Karlsruher Institut für Technologie

ETP-Bachelor-KA/2022-02

# Automatische Optische Defekterkennung an Siliziumstreifensensoren

## Automatic Optical Defect Recognition on Silicon Strip Sensors

Bachelorarbeit

von

Alexander Salameh

Bearbeitungszeitraum: 24.11.2021-11.04.2022

Referent: Prof. Dr. Ulrich Husemann Institut für Experimentelle Teilchenphysik
Korreferent: Dr. Alexander Dierlamm Institut für Experimentelle Teilchenphysik

Alexander Salameh:
*Automatische Optische Defekterkennung an Siliziumstreifensensoren*
April 2022

# Selbständigkeitserklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

*Karlsruhe, den 11. April 2022*

Alexander Salameh

Diese Arbeit wurde vom Erstgutachter der Bachelorarbeit akzeptiert.

*Karlsruhe, den 11. April 2022*

Prof. Dr. Ulrich Husemann

# Inhaltsverzeichnis

# 1 Einleitung

Großexperimente der Teilchenphysik werden viele Jahre unter schwierigen Bedingungen betrieben. Daher erfordern diese eine industrieähnliche Bauweise, womit ein hohes Maß an Qualitätssicherung einhergeht. Hierzu gehören umfangreiche Tests und Qualitätssicherungsprozesse der verbauten Elemente. Dabei steht insbesondere die Qualitätskontrolle der im Detektor verwendeten Bauteile im Vordergrund, um die Messgenauigkeit des Experiments zu maximieren.

Ein wichtiges Element solcher Detektoren stellen Siliziumstreifensensoren dar. Diese Sensortechnologie findet sich im **C**ompact **M**uon **S**olenoid (CMS) Experiment am **L**arge **H**adron **C**ollider (LHC). Hier bilden Siliziumstreifensensoren den äußeren Spurdetektor, welcher die innersten vier Lagen an Pixeldetektoren umgibt. Das Phase-2-Upgrade des CMS-Spurdetektors der Nutzung des CMS-Detektors im Rahmen des High-Luminosity LHC-Projekts, für welches nach 10 Jahren Laufzeit [Con15] seit 2017 gemäß [Har17] eine integrierte Luminosität von $\mathcal{L} = 3000\,\mathrm{fb}^{-1}$ vorgesehen ist.

Im neuen Spurdetektor kommen zwei verschiedene Sensortypen zum Einsatz. Hierbei handelt es sich zum einen um Makropixelsensoren (PS-p-Sensoren). Diese bestehen aus 32 Reihen mit $1,5\,\mathrm{mm}$ langen Makropixeln, welche einen Abstand von $100\,\mu\mathrm{m}$ zueinander besitzen. Außerdem werden Streifensensoren mit zwei Reihen an jeweils 960 $2{,}5\,\mathrm{cm}$ langen Streifen mit $100\,\mu\mathrm{m}$ Abstand (PS-s-Sensoren) und Streifensensoren mit zwei Reihen an jeweils 1016 $5\,\mathrm{cm}$ langen Streifen mit $90\,\mu\mathrm{m}$ Abstand (2S-Sensoren) eingesetzt. [Con15] Neben der Qualitätskontrolle durch Prüfung der elektronischen Eigenschaften der Sensoren dient eine optische Qualitätskontrolle zur Ermittlung mechanischer Beschädigungen, welche sich negativ auf die elektronischen Eigenschaften auswirken können. Diese Qualitätskontrolle wird von fünf an der Sensorentwicklung beteiligten Instituten durchgeführt. Hierfür wird ein Mikroskop-Scan mit hoher Vergrößerung über den Bereich eines gesamten Sensors durchgeführt. Die sich dabei ergebenden Bilder erfordern jeweils eine Überprüfung zur Sicherstellung der Funktionalität des Sensors. Das Finden von Beschädigungen auf einem solchen Scan des Sensors ist bei manueller Prüfung mit hohem Aufwand verbunden. Daher besteht das Ziel dieser Arbeit in der Entwicklung einer Software zur automatischen Defekterkennung an Siliziumstreifensensoren. Diese soll zur Qualitätskontrolle der Siliziumstreifensensoren für das Phase-2-Upgrade des CMS-Spurdetektors eingesetzt werden. Bei der Entwicklung der Software steht die Ausgabe einer Liste an Bildern des Sensors im Vordergrund, welche oberflächliche Beschädigungen zeigen. Diese Bilder stellen Bereiche des Sensors dar, welche eine nähere Inspektion erfordern, wobei durch die Vorauswahl der benötigte Aufwand erheblich verringert wird. Weiterhin dient die Software zur Gewinnung weiterer Informationen wie der Sensornummer, den Winkeln zwischen den Schnittkanten des Sensors und dem aufgebrachten Aluminium. Ein weiteres Ziel dieser Arbeit ist die übersichtliche Darstellung der gewonnenen Daten.

Die Entwicklung der Software in der Programmiersprache Python basiert auf der Open-Source-Bibliothek **O**pen **C**omputer **V**ision (OpenCV) [Opec] für Bildverarbeitung und maschinelles Sehen. Diese stellt eine Vielzahl an Funktionen aus diesen Bereichen bereit, welche in den zur Ermittlung von Defekten angewandten Algorithmen benötigt werden.

Der Beginn der Arbeit fokussiert sich in Kapitel 2.1 auf die Grundlagen und Eigenschaften von Halbleiterdetektoren als Teilchendetektoren, wobei auf die Funktionsweise und Herstel-

lungsverfahren eingegangen wird. Dabei steht vor allem die in dieser Arbeit zentrale Bauform der Siliziumstreifensensoren im Vordergrund. Das anschließende Kapitel 2.2 basiert auf der Erklärung der für diese Arbeit relevanten Funktionen der automatischen Bildverarbeitung. Hierzu gehört die Beschreibung verschiedener Bildtransformationen sowie die Erläuterung von Verfahren zur Erkennung von Mustern und Formen auf Bildern. Der zweite Teil der Arbeit befasst sich mit der Implementierung der automatischen Defekterkennung sowie der grafischen Benutzeroberfläche. Neben einer Beschreibung der Softwarestruktur sowie den zugehörigen Datenstrukturen in Kapitel 3.1 erfolgt in Kapitel 3.2 die Erklärung der Funktionsweise aller Bestandteile der Software. Dies schließt die Defekterkennung sowie die Funktionen zur Ermittlung weiterer Sensordaten ein. Das darauf folgende Kapitel 4 stellt die mit der Software erzielten Testergebnisse vergleichend dar und befasst sich mit einer geeigneten Parametrisierung des Scanvorgangs. Zuletzt erfolgt in Kapitel 5 eine Zusammenfassung der Ergebnisse der Arbeit im Hinblick auf Funktionalität. Dabei stehen der aktuelle Stand sowie mögliche Verbesserungen im Fokus.

# 2 Theoretische Grundlagen

Dieses Kapitel beschreibt zunächst die Grundlagen zu Siliziumstreifensensoren, für welche die Defekterkennung in dieser Arbeit dient. Der Fokus liegt hierbei auf den physikalischen Vorgängen in Halbleiterdetektoren sowie dem Aufbau von Siliziumstreifensensoren. Weiterhin wird auf das Herstellungsverfahren und die Qualitätskontrolle solcher Sensoren eingegangen.

Der zweite Teil des Kapitels befasst sich mit der Funktionsweise der Methoden zur Bildverarbeitung, welche in dieser Arbeit angewandt werden.

## 2.1 Siliziumstreifensensoren als Teilchendetektoren

Detektoren stellen die zentralen Elemente zahlreicher Experimente der Teilchenphysik dar. Sie dienen in der Grundlagenforschung zur Vermessung der Eigenschaften und Kinematik einzelner Teilchen. Bei kern- und teilchenphysikalischen Experimenten handelt es sich zumeist um Streuexperimente, aus welchen die Eigenschaften und Wechselwirkungen der Teilchen ermittelt werden. Vor der Entwicklung von Teilchenbeschleunigern zur Durchführung gezielter Streuexperimente mit hohen Intensitäten wurden diese mittels radioaktiver und kosmischer Strahlung durchgeführt. [Kol16]

Der Querschnitt von Teilchendetektoren, wie sie in Teilchenbeschleunigern zu Einsatz kommen, ist in Abbildung 2.1 dargestellt. [Kol16] Als Spurdetektor werden in solchen Detektoren beispielsweise Siliziumstreifensensoren eingesetzt.

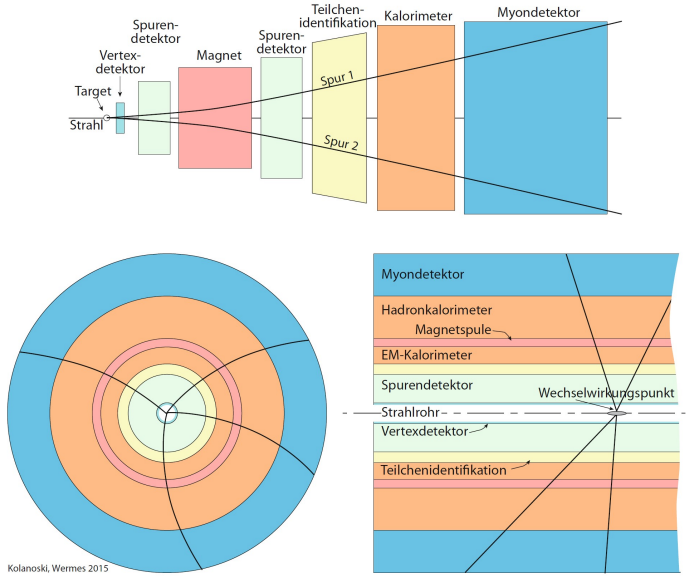



**Abbildung 2.1:** Querschnitt eines Teilchendetektors für Fixed-Target-Experimente (oben) und Collider-Experimente (unten). Aus [Kol16].

### 2.1.1 Aufbau und Funktionsweise von Siliziumstreifensensoren

Die folgenden Erläuterungen orientieren sich an [Kol16].

Siliziumstreifendetektoren beschreiben eine spezielle Bauart von Halbleiterdetektoren. Halbleiter zeichnen sich insbesondere durch ihre Energieniveaus, die Bandstruktur, aus, welche in Abbildung 2.2 im Vergleich mit Isolatoren und Leitern schematisch dargestellt ist. Isolatoren haben hierbei eine sehr große Bandlücke von $E_\mathrm{G} \sim 9\,\mathrm{eV}$, welche eine Anhebung von Ladungsträgern vom Valenz- ins Leitungsband sehr unwahrscheinlich macht. Im Gegensatz dazu weisen Leiter eine Überlappung des Valenz- und Leitungsbands oder ein teilweise gefülltes Leitungsband auf. Dies führt mit sehr geringem Energieaufwand zu Stromfluss. Halbleiter besitzen eine klare Trennung zwischen Valenz- und Leitungsband. Im Gegensatz zu Isolatoren ist die Bandlücke aufgrund der schwächeren Bindung zwischen benachbarten Atomen jedoch wesentlich kleiner. Bei Silizium liegt die Größe der Bandlücke bei $E_\mathrm{G} = 1{,}12\,\mathrm{eV}$. Diese kann durch thermische Vibrationen oder ein äußeres elektrisches Feld überwunden werden, wodurch ein freies Elektron und ein freies Loch entstehen, welche einen Stromfluss ermöglichen. Weiterhin erfolgt die Unterscheidung in direkte und indirekte Halbleiter, welche durch die Energie-Impuls-Beziehung der Elektronen in Valenz- und Leitungsband charakterisiert sind. Direkte Halbleiter wie Galliumarsenid zeichnen sich dadurch aus, dass das Maximum des Valenzbands im Raum der Wellenzahlen ($k$-Raum) über dem Minimum des Leitungsbands liegt. Da dies bei indirekten Halbleitern wie Silizium und Germanium nicht der Fall ist, muss für einen Übergang zwischen den Bändern ein Impulsübertrag an das Kristallgitter erfolgen.

