Methoden zur Bildsegmentation und zum Connected Component Labeling auf einem FPGA für eine zukünftige On-Board Bilddatenverarbeitung

Dipl.-Math. Kurt Schwenk

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktor-Ingenieur
(Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. habil. Hans-Joachim Gudladt (UniBw. M.)
1. Gutachter:	Univ.-Prof. Dr.-Ing. Felix Huber (UniBw. M.)
2. Gutachter:	Univ.-Prof. Dr. Thomas Rösgen (ETH Zürich)

Die Dissertation wurde am 01.12.2015 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 11. Mai 2016 angenommen. Die mündliche Prüfung fand am 9. September 2016 statt.

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich im Rahmen dieser Arbeit begleitet haben.

Ich möchte mich bei Professor Dr. Felix Huber, Frau Dr. Maria Schönermark und dem Deutschen Zentrum für Luft- und Raumfahrt bedanken, die mir ermöglicht haben, diese Arbeit anzufertigen.

Ganz besonderer Dank gilt Frau Katharina Willburger, die Maßgeblich an der Korrektur des Manuskriptes beteiligt war und mir auch fachlich mit Rat und Tag zur Seite stand.

Im weiteren möchte ich mich bei Herrn Benjamin Koch bedanken, der die finale Fassung überprüft hat, und bei Herrn Florian Waschbichler der den mathematischen Teil dieser Arbeit begutachtet hat.

Danken möchte ich außerdem meinem Mitstreiter Dr. Sebastian Beulig für die vielen schönen Stunden, die wir trotz zahlreicher Anstrengungen, miteinander verbringen durften.

Abstract

The concept of the current generation of remote sensing satellites, is to send raw image data to the ground. Due to the extremely high volume of the image data, transmission capacity is a bottleneck. Also there is a high latency between image acquisition and the delivery of the product data, which can be in the range of several hours. This can be a real restriction for time critical applications, as disaster recognition.
By performing data analysis directly after the image acquisition on board, the satellite useless data can be filtered out and time critical information can be delivered to the customer within a few minutes. The downside of this concept is that image processing has to be performed on board of the satellite. Many image processing steps are computationally too intensive for a software implementation on the on-board computer. One possible solution is the implementation of these demanding processing steps on a special computing unit, called field-programmable gate array (FPGA). If a processing step is suitable for an FPGA implementation, high performance and low power consumption can be provided.
In this work a high-performance FPGA implementation of image segmentation, including connected-component labeling (CCL), is presented. Especially two processing algorithms, the gap-smoothing algorithm and the two-pass single storage CCL (TPSS-CCL) method, which are the main components of the process chain, are explained in detail. The gap-smoothing algorithm is a edge preserving smoothing filter, that can handle heavy image noise. The TPSS-CCL is a special memory efficient implementation of the classical two-pass connected-component labeling method, able to process images without restrictions in complexity and the number of connected components. Both filters have been specially designed delivering high performance on an FPGA and are able to process large complex image data. The performance of the FPGA implementations of these algorithms have been studied in detail. Also the the hardware requirements for the implementations are presented.
The second part of this work focuses on the evaluation of segmentation quality. An accurate, an objective and an good to interpret method for measuring the quality of segmentation methods is presented. The measurement process is based on the new developed SA_{EQ} metric. This metric is able to determine the difference between image segmentations.
With SA_{EQ} the qualitative performance of the segmentation process, presented in this work, has been studied in detail. Finally a comparison with different segmentation methods is presented.

Zusammenfassung

Das Konzept der aktuellen Generation von Fernerkundungssatelliten, Aufnahmen zu einem fest vorgegeben Zeitpunkt durchzuführen und ohne weitere Zwischenverarbeitung zur Erde zu senden, stößt gerade bei der Nutzung von moderneren hochauflösenden Sensoren aufgrund der extrem hohen Datenmenge und gleichzeitig begrenzter Übertragungskapazitäten an seine Grenzen. Durch die Auswertung von Fernerkundungsdaten an Bord eines Satelliten, können nicht verwertbare Bilddaten bereits vor der Übertragung zur Erde ausgefiltert werden, wenn nicht sogar die meist wesentlich kompaktere Nutzinformation, direkt an Bord extrahiert werden. Einer der wesentlichsten Vorteile der an Bord Datenverarbeitung ist jedoch, dass sich der Zeitraum vom Zeitpunkt der Aufnahme bis zur Auslieferung der Nutzinformation in vielen Fällen deutlich verkürzen lässt. Gerade in Hinblick auf Katastrophenwarnsysteme ist dies ein entscheidender Faktor.
Viele Verarbeitungsschritte, welche bei einer Bildanalyse anfallen, sind jedoch sehr rechenaufwendig. Sie können in Software auf heutigen On-Board Computern oftmals nicht im gewünschten Maße durchgeführt werden. Ein Lösungsansatz stellt die Implementation dieser Verarbeitungsschritte in Hardware dar. Falls sich diese Verfahren für eine Hardwareimplementation eignen, kann eine hohe Rechenleistung bei einem relativ geringen Energieverbrauch bereitgestellt werden. Dies ist jedoch meistens mit erheblichen Entwicklungsaufwand verbunden. Ein Möglichkeit Hardware mit vergleichsweise geringem Aufwand zu realisieren, stellen konfigurierbare Hardwarebausteine, sogenannte Field Programmable Gate Arrays (FPGAs) dar.
In dieser Arbeit wird gezeigt, wie eine Bildsegmentation auf einem FPGA realisiert werden kann. Die Bildsegmentation ist ein rechenaufwendiger Schritt, welcher ein essentieller Teil in der Verarbeitungskette vieler Bildanalyseverfahren ist. Sie lässt sich jedoch aufgrund der Komplexität nicht ohne weiteres auf einem FPGA implementieren. Im Rahmen dieser Arbeit wurde daher ein, speziell für eine FPGA-Implementation geeignetes, kantenerhaltendes Glättungsverfahren entwickelt. Außerdem musste ein Ansatz entwickelt werden, mit welchem die Bestimmung der Zusammenhangskomponenten eines Bildes – dass sogenannte Connected Component Labeling – großer und komplexer Bilder performant und speicherschonend auf einem FPGA durchgeführt werden kann. Das Laufzeitverhalten und der Ressourcenverbrauch wurde anhand verschiedener Modellimplementationen detailliert untersucht, und um die Leistungsfähigkeit besser einordnen zu können ebenfalls ein Vergleich mit Softwareimplementationen durchgeführt.
Ein weiterer Schwerpunkt dieser Arbeit liegt auf der qualitativen objektiven Bewertung eines Segmentationsprozesses. Es wird ein möglicher Weg aufgezeigt, wie die qualitative Leistungsfähigkeit eines Segmentationsverfahrens ermittelt werden kann. Es wurde hierbei insbesondere darauf Wert gelegt, den Begriff der Segmentation und der Segmentationsqualität auf exakte, mathematisch formulierte Grundlagen zu stellen. Mithilfe der mathematisch exakten Darstellung, konnte ein existierendes Bewertungsverfahren maßgeblich erweitert werden. Hierdurch konnte die Qualität des entwickelten Segmentationsverfahrens deutlich verbessert und eine detaillierte Untersuchung der Leistungsfähigkeit des Segmentationsverfahren vorgenommen werden. In diesem Rahmen wurde auch ein Vergleich mit alternativen Segmentationsmethoden durchgeführt.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Ziel dieser Arbeit ist es, der Gruppe On-Board-Classification (OBC) am Deutschen Zentrum für Luft- und Raumfahrt e.V. (DLR) Verfahren zur Analyse von optischen Fernerkundungsdaten an Bord eines Satelliten bereitzustellen.
Dies ist eine der Grundvoraussetzungen für moderne, intelligente Erdbeobachtungssatellitensysteme [1], die nicht nur fest vorgegebene Bildaufnahmen erstellen und diese in einer fest vorgegebenen Reihenfolge zur Erde senden, sondern selbstständig entscheiden sollen, ob und wann es sinnvoll ist, eine Aufnahme zu machen, welche Daten, zu welchem Zeitpunkt, in welcher Reihenfolge, auf welchem Kommunikationskanal übertragen werden. Bei unvorhersehbaren Ereignissen, wie zum Beispiel Naturkatastrophen, sollen zukünftige Erdbeobachtungssatellitensysteme in der Lage sein, diese Krisensituation zu erkennen, eine erste Analyse durchzuführen, um gegebenenfalls unmittelbar Alarm schlagen und eigenständig weitere Aufnahmen des Krisengebietes einplanen und durchzuführen zu können.
Um in Zukunft derart komplexe Satellitensysteme realisieren zu können, besteht in vielen unterschiedlichsten Themenfeldern, vor allem im Bereich der Datenverarbeitung und On-Board-Intelligenz, noch ein gewaltiger Entwicklungsbedarf. Eine Hauptaufgabe der Datenverarbeitung an Bord eines Erdbeobachtungssatellit ist die zielgerichtete Analyse von aufgenommenem Bildmaterial. Die Nutzinformation soll vollständig automatisch, innerhalb einer meistens kurzen Zeitspanne extrahiert werden. Um dies durchführen zu können, müssen geeignete, vollständig automatisch ablaufende Methoden zur Verfügung stehen oder entwickelt werden. Und es muss ihnen ausreichend Rechenleistung an Bord des Satelliten zur Verfügung gestellt werden.
Diese Arbeit beschäftigt sich in erster Linie mit der letzteren Problematik. Mit gewöhnlichen, auf Standartprozessoren basierten Bordrechnern ist es in vielen Fällen nicht möglich, die erforderliche Rechenleistung aufzubringen. Daher müssen alternative Wege gefunden werden. Eine Lösungsmöglichkeit besteht darin, die Verfahren direkt in Hardware zu realisieren. Sind die Verfahren gut für eine Hardwareimplementation geeignet, kann eine sehr hohe Verarbeitungsleistung erreicht werden. Dies ist beispielsweise bei fensterbasierten Glättungs- oder Kantendetektionsfiltern, wie sie häufig in der Bildverarbeitung eingesetzt werden, möglich [2]. Die Hardwareentwicklung ist jedoch ein komplexes Unterfangen. Software basierte Verfahren müssen oftmals erst aufwendig angepasst werden, um sie performant in Hardware umsetzen zu können. Dieser Aufwand ist jedoch in vielen Fällen notwendig. Beispielsweise wäre Full High Definition Videowiedergabe auf modernen Smartphones ohne Hardwaredecodierung selbst mit den modernsten mobilen Prozessoren nicht annähernd ruckelfrei möglich. Eine relativ komfortable Möglichkeit Hardwareschal-

tungen zu realisieren stellen Field Programmable Gate Arrays (FPGAs) dar.
Der Schwerpunkt dieser Arbeit liegt im Themengebiet der Bildsegmentation, in erster Linie für den Einsatz in Fernerkundungsanwendungen. Eine Bildsegmentation ist ein Verarbeitungsschritt, der in vielen Bereichen der Bilddatenverarbeitung, insbesondere der Objekterkennung, benötigt wird. Eine Bildsegmentation stellt bei vielen Aufgaben einen der rechenaufwendigsten Verarbeitungsschritten dar. Eine zeitnahe Verarbeitung, gerade von sehr großen Datenmengen, wie sie in der Fernerkundung anfallen, ist mit gewöhnlichen Bordrechnern kaum erreichbar. Mit einer auf FPGA basierten Hardwarelösung könnte die notwendige Rechenleistung jedoch aufgebracht werden. Dies zu realisieren ist das Hauptziel dieser Arbeit.

1.2 On-Board Bilddatenverarbeitung

Die meisten derzeit eingesetzten optischen Erdbeobachtungssatelliten, wie zum Beispiel die der Landsat Reihe, arbeiten nach dem Store-And-Forward Prinzip. Die Bilddaten werden nach der Aufnahme an Bord des Satelliten zwischengespeichert und bei einem Kontakt mit einer Bodenstation zur Erde gesendet [3]. Auf den Inhalt oder die Qualität der aufgenommen Daten wird bei dieser Prozedur nicht eingegangen. Dies ist aber bei modernen Erdbeobachtungssatelliten, wie im Folgenden erläutert wird, unerlässlich, um sie effizient betreiben zu können.
Ein Engpass, der bei vielen modernen und vor allem auch zukünftigen Erdbeobachtungssatelliten immer mehr in Erscheinung tritt, ist die Nutzdatenübertragung zur Erde. In den letzten Jahrzehnten hat sich die Sensorik weitaus schneller weiterentwickelt als die Kommunikationstechnologie [4]. Sowohl mit räumlich hochauflösenden Sensoren, als auch mit hyperspektralen Sensoren, die Hunderte von Spektralkanälen besitzen [5], werden extrem große Datenmengen erzeugt, welche mit heutigen Übertragungsverfahren nicht mehr ohne Weiteres zum Boden transferiert werden können. Beispielsweise könnten mit dem X-Sat Satelliten, welcher eine räumliche Auflösung von 10 m im sichtbaren und nahem infraroten Bereich hat, jeden Orbit 38 GByte an Daten gewonnen werden. Es können davon aber pro Orbit nur ungefähr 3,5 GByte Daten abgesetzt werden [4]. Um diesen Engpass zu lösen, wird an zwei Stellen angesetzt. Erstens wird versucht, die physikalische Datenübertragungsbandbreite zu erhöhen. Moderne Erdbeobachtungssatelliten, wie die der Sentinel-2 Reihe [6], verwenden leistungsfähige X-Band Transmitter und setzen auf eine Vielzahl von Bodenstationen. Auch sollen bei den Sentinel Satelliten bereits neuartige Übertragungskonzepte, wie eine Datenübertragung über ein Netzwerk aus geostationären Kommunikationssatelliten, eingesetzt werden[1]. Zur Datenübertragung zwischen den Satelliten sollen auch optische Laser Verbindungen eingesetzt werden. Auf der anderen Seite wird On-Board Datenverarbeitung eingesetzt, um die zu übertragende Datenmenge zu verringern. Durch Komprimierung der Rohbilddaten kann Datenvolumen reduziert werden. Mit verlustfreier Kompression können Rohbilddaten beispielsweise durchschnittlich um den Faktor 3 komprimiert werden [2]. Zusätzlich kann durch eine erste Datenanalyse an Bord des Satelliten die Übertagung nicht verwertbarer Daten vermieden werden.

[1] European Data Relay Satellite System (EDRS)

In vielen Fällen ist es beispielsweise sinnvoll, die Wolkenbedeckung in Satellitenaufnahmen des Erdoberfläche schon an Bord zu ermitteln. Stark bewölkte Szenen können dann herausgefiltert werden. Es kann auch von Vorteil sein Nutzdaten direkt an Bord des Satelliten zu extrahieren, da diese meist ein sehr viel kleineres Datenvolumen, im Vergleich zu den Rohdaten, besitzen.
Eine weitere Problematik heutiger Erdbeobachtungssatelliten ist die Latenz der Datengewinnung und der damit verbundenen fehlenden Flexibilität bei einer sich verändernden Situation. Tritt ein nicht vorhersagbares Ereignis wie eine Überflutung oder ein Feuer auf, ändert sich bei den meisten heutigen Erdbeobachtungssatelliten die Operationsausführung nicht. Bis die Aufnahme zum Datenanalysezentrum geschickt wurde, hier das Ereignis detektiert wurde, vergehen wertvolle Stunden, teils Tage, in denen schon eine Analyse der Lage am Boden hätte erfolgen können, Alarm gegeben und weitere Aufnahmen des Krisengebietes angefertigt hätten werden können. Wird jedoch nach einer Bildaufnahme sofort an Bord des Satelliten ein Katastrophenereignis wie eine Überflutung oder ein Feuer festgestellt, kann unmittelbar darauf reagiert werden. Es könnte über spezielle Kommunikationskanäle eine sofortige Warnmeldung an Einsatzkräfte erfolgen und durch Datenpriorisierung ein möglichst schnelles Absetzen der für den Katastrophenschutz wichtigen Daten erreicht werden. Außerdem könnte sofort veranlasst werden, das Katastrophengebiet nach Möglichkeit weiter zu beobachten, zum Beispiel durch eine Drehung des Satelliten oder mit Hilfe weiterer Satelliten in einem Satellitenverbund.
Vollständig unerlässlich ist eine On-Board Datenanalyse bei Echtzeitanwendungen wie Navigation, bei Rendezvousmanövern oder Landemanövern auf Asteroiden oder fremden Planeten. Eine Steuerung des Satelliten von der Erde aus ist aufgrund der Verzögerung in vielen Fällen nicht möglich.
Um eine solche Funktionalität zur Verfügung zu stellen, muss die Datenverarbeitung an Bord eines zukünftigen Erdbeobachtungssatelliten in der Lage sein folgende Aufgaben selbstständig bewältigen zu können: Bilddatenverarbeitung, Datenverwaltung, Datenverteilung, Organisation, Ressourcen-Verwaltung, Sensor-/Plattformsteuerung, Missionsscheduling und Missionsplanung. Zur Konstruktion eines solch komplexen Satellitensystems sind Beiträge von Wissenschaftlern und Ingenieuren aus vielen unterschiedlichen Bereichen notwendig [1].
Im Folgenden werden einige wichtige Missionen kurz vorgestellt, bei denen einige der oben genannten Funktionalitäten experimentell untersucht wurden. Auf dem am 22. Oktober 2001 gestarteten, vom DLR entwickelten Bispectral Infrared Detection Satelliten (BIRD) wurde eine multispektrale Klassifikation von Wasserflächen, Brache, Wolken, Städten und eine Feuerdetektion durchgeführt [7]. Ziel war es, thematische Karten an Bord des Satelliten zu erstellen und die Fähigkeiten eines On-Board Feuerdetektionssystems zu untersuchen.
Ein weitaus autonomeres System, und eines der am weitesten entwickelten Systeme in diesem Feld, ist das von der NASA entwickelte Autonomous Sciencecraft Experiment (ASE) welches auf dem Earth Observing One Satelliten (EO-1) geflogen ist. Der EO-1 Satellit wurde im November 2000 gestartet. Er ist unter anderem mit einem multispektralen Sensor ausgestattet und war der erste Satellit auf dem ein Hyperspektralsensor montiert wurde. Eine weitere Besonderheit von EO-1 ist die Cross-Track-Pointing Fähigkeit,

welche es trotz einer Wiederkehrrate von 16 Tagen ermöglicht ein Ziel fünf Tage hintereinander beobachten zu können[2] [8]. Mit dem ASE-Experiment konnte demonstriert werden, dass an Bord eine vollständig autonome Missionsdurchführung möglich ist. Dabei wurden Bilddaten an Bord des Satelliten untersucht und in Abhängigkeit dieser Analyse wurde autonom der weitere Missionsverlauf geplant und auch ausgeführt [9]. Im Detail wurden ausgewählte Zielgebiete im Hinblick auf bestimmte Ereignisse überwacht. Drei unterschiedliche Szenarien wurden untersucht: Flut-Detektion, Schnee-Wasser-Eis-Land Detektion und die Detektion vulkanischer Eruptionen [8, 10, 11]. Eine Fähigkeiten des Satelliten war es, bei bestimmten Ereignissen, beispielsweise bei der Entdeckung eines neuen Lavastromes, für die nachfolgenden Orbits selbstständig weitere Beobachtungen des entsprechenden Zielgebiets einplanen und durchgeführt zu können.

1.3 Anforderungen an ein System zur On-Board Bilddatenverarbeitung

Eine wichtige Anforderung, welche an eine On-Board Datenverarbeitung gestellt werden muss, ist, dass sie voll automatisch abläuft, um autonomes Handeln zu ermöglichen. Ein weiterer wichtiger Punkt für viele Anwendung ist die Echtzeitfähigkeit. Der Begriff Echtzeit ist nach DIN 44300 folgendermaßen definiert:

„Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.“

In dieser Beschreibung verstecken sich zwei wesentlichen Anforderungen, eine an das Verfahren, welches dem auszuführendem Programm zugrunde liegt, und eine an das System, auf dem der Algorithmus implementiert wird.

- Die erste Forderung ist, dass sich die Laufzeit des Verfahrens abschätzen lässt. Viele Verfahren sind von den zu bearbeitenden Daten abhängig und es können sehr große Unterschiede in der Bearbeitungszeit auftreten. Kann ein Datensatz in wenigen Minuten abgearbeitet werden, könnten für einen weiteren Datensatz der gleichen Größe mehrere Stunden benötigt werden. Man denke hier an Verfahren, welche in einer großen Datenbank einen bestimmten Wert finden müssen, der nicht indiziert ist, oder an Verfahren, welche eine möglichst kurze Verbindungsroute zwischen zwei Wegpunkten finden sollen.
- Die zweite wesentliche Anforderung an ein Echtzeitsystem ist, dass das System genügend Rechenleistung bereitstellen muss, um die Datenverarbeitung in der gewünschten Zeit durchführen zu können. Muss eine aufwendige Berechnung in kurzer

[2] eine Beobachtung pro Tag

Zeit durchgeführt werden, wird eine hohe Prozessierungsleistung benötigt. Gerade Videoanwendungen benötigen aufgrund der Größe der zu verarbeitenden Daten, der oft komplexen Rechenoperationen und der kurzen zur Verfügung stehenden Verarbeitungszeit, sehr große Rechenleistung [12]. Außerdem muss sichergestellt werden, dass die zur Datenverarbeitung benötigten Ressourcen auch zum passenden Zeitpunkt zur Verfügung stehen. Auf Software basierten Systemen, auf denen sich unterschiedliche Prozesse die zur Verfügung stehenden Ressourcen teilen, müssen hierfür oftmals aufwendige Vorkehrungen getroffen werden[3].

Softwarelösungen können oftmals die erforderlichen Echtzeitanforderungen nicht erfüllen. Dies liegt in erster Linie daran, dass die Prozessoren, welche auf Satelliten eingesetzt werden können, nicht genügend Rechenleistung zur Verfügung stellen können. Viele Anwendungen sind jedoch derart rechenaufwendig, dass selbst Hochleistungsprozessoren nicht in der Lage sind, genügend Rechenleistung zu liefern. Der Grund ist die Prozessorarchitektur von Standartprozessoren, welche darauf ausgelegt ist ein breites Aufgabenspektrum bearbeiten zu können.
Eine spezialisierte Hardwarelösung kann Abhilfe schaffen. Um einem hohen Datendurchsatz zur Verfügung zu stellen, werden beispielsweise zur Bildkompression auf Satelliten häufig Application Specific Integrated Circuits (ASICs) eingesetzt [2]. Diese liefern oftmals eine weitaus höhere Rechenleistung als mit Standartprozessoren oder sogar Digital Signal Prozessoren (DSPs) möglich wäre.
In den letzten Jahren werden auch Static Random Access Memory (SRAM) basierte FPGA Lösungen populärer, da sie günstiger und einfacher zu entwickeln sind als ASIC Lösungen und inzwischen oftmals genügend Leistung zur Verfügung stellen [2]. Ein großer Vorteil gegenüber ASICs ist ihre (Re-)Konfigurierbarkeit. Daher können Änderungen am System in der Entwicklungsphase und sogar innerhalb der Laufzeit einer Mission durchgeführt werden [2]. FPGAs werden bereits auf Satelliten eingesetzt, beispielsweise zur Bildkompression auf den Sentinel-2 Satelliten.

1.4 Wissenschaftliche Zielsetzung dieser Arbeit

Die Zielsetzung dieser Arbeit ist es, eine Methode zur Bildsegmentation bereitzustellen, welche vorteilhaft auf einer FPGA-Plattform implementiert werden kann und mit welcher Fernerkundungsdaten prozessiert werden können. Konkret sollen Grauwertbilder segmentiert werden und als Ergebnis ein Indexbild ausgegeben werden. Es soll außerdem die Qualität, die Leistungsfähigkeit und der Ressourcenverbrauch dieser Methode untersucht und ein Vergleich mit alternativen Methoden durchgeführt werden.
Für eine Bildsegmentation mit dem Ergebnis eines Indexbildes sind zwei grundlegende Prozessierungsschritte notwendig. Zuerst wird der lokale Zusammenhang ermittelt. Hierbei wird ausgehend von vorgegebenen Segmentationskriterien untersucht, ob benachbarte Bildpunkte in einem gemeinsamen Segment liegen oder nicht. Dies ist der eigentliche Segmentationsschritt. Das Ergebnis gibt jedoch nur Auskunft über den Zusammenhang

[3] beispielsweise ein Echtzeitbetriebssystem mit speziell angepasste Anwendungen

benachbarter Bildpunkte. Ob zwei beliebige Bildpunkte in einem gemeinsamen Segment liegen, ist noch nicht bekannt. Hierfür ist ein weiterer Prozessierungsschritt notwendig. Bei diesem werden alle Bildpunkte, welche in einem Segment liegen, mit einem eindeutigen gemeinsamen Index versehen. Dieser in der Bildverarbeitung grundlegende Schritt wird Connected Component Labeling (CCL) genannt [13].
Im Detail sollen in dieser Arbeit folgende Einzelfragen geklärt werden. Es soll zuerst geklärt werden, welchen Anforderungen der Segmentationsprozess genügen muss. Beispielsweise muss sichergestellt sein, dass auch umfangreiche und komplexe Datenmengen in einem relativ kurzen Zeitraum prozessiert werden können. Auch muss festgelegt werden, nach welchem Segmentationskriterium die Segmentation durchgeführt werden soll. Anschließend muss ermittelt werden, wie der Segmentationsprozess durchgeführt werden kann. Die hierfür benötigten Verfahren sollten sich prinzipiell gut für eine FPGA-Implementation eignen und sind derart zu modifizieren, dass sie die Vorteile, welche FPGAs bieten, nutzen können.
Die Verarbeitungsgeschwindigkeit und die Echtzeiteigenschaften der (Einzel-)Verfahren[4] sind ebenfalls zu ermitteln. Auch soll eine qualitative Einschätzung des Segmentationsverfahrens durchgeführt werden. Insbesondere soll der qualitative Unterschied zwischen dem Orginalverfahren und der für das FPGA optimierten Version ermittelt werden. Wenn möglich, soll außerdem ein Vergleich mit verschiedenen alternativen Methoden im Hinblick auf Laufzeit, Echtzeitfähigkeit, Ressourcenverbrauch und Qualität durchgeführt werden.
Abschließend muss ein Prototyp erstellt werden, um die prinzipielle Durchführbarkeit zu bestätigen und weitere Eigenschaften wie Ressourcenverbrauch sowie die Verarbeitungsgeschwindigkeit und Echtzeitfähigkeit ermitteln zu können. Im Folgenden sind diese Punkte nochmals stichpunktartig aufgeführt.

Teil 1: Segmentation

- Erstellung eines Anforderungsprofils
- Wahl eines für das FPGA prinzipiell gut geeigneten Segmentationsverfahrens für Fernerkundungsdaten
- Optimierung des Verfahrens im Hinblick auf eine möglichst günstige FPGA-Implementation
- Abschätzung der benötigten Ausführungszeit, der Echtzeitfähigkeit und der Qualität des Segmentationsverfahrens
- Vergleich des Verfahrens mit alternativen Methoden, hinsichtlich der Segmentationsqualität, der Laufzeit, der Echtzeitfähigkeit und des Ressourcenverbrauchs
- Implementierung eines Prototypen auf einer FPGA Plattform zur Ermittlung der Verarbeitungsgeschwindigkeit, Echtzeitfähigkeit und des Ressourcenverbrauchs

[4] Segmentationsprozess, CCL

Teil 2: Connected Component Labeling

- Erstellung eines Anforderungsprofils
- Wahl eines für das FPGA prinzipiell gut geeigneten CCL Verfahrens für die Anwendung in der Fernerkundung
- Optimierung des Verfahrens im Hinblick auf eine möglichst günstige FPGA-Implementation
- Abschätzung der benötigten Ausführungszeit und Echtzeitfähigkeit des CCL Verfahrens
- Vergleich der Laufzeit und Echtzeitfähigkeit mit der ursprünglichen und mit alternativen Methoden
- Implementierung eines Prototypen auf einer FPGA Plattform zur Ermittlung der Verarbeitungsgeschwindigkeit, Echtzeitfähigkeit und des Ressourcenverbrauchs

1.5 Gliederung der Arbeit

Kapitel 1: Einleitung Im Einleitungskapitel wird auf die Motivation welche hinter diese Arbeit steht und auf den Bereich On-Board Bilddatenverarbeitung, in welchem diese Arbeit angesiedelt ist, eingegangen. Abschließend wird die wissenschaftliche Zielsetzung und der Aufbau dieser Arbeit erläutert.

Kapitel 2: Bildsegmentation In diesem Kapitel wird zuerst dargelegt, was unter der Segmentation eines Bildes zu verstehen ist. Anschließend werden aufgrund des Umfeldes, des Einsatzgebietes und der Zielsetzung, Anforderungen an die Segmentationsmethode abgeleitet. Im Weiteren wird eine Übersicht über gängige Segmentationsverfahren und Hardware-, insbesondere FPGA-Implementationen, gegeben. Ausgehend vom Anforderungsprofil wird nun eine Methode gewählt, welche als Grundlage der FPGA-Implementation verwendet werden soll.

Kapitel 3: Gap-Segmentation In diesem Kapitel wird das Gap-Segmentationsverfahren vorgestellt, welches im Rahmen dieser Arbeit entwickelt wurde. Die grundsätzlichen Prozessierungsschritte werden näher erläutert, insbesondere die kantenerhaltende Gap-Glättung. Anschließend wird auf den grundlegenden Aufbau eines FPGAs und dessen Programmierung eingegangen. Es wird näher dargestellt, wie der Gap-Glättungsprozess auf dem FPGA implementiert wurde. Zum Schluss wird aufgezeigt, welche Leistung mit der FPGA-Implementation erzielt werden kann, auch im Vergleich zu einer Softwareimplementation.

Kapitel 4: Qualitätsuntersuchung In diesem Kapitel wird erläutert, wie die qualitative Leistung eines Segmentationsverfahrens bestimmt werden kann. Anschließend werden (optimale) Parameter für das Gap-Segmentationsverfahren ermittelt und seine qualitative Leistungsfähigkeit bestimmt. Es wird ebenfalls ein Vergleich mit alternativen Segmentationsverfahren durchgeführt. Zum Schluss wird demonstriert, wie sich die mit der Qualitätsuntersuchung gewonnenen Ergebnisse auf reale Daten übertragen lassen.

Kapitel 5: Connected Component Labeling In diesem Kapitel wird dargestellt, wie das CCL großer komplexer Bilder auf einem FPGA durchgeführt werden kann. Zuerst wird dargelegt, was unter CCL zu verstehen ist und es werden gängige Verfahren vorgestellt. Ausgewählte Verfahren werden auf ihre Eignung untersucht. Im Weiteren wird der Algorithmus vorgestellt, welcher in dieser Arbeit entwickelt wurde. Im diesem Rahmen wird auch bewiesen, dass dieser Algorithmus fehlerfrei ist. Anschließend wird dargelegt, wie das CCL Verfahren auf einem FPGA implementiert werden kann, und genau untersucht, welche Verarbeitungszeit für das CCL benötigt wird. Um die Leistungsfähigkeit der FPGA Lösung einschätzen zu können, wurde ebenfalls ein Vergleich mit einer Softwareimplementation des verwendeten CCL-Verfahrens durchgeführt.

Kapitel 6: Schlussbemerkung Abschließend werden die Ergebnisse zusammengefasst, ein kurzer Ausblick gegeben, und ein Fazit gezogen.

2 Bildsegmentation

2.1 Segmentierung von Bilddaten

Unter einer Bildsegmentation versteht man die flächenhafte Gliederung eines Bildes [13, 14, 15]. Die Bildpunkte werden anhand vorgegebener Kriterien zu Regionen, den Segmenten, zusammengefasst. Jeder Bildpunkt wird einem Segment zugeordnet. Die Segmente überdecken folglich das gesamte Bild und überlappen sich nicht. Zusätzlich wird in den allermeisten Fällen gefordert, dass ein Segment eine örtlich zusammenhängende Region darstellt. Im mathematischen Sinne ist eine Segmentation eines Bildes eine Partitionierung des Ortsraumes, wobei zusätzlich gefordert wird, dass die einzelnen Komponenten zusammenhängend sind. Eine exakte Definition wird in Abschnitt 4.2 gegeben.

Die Kriterien, nach denen eine Segmentation durchgeführt werden soll, sind je nach Anwendung entsprechend zu wählen. Sie beruhen in den allermeisten Fällen auf Homogenitäts- und Heterogenitätsmerkmalen. Eines der gebräuchlichsten Merkmale, welches auch in dieser Arbeit verwendet wird, ist das Merkmal des mittleren Grauwerts. Die Bildpunkte eines Segments sollen ähnliche Grauwerte besitzen und sich von den Grauwerten benachbarter Segmente signifikant unterscheiden.

Das Ergebnis einer Bildsegmentation lässt sich auf unterschiedlichste Arten darstellen. Eine der häufigsten Darstellungsarten ist es, den Rand der einzelnen Segmente anzugeben, beispielsweise in Form eines Polygonzugs. Hierbei können die Bildpunkte jedoch noch nicht direkt den einzelnen Segmenten zugeordnet werden. Mithilfe eines zusätzlichen Verarbeitungsschrittes, dem Connected Component Labeling, kann eine Labelmaske erstellt werden. Hierbei wird jedem Bildpunkt ein Zahlenwert zugeordnet, welcher eineindeutig seiner Zusammenhangskomponente, dem Segment, entspricht. Dieser Zahlenwert wird auch als Label oder Index bezeichnet. Dies ist auch die in dieser Arbeit verwendete Darstellungsform, da hierdurch unmittelbar abgelesen werden kann, welchem Segment ein Bildpunkt zugeordnet ist. Diese Information wird für viele weiterführenden Verarbeitungsschritte benötigt, beispielsweise für eine Objekterkennung.

In Abbildung 2.1 ist eine Segmentation einer panchromatischen WorldView-1 Szene in der Nähe von Peking gezeigt. Auf der linken Seite befindet sich das Satellitenbild. Hierauf sind unterschiedliche Bildbereiche, wie Ackerflächen, Wasserflächen, Stadtbereiche, ein Fluss und einzelne Gebäudekomplexe zu erkennen. Diese Bildbereiche lassen sich in erster Linie dadurch unterscheiden, dass ihre Bildpunkte ähnliche Grauwerte besitzen, welche sich größtenteils signifikant von ihrer Umgebung unterscheiden. Sie genügen daher dem Kriterium des mittleren Grauwertes, nach welchem in diesem Beispiel segmentiert wurde.

Auf der rechten Seite ist die Labelmaske dargestellt. Die Segmente sind hierauf unter-

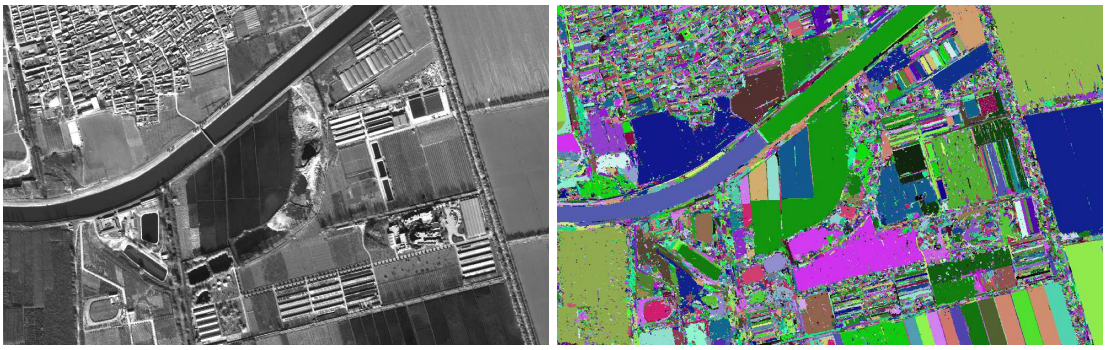


Abbildung 2.1: Segmentation einer panchromatischen World View 1 Szene, links die Satellitenbildaufnahme, rechts die Labelmaske

schiedlich eingefärbt[1]. Es ist zu erkennen, dass die Segmente örtlich zusammenhängende Regionen bilden. Beispielsweise zerfällt der Fluss in zwei Segmente, da er durch eine Brücke getrennt wird.
Viele Bildbereiche, wie Wasserflächen, Ackerflächen und Gebäudekomplexe werden subjektiv betrachtet gut separiert. Einige wurden jedoch vom subjektiven Standpunkt aus weniger gut segmentiert. Beispielsweise gibt es ein Vielzahl von kleinen Segmenten, die auf den ersten Blick störend wirken.

2.2 Was ist eine gute Segmentation?

In Abbildung 2.2 finden sich zwei weitere Segmentationsergebnisse der WorldView-1 Szene aus Abbildung 2.1.

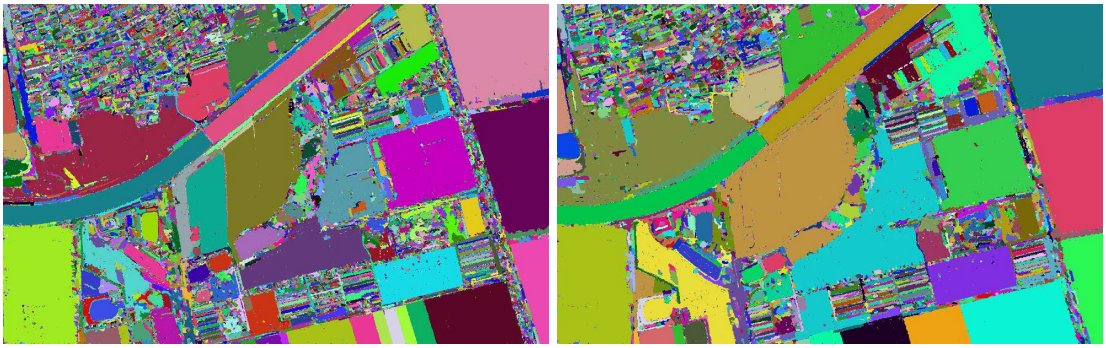


Abbildung 2.2: Weitere Segmentationen der WorldView-1 Szene aus Abbildung 2.1

Hierauf sind deutlich weniger Segmente vorhanden, die subjektiv störend wirken. Außer-

[1] Aufgrund der beschränkten Anzahl an darstellbaren Farbwerten ist verschiedenen Segmenten teilweise der gleiche Farbwert zugeordnet

dem scheinen die zwei Segmentationen insgesamt etwas gröber zu sein. Eine geeignete Segmentation dieser Szene ist natürlich anwendungsabhängig. Hier jedoch stößt man auf zwei fundamentale Probleme der Bildsegmentation:

1. Was ist eine gute Segmentation (in Abhängigkeit einer Anwendung)?
2. Wie erhalte ich diese Segmentation?

Diese Fragen stellen sich bei jeder Segmentationsaufgabe und ein generelles Verfahren, diese zu behandeln, gibt es bis jetzt nicht. Schon der erste Punkt stellt in vielen Fällen eine fast unlösbare Aufgabe dar. Denn auch wenn man eine konkrete Anwendung im Auge hat, hat man oftmals nur eine grobe Vorstellung wie das gewünschte Segmentationsergebnis auszusehen hat. Und überdies muss ein möglichst exaktes und konkretes Segmentationskriterium formuliert werden, welches diese Vorstellung widerspiegelt. In der Praxis wendet man daher oft verfügbare Segmentationsmethoden an und überprüft ihre Eignung. Da die in dieser Arbeit entwickelte Segmentationsmethode ein Grundlagenwerkzeug für verschiedenste Anwendung ist, soll nach einem Kriterium segmentiert werden, welches möglichst allgemein angewendet werden kann. Ein sehr grundlegendes Kriterium ist der mittlere Grauwert, der auch in dieser Arbeit verwendet wird. Andere Kriterien, basierend auf Rauigkeits- oder Texturmerkmalen, können behandelt werden, indem ihr Rauigkeits- oder Texturmaß in einem Grauwertbild codiert wird.
Mit der in dieser Arbeit vorgestellten Segmentationsmethode lassen sich auch Bilder mit mehreren Kanälen bearbeiten, denn beim der Segmentation ist nur die Differenz der Grauwerte benachbarten Bildpunkten entscheidend. Anstatt den Betrag der Grauwertdifferenz zu verwenden, kann bei mehrkanaligen Bildern der Betrag einer Vektordifferenz herangezogen werden.
Für eine objektive Qualitätsanalyse, welche in dieser Arbeit durchgeführt werden soll, ist ein exakt definierter Gütebegriff notwendig. In Kapitel 4.3 wird auf die in diesem Abschnitt aufgeworfenen Fragen näher eingegangen. Um aber überhaupt eine geeignete Segmentationsmethode bereitstellen zu können, müssen die an diese Methode gestellten Anforderungen möglichst exakt formuliert werden.

2.3 Anforderungen an die Segmentationsmethode

Aufgrund des Umfeldes, des Einsatzgebiets und der Zielsetzung ergeben sich, wie in den vorherigen Abschnitten bereits angesprochen, drei grundlegende Anforderungen:

1. Automatische Segmentation einkanaliger Fernerkundungsbilder
2. Erfüllung grundlegender Echtzeitanforderung
3. Eignung für eine FPGA-Implementation

Ausgehend von diesen Punkten werden im Folgenden konkrete Anforderungen abgeleitet. Der erste wichtigste Punkt ist, welche Daten verarbeitet werden müssen und welche Anforderungen an das Resultat gestellt werden. Die Segmentationsmethode soll einzelne

Bildbereiche aufgrund räumlicher Merkmale für spätere Verarbeitungsschritte aus einer Gesamtszene herauslösen. Diese Merkmale sollen möglichst grundlegend und allgemein sein, um an die Bedürfnisse möglichst vieler Anwendungen angepasst werden zu können. Wie im vorherigen Abschnitt bereits aufgezeigt, ist ein grundlegendes Merkmal, nach dem Bildbestandteile voneinander getrennt werden können ihr mittlerer Grauwert. Die Segmentationsmethode soll daher benachbarte Bildpunkte, welche einen ähnlichen Grauwert haben, zusammenfassen und Bildpunkte, die sich signifikant im Grauwert unterscheiden, trennen. Wichtig ist hierbei zu bemerken, dass das Merkmal des mittleren Grauwerts von den Absolutwerten der Bildpunkte eines Segments unabhängig ist. Eine Einteilung aufgrund absoluter spektraler Merkmale wird üblicherweise im Rahmen einer Bildklassifikation durchgeführt. Die Methode muss aufgrund des Anspruches nach einer universellen Verwendbarkeit prinzipiell auch von der Größe, Form und Anzahl der Segmente unabhängig sein. Der Segmentationsprozess muss wegen des geplanten Einsatzfeldes vollständig automatisch durchgeführt werden können. Außerdem soll die Möglichkeit bestehen auch Satellitenaufnahmen mit einer hohen Anzahl an Bildpunkten zu verarbeiten. Als Anhaltspunkt wurden panchromatische Landsat-7 Szenen herangezogen, bei welchen über 10000*10000 Bildpunkte verarbeitet werden müssen. Es wurde außerdem davon ausgegangen, dass acht Bit zur Darstellung eines Bildpunkts ausreichend sind, um den spektralen Dynamikumfang bei ausreichender spektraler Auflösung darstellen zu könne. Die Farbtiefe kann jedoch bei der in dieser Arbeit vorgestellten Methode, falls benötigt, beliebig angepasst werden.

Der zweite Punkt ist die Echtzeitanforderung. Um Echtzeitkriterien nachweislich erfüllen zu können, muss auf alle Fälle eine obere Schranke für die Laufzeit der Methode bekannt sein. Natürlich werden auch Anforderungen an die Höhe der Laufzeit gestellt. Diese ist von der gewünschten Anwendung abhängig. Jedoch ist es wichtig, eine vernünftige Größenordnung anzugeben, um die Entwicklung in die richtige Richtung zu führen. Der Fokus liegt auf der Bilddatenverarbeitung und nicht auf Videoprozessierung. Nach Abwägung verschiedener möglicher Einsatzszenarien und des technisch Möglichen, wurde festgesetzt, dass der Gesamtprozess der Segmentation mit realistischen Hardwareanforderungen bei moderater Bildgröße[2] in wenigen Sekunden und bei sehr großen Bildern[3] in wenigen Minuten abgeschlossen werden sollte.

Der dritte Punkt ergibt sich aus der Anforderung, dass die Methode auf einem FPGA möglichst performant und ressourcenschonend implementiert werden soll. Eine hohe Verarbeitungsleistung auf einem FPGA ist in erster Linie durch parallele Datenverarbeitung zu erreichen. Daher sollte sich die Methode gut parallelisieren lassen. Ein Voraussetzung für eine Parallelisierung ist, dass sich die Gesamtaufgabe in möglichst viele Teilaufgaben zerlegen lässt, die unabhängig voneinander bearbeitet werden können.

Ein weiterer wichtiger Punkt im Hinblick einer FPGA-Implementation ist, dass der Programmverlauf möglichst statisch sein sollte, denn das Programm muss mit allen seinen möglichen Programmabläufen auf dem FPGA abgebildet werden. Für jede Programmverzweigung wird ein Schaltkreis benötigt. Verzweigungen welche selten oder überhaupt

[2] $<$ 2000 * 2000 Pixel
[3] 10000*10000 Pixel

nicht benutzt werden, sollten vermieden werden. Für den Entwickler bedeutet dies, möglichst keine if- oder switch- Anweisungen zu verwenden, insbesondere falls sie komplexe Ausdrücke enthalten.
Dies führt direkt auf den letzten Punkt, der Komplexität der verwendeten Verfahren. Das verwendete Verfahren sollte mit möglichst wenigen und möglichst einfachen Rechenoperationen umsetzbar sein. Zur Implementation einer Divisionseinheit oder einer Wurzelberechnungseinheit beispielsweise sind große Schaltnetzwerk nötig, welche auf einem FPGA realisiert werden müssten. Erschwerend kommt hinzu dass sich komplexe Rechenoperation negativ auf die maximale Taktfrequenz, mit welchem das FPGA betrieben werden kann, auswirken. Komplexe Rechenoperationen sollten daher soweit es möglich ist vermieden werden. Zusammenfassend sind folgende Anforderungen herausgearbeitet worden:

- Segmentation von Grauwertbilder mit einer Farbtiefe von acht Bit, nach dem Merkmal des mittleren Grauwerts
- Unabhängigkeit von Form, Größe und Anzahl der Segmente
- Vollständig automatischer Ablauf
- Fähigkeit prinzipiell sehr große Bilder verarbeiten zu können
- Eine obere Schranke der Laufzeit der Methode muss bekannt sein
- Die maximale Gesamtlaufzeit der Methode soll bei kleineren Bildern maximal einige Sekunden betragen, bei größeren Bildern im Bereich einiger Minuten liegen
- Parallelisierbare Datenverarbeitungsoperationen
- Wenige Programmverzweigungen
- Wenige Verarbeitungsschritte
- Keine komplexen Rechenoperationen

Dies sind die Punkte, welche bei der Wahl einer geeigneten Segmentationsmethode zu beachten sind. Diese Punkte stehen größtenteils auch in Konkurrenz zueinander. Es muss letztendlich ein Kompromiss zwischen der Qualität, der Verarbeitungszeit und des Ressourcenverbrauchs gefunden werden. Im nächsten Abschnitt werden die gängigsten Segmentationsverfahren vorgestellt.

2.4 Segmentationsverfahren

In diesem Abschnitt soll eine Übersicht über grundlegende Segmentationsverfahren gegeben werden, wie sie beispielsweise in [16] und vielen anderen Veröffentlichungen zu finden sind. Außerdem erfolgt eine erste Abschätzung, inwieweit diese Verfahren prinzipiell die

Kriterien, welche im letzten Abschnitt formuliert wurden, erfüllen.

Schwellenwertverfahren
Eine der am häufigsten benutzten Methoden stellen Schwellenwert-basierte Segmentationsverfahren dar. Bei diesen Verfahren erfolgt zuerst eine Einteilung des Farbraumes[4] eines Bildes in verschiedene Klassen. Die einzelnen Bildpunkte werden dann ausschließlich aufgrund ihres Farbwertes der entsprechenden Klassen zugeordnet und benachbarte Bildpunkte, die der gleichen Klasse angehören, zu einem Segment zusammengefasst. Die Klasseneinteilung des Farbraums erfolgt in vielen Fällen durch die Vorgabe vordefinierter Schwellenwerte. Da die Segmentation des Bildes von den absoluten Farbwerten abhängig ist, kommt diese Methode aufgrund der gestellten Anforderungen (Abschnitt 2.3) nicht in Frage. Es kann jedoch die Farbverteilung des Bildes herangezogen werden, um eine speziell an das Bild angepasste Klasseneinteilung des Farbraums vorzunehmen. Hierzu werden je nach Einsatzgebiet allgemeine Klassifikationsmethoden wie das k-Means Clustering herangezogen, aber auch spezielle Verfahrensweisen, wie die Otsu-Methode [17], welche bei bimodalen Segmentationsproblemen[5] verwendet wird. Diese Verfahren funktionieren gut, falls zur Segmentation einige wenige Klassen im Farbraum ausreichen und diese sich signifikant unterscheiden. Falls viele Bildbereiche vorhanden sind, welche ähnliche mittlere Grauwerte besitzen, kann, wie bereits in [16] gezeigt, kein vernünftiges Ergebnis mehr erzielt werden. Da dies auf viele Anwendungen in der Fernerkundung zutrifft, kommt diese Art von Methoden in dieser Arbeit nicht in Betracht. Dies könnte zwar durch lokale Schwellwertverfahren, bei denen das Bild in mehrere Bereiche eingeteilt und die Schwellenwerte jeweils für eine Region bestimmt werden oder noch dynamischere Schwellenwertverfahren abgemildert werden. Aufgrund der hohen Komplexität wird auf derartige Verfahren jedoch nicht näher eingegangen.

Regionorientierte Verfahren:
Die zweite große Klasse von Segmentationsverfahren stellen regionorientierte Verfahren dar. Benachbarte Bildpunkte, welche ähnliche Eigenschaften aufweisen, werden dabei zu Regionen zusammengefasst. Hierzu wird der Ortsraum eines Bildes oft als (math.) Graph aufgefasst, wobei die einzelnen Bildpunkte den Knoten des Graphen entsprechen. Haben benachbarte Bildpunkte ähnliche Eigenschaften, werden sie zusammengefasst. Der Unterschied verschiedener Vertreter dieser Klasse beruht auf den verwendeten Ähnlichkeitseigenschaften, insbesondere ihrer Lokalität. Die einfachsten Verfahren vergleichen benachbarte Bildpunkte bezüglich eines gewählten Kriteriums, zum Beispiel ihrer Grauwertdifferenz, direkt miteinander. Aufgrund ihrer Lokalität eignen sich derartige Verfahren vom technischen Standpunkt aus hervorragend für eine FPGA-Implementation, sind aber andererseits sehr störanfällig. Weniger empfindlich sind Methoden, welche kleine Umgebungen der Bildpunkte, zum Beispiel den Mittelwert eines 3×3-Bildfensters, zur Bestimmung der Zusammengehörigkeit heranziehen. Diese eignen sich aufgrund ihrer ho-

[4] als Farbraum wird in dieser Arbeit der Zielbereich einer Bildfunktion bezeichnet (vgl. Abschnitt 4.2.1). Es kann daher vorkommen, dass in dieser Arbeit auch im Falle eines Grauwertbildes von einem Farbraum gesprochen wird.
[5] Vordergrund- Hintergrundtrennung

hen Lokalität und eines fest vorgegebenen Untersuchungsbereichs ebenfalls gut für eine FPGA-Implementation.
Zu den globalen und weitaus dynamischeren Verfahren zählen beispielsweise die bekannten Region Growing Verfahren. Bei diesen wird ausgehend von vorgegebenen Startregionen ein Wachstumsprozess eingeleitet, welche durch Hinzunahme geeigneter Bildpunkte schrittweise erweitert werden. Als Eignungskriterium werden beispielsweise die Mittelwerte der bereits gebildeten Regionen herangezogen. Diese Verfahren sind jedoch sehr stark von der Wahl der Startregionen abhängig. Es gibt daher wiederum Verfahren, die selbständig günstige Startregionen bestimmen.
Ein anderes globales regionorientiertes Verfahren ist das sogenannte Split and Merge Verfahren. Hier wird, im Gegensatz zu Region Growing Verfahren die Anfangsannahme getroffen, dass der gesamte Ortsraum des Bildes ein Segment bildet. Anschließend wird die Zulässigkeit dieser Annahme, beispielsweise durch Berechnung der Standartabweichung, überprüft. Trifft sie nicht zu, wird das Segment in Untersegmente zerlegt, standardmäßig in vier Quadrate. Mit diesen Segmenten wird anschließend ebenso verfahren. Ist keine weitere Zerlegung des Bildes mehr sinnvoll, werden analog Segmente welche eine hohe Ähnlichkeit aufweisen zusammengefasst.
Aufgrund ihrer Dynamik sind solche Verfahren nur eingeschränkt für eine FPGA-Implementation geeignet. Der Vorteil globaler Methoden ist jedoch, dass sie im Gegensatz zu lokalen Methoden weniger störanfällig sind, und auch lokal stärker strukturierte Flächen zusammengefasst werden können.

Kantenorientierte Verfahren
Die dritte große Klasse von Segmentationsmethoden stellen kantenorientierte Verfahren dar. Im Gegensatz zu den regionorientierten Verfahren, bei denen Bildpunkte in erster Linie anhand bestimmter Homogenitätskriterien zu Regionen zusammengefasst werden, erfolgt hier eine Trennung von Bildpunkten vornehmlich nach Heterogenitätsmerkmalen. Meistens werden hierzu die Kanten zum Beispiel mit einem Sobel-Operator oder mithilfe des Canny-Algorithmus [18] bestimmt. Ein Nachteil kantenorientierter Verfahren ist, dass die Kantendetektion meistens auf sehr lokalen Kriterien beruht und bereits kleine Lücken in den Kantenzügen genügen, damit zwei benachbarte Regionen zusammenwachsen. Daher werden oftmals nach einer Kantenextraktion Kantenverfolgungsalgorithmen angewendet, um diese Lücken zu schließen.

Modellbasierte Verfahren:
Bei diesen Verfahren steht ein a-priori Wissen über die möglicherweise auftretenden Segmente zur Verfügung. Zu bekannten Vertreter dieser Klasse gehören das Template-Matching oder die Hough-Transformation. Da kein a-priori Wissen vorausgesetzt werden kann, können diese Art von Verfahren in dieser Arbeit nicht verwendet werden.

In vielen Fällen können Segmentationsverfahren nicht eindeutig einer dieser drei Klassen zugeordnet werden, da sie mehr oder weniger ausgeprägte Charakteristika verschiedener Klassen besitzen.
Abschließend sollen noch drei moderne Methoden erwähnt werden, welche sowohl kanten-

als auch regionbasierte Merkmale besitzen. Die Wasserscheidesegmentation [19], die Mean-Shift Segmentation [20] und Segmentationsmethoden beruhend auf dem Mumford-Shah Modell [21, 22, 23, 24]. Die Wasserscheidesegmentation und die Mean-Shift Segmentation beruhen im Wesentlichen auf Gradientenabstiegsverfahren. Das Mumford-Shah Modell stellt eine Formulierung des Merkmals des mittleren Grauwertes als Energiefunktion dar. Das Ziel ist es, diese zu minimieren. Auf dieses Modell wird in Kapitel 4.2.5 näher eingegangen. Aufgrund der hohen Komplexität wurden diese Methoden in dieser Arbeit nicht weiter herangezogen.

2.5 Hardware/FPGA-Implementationen

In der Literatur finden sich nur wenige Hardwareimplementationen, welche den Segmentationsprozess komplett durchführen. In der 2004 erschienen Arbeit von Neuenhahn et al. [25] wurde eine Implementation eines Wasserscheidesegmentationsverfahrens auf einer experimentellen Plattform für Videoprozessierung, basierend auf einem Altera APEX20K400 FPGA, vorgestellt. Hiermit konnte eine Echtzeitverarbeitung eines Videostroms mit einer Bildgröße von 352*288 Pixel mit 20 Bildern pro Sekunde (FPS) durchgeführt werden. Auf die genaue Umsetzung und die Leistungsfähigkeit dieses Verfahrens wurde in dieser Arbeit nicht näher eingegangen.
In der 2005 erschienen Arbeit von Bannister et al. [26] wurde eine FPGA-Implementation einer Segmentationsmethode, welche auf dem Modell von Bayes[6] beruht, vorgestellt. Das Ziel war es, die Methode, welche für die Segmentation eines Bildes von 256*256 Pixel auf einem Intel Pentium 4 @ 2.4 GHz Prozessor acht Minuten benötigt, zu beschleunigen. Es stellte sich jedoch heraus, dass einige Teile des Algorithmus ungeeignet für eine FPGA-Implementation sind. Diese Verarbeitungsschritte wurde daher auf einem PC ausgelagert. Aufgrund der Komplexität des Designs, konnte selbst auf einem damals relativ leistungsfähigen Virtex-2 FPGA nur eine Taktfrequenz von 57 MHz erreicht werden. Insgesamt lief die finale Implementation in etwa 40% langsamer als die Softwareversion und die gewünschte Beschleunigung konnte nicht erreicht werde.
In der Arbeit von Yamaoka et al. [29] (2006) wurde eine FPGA-Implementation eines Bildsegmentationsverfahrens vorgestellt, welches Teil eines Echtzeit-Objektverfolgungssystems ist. Mit diesem System können gleichzeitig über 200 Objekte in einem Videostrom der Auflösung 80*60 Pixel verfolgt werden. Bei einer Taktfrequenz von 20 MHz konnten 30 Bilder pro Sekunde verarbeitet werden. In [30, 31] (2005, 2010) werden ebenfalls FPGA-Implementationen von Segmentationsmethoden in diesem Anwendungsfeld vorgestellt. Eine genaue Einschätzung der Leistungsfähigkeit dieser Segmentationsmethoden war allerdings nicht möglich.
In [32] (2011) wird die Implementation eines Mean-Shift Filters vorgestellt, der als Grundlage einer Segmentation benutzt werden kann. Dieser wurde auf einem relativ leistungsfähigen Xilinx Virtex-4 implementiert. Auch dieses Verfahren wurde für Videoprozessierung konstruiert, und es konnten bei einer Frequenz von 138,8 MHz, bei einer Bildgröße von

[6] Standartmodell der Stochastik, welche auf der Bayes-Formel zur Berechnung bedingter Wahrscheinlichkeiten beruht [27, 28]

768*512 Pixel, mehr als 50 FPS erreicht werden. Das in dieser Arbeit beschriebene Verfahren ist relativ komplex. Es wurden keine genauen Angaben über die Filterleistung gemacht.
Ein weitaus größere Anzahl an Veröffentlichungen findet sich zum Thema Kantenextraktion. In [33] ist beispielsweise eine Implementation des Sobel-Operators gezeigt, in [34, 35] findet man Umsetzungen des komplexeren Canny-Filters. Aufgrund ihrer Lokalität sind diese Filter hervorragend für eine FPGA-Implementation geeignet, sie stellen allerdings nur einen Teil des Segmentationsprozesses dar.
Leider konnte keiner dieser Ansätze direkt benutzt werden. Das lag in erster Linie daran, dass ihre Leistungsfähigkeit und Eignung sich anhand der vorhandenen Informationen nur ungenügend abschätzen ließ und die Systeme nicht verfügbar waren. In den meisten Fällen findet sich außerdem nur eine mehr oder minder grobe Beschreibung der Implementation, ein Nachbauen ist also nicht ohne Weiteres möglich.
Die hier vorgestellten Methoden wurden außerdem bis auf [26] im Hinblick auf eine Videoprozessierung entworfen, welche andere Ansprüche stellt. Sie lieferten jedoch wertvolle Hinweise, wobei [25] und [26] der hier vorliegenden Problemstellung am nächsten kamen.

2.6 Wahl einer geeigneten Segmentationsmethode

In den letzten beiden Abschnitten wurde eine Übersicht über grundlegende Segmentationsverfahren und Beispiele von FPGA-Implementationen gegeben. Es gibt viele Verfahren, von denen die meisten in Software realisiert und nicht für Fernerkundungsanwendungen entwickelt wurden. Eine Einschätzung ihrer Eignung fällt sehr schwierig aus. Ein Hauptgrund hierfür ist, dass diese Verfahren nicht zur Verfügung standen und erst implementiert hätten werden müssen.
In dieser Arbeit wurde ein Segmentationsverfahren von Jahn [36, 37] als Grundlage verwendet. Die Gründe hierfür sind, dass dieses in der Doktorarbeit von Halle [38] ausführlich beschrieben und speziell für die Anwendung auf Fernerkundungsbilddaten, auch im Rahmen einer möglichen On-Board Datenverarbeitung und der Realisierung auf einer parallelen Prozessierungseinheit, untersucht und ausgewählt wurde. Dies entspricht der geforderten Zielsetzung und es wurde davon ausgegangen, dass diese Methode grundsätzlich geeignet ist. Aufgrund der guten Dokumentation konnte überprüft werden, dass die vorgestellte Segmentationsmethode, wie im Folgenden beschrieben, fast alle der in Abschnitt 2.3 gestellten Anforderungen erfüllt.
Das Segmentationsverfahren gehört der Klasse der regionorientierten Verfahren an. Es wird eine Segmentation von Grauwertbildern nach dem Merkmal des mittleren Grauwerts durchgeführt und ein Zusammenhangsgraph erzeugt. Durch diesen wird, wie in Abbildung 2.3 dargestellt, der Zusammenhang direkt benachbarter Bildpunkte codiert.
Das Verfahren ist unabhängig von den absoluten Grauwerten, von der Form und Anzahl der Segmente. Es ist außerdem ein lokales Verfahren, welches auf einem 3×3-Bildfenster arbeitet. Es ist laut den Untersuchungen aus [38] trotzdem unempfindlich gegenüber Störungen, wie beispielsweise Bildrauschen. Diese Eigenschaft wurde in den Arbeiten von Jahn/Halle allerdings nicht objektiv überprüft, sondern nur anhand einiger Beispiele de-

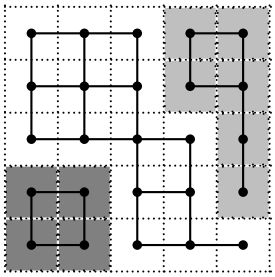


Abbildung 2.3: Beispiel eines Zusammenhangsgraph, die Grautöne dienen nur zur Veranschaulichung

monstriert. Die Störunempfindlichkeit wird durch eine im ersten Schritt durchgeführte kantenerhaltende Glättung erreicht, bei welcher teilweise über 50 hintereinander folgende Filterläufe ausgeführt werden. Da aber eine Verarbeitung von einem Pixel pro Takt, wie später auch gezeigt wird, möglich erscheint, wurde davon ausgegangen, dass mit geeigneter Hardware leicht über 50 Filterläufe pro Sekunde bei einem Bild von 1000*1000 Pixel durchgeführt werden könnten und dadurch die Anforderungen an die Verarbeitungsgeschwindigkeit erfüllt würden. Da die Laufzeit eines Filterlaufs unabhängig vom Bildinhalt ist, und die Anzahl der Filterläufe fest vorgegeben werden kann, kann die Gesamtlaufzeit des Segmentationsprozesses exakt angegeben werden.
Die Methode läuft außerdem vollkommen automatisch ab, und es können ohne Einschränkung[7] auch sehr große Bilder verarbeitet werden.
Das einzige Problem ist die technische Umsetzung. Der Programmablauf ist sehr statisch. Einige Verarbeitungsschritte sind jedoch sehr komplex. Es werden außerdem zur Berechnung Dezimalzahlen verwendet und einige relativ verschachtelte Ausdrücke wie $\frac{t^2}{t^2+x^2}$, welche für eine FPGA-Implementation nicht wünschenswert sind. Eine genau Darstellung dieses Verfahrens findet sich im Abschnitt 4.6.1. Es wurde daher letztendlich von dieser Methode ausgehend eine Methode entwickelt, die besser für eine FPGA-Implementation geeignet ist.

[7] Komplexität des Verfahrens bezüglich Laufzeit und Speicherverbrauch linear relativ zur Anzahl der Bildpunkte

3 Gap Segmentation

Im letzten Kapitel wurde erklärt, was unter einer Bildsegmentation zu verstehen ist, welche Anforderung an ein Segmentationsverfahren gestellt werden, welche grundlegenden Verfahrensweisen es gibt, und es wurde eine Übersicht über existierende Hardwareimplementationen gegeben. Schließlich wurde eine Segmentationsmethode ausgewählt, welche als Grundlage dient, jedoch aufgrund ihrer Komplexität, selbst für eine Implementation nicht in Frage kam.
In diesem Kapitel wird nun das im Rahmen dieser Arbeit entwickelte sogenannte Gap-Segmentationsverfahren vorgestellt, welches speziell für eine FPGA-Implementation entwickelt wurde. Kern dieses Verfahrens ist eine kantenerhaltende Glättung, die sogenannte Gap-Glättung. Im Folgenden wird der gesamte Segmentationsprozess vorgestellt und anschließend auf die einzelnen Prozessschritte näher eingegangen.

3.1 Der Segmentationsprozess im Ganzen

In diesem Abschnitt wird die gesamte Prozesskette des Gap-Segmentationsprozesses vorgestellt. Es sollen, wie spezifiziert, Grauwertbilder bearbeitet werden und als Ergebnis eine Labelmaske ausgegeben werden. Die Prozesskette besteht aus mehreren Verarbeitungsschritten, welche in Abbildung 3.1 schematisch dargestellt werden. In Abbildung 3.2 wird der Ablauf des Segmentationsprozesses an einem Beispiel verdeutlicht.

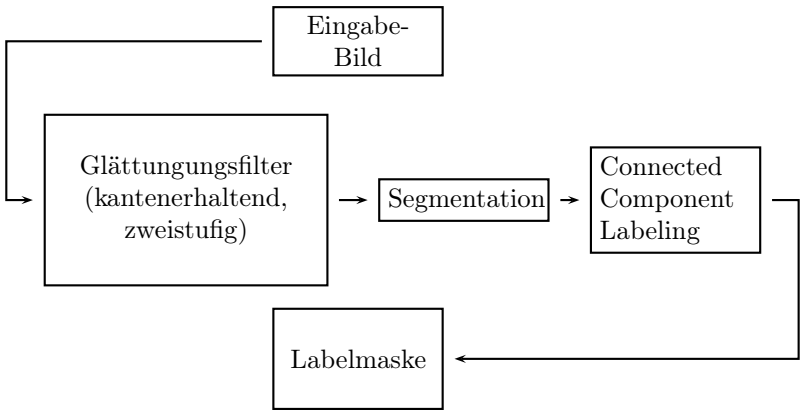
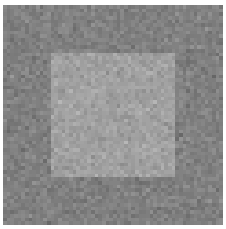
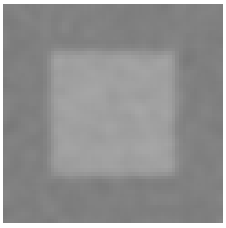


Abbildung 3.1: Schematische Darstellung des Segmentationsprozesses

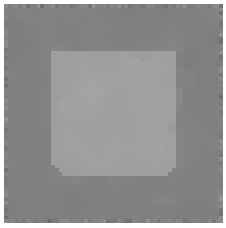
Das Gesamtverfahren besteht, wie in Abbildung 3.1 zu erkennen ist, aus drei Verarbeitungsschritten, welche im Folgenden vorstellen werden.



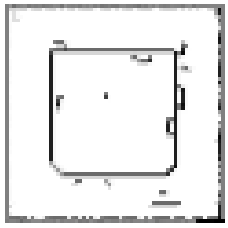
(a)



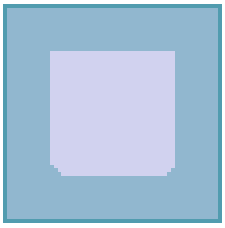
(b)



(c)



(d)



(e)

Abbildung 3.2: Stufen des Segmentationsprozess: a) Eingabebild, b) Glättungsfilter (Stufe 1), c) Glättungsfilter (Stufe 2), d) Zusammenhangsgraph, e) Labelmaske

1. **Glättungsfilter**
   In diesem Schritt sollen lokale Störungen, wie Rauschen und Textur entfernt werden, wobei Kanten möglichst erhalten oder sogar verstärkt werden sollen. Es hat sich, wie in Abschnitt 4.5.1 gezeigt wird, als günstig herausgestellt zwei Filterstufen zu verwenden. Diese müssen allerdings miteinander harmonieren und speziell aufeinander abgestimmt werden. Erst mithilfe des Testverfahrens, welches in Kapitel 4 vorgestellt wird, konnte dies erreicht werden. Es wurde zuerst eine zweimalige Vorfiltrierung mit einem gewöhnlichen 3×3-Mittelwertfilter durchgeführt. Hierdurch konnte, wie in Abbildung 3.2(b) ersichtlich, ein Großteil des Rauschens entfernt werden, allerdings auf Kosten der Kantenschärfe. Im zweiten Filterschritt wird die eigentliche kantenerhaltende Glättung durchgeführt, wodurch, wie in 3.2(c) zu erkennen ist, sogar eine Kantenschärfung eintritt. Dieser Schritt wird Gap-Glättung genannt[1].

