Lehrstuhl für Messsystem- und Sensortechnik

Echtzeitvermessung dreidimensionaler Objekte mittels Speckle-Interferometrie

Dipl.-Ing. Markus Riemenschneider

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. habil. Alexander W. Koch
	2.	Hon.-Prof. Dr. rer. nat. habil. Bernd Schürmann, Ph.D. (Univ. of Cape Town, Südafrika) Johann-Wolfgang-Goethe-Universität Frankfurt am Main

Die Dissertation wurde am 04.05.2004 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 11.08.2004 angenommen.

Zusammenfassung

Der industrielle Einsatz der Speckleinterferometrie zur dreidimensionalen Formvermessung wird bisher durch lange Mess- und Rechenzeiten erschwert. Um eine Echtzeitauswertung zu erreichen, werden zunächst die Phasenbildberechnung und deren anschließende Entfaltung untersucht. Dazu werden Algorithmen entwickelt und in VHDL auf FPGAs realisiert. Danach erfolgt die Modifizierung eines Speckle-Interferometers derart, dass es mit geschalteten Laserdioden ohne bewegte Verschlusseinrichtungen arbeitet. Die eingesetzten Laserdioden werden dazu direkt im System auf Stabilität und Wellenlänge vermessen, so dass diese Parameter stets bekannt sind. Im Ergebnis dient das Zwei-Wellenlängen-Speckleinterferometer zur Vermessung von dreidimensionalen Objekten mit einer Auflösung von 1280 x 1024 Pixeln und mittleren Verarbeitungsrate von 23 Bildern pro Sekunde.

Abstract

The operation of speckle interferometers in the case of three dimensional object measurement is a difficult task because of the long measurement and analysis duration. To realize a realtime measurement the phase calculation and unwrapping procedure are investigated. Therefore algorithms are developed and realized in VHDL on FPGAs. In the next step the mechanical shutters are replaced by switched laser diodes. To enumerate the stability and wavelength of the these laser diodes they where observed by an in system measurement. Finally the two wavelength speckle interferometer achieves a resolution of 1280 x 1024 pixel^2 and the operating frequency is 23 images per second.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Einsatz optischer Messverfahren bietet in der industriellen Messtechnik viele Vorteile [1]. Der wichtigste besteht dabei in der Möglichkeit, Objekte zu vermessen, zu denen lediglich ein optischer Zugang besteht. Diese lassen sich damit berührungslos auch unter schwierigen Bedingungen, wie beispielsweise unter großer Hitze oder in stark verschmutzter Umgebung vermessen. Solche Messsysteme arbeiten, indem sie das Messobjekt mit Licht beleuchten und es gleichzeitig mit einem optischen Aufnehmer beobachten. Die Rückwirkung auf das Messobjekt kann im Allgemeinen vernachlässigt werden, da die Lichtleistungen lediglich im mW-Bereich liegen. Die erreichbare lokale Auflösung ist dabei sehr hoch, da bereits optische Standardkomponenten eine störungsfreie Bündelung, Fokussierung und Ausrichtung optischer Strahlung erlauben. Insbesondere gerichtete Laserstrahlen eröffnen die Möglichkeit große Messabstände zu realisieren.
Die Speckle-Interferometrie ist ein etabliertes Verfahren der optischen Messtechnik. Bei diesem Messverfahren wird eine raue Oberfläche mit einem Laserstrahl beleuchtet und das reflektierte Licht wird beobachtet. Infolge der Rauheit der Oberfläche bildet sich ein Fleckchenmuster durch lokale Interferenzen aus. Diese Fleckchen werden auch als Speckle bezeichnet und geben dieser Messtechnik ihren Namen. Die Speckle-Muster enthalten eine Reihe von Informationen. Hierzu zählen vor allem die Form, die Deformation und die Ebenheit von Objektoberflächen. Bis vor einiger Zeit existierten nur sehr störanfällige und umständliche Laboraufbauten auf schwingungsgedämpften Tischen [1]. Seit kurzem ist ein kompakter Messkopf bekannt, der es ermöglicht, unter industriellen Umgebungsbedingungen mit interferometrischer Genauigkeit zu messen [2]. Das Gerät arbeitet ohne Schwingungsdämpfung und lässt sich leicht an verschiedene Messobjekte, Messabstände und Messempfindlichkeiten adaptieren. Die Entwicklung des Geräts basiert auf wesentlichen Fortschritten im Bereich der Laser- und der Kameratechnik. Hierzu zählen insbesondere hochauflösende Kameras und kompakte, leistungsstarke, monomodige Laserdioden.

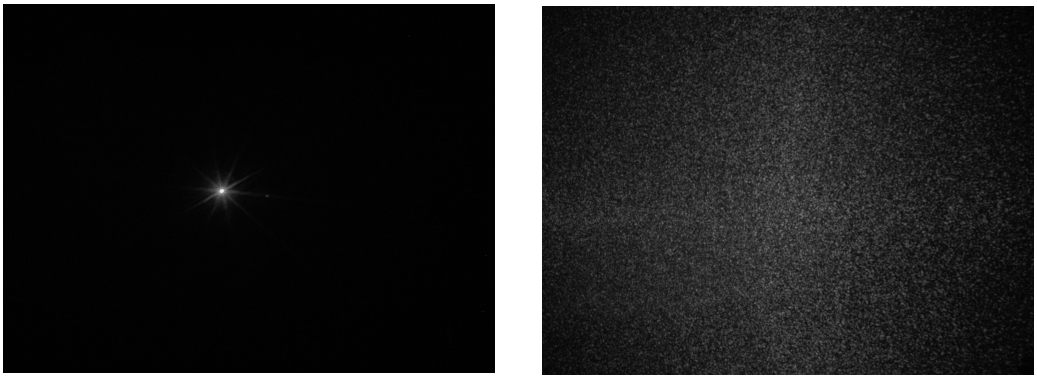
Im industriellen Umfeld finden sich eine Vielzahl von möglichen Anwendungen, vor allem im Bereich der online-Diagnose in der Fertigung. Bei immer weiter wachsenden Anforderungen an die Produktqualität reichen häufig Stichprobenkontrollen nicht mehr aus [4]. In der Fertigung von Flugzeugtriebwerken werden Schweißnähte visuell geprüft und auf ihre Qualität hin beurteilt [5]. Dies zeigt die Notwendigkeit eines universellen, robusten und schnellen Messgerätes zur online-Diagnose von Oberflächenformen und -ebenheiten. Dabei soll das Gerät in Echtzeit, d.h. im Kameratakt Messungen durchführen und auch auswerten können und im industriellen Einsatz möglichst ohne Wartung auskommen.
Zur Auswertung der verschiedenen Messgrößen existieren bereits eine Vielzahl von Algorithmen, die jedoch aufgrund der fehlenden Notwendigkeit bisher nicht auf Geschwindigkeiten im Millisekundenbereich hin optimiert wurden [21]. Auch mit immer stärker wachsenden Taktraten bei PCs und DSPs dauert die Auswertung einer vollständigen Messung noch mehrere Sekunden. Häufig übertrifft hierbei der Rechenaufwand durch die wachsende Auflösung der Kameras den Geschwindigkeitsgewinn durch schnellere Rechner bei weitem. Um dennoch in den Bereich der Echtzeitauswertung zu gelangen, bietet sich eine Auswertung der Daten in einer spezialisierten Hardware an. Hierzu existieren inzwischen auch hervorragende Entwicklungssysteme, die es ermöglichen, ohne eine aufwändige Hardwareentwicklung Algorithmen in ihrer physikalischen Umgebung zu emulieren [3]. Dazu müssen die vorhandenen Algorithmen jedoch zunächst selektiert und auf einen möglichen Hardware-Einsatz hin optimiert werden. Es werden neue Algorithmen und deren Implementierung in Hardware vorgestellt, mit welchen es möglich ist, Speckle-Interferogramme zur Oberflächenform, -deformation und -ebenheitsmessung im Kameratakt durchzuführen.
Ein weiteres wesentliches Problem betrifft den Einsatz der Laserdioden. Es stehen zwar inzwischen monomodige Laserdiodensysteme inklusive der Regler zur Verfügung. In der Herstellung der Laserdioden ist es jedoch nur schwer möglich, von einer Charge zur nächsten die Kennwerte exakt zu reproduzieren. Außerdem ändern sich diese Parameter mit der Alterung der Diode [7]. Ferner garantiert auch die Regelung der Temperatur und des Stroms der Laserdiode noch keinen dauerhaften monomodigen Betrieb. Deshalb kommt es häufig zu Fehlmessungen und einem damit verbunden Geräteausfall, was eine aufwändige Nachkalibrierung des Messgerätes nach sich zieht. Diese Nachkalibrierung ist auch bei Änderungen der Messempfindlichkeit erforderlich, da diese von den eingesetzten Wellenlängen des Lichts abhängt. Das Schalten der Laserdioden erfolgt bisher durch den Einsatz mechanischer Shutter [2], die ihrerseits die Messgeschwindigkeit erheblich einschränken, da sie Vibrationen in das Messsystem einkoppeln. Aus diesem Grund ist es erforderlich, die Laserdioden im Interferometer ständig zu überwachen und automatisch zu kalibrieren. Dazu dient ein automatisches Messsystem, das es ermöglicht, die Laserdioden in ihrer Stabilität abhängig von dem Temperatur- und

Stromfluss zu charakterisieren. Diese Charakterisierung, auch Stabilitätskarte [8] genannt, dient als Grundlage zur Umschaltung der Messempfindlichkeiten, das heißt, die stabilen Wellenlängen sind bekannt und können definiert eingestellt werden. Ein weiterer Vorteil des Systems besteht in der direkten Schaltung der Laserdioden, so dass das Messgerät vollständig ohne bewegte Teile auskommt.

Kapitel 2

Grundlagen der Speckle-Interferometrie

2.1 Speckle-Effekt



a) b)

Abbildung 2.1: a) Laserpunkt auf einem Schirm, b) objektive Speckles auf einem Schirm

Der Speckle-Effekt beruht auf der diffusen Streuung monochromatischen Lichts an rauen Oberflächen. Im Gegensatz zur Reflexion an einem Spiegel entsteht dabei kein homogener heller Fleck, es sind vielmehr charakteristische Fleckenmuster zu erkennen. Diese enstehen durch die Überlagerung vieler Teilstrahlen mit zufälligen Phasen. Bei der direkten Beobachtung der Strahlen spricht man von objektiven Speckles, bei der Abbildung der Strahlen mittels einer Linse handelt es sich um subjektive Speckles. Bild 1.1 zeigt die Photographie eines mit einem

Laser beleuchteten Spiegels, sowie einer rauen Aluminium-Platte.
Bei der Überlagerung des optischen Feldes mit einer Referenzwellenfront der gleichen Lichtquelle entsteht ein Speckle-Interferogramm. In diesem Speckle-Interferogramm sind Informationen über die Geometrie der Fläche, wie beispielsweise die Form oder die Deformation bezüglich der Referenzwellenfront enthalten.
Damit voll ausgebildete Speckles entstehen, muss die Rauheit der Oberfläche einen Rauheitswert von mindestens $R_q \geq \lambda/4$ annehmen. Der Wert R_q bezeichnet hierbei den quadratischen Mittenrauwert der Fläche [2]. Daraus ergibt sich dann, dass die Phasendifferenz zwischen den Teilstrahlen den Wert π überschreiten und es damit zur totalen Auslöschung der Strahlen kommen kann. Falls diese Bedingung nicht erfüllt ist, liegen nur teilweise ausgebildete Speckles vor [3].

2.2 Speckle-Interferometrie

Grundsätzlich lässt sich Speckle-Messtechnik in interferometrische und nicht-interferometrische Verfahren unterteilen. Die nicht-interferometrischen Verfahren beschäftigen sich im Wesentlichen mit der Rauheitsmessung [1, 10]. Die wichtigsten Einsatzgebiete der interferometrischen Verfahren sind [2]:

- Oberflächenformvermessung
- Oberflächendeformationsmessung
- Oberflächenschwingungsmessung
- Oberflächenerosionsmessung

Verfahren für die Schwingungs- und Deformationsmessung sind hinreichend bekannt und es existieren bereits eine Reihe von kommerziell erhältlichen Geräten. Die Erosionsmessung ist ein Spezialgebiet der Speckle-Interferometrie, die im Wesentlichen bei der Erforschung der kontrollierten Kernfusion zum Einsatz kommt [3, 9].
Das folgende Kapitel betrachtet ausschließlich Messsysteme zur Vermessung der Form und der Ebenheit von Oberflächen.
Prinzipiell lassen sich die interferometrischen Anordnungen in drei klassische Gruppen einteilen [2, 11, 12]:

- out-of-plane empfindliche Interferometer
- in-plane empfindliche Interferometer
- shearing Interferometer

Die folgenden Abschnitte beschäftigen sich ausschließlich mit out-of-plane empfindlichen Interferometern.
Zur Beschreibung der verschiedenen Interferometer und deren Empfindlichkeiten

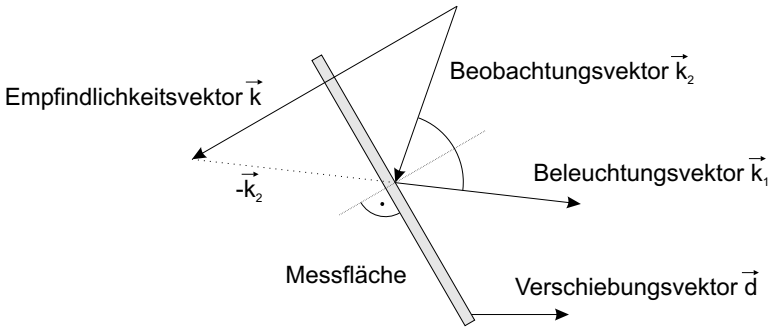


Abbildung 2.2: Empfindlichkeitsvektor

dient der Empfindlichkeitsvektor $\boldsymbol{k}$, der definiert ist als die Differenz zwischen dem Beleuchtungsvektor $\boldsymbol{k}_1$ und dem Beobachtungsvektor $\boldsymbol{k}_2$ [12]:

$$\boldsymbol{k} = \boldsymbol{k}_1 - \boldsymbol{k}_2 \tag{2.1}$$

Wie Abbildung 2.2 zeigt, gibt der Beleuchtungsvektor die Richtung der Beleuchtung, beispielsweise die eines Laserstrahls an. Der Beobachtungsvektor dagegen zeigt in Richtung des Beobachters. Die Winkel beider Vektoren werden gegen die Flächennormale der Beobachtungsfläche gemessen und ihre Länge beträgt $2\pi/\lambda$. Damit ergibt sich die Phasenänderung $\Delta\varphi$ als Anteil des Verschiebungsvektors $\boldsymbol{d}$ in Richtung des Empfindlichkeitsvektors $\boldsymbol{k}$:

$$\Delta\varphi = \boldsymbol{k} \cdot \boldsymbol{d} \tag{2.2}$$

2.2.1 Out-of-Plane Interferometer

Bei einer ausschließlich out-of-plane empfindlichen Anordnung steht der Empfindlichkeitsvektor senkrecht auf der Beobachtungsebene. Diese Bedingung ist genau dann erfüllt, wenn die Winkel α und β nach Abbildung 2.2 gleich groß sind. Für $\alpha = \beta$ folgt somit für den Betrag von $\boldsymbol{k}$:

$$|\boldsymbol{k}| = \frac{4\pi}{\lambda}\cos(\alpha) \tag{2.3}$$

Abbildung 2.3 zeigt die beiden bekanntesten Vertreter dieser Interferometeranordungen [12]. Es stehen dabei sowohl die Beleuchtungs- als auch die Beobachtungsrichtung senkrecht zur Messfläche. Daraus ergibt sich, dass auch der Emp-

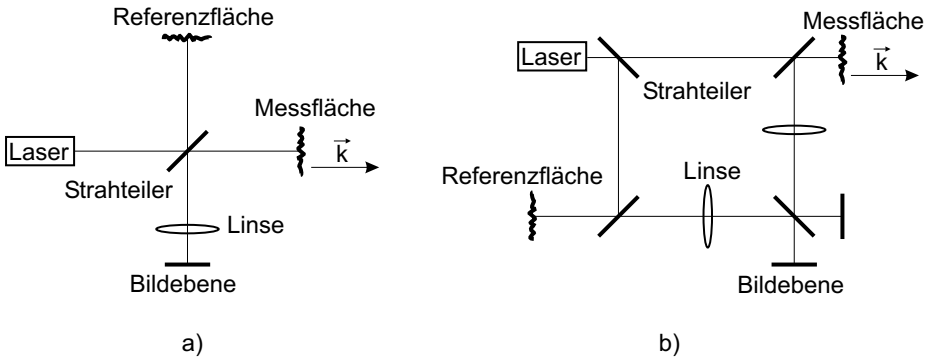


Abbildung 2.3: a) Michelson-Interferometer mit rauer Mess- und Referenzfläche, b) Mach-Zehnder-Interferometer mit rauer Mess- und Referenzfläche

findlichkeitsvektor senkrecht dazu steht und einen Betrag von $2\pi/\lambda$ besitzt. Für die resultierende Intensität I kann somit angegeben werden [2]:

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}\cos(\alpha_p)\cos\left(\Phi_0 + \frac{4\pi}{\lambda}d\right) \tag{2.4}$$

Hierbei bezeichnet I_1 die von der Referenzfläche und I_2 die von der Messfläche kommende Intensität, α_p den Winkel zwischen den interferierenden elektrischen Feldvektoren, Φ_0 die Phasendifferenz zwischen den interferierenden Feldern vor der Verschiebung, λ die Beleuchtungswellenlänge und d der Betrag der Verschiebung in out-of-plane Richtung.
In den ursprünglichen Spiegelinterferometern bestanden sowohl die Referenz- als auch die Messfläche aus Spiegeln. Bei der Speckle-Interferometrie wird mindestens einer der beiden Spiegel durch eine optisch raue Fläche ersetzt. Die Funktion und insbesondere die Empfindlichkeit der Interferometer bleiben dadurch jedoch unberührt. Um bei einem Speckle-Interferometer eine ortsaufgelöste Messung zu erhalten muss die Beobachtungsebene jedoch abgebildet werden [2].

2.3 Formvermessung

Prinzipiell lassen sich zwei unterschiedliche Verfahren zur Formvermessung mittels Speckle-Interferometrie unterscheiden [3, 12, 9]:

- Zwei-Wellenlängen-Technik
- Zwei-Winkel-Technik

Da sich die eingesetzten Algorithmen ausschließlich mit der Zwei-Wellenlängen-Technik beschäftigen, wird im Folgenden auf die Zwei-Winkel-Technik nicht weiter eingegangen.
Wie der Name bereits andeutet, arbeitet die Zwei-Wellenlängen-Technik mit zwei Speckle-Interferogrammen, die nacheinander mit zwei unterschiedlichen Wellenlängen λ_1 und λ_2 entstehen. Physikalisch liegt der Messung einer der beiden optischen Aufbauten nach Abbildung 2.3 zugrunde. Die beiden Intensitäten $I_{\lambda_{1,xy}}$ und $I_{\lambda_{2,xy}}$ auf der Bildebene ergeben sich zu:

$$\begin{aligned} I_{\lambda_1,xy} &= I_{1,xy} + I_{2,xy} + 2\sqrt{I_{1,xy}I_{2,xy}}\cos\left(\Phi_1 + \frac{4\pi}{\lambda_1}\cos(\alpha)\cdot d_{xy}\right) \\ I_{\lambda_2,xy} &= I_{1,xy} + I_{2,xy} + 2\sqrt{I_{1,xy}I_{2,xy}}\cos\left(\Phi_2 + \frac{4\pi}{\lambda_2}\cos(\alpha)\cdot d_{xy}\right) \end{aligned} \tag{2.5}$$

Als Voraussetzung gilt dabei, dass die Reflexionskoeffizienten von Mess- und Referenzfläche sowie die von diesen reflektierten Intensitäten $I_{1,xy}$ und $I_{2,xy}$ bei unterschiedlichen Beleuchtungswellenlängen λ_1 und λ_2 konstant bleiben [3, 9]. Weiterhin bezeichnen Φ_1 und Φ_2 die wellenlängenabhängigen Phasendifferenzen der interferierenden Felder bezogen auf eine feste Höhe der Mess- und Referenzfläche ohne deren Form zu berücksichtigen, α den Beleuchtungs- und Beobachtungswinkel und d_{xy} den formbedingten Höhenunterschied in Richtung des Empfindlichkeitsvektors zwischen Mess- und Referenzfläche, bezogen auf die gleiche feste Höhe. Die Indizes x und y beschreiben die örtliche Koordinate im Interferogramm.
Eine mögliche Auswertung dieser beiden Interferogramme besteht darin, die beiden Intensitäten $I_{\lambda_1,xy}$ und $I_{\lambda_2,xy}$ voneinander zu subtrahieren:

$$\begin{aligned} I_{Diff,xy} &= 4\sqrt{I_{1,xy}I_{2,xy}} \\ &\quad \cdot \sin\left(-\frac{\Phi_1+\Phi_2}{2} - 2\pi d_{xy}\cos(\alpha)\left(\frac{1}{\lambda_1}+\frac{1}{\lambda_2}\right)\right) \\ &\quad \cdot \sin\left(\frac{\Phi_1-\Phi_2}{2} - 2\pi d_{xy}\cos(\alpha)\left(\frac{1}{\lambda_1}-\frac{1}{\lambda_2}\right)\right) \end{aligned} \tag{2.6}$$

Gleichung 2.6 lässt sich erheblich vereinfachen, da gilt:

$$\frac{1}{\lambda_1}+\frac{1}{\lambda_2} \gg \frac{1}{\lambda_1}-\frac{1}{\lambda_2} \tag{2.7}$$

Damit ergibt nämlich nur der zweite Sinus-Term einen auswertbaren Zusammenhang zwischen Differenzintensität und Messflächenform [3, 9]. Mit der Einführung der synthetischen Wellenlänge Λ [3, 9, 12] ergibt sich aus Gleichung 2.6:

$$I^*_{Diff,xy} = 4\sqrt{I_{1,xy}I_{2,xy}}\ sin\left(\frac{\Phi_1-\Phi_2}{2} - \frac{\pi}{\Lambda}d_{xy}\right) \tag{2.8}$$

Wobei die synthetische Wellenlänge definiert ist als:

$$\Lambda = \frac{\lambda_1 \lambda_2}{2|\lambda_1 - \lambda_2| cos(\alpha)} \tag{2.9}$$

Im Differenzbild erscheinen dann schwarze Streifen, welche die relative Form bezüglich der Referenzfläche angeben. Der Übergang von einem Streifen zum nächsten entspricht dabei einer Höhenänderung von Λ. Die Höhenänderung lässt sich jedoch nur im Bereich der schwarzen Streifen berechnen, wobei eine Aussage über das Vorzeichen nicht möglich ist [2]. Da die Differenzintensitäten auch negative Werte annehmen können, müssen bei der Darstellung in Form eines Differenzbilds die Beträge verwendet werden. Erst die Anwendung von Phasenschiebeverfahren, die Abschnitt 2.4 beschreibt, ermöglicht eine Aussage über die gesamte Form der Messfläche inklusive des Vorzeichens.
Eine Variation der Beleuchtungswellenlängen λ_1 und λ_2 ermöglicht eine Variation der synthetische Wellenlänge Λ. Abschnitt 6.1 beschreibt ein Verfahren zur automatischen Einstellung des Messbereichs durch die Variation von Λ. Es ist zu beachten, dass die Größe d_{xy} neben der zu messenden makroskopischen Messflächenform auch einen Anteil aufgrund deren Rauheit enthält. Da es sich hierbei um eine statistische Größe handelt, macht sich dies durch mehr oder weniger starke Intensitätsschwankungen im Bereich der schwarzen Streifen bemerkbar. Um dennoch einen guten Streifenkontrast zu erhalten, müssen die synthetische Wellenlänge Λ und der quadratischen Mittenrauwert R_q folgende Relation einhalten [9]:

$$\Lambda > 8R_q \tag{2.10}$$

2.4 Phasenschieben

Zur Bestimmung der Phase existieren eine Reihe von Algorithmen, die hinsichtlich verschiedenster Eigenschaften optimiert sind. Die Beschreibung der einzelnen Algorithmen folgt in Kapitel 4.1. Dieser Abschnitt beschreibt das prinzipielle Problem und die physikalische Umsetzung.
Die gemessene Intensität I eines Interferogramms berechnet sich allgemein aus der Grundintensität I_0, der Modulation γ_0, dem bekannten Phasenschub α sowie der Phase ϕ:

$$I = I_0 \left[1 + \gamma_0 \cos(\phi + \alpha)\right] \tag{2.11}$$

In der Gleichung 2.11 treten die drei unbekannten Größen I_0, γ_0 und ϕ auf. Zur Bestimmung der gesuchten Phase ϕ sind demnach mindestens drei Messungen notwendig. Die verschiedenen Algorithmen arbeiten mit unterschiedlicher Zahl

an Intensitätsmessungen sowie mit unterschiedlichem Phasenschub α. Die Ergebnisse unterscheiden sich dann insbesondere in ihrer Empfindlichkeit gegenüber falschen Phasenschüben oder Vibrationen. Prinzipiell existieren zwei Möglichkeiten zur Realisierung des Phasenschubs:

- zeitliches Phasenschieben
- räumliches Phasenschieben

Beim zeitlichen Phasenschieben erfolgt die Änderung der Phase über die Zeit. Das bedeutet, dass sich in einer Folge von Interferogrammen die gesamte Phase von einem Interferogramm zum nächsten ändert oder dass sich die Phase während der Belichtungszeit verändert. Der erste Fall entspricht dem Phase-Stepping, der zweite Fall dem Phase-Shifting [2]. Dieses Verfahren spielt jedoch für die Echtzeitdiagnose keine Rolle, da die dazu notwendige hochpräzise Bewegungen von Spiegeln lediglich den Einsatz im Labor erlaubt [15, 16, 17, 18, 19].
Mit Hilfe des Empfindlichkeitsvektors ergibt sich aus der Phasendifferenz unterschiedlicher Bildpunkte die Deformation oder die Form der Messfläche relativ zur Referenzfläche. Aufgrund der Wellennatur des Lichts weist jedes Interferogramm eine Periodizität auf. Dies führt dazu, dass die durch Phasenschiebealgorithmen gewonnenen Flächen oder Deformationen modulo 2π gefaltet vorliegen. Die Phase ϕ liegt je nach Algorithmus im Intervall $[0, 2\pi[$ oder $[-\pi, \pi[$. Die kontinuierliche Phase ψ ergibt sich aus dem Phasenbild durch verschiedene Unwrapping-Algorithmen, die Abschnitt 2.5 näher beschreibt.
Das räumliche Phasenschieben findet zur selben Zeit, jedoch an unterschiedlichem Ort statt. Eine Technik des räumlichen Phasenschiebens [2] beruht auf der Tatsache, dass die Phase innerhalb eines Speckles konstant, zwischen benachbarten Speckles jedoch beliebig ist. Die Phasenschiebealgorithmen benutzen hierbei zur Phasenberechnung benachbarte Bildpunkte, deren Phasendifferenz α bekannt ist, wobei die Parameter I_0, γ_0 und Φ konstant sein müssen. Diese Bedingungen sind ausreichend erfüllt, wenn sich die Specklegröße mindestens über einen Bereich erstreckt, der die zur Berechnung einer Phase verwendeten Intensitätswerte umfasst [9]. In Bereichen, in denen der Phasenhub 0 oder ganzzahlige Vielfache von π beträgt, schlägt die Phasenberechnung fehl, da in diesem Bereich der Phasenschub für eine Berechnung nicht ausreicht. Deshalb bilden sich in an diesen Stellen Störstreifen aus [2].
Eine weitere Möglichkeit die Phase innerhalb eines einzigen Interferogramms zu verschieben, besteht in der Verkippung der Referenzwelle und einer anschließenden digitalen Phasenschiebung im Bildspeicher [20].
Es besteht außerdem die Möglichkeit die Phase in räumlich getrennten Interferogrammen zu verschieben. Zu dieser Klasse von Interferometern zählt insbesondere das Vier-Kamera-Interferometer [9].

2.5 Unwrapping

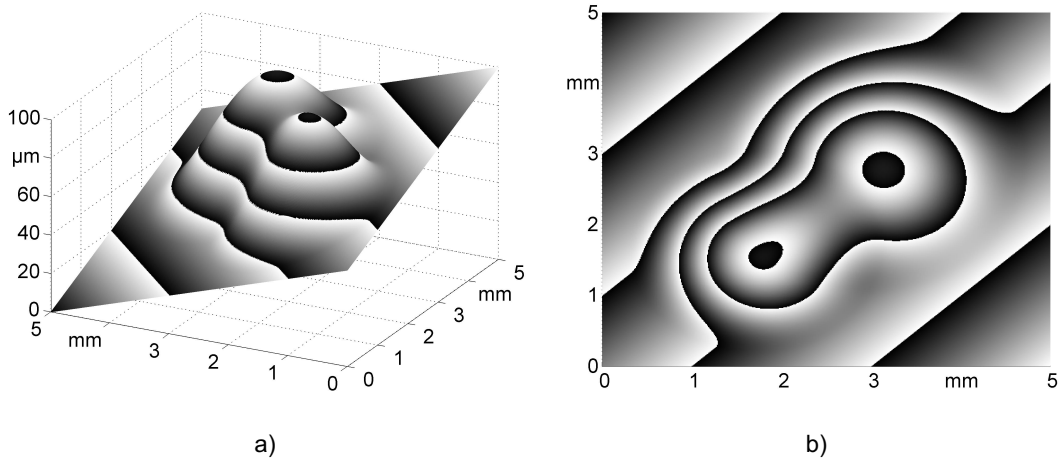


Abbildung 2.4: a) kontinuierliche dreidimensionale Oberfläche, b) Phasenbild (gefaltete Oberfläche oder „Wrapped Phase“) bei $\Lambda = 20\,\mu$m

Um aus einem Phasenbild, das mittels Phasenschieben nach Abschnitt 2.4 entsteht, das dreidimensionale Bild zu rekonstruieren existieren eine Reihe sogenannter Unwrapping-Algorithmen [21, 22]. Neben der Speckleinterferometrie kommen solche Algorithmen vor allem in der Radar-Interferometrie zum Einsatz [25]. Das prinzipielle Problem besteht darin, dass die Oberfläche in gefalteter Form (wrapped) vorliegt. Die Werte im gefalteten Bild entsprechen den tatsächlichen Höhenwerten modulo der synthetischen Wellenlänge Λ, wie Abbildung 2.4 verdeutlicht. Die Unwrapping-Algorithmen lassen sich in zwei Kategorien einteilen [21]:

1. pfadabhängige Methoden

2. pfadunabhängige Methoden

Die pfadabhängigen Methoden beginnen bei der Entfaltung mit einem Startpunkt und bearbeiten davon ausgehend dessen Nachbarn, die wiederum Startpunkte für die weitere Entfaltung darstellen. Die Algorithmen unterscheiden sich dabei vor allem in der Wahl der Pfade. Die Grundlagen der pfadabhängigen Methoden beschreibt Abschnitt 2.5.1.
Die pfadunabhängigen Methoden bearbeiten das Bild als Ganzes. Als Grundlage dient dabei ein globales Optimierungskriterium. Hierzu zählt beispielsweise die

Minimierung der Gradienten im Phasenbild. Die Grundlagen der pfadunabhängigen Methoden beschreibt Abschnitt 2.5.2.
Je nach angewendetem Algorithmus zur Phasenbildgewinnung liegen die Phasenwerte dabei in Intervallen der Breite π. Ohne Beschränkung der Allgemeinheit erfolgen die weiteren Berechnungen mit gefalteten Phasen ϕ in folgendem Intervall:

$$\phi \in [0, 2\pi[\tag{2.12}$$

Die ortsabhängigen gefalteten Phasenwerte $\phi(x, y)$ ergeben sich aus den kontinuierlichen Phasenwerten $\psi(x, y)$ durch die Anwendung des Wrapping-Operators $\mathcal{W}$:

$$\phi(x, y) = \mathcal{W}\{\psi(x, y)\} = \psi(x, y) \bmod 2\pi \tag{2.13}$$

Die tatsächlichen Höhenwerte in Abbildung 2.4 ergeben sich somit aus der kontinuierlichen Phase ψ zu

$$h(x, y) = \frac{\psi(x, y)}{2\pi} \cdot \Lambda \tag{2.14}$$

2.5.1 Pfadabhängige Methoden

Die pfadabhängigen Methoden betrachten das Bild punktweise. Ausgehend von einem Startpunkt $\boldsymbol{r}_0$ bearbeitet der Algorithmus einen benachbarten Punkt und entfaltet diesen. Für den nächsten Iterationsschritt dient einer der beiden bereits entfalteten Punkte als Startpunkt. Eine Reihe dieser Klasse von Algorithmen sind bekannt [21, 22] und werden im Abschnitt 5.1 näher erläutert.
Die kontinuierliche Phase ergibt sich durch eine Integration der Phasendifferenzen [23]. Eine weitere Formulierung des in Gleichung 2.13 eingeführten Wrapping-Operator ergibt:

$$\mathcal{W}\{\psi(n)\} = \phi(n) = \psi(n) - 2\pi k(n), \quad \text{mit } n = 0, 1, \cdots, N-1 \tag{2.15}$$

Wobei die ganzzahligen Faltungskoeffizienten $k(n)$ bewirken, dass die gefalteten Phasenwerte ϕ im Intervall $[0, 2\pi[$ nach Formel 2.12 liegen. Zur weiteren Berechnung dient der Differenzoperator Δ:

$$\begin{aligned} \Delta\{\psi(n)\} &= \psi(n+1) - \psi(n) \\ \Delta\{k(n)\} &= k(n+1) - k(n), \quad \text{mit } n = 0, 1, \cdots, N-2 \end{aligned} \tag{2.16}$$

Die Berechnung der Differenzphasen der gefalteten Phasen nach Formel 2.15 ergibt dann:

$$\Delta\{\mathcal{W}\{\psi(n)\}\} = \Delta\{\phi(n)\} - 2\pi\Delta\{k_1(n)\} \tag{2.17}$$

Eine weitere Anwendung des Wrapping-Operators auf Gleichung 2.17 führt zu:

$$\begin{aligned} \mathcal{W}\{\Delta\{\mathcal{W}\{\psi(n)\}\}\} &= \mathcal{W}\{\Delta\{\phi(n)\}\} \\ &= \Delta\{\psi(n)\} - 2\pi \cdot (\Delta\{k_1(n) + \Delta\{k_2(n)\}\}) \end{aligned} \tag{2.18}$$

Die Faltungskoeffizienten $k_1(n)$ und $k_2(n)$ entstehen durch Anwendung des Wrapping-Operators $\mathcal{W}$. Da die gefundenen Phasenwerte ϕ im Intervall $[0, 2\pi[$ liegen impliziert dies den Zusammenhang:

$$2\pi \cdot (\Delta\{k_1(n) + \Delta\{k_2(n)\}\} = 0 \tag{2.19}$$

Somit kann die kontinuierliche Phase $\psi(m)$ durch die Summation der Gradienten gewonnen werden:

$$\begin{aligned} \psi(m) &= \psi(0) + \sum_{n=0}^{m-1} \Delta\{\phi(n)\} \\ &= \psi(0) + \sum_{n=0}^{m-1} \mathcal{W}\{\Delta\{\mathcal{W}\{\psi(n)\}\}\} \end{aligned} \tag{2.20}$$

Diese Summengleichung führt zu einer Verallgemeinerung die auch für n-dimensionale Signale gilt. Die Phase eines Punktes $\boldsymbol{r}$ ergibt sich damit durch die Integration entlang eines beliebigen Weges C ausgehend von einem Startpunkt $\boldsymbol{r}_0$ zum Endpunkt $\boldsymbol{r}_1$.

$$\psi(\boldsymbol{r}) = \psi(\boldsymbol{r}_0) + \int_C \nabla\psi \cdot d\boldsymbol{r} \tag{2.21}$$

Da gemessene Signale jedoch beispielsweise durch Rauschen und Aliasing-Effekte gestört werden, führt eine einfache Integration nach Gleichung 2.21 in zwei- oder höher dimensionalen Signalen zu teilweise erheblichen Fehlern.
Aus diesen Fehlern ergeben sich dann Residuen wenn das Integral nach Gleichung 2.21 entlang eines geschlossenen Weges C nicht mehr verschwindet. Das Residuum liegt dann in der Fläche, welche der geschlossene Weg umspannt.

$$\psi(\boldsymbol{r}_0) + \oint_C \nabla\psi \cdot d\boldsymbol{r} \neq \psi(\boldsymbol{r}_0) \tag{2.22}$$

Die sich damit ergebenden gemessenen kontinuierlichen Phasen φ sind damit nicht mehr pfadunabhängig. Daraus ergibt sich, dass pfadabhängige Algorithmen im Wesentlichen mit der Suche nach geeigneten Pfaden befassen. Hierzu zählen beispielsweise einfache Algorithmen wie das lineare Scanning oder das Scanning in mehrere Richtungen [22]. Diese Algorithmen versuchen das Phasenbild linear

zeilen- oder spaltenweise oder auch in mehrere Richtungen in Kombination zu bearbeiten. Effektivere Methoden wie „Goldstein's Branch-Cut-Algorithmus“ [24] verbindet Residuen durch Linien miteinander und sperrt damit fehlerhafte Pfade. Die grundlegenden Methoden erläutert Abschnitt 5.2.1 und bewertet sie bezüglich des hardwaregestützten Einsatzes.

2.5.2 Pfadunabhängige Methoden

Pfadunabhängige Methoden betrachten die Bilder als Ganzes. Da sich das Rauschen in gemessenen Phasenbildern niemals vollständig eliminieren lässt, entsteht bei der Integration ein Fehler ϵ. Eine Optimierungsmöglichkeit besteht beispielsweise in der Minimierung der Fehlerquadrate [21]:

$$\epsilon^2 = \int_A \mathcal{W}(\nabla\varphi - \boldsymbol{g})^2 \, dA \tag{2.23}$$

Dabei repräsentiert $\nabla\varphi$ die gemessene kontinuierliche Phase, dA ein elementares Flächenelement, W eine Gewichtungsfunktion wobei $\boldsymbol{g}$ die gemessenen Gradienten darstellen. Diese ergeben sich aus den echten Gradienten $\nabla\phi$ mittels Verfälschungen durch Rauschen in Form des Vektors $\boldsymbol{n}$:

$$\boldsymbol{g} = \nabla\phi + \boldsymbol{n} \tag{2.24}$$

Aus Gleichung 2.23 ergibt sich die Motivation pfadunabhängiger Algorithmen, die auf der Minimierung einer L^p-Norm beruhen:

$$\|f\|_p = \epsilon = \left[\int_A |\mathcal{W}(\nabla\varphi - \boldsymbol{g})|^p \; dA\right]^{\frac{1}{p}} \tag{2.25}$$

In der diskreten Form liegt ein zweidimensionales Phasenbild mit der Dimension $N \times M$ vor. Die gefalteten Phasendifferenzen $\Delta^x_{i,j}$ und $\Delta^y_{i,j}$ ergeben sich dann zu:

$$\Delta^x_{i,j} = \begin{cases} \mathcal{W}\{\phi_{i+1,j} - \phi_{i,j}\} & : \quad i = 0, \cdots, M-2, \\ & \quad\; j = 0, \cdots, N-1 \\ 0 & : \quad sonst \end{cases}$$

$$\Delta^y_{i,j} = \begin{cases} \mathcal{W}\{\phi_{i,j+1} - \phi_{i,j}\} & : \quad i = 0, \cdots, M-1, \\ & \quad\; j = 0, \cdots, N-2 \\ 0 & : \quad sonst \end{cases} \tag{2.26}$$

Damit lässt sich die L^P-Norm mit den Gewichtungsfunktionen $U(i, j)$ und $V(i, j)$ in diskreter Form schreiben:

$$\epsilon^p \quad = \quad \sum_{i=0}^{M-2} \sum_{j=0}^{N-1} U(i,j) |\psi_{i+1,j} - \psi_{i,j} - \Delta^x_{i,j}|^p +$$

$$\sum_{i=0}^{M-1}\sum_{j=0}^{N-2} V(i,j)|\psi_{i,j+1} - \psi_{i,j} - \Delta^{y}_{i,j}|^{p} \tag{2.27}$$

Zur Lösung dieses Gleichungssystems existieren beispielsweise Verfahren, die auf der gewichteten sowie der ungewichteten Minimierung der Fehlerquadrate beruhen [21]. Bei allen Verfahren handelt es sich um iterative Verfahren, deren Rechenaufwand im Vorhinein nicht abschätzbar ist.

2.6 Speckle-Interferometer LAoS

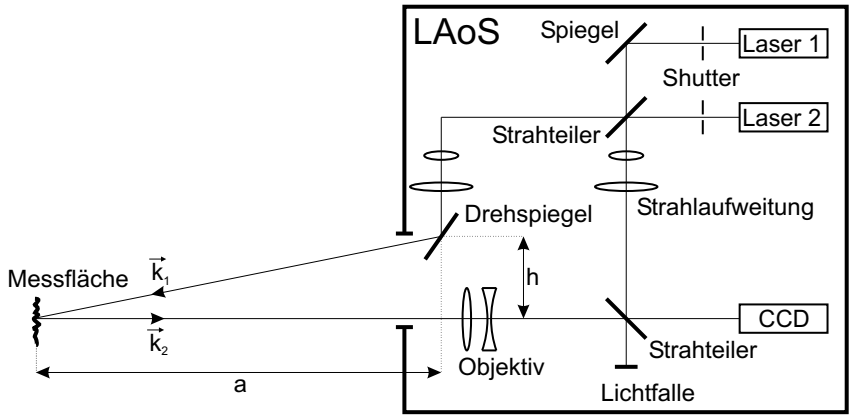


Abbildung 2.5: LAoS - Laser Analysis of Surfaces

Das Speckle-Interferometer LAoS (Laser Analysis of Surfaces) [2, 35, 36] wurde am Lehrstuhl für Messsystem- und Sensortechnik von Hern Dr. Peter Evanschitzky entwickelt und ist in seiner Doktorarbeit eingehend beschrieben. Das Gerät LAoS beruht auf einem modifizierten Mach-Zehnder-Interferometer dessen grundlegende Eigenschaften bereits in Abschnitt 2.2.1 beschrieben sind und arbeitet zur Formvermessung mit dem Zwei-Wellenlängen-Prinzip.
Abbildung 2.5 zeigt den prinzipiellen Aufbau des Gerätes LAoS. Ein Strahlteiler spaltet die unterschiedlichen Wellenlängen λ_1 und λ_2 in einen Objektstrahl und einen Referenzstrahl auf. Im Unterschied zum Mach-Zehnder-Interferometer lenkt ein Spiegel den Objektstrahl aus dem Gerät heraus auf das Messobjekt. Ein Objektiv bildet das vom Messobjekt reflektierte Licht ab. Im Strahlteiler hinter dem Objektiv überlagern sich der Objekt- und der Referenzstrahl, so dass auf dem CCD-Chip das Speckle-Interferogramm entsteht. Mit Hilfe der Strahlaufweitung entsteht aus dem Laserstrahl ein ausreichend großer Beleuchtungsfleck,

der in einem weiten Bereich variabel ist. Der Drehspiegel ermöglicht die Einstellung variabler Messabstände a. Die Shutter steuern die Einstrahlung der beiden Wellenlängen λ_1 und λ_2. Die beiden Wellenlängen lassen sich durch eine externe Temperatur- und Stromregelung manuell in einem weiten Bereich regeln. Dies ermöglicht dann die Einstellung der synthetischen Wellenlänge Λ und damit der Messempfindlichkeit des Gerätes (siehe Abschnitt 2.3). Die Kenngrößen des LA-

Tabelle 2.1: Messparameter LAoS

Messabstand a	$0,4$ m .. 2 m
Messfläche bei $a = 0,4$ m	9 mm $\times$ 8 mm .. 21 mm $\times$ 18 mm
Messfläche bei $a = 2$ m	48 mm $\times$ 40 mm .. 90 mm $\times$ 75 mm
Geräteabmessungen	$700 \times 400 \times 200$ mm
laterale Auflösung	732×579 Messpunkte auf der Messfläche
Messzeit zur Phasenbildberechnung	~ 1 s
Maximal messbarer Höhenunterschied	$\sim 2,5$ mm
Höhenauflösung	$2,9\,\mu$m .. $16,95\,\mu$m

oS fasst Tabelle 2.1 zusammen. Hierbei zeigt sich deutlich die Flexibilität des Gerätes bezüglich des Messabstands und der Messfläche. Diese Flexibilität und der kompakte Aufbau ermöglichen es erstmals, die Speckle-Interferometrie zur Formvermessung auch im industriellen Maßstab einzusetzen. Die Bedienung ist einfach und gleicht der einer herkömmlichen Kamera:

1. Aufstellen des Interferometers
2. Ausrichten der Beleuchtung
3. Einrichten der Größe des Beleuchtungsflecks
4. Abbilden der Messfläche auf die Kamera mittels des Objektivs

Die weitere Messung läuft kontinuierlich und vollautomatisch ab.
Abbildung 2.6 zeigt zur Berechnung des Empfindlichkeitsvektors den einfallenden und reflektierten Strahl an der Messoberfläche. Sein Betrag errechnet sich mit der Beleuchtungswellenlänge λ und den Abständen a und h zu:

$$|\boldsymbol{k}| = |\boldsymbol{k}_1 - \boldsymbol{k}_2| = \frac{4\pi}{\lambda} \cos\left(\frac{\alpha}{2}\right) \tag{2.28}$$

Nach den Kenngrößen des Gerätes beträgt der Messabstand zwischen 40 cm und 2 m wobei die Strecke h genau $5,5$ cm beträgt. Da der Winkel α damit sehr klein

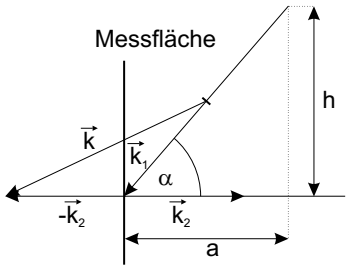


Abbildung 2.6: LAoS - Empfindlichkeitsvektor

und $\cos\alpha \approx 1$ wird, vereinfacht sich die Berechnung für den Empfindlichkeitsvektor:

$$|\boldsymbol{k}| \approx \frac{4\pi}{\lambda} \tag{2.29}$$

Kapitel 3

Mixed-Signal-Hardware-Design

Beim Mixed-Signal-Hardware-Design handelt es sich um die Entwicklung von Schaltungen, die sowohl analoge als digitale Komponenten beinhalten. Hierzu zählen beispielsweise digitale Regler, die analoge Prozesse steuern. Bei der Speckleinterferometrie handelt es sich um Messtechnik, die sowohl einen starken analogen, als auch digitalen Teil beinhaltet. Der analoge Teil besteht aus CCD-Kameras und Halbleiterlaserdioden, während den digitale Teil die Verarbeitung der gemessenen Interferogramme repräsentiert. Die wachsende Komplexität von Mixed-Signal-Designs in der Messsystem- und Kommunikationstechnik stellt Entwickler in Forschung und Industrie vor immer größere Herausforderungen. Insbesondere Hard- und Softwareentwicklung lassen sich nur schwer parallelisieren. Dabei ist es zunächst unerheblich, ob es sich bei dem Design um ein PCB-Design (PCB = Printed Circuit Board) oder einen ASIC (ASIC = Application Specific Integrated Circuit) handelt. Das Designrisiko wächst mit der Komplexität des Designs, da die Fehlerwahrscheinlichkeit zunimmt [13].
Die Realisierung digitaler Komponenten im PCB-Design erfolgt meist in programmierbarer Logik wie beispielsweise FPGAs (FPGA = Field Programmable Gate Arrays), auf die Abschnitt 3.1 näher eingeht. Digitale Schaltungskomponenten sowohl innerhalb eines FPGA als auch innerhalb eines ASIC lassen sich leicht simulieren [26]. Auch analoge Komponenten innerhalb eines ASIC sind durch die Simulation abbildbar [26]. Probleme tauchen meist erst beim Zusammenspiel des FPGA oder ASIC mit der analogen Außenwelt auf, da die Simulation dieses komplexe Verhalten nur unzureichend beschreibt.
Eine Möglichkeit zur Lösung dieses Problems besteht im Einsatz eines Hardware-Emulators. Dieses Werkzeug kann digitale und analoge Hardware direkt verbinden und unmittelbar nach der Simulation das Design in seiner analogen Umgebung zu testen. Es besteht die Möglichkeit ein Design sehr schnell von der Idee bis zum Prototypen zu entwickeln, ohne langwierige Fertigungsschritte wie die PCB- oder ASIC-Herstellung abwarten zu müssen. Gleichzeitig kann ein Software-

Entwickler, bereits lange vor Fertigstellung der Ziel-Hardware, mit dem aktuell vorhandenen Entwicklungsstand am Hardware-Emulator arbeiten.
Der Hardware-Emulator MP3C der Firma Aptix eignet sich besonders gut für die Entwicklung von Mixed-Signal-Designs. Abschnitt 3.3 beschreibt dessen Aufbau, sowie die notwendige Entwicklungsumgebung vom Entwurf über die Simulation bis zur Platinenherstellung. Prinzipiell besteht auch die Möglichkeit ein solches Design in einen ASIC zu überführen, dies ist jedoch angesichts des Prototypencharakters im Labor zu teuer und aufwändig.
Ein wesentlicher Bestandteil des MP3C sind programmierbare Logikbausteine, denen sich Kapitel 3.1 widmet. Zur Beschreibung digitaler Hardware stehen verschiedene Beschreibungssprachen zur Verfügung, die Kapitel 3.2 erläutert. Auf die Betrachtung der Grundlagen digitaler Systeme und des digitalen Schaltungsentwurfs wird an dieser Stelle verzichtet und stattdessen auf die umfangreiche Literatur verwiesen [31, 55, 56].