Eine Möglichkeit, die Leitungseigenschaften von Halbleitern zu ändern, besteht in der Dotierung durch Einfügen eines Atoms einer benachbarten Hauptgruppe. Das Einfügen eines fünfwertigen Elements in einen vierwertigen Halbleiter (n-Dotierung) führt hierbei zu einem Überschuss negativer Ladungsträger. Das Einfügen eines dreiwertigen Elements in einen vierwertigen Halbleiter (p-Dotierung) führt hingegen zu einem Überschuss an positiven Ladungsträgern. Eine Dotierung erzeugt im Beispiel von n-dotiertem Silizium zusätzliche Energieniveaus dicht unterhalb des Leitungsbands (Donatorniveaus). Bei p-dotierem Silizium ergeben sich hingegen zusätzliche Energieniveaus dicht oberhalb des Valenzbands (Akzeptorniveaus).

Der in Abbildung 2.3 dargestellte Übergangsbereich zwischen einem p-dotierten und einem n-dotierten Halbleiter zeichnet sich durch eine Verarmung an freien Ladungsträgern aus, da diese dort aufgrund der Diffusion freier Ladungsträger rekombinieren. Diese Zone wird daher als Verarmungs- oder Depletionszone bezeichnet. Durch die ionisierten Atomrümpfe ist die Raumladungsdichte der p-Grenzschicht negativ und die der n-Grenzschicht positiv, was zu einem elektrischen Feld innerhalb der Raumladungszone führt. Dieses erzeugt einen der Diffusion

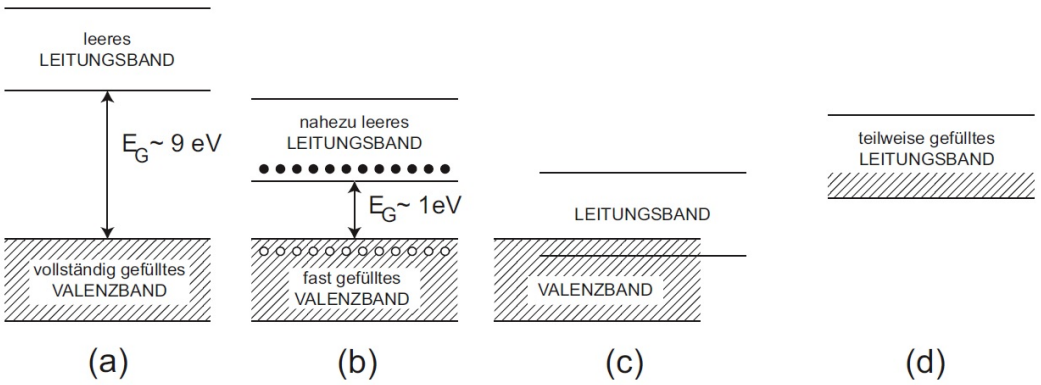



**Abbildung 2.2:** Schematische Energiebandstruktur von Isolatoren (a), Halbleitern (b) und Leitern (c,d). Aus [Kol16].

entgegengerichteten Driftstrom. Bei Anlegen einer externen Spannung mit Stromfluss von p zu n resultiert eine Verkleinerung der Raumladungszone durch Beeinflussung der Diffusion freier Ladungsträger, was freien Stromfluss ermöglicht. Ein entgegengerichteter Stromfluss führt hingegen zu einer Vergrößerung der Raumladungszone, wodurch freier Stromfluss verhindert wird.

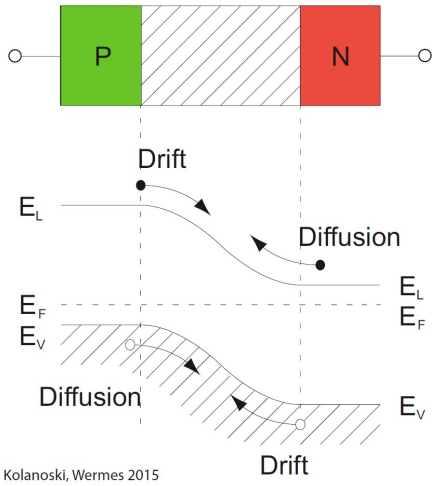



**Abbildung 2.3:** Übergangsbereich zwischen einem p-dotierten und einem n-dotierten Halbleiter. Aus [Kol16].

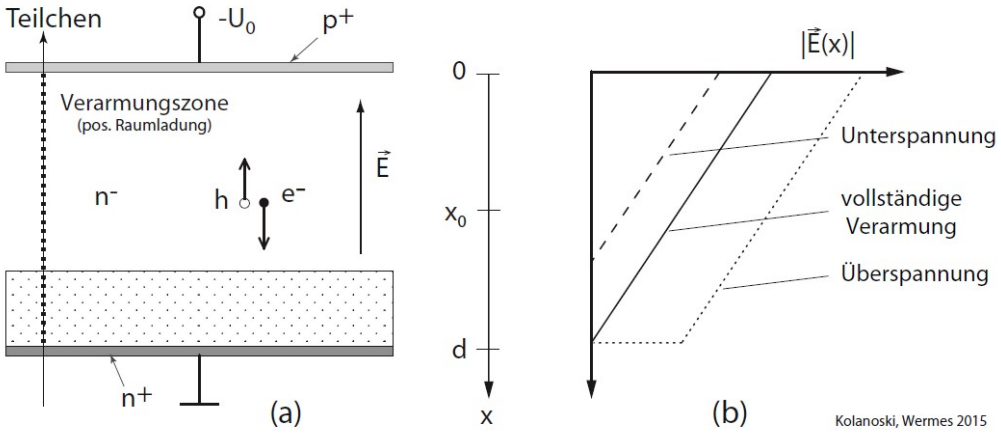



**Abbildung 2.4:** Schematischer Aufbau eines Siliziumdetektors mit unvollständiger Verarmung durch Unterspannung (a) und zugehöriger elektrischer Feldstärkeverlauf (b). Aus [Kol16].

Durch Absorption oder Energieverlust eines das Sensormaterial durchquerenden Teilchens entstehen im Halbleiter Elektron-Loch-Paare. Diese erzeugen, wie in Abbildung 2.4 gezeigt, durch Drift zu den Elektroden ein messbares Influenzsignal. Hierfür ist eine Spannung erforderlich, welche oberhalb der zur vollständigen Verarmung notwendigen Spannung $U_{\text{dep}}$ liegt, um volle Signaleffizienz und die Ankunft möglichst vieler Ladungsträger an der Elektrode zu gewährleisten. [Kol16]

Streifendetektoren bezeichnen eine spezielle Bauform der einseitig prozessierten pn-Detektoren. Diese erfordern die Unterscheidung von p-in-n-Sensoren, bei welchen p-dotierte Streifen in einen

n-dotierten Halbleiter eingefasst sind, sowie n-in-p-Sensoren, bei welchen n-dotierte Streifen in einen p-dotierten Halbleiter eingefasst sind. [Har17] Letztere besitzen den in Abbildung 2.5 dargestellten schematischen Aufbau. Die Unterseite des Detektors bildet der Rückkontakt. Dieser ist über eine sehr dünne ($\sim 10\,\mathrm{nm}$) stark p-dotierte ($p^{++}$) Schicht an das schwach p-dotierte ($p^{+}$) Substrat (p-bulk) gekoppelt. Dies ist notwendig, um die Potentialbarriere zwischen Halbleiter und Rückkontakt möglichst schmal zu halten, damit diese bereits mit geringer angelegter Spannung durchtunnelt werden kann. Dieser Kontakt wird gemäß des geringen Kontaktwiderstands durch die kleinere Austrittsarbeit im Metall als im Halbleiter als ohmscher Kontakt bezeichnet. An den schwach p-dotierten Halbleiter schließen n-dotierte ($n^{+}$) Streifen an, auf welchen sich Aluminiumstreifen befinden. Die kapazitive Kopplung zur Auslese ist in Form einer Siliziumdioxidschicht zwischen den n-dotierten Streifen und den Aluminiumstreifen realisiert. Der thermische Leckstrom wird hierbei über die Spannungsversorgung abgeführt. [Kol16] [Har17]

n-in-p-Sensoren erfordern zusätzliche stark p-dotierte $p^{+}$-Implantate, welche die n-dotierten Streifen umlaufen. Diese verhindern Kurzschlüsse von Streifen durch die Akkumulation von Elektronen am Oxid zwischen benachbarten Streifen. [Har17]

Eine Möglichkeit der hochohmigen Stromversorgung für den Detektor bieten Polysiliziumwiderstände. Das polykristalline Silizium ist hierbei in einer Mäanderstruktur aufgebracht, um durch möglichst lange Leitungen hohe Widerstände erzielen zu können. [Kol16] Diese befinden sich in der Größenordnung $1 - 2\,\mathrm{M\Omega}$.

Die Polysiliziumwiderstände sind mit dem umlaufenden Bias-Ring verbunden, welcher zur Aufrechterhaltung eines homogenen Streifenpotentials dient. Weiterhin dienen ein oder mehrere umlaufende Schutzringe zur Minimierung von Kanteneffekten und Einhaltung der Homogenität des Potentials an den Randstreifen. [Har17]

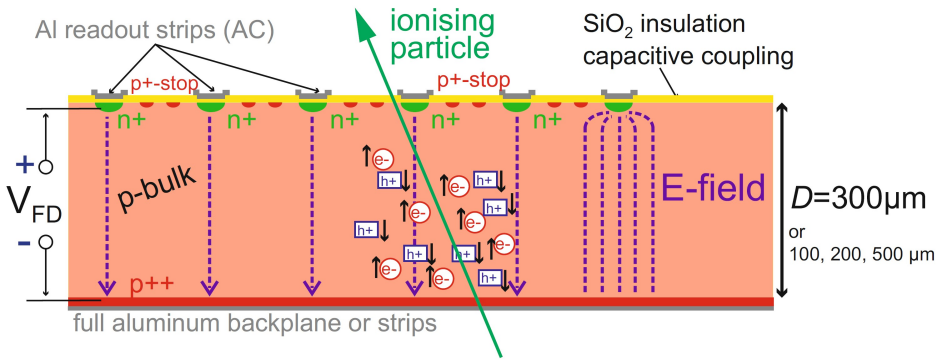



**Abbildung 2.5:** Schematischer Aufbau eines n-in-p-Siliziumstreifensensors. Aus [Har17].

### 2.1.2 Herstellung und Qualitätskontrolle von Siliziumstreifensensoren

Gemäß [Kol16] basiert das Herstellungsverfahren von p-in-n-Siliziumsensoren auf den nachfolgenden Prozessschritten. Für n-in-p-Siliziumsensoren erfolgt ein ähnlicher Herstellungsprozess. Zunächst erfolgt die Strukturierung des Halbleiters, welche in Abbildung 2.6 schrittweise aufgeführt ist. Ein schwach n-dotierter Halbleiter wird hierfür im Ofen bei 1035 °C oxidiert. Die Oberseite des Detektors wird dann mit einem Fotolack beschichtet, welcher durch eine Maske mit der gewünschten Struktur belichtet wird. Nach Entwickeln des Fotolacks wird das freie Oxid mittles Ätzen entfernt und der verbleibende Fotolack abgetragen. Zur Erzeugung einer stark p-dotierten Schicht auf den Streifen wird die Oberfläche wie in Abbildung 2.7 gezeigt mit $5 \times 10^{14}\,\mathrm{cm}^{-2}$ Bor-Atomen mit einer kinetischen Energie von 15 keV beschossen. Durch Erhitzen verdampfen diese von der Oxidschicht und die Fehlstellen im Siliziumgitter werden ausgeheilt. Abschließend werden die Aluminiumkontakte aufgedampft und mittels eines Fotolithografieprozesses analog zu Abbildung 2.6 in die richtige Form gebracht. [Kol16]