   Die kantenerhaltende Glättung ist der zeitaufwendigste Schritt in der Prozesskette, stellt jedoch den Kern des Segmentationsprozesses dar. In Abschnitt 3.2.1 wird ein erster Eindruck vermittelt, wie wichtig dieser Prozessierungsschritt ist.

2. **Segmentationsstufe**
   Bei der Segmentationsstufe wird, aus einem Grauwertbild der Zusammenhangs-

[1] näheres siehe Abschnitt 3.2.2

graph[2] erzeugt. Das Ergebnis dieses Schrittes, welches in Abbildung 3.2(d) dargestellt ist, zeigt die Codierung dieses Graphen in einem 8-Bit Graustufenbild. In dieser Arbeit wird hierfür ein sehr einfaches regionorientiertes Verfahren benutzt, bei dem zwei direkt benachbarte Bildpunkte verbunden werden, wenn ihre Grauwertdifferenz kleiner als ein vorgegebener Schwellwert ist.

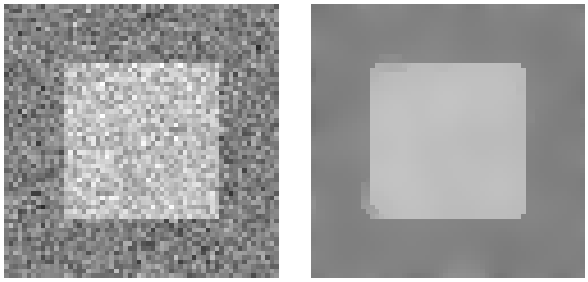
3. **Connected Component Labeling**
   Beim Connected Component Labeling wird aus den lokalen Zusammenhangsdaten der globale Zusammenhang ermittelt. Aus dem Zusammenhangsgraph, im Beispiel Abbildung 3.2(d), kann aus den Grauwerten direkt abgelesen werden, ob zwei benachbarte Punkte zusammengehören. Um aber angeben zu können, ob zwei entfernte Bildpunkte im gleichen Segment liegen, ist ein weiterer Prozessierungsschritt nötig, das Connected Component Labeling. Das Ergebnis ist eine Labelmaske, in Abbildung 3.2(e) dargestellt, bei welchem alle Bildpunkte eines Segments mit einem eineindeutigen Label versehen wurden. Auf das verwendete Verfahren wird in Kapitel 5 detailliert eingegangen.

In Abbildung 3.2(e) kann überdies beobachtet werden, dass kleine Segmentationsfehler, zum Beispiel an der unteren linken Ecke des inneren Segments, aufgetreten sind. Eine genauere Analyse dieses Fehlverhaltens, wird in Kapitel 4 durchgeführt. Im den folgenden Abschnitten werden diese einzelnen Prozessschritte genauer beschrieben.

3.2 Glättungsfilter

3.2.1 Kantenerhaltende Glättung

Die Glättung eines Bildes ist ein entscheidender Schritt. Hierbei wird das Bildrauschen und die Textur entfernt, wie in Abbildung 3.3 zu erkennen ist.



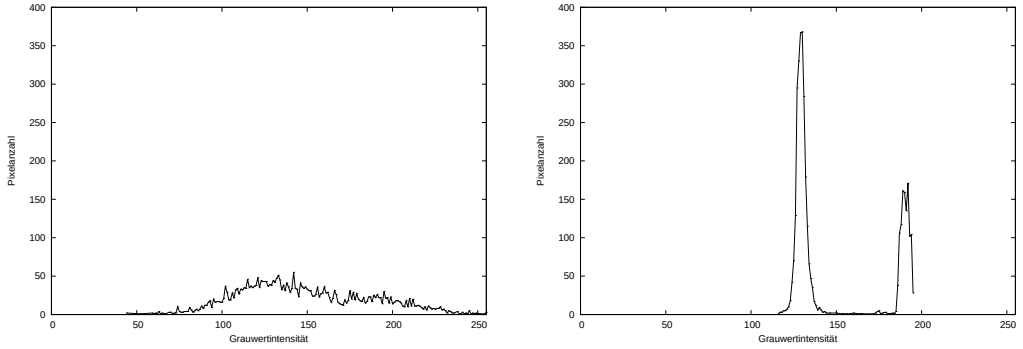
(a) Orginal (b) Geglättet

Abbildung 3.3: Synthetisches Testbild, vor und nach einer kantenerhaltenden Glättung

Obwohl sich das innere Segment vom äußeren in Bezug auf die Grauwertdifferenz deutlich unterscheidet, können anhand des (Grauwert-)Histogramms des Bildes 3.4(a), ohne

[2] Abbildung 2.3

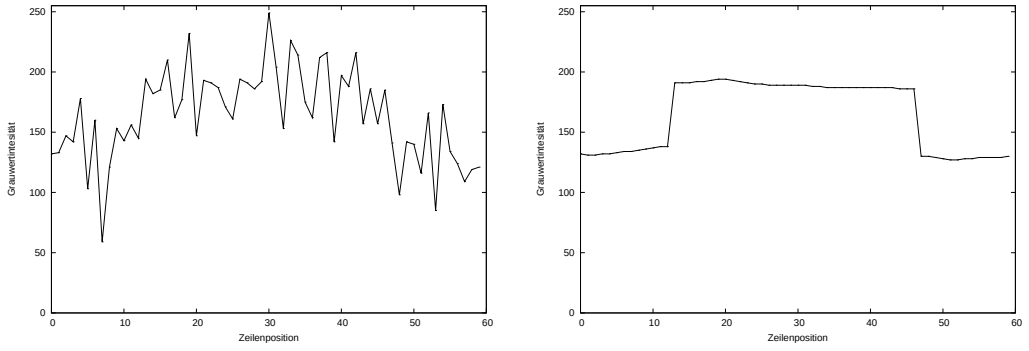
Glättung, die zwei verschiedenen Grauwertbereiche nicht unterschieden werden. Betrachtet man jedoch im Gegensatz dazu das Histogramm der geglätteten Szene 3.4(b) sind deutlich zwei Bereiche erkennbar.



(a) Orginal (b) Geglättet

Abbildung 3.4: Histogramm der Grauwerte des originalen und geglätteten Testbildes aus Abbildung 3.3

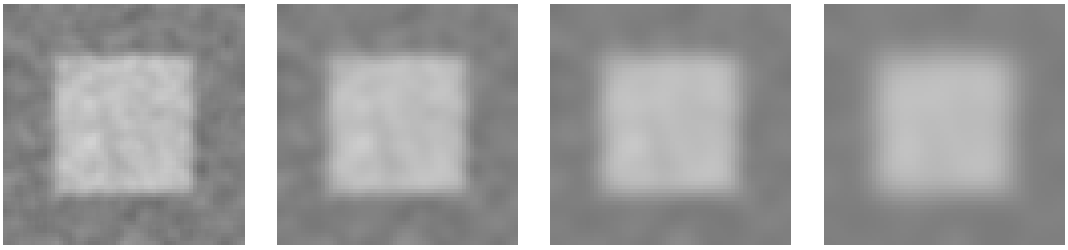
Bei der Betrachtung des Profiles der Grauwerte einer Bildzeile, dargestellt in Abbildung 3.5, ergibt sich ein ähnliches Bild. Im Profil des ungefilterten Testbildes (Abbildung 3.5(a)), sind im Gegensatz zum geglätteten Bild (Abbildung 3.5(b)), weder Kanten noch homogene Bereiche zu erkennen.



(a) Orginal (b) Geglättet

Abbildung 3.5: Zeilenprofil der Grauwerte des originalen und geglätteten Testbildes aus Abbildung 3.3, Zeile 30

Zur Glättung kam die selbst entwickelte kantenerhaltende Gap-Glättung zum Einsatz. Mit einem einfachen Mittelwertfilter sind, wie in Abbildung 3.6 zu sehen ist, solche Resultate nicht erzielbar.



(a) 2x (b) 4x (c) 8x (d) 16x

Abbildung 3.6: Testszene, Glättung mit einem 3×3-Mittelwertfilter, 2-16 Filterläufe

3.2.2 Gap-Glättung

In diesem Abschnitt wird die Verfahrensweise, auf welcher die Gap-Glättung beruht, beschrieben. Die Gap-Glättung ist, wie bereits erwähnt, ein lokaler Filter, der auf einem 3×3-Bildfenster arbeitet. Sie stützt sich in erster Linie auf Ganzzahlberechnungen. Komplexere arithmetische Operationen, bis auf eine einzige Division, werden nicht benötigt. Die Gap-Glättung ist daher gut für eine FPGA-Implementation geeignet. Im Gegensatz zu Standardfiltern, wie Mittelwertfiltern und Gauß-Filtern, deren Verfahren unabhängig vom Bildinhalt ist, wird bei der Gap-Glättung zuerst eine Analyse der Quell-Daten durchgeführt. Anschließend wird der eigentliche Filterprozess ausgeführt.
Es erfolgt zuerst eine Abschätzung, welche Bildpunkte einer 3×3-Umgebung im gleichen Segment liegen könnten, also ähnliche Grauwerte haben. In Abbildung 3.7 sind als Beispiel zwei Bereiche zu sehen, die einen hohen Grauwertunterschied aufweisen. Im Analyseschritt der Gap-Glättung sollen diese zwei Bereiche erkannt werden, und es soll eine Trennung dieser erfolgen. Beim eigentlichen Glättungsvorgang wird dann nur innerhalb der einzelnen Bereich geglättet. Signifikante Kanten, wie im gerade genannten Beispiel, bleiben hierbei vollständig erhalten.
Bei dem Analyseschritt der Gap-Segmentation wird das Grauwerthistogramm jedes 3×3-Bildfensters untersucht. Es wird angenommen, dass entweder eine unimodale oder eine bimodale Grauwertverteilung, wie in Abbildung 3.8 zur Veranschaulichung dargestellt, vorliegt.
Es soll nun festgestellt werden, welche Verteilung vorliegt. Dies ist ein aufwendiger Schritt, da über die Grauwertverteilung keine Annahmen gemacht werden können, außer dass bei einer bimodalen Verteilung signifikante Grauwertunterschiede vorhanden sein sollten. Einfache Schwellenwertverfahren, können daher nicht ohne weiteres verwendet werden.
Einen Ausweg stellen Clustermethoden dar. Clustermethoden versuchen Strukturen in Datenbeständen zu entdecken, um die Daten zu partitionieren, ohne auf Vorwissen angewiesen zu sein[3].
Die Gap-Glättung beruht auf einem einfachen, speziell zugeschnittenen Clusterverfahren,

[3] uninformierte Verfahren

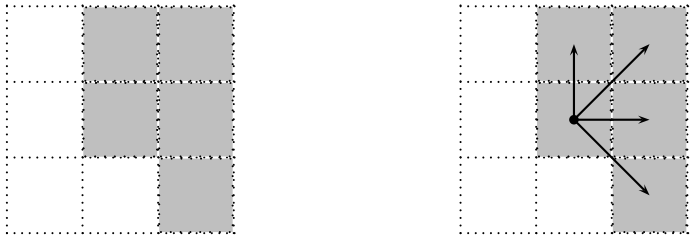


Abbildung 3.7: links 3×3-Umgebung eines Bildpunktes, rechts zum Zentralpunkt gehörige Nachbarn

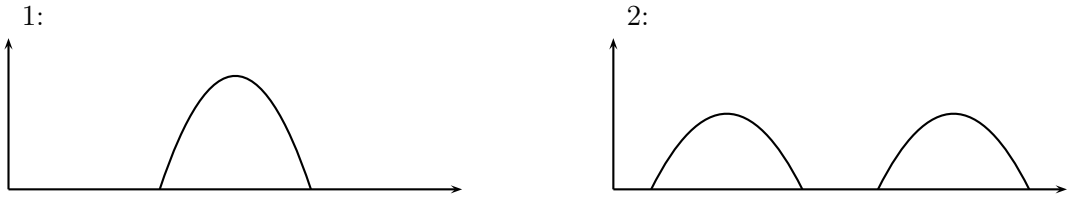


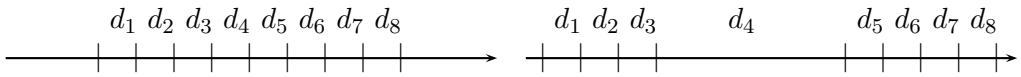
Abbildung 3.8: links unimodale Verteilung, rechts bimodale Verteilung

der sogenannten Gap-Detektion. Die Bildpunkte des 3×3-Filterfensters werden aufgrund ihres Grauwertes in zwei Cluster eingeteilt, einem Cluster C_{dunkel} der dunklen Pixel und einem Cluster C_{hell} der hellen Pixel. Unterscheiden sich diese beiden Cluster signifikant voneinander, wird angenommen, dass eine bimodale Verteilung vorliegt, falls nicht, dass eine unimodale Verteilung vorliegt. Liegt eine bimodale Verteilung vor, wird anschließend nur über die Bildpunkte des Clusters, welcher den Zentralpunkt enthält, gemittelt.
Liegt eine unimodale Verteilung vor, wird über alle Bildpunkte des 3×3-Fensters gemittelt. In diesem Fall entspricht die Gap-Glättung einer gewöhnlichen 3×3-Mittelwertfiltrierung. Durch entsprechende Parametereinstellungen kann die Gap-Glättung deswegen auch als reiner Mittelwertfilter verwendet werden. Sie ist daher für Filterstufe-1 und -2 verwendbar. Im Folgenden wird das Prinzip der Gap-Detektion erklärt.

Gap-Detektion

Die Gap-Detektion arbeitet nach folgendem Prinzip: Die neun Grauwerte werden zuerst der Größe nach sortiert. Es ergibt sich dann für unimodale und bimodale Verteilungen eine Charakteristik, wie sie in Abbildung 3.9 dargestellt ist. Danach werden die Abstände benachbarter Grauwerte berechnet und der größte Abstand bestimmt. Unterscheidet sich der größte Abstand nicht wesentlich von den anderen, wird von einer unimodalen, anderenfalls von einer bimodalen Verteilung ausgegangen. In Abbildung 3.9(b) tritt bei-

spielsweise d_4 deutlich hervor.



(a) unimodale Verteilung (b) bimodale Verteilung

Abbildung 3.9: Darstellung der nach Größe geordneten Grauwerte (vertikal Balken [||]) eines 3×3-Bildfensters, wobei die d_i $(i = 1, ..., 8)$ die Differenzen benachbarter Grauwertstufen bezeichnen.

Gap-Glättungsalgorithmus im Detail

Das Gap-Glättungsverfahren ist ein lokaler Filter, welcher auf einem 3×3-Bildfenster arbeitet, wie er in Abbildung 3.10 dargestellt ist. Die P_i entsprechen hier den Grauwerten der Bildpunkte, konkret ganze Zahlen zwischen 0 und 255[4].

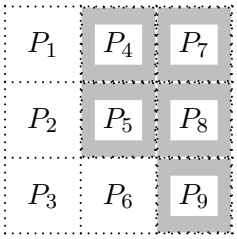


Abbildung 3.10: 3×3-Bildfenster

Im Algorithmus der Gap-Glättung treten drei Parameter auf: Sensibilität, Kantenschärfe und minimaler Glättungsschwellwert (g_1, g_2, g_3).
Die Sensibilität beschreibt, wie ausgeprägt die bimodale Charakteristik der Grauwerte des 3×3-Bildfenster sein muss, damit von einer bimodalen Verteilung ausgegangen wird. Je höher der Wert, desto stärker muss der bimodale Charakter sein. Die Kantenschärfe beschreibt, wie stark über Kanten hinweg geglättet werden soll. Dies ist ein wichtiger Wert, der Einfluss auf die Kantenschärfe hat. Es hat sich herausgestellt, dass 1,5 ein geeigneter Wert ist. Der minimale Glättungsschwellwert stellt sicher, dass selbst bei sehr kleinen Grauwertunterschieden eine Glättung durchgeführt wird. Beispielsweise haben die Werte (1 1 1 1 2 2 2 2 2) eine sehr ausgeprägte bimodale Charakteristik[5]. Die Werte können in zwei Klassen eingeteilt werden. Trotzdem soll bei einem derart kleinen Grauwertunterschied normalerweise eine Glättung erfolgen.
Es ist nicht ohne weiteres möglich,geeignete Werte für die Parameter (g_1, g_2, g_3) zu ermitteln. Erst mithilfe der in Kapitel 4 gezeigten Testmethoden, konnte dies vernünftig

[4] bei einer 8 Bit Darstellung
[5] $d_{avg} = 0$, siehe Gap-Glättungsalgorithmus unten

durchgeführt werden[6].
Bei der Gap-Glättung $Gap(g_1, g_2, g_3)$ werden folgende Einzelschritte durchgeführt:

Algorithmus Gap($\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}$) :
($g_1, g_3 \in (\mathbb{R} \geq 0)$, $g_2 \in (\mathbb{R} > 0)$)

1. Sortierung

$$(P_1, ...P_9) \mapsto (P_{s(1)}, ..., P_{s(9)}) \quad \text{so dass} \quad P_{s(1)} \leqq P_{s(2)} ... \leqq P_{s(9)}$$

2. Differenzenbildung zum Nachbarn

$$(P_{s(1)}, ..., P_{s(9)}) \mapsto (d_1, ..., d_8), \quad d_i = P_{s(i+1)} - P_{s(i)}$$

3. Bestimmung des größten Abstandes

$$d_{\mathrm{m}} = \max(d_1, ..., d_8) \quad \text{wobei} \quad \mathrm{m} \in \{1, ..., 8\}$$

4. Bestimmung des Durchschnitts der Abstände ohne den maximalen Abstand

$$d_{avg} = \frac{\sum_{i \neq \mathrm{m}} d_i}{7} \quad (d_{\mathrm{avg}} \in \mathbb{R})$$

5. Gap Detektion

$$\text{Verteilung} = \begin{cases} \text{bimodal} & \text{für} \quad d_{\mathrm{m}} \geq g_1 * d_{\mathrm{avg}} \\ \text{unimodal} & \text{sonst} \end{cases}$$

6. Schwellwertberechung

$$\text{Schwellwert} = \begin{cases} \max(\frac{d_{\mathrm{m}}}{g_2}, g_3) & \text{falls} \quad \text{Verteilung} = \text{bimodal} \\ +\inf & \text{sonst} \end{cases}$$

   wobei auf ganze Zahlen gerundet wird[7]

7. Schwellwertglättung

$$P_{\mathrm{result}} = \frac{\sum_{|P_i - P_5| \leq \text{Schwellwert}} P_i}{\sum_{|P_i - P_5| \leq \text{Schwellwert}} 1}$$

   wobei auf ganz Zahlen gerundet wird[7]

[6] Abschnitt 4.5
[7] bis einschließlich x,5 wird abgerundet

Es hat sich herausgestellt, dass es besser ist, zur Glättung nicht direkt die einzelnen Cluster C_{dunkel} und C_{hell} zu verwenden, sondern wie im Schritt 7 vorzugehen. Es sei bemerkt, dass falls $g_3 \geq 255$, die Gap-Glättung einem gewöhnlichen 3×3-Mittelwertfilter entspricht[8]. Zum besserem Verständnis wird im Folgenden der Glättungsprozess an einem Beispiel durchgeführt:

Anfangsvoraussetzung:

$$(g_1, g_2, g_3) = (1.5, 1.5, 4)$$
$$(P_1, ..., P_9) = (1, 4, 3, 6, 1, 3, 50, 5, 57)$$

1. Sortierung
$$(1, 4, 3, 6, 1, 3, 50, 5, 57) \mapsto (1, 1, 3, 3, 4, 5, 6, 50, 57)$$

2. Differenzenbildung zum Nachbarn
$$(1, 1, 3, 3, 4, 5, 6, 50, 57) \mapsto (0, 2, 0, 1, 1, 1, 44, 7)$$

3. Bestimmung des größten Abstands
$$d_m = d_7 = 44$$

4. Bestimmung des Durchschnitts der Abstände ohne den maximalen Abstand
$$d_{avg} = \frac{0+2+0+1+1+1+7}{7} = \frac{12}{7} = 1.71...$$

5. Gap Detektion
$$d_m = 44,\ d_{avg} = \tfrac{12}{7},\ g_1 = 1.5$$
$$44 \geq 1.5 * \tfrac{12}{7} \Rightarrow \text{bimodale Verteilung}$$

6. Schwellwertberechung
$$g_2 = 1.5,\ g_3 = 4, \text{Verteilung} = \text{bimodal}$$
$$\text{Schwellwert} = \max(\tfrac{44}{1.5}, 4) = \max(29.\overline{3}, 4) \overset{gerundet}{=} 29$$

7. Schwellwertglättung
$$P_i = (1, 4, 3, 6, 1, 3, 50, 5, 57),\ |P_i - P_5| = (0, 3, 2, 5, 0, 2, 49, 4, 56),\ \text{Schwellwert} = 29$$
$$P_i \text{ mit} |P_i - P_5| \leq 29 : (1, 4, 3, 6, 1, 3, 5), \quad \text{Anzahl} = 7$$
$$\Rightarrow P_{\text{result}} = \tfrac{1+4+3+6+1+3+5}{7} = \tfrac{23}{7} \overset{gerundet}{=} 3$$

[8] bei Grauwerten mit Werten zwischen 0 und 255

3.3 Segmenationsprozess

Nach der Glättung folgt, wie in Abschnitt 3.1 dargestellt, der Segmenationsprozess. Hierbei wird der lokale Zusammenhang ermittelt. Dieser wird in einem 8-Bit Grauwertbild codiert. Es gibt sehr viele verschiedene Möglichkeiten dies durchzuführen. In Abschnitt 2.4 wurde eine Übersicht über verschiedene Verfahren gegeben. Wie bereits in Abschnitt 3.1 erwähnt, wird ein einfaches regionorientiertes Verfahren verwendet, um Komplexität zu vermeiden. Es ist ein sehr lokales Verfahren, daher sehr störanfällig. Es sollte deswegen nur in Verbindung mit einer vorangehenden Glättung benutzt werden.
Es werden direkt benachbarte Bildpunkte verbunden, deren Grauwertdifferenz sich höchstens um einen vorgegebenen Schwellenwert unterscheidet. Zur Berechnung des lokalen Zusammenhangs eines Bildpunkts wird seine 3×3-Umgebung benötigt. Der Zusammenhangscode wird folgendermaßen berechnet, wobei $\mathrm{t} \in \mathbb{N}$ der Schwellwert ist und μ die Bildfunktion darstellt. Die Bezeichnungen der Bildpunkte P_i sind wie in Abbildung 3.10 gewählt.

$$\text{Zusammenhangscode} = \sum_{i=1}^{4} \delta_i(t) * 2^{(i-1)} + \sum_{i=6}^{9} \delta_i(t) * 2^{(i-2)} \in [0, 255]_{\mathbb{N}_0}$$

wobei

$$\delta_i(t) = \begin{cases} 1 & |\mu(P_i) - \mu(P_5)| \leq \mathrm{t} \\ 0 & \text{sonst} \end{cases}$$

Beispielsweise beträgt der Zusammenhangscode, falls P_4, P_5, P_7, P_8, P_9 verbunden sein sollen[9] (wie in Abbildung 3.7):

$$2^3 + 2^5 + 2^6 + 2^7 = 8 + 32 + 64 + 128 = 232$$

3.4 FPGA-Implementation

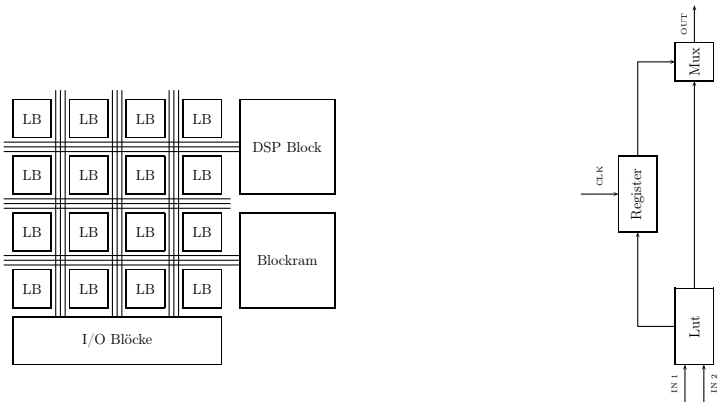
In diesem Abschnitt wird auf die FPGA-Implementation der Gap-Glättung eingegangen. Auf die Darstellung des Segmentationschrittes wird verzichtet, da er standardmäßig, ähnlich wie ein 3×3-Mittelwertfilter umgesetzt werden kann. Der Vorgang des Connected Component Labeling wird aufgrund des Umfanges, und da es als eigenständiges Thema angesehen wird, in Kapitel 5 separat besprochen. Zuerst wird jedoch auf den grundlegenden Aufbau eines FPGAs und dessen Programmierung eingegangen, um einen tieferen Einblick in die Thematik zu ermöglichen.

3.4.1 Prinzipieller Aufbau eines FPGAs

Ein FPGA ist ein programmierbarer Logikbaustein mit dem sich digitale Schaltungen implementieren lassen. Der zentrale Bestandteil eines FPGAs sind die sogenannten Logikblöcke, mit denen elementare Logikfunktionen und Speicheroperationen realisiert werden können. Durch Zusammenschaltung dieser kann prinzipiell jede digitale Schaltung

[9] d.h. $\delta_i(t) = 1$ genau für $i = 4, 7, 8, 9$.

erzeugt werden. Genauer kann prinzipiell jeder Zustandsautomat nachgebildet werden und daher kann jede berechenbare Funktion implementiert werden.



(a) (b)

Abbildung 3.11: Schematische Darstellung eines a) FPGAs b) Logik Blocks

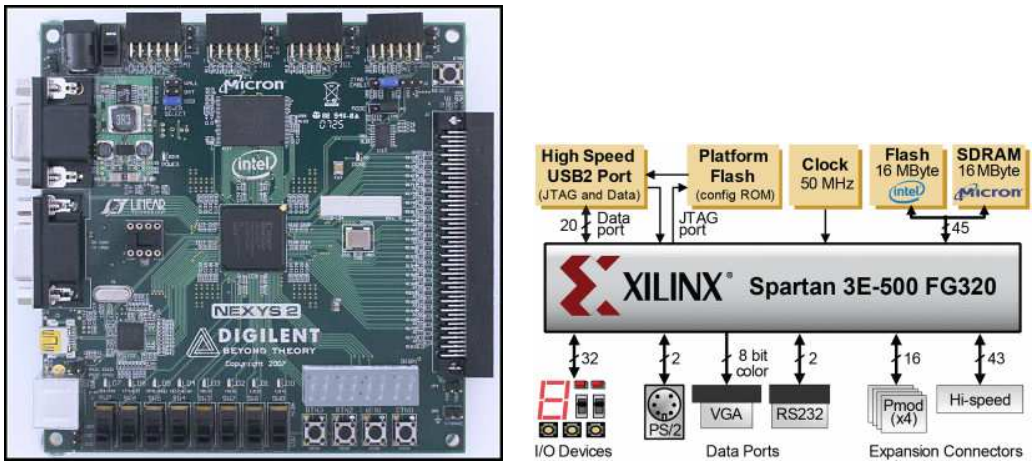
Ein Logikblock enthält wie in Abbildung 3.11(b) dargestellt, eine Look-Up Table (LUT)[10] und mindestens ein Flipflop (FF). Außer den Logikblöcken gibt es weitere Blöcke, wie Eingangs-/Ausgangs-Blöcke[11], mit denen sich Verbindungen, beispielsweise zu externen Speicherbausteinen, seriellen Schnittstellen, oder USB-Anschlüssen herstellen lassen. Viele Funktionen lassen sich aufgrund ihrer Schaltungskomplexität nur sehr ineffizient durch Logikblöcke realisieren. Deswegen finden sich auf modernen FPGAs zusätzlich viele Spezialblöcke, wie Speicherbausteine, digitale Signalprozessoren oder sogar komplette CPU-Einheiten.

Einen großen Teil des FPGAs macht das programmierbare Schaltnetzwerk aus, durch welches die einzelnen Blöcke miteinander verbunden werden. Die Verschaltung hat einen großen Einfluss auf die Komplexität einer Schaltung, deren Schaltgeschwindigkeit und damit maximal erreichbarer Taktfrequenz. Es kann vorkommen, obwohl für eine Schaltung genügend Logikblöcke zur Verfügung stehen, dass diese nicht implementiert werden kann, da die Logikblöcke durch das Schaltnetzwerk nicht wie benötigt verbunden werden können. Zusätzlich zum Schaltnetzwerk gibt es noch ein spezielles Netzwerk, durch welches Taktsignale synchron an alle Blöcke verteilt werden können[12]. Dies ist über die Digital Block Management Blöcke (DCMs), mit deren Hilfe Taktsignale erzeugt werden können, mit einem externen Taktgeber verbunden.

Es sei erwähnt, dass dies nur eine sehr vereinfachte Darstellung eines modernen FPGAs ist. Genauere Informationen findet man in Lehrbüchern wie [39] oder in der Dokumentation der Hersteller. Der in dieser Arbeit verwendete Xilinx Spartan-3E XC3S1200E FPGA

[10] gewöhnlich eine 4- oder 6-Input LUT
[11] engl. IO-Blocks (IOBs)
[12] hierbei müssen unterschiedliche Leitungslängen ausgeglichen werden

besitzt beispielsweise 17344 Logikblöcke[13], 28 * 18 Bit Multiplizierer, 504 kBit Blockram und 8DCMs [40].
Das FPGA selbst ist auf einer Leiterplatine, dem Board, mit einer Vielzahl anderer Bauteile, wie einem externen Taktgeber, externen Speicherbausteinen, Schnittstellen, LED-Anzeigen, Schaltern und vielen weiteren Bauteilen verbunden. In Abbildung 3.12 ist das Digilent Nexys2 Board, welches in dieser Arbeit verwendet wurde, dargestellt.



(a) (b)

Abbildung 3.12: a) Nexys 2 Board, b) Schematische Darstellung, Abbildung entnommen aus der Produktbeschreibung des Digilent Inc. Nexys 2 Board zu finden auf *http://www.digilentinc.com*[14]

Auf den Nexys 2 Board in Abbildung 3.12(a) findet sich in der Mitte das Sparten-3E FPGA. Darüber ist ein 16 MByte großer RAM Baustein angebracht, welcher über eine 16 Bit Speicheranbindung mit dem Spartan-3E FPGA verbunden ist und mit maximal 12,5 MHz[15] betrieben werden kann. Rechts neben dem FPGA befindet sich der Taktgeber. Auf der linken Seite ist ein VGA-Ausgang, eine serielle Schnittstelle und ein USB-Anschluss angebracht. Im unteren Teil befinden sich zahlreiche Schalter und eine LED-Anzeige. Dieses Board dient als Testplattform für die im Rahmen dieser Arbeit erstellten Algorithmen. Für spätere Anwendungen sollte ein für die Bildverarbeitung besser geeignetes Board verwenden werden, das insbesondere eine höhere Speicherbandbreite zwischen FPGA und RAM zur Verfügung stellt. Leistungsfähigere FPGAs wie der Spartan-6 oder FPGAs der Virtex Reihe stellen außerdem eine weitaus höhere Anzahl an Ressourcen[16] zur Verfügung und können aufgrund leistungsfähigerer Architektur und feineren Fertigungstechnik höher getaktet werden. Beispielsweise stellt der Spartan-6

[13] jeder Logikblock beinhaltet eine 4-Lut und 1 FF
[15] im SRAM-Modus
[16] in erster Linie mehr Logikblöcke, Blockram, DSPs und IO-Ports

XC6SLX150 laut [41] 92152 Logikblöcke[17], 4872 kBit Blockram und 180 DSP Blöcke zur Verfügung, also um den Faktor sechs mehr Ressourcen als das hier verwendete Spartan-3E FPGA.

3.4.2 Konfiguration eines FPGAs

Im letzten Abschnitt wurde der Aufbau eines FPGAs beschrieben. In diesem Abschnitt wird nun vorgestellt, wie es konfiguriert werden kann. Zur eigentlichen Beschreibung wurde die höhere Hardwarebeschreibungssprache Handel-C [42] benutzt. Handel-C wurde entwickelt, um Algorithmen, die in der Hochsprache C entwickelt wurden, auf möglichst direktem Weg auf einem FPGA umsetzen zu können. Ihr Syntax entspricht der Programmiersprache C und wurde um ein paar Schlüsselwörter erweitert, welche auf FPGA spezifische Gegebenheiten eingehen und außerdem paralleles Programmieren ermöglichen. Andere bekannte Hardwarebeschreibungssprachen, welche aber auf einer niedrigeren Abstraktionsebene arbeiten, sind VHDL und VERILOG.
Handel-C ist eine streng taktbasierte Sprache[18]. In jedem Takt kann einer oder mehreren Variablen genau ein Wert zugewiesen werden. Für alle anderen Operationen werden keine zusätzlichen Takte benötigt. In 3.13 ist der Quelltext eines einfachen Programms gezeigt.

```
1  unsigned int 8 a,b,c,d,e;
2
3
4  par
5  {
6       a=5;
7       b=3;
8  }
9  par
10 {
11      c=a+b;
12      d=a*b;
13 }
14 e=c+d;
```

Abbildung 3.13: Handel-C Beispielcode

In der ersten Zeile werden fünf 8-Bit Variablen deklariert. Wird das Programm ausgeführt wird im ersten Takt der Variablen a der Wert 5 und der Variablen b der Wert 3 zugewiesen. Dies kann parallel ausgeführt werden, da diese Operationen unabhängig sind und wird durch das Schlüsselwort *par* signalisiert. Im zweiten Takt wird der Variablen c der Wert $a + b$ zugewiesen und der Variablen d der Wert $a * b$. Diese Operationen sind ebenfalls unabhängig und können parallelisiert werden. Für die Berechnung von $a*b$ und $a + b$ wird kein weiterer Takt benötigt. Zur Berechnung der Variablen e allerdings sind

[17] wobei jeder Logikblock eine 6-Input Lut und ein FF enthält
[18] Ein ein Takt entspricht einem Schaltzyklus. Die Frequenz mit welchem das FPGA betrieben wird entspricht der Anzahl der Takte pro Sekunde.

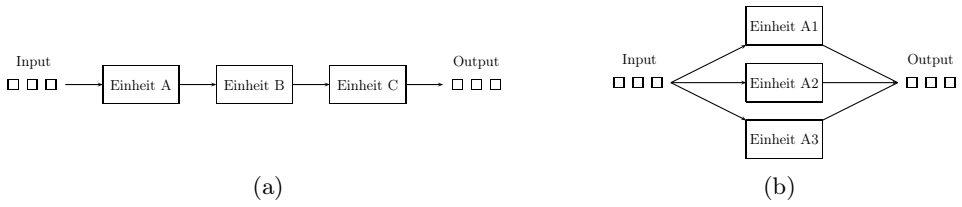
die Werte der vorherigen Berechnung notwendig, die Operation kann daher nicht parallelisiert werden, und muss im dritten Schritt ausgeführt werden. Hier haben die Variablen dann die Werte Werte $c = 8$, $d = 15$ und $e = 23$.
Beim übersetzen eines Programmes mit einem Handel-C Compiler, wird eine Netzliste[19] erzeugt. In diesem als Logiksynthese bekannten Schritt, wird vereinfacht ausgedrückt beschrieben, wie der Programmablauf durch Logikblöcke realisiert werden kann. Es findet sich hier die Beschreibung darüber, wie Logikblöcke[20] konfiguriert und verschaltet werden müssen. Es werden hier normalerweise noch keine genauen Angaben gemacht, welche Logikblöcke auf dem FPGA genau verwendet werden sollen und insbesondere nicht, wie die Verschaltung dieser explizit zu realisieren ist. Zur physikalische Realisierung wird eine Place-and-Root Electronic Design Automation (P&R EDA) Software herangezogen.
Beim P&R entscheidet sich, ob eine Schaltung auf einem ausgewählten FPGA überhaupt umgesetzt werden kann und wie hoch das FPGA maximal getaktet werden kann. Die Komplexität der im Quelltext des Programmes verwendeten Ausdrücke spielt hierbei eine entscheidende Rolle. In Handel-C benötigt die Berechnung eines Ausdrucks immer einen Takt, jedoch entscheidet die Komplexität des Ausdruckes, wie lange ein Takt dauern muss, in anderen Worten, wie hoch die maximal erreichbare Taktfrequenz des FPGAs ist. Die insgesamt zur Ausführung eines Programmes benötigte Zeit, ergibt sich aus dem Quotienten der benötigten Takte durch die Frequenz ($\frac{\text{benötigte Takte}}{\text{Frequenz}}$).
Daher ist es oft sinnvoll, komplexere Ausdrücke in mehrere weniger komplexe zu zerlegen. Auch wenn das Programm dann mehr Takte benötigt, kann aufgrund der höheren Taktgeschwindigkeit eventuell eine kürzere Laufzeit erreicht werden.
Das Ergebnis des P&R ist eine Konfigurationsdatei, mit welcher das FPGA konfiguriert werden kann. In dieser Arbeit werden SRAM basierte FPGAs verwendet, die beliebig oft (re-)konfigurieren werden können, jedoch bei Spannungsverlust ihre Funktionalität verlieren. Auf den Boards von SRAM FPGAs finden sich aber normalerweise FLASH Speicherzellen, in welchen die Konfiguration dauerhaft gespeichert werden kann. Das FPGA wird bei einer Aktivierung dann automatisch konfiguriert. Die Konfiguration eines SRAM basierte FPGAs benötigt gewöhnlicherweise einige Sekunden. Es gibt aber auch FPGAs die ihre Konfiguration dauerhaft behalten, jedoch nicht mehr rekonfiguriert werden können.

3.4.3 Programmiertechniken

In den letzten zwei Abschnitten wurde erklärt, wie ein FPGA aufgebaut ist und wie es konfiguriert werden kann. In diesem Abschnitt werden Techniken vorgestellt, mit denen ein für ein FPGA geeignetes Programm erstellt werden kann. Nach [12, 43] gibt es drei verschiedene Prozessierungsmodelle, um Bilddaten zu verarbeiten: Datenstromverarbeitung[21], Offline-Prozessierung und Hybrid-Prozessierung. Bei der Datenstromverarbeitung wird ein Bilddatenstrom, gewöhnlich im Rasterformat, verarbeitet. Es kann nur auf die Daten, welche sich gerade im Datenstrom befinden, zugegriffen werden. Alle anderen Da-

[19] in der Regel im Electronic Design Interchange Format (EDIF) dargestellt
[20] ein Teil hiervon ist die Konfiguration der LUTs und der Anfangszustand der Register (FF)
[21] stream processing

ten, welche zu einer Berechnung benötigt werden, müssen zwischengespeichert werden. Dieser Modus eignet sich insbesondere für pixelbasierte Methoden. Durch Zwischenspeicherung vorangegangener Zeilen können auch fensterbasierte Filter prozessiert werden. Globale und dynamische Verarbeitungsschritte, bei welchen ein Zugriff auf einen großen eventuell erst zur Laufzeit bekannten Datenbereich ermöglicht werden muss, eignen sich für eine Datenstromverarbeitung nicht. Durch eine Datenstromverarbeitung kann jedoch eine sehr hohe Verarbeitungsgeschwindigkeit (Pixeldurchsatz[22]) erreicht werden. Bei der Offline-Prozessierung wird meistens das gesamte Eingangsbild und Ausgangsbild in einem Bildspeicher zwischengespeichert, um beliebig darauf zugreifen zu können. Diese Verfahrensweise wird normalerweise angewendet um globale und dynamische Methoden zu realisieren. Der Nachteil ist, dass ein Bildzwischenspeicher (Framebuffer) benötigt wird, und oftmals die Verarbeitungsgeschwindigkeit nicht erreicht wird, wie es mit einer Datenstromverarbeitung möglich wäre. Bei der Hybrid-Prozessierung werden beide Verfahrensweisen verwendet. Bei der Prozessierung eines Bilddatenstroms werden hier oft komplette Bilder zwischengespeichert. Sie wird meistens eingesetzt, wenn eine Datenstromverarbeitung nicht komplett durchgeführt werden kann, oder nur an einigen kritischen Stellen erforderlich ist. Bei der Gap-Glättung erfolgt eine hybride Prozessierung. Der Gap-Glättungsfilter ist zwar vollkommen auf Datenstromverarbeitung ausgelegt. Da jedoch für eine Glättung mehrere Filterdurchläufe nötig sind, muss das Ergebnis nach jedem Filterlauf zwischengespeichert werden. Die Datenstromverarbeitung wird meistens mittels Fließbandverarbeitung[23] realisiert. Hierbei wird eine Aufgabe, wie in Abbildung 3.14(a) dargestellt, in mehrere Teilaufgaben aufgeteilt, die wie auf einem Fließband nacheinander abgearbeitet werden. Die Verarbeitungsdauer einer Dateneinheit, die sogenannte Latenz, wird dadurch im allgemeinen nicht verkürzt, sondern eher erhöht. Jedoch ermöglicht die Fließbandverarbeitung, den simultanen Betrieb aller Verarbeitungseinheiten, wodurch insgesamt eine höhere Verarbeitungsleistung erreicht wird.



(a) (b)

Abbildung 3.14: Parallele Datenverarbeitung: a) Datenstromverarbeitung b) SIMD

Dies ist eine Form von Multiple Instruction Multiple Data (MIMD). Eine weitere Form

[22] gewöhnlich Angegeben im Pixel pro Sekunde
[23] engl. pipeline processing

der parallelen Datenverarbeitung ist Single Instruction Multiple Data (SIMD) (Abbildung 3.14(b)). Ein Verarbeitungsschritt wird gleichzeitig mehrfach ausgeführt. Um dies Durchzuführen, werden mehrere gleichartige Verarbeitungseinheiten benötigt. Beide Methoden können auch kombiniert werden. Zur Übersicht ist in Abbildung 3.15 die Flynn Kategorisierung der parallelen Datenverarbeitung abgebildet.

	Single data	Multiple data
Single Instruction	SISD	SIMD
Multiple Instruction	MISD	MIMD

Abbildung 3.15: Flynns Kategorisierung der parallelen Datenverarbeitung [44]

3.4.4 Implementation des Gap-Glättungsalgorithmus

Im Folgenden wird beschrieben, wie das Gap-Glättungsverfahren auf einem FPGA umgesetzt wurde. Beim Gap-Glättungsfilter konnte eine Datenstromverarbeitung realisiert werden. Hierzu werden viele Verarbeitungsstufen benötigt. Eine Übersicht der Implementation ist in Abbildung 3.16 dargestellt.

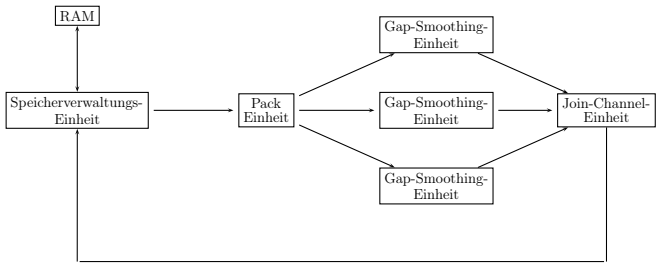


Abbildung 3.16: Gap-Smoothing Implementation, Aufbau

Der Kern sind die Gap-Glättungseinheiten. Diese führen eine Datenstromverarbeitung durch. Sie verarbeiten jeweils einen 3×3-Bildblock. Mit jeden Takt nehmen sie einen Pixelwert entgegen und liefern alle neun Takte einen Pixelwert aus. In der Abbildung sind drei Gap-Glättungseinheiten dargestellt, welche parallel betrieben werden (SIMD). Alle drei Takte kann daher ein Pixelwert ausgeben werden. Werden neun Gap-Glättungseinheiten parallel betrieben, kann in jedem Takt ein Pixelwert ausgegeben werden. Die Speicherverwaltungseinheit und Packeinheit kümmern sich darum, den Gap-Glättungseinheiten die Eingangsdaten zur Verfügung zu stellen und die prozessierten Daten abzutransportieren (Abbildung 3.17).

Jede Gap-Glättungseinheit besteht aus zahlreichen Untereinheiten, und diese wiederum aus zahlreichen weiteren Untereinheiten (Abbildung 3.18).

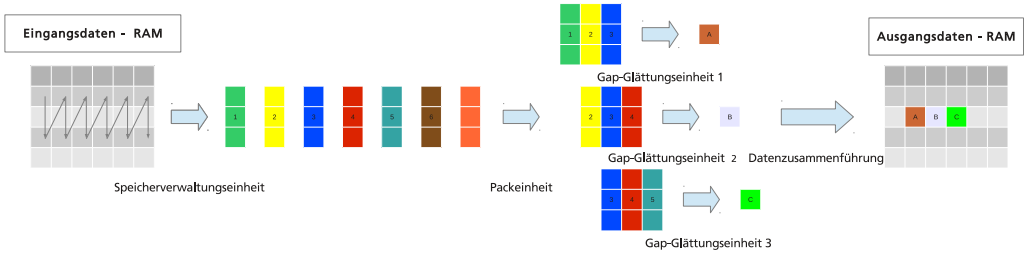


Abbildung 3.17: Gap-Glättung, Datenverwaltung

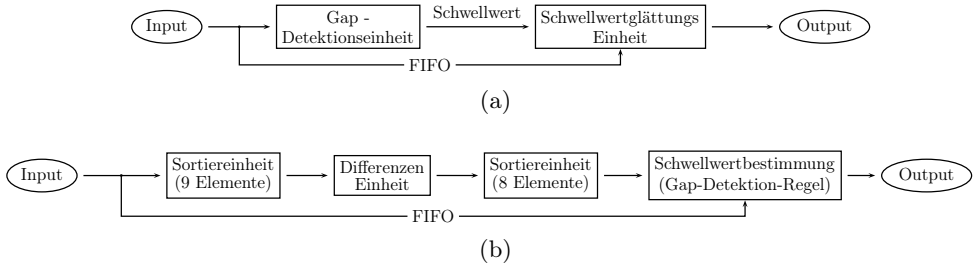


Abbildung 3.18: a) Gap-Glättungseinheit, b) Gap-Detektionseinheit

Die Module sind untereinander durch sogenannte *channels*[24] verbunden. Jedes Modul verfügt über eine Datenflusssteuerung. Der Datenfluss kann angehalten und fortgesetzt werden. Versiegt der Eingangsdatenstrom oder können die prozessierten Daten nicht abgeführt werden, wird der Betrieb erst dann wieder aufgenommen, wenn die entsprechende Ressource wieder zur Verfügung steht. Bei einer ungesteuerten Fließbandverarbeitung ist dies nicht möglich. Die Pipeline muss bei Prozessierungsbeginn erst vollständig geladen werden, bevor gültige Ergebnisse produziert werden, und der Datenfluss kann nicht einfach unterbrochen werden. Der Vorteil einer flussgesteuerten Pipeline ist eine höhere Flexibilität, der Nachteil, ein erhöhter Hardwareverbrauch für benötigte Zwischenspeicherstufen. Jedes zur Prozessierung verwendete Register muss gepuffert werden.

3.4.5 Performanz/Hardwareverbrauch der Gap-Glättung

Durch die Datenstromverarbeitung und den parallelen Betrieb mehrerer Glättungseinheiten kann eine sehr hohe Verarbeitungsleistung erreicht werden. Der Prototyp, der auf dem Nexys2-Board implementiert wurde und drei Gap-Glättungseinheiten verwen-

[24] Handel-C Ausdruck: entspricht einem FIFO über den eine synchrone Kommunikation zwischen Modulen (auch unterschiedlicher Taktbereiche) ermöglicht wird

det, benötigt acht Takte pro Pixel. Wie sich später herausstellte werden ungefähr 70 Filterdurchläufe benötigt, um sehr starkes Rauschen behandeln zu können, und 35 Filterdurchläufe, für mittelhohe Rauschstufen. In Abbildung 3.19 ist eine Übersicht über die Dauer der Glättung bei verschiedenen Implementationen gegeben.

Bildgröße	1000 * 1000		10000 * 10000	
	70 Läufe	35 Läufe	70 Läufe	35 Läufe
	Hardware			
Implementation auf Nexys-2 Board (Spartan3E @ 12.5 MHz)	44,8 sec	22,4 sec	75 min	37 min
Implementation* 3 Takte/Pixel @ 12,5 MHz	16,8 sec	8,4 sec	28 min	14 min
Implementation* 3 Takte/Pixel @ 50,0 MHz	4,2 sec	2,1 sec	7,0 min	3,5 min
Implementation* 3 Takte/Pixel @ 100,0 MHz	2,1 sec	1,1 sec	3,5 min	1,75 min
Implementation* 1 Takte/Pixel @ 12,5 MHz	5,6 sec	2,8 sec	9,4 min	4,7 min
Implementation* 1 Takte/Pixel @ 50,0 MHz	1,4 sec	0,7 sec	2,4 min	1,2 min
Implementation* 1 Takte/Pixel @ 100,0 MHz	0,7 sec	0,35 sec	1,2 min	36 sec
	Software			
Core2Duo @ 2.8 GHz** - 1 Kern	39,88 sec	22,4 sec	67 min	34 min
Core2Duo @ 2.8 GHz** - 2 Kerne	21,13 sec	10,57 sec	36 min	18 min

Abbildung 3.19: Hochrechnung Geschwindigkeit Gap-Glättung, * Simulation mit dem MentorDK-Simulator (aufgrund des Speicherinterfaces auf dem Nexy2-Board nicht umsetzbar), **genau Bezeichnung: Intel Core2 Duo CPU T9600 @ 2.8 GHz

Der Prototyp, welcher auf dem Nexys-2 Board umgesetzt wurde, benötigt für 70 Läufe bei einem 1000*1000 Bild ungefähr 45 Sekunden. Selbst mit 12,5 MHz wird dabei ungefähr die Leistung einer ausgewachsenen CPU[25] erreicht . Dies liegt in erster Linie an der Sortierung und Maximumsbestimmung welche bei der Gap-Glättung durchgeführt werden müssen (Schritt 1 und 3). Ohne diese benötigt die CPU nur 3,8 Sekunden (1 Kern) bzw. 2 Sekunden (2 Kerne). Diese Sortierung kann mittels Hardware sehr effizient durchgeführt werden. Ein Datenstrom kann blockweise sortiert werden. Mit jedem Takt wird ein Wert aufgenommen und ein Wert ausgegeben.
Die FPGA-Implementation auf dem Nexys-2 Board erfüllt nicht die Anforderung, dass der Gesamtprozess bei kleinen Bildgrößen[26] in einigen Sekunden und bei größeren Bildern[27] in wenigen Minuten abgeschlossen sein muss. Auf einer leistungsfähigeren Plattform kann der Glättungsprozess weitaus schneller ausgeführt werden. Wie in der Abbildung 3.20 gezeigt, können auf einem Spartan-6 FPGA Taktraten bis 100 MHz erreicht werden. Auf einem Spartan-6 FPGA stehen außerdem genug Ressourcen zur Verfügung, um neun Gap-Glättungseinheiten zu implementieren. Damit könnte ein Pixel pro Takt verarbeitet werden.
Bei einer Bildgröße von 1000*1000 Bildpunkten reicht eine Implementation mit drei Glättungseinheiten und einer Taktfrequenz von 50 MHz aus um auch eine Gap-Glättung mit 70 Filterdurchläufen in unter fünf Sekunden durchführen zu können. Bei extrem großen Bildgrößen, beispielsweise 10000*10000 Bildpunkten ist eine Implementation mit neun

[25] Intel(R) Core(TM)2 Duo CPU T9600 @ 2.8 GHz, termal design power (TDP) 35W
[26] $< 2000 * 2000$
[27] $\sim 10000 * 10000$

Glättungsstufen sinnvoll. Bei einer Taktfrequenz von 100 MHz werden für 70 Filterläufe 1,2 Minuten benötigt. Diese Leistung würde den Laufzeitanforderungen genügen. Mit leistungsfähigen FPGAs, wie der Xilinx Virtex-6,-7 Modellreihe, ist es jedoch durchaus im Bereich des Möglichen eine Leistungssteigerung um den Faktor zehn zu realisiert[28]. Dadurch könnte selbst ein 10000*10000 Pixel großes Bild in unter zehn Sekunden verarbeitet werden, und eine Videoprozessierung mit 14 Bilder pro Sekunde bei einer Bildgröße von 1000*1000 Pixel verwirklicht werden.
In Abbildung 3.20 ist der benötigte Hardwareverbrauch dargestellt. Auf einem Xilinx Spartan-3E FPGA benötigt eine Gap-Glättungsfilter mit drei Glättungseinheiten in etwa 43 % der verfügbaren Logikblöcke. Auf einem mittelgroßen Xilinx Spartan-6 XCSLX75 Chip werden hierfür rund 12 % der verfügbaren Logikblöcke benötigt. Es wäre daher noch genug Platz für andere Funktionseinheiten vorhanden. Auf einem Spartan-6 XC6SLX150 Chip, der ungefähr doppelt so viele Logikblöcken wie der Spartan-6 XCSLX75 besitzt, benötigen neun Glättungseinheiten, ungefähr 18 % der verfügbaren Logikblöcke. Es dürften auch in diesem Fall noch genug Ressourcen für weitere Aufgaben zur Verfügung stehen.

	Spartan-3E	Spartan-6
Benötigte/Verfügbare Slices*	3786/8672 (43 %)	1492/11662 (12 %)
Benötigte/Verfügbare 4 bzw. 6-input LUTs	4442/17344 (25 %)	3930/46648 (8 %)
Benötigte/Verfügbare FF	4877/17344 (28 %)	5268/93296 (4 %)
Benötigte/Verfügbare DSP-Blöcke	3/28 (2 %)	-/172
Benötigter/Verfügbarer Block-RAM	-/28*18 kbit	-/172*18 kbit
Maximale Frequenz	89 MHz	119 MHz

Abbildung 3.20: Hardwareverbrauch der FPGA-Implementierung der Gap-Glättung, drei Glättungseinheiten, Plattformen: Xilinx Spartan-3E xc3s1200e-5fg320, speedgrade -5, Xilinx Spartan-6 SC6SLX75-FGG484, speedgrade -3, beide ISE 14.7, Design Goal balanced; * eine Spartan-3 Slice enthält zwei 4-input Luts und zwei FlipFlops, wogegen eine Spartan-6 Slice vier 6-input LUTS und acht FlipFlops enthält

[28] auf den (mittel-)großen Virtex-6,-7 Modellen können bei einem Hardwareverbrauch von unter 75% durchaus 45 Glättungseinheiten mit einer Taktfrequenz von 200 MHz verbaut werden.

4 Qualitätsuntersuchung

4.1 Grundsätzliches

Die Bewertung eines Segmentationsprozesses hinsichtlich der Qualität des Segmentationsergebnisses ist sehr hilfreich sowohl für den späteren Nutzer, als auch für den Entwickler. Für den Entwickler kann sie wertvolle Hinweise zur Optimierung und Weiterentwicklung der Segmentationsmethode liefern. Einem Nutzer kann hierdurch ermöglicht werden, eine für seine Anwendung geeignete Segmentationsmethode auszuwählen, diese vernünftig zu implementieren, und das Verhalten dieser Methode abschätzen zu können.

4.1.1 Ziele

Die folgenden Fragestellungen waren der Anlass, im Rahmen dieser Arbeit eine qualitative Bewertung des Gap-Segmentationsverfahrens durchzuführen:

a) Parameterbestimmung
Die Durchführung einer detaillierten Qualitätsuntersuchung entstand aus der Notwendigkeit geeignete Parameter zu finden. Bei dem im Abschnitt 3.1 vorgestellten Gap-Segmentationsprozess treten insgesamt sechs Parameter auf:

1. Filterstufe 1: $\mathbf{X_1} *$ Mittelwertfilter 3×3
2. Filterstufe 2: $\mathbf{X_2} *$ Gap-Glättung$(\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3})$
3. Segmentationsstufe: Segmentationsschwellenwert $\mathbf{t}$

Die Parameter X_1, X_2 bezeichnen die Anzahl der entsprechenden Filterläufe. Es stellt sich hier die Frage, ob es einen optimalen Wert gibt. Da die Anzahl der Filterläufe natürlich direkt Einfluss auf die Laufzeit des Segmentationsprozesses hat, sollte die minimale Anzahl der Läufe bestimmt werden, mit der eine vernünftige Qualität erreicht werden kann. Die Parameter g_1, g_2 und g_3 der Gap-Glättung sind in Kapitel 3.2.2 und der Segmentationsschwellenwert t in Kapitel 3.3 beschrieben worden.

b) Bestimmung der qualitativen Leistungsfähigkeit
Ein vor allem für den späteren Anwender wichtiger Punkt ist, die qualitative Leistungsfähigkeit der hier vorgestellten Segmentationsverfahren einschätzen zu können. Welche qualitativen Ergebnisse sind bei unterschiedlich guten und schlechten Bedingungen zu erwarten? Wie störanfällig ist das Segmentationsverfahren und wie wirken sich unterschiedliche Arten von Störungen aus? Wie gut muss die Bildqualität sein, um gute Ergebnisse zu erhalten, und ab wann beginnt das Verfahren zu versagen. Des weiteren ist

es sehr hilfreich, untersuchen zu können, wie sich Segmentationsmethoden im Allgemeinen verhalten und von welchen Größen sie besonders abhängig sind. Um diese Fragen beantworten zu können, muss zunächst geklärt werden, wie man die qualitative Leistungsfähigkeit einer Segmentation überhaupt vernünftig bestimmen kann.

c) Vergleich mit alternativen Verfahren
Zuletzt sollte die Gap-Segmentation mit alternativen Verfahren verglichen werden, um die generelle Leistungsfähigkeit diese Verfahrens einzuschätzen. Gibt es bessere und schlechtere Verfahren? Inwieweit unterscheiden sich verschiedene Verfahren? Lohnt sich der Aufwand komplexerer Algorithmen? Was sind Vor- und Nachteile unterschiedlicher Verfahren (Stichwort: Komplexität vs. Geschwindigkeit vs. Qualität)? Insbesondere sollte ein Vergleich mit der Jahn-Methode erfolgen, von welcher die Gap-Segmentation abgeleitet wurde (Abschnitt 2.6).

4.1.2 Problematik und genereller Ablauf

In diesem Abschnitt soll geklärt werden, welche Schritte notwendig sind, um eine qualitative Segmentationsbewertung durchführen zu können und welche Fragen hierbei gelöst werden müssen. Um eine vernünftige qualitative Segmentationsbewertung durchführen zu können, muss vorab geklärt werden, welche Fragen mit einer Segmentationsbewertung beantwortet werden sollen. Die Fragestellung leitet sich aus der Zielsetzung ab, welche im vorhergehenden Abschnitt aufgeführt wurde. Anschließend muss geklärt werden, welche Anforderungen an den Bewertungsprozess selbst gestellt werden müssen, um diese Fragestellung vernünftig beantworten zu können (Abschnitt 4.1.3).
Ist geklärt, was mit einer Bewertung bezweckt und welchen Anforderung diese genügen soll, muss eine geeignete Bewertungsmethode gefunden werden. Die schwierigste Aufgabe, die bewältigt werden muss, um eine vernünftige qualitative Segmentationsbewertung durchführen zu können, ist es ein Maß für die Güte einer Segmentation zur Verfügung zu stellen. Dafür gibt es jedoch kein Standardverfahren und viele Wissenschaftler gehen sogar davon aus, dass dieses Unterfangen aussichtslos ist [15, 45]. Um überhaupt ein Maß für die Güte einer Segmentation konstruieren zu können, müssen zwei grundlegende Fragen geklärt werden.

- Was ist die Güte eines Segmentationsergebnisses und,
- wie kann dieses ermittelt werden?

Im Abschnitt 4.2 wird auf diese Fragen näher eingegangen. Es wird dargestellt, wie der Begriff der Güte einer Segmentation mathematisch exakt definieren werden kann, und es werden erste Folgerungen davon abgeleitet. In dieser Arbeit wird das Kriterium des mittleren Grauwertes verwendet. Segmente sollen bezüglich ihres Grauwertes möglichst einheitlich sein, und sich von den mittleren Grauwerten benachbarter Segmente möglichst signifikant unterscheiden. Dies ist keine exakte Beschreibung. Eine Modellierung des Begriffs des mittleren Grauwerts stellt das Mumford- und Shah-Modell [23] dar. Im Abschnitt 4.2.5 wird auf dieses Modell näher eingegangen. Aufgrund der Komplexität

und des daraus resultierenden Aufwands dieses Verfahren umzusetzen und der Schwierigkeit die Ergebnisse dieses Modelles zu interpretieren, wurde dieses Modell jedoch nicht direkt zur Segmentationsbewertung herangezogen.
In Abschnitt 4.1.4 werden verschiedene Methoden zur Durchführung einer Segmentationsbewertung vorgestellt und dargelegt, welcher Weg in dieser Arbeit eingeschlagen wurde, und weshalb. In dieser Arbeit wurde, vor allem auf Grund der guten Interpretierbarkeit, der Weg der überwachten Segmentationsbewertung gewählt. Das Ergebnis der Segmentation wird dabei mit einer vorgegebenen idealen Segmentation (Mustersegmentation) verglichen. Um dies durchzuführen wird, wie in Abschnitt 4.2.7 genau dargelegt, folgendes benötigt:

- eine Testmenge, bestehend aus Testbildern und den dazugehörigen Mustersegmentationen
- ein Vergleichskriterium, um die Segmentationsergebnisse mit den Mustersegmentationen vergleichen zu können

Ein vernünftiges Bewertungsverfahren, mit dem zwei Segmentationen verglichen werden können, wird in Abschnitt 4.3 vorgestellt. Es werden grundlegende Eigenschaften dieser Bewertung abgeleitet. Anhand von Beispielen wird gezeigt, dass dieses Kriterium dem subjektiven Empfinden genügt (Abschnitt 4.3.5 und 4.3.6). Im Anschluss wird in Abschnitt 4.4 dargelegt, welche Testdaten verwendet wurden. Die Wahl eines geeigneten Testdatensets[1] ist ein wesentlicher Schritt im überwachten Bewertungsprozess, denn hierdurch wird festgelegt, nach welchem Kriterium die Bewertung durchgeführt wird und welche Störgrößen Einfluss auf die Bewertung haben.
Mithilfe dieses Bewertungsverfahrens wird nun in Abschnitt 4.5 die Gap-Segmentationsmethode genauer untersucht. Im Abschnitt 4.5.1 werden zuerst (optimale) Parameter für die Gap-Segmentation ermittelt. Daraufhin wird die qualitative Leistungsfähigkeit, mit Hilfe der im Abschnitt 4.5.5 eingeführten Charakteristik einer Segmentationsmethode, ermittelt. In diesem Rahmen wird auf Besonderheiten der Gap-Segmentation eingegangen (Abschnitt 4.5.6) und das Verhalten der Gap-Segmentation unter erschwerten Bedingungen untersucht (Abschnitt 4.5.8), wodurch eine weitere Optimierung des Verfahrens durchgeführt werden konnte (Abschnitt 4.5.9).
In Abschnitt 4.6 wird das Gap-Segmentationsverfahren mit drei alternativen Ansätzen verglichen, um die absolute qualitative Leistungsfähigkeit des Verfahrens einschätzen zu können, insbesondere mit der Jahn-Segmentation, welche als Grundlage der Gap-Segmentation diente. Als Abschluss dieses Kapitels wird im Abschnitt 4.7 noch die Gap-Segmentation einer einer realer Satellitenszene analysiert und untersucht, inwieweit sich die bei der Qualitätsanalyse gewonnenen Erkenntnisse auf reale Daten übertragen lassen.

4.1.3 Anforderungen an eine Bewertung

An den qualitativen Bewertungsprozess einer Segmentationsmethode werden folgende Anforderungen gestellt:

[1] bestehend aus Testbildern und den dazugehörigen Mustersegmentationen

- Hinreichende genaue Qualitätsbestimmung
- Objektive Bewertungskriterien
- Aussagekräftige Ergebnisse
- Gute Durchführbarkeit

Die erste und zweifelsfrei wichtigste Anforderung, welche an den Bewertungsprozess gestellt werden muss, ist, dass eine vernünftige Ermittlung der qualitativen Leistungsfähigkeit einer Segmentationsmethode stattfindet. Außerdem sollte die Bewertung objektiv sein, so dass das Bewertungskriterium und die quantitative Bestimmung dieses Kriteriums präzise definiert und exakt nachvollziehbar ist. Dies spielt für das Verständnis, für die Interpretation und die (wissenschaftliche) Verwertbarkeit eine entscheidende Rolle. Außerdem sollten die Ergebnisse der Bewertungsmethode aussagekräftige Ergebnisse liefern, und daher möglichst verständlich und einfach zu interpretieren sein. In vielen Arbeiten werden oft unterschiedlichste Gütebegriffe verwendet und in einer Tabelle oder einem Graph zusammengetragen. Deren Bedeutung wird meistens nur unzureichend erklärt. Dann ist es schwierig, die Ergebnisse überhaupt einzuordnen und zu beurteilen. Letztendlich sollte das Bewertungsverfahren mit möglichst wenig Aufwand durchführbar sein, um den Bewertungsprozess zu erleichtern.

4.1.4 Verschiedene Bewertungsmethoden

Nach [15] gibt es, wie in Abbildung 4.1 dargestellt, folgende grundsätzliche Vorgehensweisen zur Bewertung der Güte einer Segmentation.
Das wichtigste Unterscheidungsmerkmal ist, ob eine Bewertungsanalyse subjektiv oder objektiv erfolgt. Unter subjektiver Bewertung versteht man gewöhnlich die visuelle Begutachtung des Segmentaktionsergebnisses durch einen oder mehrere Experten. Objektive Bewertungen werden normalerweise nach einer Bewertungsregel maschinell ohne den direkten Einfluss eines Menschen durchgeführt.
Objektive Bewertungen können dahingehend unterschieden werden, dass das Gesamtsystem welches die Segmentierung enthält, untersucht wird (Systemweite Bewertung), oder ob eine direkte Bewertung des Segmentationsschrittes vorgenommen wird.
Bei der direkten Bewertungen wird entweder die Methode selbst untersucht (Analytische Bewertung) oder es werden die Ergebnisse der Methode untersucht (Empirische Bewertung). Es gibt zwei Klassen von empirischen Messmethoden.
Bei unüberwachten Methoden wird das Segmentationsergebnis ausschließlich anhand von vorgegebenen Kriterien beurteilt. Bei überwachten Methoden erfolgt ein Vergleich mit einem vorgegebenen Segmentationsergebnis (Mustersegmentation).
Es ist zu bemerken, dass das hier vorgestellte Schema zur Grob-Charakterisierung verschiedener Bewertungsmethoden verwendet werden kann. Ein Bewertungsverfahren weist im allgemeinen Charakteristika mehrerer Kategorien auf[2].