3.1 Programmierbare Logikbausteine

Komplexe digitale Schaltungen lassen sich im Allgemeinen mit Hilfe applikationsspezifischer Schaltungen (ASIC-Application Specific Integrated Circuit) oder programmierbarer Logikbausteine (PLD-Programmable Logic Devices) realisieren. Die PLDs wiederum unterteilen sich in folgende Klassen:

- SPLD - Simple PLD
- CPLD - Complex PLD
- FPGA - Field Programmable Gate Array

Häufig ist es schwierig Bausteine einer einzigen Klasse zuzuordnen, da inzwischen eine Reihe von Bausteinen existieren, die die Vorteile der verschiedenen Typen kombinieren.
Prinzipiell basieren die Architekturen programmierbarer Logikbausteine auf der Möglichkeit der Abbildung kombinatorischer Funktionen in Speicherzellen oder Nachschlagetabellen (LUT-Lookup Table). Abbildung 3.1 zeigt ein 16-Bit ROM (Read Only Memory), das in Form einer Nachschlagetabelle die vier Funktionen der logischen Verknüpfungstabelle 3.1 zeigt. Die Realisierung komplexer Funktionen mit einfachen Speicherbausteinen beruht auf der Möglichkeit kombinatorische Funktionen mittels Boolescher Algebra umzuformen. In dem Beispiel nach Abbildung 3.1 besteht die Kombinatorik ausschließlich aus UND-Gattern im Dekoderteil und aus ODER-Gattern im Speicherbereich. Die noch fehlenden Negierer sind jeweils an den Eingängen hart verdrahtet und müssen nicht gesondert

programmiert werden.
Es existieren zwei Möglichkeiten, Funktionen mittels UND-Verknüpfungen (Disjunktion, Summe) bzw. ODER-Verknüpfungen (Konjunktion, Produkt) auszudrücken:

- KNF: Konjunktive Normalenform - Produkt aus Summen
- DNF: Disjunktive Normalenform - Summe aus Produkten

Tabelle 3.1: Wahrheitstabelle der Schaltung nach Abbildung 3.1

Adresse		**Daten**			
a_0	a_1	b_0	b_1	b_2	b_3
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0
Funktion		OR	XNOR	XOR	NAND

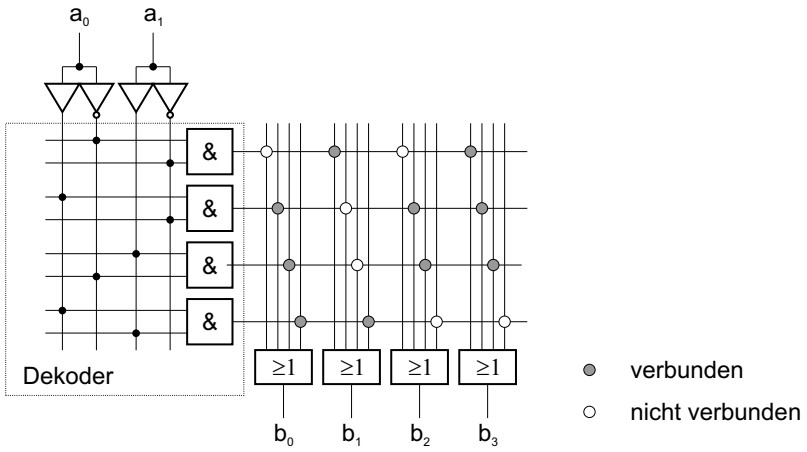


Abbildung 3.1: Prinzipieller Aufbau programmierbarer Logikbausteine mit Nachschlagetabelle

PLDs sind kombinatorisch programmierbare Logikbauelemente. Es existieren die drei alternativen Strukturen PROM (Programmable Read Only Memory), PLA (Programmable Logic Array), PAL (Programmable Array Logic) [29]. SPLDs

sind sehr schnelle Bausteine mit einem definierten Zeitverhalten, da ein Durchlauf durch die Matrix unabhängig von der realisierten Funktion die gleiche Zeit benötigt. Die Einsatzmöglichkeiten stoßen jedoch mit wachsender Komplexität der Schaltungen schnell an ihre Grenzen.
Falls die Komplexität einzelner SPLDs nicht mehr ausreicht, besteht die Möglichkeit mehrere PAL- oder PLA-Bausteine in einem komplexen Bauelement monolithisch zu integrieren [29].

Im Unterschied zu den CPLDs besitzen FPGAs wesentlich kleinere Basiszel-

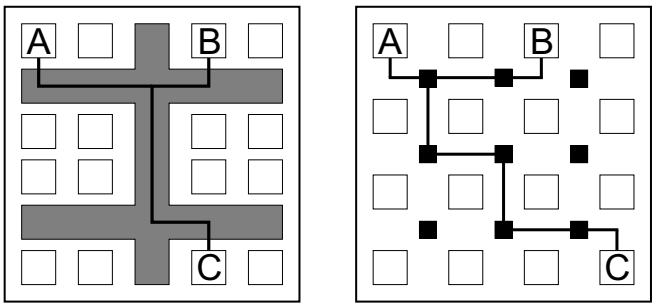


Abbildung 3.2: Charakteristik der Verbindungsstrukturen bei a) CPLD und b) FPGA

len und flexible Verbindungsstrukturen. Das bedeutet, dass nicht mehr alle Zellen untereinander mit einem globalen Bus verschaltet sind, sondern segmentierte Verbindungsstrukturen besitzen. Diesen Unterschied verdeutlicht Abbildung 3.2. Beim FPGA kommt daher den Verbindungsstrukturen eine entscheidende Bedeutung zu. Die Verzögerungszeiten, die durch die Verschaltung der Zellen untereinander entstehen, ist damit variabel [29].
Der Aufbau der Basiszellen unterscheidet sich erheblich von einem Hersteller zum anderen. In den vorgestellten Schaltungen kommen ausschließlich Virtex-1000 FPGAs [32] zum Einsatz. Diese zeichnen sich durch hervorragende Möglichkeiten aus, sowohl große interne Speicherstrukturen, als auch komplexe und schnelle Addierer zu realisieren [32, 29].
Um auch hohe Taktraten verarbeiten zu können verfügt der Virtex-1000 über vier niederohmige Taktnetze, die gewährleisten, dass das Taktsignal in allen Schaltungsteilen des FPGA zur gleichen Zeit ankommt. Ist dies nicht der Fall so entsteht ein sogenannter Schleppfehler, der die Differenz zwischen den Zeitpunkten zweier Taktflanken an aufeinander folgenden Registern beschreibt [32, 29]. Eine besondere Problematik stellt dabei die Einkoppelung von Taktsignalen niedriger Qualität dar. Dazu stehen im FPGA neben besonderer Eingangspuffer

(GBUF - Global Buffer) voll digitale DLLs (Delay-Locked Loop) zur Verfügung. DLLs bieten unter anderem die Möglichkeit, die Zeitverzögerung des Taktsignals zwischen dem Eingangs-Pad und den internen Eingangs-Pins auszugleichen und damit die dauerhafte Synchronisierung des internen und externen Signals zu gewährleisten [32]. Diese Eigenschaft eignet sich für die Einkoppelung externer Kamerasignale in Abschnitt 6.
Ein weiteres wichtiges Element eines FPGA stellen die Ein- und Ausgänge (IOB-Input/Output Block) dar. Der Virtex-1000 stellt an seinen Ein- und Ausgängen getaktete Flip-Flops zur Verfügung um Verzögerungen durch interne Laufzeiten auszuschließen [32]. Dies ermöglicht die Ansteuerung schneller SRAM-Bausteine in Abschnitt 6.

3.2 Beschreibungssprachen für digitale Hardware

Im Allgemeinen kommen die folgenden Hardwarebeschreibungssprachen zum Einsatz:

- VHDL- IEEE Std. 1076
- Verilog - IEEE Std. 1364

VHDL steht für Very High Speed Integrated Circuit Hardware Description Language und wurde 1983 vom amerikanischen Department of Defense erstmals vorgeschlagen. VHDL und Verilog unterscheiden sich im wesentlichen durch ihre Syntax, sind aber ansonsten als gleichwertig zu betrachten [27]. Da die Umsetzung der Algorithmen in den folgenden Kapiteln ausschließlich auf VHDL basiert, wird Verilog [28] nicht weiter betrachtet.
Die drei Schlüsselwörter *Entity* , *Architecture* und *Configuration* ermöglichen den modularen Aufbau eines VHDL-Programms. Die *Entity* stellt die Schnittstelle eines Moduls mit der Außenwelt dar, sie beschreibt die externen Signale. Die *Architecture* beschreibt das Verhalten des Moduls und damit dessen Funktionalität. Da zu jedem Modul mehrere *Architectures* existieren können, gibt die *Configuration* die gültige *Architecture* an.
Ein wesentliches Element innerhalb eines VHDL-Programms stellt der Prozess dar. Ein Prozess besitzt Ein- und Ausgänge sowie eine interne Funktionalität. Bild 3.3 zeigt die verschiedenen Prozessformen.
Der asynchrone Prozess dient der Realisierung rein kombinatorischer Schaltnetze und es kommt je nach Synthese zu unterschiedlichem Laufzeitverhalten. Der synchrone Prozess arbeitet dagegen mit einem flankengesteuerten Eingangspuffer, so dass die Eingangssignale immer taktsynchron die dahinterliegende kombinatorische Logik durchlaufen. Synchrone Prozesse bilden durch eine kaskadierte Verschaltung eine Pipeline mit vollkommen unabhängig arbeitenden Prozessen. Bei

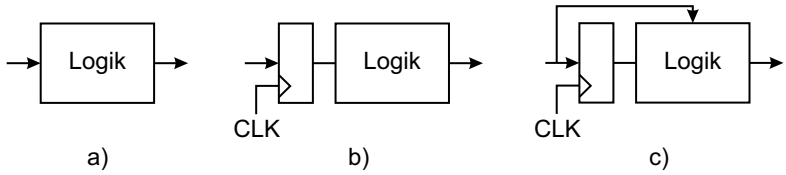


Abbildung 3.3: Formen von Prozessen in VHDL a) asynchroner Prozess, b) synchroner Prozess, c) Mischform

der Synthese ist auf die Einhaltung der Laufzeit innerhalb der kombinatorischen Logik zwischen den einzelnen Eingangspuffern zu achten.
Weitere Sprachelemente ermöglichen die Realisierung von Variablen, Signalen, Entscheidungen oder auch Tristate-Bussen [27, 33, 34].

3.3 Aptix-Explorer

Als wesentliches Werkzeug zur Realisierung der in den folgenden Kapiteln beschriebenen Schaltnetze dient der Hardwareemulator Explorer-MP3C der Firma Aptix. Bild 3.4 zeigt den prinzipiellen Aufbau des MP3C.
Das Kernstück des MP3C besteht aus den sogenannten FPICs (FPIC: Field Programmable Interconnect). Ein FPIC besitzt 1024 Ein- und Ausgänge und seine einzige Aufgabe besteht darin, diese frei programmierbar miteinander zu verschalten. Die erreichbare Grenzfrequenz beträgt dabei 50 MHz [37]. Insgesamt stehen sechs FPICs zur Verfügung. Die FPICs $F1$-$F3$ übernehmen dabei die Aufgabe der Verdrahtung der Standard-Pins. Diese sind in sechs Blöcke aufgeteilt, wobei FPIC $F1$ Standard-Pins im Bereich $ST1$ und $ST4$, FPIC $F2$ die Bereiche $ST2$ und $ST6$ und FPIC $F3$ die Bereiche $ST5$ und $ST3$ verbindet. Die FPICs $F1$-$F3$ sind untereinander wiederum mit Bussen verbunden. Zum Debuggen dienen die FPICs $F4$-$F6$, die jeweils einem der FPICs $F1$-$F3$ zugeordnet sind. Über die Debug-Ports $P5$-$P10$ stehen wahlweise die Pins der FPICs $F4$-$F6$ zur Verfügung. Durch diese Struktur bilden jeweils zwei Standard-Pin Bereiche mit einem FPIC zu den Debug-Ports eine Einheit. Bei Verbindungen zwischen solchen Bereichen kommt mindestens ein zusätzlicher FPIC zum Einsatz, wodurch die Grenzfrequenz sinkt. Die Programmierung der FPICs erfolgt durch den Mikrocontroller $C1$.
Zur Aufnahme der Module, wie beispielsweise FPGAs oder A/D-Wandler dienen zum Einen die Standard-Pin-Bereiche und zum Anderen die Freien-Pins die über Ports von außen erreichbar sind. Die Einkoppelung der Signale in das System erfolgt dann normalerweise durch einen FPGA, der sowohl im Bereich der Standard-Pins, als auch im Bereich der freien Pins steckt.

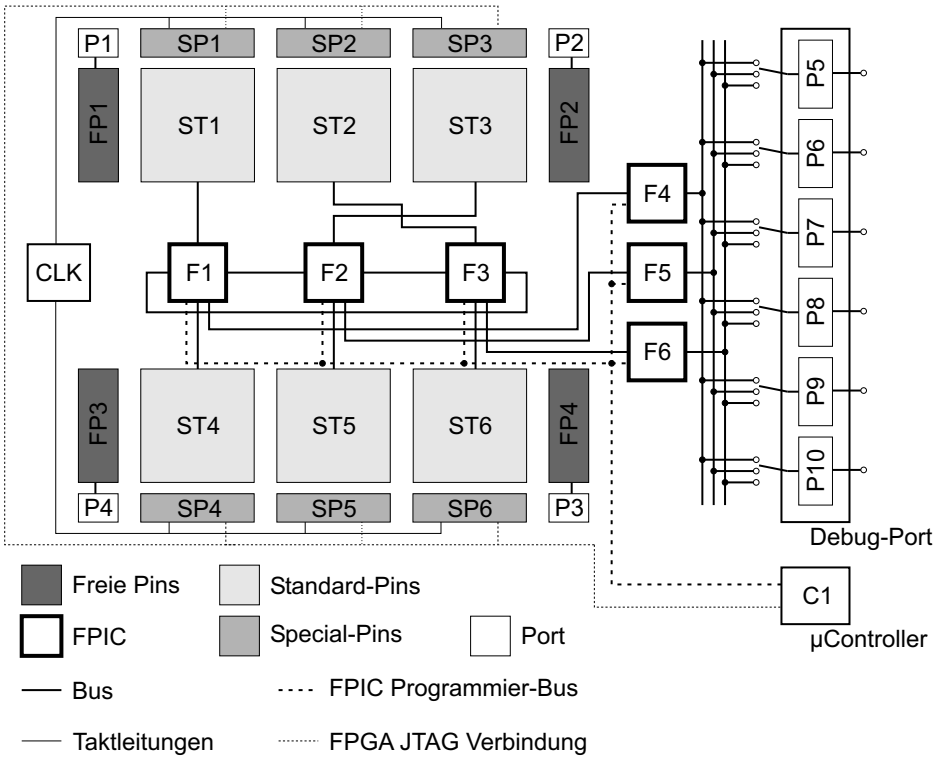


Abbildung 3.4: Prinzipieller Aufbau des MP3C-Explorer der Firma Aptix

Die Spezial-Pins führen neben der Spannungsversorgung auch die Programmierpins für die aufgesteckten FPGAs sowie die Taktleitungen. Insgesamt stehen acht optimierte Taktleitungen zur Verfügung. Diese Leitungen werden wegen der hohen Anforderungen an die Güte direkt beschaltet und nicht programmiert. Die Programmierung der FPGAs erfolgt über eine JTAG-Schnittstelle durch den FPGA.
Die detaillierte Funktionsweise der FPICs sowie nähere Informationen zum MP3C finden sich in der Literatur [37, 38]. Die Entwicklungsumgebung des Aptix-Explorers besteht aus den folgenden Komponenten verschiedener Hersteller:

- Design Analyzer der Firma Synopsis in der Version 2000.11-SP2 [39]
- Synplicity der Firma Synplify [40]
- XILINX Core Generator (siehe Abschnitt [41]
- Explorer Software der Firma Aptix in der Version v2001.Q4.SP1 [38]

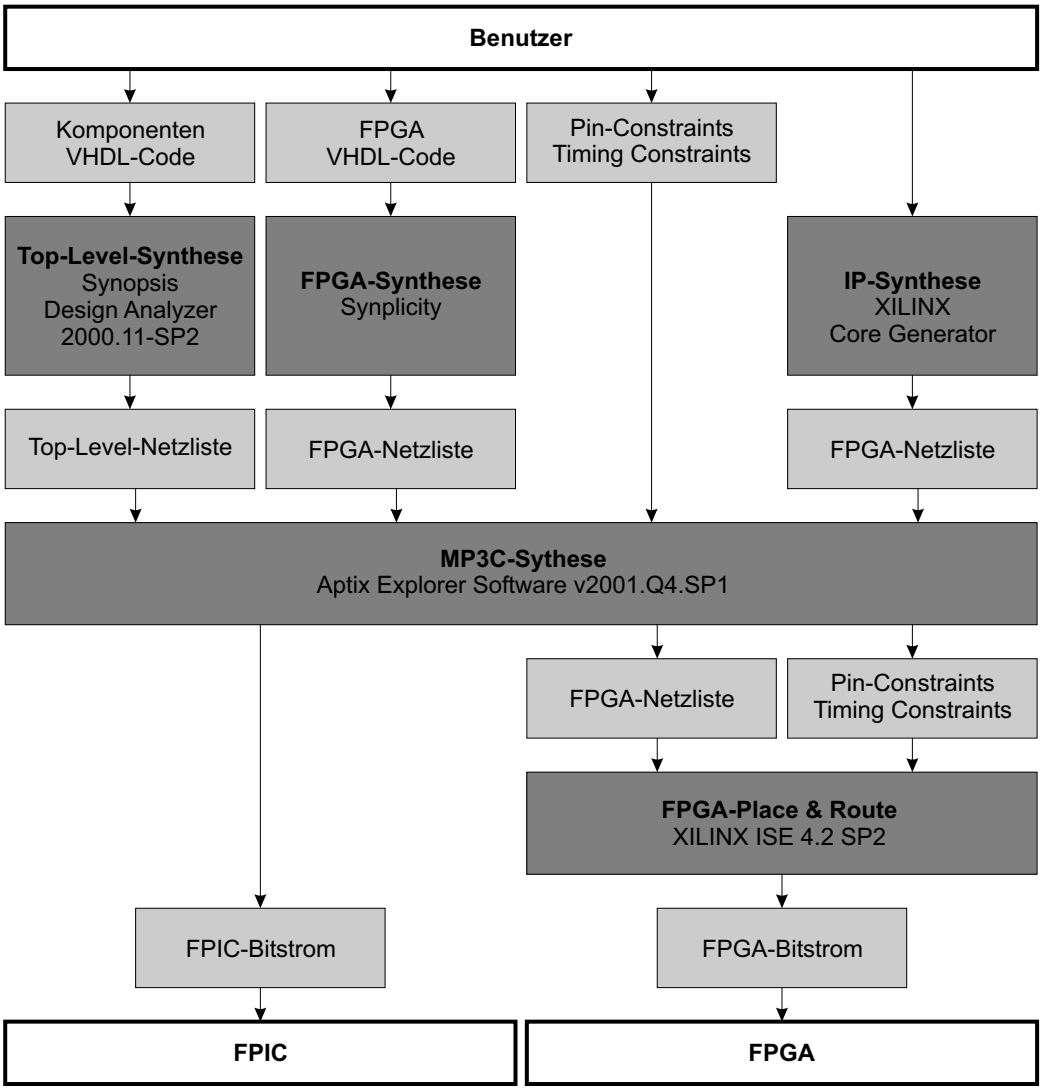


Abbildung 3.5: Entwicklungsablauf von der Benutzereingabe bis zum Bitstrom zur Programmierung von FPGA und FPIC

- Place & Route der Firma XILINX in der Version ISE .42 SP2 [42]
- Modeltech Modelsim XE Starter 5.5e_p1 von XILINX [43]

Abbildung 3.5 zeigt den Ablauf von der Benutzereingabe bis zu den Bitströmen zur Programmierung der FPGAs und FPICs.
Die Aptix-Explorer-Software stellt das zentrale Werkzeug der Entwicklungsumgebung dar. Hier werden die verschiedenen Netzlisten und andere Eingaben zusammengeführt und für den MP3C aufbereitet. Eine zentrale Aufgabe besteht dabei in der Verdrahtung der verschiedenen Komponenten untereinander durch die Programmierung der FPICs. Eine weitere wesentliche Aufgabe betrifft das Debugging. Zunächst besteht die Möglichkeit Signale, die über die FPICs geroutet werden, auf den Debug-Port umzulenken. Bei einigen FPGAs, wie beispielsweise dem Virtex-1000 von XILINX, können auch interne Signale automatisch über den Debug-Port sichtbar gemacht werden. Das Programm bietet ferner die Möglichkeit Pin- sowie Timing-Constraints manuell zu manipulieren.
Der Design Analyzer erstellt die Top-Level-Netzliste aus dem Toplevel-VHDL-Code. Die Top-Level-Netzliste nimmt eine besondere Stellung ein, da sie die Grundlage für die Programmierung der FPICs darstellt. Sie beschreibt die Pin-Belegung der in den Standard-Pin-Bereichen eingesteckten Komponenten und deren Verbindung untereinander. Das Programm Synplicity dient der Synthese der VHDL-Programme für die FPGAs und erzeugt eine FPGA-Netzliste. Der XILINX Core-Generator erzeugt Netzlisten für IP-Module, die in den FPGA-Programmen als Black-Boxen verwendet werden. Die Überführung der FPGA-Netzlisten in Bitströme zur Programmierung übernimmt das Place & Route-Programm. Neben den FPGA-Netzlisten erhält diese Software Eingaben in Form der Pin- und Timing-Constraints. Die Pin-Constraints definieren, welche Signale an welchen physikalisch vorhandenen Pins eines FPGA liegen. Nicht explizit vom Benutzer festgelegte Pins definiert die Explorer Software automatisch. Die Timing Constraints zwingen das Place & Route-Programm dazu, definierte Signallaufzeiten einzuhalten um die Funktionalität des FPGA-Programms zu gewährleisten.
Neben dieser Emulationsmöglichkeit steht auch noch die Simulationsumgebung zur Verfügung. Dieses Tool dient der prinzipiellen Überprüfung der Funktionalität der programmierten Module. Erst nach einer erfolgreichen Simulation beginnt die Emulation, da ein Designdurchlauf bei der Simulation erheblich schneller erfolgt. Diese Simulation beschränkt sich jedoch auf einzelnen Module innerhalb des FPGA. Das Zusammenwirken der einzelnen Module, insbesondere mit der Außenwelt, erfolgt ausschließlich durch die Emulation, da die physikalische Realität sich einer geschlossenen Beschreibung im Allgemeinen entzieht.

Kapitel 4

Echtzeit Phasenbildberechnung

Der erste Schritt bei der Auswertung von Speckleinterferogrammen zur Formvermessung besteht in der Phasenbildberechnung aus einem oder mehreren Interferogrammen nach Abschnitt 2.4. Abschnitt 4.2 zeigt die Umsetzung solcher Auswertungen in spezialisierter Hardware, nachdem Abschnitt 4.1 die wichtigsten Algorithmen vorgestellt hat.

4.1 Algorithmen

Wie bereits in Abschnitt 2.4 erwähnt, sind für eine Berechnung der Grundintensität I_0, der Modulation γ_0 sowie der Phase ϕ mindestens drei Messungen mit bekanntem Phasenschub notwendig. Die Bestimmung des exakten Phasenschubs bei räumlichem Phasenschieben ist jedoch bei dem verwendeten Messgerät LAoS nach Abschnitt 2.6 nicht möglich [2]. Durch die Hinzunahme mehrerer Messungen entstehen überbestimmte Gleichungssysteme, die eine Optimierung der Lösungen nach verschiedenen Kriterien zulassen. Die folgenden Algorithmen kommen im Allgemeinen bei Messsystemen zur Form- bzw. Deformationsmessung zum Einsatz:

- Algorithmus nach Hariharan-Schwider
- Algorithmus nach Carré
- 3-Schritt-Algorithmus
- 4-Schritt-Algorithmus
- 5-Schritt-Algorithmus

Die Algorithmen nach Hariharan-Schwider [45] und Carré [46] sind spezielle 4-Schritt-Algorithmen. Die eingesetzten Messsysteme arbeiten am Besten mit diesen beiden Algorithmen, so dass die Betrachtung der anderen Verfahren eine untergeordnete Rolle spielt [2].
Der Algorithmus nach Hariharan-Schwider erfordert eigentlich einen Phasenhub von $\alpha = \pi/2$. Er erweist sich jedoch im praktischen Einsatz als extrem robust gegenüber Abweichungen von dieser Voraussetzung [44] und kommt damit sowohl beim zeitlichen als auch beim räumlichen Phasenschieben zum Einsatz. Mit Gleichung 2.11 ergibt sich somit für die vier Messungen:

$$\begin{aligned} I_1 &= I_0\left[1+\gamma_0\cos\left(\phi+0\cdot\frac{\pi}{2}\right)\right] \\ I_2 &= I_0\left[1+\gamma_0\cos\left(\phi+1\cdot\frac{\pi}{2}\right)\right] \\ I_3 &= I_0\left[1+\gamma_0\cos\left(\phi+2\cdot\frac{\pi}{2}\right)\right] \\ I_4 &= I_0\left[1+\gamma_0\cos\left(\phi+3\cdot\frac{\pi}{2}\right)\right] \end{aligned} \tag{4.1}$$

Die gesuchte Phase errechnet sich daraus zu [45]:

$$\phi = tan^{-1}\left[\frac{3I_2-(I_1+I_3+I_4)}{(I_1+I_2+I_4)-3I_2}\right] \tag{4.2}$$

Der Algorithmus nach Carré arbeitet im Gegensatz dazu mit variablem Phasenschub:

$$\phi = tan^{-1}\left[\frac{\sqrt{[(I_1-I_4)+(I_2-I_3)][3(I_2-I_3)-(I_1-I_4)]}}{(I_2+I_3)-(I_1+I_4)}\right] \tag{4.3}$$

Theoretisch bringt dieser Algorithmus bessere Ergebnisse. In der Praxis ist Unterschied im Messergebnis jedoch zu vernachlässigen [2]. Da die zusätzliche Berechnung der Wurzel einen erheblichen Aufwand in der Verarbeitungskette der Interferogramme darstellt wird im folgenden nur noch der Algorithmus nach Hariharan-Schwider betrachtet. Bei der zwei Wellenlängen-Interferometrie werden zwei komplette Sätze Intensitäten $I_1 - I_x$ bei den Wellenlängen λ_1 und λ_2 aufgenommen. Daraus ergeben sich zwei unterschiedliche Phasen ϕ_1 und ϕ_2, wobei w_A den Zähler des Argumentes unter dem Arkustangens nach Gleichung 4.2 darstellt und w_B den Nenner:

$$\phi_1 = \tan^{-1}\left(\frac{w_A}{w_B}\right) \tag{4.4}$$

$$\phi_2 = \tan^{-1}\left(\frac{w_C}{w_D}\right) \tag{4.5}$$

Die Differenz $\Delta\phi$ der beiden Phasen ϕ_1 und ϕ_2 ergibt somit:

$$\Delta\phi = \tan^{-1}\left(\frac{w_A \cdot w_D - w_B \cdot w_C}{w_B \cdot w_D + w_A \cdot w_C}\right) \tag{4.6}$$

Bei der Anwendung der Arkustangens-Funktion ergibt sich ein Wertebereich im Intervall $[-\pi/2; \pi/2[$. Die Berechnung des Phasensignals im Intervall von $[0; 2\pi[$ ergibt sich aus einer Vorzeichenbetrachtung der Terme $(w_A \cdot w_D - w_B \cdot w_C)$ und $(w_B \cdot w_D + w_A \cdot w_C)$. Damit erfolgt die Bestimmung des Quadranten, in dem das Ergebnis liegt [11]. Diese Operation wird auch als Arkustangens2 [47] bezeichnet, so dass sich aus Gleichung 4.6 ergibt:

$$\Delta\phi = \arctan 2\left(\frac{w_A \cdot w_D - w_B \cdot w_C}{w_B \cdot w_D + w_A \cdot w_C}\right) \tag{4.7}$$

4.2 Umsetzung in Hardware

Die Umsetzung der in Abschnitt 4.1 vorgestellten Algorithmen lässt sich zunächst in zwei Bereiche aufteilen:

1. Berechnung von w_A, w_B, w_C und w_D aus Gleichung 4.4 und 4.5
2. Berechnung der Phasendifferenz $\Delta\phi$ nach Gleichung 4.7

Dabei verändert sich Bereich 1 mit dem eingesetzten Phasenschiebealgorithmus. Teil 2 dagegen bleibt in allen Fällen gleich. Abschnitt 4.2.1 beschreibt die Umsetzung der Addition und Subtraktion, Abschnitt 4.2.2 beschäftigt sich mit der Multiplikation, Abschnitt 4.2.3 mit der Division und Abschnitt 4.2.4 zeigt die Umsetzung des Arkustangens.

4.2.1 Addition, Subtraktion

Die prinzipielle Funktionsweise eines Addierers [31] zeigt Abbildung 4.1 [31]. Die kleinste Einheit eines Addierers stellt der Halbaddierer dar, der zwei Bit A_0 und A_1 addiert. Ein Volladdierer für ein Bit berechnet zunächst die Summe aus den Eingängen A_n und B_n mit Hilfe des Halbaddierers HA_1 und addiert zu diesem Ergebnis den Übertrag der vorhergehenden Additionsstufe C_{n-1} mittels des Halbaddierers HA_2. Ein Volladdierer oder Ripple-Carry-Addierer, für n Bit besteht prinzipiell aus der Kaskadierung von n Volladdierern. Der Nachteil dieser Art der Verschaltung besteht jedoch darin, dass durch die Kaskadierung das Übertragsbit alle Stufen nacheinander durchlaufen muss. Der Addierer nach dem Local-ahead-Carry-Prinzip [31] behebt dieses Problem, indem er in einer speziellen Logik die Überträge gesondert berechnet. Die binäre Subtraktion von A und B beruht auf

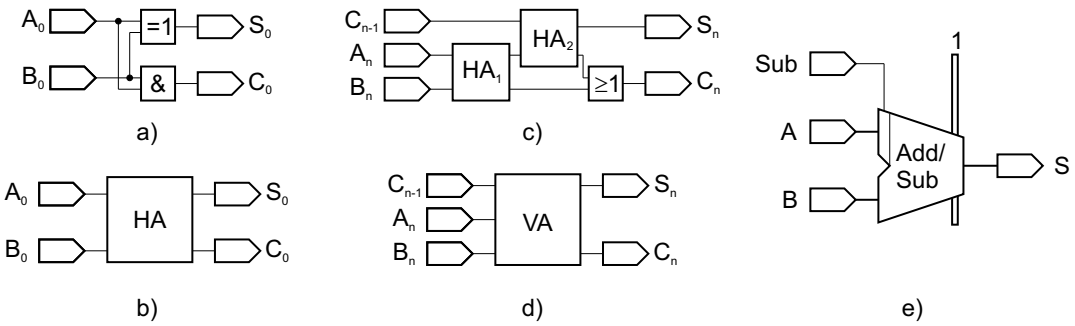


Abbildung 4.1: Halbaddierer: a) Schaltung, b) Symbol. Ein Bit Volladdierer: c) Schaltung, d) Symbol. e) Symbol Addierer und Subtrahierer für n Bit

der binären Addition des Zweierkomplements von *B* mit *A*.
Die Realisierung der Additionen und Subtraktionen erfolgt im Folgenden mit Hilfe eines XILINX-Cores, der einen parametrisierbaren Addierer und Subtrahierer zur Verfügung stellt [57].
Das gestreckte Rechteck symbolisiert die taktsynchronen Register, wobei die jeweiligen Bearbeitungsschritte nummeriert sind.

4.2.2 Multiplikation

Die binäre Multiplikation beruht auf dem Schulalgorithmus zur Multiplikation. Dabei werden einzelne Teilprodukte gebildet und diese aufsummiert [59]. Es existieren drei grundlegende Typen von Multiplizierern:

- Sequentieller Multiplizierer
- Paralleler Multiplizierer
- Array-Multiplizierer

Der sequentielle Multiplizierer erzeugt die Teilprodukte nacheinander und addiert jedes Teilprodukt zur bisherigen Zwischensumme. Der parallele Multiplizierer erzeugt alle Teilprodukte simultan und benutzt dann einen Mehroperanden-Addierer [60] zur Summenbildung. Der Array-Multiplizierer besteht aus identischen Zellen, welche die Teilprodukte gleichzeitig erzeugen und aufsummieren [59, 31].
Zur Realisierung der Multiplikation dient im Folgenden ein parametrisierbarer Multiplizierer als Core der Firma XILINX [58].

4.2.3 Division

Die Standard-Methoden zur Division sind Module mit extrem niedrigen Grenzfrequenzen, welche für Echtzeitalgorithmen ausscheiden [41]. Lediglich die Division einer Zahl w durch eine ganzzahlige Potenz von zwei lässt sich einfach mit Hilfe einer Shift-Operation ausführen, da gilt [48]:

$$\frac{w}{2^{s_r}} = w \cdot 2^{-s_r} \tag{4.8}$$

Dazu wird der Wert w um s_r Bits nach rechts verschoben. In der weiteren Realisierung der Algorithmen kommen deshalb keine Divisionen, außer durch ganzzahlige Potenzen von zwei zum Einsatz.

4.2.4 Arkustangens

Die Berechnung des Arcustangens2 und der Quadratwurzel sind keine, im Sinne eines FPGA, einfach zu realisierenden Funktionen. Zur Berechnung dieser Funktionen stehen prinzipiell folgende Möglichkeiten zur Verfügung [49, 50]:

- Reihenentwicklung
- Polynomiale Interpolation
- Nachschlagetabelle
- Iterative Verfahren

Zur Berechnung des Arkustangens dienen die beiden folgenden Reihenentwicklungen:

$$\tan(x)^{-1} = \begin{cases} x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots + \frac{(-1)^n \cdot x^{2n+1}}{2n+1} & |x| \leq 1 \\ \\ \pm\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^2} - \frac{1}{5x^5} + \cdots + \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} & |x| > 1 \end{cases} \tag{4.9}$$

Der Nachteil dieses Verfahrens liegt auf der Hand, da beide Reihenentwicklungen, für den Hardwareeinsatz ungünstige, Divisionen verwenden. Ein weiteres Argument gegen den Einsatz der Reihenentwicklung besteht im maximalen relative Fehler, der auch bei einer Approximation von sieben Schritten immer noch bei $f_{rel} \approx 2,4\%$ liegt.
Eine Interpolation der Arkustangens-Funktion mit einem Polynom n-ten Grades kommt ohne Divisionen aus:

$$\tan(x)^{-1} = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 x^0 \tag{4.10}$$

Der relative Fehler f_{rel} hängt jedoch hierbei stark vom Grad n des Polynoms ab. Bei einem Polynom sechster Ordnung beträgt der maximale relative Fehler immer noch $f_{rel,max} \approx 8\%$. Ein weiterer Nachteil besteht in der notwendigen Beschränkung des Messbereichs. Eine Beschränkung der Argumente auf $|x| < 32$ führt bei $|x| >= 32$ bei dem dann angenommenen Ergebnis von $\pi/2$ zu einem maximalen relativen Fehler von $f_{rel,max} \approx 1\%$. Es besteht prinzipiell auch die Möglichkeit einer schrittweisen linearen Approximation. Zu deren Realisierung ist jedoch auch eine Nachschlagetabelle notwendig.
Die Nachschlagetabelle legt die Ergebniswerte zu einem Argument im Speicher ab. Dies erfordert die Nutzung eines FPGA-internen oder eines zusätzlichen externen Speichermoduls. Um einen relativen Fehler von $f_{rel} < 1\%$ zu erhalten sind mindestens sieben Bit für die Nachkommastellen erforderlich, da die maximale Steigung der Arkustangensfunktion $d\tan^{-1}(x)/dx = 1$ beträgt. Bei einer Beschränkung der Argumente auf $|x| < 32$ und einer Genauigkeit der Einträge von 12 Bit ergibt sich eine Tabellengröße von $2^{(log_2(32)+7)} \cdot 12$ Bit = 49152 Bit. Speicher dieser Größe überschreiten sehr schnell die Kapazität eines FPGA. Ein externes Speichermodul erfordert jedoch einen eigenen Bus, was die Auswertung um mindestens den Faktor zwei bremsen würde.
Iterative Verfahren bieten die Möglichkeit zur Berechnung komplexer Funktionen mit einfachen, in Hardware realisierbaren, Operationen. Neben einigen Verfahren wie der Iteration nach De Lugish oder nach Baker erweist sich das CORDIC-Verfahren als der geeignetste Algorithmus dieser Klasse. Die entscheidenden Vorteile gegenüber anderen Verfahren besteht in der immer konstanten Zahl der Zyklen einer Berechnung und der Möglichkeit die Genauigkeit durch ein Pipelining-Verfahren einzustellen, ohne dabei die Verarbeitungsdauer großer Datenmengen entscheidend zu erhöhen.
Der CORDIC-Algorithmus basiert auf der Drehung eines Vektors $\boldsymbol{l}$ mit den Koordinaten $\boldsymbol{l} = (x_0, y_0)$ um den Winkel α:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{4.11}$$

Diese Gesamtdrehung des Vektors um den Winkel α wird auch als Makrorotation bezeichnet. Gleichung 4.11 lässt sich Umformen zu [49, 48]:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{1 + \tan^2(\alpha)}} \begin{bmatrix} 1 & -\tan(\alpha) \\ \tan(\alpha) & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{4.12}$$

Der Grundgedanke des CORDIC-Verfahrens besteht nun darin, die Drehung um den Winkel α nicht in einem einzigen Schritt auszuführen, sondern durch einzelne Drehungen, den sogenannten Mikrorotationen, um fest vorgegebene Teilwinkel α_i

anzunähern:

$$\alpha = \sum_{i=0}^{n-1} \sigma_i \alpha_i \tag{4.13}$$

Dabei beschreibt $\sigma_i \in \{1; -1\}$ die Drehrichtung der $i+1$-ten, so dass die aus den Teildrehungen resultierende Gesamtdrehung den Winkel α möglichst gut approximiert. Die Teildrehungen vereinfachen sich in der Ausführung durch folgende Definition der Teilwinkel:

$$\alpha_i = \tan^{-1}(2^{-i}) \tag{4.14}$$

Dies bedeutet, dass nicht der Winkel α selbst, sondern der Tangens des Winkels α gedreht wird. Damit folgt für die Mikrorotation:

$$\begin{aligned} x_{i+1} &= k_i \left(x_i - \sigma_i y_i 2^{-i} \right) \\ y_{i+1} &= k_i \left(y_i + \sigma_i y_i 2^{-i} \right) \end{aligned} \tag{4.15}$$

Dabei ist k_i unabhängig vom Winkel der Drehung:

$$k_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \tag{4.16}$$

Um die Konvergenz des Verfahrens zu gewährleisten, müssen die Teilwinkel α_i eine monoton fallende Folge bilden. Die Konvergenzbedingung lautet [49, 51]:

$$\alpha_i - \sum_{j=i+1}^{n-1} \alpha_j < \alpha_{n-1} \tag{4.17}$$

Die Einführung der Steuervariablen z_i ermöglicht die Ermittlung der Drehrichtung der nächsten Iteration ausgehend von einem Anfangsdrehwinkel $z_0 = \alpha$. Insgesamt ergeben sich damit die CORDIC-Gleichungen, unter Vernachlässigung des Streckungsfaktors k_i einer Mikrorotation, zu [52]:

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i y_i 2^{-i} \\ y_{i+1} &= y_i + \sigma_i x_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i \tan^{-1}(2^{-i}) \end{aligned} \tag{4.18}$$

Wobei der Drehwinkel sich nach folgender Gleichung berechnet:

$$\sigma_i = \begin{cases} 1 & : \quad z_i \geq 0 \\ -1 & : \quad z_i < 0 \end{cases} \tag{4.19}$$

Gleichung 4.15 beschreibt eine Drehstreckung, wobei sich die wahre Länge des Vektors mit Hilfe eines Korrekturfaktors k_n ergibt, der sich als Produkt der einzelnen Streckungen k_i ergibt:

$$k_n = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1 + 2^{-2i}}} \tag{4.20}$$

Für $n \to \infty$ ergibt sich ein Wert von $k_\infty \approx 0,6073$.
Diese Art des CORDIC-Algorithmus, der als Rotation bezeichnet wird, erzielt nach einer Makrorotation das folgende Ergebnis.

$$\begin{aligned} x_n &= k_n (x_0 \cos z_0 - y_0 \sin z_0) \\ y_n &= k_n (y_0 \cos z_0 - x_0 \sin z_0) \\ z_n &= 0 \end{aligned} \tag{4.21}$$

Daneben existiert noch eine Art des CORDIC-Algorithmus, bei der die Variation des Vektors $\boldsymbol{l} = (x_0, y_0)$ in Form einer Dreh-Streckung erfolgt, so dass er nach Abschluss aller Iterationen in der x-Achse liegt. Diese Betriebsart wird auch als Vectoring bezeichnet. Durch die Summation der Teildrehwinkel α_i ergibt sich die Phase und über die Betrachtung der resultierenden x-Koordinate der Betrag des Vektors $\boldsymbol{l}$. Der Winkel α beträgt in diesem Fall:

$$\alpha = -\tan^{-1}\left(\frac{x_0}{y_0}\right) \tag{4.22}$$

Daraus ergibt sich dann mit Gleichung 4.12 [49, 51]:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sqrt{x_0^2 + y_0^2} \\ 0 \end{bmatrix} \tag{4.23}$$

Für den Fall des Vectoring bleiben die CORDIC-Gleichung nach 4.15 unverändert, lediglich die Berechnung des Drehwinkels ändert sich [52]:

$$\sigma_i = \begin{cases} -1 & : \quad y_i \geq 0 \\ 1 & : \quad y_i < 0 \end{cases} \tag{4.24}$$

Das Ergebnis der Iteration nach dem Vectoring ergibt damit:

$$\begin{aligned} x_n &= k_n \sqrt{x_0^2 + y_0^2} \\ y_n &= 0 \\ z_n &= z_0 + \tan^{-1}\left(\frac{x_0}{y_0}\right) \end{aligned} \tag{4.25}$$

Um eine definierte Genauigkeit zu erhalten sind mindestens fünf Iterationen notwendig [49, 51].
Somit lassen sich Vektor-Drehungen leicht durch Shift-Operationen und Summationen ausführen. Durch geschickte Substitutionen sowie die Wahl der Anfangsbedingungen lassen sich eine Reihe von Funktionen, wie beispielsweise die Wurzel oder hyperbolische Funktionen berechnen [49, 51, 52].
Aus diesen Vorüberlegungen lässt sich die CORDIC-Iteration für die Berechnung des Arkustangens2 ableiten, der zur Ermittlung der Phase nach Abschnitt 4.1 dient. Dabei wird der Arkustangens des Bruchs aus den Werten x_0 und y_0 unter Berücksichtigung des Quadranten, der sich aus den Vorzeichen von x_0 und y_0 ergibt, bestimmt. Der Arkustangens2 ergibt sich unmittelbar aus der CORDIC-Iteration nach Gleichung 4.25. Der Startwert z_0 addiert sich dabei als Offset auf das Endergebnis. Für die Bestimmung des Winkels ist keine Längenkorrektur notwendig.
Die Realisierung einer vollständig parallelen Mikrorotation zeigt Abbildung 4.2. Die Schaltung berechnet die Gleichungen 4.15.
Jede Mikrorotation besitzt am Eingang die Register R_1, R_2 und R_3 welche die

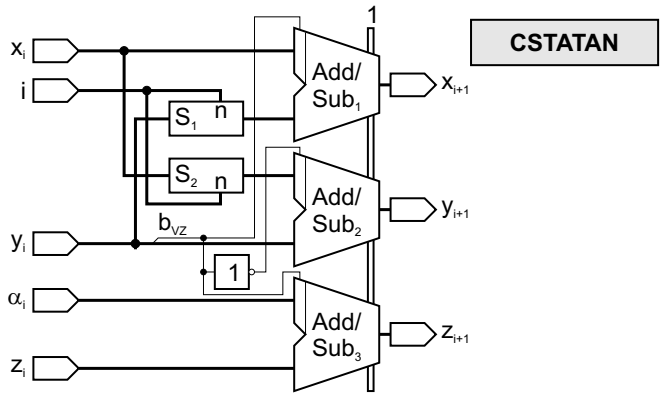


Abbildung 4.2: a) Makrorotation zur Berechnung des Arkustangens

Eingangswerte x_i, y_i und z_i zum Synchronisieren des Prozesses zwischenspeichern. Die Shift-Operatoren S_1 und S_2 schieben ihren Eingangswert um n-Bit nach rechts und realisieren damit die Berechnung des Wertes von 2^{-n}. b_{VZ} entspricht dem höchstwertigsten Bit, was in der Notation des Zweierkomplements gerade dem Vorzeichenbit entspricht [31]. Die Addierer Add/Sub_1 und Add/Sub_3 subtrahieren bei negativem und der Addierer Add/Sub_2 bei positivem Vorzeichen des Wertes y_i. Der Eingangsgröße α_i entspricht dem Wert von $tan^{-1}(2^{-i})$.
Zur Berechnung einer Makrorotation steht die Möglichkeit einer rückgekoppelten Struktur sowie einer Pipeline zur Verfügung. Die Pipeline, nach Abbildung

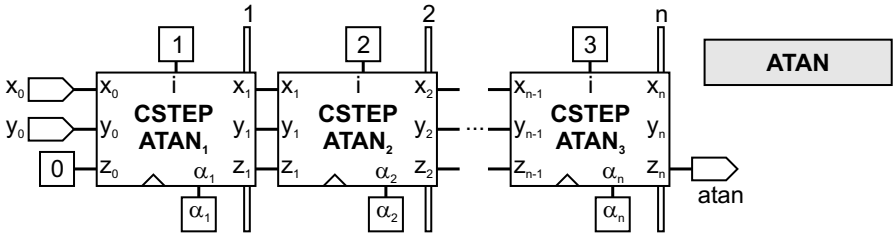


Abbildung 4.3: Makrorotation mit Pipeline

4.3, kann, im Gegensatz zur rückgekoppelten Struktur, nach jedem Taktzyklus neue Werte aufnehmen und einen Ergebniswert liefern. Lediglich das erstmalige Füllen der Pipeline erzeugt eine Verzögerung in der Bearbeitung. Der Gesamtdatendurchsatz einer Pipeline übersteigt die der rückgekoppelten Struktur um ein Vielfaches, so dass im Folgenden nur noch die Pipelinestruktur betrachtet wird.
Die Einstellung des CORDIC-Algorithmus erfolgt durch die Betrachtung des kleinsten Drehwinkels, der sich beim letzten Iterationsschritt I ergibt. Daraus errechnet sich der Skalierungsfaktor $S_{k,min}$ so, dass die letzte Drehung mindestens noch ein LSB (least significant bit) ergibt.

$$S_{k,min}(I) = \frac{1}{\tan^{-1}(2^{I-1})} \tag{4.26}$$

Durch die Skalierung $S_{k,min}$ ergibt sich dann ein ganzzahlig darstellbarer Wertebereich $W_{k,min}$ von

$$W_{k,min} = \text{floor}\,[2\pi S_{k,min}] \tag{4.27}$$

Die Funktion floor $[w]$ rundet den Wert w auf die nächste ganze Zahl ab. Der Wertebereich $W_{k,min}$ nutzt jedoch nicht vollständig die zur Darstellung benötigte Bittiefe aus, so dass sich Wertebereich durch einen günstigeren Skalierungsfaktor auf den Wertebereich $W_{k,max}$ erweitern lässt.

$$W_{k,max} = 2^{\text{ceil}[\log_2[W_{k,min}]]} - 1 \tag{4.28}$$

Die Funktion ceil $[w]$ rundet den Wert w auf die nächste ganze Zahl auf. Mit den Gleichungen 4.26, 4.27 und 4.28 ergibt sich dann der maximale Skalierungsfaktor $S_{k,max}$, der bei gegebener Iterationstiefe I eine maximale Ausnutzung des Wertebereichs $W_{k,max}$ garantiert.

$$S_{k,max}(I) = \frac{2^{e_s} - 1}{2\pi} \tag{4.29}$$

Tabelle 4.1: Fehler beim CORDIC-Algorithmus in Abhängigkeit der Iterationstiefe I, und dem Skalierungsfaktor S_k

I	$W_{k,min}$	$S_{k,min}$	$f_{rel,\arctan,max}$	$W_{k,max}$	$S_{k,max}$	$f_{rel,\arctan,min}$
5	100	16,021	1,4967	127	20,21	1,2841
6	201	32,01	0,7461	255	40,58	0,6408
7	402	64,005	0,3731	511	81,33	0,32
8	804	128,003	0,1865	1023	162,82	0,1599
9	1608	256,001	0,0933	2047	325,79	0,0799
10	3216	512,001	0,0466	4095	651,74	0,04
11	6433	1024	0,0233	8191	1303,64	0,02
12	12867	2048	0,0117	16383	2607,44	0,01

Dabei gilt für den Exponent e_s:

$$e_s = \text{ceil}\left[\log_2\left[\text{floor}\left[2\pi\frac{1}{\tan^{-1}(2^{I-1})}\right]\right]\right] \tag{4.30}$$

Die Tabelle 4.1 zeigt den Zusammenhang der Zahl der Iterationen I und einer geforderten Bitgenauigkeit n:

$$I = n - 2 \tag{4.31}$$

Der resultierende Gesamtfehler $f_{rel,\arctan}(I)$ bei der Berechnung des Arkustangens ergibt sich somit aus der Summe der beiden folgenden Komponenten:

1. Fehler durch die Rotation $f_{rel,\arctan,rot}(I)$
2. Fehler durch die ganzzahlige Darstellung $f_{rel,\arctan,g}(I)$

Der Fehler durch die Rotation beträgt im ungünstigsten Fall den Arkustangens des minimalen Drehwinkels, falls mit der vorletzten Drehung das Ergebnis bereits exakt erzielt wurde. Der Fehler durch die Darstellung ergibt sich aus dem Quotient des Ergebnisbereichs im Intervall von $[0..2\pi[$ und dem Wertebereich W_k. Dieser Quotient entspricht jedoch gerade dem Kehrwert von S_k, so dass insgesamt für den Fehler $f_{\arctan}(I)$ folgt:

$$\begin{aligned} f_{rel,\arctan}(I) &= f_{rel,\arctan,rot}(I) + f_{rel,\arctan,g}(I) \\ &= \tan^{-1}\left(2^{I-1}\right) + \frac{1}{S_k(I)} \end{aligned} \tag{4.32}$$

Tabelle 4.1 zeigt die Fehler der Arkustangensberechnung in Abhängigkeit von der Iterationstiefe I, und dem Skalierungsfaktor S_k. Der Fehler verringert sich bei der Ausnutzung des gesamten Wertebereichs abhängig von der Iterationstiefe um ca. 14 %.