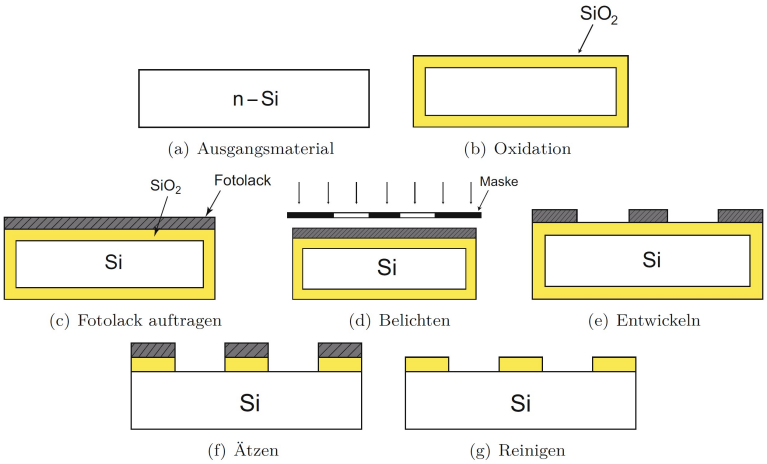



**Abbildung 2.6:** Strukturierung eines Siliziumstreifensensors mittels Fotolithografie. Aus [Kol16].

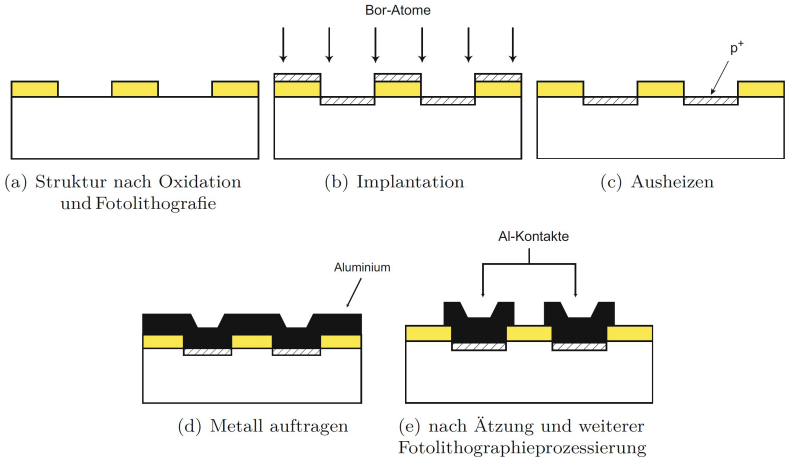



**Abbildung 2.7:** Aufbringen der Streifen auf einen Siliziumstreifensensor. Aus [Kol16].

Die Qualitätskontrolle von Siliziumstreifensensoren setzt sich aus drei Schritten zusammen, der optischen, der elektronischen und der Qualitätskontrolle unter Bestrahlung. Von der Produktion des Sensors wird für die Streifen gefordert, dass mindestens 98-99 % der Streifen den gestellten Anforderungen entsprechen. Hierzu gehört, dass der Leckstrom begrenzt ist und die zur Verarmung notwendige Spannung $U_{\text{dep}}$ innerhalb eines bestimmten Bereiches liegt. Außerdem werden bei der Produktion die Kapazität und der Widerstand zwischen den Streifen, die Kopplungskapazität, der Bias-Widerstand sowie weitere elektronische Parameter festgelegt. [Har17]

Zur optischen Qualitätskontrolle wird der Sensor mittels eines Mikroskops bei hohem Vergrößerungsfaktor inspiziert. Dabei steht die Ermittlung mechanischer Beschädigungen im Vordergrund. Hierzu gehören bei Transport und Handhabung entstehende Kratzer, welche wie in Abbildung 2.8 als Beschädigungen an der Oberfläche des Sensors sichtbar sind. Kratzer, welche mehrere Streifen einschließen, führen möglicherweise zu einer Beeinträchtigung der elektronischen Eigenschaften. Eine optisch erkennbare Beschädigung zeigt sich daher eventuell auch bei der elektronischen Qualitätskontrolle. Die zuverlässige Erkennung optischer Mängel und mechanischer Beschädigungen ist folglich ein wichtiger Schritt in der Qualitätssicherung von Sensoren. Die manuelle Durchführung dieser Inspektion für einen gesamten Sensor ist aufgrund des hierfür nötigen hohen Vergrößerungsfaktors sehr aufwendig. Eine Möglichkeit, den Aufwand der optischen Qualitätskontrolle zu minimieren, besteht in der automatischen Ermittlung möglicher Beschädigungen durch Bildverarbeitung. Dies reduziert die manuelle optische Qualitätskontrolle auf einen geringen Bereich des Sensors.

Die elektronische Qualitätskontrolle beinhaltet gemäß [Har17] folgende Tests der globalen Strom-Spannungs-Charakteristik (IV-Scan) sowie die Durchführung von Kapazitäts-Spannungs-Rampen (CV-Scan) zur Bestimmung von $U_{\text{dep}}$. Zum Test des gesamten Sensors wird ein IV-Scan mit $U \in [0\,\text{V}, U_{\text{IV}_{max}}]$ und ein CV-Scan mit $U \in [0\,\text{V}, U_{\text{CV}_{max}}]$ durchgeführt. Für die einzelnen Streifen werden in der Regel die Leckströme jedes Streifen bei der Spannung $U_{\text{strip}}$ sowie der Bias-Widerstand jedes Streifens gemessen. Zur Identifikation von Pinholes, Löchern in der Siliziumdioxidschicht zwischen den Streifen und den Aluminiumelektroden, wird der Strom $I_{\text{diel}}$ zwischen diesen Schichten bei der Spannung $U_{\text{diel}}$ gemessen. Zusätzlich wird die Kapazität der Kopplung mittels eines LCR-Meters ermittelt. Zusätzlich werden stichprobenartig oder bei Auftreten einer Abweichung in den vorigen Messungen die Kapazität bei hoher LCR-Frequenz und der Widerstand zwischen Streifen gemessen. [Har17] Die hohe Frequenz bei der Messung resultiert aus der geringen Zwischenstreifenkapazität, da hierbei mit hohen Frequenzen bessere Ergebnisse erzielt werden.

Die Qualitätskontrolle unter Bestrahlung beinhaltet Messungen analog zur elektronischen Qualitätskontrolle, welche an speziellen bestrahlten Testsensoren durchgeführt werden. Dazu gehören IV- und CV-Scans, die Kapazität und der Widerstand zwischen Streifen sowie die Messung weiterer Größen wie der Kopplungskapazität, des Bias-Widerstands und der Leckströme. Zusätzlich wird die Stärke des Rauschens und das Verhältnis zwischen Signal- und Rauschstärke

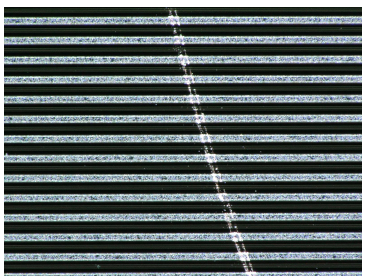


**Abbildung 2.8:** Ein Siliziumstreifensensor mit oberflächlicher Beschädigung.

ermittelt. Im Gegensatz zur elektronischen Qualitätskontrolle müssen hier nicht alle Streifen gemessen werden, da der Vergleich mit unbestrahlten Sensoren im Vordergrund steht. [Har17]

## 2.2 Elemente der automatischen Bildverarbeitung

### 2.2.1 Simple Thresholding

Thresholding beschreibt eine Operation, welche dazu dient, ein Bild in Graustufen in ein binäres Bild umzuwandeln. Dies hat den Nutzen, wichtige Information aus einem Bild mit hohem Kontrast extrahieren zu können. Hierzu wird beim Simple Thresholding ein bestimmter Schwellwert zwischen 0 und 255, der Threshold Value, festgelegt, welcher die Schwelle markiert, ab der ein Helligkeitswert dem höreren Wert zugeordnet werden soll. Alle Helligkeitswerte unter dem Schwellwert werden dem Wert 0 zugeordnet. [Sha20]

Diese Operation und ihr Inverses lassen sich gemäß [Opeb] darstellen als

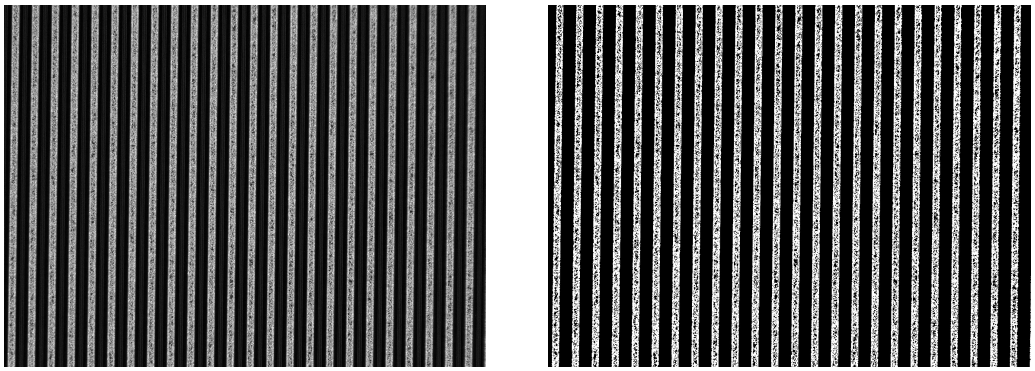
$$\mathrm{dst}(x,y) = \begin{cases} \mathrm{maxval} & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$\mathrm{dst}(x,y) = \begin{cases} 0 & \text{if } \mathrm{src}(x,y) > \mathrm{thresh} \\ \mathrm{maxval} & \text{otherwise} \end{cases}, \tag{2.2}$$

wobei $\mathrm{dst}(x,y)$ den sich ergebenden Wert des Pixels mit den Koordinaten $(x,y)$, $\mathrm{src}(x,y)$ die ursprüngliche Helligkeit des Pixels und thresh den Schwellwert bezeichnet.

Am in Abbildung 2.9 dargestellten Beispiel lässt sich erkennen, dass die dunklen Streifen nach Anwendung des Simple Thresholding einfarbig sind, was eine Weiterverarbeitung des Bildes stark vereinfacht.

