LEHRSTUHL FÜR RECHNERARCHITEKTUR UND PARALLELE SYSTEME

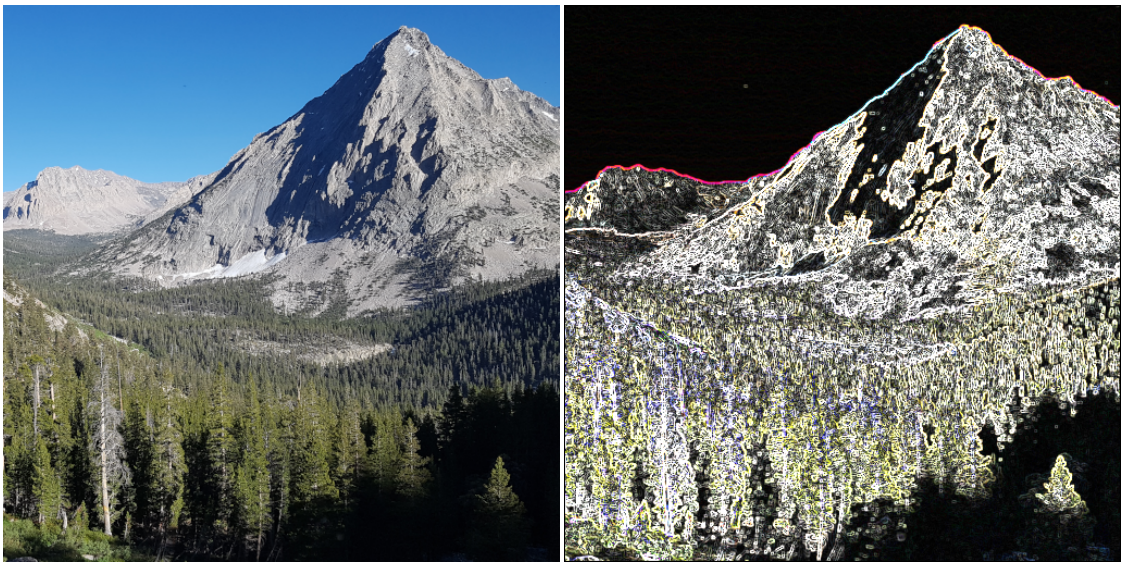
Grundlagenpraktikum: Rechnerarchitektur

Gruppe 110 – Abgabe zu Aufgabe A201
Sommersemester 2023

Simon Bußmann Nico Lintner Manuel Walter Mußbacher

1 Einleitung

In diesem Projekt haben wir uns damit beschäftigt den Sobel-Filter Algorithmus zu implementieren. Der Algorithmus wird dazu verwendet um Kanten in Bildern zu erkennen. Dabei werden die Pixel eines Bildes mit zwei Matritzen verrechnet. Diese Matritzen werden auch Filter genannt. Die aus der Verrechnung resultierenden Werte werden dann in einem neuen Bild gespeichert.



(a) Input-Bild (b) Output-Bild

In Abbildung 1a und 1b ist ein Beispiel für die Anwendung des Sobel-Filters zu sehen.

2 Lösungsansatz

Unser gewählter Lösungsansatz besteht aus drei verschiedenen Versionen. In der ersten Version haben wir den Sobel-Filter nach seiner mathematischen Definition implementiert. Zusätzlich dient und die erste Version als Vergleichsimplementierung. Die zweite und dritte Version sind jeweils eine Optimierung der ersten Version. In der zweiten Version haben wir die den Algorithmus mit Hilfe von SIMD-Instruktionen implementiert und in der dritten die SIMD Implementierung mit Threading kombiniert. Alle

Implementierungen arbeiten mit 24-Bit BMP Bildern. Ein Pixel besteht dabei aus drei Bytes, die die Farbwerte Rot, Grün und Blau repräsentieren.