4.2.5 Phasenbild

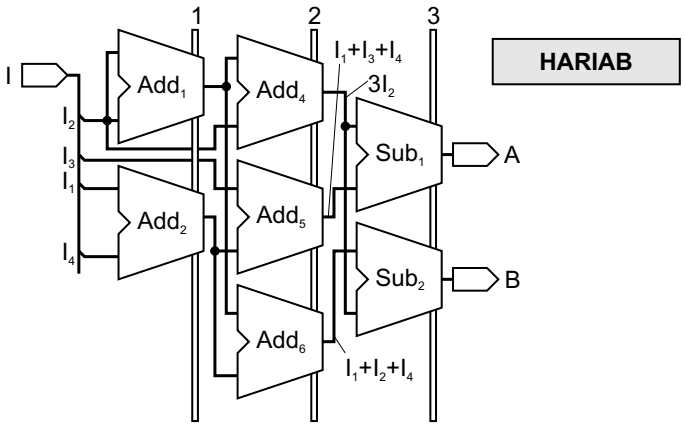


Abbildung 4.4: Berechnung von A und B aus Gleichung 4.4 und 4.5

Zur Berechnung des Phasenbilds aus den Intensitäten zweier Kamerabilder mit unterschiedlichen Wellenlängen λ_1 und λ_2 müssen die bisher vorgestellten Module zur „Hariharan-Pipeline“ zusammengeschaltet werden. Wie bereits erwähnt, erfolgt die Berechnung schrittweise, beginnend mit den Zwischenvariablen w_A, w_B, w_C und w_D nach Gleichung 4.4 und 4.5. Diese Aufgabe übernimmt das Modul *HARIAB* in Abbildung 4.4, das als Eingabe die Intensitäten I_1, I_2, I_3 und I_4 erhält. Abbildung 4.5 zeigt dann die vollständige Zusammenschaltung aller Module. Aus den Zwischenvariablen ergeben sich durch den Addierer Add_1 und Subtrahierer Sub_1 Zähler und Nenner von Gleichung 4.7. Die „CORDIC-Pipeline“ in Form des Moduls *ATAN* berechnet schließlich die Phase $\Delta\phi$.

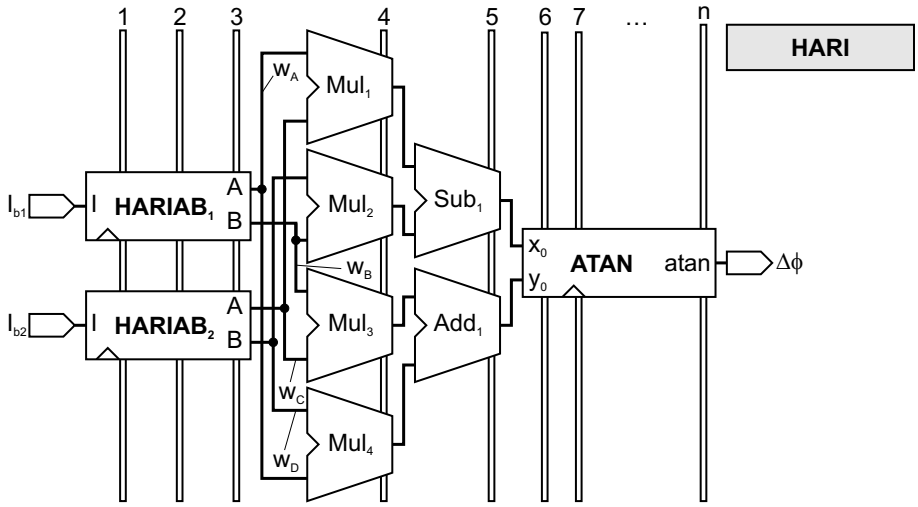


Abbildung 4.5: Gesamte Pipeline zur Berechnung des Phasenbilds nach dem Algorithmus von Hariharan-Schwider

Kapitel 5

Echtzeit Unwrapping

5.1 Grundlegende Algorithmen

Das zweidimensionale Entfaltungsproblem wird in Zusammenhang mit der Specklemesstechnik erstmals 1980 erwähnt [66]. Seitdem haben sich viele Wissenschaftler in einer Vielzahl von Publikationen mit dieser Aufgabe beschäftigt. Es existieren prinzipiell zwei Lösungsansätze, mit deren Grundlagen sich bereits Kapitel 2.5 befasste:

1. Pfadabhängige Methoden
2. Pfadunabhängige Methoden

Dieser Abschnitt beschreibt die prinzipielle Funktionsweise der Algorithmen und bewertet diese vor dem Hintergrund eines industriellen Einsatzes und der Möglichkeit einer Umsetzung in einer spezialisierten Hardware.
Abbildung 5.1 verdeutlicht die Aufgabenstellung mit ihren wesentlichen Detailfragen qualitativ. Quantitative Aussagen zur Qualität des Entfaltungsalgorithmus finden sich in Abschnitt 6.7. Das Beispiel zeigt ein gehämmertes Stück Blech, das eine unregelmäßige Form aufweist, in verschieden Stadien der Bildbearbeitung. Die Messung erfolgt mit zwei unterschiedlichen Wellenlängen und einem räumlichen Phasenschub. Unter Anwendung des Hariharan-Schwider-Algorithmus nach Abschnitt 4.1 entsteht das Phasenbild 5.1.a. Das gefilterte Phasenbild 5.1.b ist das Ergebnis der Anwendung einer sogenannten Pixelwaage auf das Phasenbild. Bei diesem einfachen Filteralgorithmus werden drei benachbarte Pixel der Größe nach sortiert und dem zu filternden Pixel der Wert des zweitgrößten Pixel der Sortierung zugeordnet [67]. Dieses Filterverfahren ist robust gegen Ausreißer. Den Algorithmus zur Entfaltung des Phasenbilds beschreibt der folgende Abschnitt 5.2. Um das Ergebnis visuell mit dem gemessenen Phasenbild vergleichen zu können können, wird das entfaltete Bild 5.1.c mittels eines Medianfilters [67] geglättet und

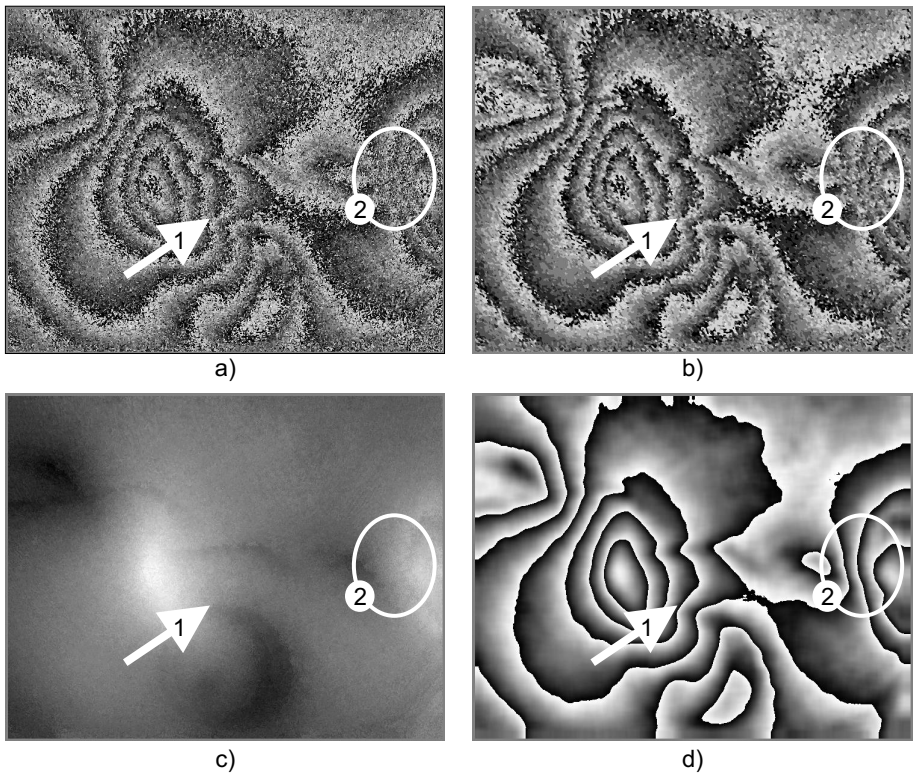


Abbildung 5.1: Kachelung eines Phasenbilds: a) Phasenbild, b) gefiltertes Phasenbild c) dreidimensionale Rekonstruktion des Objektes, d) Wrappingoperator $\mathcal{W}$ angewandt auf die dreidimensionale Rekonstruktion

anschließend durch die Anwendung des Wrappingoperators $\mathcal{W}$ gefaltet, so dass Abbildung 5.1.d entsteht.
Das gemessene Phasenbild 5.1.a wird im Wesentlichen durch folgende Störungen verfälscht [2]:

- Ungenauigkeiten in der Justage des Messaufbaus
- mechanische Schwingungen des Messaufbaus
- Rauschen der Kamera
- Quantisierungsrauschen der AD-Wandler

Die daraus resultierenden Fehler erschweren die Entfaltung des Phasenbilds. Es enstehen Uneindeutigkeiten, die insbesondere bei den pfadabhängigen Methoden zu Fehlern im Ergebnis führen, da in solchen Fällen Residuen auftreten, die bereits Abschnitt 2.5 beschreibt. Dabei verschwindet das Integral, wie es Gleichung 2.22 zeigt, entlang eines geschlossenen Weges nicht. Der Pfeil eins in Abbildung 5.1 zeigt das Beispiel eines solchen unterbrochenen Phasenübergangs, der bei einfachen Entfaltungsalgorithmen sehr leicht zu Fehlern führt. Insbesondere besteht das Problem der Fehlerfortpflanzung, d.h. eine kleine Lücke im Phasenübergang setzt sich über einen ganzen Bereich des Bildes als Fehler fort. Im umrandeten Gebiet zwei befindet sich eine insgesamt verrauschte Stelle, die in diesem Beispiel zu einem gravierenden Fehler führt. Wie der Vergleich der Abbildungen 5.1.a und 5.1.d im Bereich zwei zeigt, geht ein ganzer Phasenübergang verloren, während die Lücke im Phasenübergang das Ergebnis nicht beeinflusst.
Ein wichtiger Vorverarbeitungsschritt besteht in der Filterung der Daten nach den verschiedensten Techniken [21]. Hierbei kommen sowohl klassische Filter, wie beispielsweise Medianfilter oder Gauß'sche Filter [76] als auch speziell auf die Filterung von Phasenbildern zugeschnittene Techniken, wie beispielsweise der Filter von Vikhagen [77], zum Einsatz. Die verschiedenen Filtertechniken spielen zur Beurteilung der Algorithmen im Folgenden keine wesentliche Rolle, zumal die Wahl der geeigneten Filtertechnik im Allgemeinen abhängig ist von der Art des Entfaltungsproblems und sich deshalb nicht generell automatisieren lässt [21].
Die pfadabhängigen Methoden arbeiten sich Pixel für Pixel durch das zu entfaltende Phasenbild. Einfach ausgedrückt geht es stets um die Frage den richtigen Pfad im Bild zu finden, so dass keine Fehler durch korrupte Phasenübergänge oder verrauschte Bereiche enstehen. Dabei kommen verschiedene Techniken zum Einsatz.
Der „Branch-Cut-Algorithmus“ verbindet positive und negative Residuen mit einer Barriere, die der Pfad der Entfaltung nicht kreuzen darf [68, 69, 70, 71, 72, 73, 74, 75]. Die wesentliche Aufgabe bei dieser Klasse von Algorithmen besteht also darin, die gefundenen positiven und negativen Residuen auf die richtige Art und Weise miteinander zu verbinden. In einem Phasenbild können mitunter $n_{bc} = 10^3$ Residuen auftreten, so dass insgesamt $n_{bc}!$ Verbindungsmöglichkeiten bestehen [70]. Da dieses Problem weder eindeutig noch geschlossen lösbar ist, kommen verschiedene Optimierungsverfahren zum Einsatz. Hier sind insbesondere folgende Verfahren zu nennen:

- Nächstliegende Residuen miteinander verbinden [68]
- Balancieren der Residuen [24]
- Simulated Annealing für die Suche nach dem globalen Minimum der Summe der Länge aller Branch-Cuts [70, 71]

- Methoden der Graphentheorie für die Suche nach dem globalen Minimum der Summe der Länge aller Branch-Cuts [72]

Eine weitere Möglichkeit den richtigen Weg im Phasenbild zu finden ist der Einsatz von Qualitätskarten. Diese werden beim Entfalten von Phasenbildern in der Literatur jedoch ausschließlich im Zusammenhang mit Radar-Interferometrie erwähnt und ergeben sich dort aus dem physikalischen Messprinzip, so dass diese Techniken nicht auf die Speckle-Messtechnik übertragbar sind [21].
Bei den verschiedenen Techniken spannender Bäume, geht es darum, einen Pfad im Bild zu suchen der die Gewichte aller Kanten maximiert [78, 79, 80, 81].
Das Entfalten nach Regionen beruht darauf, zunächst nur Bereiche zu entfalten, die an den Rändern der Phasensprünge enden. Dadurch wird eine Fehlerfortpflanzung verhindert. Zum Entfalten der Regionen, sowie zum Erkennen der Kanten der Regionen stehen alle Entfaltungsalgorithmen zur Verfügung.
Die Kachelung der Phasenbilder erfolgt in festgelegten Größen. Diese Kacheln werden dann wieder nach den oben genannten Algorithmen entfaltet. Die Kacheln müssen nun zum Gesamtbild zusammengesetzt werden. Die Methoden hierzu entsprechen den Methoden des Entfaltens auf Pixelebene.
Bei den Pfadunabhängigen Methoden lässt sich generell sagen, dass die Rechenzeiten bei bis zu mehreren Stunden und liegen können. Ferner kann der Speicherbedarf je nach Algorithmus auf einige Gigabyte anwachsen. Ferner wächst sowohl die Rechenzeit als auch der Speicherbedarf stets überproprotional mit der Zahl der zu verarbeitenden Pixel.
Zusammenfassend lässt sich feststellen, dass keiner der bekannten Algorithmen unmittelbar als Echtzeitalgorithmus geeignet ist. Die pfadabhängigen Methoden sind zwar sehr schnell, benötigen aber vorverarbeitete Phasenbilder. Diese Vorverarbeitung in Form einer Filterung der Bilder ist nicht generell automatisierbar, da je nach Art und Qualität des Bildes die Filterparameter manuell angepasst werden müssen. Die pfadunabhängigen Algorithmen erzielen zwar hervorragende Ergebnisse, scheiden jedoch wegen ihres großen Speicherbedarfs und ihrer langen Laufzeit aus.
Daraus ergibt sich die Konsequenz, einen neuen Algorithmus entwickeln zu müssen, der sich teilweise auf bereits vorhandene Ideen stützt. Diesen Algorithmus, sowie seine Umsetzung in Hardware beschreiben dei folgenden Abschnitte 5.2 und 5.3.

5.2 Echtzeitalgorithmus

Die Anforderungen an den Echtzeitalgorithmus leiten sich aus den grundlegenden Gedanken aus Abschnitt 5.1 ab:

- Zerlegbarkeit in Teilprobleme zur Parallelisierung und zum Pipelining
- Verhindern der Fehlerfortpflanzung
- Definierte Laufzeit
- Keine Datenvorverarbeitung durch Filter
- Berechnung einer Qualitätskarte
- Robustheit gegen Rauschen und Fehler

Die Grundidee des folgenden Algorithmus beruht darauf, das gesamte Bild in ausreichend kleine, sich überlappende Kacheln zu zerlegen. Diese Kacheln werden dann unabhängig voneinander nach demselben Verfahren entfaltet. Die Ausgangswerte für die unabhängigen Optimierungen stellen jeweils die Phasenwerte $\phi(P_i)$ des Phasenbildes dar. Abbildung 5.2 zeigt die prinzipielle Idee dieser Art der

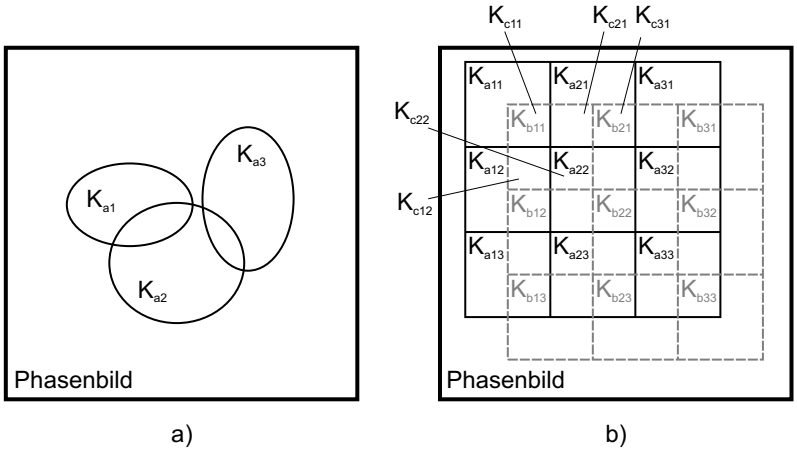


Abbildung 5.2: Kachelung eines Phasenbildes: a) Allgemeine Betrachtung b) Kachelung in der Umsetzung

Kachelung. Die einzelnen Kacheln K_{a1}, K_{a2} und K_{a3} werden unabhängig voneinander, ausgehend von den Phasenwerten $\phi(P_i)$ für alle $P_i \in K_{a1}, K_{a2}, K_{a3}$, entfaltet. Der Algorithmus nach der die Entfaltung abläuft spielt dabei keine Rolle, muss jedoch für die beiden sich überlappenden Kacheln identisch sein. Neben den bereits in Abschnitt 5.1 diskutierten Algorithmen beschreibt Abschnitt 5.2.1 eine neue und schnelle Methode zur Entfaltung von Kacheln. Die Betrachtung der überlappenden Bereiche $K_{a1} \cap K_{a2}$ sowie $K_{a2} \cap K_{a3}$ eröffnet nun folgende Möglichkeiten:

1. Beurteilung der Qualität der überlappenden Bereiche durch die Berechnung der Korrelation (5.2.2). Dies führt zu einem Qualitätsmaß Q_k der Überlappungsbereiche.

2. Zusammenfügen benachbarter Kacheln zu einer geschlossenen Einheit durch die Berechnung des Offsets O_k im Überlappungsbereich.

Abbildung 5.2 zeigt zur Verdeutlichung die Kachelung eines Phasenbilds in der Umsetzung. Im Folgenden wird die Optimierung K_a als Kachelebene und die Optimierung K_b als Klebeebene bezeichnet. Dabei überlappen sich die unabhängigen Bereiche, außer am Rand des Bildes, vollständig. So wird beispielsweise Kachel K_{a22} von den Kacheln K_{b11}, K_{b12}, K_{b21} und K_{b22} überlappt. Die Kacheln K_c ergeben sich aus K_a und K_b:

$$K_c(x_c, y_c) = K_a|_{K_c} \cap K_b|_{K_c} = K_a(x_a, y_a) \cap K_b(x_b, y_b) \tag{5.1}$$

Wobei sich die Indizes aus folgender Vorschrift ergeben:

$$x_a = \text{floor}\left[\frac{x_c}{2}\right] + 1 \tag{5.2}$$

$$y_a = \text{floor}\left[\frac{y_c}{2}\right] + 1 \tag{5.3}$$

$$x_b = \text{floor}\left[\frac{x_c}{2}\right] \tag{5.4}$$

$$y_b = \text{floor}\left[\frac{y_c}{2}\right] \tag{5.5}$$

Der nächste Schritt des Algorithmus prüft benachbarte Kacheln auf Konsistenz, wie dies Abschnitt 5.2.3 beschreibt. Die Konsistenz ergibt sich durch die Bildung des Ringintegrals über die Abstände $O_k(i, j)$ der Kachelebenen K_a und K_b im Bereich der Kacheln $K_c(i, j)$, welches im Falle der Konsistenz null ergeben muss.

$$\oint_{K_c(i,j)} O_k(i, j) = 0 \tag{5.6}$$

Dabei ist $O_k(i, j)$ definiert als:

$$O_k(i, j) = K_a - K_b \tag{5.7}$$

Ist die Bedingung nach Gleichung 5.6 nicht erfüllt, so werden diese Überlappungsbereiche bei der späteren Entfaltung zunächst nicht berücksichtigt.
Abschnitt 5.2.4 erläutert den nächsten Schritt des Algorithmus, der Lücken innerhalb der inkonsistenten Bereiche schließt. Dieser Teil des Algorithmus ist empirisch aus Erfahrungen mit realen Messungen an komplexen Objekten entstanden. Er dient im Wesentlichen dazu, eine Fehlerfortpflanzung zu verhindern. Die

gefundenen Lücken scheiden als Übergänge beim Zusammensetzen der Kacheln aus.
Durch die Entfaltung nach Abschnitt 5.2.5 entstehen großflächige zusammenhängende Bereiche. Die folgenden Schritte arbeiten nur noch auf diesen großen Bereichen, was zu einer erheblichen Datenreduktion führt, welche die Verarbeitungsgeschwindigkeit erhöht und den Speicherbedarf verringert.
Durch die Berechnung verschiedener Qualitätsmaße nach Abschnitt 5.2.6 entsteht zum Einen eine Qualitätskarte zur Beurteilung des Messergebnisses. Zum Anderen spielen die Kennzahlen zur Beurteilung der Beziehung benachbarter Bereiche beim späteren Zusammensetzen des Gesamtbildes eine entscheidende Rolle.
Der letzte Schritt des Algorithmus setzt die großflächigen Bereiche zum Gesamtbild zusammen. Die Reihenfolge beim Puzzeln der Bereiche ergibt sich aus dem Produkt der verschiedenen Kennzahlen, die aus vorangegangen Arbeitsschritten gewonnen wurden.
Die folgenden Kapitel erläutern die verschiedenen Schritte des Algorithmus zur Entfaltung des Gesamtbildes aus dem Phasenbild und untersuchen die Umsetzbarkeit in einer spezialisierten Hardware.
Als Beispiel dient in den folgenden Kapiteln die Messung nach Abbildung 5.3. Das Phasenbild zeigt eine Delle in einem Alublock. Die Kameraauflösung beträgt dabei 1280 Pixel horizontal und 1024 Pixel vertikal. Anhand dieses Beispiels lassen sich die Schritte des Entfaltungsalgorithmus qualitativ zeigen und analysieren.

Abbildung 5.3: Phasenbild der Delle in einem Alublock

5.2.1 Kachelentfaltung

Eine wichtige Frage bei der Suche nach einem geeigneten Algorithmus zur Kachelentfaltung stellt die Konvergenz und das damit verbundene Abbruchkriterium dar. Bei der Wahl ausreichend kleiner Kacheln kann die Kachelfläche K_a als lokal eben angenommen werden. Diese Einschränkung quantifiziert Gleichung 2.15. Ein mögliches Qualitätskriterium o ergibt sich durch die Summe der Fehlerquadrate der kontinuierlichen Phasenwerte $\psi(P_i)$ in den Punkten $P_i \in K_a$ bezüglich der optimalen ebenen Fläche mit Höhe c, die insgesamt n Punkte enthält:

$$o = \|f\|_2^2 = \sum_{P_i \in K_a} (c - \psi(P_i))^2 \tag{5.8}$$

mit Gleichung 2.15

$$o = \sum_{P_i \in K_a} (c - (\phi(P_i) + 2\pi k(P_i)))^2 \tag{5.9}$$

Als Lösung für c ergibt sich damit der Mittelwert [48]:

$$c = \frac{\sum\limits_{P_i \in K_a} (\phi(P_i) + 2\pi k(P_i))}{n} \tag{5.10}$$

Die Aufgabe der Optimierung besteht nun darin, alle $k(P_i)$ so zu bestimmen, dass o minimal wird. Dazu wird der Faltungskoeffizient k in genau einem Punkt P_0 so zu k^* variiert so dass gilt:

$$|c - (\phi(P_0) + 2\pi k(P_0))| > |c - (\phi(P_0) + 2\pi k^*(P_0))| \tag{5.11}$$

mit

$$\begin{aligned} k^*(P_i) &= k(P_i) \quad \forall \quad P_i \neq P_0 \\ k^*(P_i) &\neq k(P_i) \quad \text{für} \quad P_i = P_0 \end{aligned} \tag{5.12}$$

Somit verändert sich der Mittelwert in folgender Weise:

$$\begin{aligned} c^* &= \frac{\sum\limits_{P_i \in K_a} (\phi(P_i) + 2\pi k(P_i))}{n} - \frac{\phi(P_0) + 2\pi k(P_0)}{n} + \frac{\phi(P_0) + 2\pi k^*(P_0)}{n} \\ &= \frac{\sum\limits_{P_i \in K_a} (\phi(P_i) + 2\pi k(P_i))}{n} + \frac{2\pi}{n}(k^*(P_0) - k(P_0)) \\ &= c + \frac{2\pi}{n}(k^*(P_0) - k(P_0)) \end{aligned} \tag{5.13}$$

Das Qualitätskriterium o verändert sich damit zu o^* wobei gilt:

$$o^* = \sum_{P_i \in K_a} (c^* - (\phi(P_i) + 2\pi k(P_i)))^2 \tag{5.14}$$

Dies ergibt durch Einsetzen von Gleichung 5.13

$$\begin{aligned} o^* &= \sum_{P_i \in K_a} \left(c - \phi(P_i) - 2\pi k^*(P_i) + \frac{2\pi}{n}(k^*(P_0) - k(P_0)) \right)^2 \\ &= \sum_{P_i \in K_a} (c - \phi(P_i) - 2\pi k^*(P_i))^2 + \qquad (5.15) \\ &\quad \sum_{P_i \in K_a} \frac{4\pi}{n}(k^*(P_0) - k(P_0))(c - \phi(P_i) - 2\pi k^*(P_i)) + \qquad (5.16) \\ &\quad \sum_{P_i \in K_a} \frac{4\pi^2}{n^2}(k^*(P_0) - k(P_0))^2 \qquad (5.17) \end{aligned}$$

Im Folgenden werden die Einzelterme 5.15, 5.16 und 5.17 vereinfacht. Zunächst ergibt sich für Term t_1 mit Gleichung 5.15 unter Berücksichtigung der Voraussetzung nach 5.12:

$$\begin{aligned} t_1 &= \sum_{P_i \in K_a} (c - \phi(P_i) - 2\pi k^*(P_i))^2 \\ &= \sum_{P_i \in K_a} (c - \phi(P_i) - 2\pi k(P_i))^2 \\ &\quad -(c - \phi(P_0) - 2\pi k(P_i))^2 + (c - \phi(P_i) - 2\pi k^*(P_0))^2 \\ &= o - (c - \phi(P_0) - 2\pi k(P_i))^2 + (c - \phi(P_i) - 2\pi k^*(P_0))^2 \qquad (5.18) \end{aligned}$$

Analog erfolgt die Umformung für Term t_2 nach Gleichung 5.16, wobei zu beachten ist, dass die Summe über die Differenzen der Phasenwerte $\phi(P_i)$ zum Mittelwert c stets null ergeben [48].

$$\begin{aligned} t_2 &= \sum_{P_i \in K_a} \frac{4\pi}{n}(k^*(P_0) - k(P_0))(c - \phi(P_i) - 2\pi k^*(P_i)) \\ &= \frac{4\pi}{n}(k^*(P_0) - k(P_0)) \sum_{P_i \in K_a} (c - \phi(P_i) - 2\pi k(P_i)) + \\ &\quad \frac{4\pi}{n}(k^*(P_0) - k(P_0)) \left[(c - \phi(P_0) - 2\pi k^*(P_0)) - (c - \phi(P_0) - 2\pi k(P_0)) \right] \\ &= -\frac{8\pi^2}{n}(k^*(P_0) - k(P_0))^2 \qquad (5.19) \end{aligned}$$

Term t_3 ist unabhängig von den Punkten P_i und vereinfacht sich damit zu:

$$t_3 = \sum_{P_i \in K_a} \frac{4\pi^2}{n^2}(k^*(P_0) - k(P_0))^2$$

$$= \frac{4\pi^2}{n}(k^*(P_0) - k(P_0))^2 \tag{5.20}$$

Insgesamt ergibt sich damit durch die Addition der Terme t_1, t_2 und t_3 für das Qualitätskriterium o^*:

$$\begin{aligned} o^* &= t_1 + t_2 + t_3 \\ &= o - \\ &\quad (c - \phi(P_0) - 2\pi k(P_0))^2 + (c - \phi(P_0) - 2\pi k^*(P_0))^2 - \qquad (5.21) \\ &\quad \frac{4\pi^2}{n}(k^*(P_0) - k(P_0))^2 \qquad (5.22) \end{aligned}$$

Unter Verwendung der Voraussetzung nach 5.11 folgt, dass

$$(c - \phi(P_0) - 2\pi k(P_0))^2 > (c - \phi(P_0) - 2\pi k^*(P_0))^2 \tag{5.23}$$

ferner gilt stets

$$n > 0 \tag{5.24}$$

Aus den Gleichungen 5.21, 5.22, 5.23 und 5.24 ergibt sich dann, dass das Optimierungskriterium o^* stets kleiner ist als o:

$$o^* < o \tag{5.25}$$

Der Abstand zweier beliebiger Faltungskoeffizienten $k(P_1)$ und $k(P_2)$ lässt sich noch einschränken. Es gilt für die beiden Punkte $P_1 \in K_a$ und $P_2 \in K_a$

$$|(\phi(P_2) + 2\pi k(P_2)) - (\phi(P_1) + 2\pi k(P_1))| \leq 2\pi \tag{5.26}$$

da ansonsten ein $k^*(P_2)$ existieren würde, das Gleichung 5.11 erfüllt. Dies lässt sich umformen [48] zu:

$$|\phi(P_2) - \phi(P_1)| + 2\pi|k(P_2)) - k(P_1)| \leq 2\pi \tag{5.27}$$

Da nach Gleichung 2.12 stets

$$|\phi(P_2) - \phi(P_1)| < 2\pi \tag{5.28}$$

gilt, ergibt sich für $|k(P_2) - k(P_1)|$ folgende Einschränkung:

$$|k(P_2) - k(P_1)| \leq 1 \tag{5.29}$$

Ferner lässt sich wegen 2.12 und 5.25 feststellen, dass für $k(x, y)$ entweder

$$k(x, y) = 0 \;\; \forall \; x_i \in [x_s; x_e], \; y_i \in [y_s; y_e] \tag{5.30}$$

oder

$$\begin{aligned} &\exists\, k(x_0, y_0) \neq 0 \;\text{ mit }\; k(x_0, y_0) \neq k(x_i, y_i) \\ \forall\; x_i \in [x_s; x_e],\; y_i \in [y_s; y_e] \;&\text{ und }\; x_i \neq x_0,\; y_i \neq y_0 \end{aligned} \tag{5.31}$$

gilt.

Zusammenfassend ergeben sich aus den vorangegangen Betrachtungen folgende Schlussfolgerungen:

1. Eine fortwährende Änderung des Faltungsparameters $k(P_0)$ konvergiert wegen Gleichung 5.25

2. Das Abbruchkriterium ist erreicht, wenn genau n Iterationen durchgeführt wurden, ohne dass sich eine Änderung der Faltungsparameter $k(P_i)$ ergeben hätte, da dann kein o^* mehr gefunden werden kann, welches Gleichung 5.25 erfüllt.

3. Für den Wertebereich der Faltungskoeffizienten $k(P_i)$ ergibt sich aus den Gleichungen 5.29, 5.30 und 5.31

$$k(P_i) \in \{-1, 0, 1\} \tag{5.32}$$

4. Wegen Gleichung 5.29 und Bedingung 5.31 und 5.32 können sich alle Faltungsparameter $k(P_i) \in K_a$ maximal einmal ändern. Da auf jeden Fall ein $k(P_0) = 0$ sein muss, ist nach maximal

$$s_k = n - 1 \tag{5.33}$$

   Änderungen die bestmögliche Entfaltung erreicht.

5. Wegen Gleichung 5.29 können sich alle Faltungsparameter $k(P_i) \in K_a$ nur in eine Richtung ändern. Die Richtung ergibt sich durch die Betrachtung der Höhe der optimalen Fläche c:

$$k(P_i) \begin{cases} > 0 & \text{für } c > \frac{\pi}{2} \\ = 0 & \text{für } c = \frac{\pi}{2} \\ < 0 & \text{für } c < \frac{\pi}{2} \end{cases} \tag{5.34}$$

Die Prämisse dieses Entfaltungsverfahrens beruht auf der Annahme, dass die Kachel lokal eben ist. Diese Einschränkung lässt sich mit Gleichung 5.32 präzisieren. Danach gilt für den Abstand der Werte der kontinuierlichen Phase $\psi(P_i)$ innerhalb der Kachelfläche K_a:

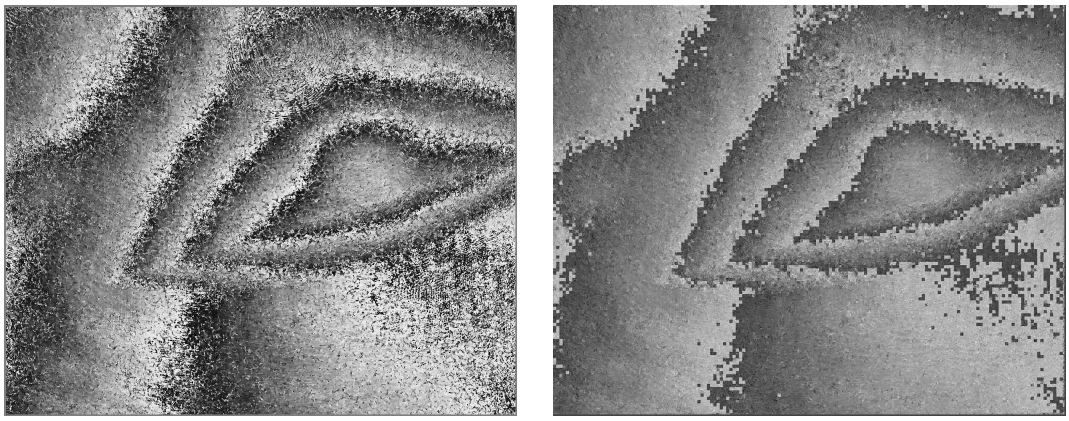
$$|\psi(P_i) - \psi(P_j)| < 2\pi \quad \forall \quad P_i \in K_a \tag{5.35}$$

Aus diesem Ergebnis lässt sich unter Berücksichtigung der Startvoraussetzung nach Gleichung 5.11 folgender Algorithmus zur Kacheloptimierung ableiten:

1. Berechne den Mittelwert über alle Punkte $P_i \in K_a$
2. Nimm einen Punkt P_0 aus der Menge aller Punkte P_i und optimiere den Punkt nach Gleichung 5.11:

$$k^*(P_i) = \begin{cases} 1 & \text{für } \phi(P_i) - c \leq -\frac{\pi}{2} \\ 0 & \text{für } -\frac{\pi}{2} < \phi(P_i) - c < \frac{\pi}{2} \\ -1 & \text{für } \phi(P_i) - c \geq \frac{\pi}{2} \end{cases} \tag{5.36}$$

3. Breche ab, falls kein Punkt zu optimieren war
4. Fahre fort mit Schritt 1



a) b)

Abbildung 5.4: Phasenbild der Delle in einem Alublock

Das Beispiel nach Abbildung 5.6 zeigt das Messbeispiel aus dem einführenden Abschnitt 5.2. Die Abbildung 5.3.b zeigt das Phasenbild nach der Kachelentfaltung.

5.2.2 Kachelkorrelation

Die Kachelkorrelation dient dazu die Ähnlichkeit der unabhängigen Entfaltungen zweier Kacheln zu bestimmen. Abbildung 5.5 verdeutlicht die Aufgabenstellung. Die Kacheln K_a und K_b durchlaufen jeweils eine unabhängige Entfaltung, wie

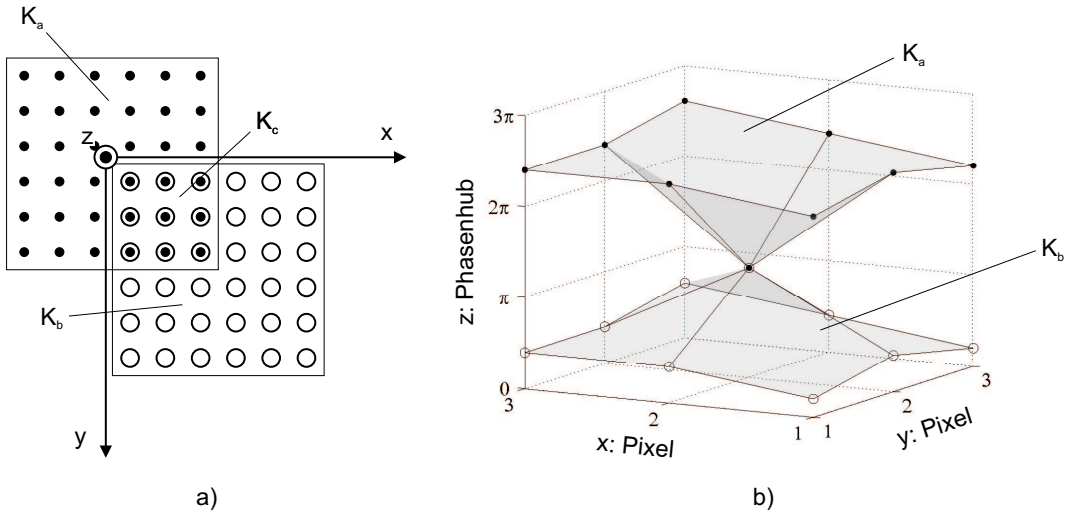


Abbildung 5.5: Beispiel für die Ähnlichkeit zweier überlappender Kacheln: a) Definition der Kacheln K_a, K_b und des Überlappungsbereichs K_c b) dreidimensionale Ansicht des Ergebnisses der Entfaltung der Bereiche K_a und K_b im Überlappungsbereich K_c

sie der vorangegangene Abschnitt 5.2.1 beschreibt. Der Überlappungsbereich K_c ergibt sich aus

$$K_c = K_a \cap K_b \tag{5.37}$$

Die Zahl der Punkte innerhalb des Bereich K_c ist dabei n_{K_c}. Die Phasenwerte ϕ des Beispiels in Abbildung 5.5 der Kacheln K_a und K_b im Bereich K_c sind:

$$K_a^{\phi}|_{K_c} = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 6 & 2 \\ 2 & 2 & 3 \end{pmatrix} \frac{2\pi}{10};\; K_b^{\phi}|_{K_c} = \begin{pmatrix} 11 & 12 & 11 \\ 12 & 6 & 12 \\ 12 & 12 & 13 \end{pmatrix} \frac{2\pi}{10} \tag{5.38}$$

Wie leicht per Augenschein zu erkennen ist, findet sich die größte Übereinstimmung der Kacheln genau dann, wenn von allen Phasenwerten ϕ im Bereich K_b der Phasenhub von 2π subtrahiert wird.

Allgemein formuliert besteht die Aufgabe darin, die Ähnlichkeit zweier geometrischer Objekte, in diesem Fall zweier Flächen K_a und K_b zu bestimmen. Um die Ähnlichkeit zweier Objekte zu quantifizieren, besteht die Möglichkeit diese als Objekte der Metrik $\delta(P_1, P_2)$ aufzufassen [48] und den Grad ihrer Ähnlichkeit durch ihren Abstand zu definieren [61, 62]. Der Abstand $\delta^*(P_1, X_u)$ zwischen einem Punkt P_1 und einem Unterraum $X_u \subseteq X$ definiert sich durch [61]:

$$\delta^*(P_1, X_u) = \inf_{P_i \in X_u} [\delta(P_1, P_i)] \tag{5.39}$$

Der Abstand δ_h^* zwischen den beiden Unterräumen $X_u \subseteq X$ und $X_v \subseteq X$ berechnet sich zu [62]:

$$\delta_h^*(X_u, X_v) = \sup_{P_i \in X_u} [\delta^*(P_i, X_v)] \tag{5.40}$$

Daraus lässt sich dann der Hausdorff-Abstand angeben [62]:

$$\delta_h(X_u, X_v) = \sup [(\delta_h^*(X_u, X_v), \delta_h^*(X_v, X_u))] \tag{5.41}$$

Ein Maß für die Ähnlichkeit Q_k^* zweier Objekte ergibt sich durch die Minimierung des Hausdorff-Abstandes zwischen dem Unterraum X_u und einer Transformation T aus der Menge aller möglicher Transformationen $\mathcal{T}$, die sich auf den Unterraum X_v anwenden lassen [62]:

$$Q_k^* = \inf_{T \in \mathcal{T}} [\delta_h(X_u, T(X_v))] \tag{5.42}$$

Die Algorithmen zur Ähnlichkeitsbestimmung sind vielfältig [61, 62, 63] und sehr stark abhängig von der Wahl des Abstandsmaßes δ und der Menge der Transformationen $\mathcal{T}$. Als Abstandsmaße stehen beispielsweise die euklidische Metrik, Maximumsmetrik oder auch die Metrik über die Summe der Beträge der Differenzen zur Verfügung [64, 65]. Bei der vorliegenden Aufgabe können sich die einzelnen Phasenwerte $\psi(P_i)$ durch unterschiedliche Ergebnisse der unabhängigen Entfaltungen der Bereich K_a und K_b, nach Gleichung 2.15 lediglich um ganzzahlige Vielfache von 2π unterscheiden. Deshalb eignet sich insbesondere die diskrete Metrik [64, 65]:

$$\delta_d(\psi_{K_a}(P_i), \psi_{K_b}(P_i)) = \begin{cases} 1 & \text{für } \psi_{K_a}(P_i) \neq \psi_{K_b}(P_i) \\ 0 & \text{für } \psi_{K_a}(P_i) = \psi_{K_b}(P_i) \end{cases} \tag{5.43}$$

K_a, K_b und K_c sind dabei Elemente des Phasenraums X_p:

$$K_a \in X_P,\ K_b \in X_P,\ K_c \in X_P \tag{5.44}$$

Die Menge aller möglichen Transformationen $\mathcal{T}$ im Phasenraum X_P beschränkt sich auf eine einzige Transformation T_k, nämlich die Verschiebung der Ebene um ganzzahlige Vielfache von 2π nach Gleichung 2.15:

$$T_{k_o}(X_P) = X_P(P_i) + 2\pi k_o \quad \forall\ P_i \in X_P, \quad k \in \mathbb{Z} \tag{5.45}$$

Die Anwendung des Hausdorff-Abstandes auf das Abstandsmaß δ_d sowie die Transformation T_k beantwortet jedoch nur die Frage, ob die beiden Flächen vollständig übereinstimmen oder nicht:

$$\delta_d(\psi_{K_a}(P_i), \psi_{K_b}(P_i)) = \begin{cases} 0 & \text{für } \psi_{K_a}(P_i) = \psi_{K_b}(P_i) \\ 1 & \text{für } \psi_{K_a}(P_i) \neq \psi_{K_b}(P_i) \end{cases} \tag{5.46}$$

Für die Abstände der Unterräume K_a und K_b im Überlappungsbereich K_c ergibt sich aus den Gleichungen 5.39, 5.40, 5.41 und 5.43:

$$\delta^*(K_a, T_k(K_b))|_{K_c} = \sup_{P_i \in K_c} \left[\inf_{P_j \in K_c} \left[\delta_b(\psi_{K_a}(P_i), \psi_{K_b}(P_j) + 2\pi k_o) \right] \right] \tag{5.47}$$

$$\delta^*(T_k(K_b), K_a)|_{K_c} = \sup_{P_i \in K_c} \left[\inf_{P_j \in K_c} \left[\delta_b(\psi_{K_a}(P_i) + 2\pi k_o, \psi_{K_b}(P_j)) \right] \right] \tag{5.48}$$

Da aber die Transformation T_k lediglich den Abstand der beiden Flächen zueinander variiert, lässt sich zu jeder Verschiebung k_{o1} eine Verschiebung k_{o2} finden, so dass gilt:

$$\delta_b(\psi_{K_a}(P_i), \psi_{K_b}(P_j) + 2\pi k_{o1}) = \delta_b(\psi_{K_a}(P_i) + 2\pi k_{o2}, \psi_{K_b}(P_j)) \tag{5.49}$$

Somit folgt sofort

$$\delta^*(K_a, T_k(K_b))|_{K_c} = \delta^*(T_k(K_b), K_a)|_{K_c} \tag{5.50}$$

Da die Flächen niemals lateral gegeneinander verschoben sind, reicht es aus, die Phasenwerte ϕ_{K_a} und ϕ_{K_b} nur an den Punkten $P_i \in K_c$ zu vergleichen. Der Kreuzweise Vergleich von Phasenwerten in unterschiedlichen Punkten entfällt. Aus Gleichung 5.42 ergibt sich somit für das Ähnlichkeitsmaß Q_k^*:

$$\begin{aligned} Q_k^*(\delta_b, T_k, K_a, K_b)|_{K_c} &= \inf_{k_o \in \mathbb{Z}} \left[\sup_{P_i \in K_c} \left[\inf_{P_i} \left[\delta_b(\psi_{K_a}(P_i), \psi_{K_b}(P_j) + 2\pi k) \right] \right] \right] \\ &= \begin{cases} 0 & \text{wenn} \quad \forall\ P_i, P_j \in K_c \ \text{ gilt} \\ & \qquad \psi_{K_a}(P_i) = \psi_{K_b}(P_i) \\ 1 & \text{sonst} \end{cases} \end{aligned} \tag{5.51}$$

Das heißt bei vollkommener Übereinstimmung der entfalteten Flächen K_a und K_b ergibt sich der Wert null, in allen anderen Fällen der Wert eins.