Ein Nachteil dieses Verfahrens besteht darin, dass je nach Intensitätsverteilung des Bildes wichtige Informationen verloren gehen können. Dieser Effekt lässt sich durch Nutzung der auf Simple Thresholding aufbauenden Varianten Adaptive Thresholding und Otsu's Binarization minimieren. Adaptive Thresholding beschreibt hierbei ein Verfahren, bei welchem der bezüglich einer Region um den Pixel ideale Schwellwert automatisch ermittelt wird. Dies dient dazu, die Ergebnisse des Thresholding für ein Bild variierender Intensität zu verbessern. Otsu's Binarization vereinfacht das Simple Thresholding durch das automatische Ermitteln des idealen Schwellwerts durch Histogrammieren der Helligkeitswerte des Bildes. [Sha20]



**(a)** Bild in Graustufen. **(b)** Simple Thresholding.

**Abbildung 2.9:** Ein Bild in Graustufen (a) und angewandtem Simple Thresholding mit einem Schwellwert von 127 (b).

### 2.2.2 Non-Local Means Denoising

Denoising dient zur Rauschminderung auf einem Bild. Rauschen tritt durch Zufälligkeiten im Kamerasensor auf und kann durch Weiterverarbeitung des Bildes verstärkt werden. Eine Art von Denoising-Verfahren wird durch das Non-Local Means Denoising-Verfahren beschrieben. [Bua11] Dieses basiert auf der Ersetzung der Farbe des Pixels mit dem mittleren Farbwert ähnlicher Pixel aus einem größeren Bereich um den betreffenden Pixel. Im Gegensatz zu anderen Verfahren besitzt das Non-Local Means Denoising-Verfahren den Vorteil, dass sich hierbei ein besseres Ergebnis für periodische Muster oder Kanten ergibt. [Bua11]

Dies lässt sich nach [Bua11] (Klammer ergänzt) beschreiben durch

$$NLu(p) = \frac{1}{C(p)} \int f(d(B(p), B(q)))u(q)dq. \tag{2.3}$$

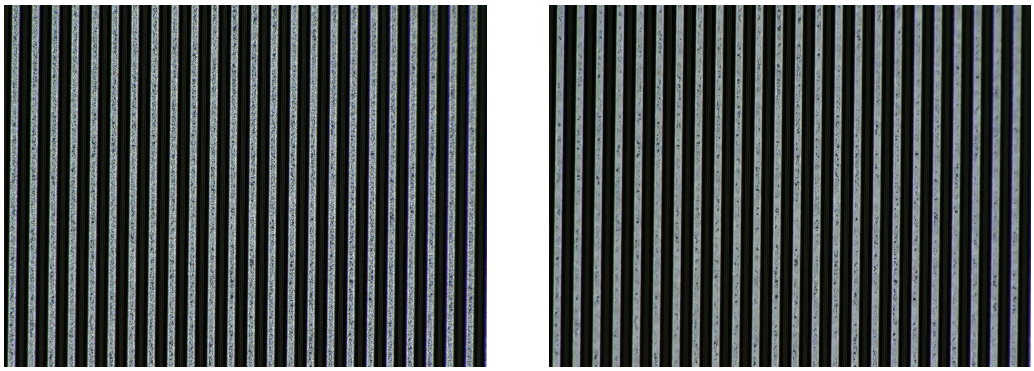
Hierbei stellt $NLu(p)$ den Farbwert des Pixels $p$ mit Denoising dar. Dieser berechnet sich durch das mittels des Normierungsfaktors $C(p)$ normierte Integral über die gewichteten Farbwerte der Pixel $u(q)$ im Bildbereich $B(q)$. Zur Gewichtung der Bildbereiche dient die monoton fallende Funktion $f(d(B(p), B(q)))$, wobei $d(B(p), B(q))$ den Abstand zwischen zwei Bildbereichen mit den Pixeln $p$ und $q$ als Zentrum bezeichnet. [Bua11]

Das Beispiel in Abbildung 2.10 stellt Non-Local Means Denoising in einem Bereich von 21 Pixeln um $p$, einem Bereich von 7 Pixeln zur Berechnung der Gewichtung und einer Filterstärke von 20 für Helligkeit und Farbkomponenten dar. Hierbei zeigt sich eine Verringerung der dunklen Stellen innerhalb der hellen Streifen.

### 2.2.3 Median Blurring

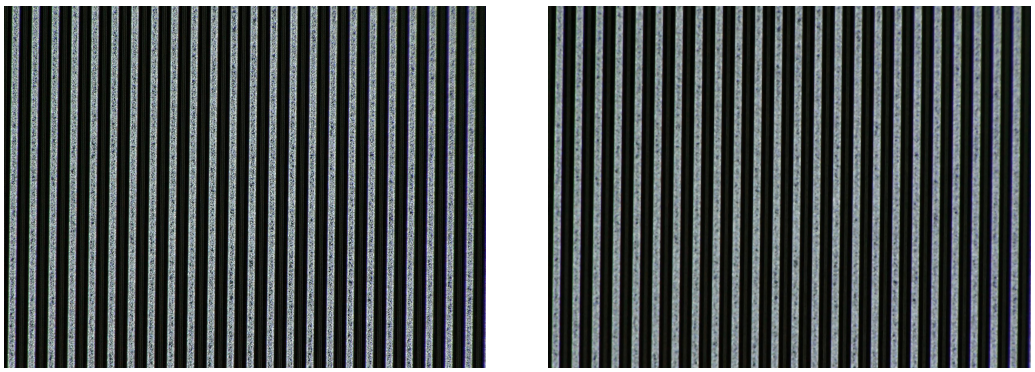
Median Blurring beschreibt eine Methode der Rauschentfernung durch Verwischen des Bildes mittels Mittelwertbildung. Hierbei wird für jeden Pixel des Bildes eine $k \times k$-Matrix der umliegenden Pixel, der sogenannte Kernel, gebildet. Der neue Farbwert des Pixels ergibt sich durch den Median der $k \times k$-Matrix. [Sha20]

Das in Abbildung 2.11 gezeigte Beispiel weist ähnlich zum Non-Local Means Denoising eine Verringerung der dunklen Stellen innerhalb der hellen Streifen auf. Aufgrund der Unabhängigkeit des Verfahrens von Gewichtung nach Farbwerten erscheint das gesamte Bild insbesondere in den Grenzbereichen unklarer als beim Non-Local Means Denoising.



**(a)** Bild in Farbe. **(b)** Non-Local Means Denoising.

**Abbildung 2.10:** Ein Bild in Farbe (a) und angewandtem Non-Local Means Denoising (b).



**(a)** Bild in Farbe. **(b)** Median Blur.

**Abbildung 2.11:** Ein Bild in Farbe (a) und angewandtem Median Blur mit Kernel-Größe 7 (b).

### 2.2.4 Formerkennung mittels Hough-Transformation

Die Hough-Transformation [Kle14] stellt ein Verfahren zur automatischen Erkennung von Formen dar. Hierfür ist zuvor eine automatische Kantenerkennung notwendig, deren Ergebnisse der Hough-Transformation übergeben werden. Das Verfahren der Hough-Transformation am Beispiel einer geraden Linie basiert auf einer Parametrisierung der Geraden, bei welcher die Parameter innerhalb eines begrenzten Bereichs liegen (z.B. Polarkoordinaten). Für jeden Pixel der Kantenerkennung werden die Parameter zu allen Realisierungen ermittelt, auf welchen der Pixel liegen kann. Im Falle einer Geraden entspricht dies allen Geraden, welche den Pixel schneiden. Ist eine Mindestanzahl an Pixeln Teil derselben Geraden, besitzt also dieselbe Kombination an Parametern, gilt diese Gerade als erkannt. Für dieses Beispiel ergibt sich hier als Parametrisierung gemäß [Opea] die in Abbildung 2.12 dargestellte Gerade mit der Gleichung

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{r}{\sin\theta}\right). \tag{2.4}$$

Hieraus folgt für einen bestimmten Punkt $(x_0, y_0)$ nach [Opea] die Menge der Geraden $(r_\theta, \theta)$, welche den Punkt $(x_0, y_0)$ schneiden, als

$$r_\theta = x_0 \cos\theta + y_0 \sin\theta, \tag{2.5}$$

was sich in Abhängigkeit von $\theta \in [0, 2\pi)$ [Kle14] gemäß Abbildung 2.13a im sogenannten $r\theta$-Hough-Raum darstellen lässt. Wird dieses Verfahren, wie in Abbildung 2.13b gezeigt, für alle Punkte des Bildes weitergeführt, stellt der Schnittpunkt zweier Kurven dar, dass die zugehörigen Punkte auf der durch den Schnittpunkt beschriebenen Gerade liegen. Durch

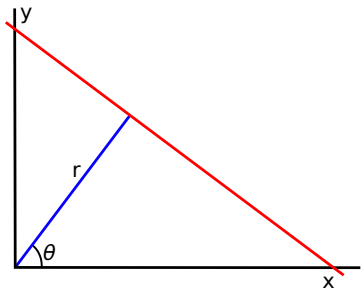
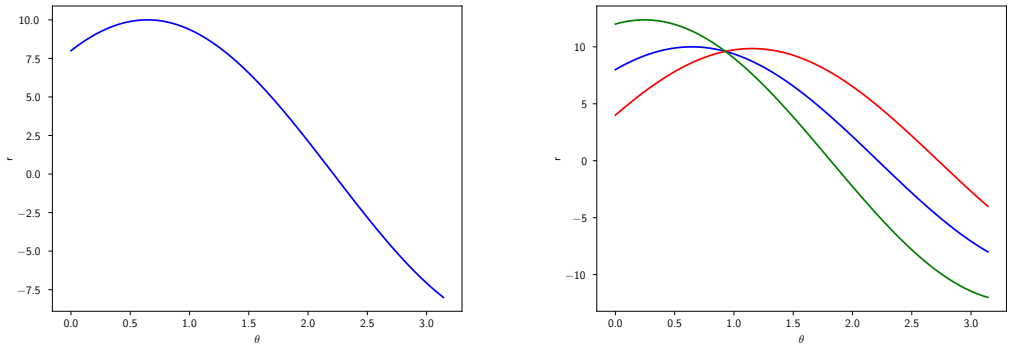
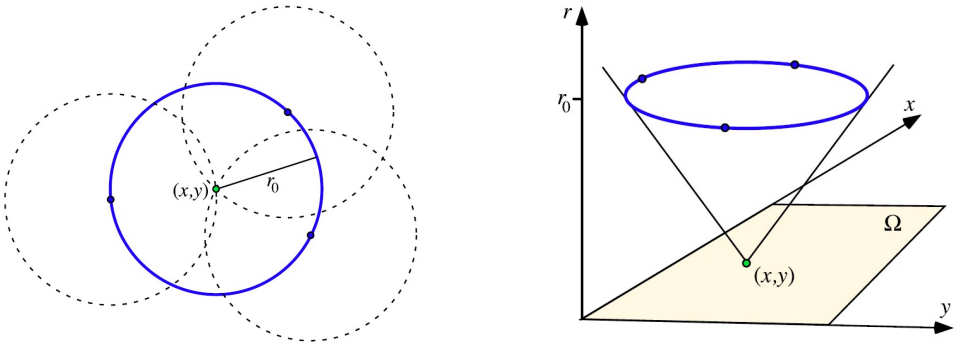



**Abbildung 2.12:** Darstellung einer Geraden in Polarkoordinaten. Nach [Opea].




**(a)** **(b)**

**Abbildung 2.13:** Menge der Geraden, welche den Punkt $(8, 6)$ schneiden (a) und Mengen der Geraden, welche die Punkte $(8, 6)$ (blau), $(4, 9)$ (rot), $(12, 3)$ (grün) schneiden (b). Nach [Opea].




**(a)** **(b)**

**Abbildung 2.14:** Drei Kreise mit Radius $r_0$, welche sich im selben Punkt schneiden und somit einen vierten Kreis mit Radius $r_0$ definieren (a) und Kreis mit Radius $r_0$ als Kegelschnitt im $xyr$-Hough-Raum (b). Aus [Kle14].

