Technische Universität Dresden
Fakultät Informatik
Institut für Technische Informatik
Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Bachelorarbeit

Evaluation einer modernen Zynq-Plattform am Beispiel der Implementierung einer Hough Transformation

Dominik Weinrich
geboren am 11.08.1990 in Kassel
(Mat.-Nr.: 3914410)

Betreuender Hochschullehrer:
Prof. Dr.-Ing. habil. Rainer G. Spallek

Betreuer:
Oliver Knodel

Dresden, Datum

Aufgabenstellung

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Studienarbeit zum Thema

Evaluation einer modernen Zynq-Plattform am Beispiel der Implementierung einer Hough Transformation

selbstständig verfasst und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt, nur die angegebenen Quellen benutzt und die in den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Name, Dresden, Datum

Wettbewerbsrechtlicher Hinweis

Die bloße Nennung von Namen, Produkten, Herstellern und Firmennamen dient lediglich als Information und stellt keine Verwendung des Warenzeichens sowie keine Empfehlung des Produktes oder der Firma dar.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

ABS Antiblockiersystem

CPU Central Processing Unit

ESP Elektronisches Stabilitätsprogramm

FPGA Field Programmable Gate Array

GPU Graphics Processing Unit

HLS High Level Synthese

HT Hough-Transformation

HW Hardware

SW Software

1 Einleitung und Motivation

Vor einigen Jahren hatten die meisten Autos ein ABS[1] und ein ESP[2] zur Unterstützung bei Gefahrenbremsungen und zur Stabilisierung der Fahrt in scharfen Kurven. Mittlerweile ist die Liste der Fahrassistenzsysteme deutlich größer geworden und umfasst unter anderen auch Adaptive Geschwindigkeitsregelanlagen, Spurhalte- und Spurwechselassistenten, Einparkhilfen und Autonome Notbremssysteme. Viele dieser Systeme basieren auf Bildbearbeitungs- und Bildanalysealgorithmen, welche gerade in Gefahrensituationen schnellstmöglich ausgewertet werden müssen. Hierfür ist die Nutzung einer CPU[3] meist keine ausreichende Lösung, da diese zwar eine vergleichsweise hohe Taktrate besitzt, aber aktuell mit bis zu 8 Kernen nicht genug Parallelität bietet, um große Bilder effektiv auswerten zu können. Solche Probleme werden daher vermehrt in Hardware ausgelagert durch einen FPGA[4] oder eine GPU[5] gelöst.
Diese Bachelorarbeit behandelt eine solche Auslagerung von Software in Hardware. Dazu wird im zweiten Kapitel einleitend die Zielarchitektur vorgestellt und ein Einblick in die Grundlagen des HW[6]/SW[7] Codesigns, so wie der HLS[8] gegeben. Im dritten Kapitel wird am Beispiel einer Hough Transformation eine HLS durchgeführt. Dazu wird zunächst die Softwareimplementierung vorgestellt. Anschließend werden iterativ einzelne Komponenten der Hough Transformation von einer CPU auf einen FPGA ausgelagert.
Das vierte Kapitel behandelt die Auswertung des HW/SW Codesigns hinsichtlich des erbrachten Speedups und des Ressourcenverbrauchs und im fünften Kapitel wird abschließend ein Fazit gezogen und ein Ausblick in das Thema gegeben.

[1] Antiblockiersystem
[2] Elektronisches Stabilitätsprogramm
[3] Central Processing Unit
[4] Field Programmable Gate Array
[5] Graphics Processing Unit
[6] Hardware
[7] Software
[8] High Level Synthese

2 Grundlagen

2.1 Zielarchitektur - Zynq Systemarchitektur

2.2 Hardware/Software Codesign

2.3 High Level Synthese

2.4 Hough Transformation

Die HT[1] ist ein Verfahren zur Erkennung von beliebigen parametrisierbaren Objekten in einem Bild. Dabei können durch ein Votingverfahren auch unvollständige, z.B. teilweise verdeckte Objekte, erkannt werden. Da die HT nur auf binäre Gradientenbilder angewandt werden kann, muss ein Eingabebild zunächst in ein solches umgewandelt werden. Hierfür sind mehrere Schritte erforderlich, die in den folgenden Unterkapiteln behandelt werden.

2.4.1 Umwandlung eines RGB Bildes in Graustufen

Damit eine Kantenextraktion mit dem Canny-Algorithmus durchgeführt und somit ein binäres Gradientenbild erzeugt werden kann, muss zuvor eine Umwandlung eines Farbbildes in ein Graustufenbild erfolgen. Ein Bild ist grau, wenn die rot, grün und blau (R, G, B) Komponenten den selben Wert haben. Dann kann allerdings auch anstatt der zuvor benötigten drei Farbkanäle (R, G, B) mit je 8 Bit pro Pixel nur noch ein Farbkanal mit 8 Bit pro Pixel verwendet werden. Damit wird zusätzlich die Größe des Bildes und damit die Größe, der zu verarbeitenden Daten deutlich reduziert.

Für jedes Pixel werden die drei Farbkomponenten (R, G, B) in einen Intensitätswert Y umgerechnet, welcher die Helligkeit des Pixels beschreibt. Es gibt eine Vielzahl verschiedener Ansätze, die sich unterschiedlich gut zur Kantenextraktion eignen. An dieser Stelle soll nicht näher auf die Unterschiede eingegangen werden. Zwei detaillierte Vergleiche zu dem Thema finden sich in [KC] und [AMS]. Daraus geht hervor, dass GLuminance als Standardalgorithmus zur Bildverarbeitung gilt. Die Intensität berechnet sich nach GLuminance gewichtet, da wir rot und grün deutlich heller als blau wahrnehmen.

