LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

Grundlagenpraktikum: Rechnerarchitektur

Pixelwiederholung (A205)
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 C-Code anzufertigen ist, sind in C nach dem C17-Standard zu schreiben.

Der **Abgabetermin** ist **Sonntag 16. Juli 2023, 23:59 Uhr (CEST)**. Die Abgabe erfolgt per Git in das für Ihre Gruppe eingerichtete Projektrepository. Bitte beachten Sie die in der `README.md` angegebene Liste von abzugebenden Dateien.

Die **Abschlusspräsentationen** finden in der Zeit vom **21.08.2023 – 01.09.2023** statt. Weitere Informationen werden noch bekannt gegeben. Beachten Sie, dass die Folien für die Präsentation am obigen Abgabetermin im PDF-Format abzugeben sind und keine nachträglichen Änderungen akzeptiert werden können.

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

[1] `https://gra.caps.in.tum.de`

2 Pixelwiederholung

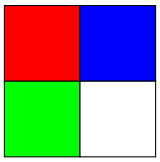
2.1 Überblick

Im Zuge Ihrer Projektaufgabe werden Sie theoretisches Wissen aus der Mathematik im Anwendungszusammenhang verwenden, um einen Algorithmus in C 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.

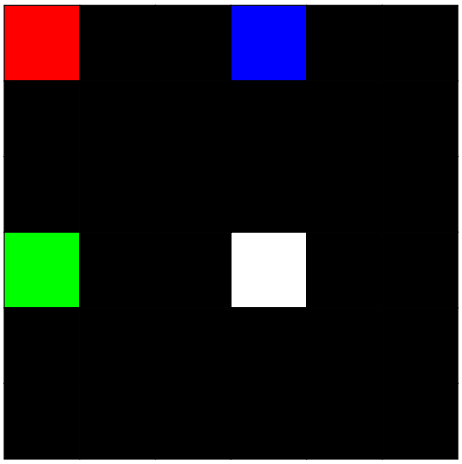
2.2 Funktionsweise

Wir beschäftigen uns in dieser Projektaufgabe mit Pixelgrafiken, die im BMP-Format vorliegen. Ziel der Aufgabe ist es, aus einem gegebenem unkomprimierten Bitmap ein Fenster auszuschneiden und dieses zu vergrößern. Während der Schritt des Ausschneidens algorithmisch trivial ist, soll das Vergrößern nach dem Pixelwiederholungsverfahren (engl. *Nearest Neighbor*) erfolgen, bei dem die entstehenden Lücken nach dem folgenden Verfahren wieder aufgefüllt werden.

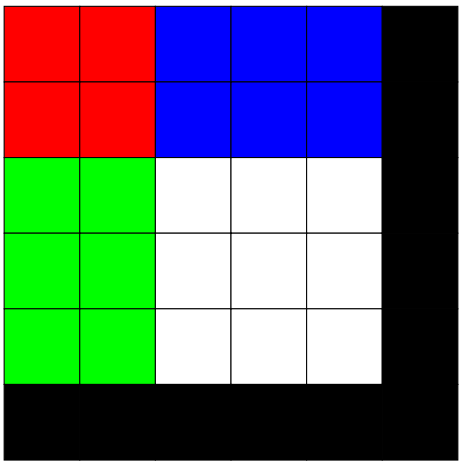
Ausgangsbild



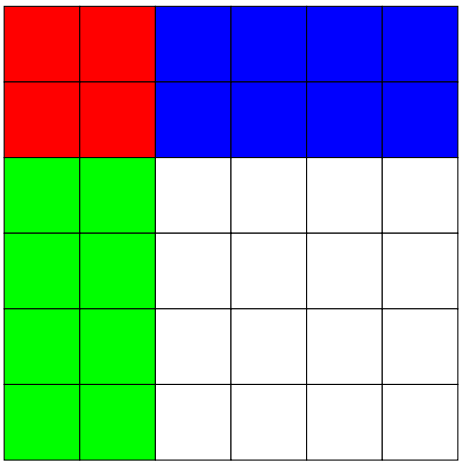
Skaliert mit Faktor 3 Zunächst werden die vorhandenen Pixel auf ihre neuen Positionen bewegt.



Lücken gefüllt mit direkten Nachbarn Anschließend werden die Lücken mit ihren direkten Nachbarn gefüllt.



Lücken gefüllt mit alternativen Nachbarn Uneindeutige Lücken, die von Pixeln gleich weit entfernt sind oder am Rand liegen, werden einer beliebig festgelegten Regel folgend aufgefüllt. In diesem Beispiel wird immer der nächste Pixel links neben der Lücke verwendet, es sind aber auch andere Strategien möglich.



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 Ihren 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.
- Welche Laufzeit hat Ihr Algorithmus für ein 24Bit-Eingabebild mit $n \times n$ Pixeln (größenordnungsmäßig in $\mathcal{O}$-Notation)?