Forderung einer Mindestanzahl an Kurven, die sich im selben Punkt schneiden müssen, sind durch diese Schnittpunkte die Geraden im Bild gegeben. [Opea]

Hierbei zeigt sich die Notwendigkeit der Darstellung in Polarkoordinaten, da die Parameter einer Geraden der Form

$$y = mx + c \tag{2.6}$$

jeden Wert aus den reellen Zahlen $\mathbb{R}$ annehmen können. Folglich müssten die Funktionen über einen unendlich großen Bereich evaluiert werden. Da der Winkel $\theta$ in Polarkoordinaten begrenzt ist und die Funktionen periodisch sind, lassen sich diese in Polarkoordinaten über einen begrenzten Bereich evaluieren. [Kle14]

Die Hough-Transformation lässt sich somit für alle weiteren Formen durchführen, welche eine Darstellung mittels begrenzter Parameter besitzen. Eine eindeutige Definition eines Kreises ist gegeben durch das Zentrum $(x_c, y_c)$ und den Radius $r$. Die Diskretisierung des $xyr$-Hough-Raumes ist gegeben durch die Bildgröße sowie die Begrenzung des Suchradius $r$ auf ein Intervall $[r_0, r_1]$ mit $r_0 < r < r_1$. Zur Detektion eines Kreises mit Radius $r = r_0$ wird nach mindestens

drei Kreisen mit Radius $r_0$ gesucht, welche sich im selben Punkt schneiden. Die Mittelpunkte dieser Kreise liegen alle gemäß Abbildung 2.14a auf demselben Kreis und legen diesen somit eindeutig fest. Im $xyr$-Hough-Raum ist dieser Kreis als Kegelschnitt analog zu Abbildung 2.14b gegeben. [Kle14]

Im praktischen Anwendungsfall wird zur Bestimmung der Schnittpunkte ein diskretes Akkumulator-Array als Nullmatrix definiert, dessen Dimension der Anzahl an Parametern entspricht. Ein Akkumulator-Array dient der Berechnung der Schnittpunkte von Kurven durch Zählung der Schnitte dieser Kurven mit den vom Array aufgespannten Zellen. Im Fall der Hough-Transformation repräsentiert dieses Array den entsprechenden Hough-Raum. Für jede eingefügte Kurve wird der Eintrag der von ihr geschnittenen Array-Elemente um 1 erhöht. Hieraus manifestiert sich das Ergebnis der Formerkennung nach Einfügen aller Kurven durch Maxima im Akkumulator-Array. [Kle14]

### 2.2.5 Template Matching

Template Matching bezeichnet in der Bilderkennung die Überprüfung der Existenz eines Bildes in Graustufen $T$ der Größe $w_\mathrm{T} \times h_\mathrm{T}$, des Templates, innerhalb eines Quellbildes in Graustufen $I$ der Größe $w_\mathrm{I} \times h_\mathrm{I}$ mit $w_\mathrm{I} \geq w_\mathrm{T}$ und $h_\mathrm{I} \geq h_\mathrm{T}$. Hiermit lassen sich zuvor bekannte Muster in Bildern detektieren. Dafür wird für jeden Pixel $(x, y)$ des Quellbildes bestimmt, wie sehr das Template mit dem durch $(x, y)$ und $x + w, y + h$ aufgespannten Bildbereich übereinstimmt. Hieraus ergibt sich eine Ergebnismatrix $R$ der Größe $(w_\mathrm{I} - w_\mathrm{T} + 1) \times (h_\mathrm{I} - h_\mathrm{T} + 1)$, deren Einträge die Ähnlichkeit kennzeichnen. [Oped]

Die Ähnlichkeit lässt sich beispielsweise über normierte Korrelationskoeffizienten definieren, diese berechnen sich gemäß [Oped] zu

$$R(x, y) = \frac{\sum_{x',y'}(T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x',y'} T'(x', y')^2 \cdot \sum_{x',y'} I'(x + x', y + y')^2}} \tag{2.7}$$

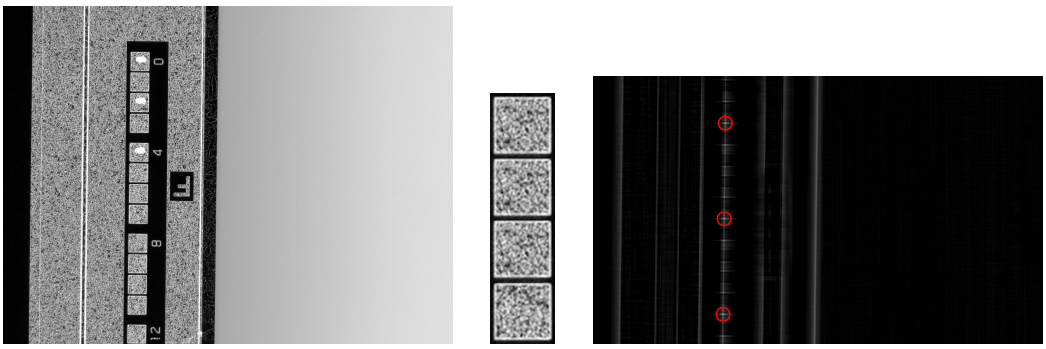
mit $x' \in [0, w]$ und $y' \in [0, h]$ und

$$T'(x', y') = T(x', y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'', y'') \tag{2.8}$$

$$I'(x + x', y + y') = I(x + x', y + y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x + x'', y + y''). \tag{2.9}$$

$T(x, y)$ und $I(x, y)$ bezeichnen hier jeweils den Helligkeitswert des Pixels am Punkt $(x, y)$ von Template bzw. Quellbild. Hieraus resultiert ein Wertebereich der Ergebnismatrix zwischen 0 und 1. Der Punkt, an welchem die maximale Übereinstimmung zwischen Template und Quellbild besteht, ist durch das Maximum der Ergebnismatrix gegeben. Bei mehreren Vorkommen des Templates kann ein Threshold-Wert definiert werden, über welchem die Einträge der Matrix liegen, welche die möglichen Positionen des Templates markieren. [Opee]

Das in Abbildung 2.15 dargestellte Beispiel veranschaulicht die Detektion mehrerer Marker der in Abbildung 2.15b dargestellten Form. Hierbei ergibt sich die in Abbildung 2.15c gezeigte Ergebnismatrix, deren drei maximale Peaks (mit roten Kreisen markiert) mittels eines Threshold-Wertes von $0,6$ herausgefiltert werden.



**(a)** **(b)** **(c)**

**Abbildung 2.15:** Bild, welches drei Vorkommen des Templates enthält (a), Template (b) und die zugehörige Ergebnismatrix (c).

### 2.2.6 OpenCV

OpenCV (**Open C**omputer **V**ision) [Opec] ist eine Open-Source-Bibliothek mit einer Vielzahl an Funktionen aus dem Bereich des Maschinellen Sehens. Dieses beschreibt den Prozess der Transformation von Daten in eine Entscheidung oder eine neue Repräsentation zu einem bestimmten Zweck. Das Maschinelle Sehen wird beispielsweise zur Produktinspektion, Bildverarbeitung in der Medizin, Sicherheit, für User Interfaces, zur Kamerakalibration, Stereosicht und in der Robotik eingesetzt. Zusätzlich zu den Funktionen des Maschinellen Sehens beinhaltet OpenCV eine Unterbibliothek für maschinelles Lernen. Der Ursprung der Bibliothek liegt in einer Forschungsinitiative von Intel mit dem Ziel, prozessorintensive Anwendungen zu verbessern. Die Bibliothek ist in den Programmiersprachen C und C++ geschrieben, außerdem existieren Interfaces zur Nutzung der Bibliothek in Python, Java, MATLAB und weiteren Sprachen. [Kae17]

Die in diesem Kapitel beschriebenen Funktionen besitzen alle eine Implementierung in OpenCV:

- Funktion zur Anwendung von Simple Thresholding (2.2.1):
  **cv::threshold**
- Funktion zur Anwendung von Non-Local Means Denoising (2.2.2):
  **cv::fastNlMeansDenoisingColored**
- Funktion zur Anwendung von Median Blurring (2.2.3):
  **cv::medianBlur**
- Funktion zur Erkennung von Geraden mittels Hough-Transformation (2.2.4):
  **cv::HoughLines**
- Funktion zur Erkennung von Kreisen mittels Hough-Transformation (2.2.4):
  **cv::HoughCircles**
- Funktion zur Anwendung von Template Matching (2.2.5):
  **cv::matchTemplate**

# 3

# Implementierung der automatischen Defekterkennung

Das folgende Kapitel beschreibt zunächst die Strukturierung des Programms, welches im Rahmen dieser Arbeit entstanden ist. Hierbei liegt der Fokus auf dem strukturellen Ablauf der Defekterkennung sowie den bei der Auführung und Speicherung verwendeten Datenstrukturen. Im zweiten Teil des Kapitels erfolgt die Beschreibung der Funktionsweise der einzelnen Bestandteile der Defekterkennung und der Visualisierung. Hierbei zeigt sich unter anderem der Einsatz der in Kapitel 2.2 erklärten Funktionen.

## 3.1 Struktur der Software

Die Software setzt sich aus den in Abbildung 3.1 dargestellten Bestandteilen zusammen. Die Basis der Defekterkennung bilden die Skripte **thresholding_fault_detection.py** zur Lokalisierung und **fault_type_detection.py** zur Charakterisierung der Fehler. Der Aufruf der zugehörigen Funktionen erfolgt über **fault_detection_utils.py**, dessen Methoden neben den Funktionen zum Ausführen der Defekterkennung alle weiteren für die Detektion relevanten Funktionen beinhalten. Dies schließt Funktionen zum Laden der Daten, zur Detektion der Ecken, Zuordnung der Bilder, Schnittkantenvermessung sowie zur Ermittlung der Sensornummer ein. Das Skript **fault_detection.py**, welches das vom Benutzer ausgeführte Skript darstellt, ist für die Ausführung aller Funktionen in **fault_detection_utils.py** und die Speicherung der gewonnenen Daten in **data.txt** verantwortlich. Zur Darstellung der Ergebnisse liest **fault_detection_viewer.py** die Ausgabedatei ein und erzeugt eine grafische Benutzeroberfläche. Diese ermöglicht neben **config.py** ebenfalls das Beschreiben der Konfigurationsdatei **config.ini**, welche alle Parameter des Programms enthält, die zur Einstellung des Scans dienen. Diese sind in acht verschiedene Gruppen aufgeteilt, welche verschiedene Funktionen des Programms mit Werten versorgen. Die Konfigurationsobjekte in der Konfigurationsdatei sind in Tabelle 3.1 aufgeführt.