Diese binäre Entscheidung, ob die Flächen in ihrer Form übereinstimmen oder nicht, reicht als Qualitätsmaß jedoch nicht aus. Die Kacheln K_a und K_b überlagern sich nur in einer endlichen Zahl von Punkten K_c. Außerdem lassen sich die Punkte $P_i \in K_c$ durch die Transformation T_k exakt ineinander transformieren. Dies ermöglicht die Definition eines Qualitätsmaßes Q_k, so dass dieses die größtmögliche Übereinstimmung der beiden Flächen K_a und K_b als die Zahl der exakt übereinander liegenden Punkte ausdrückt.

$$Q_k(\delta_b, T_k, K_a, K_b)|_{K_c} = \sup_{k_o \in \mathbb{Z}} \left[\sum_{P_i \in K_c} (1 - \delta_b(\psi(K_a)(P_i), \psi(K_b)(P_i) + 2\pi k)) \right] \tag{5.52}$$

Dieses Qualitätsmaß wird im Folgenden als Kachelkorrelation Q_k bezeichnet. Unter der Berücksichtigung, dass die Faltungskoeffizienten $k(P_i)$ nach Gleichung 5.32 Elemente der Menge $\{0, 1, 2\}$ sind und sich außerdem die Faltungskoeffizienten nach Gleichung 5.34 nur in eine Richtung ändern dürfen, lässt sich die Menge der zu untersuchenden Transformationen T_{k_o} einschränken.

$$k_o = \begin{cases} -1 & \text{für } c_{K_a} < c_{K_b} \\ 0 & \text{für } \operatorname{sign}\left[c_{K_a} - \frac{\pi}{2}\right] = \operatorname{sign}\left[c_{K_b} - \frac{\pi}{2}\right] \\ 1 & \text{für } c_{K_a} > c_{K_b} \end{cases} \tag{5.53}$$

Dabei sind c_{K_a} und c_{K_b} die optimalen Flächen der unabhängigen Entfaltungen der Kacheln K_a und K_b. Insbesondere Gleichung 5.34 ermöglicht die direkte Berechnung Kachelkorrelation Q_k. Ausgangspunkt dafür ist die Differenz Δc der Mittelwerte der Flächen K_a und K_b im Überlappungsbereich K_c, für die nach Gleichung 5.10 gilt:

$$\Delta c = c_{K_b}|_{K_c} - c_{K_b}|_{K_c} = \frac{2\pi}{n} \sum_{P_i \in K_c} (k_b(P_i) - k_a(P_i)) \tag{5.54}$$

Aus Gleichung 5.34 ergibt sich, dass das Vorzeichen der Differenz $k_b(P_i) - k_a(P_i)$ für alle $P_i \in K_c$ identisch ist. Damit ergibt sich für das Qualitätsmaß Q_k:

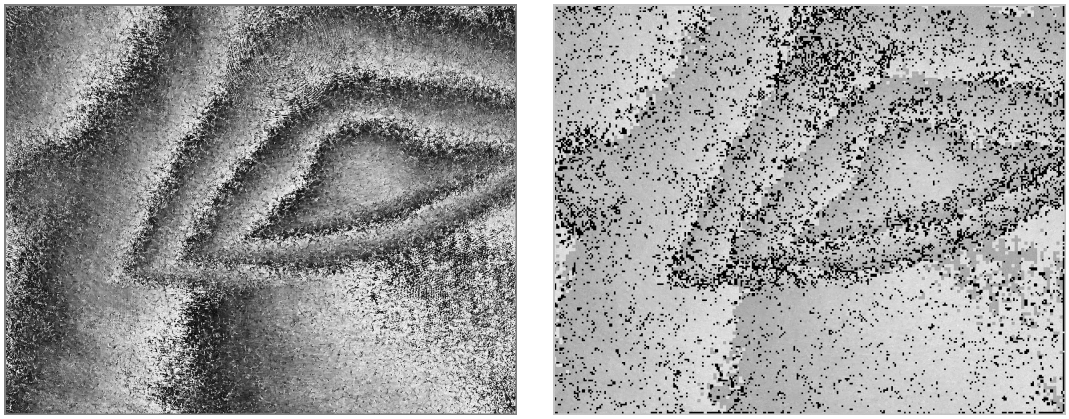
$$Q_k = \sup \left[n\frac{|\Delta c|}{2\pi}; n\left(1 - \frac{|\Delta c|}{2\pi}\right) \right] \tag{5.55}$$

Das Qualitätsmaß Q_k beschreibt die Qualität einer entfalteten Kachel K_c durch den Vergleich zweier unabhängig entfalteter Kacheln K_a und K_b, deren Überlappungsbereich K_c entspricht. Mit Hilfe der Kachelkorrelation Q_k lässt sich der Prozess der Entfaltung steuern, indem Bereiche zunächst ausgenommen und somit Fehler vermieden werden. Dies entspricht der Idee des Einsatzes von Qualitätskarten zum Entfalten der Phasenbilder nach Abschnitt 5.2.1, mit dem Unterschied, dass

die Kachelkorrelation sich ausschließlich aus dem Phasenbild, ohne Kenntnis des physikalischen Messprinzips, ergibt. Weiterhin kann mit Hilfe von Q_k die Qualität des entfalteten Bildes örtlich aufgelöst werden.
In den folgenden Kapiteln ist häufig lediglich die Frage von belang, ob eine Kachel maximale Korrelation aufweist oder nicht. Dies erfordert die Definition des folgenden Qualitätsmaßes Q_g:

$$Q_g(K_c) = \begin{cases} 1 & \text{für } Q_g(K_c) = n(K_c) \\ 0 & \text{für } Q_g(K_c) < n(K_c) \end{cases} \tag{5.56}$$

Im Beispiel zu Beginn dieses Kapitels ergibt sich $Q_k = 8$ und mit $n(K_c) = 9$ ergibt sich für $Q_g = 0$.
Das Beispiel nach Abbildung 5.6 zeigt das Messbeispiel aus dem einführenden



a) b)

Abbildung 5.6: a) Phasenbild, b) Pixel für die gilt $Q_k(K_c) \neq n(K_c)$ geschwärzt

Abschnitt 5.2. In der Abbildung 5.6.b sind alle Kacheln geschwärzt, die nicht maximal korrelieren, also $Q_k(K_c) \neq n(K_c)$.

5.2.3 Kachelkonsistenz

Das Ziel der Suche nach inkonsistenten Kacheln ist es, Residuen im Phasenbild zu finden, um dadurch Fehler bei der Entfaltung zu vermeiden. Im Gegensatz zu den Residuen, wie sie Abschnitt 2.5 auf Pixelebene definiert, sucht dieser Teil des Algorithmus nach Residuen auf Kachelebene. Dabei werden nur die kleinsten möglichen Pfade untersucht, nämlich das Integral über vier benachbarte Kacheln. Abbildung 5.7 zeigt exemplarisch die Bestimmung der Konsistenz der Ka-

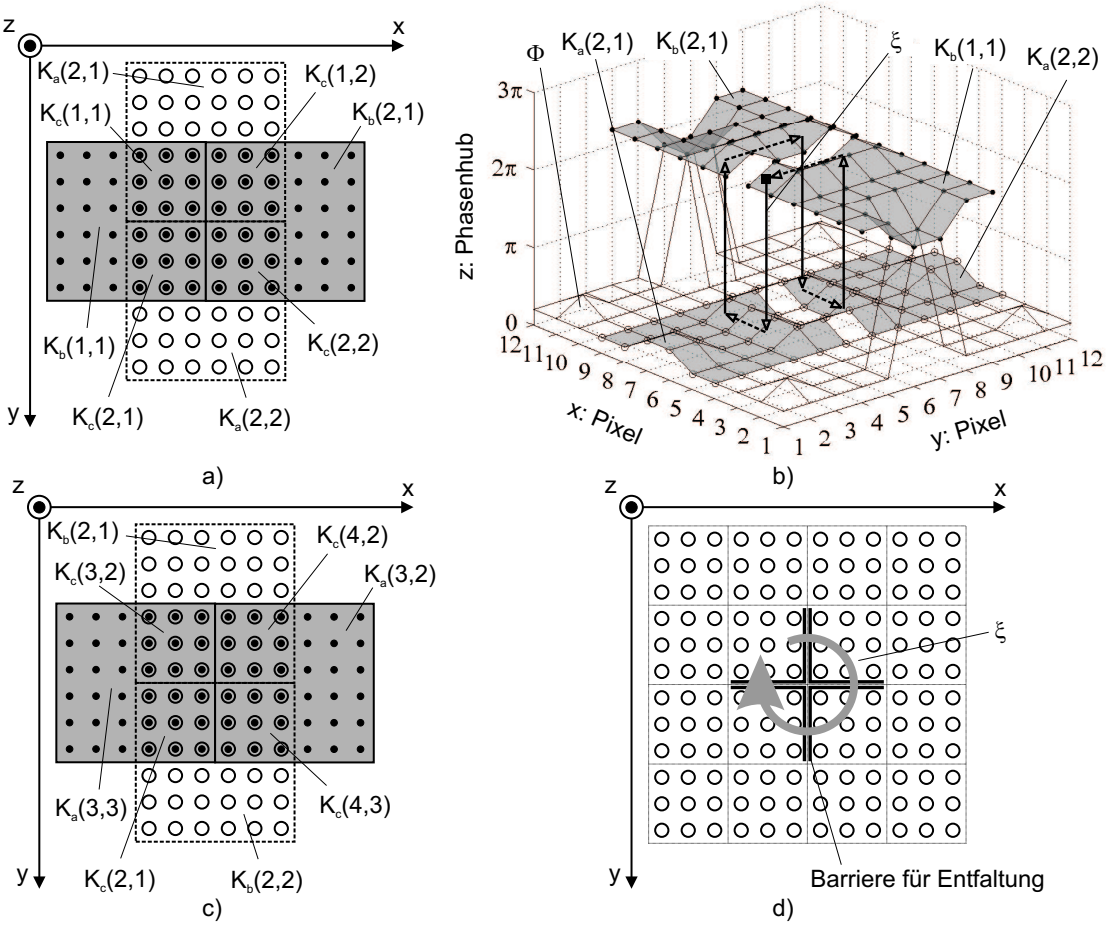


Abbildung 5.7: Residuen: a) Erste Kachelaufteilung, b) dreidimensionale Ansicht der entfalteten Flächen, c) zweite Kachelaufteilung, d) Integrationsweg ξ

cheln $K_c(1,1)$, $K_c(1,2)$, $K_c(2,1)$ und $K_c(2,2)$ in der linken oberen Ecke eines Bildes, wie dies Abbildung 5.2 verdeutlicht. Diese entstehen durch die Kachelentfaltung der Kacheln $K_a(1,2)$, $K_a(2,2)$, $K_b(1,1)$ und $K_b(2,1)$ wie sie Abschnitt 5.2.1 beschreibt. Allgemein gilt für die Bildung des Ringintegrals über die Abstände $\Delta O_k(i,j)$ der Kachelebenen K_a und K_b entlang des Weges ξ nach Gleichung 5.6:

$$\oint_{\xi} \Delta O_k(i,j) = 0 \tag{5.57}$$

Dabei definiert sich der Offset $\Delta O_k(i, j)$ als:

$$\Delta O_k(i, j) = \mathrm{Int}\left[\frac{c_b\left(K_b|_{K_c(i,j)}\right) - c_a\left(K_b|_{K_c(i,j)}\right)}{2\pi}\right] \tag{5.58}$$

Die Funktion $\mathrm{Int}[w]$ rundet die Zahl w auf den nächstliegenden ganzzahligen Wert. Die Kacheln der Ebenen K_a und K_b repräsentieren jeweils dieselben Überlappungsbereiche K_c. Die Verbindung der Kacheln K_c zu einem Gesamtbild erfolgt durch die Überlappung der Kachelebenen ´K_a und K_b. Die Pfade bei der Suche nach Residuen durchlaufen genau vier Kacheln in Anlehnung an die Suche nach Residuen auf Pixelebene [21], die in der Menge $K_c^{\square}(i, j)$ enthalten sind:

$$\begin{aligned} K_c^{\square}(x_{\square}, y_{\square}) \quad \in \{ \quad & K_c(x_{\square}, y_{\square}), K_c(x_{\square}, y_{\square} + 1), \\ & K_c(x_{\square} + 1, y_{\square} + 1), K_c(x_{\square} + 1, y_{\square}) \ \} \end{aligned} \tag{5.59}$$

Durch die Kachelung nach Abbildung 5.2 entstehen vier verschiedene Fälle. Hier bilden sich die Kachelkoordinaten x_a, y_a, x_b und y_b aus $x_{\square}$ und $x_{\square}$ durch das Bildungsgesetz nach den Gleichungen 5.2-5.5.

1. Alle Kacheln K_c sind Elemente einer Kachel von $K_a(x_a, y_a)$, wenn für $x_{\square}$ gilt:

$$x_{\square} \bmod 2 = 0 \quad \wedge \quad y_{\square} \bmod 2 = 0 \tag{5.60}$$
$$K_c(x_{\square}, y_{\square}) \quad = \quad K_a(x_a, y_a) \tag{5.61}$$

2. Alle Kacheln K_c sind Elemente einer Kachel von $K_b(x_b, y_b)$, wenn für $x_{\square}$ gilt:

$$x_{\square} \bmod 2 = 0 \quad \wedge \quad y_{\square} \bmod 2 = 0 \tag{5.62}$$
$$K_c(x_{\square}, y_{\square}) \quad = \quad K_b(x_b, y_b) \tag{5.63}$$

3. Die Kacheln K_a und K_b im Bereich $K_c^{\square}$ überlappen sich analog zu Abbildung 5.7.a:

$$x_{\square} \bmod 2 = 0 \quad \wedge \quad y_{\square} \bmod 2 = 1 \tag{5.64}$$

4. Die Kacheln K_a und K_b im Bereich $K_c^{\square}$ überlappen sich analog zu Abbildung 5.7.c:

$$x_{\square} \bmod 2 = 1 \quad \wedge \quad y_{\square} \bmod 2 = 0 \tag{5.65}$$

In den Fällen eins und zwei kann kein Residuum auftreten, da dort K_a bzw. K_b die einzige verbindende Kachel darstellt. In den Fällen drei und vier ergibt sich der Weg des Ringintegrals mit dem Startpunkt der Kachel auf der Ebene K_b an der Stelle der Kachel $K_c(x_\square, y_\square)$. Der Weg ξ, der in Abbildung 5.7.b skizziert ist, folgt dann einem Sprung entlang der z-Achse auf die Ebene K_a, einer Seitwärtsbewegung auf den nächsten Punkt $K_c(x_\square + 1, y_\square)$ einem Sprung auf die Ebene K_b und so fort. Dadurch ergibt sich für das Ringintegral folgende Summe:

$$\begin{aligned} Q_r(K_c^\square) &= \sum_{K_c^\square} \Delta O_k \qquad (5.66)\\ &= \Delta O_k(i,j) - \Delta O_k(i,j+1) + \Delta O_k(i+1,j+1) - \Delta O_k(i+1,j) \end{aligned}$$

Das Qualitätsmaß Q_r wird im Folgenden als Kachelkonsistenz bezeichnet. Falls $Q_r \neq 0$ ist, darf der Weg der Entfaltung die Barrieren entlang der inkonsistenten Kacheln nach Abbildung 5.7.d nicht mehr kreuzen. Kachelkonsistenz kann auch binär ausgedrückt werden:

$$Q_r^*(K_c^\square) = \begin{cases} 0 & \text{für} \quad Q_r = 0 \\ 1 & \text{für} \quad Q_r \neq 0 \end{cases} \qquad (5.67)$$

Die Übergänge von einer Kachel zur nächsten definieren die Größen Q_b^r, Q_b^l, Q_b^o und Q_b^u. Die Übergänge zwischen den Kacheln ergeben sich mit Q_r zu:

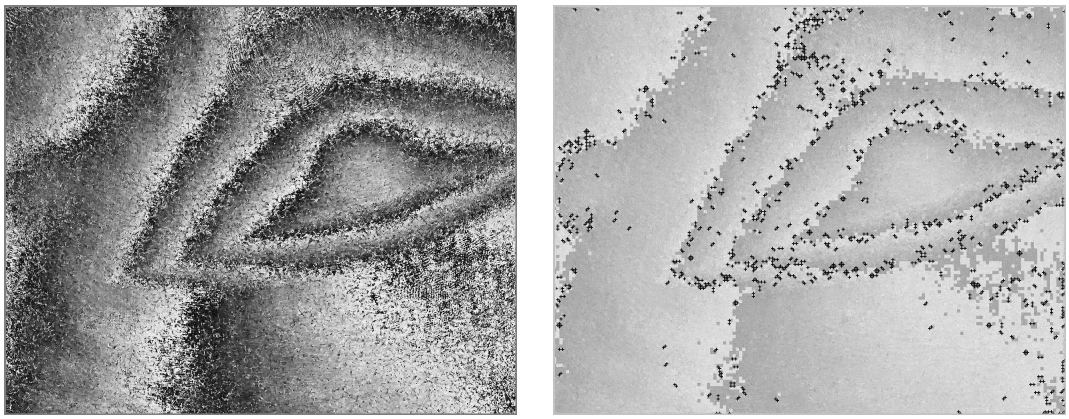
$$\begin{aligned} Q_b^o(K_c(x_\square, y_\square)) &= 0 \qquad (5.68)\\ Q_b^r(K_c(x_\square, y_\square)) &= Q_r^*(K_c^\square) \qquad (5.69)\\ Q_b^u(K_c(x_\square, y_\square)) &= Q_r^*(K_c^\square) \qquad (5.70)\\ Q_b^l(K_c(x_\square, y_\square)) &= 0 \qquad (5.71) \end{aligned}$$

$$\begin{aligned} Q_b^o(K_c(x_\square + 1, y_\square)) &= 0 \qquad (5.72)\\ Q_b^r(K_c(x_\square + 1, y_\square)) &= 0 \qquad (5.73)\\ Q_b^u(K_c(x_\square + 1, y_\square)) &= Q_r^*(K_c^\square) \qquad (5.74)\\ Q_b^l(K_c(x_\square + 1, y_\square)) &= Q_r^*(K_c^\square) \qquad (5.75) \end{aligned}$$

$$\begin{aligned} Q_b^o(K_c(x_\square + 1, y_\square + 1)) &= Q_r^*(K_c^\square) \qquad (5.76)\\ Q_b^r(K_c(x_\square + 1, y_\square + 1)) &= 0 \qquad (5.77)\\ Q_b^u(K_c(x_\square + 1, y_\square + 1)) &= Q_r^*(K_c^\square) \qquad (5.78)\\ Q_b^l(K_c(x_\square + 1, y_\square + 1)) &= 0 \qquad (5.79) \end{aligned}$$

$$Q_b^o(K_c(x_\square, y_\square + 1)) = Q_r^*(K_c^\square) \quad (5.80)$$
$$Q_b^r(K_c(x_\square, y_\square + 1)) = Q_r^*(K_c^\square) \quad (5.81)$$
$$Q_b^u(K_c(x_\square, y_\square + 1)) = 0 \quad (5.82)$$
$$Q_b^l(K_c(x_\square, y_\square + 1)) = 0 \quad (5.83)$$

Das Beispiel nach Abbildung 5.8 zeigt das Messbeispiel aus dem einführenden



a) b)

Abbildung 5.8: a) Phasenbild, b) Barrieren an Residuen

Abschnitt 5.2. In Abbildung 5.8 sind in die Übergänge inkonsistenter Kacheln analog zu Abbildung 5.7.d Barrieren eingefügt.

5.2.4 Inkonsistente Lücken

Eine Entfaltung, basierend auf den bisher vorgestellten Qualitätskriterien, führt zu gravierenden Fehlern. Dies verdeutlicht Abbildung 5.9, in der nach der Entfaltung ein Phasensprung quer durch das Bild erhalten bleibt. Die Ursache dieses Fehlers liegt in der eingeschränkten Suche nach Residuen, die sich auf die kleinsten möglichen Pfade ξ beschränkt. Die Lokalisierung des fehlerhaften Umlaufs um das Residuum zeigt Abbildung 5.9.b. Man sieht deutlich, wie sich dieses lokale Problem durch das gesamte Bild fortpflanzt. Umgekehrt führt aber nicht zwangsläufig jeder Umlauf um ein Residuum zum Fehler und dessen Fortpflanzung. Mögliche Strategien zur Behebung dieses Problems beschreibt Abschnitt 5.1. Die dort vorgestellten Strategien beruhen jedoch auf Methoden, deren Laufzeit entweder nicht abzuschätzen ist oder die zumindest größer als linear mit der Kachelzahl wächst [21]. Ferner eignen sich die meisten dieser Algorithmen a priori aufgrund ihrer

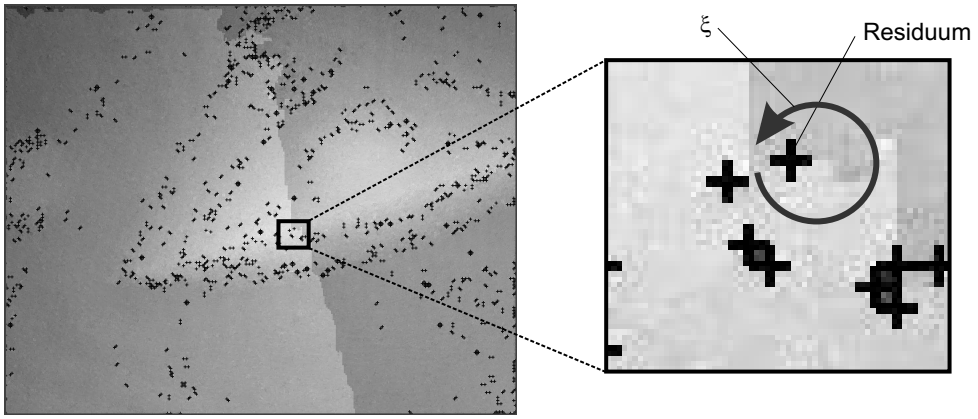


Abbildung 5.9: a) Kachelbild mit Residuen, b) Ausschnittsvergößerung

Komplexität nicht für eine Realisierung in Hardware. Ein wesentlich einfacherer Algorithmus ergibt sich aus der Überlegung heraus, dass ein Residuum im Wesentlichen lokale Störungen verursacht. Als Randbedingung lässt sich ferner noch feststellen, dass Residuen praktisch nur dort auftreten, wo die Kachelkorrelation $Q_k(K_c) < n(K_c)$, also $Q_g = 0$ ist. Es lassen sich zwar prinzipiell Fälle konstruieren, die diese Bedingung nicht erfüllen, die statistische Untersuchung zeigt jedoch, dass dieser Fall praktisch nie eintritt. Aus den Vorüberlegungen ergeben sich folgende Hypothesen:

1. In schwierigen Bereichen mit lokalen Störungen ist die Dichte an Kacheln mit nicht maximaler Kachelkorrelation $Q_g = 0$ so hoch, dass lediglich in ein Kacheln breiten Durchgängen sich fortpflanzende Fehler entstehen können.

2. Für die Entfaltung in Bereichen, in denen lediglich vereinzelt Residuen auftreten, schützen die Barrieren ausreichend vor einer Fehlerfortpflanzung.

Aus diesen Überlegungen lässt sich folgender Algorithmus ableiten:

1. Maskieren aller Kacheln:

$$Q_m(K_c) = \overline{Q}_{gK_c}. \tag{5.84}$$

2. Maskieren aller Kacheln im Bild, die einen Durchgang der Breite eines Pixels zwischen denen in Schritt 1 maskierten Pixel bilden.

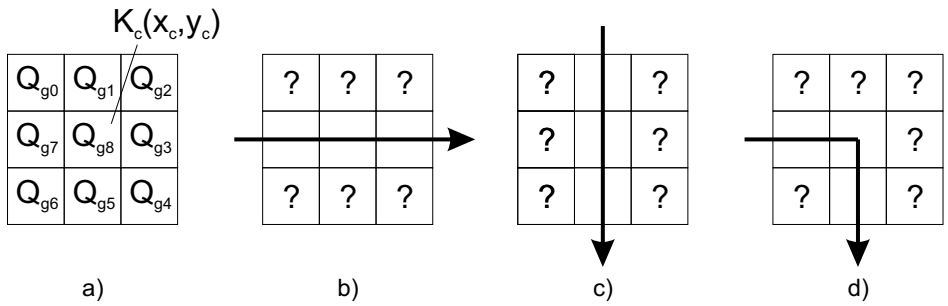


Abbildung 5.10: Eine Kachel breite Durchgänge zwischen Kacheln: a) Bezeichnung der Kacheln, b) horizontaler Durchgang, c) vertikaler Durchgang, d) gewinkelter Durchgang

Die Funktionstüchtigkeit der Hypothesen und dieses Algorithmus, der rein empirisch entstanden ist, bestätigen die Stabilitätsuntersuchungen in Abschnitt 6.7.
Die Aufgabe, genau eine Kachel breite Durchgänge zwischen maskierten Kacheln zu finden, lässt sich auf einfache Fallunterscheidung zurückführen. Abbildung 5.10 verdeutlicht die Aufgabenstellung.
Bei der Betrachtung der Kachel $K_c(x_a, y_a)$ ergeben sich für die Qualitätsmaße Q_{g0} bis Q_{g7} aus Abbildung 5.10.a:

$$Q_{g0} = Q_g(K_c(x_a - 1, y_a - 1)) \tag{5.85}$$
$$Q_{g1} = Q_g(K_c(x_a, y_a - 1)) \tag{5.86}$$
$$Q_{g2} = Q_g(K_c(x_a + 1, y_a - 1)) \tag{5.87}$$
$$Q_{g3} = Q_g(K_c(x_a + 1, y_a)) \tag{5.88}$$
$$Q_{g4} = Q_g(K_c(x_a + 1, y_a + 1)) \tag{5.89}$$
$$Q_{g5} = Q_g(K_c(x_a, y_a + 1)) \tag{5.90}$$
$$Q_{g6} = Q_g(K_c(x_a - 1, y_a + 1)) \tag{5.91}$$
$$Q_{g7} = Q_g(K_c(x_a - 1, y_a)) \tag{5.92}$$
$$Q_{g8} = Q_g(K_c(x_a, y_a)) \tag{5.93}$$

Daraus lässt sich die Besetzung der Kacheln in der unmittelbaren Umgebung von K_c als ein Binärwort $KB(K_c)$ darstellen:

$$KB(K_c) = \sum_{i=0}^{7} Q_{gi} 2^i \tag{5.94}$$

Die Rotation der Kachel um 90° entspricht damit der bitweisen Rotation des Binärworts $KB(K_c)$ um 3 Bit nach links:

$$KB(K_c)_{90°} = RoL(KB(K_c), 3) \tag{5.95}$$

Zur Beantwortung der Frage, ob eine Kachel $K_c(x_c, y_c)$ einen Durchgang nach Abbildung 5.10 darstellt, reicht eine Betrachtung der unmittelbar benachbarten Kacheln aus. Die Fallunterscheidung besteht darin, alle möglichen Kombinationen der Nachbarkacheln zu suchen, die ein Maskieren der Kachel $K_c(x_c, y_c)$ erfordern. Es bieten sich für die Beschreibung dieser Aufgabe zwei Möglichkeiten an. Die beiden Betrachtungsweisen führen jeweils zu unterschiedlichen Hardwarerealisierungen, deren Bewertung im Abschnitt 5.3.4 folgt.
Im ersten Fall existieren nur zwei Arten von elementaren Durchgängen. Alle wei-

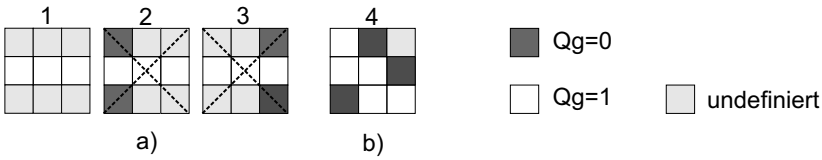


Abbildung 5.11: Charakteristik der elementaren Kacheldurchgänge der Breite eins: a) horizontaler Durchgang, b) gewinkelter Durchgang

teren Möglichkeiten ergeben sich dann durch Drehungen um 90°, 180° oder 270°:

1. Horizontaler Durchgang nach Abbildung 5.10.a:

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g8} \wedge Q_{g7} \wedge Q_{g3} \wedge (\overline{Q}_{g0} \vee \overline{Q}_{g1} \vee \overline{Q}_{g2}) \wedge \\ & \quad (\overline{Q}_{g4} \vee \overline{Q}_{g5} \vee \overline{Q}_{g6}) \wedge \\ & \quad ((Q_{g0} \vee Q_{g6}) \wedge (Q_{g2} \vee Q_{g4})) \\ 0 & \text{sonst} \end{cases} \tag{5.96}$$

2. Gewinkelter Durchgang nach Abbildung 5.10.d:
   Die einzig mögliche Kombination ergibt sich aus Abbildung 5.11.b:

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g0} \wedge Q_{g4} \wedge Q_{g5} \wedge Q_{g7} \wedge Q_{g8} \wedge \\ & \quad \overline{Q}_{g6} \wedge \overline{Q}_{g7} \wedge \overline{Q}_{g3} \\ 0 & \text{sonst} \end{cases} \tag{5.97}$$

Im zweiten Fall gibt es insgesamt sechs verschiedene elementare Durchgänge, die sich durch die explizite Ausformulierung aller möglichen Drehungen ergibt.

1. Horizontaler Durchgang nach Abbildung 5.10.a:

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g8} \wedge Q_{g7} \wedge Q_{g3} \wedge (\overline{Q}_{g0} \vee \overline{Q}_{g1} \vee \overline{Q}_{g2}) \wedge \\ & \quad (\overline{Q}_{g4} \vee \overline{Q}_{g5} \vee \overline{Q}_{g6}) \wedge \\ & \quad ((Q_{g0} \vee Q_{g6}) \wedge (Q_{g2} \vee Q_{g4})) \\ 0 & \text{sonst} \end{cases} \tag{5.98}$$

2. Horizontaler Durchgang nach Abbildung 5.10.a um 90° gedreht:

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g8} \wedge Q_{g1} \wedge Q_{g5} \wedge (\overline{Q}_{g2} \vee \overline{Q}_{g3} \vee \overline{Q}_{g4}) \wedge \\ & \quad (\overline{Q}_{g0} \vee \overline{Q}_{g7} \vee \overline{Q}_{g6}) \wedge \\ & \quad ((Q_{g0} \vee Q_{g2}) \wedge (Q_{g6} \vee Q_{g4})) \\ 0 & \text{sonst} \end{cases} \tag{5.99}$$

3. Gewinkelter Durchgang nach Abbildung 5.10.b

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g0} \wedge Q_{g4} \wedge Q_{g5} \wedge Q_{g7} \wedge Q_{g8} \wedge \\ & \quad \overline{Q}_{g6} \wedge \overline{Q}_{g7} \wedge \overline{Q}_{g3} \\ 0 & \text{sonst} \end{cases} \tag{5.100}$$

4. Gewinkelter Durchgang nach Abbildung 5.10.b um 90°gedreht

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g2} \wedge Q_{g3} \wedge Q_{g5} \wedge Q_{g6} \wedge Q_{g8} \wedge \\ & \quad \overline{Q}_{g1} \wedge \overline{Q}_{g4} \wedge \overline{Q}_{g7} \\ 0 & \text{sonst} \end{cases} \tag{5.101}$$

5. Gewinkelter Durchgang nach Abbildung 5.10.b um 180°gedreht

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g0} \wedge Q_{g1} \wedge Q_{g3} \wedge Q_{g4} \wedge Q_{g8} \wedge \\ & \quad \overline{Q}_{g2} \wedge \overline{Q}_{g5} \wedge \overline{Q}_{g7} \\ 0 & \text{sonst} \end{cases} \tag{5.102}$$

6. Gewinkelter Durchgang nach Abbildung 5.10.b um 270°gedreht

$$Q_m(K_c) = \begin{cases} 1 & \text{für} \quad Q_{g2} \wedge Q_{g3} \wedge Q_{g5} \wedge Q_{g6} \wedge Q_{g8} \wedge \\ & \quad \overline{Q}_{g1} \wedge \overline{Q}_{g4} \wedge \overline{Q}_{g7} \\ 0 & \text{sonst} \end{cases} \tag{5.103}$$

Das Beispiel nach Abbildung 5.8 zeigt das Messbeispiel aus dem einführenden Abschnitt 5.2. Abbildung 5.12.a zeigt die maskierten Kacheln nach Gleichung 5.84. In Abbildung 5.12.b sind dann alle maskierten Kacheln zu sehen.

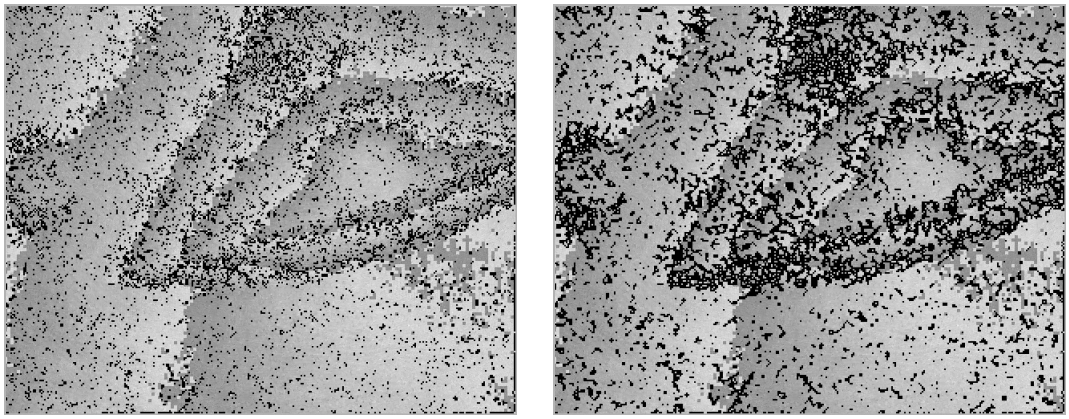


Abbildung 5.12: Maskierte Kacheln a) ohne geschlossene Lücken, b) mit geschlossenen Lücken

5.2.5 Entfalten

Das Entfalten der maskierten Bilder erfolgt nun nach der Methode der Flächenfüllung aus der Graphikprogrammierung [82]. Dies ist ein rekursiver Algorithmus, der das Bild Kachel für Kachel durchläuft und jeweils versucht, den Bereich nach oben, unten, rechts und links zu erweitern. Der Algorithmus benötigt an dieser Stelle keine weitere Intelligenz, da die fehlerhaften Übergänge maskiert sind. Abbildung 5.13 veranschaulicht den Weg der Entfaltung innerhalb eines Bildes. Durch das Maskieren der Kacheln entstehen unabhängige Bereiche A_c innerhalb des Bildes, die getrennt voneinander entfaltet werden. Jeder Bereich A_c erhält eine Nummer N_A, die den Kacheln $K_c \in A_c$ zugeordnet wird:

$$N_A(K_c) = N_A(A_c) \quad \text{mit} \quad K_c \in A_c \tag{5.104}$$

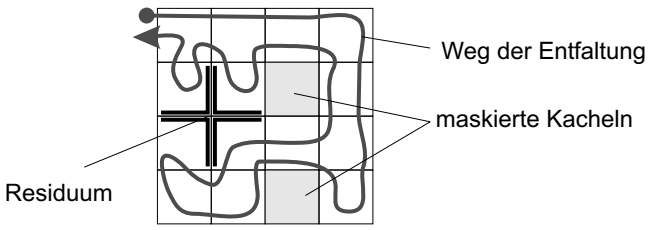


Abbildung 5.13: Weg der Entfaltung

Das Qualitätsmaß Q_u markiert die entfalteten bzw. nicht entfalteten Kacheln:

$$Q_u(K_c) = \begin{cases} 1 & \text{falls Kachel entfaltet} \\ 0 & \text{falls Kachel nicht entfaltet} \end{cases} \tag{5.105}$$

Der Algorithmus beginnt damit, zunächst systematisch nach nicht entfalteten Kacheln zu suchen:

1. Suche nach einer Kacheln für die gilt:

$$Q_u(K_c) = 0 \ \wedge \ Q_g = 0 \tag{5.106}$$

2. Entfalte den Bereich A_c, wobei gilt:

$$N_A(A_c) = 1 + \inf_{A_c} [N_A(A_c)] \tag{5.107}$$

Die Entfaltung eines Bereich A_c läuft nach dem folgenden rekursiven Algorithmus ab. Dabei wandert der Algorithmus von Kachel zu Kachel und entfaltet diese. Ein Übergang von einer Kachel K_c zu einer Kachel K_c^* darf unter der folgenden Voraussetzung stattfinden:

$$\begin{aligned} Q_b^o(Kc(x_a, y_a)) &= 0 \\ Q_m(K_c^*) &= 0 \\ Q_u(K_c^*) &= 0 \end{aligned} \tag{5.108}$$

Da in diesem Schritt der Entfaltung ausschließlich Kacheln mit maximaler Korrelation Q_k betrachtet werden, erhalten die Ebenen K_a und K_b getrennte Funktionen zugewiesen:

- K_a erhält die Bezeichnung *Bildebene*.
- K_b erhält die Bezeichnung *Klebeebene*. Sie hat die Funktion die Elemente der Bildebene zu verbinden.

Beim Übergang von einer Kachel $K_c(x_a, y_a)$ zu einer Kachel K_c^* dient die Klebeebene dazu, abhängig von der Laufrichtung, den Offset der benachbarten Kacheln zu bestimmen. Dabei muss folgende Bedingung erfüllt sein:

$$\begin{array}{llll} \text{für} & x_a^* = x_a & , \ y_a^* = y_a - 1 & \text{wenn} \quad y_a \bmod 2 = 0 \\ \text{für} & x_a^* = x_a + 1 & , \ y_a^* = y_a & \text{wenn} \quad x_a \bmod 2 = 0 \\ \text{für} & x_a^* = x_a & , \ y_a^* = y_a + 1 & \text{wenn} \quad y_a \bmod 2 = 1 \\ \text{für} & x_a^* = x_a - 1 & , \ y_a^* = y_a & \text{wenn} \quad x_a \bmod 2 = 1 \end{array}$$

Um ausgehend von einer Kacheln die vorhergehende Kachel wiederzufinden dient der Kacheleintritt e_k, der aus folgenden Elementen besteht:

$$e_k \in \{s_o, s_r, s_u, s_l\} \tag{5.109}$$

Im Einzelnen läuft der Algorithmus zur Entfaltung in folgenden Schritten ab:

1. Initialisiere die Variablen beim Startpunkt $K_c(x_a, y_a - 1)$ und fahre fort mit Schritt 4:

$$O(Kc(x_a, y_a)) = 0 \quad (5.110)$$
$$e_k(Kc(x_a, y_a)) = s_b \quad (5.111)$$

2. Berechne den Kacheloffset:

$$O(K_c^*) = \begin{cases} O(K_c) - O_k(K_c) + O_k(K_c^*) & \text{für Bed. 5.109} \\ O(K_c) & \text{sonst} \end{cases} \quad (5.112)$$

3. Markiere die Kachel als entfaltet, weise die Nummer des Bereiches A_c zu und gehe zur neuen Kachel:

$$N_A(Kc(x_a, y_a)) = N_A(A_c) \quad (5.113)$$
$$Q_u(Kc(x_a, y_a)) = 1 \quad (5.114)$$
$$K_c = K_c^* \quad (5.115)$$

4. Gehe zur oberen Kachel falls Gleichung 5.108 gilt und fahre fort mit Schritt 2, wobei gilt

$$K_c^* = K_c(x_a, y_a - 1) \quad (5.116)$$
$$e_k(K_c^*) = s_u \quad (5.117)$$

5. Gehe zur rechten Kachel falls Gleichung 5.108 gilt und fahre fort mit Schritt 2, wobei gilt

$$K_c^* = K_c(x_a + 1, y_a) \quad (5.118)$$
$$e_k(K_c^*) = s_l \quad (5.119)$$

6. Gehe zur unteren Kachel falls Gleichung 5.108 gilt und fahre fort mit Schritt 2, wobei gilt

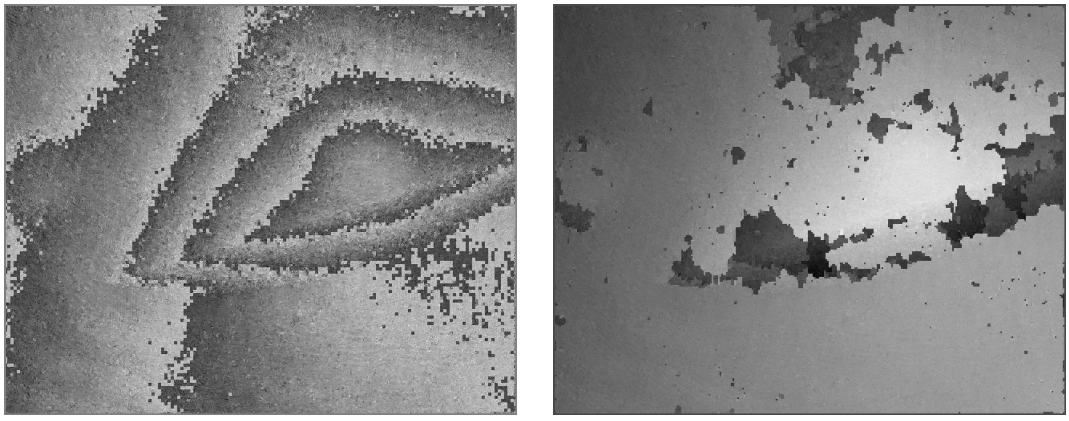
$$K_c^* = K_c(x_a, y_a + 1) \quad (5.120)$$
$$e_k(K_c^*) = s_o \quad (5.121)$$

7. Gehe zur linken Kachel falls Gleichung 5.108 gilt und fahre fort mit Schritt 2, wobei gilt

$$K_c^* = K_c(x_a - 1, y_a) \quad (5.122)$$
$$e_k(K_c^*) = s_r \quad (5.123)$$

8. Kehre zur vorhergehenden Kachel zurück und fahre mit der Suche nach weiteren entfaltbaren Nachbarkacheln fort. Falls keine vorhergehende Kachel nach Bedingung 5.124 mehr existiert, brich ab.

$$e_k(K_c^*) = s_r \quad (5.124)$$



a) b)

Abbildung 5.14: Entfaltung nach dem Algorithmus zur Flächenfüllung: a) Bildebene K_a, b) entfaltete Bereiche

Im nächsten Schritt müssen alle maskierten Kacheln entfaltet werden. Dies erfolgt nach dem sogenannten umgekehrten Morphing [83]. Dabei werden alle Kacheln K_c^*, die nicht entfaltete sind, aber an eine entfaltete Kachel K_c angrenzen, dem entsprechenden Bereich zugeordnet. Der Offset O der Kachel K_c^* berechnet sich dabei nach Gleichung 5.112.

$$N_A(K_c^*) = N_A(K_c) \quad \text{wenn} \quad Q_u(K_c) = 1, \; x_a^* \in \{x_a + 1, x_a - 1\}, \; y_a^* \in \{y_a + 1, y_a - 1\} \quad (5.125)$$

Dieser Vorgang wird solange wiederholt bis alle Kacheln des Bildes einem Bereich zugeordnet sind.

Das Beispiel nach Abbildung 5.14 zeigt das Messbeispiel aus dem einführenden Abschnitt 5.2. Abbildung 5.14 zeigt den Übergang vom Kachelbild zu entfalteten Bereichen.

