende Algorithmus wird verwendet, um in einem Bild die Kanten sichtbar zu machen. Gemeinhin ist er bekannt unter dem Namen *Laplace-Filter*. Der Laplace-Filter wird auf die folgende Art berechnet:

$$A = M * E$$

mit der Faltungsmatrix M der Dimension $2k + 1 \times 2k + 1$ (hier $k = 1$):

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

E bezeichnet dabei das Eingabebild und A das Ausgabebild. Der Operator $*$ ist der Faltungsoperator. Er ist definiert als

$$(f * g)[n] \; = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

wobei n die Gesamtzahl aller Pixel des Bildes ist mit beliebigen Funktionen f und g.

Da die Matrix, welche der Laplace-Filter verwendet, sehr klein ist, wird das Produkt bei der Auswertung der Faltung sehr oft den Wert 0 ergeben. Definiert man einen Pixel p an Bildposition (x, y) mit den Farbkomponenten RGB für Rot- Grün- und Blauanteil als

$$p_{x,y} = \left(R_{x,y}, G_{x,y}, B_{x,y}\right),$$

so lässt sich die Faltungsoperation für einen einzelnen Farbkanal (F) eines Pixels folgendermaßen darstellen (i und j bezeichnen wie gewohnt Spalte und Reihe einer Matrix):

$$A^F_{(x,y)} = \sum_{i=-k}^{k} \sum_{j=-k}^{k} M_{(k+i,k+j)} \cdot E^F_{(x+i,y+j)}$$

Ihre Aufgabe ist es nun, die Kantenfindung für das Testbild *lena.bmp* in Assembler zu implementieren.

2.3 Aufgabenstellungen

Ihre Aufgaben lassen sich in die Bereiche Konzeption (theoretisch) und Implementierung (praktisch) aufteilen. Sie können (müssen aber nicht) dies bei der Verteilung der Aufgaben innerhalb Ihrer Arbeitsgruppe ausnutzen. Antworten auf konzeptionelle Fragen sollten an den passenden Stellen in Ihrer Ausarbeitung in angemessenem Umfang erscheinen. Entscheiden Sie nach eigenem Ermessen, ob Sie im Rahmen Ihres Abschlussvortrags auch auf konzeptionelle Fragen eingehen. Die Antworten auf die Implementierungsaufgaben werden durch Ihrem Code reflektiert.

2.3.1 Theoretischer Teil

- Machen Sie sich mit dem unkomprimierten BMP-Bildformat für 24 Bit Farbtiefe vertraut. Ihr Rahmenprogramm muss in der Lage sein, den Header des Testbilds soweit zu parsen, dass der Anfang der Pixeldaten, die Breite und die Höhe des Bildes zuverlässig gefunden werden.
- Entwickeln Sie einen Algorithmus, welcher die Faltungsoperation für jedes Pixel ausführt. Besprechen Sie Ihren Pseudocode mit Ihrem Tutor, bevor Sie mit der Implementierung beginnen, falls nötig.
- Welche Laufzeit hat Ihr Algorithmus für ein 24Bit-Eingabebild mit $n \times n$ Pixeln (größenordnungsmäßig in $\mathcal{O}$-Notation)?
- Überlegen Sie sich Strategien, um Ihren Algorithmus schneller zu machen. Betrachten Sie dazu SIMD-Operationen und führen Sie, wenn Sie Optimierungsmöglichkeiten finden, einen Performanzvergleich in Form eines Microbenchmarks durch. Gestalten Sie Letzteren sinnvoll nach eigenem Ermessen!

2.3.2 Praktischer Teil

- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie Ihrer Assemblerroutine einen Pointer auf die sequentiellen Bilddaten sowie Höhe und Breite des Bildes übergeben können. Das Rahmenprogramm sollte dann ein neues Bitmap erstellen und die von der Assemblerfunktion berechneten Werte nach dem Aufruf dort hineinschreiben. Das Bitmap sollte mit den üblichen Bildbetrachtern lesbar sein, kopieren Sie also den Header der alten Datei in die neue Datei!

- Implementieren Sie in der Datei mit dem Assemblercode Ihren Algorithmus. Die Assemblerfunktion nimmt dabei einen Pointer auf die Pixelwerte des Bildes als Argument und liefert einen Pointer auf ein Array mit den vom Faltungsalgorithmus berechneten Werten zurück. Sie haben freie Wahl, ob Sie im Assemblercode freien Speicher allokieren oder einen Pointer auf das Ergebnisarray by-reference übergeben. Zusätzlich nimmt die Funktion die Breite und die Höhe des Bildes als Argumente. (Ihr Algorithmus wird diese Daten höchstwahrscheinlich ebenfalls benötigen.)
- Implementieren Sie Ihren Benchmark, um abschätzen zu können, welche Performanzsteigerung die von Ihnen gefundenen Optimierungen bieten (falls zutreffend).

2.3.3 Zusatzaufgabe

- Beschaffen Sie sich Sekundärliteratur zu den Themen Faltung und Laplace-Filter. Wählt man statt der 3×3-Matrix eine 5×5- oder gar 7×7-Matrix, verbessert sich das visuelle Ergebnis. Wie würden Sie die Einträge einer solchen Matrix wählen?
- Das Dateiformat BMP ist aufgrund fehlender Kompressionsalgorithmen und daraus resultierenden Dateigrößen heutzutage nicht mehr häufig in Gebrauch. Erweitern Sie Ihr Programm mithilfe einer JPG-Bibliothek so, dass auch Dateien im JPG-Format gelesen und geschrieben werden können!

2.4 Allgemeine Bewertungshinweise

Die folgende Liste soll Ihnen als Gedächtnisstütze beim Bearbeiten der Aufgaben dienen. Beachten Sie ebenfalls die in der Praktikumsordnung angegebenen Hinweise.

- Stellen Sie unbedingt sicher, dass Ihre Abgabe auf der Referenzplattform des Praktikums (HimMUC-Cluster) kompiliert und funktionsfähig ist.
- Fügen Sie Ihrem Projekt ein funktionierendes `Makefile` hinzu, welches durch den Aufruf von `make` Ihr Projekt kompiliert.
- Verwenden Sie keinen Inline-Assembler.
- Verwenden Sie SIMD-Befehle, wenn möglich.
- Sie dürfen die Signatur der in Assembler zu implementierenden Funktion nur dann ändern, wenn Sie dies (in Ihrer Ausarbeitung) rechtfertigen können.
- I/O-Operationen dürfen grundsätzlich in C implementiert werden.
- Denken Sie daran, das Laufzeitverhalten Ihres Codes zu testen (Sichere Programmierung, Performanz) und behandeln Sie alle möglichen Eingaben, auch Randfälle. Ziehen Sie ggf. alternative Implementierungen als Vergleich heran.

- Eingabedateien, welche Sie generieren, um Ihre Implementierungen zu testen, sollten mit abgegeben werden.
- Verwenden Sie für die Ausarbeitung die bereitgestellte TEX-Vorlage und legen Sie sowohl die PDF-Datei als auch sämtliche TEX-Quellen in das Repository.
- Stellen Sie Performanz-Ergebnisse nach Möglichkeit grafisch dar.
- Vermeiden Sie unscharfe Grafiken und Screenshots von Code.
- Geben Sie die Folien für Ihre Abschlusspräsentation im PDF-Format ab. Achten Sie auf hinreichenden Kontrast (schwarzer Text auf weißem Grund!) und eine angemessene Schriftgröße. Verwenden Sie 4:3 als Folien-Format.
- Zusatzaufgaben (sofern vorhanden) müssen nicht implementiert werden. Es gibt keine Bonuspunkte.