Bei der Ausführung des Skriptes **fault_detection.py** liest dieses zunächst die benötigten Werte aus **config.ini** ein. Zu Beginn fordert das Programm ebenfalls zur Bestätigung des in **config.ini** vermerkten Pfades zur Speicherung der Ergebnisse oder Eingabe eines Pfades auf, um die entsprechenden Unterordner zu erzeugen. Auf die Funktion zum Einlesen der Daten folgt der **main**-Block, welcher dazu dient, die Parallelisierung der Detektion korrekt auszuführen. In diesem wird die Eingabe in der Kommandozeile auf Argumente geprüft, welche die Ausführung der Lokalisierung oder Charakterisierung verhindern. Die Detektion beginnt mit dem Laden der Bilddaten in eine Datenklasse *Data*, welche die in Tabelle 3.2 beschriebenen Variablen beinhaltet. Die Ergebnisse der Lokalisierung in der Variable fault_location stellen hierbei dar, ob der erste oder zweite Algorithmus einen möglichen Defekt lokalisiert hat (True = möglicher Defekt lokalisiert). Das Resultat der Charakterisierung in der Variable fault_type zeigt, ob diese einen Defekt erkannt hat (True = Defekt erkannt). Für die Detektion der Ecken erfolgt ein Zugriff auf die entsprechende Funktion in **fault_detection_utils.py**, deren Ergebnisse

zusätzlich zur Bestimmung der Höhe und Breite des Sensors in Einheiten von Bildern dienen. Die hierbei gewonnenen Ergebnisse werden in *Data* gespeichert. Die Schnittkantenvermessung sowie die Erkennung der Sensornummer nutzen ebenfalls die entsprechenden Funktionen aus **fault_detection_utils.py**, wobei die Speicherung der Rückgabewerte in *Data* erfolgt. Daraufhin ruft das Skript die Lokalisierung der Fehler auf. Ist diese mittels der eingegebenen Argumente verhindert, wird die Datenklasse so befüllt, als würden keine Fehler detektiert und die Daten gespeichert. Erfolgt die Lokalisierung, startet **fault_detection_utils.py** den parallelisierten Prozess. Die hierbei verwendete Anzahl an Prozessorkernen ist in **config.ini** festgelegt. Nach Rückgabe von *Data* folgt die Ausführung der Charakterisierung, sofern diese nicht durch eingegebene Argumente unterbunden ist. Diese wird ausschließlich für die Bilder durchgeführt, bei denen die Lokalisierung eine Abweichung vom Streifenmuster detektiert. Die anschließende Speicherung der Daten erfolgt mittels der durch die Charakterisierung vollständige Datenklasse *Data*. Die hierin gespeicherten Parameter werden mit der in Abbildung 3.2 anhand eines Beispiels dargestellten Struktur in **data.txt** gespeichert. Die Überschriften dienen beim Auslesen der Datei zur Sortierung der Daten.

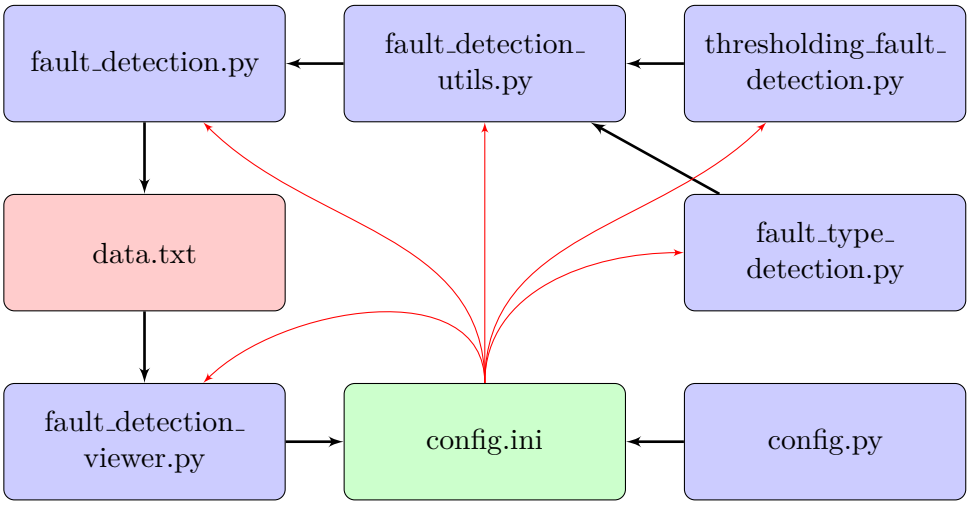



**Abbildung 3.1:** Strukturdiagramm der Software.

**Tabelle 3.1:** Bestandteile von **config.ini**.

| Gruppe | Inhalt |
|---|---|
| PATH | Verzeichnisse zum Scan und zur Darstellung der Ergebnisse |
| THRESHOLDING_VALUES | Parameter zur Lokalisierung der Fehler |
| CORNER_VALUES | Parameter zur Erkennung der Ecken |
| ANGLE_VALUES | Parameter zur Schnittkantenvermessung |
| NUMBER_VALUES | Parameter zur Bestimmung der Sensornummer |
| TYPE_VALUES | Parameter zur Charakterisierung der Fehler |
| MULTIPROCESSING | Anzahl der bei der Parallelisierung verwendeten Kerne |
| VIEWER | Parameter zur Einstellung der Anzeige im User Interface |

**Tabelle 3.2:** Struktur der Datenklasse *Data*.

| Variablenname | Datentyp | Beschreibung |
|---|---|---|
| directories | list of string | Pfade zu den Ordnern mit den Bilddateien |
| files | list of string | Pfade zu allen Bildern |
| fault_location | list of [bool,bool] | Ergebnisse der Lokalisierung |
| fault_type | list of bool | Ergebnisse der Charakterisierung |
| width | int | Breite in Einheiten von Bildern |
| height | int | Höhe in Einheiten von Bildern |
| corners | list of int | Indizes der Bilder, welche Ecken zeigen |
| angles | list of float | Ergebnis der Schnittkantenvermessung |
| sensor_number | string | Sensornummer |

```
DIRECTORIES
PfadZuOrdner\front_a\
PfadZuOrdner\front_b\

CORNERS
0
49
3700
3750

SIZE
50
76

ANGLES
-6.391895898183459e-06
8.767593261001732e-06
-4.904213393557478e-06

SENSORNUMBER
000015

DATA
PfadZuOrdner\front_a\Bild1.jpg,[True, True],False
PfadZuOrdner\front_a\Bild2.jpg,[True, False],False
PfadZuOrdner\front_a\Bild3.jpg,[True, True],True
...
PfadZuOrdner\front_b\Bild1.jpg,[False, True],True
PfadZuOrdner\front_b\Bild2.jpg,[False, False],False
PfadZuOrdner\front_b\Bild3.jpg,[True, True],False
...
```

**Abbildung 3.2:** Struktur der Ausgabedatei **data.txt**.

## 3.2 Funktion der Software-Bestandteile

### 3.2.1 Einlesen der Daten

Der erste Schritt der Defekterkennung beschreibt das Einlesen der auf Defekte zu prüfenden Daten. Bei diesen handelt es sich um alle Bilddateien aus dem vollständigen oder teilweisen Scan eines Sensors. Die Auswahl der Daten, welche einzulesen sind, erfolgt über Eingaben in die Kommandozeile. Hier ist zunächst die Entscheidung zu treffen, ob die Daten aus dem in der Konfigurationsdatei hinterlegten Pfad oder Daten aus einem einzugebenden Pfad genutzt werden sollen. Da die Bilddateien eventuell in zwei separaten Ordnern gespeichert sind, erfolgt eine Prüfung auf Unterordner. Enthält das Verzeichnis keinen oder mehr als zwei Unterordner, so besteht die Möglichkeit der Eingabe eines neuen Pfades. Die Pfade zu den Unterordnern sowie zu den dort gespeicherten Bildern werden in jeweils einem Array gespeichert, und in einer neuen Instanz der Datenklasse *Data* zurückgegeben.

### 3.2.2 Funktionsweise der Defekterkennung

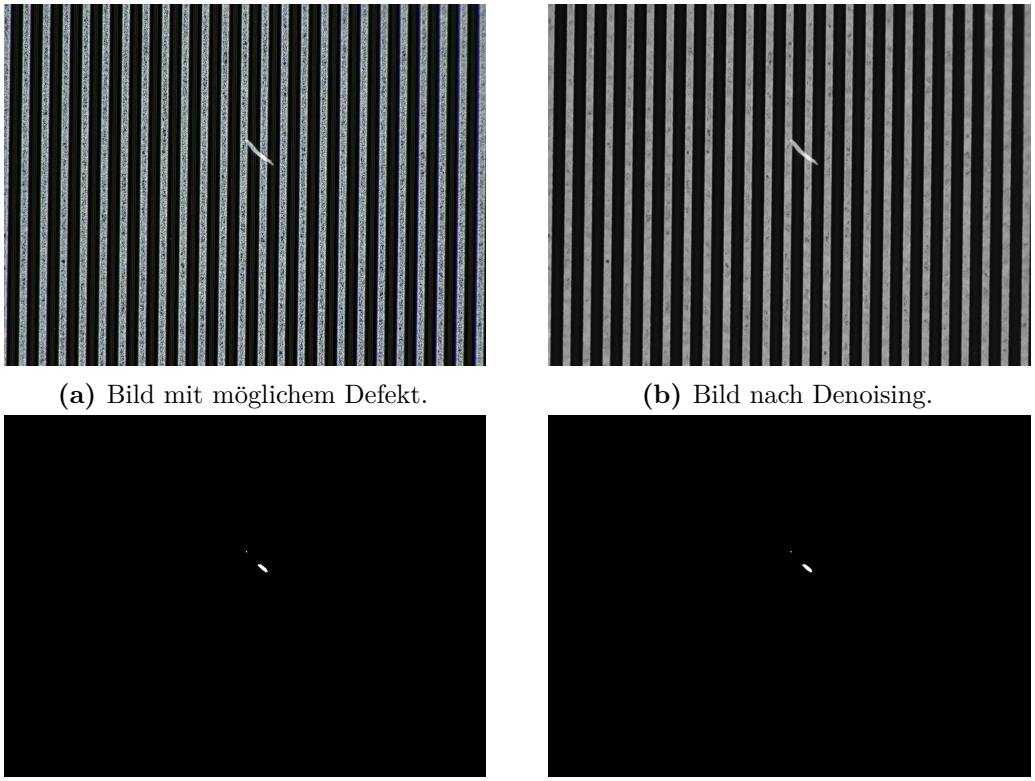
Die Basis der Defekterkennung bildet ein zweistufiges Verfahren, welches sich aus der Lokalisierung und der Charakterisierung von Defekten zusammensetzt. Das Lokalisierungsverfahren ermittelt zunächst die Bilder, welche Abweichungen vom Streifenmuster aufweisen. Hieraus ergibt sich eine Liste von Bildern, welche den Randbereich sowie alle möglichen Defekte auf dem Streifenmuster mit einschließt. Diese Bilder werden im zweiten Schritt der Defekterkennung mittels Vergleich von Farbwerten auf Abnormalitäten untersucht, welche auf einen Kratzer hindeuten.

#### 3.2.2.1 Lokalisierung von Defekten auf Basis von Thresholding

Das Lokalisierungsverfahren nutzt die Eigenschaft binärer Bilder, Übergänge und Abweichungen klar darzustellen. Zur Erkennung solcher Abweichungen dienen zwei verschiedene Algorithmen. Der erste Algorithmus, welcher in Abbildung 3.3 anhand eines Beispiels dargestellt ist, basiert auf der Erkennung besonders heller Stellen auf den Bildern. Hierfür wird auf das Bild in Graustufen zunächst ein Non-Local Means Denoising-Algorithmus angewandt, welcher kleine Bildfehler, welche zu falsch positiven Erkennungen führen, entfernt. Das resultierende Bild wird mit dem Ziel, sehr helle Stellen zu ermitteln, durch Simple Thresholding mit hohem Schwellwert in ein binäres Bild umgewandelt. Zur Entfernung von hellen Stellen mit einer sehr geringen Größe, welche für die weitere Betrachtung daher nicht relevant sind, dient erneut die Anwendung eines Non-Local Means Denoising-Algorithmus. Bleiben weiße Stellen auf dem Bild zurück, gibt der Algorithmus das Ergebnis True aus. Das Beispiel zeigt, dass die besonders hellen Stellen durch die Überdeckung des möglichen Defektes, in diesem Fall einer Faser, mit den hellen Streifen gegeben sind. Im Beispiel wurde ein Schwellwert von 200 verwendet, wobei der mögliche Defekt für Schwellwerte von 188 bis 242 erkannt wird, ohne dass falsch positive Ergebnisse entstehen. Dieser relativ große Bereich unterliegt bei anderer Wahl der Belichtung Schwankungen. Diese können berücksichtigt werden, indem die Schwellwerte für einzelne Bilder manuell ermittelt werden. Hierzu ist es erforderlich, die untere und obere Grenze der korrekten Erkennung durch Anwendung dvon Simple Thresholding mit verschiedenen Schwellwerten einzuschränken, bis sich ein eindeutiger Bereich ergibt. Diese stichprobenweise Ermittlung ist ebenfalls für alle weiteren Schwellwerte des Programms durchzuführen.