[2] die Charakteristika selbst lassen sich in den meisten Fällen auch nicht eindeutig unterscheiden

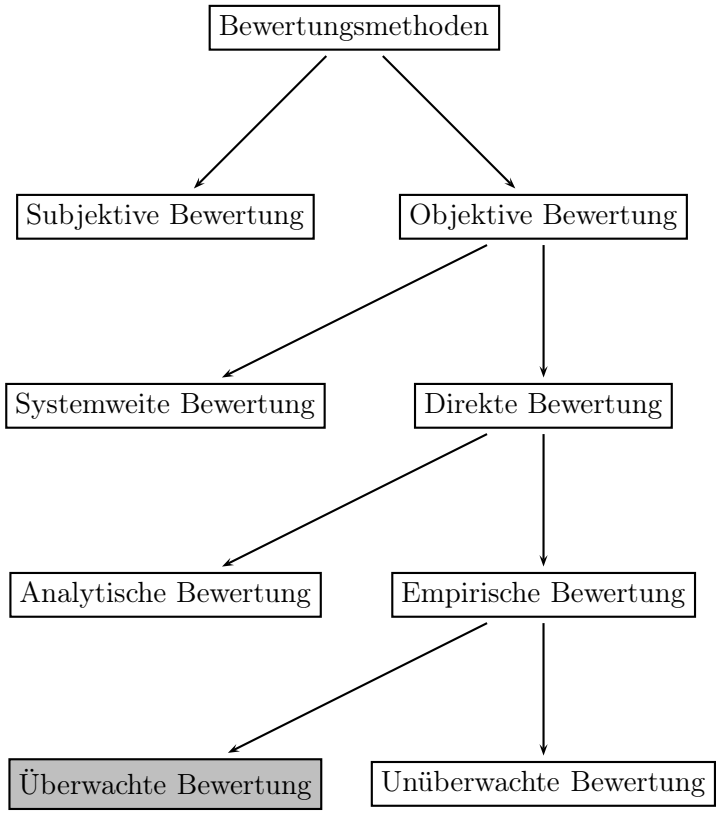


Abbildung 4.1: Schematische Klassifikation von Segmentationsbewertungsmethoden, nach [15]

Anhand der Anforderungen an das Bewertungsverfahren, welche im letzten Abschnitt aufgestellt wurden, erfolgte eine Einschätzung der Eignung dieser Verfahren. Eine visuelle Interpretation durch einen Experten kommt in dieser Arbeit aufgrund der nur sehr schwer sicherzustellenden Objektivität der Messergebnisse, der zeitaufwändigen Durchführung und der Schwierigkeit dritter Personen diese Ergebnisse genau einschätzen und reproduzieren zu können nicht in Betracht.
Da die Segmentationsmethode als Modul für verschiedenste Anwendungen entwickelt wurde, ist eine systemweite Untersuchung nicht möglich. Sie wird aber spätestens bei einem Einsatz der Segmentationsmethode durch den Benutzer durchgeführt werden.
Eine analytische Untersuchung scheidet aufgrund fehlender Methodik ebenfalls aus.
Die unüberwachte Bewertung ist ein komplexes und nur spärlich erforschtes Themengebiet [15]. Es existieren große Schwierigkeiten, geeignete Kriterien zu finden und diese dann zu bewerten. Daher wurde auch von Verwendung derartiger Bewertungsverfahren abgesehen.
Letztendlich wurde entschieden, eine überwachte Bewertung durchzuführen. Der Hauptent-

scheidungsgrund liegt darin, dass bei einer überwachten Bewertung die Interpretation und damit die Auswertung der Ergebnisse im Vergleich zu den anderen Verfahren bei weitem einfacher und am gewinnbringendsten erschien.
In den folgenden Abschnitten wird die in dieser Arbeit verwendete Bewertungsmethodik vorgestellt.

4.2 Theoretische Grundlagen

In diesem Abschnitt wird die Grundlage gelegt, um den Begriff der Güte einer Segmentation exakt formulieren zu können. Zunächst wird kurz erklärt, wie ein Bild definiert werden kann, was genau eine Bildsegmentation ist, was die Güte eine Segmentation ist und wie diese bestimmt werden kann.

4.2.1 Definition eines Bildes

In Folgenden wird eine mathematische Definition des Bildbegriffs gegeben. Dies ist für eine exakte Formulierung der Begriffe innerhalb dieses Kapitels hilfreich.

Definition 1 (Bild) *Ein Bild ist eine Abbildung*

$$\mu : \Omega \to \mathbb{F}$$

wobei $\Omega \subseteq \mathbb{R}^n, \mathbb{F} \subseteq \mathbb{R}^m$ mit $n, m \in \mathbb{N}$. Dabei wird Ω als der Ortsraum des Bildes und $\mathbb{F}$ als der Farbraum des Bildes bezeichnet. n ist die Dimension des Ortsraumes und m die Dimension des Farbraumes.

Bildräume werden folgendermaßen bezeichnet.

Bemerkung 1 (Bildräume) *Sei ein Ortsraum Ω und ein Farbraum $\mathbb{F}$ vorgegeben. Die Menge aller Bilder über Ω und $\mathbb{F}$ werden mit*

$$Bild(\Omega, \mathbb{F})$$

bezeichnet, oder, wenn der Farbraum unerheblich oder im Kontext nicht benötigt wird, vereinfacht mit

$$Bild(\Omega)$$

In dieser Arbeit werden ausschließlich zweidimensionale Bilder, sowohl im Ortsraum als auch Farbraum diskret und beschränkt, betrachtet. Konkret haben die Bilder dann folgende Form

$$\Omega = [0, h-1]_{\mathbb{N}} \times [0, b-1]_{\mathbb{N}}$$

und

$$\mathbb{F} = \prod_{i=1}^{m} [0, DN_i]_{\mathbb{N}}$$

wobei $h \in \mathbb{N}$ die Bildhöhe, $b \in \mathbb{N}$ die Bildbreite, m die Dimension des Farbraums und $DN_i \in \mathbb{N}$ der maximale Wert des i-ten Farbkanals ist. Hierbei ist

$$[a, b]_{\mathbb{N}} = \{x \in \mathbb{N} | a \leq x \leq b\},\ a, b \in \mathbb{N}$$

ein (diskretes) Intervall natürlicher Zahlen von a bis b. Ein monochromatisches Bild (d.h. $m = 1$) mit einer 8-Bit Quantisierung (Farbtiefe) hat dann die Form

$$\mu : [0, b]_{\mathbb{N}} \times [0, h]_{\mathbb{N}} \to [0, 255]_{\mathbb{N}} \quad (= [0, 2^8 - 1]_{\mathbb{N}}), \quad h, b \in \mathbb{N}$$

Dies ist die Beschreibung eines Bildes, welche dieser Arbeit zugrunde gelegt wurde. Abschließend sei bemerkt, dass ein Bild auch auf viele andere Arten beschrieben werden kann. Ein Bild kann auch als Zufallsvariable, wie sie in der Wahrscheinlichkeitstheorie verwendet wird, aufgefasst werden. Konkret wäre ein Bild hier eine Funktion von

$$\mu : (\Omega, \xi_\Omega, P) \to (\mathbb{F}, \xi_\mathbb{F})$$

wobei (Ω, ξ_Ω, P) ein reeller oder diskreter Maßraum und $(\mathbb{F}, \xi_\mathbb{F})$ ein reeller oder diskreter Meßraum wäre. Der Vorteil dieser Beschreibung liegt darin, dass die ganze wahrscheinlichkeitstheoretische/stochastische Maschinerie direkt verwenden werden könnte, beispielsweise die Testtheorie, die Signaltheorie oder die Informationstheorie. Die wahrscheinlichkeitstheoretische Beschreibung ist eine Erweiterung des hier gewählten Begriffes und, falls benötigt, kann darauf zurückgriffen werden. Auch würde ein Vektorfeld auf einer Mannigfaltigkeit eine Erweiterung des Bildbegriffs darstellen. Aufgrund der Komplexität wurde aber darauf verzichtet, einen allgemeineren Bildbegriff von Grund auf zu verwenden. Eine derartige Darstellung wird in dieser Arbeit auch nicht benötigt. Es folgt ein Einschub über lokale Eigenschaften eines Bildes.

4.2.2 Lokale Eigenschaften

In diesem Abschnitt wird der Begriff der Lokalität eines Bildes eingeführt. Es sei bemerkt, dass in dieser Arbeit unter Lokalität immer die Lokalität bezüglich des Ortsraumes verstanden wird.

Definition 2 (Einschränkung eines Bildes) *Sei $\mu \in Bild(\Omega, \mathbb{F})$ ein Bild und $\Omega' \subseteq \Omega$. Die Einschränkung von μ auf Ω', ist das Bild $\mu|_{\Omega'} \in Bild(\Omega', \mathbb{F})$ mit der Eigenschaft*

$$\forall x \in \Omega' \quad \mu(x) = \mu|_{\Omega'}(x)$$

Der Begriff *lokale Bildeigenschaft*, wie er in dieser Arbeit bereits schon häufiger verwendet wurde, kann folgendermaßen exakt definiert werden.

Definition 3 (Eigenschaften von Bildern und Lokalität) *Sei ein Bildraum $Bild(\Omega, \mathbb{F})$ gegeben.*

- *Eine Bildeigenschaft auf $Bild(\Omega, \mathbb{F})$ ist eine Abbildung*

$$E : Bild(\Omega, \mathbb{F}) \to M$$

  Dabei ist M eine beliebige Menge, der sogenannte Merkmalsraum.

- *Sei $\Omega' \subseteq \Omega$. Die Eigenschaft E ist lokal auf Ω' genau dann, wenn eine Eigenschaft $E' : Bild(\Omega', \mathbb{F}) \to M$ existiert, so dass*

$$\forall \mu \in Bild(\Omega, \mathbb{F}) \quad E'(\mu|_{\Omega'}) = E(\mu)$$

Hierdurch wird ausgedrückt, dass die Information, welche zur Bestimmung einer Bildeigenschaft E benötigt wird, bereits in Ω' vorhanden ist. Hiermit endet der Einschub über die Lokalität.

4.2.3 Partition einer Menge

Ein weiterer, im Hinblick auf Segmentation wichtiger Begriff ist die Partition einer Menge.

Definition 4 (Partition) *Eine Partition einer Menge M ist eine Menge P, deren Elemente nichtleere, disjunkte Teilmengen von M sind, so dass jedes Element von M in genau einem Element von P enthalten ist. In anderen Worten*

- $P \subset \mathcal{P}(M)$
- $\forall m \in P \; m \neq \emptyset$
- $\cup_{m \in P} \; m = M$ *[Gesamtheit]*
- $\forall m, n \in P \; m \cap n = \emptyset$ *[Exklusivität]*

Die Menge aller Partitionen über M wird mit $Part(M)$ bezeichnet.

$\mathcal{P}(M)$ ist die Potenzmenge der Menge M, in anderen Worten die Menge aller Teilmengen von M.

4.2.4 Bildsegmentation

Im Folgenden wird eine exakte mathematische Formulierung des Begriffes der Bildsegmentation gegeben.

Definition 5 (Segmentation) *Sei $\mu : \Omega \longrightarrow \mathbb{F}$ ein Bild. Eine Segmentation S von μ ist eine Partition des Ortsraumes Ω. Die Menge aller Segmentationen über Ω wird mit $Seg(\Omega)$ bezeichnet.*

Es sei bemerkt, dass Part(Ω) nach Definition exakt Seg(Ω) entspricht. Da aber von Segmentationen und nicht Partitionen gesprochen werden soll, wird die Schreibweise Seg(Ω) verwendet.
Es ist zu bemerken, dass der Begriff der Segmentation von μ zuerst einmal nur vom Ortsraum Ω abhängt und nicht von der Bildfunktion μ. Die Abhängigkeit zu μ kommt erst durch das Setzen von Segmentationskriterien ins Spiel.

Definition 6 (Segmentationskriterium) *Ein Segmentationskriterium ist durch seine Gütefunktion*

$$K : Bild(\Omega) \times Seg(\Omega) \to \mathbb{R}$$

gegeben. Hier wird die Vereinbarung getroffen, dass große K Werte für eine höhere Güte stehen.

$K(\mu, s)$ beschreibt die Güte der Segmentation s des Bildes μ im Hinblick auf das gewählte Kriterium. Je größer dieser Wert, desto besser ist die Segmentation bezüglich des gewählten Kriteriums. Dies ist ein sehr entscheidender Begriff und zeigt auf, wie ein Kriterium, nach dem segmentiert werden soll, überhaupt beschrieben werden kann. Es ist wichtig, dass der Güte einer Segmentation eine Zahl (Skalar) zugewiesen wird, denn Zahlen können verglichen werden ($<$,$>$,$=$). Sollen mehrere Segmentationskriterien herangezogen werden, müssen diese daher mit einer geeigneten Funktion in die Form von Definition 6 gebracht werden, beispielsweise durch eine gewichtetet Summe $\sum_i \lambda_i * K_i,\ \lambda_i \in \mathbb{R}$.
Das optimale Segmentationsergebnis zu finden kann nun als Optimierungsproblem formuliert werden.

Definition 7 (Optimale Segmentation) *Sei eine Bildfunktion $\mu : \Omega \to \mathbb{F}$, ein Kriterium K und eine Segmentation $S \in Seg(\Omega)$ vorgegeben. Dann ist*

$$S \text{ optimal} \iff \forall S' \in Seg(\Omega)\ K(\mu, S') \leq K(\mu, S)$$

unter der Voraussetzung, dass höhere K Werte ein höhere Güte darstellen.

Hier wird schon deutlich, dass in den meisten Fällen von dem einen optimalen Segmentationsergebnis überhaupt nicht gesprochen werden darf. Bei Bildern mit endlich vielen Bildpunkten gibt es zwar mindestens eine optimale Segmentation[3], es könnte jedoch mehrere geben.
Es ist außerdem sehr wichtig zu bemerken, dass von Güte eines Segmentationsergebnisses nur im Zusammenhang mit einem Kriterium gesprochen werden kann. Eine in diesem Zusammenhang wichtige Erkenntnis ist, dass jede Segmentationsmethode[4] selbst ein Kriterium darstellt.

Bemerkung 2 *Sei ein Bildraum $Bild(\Omega)$ vorgegeben. Außerdem sei $M : Bild(\Omega) \to Seg(\Omega)$ eine Segmentationsmethode. M definiert dann ein Kriterium,*

$$K_M(\mu, S) := \begin{cases} 1 & falls\ S = M(\mu) \\ 0 & sonst \end{cases}$$

unter welchem M immer ein optimales Segmentationsergebnis liefert.

Diese Bemerkung scheint zuerst unsinnig, sie verdeutlicht jedoch, dass zu jeder Segmentationsmethode ein Kriterium existiert, unter welchem dieses optimale Ergebnisse liefert. Bei Vergleichen von Methoden werden oft Verfahren unter Kriterien verglichen, die die gewählte Methode begünstigen. Es wird hieraus dann oft geschlossen, dass das gewünschte Verfahren allgemein besser als andere Verfahren ist. Im Folgenden wird das Kriterium des mittleren Grauwertes mittels der in diesem Abschnitt bereitgestellten Begriffe dargestellt.

4.2.5 Modellkriterien

Dieser Abschnitt dient dazu, einen theoretischen Einblick zu gewähren und zu zeigen, wie das Merkmal des mittleren Grauwerts exakt formuliert werden kann.
Die Angabe eines Segmentationskriteriums in der im letzten Abschnitt dargestellten Schärfe (Definition 6) ist eine schwierige Aufgabe. Eines der allerersten Segmentationskriterien ist in [16] zu finden - *"Regions of an image segmentation should be uniform and homogeneous with respect to some characteristic such as gray tone or texture. Region interiors should be simple and without many small holes. Adjacent regions of a segmentation should have significantly different values with respect to the characteristic on which they are uniform. Boundaries of each segment should be simple, not ragged, and must be spatially accurate"*. Dies ist eine grobe, allgemeingültige Beschreibung. Auch das in dieser Arbeit verwendete Kriterium des mittlerem Grauwertes ist hiervon abgeleitet - Grauwerte eines Segments sollen möglichst einheitlich (Homogenität) sein und sie sollen sich von den Grauwerten anderer Segmente möglichst signifikant unterscheiden (Heterogenität).

[3] da $Seg(\Omega)$ eine endliche Menge ist
[4] bei festgelegter Parameterwahl

Wie kann das Kriterium des mittleren Grauwertes präzisiert werden? Ein mathematisches Modell des mittleren Grauwertes wird durch das Mumford- und Shah-Modell [23] gegeben, welches im Folgenden in Form eines Segmentationskriteriums dargestellt wird.

Beispiel 1 (Mumford- und Shah-Modell) *Sei $\mu : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ eine beschränkte Bildfunktion (d.h. Ω beschränkt). Das Mumford- und Shah-Modell lässt sich durch drei Teilkriterien beschreiben.*

$$K_1(\mu, S, \mu_0) = \int_\Omega (\mu - \mu_0)^2 \, dxdy \quad \textit{(Gesamtabstand)}$$
$$K_2(\mu, S, \mu_0) = \int_{\Omega - S_{Rand}} |\bigtriangledown \mu_0^2| \, dxdy \quad \textit{(Homogenität der Segmente)}$$
$$K_3(\mu_0, S) = |S_{Rand}| \quad \textit{(Seperationsgrad)}$$

wobei $\mu_0 : \Omega \to \mathbb{R} \in Bild(\Omega, \mathbb{R})$ ein Kopplungsparameter ist, und S_{Rand} die Menge der Randpunkte der Segmente ist ($x \in S_{Rand} \subset \Omega \Leftrightarrow \forall$offenen Umgebungen von x (U_x) $\exists y \in U_x$ so dass x und y in verschiedenen Segmenten liegen). Dann kann ein Segmentationskriterium, basierend auf dem Mumford- und Shah-Modell, folgendermaßen definiert werden.

$$K(\mu, S) = -\inf\{K_1(\mu, S, \mu_0) + \lambda K_2(\mu, S, \mu_0) + \nu K_3(\mu_0, S) \mid \mu \in Bild(\Omega)\}$$

Hierbei sind $\lambda, \nu \in \mathbb{R}$ beliebig wählbare Parameter.

Das Kriterium ist für im Ortsraum kontinuierliche Bilder (d.h. $\Omega \subset \mathbb{R}$) gegeben. Diskrete Bilder (d.h. $\Omega \subseteq \mathbb{N}^2$) können durch eine Einbettung von Ω in $\mathbb{R}^2$, beispielsweise durch Darstellung der Bildpunkte als Kacheln, ebenfalls behandelt werden.
Die Teilkriterien K_1, K_2 und K_3 bilden zusammen ein Maß für die Einheitlichkeit der Grauwerte eines Segments, und für die Signifikanz der Differenz der Grauwerte unterschiedlicher benachbarter Segmente.
Dieses Verhalten kann über die Parameter λ, ν gesteuert werden. Über λ kann die Gewichtung der Homogenität und über ν die Gewichtung der Heterogenität geregelt werden. Setzt man $\lambda = 0,\ \nu \geq 0$ wird auf die Homogenität keine Rücksicht genommen und es wird versucht die Heterogenität zu maximieren. Die Heterogenität wird maximiert, wenn nur ein einziges Segment gebildet wird. Umgekehrt wird, falls $\lambda > 0,\ \nu = 0$, die Homogenität maximiert. Im Falle eines diskreten Bilds wäre eine Segmentation, in welcher jeder Punkt ein eigenes Segment darstellt, ein optimales Ergebnis.
Da die Güte der Segmentation ausschließlich aus scharf definierten Kriterien abgeleitet werden kann, gehört es der Klasse der unüberwachten Segmentationskriterien an (Abschnitt 4.1). Hat man jedoch eine Gütefunktion K, wie zum Beispiel das Mumford- und

Shah-Modell, zur Verfügung, kann eine Segmentationsbewertung, wie im folgenden Abschnitt beschrieben, durchgeführt werden. Auf das Mumford- und Shah-Modell speziell wird aber in dieser Arbeit nicht mehr explizit eingegangen, da dessen Anwendung für eine erste Segmentationsbewertung bei weitem den Rahmen gesprengt hätte.

4.2.6 Gütebestimmung einer Segmentationsmethode

In diesem Abschnitt wird dargestellt, wie mithilfe eines Segmentationskriteriums, welches im letzten Abschnitt eingeführt wurde, die Güte einer Segmentationsmethode bestimmt werden kann, welche Mittel dafür notwendig sind.
Zur Gütebestimmung benötigt man ein Segmentationskriterium K und eine Menge von Testbildern.

Definition 8 *Sei ein Bildraum $Bild(\Omega)$ gegeben. Eine (gewichtete) Testmenge ist eine Familie $T_\lambda = \{(\mu_i \in Bild(\Omega), \lambda_i \in \mathbb{R})_{i \in I}\}$*

Durch die λ_i können die Testbilder unterschiedlich gewichtet werden. Wird dies nicht benötigt, werden die Gewichte weglassen. Die Gesamtgüte einer Segmentationsmethode kann dann folgendermaßen bestimmt werden

Definition 9 (Güte einer Segmentationsmethode) *Sei $T_\lambda = \{(\mu_i \in Bild(\Omega), \lambda_i \in \mathbb{R})_{i \in I}\}$ eine gewichtete endliche Testmenge und K ein Kriterium auf $Bild(\Omega)$. Sei $M : Bild(\Omega) \to Seg(\Omega)$ eine Segmentationsmethode auf $Bild(\Omega)$. Die Güte $G_{T_\lambda,K}$ der Segmentationsmethode M über $Bild(\Omega)$, bezüglich des Kriteriums K und der Testmenge T_λ, ist dann gegeben durch*

$$G_{T_\lambda,K}(M) := \sum_{i \in I} \lambda_i * K(\mu_i, M(\mu_i))$$

Normalerweise wird hierbei eine normierte Gewichtung verwendet, d.h. $\sum_{i \in I} \lambda_i = 1$. Wichtig ist, dass von der Güte einer Segmentation nur in Zusammenhang mit der (gewichteten) Testmenge und dem Segmentationskriterium gesprochen werden kann. Da bei der Bewertung ein vorgegebenes Kriterium K verwendet wird, welches einer Segmentation direkt eine Güte zuweist, entspricht dies im Grunde einer unüberwachten Segmentationsbewertung (Abschnitt 4.1).

4.2.7 Überwachte Segmentationsbewertung

Zur Durchführung einer Segmentationsbewertung wird, wie im letzten Abschnitt dargestellt, ein Segmentationskriterium und eine Testmenge benötigt. Die größte Schwierig-

keit besteht darin, ein geeignetes Segmentationskriterium K zu finden, zu formulieren und auszuwerten zu können. Ein geeignetes Segmentationskriterium steht in den meisten Fällen nicht zur Verfügung. Es kann jedoch konstruiert werden, wenn man sich zu den Testbildern Mustersegmentationen[5] vorgibt und eine Vergleichsfunktion zur Verfügung steht, mit der die Segmentationsergebnisse mit der Mustersegmentation verglichen werden können. Dieses Vorgehen kann der überwachten Segmentationsbewertung zugeordnet werden (Abschnitt 4.1.4).
Um eine überwachte Segmentationsbewertung durchführen zu können, wird zu jedem Testbild eine Mustersegmentation benötigt.

Definition 10 (Bewertete (gewichtete) Testmenge) *Eine bewertete (gewichtete) Testmenge auf $Bild(\Omega)$ ist eine Familie*

$$T_{\lambda,S} := \{(\mu_i \in Bild(\Omega), S_i \in Seg(\Omega), \lambda_i \in \mathbb{R})_{i \in I}\}$$

Die S_i sind Mustersegmentationen zu μ_i, und $\lambda_i \in \mathbb{R}$ Gewichtsparameter. Spielt die Gewichtung keine Rolle, spricht man von einer bewerteten Testmenge T_S.

Im Weiteren wird eine Funktion benötigt, mit der ein Vergleich zweier Segmentationen durchgeführt werden kann.

Definition 11 (Vergleichsfunktion) *Sei der Ortsraum Ω gegeben. Ein Vergleichsfunktion auf $Seg(\Omega)$ ist*

$$v : Seg(\Omega) \times Seg(\Omega) \to \mathbb{R}$$

Es wird die Vereinbarung getroffen, dass größere Werte von v für eine höhere Übereinstimmung stehen.

Ist eine Vergleichsfunktion gegeben, kann ein Kriterium $K_{T_S,v}$ auf der bewerteten Testmenge T_S konstruiert werden.

Definition 12 *Sei $T_S = \{(\mu_i \in Bild(\Omega),\ S_i \in Seg(\Omega))_{i \in I}\}$ eine bewertete Testmenge und $v : Seg(\Omega) \times Seg(\Omega) \to \mathbb{R}$ eine Vergleichsfunktion auf $Seg(\Omega)$. Dann ist auf der Testmenge $T = \{(\mu_i \in \Omega)_{i \in I}\}$ folgendermaßen ein Segmentationskriterium gegeben*

$$K_{T_S,v}(\mu_i, S) := v(S_i, S) \quad (\forall i \in I)$$

[5] i.a.W. ideale Segmentationsergebnisse

Hiermit kann, wie in Definition 4.2.6 dargestellt, die Güte einer Segmentationsmethode auf einer bewerteten Testmenge bestimmt werden.

Definition 13 (Überwachte Bewertung einer Segmentationsmethode) *Sei $T_{\lambda,S} = \{(\mu_i \in Bild(\Omega), S_i \in Seg(\Omega), \lambda_i \in \mathbb{R})_{i \in I}\}$ eine endliche (bewertete) gewichtete Testmenge und $v : Seg(\Omega) \times Seg(\Omega) \to \mathbb{R}$ eine Vergleichsfunktion.*
Sei $M : Bild(\Omega) \to Seg(\Omega)$ eine Segmentationsmethode. Die Güte $G_{T_{\lambda,S},v}$ der Segmentationsmethode M über $Bild(\Omega)$, bezüglich der bewerteten gewichteten Testmenge $T_{\lambda,S}$ und der Vergleichsfunktion v, ist gegeben durch

$$G_{T_{\lambda,S},v}(M) := \sum_{i \in I} \lambda_i * K_{T_S,v}(\mu_i, M(\mu_i)) = \sum_{i \in I} \lambda_i * v(S_i, M(\mu_i))$$

Es ist sehr wichtig festzuhalten, dass der Gütebegriff untrennbar von einer bewerteten (gewichteten) Testmenge und einer Vergleichsfunktion ist.
Dies ist auch der Weg, der in dieser Arbeit eingeschlagen wird. In dem folgenden Abschnitt 4.3 wird zuerst eine Vergleichsfunktion konstruiert, und in Abschnitt 4.4 die bewertete (gewichtete) Testmenge vorgestellt.

4.3 Bewertungsverfahren

Wie in Abschnitt 4.1.4 dargelegt, soll eine überwachte qualitative Segmentationsbewertung durchgeführt werden. Hierzu wird das Ergebnis einer Segmentation mit einer vorgegebenen Mustersegmentation verglichen. Um dies durchführen zu können, wird eine Funktion benötigt, mit welcher zwei Segmentationen verglichen werden können.
In diesem Abschnitt wird die Vergleichsfunktion

$$SA_{EQ} : Seg(\Omega) \times Seg(\Omega) \to (0, 1]$$

eingeführt, mit der die Übereinstimmung zweier Segmentationen bestimmt werden kann. Je höher der Wert desto größer ist die Übereinstimmung. Im folgenden Abschnitt wird diese Funktion definiert und es werden erste Folgerungen dieser Funktion abgeleitet. Daran anschließend wird diese Funktion anhand von Beispielen betrachtet und gezeigt, dass damit eine sinnvolle Qualitätsmessung ermöglicht wird (Abschnitt 4.3.5 und 4.3.6).

4.3.1 Grundlegende Begriffe

Um die Prinzipien, auf denen SA_{EQ} beruht, verstehen zu können, wird der Begriff der Verfeinerung einer Segmentation benötigt. Eine Segmentation eines Bildes ist nichts anderes als eine Zerlegung des Ortsraumes in Teilbereiche. Eine Verfeinerung ist nun, umgangssprachlich ausgedrückt, eine feinere Zerlegung.

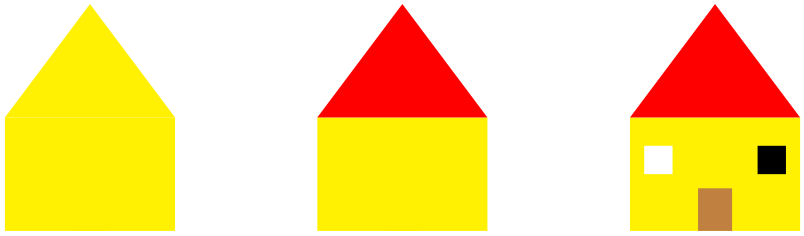
Ein Segmentation entspricht, wie in Abschnitt 4.2.1 definiert, einer Partition des Ortsraumes eines Bildes. Die Partition ist ein zentraler Begriff aus der mathematischen Mengenlehre. Die Folgenden zwei Abschnitte wurden aufgrund größerer Allgemeinheit und Übersichtlichkeit mengentheoretisch formuliert. Es sollte jedoch keine Mühe erforderlich sein, um die entsprechenden Begriffe auf die Segmentation eines Bildes zu übertragen.

Definition 14 (Verfeinerung) *Seien P_1, P_2 Partitionen einer Menge M. Dann ist P_1 eine Verfeinerung von P_2, wenn gilt*

$$\forall m \in P_1 \; \exists n \in P_2 \; m \subseteq n$$

Dann schreibt man $P_1 \leq P_2$, und falls $P_1 \neq P_2$, auch $P_1 < P_2$.

Im Falle einer Segmentation ist $M = \Omega$ der Ortsraum eines Bildes und P_1, P_2 zwei Segmentationen. In Abbildung 4.2 ist ein Beispiel zur Veranschaulichung dargestellt.



(a) (b) (c)

Abbildung 4.2: Verfeinerung einer Segmentation/Partition, (a) >(b) > (c)

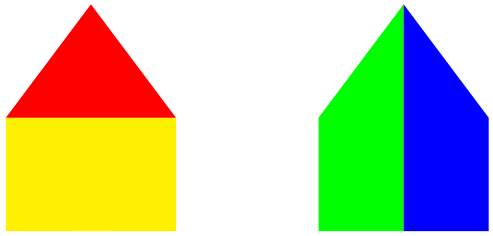
Hierauf sind drei verschiedene Partitionen eines Hauses dargestellt. In Abbildung 4.2(a) ist nur der Grobumriss zu erkennen, wogegen in Abbildung 4.2(b) das Dach extra dargestellt ist. In Abbildung 4.2(c) ist das Haus noch feiner aufgelöst. Durch Herausnehmen der Details[6] erhält man Abbildung 4.2(b). Daher ist das Haus in Abbildung 4.2(c) eine Verfeinerung des Hauses in Abbildung 4.2(b). Ebenso ist das Haus in Abbildung 4.2(b) eine Verfeinerung dessen in Abbildung 4.2(a). Das Haus in Abbildung 4.2(c) ist dann natürlich auch eine Verfeinerung des Hauses in Abbildung 4.2(a). Dies und noch weitere elementare Gesetzmäßigkeiten, die sich direkt aus der Definition ablesen lassen, finden sich in folgender Bemerkung.

[6] dies entspricht dem Gelb Färben der Türen und Fenster in Abbildung 4.2(c)

Bemerkung 3 *Seien P_1, P_2, P_3 Partitionen einer Menge. Dann gilt*

- $P_1 \leq P_1$ *(Reflexivität)*
- $P_1 \leq P_2$ *und* $P_2 \leq P_3 \Longrightarrow P_1 \leq P_3$ *(Transitivität)*
- $P_1 \leq P_2$ *und* $P_2 \leq P_1 \Longrightarrow P_1 = P_2$ *(Antisymmetrie)*

Eine Verfeinerung bildet daher eine Halbordnung auf den Partitionen einer Menge $Part(M)$, $(\mathcal{P}(M), \leq)$ genannt. Jedoch ist $(\mathcal{P}(M), \leq)$ im Allgemeinen keine Totalordnung, d.h. $\exists P_1, P_2 \in \mathcal{P}(M)$, so dass weder $P_1 \leq P_2$ noch $P_2 \leq P_1$ gilt. Die Häuser in Abbildung 4.3 können beispielsweise, durch Umfärben nicht ineinander übergeführt werden. Es gilt daher weder $(a) \leq (b)$ noch $(a) \geq (b)$.



(a) (b)

Abbildung 4.3: Zwei Segmentationen/Partitionen für die weder $(a) \leq (b)$ noch $(a) \geq (b)$ zutrifft

Im Zusammenhang mit einer Segmentation wird auch oft von Übersegmentation und einer Untersegmentation gesprochen. Diese Begriffe sind folgendermaßen zu verstehen:

Bemerkung 4 *Seien S, S_1, S_2 Segmentationen eines Bildes B, wobei S das gewünschte Segmentationsergebnis darstellen soll. Man sagt dann S_1 ist eine (reine) Übersegmentation von B, falls $S_1 \leq S$ und S_2 ist eine (reine) Untersegmentation von B, falls $S \leq S_2$.*

In der Praxisanwendung ist dies ein viel zu strenger Begriff, denn es wird fast nie eine Verfeinerung im strengen Sinne, wie in Definition 14 dargelegt, auftreten. Es wird Üblicherweise von einer Verfeinerung S_{fein} von S gesprochen, wenn es eine Verfeinerung $\tilde{S}_{\text{fein}}$ von S im strengen Sinne gibt, die S_{fein} sehr ähnlich ist. Für die Begriffe Unter- und

Übersegmentation gilt dies entsprechend. Ähnlich bedeutet hier, dass die Vergleichsfunktion $SA_{EQ}(\tilde{S}_{\text{fein}}, S_{\text{fein}})$ hohe Werte annimmt. Die Grundlagen für die Vergleichsfunktion SA_{EQ} werden im folgenden Abschnitt gelegt.

4.3.2 Segmentationsgenauigkeit SA

In [46] wird eine sogenannte SA-Funktion (Segmentation Accuracy) verwendet, um die Qualität eins Segmentationsergebnisses abschätzen zu können[7]. Diese Funktion stellt die Grundlage der Vergleichsfunktion SA_{EQ} dar.

Definition 15 *Sei M eine endliche Menge, P eine Partition von M und $a \subseteq M, a \neq \emptyset$ eine Teilmenge von M. Dann ist*

$$SA(a|P) := \sum_{m \in P} \frac{|a \cap m|}{|m|} * \frac{|a \cap m|}{|a|} \in \mathbb{R}$$

wobei der Betrag $|m|$ einer endlichen Menge m die Anzahl der Elemente von m ist.

Es ist leicht einzusehen, dass SA wohldefiniert ist[8]. Etwas schwieriger ist es folgende Behauptung abzuleiten.

Satz 1 *Sei M eine endliche Menge, P eine Partition von M und $a \neq \emptyset, a \subseteq M$ eine Teilmenge von M. Dann gilt*

a) $SA(a|P) \in (0, 1]$

b) $SA(a|P) = 1 \Leftrightarrow \forall m \in P$ ist $m \cap a = \emptyset$ oder $m \subseteq a$

Beweis: *Dass $SA(a|P) > 0$ folgt aus der Tatsache, dass eine Menge $m \in P$ existiert, so dass $m \cap a \neq \emptyset$[9]. Dass $SA(a|P) \leq 1$ ergibt sich aus folgender Ungleichung:*

$$SA(a|P) = \frac{1}{|a|} * \sum_{m \in P, (m \cap a) \neq \emptyset} \frac{|a \cap m|}{|m|} * |a \cap m| \leq \frac{1}{|a|} * \sum_{m \in P, (m \cap a) \neq \emptyset} |a \cap m| \leq \frac{1}{|a|} * |a| \leq 1$$

Betrachtet man dieser Ungleichung genauer, kann, wie im Folgenden dargestellt, Aussage

[7] diese Funktion geht wahrscheinlich auf [47] zurück
[8] da keine Division durch 0 auftreten kann
[9] da $\cup_{m \in P} = M$ und $a \neq \emptyset$

b) abgeleitet werden.

$$SA(a|P) = 1 \Leftrightarrow \forall(m \in P, (m \cap a) \neq \emptyset) : \ \frac{|a \cap m|}{|m|} = 1 \Leftrightarrow$$
$$\forall(m \in P, (m \cap a) \neq \emptyset) : \ |a \cap m| = |m| \Leftrightarrow$$
$$\forall(m \in P, (m \cap a) \neq \emptyset) : \ m \subseteq a \Leftrightarrow \forall m \in P \text{ ist } m \cap a = \emptyset \text{ oder } m \subseteq a \quad \square$$

Aus Aussage b) des obigen Satzes kann eine sehr interessante Folgerung abgeleitet werden:

Satz 2 *Sei M eine endliche Menge und P_1, P_2 zwei Partitionen von M. Dann gilt*

$$P_1 \leq P_2 \iff \forall a \in P_2 \ (SA(a|P_1) = 1)$$

Beweis: *Aus Aussage b) des obigen Satzes und der Definition der Partition und Verfeinerung folgt:*

$$\forall a \in P_2 \ SA(a|P_1) = 1 \Leftrightarrow \forall a \in P_2 \ \forall m \in P_1 \text{ ist } m \cap a = \emptyset \text{ oder } m \subseteq a \Leftrightarrow P_1 \leq P_2 \quad \square$$

Mit dem SA-Begriff aus Definition 15 können zwei Partitionen (Segmentationen) noch nicht direkt miteinander verglichen werden. Um dies zu ermöglichen wurde in [46] SA auf ganze Partitionen ausgeweitet.

Definition 16 *Seien P_1, P_2 zwei Partitionen einer endlichen Menge M. Dann sei*

$$SA(P_2|P_1) := \frac{\sum_{a \in P_2} SA(a|P_1) \ * \ |a|}{\sum_{a \in P_2} |a|} = \sum_{a \in P_2} SA(a|P_1) * \frac{|a|}{|M|} \in \mathbb{R}$$

$SA(.|.)$ besitzt folgende Eigenschaften.

Satz 3 *Seien P_1, P_2 zwei Partitionen einer endlichen Menge M. Dann gilt*

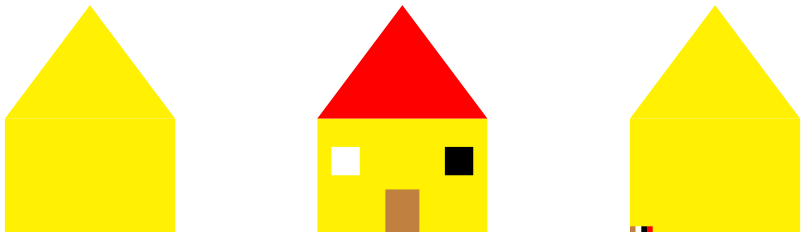
a) $SA(P_2|P_1) \in (0, 1]$

b) $SA(P_2|P_1) = 1 \Leftrightarrow P_1 \leq P_2$

Beweis: *Aussage a) folgt direkt aus Satz 1 a). Aussage b) folgt schnell aus Satz 2, denn es gilt*

$$SA(P_2|P_1) = 1 \Leftrightarrow \forall a \in P_2 \; SA(a|P_1) = 1 \overset{Satz\ 2}{\Longleftrightarrow} P_1 \leq P_2 \quad \square$$

Ist nun $SA(.|.)$ ein geeignetes Mittel, um zwei Partitionen P_1, P_2 zu vergleichen? Nein, denn laut Satz 3b kann, falls P_1 feiner ist als P_2, keine Aussage mehr gemacht werden, da $Sa(P_2|P_1)$ dann immer 1 ist. In [46] wurde deswegen zusätzlich zu $SA(.|.)$ die Anzahl der auftretenden Segmente verglichen. Dies ist aber eine sehr unbefriedigende Lösung, da zwei unterschiedliche Kriterien benötigt werden, um Segmentationen vergleichen zu können. Dies erschwert die Interpretation ungemein. Ein viel schwerwiegender Punkt ist jedoch, dass diese Vorgehensweise, nicht dem subjektiven Empfinden entspricht. In Abbildung 4.4 ist beispielsweise $SA([b]|[a]) = SA([c]|[a]) = 1$, da $[b] \leq [a]$ und $[c] \leq [a]$. Sowohl Abbildung 4.4(b) und 4.4(c) bestehen aus fünf Segmenten. Nach der Bewertung, gegeben durch $SA(.|.)$ und der Anzahl der Segmente, unterscheiden sich 4.4(b) und 4.4(c) demnach im gleichen Maße von 4.4(a). Dies ist aber vom subjektiven Eindruck her sicher nicht der Fall, da die zusätzlichen Segmente in 4.4(b) von der Fläche her viel größer sind als in 4.4(c). In dieser Arbeit wurde daher eine andere Möglichkeit gesucht, die Bewertung mit $SA(.|.)$ zu erweitern.



(a) (b) (c)

Abbildung 4.4: Beispiel SA(.|.): SA([b]||[a]) = SA([c]||[a]) = 1, #[b] = #[c] = 5

4.3.3 SA_{EQ} Vergleichsfunktion

Die SA-Funktion liefert, wie im letzten Abschnitt dargestellt, noch keine zufriedenstellende Segmentationsbewertung, da Verfeinerungen nicht genauer betrachtet werden können (Satz 3b). Daher wurde die SA-Funktion durch eine Symmetrisierung zu der SA_{EQ}-Funktion erweitert.

Definition 17 *Sei M eine endliche Menge, P_1, P_2 zwei Partitionen von M. Dann sei*

$$SA_{EQ}(P_2, P_1) := \sqrt{SA(P_1|P_2) * SA(P_2|P_1)} \in \mathbb{R}$$

SA_{EQ} besitzt folgende Eigenschaften.

Satz 4 *Sei M eine endliche Menge, P_1, P_2 zwei Partitionen von M. Dann ist*

a) $SA_{EQ}(P_1, P_2) \in (0, 1]$

b) $SA_{EQ}(P_1, P_2) = SA_{EQ}(P_2, P_1)$

c) $SA_{EQ}(P_1, P_2) = 1 \Leftrightarrow P_1 = P_2$

Beweis: *Aussage a) folgt direkt aus Satz 3a, 3b folgt direkt aus der Definition und Aussage c) folgt aus Satz 3b und der Tatsache, dass die Verfeinerung antisymmetrisch ist.* □

All dies sind Eigenschaften, welche diese Bewertung sehr sinnvoll erscheinen lassen. Es gibt eine feste Skala (Satz 4a), die Vergleichsfunktion SA_{EQ} ist, wie gängige Abstandsbegriffe, symmetrisch (Satz 4b). Der wichtigste Punkt ist jedoch die Definitheit (Satz 4c), also die Eigenschaft, dass bei exakter Übereinstimmung, und nur dann, der Maximalwert angenommen wird, d.h. $SA_{EQ}(P_1, P_2) = 1$. Es kann also davon ausgegangen werden, dass ein SA_{EQ}-Wert nahe 1 eine hohe Übereinstimmung der Partitionen (Segmente) widerspiegelt. Dies war bei der SA-Funktion, wie in Abbildung 4.4 gezeigt, subjektiv nicht der Fall. Die subjektive Güte der SA_{EQ} Funktion muss jedoch noch untersucht werden (Abschnitt 4.3.5).

Bemerkung 5 *Allgemein könnte die SA_{EQ} Vergleichsfunktion auch definiert werden als*

$$SA_{EQ}(P_1|P_2) = f(SA(P_1|P_2), SA(P_2|P_1))$$

wobei f(x,y) eine Funktion sein soll, die sowohl in x als auch y monoton steigend ist. Hier wird das geometrische Mittel $f(x, y) = \sqrt{x * y}$ verwendet, denn dieses wirkt beruhigend auf Werte nahe eins.

$$0,99 * 0,99 = 0.9801.. \quad \sqrt{0,99 * 0,99} = 0,99$$
$$0,98 * 0,98 = 0,9601.. \quad \sqrt{0,98 * 0,98} = 0,98$$
$$0,97 * 0,97 = 0,9409.. \quad \sqrt{0,97 * 0,97} = 0,97$$

Einen tieferen Grund für die Verwendung der Wurzel gibt es jedoch nicht.

Mithilfe von SA_{EQ} ist es möglich die Übereinstimmung zweier Segmentationen anzugeben. Für eine Qualitätsuntersuchung wird jedoch, wie in Abschnitt 4.1.2 beschrieben, ein Gütebegriff einer Segmentation benötigt.

Definition 18 *Sei M eine endliche Menge M und P eine Partition von M. Durch P wird folgendermaßen eine Norm auf der Menge der Partitionen gebildet*

$$|Q|_P^{SA_{EQ}} := SA_{EQ}(P, Q) \in \mathbb{R}$$

Sei $S_{opt} \in Seg(\Omega)$ eine Mustersegmentation eines Bildes. Die Norm $|.|_{S_{opt}}^{SA_{EQ}}$ bildet dann ein Gütekriterium, welches beschreibt inwieweit ein beliebige Segmentation S der Mustersegmentation S_{opt} entspricht.
Die Norm $|.|_P^{SA_{EQ}}$ hat folgende Eigenschaften.

Bemerkung 6 *Sei M eine endliche Menge und P eine ausgezeichnete Partition von M. Seien P_1, P_2, P_3 drei beliebige Partitionen von M dann gilt*

a) $|P_1|_P^{SA_{EQ}} \leq |P_1|_P^{SA_{EQ}}$ *(Reflexivität)*

b) $|P_1|_P^{SA_{EQ}} \leq |P_2|_P^{SA_{EQ}}$ *und* $|P_2|_P^{SA_{EQ}} \leq |P_3|_P^{SA_{EQ}} \Longrightarrow |P_1|_P^{SA_{EQ}} \leq |P_3|_P^{SA_{EQ}}$ *(Transitivität)*

c) $|P_1|_P^{SA_{EQ}} \leq |P_2|_P^{SA_{EQ}}$ *oder* $|P_2|_P^{SA_{EQ}} \leq |P_1|_P^{SA_{EQ}}$ *(Totalität)*

Beweis: Beide Aussagen folgen direkt aus der Definition 18.[10] □

Es ist jedoch folgender wichtiger Punkt zu beachten.

Bemerkung 7 *Im allgemeinen gilt* ***nicht!!!***

$$|P_1|_P^{SA_{EQ}} = |P_2|_P^{SA_{EQ}} \Rightarrow P_1 = P_2$$

Die Norm $|.|_P^{SA_{EQ}}$ partitioniert daher die Menge der Partitionen auf M.

[10] Die Betragsfunktion $|.|_P^{SA_{EQ}}$ bildet daher auch eine totale Quasiordnung auf der Menge der Partitionen von M

Definition 19 *Seien M eine endliche Menge und P, Q Partitionen von M. Dann sei*

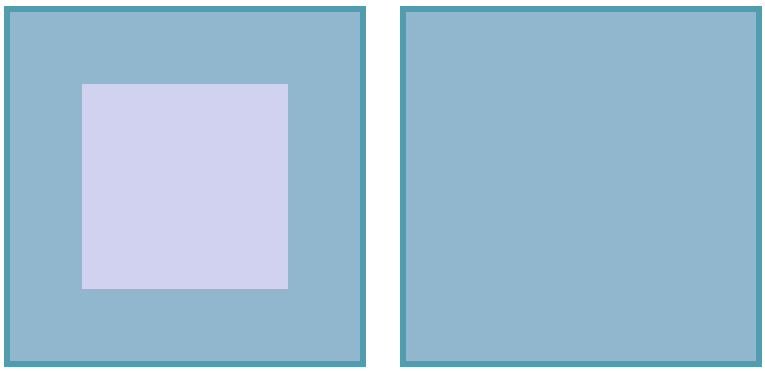
$$[Q]_P^{SA_{EQ}} := \{R \in Part(M) \mid |R|_P^{SA_{EQ}} = |Q|_P^{SA_{EQ}}\}$$

die Äquivalenzklasse von Q bzgl. der Ordnung $|.|_P^{SA_{EQ}}$.

Es gilt insbesondere $Q \in [Q]_P^{SA_{EQ}}$ und $[P]_P^{SA_{EQ}} = \{P\}$. Diese Äquivalenzklassen entsprechen Güteklassen von SA_{EQ}. Sie geben unter anderem wieder, wie unterschiedliche Fehlerarten relativ zueinander gewichtet werden. Sie stellen ein weiteres wichtiges Kriterium dar, an welchem überprüft werden kann ob SA_{EQ} ein vernünftiger Gütebegriff ist. In Abschnitt 4.3.6 wird hierauf genauer eingegangen.

4.3.4 SA_{EQ} Beispielrechnung

In diesem Abschnitt soll an einem praktischen Beispiel gezeigt werden, wie die SA_{EQ} Funktion berechnet werden kann. Hierzu werden die in Abbildung 4.5 dargestellten Segmentationen S_{bimodal}, S_{unimodal} verwendet.



(a) (b)

Abbildung 4.5: Test-Segmentationen *a)* S_{bimodal}, *b)* S_{unimodal},
Bildgröße 60 x 60 Pixel, inneres Viereck 34 x 34 Pixel,
Anzahl Segmente: S_{bimodal} 2, S_{unimodal} 1

Es soll nun $|S_{\text{uni}}|_S^{SA_{EQ}} = SA_{EQ}(S_{\text{uni}}, S)$ berechnet werden. Hierfür müssen zuerst die einzelnen Segmente bezeichnet werden:

$$S := S_{\text{bimodal}} = \{s_{\text{außen}}, s_{\text{innen}}\},\ S_{\text{uni}} := S_{\text{unimodal}} = \{s_{\text{ges}}\}$$

s_{innen} ist hier das innere Viereck (hell), $s_{\text{außen}}$ das innere Viereck umgebende Segment (dunkel) von S_{bimodal}. s_{ges} ist das einzige Segment von S_{uni}.

Es gilt:

$$|s_{\text{ges}}| = 60 * 60 = 3600,\ |s_{\text{innen}}| = 34 * 34 = 1156$$

und

$$|s_{\text{außen}}| = |s_{\text{ges}}| - |s_{\text{innen}}| = 60 * 60 - 34 * 34 = 2444$$

Nun kann mit der Berechnung begonnen werden:

$$|S_{\text{uni}}|_S^{SA_{EQ}} = SA_{EQ}(S, S_{\text{uni}}) = \sqrt{SA(S_{\text{uni}}|S) * SA(S|S_{\text{uni}})}$$

Außerdem:

$$SA(S_{\text{uni}}|S) = \frac{\sum_{a \in S_{\text{uni}}} SA(a|S) * |a|}{\sum_{a \in S_{\text{uni}}} |a|} = \frac{SA(s_{\text{ges}}|S) * |s_{\text{ges}}|}{|s_{\text{ges}}|} = SA(s_{\text{ges}}|S)$$

$$SA(s_{\text{ges}}|S) = \sum_{m \in S} \frac{|s_{\text{ges}} \cap m|}{|m|} * \frac{|s_{\text{ges}} \cap m|}{|s_{\text{ges}}|} = \frac{|s_{\text{ges}} \cap s_{\text{innen}}|^2}{|s_{\text{innen}}| * |s_{\text{ges}}|} + \frac{|s_{\text{ges}} \cap s_{\text{außen}}|^2}{|s_{\text{außen}}| * |s_{\text{ges}}|} =$$
$$(\frac{|s_{\text{innen}}|^2}{|s_{\text{innen}}|} + \frac{|s_{\text{außen}}|^2}{|s_{\text{außen}}|}) * \frac{1}{|s_{\text{ges}}|} = \frac{|s_{\text{innen}}| + |s_{\text{außen}}|}{s_{\text{ges}}} = \frac{s_{\text{ges}}}{s_{\text{ges}}} = 1$$

Daher ist $SA(S_{\text{uni}}|S) = 1$. Dieses Ergebnis hätten mit Satz 3b) sofort eingesehen werden können, da $S < S_{\text{uni}}$ eine Verfeinerung ist. Weiterhin gilt:

$$SA(S|S_{\text{uni}}) = \frac{SA(s_{\text{außen}}|S_{\text{uni}}) * |s_{\text{außen}}| + SA(s_{\text{innen}}|S_{\text{uni}}) * |s_{\text{innen}}|}{|s_{\text{außen}}| + |s_{\text{innen}}|}$$

Es ist

$$SA(s_{\text{außen}}|S_{\text{uni}}) = \frac{|s_{\text{außen}}|^2}{|s_{\text{ges}}| * |s_{\text{außen}}|} = \frac{|s_{\text{außen}}|}{|s_{\text{ges}}|}$$

und

$$SA(s_{\text{innen}}|S_{\text{uni}}) = \frac{|s_{\text{innen}}|^2}{|s_{\text{ges}}| * |s_{\text{innen}}|} = \frac{|s_{\text{innen}}|}{|s_{\text{ges}}|}$$

daher

$$SA(S|S_{\text{uni}}) = \frac{(|s_{\text{außen}}|^2) + |s_{\text{innen}}|^2) * \frac{1}{|s_{\text{ges}}|}}{|s_{\text{ges}}|} = \frac{|s_{\text{außen}}|^2 + |s_{\text{innen}}|^2}{|s_{\text{ges}}|^2}$$

Nun kann das Gesamtergebnis berechnet werden

$$|S_{\text{uni}}|_S^{SA_{EQ}} = \sqrt{\frac{|s_{\text{außen}}|^2 + |s_{\text{innen}}|^2}{|s_{\text{ges}}|^2} * 1} = \frac{2444^2 * 1156^2}{3600^2} = 0,7510009780...$$

4.3.5 Subjektive Bewertung von SA_{EQ}

In den letzten Abschnitten wurden die SA_{EQ} Funktion eingeführt und einige theoretische Aussagen abgeleitet. Ob diese Funktion aber für eine Qualitätsanalyse geeignet ist, ist hierdurch noch nicht gezeigt. In diesem Abschnitt wird eine subjektive Bewertung von SA_{EQ} vorgenommen. Diese ist sehr wichtig, um einschätzen zu können, ob die SA_{EQ} Norm für eine Qualitätsanalyse geeignet ist. In der Abbildung 4.6 sind zunächst Beispiele mit unterschiedlichen SA_{EQ} Werten dargestellt. Zusätzlich zum Gesamtqualitätsmaß $|.|_S^{SA_{EQ}}$ sind hier zusätzlich die $SA(.|S)$ und $SA(S|.)$ Werte angegeben. Ein großer $SA(S|.)$ und ein kleiner $SA(.|S)$ Wert ist ein Indiz für eine hohe Übersegmentation, umgekehrt für eine hohe Untersegmentation[11].

Die SA_{EQ} Norm scheint im Einklang mit dem subjektiven menschlichen Eindruck zu sein. In Abbildung 4.6(a) wird ein SA_{EQ} Wert von eins angenommen, die Segmentation entspricht infolgedessen der Mustersegmentation[12]. SA_{EQ} Werte über 0,97 liefern visuell betrachtet sehr gute Ergebnisse, SA_{EQ} Werte über 0,9 machen bis auf kleinere Störungen einen guten Eindruck. SA_{EQ} Werte im 0,8 Bereich zeigen bereits erhebliche Störungen. Das innere Segment ist allerdings noch gut zu erkennen. Unter einem SA_{EQ} Wert von 0,8 wird es kritisch, große Störungen treten auf. Im Falle von Abbildung 4.6(g) fehlt das mittlere Segment völlig, bei Abbildung 4.6(h) ist das mittlere Segment noch gut zu erkennen, aber der Rand zerfällt völlig und bei Abbildung 4.6(i) ist kaum noch etwas zu erkennen.

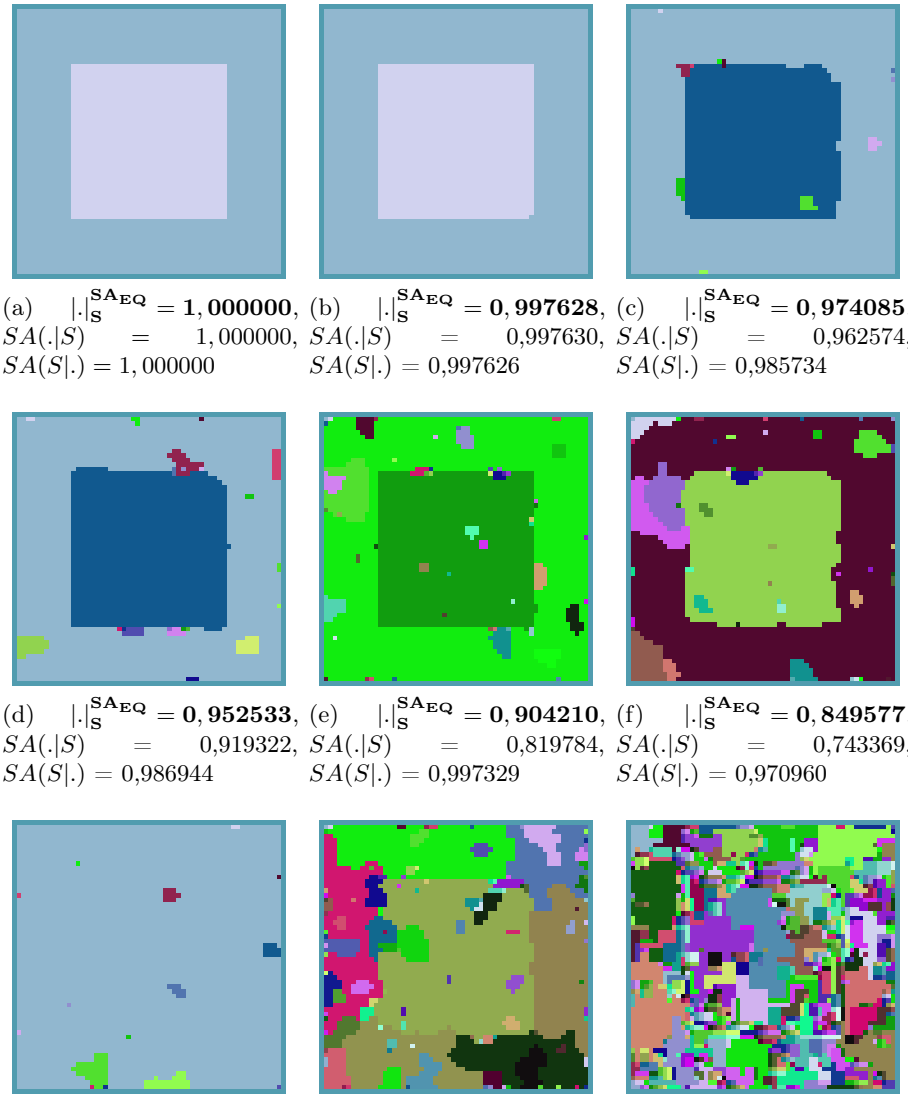
Dass Abbildung 4.6(g) besser gewertet wird als Abbildung 4.6(h), widerspricht dem visuellen Eindruck. Das liegt daran, dass dem inneren Segment subjektiv eine sehr hohe Bedeutung zugemessen wird, bezüglich der SA_{EQ} Norm aber beide Segmente gleichwertig sind.

Aufgrund des hohen $SA(S|.)$ Wertes ist zu erkennen, dass in den meisten Bildern eine Übersegmentation vorliegt. Nur im Falle von Abbildung 4.6(g) tritt am hohen $SA(.|S)$ Wert erkennbar eine Untersegmentation auf. Auch hier stimmt der visuelle Eindruck mit dem gemessenen überein.

An diesen Beispielen ist zu erkennen, dass weder der $SA(S|.)$Wert noch der $SA(.|S)$ Wert alleine für eine Qualitätsbestimmung ausreichen. Würde nur der $SA(S|.)$ Wert verwendet werden, würde Abbildung 4.6(i) höher als Abbildung 4.6(c) bewertet. Würde im Gegensatz nur der $SA(.|S)$ Wert verwendet werden, würde Abbildung 4.6(g) höher als Abbildung 4.6(d) bewertet werden. Beides stände im völligen Widerspruch zum subjektiven visuellen Eindruck.

Es ist zu bemerken, dass der konkrete SA_{EQ} Wert von der zugrunde liegenden Mustersegmentation abhängt. Der relative Eindruck bleibt aber erhalten. Es wird sich aber später zeigen, dass die SA_{EQ} Norm für erste Untersuchungen sehr gut geeignet ist. Im Vergleich zu einer visuellen Untersuchung können viel genauere und objektivere Aussagen getroffen werden. Es wurden Ergebnisse erzielt, die mit einer subjektiven Begutachtung nicht gewonnen hätten werden können.

[11] Abschnitt 4.3.2, Satz 3
[12] vgl. Satz 4

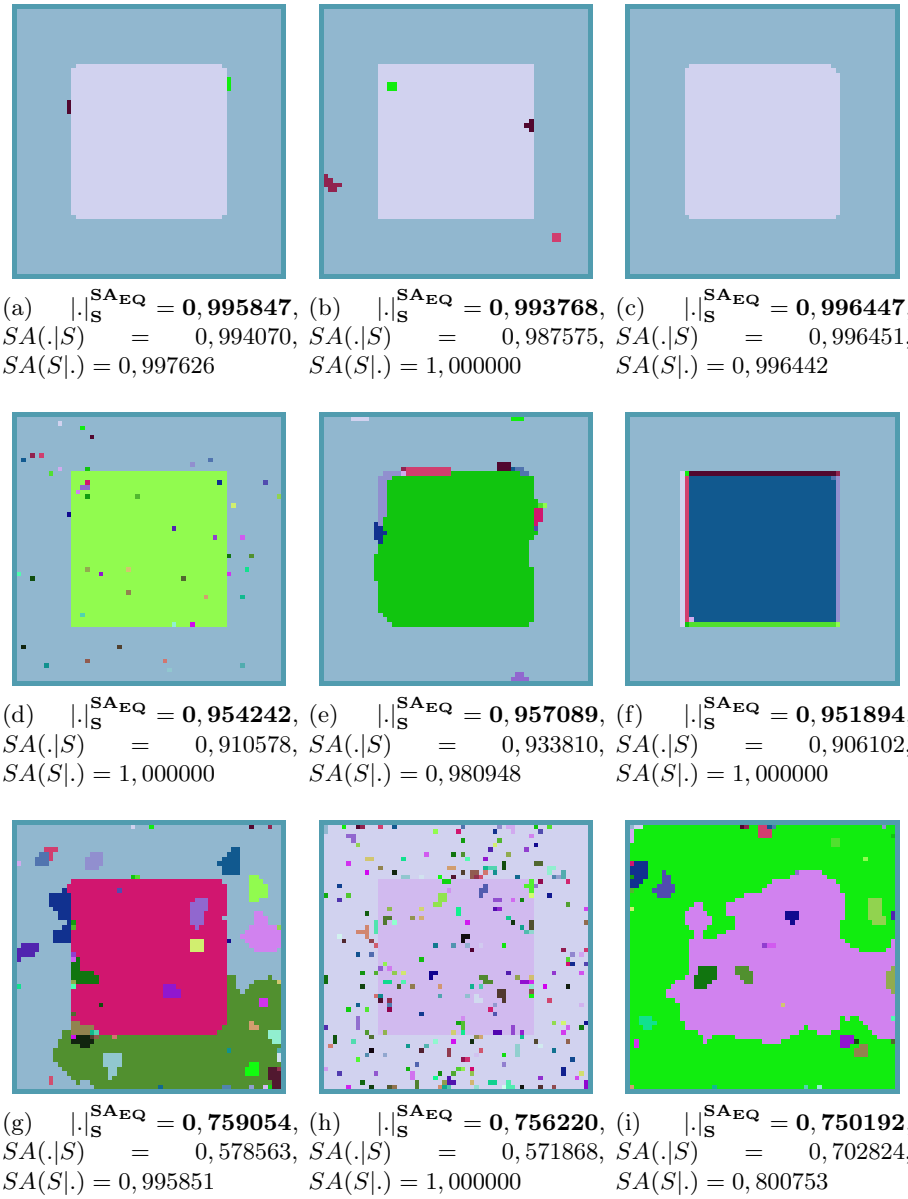


(a) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{1,000000}$, $SA(.|S) = 1,000000$, $SA(S|.) = 1,000000$
(b) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,997628}$, $SA(.|S) = 0{,}997630$, $SA(S|.) = 0{,}997626$
(c) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,974085}$, $SA(.|S) = 0{,}962574$, $SA(S|.) = 0{,}985734$
(d) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,952533}$, $SA(.|S) = 0{,}919322$, $SA(S|.) = 0{,}986944$
(e) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,904210}$, $SA(.|S) = 0{,}819784$, $SA(S|.) = 0{,}997329$
(f) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,849577}$, $SA(.|S) = 0{,}743369$, $SA(S|.) = 0{,}970960$
(g) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,724951}$, $SA(.|S) = 0{,}939033$, $SA(S|.) = 0{,}559676$
(h) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,541057}$, $SA(.|S) = 0{,}302299$, $SA(S|.) = 0{,}968390$
(i) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,169027}$, $SA(.|S) = 0{,}028713$, $SA(S|.) = 0{,}995005$

Abbildung 4.6: Beispiele für verschiedene SA_{EQ} Bewertungsstufen

4.3.6 Äquivalenzklassen von SA_{EQ}

Im letzten Abschnitt haben wir Beispiele mit unterschiedlichen SA_{EQ} Werten betrachtet und festgestellt, dass die Bewertung in dieser Hinsicht vernünftig zu sein scheint. In Abbildung 4.7 werden nun verschiedene Beispiele mit vergleichbarer Güte vorstellen. Ziel ist es, einen besseren Eindruck des von $|.|_S^{SA_{EQ}}$ induzierten Gütebegriffs zu bekommen.



(a) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,995847}$, $SA(.|S) = 0,994070$, $SA(S|.) = 0,997626$

(b) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,993768}$, $SA(.|S) = 0,987575$, $SA(S|.) = 1,000000$

(c) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,996447}$, $SA(.|S) = 0,996451$, $SA(S|.) = 0,996442$

(d) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,954242}$, $SA(.|S) = 0,910578$, $SA(S|.) = 1,000000$

(e) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,957089}$, $SA(.|S) = 0,933810$, $SA(S|.) = 0,980948$

(f) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,951894}$, $SA(.|S) = 0,906102$, $SA(S|.) = 1,000000$

(g) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,759054}$, $SA(.|S) = 0,578563$, $SA(S|.) = 0,995851$

(h) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,756220}$, $SA(.|S) = 0,571868$, $SA(S|.) = 1,000000$

(i) $|.|_{\mathbf{S}}^{\mathbf{SA_{EQ}}} = \mathbf{0,750192}$, $SA(.|S) = 0,702824$, $SA(S|.) = 0,800753$

Abbildung 4.7: Relativer Vergleich verschiedener SA_{EQ} Bewertungen

Sind die $|.|_S^{SA_{EQ}}$ Werte nahe eins, stimmen die Ergebnisse noch mit der natürlichen Empfindung überein, d.h. die Segmentierungen mit SA_{EQ} Werten auf dem gleichen Niveau hinterlassen einen vergleichbaren (subjektiven) Qualitätseindruck. Bei den Beispielen mit SA_{EQ} Werten unter 0,75 verliert man jedoch das Gefühl, was gut oder schlecht ist. Dass die Segmentationen in den Abbildungen 4.7(g) - 4.7(i) qualitativ gleichwertig sind, ist nicht mehr ganz so leicht nachzuvollziehen. Bei Betrachten der $SA(.|S)$ und $SA(S|.)$

Werte fällt auf, dass sowohl in Abbildung 4.7(g) als auch in Abbildung 4.7(h) eine starke Übersegmentation vorliegt, wohingegen in Abbildung 4.7(i) weder eine ausgeprägte Untersegmentation noch Übersegmentation stattfindet. Dass Abbildung 4.7(i) aber bezüglich SA_{EQ} genauso gewichtet wird, liegt an der Symmetrie der SA_{EQ} Vergleichsfunktion. Dies bedeutet, dass nicht nur die Abweichung von der Mustersegmentation bestimmt wird, sondern die Abweichungen voneinander. Würde zum Beispiel in Abbildung 4.7(h) davon ausgegangen, dass das fehlerhafte Bild die Mustersegmentation, und S das gestörte Bild darstellt, dann würde ein massiver Informationsverlust vorliegen.
Gerade bei hohen SA_{EQ} Werten stimmt aber das subjektive Empfinden und ein Unterschied zwischen den Qualitätsstufen ist gerade dort deutlich zu erkennen. Die $|.|_S^{SA_{EQ}}$ Norm eignet sich daher auf alle Fälle gut für eine erste Qualitätsuntersuchung. Dies gilt im Besonderen, wenn ein einzelnes Verfahren untersucht wird[13]. Denn hier tritt in den meisten Fällen oft die gleiche Fehlerart auf[14].