5.2.6 Bereiche verbinden

Die Aufgabe des nächsten Schrittes besteht darin, die bisher entstandenen Bereiche miteinander zu verbinden. Diese Aufgabe kann wiederum als klassisches Entfaltungsproblem angesehen werden und damit auch mit allen Algorithmen aus Abschnitt 5.1 bearbeitet werden. Auch wenn die Zahl der zu verarbeitenden Daten von Pixel- über die Kachel- bis zur Bereichsebene stark reduziert wurde, scheiden für die folgenden Aufgaben die nicht deterministischen und langwierigen Verfahren aus.
Der folgende Algorithmus ist von der Idee der durch die Qualitätskarte geleiteten Entfaltung geprägt. Zur Beurteilung einer Kachel steht das Qualitätskriterium Q_k zur Verfügung. Darauf basierend lassen sich nun auf Bereichsebene eine Reihe von Qualitätskriterien für die Bereiche definieren. Diese beziehen sich nicht nur auf die Qualität der Kachel an sich, sondern insbesondere auf die Beurteilung der Nachbarschaftsbeziehung zwischen den Kacheln.
Zunächst ergibt sich die Summe Q_a^Σ der Kachelkorrelationen Q_k aller Kacheln des Bereichs A_c zu

$$Q_a^\Sigma(A_c) = \sum_{K_c \in A_c} Q_k(K_c) \tag{5.126}$$

Aus Q_a^Σ ergibt sich die Bereichskorrelation Q_a^* durch die Normierung auf die Zahl der Kacheln des Bereichs $N_A(A_c)$. Damit lässt sich die Qualität eines Bereiches quantitativ bewerten.

$$Q_a^*(A_c) = \frac{Q_a^\Sigma(A_c)}{n_A(A_c)} \tag{5.127}$$

Daraus ergibt sich die Beurteilung Q_a der durch die Verbindung entstehenden Gesamtfläche zu:

$$Q_a(A_{c1}, A_{c2}) = \frac{Q_a^\Sigma(A_{c1}) + Q_a^\Sigma(A_{c2})}{n_A(A_{c1}) + n_A(A_{c2})} \tag{5.128}$$

Die Beurteilung des Übergangs Q_n zwischen den Bereichen spielt eine entscheidende Rolle. Zwei Kacheln K_{c1} und K_{c2} aus den Bereichen A_{c1} und A_{c2} gelten genau dann als benachbart, wenn sie übereinander oder nebeneinander liegen. Insbesondere gelten sie nicht als benachbart, wenn sie sich lediglich an einer Ecke

berühren. Unter Berücksichtigung der Residuen ergibt sich damit für die Nachbarschaft Q_M:

$$Q_M(K_{c1}, K_{c2}, A_{c1}, A_{c2}) = \begin{cases} 1 \text{ für} & |x_{c1} - x_{c2}| + |y_{c1} - y_{c2}| = 1; \\ & K_{c1} \in A_{c1};\ K_{c2} \in A_{c2}; \\ & Q_b(K_{c1};\ K_{c2}) = 0 \\ & Q_u(K_{c1}) = 1;\ Q_u(K_{c2}) = 1; \\ 0 \text{ sonst} & \end{cases} \tag{5.129}$$

Daraus ergibt sich die Menge N_k aller benachbarter Kacheln K_{c1} und K_{c2} entlang der Grenze zweier Bereiche A_{c1} und A_{c2}:

$$N_k(A_{c1}, A_{c2}) = \{K_{c1}, K_{c2} \mid Q_M(K_{c1}, K_{c2}, A_{c1}, A_{c2}) = 1\} \tag{5.130}$$

Die Summe $Q_f^{\Sigma}(N_k)$ aller Kachelkorrelationen Q_k benachbarter Kacheln N_k zweier Bereiche A_{c1} und A_{c2} ergibt sich zu:

$$Q_f^{\Sigma}(N_k) = \sum_{N_k(A_{c1},A_{c2})} (Q_k(K_{c1}) + Q_k(K_{c1})) = \sum_{N_k(A_{c1},A_{c2})} Q_f^s \tag{5.131}$$

Aus $Q_f^{\Sigma}(N_k)$ ergibt sich durch die Normierung auf die Zahl der einbezogenen Kacheln $n_n(N_k)$ die Bewertung des Grenzübergangs Q_f :

$$Q_f(N_k) = \frac{Q_f^{\Sigma}(N_k)}{n_n(N_k)} \tag{5.132}$$

Um eine Fehlerfortpflanzung zu vermeiden muss noch die Größe der Bereiche bezogen auf die Gesamtzahl der Kacheln des Bildes n_k in die Beurteilung einfließen:

$$Q_z(A_{c1}, A_{c2}) = \frac{n_A(A_{c1}) + n_A(A_{c2})}{n_k} \tag{5.133}$$

Um nun zu entscheiden in welcher Reihenfolge die verschiedenen Bereiche zusammengesetzt werden, dient das Nachbarschaftskriterium Q_n^*, das sich aus dem Produkt der verschiedenen Kriterien Q_a, Q_f und Q_z aus den Gleichungen 5.128, 5.132 und 5.133 ergibt:

$$\begin{aligned} Q_n^*(A_{c1}, A_{c2}) &= Q_a(A_{c1}, A_{c2}) \cdot Q_f(N_k(A_{c1}, A_{c2})) \cdot Q_z(A_{c1}, A_{c2}) \\ &= \frac{(Q_a^{\Sigma}(A_{c1}) + Q_a^{\Sigma}(A_{c2})) \cdot Q_f^{\Sigma}(N_k)}{n_n(N_k) \cdot n_k} \end{aligned} \tag{5.134}$$

Alle Nachbarschaftsbeziehungen zwischen aneinander grenzenden Bereichen A_{c1} und A_{c2} werden der Größe nach sortiert und anschließend in dieser Reihenfolge entfaltet. Der Offset O_a berechnet sich dabei zu:

$$O(A_{c1}, A_{c2}, N_k(A_{c1}, A_{c2})) = \text{Int}\left[\frac{\sum_{K_c \in N_k} O_k(K_c)}{n_n(N_k)}\right] \tag{5.135}$$

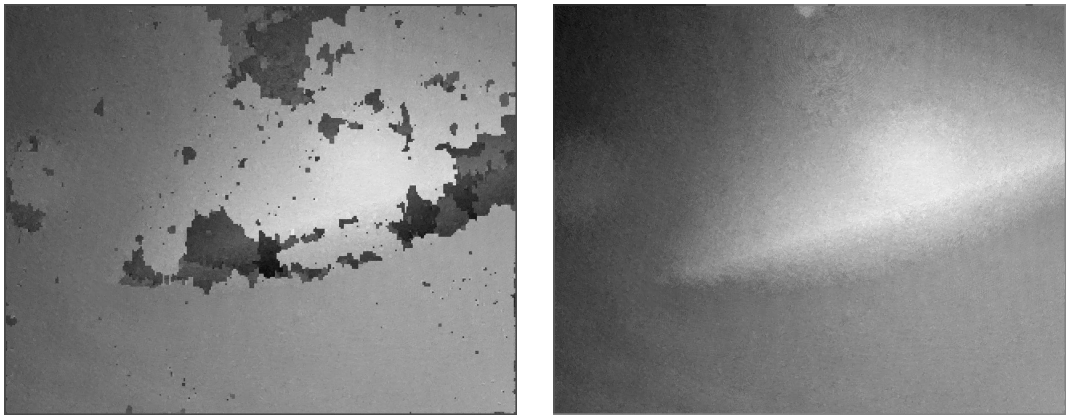


Abbildung 5.15: Übergang vom a) entfalteten Bereichen zum b) entfalteten Gesamtbild

Das Beispiel nach Abbildung 5.14 zeigt das Messbeispiel aus dem einführenden Abschnitt 5.2. Abbildung 5.15 zeigt den Übergang von den entfalteten Bereichen zum entfalteten Gesamtbild.

5.3 Umsetzung in Hardware

Der folgende Abschnitt beschreibt die Umsetzung der Algorithmen aus Kapitel 5.2 in Hardware. Die einzelnen Teile des Algorithmus, analog zum Kapitel 5.2, stellen Schritte in einer Pipelining-Architektur dar. Als Eingabe erhält die Pipeline ein Phasenbild, das durch die Bearbeitung der Specklebilder nach den in Abschnitt 4.1 beschriebenen Algorithmen entsteht. Im Falle eines Kamerabilds entsprechen die Punkte P_i Positionen von Kamerapixeln an der Position (x, y). Daraus ergibt sich für die Phasenwerte $\phi(P_i)$:

$$\phi(P_i) = \phi(x, y) \tag{5.136}$$

Die Phasenwerte $\phi(x, y)$ liegen, ausgehend von der Bittiefe B^*_ϕ, in einem Intervall von

$$\phi(x, y) \in [0; 2^{B^*_\phi}[\tag{5.137}$$

Der Phasenhub p_ϕ entspricht bei der ganzzahligen Betrachtung dem Wert 2π so dass gilt:

$$p_\phi = 2^{B^*_\phi} \tag{5.138}$$

Da während der Kachelentfaltung die Faltungskoeffizienten $k^*(x,y)$ direkt mit $\phi(x,y)$ verrechnet werden, muss die Bittiefe für ϕ um ein Bit für den Wertebereich und um ein Bit für das Vorzeichen erweitert werden:

$$B_\phi = B_\phi^* + 2 \tag{5.139}$$

Die betrachtete Kachelfläche K_a wird dabei als ein rechteckiger Ausschnitt des Kamerabilds beschrieben, bei dem die Positionen x_i und y_i in den folgenden Intervallen variieren.

$$x_i \in [x_s; x_e] \quad \text{mit} \quad n_x = x_e - x_s + 1 \tag{5.140}$$
$$y_i \in [y_s; y_e] \quad \text{mit} \quad n_y = y_e - y_s + 1 \tag{5.141}$$

Die Zahlen n_x und n_y beschreiben dabei die Zahl der Pixel des Rechtecks in x- und y-Richtung. Die Zahl der Pixel n der Fläche berechnet sich damit zu:

$$n = n_x n_y \tag{5.142}$$

Tabelle 5.3 enthält die Beschreibung der verwendeten Speicherobjekte. Diese bilden die zur Berechnung der Entfaltung benötigten Variablen aus dem vorhergehenden Kapitel 5.2. Im späteren Messsystem bilden diese Objekte die zu speichernden Elemente, wobei die Speichergröße und der Speicherort erst in Abschnitt 6 festgelegt werden.

5.3.1 Kachelentfaltung

Eine wesentliche Operation der Entfaltung einer Kachel K ist die Berechnung des Mittelwertes $c(K)$ als optimale Fläche der als lokal eben betrachteten Kachel. Zur Berechnung des Mittelwertes muss die Summe der Phasenwerte ϕ durch die Zahl der Pixel in der Kachel $n(K)$ dividiert werden. Diese Division lässt sich durch eine einfache Shift-Operation realisieren, wenn die Pixelzahl $n(K)$ in der Fläche K_a eine Potenz von zwei darstellt. Als Voraussetzung wird daher im folgenden angenommen dass die Dimensionen n_x und n_y die Bittiefe B_x und B_y besitzen, so dass gilt

$$n(K) \;=\; 2^{B_x} \cdot 2^{B_y} \tag{5.143}$$

Die Division lässt sich dann durch eine Shift-Operation nach rechts um s_r Bits, wie in Abschnitt 4.2.3 beschrieben, realisieren:

$$s_r = B_x + B_y \tag{5.144}$$

Aus der Zahl der Kacheln n und der Bittiefe B des Bildes ergibt sich die Änderung des Mittelwertes Δc bei Änderung eines einzigen Faltungskoeffizienten von

$k(x_0, y_0)$ nach $k^*(x_0, y_0)$ nach den Gleichungen 5.13, 5.138, 5.143 und der Bedingung 5.32 zu:

$$\Delta c = \pm\frac{2^B}{n} = \pm 2^{B_\phi - B_x - B_y} \tag{5.145}$$

Die Änderung des Mittelwertes Δc ist damit konstant und muss während der Be-

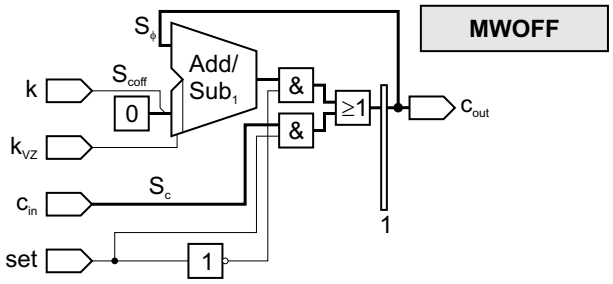


Abbildung 5.16: Berechnung des verschiebungsabhängigen Mittelwertes

rechnung im FPGA nicht mehr ausgeführt werden. Daraus ergibt sich die Realisierung der Berechnung des gleitenden Mittelwertes in Modul $MWOFF$ nach Abbildung 5.16. Die Position des Bits aus dem Signal S_{MWOff}, welches die Addition des festen Offsets darstellt, ergibt sich aus Gleichung 5.145.

$$b(S_{coff}) = B_\phi - B_x - B_y \tag{5.146}$$

Das Vorzeichenbit k_{VZ} definiert, ob der Offset addiert oder subtrahiert wird. Initial erhält das Modul seinen Startmittelwert durch die Leitungen c_{in}, der in das Register R_5 geschrieben wird, wenn das Signal *set* auf eins liegt. Den initialen

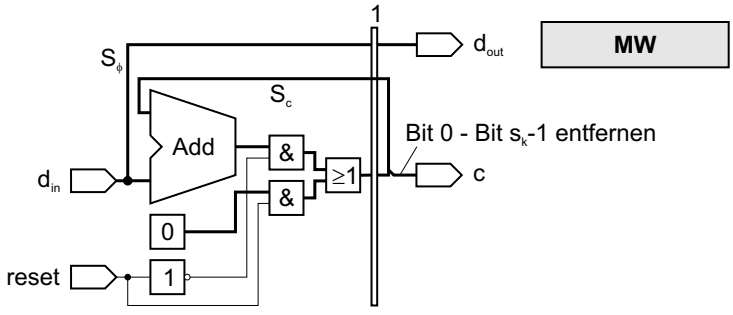


Abbildung 5.17: Berechnung des Mittelwertes der einlaufenden Daten

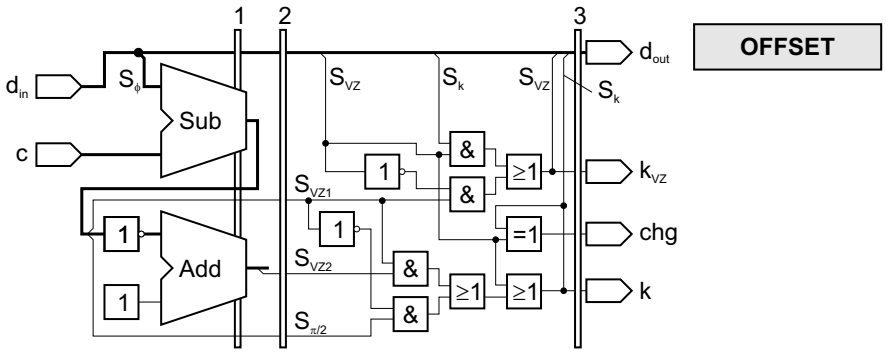


Abbildung 5.18: Offsetberechnung

Mittelwert berechnet das Modul *MW* nach Abbildung 5.17. Über die Resetleitung wird das Register R_4 zurückgesetzt. Die anschließend einlaufenden Daten werden summiert und dann durch das Entfernen der unteren s_r Bits nach Gleichung 5.144 dividiert. Das Register R_2 verzögert die Eingangsdaten um einen Takt und synchronisiert damit das Ergebnis der Berechnung des Mittelwertes mit den Eingangsdaten. Für die Busbreite der Summe S_c ergibt sich:

$$B(S_c) = B_\phi + B_x + B_y \tag{5.147}$$

Die Entfaltung der einzelnen Pixel sowie die Berechnung des gleitenden Mittelwertes beginnt mit der Berechnung der Verschiebung $k^*(P_i)$ nach Gleichung 5.36. Das Modul *HWOFF* berechnet zunächst die Differenz zwischen dem Mittelwert c und den einlaufenden Daten d_{in}. Der Offset ergibt sich dann durch Betrachtung des Bits $S_{\pi/2}$ mit:

$$b(S_{\pi/2}) = B_\phi - 2; \tag{5.148}$$

Falls die Differenz jedoch negativ wird, ermöglicht die Berechnung des Zweierkomplements ebenso eine einfache Betrachtung des Bits $b_{\pi/2}$. Register R_3 synchronisiert die Ausgabe der Daten mit der Berechnung des Offsets. Eine Eigenschaft des Algorithmus aus Abschnitt 5.2 besteht in der Tatsache, dass sich ein Offset maximal einmal ändern kann. Daraus ergibt sich, dass im Falle eines Wertes von S_{VZ} gleich eins einfach die alten Werte übernommen werden können. Dies spart insbesondere eine weitere Addition des Phasenwerts mit dem Faltungsparameter.
Die Kachelpipeline hält an, wenn das Abbruchkriterium nach Abschnitt 5.2 erreicht ist. Dies erfolgt, wenn genau n mal hintereinander keine Änderung der Faltungskoeffizienten $k(x, y)$ vorgenommen wurde. Diese Berechnung führt das Modul *TDONE* nach Abbildung 5.19 durch. Das Register R_2 wird im Falle einer

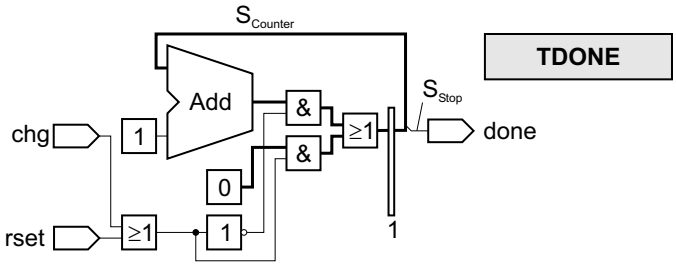


Abbildung 5.19: Zähler zum Berechnen der Konvergenz der Kachelpipeline

Änderung, welche das Signal *change* signalisiert, zurückgesetzt. Im anderen Falle inkrementiert der Addierer das Register R_2. Die Breite des Zählers $B_{Counter}$ ergibt sich zu:

$$B(S_{Counter}) = \text{ceil}\,[\ln_2 n] \tag{5.149}$$

Die Abbruchbedingung ist damit genau dann erreicht, wenn das Signal S_{Stop} gleich eins wird. Dies geschieht jedoch wegen der Verzögerung durch das Eingangsregister einen Takt später als die tatsächliche Konvergenz erreicht wäre, wenn zwischenzeitlich eine Änderung erfolgt ist.

$$b(S_{Stop}) = B(S_{Counter}) - 1 \tag{5.150}$$

Abbildung 5.20 zeigt das Zusammenspiel der bisher vorgestellten Module. Die

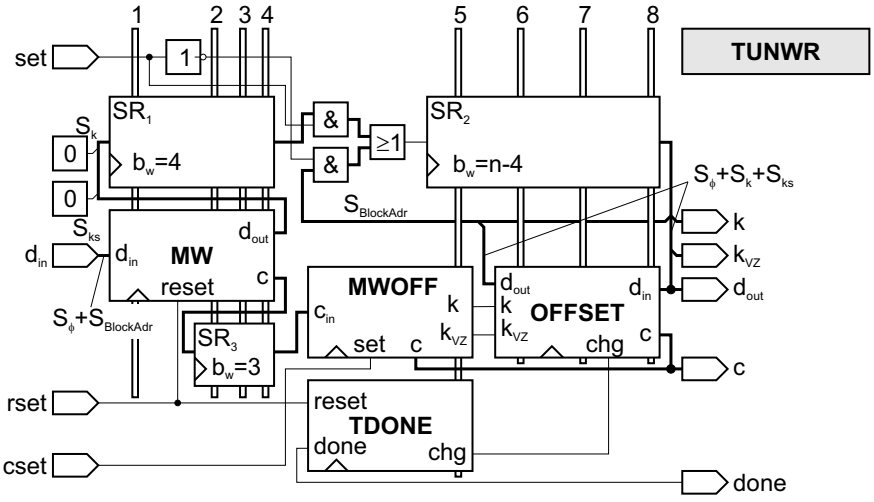


Abbildung 5.20: Kachelentfaltung

Werte einer Kachel gelangen zunächst in das Modul zur Mittelwertberechnung MW und von dort in das Schieberegister SR_1 mit der Länge von drei Worten. Der vollständig berechnete Mittelwert wird nach $n + 3$ Schritten durch Setzen des Signals *cset* direkt in das Modul $MWOFF$ geschrieben, wo der Mittelwert dann zur ersten Berechnung zur Verfügung steht. Das Schieberegister SR_1 liefert seine Werte weiter zum Schieberegister SR_2, wobei die zwischengeschaltete Logik entscheidet, ob die Pipeline noch mit Werten von außen gespeist wird, oder ob das Modul bereits die Kachel entfaltet. Die Umschaltung erfolgt dann, wenn n Werte in das Schieberegister SR_2 eingelaufen sind. Damit ist dann auch die nachgeschaltete Pipeline, die vier Takte zum Rechnen benötigt, vollständig gefüllt. Die Daten d_{in} enthalten neben den Pixelwerten auch noch die Blockadresse der Daten, so dass sich beim anschließenden Schreiben der Daten an jeder Stelle der Pipeline ohne weitere Berechnungen die Adresse ergibt. Diese Blockadresse wird vor Beginn der Berechnungen aus dem Datenstrom ausgekoppelt und durch die Register $R4$, R_5 und R_6 verzögert. Die Breite der Blockadresse berechnet sich zu:

$$B(S_{BlockAdr}) = \text{ceil}\,[\log_2[n]] \tag{5.151}$$

5.3.2 Kachelkorrelation

Bei der Kachelkorrelation werden die überlappenden Bereiche K_c der unabhängig optimierten Kachel K_a und K_b nach Abbildung 5.2.b betrachtet. Die Berechnung der Kachelkorrelation basiert dabei auf Gleichung 5.55:

$$Q_k = \sup\left[\left|\sum_{P_i \in K_c} (k_b(P_i) - k_a(P_i))\right|;\ n(K_c) - \left|\sum_{P_i \in K_c} (k_b(P_i) - k_a(P_i))\right|\right] \tag{5.152}$$

Die Schaltung nach Abbildung 5.21 berechnet die Werte Q_k und Q_g. Als Eingangsdaten erhält die Korrelationspipeline die Faltungsparameter $k_a(x, y)$ und $k_b(x, y)$ aller beteiligten Kacheln, im Zweierkomplement als vorzeichenbehaftete Binärzahlen:

$$k(x, y) = ks \cdot 2^1 + k \cdot 2^0 \tag{5.153}$$

Zunächst bilden die Subtrahierer SUB_1 bis SUB_n die Differenzen, die dann parallel paarweise durch die Addierer ADD_1 bis ADD_{n-1} aufsummiert Δc nach Gleichung 5.54 ergibt. Der Addierer ADD_n mit dem vorgeschalteten Negierer bildet das Zweierkomplement der Summe also $-\Delta c$. Anschließend entscheidet die Vorzeichenbit b_{VZ1} von Δc und b_{VZ2} von $-\Delta c$ welcher der Werte größer als null ist und damit übernommen wird. Danach entscheidet das Vorzeichenbit b_{VZ3} der Differenz der beiden Terme aus Gleichung 5.152, welcher Wert größer ist und damit

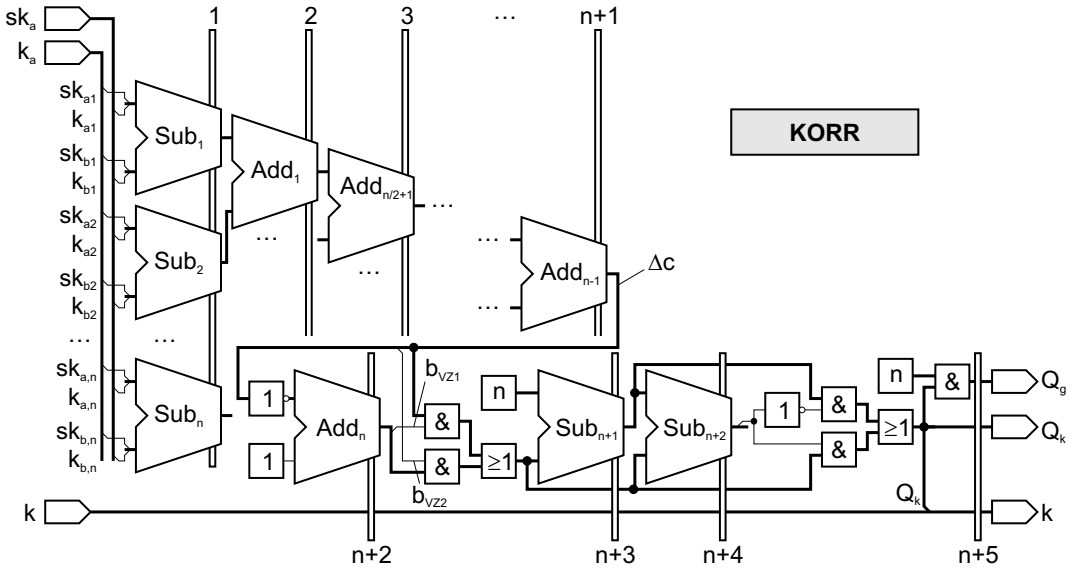


Abbildung 5.21: Berechnung der Kachelkorrelation Q_k

als Q_k übernommen wird. Ein Vergleich von Q_k mit dem Maximalwert der Korrelation n, der wegen 5.143 lediglich einer UND-Verknüpfung bedarf, ergibt Q_g. Die Busbreite Signals S_{sum} ergibt sich aus der Zahl der Pixel n je Kachel:

$$B(S_{sum}) = \text{ceil}\left[\log_2[2n]\right] \tag{5.154}$$

5.3.3 Kachelkonsistenz

Wie Abschnitt 5.3.3 erläutert, sucht dieser Teil des Algorithmus nach Residuen bei vier benachbarten Kacheln. Als Grundlage dient dabei die Kachelkonsistenz Q_r nach Gleichung 5.67. Das Modul berechnet zunächst die beiden Teildifferenzen $O_k(x_\square, y_\square) - O_k(x_\square, y_\square + 1)$ sowie $O_k(x_\square + 1, y_\square + 1) - O_k(x_\square + 1, y_\square)$. Der nächste Schritt summiert die Ergebnisse der beiden Terme zum Wert Q_r^* auf. Daraus ergibt sich durch eine *ODER*-Verknüpfung aller einzelnen Bits $S_{Q,0}$ bis $S_{Q,B-1}$ des Ergebnisses S_Q der Wert Q_r. Mit den Gleichungen 5.68-5.83 lassen sich damit die Barrieren der vier Kacheln sofort ermitteln. Das Ergebnis wird im Signal Q_{br} zusammengefasst.

5.3.4 Inkonsistente Lücken

Für die Berechnung der maskierten Pixel stehen prinzipiell die beiden Möglichkeiten aus Abschnitt 5.2.4 zur Verfügung. Bei der ersten Möglichkeit müssen die

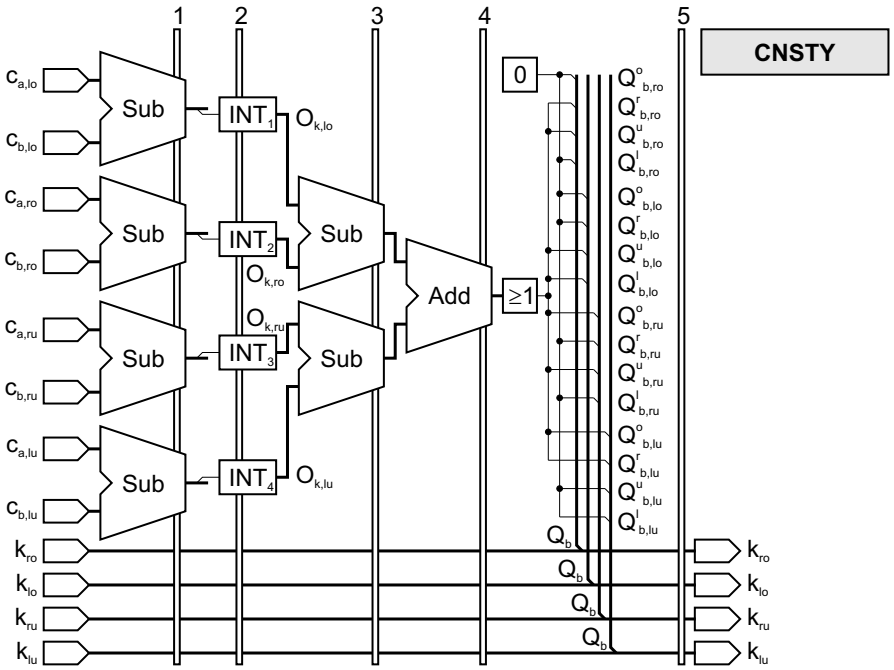


Abbildung 5.22: Berechnung der Kachelkonsistenz Q_k mit Barrieren

Kacheln rotiert werden, um dann mit der gleichen Logik $Q_{m,r}$ verarbeitet werden zu werden:

$$Q_{m,r} = Q_{m,1} \vee Q_{m,3} \tag{5.155}$$

Diese Variante kann dann auf zwei Arten erfolgen:

- Die Logik zur Berechnung ist einmal vorhanden. Dann spart dies Platz, die Pipeline benötigt jedoch jeweils vier Takte Rechenzeit, in der keine neuen Werte verarbeitet werden können.
- Die Logik zur Berechnung ist vier mal vorhanden. Dann ist die Platzersparnis jedoch aufgehoben. Die Rotation kann zwar ohne Rechenzeit durch ein einfaches verdrehen der Leitungen erreicht werden, die anschließende Logik benötigt jedoch für jede Drehung 13 Minterme, also insgesamt 52 Minterme.

Die zweite Variante ergibt sich aus der expliziten Betrachtung aller Möglichkeiten:

$$Q_{m,r} = Q_{m,1} \vee Q_{m,2} \vee Q_{m,3} \vee Q_{m,4} \vee Q_{m,5} \vee Q_{m,6} \tag{5.156}$$

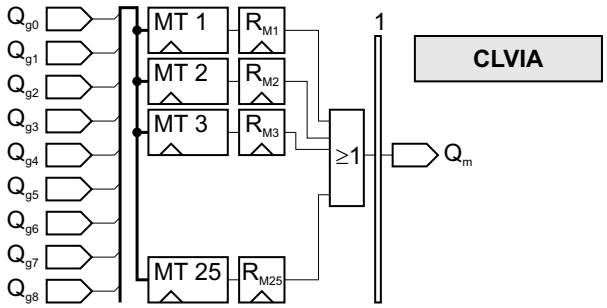


Abbildung 5.23: Berechnung der maskierten Pixel auf Basis von Residuen

Dies führt dann insgesamt zu 25 Mintermen. Aus diesem Grund wird diese Variante gegenüber der Variante mit Rotation bevorzugt. Abbildung 5.23 zeigt den Aufbau der Pipeline zum schließen der Lücken. Die Logik ist zweistufig aufgebaut, so dass die Grenzfrequenz der Schaltung nicht zu gering wird.

5.3.5 Entfalten

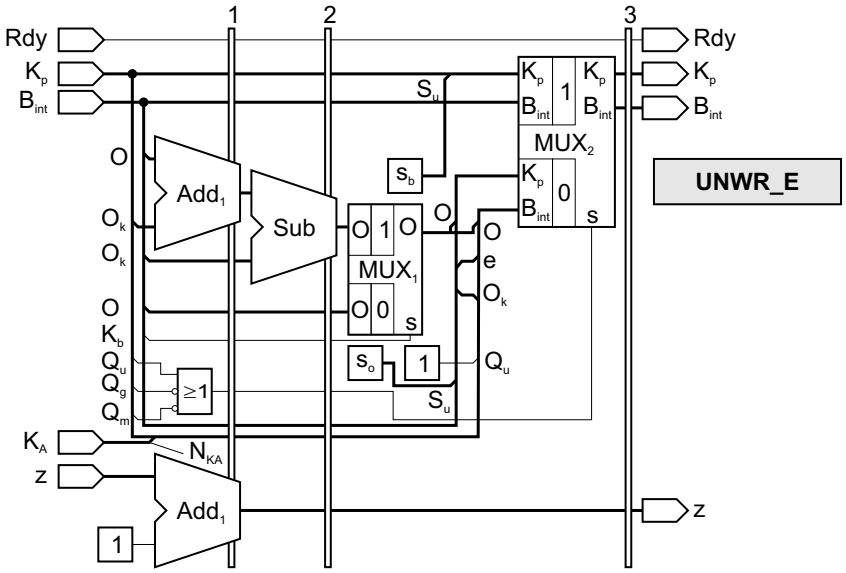


Abbildung 5.24: Eintritt die Kachel beim Entfalten

Den Algorithmus zur Entfaltung beschreibt bereits Abschnitt 5.2.5. Die Entfaltung in Form eines rekursiven Algorithmus würde dabei jedoch zu viele Ressour-

cen in Form eines Stacks benötigen. Dabei kann sich dann im ungünstigsten Fall der Speicherbedarf verdoppeln, wobei sich kein Geschwindigkeitsgewinn ergibt, da der Vorgang des Speicherns in jedem Fall durchgeführt werden muss. Deshalb werden alle notwendigen Informationen direkt in den Kachelinformationen K_p gespeichert. Das Signal K_p setzt sich aus dem Offset der Kachel O, dem Abstand O_k zwischen Kachel- und Klebeebene, der Kachelentfaltung Q_u, der binären Kachelqualität Q_g, der maskierten Kachel Q_m, dem Eintritt in die Kachel e_k sowie der Kachelposition $P(x,y)$ zusammen. Um zwischen den einzelnen Schritten des Algorithmus Daten austauschen zu können, dient das Signal B_{int}, welches aus dem Kacheloffset O, dem Abstand O_k zwischen Kachel- und Klebeebene, dem Status S_b des Entfaltungsalgorithmus sowie dem Wert K_b besteht. Für K_b gilt, analog zu Schritt zwei des Algorithmus:

$$O(K_c^*) = \begin{cases} 1 & \text{für Bedingung 5.109} \\ 0 & \text{sonst} \end{cases} \tag{5.157}$$

Die Initialisierung des Algorithmus übernimmt die übergeordnete Steuerung nach Abschnitt 6.
Abbildung 5.24 zeigt das Modul *UNWRE*, das die Berechnung des Kacheloffsets

Tabelle 5.1: Parametrierung des Moduls UNWR_X. Elemente sind je nach Variante vorhanden (v.) oder nicht vorhanden (n. v.)

	UNWR_O	*UNWR_R*	*UNWR_U*	*UNWR_L*
C_x	Sub	Add	Add	Sub
B_k	y	x	y	x
B_{ko}	y_o	x_o	y_o	x_o
Q_x	Q_b^o	Q_b^r	Q_b^u	Q_b^l
Neg_x	v.	n. v.	v.	n. v.
e_x	n. v.	n. v.	n. v.	v.
b_{mod2}	y_{mod2}	x_{mod2}	y_{mod2}	x_{mod2}
S_{x1}	S_r	S_u	S_l	S_b
S_{x2}	S_u	S_l	S_o	S_r
S_{x3}	S_o	S_r	S_u	S_l

nach den Punkten zwei und drei des Algorithmus durchführt. Der Multiplexer MUX_1 entscheidet über die Berechnung nach Gleichung 5.112. In der Variablen von z wird die Zahl der Kacheln eines Bereiches A berechnet. Der Multiplexer MUX_2 repräsentiert die Entscheidung nach Gleichung 5.108.
Das Modul *UNWR_X* nach Abbildung 5.25 beschreibt den Übergang in die Kacheln nach den Punkten vier bis sieben. Die Parametrierung zeigt Tabelle 5.1.

Der Negierer Neg_x sowie die Verbindung e_x sind je nach Variante vorhanden (v.) oder nicht vorhanden (n. v.).
Das Modul *UNWR_B* nach Abbildung 5.26 beschreibt den Schritt aus einer

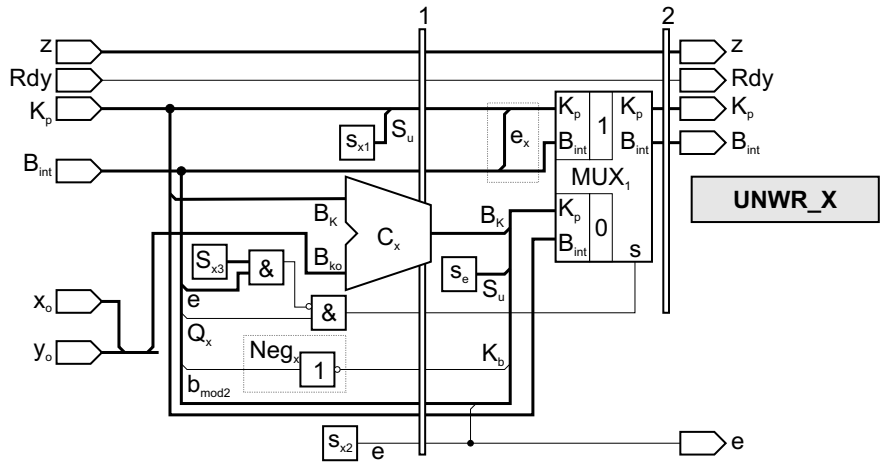


Abbildung 5.25: Suche der nächsten Kachel

Kachel zurück in die vorhergehende Kachel. Dabei wird abhängig vom Kacheleintritt e_k die vorhergegangene Kachelposition ermittelt. Falls keine weitere Kachel existiert, ist der Algorithmus am Ende angekommen und setzt das Signal St.
Die Zusammenschaltung der Module *UNWR_E*, *UNWR_O*, *UNWR_R*, *UNWR_U*, *UNWR_L* und *UNWR_B* nach Abbildung 5.26 bearbeitet eine Kachel, so dass nach Setzen des Signals S_t ein Bereich A vollständig entfaltet ist. Der Multiplexer *MUX* steuert die Ausgänge abhängig vom aktuellen Status S_u des Entfaltungsalgorithmus. Dieser wird zu Beginn aus dem Datenstrom ausgekoppelt und zwischengespeichert, da sich der Status innerhalb der Module ändert. Da die einzelnen Module unterschiedliche Laufzeiten besitzen, zeigt die Variable *Rdy* an, wenn das Modul seine Berechnungen abgeschlossen hat. Initial ist *Rdy* gleich eins und wird dann wie ein Token solange durch die einzelnen Module geschoben bis es am Ende wieder erscheint.
Das Modul *UNWRT* nach Abbildung 5.28 zeigt die Realisierung der Bereichsausdehnung. Dabei wird jeweils eine Kachel K_c mit den benachbarten Kacheln K_{co}, K_{cr}, K_{cu} und K_{cl} mit den dazugehörenden Kachelinformationen K_p betrachtet.

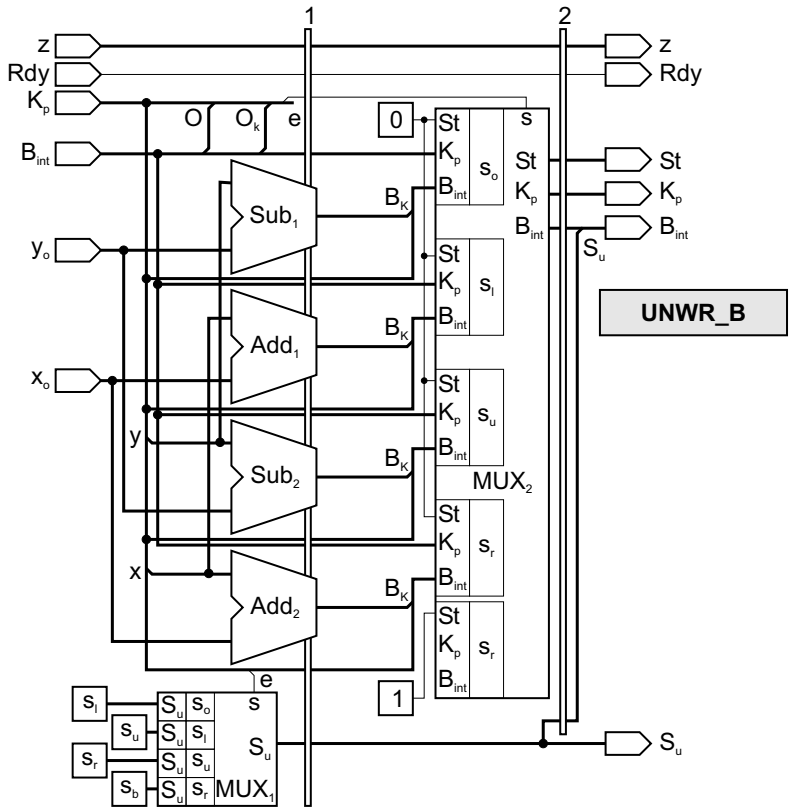


Abbildung 5.26: Weg zurück in die vorhergehende Kachel

Die Kachel K_c^* ergibt sich nach folgender Regel:

$$K_c^* = \begin{cases} K_{co} & \text{für} \quad Q_u(K_{co}) = 1 \\ K_{cr} & \text{für} \quad Q_u(K_{co}) = 0 \wedge Q_u(K_{cr}) = 1 \\ K_{cu} & \text{für} \quad Q_u(K_{co}) = 0 \wedge Q_u(K_{cr}) = 0 \wedge \\ & \qquad Q_u(K_{cu}) = 1 \\ K_{cl} & \text{für} \quad Q_u(K_{co}) = 0 \wedge Q_u(K_{cr}) = 0 \wedge \\ & \qquad Q_u(K_{cu}) = 0 \wedge Q_u(K_{cl}) = 1 \end{cases} \tag{5.158}$$

Für die Nummer des Bereiches gilt dann

$$N_a(K_c) = N_a(K_c^*) \tag{5.159}$$

Der Offset berechnet sich nach Gleichung 5.112.

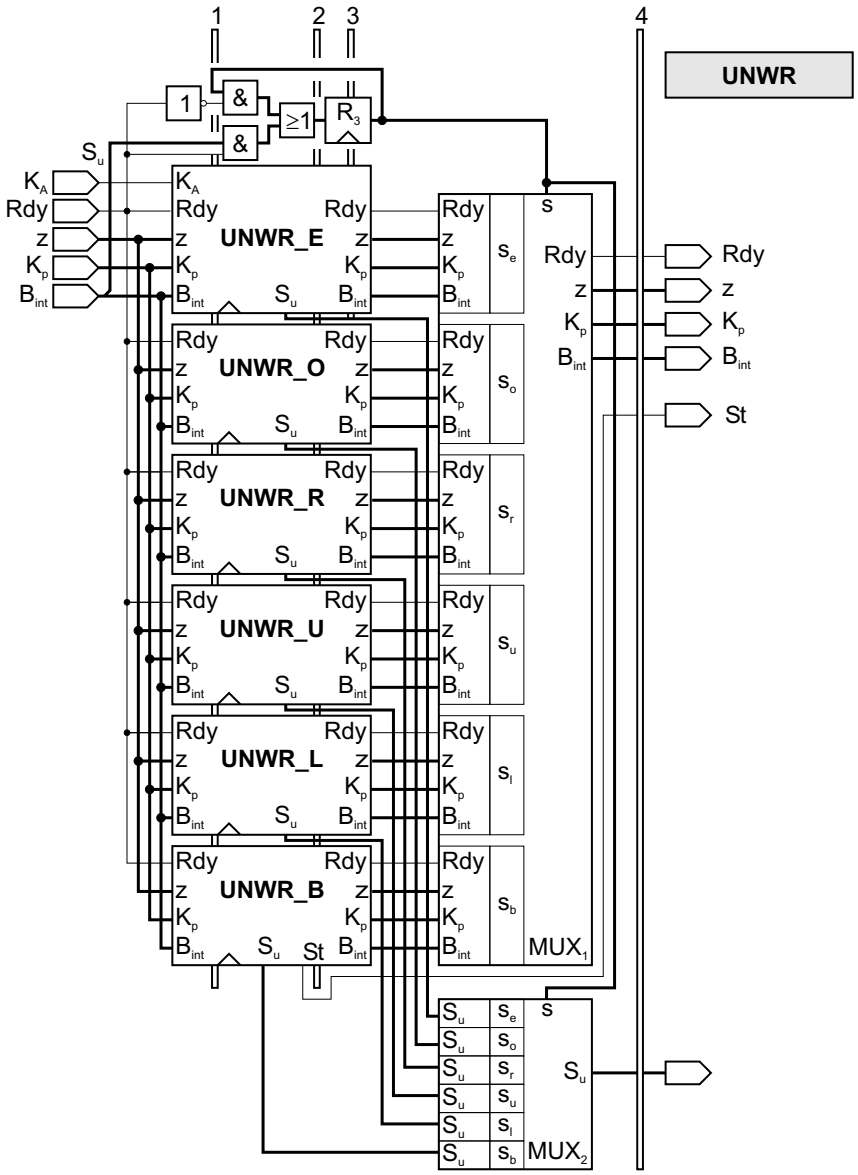


Abbildung 5.27: Kachelentfaltung

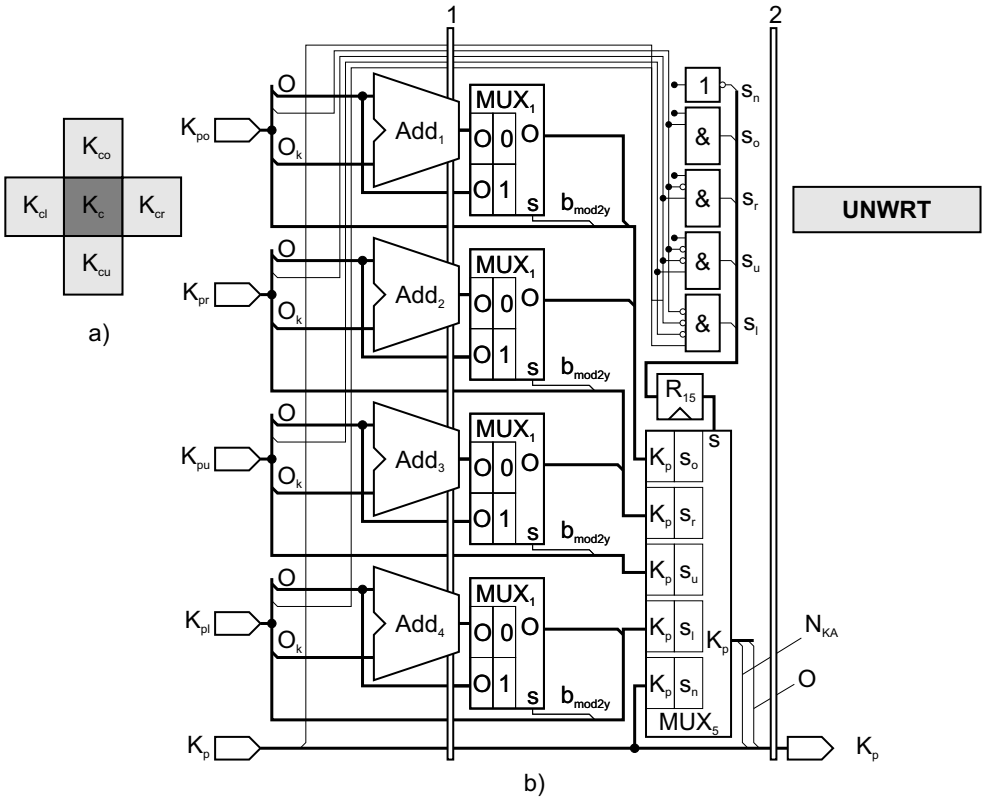


Abbildung 5.28: Bereichsausdehnung

5.3.6 Bereiche verbinden

Zum Verbinden der Bereiche dient die Bewertung ihrer Nachbarschaftsbeziehung nach Gleichung 5.134. Wie Abschnitt 6.7 zeigt, kann die Zahl der Bereiche sehr groß werden. Deshalb scheidet das Aufstellen einer vollständigen Nachbarschaftsmatrix aus. Zum Einen würde dies zu einem erheblichen Bedarf an Speicher führen, zum Anderen benötigt die Auswertung der Matrix viel Zeit. Effektiver ist da die Möglichkeit der Verwendung von Listen nach Abbildung 5.29. Dabei werden zu einem Bereich A_{c1} im Rahmen der Nachbarschaftsinformationen immer die Nachbarschaften zu Bereichen A_{c2} mit niedrigeren Nummern gespeichert:

$$N_a(A_{c1}) > N_a(A_{c2}) \tag{5.160}$$

Dadurch wird jede Nachbarschaftsbeziehung genau einmal repräsentiert. Für Bereiche, die keine Nachbarschaftsbeziehung besitzen wird somit kein Speicherplatz reserviert. Die Suche nach einer bestimmten Nachbarschaftsbeziehung zwischen zwei Bereichen A_{c1} und A_{c2} läuft dadurch in zwei Stufen ab:

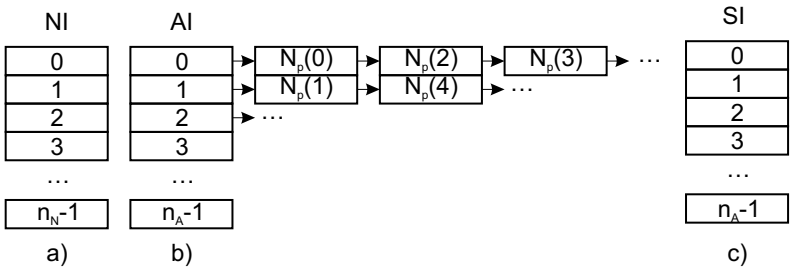


Abbildung 5.29: Speichermanagement Bereiche Zusammenfügen: a) Nachbarschaftsinformation N_p, b) Bereichsinformation A_p, c) Sortierinformation N_{pa}

1. Suche die größere Nummer $N_{a,max}$ der Bereiche und indiziere den Bereich.

$$N_{a,max} = \sup\left[N_a(A_{c1}), N_a(A_{c2})\right] \tag{5.161}$$

   Die Indizierung stellt eine Multiplikation dar. Durch die Wahl einer Zweierpotenz der Bereichsinformation A_p lässt sich die Multiplikation durch eine Shift-Operation ausführen.