[1] Hough-Transformation

$$Y = 0.3 * R + 0.59 * G + 0.11 * B \tag{2.1}$$

Die Umwandlung eines Bildes der Größe $M \times N$ benötigt $5(M \cdot N)$ Operationen. Unter der vereinfachten Annahme, dass die Bildgröße $N \times N$ ist folgt damit eine Komplexität von $O(N^2)$.

2.4.2 Gauß-Filter

Einige Bilder enthalten ein Bildrauschen, welches bei einer späteren Kantenextraktion zu fehlerhaften Kanten führen kann. Um dieses Rauschen zu verringern wird zur Glättung des Bildes ein lineares Weichzeichnungsfilter, wie das Gauß-Filter, angewandt.

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \qquad G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

Die eindimensionale Filterfunktion $G(x)$ entspricht der Funktion der Normalverteilung. Die zweidimensionale Funktion $G(x, y)$ des Gauß-Filters ergibt sich aus dem Produkt der Funktionen in x- und y-Richtung. Die Argumente x und y bezeichnen jeweils die Entfernungen in horizontaler und vertikaler Richtung zu dem aktuell gefilterten Pixel und σ bezeichnet die Standardabweichung der Normalverteilung. [BB09, S. 106-109]

Ein Filterkernel für einen Gauß-Filter lässt sich nun über die zweidimensionale Filterfunktion berechnen. Ein Kernel der Größe $N \times N$ mit $N \in \{2i + 1 | i \in \mathbb{N} \setminus 0\}$ berechnet sich wie folgt:

$$X = \sum_{x=-B}^{B} \sum_{y=-B}^{B} G(x,y)$$
$$B = (N-1)/2$$

$$\frac{1}{X} \begin{pmatrix} h(-B,-B) & ... & ... & h(0,-B) & ... & ... & h(B,-B) \\ h(-B,-1) & ... & h(-1,-1) & h(0,-1) & h(1,-1) & ... & h(B,-1) \\ h(-B,0) & ... & h(-1,0) & h(0,0) & h(1,0) & ... & h(B,0) \\ h(-B,1) & ... & h(-1,1) & h(0,1) & h(1,1) & ... & h(B,1) \\ h(-B,B) & ... & ... & h(0,B) & ... & ... & h(B,B) \end{pmatrix} \tag{2.3}$$

Ein Filter der Größe $(2K + 1) \times (2L + 1)$ angewandt auf ein Bild mit $M \times N$ Pixeln benötigt $2K \cdot 2L \cdot M \cdot N = 4KLMN$ Operationen. Bei der vereinfachten Annahme, dass Bild und Filter die Größe $N \times N$ haben folgt daraus eine Komplexität von $O(N^4)$.
Durch Ausnutzen der Separierbarkeit des Gauß-Filters kann die Rechenzeit allerdings deutlich reduziert werden, indem das zweidimensionale Filter in zwei eindimensionale Filter $G(x)$ und

$G(y)$ aufgeteilt wird (2.4). Diese werden dann nacheinander auf das Bild angewandt. [BB09, S.113f]

$$\frac{1}{16}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} = \frac{1}{4}\begin{pmatrix}1\\ 2\\ 1\end{pmatrix} * \frac{1}{4}\begin{pmatrix}1 & 2 & 1\end{pmatrix} \tag{2.4}$$

In der Bildverarbeitung wird häufig nur eine Approximation für den Filterkernel verwendet, die auf dem Pascalschen Dreieck beruht. Diese bietet den Vorteil, dass der Kernel aus ganzen Zahlen aufgebaut ist und zur Normierung durch eine Zweierpotenz geteilt werden kann. Für einen Kernel der Größe N wird die N-te Zeile des Pascalschen Dreiecks in ein Feld mit N Elementen geladen. Anschließend wird der Kernel wie folgt aufgebaut: Die i-te Zeile des Kernels entsteht durch Multiplikation des Feldes mit dem i-ten Element des Felds. Der Faktor zur Normierung ist dann $X = 2^N$.

Die Pixel am Rand eines Bildes müssen bei der Filterung besonders betrachtet werden, da hier nicht das gesamte Filter angewandt werden kann. Um das Filter auf ein Randpixel anzuwenden gibt es verschiedene Möglichkeiten. An dieser Stelle soll nur die in dieser Arbeit verwendete Methode erläutert werden:
Das Bild wird um die benötigte Pixelanzahl erweitert und die Pixel außerhalb des Bildes auf den Wert des Randpixels gesetzt. Dies hat nur kleinere Artefakte zur Folge und wird Aufgrund der Einfachheit häufig verwendet. [BB09, S. 125f]

2.4.3 Canny Edge Detection

2.4.4 Circle Hough Transformation

2.4.5 Optimierungen

3 Hardware/Software Codesign am Beispiel einer Hough Transformation

3.1 Softwareimplementierung

3.2 Iterative Auslagerung einzelner Komponenten auf den FPGA

4 Evaluation

5 Fazit und Ausblick

Literaturverzeichnis

[AMS] Ahmad, Ijaz ; Moon, Inkyu ; Shin, Seok J.: Color-to-grayscale algorithms effect on edge detection - A comparative study. In: editor (Hrsg.) ; IEEE (Veranst.): *Electronics, Information, and Communication (ICEIC), 2018 International Conference on* IEEE

[BB09] Burger, Wilhelm ; Burge, Mark J.: *Principles of Digital Image Processing.* London, 2009

[KC] Kanan, Christopher ; Cottrell, Garrison W.: Color-to-Grayscale: Does the Method Matter in Image Recognition.