Der zweite Algorithmus zur Lokalisierung der Fehler basiert auf der Prüfung, ob das auf dem Bild gezeigte Streifenmuster unterbrochen ist. Diese ist beispielhaft in Abbildung 3.4 gezeigt. Analog zum ersten Algorithmus wird hier ein Non-Local Means Denoising-Algorithmus genutzt. Dieser dient jedoch der Glättung der Struktur der hellen Streifen und besitzt daher separate

Parameter in der Konfigurationsdatei. Die Glättung ist notwendig, um nach Anwendung von Simple Thresholding mit geringerem Schwellwert als beim ersten Algorithmus ein klares Streifenmuster zu erhalten. Um Unterbrechungen im Streifenmuster wie in Abbildung 3.4c zu detektieren, wird das Bild zeilenweise auf diese geprüft. Zur Ermittlung der Übergänge werden zunächst die Stellen im Array der Helligkeitswerte der Zeile gesucht, an welchen die Differenz zwischen zwei Einträgen nicht Null ist. Die Differenz dieser Stellen entspricht der Breite der einzelnen Streifen. Aufgrund der unterschiedlichen Breite der hellen und dunklen Streifen werden diese Breiten dann in ein Array für breite Streifen und ein Array für schmale Streifen eingeordnet. Diese Ordnung wird über den Index der Breite im Array sowie den Anfangswert festgelegt. Unterbrechungen der Streifen zeigen sich durch eine Prüfung der beiden Arrays auf relative Abweichungen der Breiten vom Mittelwert oberhalb eines Grenzwertes. Liegt die relative Abweichung eines Streifens über diesem Grenzwert, gibt der Algorithmus das Ergebnis True zurück. Das in Abbildung 3.5 dargestelle Beispiel zeigt den Plot der Helligkeitswerte einer Zeile mit Unterbrechung eines Streifens.



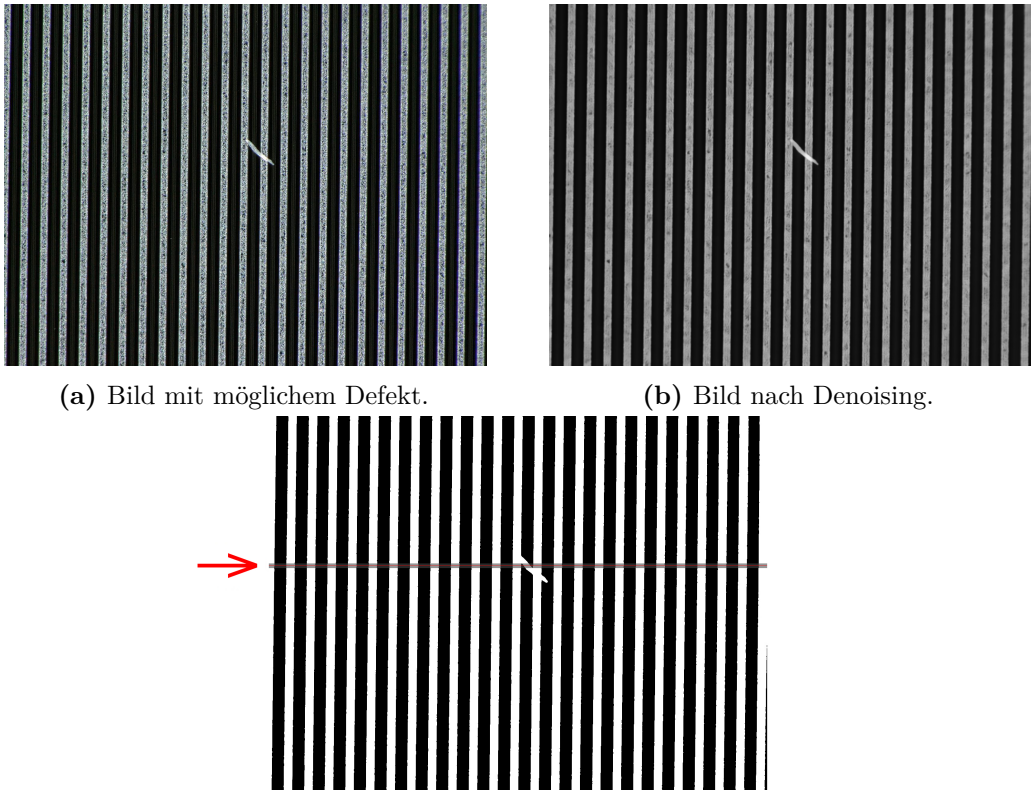
**(a)** Bild mit möglichem Defekt.

**(b)** Bild nach Denoising.

**(c)** Bild nach Thresholding.

**(d)** Bild nach erneutem Denoising.

**Abbildung 3.3:** Ein Bild mit möglichem Defekt (a), nach Non-Local Means Denoising (b), Simple Thresholding mit Schwellwert 200 (c) und weiterem Non-Local Means Denoising (d).



**(a)** Bild mit möglichem Defekt.

**(b)** Bild nach Denoising.

**(c)** Bild nach Thresholding.

**Abbildung 3.4:** Ein Bild mit möglichem Defekt (a), nach Non-Local Means Denoising (b) und Simple Thresholding mit Schwellwert 40 (c). Für andere Belichtungen des Sensors ist eine Anpassung des Schwellwerts erforderlich. Zeile 480, in welcher ein Streifen unterbrochen ist, ist im letzten Bild in rot markiert und zur besseren Sichtbarkeit grau hinterlegt.

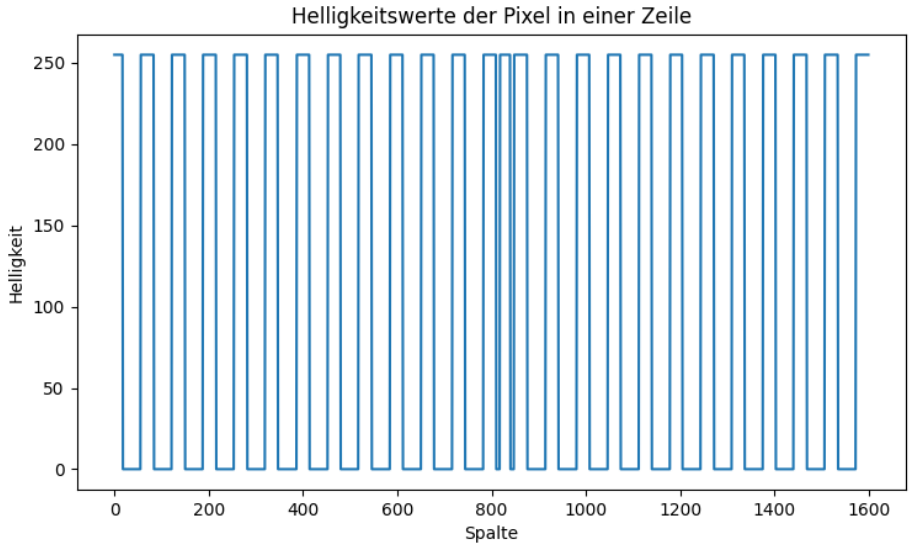



**Abbildung 3.5:** Plot der Helligkeitswerte in Zeile 480 von Abbildung 3.4c.

#### 3.2.2.2 Charakterisierung der Defekte mittels Farbwertvergleichen

Die Charakterisierung der Defekte basiert auf der Annahme, dass sich Kratzer gegenüber Verunreinigungen durch Farbabweichungen manifestieren. Diese weisen besonders hohe rote und gelbe Farbanteile auf, wie die Abbildungen 3.6b und 3.6c beispielhaft zeigen. Zu Beginn des Verfahrens dient ein Non-Local Means Denoising-Algorithmus der Entfernung kleiner Fehler im Bild, welche zu einem falsch positiven Ergebnis führen können. Für die Abweichungen der Farbanteile werden zwei Listen für rote und gelbe Farbanteile generiert, welche jeweils die Koordinaten enthalten, für welche die Bedingung

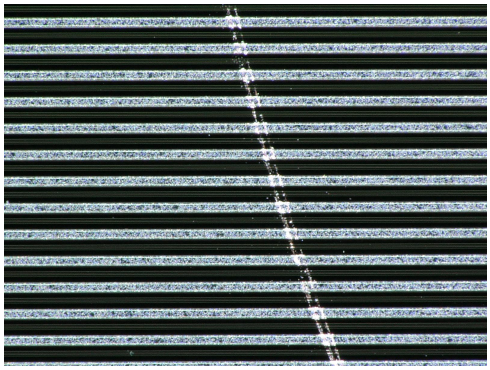
$$F_\mathrm{r} > \frac{F_\mathrm{b} + F_\mathrm{g} + F_\mathrm{r}}{3} + A_r \quad \text{(rote Farbanteile)} \tag{3.1}$$

$$\frac{F_\mathrm{g} + F_\mathrm{r}}{2} > \frac{F_\mathrm{b} + F_\mathrm{g} + F_\mathrm{r}}{3} + A_y \quad \text{(gelbe Farbanteile)} \tag{3.2}$$

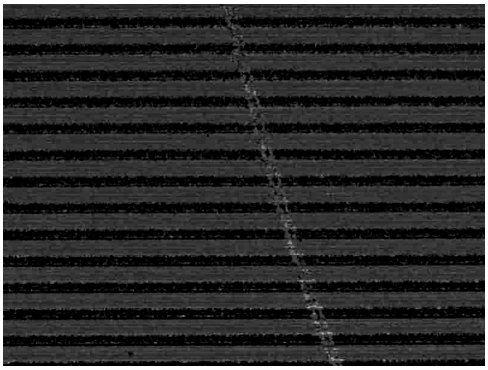
erfüllt ist. Hierbei stellt $F$ den entsprechenden Farbwert und $A$ den Grenzwert der Abweichung dar. Die Koordinaten, für welche die erste Bedingung erfüllt ist, werden auf dem Ausgabebild rot markiert, die Koordinaten, für welche die zweite Bedingung erfüllt ist, werden gelb markiert. Außerdem lässt sich als zusätzliche Bedingung für die Markierung auf dem Ausgabebild ein Bereich für den mittleren Farbwert einstellen. Dieser ergibt sich durch die Bedingung

$$H_\mathrm{min} < \frac{F_\mathrm{b} + F_\mathrm{g} + F_\mathrm{r}}{3} < H_\mathrm{max}, \tag{3.3}$$

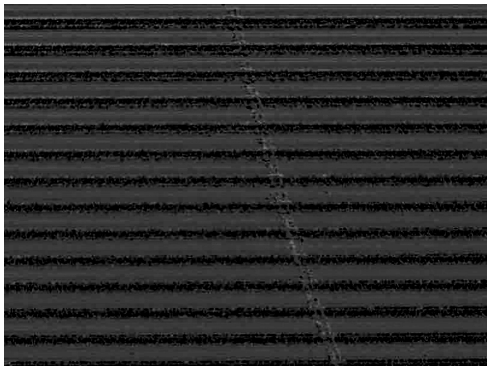
wobei $H_\mathrm{min}$ und $H_\mathrm{max}$ die minimale und maximale erlaubte Helligkeit darstellen. Sind für eine Koordinate alle drei Bedingungen erfüllt, so gilt dies als erkannter Kratzer und die Koordinate wird grün markiert und mit einem grünen Kreis gekennzeichnet. Dies ist am Beispiel in Abbildung 3.6d zu erkennen. In diesem Fall wird das markierte Bild im Ausgabeverzeichnis gespeichert. Abbildung 3.7 zeigt, dass die Unterscheidung von Verunreinigungen und Kratzern zumeist geringere Werte für $A_\mathrm{r}$ und $A_\mathrm{y}$ als in Abbildung 3.6d benötigt. Dies verhindert, dass Verunreinigungen oder die Kanten der Streifen als falsche Defekte erkannt werden, beschränkt die idealen Parameter jedoch auf einen kleinen Bereich, welcher stark von den Scaneinstellungen abhängt. $A_\mathrm{r}$ und $A_\mathrm{y}$ liegen zumeist in einem Bereich von 0-20, die genaue Eingrenzung dieser Parameter wird jedoch wie in Kapitel 4.2 durchgeführt.