2. Laufe entlang der verketteten Liste und suche die Stelle $N_a(A_{c2})$. Diese Suche wird linear und nicht binär [85] durchgeführt, da ansonsten die Nachbarschaftsinformationen zunächst nach Bereichsnummern sortiert werden müssten. Dieser Aufwand lohnt sich nicht in Anbetracht der Tatsache, dass die Zahl der Nachbarschaften eines Bereiches in Bezug auf eine binäre Suche als klein angenommen werden können, was Abschnitt 6.7 zeigt.

Beim Aufbauen der Liste muss diese Prozedur immer wieder durchlaufen werden. Falls die Suche erfolglos bleibt, wird die Liste um eine Nachbarschaftsbeziehung $N_p(n_N)$ erweitert, wobei n_N anschließend inkrementiert wird. Im Einzelnen basieren die Listen nach Abbildung 5.29 auf folgenden Speicherblöcken:

1. N_p beinhaltete die Informationen der Nachbarschaftsbeziehungen. Diese setzt sich zusammen aus den Nummern der angrenzenden Bereiche $N_a(A_{c1})$ und $N_a(A_{c2})$, den Qualitätskriterien $Q_n(N_k)$ und Q_A^{Σ}, der Zahl der Nachbarschaftskacheln $n_n(N_k)$ und einem Zeiger auf das nächste Element der Liste $N_{p,next}$.

2. A_p beschreibt die Parameter eines Bereiches, der sich aus der Nummer des Bereiches $N_A(A_{c1})$, der Zahl der Kacheln innerhalb des Bereiches $n_a(A_{c1})$, der Zahl der Nachbarn des Bereiches $n_{na}(A_{c1})$, dem Offset des Bereiches $O_a(A_{c1})$ und der ersten Nachbarschaftsbeziehung $N_{p,root}$ zusammensetzt.

Zusätzlich dienen die Zeiger $A_{p,next}$ und $A_{p,prev}$ dem Bilden einer verketteten Liste beim Zusammensetzen der Bereiche.

3. N_{ps} setzt sich aus Zeigern auf den Nachbarschaftsbeziehungen sowie dem Qualitätskriterium $Q_n(N_k)$ zusammen. Diese Liste wird sortiert und ergibt damit die Reihenfolge beim Zusammensetzen der Bereiche.

Zur Berechnung der Nachbarschaftsbeziehung wird das Bild einmal in horizontaler und einmal in vertikaler Ebene durchlaufen:

$$S_d = \begin{cases} 0 & \text{horizontale Laufrichtung} \\ 1 & \text{vertikale Laufrichtung} \end{cases} \tag{5.162}$$

Zunächst sucht der Algorithmus Kacheln K_{c1} und K_{c2}, die nach Bedingung 5.129

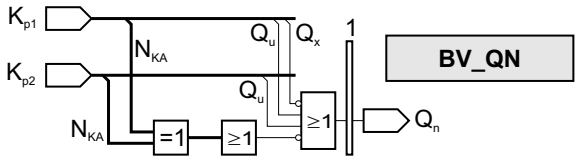


Abbildung 5.30: Suche benachbarter Kacheln

benachbart sind. Diese Aufgabe erfüllt das Modul *BV_QN* nach Abbildung 5.30. Zum Vergleich der beiden Bereichszugehörigkeiten wird zunächst bitweise die Antivalenz errechnet. Falls die beiden Zahlen $N_A(K_{c1})$ und $N_A(K_{c2})$ identisch sind, ergibt sich daraus jeweils als Ergebnis eine null. Dies kann mit Hilfe einer bitweisen *ODER*-Verknüpfung getestet werden, die in diesem Fall ebenfalls gleich null sein muss.
Darauf hin erfolgt die Berechnung des Offsets zwischen den beiden Kacheln durch das Modul *BV_OFF* nach Abbildung 5.31 analog zur Kachelentfaltung nach Gleichung 5.112. Das Objekt wird, wie Tabelle 5.2 zeigt, für die horizontale *BV_OFF_H* und die vertikale Richtung *BV_OFF_V* parametriert. Hinzu kommt die Sortierung der Bereichszugehörigkeiten $N_A(K_{c1})$ und $N_A(K_{c2})$ nach der Größe um damit die Speicherstruktur nach Abbildung 5.29 aufbauen zu können. Abhängig von der Sortierung ändert sich dann auch das Vorzeichenbit b_{VZ} des Offsets. Die Größe Q_f^{Σ} wird zur Summenbildung nach Gleichung 5.131 berechnet:

$$Q_f^{\Sigma} = Q_K(K_{c1}) + Q_K(K_{c2}) \tag{5.163}$$

Daraus ergibt sich zunächst der zu indizierende Bereich A_p aus dem Wert N_{KA1}. Danach sucht das Modul *BV_SN* in Abbildung 5.32 nach der Nachbarschaftsinformation N_p, deren Wert N_A gleich der Bereichsnummer N_{KA2} ist. Wird diese

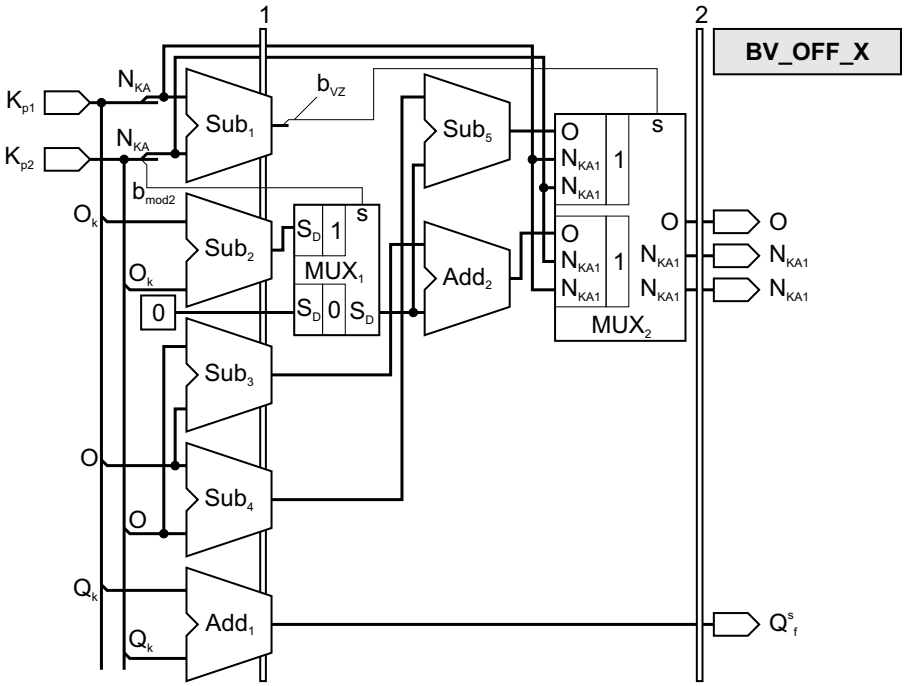


Abbildung 5.31: Berechnung des Bereichoffsets für eine Kachel

Tabelle 5.2: Parametrierung des Moduls BV_OFF_X

	BV_OFF_H	*BV_OFF_V*
b_{mod2}	x_{mod2}	y_{mod2}

gefunden, so wird das Signal S_F gesetzt. Im anderen Fall, wenn kein Nachfolger $Np, next$ mehr existiert, wird das Signal S_{iN} gesetzt. Damit kann der überlagerte Prozess eine neue Nachbarschaftsinformation aus N_p erzeugen und dem nächsten Schritt als Eingabe zur Verfügung stellen.
Das Modul *BV_CN* nach Abbildung 5.33 berechnet dann mit der Bereichsinformation A_p und der Nachbarschaftsinformation N_p die Summe aller Offsets zwischen den Bereichen, den Wert Q_f^{Σ} nach Gleichung 5.131 sowie die Zahl n_n der Kacheln, die in den beiden angrenzenden Bereichen nach Gleichung 5.129 benachbart sind.
Abbildung 5.34 zeigt das Modul *BV_AI*, welches die Zahl der Kacheln eines Bereiches sowie Q_a^{Σ} ermittelt. Da die Entfaltung in mehreren Schritten erfolgt, würde die direkte Berechnung dieser Informationen insbesondere beim umgekehrten Morphing eine erhebliche Zahl an unnötigen Speicherzugriffen erfordern. Des-

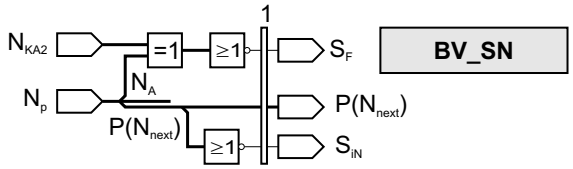


Abbildung 5.32: Suche der Nachbarschaftsinformation N_p

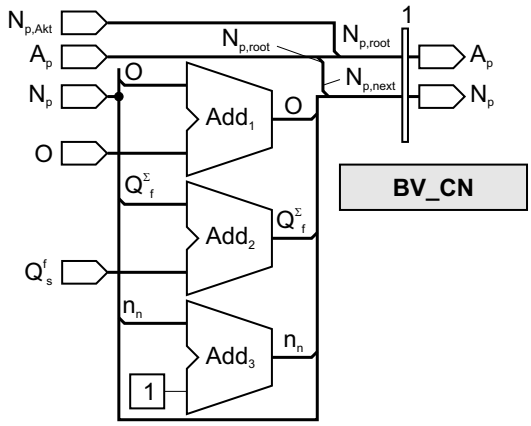


Abbildung 5.33: Berechnung von Q_f^Σ und n_n eines Bereiches A_c

halb bietet sich die Berechnung in einem gesonderten Schritt an.
Danach erfolgt die Berechnung des Qualitätskriteriums Q_n für alle Nachbarschaften N_p mit Hilfe des Moduls *BV_QN* nach Abbildung 5.35. Die Multiplikation und die Division [84] werden an dieser Stelle mit Pipelinemodulen realisiert. Die Latenzzeit des Moduls ist daher hoch.
Nachdem die Qualitätskriterien Q_n berechnet sind, müssen die Nachbarschaftsinformationen sortiert werden. Zum Sortieren im FPGA bieten sich insbesondere zwei Verfahren an:

1. Schneller Vergleich mit mehreren Registern: Für dieses Verfahren, das in verschiedenen Ausprägungen bezüglich Platzbedarf und Geschwindigkeit existiert, eignen sich ausschließlich kleine Listen, da das gesamte Feld im Speicher abgelegt werden muss. Im Sinne des Speichervorrats eines FPGAs ist die Zahl der möglichen Nachbarschaften bereits groß.

2. Sortieren mit zwei Speicherbereichen: Dieses Verfahren arbeitet langsamer, besitzt aber im Vergleich zu Bubble-Sort beispielsweise erhebliche Vorteile. Komplexe und schnelle Sortierverfahren, wie beispielsweise Quicksort

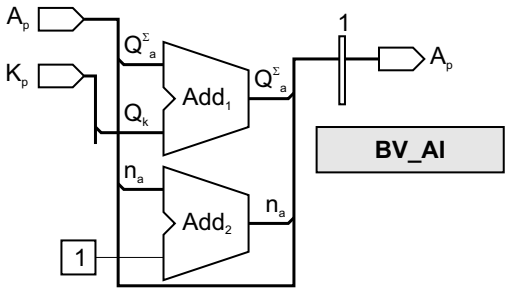


Abbildung 5.34: Berechnung der Zahl der Kacheln n_a und Q_a^Σ eines Bereiches

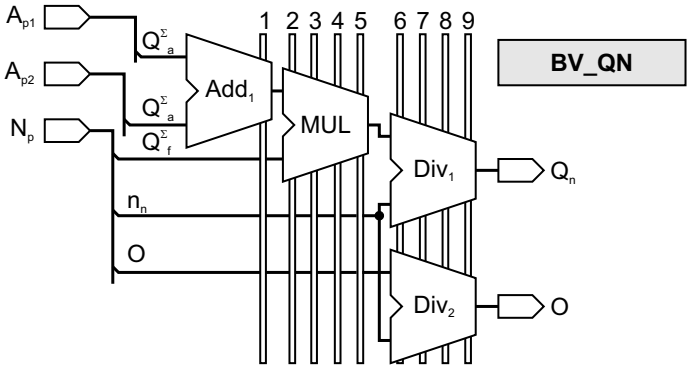


Abbildung 5.35: Berechnung des Qualitätskriteriums Q_n

eignen sich prinzipiell nicht für eine Implementierung im FPGA [86].

Der Algorithmus, dessen Prinzip Abbildung 5.36.a zeigt, liest die zu sortierenden Werte *w* nacheinander aus dem ersten Speicherbereich ein. Ist das LSB des Wertes *w* gleich null, so wird der Wert in den zweiten Speicherbereich geschrieben. Im nächsten Durchlauf wird der Wert *w* in den zweiten Speicherbereich geschrieben wenn das LSB gleich eins ist. Der nächste Schritt läuft analog ab, mit dem Unterschied dass nun das nächsthöherwertige Bit betrachtet wird. Der zweite Durchlauf lässt sich einsparen, indem der Algorithmus auf vier Speicherbereichen arbeitet. In Abhängigkeit vom Ergebnis des Bitvergleichs wandert der Wert *w* in unterschiedliche Speicherbereiche. Dabei muss jedoch die Zahl der Werte in beiden Speicherbereichen verwaltet werden. Abbildung 5.36 zeigt das Modul *BV_SORT*, das den Bitvergleich durchführt. Im Signal S_{vgl} ist nur das zu testende Bit b_{vgl} gesetzt. Der Vergleich entscheidet außerdem darüber, welcher Speicherzähler integriert wird. Somit entsteht die umgekehrt sortierte Liste L_{Qn}.
Im letzten Schritt müssen die Bereiche miteinander verbunden werden. Dies er-

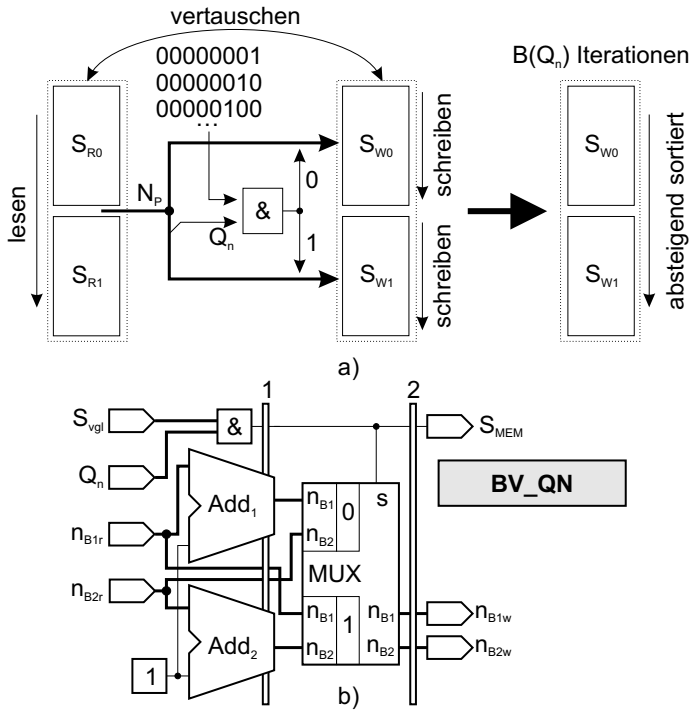


Abbildung 5.36: Sortieren mit zwei Speicherbereichen: a) Prinzip, b) Schaltplan

folgt in Form verketteter Listen. Die sortierte Liste des Qualitätskriteriums Q_n dient als Grundlage für die Erstellung der Listen. Beginnend mit dem größten Q_n sucht der Algorithmus aus dem Speicher die Nachbarschaftsinformation N_p heraus und ermittelt dadurch die beiden Bereichsinformationen A_{p1} und A_{p2}. Der Offset O_a der beiden Bereiche wurde bereits berechnet und ist in der Nachbarschaftsinformation enthalten. Die Bereiche werden dann über die Zeiger $A_{p,next}$ und $A_{p,prev}$ vorwärts und rückwärts miteinander verkettet. Abbildung 5.37 zeigt das Prinzip der verketteten Bereichslisten. Die Kette q_2 muss dann entsprechend dem sich ergebenden Offset angehoben werden, d. h. zu jedem Element wird der Offset O_a addiert. Ferner wird die Kette q_2 mit der Kette q_1 verschmolzen. Die Zahl der Ketten beträgt q_n. Der Algorithmus zur Verkettung läuft demnach in folgenden Schritten ab:

1. Suche aus der sortierten Liste L_{Qn} das größte noch nicht bearbeitete Element N_p heraus.

2. Ermittle mit Hilfe von N_p die beiden Bereiche A_{c1} und A_{c2}.

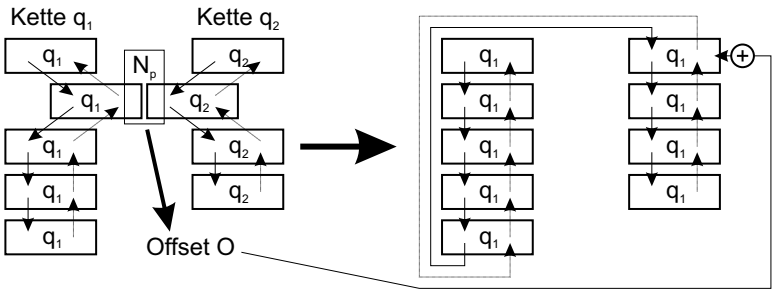


Abbildung 5.37: Prinzip des Verschmelzens zweier verketteter Bereiche q_1 und q_2

3. Wenn gilt:

$$q(A_{c1}) = q(A_{c2});\ q(A_{c1}) \neq 0 \tag{5.164}$$

   breche ab und fahre fort mit Schritt eins.

4. Laufe die Kette $q(A_{c1})$ entlang der Zeiger $A_{p1,next}$ bis zum Ende der Kette. Dies sei der Bereich $A_{end}(A_{c1})$.

5. Laufe die Kette $q_(A_{c2})$ entlang der Zeiger $A_{p2,prev}$ bis zum Beginn der Kette. Dies sei der Bereich $A_{beg}(A_{c2})$.

6. Ermittele ΔO_a:

$$\Delta O_a = \begin{cases} O(A_{c1}) - O(A_{c2}) + O_a(A_{c1}, A_{c2}) \text{ für } N_A(A_{c1}) \geq N_A(A_{c2}) \\ O(A_{c2}) - O(A_{c1}) - O_a(A_{c1}, A_{c2}) \text{ sonst} \end{cases} \tag{5.165}$$

7. Ermittle die Nummer der verschmolzenen Kette q_{akt}:

$$q_{akt} = \begin{cases} q_n & \text{für} \quad q(A_{c1}) = 0;\ q(A_{c2}) = 0 \\ q(A_{c2}) & \text{für} \quad q(A_{c1}) \neq 0;\ q(A_{c2}) = 0 \\ q(A_{c1}) & \text{sonst} \end{cases} \tag{5.166}$$

8. Inkrementiere die Zahl der Ketten:

$$q_n = q_n + 1 \quad \text{falls} \quad q(A_{c1}) = 0;\ q(A_{c2}) = 0 \tag{5.167}$$

9. Setze die Nummer der Kette für den Bereich A_{c1}:

$$q(A_{c1}) = q_{akt} \tag{5.168}$$

10. Verkette die beiden Bereiche A_{c1} miteinander, indem die Zeiger der Bereiche gesetzt werden:

$$P_{next}(A_{end}(A_{c1}) = P(A_{beg}(A_{c2})) \tag{5.169}$$
$$P_{prev}(A_{beg}(A_{c2}) = P(A_{end}(A_{c1})) \tag{5.170}$$

11. Subtrahiere vom ersten Element der Kette $q(A_{c2})$ den Offset ΔO.

$$O_a(A_{beg}(A_{c2})) = O_a(A_{beg}(A_{c2})) - \Delta O \tag{5.171}$$

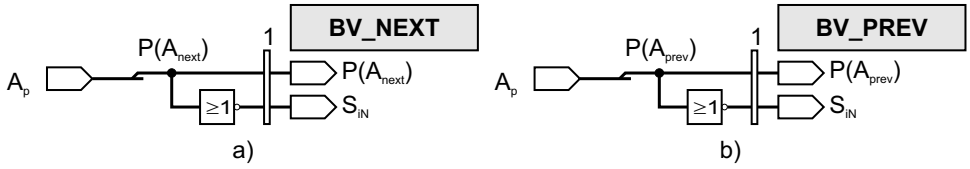


Abbildung 5.38: Laufen innerhalb einer verketten Liste. Vorwärts laufen: a) Schaltplan, b) Symbol, rückwärts laufen

Abbildung 5.38 zeigt die Module zum Laufen innerhalb der verketten Listen. Hierzu werden jeweils die entsprechenden Signale aus den Modulen ausgekoppelt und am Ausgang als Adresse des nächsten Lesevorgangs zur Verfügung gestellt. Falls der Anfang bzw. das Ende der Liste erreicht ist, wird das Signal S_{iN} gesetzt. Das Modul *BV_NEXT* realisiert den Punkt 4, das Modul *BV_PREV* den Punkt 5 des Algorithmus.
Abbildung 5.39 zeigt das Modul *BV_PUZOFF*, das den Schritt 11 des Algorithmus realisiert.
Abbildung 5.40 zeigt die Berechnung des Offsets nach Punkt 6. Die beiden verschiedenen Varianten werden parallel realisiert, während die Fallunterscheidung durch die Subtraktion in SUB_1 der beiden Bereichsnummern und das anschließende Auskoppeln des Vorzeichenbits b_{VZ} realisiert wird.
Wie sich die aktuelle Kettennummer q_{akt} nach Schritt 7 errechnet, entscheidet sich durch die Signale s_1, s_2 und s_3, die sich nach folgender Vorschrift errechnen:

$$s_1 = \begin{cases} 1 & \text{für} \quad q(A_{c1}) = 0 \wedge q(A_{c2}) = 0 \\ 0 & \text{sonst} \end{cases} \tag{5.172}$$

$$s_2 = \begin{cases} 1 & \text{für} \quad q(A_{c1}) \neq 0 \wedge q(A_{c2}) = 0 \\ 0 & \text{sonst} \end{cases} \tag{5.173}$$

$$s_3 = \begin{cases} 1 & \text{für} \quad s_1 = 0 \wedge s_2 = 0 \\ 0 & \text{sonst} \end{cases} \tag{5.174}$$

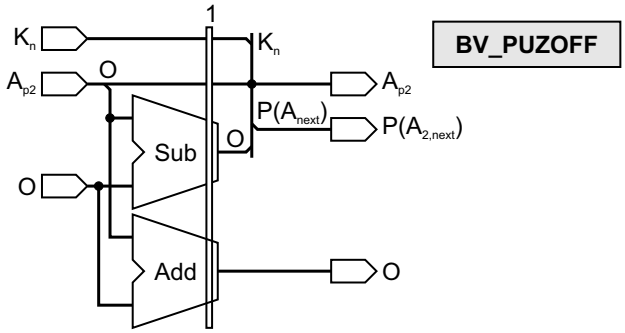


Abbildung 5.39: Berechnung der Parameter zum Verbinden zweier verketteter Listen

Das Bit b_{vkt} entspricht der Bedingung nach Gleichung 5.164:

$$b_{vkt} = \begin{cases} 1 & \text{für} \quad q(A_{c1}) = q(A_{c2}) \wedge q(A_{c1}) \neq 0 \\ 0 & \text{sonst} \end{cases} \tag{5.175}$$

Die Verkettung der beiden Listen nach Punkt 10 erfolgt durch das Einschleifen der Zeiger in die entsprechenden Ausgangssignale.

5.3.7 Pixelwerte berechnen

Um zum endgültigen entfalteten Bild zu kommen, müssen die Faltungskoeffizienten aller Pixel des Bildes berechnet werden. Beim Schritt der Kachelentfaltung wurden die Faltungskoeffizienten bereits zu den Pixeln der Kachel sowie der Klebeebene addiert. Die durch die Entfaltung der Kacheln und Bereiche zusätzlich berechneten Faltungskoeffizienten müssen in diesem Schritt zu den einzelnen Pixeln addiert werden:

$$\psi(P_i) = P_i + O(K_a(P_i)) + O(A_c(K_a(P_i))) \; \forall \; P_i \in K_a \tag{5.176}$$

Abbildung 5.41 zeigt das Modul $ENTFALT$, welches die Berechnung zur endgültigen Entfaltung des Pixels P_i durchführt.

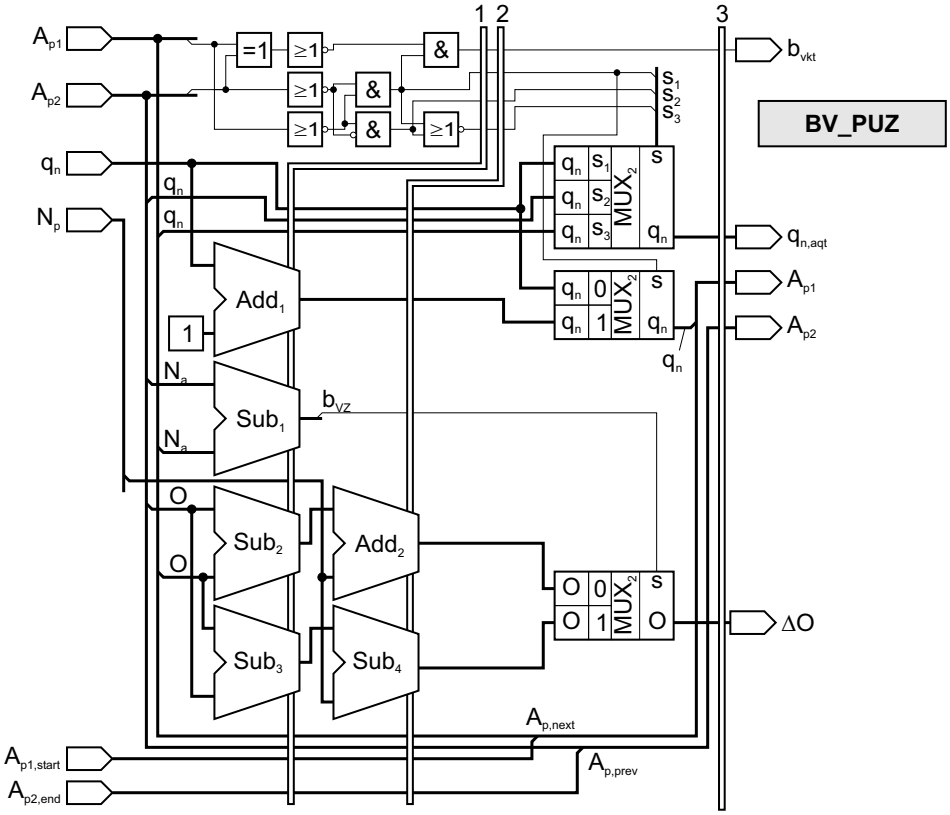


Abbildung 5.40: Berechnung von ΔO_a

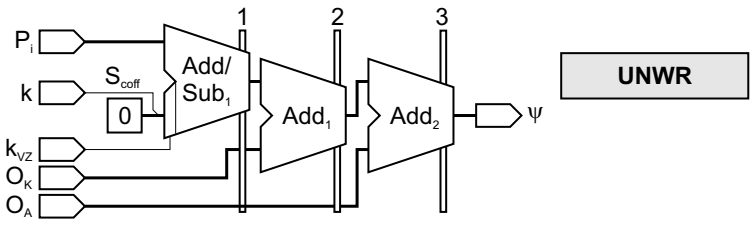


Abbildung 5.41: Berechnung des entfalteten Pixelwerts $\psi(P_i)$

Tabelle 5.3: Speicherobjekte

Kachelinformation (KI)	
Bereichsnummer	N_a
Offset	O
Kachelkorrelation	Q_k
Barrieren	Q_b
maskierte Kacheln	Q_m
entfaltete Kachel	Q_u
maximale Korrelation	Q_g
Bereichsinformation (AI)	
Bereichsnummer	N_a
Anzahl der Kacheln	n_a
Zahl der Nachbarschaften	n_{na}
Offset	O_a
Kettennummer	N_K
Zeiger zur ersten NI	$P(NI_{root})$
Verkettung nächste AI	$P(AI_{next})$
Verkettung vorhergehende AI	$P(AI_{prev})$
Nachbarschaftsinformation (NI)	
Bereichsnummer 1	$N_a(A_{c1})$
Bereichsnummer 2	$N_a(A_{c2})$
Nachbarschaftskorrelation	Q_f
Nachbarschaftsqualität	Q_n
Offset zwischen A_c1 und A_c2	O_n
Verkettung nächste NI	$P(NI_{next})$
NI schnelle Suche (NIS)	
Bereichsnummer 2	$N_a(A_{c2})$
Zeiger zu korrespondierer NI	$P(NI)$
Sortierinformation (SI)	
Nachbarschaftsqualität	Q_n
Zeiger zu korrespondierer NI	$P(NI)$
Faltungsinformation (FI)	
Faltungskoeffizient Kachelebene	k_a
Faltungskoeffizient Klebeebene	k_b
Maximale Korrelationsinformation (MKI)	
Maximale Korrelation	Q_g

Kapitel 6

Messsystem

Als Messgerät dient das Speckleinterferometer LAoS [2], das bereits Abschnitt 2.6 beschreibt. Die Vermessung einer Oberflächenform mit dem LAoS erfolgt in den folgenden Schritten. Die vorhergehenden Abschnitte beschreiben die einzelnen Schritte detailliert.

1. Aufnahme eines Specklebilds mit der Wellenlänge λ_1
2. Aufnahme eines Specklebilds mit der Wellenlänge λ_2
3. Berechnung des Phasenbilds P_ϕ
4. Entfaltung der Kachelebene K_a und der Klebeebene K_b
5. Berechnung der Korrelation der überlappenden Bereiche
6. Berechnung der Kachelkonsistenz
7. Berechnung der inkonsistenten Lücken
8. Entfalten der Kachelebene
9. Expandieren der Bereiche noch nicht entfalteter Kacheln
10. Berechnen der Nachbarschaftsinformationen (NI)
11. Berechnen der Bereichsinformationen (AI)
12. Berechnen der Qualitätskriterien Q_n
13. Sortieren der NI nach Q_n
14. Verketten der Bereiche in der Reihenfolge der sortierten Nachbarschaften

15. Berechnung des Höhenwerts jedes Bildpunkts
16. Speichern der Höhenwerte
17. Visualisieren der Höhenwerte

Bisher arbeitet das LAoS mit mechanischen Verschlüssen, die als bewegte Elemente Schwingungen in das Messsystem einkoppeln. Dies führt dazu, dass zwischen der Bewegung des Verschlusses und der Aufnahme eines Specklebilds mindestens 200 ms liegen müssen. Die Messrate des Gerätes ist damit, auch bei einer schnelleren Kamera, auf etwa zwei Messungen pro Sekunde beschränkt. Die Einstellung der Messgenauigkeit erfolgt dabei durch die manuelle Justierung der Wellenlängen der Laserdioden. Abschnitt 6.1 beschreibt, wie ohne bewegte Verschlüsse geschaltet werden kann und eine automatische Messbereichseinstellung erfolgt. Als Aufnahmegerät dienen zwei unterschiedliche CCD-Kameras, deren Ansteuerung Abschnitt 6.2 zeigt. Mit der Datenauswertung befasst sich Abschnitt 6.3. Dabei werden die in Abschnitt 4 und 5 beschriebenen Hardwaremodule zu einer Pipeline verschaltet. Untersucht werden zwei Konfigurationen für zwei verschiedene CCD-Kameras. Diese werden den Ergebnissen, gewonnen mit Hilfe eines handelsüblichen PCs, gegenübergestellt. Abschnitt 6.4 beschreibt wie die Daten vom FPGA zu einem PC oder ins Datennetz gelangen. Abschnitt 6.5 beschreibt ein plattformunabhängiges Visualisierungstool der Daten. Abschnitt 6.6 zeigt dann das Zusammenspiel aller Komponenten und eine Übersicht des Gesamtsystems. Zum Schluss belegen Messdaten in Abschnitt 6.7 die Leistungsfähigkeit des Gesamtsystems.

6.1 Beleuchtung

Wie dies bereits die Einleitung und Abschnitt 2 beschreiben, können Speckleinterferometer mit Laserdioden [2] arbeiten. Die Streifenempfindlichkeit bei einem Interferometer das mit zwei Wellenlängen arbeitet hängt dabei nach Gleichung 2.8 von der Frequenz der eingesetzten Laser ab. Für den Einsatz im Interferometer muss das Laserlicht folgende Bedingungen erfüllen. Es muss

- kohärent und
- Schmalbandig sein.

Für den industriellen Einsatz sind zusätzlich folgende Eigenschaften wichtig:

- Ausreichende Lichtleistung bzw. Lichtintensität

- Variable Wellenlängen und damit variable Höhenauflösung
- Schnelle Modulation um auf bewegte Verschlüsse verzichten zu können
- Langzeitstabilität
- In-situ Fehlererkennung
- Kalibrierung im System

Um diese Anforderungen erfüllen zu können, bietet sich der Einsatz von Laserdioden an. Um Laserdioden kohärent und schmalbandig betreiben zu können sind jedoch eine Reihe von Maßnahmen notwendig. Der folgende Abschnitt 6.1.1 erläutert deshalb die Arbeitsweise von Laserdioden.
Prinzipiell besteht auch die Möglichkeit mit Hilfe optischer Rückkopplungen durch externe, abstimmbare Resonatoren schmale Verstärkungskurven zu erzeugen [91]. Dies hat jedoch den Nachteil, dass zum Einen die Laser bereits auf das Resonatorsystem zugeschnitten sein müssen um nicht zu hohe Verluste zu erzeugen. Zum Anderen ist eine Adaption des Systems an neue Wellenlängen stets mit einer manuellen Modifikation verbunden. Die Betrachtung solcher Systeme scheidet deshalb im Folgenden aus.

6.1.1 Theoretische Grundlagen

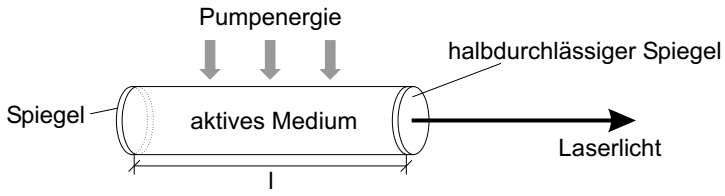


Abbildung 6.1: Prinzipieller Aufbau eines Halbleiterlasers

Prinzipiell kann die Laserdiode als Lichtoszillator beschrieben werden, der einen Verstärker, das aktive Medium und eine Rückkopplung besitzt. Zwei begrenzende Spiegel realisieren die Rückkopplung indem sie einen Resonator bilden. Ein laufendes Photon im aktiven Medium erzeugt idealerweise durch stimulierte Emission weitere Photonen gleicher Phase und einer definierten Wellenlänge Λ. Der Abstand der Spiegel, die Cavity-Länge l, bestimmt dabei die möglichen longitudinalen Moden des Resonators. Diese entsprechen der Bedingung, dass die optische Länge L eines kompletten Durchlaufs durch den Resonator ein ganzzahliges Vielfaches der Wellenlänge λ sein muss.

$$L = 2 \cdot n(\vartheta, I) \cdot l = i\lambda \quad \text{mit} \quad i = 1, 2, 3, ... \tag{6.1}$$

Dabei entspricht l der Resonatorlänge und n dem Brechungsindex, der von dem Pumpstrom I sowie der Temperatur ϑ abhängt. Ein solcher Aufbau eines Halbleiterlasers wird als Fabry-Perot-Laser bezeichnet.
Da die angeregten Energiezustände eine endliche Lebensdauer besitzen, verbrei-

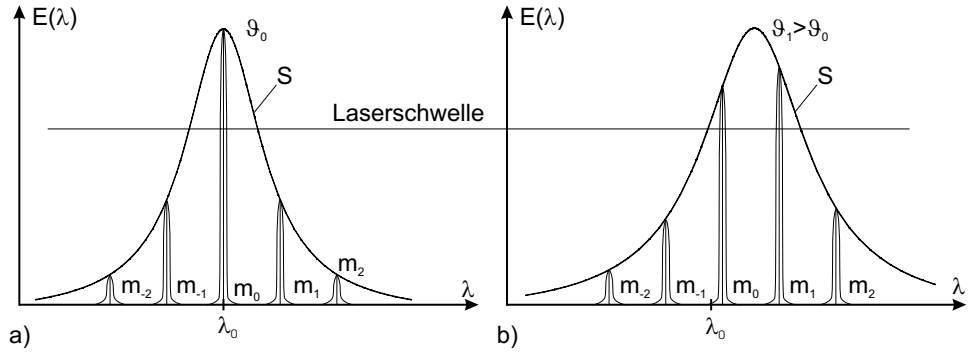


Abbildung 6.2: Auswirkungen einer Temperaturänderung auf das emittierte Spektrum einer Laserdiode [94]: a) Temperatur ϑ_0, b) Temperatur $\vartheta_1 > \vartheta_0$

tert sich das Verstärkungsprofil im Frequenzbereich. Diese Verstärkungskurve S heißt natürliche Linienbreite und besitzt die Form des Lorentz-Profils, das noch zusätzlich vor allem durch thermische Prozesse verbreitert wird. Alle longitudinalen Moden, deren Frequenz innerhalb dieser Kurve liegen, finden sich daher auch im emittierten Spektrum des Lasers wieder. Ein weiterer Effekt, der zur Linienverbreiterung beiträgt ist der Dopplereffekt. Um solche Abhängigkeiten zu minimieren und eine Auswahl der lasenden Frequenzen zu treffen, erhalten die sogenannten Distributed Feedback Laser (DFB) periodische Strukturen [90]. Durch die damit erzeugten verteilten Reflexionen und Brechungen bestimmt die geometrische Struktur die Ausbildung der stehenden Welle entscheidend mit.
Die Temperatur hat somit einen weitreichenden Einfluss auf die Abstrahlcharakteristik der Laserdiode. Bei wachsender Temperatur verändern sich insbesondere folgende Parameter:

- Verlängerung der Resonatorlänge l
- Wachsen der Brechzahl durch die Erhöhung der Gitterschwingung
- Verschiebung der Verstärkungskurve durch abnehmende Bandlücke bei wachsender Temperatur
- Lorentz-Verbreiterung
- Doppler-Verbreiterung

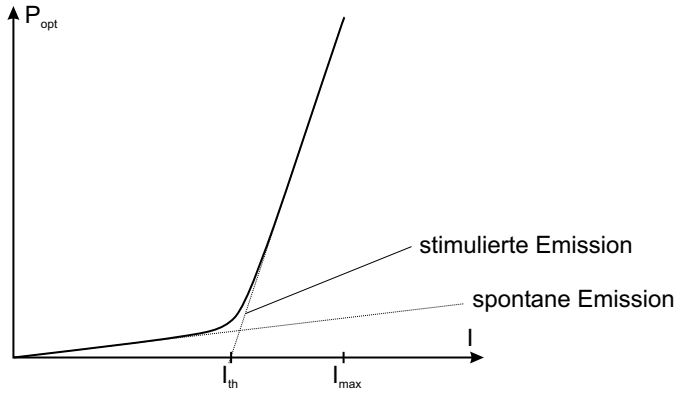


Abbildung 6.3: Zusammenhang zwischen dem Diodenstrom I und der emittierten Leistung P_{opt}

Die Auswirkungen dieser Temperatureffekte sind:

- Verschiebung der Moden zu höheren Wellenlängen
- Verschiebung der Verstärkungskurve zu höheren Wellenlängen
- Mehr- oder Multimodigkeit
- Modensprünge oder Mode-Hopping [92]

Abbildung 6.2 zeigt, wie sich bei einer Temperaturerhöhung von ϑ_0 auf ϑ_1 die Verstärkungskurve S verbreitert und verschiebt. Die Moden m selbst verschieben sich jedoch nicht so stark wie die Verstärkungskurve. Dies hat zur Folge, dass nach der Temperaturerhöhung der Laser multimodig arbeitet. Ebenso ändern sich die Laserdiodenparameter durch Alterungsprozesse. Die Lebensdauer einer Laserdiode halbiert sich beim Betrieb mit einer Temperatur von $\vartheta = 30°C$ [94].
Das Licht wird im Halbleiter innerhalb eines aktiven Bereichs in der sogenannten Inversionszone verstärkt. Das energetisch höher gelegene ist dabei stärker besetzt als das energetisch tiefer liegende Niveau. Eine solche Inversion wird erreicht, wenn mindestens eine der Ladungsträgersorten in entarteter Konzentration vorliegt.
In der Praxis bestehen Laserdioden im einfachsten Fall aus zwei aneinander grenzenden Bereichen des gleichen Grundmaterials, die entartet p-dotiert bzw. entartet n-dotiert sind. Legt man über der Sperrschicht der entstandenen Diode eine äußere Spannung in Vorwärtsrichtung an, so fließt ein Strom durch den Halbleiter, der Minoritätsladungsträger nachliefert. Die aktive Schicht bildet sich wegen der höheren Beweglichkeit der Elektronen fast ausschließlich im p-Bereich

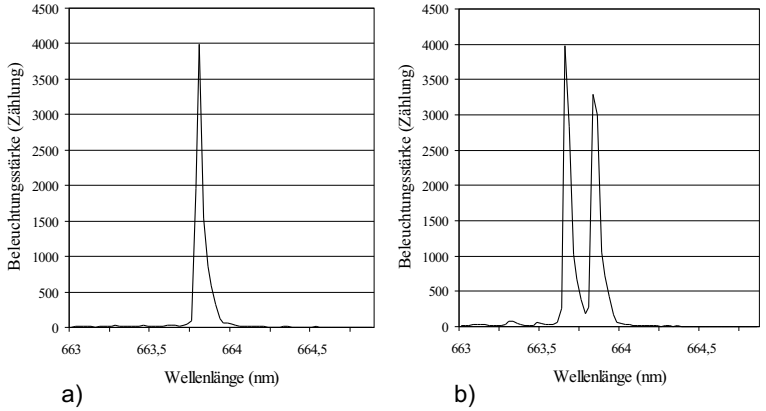


Abbildung 6.4: Vergleich des Spektrums einer Laserdiode beim Betrieb mit einer Temperatur von a) $\vartheta = 20°C$ und b) von $\vartheta = 24°C$

aus. Ein Aufrechterhalten der Inversion in der aktiven Zone ist nur bei genügender Nachlieferung an Ladungsträgern möglich. Ansonsten käme es nur zu spontaner, strahlender sowie nicht-strahlender Rekombination, nicht aber zu stimulierter Emission. Mit zunehmendem Pumpstrom spielt die spontane Emission, die sogenannte LED-Strahlung, gegenüber der stimulierten Emission keine Rolle mehr. Es gibt abhängig von Material, Temperatur und Bauart der Laserdiode einen Schwellstrom I_{th}, ab dem eine optische Verstärkung erst möglich wird. Darüber bestimmt die Stromstärke des Pumpstroms I den Grad der Verstärkung im aktiven Medium und somit die Ausgangsleistung P_{opt} des Lasers. Den Zusammenhang zwischen dem Pumpstrom I und der emittierten Energie P_{opt} zeigt Abbildung 6.3. Der Pumpstrom hat aber neben seinem Einfluss auf die Ausgangsleistung der Diode noch einen weiteren Effekt: Eine Änderung des Betriebsstromes bewirkt eine Veränderung der Inversionsdichte. Die Inversionsdichte bestimmt auch den Brechungsindex des Halbleiters. Das heißt über den Pumpstrom kann man auch die optische Weglänge und damit die Eigenschaften des Resonators verändern. Das führt dazu, dass es bei ungenügender Stabilisierung des Pumpstromes zu einer mehrmodigen und damit nicht-kohärenten Abstrahlcharakteristik kommt. Oberhalb der Schwellstromdichte erfolgt jedoch keine wesentliche Änderung der Ladungsträgerkonzentration. Die Wellenlängenabhängigkeit vom Injektionsstrom resultiert nur aus der $I^2 \cdot R$ - Erwärmung des Kristalls [93].
Darüber hinaus ist es wichtig den Arbeitsbereich der Laserdioden einzuhalten und die empfindlichen Bauteile vor Strom- und Spannungsspitzen zu schützen. Laserdioden werden vom Hersteller immer mit ihrem maximalen Betriebsstrom I_{max} gekennzeichnet. Das heißt, die Dioden können in einem Strombereich von I_{th} bis

I_{max} mit dem Betriebsstrom I_{op} betrieben werden. Ein sicherer Betrieb fordert indes einen oberen Schutzbereich von etwa 10%.
Die Abhängigkeit des Emissionsspektrums von Strom und Temperatur kann in ei-

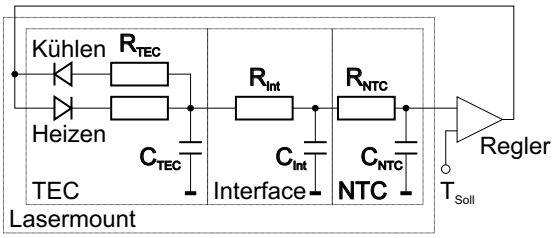


Abbildung 6.5: Thermische Netzwerk der Laserhalterung

ner Stabilitätskarte [8] dargestellt werden. Das eigentliche Maß für die Stabilität ist dabei die Kohärenzlänge des Lichts. Dieses Maß ist jedoch nur unter Laborbedingungen und unter erheblichem Aufwand zu ermitteln, so dass dieses Maß für eine Online-Bewertung der Laserdiode ausscheidet [98]. Dagegen hat sich als Maß für die Qualität der Laserdiode der Abstand Signal- zu Rauschabstand SNR zwischen der Hauptmode m_0 und der nächstgrößten Nebenmode m_1 als brauchbar herausgestellt:

$$SNR = \frac{I(m_0)}{I(m_1)} \tag{6.2}$$

Eine weitere Möglichkeit würde noch darin bestehen, die Breite des Hauptmaximums in das Gütemaß mit einzubeziehen. Es hat sich jedoch gezeigt, dass bei ausreichendem Signal- zu Rauschabstand die Messung stets durchführbar ist.
Um Halbleiter-Laserdioden in der Speckleinterferometrie sinnvoll einsetzen zu können, muss ein Mess- und Regelsystem folgende Aufgaben erfüllen:

- Betrieb der Laserdioden in einem stabilen und definierten Arbeitspunkt
- Automatische Aufnahme der Stabiltätskarte im Speckleinterferometer
- Überprüfung der Laserwellenlänge und Beurteilung der Stabilität während jeder Messung

Die folgenden Abschnitte beschäftigen sich mit der Regelung der beiden Parameter Strom und Temperatur zum Betrieb der Laserdiode.