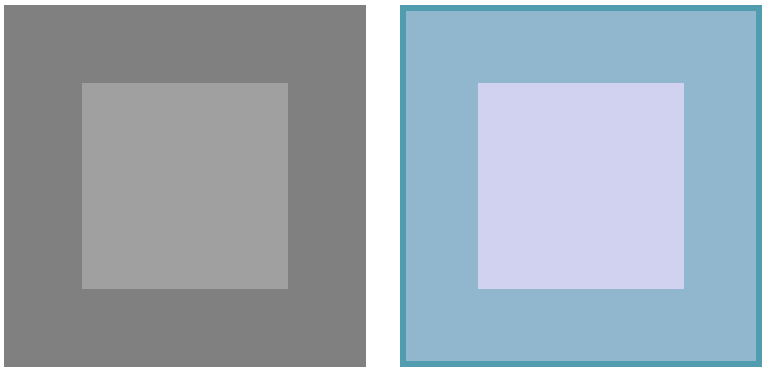
4.4 Testdaten

Mithilfe der Vergleichsfunktion SA_{EQ} kann, wie in den letzten Abschnitten aufgezeigt, eine Segmentationsbewertung durchgeführt werden, indem die Abweichung des Ergebnisses einer Segmentationsmethode von einem Musterergebnis, bezüglich einer Testdatenmenge ermittelt wird. Diese Testdatenmenge hat einen großen Einfluss auf das Bewertungsergebnis, und muss daher sorgfältig ausgewählt werden. Die Wahl der Testmenge und der dazugehörigen Mustersegmentationen entscheiden ebenfalls im großem Maße darüber, nach welchem Segmentationskriterium eine Bewertung durchgeführt wird[15], und welche Bildstörungen in die Bewertung einfließen[16]. In dieser Arbeit wird, wie in Abschnitt 2.3 begründet, das Kriterium des mittleren Grauwerts verwendet. Daher sollte sich dieses Kriterium anhand des Testdatensets überprüfen lassen.
Es sei angemerkt, dass hier nicht überprüft werden soll, wie geeignet ein Segmentationskriterium für eine bestimme Aufgabe ist. Dies ist eine andere Frage. Die Fragen, nach der Eignung eines Segmentationskriteriums und wie gut nach diesem Kriterium segmentiert wird, wird leider oft vermischt.

4.4.1 Testmodell

Die Grundlage der Testdatenmenge stellt das in Abbildung 4.8 gezeigte und bereits wohlbekannte Testbild dar.
Aus den folgenden Gründen wurde entschieden, ein synthetisches Testmodell, bestehend aus einem Vordergrundobjekt und einem Hintergrundobjekt, zu verwenden. Der erste Punkt ist, dass eine Mustersegmentation zur Verfügung steht. Hier ist es eindeutig klar, wie nach dem Merkmal des mittleren Grauwerts segmentiert werden muss. Es kann gezeigt werden, dass die Mustersegmentation, bei einer vernünftigen Parameterwahl, auch

[13] Beispielsweise im Falle einer Parameterbestimmung
[14] vgl. Abbildungen 4.7(d) und 4.7(h)
[15] vgl. Abschnitt 4.2.7, Definition 12
[16] vgl. Abschnitt 4.2.7, Definition 13



(a) (b)

Abbildung 4.8: Grundlegendes Testmodell: a) Testbild b) Mustersegmentation Gesamtgröße 60×60 Pixel, inneres Viereck 34×34 Pixel

die einzige optimale Lösung des Mumford- und Shah-Modells (Abschnitt 4.2.5) darstellt. Dagegen ist es in den meisten Fällen schwierig, eine geeignete Mustersegmentation einer realen Szene zu erzeugen. Hierbei ist oft überhaupt nicht ersichtlich, wie nach Merkmal des mittleren Grauwertes segmentiert werden soll.
Der zweite wichtige Punkt ist, dass das Testmodell auf die wesentlichsten Charakteristika beschränkt ist, die zur Untersuchung des Segmentationskriteriums (mittlerer Grauwert) benötigt werden. Erst hierdurch wird eine gezielte Interpretation der Messergebnisse ermöglicht und dadurch die Grundlage für ein tieferes Verständnis geschaffen. Da im Testbild nur zwei Segmente auftreten, kann beispielsweise sehr eindeutig abgeleitet werden, wie hoch die Grauwertdifferenz zweier benachbarter Bildbereiche sein muss, damit eine Separation dieser stattfindet (Abschnitt 4.5.5). Auf realen Satellitenszenen mit vielen Bereichen unterschiedlichen mittleren Grauwertes kann dies nicht erfolgen.
Der dritte Punkt ist, dass Störgrößen gezielt eingefügt, exakt beschreiben und modifiziert werden können. Auf realen Szenen sind im Allgemeinen verschiedenste Arten von Störungen vorhanden. Der Einfluss einzelner Störgrößen kann dann nur noch eingeschränkt analysiert werden, auch weil sich die Stärke dieser Störung nicht ohne weiteres verändern lässt.
Der vierte und letzte Punkt, welcher für die Verwendung eines einfachen synthetischen Testmodells spricht, ist, dass sich die Testergebnisse relativ leicht interpretieren und mit relativ geringem Aufwand reproduzieren lassen. Dies erleichtert Anwendern und Wissenschaftlern, die Leistungsfähigkeit einer Segmentationsmethode einzuschätzen und eine Bewertung bei vergleichbaren Testbedingungen durchzuführen zu können.

4.4.2 Störmodell

Ein Störmodell, basierend auf Gaußrauschen und dem Verschmieren der Kanten durch einen 3×3 Mittelwertfilter wurde angewendet, um reale Bedingungen zu simulieren. Es

wurden auch reale Satellitenszenen analysiert (Abschnitt 4.7) und gezeigt, dass dieses Störmodell eine gute Näherung realer Bedingungen darstellt.

4.4.3 Testbilder

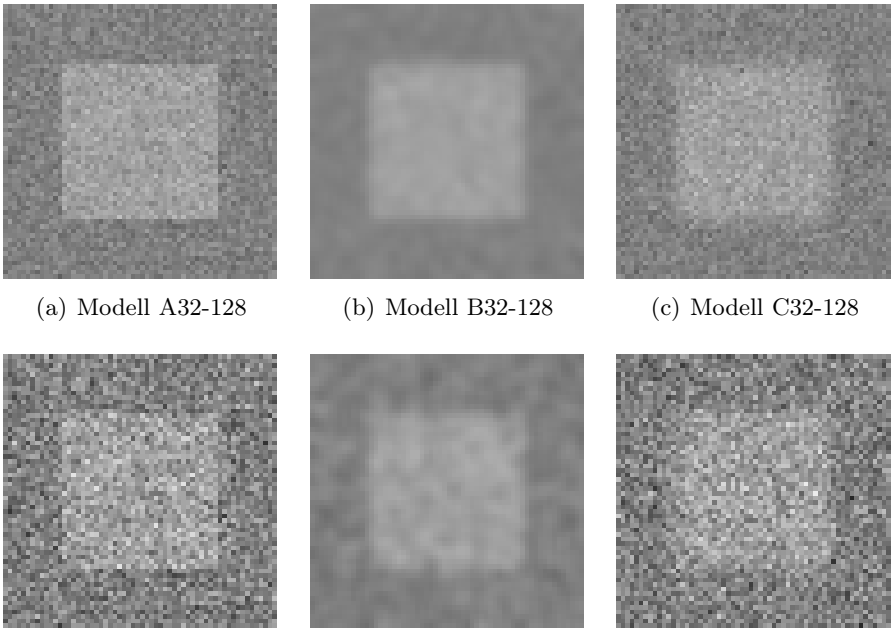
Auf dem Testbild finden sich zwei Segmente, ein Viereck in der Mitte und dessen Umrandung. Das mittlere Viereck hat eine Größe von 34×34 Bildpunkten und das ganze Bild eine Größe von 60×60 Bildpunkten. Die Grauwertdifferenz zwischen dem inneren Viereck und dem Hintergrund wurde zwischen 0 und 64 Grauwertstufen variiert. Drei Störmodelle wurden verwendet

1. Modell A
   a) Versehen des Testmodells mit Gaußrauschen
2. Modell B
   a) Versehen des Testmodells mit Gaußrauschen
   b) Anschließend zweimalige Filterung mit einem 3×3 Mittelwertfilter
3. Modell C
   a) Zweimalige Filterung des Testmodells mit einem 3×3 Mittelwertfilter
   b) Anschließend Hinzufügen von Gaußrauschen

Im zweiten und dritten Modell wurde ein gewöhnlicher 3×3-Mittelwertfilter (Tiefpass) verwendet, um die Bildkanten zu verschmieren. In dieser Weise sollten Nachbarschaftseffekte an den Kanten simuliert werden. Zur Größenangabe des Gaußrauschens wurde die Standardvarianz benutzt. Die Standardvarianz entspricht der Quadratwurzel der Standardabweichung ($\delta = \sqrt{\text{var}}$).
Bei den Messungen stellte sich heraus, dass nicht nur die Stärke des Rauschens eine Rolle spielt, sondern im großen Maße auch die konkrete Verteilung des Rauschens. Rauschen der gleichen Stärke kann günstig oder ungünstig verteilt sein. In Abschnitt 4.5.5 wird hierauf näher eingegangen. Aus diesem Grund wurden pro Messpunkt mithilfe des Rauschgenerators jeweils 129 verschiedene Versionen eines Testbildes erzeugt. Jeder Messwert entspricht dem Mittel der über die 129 Versionen gewonnenen Messwerte.
In der formalen Darstellung (Abschnitten 4.2.7) entspricht die bewertete gewichtet Testmenge, welche zur Bestimmung eines Messpunktes verwendet wurde, folgendem Term:

$$T_{\frac{1}{129},S} := (\mu_i \in Bild(\Omega, \mathbb{F}), s_i = S \in Seg(\Omega), \lambda_i = \frac{1}{129} \in \mathbb{R})_{i \in [0,128]_{\mathbb{N}}}$$

Hierbei sind μ_i die mithilfe des Rauschgenerators erzeugten Testbilder, $\Omega = 60 \times 60$ ist der Ortsraum, $\mathbb{F} = [0, 255]_{\mathbb{N}}$ der Farbraum und S die Mustersegmentation (Abbildung 4.8(b)).
Beispiele der Störmodelle sind in Abbildung 4.9 dargestellt.



(a) Modell A32-128 (b) Modell B32-128 (c) Modell C32-128

(d) Modell A32-576 (e) Modell B32-576 (f) Modell C32-576

Abbildung 4.9: Beispiele der Störmodelle A/B/C bezüglich einer mittleren (Varianz: 128) bzw. hohen (Varianz: 576) Rauschstufe und einer Grauwertdifferenz der Bildbereiche von 32 Grauwertstufen.

4.5 Gap-Segmentation

4.5.1 Parameterbestimmung - Übersicht

In diesem Abschnitt wird eine Parameterbestimmung für den Gap-Segmentationsalgorithmus vorgenommen. Wie bereits in Abschnitt 4.1.1 aufgezeigt, besteht das Gap-Segmentationsverfahren aus drei Teilschritten:

1. Filterstufe 1: $\boldsymbol{X}$-*mal* Mittelwertfilter 3×3
2. Filterstufe 2: $\boldsymbol{X}$-*mal* Gap-Glättungsfilter $\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3}$
3. Segmentationsstufe: Schwellwert $\boldsymbol{t}$

Es wird untersucht, wie viele Vorfiltrierungs- und Hauptfiltrierungsläufe (Filterstufe 1, 2) für ein zufriedenstellendes Ergebnis notwendig sind. Der Gap-Glättungsfilter ist außerdem von den drei Parametern g_1, g_2, g_3 (Abschnitt 3.2.2) abhängig, welche vernünftig gewählt werden sollten. Auch muss ein geeigneter Schwellwert t für die Segmentationsstufe (Abschnitt 3.3) festlegt werden. Da diese Parameter voneinander abhängig sein könnten, müsste eine 6-dimensionale Parameteruntersuchung[17] durchgeführt werden. Diese Unter-

[17] Läufe Vorfilter, Läufe Hauptfilter, g_1, g_2, g_3, t

suchung sollte außerdem jeweils für die in Abschnitt 4.4 dargestellten Testszenarien A, B und C vorgenommen werden, um ihre Eignung unter unterschiedlichen Bedingungen analysieren zu können. Da dies einen viel zu hohen Aufwand bedeuten würde, wird ein anderer Ansatz gewählt. Es wird von zwei Parametereinstellungen ausgegangen, einer empfindlicheren Einstellung und einer moderaten Einstellung:

Filtereinstellung a): 2 × Mittelwertfilter 3 × 3; *15*× Gap-Glättung 1.5, 1.5, 2; Segmentationsschwellwert 1 (hohe Empfindlichkeit)

Filtereinstellung b): 2 × Mittelwertfilter 3 × 3; 15 × Gap-Glättung 1.5, 1.5, 4; Segmentationsschwellwert 1 (moderate Empfindlichkeit)

Diese haben sich unter vielen im Vorfeld untersuchten Stichproben als günstig erwiesen. Es werden nun, von diesen Filtereinstellungen ausgehend, jeweils einzelne Parameter abgeändert, mit dem Ziel zu überprüfen, ob diese Parameter vernünftig gesetzt wurden oder es bessere Alternativen gibt. Diese Parameteruntersuchung wurde zuerst an Testszenario A durchgeführt. Anschließend wurde überprüft, inwieweit diese Parameter auch für die Testszenarien B und C sinnvoll sind (Abschnitt 4.5.8). Es wurden hiervon ausgehend weitere Verbesserungen vorgenommen. Die finalen Filtereinstellungen sind in Abschnitt 4.5.9 zu finden.

4.5.2 Läufe Vorfilter - Filterstufe 1

Im Folgenden wird der Einfluss der Vorfilterläufe[18] auf die Segmentationsqualität untersucht. In Abbildung 4.10 ist das Untersuchungsergebnis für die Modell-A32 Testreihe (Filtereinstellung b) dargestellt. Die Qualität der Segmentation (SA_{EQ}) ist in Abhängigkeit verschiedener Rauschstufen[19] und der Anzahl der Vorfilterläufe aufgetragen.
Es stellte sich heraus, dass eine Vorfiltrierung einen großen Einfluss auf das Testergebnis hat und auf jeden Fall sinnvoll ist[20]. Mit zweimaliger Vorglättung kann bei moderaten Rauschstufen (bis 400) ein sehr guter SA_{EQ} Wert nahe eins erreicht werden. Erst bei höheren Rauschstufen (ab 400) wird mit einer 3- oder 4- fachen Vorglättung ein besseres Ergebnis erzielt. Es wird, auch bei niedrigen Rauschstufen, maximal ein SA_{EQ} Wert von 0,95 erreicht. Die Qualität nimmt generell bei höheren Rauschstufen, wie zu erwarten, ab.
Diese Untersuchung wurde ebenfalls anhand der Modelle A16, A32, B16, B32, C16, C32 Testbildern[21] bezüglich den Filtereinstellungen a und b durchgeführt. Bei allen an Modell-A durchgeführten Untersuchungen ergibt sich das eben aufgezeigte Bild. Mit 2-facher Vorglättung kann bei niedrigen und moderaten Rauschstufen ein sehr gutes Ergebnis erzielt werden. Die Modell-B Testbilder unterscheiden sich von den Modell-A Bildern dadurch, dass bei ihnen, um die Kanten zu verschmieren, eine 2 - fache 3 × 3-Mittelwertfilterung vorgenommen wurde[22]. Dies entspricht einer zweifachen Vorfiltrie-

[18] Filterstufe 1, 3 × 3-Mittelwertfilter
[19] Abschnitt 4.4
[20] vgl. Abbildung 4.10, $x = 0$ zu $x = 1$ - 4
[21] Abschnitt 4.4
[22] Abschnitt 4.4

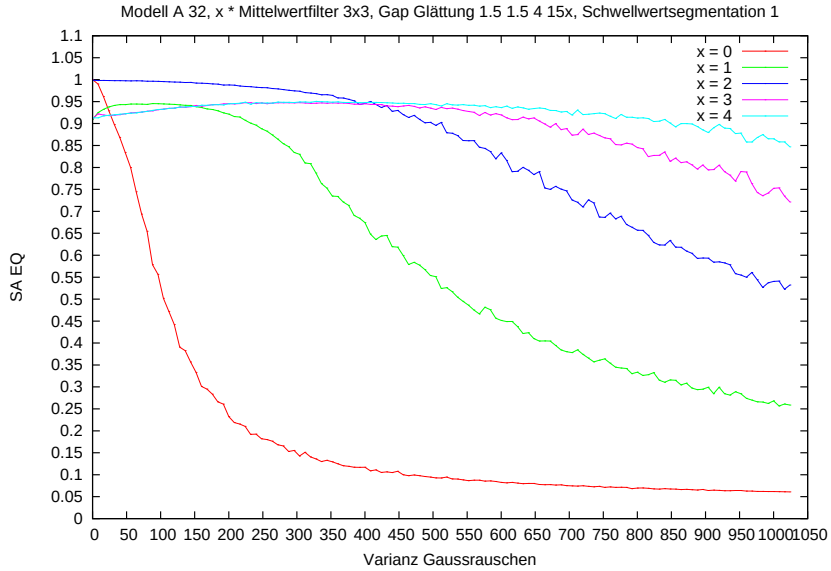


Abbildung 4.10: Vorfilterläufe, Modell-A32, Filtereinstellung b, x Anzahl der Vorfilterläufe

rung. Die Ergebnisse des Modells A mit $(X = x+2)$ entsprechen daher denen des Modells B mit $(X = x)$. Es ist daher nicht verwunderlich, dass in Modell-B bei null Filterläufen das beste Ergebnis erreicht wurde. Bei den Modell-C Testbildern, wurde ebenfalls mit zwei Vorfiltrierungsläufen, bei geringen und moderaten Rauschstufen, sowohl bei den Filtereinstellungen a und b insgesamt eine sehr gutes Ergebnis erzielt.
Es wurde entschieden, zwei Vorfiltrierungsläufe auszuführen, da nur hierdurch bei Modell-A und Modell-C sehr gute Qualität bei niedrigen und moderaten Rauschstufen erreicht werden konnte. In Abschnitt 4.5.9 wird die Filterleistung durch weitere Optimierungen noch weitaus verbessert. Hierdurch konnten auch die Nachteile der 2 - fachen Vorglättung, welche in erster Linie im Modell-B Testszenario auftraten, beseitigt und auch bei höheren Rauschstufen gute Ergebnisse erzielt werden.

4.5.3 Läufe Hauptfilter - Filterstufe 2

Im vorherigen Abschnitt wurde der Einfluss der Vorfilterläufe auf die Segmentationsqualität untersucht. Im Folgenden wird der Einfluss der Hauptfilterläufe auf die Segmentationsqualität untersucht. Es soll in erster Linie die minimale Anzahl der Läufe, welche für ein gutes Segmentationsergebnis notwendig ist, bestimmt werden. Dies ist wichtig, da die Anzahl der Hauptfilterläufe einen großen Einfluss auf die Gesamtlaufzeit des Segmentationsprozesses hat.
Außerdem stellt sich die Frage, ob nach Erreichen eines optimalen Segmentationsergebnisses weitere Filterläufe einen negativen Einfluss auf die Segmentationsqualität haben oder nicht. In dieser Arbeit wird in diesem Zusammenhang über **die Stabilität eines**

Filters bezüglich seiner Läufe gesprochen. Beispielsweise ist ein gewöhnlicher Mittelwertfilter bezüglich seiner Läufe nicht stabil, da das Bild mit jedem zusätzlichen Filterlauf immer weiter verwaschen wird bis letztendlich das gesamte Bild einen einheitlichen Grauwert annehmen würde (Abbildung 3.6).
Die Segmentationsqualität wurde in Abhängigkeit von den Hauptfilterläufen und den Rauschstufe untersucht. Beispielsweise ist, bezüglich den Modell-A32 Testbildern und Filtereinstellung b, auf Abbildung 4.11 auf den ersten Blick zu erkennen, dass mit einer steigenden Zahl an Hauptfilterläufen die Segmentationsqualität zunimmt, bis sie ein Maximum erreicht hat. Die Qualität bleibt ab diesen Punkt maximal, die Gap-Glättung ist also bezüglich der Anzahl der Hauptfilterläufe stabil.

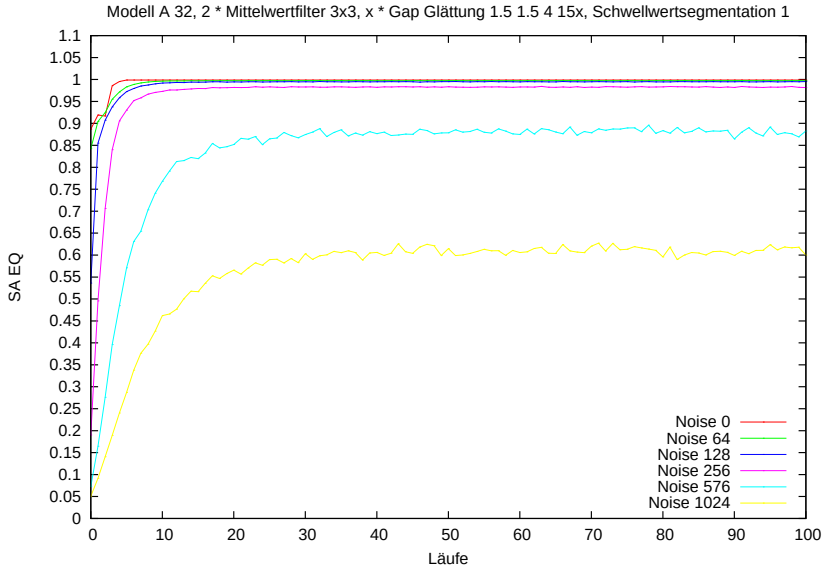


Abbildung 4.11: Haupftfilterläufe, Modell A32, Filtereinstellung b

Es wurden ebenfalls die Testmodelle A16, A32, B16, B32, C16, C32 bei den Filtereinstellungen a und b auf die Auswirkung der Hauptfilterläufe untersucht. Bei allen Testszenarien wird in den meisten Fällen, auch bei hohen Rauschstufen, ab 15 Durchläufen die maximal erreichbare Qualität erzielt. Bei niedrigen Rauschstufen, bis 128, reichen teilweise schon fünf Läufe aus. Fünf Läufe liefern jedoch bei mittleren und höheren Rauschstufen bei weitem kein optimales Ergebnis. Daher wurde entschieden, 15 Hauptfilterdurchläufe durchzuführen, da dies einen guten Kompromiss zwischen der Geschwindigkeit und der Qualität darstellt.

4.5.4 Weitere Parameter

Die Parameter der Gap-Glättung, g_1, g_2, g_3 (Abschnitt 3.2.2), wurden ebenfalls untersucht. Es hat sich herausgestellt, dass die Parameter $g_1 = 1,5$, $g_2 = 1,5$, $g_3 =$

2 und $g_3 = 4$, welche auch bei den Filtereinstellungen a und b verwendet wurden, eine gute Wahl darstellen.
Dies trifft auch auf den Schwellwert der Segmentation $t = 1$ zu. Obwohl man geneigt ist anzunehmen, dass dieser viel zu gering und daher eigentlich sehr störanfällig sein sollte, hat sich $t = 1$ als günstiger Wert erwiesen. Durch Erhöhung des Segmentationsschwellwertes kann die Empfindlichkeit der Segmentation gesenkt werden. Hierdurch kann einer möglichen Übersegmentation entgegengewirkt werden.
Eine Schwierigkeit hierbei ist, dass für jede Änderung des Segmentationsschwellwertes wieder eine komplette Parameteruntersuchung durchgeführt werden müsste, da eventuell eine Abhängigkeit zu anderen Parametern besteht. Außerdem ist nicht unbedingt gesagt, dass ein anderer Segmentationsschwellwert ebenso gut mit den anderen Parametern harmoniert. Für einen Einsatz in der Praxis wird empfohlen, die sehr gut harmonierenden und dokumentierten finalen Filtereinstellungen, wie sie in Abschnitt 4.5.9 vorgestellt werden, zu verwenden. Die Änderung der Empfindlichkeit einer Segmentation wird durch eine im Vorfeld durchgeführte Kontrastspreizung/-senkung des Eingangsbildes erreicht. Für diese Aufgabe kann eine einfache affine Transformation verwendet werden, $\mu_{neu}(P) = a * \mu_{alt}(P) + b$, wobei $a, b \in \mathbb{R}$ die Transformationsparameter darstellen. Da diese Operation lokal ist[23], kann sie einfach, ohne Geschwindigkeitsverlust, als eine weitere Vorfilterstufe in das Gesamtsystem integriert werden[24].

4.5.5 Empfindlichkeit

In den letzten Abschnitten wurde eine Parameterbestimmung durchgeführt. Es wurde jedoch noch keine Aussage über die Leistungsfähigkeit der Gap-Segmentationsmethode getroffen. Dies soll nun in diesem Abschnitt nachgeholt werden.
Es muss zuerst geklärt werden, was unter dem Begriff Leistungsfähigkeit einer Segmentationsmethode zu verstehen ist. In der Literatur wird dieses Thema leider nur sehr dürftig behandelt, doch es konnte Inspiration in einem anderen Themengebiet, der Messtechnik, gefunden werden. In diesem spielt die Empfindlichkeit, beispielsweise eines Messgeräts, eine entscheidende Rolle und gibt Auskunft darüber, wie stark ein Eingangssignal sein muss, damit es erkannt werden kann. In dieser Arbeit wird nach dem Merkmal des mittleren Grauwerts segmentiert - die Bildpunkte eines Segments sollen ähnliche Grauwerte besitzen und sich von den Grauwerten benachbarter Segmente möglichst signifikant unterscheiden (Abschnitt 2.3).
Was liegt nun näher, als zu untersuchen, wie die Segmentationsqualität von der Differenz der Grauwerte zweier benachbarter Bildbereiche abhängt und insbesondere ab welcher Grauwertdifferenz eine Trennung stattfindet. Konkret wurde die Segmentationsqualität in Abhängig der Grauwertdifferenz zwischen dem inneren und äußeren Bereich der Testbilder untersucht. In dieser Arbeit wird in diesem Zusammenhang von der Empfindlichkeit, aber auch von der (Filter-)Kennlinie oder (Filter-)Charakteristik einer Segmentationsmethode gesprochen.

[23] sogar nur vom Bildpunkt direkt abhängig
[24] Abschnitt 3.1

Die (Filter-)Charakteristik der Gap-Segmentationsmethode, bezüglich des Modell-A Testszenarios und den Filtereinstellungen a und b (Abschnitt 4.5.1) findet sich in den Abbildungen 4.12 und 4.13. Bei beiden Filtereinstellungen steigt die Segmentationsqualität mit Zunahme der Grauwertdifferenz, bis auf den Sonderfall der Rauschstufe 0 (kein Rauschen). Bei höheren Rauschstufen wird außerdem ein schlechteres Ergebnis erzielt als bei niedrigeren Rauschstufen. Dies war zu erwarten, kann jedoch durch dieses Ergebnis substantiell belegt werden.

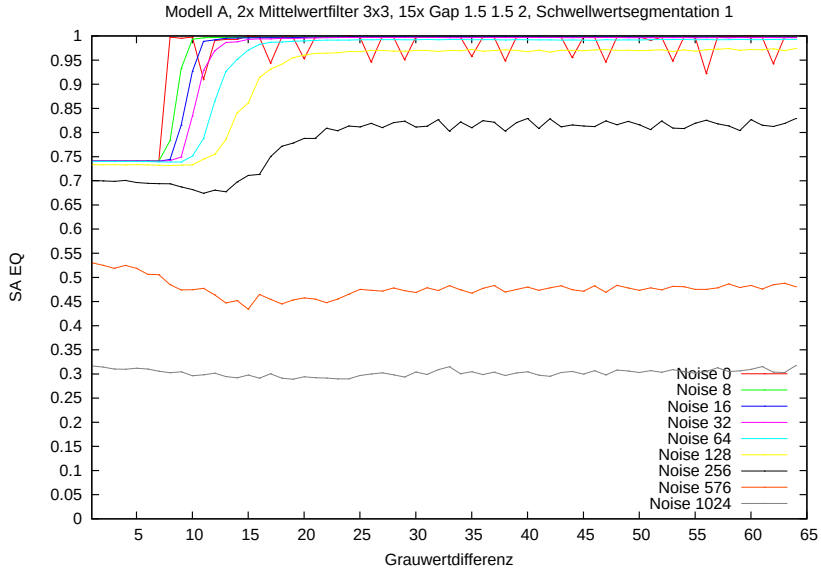


Abbildung 4.12: Empfindlichkeitscharakteristik der Gap Segmentation, Filtereinstellung a

Bei Rauschstufe 0 tritt überraschenderweise bei beiden Parametereinstellungen ein unregelmäßiges Verhalten auf. Auf diese Besonderheit wird in Abschnitt 4.5.6 näher eingegangen. Auch ist der Unterschied zwischen den Filtereinstellungen a und b erkennbar. Bei der empfindlicheren Filtereinstellung a wird bei kleinen Rauschstufen bei einer Grauwertdifferenz ab 10 aufwärts sehr gute Qualität erreicht, wogegen bei Parametereinstellung b hierfür Grauwertdifferenz ab 15 benötigt werden. Dagegen kann mit Parametereinstellung b ab Rauschstufe 256 eine deutlich höhere Segmentationsqualität erzielt werden.
Die Empfindlichkeit ist natürlich von der Rauschstufe und der Kantenschärfe[25] abhängig. Je schwieriger die Bedingungen sind[26], desto schlechter werden die Ergebnisse. Die Empfindlichkeit einer Segmentationsmethode gibt insbesondere darüber Auskunft, wie groß die Grauwertdifferenz bei einer gewissen Rauschstufe und Kantenschärfe sein muss, damit ein gutes Segmentationsergebnis erreicht wird. Hierdurch kann die Leistungsfähigkeit einer Segmentationsmethode eingeschätzt werden. Somit kann untersucht werden, in

[25] simuliert durch Modell A, B oder C
[26] höhere Rauschstufe, niedrigere Kantenschärfe

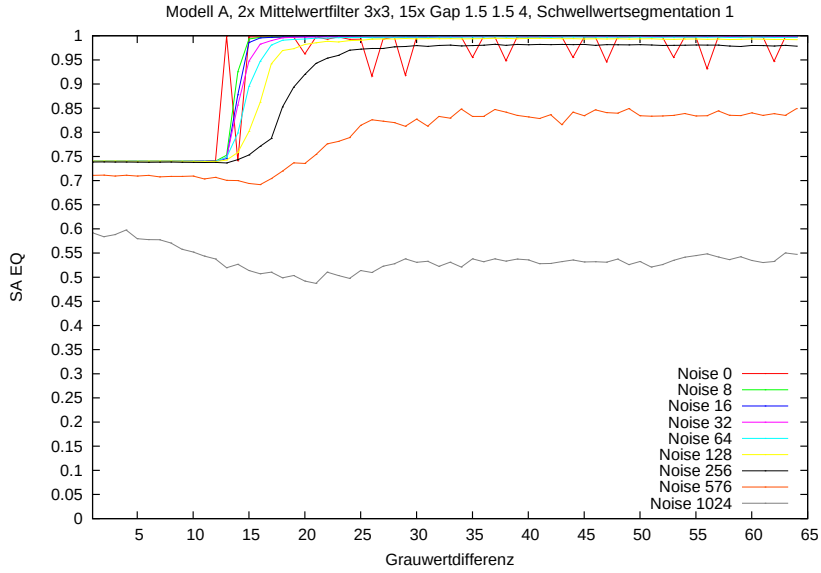


Abbildung 4.13: Empfindlichkeitscharakteristik der Gap Segmentation, Filtereinstellung b

welchem Bereich die Segmentationsmethode einsetzbar ist und ab wann sie versagt.
Der Kurvenverlauf der Empfindlichkeit einer Segmentationsmethode, bezüglich einer Rauschstufe, kann in drei Bereiche eingeteilt werden:

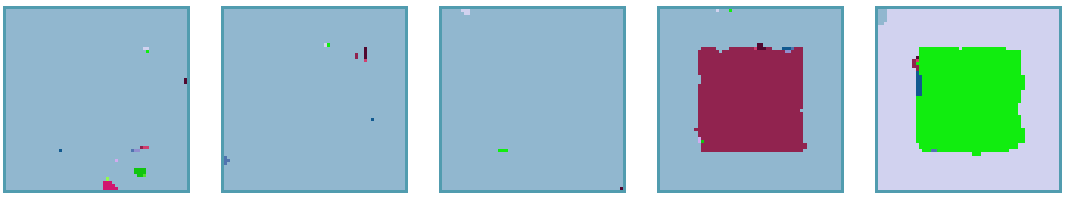
1. Nicht sensibler Bereich
2. Übergangsbereich
3. Maximaler Bereich

Auf diese drei Bereiche wird am Beispiel der Empfindlichkeitscharakteristiken in Abbildung 4.12 und 4.13 genauer eingegangen.

Nicht sensibler Bereich
Der nicht sensible Bereich definiert sich dadurch, dass hier der Grauwertunterschied für eine Trennung der zwei Bereiche des Testbildes zu gering ist. Der innere und der äußere Bereich werden nicht getrennt. Dies äußert sich, wie in Abschnitt 4.3.4 berechnet, durch einen konstanten SA_{EQ}-Wert von ungefähr 0,75. Bezüglich der Rauschstufe 128 nimmt der nicht sensible Bereich bei Filtereinstellung a (Abbildung 4.12) Werte zwischen 0 - 10 und Filtereinstellung b (Abbildung 4.13) zwischen 0 - 15 an. Es ist auch deutlich zu erkennen, dass der nicht sensible Bereich sich bei höheren Rauschstufen ausweitet, mit anderen Worten, die Empfindlichkeit sinkt. Bei sehr hohen Rauschstufen wachsen die Segmente im nicht sensiblen Bereich nicht nur zusammen, sondern es treten zusätzlich weitere Störungen auf. Dies hat zur Folge, dass der SA_{EQ}-Wert beispielsweise unter

Filtereinstellung a (Abbildung 4.12) bei den Rauschstufe 256, 576 und 1024 unter 0,75 rutscht.

Übergangsbereich
Als Übergangsbereich wird der Bereich bezeichnet, welcher den nicht sensiblen Bereich in den Bereich stabiler maximaler Qualität überführt.
Der Beginn und die Länge des Übergangsbereiches sind von der Rauschstufe abhängig. Sowohl bei Filtereinstellung a als auch b (Abbildung 4.12 und 4.13) ist deutlich zu erkennen, dass der Übergangsbereiches bei höheren Rauschstufen erheblich breiter wird. Interessant ist außerdem, dass innerhalb des Übergangsbereiches der Kurvenverlauf der Filtercharakteristik beinahe geradlinig ansteigt und keinen komplexen Verlauf hat.
Im Übergangsbereich kann kein zuverlässiges Segmentationsergebnis erzielt werden, da die Segmentationsqualität in diesem Bereich sehr stark von der räumlichen Verteilung des Rauschens abhängt. Um dies verstehen zu können, sei daran erinnert, dass ein Messwert das Mittel einer Testreihe, bestehend aus 129 Testbildern, darstellt[27]. Im Übergangsbereich schwanken die SA_{EQ}-Werte innerhalb einer Testreihe sehr stark. Als Beispiel soll die Filtercharakteristik bei Filtereinstellung a (Abbildung 4.12) herangezogen werden. Bei Rauschstufe 64 liegt der SA_{EQ}-Wert bei der Grauwertdifferenz 12, im unteren Übergangsbereich. Die ersten fünf Testbilder dieser Testreihe haben die SA_{EQ}-Werte 0,74, 0,74, 0,74, 0,98 und 0,96. Das innere Viereck wird hierbei, wie in Abbildung 4.14 dargestellt, in den ersten drei Fällen nicht erkannt.

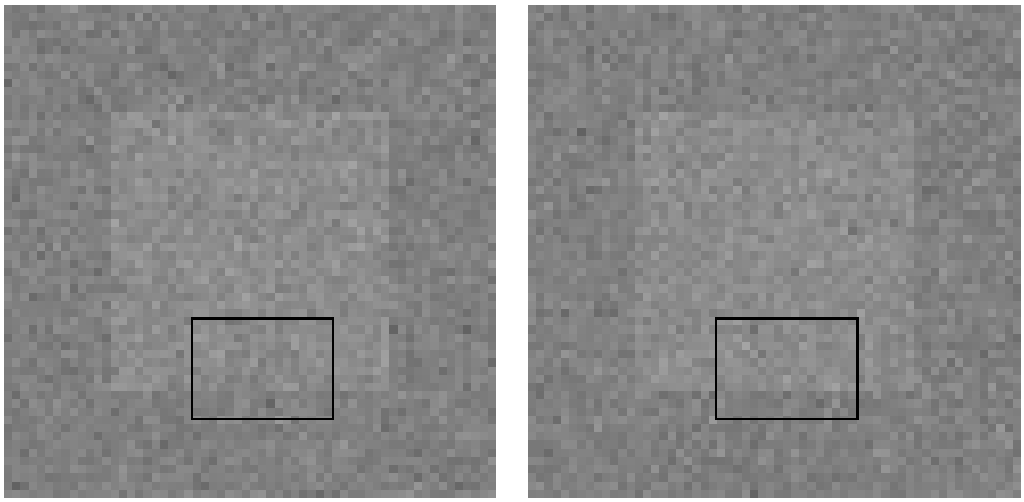


(a) (b) (c) (d) (e)

Abbildung 4.14: Segmentationsergebnisse der ersten fünf Testbilder einer Testreihe, welche im Rahmen der Bestimmung der Filtercharakteristik bezüglich Filtereinstellung a bei einer Grauwertdiffrenz von 12 und einer Rauschstufe von 64 durchgeführt wurde.

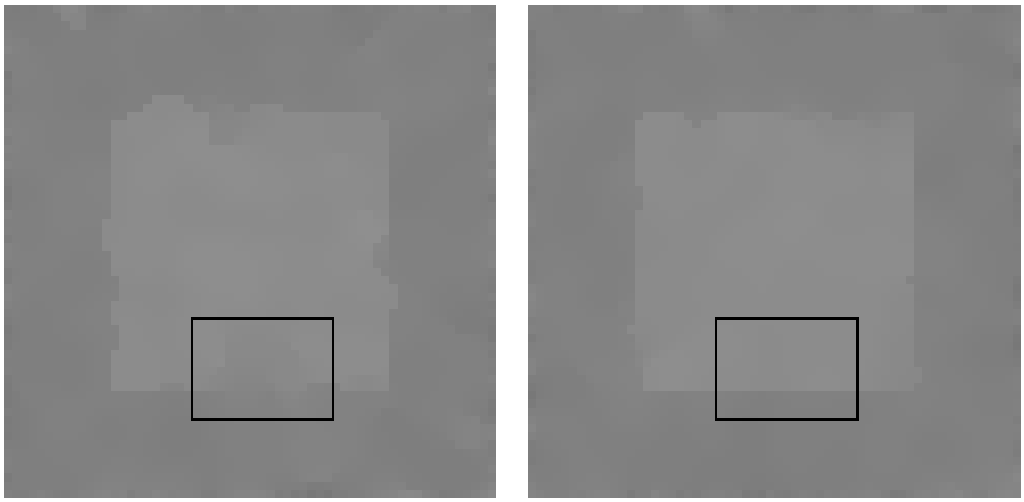
Was ist nun unter der räumlichen Abhängigkeit des Rauschens zu verstehen? Zur Verdeutlichung ist in Abbildung 4.15 das dritte und vierte Bild der Testreihe abgebildet. Beide Bilder gehören der gleichen Rauschstufe an. Bei Betrachten von Abbildung 4.15 kann festgestellt werden, dass beim linken Bild die unteren Kante des inneren Bereichs deutlich verwaschener ist als beim rechten Bild. Dies führt, anhand der geglätteten Bilder (Abbildung 4.16) gut zu erkennen, zu einer Brückenbildung. Der innere und äußere Bereich werden hierüber verbunden.
Innerhalb des Übergangsbereiches ist die Segmentationsqualität, und damit auch die

[27] Abschnitt 4.4



(a) (b)

Abbildung 4.15: zwei Modell-A Testbilder, Grauwertdifferenz 12, Rauschstufe 64



(a) (b)

Abbildung 4.16: Ergebnis einer Gap-Glättung der Testbilder aus Abbildung 4.15, Filtereineinstellung a

Wahrscheinlichkeit, ob eine Trennung der Bildbereiche des Testbildes stattfindet, sehr stark von den Grauwertdifferenzen abhängig. Betrachtet sei hier, wie im obigen Beispiel, die Empfindlichkeit bei Parametereinstellung a (Abbildung 4.12), nun nicht bei einer Grauwertdifferenz von 12, sondern von 13. Grauwertdifferenz 13 befindet sich im oberen

Übergangsbereich. Die ersten fünf Testbilder der Testreihe haben die SA_{EQ}-Werte 0,98, 0,97, 0,98, 0,98 und 0,98. Wie Abbildung 4.17 ersichtlich werden alle inneren Vierecke erkannt.

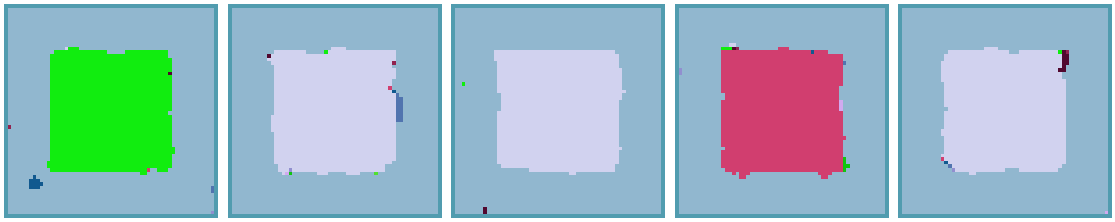


Abbildung 4.17: Segmentationsergebnisse der ersten fünf Testbilder einer Testreihe, welche im Rahmen der Bestimmung der Filtercharakteristik bezüglich Filtereinstellung a bei einer Grauwertdiffrenz von 13 und einer Rauschstufe von 64, durchgeführt wurde.

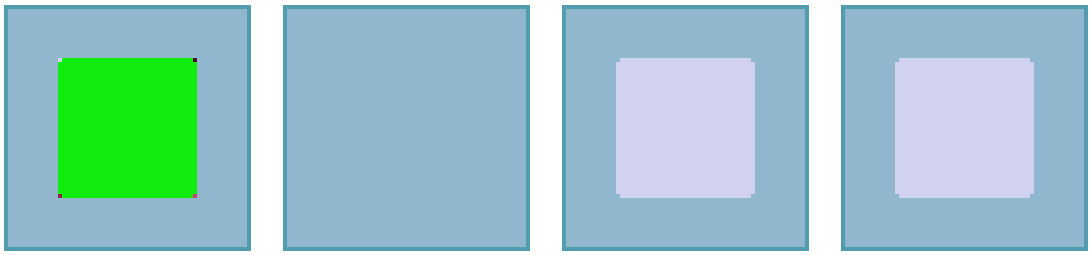
Maximaler Bereich
Dieser Bereich schließt sich dem Übergangsbereich an. In diesem Bereich wird maximale Segmentationsqualität erreicht, welche natürlich von der Rauschstufe und der Filtereinstellungen abhängig ist. Eine interessante Erkenntnis ist, dass die Segmentationsqualität ab einer gewissen Grauwertdifferenz einen maximalen Wert erreicht und nicht mehr abfällt. Je höher die Rauschstufe, desto höher ist die Grauwertdifferenz, die hierfür nötig ist. Bei Parametereinstellung b (Abbildung 4.13) beginnt der maximale Bereich bei Rauschstufe 256 ungefähr bei 25 und nimmt einen SA_{EQ}-Wert von ungefähr 0,98 an, wogegen er bei Rauschstufe 8 bei ungefähr 16 beginnt und einen SA_{EQ}-Wert nahe 0,99 annimmt.

Es sei am Ende dieses Abschnittes darauf hingewiesen, dass die Filtercharakteristik der Gap-Segmentation anhand zweier Parametereinstellungen besprochen wurde. Der sehr klar abgegrenzte Verlauf, der eine klare Unterteilung in die drei vorgestellten Bereiche möglich macht, ist charakteristisch für die Gap-Segmentation. Wie in Abschnitt 4.6 gezeigt, haben die meisten Segmentationsmethoden im Großen und Ganzen ähnliche Filtercharakteristiken. In vielen Fällen können die einzelnen Bereiche jedoch nicht mehr in dieser Deutlichkeit unterteilt werden und haben eine komplexere Form. Der Übergangsbereich ist bei einigen beispielsweise ziemlich breit und hat keinen gradlinig steigenden Verlauf. Oft wird auch nur in einem kleinen Grauwertbereich maximale Filterqualität erreicht. Die Empfindlichkeit ist außerdem nicht nur von der Rauschstufe und der verwendeten Segmentationsmethode abhängig, sondern auch im starken Maße von der Kantenschärfe[28]. Auf die Filtercharakteristik der Gap-Segmentationsmethode bezüglich Modell B und C wird in Abschnitt 4.5.8 genauer eingegangen. Im folgenden Abschnitt wird jedoch zuerst auf das sonderbare Verhalten bei Rauschstufe 0 eingegangen.

[28] hier: Modell A,B und C

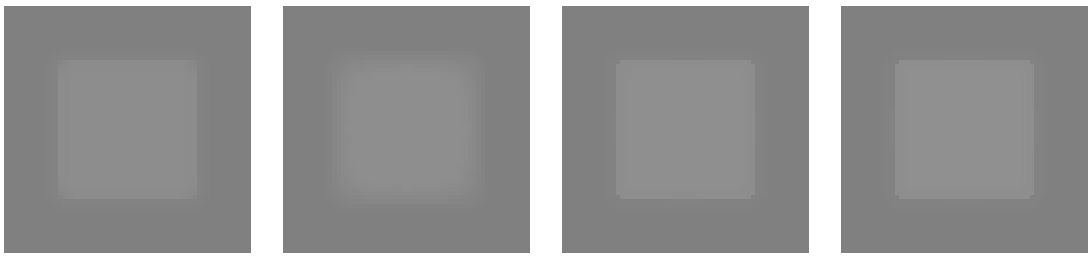
4.5.6 Das Noise 0 Phänomen

Eine Besonderheit des Gap-Filters tritt bei Rauschstufe 0 (kein Rauschen) auf. Wie in Abbildung 4.13 zu erkennen, treten bei einzelnen isolierten Grauwertdifferenzstufen anormale Zustände auf. Ansonsten haben die Kurven die im vorherigen Abschnitt beschriebene Charakteristik. Die erste Anomalie findet sich bei einer Grauwertdifferenz von 14. In Abbildung 4.18 sind die Segmentationsergebnisse der Grauwertdifferenzen 13, 14, 15 und 16 dargestellt. Bei den Grauwertdifferenzen 13, 15 und 16 wird der innere Bereich sehr gut erkannt, bei 14 allerdings überhaupt nicht. Der Grund hierfür ist, dass bei Grauwertdifferenz 14 im Gegensatz zu 13, 15, 16 über die Kanten des inneren Vierecks hinweg geglättet wurde (Abbildung 4.19).



(a) Filtereinstellung b, Modell A13 (b) Filtereinstellung b, Modell-A14 (c) Filtereinstellung b, Modell-A15 (d) Filtereinstellung b, Modell-A16

Abbildung 4.18: Segmentationsergebnisse von Modell-A Testbilder der Rauschstufe 0, die dazugehörige Filtercharakteristik findet sich in Abbildung 4.13

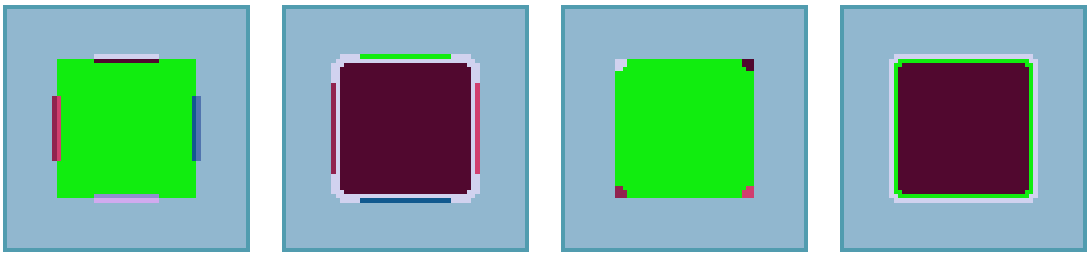


(a) Filtereinstellung b, Modell-A13 (b) Filtereinstellung b, Modell-A14 (c) Filtereinstellung b, Modell-A15 (d) Filtereinstellung b, Modell-A16

Abbildung 4.19: Geglättete Modell-A Testbilder der Rauschstufe 0, die dazugehörige Segmentationsergebnisse sind in Abbildung 4.18 dargestellt

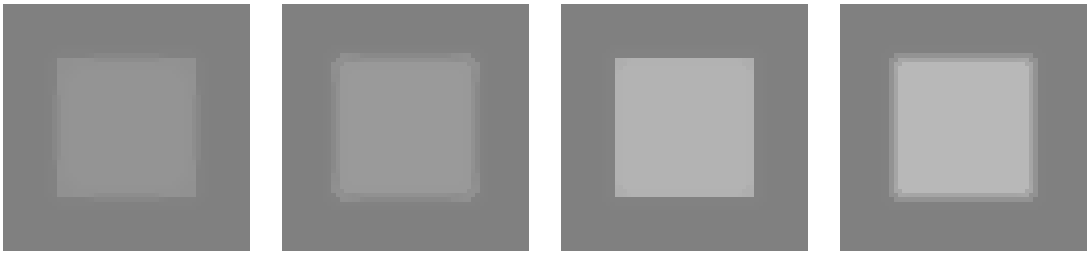
Um dies besser zu verstehen, ist es hilfreich vier weitere Singularitäten zu untersuchen. Deren Segmentationen sind in Abbildung 4.20 dargestellt. Die dazugehörigen geglätteten Bilder sind in Abbildung 4.21 zu finden. Hier ist zu erkennen, dass sich an den Rändern der zwei Bildbereiche Zwischenzustände bilden. Diese bilden sich auch bei der am Anfang

des Abschnitts gezeigten Singularität[29]. Die Graustufendifferenz zwischen den Bildbereichen beträgt in diesem Fall 14. Diese Differenz ist aber so gering, dass es anstatt zur Bildung von Übergangszuständen zu einer Brückenbildung führt.



(a) Filtereinstellung a, Modell-A20 (b) Filtereinstellung a, Modell-A26 (c) Filtereinstellung b, Modell-A51 (d) Filtereinstellung b, Modell-A56

Abbildung 4.20: Segmentationsergebnisse von Modell-A Testbildern der Rauschstufe 0, die dazugehörige Filtercharakteristiken findet sich in Abbildung 4.12 bzw. 4.13



(a) Filtereinstellung a, Modell-A20 (b) Filtereinstellung a, Modell-A26 (c) Filtereinstellung b, Modell-A51 (d) Filtereinstellung b, Modell-A56

Abbildung 4.21: Geglättete Modell-A Testbilder der Rauschstufe 0, die dazugehörige Segmentationsergebnisse sind in Abbildung 4.20 dargestellt

Um besser verstehen zu können, wie solche Zwischenstufen zustande kommen, müsste der Algorithmus der Gap-Segmentation genauer untersucht werden, was in dieser Arbeit jedoch nicht berücksichtigt wurde, da dieser Fall in der Praxis keine Rolle spielt. Denn interessanterweise verschwindet dieser Effekt schon bei der geringsten Rauschstufe, welche untersucht wurde.

4.5.7 Signalrauschabstand

Bis jetzt wurden alle Untersuchungen bezüglich absoluter Rauschstufen durchgeführt. Wird stattdessen, bei der Filtercharakteristik einer Segmentationsmethode, der Signal-

[29] zweites Bild von links, Abbildung 4.18 beziehungsweise 4.19

rauschabstand betrachtet, führt dies zu einem in Abbildung 4.22 dargestellten Ergebnis. Der Signalrauschabstand ist hier folgendermaßen definiert:

$$\text{Signalrauschabstand} := \frac{\text{Grauwertdifferenz innerer-äußerer Bereich}}{\text{Standardabweichung der Rauschstufe}^{30}}$$

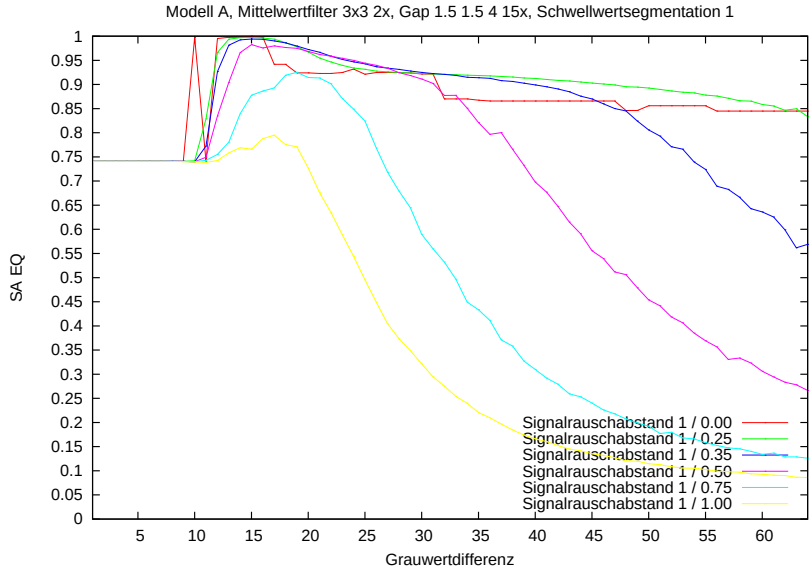


Abbildung 4.22: Filtercharakteristik bezüglich des Signalrauschabstand, Modell-A, Filtereinstellung b

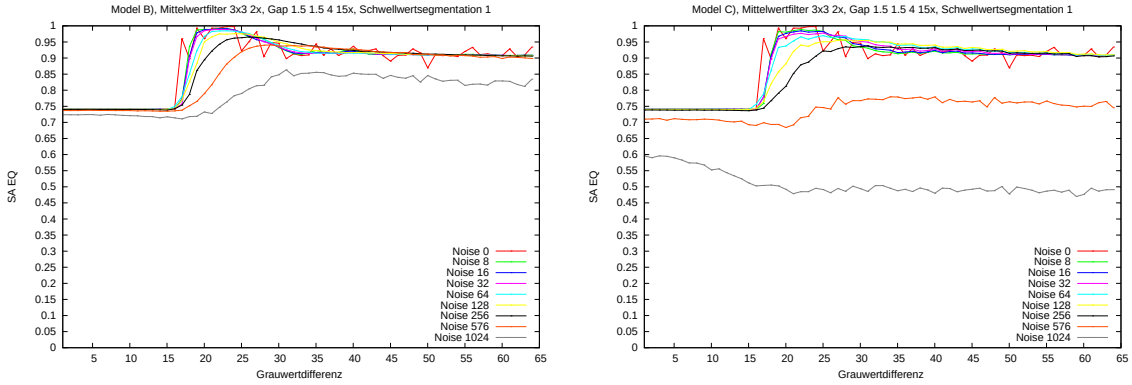
Im Gegensatz zur einer Filtercharakteristik, welche sich auf absolute Rauschstufen bezieht, hat diese keinen einfach zu interpretierenden Verlauf mehr. Der nicht sensible Bereich und der Übergangsbereich sind hierauf zwar ebenfalls zu finden, an den maximalen Bereich schließt sich jedoch ein wieder abfallender Bereich an. Die Darstellung der Filtercharakteristik bezüglich des Signal-Rauschabstands wird daher in dieser Arbeit nicht weiter verwendet.

4.5.8 Weiche Kanten

Die Charakteristik einer Segmentationsmethode wurde bis jetzt nur für das Modell-A Testszenario ermittelt. Modell-A ist, wie in Abschnitt 4.4 dargestellt, ein Modell mit harten Kanten. In realen Anwendungen treten jedoch nicht nur harte Kanten, sondern in vielen Fällen auch weiche Grauwertübergänge zwischen verschiedenen Bildbereichen auf. Weiche Grauwertübergänge zwischen Bildbereichen werden hier auch als weiche Kanten

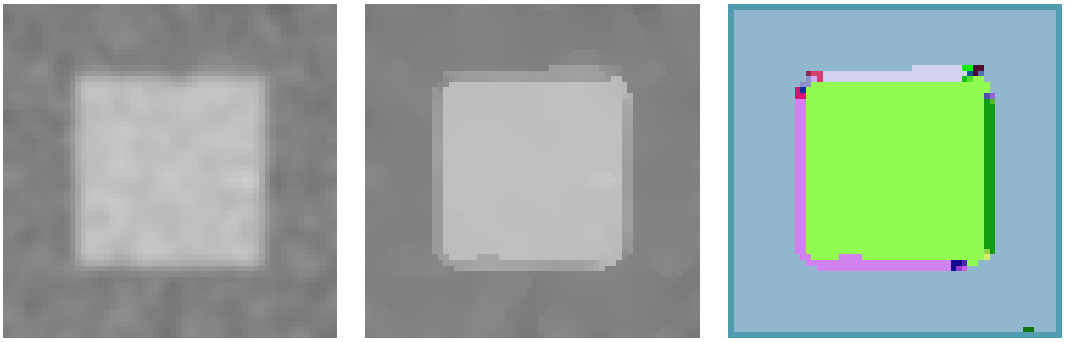
[30] bis auf diesen Abschnitt wird die Varianz der Rauschstufe angegeben (Abschnitt 4.4), Bemerkung: Standartabweichung := $\sqrt{\text{Varianz}}$

bezeichnet. Anhand Modell-B und Modell-C wurde der Einfluss weicher Kanten auf das Segmentationsergebnis untersucht.
Der Unterschied der Filtercharakteristik zu Modell-A bei Filtereinstellung b, wie sie in Abbildung 4.13 dargestellt wurde, ist in Abbildung 4.23 deutlich zu erkennen. Sowohl beim Modell-B als auch Modell-C können sehr hohe SA_{EQ} Ergebnisse nahe 1,0 nur noch in einem kleinen Bereich erzielt werden. An diesen, jetzt sehr schmalen maximalen Bereich, schließt sich der sogenannte abfallende Bereich an, der sich bei einem SA_{EQ} Wert um 0,9 herum einpendelt.



(a) Modell-B (b) Modell-C

Abbildung 4.23: Filtercharakteristiken bei Filtereinstellung b



(a) Orginal (b) Geglättet (c) Segmentiert

Abbildung 4.24: Die verschiedenen Verarbeitungsstufen eines Modell-B Testbildes bei denen der Randbereich nicht wie gewünscht separierter wird, sondern sich Zwischenstufen bilden

Dass in weiten Teilen kein sehr gutes Ergebnis erzielt werden kann, ist darauf zurückzuführen, dass der Randbereich, wie in Abbildung 4.24 dargestellt, nicht mehr wie gewünscht separiert wird, sondern sich Zwischenstufen bilden. Das Grund hierfür ist, dass

nach der zweifach durchgeführten Vorfiltrierung[31] mit dem 3×3-Mittelwertfilter die weichen Kanten noch weiter verschmiert werden. Der anschließende Gap-Glättungsprozess[32] kann den Ursprungsverlauf der Kante dann nicht mehr vollständig wiederherstellen.
Im folgenden Abschnitt wird vorgestellt, wie hier eine deutliche Verbesserung erzielt werden kann.

4.5.9 Kombinationsglättung - Finale Parametereinstellungen

Die Segmentationsqualität kann, insbesondere bei den Modellen B und C, deutlich erhöht werden, wenn die oben durchgeführte Filterprozedur, bestehend aus 2-maliger Vorfiltrierung und 15-maliger Hauptfiltrierung, mehrmals hintereinander ausgeführt wird.
Es hat sich allerdings als Vorteil herausgestellt, bei der erstmaligen Durchführung auf die Vorfiltrierung zu verzichten. Dies führt zu deutlichen Verbesserungen bei Modell-B, aber auch bei Modell-C. Wird der Gap-Filterprozess 4-mal hintereinander ausgeführt, aber beim ersten Lauf auf die Vorfiltrierung verzichtet, wird dies hier als Combo 3.5x Gap-Glättung bezeichnet. Analog wird die 2-malige Ausführung des Gap-Filterprozesses, mit Verzicht auf die Vorfiltrierung beim ersten Lauf, als Combo 1.5x Gap-Glättung bezeichnet. Der gesamte Segmentationsprozess wird entsprechend als Combo 1.5x Gap-Segmentation (Filtereinstellung b) bezeichnet.
In Abbildung 4.25 ist die Filtercharakteristik einer Combo 3.5x Gap-Segmentation (Filtereinstellung b) bezüglich Modell-B und Modell-C dargestellt. Betrachtet man Abbildung 4.23, kann festgestellt werden, dass eine deutliche Verbesserungen erzielt werden konnte.

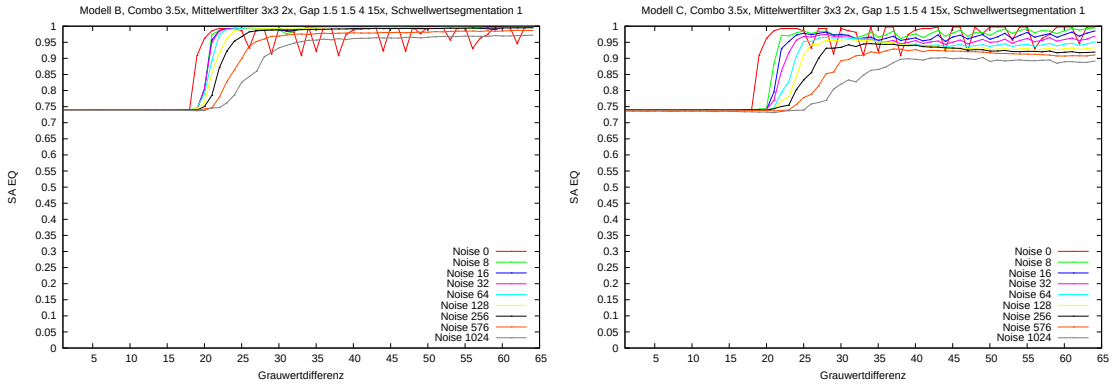


Abbildung 4.25: Combo 3.5x Gap-Segmentation, Filtereinstellung b, links Modell-B, rechts Modell-C

Bei Modell-B und Modell-C wird nun bei niedrigeren und moderaten Rauschstufen durchweg eine gute bis sehr gute Qualität erreicht[33]. Beeindruckend ist, dass selbst im Falle von

[31] entspricht Filterstufe 1
[32] entspricht Filterstufe 2
[33] entspricht einer $SA_{EQ} \geq 0,95$

Modell-C bei hohen Rauschstufen[34] zufriedenstellende Ergebnisse erzielt werden konnten. Das Testergebnis der Filtereinstellung a bezüglich Modell C findet sich in Anhang Abbildung 7.1. Bei niedrigen und moderaten Rauschstufen ist, wie hier festgestellt werden konnte, durchwegs eine Combo 1.5x Glättung ausreichend. Folgende Filtereinstellungen haben sich als günstig erwiesen:

1. **Moderate Rauschstufen (bis 128):** Combo 1.5x, 2 x Mittelwertfilter 3x3, 15 x Gap-Glättung 1.5 1.5 2, Segmentationsschwellwert 1

2. **Hohe Rauschstufen (bis 1024):** Combo 3.5x, 2 x Mittelwertfilter 3x3, 15 x Gap-Glättung 1.5 1.5 4, Segmentationsschwellwert 1

Der Nachteil der Combo-Glättung ist jedoch, dass eine hohe Anzahl an Filterläufen benötigt wird. Bei einer Combo 3.5x Filtrierung werden $3 * 2$ Vorfilterläufe und $4 * 15$ Hauptfilterläufe benötigt, also 66 Filterläufe. Bei einer Combo 1.5x Filterung werden immerhin 32 Filterläufe benötigt. Da mit einer FPGA-Implementation, bei einer Szene mit 1000*1000 Bildpunkten jedoch 100 Filterläufe pro Sekunde durchgeführt werden können[35], kann bei kleineren Szenen durchaus eine Bearbeitungszeit von unter einer Sekunde erreicht werden. Bei extrem großen Szenen mit beispielsweise 10000*10000 Bildpunkten kann bei dieser Konfiguration immerhin noch ein Filterlauf pro Sekunde durchgeführt werden, was eine Bearbeitung unter zwei Minuten ermöglicht. Die Anforderungen an die Bearbeitungszeit, welche in Abschnitt 2.3 gestellt wurden, können daher eingehalten werden.

4.6 Vergleich mit verwandten Verfahren

In den bisherigen Abschnitten dieses Kapitels wurden Parametereinstellungen für das Gap-Segmentationsverfahren ermittelt und die qualitative Leistungsfähigkeit anhand der in Abschnitt 4.5.5 eingeführten Filtercharakteristik ermittelt. Im Folgenden wird ein Vergleich mit drei verwandten Verfahren durchgeführt. Zuerst wird begründet, warum diese Verfahren herangezogen wurden:

Jahn-Segmentationsverfahren:
Das Gap-Segmentationsverfahren beruht auf einer von Jahn [37] entwickelten Segmentationsmethode[36]. Die Gap-Segmentation wurde im Hinblick einer FPGA-Implementation entworfen. Allerdings sollte sich eine zu der Methode von Jahn vergleichbare Segmentationsqualität erzielen lassen. Dies soll in diesem Abschnitt überprüft werden.

Schwellenwertsegmentation:
Das Gap-Segmentationsverfahren beruht, wie im Abschnitt 3.1 beschrieben, auf dem

[34] entspricht einer Varianz von 576 und 1024
[35] bei neun Gap-Glättungsverarbeitungseinheiten und einer Taktfrequenz von 100 MHz (Abschnitt 3.4.5)
[36] Abschnitt 4.6.1

Gap-Glättungsverfahren (Filterstufe 2). Es kann als dynamisches Schwellenwertverfahren bezeichnet werden. Die 3×3-Umgebung jedes Bildpunktes wird analysiert und ein Schwellenwert abgeleitet[37]. Zur eigentlichen Glättung werden nun nur die Punkte der Umgebung herangezogen, deren Differenz vom Zentralpunkt kleiner als der Schwellenwert ist.
Es stellt sich die Frage, welche Verbesserung durch die Verwendung eines dynamisch ermittelten Schwellwerts erzielt wird. Lohnt sich dieser Aufwand überhaupt? Es wird daher untersucht, was passiert, wenn anstatt des aufwendig dynamisch ermittelten Schwellenwerts ein konstanter Wert verwendet wird.

Bilateral-Median Segmentation:
Der Segmentationsprozess dieser Methode entspricht dem der Gap-Segmentation, abgesehen davon, dass die Filterstufe 1 (Mittelwertfilter) durch einen Bilateralfilter und die Filterstufe 2 (Gap-Glättung) durch einen Medianfilter ersetzt wurde.
Es soll untersucht werden, wie sich die Gap-Glättung im Vergleich zu einer alternativen kantenerhaltenden Glättungsmethode verhält. Diese Untersuchung wurde in erster Linie im Hinblick auf eine Softwareimplementation durchgeführt. Weder das bei der Jahn-Segmentation verwendete Glättungsverfahren, noch das Gap-Glättungsverfahren sind für eine Softwareimplementation wirklich geeignet. Es soll untersucht werden, inwieweit diese durch ein alternatives Verfahren, welches für eine Softwareimplementation besser geeignet ist, ersetzt werden können.

Es ist wichtig zu erwähnen, dass die drei Verfahren und das in dieser Arbeit entwickelte Verfahren eine Segmentation nach dem Merkmal des mittleren Grauwerts durchführen. Die Qualitätsuntersuchung und damit auch der Vergleich wird nach diesem Merkmal durchgeführt[38]. Es wurde ebenfalls versucht, für jedes Verfahren optimale Filtereinstellungen zu ermitteln. Bei der Parameterbestimmung wurde ähnlich wie bei der Gap-Segmentation vorgegangen.
Zuerst werden die drei Verfahren vorgestellt. Anschließend folgt in Abschnitt 4.6.4 ein Vergleich mit dem Gap-Segmentationsverfahren. Für eine genau Beschreibung der in diesem Abschnitt verwendeten (mathematischen) Bezeichnungen sei auf Abschnitt 4.2.1 und 4.2.2 verwiesen.

4.6.1 Jahn-Segmentation

Auf das auf Jahn [37] beruhende und in der Arbeit von Halle [38] verwendete Segmentationsverfahren, welches als Grundlage für die Gap-Segmentation diente, soll im Folgenden eingegangen werden. Vom Aufbau her entspricht das Jahn-Verfahren der Gap-Segmentationsmethode. Es wird zuerst ebenfalls eine kantenerhaltende Glättung durchgeführt (Jahn-Glättung). Anschließend wird der Segmentationsschritt ausgeführt (Jahn-Segmentation)[39].