2.3.2 Praktischer Teil

- Implementieren Sie im Rahmenprogramm I/O-Operationen in C, mit welchen Sie eine BMP-Datei einlesen können und Ihrer C-Funktion einen Pointer auf die sequentiellen Bilddaten sowie die relevanten Metadaten (z.B. Höhe und Breite) übergeben können. Das Rahmenprogramm sollte dann ein neue BMP-Datei erstellen und die berechneten Werte nach dem Aufruf dort hineinschreiben. Das Ergebnis sollte mit den üblichen Bildbetrachtern lesbar sein.
- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

```
void window(uint8_t* img, size_t x, size_t y, size_t width, size_t
                    height, uint8_t* result)
```

  Die Funktion berechnet aus den Eingabedaten des Bildes ein Fenster und schreibt nur den Teil an Pixelkoordinaten im Bereich von (x, y) bis $(x + width, y + height)$ in `result`. Allokieren Sie hierzu in Ihrem Rahmenprogramm einen Buffer passender Größe.

- Implementieren Sie in der Datei mit Ihrem C-Code die Funktion:

```
void zoom(const uint8_t* img, size_t width, size_t height, size_t
                  scale_factor, uint8_t* result)
```

  Die Funktion nimmt einen Pointer auf die Pixelwerte des Bildes sowie die Größe des ursprünglichen Bildes und den Skalierungsfaktor als Parameter und schreibt die skalierten Bilddaten in `result`. Allokieren Sie hierzu in Ihrem Rahmenprogramm einen Buffer passender Größe.

2.3.3 Rahmenprogramm

Ihr Rahmenprogramm muss bei einem Aufruf die folgenden Optionen entgegennehmen und verarbeiten können. Wenn möglich soll das Programm sinnvolle Standardwerte definieren, sodass nicht immer alle Optionen gesetzt werden müssen.

- `-V<Zahl>` — Die Implementierung, die verwendet werden soll. Hierbei soll mit `-V 0` Ihre Hauptimplementierung verwendet werden. Wenn diese Option nicht gesetzt wird, soll ebenfalls die Hauptimplementierung ausgeführt werden.
- `-B<Zahl>` — Falls gesetzt, wird die Laufzeit der angegebenen Implementierung gemessen und ausgegeben. Das *optionale* Argument dieser Option gibt die Anzahl an Wiederholungen des Funktionsaufrufs an.
- `<Dateiname>` — Positional Argument: Eingabedatei
- `-s<Zahl>,<Zahl>` — Startpunkt des Fensterausschnitts (x,y)
- `-w<Zahl>` — Breite des Bildfensters
- `-h<Zahl>` — Höhe des Bildfensters
- `-f<Zahl>` — Skalierungsfaktor
- `-o<Dateiname>` — Ausgabedatei
- `-h` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.
- `--help` — Eine Beschreibung aller Optionen des Programms und Verwendungsbeispiele werden ausgegeben und das Programm danach beendet.

Sie dürfen weitere Optionen implementieren, beispielsweise um vordefinierte Testfälle zu verwenden. Ihr Programm muss jedoch nur unter Verwendung der oben genannten Optionen verwendbar sein. Beachten Sie ebenfalls, dass Ihr Rahmenprogramm etwaige Randfälle korrekt abfangen muss und im Falle eines Fehlers mit einer aussagekräftigen Fehlermeldung auf `stderr` und einer kurzen Erläuterung zur Benutzung terminieren sollte.

2.4 Allgemeine Bewertungshinweise

Beachten Sie grundsätzlich alle in der Praktikumsordnung angegebenen Hinweise. Die folgende Liste konkretisiert einige der Bewertungspunkte:

- Stellen Sie unbedingt sicher, dass *sowohl* Ihre Implementierung *als auch* Ihre Ausarbeitung auf der Referenzplattform des Praktikums (`lxhalle`) kompilieren und vollständig korrekt bzw. funktionsfähig sind.
- Die Implementierung soll mit GCC/GNU as kompilieren. Verwenden Sie keinen Inline-Assembler. Achten Sie darauf, dass Ihr Programm keine x87-FPU- oder MMX-Instruktionen und SSE-Erweiterungen nur bis SSE4.2 verwendet. Andere ISA-Erweiterungen (z.B. AVX, BMI1) dürfen Sie nur benutzen, sofern Ihre Implementierung auch auf Prozessoren ohne derartige Erweiterungen lauffähig ist.

- Sie dürfen die angegebenen Funktionssignaturen (nur dann) ändern, wenn Sie dies (in Ihrer Ausarbeitung) begründen.
- Verwenden Sie die angegebenen Funktionsnamen für Ihre Hauptimplementierung. Falls Sie mehrere Implementierungen schreiben, legen wir Ihnen nahe, für die Benennung der alternativen Implementierungen mit dem Suffix „`_V1`“, „`_V2`“ etc. zu arbeiten.
- 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; größere Eingaben sollten stattdessen stark komprimiert oder (bevorzugt) über ein abgegebenes Skript generierbar sein.
- 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 16:9 als Folien-Format.