2.1 Vergleichsimplementierung

Die Vergleichsimplementierung ist eine naive Implementierung des Sobel-Filter Algorithmus. Dabei werden die beiden Filtermatritzen M^v und M^h mit jedem Pixel des Bildes verrechnet.

$$M^v : \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad M^h : \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1}$$

$$A^h = M^h * Image \tag{2}$$

$$A^v = M^v * Image \tag{3}$$

Jeder Pixel besteht aus drei Bytes, die die Farbwerte Rot, Grün und Blau repräsentieren. Deswegen wird das Bild B in drei Farbkanäle F aufgeteilt. Um einen Pixel mit den Filtermatritzen zu verrechnen, werden die Werte der Filtermatrix mit den Werten der umliegenden Pixel multipliziert und anschließend aufsummiert.

$$A_(x,y)^{v,F} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} M_{i,j}^{v} * B_{(x+i,y+j)}^{F} \tag{4}$$

$$A_(x,y)^{h,F} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} M_{i,j}^{h} * B_{(x+i,y+j)}^{F} \tag{5}$$

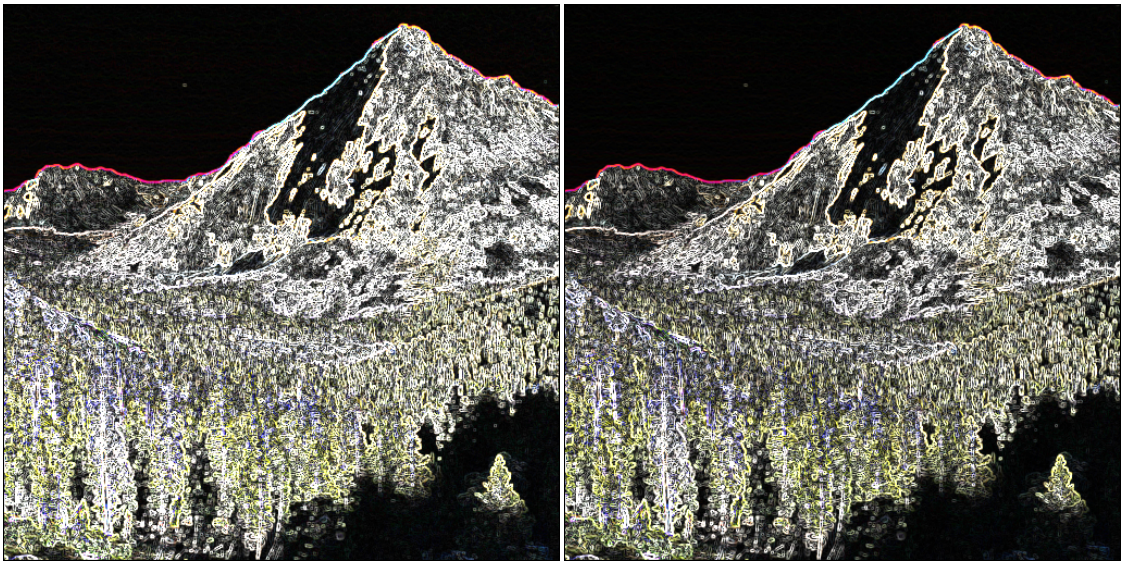
Pixel, die sich am Rand des Bildes befinden, können nicht mit allen Werten der Filtermatritzen verrechnet werden. Der Fall wird behandelt, indem die Sobelwerte dieser Pixel als schwarz angenommen werden. Hierdurch entsteht ein schwarzer Rand um das Bild.

Um nun den Sobelwert eines Pixels zu berechnen wird der Betrag der horizontalen und vertikalen Sobelwerte berechnet.

$$O_{x,y}^{F} = \left|A_{x,y}^{v,F}\right| + \left|A_{x,y}^{h,F}\right| \tag{6}$$

Normalerweise wird dieser Schritt mit der Wurzel der Summe der Quadrate berechnet.

$$O_{x,y}^{F} = \sqrt{(A_{x,y}^{v,F})^2 + (A_{x,y}^{h,F})^2} \tag{7}$$



(a) Abs Version 6 (b) Sqrt Version 7

Zwischen 2a und 2b besteht nur ein kleiner Unterschied in der Helligkeit des Bildes, was für Kantenerkennung nicht relevant ist. Da keine brauchbare SIMD Instruktion zur Berechnung der Wurzel eines 8 bzw. 16 Bit Integer existiert haben wir uns für die Variante 6 entschieden.

2.2 SIMD Implementierung

2.3 SIMD Implementierung mit Threading

3 Genauigkeit

TODO: Geben Sie hier die Genauigkeit Ihres Lösungsansatzes ein.

4 Performanzanalyse

TODO: Geben Sie hier die Performanz Ihres Lösungsansatzes ein.

5 Zusammenfassung und Ausblick

TODO: Geben Sie hier eine kurze Zusammenfassung und einen Ausblick ein.

Literatur