[37] Abschnitt 3.2.2
[38] Abschnitt 4.4
[39] beide Filter beruhen auf Fuzzy-Logik

Die Jahn-Glättung ist, ein auf einem 3×3-Bildfenster basierender, dynamisch gewichteter Filter,

$$Jahn^l(\mu|_B) := \sum_{i=1}^{9} \lambda_i(\mu|_B)^l * \mu(P_i)$$

wobei

$$\lambda_i(\mu|B)^l := \begin{cases} \frac{\gamma_{t_B}(|\mu(P_i)-\mu(P_Z)|)}{\sum_{i=j}^{9}\gamma_{t_B}(|\mu(P_j)-\mu(P_Z)|)} * \alpha(l) & \text{falls } P_i \neq P_Z \\ \frac{\alpha(l)}{\sum_{j=1}^{9}\gamma_{t_B}(|\mu(P_j)-\mu(P_Z)|)} + (1-\alpha(l)) & \text{sonst} \end{cases}$$

und

$$\gamma_t(x) := \frac{t^2}{t^2+x^2} \text{ und } \gamma_0(x) := \begin{cases} 1 & \text{falls } x = 0 \\ 0 & \text{sonst} \end{cases}$$

den sogenannten Zusammenhangsgrad darstellt. Außerdem ist

$$t_B := \overline{min}^3_B(|\mu(P_i) - \mu(P_Z)|)$$

der Durchschnitt der drei kleinsten Werte von $|\mu(P_i) - \mu(P_Z)|,\ P_i \in B$.
Mit B wird das 3×3-Filterfenster[40], mit $P_Z := P_5$ der zentrale Bildpunkt des Fensters bezeichnet. Das Eingangsbild wird wieder durch seine Bildfunktion μ dargestellt. Eine genaue Definition der hier verwendeten Begriffe ist in Abschnitt 4.2.1 und 4.2.2 zu finden. Die Filtrierung wird mehrfach durchgeführt, wobei die Variable l die Nummer des Laufes darstellt. Zur Stabilisierung des Filters wurde der Konvergenzfaktor $\alpha(l) = 4 * (0.95)^l$ verwendet. Folgende Eigenschaften wurden ermittelt:

1. $\gamma_t(0) = 1$
2. $\lim_{t\to\infty} \gamma_t(x) = 1$
3. $\sum_{i=1}^{9} \lambda_i(\mu|B)^l = \alpha(l) + (1-\alpha(l)) = 1$
4. $\lim_{l\to\infty} \alpha(l) = 0$
5. $\lim_{l\to\infty} \lambda_i(\mu|B)^l = \begin{cases} 0 & \text{falls } P_i \neq P_Z \\ 1 & \text{sonst} \end{cases}$

Es sei bemerkt, dass bei großem l laut Punkt 5 keine weitere Glättung mehr durchgeführt wird. Für Näheres sei auf die Arbeit von Halle und Jahn verwiesen. Halle hat angedeutet, dass er bei stark verrauschten Bildern anstelle des t Wertes eine Mittlung aller t Werte in einem 11×11 Fenster verwendet hat. Dies wurde in seiner Arbeit jedoch nur angerissen und deswegen hier auch nicht durchgeführt. Allerdings wurde in Halles Arbeit keine Vorfiltrierung durchgeführt, mit welcher bei hohen Rauschstufen eine deutlich bessere Qualität erreicht werden kann.

[40] Abbildung 3.10

Beim nachfolgenden Segmentationsschritt werden zwei benachbarte Punkte P_i, P_j verbunden, falls ihr Zusammenhangsgrad

$$\gamma_{t_{B(P_i),\epsilon}}(|\mu(P_j) - \mu(P_i)|) \geq \delta$$

und

$$\gamma_{t_{B(P_j),\epsilon}}(|\mu(P_i) - \mu(P_j)|) \geq \delta$$

größer als eine vorgegebene Schwelle δ ist. Es sei bemerkt, dass $B(P)$ der Bildblock mit Mittelpunkt P ist und

$$\gamma_{t_B,\epsilon}(x) := \gamma_{max\{t_B,\epsilon\}}(x)$$

ist. Der Parameter ϵ sorgt dafür, dass bei kleinen Grauwertdifferenzen keine Trennung durchgeführt wird. Wie bei Halle wurde $\delta = 0,5$ und $\epsilon = 5$ gesetzt. Es wurde eine mit der Gap-Segmentation vergleichbare Parameteruntersuchung durchgeführt. Folgende Filtereinstellungen wurden ermittelt:

1. **Moderate Rauschstufen (bis 128)[41]:** 30× Jahnglättung, Jahnsegmentation (wobei $\delta = 0,5$ und $\epsilon = 5$)
2. **Hohe Rauschstufen (bis 1024)[41]:** 2× Mittelwertfilter 3×3, 30× Jahnglättung, Jahnsegmentation (wobei $\delta = 0,5$ und $\epsilon = 5$)

4.6.2 Schwellenwertsegmentation

Die Gap-Segmentation beruht auf dem Gap-Glättungsverfahren. Das Gap-Glättungsverfahren ist ein dynamischer Schwellenwertfilter. Zuerst wird anhand der 3×3-Umgebung eines Bildpunkts ein geeigneter Schwellenwert ermittelt und anschließend auf dieser Umgebung eine Schwellenwertglättung durchgeführt. Im Algorithmus aus Kapitel 3.2.2 wird die Schwellenwertglättung im Schritt 7 durchgeführt. Die Schritte 1-6 werden benötigt um den Schwellwert zu ermitteln. Dieser Vorgang wurde auch als Gap-Detektion bezeichnet. Die Anzahl der benötigten Schritte und deren Analyse macht deutlich, dass, dass die Gap-Detektion ein sehr aufwendiger Vorgang ist. Die Frage, die sich stellt, ist, welche Vorteile die Gap-Detektion bringt und ob es eventuell ausreicht, einfach global einen statischen Schwellwert zu setzen. Daher wurde einfach die Gap-Glättung durch einen statischen Schwellenwertglättungsfilter ersetzt, und untersucht, wie viel Einfluss die Gap-Glättung wirklich hat. Im Folgenden ist der Schwellenwertglättungsfilter in Formeln dargestellt:

$$\text{Schwellenwertglättung}_C(\mu|_B) = \frac{\sum_{i=1}^{9} \lambda_i(\mu|_B) * \mu(P_i)}{\sum_{i=1}^{9} \lambda_i(\mu|_B)}$$

wobei

$$\lambda_i(\mu|_B) = \begin{cases} 1 & \text{falls } \mid \mu(P_i) - \mu(P_Z) \mid \leq C \\ 0 & \text{falls } \mid \mu(P_i) - \mu(P_Z) \mid > C \end{cases}$$

[41] die Angabe entspricht der Varianz der Rauschstufe

und $C \geq 0$ der gewählte Schwellwert ist. B bezeichnet hier wieder das 3×3-Filterfenster und μ die Bildfunktion. Bei diesem Verfahren wird nur über Werte gemittelt, welche vom zentralen Punkt P_Z von B eine Grauwertdifferenz kleiner oder gleich dem Schwellwert C haben. Als Vorfilter (Filterstufe 1) wurde, wie beim Gap-Segmentationsprozess ebenfalls, ein 3×3-Mittelwertfilter verwendet. Es wurde eine Parameteruntersuchung durchgeführt, bei der sich folgende Einstellungen als sinnvoll herausstellten:

1. **Moderate Rauschstufen (bis 128)[42]:** 2 ×-Mittelwertfilter 3×3, 15× Schwellenwertglättung $C = 3$, Segmentationsschwellenwert 1
2. **Hohe Rauschstufen (bis 1024)[42]:** 2 × Mittelwertfilter 3×3, 15× Schwellenwertglättung $C = 6$, Segmentationsschwellenwert 1

Filtereinstellung 1 ist für leichtes bis mittelschweres Rauschen geeignet, wohingegen Filtereinstellung 2 für mittelschweres bis starkes Rauschen optimiert wurde.
Eine weitere Verbesserung der Schwellenwertglättung, wie sie mit der Kombinationsglättung bei der Gap-Glättung erreicht wird, ist leider nicht gelungen. Es findet zwar eine Verbesserung im maximalen Bereich statt, aber auch eine Verschlechterung im abfallenden Bereich. Bei der Kombinationsglättung ist anscheinend die dynamische Steuerung der Gap-Glättung von erheblicher Bedeutung.

4.6.3 Bilateral-Median Segmentation

Bei der Bilateral-Median Segmentation wurde anstatt des Gap-Glättungsverfahrens ein auf einer bilateralen Filterung basiertes Verfahren verwendet. Der Bilateralfilter ist ein gewichteter Glättungsfilter, welcher zur Berechnung der Gewichte $\lambda_i(\mu|_B)$ sowohl die Abstände der Bildpunkte im Ortsraum, als auch im Farbraum heranzieht [48]. Konkret wird als Filterstufe 1 ein Bilateralfilter, als Filterstufe 2 ein Median Filter verwendet. Diese Filter sind in vielen Software-Bibliotheken, wie zum Beispiel *OpenCV* [49], enthalten. Sie eignen sich daher hervorragend für eine Softwareimplementation. Als Filterkern wird bei beiden Filtern ein 3×3 Fenster verwendet. Der Bilateralfilter ist folgendermaßen definiert:

$$Bilateral_{\epsilon,\delta}(\mu_{|B}) = \sum_{i=1}^{9} \lambda_i(\mu_{|B}) * \mu(P_i)$$

wobei

$$\lambda_i(\mu_{|B}) = \frac{c(\epsilon, P_i) * s(\delta, \mu(P_i))}{\sum_{i=1}^{9}(c(\epsilon, P_i) * s(\delta, \mu(P_i))}$$

Die Funktion c ist die Gewichtung der Abstände der Bildpunkte bezüglich der Position im Ortsraum

$$c(\epsilon, P_i) = \exp\left(-\frac{1}{2}(\frac{d(P_i, P_Z)}{\epsilon})^2\right)$$

[42] Angabe entspricht Varianz der Rauschstufe

Die Funktion s ist die Gewichtung der Abstände der Bildpunkte bezüglich des Farbraums

$$s(\delta, \mu(P_i)) = \exp\left(-\frac{1}{2}(\frac{\sigma(\mu(P_i), \mu(P_Z))}{\delta})^2\right)$$

Hierbei sind $\epsilon, \delta > 0$ Gewichtungsparameter. Zur Differenzenbildung im Ortsraum wird die euklidische Abstandsfunktion verwendet

$$d(P_i, P_Z) = \sqrt{[(P_i)_x - (P_Z)_x]^2 + [(P_i)_y - (P_Z)_y]^2}$$

Mit $(P)_x$ bzw. $(P)_y$ werden hierbei die x bzw. y Koordinate des Bildpunktes P bezeichnet. Zur Differenzbestimmung im Farb-/Grauwertraum wird Standartbetrag verwendet

$$\sigma(\mu(P_1), \mu(P_2)) = |\mu(P_1) - \mu(P_2)|$$

B bezeichnet hier wieder das 3×3 Filterfenster und μ die Bildfunktion.
Über den Parameter ϵ wird der Einfluss des Ortsraums, über den Parameter δ der Einfluss des Farbraums gesteuert. Je höher der ϵ bzw. δ Wert gesetzt wird, desto höher wird die Ortsraum- bzw. Farbraumabhängigkeit.
Als Vorfilter wurde ein 3×3-Medianfilter verwendet. Dieser berechnet den Median der Grauwerte eines 3×3 Bildfensters und gibt diesen aus. Es stellte sich bei der Parameteruntersuchung heraus, dass die Kombination dieser beider Filter sehr vorteilhafte Auswirkungen auf die Segmentationsqualität hat[43]. Die Glättung erfolgt daher immer in Kombination, erst wird eine Bilateralfiltrierung und anschließend eine Medianfiltrierung durchgeführt. Diese Kombination wird dann gegebenfalls wiederholt[44].
Durch die Parameteruntersuchung wurden folgende Einstellungen ermittelt:

1. **Moderate Rauschstufen (bis 128)[45]:** Combo 30×, 1 × Bilateralfiler 6 3 1, 1 × Medianfilter, Segmentationschwellenwert 1

2. **Hohe Rauschstufen (bis 1024)[45]:** Combo 30×, 1 × Bilateralfilter 6 3 1, 1 × Medianfilter, Segmentationschwellenwert 2

Es sei bemerkt, dass zwar bekannt ist, dass der Bilateralfilter ein kantenerhaltendes oder zumindest kantenschonendes Verfahren ist. Dem Autor ist aber keine Arbeit bekannt, bei welcher eine nur ansatzweise vergleichbare Untersuchung im Zusammenhang mit einer Bildsegmentation durchgeführt wurde. Auch war dem Autor im vornherein nicht bekannt, dass der Bilateralfilter sehr gut mit einem Median-Filter harmoniert. Diese Erkenntnis konnte erst mit Hilfe des Testverfahrens ermittelt und (objektiv) belegt werden, ebenso wie die Wahl günstiger Filtereinstellungen.

[43] es stellt insbesondere heraus, dass der Medianfilter mit dem Bilateralfilter besser als mit einem Mittelwertfilter harmoniert
[44] Combo 30× entspricht einer 30-malige Wiederholung
[45] Angabe entspricht Varianz der Rauschstufe

4.6.4 Vergleich

In diesem Abschnitt wird ein Vergleich der qualitativen Leistungsfähigkeit der Jahnsegmentation, der Schwellenwertsegmentation, der Bilateral-Median-Segmentation und der Gap-Segmentation durchgeführt. Die Filtercharakteristiken werden bezüglich folgender Szenarien verglichen:

1. **Moderate Rauschstufen (bis 128)[45]:** Modell A, B, C
2. **Hohe Rauschstufen (bis 1024)[45]:** Modell A, B, C

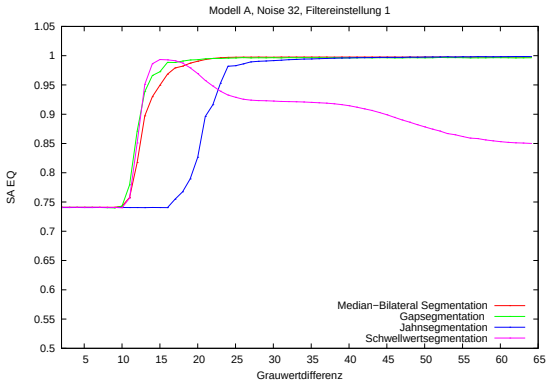
Segmentation bei moderaten Rauschstufen

In diesem Szenario sollen die Segmentationsverfahren bei einer Anwendung auf Bilder mit potentiell moderat auftretendem Bildrauschen untersucht werden. Dazu wurden die Filtereinstellungen, welche in den letzten Abschnitten vorgestellt wurden, herangezogen:

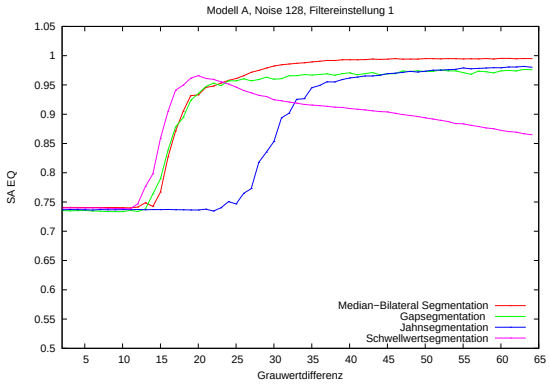
Filtereinstellung 1:

1. **Gap Segmentation:** Combo 1.5×, 2 × Mittelwertfilter 3×3, 15 × Gapglättung 1.5 1.5 2, Segmentationsschwellwert 1
2. **Jahnsegmentation:** 30 × Jahnglättung, Jahnsegmentation 0.5 5
3. **Schwellwertsegmentation:** 2 × Mittelwertfilter 3×3, 15 × Schwellwertglättung 3, Segmentationsschwellwert 1
4. **Bilateral-Median Segmentation:** Combo 30×, 1 × Bilateralfilter 6 3, 1× Medianfilter 3×3, Segmentationsschwellwert 1

In den Abbildungen 4.26, 4.27 und 4.28 sind die Ergebnisse bezüglich der Testmodelle A, B und C und der Rauschstufen 32 und 128 dargestellt.
Zuerst wird auf Modell-A eingegangen. Bei beiden Rauschstufen wird mit der Gap-Segmentation und der Bilateral-Median Segmentation die beste Leistung erreicht. Bei Rauschstufe 128 ist die Bilateral-Median Segmentation sogar einen bisschen besser. Die Filtercharakteristiken dieser beiden Verfahren sind, wie hier festgestellt werden kann, außerdem sehr ähnlich. Mit dem Schwellwertverfahren kann, bei beiden Rauschstufen, nur in einem sehr kleinen Bereich (Grauwertdifferenzen um 15) sehr gute Qualität erreicht werden, welche danach abfällt. Mit der Jahn-Segmentation kann eine hohe Qualität erreicht werden, jedoch wird die Empfindlichkeit der anderen Verfahren damit nicht erreicht.
Im Modell-B Testszenario schneidet die Gap-Segmentation am besten ab, dicht gefolgt von der Jahn-Segmentation. Im maximalen Bereich wird optimale Qualität erreicht. Dies ist ein Anzeichen dafür, dass weiche Kanten, sowohl mit der Gap-Glättung als auch mit der Jahn-Glättung sehr gut geschärft werden können. Die Schwellenwertglättung und die Bilateral-Median Glättung schneiden in diesem Fall weitaus schlechter ab.
Bei Modell-C erzielen die Gap-Segmentation und die Bilateral-Median Segmentation die besten Resultate. Jedoch kann nur noch ein maximal zufriedenstellendes Ergebnis um

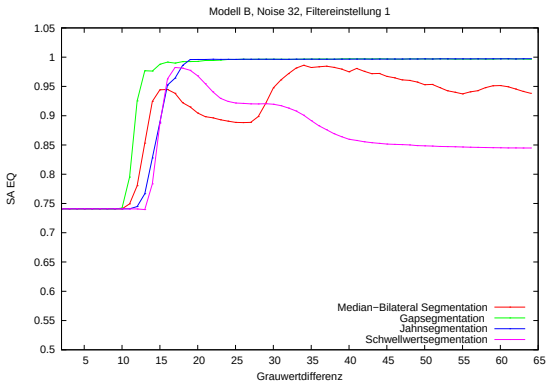


(a) Rauschstufe 32

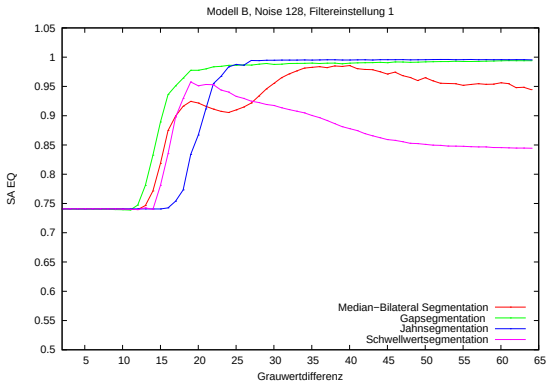


(b) Rauschstufe 128

Abbildung 4.26: Vergleich der Filtercharakteristiken bezüglich Modell-A



(a) Rauschstufe 32



(b) Rauschstufe 128

Abbildung 4.27: Vergleich der Filtercharakteristiken bezüglich Modell-B

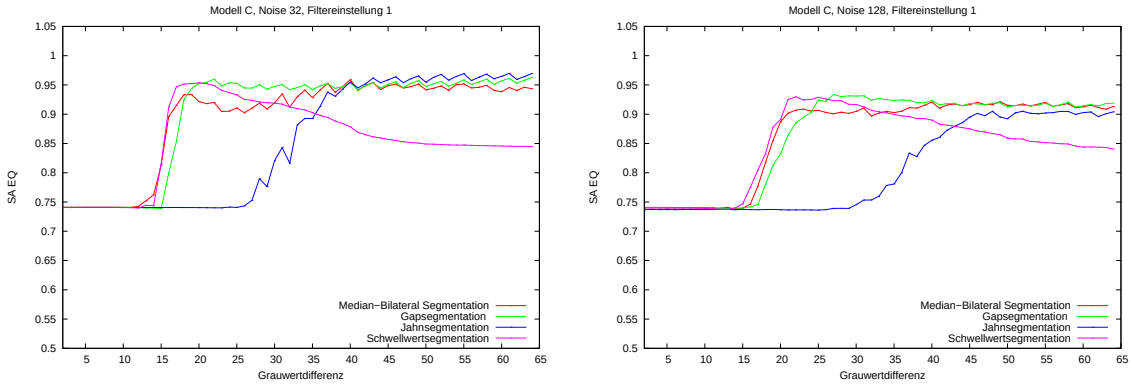
$SA_{EQ} = 0,9$ erreicht werden. Mit der Jahn Segmentation wird eine sichtbar geringere Empfindlichkeit erreicht und mit der Schwellenwertsegmentation kann kein vernünftiges Ergebnis mehr erzielt werden.

Segmentation bei hohen Rauschstufen

Nun wird davon ausgegangen, dass potentiell hohe Rauschstufen (bis 576) vorliegen können. Es wurden daher folgende Filtereinstellungen gewählt:

Filtereinstellung 2:

1. **Gap Segmentation:** Combo 3.5×, 2 × Mittelwertfilter 3×3, 15 × Gapglättung 1.5 1.5 4, Segmentationsschwellwert 1

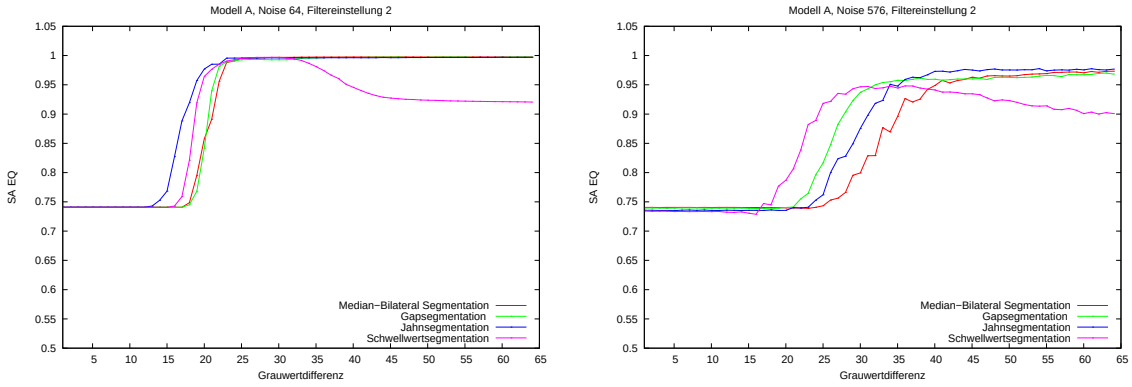


(a) Rauschstufe 32 (b) Rauschstufe 128

Abbildung 4.28: Vergleich der Filtercharakteristiken bezüglich Modell-C

2. **Schwellwertsegmentation:** 2 × Mittelwertfilter 3×3, 15 × Schwellwertglättung 6, Segmentationsschwellwert 1
3. **Jahnsegmentation:** 2 × Mittelwertfilter 3×3, 30 × Jahnglättung, Jahnsegmentation 0.55
4. **Bilateral-Median Segmentation:** Combo 30×, 1 × Bilateralfilter 6 3, 1× Medianfilter 3×3, Segmentationsschwellwert 2

Die Filtercharakteristiken dieser vier Segmentationsverfahren wurden anhand der Testmodelle A, B und C bei einer hohen 576 Rauschstufe und einer niedrigen 64 Rauschstufe verglichen. Die Ergebnisse sind in den Abbildungen 4.29, 4.30 und 4.31 dargestellt.

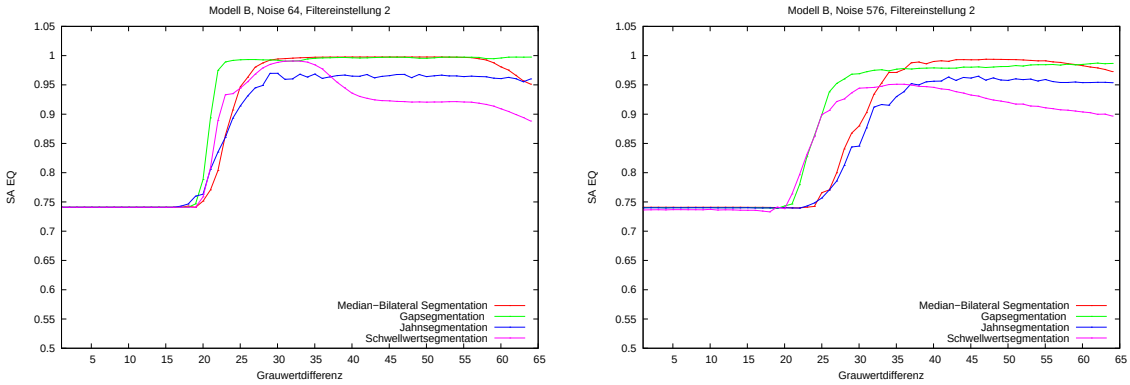


(a) Rauschstufe 64 (b) Rauschstufe 576

Abbildung 4.29: Vergleich der Filtercharakteristiken bezüglich Modell-A

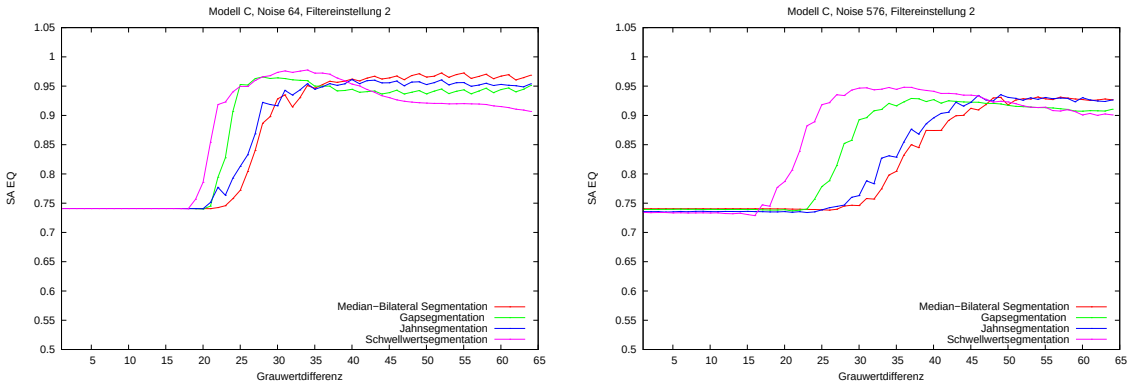
Die Gap-Segmentation, die Jahn-Segmentation und die Bilateral-Median Segmentation weisen bezüglich Modell-A, vergleichbare Filtercharakteristiken auf. Bei Rauschstufe 64,

wird ab einer Grauwertdifferenz von 25 Graustufen ein fast perfektes Ergebnis erreicht. Mit der Schwellenwertsegmentation wird nur in einem kleinen Bereich eine gute Qualität erzielt.
Bei Modell-B ergibt sich ein ähnliches Bild. Nur die Jahn-Segmentation schneidet hier schlechter ab. Die Bilateral-Median und Gap-Segmentation sind auf einem ähnlichen Niveau.
Bei Modell-C ändert sich jedoch das Bild. Hier erreicht die Schwellenwertglättung, gefolgt von der Gap-Glättung die beste Empfindlichkeit, insbesondere im Fall der Rauschstufe 576. Bei Rauschstufe 64 erreicht die Gap-Detektion bei hohen Rauschstufen zwar eine geringfügig bessere Qualität, bei Rauschstufe 576 ist die Schwellenwertglättung aber überraschenderweise erkennbar empfindlicher. Mit den anderen Verfahren wird eine deutlich niedrigere Empfindlichkeit erreicht.



(a) Rauschstufe 64 (b) Rauschstufe 576

Abbildung 4.30: Vergleich der Filtercharakteristiken bezüglich Modell-B



(a) Rauschstufe 64 (b) Rauschstufe 576

Abbildung 4.31: Vergleich der Filtercharakteristiken bezüglich Modell-C

Fazit

Mit der Gap-Segmentation wird in fast allen Testszenarien eine oft bessere Segmentationsqualität als mit der Jahn-Segmentation erreicht. Das Ziel, eine vergleichbare Segmentationsqualität zu erreichen, konnte daher erfüllt werden.
Bei harten Kanten (Modell-A) und weichem Rauschen (Modell-B) wird mit der Gap-Segmentation ein teilweise erheblich besseres Ergebnis als mit der Schwellwertsegmentation erzielt. Dies liegt daran, dass bei den Kanten eine bei Weitem sauberer Trennung erfolgt[46]. Dies belegt, dass die (aufwendige) dynamische Schnellwertberechnung, welche bei der Gap-Glättung durchgeführt wird, einen deutlich positiven Einfluss hat und für ein sehr gutes Ergebnis benötigt wird.
Bei schwierigen Bedingungen (Modell-C) erfolgt auch bei der Gap-Segmentation keine saubere Trennung der Kanten, und es kann keine Verbesserung zum (statischen) Schwellwertverfahren erreicht werden. Bei Rauschstufe 576 ist das Schwellwertverfahren sogar empfindlicher. An dieser Stelle sei aber erwähnt, dass bei der sehr hohen Rauschstufe 1024 mit der Gap-Segmentation, im Gegensatz zur Schwellenwertsegmentation, noch eine zufriedenstellende Segmentation möglich ist[47]. Dies legt nahe, dass zur Behandlung von extrem großen Rauschstufen dynamische Verfahren nötig sind. Es zeigt sich jedoch, dass eine grobe Segmentation auch mit einem einfachen Schwellwertverfahren erzielt werden kann. Der Randbereich wird zwar im Allgemeinen nicht mehr so exakt aufgelöst wie bei den dynamischeren Verfahren, dies wird jedoch von vielen Anwendungen nicht benötigt.
Mit der Bilateral-Median Segmentation konnte in den meisten Fällen ein mit der Gap-Segmentation vergleichbares Ergebnis erzielt werden. Für eine Softwareimplementation stellt sie daher in den meisten Fällen eine brauchbare Alternative dar[48].

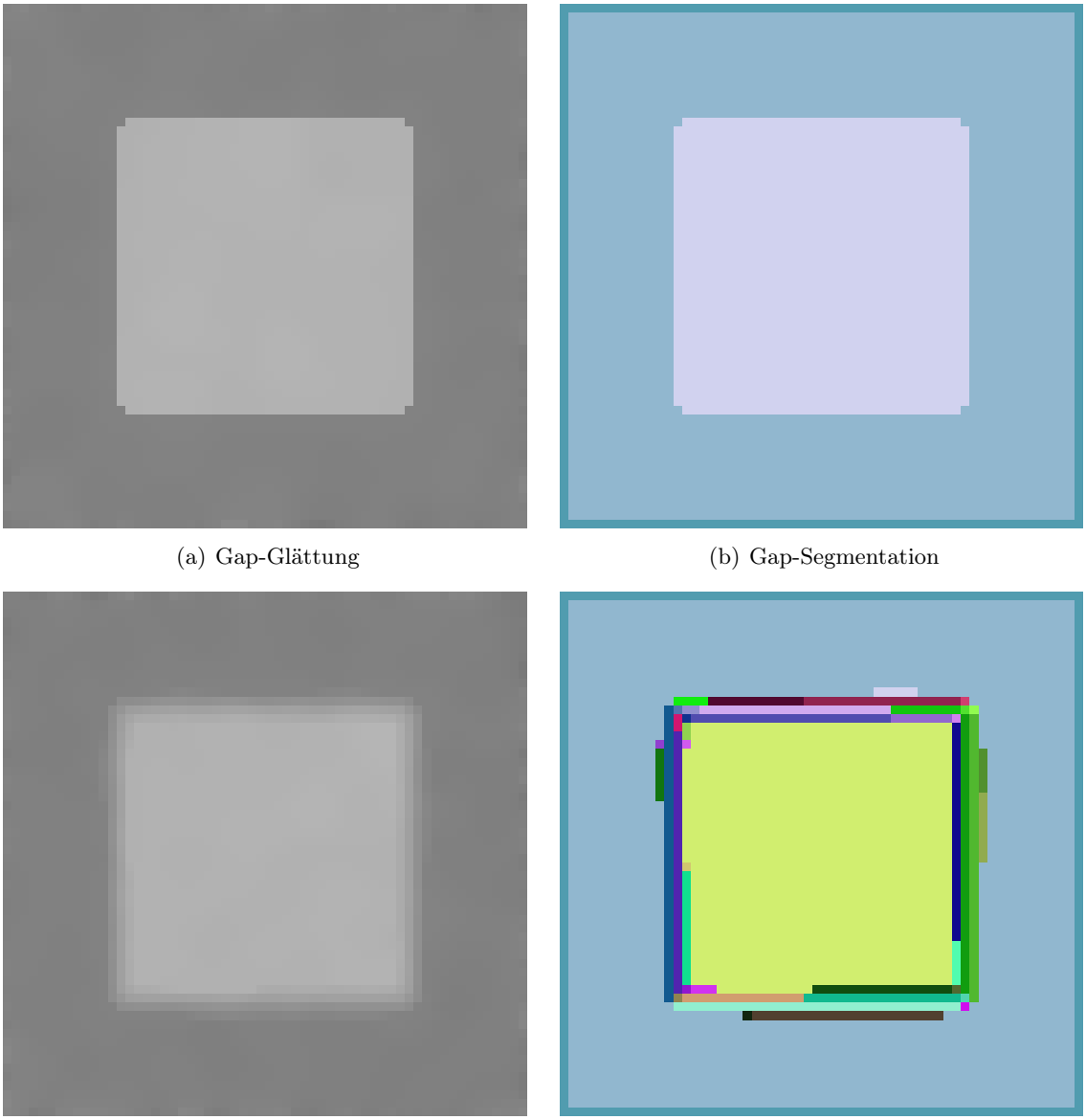
4.7 Reale Beispiele

In diesem Abschnitt soll untersuchen werden, inwieweit sich die an de Testmodellen gewonnen Erkenntnisse auf reale Daten übertragen lassen.

4.7.1 Anwendung der Filtercharakteristik

Für diese Untersuchung wurde eine Landsat7 ETM+ [50] panchromatische Szene (Abbildung 4.33) mit einer räumlichen Auflösung von $15\,m$ und einer spektralen Bandbreite von $0,52 - 0,90\,\mu m$ herangezogen. Die Szene zeigt ein landwirtschaftlich genutztes Gebiet.
Sie wurde mit dem Gap-Segmentationsverfahren segmentiert[49]. Das Segmentationsergebnis ist in Abbildung 4.34 dargestellt. Diese Segmentation wird in diesem Abschnitt mit *Seg1* bezeichnet.

[46] Abbildung 4.32
[47] Auf derartig hohe Rauschstufen wird in dieser Arbeit allerdings nicht eingegangen.
[48] es sei hier jedoch erwähnt dass in Software weitaus modernere Verfahren existieren (Abschnitt 2.4). Inwieweit diese aber überhaupt Leistungsfähiger sind, wurde in dieser Arbeit nicht untersucht.
[49] Filtereinstellung 2, hohe Rauschstufen (Abschnitt 4.5.9)



(a) Gap-Glättung (b) Gap-Segmentation

(c) Schwellwertglättung (d) Schwellwertsegmentation

Abbildung 4.32: Exemplarischer Vergleich des Gap-Verfahrens und des (statischen) Schwellenwertverfahrens anhand eines Modell A50-32 Testbildes, beide Verfahren Filtereinstellung 1

Es wurde anhand von fünf Fallbeispielen untersucht, inwieweit sich das Segmentationsergebnis mithilfe der in Abschnitt 4.5 gewonnenen Filtercharakteristik erklären lässt. Dazu wurden sowohl der Mittelwert, als auch die Varianz der Verteilung der Grauwerte, innerhalb der in der Szene markierten Bereiche ermittelt.

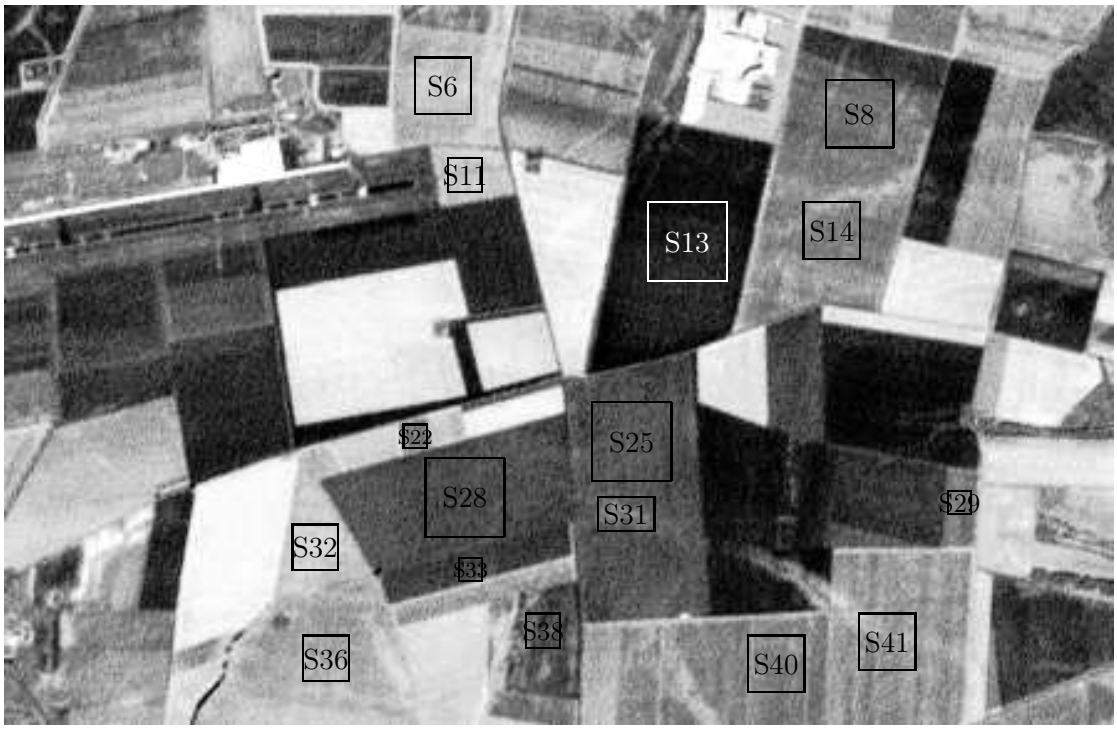


Abbildung 4.33: Landsat 7 ETM+ panchromatische Szene. Landwirtschaftlich genutztes Gebiet in der Nähe von Bitterfeld.

Fallbeispiel 1

Index	Segmente	Mittlerer Grauwertabstand	Maximale Varianz	Zsh.[50]
1.1	S28, S33	$\|76,6 - 62,8\| = 13,8$	$(13,9)^2 = 193,21$ (S28)	ja
1.2	S28, S25	$\|76,6 - 84,4\| = 7,8$	$(14,1)^2 = 198,81$ (S25)	ja
1.3	S25, S38	$\|84,4 - 81,0\| = 3,4$	$(21,0)^2 = 441,00$ (S38)	ja
1.4	S22, S28	$\|119,4 - 76,6\| = 42,8$	$(13,9)^2 = 193,21$ (S28)	nein
1.5	S13, S25	$\|27,9 - 84,4\| = 56,5$	$(15,9)^2 = 252,81$ (S13)	nein
1.6	S25, S31	$\|84,4 - 121,8\| = 37,4$	$(19,7)^2 = 388,09$ (S31)	nein

Eine Trennung benachbarter Grauwertbereiche erfolgt bei der Gap-Segmentation (Filterstellung 2), auch bei sehr guten Bedingungen[51] erst ab einer Grauwertdifferenz von 20. Eine Trennung der Testfälle 1.1-1.3 kann daher überhaupt nicht erfolgen. Ab einer Grauwertdifferenz von 30 erfolgt eine Trennung auch bei schwierigen Bedingungen[52]. Daher ist es nicht verwunderlich, dass in den Testfällen 1.4-1.6 eine Trennung stattfindet.

[50] Zusammenhängend (Zsh) bedeutet hier, dass die zwei Testbereiche SXX bezüglich der Segmentation *Seg1* in einem Segment liegen

[51] Modell-A, (Abbildung 7.2)

[52] Modell-C, Rauschstufen bis 576 (Abbildung 7.3)

Abbildung 4.34: Segmentationsergebnis *Seg1* der Szene aus Abbildung 4.33

Fallbeispiel 2

Index	Segmente	Mittlerer Grauwertabstand	Maximale Varianz	Zsh.
2.1	S40, S41	$\|132,5 - 156,9\| = 24,4$	$(17,6)^2 = 309,76$ (S28)	nein

Da im Fallbeispiel 2.1 die Grauwertdifferenz mit 24,4 zwischen 20 und 30 liegt, reicht die Grauwertdifferenz alleine nicht aus, um die Trennung der Segmente erklären zu können. Diese Situation muss daher genauer betrachtet werden. Würde man Modell-C verwenden, liegt die Grauwertdifferenz 24,4 bei Rauschstufe 256 im unteren Übergangsbereich[53]. Eine Trennung ist daher unwahrscheinlich. Bei Betrachten der Kante zwischen den Bereichen S40, S41 kann aber festgestellt werden, dass diese relativ hart ist und damit eher Modell-A entspricht. In Modell-A wird bei einer Grauwertdifferenz von 25 bei Rauschstufe 256 ein SA_{EQ} Wert von 0,95 erreicht (Abbildung 7.2). Dieser Wert liegt im oberen Übergangsbereich. Die vorliegende Trennung der Bereiche lässt sich daher anhand der Filtercharakteristik erklären.

[53] Abbildung 7.3

Fallbeispiel 3

Index	Segmente	Mittlerer Grauwertabstand	Maximale Varianz	Zsh.
3.1	S6, S11	$\|199,0 - 198,7\| = 0,3$	$(11,8)^2 = 139,24$ (S6)	ja

Nach der Differenz der mittleren Grauwerte müssen die zwei Segmente zusammenhängen. Interessant ist dieses Beispiel aber in der Hinsicht, dass die Segmente S6, S11 durch eine dunkle Linie räumlich getrennt werden. Diese ist für diese Filtereinstellung aber zu schwach ausgeprägt, um zu einer Trennung der Segmente zu führen. Es ist eine berechtigte Frage, wie breit eine Trennlinie in Abhängigkeit von Grauwertdifferenz, Rauschstufe und Kantenschärfe sein müsste, um zwei Grauwertbereiche mit gleichem mittleren Grauwert zu trennen. Diese Fragestellung ist ein guter Anknüpfungspunkt für weiterführende Arbeiten.

Fallbeispiel 4

Index	Segmente	Mittlerer Grauwertabstand	Maximale Varianz	Zsh.
4.1	S8, S14	$\|119,5 - 129,4\| = 9,9$	$(16,5)^2 = 272,25$ (S14)	nein
4.2	S8, S42	$\|119,5 - 151,4\| = 39,9$	$(15,2)^2 = 231,06$ (S 8)	nein
4.3	S14,S42	$\|129,4 - 151,4\| = 22,0$	$(16,5)^2 = 272,25$ (S14)	ja.

Nach der mittleren Grauwertdifferenz dürfte bei Testfall 4.1 keine Trennung der Bereiche S8 und S14 stattfinden. Bei genauerer Betrachtung des Randbereiches fällt jedoch eine linienförmige Störung an der Segmentgrenze auf. Diese wird teilweise auch als eigenes Segment gewertet. Der Testbereich S42 liegt in einem relativ schwach ausgeprägten Teil der Störung. Trotzdem liegt zwischen den Bereichen S42 und S8 eine Grauwertdifferenz von 39,9 vor. Nach der Filtercharakteristik sollte daher eine Trennung stattfinden. Zwischen den Bereiche S42 und S14 tritt jedoch nur eine Grauwertdifferenz von 22,0 auf. Da der Übergang zwischen diesen Bereichen relativ verwaschen ist, dürfte es sinnvoll sein, die Model-C Filtercharakteristik heranzuziehen[54]. Bei dieser liegt die Grauwertdifferenz 22 bei einer 256 Rauschstufe im unteren Übergangsbereich ($\cong SA_{EQ}$ 0,8). Eine Trennung ist daher laut der Modell-C Filtercharakteristik unwahrscheinlich und erklärt den Zusammenhang der Bereiche S14 und S42.

Fallbeispiel 5

Index	Segmente	Mittlerer Grauwertabstand	Maximale Varianz	Zsh.
5.1	S32, S36	$\|211,5 - 165,9\| = 45,6$	$(15,3)^2 = 234,09$ (S36)	ja

Nach den mittleren Grauwerten müssten die Bereiche S32, S36 getrennt sein. Sie sind es jedoch nicht. Das Problem ist, dass der Rand sehr undeutlich ist und beide Segmente ineinander überführt. Der Übergang der zwei Bereiche ist dem ersten Eindruck nach weitaus undeutlicher als jener von Modell-C. Daher ist Modell-C in diesem Fall offensichtlich nicht anwendbar.

[54] Abbildung 7.3

Fazit

Mithilfe der Filtercharakteristik, des mittleren Grauwertabstandes und der Varianz der Grauwerte (Rauschstufe) kann in vielen Fällen bereits recht gut abgeschätzt werden, wann eine Trennung der Bildbereiche auftritt und wann nicht (Fallbeispiel 1). In Grenzfällen ist es sinnvoll die Kantenschärfe der Betrachtung hinzuzufügen (Fallbeispiel 2 und 5). Auch der Einfluss von Störungen lässt sich begründen (Fallbeispiel 4). Es gibt jedoch auch Fälle, bei denen sich das Verhalten nicht eindeutig abschätzen lässt (Fallbeispiel 3). Hier wären weitere Untersuchungen von Nöten, welche im Rahmen dieser Arbeit nicht durchgeführt wurden.

4.7.2 Vergleich verschiedener Segmentationen

Im Abbildung 4.35 ist ein weiteres Segmentationsergebnis der Szene aus Abbildung 4.33 dargestellt. Diese Segmentation wird mit *Seg2* bezeichnet. Es soll im Folgenden anhand *Seg1* und *Seg2* aufgezeigt werden, wie verschiedene Segmentationsergebnisse verglichen werden können.

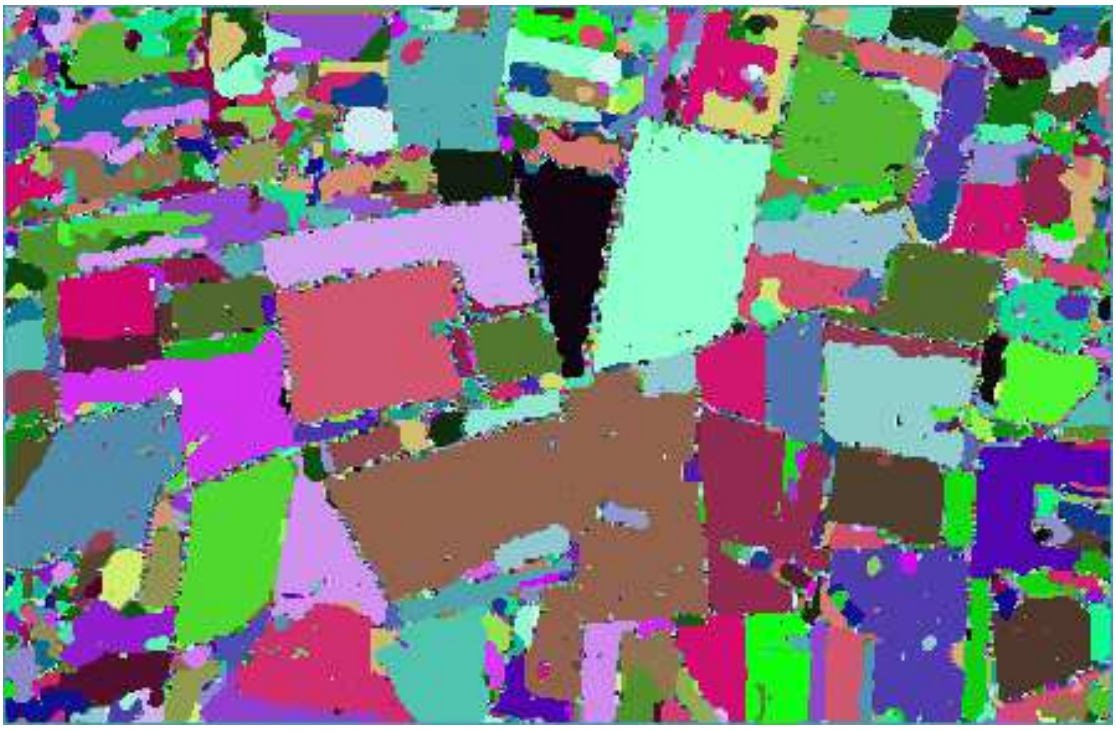


Abbildung 4.35: Segmentationsergebnis (*Seg2*) der Szene aus Abbildung 4.33, Filtereinstellungen: Combo 3.5 2×, Mittelwertfilter 3 × 3, Gap-Glättung 1.5 1.5 2 15×, Schwellwertglättung 1

Bei genauerer Betrachtung fällt auf, dass sich *Seg1* und *Seg2* in einigen Bereichen deutlich

unterscheiden. Dieser Eindruck wird durch einen $SA_{EQ}(Seg1, Seg2)$-Wert von ungefähr $0,75$ bestätigt.
Stellt nun die eine Segmentation eine Unter- oder Übersegmentation der anderen dar? Um dies zu überprüfen können die SA-Werte[55] herangezogen werden:

$$SA(Seg1|Seg2) = 0.63,\ SA(Seg2|Seg1) = 0,89$$

Aus diesen Werten kann abgelesen werden, dass *Seg2* tendenziell eine Verfeinerung von *Seg1* ist[56]. Es kann jedoch nicht von einer reinen Verfeinerung gesprochen werden[57].
Für eine genauere Analyse ist es hilfreich, die im Folgenden definierten lokalen SA-, SA_{EQ}-Werte zu betrachten.

Definition 20 (Lokale SA, SA_{EQ}-Werte) *Sei M eine endliche Menge, $P_1, P_2 \in Part(M)$ zwei Partitionen von M und $x \in M$.*

- *Der lokale $SA(P_2, P_1)$-Wert $SA_x(P_2, P_1)$, an der Stelle x wird definiert als*

$$SA_x(P_2|P_1) := SA(P_2(x)|P_1) \in (0, 1]_{\mathbb{R}}$$

- *Der lokale SA_{EQ}-Wert $SA_{EQ_x}(P_2, P_1)$ an der Stelle x wird definiert als*

$$SA_{EQ,x}(P_2, P_1) := \sqrt{SA_x(P_2|P_1) * SA_x(P_1|P_2)} \in (0, 1]_{\mathbb{R}}$$

$P_1(x) \in P_1$ ist hierbei das eindeutige Element der Partition P_1, welches x enthält. Für $P_2(x)$ gilt dies entsprechend.

Die lokalen SA- und SA_{EQ}-Werte können skaliert und bildlich dargestellt werden.

Bemerkung 8 (SA-, SA_{EQ}-Bilder) *Seien $S_1, S_2 \in Seg(\Omega)$ zwei Segmentationen.*

- *Ein SA-Bild bezüglich S_1 und S_2 ist ein Bild $\mu_{SA}(S1|S2) : \Omega \to [0, 255]_{\mathbb{N}}$,wobei der Grauwert jedes Bildpunktes x folgendermaßen definiert ist*

$$\mu_{SA}(S1|S2)[x] := round(SA_x(S_1|S_2) * 255)$$

- *Ein SA_{EQ}-Bild bezüglich S_1 und S_2 ist ein Bild $\mu_{SA_{EQ}}(S_1, S_2) : \Omega \to [0, 255]_{\mathbb{N}}$, wobei der Grauwert jedes Bildpunktes folgendermaßen definiert ist*

$$\mu_{SA_{EQ}}(S_1, S_2)[x] := round(SA_{EQ,x}(S_2, S_1) * 255)$$

[55] Abschnitt 4.3.2
[56] Abschnitt 4.3.2, Satz 3
[57] Abschnitt 4.3.1, Bemerkung 4

Ein SA_{EQ}-Bild eignet sich sehr gut, um festzustellen, an welchen Stellen sich zwei Segmentationen unterscheiden. In Abbildung 4.35 ist das $SA_{EQ}(Seg1, Seg2)$-Bild dargestellt. In den hellen Bereichen stimmen die Segmentationen größtenteils überein. In den dunkleren Bereichen treten größere Unterschiede auf.

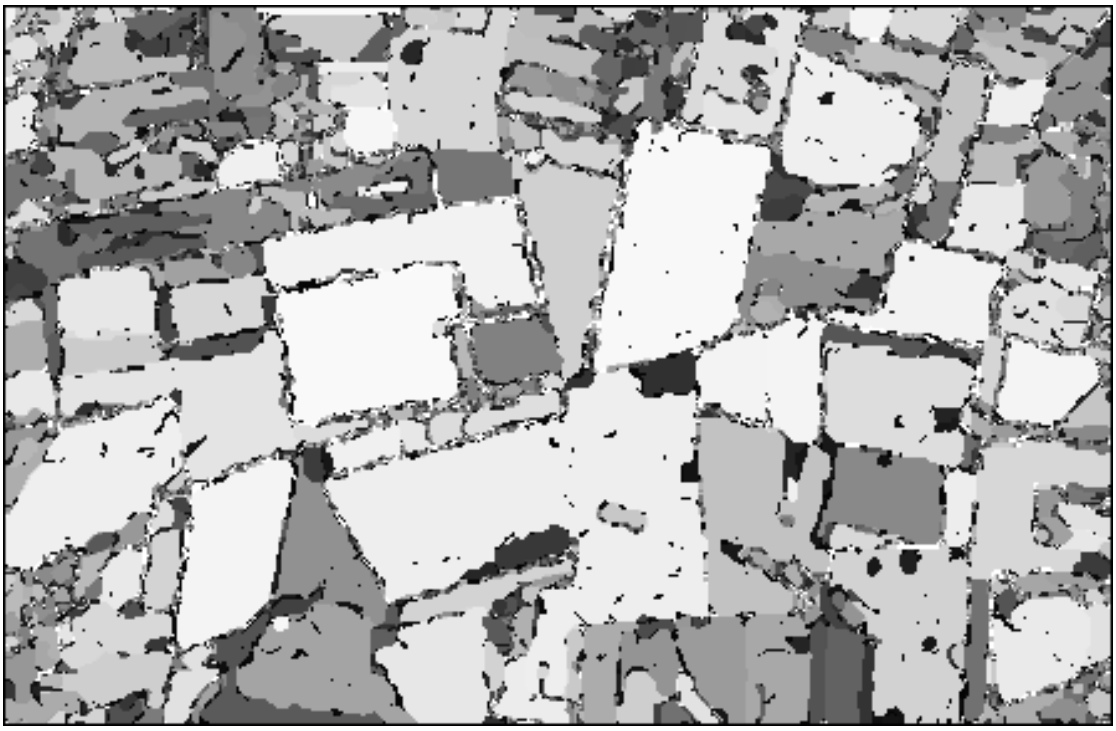


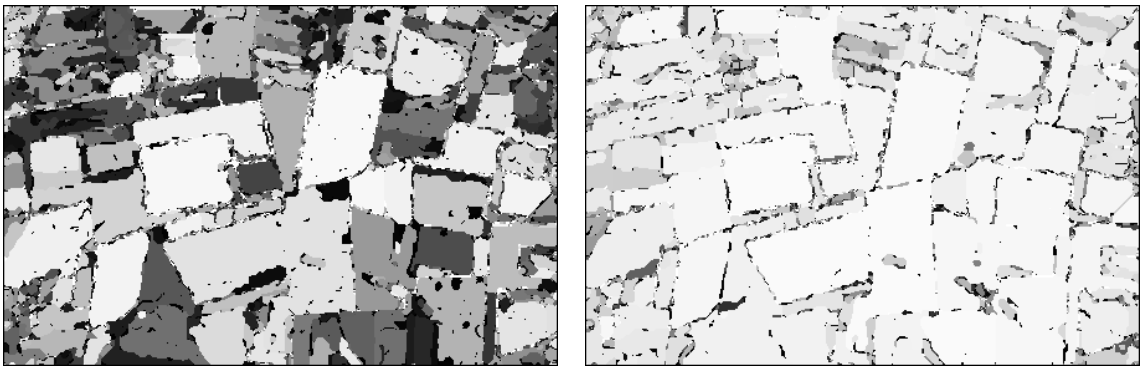
Abbildung 4.36: SA_{EQ}-Vergleichsbild der Segmentationen *Seg1* und *Seg2*

Mit Hilfe der $SA(Seg1|Seg2)$- und $SA(Seg2|Seg1)$-Bilder kann nun überprüft werden, an welchen Stellen eine Segmentation feiner ist als die andere, und umgekehrt. In Abbildung 4.37 finden sich die SA-Bilder der Segmentationen *Seg1* und *Seg2*.
Es gilt hier folgende Regel[58]:

- $\mu_{SA}(Seg1|Seg2)[x]$ dunkel, $\mu_{SA}(Seg2|Seg1)[x]$ hell $\Longrightarrow$ an der Stelle x ist *Seg2* eine Verfeinerung von *Seg1*
- $\mu_{SA}(Seg2|Seg1)[x]$ dunkel, $\mu_{SA}(Seg1|Seg2)[x]$ hell $\Longrightarrow$ an der Stelle x ist *Seg1* eine Verfeinerung von *Seg2*

Bei Betrachten der SA_{EQ}-Bilder fällt auf, dass $\mu_{SA}(Seg2|Seg1)$ im Großen und Ganzen sehr hell ist und $\mu_{SA}(Seg1|Seg2)$ an vielen Stellen sehr dunkle Bereiche aufweist. Dies bestätigt die Aussage, dass *Seg2* tendenziell eine Verfeinerung von *Seg1* ist. Jedoch gibt

[58] Abschnitt 4.3.2, Satz 2



(a) $\mu_{SA}(Seg1|Seg2)$ (b) $\mu_{SA}(Seg2|Seg1)$

Abbildung 4.37: SA-Vergleichsbild der Segmentationen *Seg1* und *Seg2*

es auch einige dunklere Bereiche in *Seg1*, welche aber auch in *Seg2* dunkel sind. In diesen Bereichen unterscheiden sich die Segmentationen wesentlich in ihrer Struktur[59].

[59] Abschnitt 4.3.1, Abbildung 4.3

5 Connected Component Labeling

5.1 Einleitung

Die Bestimmung der Zusammenhangskomponenten eines Bildes, das sogenannte Connected Component Labeling (CCL), ist ein grundlegender Verarbeitungsschritt in der Bildanalyse. Er folgt in natürlicher Weise unmittelbar auf eine Bildsegmentation [51]. Wird bei einer Bildsegmentation der Zusammenhang benachbarter Bildpunkte ermittelt, werden beim CCL die zusammengehörigen benachbarten Bildpunkte zu einem Bildbereich[1] zusammengefasst, indem sie mit einem eindeutigen gemeinsamen Kennzeichen (Label) versehen werden. Einzelne Bildbereiche können hierdurch angesprochen und voneinander unterschieden werden [52]. CCL stellt damit den Übergang von einer pixelbasierten auf eine objektbasierte Betrachtungsweise dar. Eine detaillierte Beschreibung der Aufgaben des CCL wird im nächsten Abschnitt gegeben.
Das Connected Component Labeling wird für viele Bildanalyseverfahren, insbesondere im Bereich der Objekterkennung, benötigt[53, 54]. Es sei beispielsweise die Erkennung und Analyse von Schiffen, Häusern oder Personen genannt, welche für viele zukünftige On-Board Anwendungen von zentraler Bedeutung ist.
Im Rahmen dieser Arbeit soll, wie im Abschnitt 1.4 aufgeführt, eine CCL-Methode bereitgestellt werden, welche vorteilhaft auf einer FPGA-Plattform implementiert werden kann und mit welcher Fernerkundungsdaten prozessiert werden können. Dabei soll das Verfahren in hohem Maße echtzeitfähig sein. Die Anforderungen, die an die CCL-Methode gestellt werden, werden in Abschnitt 5.4.1 ausführlich dargestellt. Zusammengefasst sollen große komplexe Bildszenen in wenigen Sekunden prozessiert werden können.
Diese Anforderungen unterscheiden sich von denen fast aller publizierten Hardware- und FPGA-basierten Verfahren. Der Fokus dieser Verfahren liegt in aller erster Linie darin, Bilder mit niedriger Auflösung und Komplexität mit einer möglichst hohen Bildwiederholrate prozessieren zu können [55]. Dies sind wichtige Voraussetzungen, um eine Objekterkennung in Videoanwendungen durchführen zu können, wie sie bei automatischen Überwachungsaufgaben, optischer Personenerkennung und in der Robotik eingesetzt wird [51].
Dieses Kapitel ist folgendermaßen aufgebaut. Zuerst werden unterschiedliche bekannte CCL-Verfahren vorgestellt (Abschnitt 5.3). Anschließend wird eine Anforderungsanalyse durchgeführt und untersucht welche Methode prinzipiell den Anforderungen am nächsten kommt. Insbesondere soll die Methode performant auf einem FPGA umgesetzt werden können (Abschnitt 5.4). Ausgehend von diesem Verfahren wird ein spezielles, an das FPGA angepasstes Verfahren vorgestellt (Abschnitt 5.5). Anhand einer Prototyp-

[1] Segment, Objekt

Implementation wird gezeigt wie dieses Verfahren auf einem FPGA realisiert werden kann (Abschnitt 5.6), welcher Ressourcenverbrauch (5.6.4) notwendig ist, und welche Leistung erzielt werden kann (Abschnitt 5.6.3). Um die Verarbeitungsgeschwindigkeit der FPGA-Implementation besser einschätzen zu können, wird zum Schluss ein Vergleich zu einer Softwareimplementationen durchgeführt (Abschnitt 5.6.5).

5.2 Grundlage

Die Aufgabe eines CCL-Verfahrens ist es, den Zusammenhangskomponenten eines Bildes eine eindeutige Identifikationsnummer zuzuordnen [56, 57]. In der Literatur werden meistens CCL-Verfahren auf Binärbildern behandelt, bei welchen ausschließlich die Vordergrundobjekte gelabelt werden sollen (Abbildung 5.1).

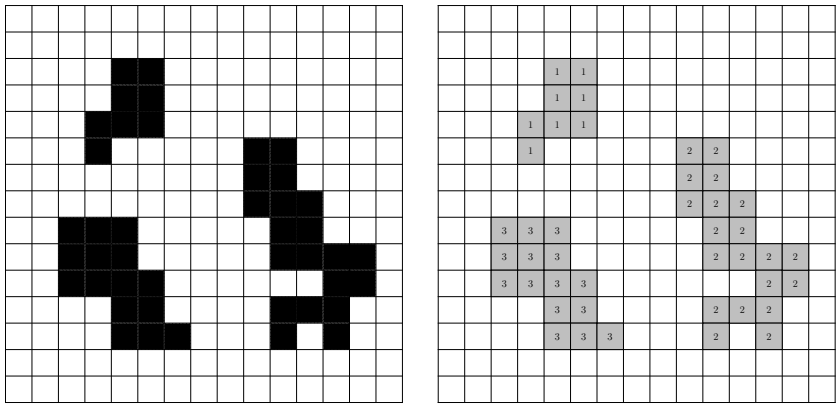


Abbildung 5.1: links Binärbild, rechts gelabeltes Bild

Im Rahmen dieser Arbeit ist jedoch der allgemeine Fall wichtig, bei dem wie auf einer Landkarte jeder Bildpunkt einem Bereich zugeordnet wird (Abbildung 5.2). Zwei benachbarte Bildpunkte haben im allgemeinen Fall die Freiheit zusammenhängend zu sein oder auch nicht. Diese Freiheit ist bei einer Binärdarstellung nicht gegeben. Eine (multimodale) Zerlegung, wie in Abbildung 5.2 gezeigt, kann bei einem Binärbild nicht auftreten.
Zwei beliebige Bildpunkte werden genau dann zusammenhängend genannt, wenn es einen Weg über zusammenhängende benachbarte Bildpunkte gibt, der sie verbindet (Abbildung 5.3). Zusammenhängende Bildpunkte bilden eine Zusammenhangskomponente. Wie in den Abbildungen 5.1 und 5.2 ersichtlich, wird jedem Bildpunkt eine Nummer zugewiesen, so dass Bildpunkte in der gleichen Zusammenhangskomponente die gleiche Nummer, Bildpunkte in verschiedenen Zusammenhangskomponenten verschiedene Nummern bekommen.
Es wird außerdem zwischen dem 4- und 8-Zusammenhang unterschieden [58]. Beim 4-Zusammenhang darf der Weg nur über die oberen, unteren, links, und rechts gelegenen Nachbarn verlaufen. Beim 8-Zusammenhang darf der Weg zusätzlich über die dia-

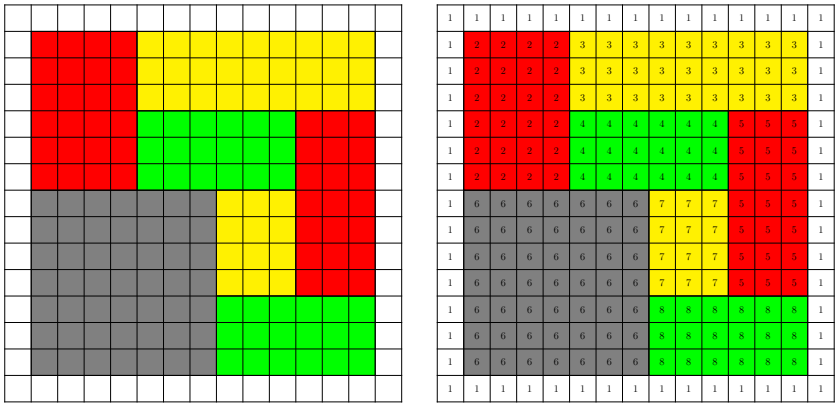


Abbildung 5.2: links ein (multimodales) Zusammenhangsbild (Farbcodierung), rechts das gelabeltes Bild

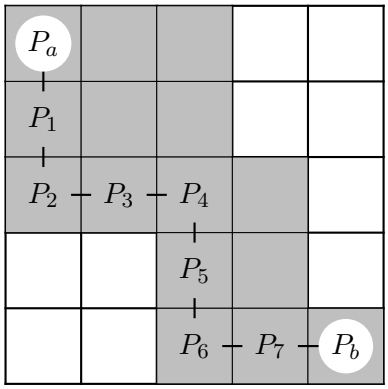


Abbildung 5.3: Zusammenhang zwischen zwei Punkten

gonal gelegenen Nachbarn (Abbildung 5.4) verlaufen. Es sei darauf hingewiesen, dass sich der 4- und 8-Zusammenhang unterscheiden. 4-Zusammenhangskomponenten stimmen im Allgemeinen nicht mit den 8-Zusammenhangskomponenten übereinstimmen. In Abbildung 5.5 sind zwei Flächen dargestellt, welche durch eine Diagonale getrennt werden. Die zwei Flächen sind bezüglich des 4-Zusammenhangs getrennt, bezüglich des 8-Zusammenhangs aber zusammenhängend. Die Diagonale ist außerdem nur bezüglich des 4-Zusammenhangs zusammenhängend. Eine Zerlegung, bei der die zwei Flächenstücke getrennt werden und die Diagonale ein Segment darstellt, entspricht weder dem 4- noch dem 8-Zusammenhang (Abbildung 5.5(d)).