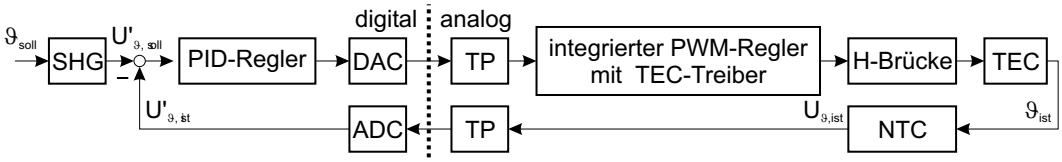


Abbildung 6.6: Kaskadenregelkreis zur Temperaturregelung mit Hilfe eines TECs.

6.1.2 Temperaturregelung

Die Laserdiode befindet sich im Betrieb in einer Laserhalterung. Thermisch betrachtet lässt sich diese in drei Teile aufspalten:

1. Wärmesenke oder Kühlkörper
2. Peltierelement
3. thermische Last

Das Peltierelement befindet sich dabei zwischen thermischer Last und der Wärmesenke. Die thermische Last besteht aus der Laserdiode, deren Halterung und dem in unmittelbarer Nähe der Laserdiode angebrachten Temperatursensor. Die Größe der thermischen Last stellt insgesamt einen Kompromiss zwischen schneller Reaktionsfähigkeit und einer mit der Trägheit des Systems einhergehenden Stabilität dar. Die Reaktionsfähigkeit des Systems wird vom Peltierelement (TEC), dem Sensor (NTC) sowie dem Interface (INT) zwischen Peltierelement und Sensor bestimmt. Alle Teile bestehen aus einem thermischen Widerstand R und einer thermischen Kapazität C. Die Zeitkonstante τ_{ges} des thermischen Netzwerks, das Abbildung 6.5 zeigt, ergibt sich damit zu:

$$\tau_{ges} = (R_{TEC} \cdot C_{TEC}) \cdot (R_{INT} \cdot C_{INT}) \cdot (R_{NTC} \cdot C_{NTC}) \tag{6.3}$$

Da das System drei Energiespeicher beinhaltet, kann die Strecke als PT_3-System modelliert werden. Aufgrund der geringen thermischen Masse des Sensors lässt sich dessen Verhalten und das der Halterung gemeinsam beschreiben. Aus diesem Grund ergibt sich insgesamt ein PT_2-System, wobei die Zeitkonstante des TEC und des Interfaces wegen ihrer großen Differenz zu einer gemeinsamen Block zusammengefasst werden können. Bei der Betrachtung der Sprungantwort des Systems ergibt sich, dass es sich um ein stabiles, aperiodisches System höherer Ordnung handelt. Daraus ergibt sich dann die Möglichkeit zur Streckenidentifikation das Verfahren von Strejc oder Radtke [97, 96] anzuwenden. Daraus ergibt sich die Übertragungsfunktion des Systems zu [100]:

$$G(s) = \frac{K_p}{1 + T_E \cdot s} = \frac{18}{1 + 43,1 \cdot s} \tag{6.4}$$

Die Regelung des Systems erfolgt mit Hilfe eines Kaskadenreglers nach Abbildung 6.6. Dieser besteht zum Einem aus einem integrierten analogen PWM-Regler mit der Ansteuerung für die H-Brücke, die den TEC treibt. Zum Anderen aus einem überlagerten digitalen PID-Regler, der als Führungsregler die bleibende Regeldifferenz minimiert [95] und nach folgender Rechenvorschrift arbeitet:

$$y_i = K_p \left[e_i + \frac{T_a}{T_N} \sum_{j=1}^{i} e_j + \frac{T_V}{T_A}(e_i - e_{i-1}) \right] \tag{6.5}$$

Die Sollspannung $U'_{\vartheta,soll}$ ergibt aus der Steinhartgleichung, die den Zusammenhang zwischen der Temperatur und dem Widerstand beschreibt [94]. Bei diesem Algorithmus sind bei der Berechnung die Stoßfreiheit und der integrale Anfahreffekt zu beachten. Die Maßnahme des Anti-Wind-Up verhindert ein starkes Überschwingen beim Anfahren des Systems [100].
Der unterlagerte analoge Regler besteht im Wesentlichen aus einem integrierten

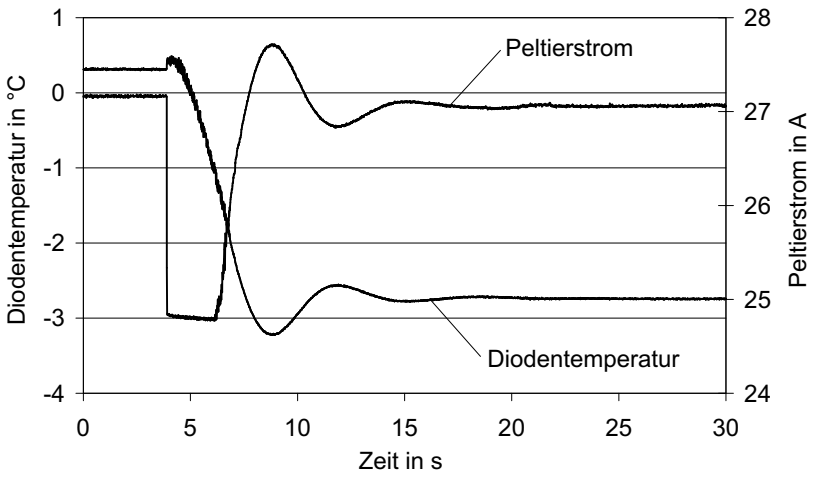


Abbildung 6.7: Einschwingverhalten der geschalteten Temperaturregelung

Chip mit wenig externer Beschaltung [101]. Abbildung 6.7 zeigt das Einschwingverhalten des Reglers bei Änderung der Solltemperatur von $27,5\,°C$ auf $25\,°C$. Dabei ist festzustellen, dass der Regler zunächst in die Sättigung fährt und einen maximalen Strom von -3 A liefert. Während dieser Zeit ist das Temperatursignal stark verrauscht. Wenn jedoch der Betrag des Stroms unter $1,5$ A sinkt, dann arbeitet der Regler wieder rauschfrei. Da während der Einschwingphase ohnehin keine Messungen möglich sind, stört dieses Verhalten nicht weiter. Die Parametrierung des digitalen Reglers erfolgt nach dem experimentellen Verfahren von Ziegler-Nichols [95].

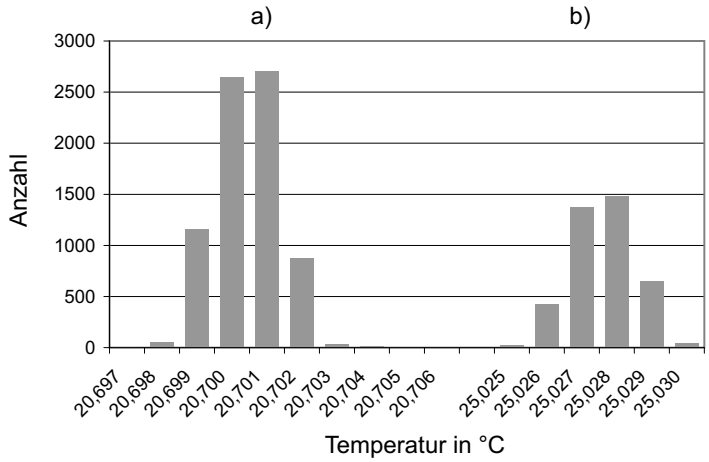


Abbildung 6.8: Sollwertabweichung bei eingeschwungener Regelung für a) $\vartheta_{soll} = 20\ °C$, *b)* $\vartheta_{soll} = 25\ °C$

Die hohe Präzision des Reglers dokumentieren die Regelversuche auf 20 °C sowie 25 °C, deren Ergebnis Abbildung 6.8 zeigt. Im einzelnen ergeben sich für den Mittelwert m und die Standardabweichung σ die Werte in Tabelle 6.1. Die hohe stationäre Abweichung ΔI_{stat} kann durch einen größeren I-Anteil im Führungsregler minimiert werden, was den Regler jedoch langsamer macht. Die absolute Genauigkeit spielt für die Regelung keine entscheidende Rolle. Vielmehr ist die hohe Wiederholgenauigkeit, die sich in der geringen Standardabweichung niederschlägt, von großer Bedeutung, da die Diode im System vermessen wird. Damit ist lediglich das präzise Erreichen der Arbeitspunkte wichtig.

Tabelle 6.1: Regelversuche mit dem kaskadierten Temperaturregler

ϑ_{soll}	m	σ	ΔI_{stat}	$\Delta I_{stat,rel}$
20 °C	20,70145 K	1,03 mK	−0,7015 K	−3,507%
25 °C	25,02860 K	0,94 mK	−0,0286 K	−0,114%

6.1.3 Stromregelung

Für die Stabilität der Laserdiode ist es wichtig, einen stabilen Pumpstrom zu liefern. Außerdem soll die Diode dunkel geschaltet werden können. Der Sollwert des Stroms soll mit einer Toleranz von weniger als 0,5 % angefahren werden, die Stabilität des eingestellten Wertes darf jedoch 0,05 % nicht überschreiten,

da die Grenzfrequenz von Standardlaserdioden oberhalb einiger 100 MHz liegt. Insbesondere bei Schaltreglern ist deshalb eine effektive Filterung notwendig. Der Ausgangsstrom des Treibers soll bei maximal 250 mA liegen. Der Spannungsabfall über der Laserdiode liegt typischerweise zwischen 2, 2 und 2, 6 V. Um die Laserdiode nicht zu zerstören, ist es außerdem wichtig, dafür zu sorgen, den maximalen Strom der Laserdiode nicht zu überschreiten.
Das Modell der Regelstrecke der Stromregelung entspricht wegen der extrem

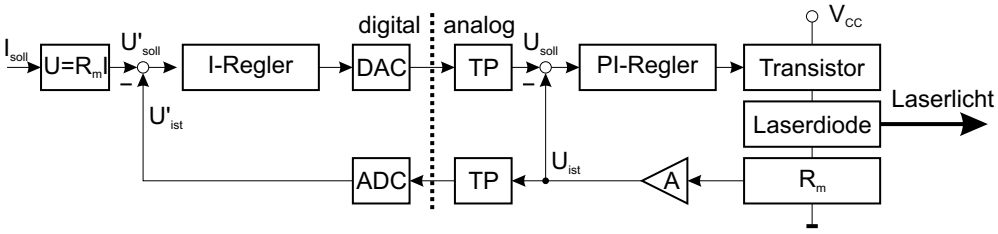


Abbildung 6.9: Kaskadenregelkreis zur Stromregelung.

kleinen Zeitkonstanten einem Proportionalglied. Die Einstellung der Regelparameter erfolgt deshalb experimentell [95]. Bei der Stromregelung wird kein Kaskadenregler im klassischen Sinn verwendet, da beide Strecken einen gemeinsamen Sensor besitzen. Da die AD-Wandlung im Sensorzweig jedoch wie ein Verzögerungsglied wirkt, unterscheiden sich beide Messgrößen zum Zeitpunkt der Regelung. Somit lassen sich alle Vorzüge der Kaskadenregelung nutzen [96]. Insbesondere verbessert der Einsatz eines integralen Führungsreglers die stationäre Genauigkeit entscheidend. Der analoge Teil der Schaltung besteht aus einem integrierten Messverstärker [103] sowie einem einfachen PI-Regler [102]. Der digitale Führungsregler, der als PID-Regler ausgelegt ist, arbeitet nach dem gleichen Prinzip wie der bereits besprochene Temperaturregler.
Abbildung 6.10 zeigt das Einschwingen des Diodenstroms bei einer sprungförmigen Änderung des Sollwertes von 60 mA auf 100 mA. Der Einschwingvorgang Δt_e dauert dabei etwa 40 ms.
Abbildung 6.11 zeigt die Auswertung von Regelergebnissen im eingeschwungenen Zustand. Im Einzelnen ergeben sich die Ergebnisse, die Tabelle 6.2 zusammenfasst. Die Standardabweichung σ bewegt sich unterhalb der geforderten Schranke. Die Absoltgenauigkeit spielt hier, wie bereits bei der Temperaturregelung beschrieben, eine untergeordnete Rolle.

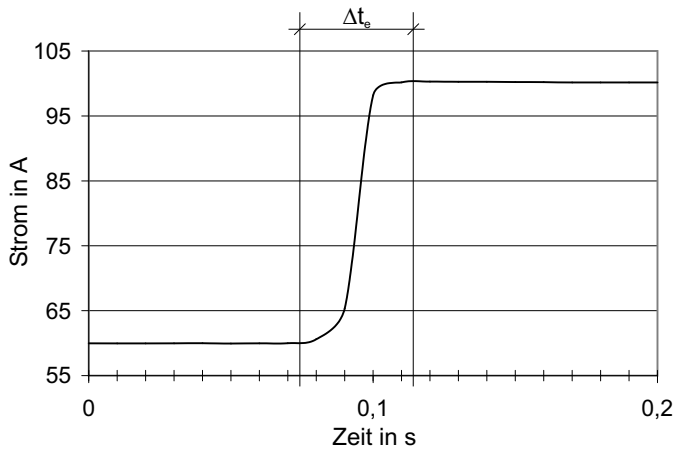


Abbildung 6.10: Einschwingen des Diodenstroms nach einer sprungförmigen Änderung des Sollwertes von 60 mA *auf* 100 mA

Tabelle 6.2: Regelversuche mit dem kaskadierten Temperaturregler

I_{soll}	m	σ	ΔI_{stat}	$\Delta I_{stat,rel}$
80 mA	79,9705 mA	7,2 μA	29,4 μA	0,037%
100 mA	99,9664 mA	13,8 μA	33,6 μA	0,034%

6.2 Bildaufnahme

Zur Aufnahme der Specklebilder dienen zwei unterschiedliche Kameras:

1. Pixelfly der Firma PCO mit einer Auflösung von 1280x1024 Pixeln bei einer Wiederholrate von 9 Bildern pro Sekunde

2. CV-M10 der Firma JAI mit einer Auflösung von 740x581 Pixeln bei einer Wiederholrate von 15 Bildern pro Sekunde

Die beiden verwendeten Kameras stellen ihre Daten in dem zum Framegrabber kompatiblen Format zur Verfügung [106]. Physikalisch arbeitet die Schnittstelle im Wesentlichen mit den folgenden Signalen:

- VINIT: Initialisiert die Kamera und startet die Belichtung des CCD-Chips. Die Kamera belichtet dann entsprechend der vorher programmierten Belichtungszeit.

- Frame Data Valid (VDF): Signalisiert, dass die Kamera mit dem Senden eines Bildes beginnt.

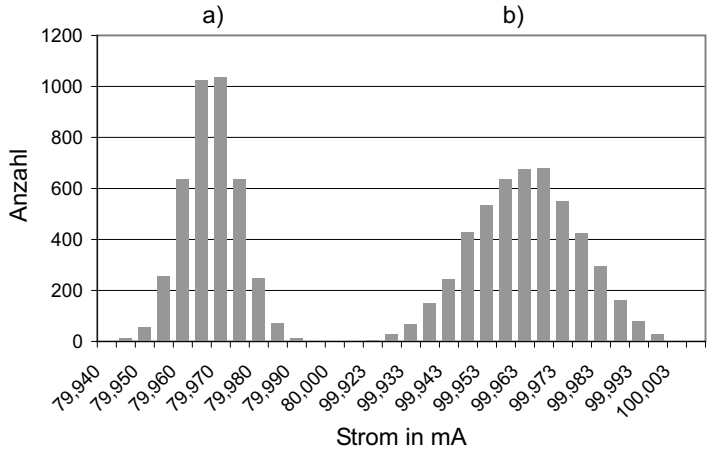


Abbildung 6.11: Sollwertabweichung bei eingeschwungener Regelung für a) I_{soll} = 80 mA, *b)* I_{soll} = 100 mA

- Line Data Valid (LDF): Signalisiert, dass die Kamera mit dem Senden einer Zeile beginnt.
- Datenleitungen (DATA): Je nach Kamera bis zu 16 Bit.
- Kamera Takt (CCLK): Synchronisiert die Daten.

Die Aufnahme eines Bildes erfolgt dann in den folgenden Schritten [50, 105]:

1. VINIT senden
2. FDV abwarten
3. Die ungültigen Zeilen zu Beginn des Kamerabilds abwarten.
4. Die ungültigen Pixel zu Beginn einer gültigen Zeile abwarten.
5. Die gültigen Pixel in den Speicher übernehmen.

Die Datenaufnahme erfolgt mit Hilfe einer digitalen Abtastung der Signale. Das bedeutet, wenn der Kameratakt gültige Daten anzeigt, entscheidet der Komparator am Eingang des FPGA, ob eine logische eins oder null anliegt. Ein wesentliches Problem stellt bei der Datenaufnahme die Signalqualität dar. Fehlinterpretationen bei den Datenleitungen betreffen dabei lediglich das entsprechende Pixel, während insbesondere CCLK, das zur Synchronisation der Daten dient, eine Fehlerfortpflanzung nach sich zieht, falls beispielsweise ein steigende Flanke nicht erkannt wird. Um eine qualitativ hochwertige Digitalisierung dennoch zu gewährleisten,

läuft der Kameratakt zunächst über eine in Abschnitt 3.1 beschriebene DLL. Die Abtastung des Kameratakts sowie der Daten und Steuerleitungen erfolgt mit Hilfe des FPGA Systemtakts, so dass dieser mindestens zwei mal so groß wie CCKL sein muss.

6.3 Datenauswertung

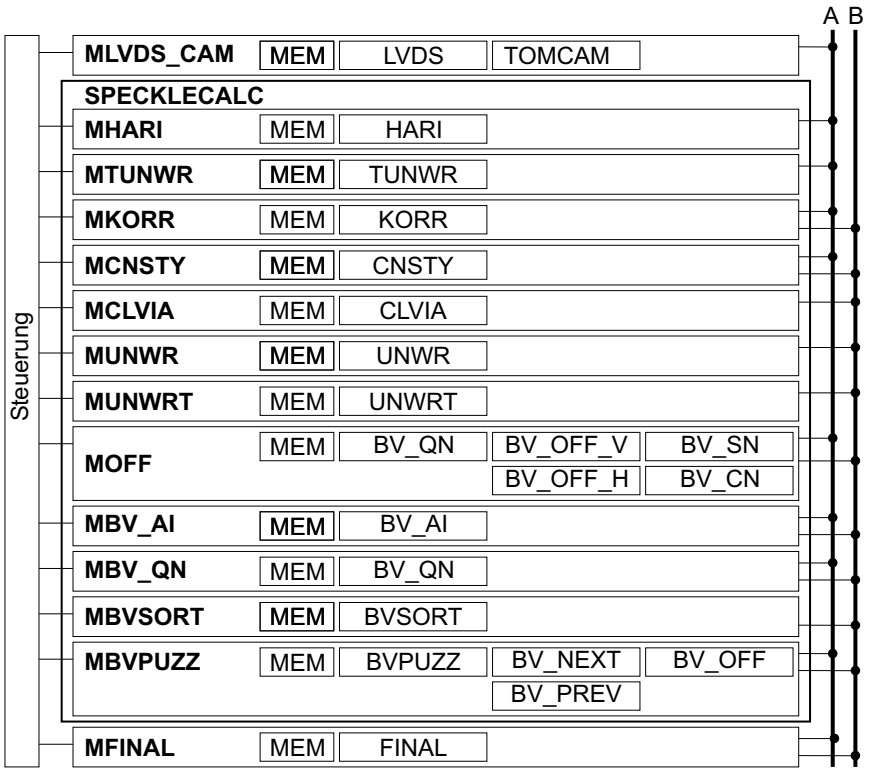


Abbildung 6.12: Zusammenschaltung der Verarbeitungsmodule

Der folgende Abschnitt beschreibt die Zusammenschaltung der in den Abschnitten 4 und 5 vorgestellten Module. Das wesentliche begrenzende Element bei der Datenauswertung stellt dabei die Geschwindigkeit der Kamera, sowie die des Speicherzugriffs dar. Das heißt, der maximale Datendurchsatz ergibt sich durch die Busbreite, die Zahl der Busse und die Geschwindigkeit der Speicherbausteine. Das Ziel des Designs der Pipelines besteht darin, den maximalen Datendurchsatz zu erreichen.
Im Folgenden kommen Dual-Ported-RAM-Bausteine zum Einsatz [104], die es

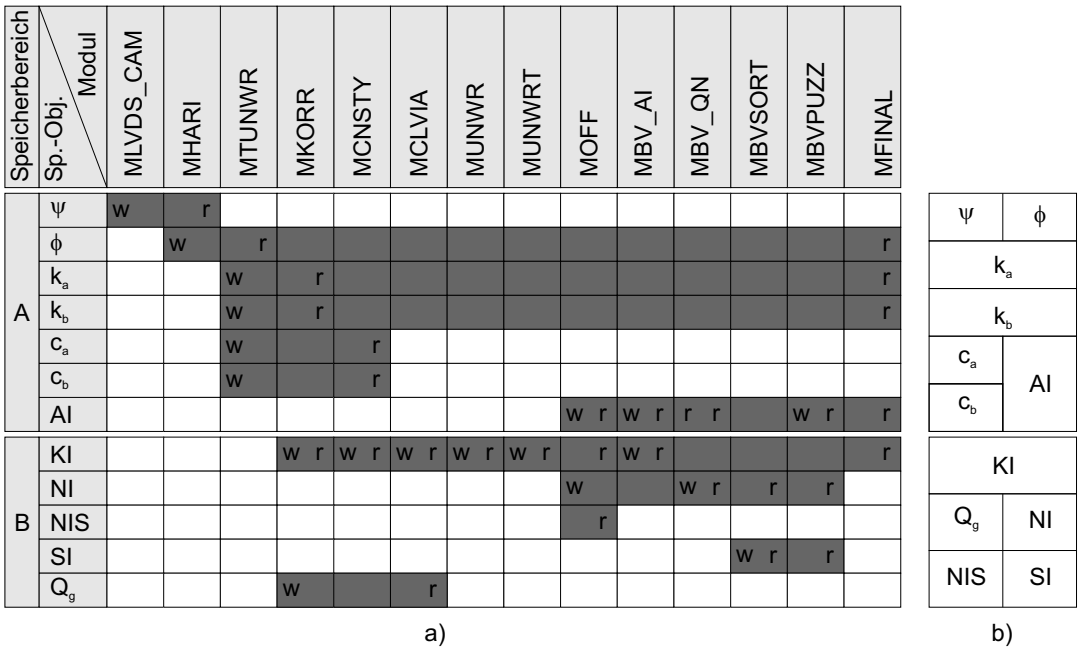


Abbildung 6.13: a) Lebensdauer und Operation (r: lesen, w: schreiben) der Speicherobjekte (Sp.-Obj.) in den einzelnen FPGA Modulen, b) logische Speicheraufteilung

erlauben, auf zwei getrennten Adress- und Datenbussen gleichzeitig auf denselben Speicherbaustein zuzugreifen. Durch den Einsatz statischer RAM-Bausteine sind keine Zyklen zum Auffrischen des Speichers notwendig und der Lese- bzw. Schreibzugriff erfolgt in einem einzigen Takt. Dadurch können die Pipelines zur Verarbeitung gleichzeitig Daten lesen und schrieben. Insgesamt arbeitet das System mit zwei getrennten Dual-Ported Speicherbereichen mit vier Adress und Datenbussen. Die Breite der Datenbusse beträgt 128 Bit. Die beiden Speicherbereiche werden mit *A* und *B* bezeichnet, die Busse mit A_1, A_2, B_1 und B_2. Abbildung 6.12 zeigt die Zusammenschaltung der einzelnen Module. Dem ganzen System ist eine Steuerung überlagert, welche die einzelnen Module synchronisiert. Die Speicherzugriffe realisieren die einschließenden Module, der Bezeichnung im Folgenden mit einem *M* beginnt. Tabelle 6.4 zeigt die Breite der Busse sowie die Größe der notwendigen Speicherbereiche für die verschiedenen Speicherobjekte aus Tabelle 5.3. Abbildung 6.13 zeigt die logische Speicheraufteilung und die Zugriffe der einzelnen Arbeitsschritte auf die einzelnen Busse. Durch die unterschiedliche Lebensdauer der einzelnen Speicherobjekte kann der Speicher mehrfach verwendet werden.
Im ersten Schritt arbeiten die Pipelines mit der maximalen Lesegeschwindigkeit

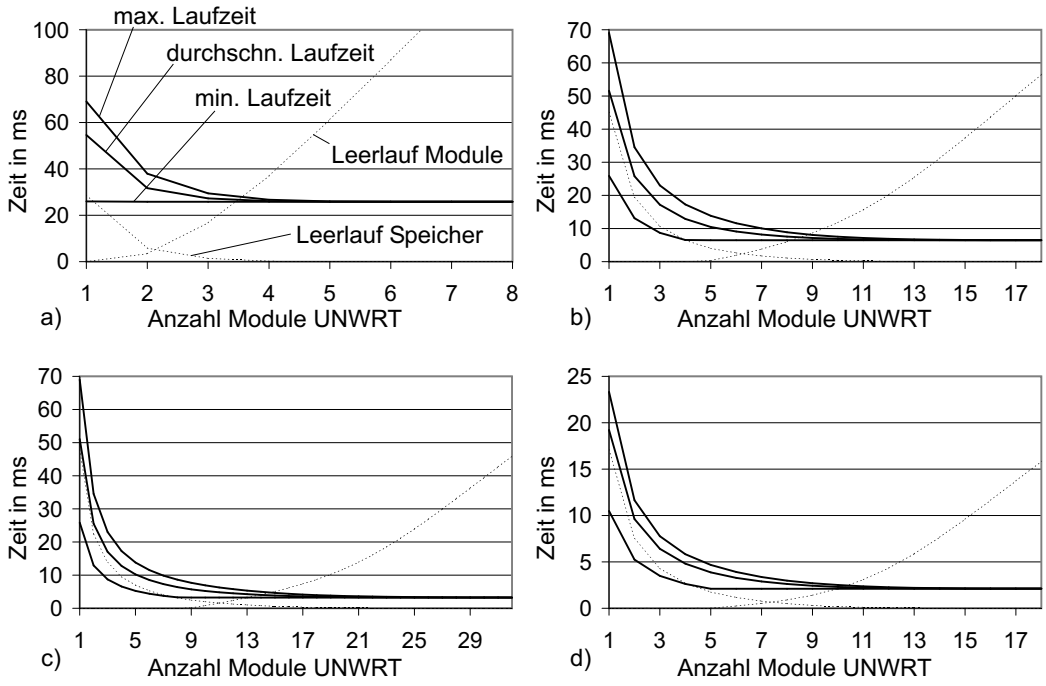


Abbildung 6.14: Maximale, minimale und durchschnittliche Laufzeit des Moduls MUNWRT sowie die integrale Leerlaufzeit aller Module UNWRT. Kameratyp und Fülldauer des Schieberegisters: a) Pixelfly, 64 Takte, b) Pixelfly, 16 Takte, c) Pixelfly, 8 Takte, d) CV-M10, 16 Takte

der Speicherbausteine, die bei 12 ms liegt. Dies erlaubt theoretisch eine maximale Taktrate von 83, 3 MHz, wobei die Versuche bei der etwas geringeren Taktrate von 80 MHz liefen. Die Vorversuche mit dem Hardwareemulator erfolgten bei dessen maximaler Taktrate von 50 MHz.
Die Verarbeitungsmodule können in drei Klassen eingeteilt werden:

1. Module mit definierter Laufzeit: *MHARI*, *MTUNWR*, *MKORR*, *MCNSTY*, *MCLVIA*, *MFINAL*
   Die Laufzeit dieser Module hängt linear mit der Pixelzahl des Bildes zusammen. Sie arbeiten als Pipeline und führen auf einem Bus gleichzeitig Lese- und Schreibzugriffe durch, um so den maximalen Durchsatz zu garantieren.

2. Module mit definierter Laufzeit, abhängig von der Zahl der Bereiche und Nachbarschaften: *MUNWRS*, *MUNWRE*, *MUNWRT*, MBV_AI, MBV_QN, *BVSORTS*, *MBVSORT*, *BVOFF*

Deren Laufzeit lässt sich nicht vorhersagen, hängt aber linear von n_N und n_A ab.

3. Module mit undefinierter Laufzeit: *MUNWR, MOFF, MBVPUZZ*
   Hierzu zählt insbesondere die Entfaltung des Bildes. Dieser Algorithmus kann nie mit dem maximalen Durchsatz arbeiten, da er immer zuerst entscheiden muss, welche Kachel als nächste zu entfalten ist.

In vielen Fällen können mit einem Speicherzugriff mehrere Objekte zur Auswertung gleichzeitig gelesen werden. In diesem Fall bietet sich eine parallele Verarbeitung mehrerer Module an. Die Tabellen 6.5 und 6.6 fassen die Laufzeiten der Module zusammen. Die Zahl der benötigten Takte c_g ergibt sich dabei zu:

$$c_g = c_b \cdot n_b + c_e \cdot n_e + \frac{c_r \cdot n_r}{n_m} \tag{6.6}$$

Dabei bezeichnet c_b die Zahl der Takte zum Einlaufen, n_b die Zahl der Einlaufvorgänge, c_e die Zahl der Takte zum Auslaufen, n_e die Zahl der Auslaufvorgänge, c_r die Zahl der benötigten Takte zum Abschluss eines Verarbeitungsvorgangs in einem Takt, n_r die Zahl der Verarbeitungsschritte und n_m die Zahl der parallel arbeitenden Instanzen eines Modultyps. Die Tabellen 6.5 und 6.6 zeigen die Laufzeiten der Module, wobei auf die Auflistung der Einlauf- und Auslaufvorgänge verzichtet wird, da ihr Einfluss auf die Gesamtlaufzeit zu vernachlässigen ist.
Um die Zahl der Bereiche, der Nachbarschaftsbeziehungen und die Laufzeit der Module des Types mit unbestimmter Laufzeit bestimmen zu können, diente die Auswertung von 417 Phasenbildern mit der Pixelfly sowie 94 Phasenbilder mit der CV-M10. Die Auswertung der Entfaltung dieser Bilder ergibt zunächst die Zahl der Bereichsinformationen n_A, die der Nachbarschaftsinformationen n_N und weitere Variablengrößen nach Abbildung 6.3. Die Auswertung der Laufzeitmessungen zeigen die Tabellen 6.5 und 6.6.
Die Entfaltung der Kacheln nach Abschnitt 5.2.1 bietet ein hohes Maß an Optimierungsmöglichkeiten. Die Kachelpipeline arbeitet zwar eine zunächst unbestimmte Zeit, die Entfaltung der Kacheln kann jedoch problemlos parallel ausgeführt werden. Die Pipeline selbst benötigt, wegen der initialen Berechnung des Mittelwertes, 64 Takte Anlaufzeit. Um die maximale Auswertungsrate zu erreichen, erhalten die Module vorgeschaltete Schieberegister, welche möglichst gleichmäßig aus dem Speicher gefüllt werden. Diese können auch parallel beschrieben werden. Neben der Geschwindigkeit, mit der die Schieberegister beschrieben werden, kann nun die Zahl der parallelen Pipelines variiert werden. Abbildung 6.14 zeigt die Auswertung dieser Variationen. Hierbei ist deutlich zu erkennen wie sich die Auswertegeschwindigkeit mit zunehmender Zahl an parallelen Pipelines dem maximalen Speicherdurchsatz nähert. Gleichzeitig steigt jedoch die integrale Leerlaufzeit aller parallel arbeitenden Module. Da eine Kachel wenigstens acht Lesezyklen benötigt, wobei jeweils acht Pixel gleichzeitig ausgelesen werden, kann

das Schieberegister in minimal acht Takten gefüllt werden. Bei einem Einsatz von etwa 30 parallelen Modulen ist dann die maximale Geschwindigkeit erreicht. Abbildung 6.14.d zeigt als Beispiel die Verarbeitung des Kamerabilds von CV-M10.

6.4 Datenübermittlung

Zur Datenübermittlung stehen zwei Wege zur Verfügung:

1. Übertragung mittels eines Framegrabbers zum PC.
2. Übertragung mittels TCP/IP über LAN, WLAN oder GSM.

Insbesondere bei Echtzeitanwendungen bietet sich die Übertragung mittels Framegrabbers an. Der FPGA emuliert eine Kamera und übermittelt dabei jedoch die Höheninformationen des Bildes. Der PC speichert die Daten und bietet damit die Möglichkeit, sie mit gängigen Bildverarbeitungstools weiterzuverarbeiten [110]. Eine wichtige Funktion übernimmt dabei ein Server, der als Mediator zwischen den verschiedenen Übertragungswegen arbeitet, den Zugriff vereinheitlicht und es ermöglicht das gesamte System Remote zu steuern. Dieser Server arbeitet aus Sicht des Netzwerks mit einem im Klartext lesbaren Protokoll dem SlimRPC-Transfer-Protokoll (RPC: Remote-Procedure Call). Dieses stellt ein stark vereinfachtes Protokoll zur Fernsteuerung des Messsystems dar. Der Server kennt dabei die Eigenschaften des Messsystems und insbesondere weiß er über die Leistungsfähigkeit der verwendeten Schnittstelle. Er gibt der Client-Applikation, ähnlich JINI [111] von JAVA Auskunft über die zur Verfügung stehenden Funktionen des Messgerätes.

6.5 Visualisierung

Zur Visualisierung der dreidimensionalen Messdaten stehen prinzipiell alle gängigen Methoden [110] zur Verfügung. Um die Visualisierung und die Steuerung des Systems in einer betriebssystemunabhängigen Oberfläche zusammenzuführen bietet sich die Programmiersprache Java [114] mit der Java3D-API [112] an. Aus Gründen der Performance ist die Java3D-API jedoch plattformabhängig. Java3D kann nicht nur dreidimensionale Objekte darstellen, sondern lässt sich auch für die Programmierung ganzer 3D-Welten heranziehen. Das Visualisierungssystem basiert auf einem Tomcat-Server [115], der den in Abschnitt 6.4 beschriebenen Mediator implementiert. Die Applets zur Visualisierung liegen auf dem Tomcatserver zum herunterladen bereit. Die 3D-Daten selbst holt das Applet sich über eine separate Portverbindung vom Server. Der Mediator wird zur Steuerung der Daten ebenfalls über eine separate Portverbindung angesprochen [108].

6.6 Gesamtsystem

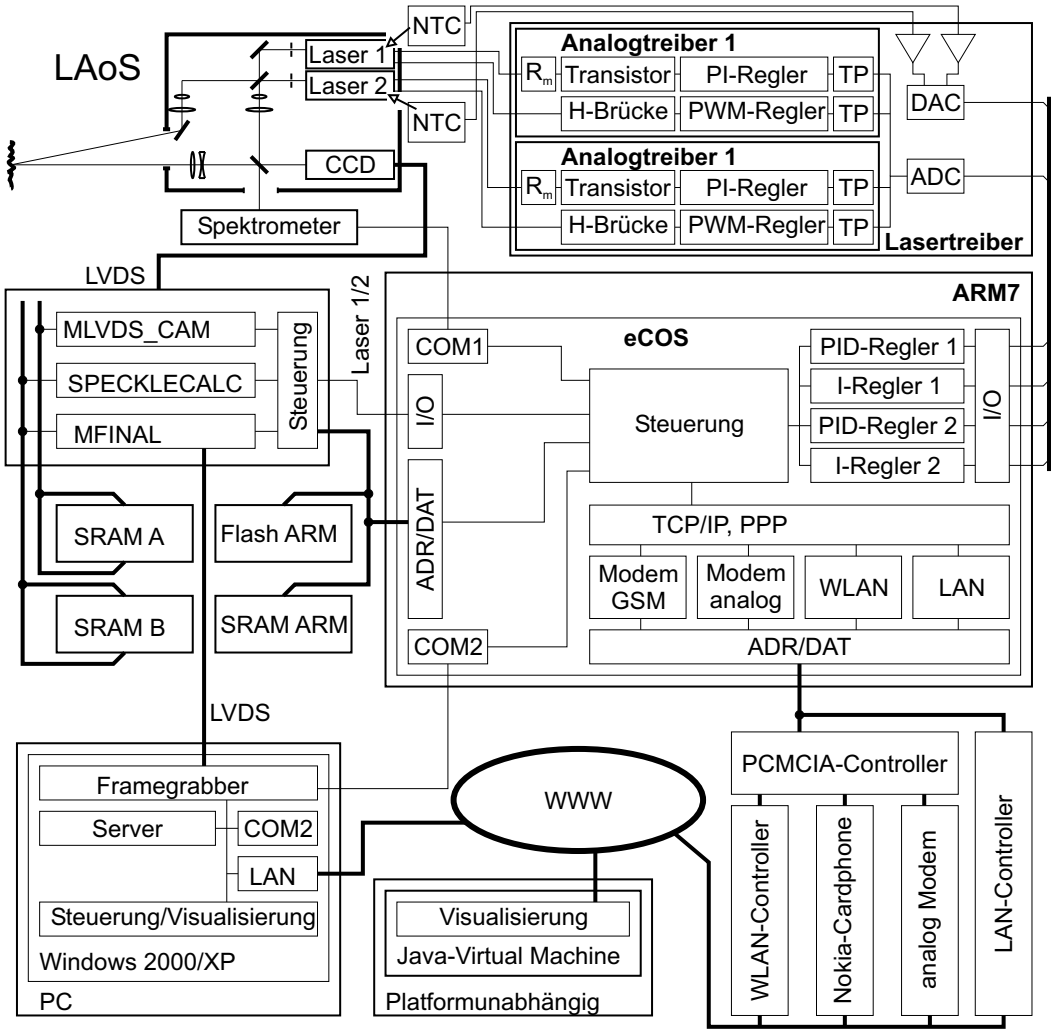


Abbildung 6.15: Gesamtsystem zur Echtzeitvermessung von Oberflächenformen mittels Speckleinterferometrie

Abbildung 6.15 zeigt das Gesamtsystem, das sich aus den in diesem Abschnitt bisher vorgestellten Komponenten zusammensetzt. Die wesentliche Komponente zur Steuerung des Gesamtsystems und zur Kommunikation stellt der ARM7-TDMI Mikrocontroller der Firma Atmel dar. Er verfügt über ein internes RAM von 4 kByte sowie über ein externes RAM von 512 kByte und ein externes Flash von 2 MByte. Als Betriebssystem arbeitet ein eCOS Echtzeitkernel [116] mit TCP/IP-Stack [121, 122, 123]. Als Datenkanäle können über einen PCMCIA-Controller

Tabelle 6.3: Größe der Variablen (max: maximal, av: durchschnittlich, Reg: Registerbreite)

	max	av	Reg	max	av	Reg
N_a	5028	2120	13 Bit	1651	924	11 Bit
O_k	13	3	4 Bit	6	2	3 Bit
O_n	13	4	4 Bit	7	3	3 Bit
O_a	24	6	5 Bit	12	5	4 Bit
$n_n a$	497	153	9 Bit	179	53	8 Bit
Q_n	4704	1416	13 Bit	3712	1087	12 Bit
n_N	9467	4331	14 Bit	3009	1895	12 Bit

die Wireless LAN-Karte Type II der Firma Agere [119], ein analoges Modem 56K Global Modem PC Card der Firma 3COM [117] sowie das Nokia Cardphone 2.0 der Firma Nokia [118] angeschlossen werden. All diese Übertragungswege bieten einen Zugang zum Inter- oder Intranet, in dem sich der Tomcat-Server des Abschnitts 6.5 befindet. Im Mikrocontroller ist ferner das SlimRPC implementiert, das sowohl über TCP/IP als auch über die serielle Schnittstelle *COM*2 angesprochen werden kann.
Als Messkopf dient das bereits in Abschnitt 2.6 beschriebene Speckleinterferometer LAoS. Im LAoS befindet sich neben den beiden Laserdioden zur Beleuchtung der Messoberfläche die CCD-Kamera zur Aufnahme der Specklebilder. Die Laserdiodenansteuerung aus Abschnitt 6.1 regelt die Laserdioden in ihren gewünschten Arbeitspunkt, in dem sie monomodig eine definierte Wellenlänge erzeugen. Der Laserdiodentreiber besteht aus einem analogen und einem digitalen Teil. Der analoge Teil wiederum besteht zum einen aus dem Leistungstreiber für die Ansteuerung der beiden Peltierelemente und dem analogen PI-Regler. Unmittelbar neben den Laserdioden in deren Aufnahmen sitzen PTCs, die als Sensor zur Regelung dienen. Weiterhin gehören die beiden Transistoren mit ihren vorgeschalteten PI-Reglern zur Stromregelung mit den Messwiderständen als Sensoren zum Analogteil. Die Schnittstelle zum Mikrocontroller bilden ein ADC und DAC, die bereits Abschnitt 6.1 beschreibt. Der digitale Teil der Regelung läuft als Task hoher Priorität im eCOS. Um aus dem entfalteten Bild die Höheninformation zu gewinnen, müssen die Wellenlängen der Laserdioden bekannt sein. Diese überwacht das Spektrometer HR2000 der Firma Ocean Optics [120] mit einem Messbereich von 350 nm bis 850 nm, das eine Auflösung von $0,25$ nm besitzt und mit Hilfe einer seriellen Schnittstelle an den Mikrocontroller angeschlossen ist. Mit Hilfe dieses Systems lässt sich auch die Vermessung der Stabilitätskarten und damit verbunden die Regelung der Wellenlänge der Laserdiode sowie die Überwachung

Tabelle 6.4: Speicherobjekte (Adr: Adressbus, Dat: Datenbus, max: maximale Anzahl)

	Pixelfly		CV-M10	
Bildhöhe	1024 Pixel		581 Pixel	
Bildbreite	1280 Pixel		740 Pixel	
gesamt	1310720 Pixel		429940 Pixel	
Bildhöhe	256 Kacheln		145 Kacheln	
Bildbreite	320 Kacheln		185 Kacheln	
gesamt	81920 Kacheln		26825 Kacheln	
Speicherbereich A				
	Adr	Dat	Adr	Dat
B_ϕ	21 Bit	16 Bit	20 Bit	16 Bit
B_P	21 Bit	16 Bit	20 Bit	16 Bit
AI	17 Bit	128 Bit	15 Bit	128 Bit
K_a	15 Bit	2 Bit	13 Bit	2 Bit
K_b	15 Bit	2 Bit	13 Bit	2 Bit
c_a	18 Bit	16 Bit	16 Bit	16 Bit
c_b	18 Bit	16 Bit	16 Bit	16 Bit
Speicherbereich B				
	Adr	Dat	Adr	Dat
MKI	14 Bit	1 Bit	12 Bit	1 Bit
KI	19 Bit	32 Bit	17 Bit	32 Bit
NI	18 Bit	128 Bit	16 Bit	128 Bit
NIS	16 Bit	32 Bit	14 Bit	32 Bit
SI	16 Bit	32 Bit	14 Bit	32 Bit
Speichergröße				
	A	B	A	B
Speicher	2990080 Byte	851968 Byte	1130496 Byte	212992 Byte

Tabelle 6.5: Laufzeit der einzelnen Module bei der Kamera Pixelfly

Taktfrequenz des FPGA					50 MHz	80 MHz
Modul	c_r	n_r	n_m	c_g	t_g in ms	t_g in ms
Maximale Laufzeit						
MHARI	1	1310720	1	1310737	26,21	16,38
MTUNWR	863720	1	1	863720	17,27	10,80
MKORR	1	81920	4	20490	0,41	0,26
MCNSTY	2	81920	1	165370	3,31	2,07
MCLVIA	1	320	32	772	0,02	0,01
MUNWRS	1	40960	8	5121	0,10	0,06
MUNWR	724905	2	4	362453	7,25	4,53
MUNWRE	1	40960	8	5121	0,10	0,06
MUNWRT	81920	1	1	81922	1,64	1,02
MOFF	1552958	1	1	1552958	31,06	19,41
MBV_AI	81920	1	1	81922	1,64	1,02
MBV_QN	1	9467	1	9477	0,19	0,12
BVSORTS	1	9467	1	9468	0,19	0,12
MBVSORT	1	198807	4	49723	0,99	0,62
MBVPUZZ	913749	1	1	913749	18,27	11,42
BVOFF	5028	1	1	5029	0,10	0,06
MFINAL	1	1310720	8	163843	3,28	2,05
gesamt				5601874	112,04	70,02
Durchschnittliche Laufzeit						
MHARI	1	1310720	1	1310737	26,21	16,38
MTUNWR	638874	1	1	638874	12,78	7,99
MKORR	1	81920	4	20490	0,41	0,26
MCNSTY	2	81920	1	165370	3,31	2,07
MCLVIA	1	320	32	772	0,02	0,01
MUNWRS	1	40960	8	5121	0,10	0,06
MUNWR	404003	2	4	202001	4,04	2,53
MUNWRE	1	40960	8	5121	0,10	0,06
MUNWRT	81920	1	1	81922	1,64	1,02
MOFF	356897	1	1	356897	7,14	4,46
MBV_AI	81920	1	1	81922	1,64	1,02
MBV_QN	1	4331	1	4341	0,09	0,05
BVSORTS	1	4331	1	4332	0,09	0,05
MBVSORT	1	90951	4	22759	0,46	0,28
MBVPUZZ	323340	1	1	323340	6,47	4,04
BVOFF	2120	1	1	2121	0,04	0,03
MFINAL	1	1310720	8	163843	3,28	2,05
gesamt				3430106	68,60	42,88

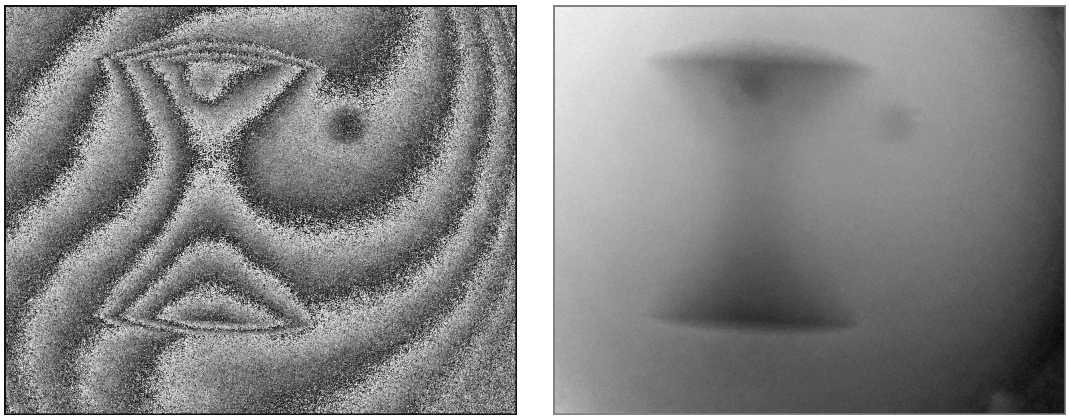
Tabelle 6.6: Laufzeit der einzelnen Module bei der Kamera CV-M10

Taktfrequenz des FPGA					50 MHz	80 MHz
Modul	c_r	n_r	n_m	c_g	t_g in ms	t_g in ms
Maximale Laufzeit						
MHARI	1	429940	1	429957	8,60	5,37
MTUNWR	291640	1	1	291640	5,83	3,65
MKORR	1	26871	4	6728	0,13	0,08
MCNSTY	2	26871	1	54608	1,09	0,68
MCLVIA	1	185	32	436	0,01	0,01
MUNWRS	1	13436	8	1680	0,03	0,02
MUNWR	152517	2	4	76259	1,53	0,95
MUNWRE	1	13436	8	1680	0,03	0,02
MUNWRT	26871	1	1	26873	0,54	0,34
MOFF	224164	1	1	224164	4,48	2,80
MBV_AI	26871	1	1	26873	0,54	0,34
MBV_QN	1	3009	1	3019	0,06	0,04
BVSORTS	1	3009	1	3010	0,06	0,04
MBVSORT	1	63189	4	15818	0,32	0,20
MBVPUZZ	155562	1	1	155562	3,11	1,94
BVOFF	1651	1	1	1652	0,03	0,02
MFINAL	1	429940	8	53746	1,07	0,67
gesamt				1373704	27,47	17,17
Durchschnittliche Laufzeit						
MHARI	1	429940	1	429957	8,60	5,37
MTUNWR	239423	1	1	239423	4,79	2,99
MKORR	1	26871	4	6728	0,13	0,08
MCNSTY	2	26871	1	54608	1,09	0,68
MCLVIA	1	185	32	436	0,01	0,01
MUNWRS	1	13436	8	1680	0,03	0,02
MUNWR	105577	2	4	52789	1,06	0,66
MUNWRE	1	13436	8	1680	0,03	0,02
MUNWRT	26871	1	1	26873	0,54	0,34
MOFF	56127	1	1	56127	1,12	0,70
MBV_AI	26871	1	1	26873	0,54	0,34
MBV_QN	1	1895	1	1905	0,04	0,02
BVSORTS	1	1895	1	1896	0,04	0,02
MBVSORT	1	39795	4	9970	0,20	0,12
MBVPUZZ	75211	1	1	75211	1,50	0,94
BVOFF	924	1	1	925	0,02	0,01
MFINAL	1	429940	8	53746	1,07	0,67
gesamt				1040825	20,82	13,01

deren Stabilität realisieren. Die Stabilitätskarte legt der Mikrocontroller nach der Vermessung im Flash ab.
Die Datenverarbeitung erfolgt in einer Kette von FPGAs. Der erste FPGA stellt dabei den LVDS-Treiber zum Auslesen der Kameradaten bereit. Entsprechend der Tabelle 6.4 erhält jeder FPGA einen ausreichenden Speicher. Alle FPGAs sind außerdem über den Adress- und Datenbus mit dem Mikrocontroller verbunden. Die interruptgesteuerte Ansteuerung der Laserdioden erfolgt über I/O-Leitungen des Mikrocontrollers. Der letzte FPGA der Kette enthält ggf. die Ausgabe der Daten an den Framegrabber. Ansonsten bezieht der Mikrocontroller die Daten über Adress- und Datenbus und überträgt sie mittels einen der anderen Kanäle zum Server.
Tabelle 6.4 zeigt den Speicherbedarf des Gesamtsystems. Dabei ist festzustellen, dass die maximale Gesamtgröße des Speichers bei einer Auflösung von $1280x1024$ Pixeln insgesamt 3842048 Byte beträgt, so dass bei üblichen Speichergrößen insgesamt 4 MByte SRAM zum Einsatz kommen.
Die Tabellen 6.5 und 6.6 zeigen die Laufzeiten der einzelnen Module, aufgesplittet nach maximaler und durchschnittlicher Laufzeit. Dabei zeigt sich, dass bei maximaler Laufzeit und einer Taktrate von 80 MHz immer noch eine Verarbeitungsrate von 8,93 Bildern pro Sekunde möglich ist. Dies übersteigt bereits die maximale Bildrate der Pixelfly, die im gleichen Zeitraum damit $17,86$ aufnehmen müsste, aber eine maximale Aufnahmerate von 10 besitzt. Die durchschnittliche Verarbeitungsrate beträgt bei 80 MHz sogar $23,32$ Bilder pro Sekunde. Bei der CV-M10 übersteigen die Bildwiederholraten selbst bei maximaler Laufzeit ebenso die Leistungsfähigkeit der Kamera. Dabei ist bei 80 MHz eine durchschnittliche Verarbeitungsgeschwindigkeit von 76,86 Bildern pro Sekunde möglich.