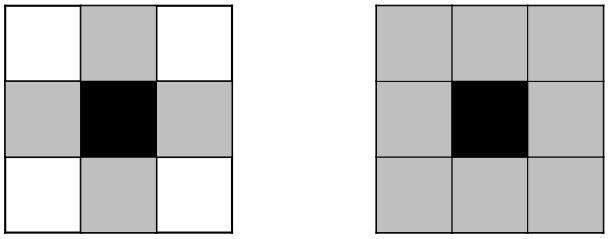
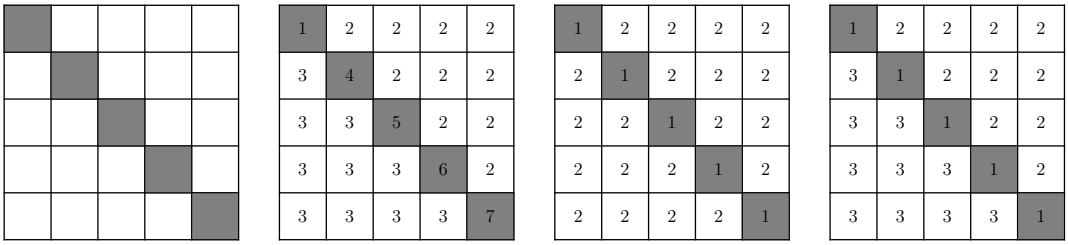


Abbildung 5.4: links: 4-Nachbarn des Zentralpunktes, rechts: 8-Nachbarn des Zentralpunktes, Zentralpunkt ist schwarz dargestellt, Nachbarn sind grau dargestellt



(a) Beispielmuster

1	2	2	2	2
3	4	2	2	2
3	3	5	2	2
3	3	3	6	2
3	3	3	3	7

(b) 4-Zusammenhang

1	2	2	2	2
2	1	2	2	2
2	2	1	2	2
2	2	2	1	2
2	2	2	2	1

(c) 8-Zusammenhang

1	2	2	2	2
3	1	2	2	2
3	3	1	2	2
3	3	3	1	2
3	3	3	3	1

(d) weder 4- noch 8-Zusammenhang

Abbildung 5.5: Unterschied 4-, 8-Zusammenhang

5.3 Verfahren

Es gibt verschiedene CCL-Verfahren, von denen im Folgenden die gängigsten vorgestellt werden. Dabei liegt der Fokus in erster Linie auf Methoden, welche bereits auf einer Hardware- oder FPGA-Plattform umgesetzt wurden.
Eine schöne allgemeine Kategorisierung verschiedenster CCL-Verfahren findet sich bei Wu et al. [52] und moderne Softwarelösungen bei Lacassagne und Zavidovique [53].

5.3.1 Multipass Verfahren

Beim Multipassverfahren werden mehrere Bilddurchläufe benötigt. Ein erstes Verfahren wurde 1966 von Rosenfeld und Pfaltz [56] vorgestellt. Dieses ist aber relativ umständlich und rechenaufwendig. Eine Weiterentwicklung dieses Verfahrens wurde 1981 von Haralick [59] präsentiert. Ein Binärbild wird dabei spaltenweise zuerst von oben links bis unten rechts mit einer Vorwärtsmaske und anschließend von unten rechts bis oben links mit einer Rückwärtsmaske durchlaufen (Abbildung 5.6). Dem zentralen Bildpunkt der Maske wird dabei der kleinste Index, aller mit ihm zusammenhängenden Nachbarn der Maske, zugeordnet. Falls kein bereits indizierter Bildpunkt in der Maske mit ihm verbunden ist, wird ihm ein neuer, noch nicht benutzter Index zugeordnet. Die Anzahl der Durchläufe

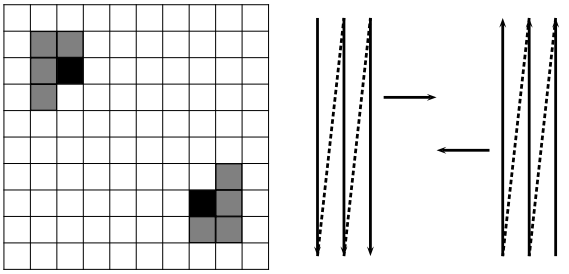


Abbildung 5.6: linkes Bild: oben links Vorwärtsmaske, unten rechts Rückwärtsmaske, schwarz Zentralpunkt; rechtes Bild: Scandurchlauf, links Vorwärtsscan, rechts Rückwärtsscan

ist vom Bildinhalt abhängig. In Abbildung 5.7 dargestellten Beispiel wird genau ein Vorwärts- und ein Rückwärtslauf benötigt.

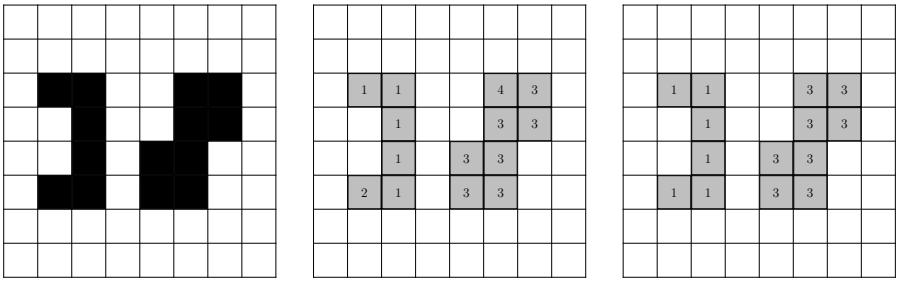


Abbildung 5.7: links Binärbild, in der Mitte Label nach Vorwärtsscan, rechts Label nach Rückwärtsscan

Eine FPGA-Implementation dieses Verfahrens zur Prozessierung eines Videostroms mit einer Bildgröße von 256*256 Bildpunkten wurde 1999 von Crookes und Benkrid [60] vorgestellt. Bei einer Taktfrequenz von 76 MHz konnten um die 70 Durchläufe pro Sekunde durchgeführt werden. Laut Autoren wurden durchschnittlich zwei Läufe für einen Labelvorgang benötigt und folglich konnten 35 Bilder pro Sekunde verarbeitet werden. Es wird allerdings in einer 2003 veröffentlichten Arbeit von Suzuki et al. [61] bereits darauf hingewiesen, dass mit diesem Verfahren vor allem bei größeren und komplexeren Bildern im Durchschnitt viel mehr als zwei Durchläufe pro Bild benötigt werden. Das Verfahren wurde in dieser Arbeit daher um eine eindimensionale Lookup-Tabelle erweitert. Dieses Verfahren benötigte bei 2314 untersuchten Binärbildern mit einer Bildgröße von 512*512 Bildpunkten maximal vier Durchläufe. Es konnte aber nicht ausgeschlossen werden, dass mehr als vier Läufe gebraucht werden. Über eine Hardware-Implementation dieses Verfahrens ist nichts bekannt.
Aufgrund der Einfachheit des Verfahrens und des statischen Programmverlaufes wäre das von Crookes und Benkrid verwendete Verfahren jedoch sehr gut für eine FPGA-

Implementation geeignet. Dieses Verfahren wird daher in Abschnitt 5.4.3 einer genaueren Untersuchung unterzogen um festzustellen wie viele Läufe bei größeren und komplexeren multimodalen Labelingproblemen benötigt werden.

5.3.2 Two-Pass Verfahren

Das Verfahren und dessen Entwicklung

Bei klassischen Two-Pass Verfahren benötigt man im Gegensatz zu Multipassverfahren genau zwei Bilddurchläufe und einen Zwischenschritt. Die ersten bekannten Versionen eines Two-Pass Verfahrens findet sich in einer 1966 veröffentlichten Publikation von Rosenfeld und Pfaltz [56]. Im ersten Schritt wird das Bild, wie beim Multipassverfahren, zeilenweise mit einer Maske durchlaufen. Die Masken für den 4- und 8-Zusammenhang sind in Abbildung 5.8 dargestellt.

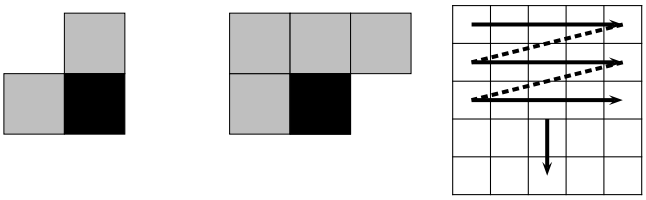
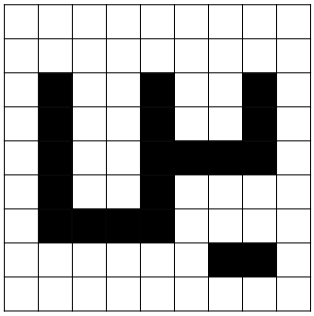
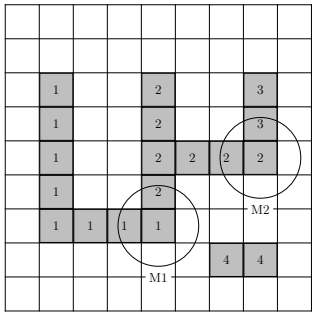


Abbildung 5.8: von links nach rechts: Maske für den 4-Zusammenhang, Maske für den 8-Zusammenhang, Bilddurchlauf der Maske beim ersten und zweiten Lauf

Dem zentralen, ausgezeichneten Punkt der Maske wird, wie bei Multipass-Verfahren, der kleinste Index der mit ihm verbundenen Punkte der Maske zugeordnet. Ist kein Punkt in der Maske mit ihm verbunden, wird dem ausgezeichneter Punkt ein neuer Index zugeordnet. Tritt nun der (Konflikt-)Fall ein, dass zwei oder mehr Punkte der Maske miteinander verbunden sind, diesen aber verschiedene Indizes zugeordnet sind, werden diese Indizes als zusammengehörig in einer Äquivalenztabelle[2] vermerkt. Nach Durchlaufen des ersten Schrittes haben Bildpunkte in verschiedenen Zusammenhangskomponenten verschiedene vorläufige Indizes (Prälabels). Falls Konfliktfälle aufgetreten sind, besitzen die Bildpunkte einer Zusammenhangskomponente aber noch unterschiedliche vorläufige Indizes.
Diese werden im zweiten Durchlauf gleichgesetzt. Um dies durchzuführen, wird in einem Zwischenschritt die Equivalenztabelle ausgewertet und eine Lookup-Tabelle erstellt, welche den vorläufigen Indizes ihre finalen Indizes zuordnet. Bei dem in Abbildung 5.9 gezeigten Beispielfall treten zwei Konfliktfälle, $M1$ und $M2$ auf.
Der kritische Punkt bei diesem Verfahren ist die Auswertung und Größe der Merge-Tabelle. Treten viele Konfliktfälle und vorläufige Indizes auf, was gerade bei größeren Bildern mit komplexem Bildinhalt der Fall ist, kann mit dem ursprünglichen Verfahren von Rosenfeld und Pfaltz das Labeling nicht mehr in vernünftiger Zeit abgeschlossen

[2] auch Merge-Tabelle genannt





Äquivalenztabelle	
M1	(1,2)
M2	(2,3)

Lookup-Tabelle	
1	1
2	1
3	1
4	4

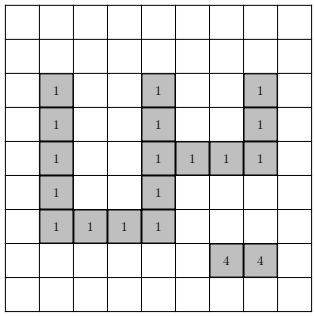


Abbildung 5.9: oben links: Binärbild, oben rechts: Ergebnis nach dem ersten Bilddurchlauf, bei dem 2 Konfliktfälle M1 und M2 aufgetreten sind, unten links: Äquivalenztabelle nach dem ersten Lauf und die Lookup-Tabelle nach Auswertung der Äquivalenztabelle (Zwischenschritt), unten rechts: Ergebnis nach dem zweiten Bilddurchlauf

werden. Daher wurden viele Algorithmen entwickelt, welche auf dem ursprünglichen Verfahren aufbauen. Eine ausführliche Beschreibung des klassischen Verfahrens und weiterer darauf aufbauender Verfahren findet man im Lehrbuch Computer and Robot Vision von Robert M. Harlick und Linda G. Shapiro von 1992 [13]. Aber auch in neueren Arbeiten wie zum Beispiel von Rachakonda et al. [62] werden Verfahren vorgestellt mit welchen, wie in Abschnitt 5.4.3 gezeigt wird, bei komplexen Bildern keine zeitnahe Verarbeitung möglich ist. Ein Durchbruch gelang erst mit der Verwendung von Union-Find Datenstrukturen[3] wie sie unter anderem in den Arbeiten von Samet und et al. [54, 64] Mitte der 1980 Jahre vorgestellt wurden. Fast alle modernen Two-Pass Verfahren (wie in [53]) verwenden Union-Find Datenstrukturen mit der auch die Auflösung sehr großer Äquivalenztabellen, wie in Abschnitt 5.4.3 gezeigt wird, mit der heutigen verfügbaren Rechenleistung, in Sekundenbruchteilen möglich ist.

[3] abstrakte Datenstruktur zur Verwaltung von Partitionen einer Menge [63]

Hardware/FPGA-Implementationen

Aufgrund der Bedeutungen von CCL für viele Anwendungen, beispielsweise im Bereich der Robotik und insbesondere der daraus ergebenden Anforderung, CCL auf einem Videostrom durchführen zu müssen, wurden Hardware/FPGA-Lösungen entwickelt. Im Folgenden werden einige von diesen vorgestellt.
In der Arbeit von Rackakonda von 1995 [62] wurde ein (klassisches) Two-Pass Verfahren für eine rekonfigurierbare Computerplattform (Splash 2), bestehend aus neun Xilinx XC4010 FPGAs, vorgestellt. Das System wurde zur Videoprozessierung entwickelt und sollte Binärbilder mit einer Größe von 512*512 Bildpunkten mit 30 FPS verarbeiten können. Um 30 FPS erreichen zu können, wurden auf diesem System zwei CCL-Einheiten integriert, welche die Bilder des Videostroms abwechselnd bearbeiteten. Es wurde hier schon darauf hingewiesen, dass die Verarbeitungsgeschwindigkeit sehr stark vom Bildinhalt abhängig ist. Treten viele Konfliktfälle auf, konnte der Labeling Schritt nicht mehr in den für 30 FPS notwendigen 0,033 Sekunden durchgeführt werden. Aufgrund der Einfachheit des bei diesem Verfahren verwendeten Algorithmus wurde dieser für eine Implementation in Betracht gezogen. Es wurde daher genau untersucht, wie sich dieses CCL-Verfahren bezüglich multimodalen labelings auf großen Bildern mit komplexen Bildinhalt verhält. Diese Untersuchung findet sich in Abschnitt 5.4.3. Es stellte sich aber leider heraus, dass es bei größeren Bilder mit komplexen Bildinhalt viel zu langsam ist.
Ein weiteres auf einer rekonfigurierbaren Computerplattform umgesetztes Two-Pass Verfahren wurde 2003 von Schmidt und Koch [65] vorgestellt. Das System (ACE-V) bestand aus einer rekonfigurierbaren Berechnungseinheit (einem Xilinx Virtex 1000 FPGA), welche über einen Bus mit einer konventionellen CPU [4] verbunden wurde. Das CCL-Verfahren ist in dieser Arbeit jedoch nicht ganz korrekt dargestellt. Das FPGA wurde mit 36 MHz betrieben. Ein Binärbild der Größe 512*512 Pixel konnte in 15 ms (66 FPS) gelabelt werden. Leider wurde nicht genauer beschrieben, wie viele Labels und Konfliktfälle maximal auftreten durften. Es wurde allerdings ein Vergleich mit einer Softwareumsetzung des Algorithmus auf einem AMD Athlon XP 1533 MHz durchgeführt. Sie stellten fest, dass obwohl die CPU mit $42,6\times$ der Taktfrequenz des FPGAs lief, sie nur um Faktor 1,16 schneller war. Dies lässt darauf schließen, dass das Two-Pass Verfahren im Vergleich zu einer CPU sehr effizient auf einem FPGA durchgeführt werden kann. Diese Aussage konnte bei einem Vergleich des in dieser Arbeit entwickelten CCL-Verfahrens mit Softwareverfahren nur teilweise bestätigt werden (Abschnitt 5.6.5).
Jablonski und Gorgon stellten 2004 eine FPGA-Implementation eines Two-Pass Verfahrens vor[5]. Eine Besonderheit dieser Arbeit ist, dass das Verfahren mit Handel-C implementiert wurde, welches auch in dieser Arbeit verwendet wurde. Sie bemerkten, dass, um die gewünschte Leistung zu erzielen, nicht genügend Speicherbandbreite zur Verfügung stand, um die Äquivalenztabelle im RAM unterzubringen. Sie mussten diese daher im internen Block-RAM des FPGAs unterbringen. Da dieser aber sehr begrenzt ist, mussten sie die maximale Anzahl möglicher Zusammenhangskomponenten auf 4095 beschränken. Mithilfe von zwei CCL-Einheiten, welche parallel betrieben wurden, konnten sie

[4] Sun microSPARC-II RISC
[5] RC-1000 PP Board (Virtex XCV1000BG560-6 FPGA)

schließlich einen Videostrom von Binärbildern der Größe von 512×512 Bildpunkten mit ungefähr 60 FPS prozessieren[6]. Es fehlt leider eine genau Beschreibung des Verfahren, insbesondere wie die Auflösung der Äquivalenztabelle durchgeführt wurde.
Um ebenfalls Speicherengpässen zu entgehen und den Labeling Prozess selbst zu beschleunigen, entwickelten Appiah et al. eine 2008 vorgestelltes Two-Pass Verfahren [66], welches zuerst eine Lauflängenkodierung eines binären Bildes vornimmt und anschließend das Labeling durchführt. Der Algorithmus wurde ebenfalls mithilfe der Hardwarebeschreibungssprache Handel-C entwickelt und auf einem Virtex-4 FPGA [7] implementiert. Bei einer Taktfrequenz von 50 MHz konnten sie einen Videostrom bestehend aus Binärbilder der Größe 640*480 mit ungefähr 79 FPS labeln. Sie bemerkten, dass das Verfahren stark vom Bildinhalt abhängig ist. Auch mussten sie ebenfalls die Anzahl der maximalen auftretenden Objekte auf 4095 beschränken[8], um die Äquivalenztabelle im Block-RAM unterbringen zu können. Sie verglichen die Performance ihres Algorithmus mit einer mit MATLAB erstellten Softwareimplementation, welche auf einem Intel Pentium 4 2.8 GHz mit 2.0 GByte SDRAM ausgeführt wurde. Mit der FPGA-Lösung konnte eine Leistungssteigerung um etwa Faktor fünf gegenüber der MATLAB-Implementation erreicht werden[9].

5.3.3 One-Pass Verfahren

One-Pass Verfahren benötigen, im Gegensatz zu Multipass- und Two-Pass Verfahren nur einen Bilddurchlauf. Es wird gewöhnlich nur von einem One-Pass Verfahren gesprochen, wenn eine Verarbeitung im Rasterformat stattfindet[10].
Mit einem One-Pass Verfahren kann jedoch, ohne Einschränkung an den Bildinhalt, keine Labelmaske erzeugt werden. Zur Erklärung stelle man sich ein Bild vor, auf dem ein großes U abgebildet ist. Wird das Bild zeilenweise eingelesen, ist in den ersten Zeilen noch nicht klar, ob die beiden vertikalen Linien des Us (| |) zusammengehören oder nicht. Dies kann frühestens in der Bildzeile entschieden werden, in der die beiden Linien verbunden werden.
Der Vorteil von One-Pass Verfahren ist jedoch, dass sie im Gegensatz zu Two-Pass Verfahren theoretisch doppelt so schnell ausgeführt werden können [11] und im Besonderen, dass kein Bildzwischenspeicher (Framebuffer) benötigt wird.

[6] dies entspricht vier Takte pro Pixel
[7] Modell: XC4VLX160, Celoxica RC340 Board
[8] tatsächlich ist der Wert der maximal auftreten Segmente von der Struktur der Segmente abhängig und im schlimmsten Fall kann nicht einmal ein Segment verarbeitet werden. Der Grund hierfür ist die Lauflängenkodierung, welche zur Darstellung beispielsweise von Segmente mit vielen kleinen Löchern nicht geeignet ist.
[9] es sei jedoch bemerkt, dass die in dieser Arbeit entwickelte Softwareimplementation weitaus leistungsfähiger ist und sich dieses Ergebnis daher nicht in dieser Höhe auf Softwarelösungen verallgemeinern lässt (Abschnitt 5.6.5)
[10] Verfahren, bei denen die Bildpunkte dynamisch angesprochen werden, wie z.B. Region-Growing Verfahren (Abschnitt 5.3.4: Sonstige Verfahren), werden gewöhnlich nicht als One-Pass-Verfahren bezeichnet, auch wenn sie nur einen Bilddurchlauf benötigen.
[11] da nur ein Bilddurchlauf benötigt wird

Eine FPGA-Implementation eines One-Pass Verfahren, bei welcher eine Labelmaske generiert wird, wurde 2008 vorgestellt [57]. Das Verfahren kann nur eine spezielle Klasse von Bildern[12] korrekt verarbeitet. Das Besondere an dieser Arbeit ist, dass eine Datenstromverarbeitung des CCL verwirklicht wird, bei welcher eine Labelmaske erzeugt wird. Auf einem Altera-Stratix-EP1S24-FPFA wurde damit bei einer Taktfrequenz von 60 MHz ein Videostrom[13] von binären-Bildern der Größe von 2048*2048 Pixeln mit 14 FPS gelabelt[14].

Muss keine Labelmaske erstellt werden, sondern nur bestimmte Objekteigenschaften, wie Anzahl der Objekte, die Objektgröße, eine Objektbox (Bounding-Boxes) oder der Objektschwerpunkt ermittelt werden, ist dies ebenfalls in einem Bilddurchlauf möglich.

In einer Reihe von Veröffentlichungen von Bailey und Johnston [67, 68] wurde 2008 ein Verfahren und dessen FPGA-Implementation vorgestellt, mit welcher die Anzahl der Objekte eines Binärbildes bestimmt werden kann. Es gelang ihnen eine Datenstromverarbeitung durchzuführen. Konfliktfälle wurden nach jeder Bildzeile aufgelöst. Die Auflösung der Konfliktfälle erfolgte in der Zeilenaustastlücke (analoger) Videosignale, welche in ihrem Falle ungefähr 20% - 25% der Länge einer aktiven Zeile entsprach. Bei komplexen Bildinhalten war diese Zeit jedoch in vielen Fällen bei weitem nicht ausreichend[15]. Das Verfahren wurde auf einem RC100 Entwicklunsgbord von Celoxica, welcher auf einem Spartan-II XC2s200 FPGA basiert, umgesetzt. Sie verwendeten die Hardwarebeschreibungssprache Handel-C. Aufgrund der beschränkten Ressourcen mussten sie die maximale Anzahl der Labels auf 512 beschränken. Einen Videostrom, bestehend aus Binärbilder der Bildgröße 640*480 Pixel, konnten sie mit durchschnittlich 60 FPS prozessieren. Leider fanden sich keine genauen Angaben, welchen Einfluss die Bildbeschaffenheit auf die Laufzeit des Verfahrens hat. In [55] wurde 2013 eine Weiterentwicklung dieses Verfahrens und seine FPGA-Implementation vorgestellt, mit welchem ein extrem hoher Datendurchsatz von um die 3 GPixeln[16] erreicht werden konnte[17].

In [69] wurde 2011 ein zu Bailey und Johnston vergleichbares Verfahren vorgestellt mit welchen Objektboxen in binären Bildern ermittelt wurden. Ein Videostrom, bestehend aus binären Bilddaten der Größe 800*600, konnte ebenfalls mit extrem hohen Geschwindigkeiten (580 FPS) prozessiert werden. Sie verglichen ihre FPGA-Implementation mit einer Softwareimplementation, mit welchem sie gegenüber einem 3.18 GHz Dual Core Xeon eine Leistungssteigerung um den Faktor vier erreichten. Gegenüber einer auf dem FPGA integrierten PowerPC CPU, welcher mit 300 MHz betrieben wurde, erreichten sie sogar eine Steigerung um den Faktor 215.

[12] sogenannte k-Concave Bilder. Zur Bestimmung des Labels eines Bildpunktes ist es bei k-Concave Bildern ausreichend, alle Bildzeilen vor diesem und k-Zeilen nach diesem zu betrachten.
[13] 10-Conkave Bilder
[14] dies entspricht ungefähr einem Takt pro Bildpunkt
[15] zur Auflösung der Konfliktfälle wäre ungefähr mehr als das doppelte der verfügbaren Zeit benötigt worden.
[16] dies entspricht 2861 FPS bei einer Bildgröße von 1024*1024
[17] diese hohe Geschwindigkeit wurde in erster Linie dadurch erreicht, dass ein Bild in bis zu über 35 Bildstreifen zerlegt wurde, welche parallel prozessiert wurden

5.3.4 Sonstige Verfahren

Neben den One-, Two- oder Multipass-verfahren existiert noch eine Vielzahl weiterer CCL-Verfahren, von welchen im Folgenden die bekanntesten Methoden vorgestellt werden.

Einen großen Anteil an diesen haben Region-Growing und Contour-Tracing (CT) Verfahren. Diese haben gemein, dass die Bildobjekte nacheinander gelabelt werden. Da das Bild hierbei jedoch nicht mehr im Rasterformat bearbeitet werden kann, treten bei diesen Verfahren sehr viele verteilte Speicherzugriffe auf. Daher sind diese nur eingeschränkt für eine FPGA-Implementation zu empfehlen. Eine FPGA-Implementation eines CT-Verfahrens wurde 2007 veröffentlicht [70]. Trotz einer Taktfrequenz von 67 MHz erzielten sie mit diesem Verfahren, selbst bei kleinen Bildern der Größe 320*240 Pixel, nur eine mäßige Bildwiederhohlrate von 25 FPS. Außerdem war die maximale Anzahl der Objekte auf 61 beschränkt.

Eine eher exotische Art von CCL-Verfahren stellen hoch-parallele Verfahren dar. Bei diesen werden binäre Bilder in Netzstrukturen abgelegt[18]. Anschließend folgt eine pixel-parallele Verarbeitung. In [71] wurde 1988 ein derartiges Verfahren für den „NASA Goddard Space Flight Center's massively parallel processor“ vorgestellt. Ein Bild mit einer Größe von 128*128 Pixeln konnte in ungefähr 94,6 ms ($\sim$ 10 FPS) gelabelt werden. Diese Berechnung wurde parallel auf 16384 Prozessorkernen durchgeführt, wobei für jedes Pixel ein Kern verwendet wurde. Ein ähnliches Verfahren wurde in [72] 1996 vorgestellt. In dieser Arbeit wurde ein Spezialchip entwickelt, um die Netzstruktur in Hardware abzubilden. Im darauf folgenden Jahr wurde in [73] ein Verfahren vorstellt, dass nicht mehr pixel-parallel arbeitete, aber immerhin noch 128 Prozessierungselemente verwendete, und ein Bild der Größe 128*128 Pixel in 0,992 ms ($\sim$ 1008 FPS) labeln konnte. Ein großer Nachteil dieser Verfahren ist, dass mit vertretbarem Hardwareaufwand nur sehr kleine Bilder behandelt werden können[19]. Da die Bilddaten im Allgemeinen sequentiell vorliegen, wird beim Laden und Entladen eines Bildes außerdem der größte Teil des Geschwindigkeitsvorteils zunichte gemacht[20]. Die vorgestellten Implementationen dieser hoch-parallelen Verfahren sind nicht mehr zeitgemäß. Aber hoch-parallelisierte Architekturen dieser Art finden ihre natürliche Fortsetzung in GPU-Implementationen [74, 75, 76]. Die Verarbeitungsgeschwindigkeit ist hier ebenfalls vom Bildinhalt abhängig. Beispielsweise werden mit der 2011 in [75] vorgestellten Implementation auf einer NVIDIA TESLA C1016, zum labeling von Binärbild der Größe 2048*2048 Pixel, zwischen 10 und 40 ms benötigt ($\sim 40 - 100$ FPS).

Im den nächsten Abschnitt wird die Frage gestellt, welche Methode, im Rahmen dieser

[18] Pixel werden in den Knotenpunkten des Netzes abgelegt

[19] da für jeden Bildpunkt ein Netzknoten benötigt wird

[20] Anmerkung des Autors: Vom akademischen Standpunkt aus ist diese Art von Verfahren aus folgendem Grund trotzdem sehr interessant. Da die hierbei benötigte zweidimensionale Netzarchitektur der Logikblockanordnung auf einem FPGA entspricht, könnte man eine Schaltung realisieren, welche das CCL eines kompletten Bildes in wenigen Takten durchführt. Dies entspräche einem 2D-Berechnungsschema und es stellt sich die Frage, in welchem Rahmen dies umgesetzt werden könnte. Eine weitere interessante Frage, die sich in dieser Hinsicht stellt, ist, ob das (menschliche) Gehirn CCL in ähnlicher Weise durchführt.

Arbeit, prinzipiell am ehesten für eine FPGA-Implementation in Frage kommt. Dazu muss aber zuerst ein Anforderungsprofil entwickelt werden.

5.4 Wahl eines geeigneten Verfahrens

5.4.1 Anforderungen

Die Aufgabe von CCL-Verfahren ist es, wie in Abschnitt 3.1 dargestellt, anhand eines lokalen Zusammenhanggraphen eine Labelmaske zu erstellen. Aufgrund des Einsatzgebietes ergeben sich folgende Anforderungen[21]:

- Die Verarbeitung von Einzelbildern
- Allgemeines multimodales Labeling muss durchführbar sein[22]
- Die Erzeugung eines Indexbildes
- Die Unabhängigkeit des Verfahrens von Form, Größe und Anzahl der Segmente
- Sehr große Bilder müssen verarbeitet werden können[23]
- Die maximale Laufzeit der Methode muss bekannt sein
- Die maximale Laufzeit der Methode soll bei kleineren Bildergrößen unterhalb einer Sekunde betragen, bei größeren Bildgrößen im Bereich weniger Sekunden liegen[24]
- Die Laufzeit sollte möglichst unabhängig vom Bildinhalt sein
- Ein vollständig automatischer Ablauf
- Das Verfahren sollte möglichst geeignet für eine FPGA-Implementation sein[25]
- Ein möglichst geringer Ressourcenverbrauch

Die Hardwareimplementationen, welche im letzten Abschnitt vorgestellt wurden, sind in erster Linie für die Verarbeitung von Videodatenströmen ausgelegt. Die Verarbeitung von Videodaten stellt andere Anforderungen als eine Einzelbildverarbeitung. Bildsequenzen sollen mit möglichst hohem Durchsatz bearbeitet werden können. Die Einzelbilder sind von geringerer Größe[26] und in allen bekannten Fällen ist nur ein Labeling an Binärbildern vorgenommen worden[27]. Um Bilder in ausreichender Geschwindigkeit[28] bearbeiten

[21] Abschnitt 5.1
[22] Abschnitt 5.2
[23] mit einer Größe von mehr als 10000*10000 Bildpunkten
[24] kleine Bilder: $\sim 1024 * 1024$, große Bilder $\sim 10000 * 10000$
[25] insbesondere sollten die Datenverarbeitungsoperationen möglichst lokal und statisch sein (Abschnitt 2.3)
[26] bei den Beispielimplementationen unter 2048*2048 Pixel
[27] i.a.W. kein multimodales Labeling
[28] in den meisten Fällen ~ 30 FPS

zu können, mussten außerdem in den meisten Fällen Einschränkungen bei der Beschaffenheit des Bildes in Kauf genommen werden[29]. Deswegen konnte keines der vorgestellten Verfahren ohne eine Anpassung verwendet werden.
Es erfolgte zuerst eine grobe Einschätzung, welche Verfahren aus der Literatur prinzipiell den Anforderungen genügen. Die geeignetsten Verfahren wurden daraufhin genauer untersucht. Zu Analysezwecken, wurden diese Verfahren in Software implementiert. Es wurden Anpassungen vorgenommen, um die Bearbeitung großer Bilder und multimodales Labeling zu ermöglichen, ohne dass Einschränkungen an den Bildinhalt gestellt werden müssen. Es wurde darauffolgend an Beispielfällen untersucht, ob der Labelvorgang in ausreichender Zeit durchgeführt werden kann, und wie stark die Verarbeitungszeit vom Bildinhalt abhängig ist. Außerdem wurde getestet, wie viel Speicher für den Labelvorgang benötigt wird, und inwiefern sich diese Verfahren für eine FPGA-Implementation eignen.

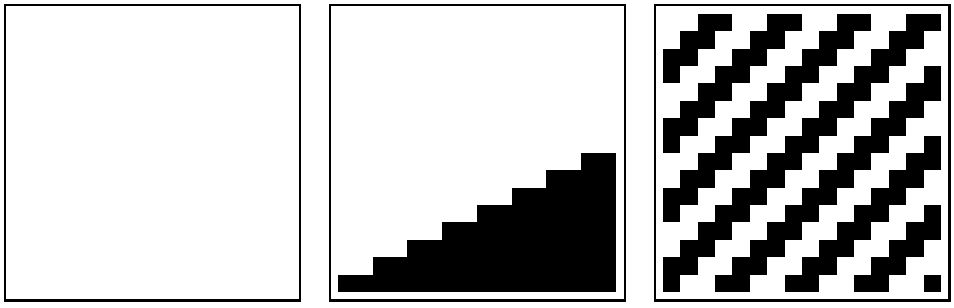
5.4.2 Testdaten

In diesem Abschnitt werden zunächst die Testdaten vorgestellt, welche zur Analyse der CCL-Verfahren benutzt wurden. Es wurden sowohl synthetische Testdaten als auch reale Daten in Form von Wolkenmasken verwendet.

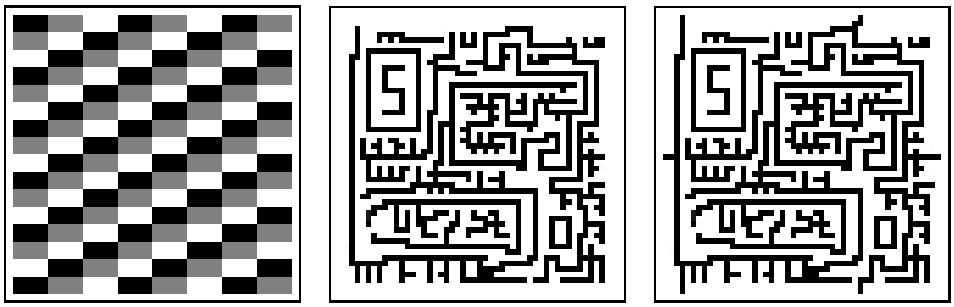
Synthetische Testdaten

Zur Erstellung der synthetischen Testdaten wurden die in Abbildung 5.10 ausgewählten Muster verwendet. Diese Muster werden aneinandergereiht, um Testbilder von einer Größe von 50*50 bis 16384*16384 Pixel zu erstellen. In Abbildung 5.11 ist zur Veranschaulichung ein Testbild und dessen Indexbild dargestellt.
Jedes Testmuster besitzt Eigenarten mit denen jeweils spezielle Aspekte untersucht werden sollen. Anhand der **weißen Testbilder** soll das Verhalten bei Bildern mit sehr niedriger Komplexität untersucht werden, in anderen Worten die minimale Laufzeit ermittelt werden[30].
Das **Dreieck-Bild** besteht aus zwei großen Objekten. Daran soll überprüft werden, wie sich die Verfahren bei Bildern mit wenigen, aber großen Segmenten verhalten.
Bei den **Diagonal-1** beziehungsweise **Diagonal-2** Testbildern treten bei Two-Pass Verfahren bezüglich des 4- bzw. 8-Zusammenhangs die meisten Konfliktfälle auf[31].
Mit den **Labyrinth-Bildern** soll getestet werden, wie die CCL-Verfahren auf viele, sehr verschachtelte Segmente reagieren. Im Unterschied zum **Labyrinth-1 Bild** sind im **Labyrinth-2** die Teillabyrinthe miteinander verbunden. Dies führt zu einer höheren Verschachtlung - insbesondere sind die Segmente noch stärker ineinander verschlungen.

[29] Beispielsweise eine maximale Anzahl an Objekten.
[30] Bemerkung: die Bilder auf denen CCL mit der minimalen Laufzeit ausgeführt werden können sind vom Verfahren und dessen Umsetzung abhängig. Tatsächlich stellt sich in Abschnitt 5.6.5 , heraus, dass die minimale Laufzeit nicht immer bei den weißen Testbildern eintritt. Jedoch wurden bei den weißen Testbilder immer die mit geringsten Laufzeiten erreicht.
[31] Abschnitt 5.6.3 wird darauf näher eingegangen

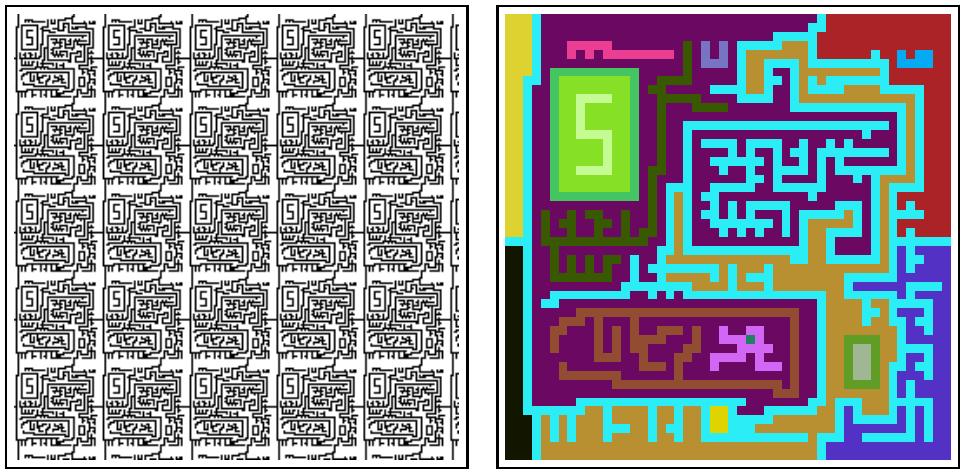


(a) Weiß (16*16) (b) Dreieck (16*16) (c) Diagonal 1 (16*16)



(d) Diagonal 2 (16*16) (e) Labyrinth 1 (50*50) (f) Labyrinth 2 (50*50)

Abbildung 5.10: Synthetische Testmuster, Größenangabe in Pixel



(a) Labyrinth 2 Testbild (256*256) (b) Labyrinth 2 Indexbild (50*50)

Abbildung 5.11: links: Synthetisches Testbild der Größe 256*256 Bildpunkte, rechts: Labelmaske des Testmusters

Reale Testdaten

Zusätzlich zu den synthetischen Testdaten wurde ein Datensatz von 365 binären Wolkenmasken verwendet[32]. Die einzelnen Wolkenmasken haben ungefähr ein Größe von 8000*8000 Bildpunkten[33]. In Abbildung 5.12 ist eine dieser Wolkenmaske abgebildet.
Anhand der Wolkenmasken sollte untersucht werden, wie sich die CCL-Verfahren bei realistischen Daten verhalten. Hervorzuheben ist hierbei, dass sich die Wolkenmasken stark in der Anzahl ihrer Wolkenpixel und in ihrer Struktur unterscheiden[34].

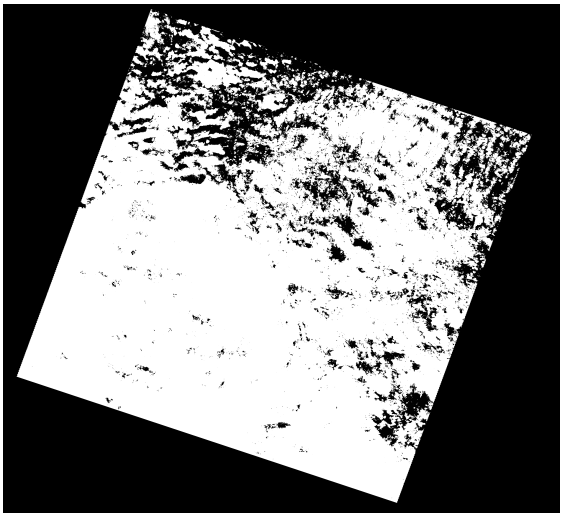


Abbildung 5.12: Wolkenmaske einer Landsat-ETM Satellitenszene, Orginalgröße 8151*7191 Bildpunkte, weiß Wolke, schwarz nicht Wolke und Rand

5.4.3 Eignung verschiedener CCL-Verfahren

Das Ziel in diesem Abschnitt ist festzustellen welche CCL-Methode die gestellten Anforderungen am ehesten erfüllt.

Vorauswahl

Nach der groben Einschätzung der in Abschnitt 5.3 vorgestellten Verfahren wurde, unter Berücksichtigung der in Abschnitt 5.4.1 gestellten Anforderungen, der Schluss gezogen, dass **Multi- und Two-Pass Methoden** am ehesten für eine FPGA-Implementation in Frage kommen.

[32] Die Daten wurden mit dem ACCA-Wolkendetektionsalgorithmus (freundlicherweise von Matthias Eder zur Verfügung gestellt) aus Landsat ETM Daten extrahiert

[33] die Bildgrößen georeferenzierter Landsat-Bilder sind im Allgemeinen nicht vollkommen konstant. Allerdings spielt der geringere Unterschied bei diesen Untersuchungen keine wesentliche Rolle.

[34] da die Wolkenbedeckung auf den Testbildern sehr variabel ist

Die Begründung hierfür ist, dass FPGA-Implementationen wie das Multi-Pass Verfahren von Crookes und Benkrid[35] und das Two-Pass Verfahren von Jablonski und Gorgon[36] den gewünschten Vorstellungen ab nächsten kamen.
Single-Pass Methoden wie die von Bailey et al.[37], obwohl sehr gut für eine Hardware-/FPGA-Implementation geeignet, schieden aus, da mit den vorgestellten Verfahren entweder keine Labelmaske erzeugt werden kann und/oder starke Einschränkungen bei der Komplexität der Bilder hingenommen werden müssen. Ein Anpassung dieser Verfahren an die gestellten Anforderungen schien im Vergleich zu Multipass- bzw. Two-Pass Verfahren unverhältnismäßig aufwändig und daher nicht sinnvoll. Insbesondere da davon ausgegangen werden kann, dass eine Implementation, falls überhaupt möglich[38], überaus komplex werden würde.
Von den sonstigen Verfahren wurden nur **Region-Growing Verfahren** in Betracht gezogen. Die anderen Verfahren waren schlicht zu speziell, um mit vertretbaren Aufwand an die Anforderungen angepasst werden zu können. Aufgrund des dynamischen Speicherzugriffes sind auch Region-Growing Verfahren vom technischen Standpunkt eher ungeeignet für eine FPGA Umsetzung [39], insbesondere im Vergleich mit einer CPU-Implementation. Es wurde trotzdem ein Region-Growing Verfahren untersucht um diese Verfahren einordnen zu können.
In Folgenden werden nun die Verfahren, welche dem ersten Eindruck nach am besten geeignet sind, näher untersucht. Sie wurden derart modifiziert, dass sie ein multimodales Labeling[40] von großen Bildern durchführen können.

Laufzeitverhalten von Multi-Pass Verfahren

Als sehr geeignet für eine FPGA-Implementation erschien das Multipass-Verfahren welches in der Arbeit von Crookes und Benkrid[60] verwendet wurde[41]. Es hat den Vorteil, dass es ein sehr statisches Verfahren ist, bei dem keine dynamischen Speicherzugriffe notwendig sind und nur ein relativ einfach umzusetzender Verarbeitungsschritt benötigt wird[42]. Dieses Verfahren lässt sich außerdem fast ohne Modifikation dahingehend erweitern, so dass multimodales Labeling durchführt werden kann. Es kann daher davon ausgegangen werden, dass sich dieses Verfahren mit geringem Aufwand und Ressourcenverbrauch auf einem FPGA umsetzen lässt. Es muss aber geklärt werden, inwiefern dieses Verfahren in der Lage ist, große Bilder mit komplexen Bildinhalten zu verarbeiten[43].
Die Anzahl der Läufe, welche für das Labeling der synthetischen Testbilder benötigt werden, ist in Abbildung 5.13 dargestellt. Es wird deutlich, dass die Anzahl der Läufe

[35] Abschnitt 5.3.1
[36] Abschnitt 5.3.2
[37] Abschnitt5.3
[38] Bemerkung: natürlich unter den gegebenen Umständen
[39] Bemerkung: Nur wenige Programmteile können parallelisiert werden
[40] s. Abschnitt 5.2
[41] Abschnitt 5.3.1
[42] d.h. das Verfahren kann mit einer sehr kleinen Logik realisiert werden, die mit hoher Taktfrequenz ausgeführt werden kann.
[43] i.a.W. wie viele Läufe nötig sind um das CCL abzuschließen

extrem vom Bildinhalt abhängig ist. Die Treppenstufen, welche im Diagonal-2* Testbild [44] auftreten, stellen das schlimmste Szenario dar. Die Bildhöhe in Pixeln entspricht hier in etwa der Anzahl der Läufe. Aber nicht nur in diesem speziellen Fall werden sehr viele Läufe benötigt, sondern auch bei den Labyrinth-2 Bildern wird eine sehr hohe Anzahl an Läufen benötigt.

Syn. Testbilder	Anzahl der Läufe						
Bildhöhe/breite in Pixel	50	512	1024	2048	4096	8192	16384
Weiß	2	2	2	2	2	2	2
Diagonal 1	4	4	4	4	4	4	4
Diagonal 2	2	2	2	2	2	2	2
Diagonal 2*	47	509	1021	2045	4093	8189	-
Dreieck	2	2	2	2	2	2	2
Labyrinth 1	23	23	23	25	23	23	23
Labyrinth 2	23	186	363	745	1479	2943	5889

Abbildung 5.13: Multipass-Verfahren, Synthetische Testbilder, Anzahl der benötigten Filterläufe

Es wurde auch die Anzahl der Läufe ermittelt, welche bei den realen Testdaten benötigt werden (Abbildung 5.14). Es stellt sich heraus, dass auch hier die Anzahl der Läufe je nach Testbild sehr unterschiedlich ist. Die höchste Anzahl an Läufen liegt bei 300.

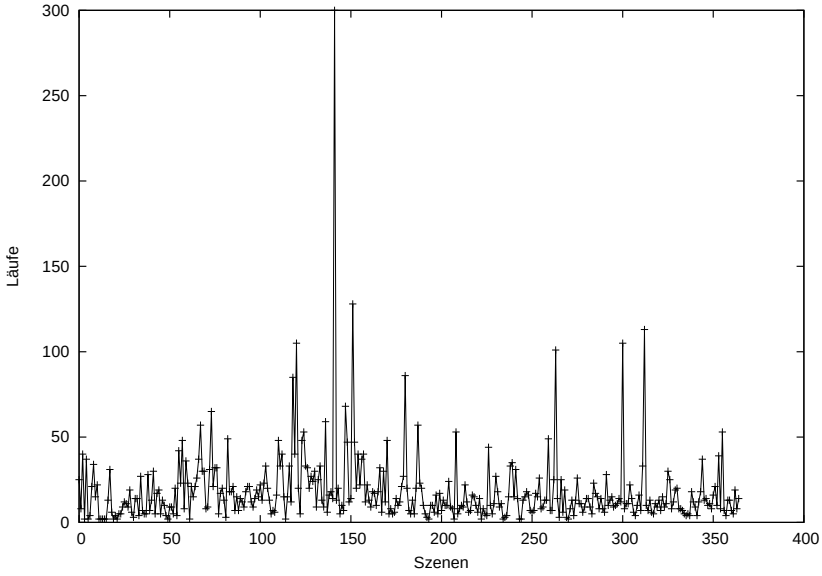


Abbildung 5.14: Mulitpassverfahren, Wolkenmasken, Anzahl der benötigten Filterläufe

[44] −90 deg gedrehtes Diagonal-2 Bild (aufgrund der spaltenweiser Verarbeitung der Bildpunkte)

Wird davon ausgegangen, dass 100 FPS bei Bildern der Größe 1000*1000 Pixel erreicht werden können, werden für die 300 Läufe des 8681*7941 großen realen Testbildes ungefähr 3:30 Minuten benötigt. Dies ist aber bei weitem nicht der schlimmste anzunehmende Fall, der auftreten kann. Für das Labyrinth-2 Bild dieser Größe würden ungefähr 33 Minuten, und für das Diagonal-2* Bild ungefähr 94 Minuten benötigt.

Laufzeitverhalten von Two-Pass Verfahren

Als nächstes wurde ein Two-Pass Verfahren untersucht, welches in [62] und [77] Anwendung fand. Der Grund hierfür war, dass dieses Verfahren in diesen Arbeiten relativ gut beschrieben ist und fast ohne Modifikationen auf multimodales Labeling erweitert werden konnte. Der Pseudo-Programmcode dieses Algorithmus, in welchem auch die Speicherzugriffe markiert sind, findet sich in Anhang Abbildung 7.4.
Anhand der Anzahl der Speicherzugriffe auf den Arbeitsspeicher wurde abgeschätzt, wie viele Takte eine FPGA-Implementation für das Labeling eines Bildes benötigt[45].
In Abbildung 5.15 und 5.16 finden sich die Ergebnisse der Laufzeituntersuchung für die synthetischen beziehungsweise realen Testdaten.

Syn. Testbilder	Durchschnittliche Anzahl der Takte pro Pixel						
Bildhöhe/Breite in Pixel	50	512	1024	2048	4096	8192	16384
Weiß	4,00	4,00	4,00	4,00	4,00	4,00	4,00
Dreieck	4,38	4,38	4,38	4,38	4,38	4,38	4,38
Diagonal 1	264	$322 * 10^2$	$130 * 10^3$	$522 * 10^3$	$209 * 10^4$	-	-
Diagonal 2	4,00	4,00	4,00	4,00	4,00	4,00	4,00
Labyrinth 1	14	892	$359 * 10^1$	$145 * 10^2$	$580 * 10^2$	-	-
Labyrinth 2	14	1025	$407 * 10^1$	$164 * 10^2$	$657 * 10^2$	-	-

Abbildung 5.15: Durchschnittliche Anzahl der Takte pro Bildpunkt des auf [62] basierten Two-Pass Verfahrens (4 Zusammenhang), welche für die synthetischen Testbilder benötigt werden. Bei „-“ wurde aufgrund der hohen Laufzeit des Algorithmus die Messung abgebrochen.

Das Ergebnis ist eindeutig, dieses Two-Pass Verfahren ist nicht für größere, komplexere Bilder geeignet. Es können extrem hohe Laufzeiten auftreten. Dies liegt daran, dass zur Auflösung eines jeden Konfliktfalles die gesamte Lookup-Tabelle durchlaufen werden muss[46]. Die Laufzeit summiert sich dann sehr schnell, gerade wenn sehr viele Konfliktfälle auftreten und die Lookup-Tabelle Tabelle sehr viele Einträge besitzt.
Wird angenommen, dass $100 * 10^6$ (100 MHz) Takte pro Sekunde ausgeführt werden können, wird für das weiße synthetische Testbild der Größe 4096*4096 Pixel ungefähr 0,7 Sekunden benötigt. Für das Labyrinth-2 Testbild werden jedoch 87 Stunden und für das Diagonal-1 Testbild sogar 16 Tage benötigt. Aber auch für das reale Testbild werden im

[45] Bemerkung: Es wurde hierbei davon ausgegangen, dass pro Takt ein Speicherzugriff erfolgen kann. Für eine erste Abschätzung scheint diese Annahme gerechtfertigt.
[46] vgl. Programmzeile 69-72, Abbildung 7.4

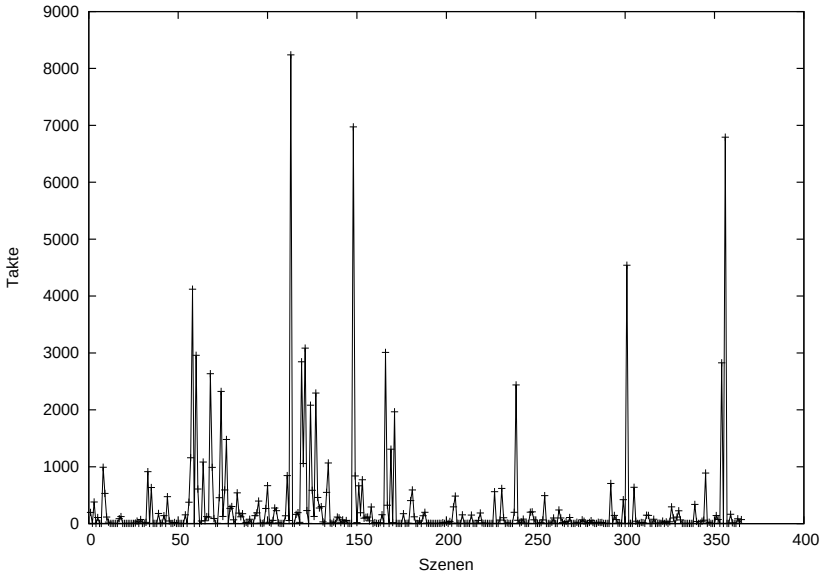


Abbildung 5.16: Two Pass Klassik, 4 Zusammenhang, Wolkenmasken, Durchschnittliche Anzahl der Takte pro Pixel

schlimmsten Fall, welcher im Test aufgetreten ist, durchschnittlich 8000 Takte pro Pixel benötigt. Das heißt, dass der Labelvorgang für das 8681*7941 Pixel große Testbild erst nach ungefähr 91 Minuten abgeschlossen ist.
Daher ist ein weiteres Two-Pass Verfahren untersucht worden, welches zur Auflösung von Konfliktfällen Union-Find Datenstrukturen verwendet[47]. Wie in Abbildung 5.17 und 5.18 zu erkennen ist, ist dieses Verfahren geeignet, um große und komplexe Bilder zu verarbeiten.

Syn. Testbilder	Durchschnittliche Anzahl der Takte pro Pixel						
Bildgröße in Pixel (quadratisch)	50	512	1024	2048	4096	8192	16384
Weiß	4,00	4,00	4,00	4,00	4,00	4,00	4,00
Dreieck	4,07	4,01	4,00	4,00	4,00	4,00	4,00
Diagonal 1	5,77	5,98	5,98	6,00	5,99	6,00	6,00
Diagonal 2	4,00	4,00	4,00	4,00	4,00	4,00	4,00
Labyrinth 1	4,34	4,34	4,35	4,35	4,35	4,35	4,35
Labyrinth 2	4,35	4,38	4,38	4,38	4,38	4,39	4,39

Abbildung 5.17: Durchschnittliche Anzahl der Takte pro Pixel eines auf Union-Find Datenstrukturen basierenden Two-Pass Verfahrens (4-Zusammenhang), welche für die synthetischen Testbilder benötigt werden.

[47] im Grunde entspricht das Verfahren dem in [53] beschrieben Verfahren von Rosenfeld (Abschnitt 2.2.1)

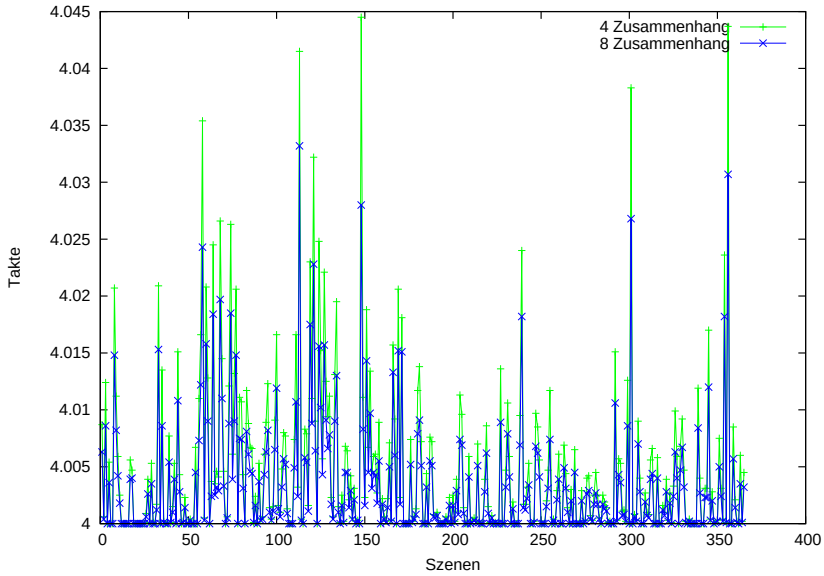


Abbildung 5.18: Durchschnittliche Anzahl der Takte pro Bildpunkt eines auf Union-Find Datenstrukturen basierenden Two-Pass Verfahrens, welche für die realen Testbilder benötigt werden.

Für das Labeling werden für ein Bild der Größe 8192*8192 Pixel maximal 4.1 Sekunden benötigt, falls $100 * 10^6$ Takte pro Sekunde (100 MHz) ausgeführt werden können. Für ein Bild der Größe 1024*1024 werden maximal 0,063 Sekunden (15 FPS) benötigt.
Diese Werte werden aber nur erreicht, wenn bei jeden Takt ein Speicherzugriff ausgeführt werden kann. Dies bedeutet, dass alle Operationen des Algorithmus parallel zu den Speicherzugriffen durchgeführt werden müssen. Um dies zu realisieren, muss eine geeignete Darstellungsform für die Union-Find Datenstruktur gefunden werden. In Abschnitt 5.5 wird eine Möglichkeit vorgestellt.

Laufzeit eines Region-Growing Verfahrens

Zum Schluss wurde ein einfaches Region-Growing Verfahren untersucht[48]. Aber wie bereits angedeutet, stellte sich bereits bei der Softwareimplementation heraus, dass Region-Growing Verfahren aufgrund ihrer vielen dynamischen Speicherzugriffe nicht besonders gut für ein FPGA-Implementation geeignet sind. Trotzdem wurde die Untersuchung fortgeführt, um darzustellen wie Region-Growing Verfahren im Vergleich zu den anderen Verfahren abschneiden. Der Pseudo-Programmcode des Algorithmus ist in der Abbildung 7.5 zu finden.
Wie bei den Two-Pass Verfahren wurde anhand der Anzahl Speicherzugriffe auf den

[48] dieses Verfahren beruht auf dem Single-Linkage Region Growing Verfahren ([13], S.525) und wurde modifiziert um multimodales Labeling durchführen zu können.

Arbeitsspeicher abgeschätzt wie viele Takte eine FPGA-Implementation für das Labeling eines Bildes benötigt. Allerdings ist diese Schätzung sehr optimistisch und es kann eher davon ausgegangen werden, dass eine reale Umsetzung doppelt so viel Takte benötigt[49]. Die Ergebnisse, welche in Abbildung 5.19 und 5.20 dargestellt sind, zeigen, dass große komplexe Bilder durchaus mit einem Region-Growing-Verfahren bearbeitet werden können. Da aber weitaus mehr Takte als bei dem Two-Pass Verfahren (beruhend auf Union-Find Datenstrukturen) benötigt werden und aufgrund der voraussichtlich sehr aufwendigen Portierung auf ein FPGA, wurde von der weiteren Verwendung abgesehen.

Syn. Testbilder	Durchschnittliche Anzahl der Takte pro Pixel						
Bildhöhe/breite in Pixel	50	512	1024	2048	4096	8192	16384
Weiß	9,92	9,99	10,00	10,00	10,00	10,00	10,00
Dreieck	9,86	9,99	9,99	10,00	10,00	10,00	10,00
Diagonal 1	7,90	7,99	8,00	8,00	8,00	8,00	8,00
Diagonal 2	5,50	5,50	5,50	5,50	5,50	5,50	5,50
Labyrinth 1	8,46	8,55	8,54	8,54	8,54	8,54	8,54
Labyrinth 2	8,44	8,53	8,52	8,52	8,52	8,53	8,53

Abbildung 5.19: Durchschnittliche Anzahl der Takte pro Pixel des Region-Growing Verfahrens (4-Zusammenhang), welche für die synthetischen Testbilder benötigt werden.

5.5 Ein Two-Pass Algorithmus basierend auf Union-Find Datenstrukturen

In der Voruntersuchung welche in den vorangegangenen Abschnitten durchgeführt wurde, hat sich herausgestellt, dass Two-Pass CCL-Verfahren basierend auf Union-Find Datenstrukturen die beste Wahl für eine FPGA-Umsetzung darstellen[50].
Es hat sich herausgestellt, dass mit der Union-Find Datenstruktur eine schnelle Verarbeitung, auch komplexer Bilder, möglich ist. Die Herausforderung liegt insbesondere darin, eine geeignete Darstellungsweise der in diesem Verfahren verwendeten Union-Find Datenstruktur zu finden, welche gut für eine FPGA Umsetzung geeignet ist.

5.5.1 Herausforderungen

Wie bereits in Abschnitt 5.3.2 beschrieben, werden bei klassischen Two-Pass Verfahren drei Verarbeitungsschritte durchgeführt. Zuerst wird ein erstes Labeling[51] durchgeführt,

[49] Aufgrund der hohen Dynamik des Verfahrens ist es schwierig Rechenoperationen hinter Speicherzugriffen zu verstecken
[50] in Hinblick auf die in Abschnitt 5.4.1 gestellten Anforderungen
[51] i.a.W. der erste Lauf(Pass)

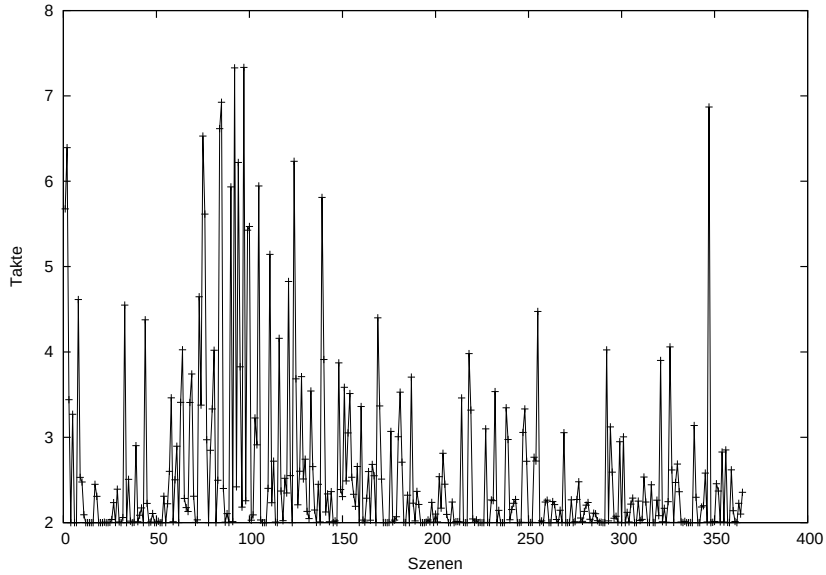


Abbildung 5.20: Durchschnittliche Anzahl der Takte pro Pixel des Region-Growing Verfahrens (4-Zusammenhang), welche für die realen Testbilder benötigt werden.

bei dem den Bildpunkten vorläufige Indizes zugeordnet werden. Während dieses Prozesses kann es jedoch vorkommen, dass zwei verschiedenen Indizes der gleichen Zusammenhangskomponente zugeordnet werden (Labeling-Konflikt). Die Zusammengehörigkeit dieser Indizes wird in einer Äquivalenztabelle[52] vermerkt. Mithilfe dieser Äquivalenztabelle wird nach Durchführung des ersten Laufes eine Lookup-Tabelle erstellt, bei welcher jedem vorläufigen Label ein finales Label zugeordnet ist. Schließlich werden mit Hilfe dieser Lookup-Tabelle im zweiten Lauf allen Bildpunkten ihre finalen Labels zugeordnet[53]. Mit einer geschickten Verwendung von Union-Find Datenstrukturen kann dieser Schritt aber in ausreichend kurzer Zeit durchgeführt werden.

Ein kritischer Punkt in diesem Zusammenhang ist die Größe der Äquivalenztabelle und der Lookup-Tabelle. Diese Größe ist abhängig von der maximalen Anzahl der Labels[55] welche auftreten kann. Im allgemeinen Fall[56] kann jeder Bildpunkt eine eigene Zusammenhangskomponente darstellen. Es werden folglich maximal **Bildhöhe** $*$ **Bildbreite**

[52] auch Merge-Tabelle genannt

[53] Die Lookup-Tabelle, kann auch während der ersten Phase erstellt und aktualisiert werden. Eine Äquivalenztabelle wird dann nicht benötigt. Das (klassische) Two-Pass Verfahren, welches in der Voruntersuchung (Abschnitt 5.4.3) analysiert wurden, führt das Labeling auf diese Weise durch[54]. Die größte Herausforderung stellt die Erstellung der Lookup-Tabelle dar. Beim ersten näher untersuchten Two-Pass Verfahren, wird, falls viele Konfliktfälle auftreten, sehr viel Zeit benötigt, um dies durchzuführen (Abschnitt5.4.3

[55] i.a.W der maximalen Anzahl an Zusammenhangskomponenten

[56] Abschnitt 5.2

Labels benötigt. Die Bit-Tiefe eines Labels muss daher mindestens

$$\lceil \log_2(\text{Bildhöhe} * \text{Bildbreite}) \rceil \text{ Bit}$$

betragen[57].
Die maximale Anzahl der Konfliktfälle, welche auftreten können, ist:

$$0,5 * \text{Bildhöhe} * \text{Bildbreite}$$

Sie wird beim 4-Zusammenhang beim Diagonal-1 Testbild und beim 8-Zusammenhang beim Diagonal-2-Testbild erreicht[58]. In der Äquivalenztabelle müssen für jeden Konfliktfall die zwei zusammengehörigen Labels gespeichert werden[59]. Die Äquivalenztabelle hat daher eine maximale Größe von:

$$2 * 0,5 * \text{Bildhöhe} * \text{Bildbreite} * \lceil \log_2(\text{Bildhöhe} * \text{Bildbreite}) \rceil \text{ Bit}$$

Dies entspricht genau der Größe der Lookup-Tabelle, in welcher für jedes potentiell auftretende Label einen Eintrag vorgehalten werden muss[60]:

$$\text{Bildhöhe} * \text{Bildbreite} * \lceil \log_2(\text{Bildhöhe} * \text{Bildbreite}) \rceil \text{ Bit}$$

Dies entspricht außerdem genau der Größe des Labelmaske, bei welchem jedem Bildpunkt ein Index zugeordnet wird.
Die Äquivalenztabelle, die Lookup-Tabelle und die Labelmaske haben demzufolge einen gleich hohen Speicherverbrauch. In Abbildung 5.21 ist der benötigte Speicherverbrauch dargestellt, in Abhängigkeit einiger ausgewählter Bildgrößen.

Bildhöhe/breite	50	512	1024	2048	4096	8192	16384
Größe in kBit	30	4719	20972	92275	402654	1744831	7516193
Größe in MB	0,004	0,6	2,7	12	51	219	940
Bittiefe in Bit	12	18	20	22	24	26	28

Abbildung 5.21: Größe der Äquivalenztabelle, der Lookup-Tabelle und der Labelmaske, Bildhöhe/breite in Pixel, Werte gegebenenfalls aufgerundet

Alle bekannten Two-Pass-Verfahren welche auf FPGA/Hardware umgesetzt wurden, benötigen eine Äquivalenztabelle und/oder eine Lookup-Tabelle und eine Labelmaske. Der benötigte Speicherverbrauch wächst bei größeren Bildern stark an. Dies ist insbesondere ungünstig, da es bei FPGA-Implementationen sinnvoll ist, wenn auf die Äquivalenztabelle, die Lookup-Tabelle und die Labelmaske unabhängig voneinander zugegriffen werden kann[61]. Dies ermöglicht eine höhere Verarbeitungsgeschwindigkeit und vereinfacht die Implementation.

[57] $\lceil . \rceil$ ist die Aufrundungsfunktion
[58] Abbildung 5.10
[59] Abbildung 5.9
[60] dies entspricht der maximalen Anzahl an Label die auftreten können
[61] um eine parallele Datenverarbeitung zu ermöglichen

Um dies zu ermöglichen wurden in [78] die Äquivalenztabelle und die Lookup-Tabelle im Block-RAM des FPGAs untergebracht. Aber obwohl nur ein binäres Labeling durchgeführt wurde und trotz der relativ kleiner Bildgrößen[62], war dies nur durch Einschränkungen möglich[63]. Es konnten nur Bilder, bei denen eine beschränkte Anzahl an Konfliktfällen[64] und Labels[65] auftraten, fehlerfrei bearbeitet werden.
Derartige Einschränkungen können aufgrund des Anforderungsprofils nicht hingenommen werden. Es wurde daher ein anderer Ansatz gewählt. Das im Folgenden vorgestellte Verfahren benötigt zur Durchführung des CCL nur die ohnehin benötigte Labelmaske. Dies spart wertvollen Speicherplatz, und es kann ohne Ausnahmen jeder Fall bearbeitet werden[66].

5.5.2 Der Algorithmus

Im Folgenden wird das im Rahmen dieser Arbeit entwickelte, speziell an die Anforderungen angepasste Two-Pass-Single-Storage-Verfahren (TPSS) vorgestellt.

Grundlegende Begriffe

Zur Beschreibung des TPSS-Verfahrens werden im Folgenden einige grundlegende Begriffe eingeführt. Eine genaue Beschreibung des Bildbegriffes, der hierzu verwendet wird, ist in Abschnitt 4.2.1 zu finden.
Um das Verfahren beschreiben zu können, müssen zuerst die Ein- und Ausgangsdaten[67] spezifiziert werden:

- $\Omega = [0, h]_{\mathbb{N}} \times [0, b]_{\mathbb{N}}$
- $\mu_{zsh} : \Omega \to [0, 255]_{\mathbb{N}}$
- $\mu_{addr} : \Omega \to \Omega$
- $\mu_{index} : \Omega \to [1, \#\Omega]_{\mathbb{N}}$[68]

Mit Ω wird der Ortsraum eines Bildes bezeichnet, wobei h die Bildhöhe und b die Bildbreite in Pixel ist[69].

[62] 640*480 Bildpunkte
[63] der Grund hierfür ist, dass auch auf 2014 verfügbaren FPGAs nur wenige MB an Block-RAM zur Verfügung stehen (bei der Low-Mid-FPGAs wie Xilinx-Spartan-6 Reihe maximal 0,5 MB [41], bei Hochleistungs-FPGAs wie der Xilinx-Virtex-7 Reihe maximal 68 MB [79].
[64] die Anzahl wurde nicht genannt
[65] maximal 4095 Labels
[66] i.a.W allgemeines Labeling kann durchgeführt werden
[67] in diesem Fall Ein- und Ausgangsbilder
[68] Mit $\#\Omega$ wird die Anzahl der Segmente von Ω bezeichnet, in diesem Fall also die Anzahl der Pixel des Bildes.
[69] da das Intervall bei 0 beginnt hat das Bild tatsächlich eine Höhe bzw. eine Breite von $h+1$ und $b+1$. Aufgrund der Übersichtlichkeit wird in Zusammenhang mit h und b trotzdem von Bildhöhe bzw. Bildbreite gesprochen.

Die lokalen Nachbarschaftsbeziehungen[70] sind im (lokalen) Zusammenhangsbild μ_{zsh} kodiert (Abschnitt 3.3). Auf deren Grundlage wird das Labeling durchgeführt. Die Nachbarschaftsbeziehungen werden gewöhnlicherweise durch eine im Vorfeld durchgeführte Bildsegmentation ermittelt (Abschnitt 3.1).
Das Ergebnis des 1-Laufes ist die vorläufige Labelmaske μ_{addr}. Das Spezielle an diesem Verfahren ist jedoch, dass hier keine vorläufigen Labels abgespeichert werden, sondern die Position eines ausgezeichneten Bildpunktes einer Zusammenhangskomponente. Dieser ausgezeichnete Punkt ist der erste Punkt einer Zusammenhangskomponente, welcher gelabelt wurde.
Das finale Ergebnis beim Labeling ist die Labelmaske μ_{index}, bei welcher jeder Zusammenhangskomponente ein Index zugeordnet ist, und jeder Bildpunkt mit dem Index seiner Zusammenhangskomponente indiziert ist.
Zur Übersicht folgt eine formale Beschreibung des Labelvorgangs.Der gesamte Labelprozess

$$TPSS : \mu_{zsh} \mapsto \mu_{index}$$

unterteilt sich bei TPSS in genau zwei Teilschritte, den ersten und den zweiten Lauf.

$$\textit{1-Lauf} : \mu_{zsh} \mapsto \mu_{addr}$$

$$\textit{2-Lauf} : \mu_{addr} \mapsto \mu_{index}$$

Es sei hier bemerkt, dass zur Speicherung der Bilder μ_{addr} und μ_{index} die gleiche Datenstruktur[71] verwendet werden kann[72]. Beim zweiten Lauf kann μ_{addr} durch μ_{index} ersetzt werden. Sie können sich damit einen gemeinsamen Speicherbereich teilen. Eine Äquivalenztabelle und eine Look-Up Tabelle werden beim TPSS-Verfahren nicht benötigt.
Ein wichtiger Punkt ist, wie bereits angesprochen wurde, eine für die Prozessierung geeignete Datenstruktur zu finden, mit welcher der Zusammenhang der Bildpunkte möglichst effizient ermittelt und dargestellt werden kann. Es wird eine Union-Find Datenstruktur verwendet, die sich im Allgemeinen sehr gut zur Verwaltung der Zusammenhangskomponenten eines Graphen eignet. Der entscheidende Punkt ist, diese Datenstruktur möglichst effizient zu implementieren.
Zur Darstellung der Union-Find Datenstruktur werden verkettete Listen verwendet, die im Folgenden dargestellt werden.

Definition 21 *Sei $P \in \Omega$ ein Bildpunkt. Dann ist auf natürliche Weise auf μ_{addr} eine Folge $(P_n)_{n\in\mathbb{N}}$ mit Startpunkt P gegeben.*

$$P_0 := P,\ P_{n+1} := \mu_{addr}(P_n)$$

[70] lokaler Zusammenhang
[71] ein zweidimensionales Array
[72] da $\#\Omega = \#[1, \#\Omega]_{\mathbb{N}}$

Ausgeschrieben sieht dies folgendermaßen aus:

$$\mathbf{P_0} := P \longmapsto \mathbf{P_1} := \mu_{addr}(P) \longmapsto \mathbf{P_2} := \mu_{addr}(P_1) \longmapsto \mathbf{P_3} := \mu_{addr}(P_2) \longmapsto ...$$

Es sei bemerkt, dass eine derartige Folge immer unendlich viele Folgeglieder hat. Es gibt aber ein Äquivalent zu endlichen Folgen.

Definition 22 *Eine Folge $(P_n)_{n\in\mathbb{N}}$, deren Glieder ab einem bestimmten Glied übereinstimmen, d.h.*

$$\exists n \in \mathbb{N} \quad P_{n+i} = P_n \quad \forall i \geq 0$$

wird stationäre Folge genannt. Das kleinste $n \in \mathbb{N}$ für das dies gilt, wird Tiefe der stationären Folge genannt.

Bemerkung 9 *Sei $P \in \Omega$ ein Bildpunkt, dessen Folge auf μ_{addr} stationär ist. Das erste stationäre Element dieser Folge wird mit $head(P)$ bezeichnet.*

Der Kopf einer endlich verketteten Liste auf μ_{addr} kann auch folgendermaßen rekursiv definiert werden (C-Syntax):

$$head(P) = (\mu_{addr}(P) == P) \ ? \ P : \ head(\mu_{addr}(P))$$

Es sei aber bemerkt, dass durch Zyklusbildung nicht stationäre Folgen auf μ_{addr} auftreten können. Beispielsweise:

$$P_0 \longmapsto P_1 \longmapsto P_2 \longmapsto P_3 \longmapsto P_0 \longmapsto P_1 \longmapsto ...$$

Zyklen dürfen bei TPSS-Verfahren allerdings nicht vorkommen.
Um den Algorithmus angeben zu können, wird eine Ordnungsrelation auf dem Ortsraum eines Bildes benötigt.

Definition 23 *Seien $\Omega = [0,h]_{\mathbb{N}} \times [0,b]_{\mathbb{N}}$ und $P = (P_x, P_y),\ Q = (Q_x, Q_y) \in \Omega$ zwei Punkte. Dann ist*

$$P < Q :\Longleftrightarrow P_x < Q_x \text{ oder } (P_x = Q_x \text{ und } P_y < Q_y)$$

Hierdurch wird auch das Maximum $max(P,Q)$ und Minimum $min(P,Q)$ definiert.

Erster Lauf des TPSS-Verfahrens

Im diesem Abschnitt wird der Algorithmus für den ersten Lauf des TPSS-Verfahrens

$$\textit{1-Lauf}: \mu_{zsh} \mapsto \mu_{addr}$$

angegeben. Der lokale Zusammenhang ist durch μ_{zsh} gegeben. Der Zusammenhang zweier benachbarter Punkte $P, Q \in \Omega$ wird folgendermaßen dargestellt:

$$con(P,Q) := \begin{cases} 1 & \text{falls P, Q zusammenhängend} \\ 0 & \text{sonst} \end{cases}$$

Sei $M \subset \Omega$ eine Menge von Punkten. Dann wird die Menge aller mit P zusammenhängenden Punkte der Menge M folgendermaßen bezeichnet:

$$con_M(P) := \{Q \in M \mid con(P,Q) = 1\} \subset \Omega$$

Im weiteren Verlauf entspricht M den Umgebungsmasken, welche in Abbildung 5.8 graphisch dargestellt sind. Jede nachdem, ob nach 4- oder 8-Zusammenhang gelabelt wird, muss eine andere Maske verwendet werden.

$$M_4 = \{P_{oben}, P_{links}\}, \quad M_8 = \{P_{\text{oben}}, P_{links}, P_{oben-links}, P_{oben-rechts}\}$$

Bei der Ausführung des Algorithmus kann es im Randbereich des Ortsraumes Ω zu Unstimmigkeiten kommen, falls ein Bildpunkt, der angesprochen werden soll, nicht mehr im Bildbereich Ω liegt. Daher wird sowohl für den ersten als auch zweiten Lauf folgende Vereinbarung getroffen.

Bemerkung 10 (Konvention (Rand))

$$con(P, P_\square) := 0 \quad \textit{falls } P_\square \notin \Omega$$

Dabei sind mit $P_\square$ alle Nachbarn von P gemeint, also
$\square \in \{$*oben, unten, links, rechts, oben-links, oben-rechts, unten-links, unten-rechts*$\}$.

Die Algorithmen werden außerdem zeilenweise, von links-oben nach rechts-unten ausgeführt (Abbildung 5.8).

Definition 24 (Raster) *Sei $\Omega = [0, h-1]_{\mathbb{N}} \times [0, b-1]_{\mathbb{N}}$ und h die Bildhöhe und b die Bildbreite.*[73] *Dann sei $(P_i)_{i \in \mathbb{N}} \in \Omega$ eine Folge von Punkten derart, dass gilt:*

$$P_i = ((P_i)_x, (P_i)_y), \quad i = (P_i)_x * (h+1) + (P_i)_y \quad \text{für} \quad i = 0, ..., (b*h) - 1$$

Jeder Punkt in Ω entspricht dann genau einem P_i.

Im ersten Lauf wird μ_{addr} berechnet. Die Punkte $(P_i)_{i=0}^{b*h-1}$ werden nacheinander bearbeitet. Für jeden Punkt werden folgende Operationen ausgeführt:

Fall 1 - Falls $con_M(P) = \emptyset$:
$\mu_{addr}(P) := P$

Fall 2 - Falls $con_M(P) \neq \emptyset$:
$\mu_{addr}(P) := min\{head(Q) \mid Q \in con_M(P)\}$
$\forall Q \in con_M(P) \quad \mu_{addr}(head(Q)) := min\{head(Q) \mid Q \in con_M(P)\}$

Abbildung 5.22: Erster Lauf des TPSS-Verfahrens

Für M ist je nach gewünschtem Zusammenhang, wie bereits erwähnt, entweder die Maske M_4 oder M_8 zu verwenden. Der Algorithmus für den 4-Zusammenhang kann, wie in Abbildung 5.23 gezeigt, dargestellt werden.

Algorithmus Pass 1 (4 Zusammenhang):

Fall 1 - Falls $con(P, P_{links}) = con(P, P_{oben}) = 0$:
$\mu_{addr}(P) := P$

Fall 2 - Falls $con(P, P_{links}) = 1$ und $con(P, P_{oben}) = 0$:
$\mu_{addr}(P) := addr(P_{links})$

Fall 3 - Falls $con(P, P_{links}) = 0$ und $con(P, P_{oben}) = 1$:
$\mu_{addr}(P) := addr(P_{oben})$

Fall 4 - Falls $con(P, P_{links}) = con(P, P_{oben}) = 1$:
$\mu_{addr}(P) := min(head(P_{links}), head(P_{rechts}))$
$\mu_{addr}(head(P_{links})) := min(head(P_{links}), head(P_{oben}))$
$\mu_{addr}(head(P_{oben})) := min(head(P_{links}), head(P_{oben}))$

Abbildung 5.23: Erster Lauf des TPSS-Verfahrens bezüglich des 4-Zusammenhangs

Zweiter Lauf des TPSS-Verfahrens

Im zweiten Lauf wird mit Hilfe des Ergebnisses des ersten Laufes[74], das finale Indexbild μ_{index} erstellt:

$$\textit{2-Lauf}: \mu_{addr} \mapsto \mu_{index}$$

Der Algorithmus ist in Abbildung 5.24 abgebildet. Die Bildpunkte werden, wie beim ersten Lauf, zeilenweise von links-oben nach rechts-unten bearbeitet (Definition 24). Für

[73] Bemerkung: Hier ist die tatsächliche Bildhöhe und Bildbreite von Bedeutung.
[74] μ_{addr}

jeden Punkt werden folgende Operationen ausgeführt. **index** := **0** ist hierbei die Index-Variable, die vor Beginn des zweiten Laufes auf 0 gesetzt wird. Außerdem sei erinnert, dass die Konvention (Rand - Bemerkung 10) auch hier herangezogen wird.

Fall 1 - Falls $\mu_{addr}(P) = P$:
$\mu_{index}(P) := index$
$index := index + 1$

Fall 2 - Falls $\mu_{addr}(P) \neq P$:
$\mu_{index}(P) := \mu_{index}(\mu_{addr}(P))$

Abbildung 5.24: Zweiter Lauf des TPSS-Verfahrens

Dieser Algorithmus ist sowohl für den 4- als auch 8-Zusammenhang gültig. Nach Abschluss des zweiten Laufes ist das TPSS-Verfahren abgeschlossen.

5.5.3 Wohldefiniertheit des TPSS-Algorithmus

In diesem Abschnitt wird die Wohldefiniertheit des Algorithmus überprüft. Das heißt, dass jede Operation ordnungsgemäß, in endlich vielen Schritten ausgeführt werden kann. Im Folgenden wird gezeigt, dass der erste Lauf wohldefiniert ist, und dass er von einer eventuell bestehenden Vorbelegung von μ_{addr} unabhängig ist. Das heißt, dass μ_{addr} nicht initialisiert werden muss.

Satz 5 (Wohldefiniertheit des ersten Laufs des TPSS-Verfahrens)

1. *Wird der Algorithmus in der Reihenfolge (Raster)*[75] *ausgeführt und wird Konvention (Rand)*[76] *befolgt, ist er wohldefiniert. Dies bedeutet, dass für alle $P \in \Omega$ gilt, dass $\mu_{addr}(P) \in \Omega$ und $head(P) \in \Omega$. Dies schließt die Wohldefiniertheit von $head(P)$ mit ein, also dass P der Startpunkt eine stationäre Kette in μ_{addr} ist.*
2. *Für alle $P \in \Omega$ gilt $\mu_{addr}(P) \leq P$ und $\mu_{addr}(P_0) = P_0$*
3. *Der Algorithmus ist von der Anfangsbelegung von μ_{addr} unabhängig.*

Beweis: Dieser Satz wird für den 4-Zusammenhang bewiesen. Für den 8-Zusammenhang ist der Beweis analog durchführbar. Die erste und zweite Aussage werden durch Induktion bewiesen.

[75] Definition 24
[76] Bemerkung 10

Beweis Aussage 1 und 2:
Induktionsvoraussetzung: Sei $i < b * h$. Es muss gezeigt werden, dass unter der Voraussetzung, dass für alle P_j mit $j < i$, $\mu_{addr}(P_j) \in \Omega$, $head(P_j) \in \Omega$ und $\mu_{addr}(P_j) \leq P_j$, dies auch für P_j mit $j = i$ gilt. Hierzu werden die verschiedenen Fälle des Algorithmus einzeln untersucht. Es ist leicht einzusehen, dass immer nur genau ein Fall eintritt.
Induktionsschritt:
Zum ersten Fall: P_i wird ab jetzt mit P bezeichnet. Da $P \in \Omega$ gilt natürlich auch $\mu_{addr}(P) := P \in \Omega$. Außerdem ist $P \rightarrow addr(P) := P$ eine endliche Kette und daher $head(P) = P \in \Omega$.
Zum zweiten Fall: Da $P_{links} < P$, gilt nach Induktionsvoraussetzung $\mu_{addr}(P_{links}) \in \Omega$ und $head(P_{links}) \in \Omega$. Also gilt dies auch für $\mu_{addr}(P) := \mu_{addr}(P_{links})$. Außerdem ist $P > P_{links} > \mu_{addr}(P_{links})$.
Zum dritten Fall: Der dritte Fall verhält sich genauso wie der zweiten Fall.
Nun zum vierten Fall: Da $P_{links} < P$ und $P_{oben} < P$, folgt aus der Induktionsvoraussetzung $\mu_{addr}(P_{links})$ bzw. $\mu_{addr}(P_{oben}) \in \Omega$ und $head(P_{links})$ bzw. $head(P_{oben}) \in \Omega$.
Dann ist auch $\mu_{addr}(P) := min(head(P_{links}), head(P_{oben})) \in \Omega$,
$\mu_{addr}(head(P_{links})) := min(head(P_{links}), head(P_{oben})) \in \Omega$ und
$\mu_{addr}(head(P_{oben})) := min(head(P_{oben}), head(P_{oben})) \in \Omega$.
Da $head(min(head(P_{links}), head(P_{oben})) = min(head(P_{links}), head(P_{oben})) \in \Omega$ ist auch $head(\mu_{addr}(P)) \in \Omega$, $head(\mu_{addr}(head(P_{links}))) \in \Omega$ und $head(\mu_{addr}(head(P_{oben}))) \in \Omega$.
Damit ist der Induktionsschritt gezeigt.
Da außerdem $\mu_{addr}(P) := min(head(P_{links}), head(P_{oben})) \leq head(P_{links})$ bzw. $head(P_{oben})$, und $head(P_{links}) \leq P_{links} < P$ und $head(P_{oben}) \leq P_{oben} < P$, ist auch $\mu_{addr}(P) \leq P$.
Damit ist der Induktionsschritt zweite Aussage des Satzes gezeigt.
Beweis Induktionsanfang: Die Gültigkeit des Induktionsanfangs wird durch eine genauere Untersuchung des Punktes $P_0 = (0, 0)$ gezeigt. Nach der oben getroffenen Konvention ist $con(P_0, P_{oben}) = con(P_0, P_{links}) = 0$. Es tritt also der erste Fall ein, und daher ist $\mu_{addr}(P_0) = P_0 = (0, 0) \in \Omega$. Hiermit ist der Induktionsbeweis beendet, und die Aussagen 1 und 2 gezeigt.

Beweis Aussage 3:
Da in allen Schritten nur auf Einträge in μ_{addr} zugegriffen wurde, welche in einem vorherigen Schritt schon gesetzt wurden, trifft auch diese Behauptung zu. □

Der zweite Lauf ist wohldefiniert, wenn die folgenden Voraussetzungen zutreffen:

A) $\forall P \in \Omega \ \mu_{addr}(P) \leqq P$

B) $\mu_{addr}(P_0) = \mu_{addr}((0, 0)) = (0, 0)$

Diese sind nach erfolgreicher Durchführung des ersten Laufes erfüllt (Satz 5).

Satz 6 (Wohldefiniertheit des zweiten Laufs des TPSS-Verfahrens)

a) Gelten die obigen Voraussetzungen, und wird der Algorithmus in Reihenfolge (Raster)[77] *durchlaufen, ist der Algorithmus wohldefiniert und unabhängig von der Vorbelegung von* μ_{index}.

b) Die Indizierung weißt keine Lücken auf, das heißt:

$$\exists max \in \mathbb{N} \quad so\ dass \quad \mu_{index}(\Omega) = [0, max]_{\mathbb{N}}$$[78]

Insbesondere ist max *der größte Index ist, welcher auftritt.*

Beweis: Für den Startpunkt P_0 tritt aufgrund der Voraussetzung B der erste Fall des Algorithmus ein[79]. Aufgrund der Voraussetzung A und der Verarbeitungsreihenfolge (Raster) wird im zweiten Fall nur auf Indizes zugegriffen, welche in einem früheren Schritt schon gesetzt wurden. Daraus ergibt sich die Wohldefiniertheit und die Unabhängigkeit von der Vorbelegung von μ_{index}.
Aussage B folgt aus der Tatsache, dass die Indizierung bei 0 beginnt, dass neue Index-Werte nur im ersten Fall gesetzt werden, und hierbei die Variable *index* genau um eins erhöht wird. □

5.5.4 Semantik des TPSS-Algorithmus

Im letzten Abschnitt wurde gezeigt, dass der TPSS-Algorithmus wohldefiniert ist, also fehlerfrei ausgeführt werden kann. Jetzt wird gezeigt, dass er tut, was er tun soll.

Satz 7 *Es gelten die gleichen Voraussetzungen wie in Satz 5*[80]*. Seien* $P, Q \in \Omega$*. Dann gilt nach dem Abschluss des ersten Laufs:*

$$head(P) = head(Q) \Longleftrightarrow P,\ Q\ zusammenhängend$$

Beweis: Dieser Satz wird für den 4-Zusammenhang bewiesen. Für den 8-Zusammenhang kann er analog durchgeführt werden. Zuerst wird gezeigt, dass für alle $P \in \Omega$ die Punkte

[77] Definition 24
[78] allgemein gilt: $\exists max, min \in \mathbb{N}$ so dass $\mu_{index}(\Omega) = \{\mu_{index}(P) \mid P \in \Omega\ \} = [min, max]_{\mathbb{N}}$. Hier ist min der kleinste Index, der auftritt. Bei der Definition des 2.Laufes wurde festgelegt, dass $min = 0$. Es kann hier jedoch ohne Einschränkung ein beliebiger Startwert $min \in \mathbb{N}$ gewählt werden.
[79] Abbildung 5.24
[80] d.h. Reihenfolge (Raster)[Def. 24] und Konvention (Rand)[Bem. 10]

P und $\mu_{addr}(P)$ zusammenhängend sind. Da

$$P \mapsto \mu_{addr}(P) =: P_1 \mapsto \mu_{addr}(P_1) =: P_2 \mapsto ... \mapsto head(P)$$

kann hieraus sofort gefolgert werden, dass P und $head(P)$ zusammenhängend sind.
Der Beweis, dass P und $\mu_{addr}(P)$ zusammenhängend sind, erfolgt durch Induktion. Sei $i < b*h$. Unter der Voraussetzung, dass für alle P_j mit $j < i$ die Punkte $\mu_{addr}(P_j)$ und P_j zusammenhängend sind, muss gezeigt werden, dass dies auch für $j = i$ gilt. Dazu werden alle 4 Fälle welche im Algorithmus des ersten Laufes[81] auftreten können untersucht.
Tritt der **erste Fall** ein, folgt dies sofort, da jeder Punkt mit sich selbst zusammenhängt. Tritt der **zweite Fall** ein, sind P und P_{links} zusammenhängend. Da P_{links} und $\mu_{addr}(P_{links})$ nach Induktionsvoraussetzung zusammenhängend sind, sind es auch P und $\mu_{addr}(P) := \mu_{addr}(P_{links})$. Für den **dritte Fall** gilt dies ebenso.
Tritt der **vierte Fall** ein, sind P, P_{links} und P_{oben} zusammenhängend. Nach der Induktionsvoraussetzung, der Definition von $head(P)$ und der Tatsache dass $\mu_{addr}(P) \leq P$, folgt, dass P_{links} und $head(P_{links})$ zusammenhängend sind. Ebenfalls gilt dies für P_{oben} und $head(P_{oben})$. Daher sind auch P und $\mu_{addr}(P) := min\{head(P_{links}), head(P_{oben})\}$ zusammenhängend.
Hiermit ist der Induktionsschritt gezeigt. Da $\mu_{addr}(P_0) = P_0$[82], ist die Induktionsvoraussetzung erfüllt, und es gilt $\forall P \in \Omega$, dass $\mu_{addr}(P)$ und P zusammenhängend sind. Damit sind auch P und $head(P)$ zusammenhängend, und es gilt [$head(P) = head(Q) \Longrightarrow$ P, Q zusammenhängend].
Nun muss noch [$head(P) = head(Q) \Longleftarrow$ P, Q zusammenhängend] bewiesen werden. Seien $P, Q \in \Omega$ zusammenhängende Punkte. Es existiert daher ein 4-Weg[83] zwischen P und Q. Seien $V, W \in \Omega$ zu P beziehungsweise Q benachbarte Punkte. Zuerst eine kurze Feststellung: $V = P_{links}$, $V = P_{oben}$ oder $P = V_{links}$, $P = V_{oben}$. Das gleiche gilt für W und Q. Überprüft man die vier Fälle des Algorithmus, stellt man fest, dass immer $head(V) = head(W)$ gilt. Da dies für alle benachbarten Punkte auf dem 4-Weg zwischen P und Q zutrifft, folgt insbesondere auch $head(P) = head(Q)$. Hiermit ist die Aussage gezeigt. □

Satz 8 *Es gelten die gleichen Voraussetzungen wie in Satz 6. Seien $P, Q \in \Omega$. Dann gilt nach dem Abschluss des zweiten Laufs:*

$$\mu_{index}(P) = \mu_{index}(Q) \Longleftrightarrow P,\ Q \text{ zusammenhängend}$$

Beweis: Seien P und Q zusammenhängend. Dann folgt aus Satz 7, dass $head(P) = head(Q)$. Außerdem ist nach Definition des Kopfes $\mu_{addr}(head(P)) = head(P)$.

[81] Abbildung 5.23
[82] nach Satz 5
[83] d.h. keine diagonalen Wegverläufe

Zuerst sei die durch μ_{addr} induzierte Folge mit Startpunkt P dargestellt:

$$P \to \mu_{addr}(P) \to \mu_{addr}(\mu_{addr}(P)) \to ... \to (\mu_{addr})^n(P) = head(P)$$

Nach Satz 5 ist $\mu_{addr}(P) \leq P$. Die Punkte dieser Folge werden daher beim zweiten Lauf, von hinten abgearbeitet.[84].
Bei $head(P)$ tritt der erste Fall ein, und es wird eine neuer Wert für $\mu_{index}(head(P))$ gesetzt. Der zweite Punkt der Folge, welcher bearbeitet wird, ist $(\mu_{addr})^{n-1}(P)$. Hier tritt der zweite Fall in Kraft und:

$$\mu_{index}((\mu_{addr})^{n-1}(P)) := \mu_{index}(\mu_{addr}((\mu_{addr})^{n-1}(P))) = \mu_{index}(head(P))$$

Führt man dies weiter, folgt $\mu_{index}(P) = \mu_{index}(head(P))$. Da nach Voraussetzung $head(P) = head(Q)$, folgt daraus $\mu_{index}(P) = \mu_{index}(Q)$ und damit:

$$\mu_{index}(P) = \mu_{index}(Q) \Longleftarrow \text{P, Q zusammenhängend}$$

Es bleibt jetzt noch zu zeigen, dass, falls $\mu_{index}(P) = \mu_{index}(Q)$, P und Q zusammenhängend sind. Es gilt, wie gerade gezeigt wurde, $\mu_{index}(P) = \mu_{index}(head(P))$ und $\mu_{index}(Q) = \mu_{index}(head(Q))$. Daher ist auch $\mu_{index}(head(Q)) = \mu_{index}(head(P))$.
Bei allen Punkten P für die $head(P) = P$ gilt (Kopfpunkt), tritt im zweiten Lauf immer der 1.Fall ein. Zwei verschiedene Kopfpunkte haben daher verschiedene Indizes. Falls $\mu_{index}(P) = \mu_{index}(Q)$, muss daher auch $head(P) = head(Q)$ zutreffen. Nach Satz 7 sind dann P und Q zusammenhängend. Die Aussage ist hierdurch gezeigt. □

5.6 Implementation des TPSS-Algorithmus

In diesem Abschnitt wird anhand von Modellimplementationen aufgezeigt, wie der TPSS-Algorithmus auf einem FPGA implementiert werden kann. Anschließend wird ein Laufzeitmodell vorgestellt, mit welchem die Laufzeit verschiedener Modelle taktgenau berechnet werden kann[85]. Abschließend wird die FPGA-Implementation mit einer Softwareimplementation verglichen, um die Leistungsfähigkeit der FPGA-Implementation besser einschätzen zu können.

5.6.1 Modellimplementationen

In diesem Abschnitt wird gezeigt, wie das TPSS-Verfahren auf einem FPGA realisiert werden kann. Das Ziel ist es, das TPSS-Verfahren möglichst vorteilhaft an die FPGA-Architektur anzupassen, um eine hohe Leistung bei einem möglichst geringen Ressourcenverbrauch zu gewährleisten.[86] Eine genaue Beschreibung des FPGA und wie es konfiguriert wird findet sich in Abschnitt 5.6.4.

[84] d.h. $head(P)$ zuerst

[85] mit Hilfe dieses Laufzeitmodell wird im Weiteren genau aufgezeigt, in welchem Maße der TPSS-Algorithmus vom Bildinhalt abhängig ist und was die Gründe hierfür sind.

[86] die wichtigste Voraussetzung hierfür ist jedoch, dass der Algorithmus des (CCL-)Verfahrens gut für eine FPGA-Implementation geeignet ist. Dies ist der Grund warum das TPSS-Verfahren entwickelt wurde.

Datenstromverarbeitung

Eine sehr hohe Leistung kann erreicht werden, wenn es gelingt, eine Datenstromverarbeitung[87] durchzuführen, bei welcher ein Bildpunkt pro Takt verarbeitet werden kann. Dies ist aus den folgenden Gründen jedoch nur eingeschränkt möglich.
Da der zweite Lauf erst nach der Durchführung des ersten Laufs durchgeführt werden kann, ist eine Datenstromverarbeitung des Gesamtprozesses nicht möglich. Das Ergebnis des ersten Laufs muss zwischengespeichert werden. Beim ersten und zweiten Lauf konnte jedoch annähernd eine Datenstromverarbeitung realisiert werden, mit welcher pro Takt näherungsweise ein Bildpunkt verarbeitet werden kann.[88]
Das annähernd bezieht sich darauf, dass sowohl beim ersten und zweiten Lauf aufgrund der Konfliktfälle[89] eigentlich keine Datenstromverarbeitung möglich ist, da hier unvorhersehbare, erst während der Laufzeit ermittelte Operationen durchgeführt werden müssen[90]. Treten keine Konfliktfälle auf, könnte allerdings eine reine Datenstromverarbeitung durchgeführt werden[91].
Es wurde daher folgender Ansatz gewählt. Es wird eine Datenstromverarbeitung durchgeführt. Bei einem Konfliktfall wird jedoch die Datenstromverarbeitung unterbrochen und der Konflikt aufgelöst. Anschließend wird die Datenstromverarbeitung fortgesetzt. Die Datenstromverarbeitung muss daher, möglichst ohne Zeitverlust, gestoppt und gestartet werden können.

Speicheranbindung

Die Anbindung des externen RAMs spielt eine große Rolle hinsichtlich der Performance der Implementation. Das TPSS-CCL-Modul, das in dieser Arbeit entwickelt wurde, kann je nach Konfiguration ein oder zwei Speicherkanäle mit einer Bit-breite von 16 Bit - 32 Bit nutzen. Die Leistungsfähigkeit des TPSS-CCL Moduls wurde bezüglich vier verschiedener (Modell-)Speicheranbindungen untersucht. Informationen über diese Modellimplementationen sind in Abbildung 5.25 aufgelistet.
Je nach Modell stehen ein oder zwei Speicherbereiche, auf welche gleichzeitig zugegriffen werden kann, zur Verfügung. Bei allen Modellen wird davon ausgegangen, dass das Quellbild μ_{zsh} im Speicher abgelegt ist und das Ergebnisbild μ_{index} ebenfalls in den Speicher geschrieben wird.[92]
Modell-1 entspricht der Version, welche auf dem Nexys-2 Bord[94] implementiert wer-

[87] Abschnitt 3.4.3
[88] der TPSS-Verfahren benötigt demzufolge ungefähr zwei Takte pro Bildpunkt (Abschnitt 5.6.4, Abbildung 5.33)
[89] Abschnitt 5.3.2
[90] Abschnitt 5.5.2, Abbildung 5.22 und 5.24, jeweils Fall 2
[91] es wäre dann sogar nur der erste Lauf nötig
[92] Die Eingangsdaten μ_{zsh} könnten auch von einer externen Quelle pixelweise in der Reihenfolge Raster (Definition 24) eingespeist werden. Dann entfällt in allen Modellen die Speicherung des Bildes μ_{zsh}. Die Labelmaske μ_{index} kann auch pixelweise abgeführt werden. Im zweiten Lauf muss aber potentiell ein Zugriff auf alle bereits berechneten Indizes bestehen[93]. Die Labelmaske μ_{index} muss daher auf alle Fälle abgespeichert werden.
[94] Abschnitt 3.4.1

den konnte[95]. **Modell-2 und -3** konnten aufgrund der schmalen Speicheranbindung des RAMs an das Spartan-3E FPGAs nicht auf dem Nexy-2-Board umgesetzt werden. Modelle-2 und -3 wurden jedoch mithilfe der Simulationsumgebung, welche DK-Design Suite[96] zur Verfügung stellt, getestet. Da sich außerdem nur die Speicheranbindung ändert[97], kann davon ausgegangen werden, dass sich die Modelle ohne Probleme auf anderen FPGA-Plattformen synthetisieren lassen.

Bei **Modell-2, -3 und -4** wird vorausgesetzt, dass pro Speicherzugriff ein Bildpunkt in μ_{addr} beziehungsweise μ_{index} gelesen und geschrieben werden kann[98]. Beim Modell-1 wird davon ausgegangen, dass zum Lesen und Schreiben eines Bildpunktes in μ_{addr} beziehungsweise μ_{index} zwei Speicherzugriffe (2 * 16 Bit) nötig sind[99].

Bei **Modell-1,-2,-4** können sich μ_{addr} und μ_{index} den Speicherplatz teilen[100]. Beim **Modell-3** ist dies nur möglich, falls zwei unabhängige Speicherzugriffe gleichzeitig auf μ_{addr} bzw. μ_{index} durchgeführt werden können[101]. Dies kann beispielsweise mithilfe eines Dual-Port-RAM realisiert werden[102]. Ansonsten müssen μ_{addr} und μ_{index} auf verschiedenen Speicherbausteinen untergebracht werden.

Bei **Modell-3** können zwei Speicherzugriffe unabhängig voneinander durchgeführt werden. Das Modell-3 entspricht dem Szenario, für welches das TPSS-CCL-Modul entwickelt wurde. Erst wenn die Eingangsdaten und Ausgangsdaten gleichzeitig zugeführt und abgeführt werden können, kann die Datenstromverarbeitung ihr volles Leistungspotential entfalten.

Das **Modell-4** entspricht dem (Software-)Modell, welches für die erste Einschätzung des in Abschnitt 5.4.3 untersuchten, auf Union-Find-Datenstrukturen beruhenden TPSS-Verfahrens, implementiert wurde. Es ist mit Modell-2 vergleichbar. Allerdings wird bei Modell-4 davon ausgegangen, dass die Laufzeit des Verfahrens ausschließlich von der Anzahl der Speicherzugriffe abhängt. Dies impliziert, dass alle Operationen des Verfahrens parallel zu den Speicherzugriffen durchgeführt werden können. Wie später gezeigt wird, ist dies bei Modell-2 näherungsweise gelungen.

[95] auf dem Nexys-2-Board kann pro Takt ein 16 Bit Speicherwert auf den externen RAM geschrieben und gelesen werden

[96] Handel-C Entwicklungsumgebung (Abschnitt 3.4.2)

[97] das TPSS-CCL Modul ist in allen Modellen dasselbe

[98] dies entspricht einer Speicheranbindung von $\lceil \log_2(\text{Bildhöhe} * \text{Bildbreite}) \rceil$ Bit, gewöhnlich 24/32 Bit. Konkrete Werte finden sich in Abbildung 5.21

[99] mit einer 16 Bit Speicheranbindung ließen sich maximal Bilder der Größe 256*256 Pixel mit einem Speicherzugriff bearbeiten.

[100] da beim zweiten Lauf μ_{index} von der Vorbelegung unabhängig ist und zur Berechnung von $\mu_{index}(P)$, bezüglich nur $\mu_{addr}(P)$ benötigt wird.

[101] notwendig beim 2-Lauf (Abbildung 5.24, Fall 2)

[102] Ideal wäre es natürlich, wenn zur Speicherung von μ_{addr} bzw. μ_{index} der interne (Dual-Port-)Blockram eines FPGAs verwendet werden könnte. Dies ist mit den meisten heutigen FPGAs leider nur für kleine Bildgrößen möglich (Abschnitt 5.5.1). **Modell-3** könnte dann außerdem mit nur einer einzigen Speicheranbindung umgesetzt werden. Da laut Anforderungsprinzip große Bilder verarbeitet werden sollen, wurde dieser Ansatz in der vorliegenden Arbeit nicht weiterverfolgt. Bei Videoanwendungen könnte dies aber sinnvoll sein.

[103] Speicheranbindung abhängig von Größe des Bildes, 24 Bit bei Bildgrößen bis 4096*4096 Pixel, 32 Bit bei Bildgrößen bis 65536*65536 Pixel

Modell	1	2	3	4
Speicherzugriffe pro Takt (R/W)[103]	1 * 16 Bit	1 * 24/32 Bit	2 * 24/32 Bit	1 * 24/32 Bit

Abbildung 5.25: Übersicht der (externen) Speicheranbindung der Testmodelle

5.6.2 Laufzeitmodell

Im diesen Abschnitt soll die Laufzeit der Modellimplementationen taktgenau ermittelt werden. Die Laufzeit ist, wie bereits bei der Voruntersuchung gezeigt, von der Bildbeschaffenheit abhängig[104]. Es soll geklärt werden, inwieweit und weshalb die Laufzeit vom Bildinhalt abhängig ist, und jeweils die maximale und durchschnittliche Laufzeit der Modellimplementationen abgeleitet werden.
Um dies durchführen zu können, musste der erste und zweite Lauf näher untersucht werden. Da nur Modell-1 auf einem FPGA implementiert wurde, wurde ein Laufzeitmodell entwickelt, mit welchem für jedes Modell, für beliebige Bildgrößen und Bilder bezüglich 4- und 8-Zusammenhangs, die zu erwartende Laufzeit bestimmt werden kann.

Erster Lauf

Beim ersten Lauf treten zwei Arten von Bildpunkten auf. Sie werden **schnelle** beziehungsweise **langsame Pixel** genannt. Bei den schnellen Pixeln ist eine Datenstromverarbeitung möglich. Bei den langsamen Pixeln muss die Datenstromverarbeitung unterbrochen und eine Ausnahmebehandlung durchgeführt werden. Für eine genauere Untersuchung sei auf die detaillierte Darstellung des ersten Laufes in Abbildung 5.26 verwiesen. In den folgenden Fällen treten schnelle und langsame-Pixel auf:

- Schnelle Pixel: Fall 1, Fall 2, Fall 3, Fall 4.1
- Langsame Pixel: Fall 4.2

Um im Fall 3 und Fall 4.1 ebenfalls eine Datenstromverarbeitung durchführen zu können, muss eine komplette Zeile von μ_{addr} zwischengespeichert werden[105]. Der Zeilenpuffer hat abhängig von der Bildgröße folgende Größe:

$$\text{Bildbreite} * \lceil \log_2(\text{Bildhöhe} * \text{Bildbreite}) \rceil \text{ Bit}$$

Der Speicherverbrauch für ein paar ausgewählte Bildgrößen findet sich in Abbildung 5.27. Der Spartan-3E XC3S1200E, welcher auf dem Nexys-2 Testboard verbaut ist, stellt 663 kBit Block-RAM zur Verfügung. Für eine Bildgröße von $1024 * 1024$ Pixel, welche zu Analysezwecken benutzt wurde, konnte der Zeilenpuffer daher ohne Probleme im Block-RAM untergebracht werden. Bei modernen, leistungsfähigeren FPGAs[106], steht Block-

[104] Abschnitt 5.4.3
[105] um auf $\mu_{addr}(P_{oben})$ ohne weiteren Speicherzugriff zugreifen zu können
[106] beispielsweise der Xilinx Spartan 6 oder Xilinx Virtex 6 Baureihe

Algorithmus 1 Lauf (detailliert, 4-Zusammenhang):

Fall 1 - Falls $con(P, P_{links}) == con(P, P_{oben}) == 0$
$\mu_{addr}(P) := P$

Fall 2 - Falls $con(P, P_{links}) == 1$ und $con(P, P_{oben}) == 0$
$\mu_{addr}(P) := addr(P_{links})$

Fall 3 - Falls $con(P, P_{links}) == 0$ und $con(P, P_{oben}) == 1$
$\mu_{addr}(P) := addr(P_{oben})$

Fall 4 - Falls $con(P, P_{links}) == con(P, P_{oben}) == 1$
Fall 4.1 - Falls $\mu_{addr}(P_{links}) == \mu_{addr}(P_{oben})$
$\mu_{addr}(P) := \mu_{addr}(P_{links})$
Fall 4.2 - Falls $\mu_{addr}(P_{links}) \neq \mu_{addr}(P_{oben})$
Fall 4.2.1 - Falls $head(P_{links}) == head(P_{oben})$
$\mu_{addr}(P) := head(P_{links})$
Fall 4.2.2 - Falls $head(P_{links}) < head(P_{oben})$
$\mu_{addr}(P) := head(P_{links})$
$\mu_{addr}(head(P_{oben})) := head(P_{links})$
Fall 4.2.3 - Falls $head(P_{links}) > head(P_{oben})$
$\mu_{addr}(P) := head(P_{oben})$
$\mu_{addr}(head(P_{links})) := head(P_{oben})$

wobei: $head(P) := (\mu_{addr}(P) == P) \ ? \ P : \ head(\mu_{addr}(P))$

Abbildung 5.26: Der erste Lauf, detailliert, 4-Zusammenhang

RAM in der Größenordnung von einigen MBit zur Verfügung. Eine Zeilenpufferung stellt daher auch bei sehr großen Bildern kein Problem dar[107].

Zweiter Lauf

Es folgt eine Analyse des zweiten Laufes. Die hierzu verwendete detaillierte Darstellung des Algorithmus ist in Abbildung 5.28 zu finden.
Die Bildpunkte können hier wiederum in **langsame** und **schnelle Pixel** eingeteilt werden:

- Schnelle Pixel: Fall 1, Fall 2.1, Fall 2.2

[107] Um Missverständnisse zu vermeiden: der Block-RAM heutiger FPGAs reicht aus, um eine Zeilenpufferung durchzuführen, jedoch wie bereits erwähnt in den meisten Fällen nicht, um ein gesamtes Bild zwischenzuspeichern.

Bildhöhe/breite	50	512	1024	2048	4096	8192	16384
Größe in kBit	0,6	10	21	45	99	213	459
Bits pro Pixel	12	18	20	22	24	26	28

Abbildung 5.27: Größe des Zeilenpuffers, Werte aufgerundet

Algorithmus zweiter Lauf (detailliert, 4 Zusammenhang)

Fall 1 - Falls $\mu_{addr}(P) = P$
$\mu_{index}(P) :=$ Neuer Index
Fall 2 - Falls $\mu_{addr}(P) \neq P$
Fall 2.1 - Falls $\mu_{addr}(P) == \mu_{addr}(P_{links})$
$\mu_{index}(P) := \mu_{index}(P_{links})$
Fall 2.2 - Falls $\mu_{addr}(P) == \mu_{addr}(P_{oben})$
$\mu_{index}(P) := \mu_{index}(P_{oben})$
Fall 2.3 - Sonst
$\mu_{index}(P) := \mu_{index}(\mu_{addr}(P))$

Abbildung 5.28: Der zweite Lauf, detailliert, 4 Zusammenhang

- Langsame Pixel Fall 2.3

Bei den schnellen Pixeln kann eine Datenstromverarbeitung durchgeführt werden[108]. Bei den langsamen Pixeln muss die Datenstromverarbeitung unterbrochen und eine Sonderbehandlung durchgeführt werden.

Berechnung der Laufzeit

Die Anzahl der Takte, welche zur Verarbeitung eines Bildpunktes benötigt werden, lassen sich folgendermaßen berechnen. Sei $P \in \Omega$. Die Anzahl der Takte welche für die Verarbeitung von P benötigt werden, wird mit **Takte(P)** bezeichnet.

Erster Lauf:

- Takte(Schnelles-Pixel) := $\mathbf{clocks_{fast1}}$
- Takte(Langsames-Pixel) := $\mathbf{clocks_{slow}} + clocks_{merge}(P_{links}, P_{oben})$

wobei:

- $clocks_{merge}(P_{links}, P_{oben}) := find_{head}(P_{links}) + find_{head}(P_{oben}) + \delta_{relableCase} * \mathbf{clocks_{relabel}}$

[108] Um im Fall 2.2 ebenfalls eine schnelle Verarbeitung durchführen zu können, muss eine Zeilenpufferung von μ_{addr} und μ_{index} durchgeführt werden.

- $find_{head}(P) := depth(P) * \mathbf{clocks_{findHeadPerDepth}}$
- $depth(P) :=$ Tiefe der Folge $P_{n+1} := \mu_{addr}(P_n)$ mit Startpunkt P
- $\delta_{relableCase} := 0$ im Fall 4.2.1, $\delta_{relableCase} := 1$ im Fall 4.2.2, 4.2.3,

Zweiter Lauf:

- Takte(Schnelles Pixel) := $\mathbf{clocks_{fast2}}$
- Takte(Langsames Pixel) := $\mathbf{clocks_{lookup}}$

Die von den Modellen abhängigen Parameter sind fett dargestellt. Sie sind in Abbildung 5.29 zu finden.

Modell	1	2	3	4
Speicherzugriffe pro Takt (R/W)	16 Bit	24/32 Bit	2 * 24/32 Bit	24/32 Bit
$\text{clocks}_{\text{fast1}}$	3	2	1	2
$\text{clocks}_{\text{slow}}$	8	5	4	3
$\text{clocks}_{\text{findHeadPerDepth}}$	3	2	2	1
$\text{clocks}_{\text{relabel}}$	1	0	0	1
$\text{clocks}_{\text{fast2}}$	4	2	1	2
$\text{clocks}_{\text{lookup}}$	7	4	3	3
Takte pro Pixel	7 + Δ	4 + Δ	2 + Δ	4 + Δ

Abbildung 5.29: Parameter zur Taktberechnung

Bei einem Konfliktfall (langsames Pixel) wird eine weitaus höhere Anzahl an Takten benötigt, als für die schnellen Pixel, bei denen eine Datenstromverarbeitung möglich ist. Beispielsweise wird für den ersten Lauf im Modell-2 für die Bearbeitung eines schnellen Pixels zwei Takte benötigt, für die Bearbeitung eines langsamen Pixels werden mindestens $5 + 1 * 2 + 1 * 2 = 7$ Takte benötigt. Daher ist in erster Linie die Anzahl der Konfliktfälle für die Verarbeitungsgeschwindigkeit entscheidend. Beim Modell-3 kann im ersten und zweiten Lauf ein schneller Bildpunkt pro Takt verarbeitet werden.
Interessant ist auch ein Vergleich zwischen Modell-2 und Modell-4. Modell-4 ist das theoretische Modell, bei dem davon ausgegangen wurde, dass alle Operationen parallel zu den ohnehin benötigten Speicherzugriffen stattfinden[109]. Das Modell-2 entspricht einer realen Implementation. Bei den schnellen Pixeln gleicht sie Modell-4. Bei den langsamen Pixeln ist es nicht vollständig gelungen, jede Operation parallel zu einem Speicherzugriff durchzuführen. Dies liegt daran, dass bei den langsamen Pixeln dynamische, erst zur Laufzeit bestimmte Operationen durchgeführt werden, welche sich nicht ohne weiteres hinter Speicherzugriffen verstecken lassen.

[109] i.a.W eine optimale Implementation

Analyse der Testbilder

In der Abbildung 5.30 ist dargestellt, zu welchem Prozentsatz schnelle und langsame Pixel bei den synthetischen Testbilder auftreten. Zusätzlich ist die Tiefenverteilung[110] der Punkte $P \in \mu_{addr}$ dargestellt, welche bei der Ermittlung des Kopfs $head(P)$ auftritt. Dies ist ein weiter Grund, weshalb die Laufzeiten des TPSS-Verfahrens vom Bildinhalt abhängig sind. Da sich eine maximale Suchtiefe von 3 ergibt, wurde auf eine Pfadkompression verzichtet, welche bei Union-Find Datenstrukturen verwendet werden kann, um die *Find(x)*-Operation zu beschleunigen.
Wie die Laufzeit für den 8-Zusammenhang berechnet werden kann, findet sich in Anhang Abbildung 7.6 und die Analyse der synthetischen Daten in Anhang Abbildung 7.7. Es ergeben sich keine wesentlichen Unterschiede im Vergleich zum 4-Zusammenhang.

Testbilder	Weiß	Dreieck	Diagonal 1	Diagonal 2	Labyrinth 1	Labyrinth 2
Erster Lauf						
Anteil schneller Pixel (in %)	100,00	99,90	50,30	100,00	91,71	89,83
Anteil langsamer Pixel (in %)	-	0,10	49,71	0,00	8,30	10,17
Suchtiefe der $find_{head}(P)$ Operation[111]	Tiefenverteilung in Prozent					
1	-	75,05	100,00	-	81,32	75,52
2	-	24,95	-	-	17,50	22,19
3	-	-	-	-	1,17	2,29
>3	-	-	-	-	-	-
Zweiter Lauf						
Anteil schneller Pixel (in %)	99,71	99,56	50,05	0,80	57,97	57,97
Anteil der zeilengepufferten Pixel (in %)	0,29	0,34	0,24	0,20	30,49	30,49
Anteil der langsamen Pixel (in %)	-	0,10	49,71	0,00	11,54	11,54

Abbildung 5.30: Analyse der synthetischen Testbilder, Bildgröße 1024*1024 Pixel, 4-Zusammenhang

5.6.3 Modelllaufzeiten

In Abbildung 5.31 ist dargestellt, wie viele Takte pro Bildpunkt im Durchschnitt für die synthetischen Testbilder[112] benötigt werden. Der Bildinhalt hat wie erwartet Einfluss auf die Bearbeitungsgeschwindigkeit. Beim weißen Testbild treten keine Konfliktfälle auf[113]. Die Datenstromverarbeitung wird daher nicht unterbrochen, und es kann die höchste Verarbeitungsleistung erreicht werden.
Es sei darauf hingewiesen, dass die Bildgröße bei den synthetischen Testbildern nur einen vernachlässigbaren Einfluss auf die durchschnittliche Laufzeit pro Bildpunkt hat[114].

[110] Definition 22

[111] dies entspricht der Größe $depth(P)$ im Laufzeitmodell der ersten Laufs, welche zur Berechnung von $find_{head}(P)$ benötigt wird.

[112] bzgl. 4-Zusammenhang, 8-Zusammenhang (Abbildung 7.8)

[113] Anteil schneller Pixel 100%

[114] kleinere Abweichungen können aufgrund des geringfügig unterschiedlichen Anteils schneller und langsamer Pixel, bei Testbildern des gleichen Typs unterschiedlicher Größe, auftreten (bei Bildern der Größe $\geq 512 * 512$ Pixel: $\pm 0,01$ Takte). Der Grund hierfür ist, dass die Testmuster aus denen die

Beim Diagonal-1 Bild treten die meisten Konfliktfälle auf[115]. Zum Prozessierung dieses Bildes wird daher die meiste Zeit benötigt. Bezüglich Modell-3 wird im Vergleich zum weißen Bild ungefähr die 2, 5-fache Zeit benötigt.
Im Normalfall bewegt sich die Verarbeitungsgeschwindigkeit nahe an den optimalen Werten. Selbst bei den sehr komplexen Labyrinth-Bildern erhöht sich die Laufzeit nur ungefähr um 35%.
Die Laufzeiten, welche bezüglich den realen Testdaten ermittelt wurden, sind in Abbildung 5.32 dargestellt[116]. Sie bewegen sich nahe den optimalen Werten.
Das Ergebnis der Laufzeituntersuchung ist, dass die Laufzeiten in Regel sehr stabil sind. Im Normalfall bewegen sie sich nahe an den optimalen Werten. Auch bei den sehr komplexen Labyrinth-Bildern bleibt die Erhöhung mit ungefähr 35% der Laufzeit im Rahmen. Nur bei den Diagonalbildern erhöht sich die Laufzeit teilweise um mehr als 200%. Sie stellen allerdings auch die Worst-Case Szenarien des TPSS-Verfahrens dar und dürften für reale Anwendungen kaum eine Rolle spielen[117]. Ein weiteres Ergebnis dieser Untersuchung ist, dass die Laufzeit nur geringfügig davon abhängig ist, ob nach 4- oder 8-Zusammenhang gelabelt wird[118].

Testbilder	Durchschnittliche Takte pro Pixel			
	Model 1	Model 2	Model 3	Model 4
Weiß	7,00	4,00	2,00	4,00
Dreieck	7,01	4,01	2,01	4,00
Diagonal 1	12,97	7,48	5,48	5,99
Diagonal 2	7,00	4,00	2,00	4,00
Labyrinth 1	8,19	4,65	2,65	4,35
Labyrinth 2	8,22	4,74	2,74	4,38

Abbildung 5.31: Modelllaufzeiten synthetischer Testbilder, Bildgröße 1024*1024, 4-Zusammenhang

5.6.4 FPGA-Implementation

Die FPGA-Implementation des TPSS-CCL-Verfahren[119] wurde mithilfe der höheren Hardwarebeschreibungssprache Handel-C[120] umgesetzt. Die Implementation des Verfahrens

synthetischen Testbilder zusammengebaut sind, am Bildrand unterschiedlich abgeschnitten werden.
[115] Anteil schneller Pixel 50%
[116] bezüglich 4-Zusammenhang, 8-Zusammenhang siehe Abbildung 7.9
[117] Die Diagonal-1 bzw. -2 Bilder stellen das Worst-Case-Szenarien bei Two-Pass Verfahren bezüglich des 4- bzw. 8-Zusammenhangs dar. Dies liegt daran, dass bei diesen Bildern die maximal mögliche Zahl an Konfliktfällen auftritt (Abbildung 5.30 bzw. Abbildung7.7)
[118] wenn man von den Worst-Case Szenarien, welche beim Diagonal-1 bzw. Diagonal-2 Bild auftreten, absieht.
[119] 4-Zusammenhang
[120] Abschnitt 3.4.2

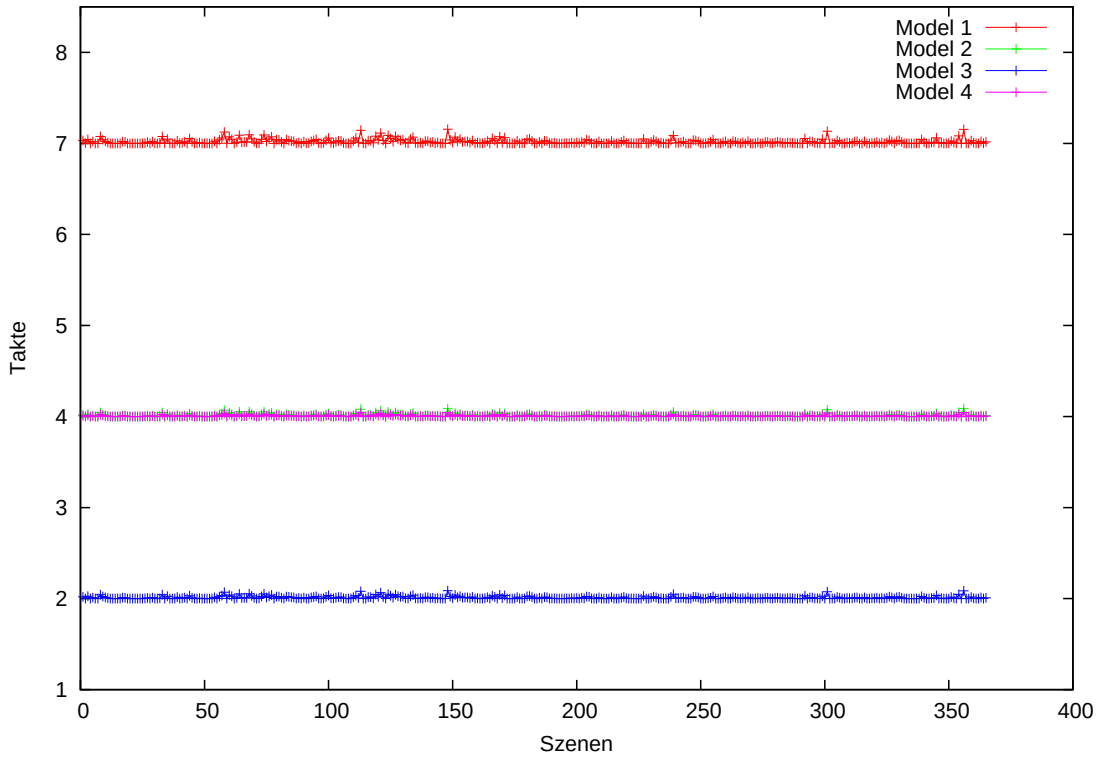


Abbildung 5.32: Modelllaufzeiten realer Testbilder bezüglich 4-Zusammenhangs, Bemerkung: die Modell-2 Laufzeiten sind in dieser Abbildung kaum sichtbar, da sie sich mit den Modell-4 Laufzeiten überlagern.

erfolgte in sechs Stufen (Opt-0,1,2,3,4,5). Opt-5 ist die finale Implementation und entspricht je nach Speicheranbindung Modell-1,-2 oder -3. Wie in Abbildung 5.33 dargestellt, wurde die Anzahl der Takte, welche durchschnittlich für die Prozessierung eines Bildpunkt benötigt werden, in jedem Schritt verringert. Technische Details zu den Implementationen der einzelnen Versionen sind in Abbildung 5.33 zu finden.
Zuerst wurde ein Prototyp TPSS-Verfahren in der Programmiersprache C (Software) implementiert. Opt-0 entspricht einer 1:1 Portierung dieses Codes auf FPGA. Dies kann mit Handel-C ziemlich gut bewerkstelligt werden.
Bei Opt-1 wurden einige erste Handel-C spezifische Optimierungen an den Kontrollstrukturen vorgenommen. Beispielsweise wurden for-Schleifen durch while-Schleifen ersetzt, wodurch die Erhöhung der Schleifenindizes parallel zu den Verarbeitungsschritten erfolgen konnte, und einige Schleifen wurden sogar komplett entfernt[121]. Hierdurch konnte

[121] durch *Loop-Unrolling*, **Bemerkung:** bei Schleifen mit sehr wenigen mit zur Compilezeit feststehender Anzahl an Durchläufen, ist es oft sinnvoll diese auszuformulieren. Das heißt den Schleifenkörper zu entfernen und die Operationen direkt anzugeben. Bei Software kann dies durch Einsparung der Schleifenkontrollstruktur zu höherer Geschwindigkeit führen und wird normalerweise vom Compiler

Version (Bit-tiefe μ_{index})	Opt-0	Opt-1	Opt-2	Opt-5 (Modell-1)	Opt-5* (Modell-3, simuliert)
Weiß	26,94	25,94	8,97	7,00	2,00
Dreieck 2	26,96	25,97	8,99	7,01	2,01
Diagonal 1	35,37	33,87	17,41	1,.97	5,48
Diagonal 2	33,86	32,37	7,47	7,00	2,00
Labyrinth 1	37,70	36,63	10,86	8,05	2,65
Labyrinth 2	38,09	37,02	11,24	8,25	2,74

Abbildung 5.33: Messergebnisse der durchschnittlichen Anzahl der benötigten Takte pro Pixel, alle Messungen bis auf Opt-5* wurden auf dem Nexys-2-Prototyp Bord durchgeführt, die Opt-5* Werte entsprechen den Modellwerten aus Abbildung 5.31, bei Opt-5 kann ein 16 Bit Speicherzugriff pro Takt durchgeführt werden, bei Opt-5* können zwei 2*32 Bit Speicherzugriffe pro Takt durchgeführt werden

zwar die Takteffizienz nicht wesentlich erhöht werden, die Komplexität des Designs wurde dadurch jedoch verringert, und es kann daher wesentlich höher getaktet werden[122]. Der Quellcode der Opt-1 Implementation, entspricht, bis auf den oben genannten eher formalen Änderungen, jedoch noch dem Quellcode der C-Implementation. Auch am Algorithmus selbst wurden keine Änderungen durchgeführt.
Bei Opt-2 wurde der original C-Code soweit es geht parallelisiert. Dieser Schritt ist bereits relativ aufwendig. Die Takteffizienz konnte, in erster Linie durch ein Veränderung der Reihenfolge der Befehlsausführung[123], enorm gesteigert werden. Allerdings entspricht diese Version im Grunde noch der Softwareversion. Eine Datenflussverarbeitung wird hier nicht durchgeführt, es erfolgt maximal ein Speicherzugriff pro Takt. Auch wenn mehrere Speicherzugriffe pro Takt möglich wären, könnten sie nicht genutzt werden. Die Komplexität des Designs steigt im Vergleich zu Modell-2, was sich in einer etwas niedrigeren maximalen Taktfrequenz niederschlägt[124].
Eine weitere Effizienzsteigerung konnte nur durch eine Umstellung auf eine Datenstromverarbeitung erzielt werden. Die Umstellung auf eine Datenstromverarbeitung war sehr aufwendig, und wurde in mehreren Schritten (Opt-3,4,5) durchgeführt. Die Takteffizienz kann hierdurch aber nochmals erheblich gesteigert werden. Pro Lauf kann nun bei entsprechender Speicheranbindung ein Bildpunkt pro Takt bearbeitet werden[125]. Das CCL-Modul, welches in dieser Arbeit entwickelt wurde, entspricht der Opt-5 Version.
Bei Opt-5 wird eine gesteuerte Datenstromverarbeitung durchgeführt. Daher muss jedes

selbstständig durchgeführt. Bei FPGA-Implementation kann ebenfalls durch Einsparung der Kontrolllogik die Komplexität des Designs verringert und die maximal erreichbare Taktgeschwindigkeit erhöht werden. Bei Handel-C wird dieser Schritt nicht automatisch vom Compiler durchgeführt, da die Funktionseinheiten im Schleifenkörper, welche bei jedem Schleifendurchlauf wiederverwendet werden können, nun mehrfach generiert werden müssten. Ob dies sinnvoll ist, sollte bei jeder Schleife einzeln entschieden werden.

[122] Abbildung 5.34

[123] dies entspricht im Grunde einer manuell durchgeführten Out-of-order Programmausführung

[124] Abbildung 5.34

[125] dies entspricht Modell-3

Arbeitsregister gepuffert werden. Dies und die hierfür benötigte zusätzliche Steuerlogik führen im Vergleich zu Opt-2 zu einem ungefähr doppelt so hohen Hardwareverbrauch[126]. Auf größeren FPGA, wie dem Xilinx Virtex-6 spielt dies jedoch keine wesentlich Rolle[127]. Auch die Komplexität der Schaltung wirkt sich beim Spartan-3E und Spartan-6 FPGA auf eine im Vergleich zu Opt-2 niedrigere maximale Taktfrequenz auf. Da aber bereits mit einem Xilinx Spartan-6 mehr als 100 MHz erreicht werden können, wurde kein weiterer Optimierungsschritt mehr vorgenommen[128].

Version (Bittiefe μ_{index})	Opt-0	Opt-1	Opt-2	Opt-5
Xilinx Spartan 3E, xc3s1200e-5fg320, speedgrade -5, min runtime				
Slices (8.672)	1111 (12%)	917 (10%)	1099 (12%)	1928(22%)
Multipler(28)	1 (3%)	1 (3%)	-	-
Bock-RAM (28 * 18 kBit)	-	-	-	6/28 (21%)
Max. Frequenz in MHz	99	101	91	69
ISE Level of Logic	11	7	8	15
Xilinx Spartan 6, SC6SLX75-FGG484, speedgrade -3, min runtime				
Slices (11.662)	498 (4%)	464 (3%)	447 (3%)	998 (8%)
Multipler (132)	-	-	-	-
Block-RAM (172 * 18 kBit)	-	-	-	6 (3 %)
Max. Frequenz in MHz	118	143	109	102
ISE Level of Logic	5	6	5	7
Xilinx Virtex 6, XC6VLX240T-FF1156, speedgrade -3, min runtime				
Slices (37.680)	522 (1%)	395 (1%)	458 (1%)	976 (2%)
Multipler()	-	-	-	-
Block-RAM (416 * 36 kBit)	-	-	-	3 (1 %)
Max. Frequenz in MHz	187	214	178	185
ISE Level of Logic	5	5	4	7

Abbildung 5.34: Daten zur FPGA-Implementation, Component Labeling inklusive Speichermanagement, Modell-1, Bildgröße $1024 * 1024$ Pixel, 4-Zusammenhang

Auf Abbildung 5.35 ist zum Vergleich der Energieverbrauch der FPGA-Implementationen unter unterschiedlichen Bedingungen dargestellt. Das komplette Bord, auf dem das FPGA aufgebracht ist, hat natürlich einen höheren Stromverbrauch[129]. Der Verbrauch eines Bords, das im Weltraum zum Einsatz kommen könnte, konnte im Rahmen dieser Arbeit nur abgeschätzt werden. Der gesamte Energieverbrauch eines speziell angepassten Bords mit beispielsweise Xilinx Spartan-6 bzw. Xilinx Virtex-6 FPGAs dürfte jedoch unter fünf

[126] Abbildung 5.34

[127] es wird daher davon ausgegangen, dass das CCL-Modul auf einer zukünftigen Plattform ohne Einschränkung umgesetzt werden kann

[128] die Xilinx Spartan-6 Reihe ist der Nachfolger der Xilinx Spartan-3 Reihe und ist leistungsmäßig am unteren Ende der Xilinx FPGA Familie angesiedelt

[129] zusätzliche Bauteile wie Speicher und I/O-Schnittstellen

beziehungsweise zehn Watt liegen.