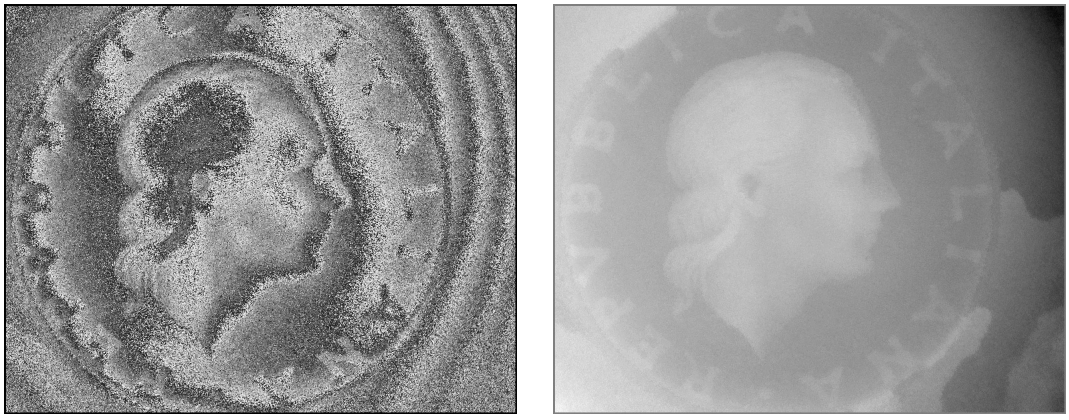
6.7 Messbeispiele

Die Messbeispiele in den Abbildungen 6.16, 6.17, 6.18, 6.19, 6.20 und 6.21 zeigen Auswertungen des Messsystems. Die Taktrate der Hardware beträgt dabei 50 MHz. Die Auswertezeit spiegelt die unterschiedliche Komplexität der verschiedenen Phasenbilder wider.



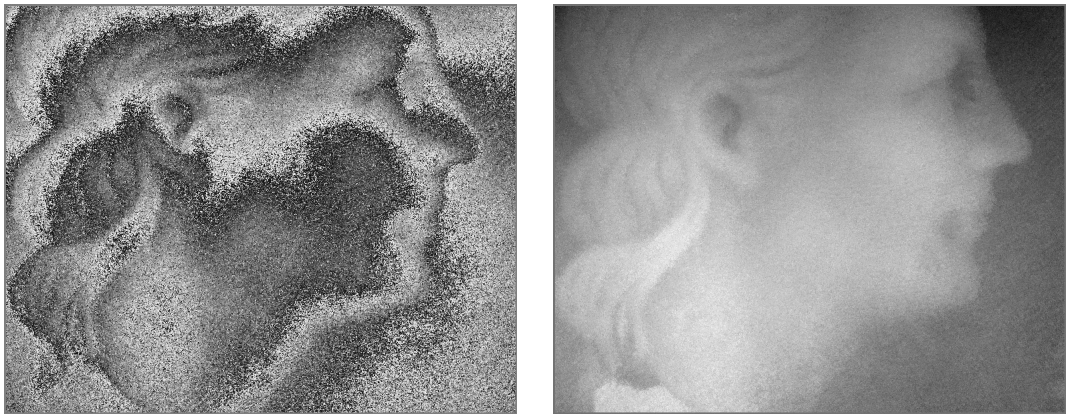
a) b)

Abbildung 6.16: Delle in einer Aluminiumplatte, $1280x1024\,\text{Pixel}^2$*, Auswertungsdauer* $65,53\,\text{ms}$*, synthetische Wellenlänge* $\Lambda = 62,5\,\mu\text{m}$*: a) Phasenbild, b) entfaltetes Bild*



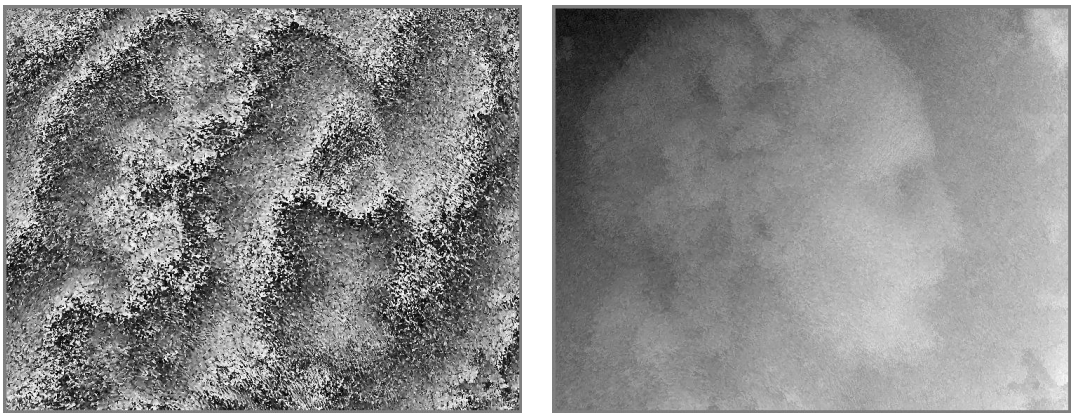
a) b)

Abbildung 6.17: 200 Lire Münze, $1280x1024\,\text{Pixel}^2$*, Auswertungsdauer* $71,34\,\text{ms}$*, synthetische Wellenlänge* $\Lambda = 74,3\,\mu\text{m}$*: a) Phasenbild, b) entfaltetes Bild*



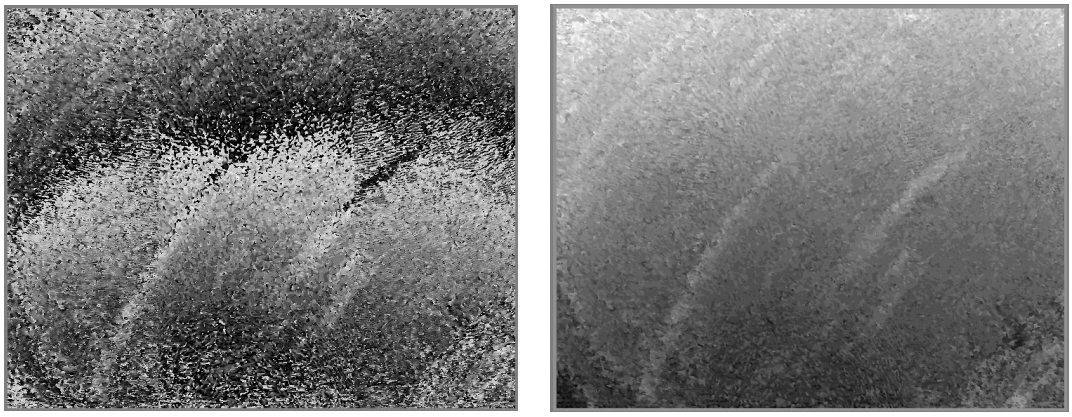
a) b)

Abbildung 6.18: 200 Lire Münze vergrößert, $1280x1024$ Pixel^2*, Auswertungsdauer* $61,89$ ms*, synthetische Wellenlänge* $\Lambda = 74,3\ \mu$m*: a) Phasenbild, b) entfaltetes Bild*



a) b)

Abbildung 6.19: 50 Lire Münze, $740x581$ Pixel^2*, Auswertungsdauer* $28,19$ ms*, synthetische Wellenlänge* $\Lambda = 74,3\ \mu$m*: a) Phasenbild, b) entfaltetes Bild*



a) b)

Abbildung 6.20: Bearbeitungsspuren bei der Herstellung einer Dichtfläche (Einspritzpumpe eines Dieselmotors), $740x581$ Pixel^2*, Auswertungsdauer* $13,11$ ms*, synthetische Wellenlänge* $\Lambda = 64\,\mu$m*: a) Phasenbild, b) entfaltetes Bild*

a) b)

Abbildung 6.21: gehämmertes Aluminiumblech, $740x581$ Pixel^2*, Auswertungsdauer* $27,77$ ms*, synthetische Wellenlänge* $\Lambda = 64\,\mu$m*: a) Phasenbild, b) entfaltetes Bild*

Kapitel 7

Zusammenfassung und Ausblick

Zur Realisierung einer Echtzeitvermessung dreidimensionaler Oberflächen mittels Speckleinterferometrie unter industriellen Umgebungsbedingungen auf der Basis des Speckleinterferometers LaOS [2] mit Zwei-Wellenlängen-Technologie, müssen drei wesentliche Aufgaben gelöst werden.

- Die Verarbeitungsgeschwindigkeit von Speckleinterferogrammen liegt typischerweise im Bereich mehrerer Sekunden und damit weit oberhalb der Bildwiederholrate herkömmlicher CCD-Kameras.

- Mechanische Shutter tragen Vibrationen in das Messsystem ein, so dass sich die Messdauer durch Wartezeiten von mehreren hundert Millisekunden zwischen den einzelnen Aufnahmen erheblich verlängert.

- Laserdioden besitzen aufgrund ihrer Herstellungsprozesse und Alterungserscheinung abweichende Eigenschaften bezüglich der Abhängigkeit von Wellenlänge und Stabilität von Strom und Temperatur.

Der erste Schritt zur Beschleunigung der Verarbeitungsgeschwindigkeit besteht in der Auswahl geeigneter Algorithmen sowie deren Optimierung bezüglich einer Realisierung in einer spezialisierten Hardware.
Zur Phasenbildberechnung, die Abschnitt 4 beschreibt, bietet sich dabei der Algorithmus nach Hariharan-Schwider an, da er qualitativ gute Ergebnisse erzielt. Die dazu nötigen Rechenoperationen lassen sich in VHDL abbilden, und damit in einem FPGA realisieren.
Zur Entfaltung nach Abschnitt 5 eignet sich ein Hybridalgorithmus, der die verschiedenen klassischen Entfaltungsalgorithmen erweitert und kombiniert. Der erste Schritt basiert auf einer pfadunabhängigen Entfaltung von Kacheln. Dieser Algorithmus arbeitet aus Sicht der Kachel global. Er vermeidet damit eine Fehlerfortpflanzung und arbeitet dennoch sehr schnell. Diese Entfaltung wird zwei

mal durchgeführt, wobei die Kacheln zwischen den Entfaltungen gegeneinander verschoben werden. Der Vergleich der sich daraus ergebenden überlappenden Bereiche führt dann dann zu einem Maß, mit dem sich die Qualität der entfalteten Kacheln bewerten lässt. Die anschließende Entfaltung der qualitativ höchstwertigsten Kacheln ergibt mehrere nicht verbundene unabhängige Bereiche. Deren Verbindung erfolgt in der Abhängigkeit eines Qualitätsmaßes, das sich aus Größe und Qualität der Bereiche sowie deren Grenzen zusammensetzt.
Die Realisierung der Algorithmen erfolgt in Form spezialisierter Hardwarelogik in FPGAs. Zur Forschung und Entwicklung dient dabei die Hardwareemulation von Mixed-Signal-Schaltungen nach Abschnitt 3. Die gesamte Hardware- und Softwareentwicklung erfolgt damit auf einer Plattform, die weitreichende Möglichkeiten bei der Fehlersuche ermöglicht. Insbesondere kann die elektrische Verbindung zwischen den einzelnen Baugruppen in kürzester Zeit mittels einer Netzliste verändert werden. Die sich daraus ergebende endgültige Schaltung kann dann ohne hohes Designrisiko gefertigt werden.
Die Mikrocontroller gestützte Laserdiodenregelung nach Abschnitt 6.1 erzielt beim Stromfluss eine Standardabweichung von 13 μA bei 100 mA und der Temperatur mit einer Standardabweichung von $1,03$ mK bei 25 °C. Die Messung der Wellenlängen des emittierten Laserdiodenlichts mittels eines Spektrometers ermöglicht die Aufnahme von Stabilitätskarten. Diese Kalibrationsmessungen können zu jederzeit im Gerät direkt erfolgen. Die Hardware ermöglicht neben der hochpräzisen Regelung von Temperatur und Strom ein Schalten Laserdioden in weniger als 30 ms. Damit kann das LAoS ohne bewegt Verschlusseinrichtungen arbeiten.
Das Gesamtsystem nach Abschnitt 6 besteht aus dem Speckleinterferometer LaOS, der rückgekoppelten Laserdiodensteuerung, der spezialisierten FPGA-Hardware, einem Mikrocontroller zur Systemsteuerung, verschiedenen Schnittstelle (LVDS, LAN, WLAN, Modem) und einer Visualisierung im Inter- und Intranet. Die erreichbaren Auswertraten bei einer Bildauflösung von $1280x1024$ Pixeln liegt bei durchschnittlich 23 und bei einer Bildauflösung von $740x581$ bei 76 Bildern pro Sekunde. Diese Taktraten übersteigen die möglichen Bildwiederholraten der eingesetzten Kameras.
Nachdem nun die Algorithmik zur schnellen Datenauswertung in spezialisierter Hardware bekannt ist, bietet sich in einem nächsten Schritt deren Umsetzung auf kleineren Plattformen wie beispielsweise PC-Steckkarten mit FPGAs oder den sogenannten „Pixelshadern“, die aus parallelen, frei programmierbaren, Rechenpipelines auf Graphikkarten bestehen an. Damit lässt sich der Aufwand zur Realisierung der Auswertung deutlich senken.

Abkürzungsverzeichnis

A/D-Wandler	Analog-Digital-Wandler
ASIC	Application Specific Integrated Cicuit
CCD	Charged Couplet Device
CCLK	Kamera Takt
CE	Clock Enable
CLB	Configurable Logic Block
CLK	Clock
CORDIC	Coordinate Rotation Digital Computer
CPLD	Complex Programmable Logic Device
D/A-Wandler	Analog-Digital-Wandler
DFB	Distributed Feedback Laser
DLL	Delay-Locked Loop
FDV	Frame Data Valid
FPGA	Field Programmable Gate Array
FPIC	Field Programmable Interconnect
GBUF	Global Buffer
INT	Interface zwischen Temperatursensor und Laserdiode
IOB	Input/Output Block
IP	Intellectual Property
LAoS	Laser Analysis of Surfaces
LC	Logic Cell
LCA	Logic Cell Array
LDV	Line Data Valid
LSB	least significant bit
LUT	Lookup-Table
MP3C	Produktbezeichnung des Hardware-Emulators der Firma Aptix
MSB	most significant bit
NTC	Temperatursensor mit negativem Temperaturkoeffizent (negative temperature coefficient)
PAL	Programmable Array Logic

PCB	Printed Circuit Board
PLA	Programmable Logic Array
PLD	Programmable Logic Device
PWM	Pulsweitenmodulation
PROM	Programmable Read Only Memory
PZT	Piezo-Electric-Transducer
RAM	Random Access Memory
RESET	Setze Flip-Flop zurück
ROM	Read Only Memory
RPC	Remote-Procedure-Call
SET	Setze Flip-Flop
SlimRPC	Reduziertes RPC
SPLD	Simple Programmable Logic Device
SRAM	Static Random Access Memory
TEC	Peltierelement
Vectoring	Drehstreckung beim CORDIC-Algorithmus, der als Ergebnis den gedrehten Vektor in auf die x-Achse transformiert
Verilog	Name einer Firma
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VINIT	CCD-Initialisierung

Symbolverzeichnis

a	Adresse
A	Zu integrierende Fläche
A_c	Bezeichnung eines Bereiches der sich aus Kacheln K_c zusammensetzt.
A_p	Parameter eines Bereiches.
b	Bit
B	Bittiefe
B_x	Bittiefe einer Kachel in x-Richtung
B_y	Bittiefe einer Kachel in y-Richtung
c	Mittelwert einer Kachel beim Unwrapping
$cset$	Signal zum Setzen des Mittelwertes
c^*	Mittelwert einer Kachel beim Unwrapping nach einem Optimierungsschritt
clk	Clock oder Takt
C	Korrelations zweier Kacheln im Überlappungsbereich
d	Verschiebung, Blendendurchmesser
d_in	Einlaufende Daten
d_out	Ausgegebene Daten
d_{xy}	Höhenunterschied an der Stelle (x, y)
$done$	Signal bei Beendigung der Kachelpipeline
e_k	Eintritt in die Kachel beim Entfalten mit $e_k \in s_o, s_r, s_u, s_l$
e_s	Exponent bei der Berechnung von S_k
$\|f\|_2^2$	L^2-Norm
$\|f\|_p$	L^p-Norm
f_{rel}	Relativer Fehler
I	Intensität
I_1	Messflächenintensität
I_2	Referenzflächenintensität
$I_{\lambda_1,xy}$	Interferogrammintensität bei λ_1 an der Stelle (x, y)
$I_{\lambda_2,xy}$	Interferogrammintensität bei λ_2 an der Stelle (x, y)
$I_{1,xy}$	Messflächenintensität an der Stelle (x, y)

$I_{2,xy}$	Messflächenintensität an der Stelle (x, y)
$I_{Diff,xy}$	Differenzintensität an der Stelle (x, y)
$I^*_{Diff,xy}$	Auswertbare Differenzintensität an der Stelle (x, y)
$I_{D,xy}$	Interferogrammintensität im deformierten Zustand (x, y)
$I_{G,xy}$	Interferogrammintensität im Grundzustand (x, y)
$\boldsymbol{k}$	Empfindlichkeitsvektor, Wellenvektor
$\boldsymbol{k}_1$	Beleuchtungsvektor
$\boldsymbol{k}_2$	Beobachtungsvektor
$\boldsymbol{k}_{11}$	Beleuchtungsvektor 11
$\boldsymbol{k}_{12}$	Beleuchtungsvektor 12
k_n	Korrekturfaktor der CORDIC-Iteration
$k(n)$	ganzzahlige Faltungskoeffizienten
$k(n)$	ganzzahlige Faltungskoeffizienten
K	Kachel als Bereich eines Bildes
KB	Kachelbesetzung in der Umgebung einer Kachel K_c
K_p	Parameter einer Kachel.
K_P	Phasenraum
l	Cavity-Länge oder Resonatorlänge
$\boldsymbol{l}$	Rotationsvektor des CORDIC-Algorithmus
L	Liste von Objekten
m	Mittelwert
n	Brechnungsindex
n_a	Zahl der Kacheln innerhalb eines Bereiches
n_k	Gesamtzahl der Kacheln eines Bildes
$n(K)$	Zahl der Punkte P_i in einer Kachelfläche K
n_n	Zahl der Kacheln einer Nachbarschaftsmenge
n_{na}	Zahl der Nachbarn eines Bereiches
n_x	Zahl der Punkte einer Kachelfläche K_a in x-Richtung
n_y	Zahl der Punkte einer Kachelfläche K_a in y-Richtung
n_A	Gesamtzahl aller Bereiche insgesamt
n_N	Gesamtzahl aller Nachbarschaftsbeziehungen
N_k	Menge aller Kacheln der Nachbarn zweier angrenzender Bereiche.
N_p	Parameter der Nachbarschaftsbeziehung.
N_A	Nummer eines entfalteten Bereiches
N_K	Nummer der Teilkette beim Entfalten von Bereichen.
o	Summe der Fehlerquadrate als Kriterium zur Kacheloptimierung
o^*	Summe der Fehlerquadrate als Kriterium zur Kacheloptimierung nach einem Optimierungsschritt
O	Offset einer Kachel oder eines Bereiches
O_k	Offset zwischen zwei sich überlappenden Kacheln

O_a	Offset zwischen zwei Bereichen
p	Phasenhub
P	Zeiger auf ein Objekt
P	Punkt
P_{opt}	Optische Leistung einer Laserdiode
q_n	Nummer der verketteten Liste der Bereichsinformationen A_p
Q_a	Bereichskorrelation
Q_b^l	Übergang zur nächsten linken Kachel
Q_b^o	Übergang zur nächsten oberen Kachel
Q_b^r	Übergang zur nächsten rechten Kachel
Q_b^u	Übergang zur nächsten unteren Kachel
Q_f	Bewertung des Grenzübergangs zweier Bereiche
Q_g	Binäre Kachelqualität
Q_k^*	Ähnlichkeitsmaß zweier Flächen
Q_k	Kachelkorrelation
Q_m	Maskierte Kacheln
Q_n	Gesamtbewertung der Nachbarschaftsbeziehung
Q_r^*	binäre Kachelkonsistenz
Q_r	Kachelkonsistenz
Q_N	Benachbarte Kacheln
Q_u	Entfaltete Kacheln
Q_z	Bewertung der Zahl der Kacheln zweier Bereiche
$reset$	Zurücksetzen von Modulen
R_q	quadratischer Mittenrauert
set	Signal zum setzen von Werten
s_b	Startkachel beim Entfalten
s_k	Anzahl Schritte zur Kacheloptimierung
s_o	Eintritt in die Kachel von oben
s_r	Anzahl der Shiftoperationen nach rechts
s_r	Eintritt in die Kachel von rechts
s_u	Eintritt in die Kachel von unten
s_l	Eintritt in die Kachel von links
S	Signale oder Leitungen
S	Verstärkungskurve der Laserdiode
S_d	Laufrichtung
S_e	Skalierungsfaktor am Eingang des CORDIC-Algorithmus
S_k	Ganzzahliger Skalierungsfaktor des CORDIC-Algorithmus
SNR	Signal-Rausch-Abstand (signal to noise ratio)
t_1	Term 1 bei der Vereinfachung von Gleichung 5.15
t_2	Term 2 bei der Vereinfachung von Gleichung 5.16
t_3	Term 3 bei der Vereinfachung von Gleichung 5.17

T	Transformation eines Unterraums
T_k	Tranformation einer Ebene im Phasenraum
$\mathcal{T}$	Menge aller Möglicher Transformationen eines Unterraums
w	Zahlenwert
W_k	Wertebereich des CORDIC-Algorithmus
x	x-Koordinate eines Kamerapixels
x_i	Laufende x-Koordinate eines Kamerapixels
X	Metrischer Raum
X_u	Metrischer Unterraum mit $X_u \subseteq X$
X_v	Metrischer Unterraum mit $X_v \subseteq X$
y	y-Koordinate eines Kamerapixels
y_i	Laufende y-Koordinate eines Kamerapixels

α	Winkel
α_i	Winkel
α_p	Winkel zwischen elektrischen Feldvektoren
β	Winkel
γ_0	Modulation
δ	Allgemeiner Abstandsbegriff im metrischen Raum
δ^*	Distanz zwischen einem Punkt und einem metrischen Unterraum
δ_d	diskrete Metrik
δ_H*	Distanz zwischen zwei metrischen Unterräumen
δ_H	Hausdorff-Distanz
Δ	Differenzoperator
Δc	Änderung des Mittelwertes c
$\Delta^x_{i,j}$	Phasendifferenzen der gefalteten Phase in x-Richtung
$\Delta^y_{i,j}$	Phasendifferenzen der gefalteten Phase in y-Richtung
ϕ	Gefaltete Phase (wrapped phase)
φ	gemessene kontinuierliche Phase (unwrapped phase)
λ	Wellenlänge
λ_1	Wellenlänge 1
λ_2	Wellenlänge 2
Λ	synthetische Wellenlänge
$\nabla\phi$	Phasengradient
Φ_1	Phasendifferenz 1
Φ_2	Phasendifferenz 2
Φ_0	Phasendifferenz
σ	Drehwinkel beim CORDIC-Algorithmus
Σ	Standardabweichung

ψ	tatsächliche kontinuierliche Phase (unwrapped phase)
θ	Tmperatur

ceil $[w]$	rundet den Wert w auf die nächste ganze Zahl auf
floor $[w]$	rundet den Wert w auf die nächste ganze Zahl ab
inf $[w]$	Infinum [48]
Int $[w]$	Rundet den Wert w auf die nächste ganze Zahl
RoL $[w, n]$	Rotiert einen binären Wert Wert w um n Bits nach links
RoR $[w, n]$	Rotiert einen binären Wert Wert w um n Bits nach rechts
sign $[w]$	Vorzeichen des Wertes w
sup $[w]$	Supremum [48]
$\overline{w}$	Negation von w
$w_1 \bmod w_2$	Berechnet den Rest der ganzzahligen Division von w_1 und w_2

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] A. W. Koch, M. W. Ruprecht, O. Toedter und G. Häusler. *Optische Messtechnik an technischen Oberflächen.* expert verlag, Renningen-Malsheim, 1998

[2] P. Evanschitzky. *Simulationsgestützte Oberflächendiagnostik mitels Speckle-Interferometrie.*, Dissertation Technische Universität München, 2002

[3] M. Ruprecht. *Ferndiagnose technischer Oberflächen mittels Laser-Speckle-Verfahren.* Shaker Verlag, Aachen, 1998

[4] J. P. Bläsing. *Qualitätssicherungssysteme und ihre Bedeutung für Klein- und Mittelbetriebe zur Wettbewerbssicherung in der Zukunft.* REFA Landesforum Landesverband Hessen, 1-22, 1990

[5] B. Larsen, L. Jeppesen, S. Börner *ZfP von Prozessanlagen, Stahlbau und Rohrleitungen in der Offshoreindustrie.* ZfP in Anwendung, Entwicklung und Forschung Berlin, 21.-23. Mai 2001 -Berichtsband 75-CD, 2001

[6] M. Courtoy. *Rapid system prototyping for real-time design validation.* Proceedings.-Ninth-International-Workshop-on-Rapid-System-Prototyping-Cat.-No.98TB100237. 1998: 108-12, IEEE Comput. Soc, Los Alamitos, CA, USA, 1998

[7] F. Dorsch, F. X. Daiminger,. *Aging tests of high power diode lasers as a basis for an international lifetime standard.* Proceedings-of-the-SPIE-The-International-Society-for-Optical-Engineering. 1996; 2870: 381-9, 1996

[8] A. Heumier, J.L. Carlsten. *Mode Hopping in Semiconductor Lasers.* http://www.ilxlightwave.com/productData/appnotes/Application

[9] M. Jakobi. *Laser Speckle Besed Surface Measurement Techniques Relevant to Fusion Devices.* Shaker Verlag, Aachen, 2001

[10] B. Ruffing. *Berührungslose Rauheitsmessungen technischer Oberflächen.* Dissertation, Universität Karlsruhe, 1987

[11] J. Evertz. *Optische 3D-Formerfassung durch den Einsatz von speckleinterferometrischen Meßverfahren.* Shaker Verlag, Aachen, 1997

[12] R. Jones, C. Wykes. *Holographic and speckle interferometry.* Cambridge University Press, London, New York, 1983

[13] M. Breitling. *Fehler und Fehlertoleranz.* http://www4.in.tum.de/misc/perlen/perlen-folien/Fehler.pdf, 2000, Stand: 1.5.2003

[14] J. C. Wyant, K Creath. *Recent advances in interferometric optical testing.* Laser Focus/Electro-Optics 21 118-132, 1985

[15] B. Packroß. *Interferometrie mit Laserdioden.* Berichte aus dem Institut für technische Optik der Universität Stuttgart, 1992

[16] J. C. Wyant, R. N. Shagam. *Use of electronic phase measurement techniques in optical testing.* Proc ICO-11 659-62, 1978

[17] M. P. Kothiyal, C. Desisle *Optical frequency shifter for heterodyne interferometry using counterrotating wave plates.* Optics Letters 9 319-21, 1984

[18] Y. Zou. *Speckleinterferometrie zur Topographiebestimmung an optisch rauen Oberflächen.* Berichte aus dem Institut für Technische Optik der Universität Stuttgart, 1996

[19] Bernhard Franze. *Formvermessung basierend auf Interferometrie mit durchstimmbaren Laserdioden.* Berichte aus dem Institut für Technische Optik der Universität Stuttgart, 1998

[20] B. P. Pfister. *Speckleinterferometrie mit neuen Phasenschiebemethoden.* Berichte aus dem Institut für Technische Optik der Universität Stuttgart, 1993

[21] D. C. Ghiglia, M. D. Pritt. *Two-Dimensional Phase Unwrapping.* John Wiley & Sons, Inc., New York, 1998

[22] D. W. Robinson, G. T. Reid. *Interferogram Analysis.* IOP Publishing Ltd, London, 1993

[23] K.Itoh. *Analysis of the phase unwrapping problem.* Applied-Optics, Vol. 21, No. 14, p. 2470, July 15, 1982

[24] R. M. Goldstein, H. A. Zebker, C. L. Werner. *Satellite radar interferometry: two-dimensional phase unwrapping.* Radio Science, Vol. 23, No. 4, pp. 713-720, 1988

[25] D. Stewart, R. Cook, I. McConnell, C. J. Oliver. *Optimal processing techniques for SAR.* Proceedings-of-the-SPIE-The-International-Society-for-Optical-Engineering, 3500: 377-84, 1998

[26] P. Söser. *Skriptum und Unterlagen zur Vorlesung aus integrierte Schaltungen.* http://www.ife.tugraz.at/LV/Skripten/is_vo_p.pdf, 2002

[27] L. M. Augustin, D. C. Luckham, B. A. Gennart, Y. Huh, A. G. Stanculescu. *Hardware Design and Simulation in VAL/VHDL.* Kluwer Academic publishers, Boston, 1991

[28] D. E. Thomas, P. Moorby. *The Verlilog Hardware Description Language.* Verlag, Ort, Jahr

[29] A. Sikora. *Programmierbare Logikbausteine.* Hanser, München, 2001

[30] K. Diepold. *Grundlagen der Informatik, Boolesche Algebra.* http://www.ldv.ei.tum.de/media/files/lehre/gi/vorlesung/GDI_03_BoolescheAlgebra.pdf, 2003, Stand: 10.3.2004

[31] M. Seifart. *Digitale Schaltungen und Schaltkreise.* Hüthig, Heidelberg, 1982

[32] *Virtex 2.5 V Field Programmable Gate Arrays.* Product Specification, DS003-2(v2.8), www.xlininx.com, 2002

[33] H. Främke. *VHDL Seminar.* http://www.fh-wedel.de/cis/archiv/seminare/ws96/vhdl/fraemke.htm, FH Wedel, 1996

[34] J. Bhasker. *Die VHDL-Syntax.* Prentice Hall, München, 1996

[35] P. Evanschitzky, A. W. Koch, M. Riemenschneider, T. Zeh. *Verfahren zum Bestimmen einer Struktur einer Oberfläche eines Messobjekts und Messvorrichtung.* Patentanmeldung DE 10122068.5, 2001

[36] A. Meixner, M. Riemenschneider, P. Evanschitzky, T. Zeh. *Homepage LASAZZ.* www.lasazz.com, 2002

[37] *Aptix Documentation.* Aptix, 1999

[38] *Explorer ™ Software, Product Brief, v2001.Q4.* www.aptix.com, 2001

[39] *Synopsis Online Help.* Synopsis, 2003

[40] *Synplicity Online Help.* Synplify, 2003

[41] *XILINX Core Generator.* XILINX, 2003

[42] *XILINX Place and Route.* XILINX, 2003

[43] *XILINX Modelsim.* XILINX, 2003

[44] A. Meixner. *Aufbau einer Speckleinterferometers für räumliches Phase-Shifting zur On-Line-Deformationsmessung.* Diplomarbeit, Lehrstuhl für Messsystem- und Sensortechnik, Technische Universität München, 2000

[45] Autor *Phasenschieben.* Verlag, Ort, Jahr

[46] P. Carré *Installation et utilisation du comparateur photoelectrique et interferentiel du bureau international des poids et mesures.* Metrologia 2 13-23, 1996

[47] *Matlab Help.* Matlab, 2003

[48] I. N. Bronstein, K. A. Semendjajew, G. Grosche. *Taschenbuch der Mathematik.* Teubner, Leipzig, 1996

[49] D. Timmermann, *CORDIC-Algorithmen, Architekturen und monolithische Realisierung mit Anwendungen in der Bildverarbeitung.* VDI Verlag, Düsseldorf, 1990

[50] A. Purde *Konzeption und Implementierung eines Framebrabbers und des Auswerte-Algorithmus für Speckle-Bilder nach Hariharan-Schwider in FPGAs.* Lehrstuhl für Messsystem- und Sensortechnik, Technische Universität München, 2001

[51] H. Hahn. *Untersuchung und Integration von Berechnungsverfahren elementarer Funktionen auf CORDIC-Basis mit Anwendungen in der adaptiven Signalverarbeitung.* VDI Verlag, Düsseldorf, 1991

[52] R. Andraka. *A survey of CORDIC algorithms for FPGA based computers.* FPGA 98 Monterey CA USA, 1998

[53] J. S. Walther. *A unified algorithm for elementary functions.* Spring Joint Computer Conference (SJCC) 1971, S. 379-385

[54] *DIN EN 60617-12.* Beuth Verlag, Berlin, 1999

[55] B. Eschermann. *Funktionaler Entwurf digitaler Schaltungen.* Springer, Berlin, 1993

[56] G. Wunsch, H. Schreiber. *Digitale Systeme.* Springer, Berlin, 1993

[57] *Adder/Subtracter V5.0, Product Specification.* XILINX, www.xilinx.com, 4.10.2001, Stand: 23.11.2002

[58] *Multiplier Generator V5.0, Product Specification.* XILINX, www.xilinx.com, 14.03.2002, Stand: 23.11.2002

[59] E. Zehender. *Vorlesung Rechnerarithmetik.* Universität Jena, Institut für Informatik,
http://www2.informatik.uni-jena.de/ nez/rechnerarithmetik/
folien/021127.pdf, 2000, Stand: 1.12.2002

[60] D. Fey. *Vorlesung Rechnerarithmetik.* Uniiversität-GH Siegen, Institut für Rechnerstrukturen,
http://www.rs.uni-siegen.de/Lehre/skripte/Rechner_Arithmetik/
kap3_73bis96.pdf, 2000, Stand: 1.12.2000

[61] Michael Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions.* Digitale Dissertation an der Technische Universität Berlin, Berlin, 1998

[62] H. Alt, O. Aichholzer, G. Rote. *Matching Shapes with a Reference Point.* International Journal on Computational Geometry and Applications 7 (1997), pp. 349-363, New York, 1997

[63] R. Osada, T. Funkhouser, B. Chazelle, D. Dobkin. *Matching 3D models with shape distributions.* Proceedings International Conference on Shape Modeling and Applications, Los Alamitos, USA, 2001

[64] H. Heuser. *Funktionanalysis.* Teubner, Stuttgart, 1986

[65] H. Alt. *Lineare Funktionanalysis.* 3. ed. Springer Verlag, Berlin, Heidelberg, 1999

[66] W.J. Cocke. *Computer simulation comparisons of speckle image reconstruction techniques.* International Optical Computing Conference, Washington DC, 1980

[67] P. Teichmann. *Einsatz von Filtertechniken in der Speckleinterfermotrie.* Bachelorarbeit am Lehrstuhlk für Messsystem- und Sensortechnik, Technische Universität München, 2001

[68] R. Cusack, J.M. Huntley. H.T. Goldrein. *Improved noise-immune phase-unwrapping algorithm.* Applied-Optics, Vol. 34, p. 781, 1995

[69] J.M. Huntley. *Three-dimensional noise-immune phase unwrapping algorithm.* Applied-Optics, Vol. 40, p. 3901, 2001

[70] B. Gutmann, H. Weber. *Phase unwrapping with the branch-cut method: role of phase-field direction.* Applied-Optics, Vol. 39, p. 4802, 2000

[71] B. Gutmann, H. Weber. *Phase unwrapping with the branch-cut method: clustering of discontinuity sources and reverse simulated annealing.* Applied-Optics, Vol. 38, p. 5577, 1999

[72] R. Schone, O. Schwarz. *Hybrid phase unwrapping algorithm extended by a minimum-cost-matching strategy.* Proc. SPIE - Int. Soc. Opt. Eng. (USA), Vol. 4933, p. 305, 2003

[73] M. Hubig, S. Suchandt, N. Adam. *Equivalence of cost generators for minimum cost flow phase unwrapping.* J. Opt. Soc. Am. A, Opt. Image Sci. Vis. (USA), Vol. 19, p. 64, 2002

[74] J. M. Huntley, J. R. Buckland. *Characterization of sources of 2 pi phase discontinuity in speckle interferograms.* : J. Opt. Soc. Am. A, Opt. Image Sci. Vis. (USA), Vol. 12, p. 1990, 1995

[75] J. R. Buckland, J. M. Huntley. *Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm.* Applied-Optics, Vol. 34, p. 5100, 1995

[76] Jähne, B. *Digitale Bildverarbeitung.* Springer Verlag, Berlin Heidelberg, 1997

[77] E. Vikhagen. *Nondestructive testing by use of TV holography and deformation phase gradient calculation.* Applied Optics, Vol. 29(1), pp. 137-144, 1990

[78] S. Takao, S. Yoneyama, M. Takashi. *Minute displacement and strain analysis using lensless Fourier transformed holographic interferometry.* Opt. Lasers Eng. (UK), Vol. 38, p. 233, 2002

[79] S. Takao, S. Yoneyama, M. Takashi. *Displacement and strain analysis using lensless Fourier-transformed holographic interferometry.* Proc. SPIE - Int. Soc. Opt. Eng. (USA), Vol. 4317, p. 568, 2001

[80] C. W. Chen, H. A. Zebker. *Network approaches to two-dimensional phase unwrapping: intractability and two new algorithms.* J. Opt. Soc. Am. A, Opt. Image Sci. Vis. (USA), Vol. 17, p. 401, 2000

[81] M. Takeda, T. Abe. *Phase unwrapping by a maximum cross-amplitude spanning tree algorithm: a comparative study.* Opt. Eng., Bellingham (USA), Vol. 35, p. 2345, 1996

[82] G. Domik . *Computergraphik.* http://www.uni-paderborn.de/fachbereich/AG/agdomik/computergrafik/cg_skript/html/node1.htm, Stand 10.3.2004

[83] H. Karpat. *Implementierung und Vergleich verschiedener Morphing-Algorithmen.* Diplomarbeit am Lehrstuhl Informatik IX, TU München, 1995

[84] S. F. Oberman, M. J. Flynn. *Division Algorithms and Implementations.* IEEE Transactions on computers, Vol. 46, No. 8, 1997

[85] R. Dumke. *Einführung, Algorithmen und Datenstrukturen.* Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik Institut für Verteilte Systeme, AG Softwaretechnik, http://ivs.cs.uni-magdeburg.de/ dumke/EAD/Skript10.html, Stand: 10.3.2004

[86] C. Reichelt. *Integers out of sort? Program on FPGA to put them in order.* http://www.e-insite.net/ednmag/archives/1997/081597/17df_03.htm, 15.08.1997, Stand: 30.12.2003

[87] W. Bludau. *Halbleiter-Optoelektronik.* Carl Hanser Verlag, München, 1995

[88] H. Bauer. *Lasertechnik.* Vogel Buchverlag, Würzburg, 1991

[89] H. Haken, H.C. Wolf. *Atom- und Quantenphysik.* Springer Verlag, Berlin, 2000

[90] M. Amann. *Tunable Laser Diodes.* Artech House, Boston, 1998

[91] C. Wieman, L. Hollberg. *Using diode lasers for atomic physics.* Rev. Sci. Instrum. 62, Boulder, 1991

[92] T. A. Heumier, J. L. Carlsten. *Mode Hopping in Semiconductor Lasers.* ILX Lightwave, Photonic Text & Measurement Instrumentation, No. 8 Application Note

[93] B. Packroß. *Interferometrie mit Laserdioden.* ITO Band 15, Stuttgart, 1992

[94] ILX Lighwave (Hrsg.). *Application Notes.* ILX Lighwave, Bozeman, 2001-2003, http://ilxlightwave.com/layout.php?topicID=5&subTopicID=21, Stand 26.08.2003

[95] G. Schmidt. *Grundlagen der Regelungstechnik.* Springer Verlag, Berlin, 1991

[96] J. Gißler, M. Schmidt. *Vom Prozess zur Regelung.* Siemens Verlag, Berlin, 1990

[97] R. Isermann. *Identifikation dynamischer Systeme.* Springer Verlag, Berlin, 1992

[98] E. Hecht. *Optik.* Addison-Wesley Publishing Company, New York, 1991

[99] Autor. *Messplatz zur Bestimmung der Kohärenzlänge von Laserstrahlung.* Diplomarbeit, Fachhochschule Wedel, Fachrichtung Physikalische Technik, 1997

[100] L. Hoffmann. *Entwicklung und Realisierung eines digitalen Laserdiodentreibers.* Diplomarbeit, Lehrstuhl für Messsystem- und Sensortechnik an der TU-Muenchen, 2003

[101] Linear Technology (Hrsg.). *LTC2600 - Datasheet.* 2003, http://www.linear.com/pdf/2600f.pdf, Stand: 26.08.2003

[102] U. Tietze, Ch. Schenk. *Halbleiter-Schaltungstechnik.* Springer Verlag, Berlin, 1989

[103] Analog Devices (Hrsg.). *AD626-Datasheet.* http://www.analog.com/ UploadedFiles/Data_Sheets/441354443AD626_d.pdf, Stand: 26.08.2003

[104] *high-speed 3.3v 64k x 36 synchronous bank-switchable dual-port static ram with 3.3v or 2.5v interface idt70v7589s.* Integrated Device Technology, 2002

[105] J. Schwärzler. *Konzeption und Implementierung einer FPGA-basierten Steuerung zur Speicherarbitrierung und zum Kachelausgleich beim Unwrapping von Phasenbildern.* Diplomarbeit am Lehrstuhl für Messsystem- und Sensortechnik an der Technischen Universität München, 2002

[106] Autor. *Titel.* Diplomarbeit am Lehrstuhl für Messsystem- und Sensortechnik an der Technischen Universität München, 2002

[107] T. Bara. *Entwurf und Implementierung eines Client/Server-Konzeptes zur Ansteuerung eines Speckle-Interferometers.* Verlag, Ort, Jahr

[108] C. Nitzl. *3D-Visualisierung und Auswertung technischer Oberflächen im Internet.* Verlag, Ort, Jahr

[109] *Image Processing Toolbox.* Mathworks, Online-Help, 2003

[110] *3-D Visualization.* Mathworks, Online-Help, 2003

[111] H. Bader, W. Huber. *Jini.* Addison-Wesley, München, 1999

[112] J. Palmer. *Essential Java3D fast, Developing 3D Graphics Applications in Java.* Springer-Verlag, London, 2001

[113] SUN (Hrsg.). *Java Products.* http://java.sun.com/products/java-media/3D, Stand: 18.12.2001

[114] SUN (Hrsg.). *Java.* http://java.sun.com, Stand: 18.12.2001

[115] Apache Group (Hrsg.). *Tomcat.* http://apache.org

[116] *eCos Homepage.* http://ecos.sourceware.org/, 2003

[117] *56K* Global Modem PC Card.* http://www.3com.de/pdf/piondat_ger_001.pdf, 1998, Stand: 3.3.2004

[118] *Nokia Cardphone 2.0 Support.* http://www.nokia.com/nokia/0,5184,2381,00.html, Stand: 3.3.2004

[119] *Short Antenna Type II Extended PCMCIA IEEE® 802.11b Wireless LAN Card.* http://www.agere.com/client/docs/DS02305.pdf, 2002, Stand: 3.3.2004

[120] *hr2000.* http://www.oceanoptics.com/products/hr2000.asp, Stand: 3.3.2004

[121] W. R. Stevens. *TCP/IP Illustrated, Vol.1 : The Protocols.* Addison-Wesley Publishing Company, New York, 1994

[122] G. R. Wright, W. R. Stevens. *TCP/IP Illustrated, Vol.2 : The Implementation.* Addison-Wesley Publishing Company, New York, 1995

[123] W. R. Stevens. *TCP/IP Illustrated, Vol.3 : TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols von W. R. Stevens.* Addison-Wesley Publishing Company, New York, 1996