Stromverbrauch	Spartan 3E @ 65Mhz	Spartan 6 @ 100Mhz	Virtex 6 @ 185Mhz
0 Grad (normal)	0,37	0,52	1,49
0 Grad (maximal)	0,45	0,58	2,72
25 Grad (normal)	0,39	0,56	1,82
25 Grad (maximal)	0,50	0,67	3,61
50 Grad (normal)	0,42	0,63	2,28
50 Grad (maximal)	0,56	0,88	5,03
85 Grad (normal)	0,47 (Warnung)	0,78	3,10 (Warnung)
85 Grad (maximal)	0,65 (Warnung)	1,18	50,24 (Fehler)

Abbildung 5.35: FPGA Implementierung, Stromverbrauch, Kommerziell, Component Labeling inklusive Speichermanagement, Modell-3, Bildgröße 1024*1024 Pixel, 4 Zusammenhang, ISE XPowerAnalyzer (Virtex ISE 12.1, alle anderen ISE 13.1)

5.6.5 Leistungsfähigkeit, Vergleich mit Software-Implementationen und Fazit

Um die Leistungsfähigkeit der FPGA-Implementationen des TPSS-Verfahrens einschätzen zu können, wurde ein Vergleich mit einer Softwareimplementierung des TPSS-Verfahrens durchgeführt. Die Softwareimplementation wurde zuerst auf einem leistungsfähigen Intel Core2Duo@2.80 GHz[130] ausgeführt. Diese CPU kommt aufgrund des Energieverbrauchs für einen On-Board Einsatz jedoch nicht in Frage. Sie wurde jedoch herangezogen, um die Performance, welche auf einem FPGA beziehungsweise einer relative leistungsfähigen CPU erzielt werden kann, zu vergleichen.
Die CPU Leistung, die auf einem Satelliten zur Verfügung gestellt werden kann, entspricht eher der eines eingebetteten Systems[131], wie dem ARM1176JZF-S@700 MHz, welches auf der RaspberryPi Plattform [80] verbaut ist, oder der eines weitaus leistungsfähigeren ARM® Cortex A9, welcher Teil des Xilinx Zynq-7000 ist. Das besondere an Xilinx Zynq ist, dass dieses eingebettete System ein FPGA enthält, dass dazu benutzt werden soll, rechenaufwendige Schritte in Hardware auszulagern[132]. Dieses System wird auch beim derzeit laufenden DLR Projekt OBC-NG verwendet, dessen Ziel es ist, eine moderne On-Board Computerplattform zu entwickeln [83]. Zu Testzwecken stand ein ZedBoard[133] zur Verfügung, welches auf dem Xilinx Zynq®-7000 SoC XC7Z020-1 basiert.
Die Ergebnisse dieses Vergleichs finden sich in Abbildung 5.36. Die Leistungs der FPGA-Implementation genügt den Anforderungen, dass kleine Bilder innerhalb einer Sekunde

[130] Thermal Design Power (TDP) 35W, Wert für die thermische Verlustleistung
[131] System on a Chip (SoC)
[132] Ein gute Übersicht über die Xilinx Zynq Familie findet sich in [81], und als Beispiel wie ein Sobel-Kantendetektionsfilter auf dem FPGA implementiert werden kann ist in [82] demonstriert.
[133] Hergestellt wird das ZedBoard unter anderem von Digilent, Dokumentation siehe [84]

und größere innerhalb weniger Sekunden gelabelt werden können[134]. Selbst mit einer Model-1@12.5 MHz Implementation kann ein Bild der Größe 1024*1024 Pixel unter einer Sekunde bearbeitet werden. Mit einer Modell-3@100 MHz Implementation kann ein sehr großes Bild mit $16384 * 16384$ Bildpunkten in ungefähr 7 Sekunden gelabelt werden.
Interessant ist, dass mit einer Modell-3@12.5 MHz Implementation in etwa die Leistung des ARM1176JZF-S@700 MHz erzielt werden kann und dass es zwei zu 100% ausgelastete relativ moderne ARM Cortex-A9@667 MHz Kerne benötigt, um in etwa die Leistung einer Modell-3@100 MHz Implementation zu erreichen.
Die Mehrkernleistung wurde gemessen, indem ganze Bilder auf verschiedenen Kernen gelabelt wurden. Die Softwareimplementation des TPSS-Verfahren bietet bis dato keine Mehrkernunterstützung. Zum Labeling eines einzigen Bildes kann daher nur ein Kern herangezogen werden. Es ist nicht ohne größere Modifikationen möglich eine Softwareversion des TPSS-Verfahren mit Mehrkernunterstützung zu implementieren und es ist fraglich, welcher Leistungsgewinn hierdurch überhaupt erreicht werden könnte.
Der schon etwas ältere Intel T9600@2.80 GHz[135] spielt in einer eigenen Liga und ist bereits mit einem Kern (2-3)-mal schneller als eine Modell-3@100 MHz Implementation. Man bedenke jedoch, dass dieser für Notebooks entwickelte Prozessor einen Leistungsverbrauch von bis zu 35 Watt hat (TDP) und eine FPGA-Implementation, beispielsweise auf einem Xilinx Spartan-6 FPGA, weniger als einen Watt benötigt[136].
In dieser Untersuchung wird auch deutlich, inwieweit das TPSS-Verfahren vom Bildinhalt abhängig ist. Beim Diagonal-1 Testbild, dem Worst-Case Szenario des TPSS-CCL Verfahrens bezüglich des 4-Zusammenhanglabelings, kommt es teilweise zu einem Einbruch der Leistung um Faktor 3 sowohl bei den Software, als auch den FPGA-Implementationen. Auch bei den sehr komplexen Labyrinth-Bildern müssen Einbußen in der Leistung hingenommen werden. Andererseits zeigt sich, dass das Verfahren bezüglich der Laufzeit relativ stabil ist, und mit dem TPSS-CCL Verfahren, auch unter ungünstigsten Umständen, eine hohe Verarbeitungsleistung erzielt werden kann.

Das Fazit, das hier gezogen werden kann ist erstens, dass die FPGA-Implementation des TPSS-Verfahrens die in Abschnitt 5.4.1 gestellten Anforderungen erfüllt, insbesondere auch sehr große Bilder in wenigen Sekunden bearbeitet werden können. Zweitens, dass mit einer Modell-3@100 MHz Implementation sogar Videoverarbeitung bis zu einer Bildgröße von 1024*1024 Pixel möglich ist, eingeschränkt sogar bis zu einer Größe von 4096*4096 Pixel. Drittens, dass eine Softwareimplementation des TPSS-CCL Verfahrens mit einer voll ausgelasteten modernen Low-Power CPU wie ein ARM Cortex-A9@667 MHz CPU eine ähnliche Leistung erzielen kann wie eine Modell-3@100 MHz Implementation. Das System ist hierbei jedoch zu 100% ausgelastet, wohingegen auf einer FPGA-Plattform parallel noch weitere Aufgaben bearbeitet werden könnten.
Bei einer Plattform wie dem Xilinx Zynq, bei der die Wahl besteht, das Labeling in Hardware oder Software zu implementieren, kann eine Umsetzung des CCL-Verfahrens

[134] Abschnitt 5.4.1
[135] Einführungsdatum 14. Juli 2008
[136] Abbildung 5.35

in Hardware dass Gesamtsystem entscheidend entlasten. Bei größeren Bildern und Videoanwendung wird man um eine Hardwareimplementation in vielen Fällen nicht herumkommen, gerade da neben dem Labeling im Normalfall viele weitere Aufgaben durchgeführt werden müssen.
Letztendlich konnte mit der FPGA-Implementation des TPSS-CCL Verfahrens gezeigt werden, dass auch das Labeling großer, komplexer Bilder[137] sehr performant mit einer FPGA Lösung bewältigt werden kann. Es wurde außerdem detailliert dargestellt, in welchem Maße dieses CCL-Verfahren vom Bildinhalt abhängig ist, und eine Begründung hierfür gegeben.

[137] beliebige Bildstruktur, beliebig viele Segmente, und Erzeugung einer Labelmaske (Abschnitt 5.2)

	Bilder pro Sekunde						
Bildgröße in Pixel (quadratisch)	50	512	1024	2048	4096	8192	16384
Testbilder	Notebook Dell Latitude E6: Intel T9600@2.80 GHz, One Thread						
Weiß	$84,7 * 10^3$	806	162	40,0	9,90	2,24	0,42
Dreieck	$83,3 * 10^3$	833	166	38,5	9,80	2,48	0,51
Diagonal 1	$83,3 * 10^3$	758	150	13,4	2,86	0,69	0,16
Diagonal 2	$12,5 * 10^4$	$126 * 10^1$	197	49,5	12,3	2,60	0,625
Labyrinth 1	$29,4 * 10^3$	318	66,5	16,4	4,18	1,00	0,235
Labyrinth 2	$29,4 * 10^3$	301	65,5	15,9	3,96	1,01	0,235
Testbilder	FPGA: Model-3@100 MHz						
Weiß	$20,0 * 10^3$	191	47,7	11,9	2,99	0,75	0,19
Dreieck	$18,8 * 10^3$	189	47,5	11,9	2,98	0,75	0,19
Diagonal 1	$7,85 * 10^3$	70.5	17,4	4,34	1,08	0,27	0,075
Diagonal 2	$20,0 * 10^3$	191	47,7	11,9	2,98	0,75	0,19
Labyrinth 1	$15,0 * 10^3$	144	36,0	8,98	2,25	0,56	0,14
Labyrinth 2	$15,0 * 10^3$	140	34,8	8,67	2,17	0,54	0,14
Testbilder	FPGA: Model-3@12.5 MHz						
Weiß	$2,50 * 10^2$	23,8	5,96	1,49	0,373	0,0931	0,0233
Dreieck	$2,35 * 10^2$	23,7	5,94	1,49	0,373	0,0931	0,0233
Diagonal 1	$9,82 * 10^2$	8,73	2,18	0,543	0,136	0,0339	0,00847
Diagonal 2	$2,50 * 10^2$	23,8	5,96	1,49	0,373	0,0931	0,0233
Labyrinth 1	$1,88 * 10^2$	18,1	4,50	1,12	0,281	0,0701	0,0175
Labyrinth 2	$1,88 * 10^2$	17,4	4,35	1,08	0,271	0,0677	0,0169
Testbilder	FPGA: Model-1@12.5 MHz						
Weiß	714	6,81	1,70	0,426	0,106	0,0266	0,00665
Dreieck	693	6,79	1,70	0,425	0,106	0,0266	0,00665
Diagonal 1	406	3,69	0,919	0,230	0,0574	0,0143	-
Diagonal 2	714	6,81	1,70	0,426	0,106	0,0266	0,00655
Labyrinth 1	618	5,91	1,48	0,369	0,0921	0,0230	-
Labyrinth 2	618	5,81	1,45	0,362	0,0904	0,0226	-
Testbilder	RaspberryPi: ARM1176JZF-S@700 MHz (ARMv6)						
Weiß	$4,63 * 10^2$	25,3	6,08	1,21	0,261	-	-
Dreieck	$4,35 * 10^2$	25,8	6,09	1,24	0,262	-	-
Diagonal 1	$4,67 * 10^2$	12,7	2,92	0,694	0,162	-	-
Diagonal 2	$5,46 * 10^2$	29,4	7,35	1,73	0,489	-	-
Labyrinth 1	$3,62 * 10^2$	17,3	3,61	0,776	0,164	-	-
Labyrinth 2	$3,62 * 10^2$	16,9	3,49	0,738	0,156	-	-
Testbilder	ZedBoard: Dual ARM Cortex-A9 MPCore@667 MHz (ARMv7), One Thread						
Weiß	$12,8 * 10^3$	88,2	21,6	5,08	1,19	-	-
Dreieck	$11,9 * 10^3$	86,7	21,1	4,97	1,16	-	-
Diagonal 1	$13,0 * 10^3$	47,5	9,13	2,04	0,464	-	-
Diagonal 2	$16,8 * 10^3$	117	29,0	7,10	1,74	-	-
Labyrinth 1	$7,72 * 10^3$	58,7	13,4	2,98	0,671	-	-
Labyrinth 2	$7,68 * 10^3$	57,7	13,1	2,88	0,646	-	-
Testbilder	ZedBoard: Dual ARM Cortex-A9 MPCore@667 MHz (ARMv7), Two Threads						
Weiß	$(12,7 + 12,7) * 10^3$	84,3 + 84,4	21,2 + 21,2	5,08 + 5,08	1,17 + 1,17	-	-
Dreieck	$(12,4 + 11,8) * 10^3$	84,6 + 82,6	20,2 + 20,2	5,01 + 4,99	1,17 + 1,17	-	-
Diagonal 1	$(13,2 + 12,8) * 10^3$	36,9 + 36,9	8,45 + 8,42	1,98 + 1,98	0,447 + 0,446	-	-
Diagonal 2	$(18,3 + 18,2) * 10^3$	104 + 105	26,9 + 26,6	6,85 + 6,83	1,63 + 1,62	-	-
Labyrinth 1	$(7,68 + 7,60) * 10^3$	55,9 + 55,9	12,8 + 12,8	2,89 + 2,89	0,618 + 0,618	-	-
Labyrinth 2	$(7,72 + 7,69) * 10^3$	56,2 + 56,1	12,4 + 12,4	2,77 + 2,76	0,594 + 0,591	-	-

Abbildung 5.36: 4 Zusammenhang Labeling, Vergleich verschiedener Plattformen; Software Plattformen: Notebook Dell Latitude E6: Intel T9600@2.80 GHz, 4.0 GB DDR3, TPM 35 W, gcc 4.7.1, compiler options O3, mtune native, 64bit; RaspberryPi Model-B: ARM1176JZF-S (700 MHz), 512 MB RAM, 3.5 W, gcc, compiler options -march=armv6 -mfpu=vfp -mfloat-abi=hard; Zed Board: Xilinx Zynq-7000 SoC XC7Z020-1, 512 MB DDR3, arm-xilinx-linux-gnueabi-gcc (gcc version 4.8.1) -O3

6 Schlussbemerkung/Zusammenfassung

6.1 Schlussbemerkung

Diese Arbeit wurde, als Teil der Gruppe On-Board-Classification, am Deutschen Zentrum für Luft- und Raumfahrt durchgeführt. Es sollen Verfahren zur Bewertung optischer Fernerkundungsdaten an Bord eines Satelliten bereitgestellt werden.
Ein wichtiger Schritt in der Verarbeitungskette stellt das Verfahren der Bildsegmentation dar. Dieser Schritt ist jedoch gerade bei großen Datenmengen sehr rechenaufwendig. Deshalb ist eine zeitnahe Durchführung auf einem gewöhnlichen On-Board Computer nur eingeschränkt möglich. Eine Möglichkeit, die Prozessierung selbst großer Datenmengen zeitnahe durchzuführen zu können, stellt die Implementation dieses Verfahrens auf einem FPGA dar.
In dieser Arbeit wurde aufgezeigt, wie eine Methode zur Bildsegmentation auf einem FPGA implementiert werden kann, mit welcher auch sehr große und komplexe Datenmengen, wie sie in der Fernerkundung vorkommen, zeitnahe prozessiert werden können. Das Augenmerk wurde vor allem darauf gelegt, dass die Segmentation nach einem möglichst allgemeinen Kriterium erfolgt, um sie in möglichst vielen verschiedenen Anwendungsszenarien verwenden zu können. Um die Wiederverwendbarkeit weiter zu erhöhen, wurde ein modulares Entwicklungskonzept verfolgt. Der Kern der Segmentationsmethode, die kantenerhaltende Gap-Glättung, kann für viele Aufgaben, bei denen es sinnvoll ist ein Bild (kantenerhaltend) zu glätten, separat verwendet werden. Auch das ebenfalls in dieser Arbeit entwickelte, speziell auf FPGA zugeschnittene Connected Component Labeling Modul, dass in der Lage ist sehr große und komplexe Bilder zu prozessieren, kann separat eingesetzt werden.
Gerade bei bei einem On-Board Einsatz, ist das Laufzeitverhalten der Prozessierungseinheiten oft von entscheidender Bedeutung.Das Laufzeitverhalten, der in dieser Arbeit vorgestellten Verfahren, wurde daher genauestens analysiert. Im Falle des CCL-Verfahrens, dessen Laufzeit vom Bildinhalt abhängig ist, wurde hierzu sogar ein Laufzeitmodell entwickelt, mit welchem die Laufzeit taktgenau bestimmt werden kann. Um die Leistungsfähigkeit der FPGA-Implementationen einordnen zu können, erfolgte außerdem ein Vergleich mit verschiedenen Softwareimplementationen.
Das Besondere der in dieser Arbeit entwickelten Verarbeitungseinheiten ist insbesondere, dass sie darauf ausgelegt sind, große komplexe Bilder, wie sie in der Fernerkundung häufig auftreten, zeitnah verarbeiten zu können. Das unterscheidet sie von fast allen Hardwareimplementationen in diesem Bereich[1], welche vorrangig zur Prozessierung eines Videodatenstroms entworfen wurden und in erster Linie darauf ausgelegt sind, Bilder mit

[1] i.a.W. Bildsegmentation und CCL

moderater Bildgröße und beschränkter Bildkomplexität, jedoch mit einer hohen Bildwiederholrate prozessieren zu könne.
Ein weiterer Schwerpunkt dieser Arbeit ist die Bewertung der Qualität von Segmentationsverfahren. Diese wurde zur Entwicklung des Segmentationsprozesses benötigt. Mit Hilfe der in dieser Arbeit entwickelten Testmethode gelang es jedoch außerdem, die Leistung des Segmentationsverfahrens besser einschätzen und darstellen zu können und einen Vergleich mit alternativen Verfahren durchzuführen. Die hierzu entwickelte Theorie und Methodik lässt sich im Allgemeinen zur Bewertung von Segmentationsverfahren verwenden und liefert daher einen wissenschaftlichen Beitrag in diesem Feld.
Im nächsten Abschnitt wird auf die Einzelergebnisse der Arbeit genauer eingegangen.

6.2 Zusammenfassung und Ergebnisse

Die in Abschnitt 1.4 gestellten (wissenschaftlichen) Fragestellungen wurden in den vier Kapiteln – *Bildsegmentation, Gap-Segmentation, Qualitätsuntersuchung und Connected-Component Labeling* – erörtert.
Es wurden folgende **Ergebnisse** erzielt:

Kapitel 2: Bildsegmentation

In diesem Kapitel wurde aufgezeigt, wie eine Bildsegmentation, welche für viele Anwendungen in der Fernerkundung geeignet ist, auf einem FPGA durchgeführt werden kann. Dazu wurde zuerst dargelegt, was unter der Segmentation eines Bildes und der Güte einer Segmentation zu verstehen ist. Anschließend wurden, anhand der (wissenschaftlichen) Zielsetzung, Anforderungen an die Segmentationsmethode abgeleitet (Abschnitt 2.3). Die Segmentation sollte nach dem Merkmal des mittleren Grauwertes erfolgen. Sie muss vollständig automatisch durchgeführt werden. Außerdem sollte die Segmentationsmethode von der Form, Größe und Anzahl der Segmente unabhängig sein. Es sollen außerdem sehr große Bilder verarbeitet werden können. Damit ein Einsatz bei echtzeitkritischen Anwendungen erfolgen kann, sollte die Laufzeit der Methode möglichst genau abgeschätzt werden können. Es wurde ebenfalls die Zielsetzung getroffen, dass kleine Bilder[2] in wenigen Sekunden und großen Bilder[3] in wenigen Minuten bearbeitet werden können.
Ausgehend von diesem Anforderungsprofil wurden Segmentationsmethoden auf ihre prinzipielle Eignung untersucht. Zu dieser Untersuchung wurden ebenfalls bereits existierende Hardware-/FPGA-Implementationen herangezogen.
Die Hauptproblematik, welche bei der Eignungsuntersuchung auftrat, war, dass die meisten der Verfahren nicht im Hinblick auf Fernerkundungsanwendungen entwickelt wurden. Da in den meisten Fällen auch keine Implementation (z.B. Softwarevariante) der Verfahren zur Verfügung standen und die Algorithmen in den meisten Fällen nur ungenügend dokumentiert wurden, war eine zufriedenstellende Einschätzung oftmals nicht möglich.

[2] $< 2000 * 2000$ Bildpunkte
[3] $\sim 10000 * 10000$ Bildpunkte

Auch wurden fast alle Hardware/FPGA-Implementationen zur Bearbeitung eines Videodatenstroms entworfen, nicht zur Verarbeitung von großen und komplexen Einzelbildern. Es fand sich jedoch eine Arbeit von Halle [38], in welcher ein Segmentationsverfahren von Jahn [36] ausführlich beschrieben und speziell für die Anwendung auf Fernerkundungsbilddaten, auch im Rahmen einer möglichen On-Board Datenverarbeitung und der Realisierung auf einer zukünftigen parallelen Prozessierungseinheit, untersucht wurde.
Dieses Verfahren entsprach fast allen Anforderungen. Nur sind viele Schritte des Algorithmus für eine FPGA-Implementation ungeeignet (Abschnitt 2.6). Es wurde daher, von dieser Methode ausgehend, die Gap-Segmentationsmethode entwickelt, welche speziell im Hinblick einer FPGA-Implementation entworfen wurde.

Kapitel 3: Gap-Segmentation

In diesem Kapitel wird das Gap-Segmentationsverfahren vorgestellt, welches im Hinblick einer möglichst günstigen FPGA-Implementation entwickelt wurde. Außerdem wurden die Ausführungszeit und der Ressourcenverbrauch verschiedener Modellimplementationen ermittelt.
Der Segmentationsprozess besteht aus drei wesentlichen Verarbeitungsschritten:

1. (Vor-)Filterstufe
2. Segmentationsstufe
3. Connected Component Labeling

In der *(Vor-)Filterstufe* werden lokale Variationen der Grauwerte[4] entfernt. Im Gegensatz zu einem gewöhnlichen Glättungsfilter bleiben jedoch die Bildkanten größtenteils erhalten und werden nicht verwaschen (Abschnitt 3.2.1). Es hat sich als günstig herausgestellt, zwei Filterstufen zu verwenden, einen gewöhnlichen Mittelwertfilter und ein kantenerhaltenden Filter (Gap-Glättungsfilter).
In der *Segmentationsstufe* wird der lokale Zusammenhang benachbarter Bildpunkte ermittelt. Es wurde ein sehr einfaches regionorientiertes Verfahren benutzt, bei dem zwei direkt benachbarte Bildpunkte verbunden werden, wenn ihre Grauwertdifferenz kleiner als ein vorgegebener Schwellwert ist.
Beim sogenannten *Connected Component Labeling* wird aus dem lokalen Zusammenhangsdaten der globale Zusammenhang ermittelt. Es wird eine Labelmaske erstellt, bei welcher jedem Segment ein Index zugeordnet wird (Kapitel 5).
Den rechenaufwendigsten Schritt des Gap-Segmentationsverfahrens stellt die kantenerhaltende Gap-Glättung dar, welche in dieser Arbeit speziell in Hinblick einer FPGA-Implementation entworfen wurde. Um die Kanten zu erhalten, wird zuerst die 3×3-Umgebung eines Bildpunktes analysiert (Gap-Detektion) und nur über Bildpunkte gemittelt, welche einen ähnlichen Grauwert haben (Abschnitt 3.2.2). Dieser Schritt ist komplex, beruht jedoch auf Operationen, welche sehr gut in Hardware umgesetzt werden konnten. Es werden ausschließlich 12-Bit Ganzzahloperationen verwendet, und bis

[4] Bsp: lokale Störungen, Rauschen, Textur

auf eine Division ausschließlich Additionen und einige wenige Multiplikationen benötigt (Abschnitt 3.2.2). Den Kern der Gap-Detektion stellen Sortiereinheiten dar, bei welchen jeweils acht Zahlen eines Datenstroms, der Größe nach sortiert werden. Dieser Schritt kann in Hardware, im Gegensatz zu einer Softwarelösung, sehr effizient durchgeführt werden.
Beim gesamten Gap-Glättungsprozess konnte eine Datenstromverarbeitung durchgeführt werden (Abschnitt 3.4). Werden neun Glättungseinheiten parallel betrieben, kann ein Bildpunkt pro Takt verarbeitet werden.
Auf einem Spartan6-FPGA@100 MHz kann der gesamte Gap-Segmentationsprozess eines Bildes mit $10000 * 10000$ Pixel in weniger als $1\frac{1}{2}$ Minuten verarbeitet werden, wogegen ein Intel Core2Duo@2.8 GHz hierfür ungefähr 36 Minuten benötigt. Dies ist umso Beeindruckender, wenn der Energieverbrauch betrachtet wird, der bei der FPGA-Implementierung ungefähr bei einem Watt liegt, wohingegen die CPU mehr als 30 Watt benötigt.

Kapitel 4: Qualitätsuntersuchung

In diesem Kapitel wurde aufgezeigt, wie die qualitative Leistungsfähigkeit eines Segmentationsverfahrens bestimmt werden kann. Mit Hilfe des in dieser Arbeit entwickelten Bewertungsverfahren, konnten (optimale) Parameter und die qualitative Leistungsfähigkeit des Gap-Segmentationsprozesses ermittelt werden, und die Gap-Segmentation mit alternativen Segmentationsverfahren verglichen werden.
Es war kein Standardverfahren bekannt mit welchem eine (qualitative) Segmentationsbewertung durchgeführt werden konnte. Auch fand sich in der Literatur keine geeignete Methode, welche direkt eingesetzt werden konnte. Es musste daher ein Bewertungsverfahren entwickelt werden. Die größte Herausforderung, die bewältigt werden musste, war, den Begriff der Güte einer Segmentation exakt zu definieren und eine Methode anzugeben, mit welcher diese ermittelt werden kann (Abschnitt 4.1.2).
Es wurden folgende Anforderungen an die Segmentationsbewertung gestellt. Sie sollte objektiv, und möglichst einfach durchführbar sein. In erster Linie sollte jedoch eine hinreichende Qualitätsbestimmung ermöglicht werden, deren Ergebnisse möglichst gut interpretiert werden können (Abschnitt 4.1.3). Es wurden verschiedene Arten der Bewertung auf ihre Eignung untersucht, und ermittelt, dass sich eine überwachte Bewertungsmethode, vor allem auf Grund der guten Interpretationsmöglichkeit, am besten eignet (Abschnitt 4.1.4)[5]
Um beschreiben zu können, wie dieser Vergleich durchgeführt wurde, musste zuerst exakt definiert werden, was unter einem Bild, einer Segmentation und der Güte einer Segmentation überhaupt zu verstehen ist (Abschnitt 4.2). Anschließend wurde die SA_{EQ} Vergleichsfunktion vorgestellt, mit welcher zwei Segmentationen verglichen werden können (Abschnitt 4.3). Es wurde vom theoretischen Standpunkt aus und an Beispielen gezeigt, dass dieses Verfahren geeignet ist.
Es stellte sich im Rahmen der Untersuchung als günstig heraus, synthetische Testbilder zu verwenden (Abschnitt 4.4). Die Gründe hierfür waren, dass eine Mustersegmentation

[5] Bei der überwachten Bewertung wird das Segmentationsergebnis mit einer Mustersegmentation verglichen.

zur Verfügung stand, exakt beschreibbare Störungen auf-modelliert werden konnten und sich die Testergebnisse sehr gut interpretieren ließen. Mit realen Szenen war dies nicht möglich.
Mit Hilfe dieser Testmethode konnten optimale Parameter ermittelt werden (Abschnitt 4.5). Es wurde gezeigt, dass die Kombination von zwei Filterstufen sehr nützlich ist. Außerdem konnte die Anzahl der minimalen Filterläufe bestimmt werden, welche für ein gutes Segmentationsergebnis benötigt werden.
Es wurde der Begriff der Empfindlichkeit einer Segmentationsmethode eingeführt, mit dem die Leistungsfähigkeit der Segmentationsmethode bestimmt werden kann. Es konnte genau beschrieben werden, welchen Einfluss der Kontrast, die Rauschstufe und die Kantenschärfe auf ein Segmentationsergebnis haben. Ohne die, in dieser Arbeit, entwickelten Bewertungsmethoden hätten derartige Ergebnisse nicht erzielt werden können.
Es wurde außerdem ein Vergleich mit alternativen Segmentationsmethoden durchgeführt (4.6). Es konnte gezeigt werden dass die Gap-Segmentationsmethode leistungsfähiger ist als die zugrundeliegende Jahn-Segmentation. Es wurde auch gezeigt, dass das dynamische Gap-Segmentationsverfahren im Vergleich zu statischen Verfahren Vorteile hat und daher als sinnvoll anzusehen ist.
Zuletzt wurde anhand einer panchromatischen Landsat-7 Szene demonstriert, dass die mit den synthetischen Testdaten gewonnenen Ergebnisse auf reale Daten übertragbar sind (Abschnitt 4.7). Die Ergebnisse der Segmentation konnten in vielen Fällen erklärt werden. Es wurde außerdem an realen Daten gezeigt, wie die Bewertungsmethode herangezogen werden können, um verschiedene Segmentationsergebnisse vergleichen und analysieren zu können.
Es sei abschließend darauf hingewiesen, dass die in diesem Abschnitt entwickelte und angewandte Bewertungsmethode aus folgendem Grund entwickelt wurde: Es fand sich in der Literatur keine geeignete Methode, mit welcher objektiv, gut interpretierbare Aussagen getroffen werden konnten. Dass fängt damit an, dass in fast allen Arbeiten nur grob dargestellt wird, was den unter der Güte einer Segmentation überhaupt zu verstehen ist, und vor allem überhaupt nicht klar war, wie diese exakt beschrieben werden kann. Ein weiteres Hindernis welches bei der Bewertung auftrat, war dass die in der Literatur verwendeten Methoden, meistens kaum interpretierbar sind, i.a.W. sich daraus nur schwer Erkenntnisse ableiten lassen. In der Arbeit wurde daher, eine Bewertungsmethode welche sich in der Literatur fand analysiert, und dahingehend erweitert, dass mit dieser Methode eine vernünftige, gut interpretierbare Bewertung durchgeführt werden kann. Zuletzt stellt sich die Frage, was bei der Bewertung einer Segmentation gezeigt werden soll. Zum Beleg der Güte einer Segmentationsmethode werden oft nur Beispiele gezeigt. Dies ist wichtig, doch ist es damit kaum möglich vorherzusagen, wie sich die Methode in einer anderen Situation verhält. In dieser Arbeit wurde daher ein anderer Ansatz gewählt, um die Leistungsfähigkeit einer Segmentationsmethode darzustellen. Es wurde zuerst eindeutig festgestellt, welche Eigenschaft einer Segmentationsmethode überhaupt bewertet werden soll. Ein nach Meinung des Autors sehr wichtiges Leistungskriterium einer Segmentationsmethode ist, wie groß der Grauwertunterschied, bei einer vorgegeben Rauschstufe sein muss, damit zwei Objekte möglichst gut getrennt werden können. Eine Bewertung nach dieser Charakteristik fand sich in der Literatur ebenfalls nicht. Der Messaufbau

wurde derart gewählt, dass diese Charakteristik möglichst gut bestimmt werden kann. Dies ist der Grund warum synthetische Messbilder verwendet wurden. Insgesamt wird eine Bewertungsmethodik vorgestellt, die so in der Literatur noch nicht zu finden war, und zum Test beliebiger Segmentationsmethoden verwendet werden kann. Diese Arbeit leistet daher einen großen Beitrag im Feld der Segmentationsbewertung, und zeigt insbesondere eine Möglichkeit auf, wie eine erste Bewertung einer Segmentationsmethode durchgeführt werden kann.

Kapitel 5: Connected Component Labeling

In diesem Kapitel wurde eine Connected Component Labeling Methode vorgestellt, welche vorteilhaft auf einer FPGA Plattform implementiert werden kann. Das Verfahren wurde aufgrund des Einsatzschwerpunktes, der On-Board Bildbewertung von Fernerkundungsbildern in Echtzeit, darauf ausgelegt, Fernerkundungsbilder mit komplexen Inhalten labeln zu können.
Es wird ein Indexbild erstellt, bei dem jedem Segment/Bildbereich ein Index zugeordnet wird. Es können sehr große Bilder gelabelt werden. Es gibt keine Einschränkung an den Bildinhalt, d.h. an die Form, Größe und Anzahl der auftretenden Segmente. Dadurch unterscheidet sich dass in dieser dieser Arbeit entwickelte CCL-Modul von existierenden Hardware/FPGA-Lösungen, welche im Hinblick einer Videoprozessierung entworfen wurden.
Es wurden mehrere Verfahren untersucht. Dabei hat sich die Two-Pass Methode, basierend auf Union-Find Datenstrukturen, als die geeignetste herausgestellt. Daher wurde ein speziell an diese Anforderungen angepasster Two-Pass Algorithmus entwickelt, und dessen korrekte Arbeitsweise bewiesen.
Der Algorithmus konnte vorteilhaft auf einem FPGA implementiert werden. Es wurden mehrere Versionen entworfen, in Abhängigkeit der parallel durchführbaren Speicherzugriffe. Es konnte größtenteils eine Datenstromverarbeitung durchgeführt werden, mit welcher eine hohe Verarbeitungsleistung erreicht wurde.
Die Verarbeitungsleistung wurde anhand mehrerer Modellszenarien abgeschätzt. Können zwei 24 Bit Speicherzugriffe durchgeführt werden, kann ein Bild mit $8192 * 8192$ Bildpunkten auf einem FPGA, das mit 100 MHz betrieben wird, in ungefähr zwei Sekunden gelabelt werden. Es können 8 Bilder mit $2048 * 2048$ Bildpunkten und mehr als 30 Bilder mit $1024 * 1024$ Bildpunkten pro Sekunde verarbeitet werden. Eine derartige Implementation könnte auf einem (low-cost) Spartan 6 FPGA realisiert werden, wobei der Ressourcenverbrauch bei mittelgroßen bis großen Modellen zwischen 5% - 10% liegt. Auf größeren FPGAs, wie dem Virtex 6 FPGAs, liegt der Ressourcenverbrauch bei weit unter 3% und es können Taktraten weit über 150 MHz erzielt werden. Steht nur ein 24 Bit Speicherzugriff pro Takt zur Verfügung, wird ungefähr die doppelte Zeit benötigt. Aber auch dies ist für die meisten Anwendungen völlig ausreichend.
Es wurde ebenfalls ein Vergleich mit einer Softwareimplementation durchgeführt, wobei sich herausstellte, dass mit einer FPGA-Implementation, welche mit 12,5 MHz betrieben wird, bereits die Leistung eines Raspberry Pi erreicht wird, welches mit 700 MHz getaktet ist. Auf einer geeigneteren FPGA-Plattform, wie beispielsweise einem Xilinx

Spartan-6, kann das CCL-Modul mit über 100 MHz betrieben werden, und es wird eine höhere Leistung erbracht, als mit zwei zu 100% ausgelastete Kernen des relativ modernen ARM Cortex-A9@667MHz. Mit dieser Leistungsuntersuchung kann abgeschätzt werden, inwiefern eine Softwareimplementation ausreicht, oder ob eine FPGA-Implementation notwendig ist.

6.3 Fazit und Ausblick

Das Hauptziel, den rechenaufwendigen Segmentationsprozess auf einem FPGA umzusetzen, konnte erreicht werden.
Der gesamte Prozess kann selbst bei sehr großen Bildern[6], mit vertretbarem Hardwareaufwand, in weniger als $1\frac{1}{2}$ Minuten durchgeführt werden. Kleinere Bilder[7] können in weniger als einer Sekunde bearbeitet werden. Der Energieverbrauch einer zukünftigen Verarbeitungseinheit, basierend auf FPGA, dürfte unter drei Watt liegen.
Das Besondere an dieser Arbeit ist, dass große komplexe Daten bearbeitet werden können. Im Gegensatz dazu sind Verfahren in der Literatur in aller erster Linie darauf ausgelegt Videodaten zu bearbeiten. Die Einzelbilder sind hierbei weitaus kleiner und weniger komplex.
Es konnte letztendlich gezeigt werden, dass die grundlegenden Verarbeitungsschritte, die Segmentation und das Connected Component Labeling, welche für viele Bildanalyseverfahren benötigt werden, an Bord eines Satelliten mit vertretbarem Aufwand durchgeführt werden können. Selbst große Datenmengen, wie sie in der Fernerkundung üblich sind, können verarbeitet werden. Dies wäre mit einer gewöhnlichen Softwareimplementation nur schwerlich zu erreichen.
Ein besonderer Wert wurde auf die Qualitätsanalyse des Segmentationsverfahrens gelegt. Das Problem, das hierbei auftrat, bestand darin, dass keine Methode zur Verfügung stand, mit der die Qualität analysiert werden konnte. Es musste daher eine eigene Methode entwickelt werden, welche in dieser Arbeit ausführlich behandelt wurde. Mit dieser Methode konnte die (qualitative) Leistungsfähigkeit einer Segmentationsmethode auf eine Art und Weise charakterisiert werden, wie sie ohne die Testmethode niemals gewonnen hätte werden können. Ein Anwender kann hiermit die Leistungsfähigkeit einschätzen. Auch können mit dieser Methode verschiedene Verfahren objektiv verglichen werden. Dies ist eine Besonderheit dieser Arbeit, wie sie der Autor in dieser Art und Weise nicht in der Literatur gefunden hat.
Einige der in dieser Arbeit entwickelten Module und Methoden, sollen in zukünftigen Projekten der Gruppe On-Board-Classification, wie der On-Board Schiffsdetektion, Verwendung finden. Eine Frage die sich hierbei stellen wird, ist, wie sich die leistungsfähigen FPGA-basierten Verarbeitungsmodule, möglichst gewinnbringend in ein Gesamtsystem integrieren lassen. Besonderes Augenmerk wird hierbei auf das effiziente Zusammenspiel zwischen den FPGA- und den Softwarekomponenten gelegt werden müssen. Gerade in Verbindung mit den neuartigen sehr leistungsfähigen eingebetteten hybriden

[6] $\sim 10000 * 10000$ Bildpunkte
[7] $\sim 1000 * 1000$ Bildpunkte

CPU-/FPGA-Plattformen, wie der Xilinx Zynq-7000 Reihe [81], könnten sich hier viele neue Möglichkeiten ergeben, und Anwendungen, welche bisher aufgrund mangelnder Rechenleistung für einen On-Board Einsatz nicht in Frage kamen, realisiert werden.

7 Anhang

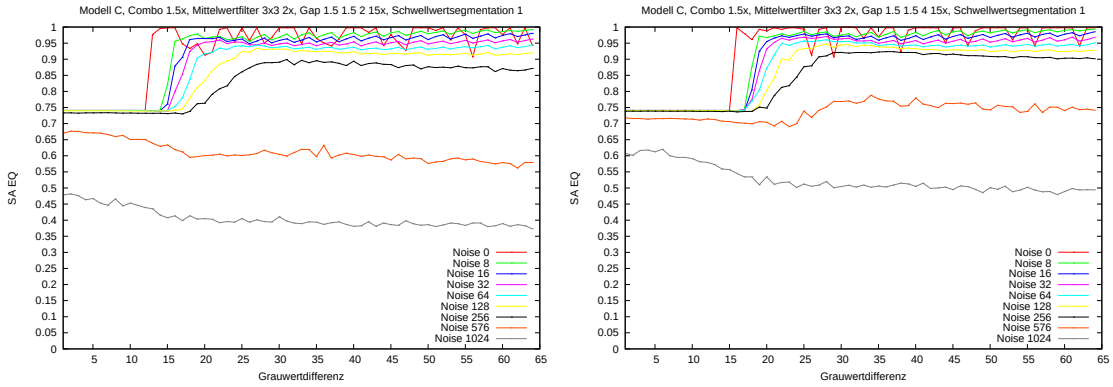


Abbildung 7.1: Modell-C, Combo 1.5x, Gap Glättung links Filtereinstellung a, rechts Filtereinstellung b

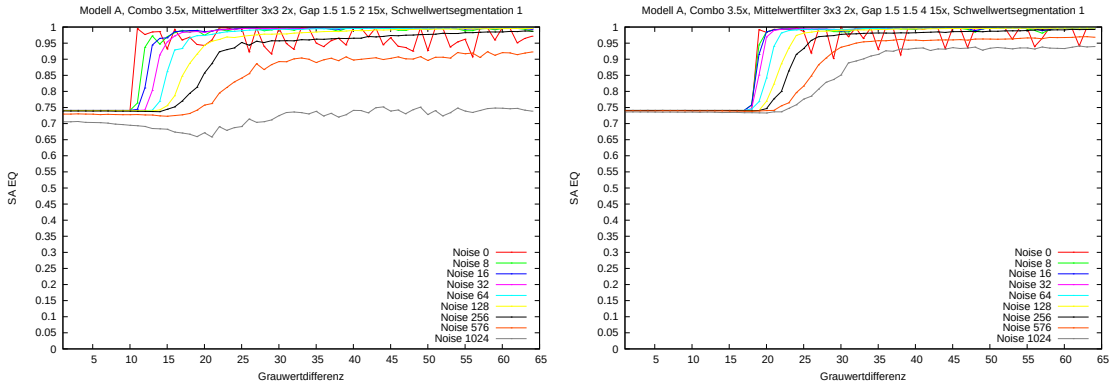


Abbildung 7.2: Modell-A, Combo 3.5x, Gap Glättung links Filtereinstellung a, rechts Filtereinstellung b

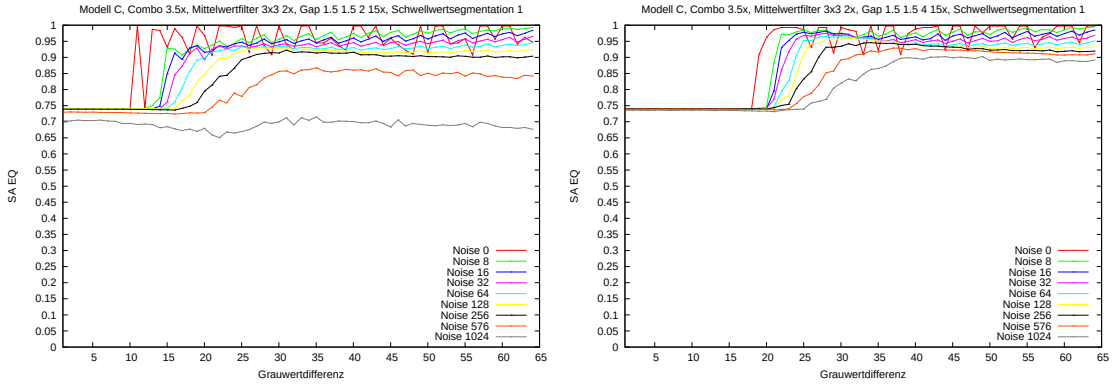


Abbildung 7.3: Modell-C, Combo 3.5x, Gap Glättung links Filtereinstellung a, rechts Filtereinstellung b

```
int hoehe   = #Bildhoehe;
int breite  = #Bildbreite;
int Lut[#max_nr_of_labels]                                  //#
int img_seg[hoehe][breite];                                 //Labelmaske
extern uchar img_zsh[hoehe][breite];
/*lokaler Zusammenhang es Bildes ist hier codiert
3x3 Umgebung von c
c: Zentraler Punkt

               ****************
               * 1  * 8  * 32 *
               * 2  * c  * 64 *
               * 4  * 16 * 128*
               ****************

Beispiel:
     *
   * c   : mit obigen und linken nachbaren verbunden: zsh(c) = 2 + 8
*/
int label_central, label_left, label_top, NewLabel = 0;
uchar zsh, con_left, con_top;

// Erster Lauf
for(i=1; i<hoehe; i++)
{
    for(j=1; j<breite; j++)
    {
        zsh = img_zsh[i][j]                                 //mem_read

                if (zsh == #background_code)
                {
                        label_central = #BackgroundLabel;
                        continue;
                }

        con_left = (zsh & 2) >> 1;
        con_top  = (zsh[i][j] & 8) >> 4;

        label_left =  img_seg[i][j-1];
        label_top = img_seg[i-1][j];

        if (con_left == 0  && con_top == 0)      label_central = NewLabel++;
        else if (con_left == 1 && con_top == 0) label_central = label_left;
        else if (con_left == 0 && con_top == 1) label_central = label_top;
        else if (con_left == 1 && con_top == 1 && (Lut[label_left] == Lut[label_top])
        {
            label_central = label_left;
        }
        else if ( con_left == 1 && con_top == 1 && (Lut[label_left] != Lut[label_top])
        {
            min_label = min(Lut[label_left], Lut[label_top]);
            max_label = max(Lut[label_left], Lut[label_top]);
            label_central = min_label;
            for(k=0; k<=Newlabel; k++)
            {
                if (Lut[k] == max_label) Lut[k]=min_label; //mem_read + if True: mem_write
            }
        }

        img_seg[i,j] = label_central;                       //mem_write
    }
}

// Zweiter Lauf
for(i=1; i<hoehe; i++)
{
    for(j=1; j<breite; j++)
    {
        img_seg[i,j] = Lut[img_seg[i,j]];                   //mem_read + mem_write
    }

}
```

Abbildung 7.4: Pseudo-Programmcode eines (klassischen) Two-Pass-Verfahrens angelehnt an das in [77] beschriebene Verfahren, 4-Zusammenhang

```
1  int hoehe   = #Bildhoehe;
2  int breite  = #Bildbreite;
3  int img_seg[hoehe][breite];                                          //label image
4  extern uchar img_zsh[hoehe][breite];                                 //zsh image
5
6  init_stack(stack);
7
8  int act_label=1;
9  for(i=0; i<hoehe; i++)
10 {
11     for(j=0; j<breite; j++)
12     {
13         uchar zsh_code = img_zsh[i][j];                              //mem_read
14
15         //Check if background
16         if(zsh_code == zsh_background_code)
17         {
18             img_label[i][j] = background_label;                      //mem_write
19             continue;
20         }
21
22         //Load Label
23         index = img_label[i][j];                                     //mem_read
24
25         //Already Labeled Case
26         if(index != 0) continue;
27
28         put_neighbours_to_stack(...);                                //see function
29
30         img_label[i][j]=act_label;                                   //mem_write
31
32         while(stack->stacklevel != 0)
33         {
34             stack_width = stack->data_widht[stack->stacklevel];      //mem_read
35             stack_height = stack->data_height[stack->stacklevel];
36             stack->stacklevel--;
37
38             zsh_code = img_zsh[stack_height][stack_width];           //mem_read
39             put_neighbours_to_stack(...);                            //see function
40         }
41         act_label++;
42 }
43
44 put_neighbours_to_stack(zsh_code, pos_x, pos_y, act_label, stack)
45 {
46     if((zsh_code & code_top) == code_top)
47     {
48         act_index = img_label[pos_y][pos_x];                         //mem_read
49
50         if(act_index == 0) //Not labeld yet
51         {
52             put_on_stack(pos_x, pos_y-1, stack);                     //see function
53             img_label[pos_y-1][pos_x]= act_label;                    //mem_write
54         }
55     }
56
57     //Analog
58     if((zsh_code & code_bottom) == code_bottom)
59     if((zsh_code & code_right)  == code_right)
60     if((zsh_code & code_left)   == code_left)
61
62     //Analog 8 connected only
63     if((zsh_code & code_upper_left) == code_upper_left)
64     if((zsh_code & code_bottom_left) == code_bottom_left)
65     if((zsh_code & code_upper_right) == code_upper_right)
66     if((zsh_code & code_bottom_right) == code_bottom_right)
67 }
68
69 put_to_stack(posx, posy, stack)
70 {
71     stack->stacklevel++;
72     stack->data_height[stack->stacklevel] = posy;                    //mem_write
73     stack->data_widht[stack->stacklevel] = posx;
74 }
```

Abbildung 7.5: Pseudo-Programmcode des zu Testzwecken implementierten Region-Growing-Verfahrens, 4-Zusammenhang

Laufzeitanalyse Algorithmus erster Lauf (detailliert, 8 Zusammenhang))

Fall 1 - Falls $con_{M_8}(P) = \emptyset$
$\mu_{addr}(P) := P$
Fall 2a - Falls $\mu_{addr}(con_{M_8}(P)) = \{a\}$ (einelementig)
$\mu_{addr}(P) := a$
Fall 2b - Falls $\#(\mu_{addr}(con_{M_8}(P))) > 1$
$\mu_{addr}(P) := min\{head(Q) \mid Q \in con_{M_8}(P)\}$
$\forall Q \in con_{M_8}(P)\ \mu_{addr}(Q) := min\{head(Q) \mid Q \in con_{M_8}(P)\}$

1. Schnelle Pixel: Fall 1, Fall 2a
2. Langsame Pixel: Fall 2b
3. Takte(Schnelles-Pixel) := $\mathbf{clocks_{fast1}}$
4. Takte(Langsames-Pixel) := $\mathbf{clocks_{slow}} + clocks_{merge}(con_{M_8}(P))$

wobei:

- $clocks\_merge(con_{M_8}(P)) := \sum_{Q \in con_{M_8}(P)} find_{head}(P) + relabel(con_{M_8}(P))$
- $find_{head}(P) := depth(P) * \mathbf{clocks_{find\_head\_per\_depth}}$
- $depth(P)$: Tiefe der Kette in mit Anfangspunkt P in μ_{addr}
- $relabel(con_{M_8}(P)) := (\#con_{M_8}(P) - 1) * \mathbf{clocks_{relabel}}$
- Parameter: $\mathbf{clocks_{fast1}}, \mathbf{clocks_{slow}}, \mathbf{clocks_{find\_head\_per\_depth}}, \mathbf{clocks_{relabel}}$ entsprechen denen des 4-Zusammenhangs (siehe Abbildung 5.29)

Bemerkung: Der 2-Lauf ist bezüglich 4- und 8-Zusammenhang identisch.
Das Laufzeitmodell des 2-Laufs ist in Abschnitt 5.6.2 zu finden.

Abbildung 7.6: Berechnung der Anzahl der Takte, welche für die Berechnung eines Bildpunktes benötigt werden, 8-Zusammenhang

Testbilder	Weiß	Dreieck	Diagonal 1	Diagonal 2	Labyrinth 1	Labyrinth 2
Erster Lauf						
Anteil schneller Pixel (in %)	100,00	99,95	100,00	50,34	91,71	87,84
Suchtiefe der $find_{head}(P)$ Operation	Tiefenverteilung in Prozent					
1	-	100,00	-	100,00	81,32	79,19
2	-	-	-	-	17,50	55,53
3	-	-	-	-	1,17	3,26
Zweiter Lauf						
Anteile schnelle Pixel (in %)	99,71	99,66	50,05	50,05	59,03	59,03

Abbildung 7.7: Analyse der synthetischen Testbilder, Bildgröße 1024*1024, 8-Zusammenhang

Testbilder	Durchschnittliche Takte pro Pixel			
	Model 1	Model 2	Model 3	Model 4
Weiß	7,00	4,00	2,00	4,00
Dreieck	7,01	4,00	2,00	4,00
Diagonal 1	7,00	4,00	2,00	4,00
Diagonal 2	12,96	7,48	5,48	5,99
Labyrinth 1	8,08	4,65	2,65	4,35
Labyrinth 2	8,29	4,78	2,78	4,38

Abbildung 7.8: Modelllaufzeiten synthetischer Testbilder, 8-Zusammenhang

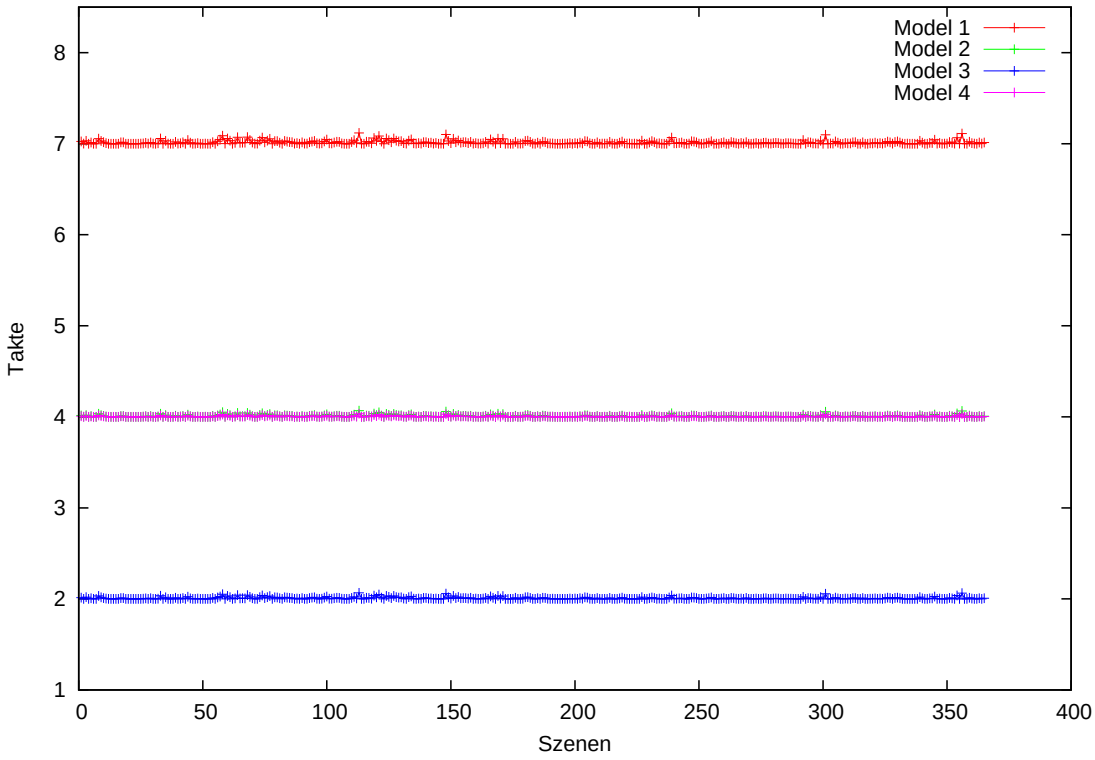


Abbildung 7.9: Modelllaufzeiten realer Testbilder, 8-Zusammenhang

Abkürzungsverzeichnis

ASIC	Application Specific Integrated Circuit
CCL	Connected Component Labeling
DLR	Deutsche Zentrum für Luft- und Raumfahrt e. V.
EDIF	Electronic Design Interchange Format
ETM	Enhanced Thematic Mapper (Landsat 7)
FF	Flipflop
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
LUT	Lookup-Tabelle
RAM	Random Access Memory
ROM	Read Only Memory
SRAM	Static Random Access Memory
TM	Thematic Mapper (Landsat 4, Landsat 5)
TPD	Thermal Design Power
TPSS	Two Pass Single Storage

Abbildungsverzeichnis

Literatur

[1] G. Zhou u. a. „Concept design of future intelligent Earth observing satellites“. In: *International Journal of Remote Sensing* 25.14 (2004), S. 2667–2685.

[2] Guoxia Yu, Tanya Vladimirova und Martin N. Sweeting. „Image compression systems on board satellites“. In: *Acta Astronautica* 64.9–10 (2009), S. 988–1005. ISSN: 0094-5765.

[3] S. Yuhaniz, T. Vladimirova und S. Gleason. „An intelligent decision-making system for flood monitoring from space“. In: *Bio-inspired, Learning, and Intelligent Systems for Security, 2007. BLISS 2007. ECSIS Symposium on*. IEEE. 2007, S. 65–71.

[4] T Bretschneider u. a. „Low-cost space-borne processing on a reconfigurable parallel architecture“. In: *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms*. Citeseer. 2004, S. 93–99.

[5] T. Stuffler u. a. „The EnMAP hyperspectral imager—An advanced optical payload for future applications in Earth observation programmes“. In: *Acta Astronautica* 61.1–6 (2007). <ce:title>Bringing Space Closer to People, Selected Proceedings of the 57th {IAF} Congress, Valencia, Spain, 2-6 October, 2006</ce:title>, S. 115–120. ISSN: 0094-5765.

[6] G. Lautenschläger u. a. *Sentinel-2: next generation satellites for optical land observation from space*. 2013.

[7] W. Halle u. a. „Autonomous onboard classification experiment for the satellite BIRD“. In: *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES* 34.1 (2002), S. 63–68.

[8] Felipe Ip u. a. „Flood detection and monitoring with the Autonomous Sciencecraft Experiment onboard EO-1“. In: *Remote Sensing of Environment* 101.4 (2006), S. 463–481. ISSN: 0034-4257.

[9] Steve Chien u. a. „Using autonomy flight software to improve science return on Earth Observing One“. In: *Journal of Aerospace Computing, Information, and Communication* 2.4 (2005), S. 196–216.

[10] T. Doggett u. a. „Autonomous detection of cryospheric change with hyperion onboard Earth Observing-1“. In: *Remote Sensing of Environment* 101.4 (2006), S. 447–462. ISSN: 0034-4257.

[11] A.G. Davies u. a. „Monitoring active volcanism with the Autonomous Sciencecraft Experiment on EO-1“. In: *Remote Sensing of Environment* 101.4 (2006), S. 427–446. ISSN: 0034-4257.

[12] CT Johnston, KT Gribbon und DG Bailey. „Implementing image processing algorithms on FPGAs“. In: *Proceedings of the Eleventh Electronics New Zealand Conference, ENZCon'04*. Citeseer. 2004, S. 118–123.

[13] R.M. Haralick und L.G. Shapiro. *Computer and robot visión*. Computer and Robot Vision Bd. 1. Addison-Wesley Pub. Co., 1992. ISBN: 9780201108774.

[14] Kevin McGuinness und Noel E. O'Connor. „A comparative evaluation of interactive segmentation algorithms“. In: *Pattern Recognition* 43.2 (2010). <ce:title>Interactive Imaging and Vision</ce:title>, S. 434–444. ISSN: 0031-3203.

[15] H. Zhang, J.E. Fritts und S.A. Goldman. „Image segmentation evaluation: A survey of unsupervised methods“. In: *Computer Vision and Image Understanding* 110.2 (2008), S. 260–280.

[16] R.M. Haralick und L.G. Shapiro. „Image segmentation techniques“. In: *Computer vision, graphics, and image processing* 29.1 (1985), S. 100–132.

[17] Nobuyuki Otsu. „A Threshold Selection Method from Gray-Level Histograms“. In: *Systems, Man and Cybernetics, IEEE Transactions on* 9.1 (1979), S. 62–66. ISSN: 0018-9472.

[18] J. Canny. „A computational approach to edge detection“. In: *Readings in computer vision: issues, problems, principles, and paradigms* 184.87-116 (1987), S. 86.

[19] Serge Beucher u. a. „The watershed transformation applied to image segmentation“. In: *SCANNING MICROSCOPY-SUPPLEMENT-* (1992), S. 299–299.

[20] D. Comaniciu und P. Meer. „Mean shift: A robust approach toward feature space analysis“. In: *IEEE Transactions on pattern analysis and machine intelligence* (2002), S. 603–619.

[21] T.F. Chan und L.A. Vese. „Active contours without edges“. In: *Image Processing, IEEE Transactions on* 10.2 (Feb. 2001), S. 266–277. ISSN: 1057-7149.

[22] LuminitaA. Vese und TonyF. Chan. „A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model“. English. In: *International Journal of Computer Vision* 50 (3 2002), S. 271–293. ISSN: 0920-5691.

[23] David Mumford und Jayant Shah. „Optimal approximations by piecewise smooth functions and associated variational problems“. In: *Communications on Pure and Applied Mathematics* 42.5 (1989), S. 577–685. ISSN: 1097-0312.

[24] Leonid I. Rudin, Stanley Osher und Emad Fatemi. „Nonlinear total variation based noise removal algorithms“. In: *Physica D: Nonlinear Phenomena* 60.1–4 (1992), S. 259–268. ISSN: 0167-2789.

[25] M. Neuenhahn, H. Blume und TG Noll. „Pareto optimal design of an FPGA-based real-time watershed image segmentation“. In: *Proceedings of the Conference on Program for Research on Integrated Systems and Circuits (ProRISC'04)*. 2004.

[26] R. Bannister u. a. „FPGA implementation of an image segmentation algorithm using logarithmic arithmetic“. In: *Circuits and Systems, 2005. 48th Midwest Symposium on*. IEEE. 2005, S. 810–813.

[27] H.O. Georgii. *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik*. de Gruyter, 2009.

[28] Hart Duda, Peter Hart u. a. *Stork, Pattern Classification*. 2001.

[29] K. Yamaoka u. a. „Image segmentation and pattern matching based FPGA/ASIC implementation architecture of real-time object tracking“. In: *Design Automation, 2006. Asia and South Pacific Conference on*. 2006, pages.

[30] T Morimoto u. a. „Pixel-parallel digital CMOS implementation of image segmentation by region growing“. In: *IEE Proceedings-Circuits, Devices and Systems* 152.6 (2005), S. 579–589.

[31] T. Koide u. a. „Architecture and FPGA-Implementation of Scalable Picture Segmentation by 2D Scanning with Flexible Pixel-Block Size“. In: *Networking and Computing (ICNC), 2010 First International Conference on*. 2010, S. 128–132.

[32] Dang Ba Khac Trieu und T. Maruyama. „An Implementation of the Mean Shift Filter on FPGA“. In: *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. 2011, S. 219–224.

[33] R.L. Rosas, A. De Luca und F.B. Santillan. „SIMD architecture for image segmentation using Sobel operators implemented in FPGA technology“. In: *Electrical and Electronics Engineering, 2005 2nd International Conference on*. Sep. 2005, S. 77–80.

[34] Wenhao He und Kui Yuan. „An improved Canny edge detector and its realization on FPGA“. In: *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*. 2008, S. 6561–6564.

[35] Hong Shan Neoh und Asher Hazanchuk. „Adaptive edge detection for real-time video processing using FPGAs“. In: *Global Signal Processing* (2004).

[36] H. Jahn. „Segmentation of Remote Sensing Images with a Layered Graph Network“. In: *International Archives of Photogrammetry and Remote Sensing* 31 (1996), S. 360–364.

[37] Herbert Jahn. „Graph structure for image segmentation“. In: *Proc. SPIE* 3026 (1997), S. 198–208.

[38] W. Halle. *Ausgewählte Algorithmen der Segmentierung und Klassifikation zur thematischen Bildverarbeitung in der Fernerkundung*. Techn. Ber. ISSN 1434-8454. Deutsches Zentrum für Luft- und Raumfahrt e. V. Institut für Weltraumsensorik und Planetenerkundung, 1999.

[39] Frank Kesel und Ruben Bartholomä. *Entwurf von digitalen Schaltungen und Systemen mit HDLs und FPGAs*. Oldenbourg Verlag, 2006.

[40] Xilinx Corporation. *Spartan-3E FPGA family: complete data sheet*. 2013.

[41] Xilinx Corporation. *Spartan-6 Family Overview*. 2013.

[42] I. Alston und B. Madahar. „From C to netlists: hardware engineering for software engineers?“ English. In: *Electronics & Communication Engineering Journal* 14 (4 Aug. 2002), 165–173(8). ISSN: 0954-0695.

[43] D.G. Bailey und C.T. Johnston. „Algorithm Transformation for FPGA Implementation“. In: *Electronic Design, Test and Application, 2010. DELTA '10. Fifth IEEE International Symposium on*. 2010, S. 77–81.

[44] A. Downton und D. Crookes. „Parallel architectures for image processing“. In: *Electronics Communication Engineering Journal* 10.3 (1998), S. 139–151. ISSN: 0954-0695.

[45] M Neubert, H Herold und G Meinel. „Evaluation of remote sensing image segmentation quality–further results and concepts“. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36.4/C42 (2006).

[46] Jörgen Kosche H. Horn. „Segmentierung von Bilddaten“. Juni 2004.

[47] Mark Everingham, Henk Muller und Barry Thomas. „Evaluating Image Segmentation Algorithms Using the Pareto Front“. English. In: *Computer Vision — ECCV 2002*. Hrsg. von Anders Heyden u. a. Bd. 2353. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, S. 34–48. ISBN: 978-3-540-43748-2.

[48] C. Tomasi und R. Manduchi. „Bilateral filtering for gray and color images“. In: *Computer Vision, 1998. Sixth International Conference on*. IEEE. 1998, S. 839–846.

[49] Gary Bradski und Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly, 2008.

[50] G. Chander, B.L. Markham und D.L. Helder. „Summary of current radiometric calibration coefficients for Landsat MSS, TM, ETM+, and EO-1 ALI sensors“. In: *Remote Sensing of Environment* 113.5 (2009), S. 893–903.

[51] J.M. Park, C.G. Looney und H.C. Chen. „Fast connected component labeling algorithm using a divide and conquer technique“. In: *Conference on Computers and Their Applications*. 2000, S. 373–376.

[52] K. Wu, E. Otoo und K. Suzuki. „Two strategies to speed up connected component labeling algorithms“. In: (2008).

[53] Lionel Lacassagne und Bertrand Zavidovique. „Light speed labeling: efficient connected component labeling on RISC architectures“. In: *Journal of Real-Time Image Processing* 6 (2 2011). 10.1007/s11554-009-0134-0, S. 117–135. ISSN: 1861-8200.

[54] M.B. Dillencourt, H. Samet und M. Tamminen. „A general approach to connected-component labeling for arbitrary image representations“. In: *Journal of the ACM (JACM)* 39.2 (1992), S. 253–280.

[55] M.J. Klaiber u. a. „A high-throughput FPGA architecture for parallel connected components analysis based on label reuse“. In: *Field-Programmable Technology (FPT), 2013 International Conference on*. Dez. 2013, S. 302–305.

[56] A. Rosenfeld und J.L. Pfaltz. „Sequential operations in digital picture processing“. In: *Journal of the ACM (JACM)* 13.4 (1966), S. 471–494.

[57] Y. Ito und K. Nakano. „Component labeling for k-concave binary images using an FPGA“. In: *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. Apr. 2008, S. 1–8.

[58] A<ol Rosenfeld. „Connectivity in digital pictures“. In: *Journal of the ACM (JACM)* 17.1 (1970), S. 146–160.

[59] R.M. Haralick. „Some Neighborhood Operators“. English. In: *Real-Time Parallel Computing*. Hrsg. von Morio Onoe, Jr. Preston Kendall und Azriel Rosenfeld. Springer US, 1981, S. 11–35. ISBN: 978-1-4684-3895-6.

[60] D. Crookes und K. Benkrid. „An FPGA implementation of image component labelling“. In: *Reconfigurable Technology: FPGAs for Computing and Applications* (1999), S. 17–23.

[61] K. Suzuki, I. Horiba und N. Sugie. „Linear-time connected-component labeling based on sequential local operations“. In: *Computer Vision and Image Understanding* 89.1 (2003), S. 1–23.

[62] R. Rachakonda, P. Athanas und A. Abbott. „High-speed region detection and labeling using an FPGA-based custom computing platform“. In: *Field-Programmable Logic and Applications*. Springer. 1995, S. 86–93.

[63] Robert Endre Tarjan. „A class of algorithms which require nonlinear time to maintain disjoint sets“. In: *Journal of Computer and System Sciences* 18.2 (1979), S. 110–127. ISSN: 0022-0000.

[64] H. Samet und M. Tamminen. „An improved approach to connected component labeling of images“. In: *International Conference on Computer Vision And Pattern Recognition*. 1986, S. 312–318.

[65] C. Schmidt und A. Koch. „Fast region labeling on the reconfigurable platform acev“. In: *Field Programmable Logic and Application* (2003), S. 1083–1086.

[66] K. Appiah u. a. „A run-length based connected component algorithm for FPGA implementation“. In: *ICECE Technology, 2008. FPT 2008. International Conference on*. Dez. 2008, S. 177–184.

[67] DG Bailey und CT Johnston. „Single pass connected components analysis“. In: *Image and Vision Computing New Zealand*. Citeseer. 2008, S. 282–287.

[68] C.T. Johnston und D.G. Bailey. „FPGA implementation of a single pass connected components algorithm“. In: *4th IEEE International Symposium on Electronic Design, Test & Applications*. IEEE. 2008, S. 228–231.

[69] V.S. Kumar u. a. „A Scalable Bandwidth-Aware Architecture for Connected Component Labeling“. In: *VLSI 2010 Annual Symposium*. Springer. 2011, S. 133–149.

[70] H. Hedberg, F. Kristensen und V. Owall. „Implementation of a labeling algorithm based on contour tracing with feature extraction“. In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE. 2007, S. 1101–1104.

[71] M Manohar und H.K Ramapriyan. „Connected component labeling of binary images on a mesh connected massively parallel processor“. In: *Computer Vision, Graphics, and Image Processing* 45.2 (1989), S. 133–149. ISSN: 0734-189X.

[72] P. Bhattacharya. „Connected component labeling for binary images on a reconfigurable mesh architecture“. In: *Journal of systems architecture* 42.4 (1996), S. 309–313.

[73] A. Rasquinha und N. Ranganathan. „C3L: a chip for connected component labeling“. In: *VLSI Design, 1997. Proceedings., Tenth International Conference on*. 1997, S. 446–450.

[74] J. Soman, K. Kishore und P. J. Narayanan. „A fast GPU algorithm for graph connectivity“. In: *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. 2010, S. 1–8.

[75] Oleksandr Kalentev u. a. „Connected component labeling on a 2D grid using {CUDA}“. In: *Journal of Parallel and Distributed Computing* 71.4 (2011), S. 615–620. ISSN: 0743-7315.

[76] Michael Backer, Jan Tünnermann und Bärbel Mertsching. „Parallel k-Means Image Segmentation Using Sort, Scan and Connected Components on a GPU“. In: *Facing the Multicore-Challenge III*. Hrsg. von Rainer Keller, David Kramer und Jan-Philipp Weiss. Bd. 7686. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, S. 108–120. ISBN: 978-3-642-35892-0.

[77] L. Di Stefano und A. Bulgarelli. „A simple and efficient connected components labeling algorithm“. In: *Image Analysis and Processing, 1999. Proceedings. International Conference on*. IEEE. 1999, S. 322–327.

[78] M. Jablonski und M. Gorgon. „Handel-C implementation of classical component labelling algorithm“. In: *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*. IEEE. 2004, S. 387–393.

[79] Kirk Saban. „Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency“. In: *Xilinx White paper: Vertex-7 FPGAs* (2011).

[80] Eben Upton und Gareth Halfacree. *Raspberry Pi User Guide*. John Wiley & Sons, 2013.

[81] Xilinx Zynq. „Zynq-7000 allprogrammable soc overview“. In: *Zynq-7000 all programmable soc overview, advance product specification-ds190(v1. 2) available on: http://www. xilinx. com/support/documentation-/data sheets/-ds190-Zynq-7000-Overview. pdf," August* (2012).

[82] Fernando Martinez Vallina, Christian Kohn und Pallav Joshi. „Zynq all programmable SoC Sobel filter implementation using the Vivado HLS tool“. In: *vol. XAPP890* (2012), S. 1–16.

[83] D. Lüdtke u. a. „OBC-NG: Towards a reconfigurable on-board computing architecture for spacecraft“. In: *Aerospace Conference, 2014 IEEE*. März 2014, S. 1–13.

[84] FPGA Digilent's ZedBoard Zynq. *Dev. board documentation.*