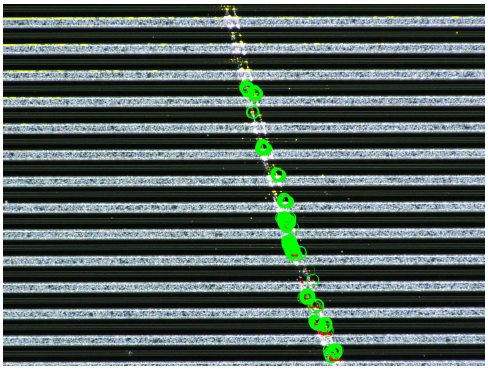
**(a)** Bild mit Kratzer.



**(b)** Abweichung rot-Mittel.

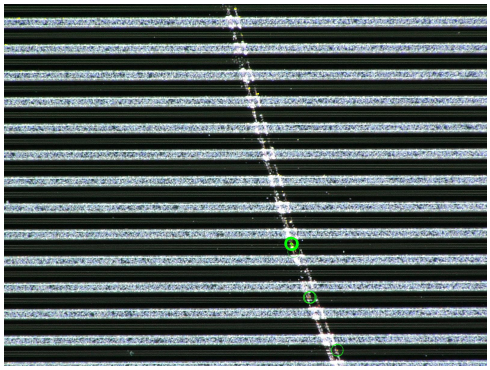


**(c)** Abweichung gelb-Mittel.

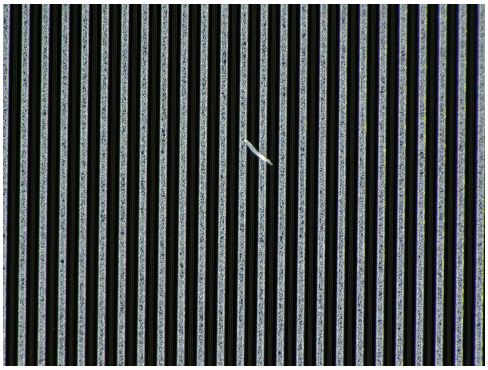


**(d)** Ergebnis der Fehlererkennung.

**Abbildung 3.6:** Ein Bild mit Kratzer (a), die absolute Abweichung des roten Farbanteils vom Mittelwert (b), die absolute Abweichung des gelben Farbanteils vom Mittelwert (c) und das Ergebnis der Fehlererkennung mit den Parametern $A_\mathrm{r} = 3$ und $A_y = 3$ (d). Besonders helle Stellen bei (b) und (c) weisen auf eine positive oder nur gering negative Abweichung hin. Für bessere Erkennbarkeit sind bei diesen Bildern Kontrast und Helligkeit angepasst.



**(a)** Bild mit Kratzer.



**(b)** Bild mit Verunreinigung.

**Abbildung 3.7:** Ein Bild mit Kratzer (a) und ein Bild mit einer Verunreinigung (b), welche mit den Parametern $A_\mathrm{r} = 11$ und $A_y = 6$ auf Defekte geprüft wurden.

### 3.2.3 Erkennung der Ecken mittels Hough-Transformation

Die Erkennung der Ecken des Bildes ist von Relevanz, um die Breite und Höhe des Sensors als Anzahl von Bildern angeben zu können. Hierzu dienen die kreisförmigen Markierungen an den Ecken des Sensors, welche in Abbildung 3.8 gezeigt sind. Die Anwendung eines Median Blurring-Verfahrens entfernt Rauschen auf dem Bild und verringert die Fehleranfälligkeit der Kreiserkennung. Diese wird mittels Hough-Transformation zur Ermittlung von Kreisen durchgeführt. Wird ein entsprechender Kreis detektiert, so wird der Index des Bildes gespeichert. Die Bilder sind hierbei durch das Mikroskop in einer oben links beginnenden Mäanderstruktur fortlaufend nummeriert. Nach dem Scan ist die Eingabe zu tätigen, ob der Scan an der unteren linken Ecke endet. Ist dies nicht der Fall, werden die Indizes der beiden unteren Ecken in der Liste der Ecken vertauschet, da es sonst aufgrund der Reihenfolge zu Problemen bei der Schnittkantenvermessung kommt. Aus den Indizes der oberen beiden Ecken und der Anzahl der Bilder ergeben sich Breite $w$ und Höhe $h$ zu

$$w = \text{Index oben links} - \text{Index oben rechts} + 1 \tag{3.4}$$

$$h = \frac{w}{\text{Anzahl der Bilder}} \tag{3.5}$$

Im Fall, dass keine vier Ecken detektiert werden, benötigt das Programm die manuelle Eingabe der Indizes. Damit entfällt die Schnittkantenvermessung und die entsprechenden Winkel werden als $[0, 0, 0, 0]$ gespeichert. Werden mehr als vier Ecken detektiert, lassen sich die korrekten Bilder manuell auswählen.

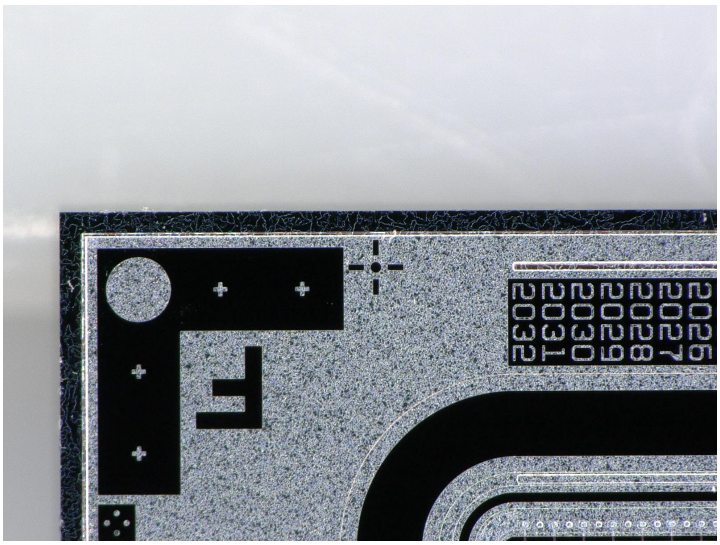


**Abbildung 3.8:** Bild einer Sensorecke mit dem zur Erkennung verwendeten Kreismarker.

### 3.2.4 Zuordnung der Bilder

Die Zuordnung der Bilder nutzt die durch die Erkennung der Ecken gewonnene Breite und Höhe zur Bestimmung der Koordinate $(x, y)$ eines Bildes. Hierzu dienen die Gleichungen

$$y = \frac{i}{w} \tag{3.6}$$

$$x = \begin{cases} i \bmod w & y \text{ gerade} \\ w - i \bmod w - 1 & y \text{ ungerade} \end{cases}, \tag{3.7}$$

wobei $i$ den Index des Bildes beschreibt. Die Fallunterscheidung ist notwendig, da die Positionen auf dem Sensor zum Fotografieren der Bilder, wie in Kapitel 3.2.3 erwähnt, in einer Mäanderstruktur angefahren werden.

### 3.2.5 Schnittkantenvermessung

Die Ermittlung des Winkels zwischen der Sensorkante und dem aufgebrachten Aluminium wird an allen Sensorkanten durchgeführt. Zur Vereinfachung des Verfahrens bestimmt die Schnittkantenvermessung den Abstand zwischen der Oberkante und der kreisförmigen Markierung an den Ecken, welche in Abbildung 3.9 dargestellt sind. Zur Vermessung links und rechts werden die entsprechenden Bilder um 90° rotiert. Für die Vermessung der Unterkante werden die Bilder um 180° gedreht. Falls eines der Bilder eine Schnittkante nicht vollständig abbildet, erfolgt die Ausgabe einer Warnung, dass das Ergebnis möglicherweise nicht aussagekräftig ist.

Zur Bestimmung des Abstands zwischen der Markierung und der Sensorkante erfolgt zunächst eine Detektion des Kreismittelpunkts $(x_\mathrm{c}, y_\mathrm{c})$ und -radius der kreisförmigen Markierung an einer Ecke mittels Hough-Transformation. Da das Bild den Sensor nicht zwangsläufig in gerader Ausrichtung zeigt, wird das Bild unterhalb des Kreises abgeschnitten. Dies dient dazu, dass bei der Detektion der Oberkante die seitliche Kante nicht berücksichtigt wird. Dieses Bild wird nun mittels Simple Thresholding in ein binäres Bild überführt, welches einen klaren Kontrast zwischen Unterlage und Sensorkante bietet. Zur Bestimmung der Oberkante wird für jede Spalte des Bildes der Übergang ermittelt und in einem Array gespeichert. Dieses Array wird zur weiteren Filterung der seitlichen Kante von allen Werten bereinigt, welche eine zu hohe Differenz zum benachbarten Wert aufweisen. Eine lineare Regression dient zur Überführung der verbleibenden Werte in eine Gerade g der Form

$$y = mx + b. \tag{3.8}$$

Zur Bestimmung des Abstandes zwischen dem Mittelpunkt des Kreises und der Geraden wird zunächst eine zu g orthogonale Gerade

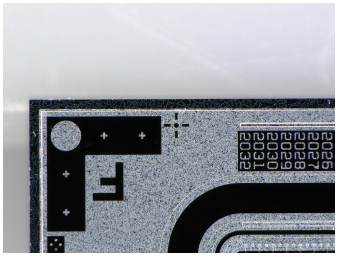
$$y = -\frac{1}{m}x + y_\mathrm{c} + \frac{1}{m}x_\mathrm{c} \tag{3.9}$$

definiert, welche den Mittelpunkt des Kreises schneidet. Aus dem Schnittpunkt der beiden Geraden

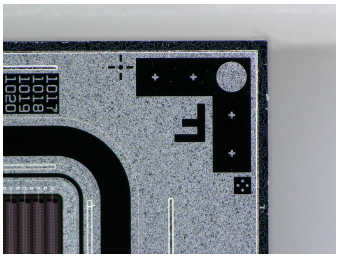
$$(x_\mathrm{i}, y_\mathrm{i}) = \left( \frac{y_\mathrm{c} + \frac{1}{m}x_\mathrm{c} - b}{m + \frac{1}{m}}, m \cdot \frac{y_\mathrm{c} + \frac{1}{m}x_\mathrm{c} - b}{m + \frac{1}{m}} + b \right) \tag{3.10}$$

ergibt sich der Abstand zwischen Mittelpunkt des Kreises und der Geraden in Einheiten von Pixeln zu

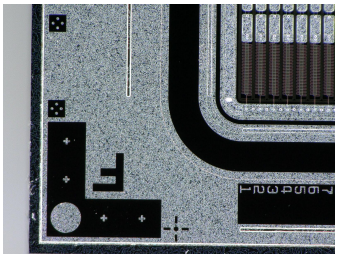
$$d = \sqrt{(x_\mathrm{i} - x_\mathrm{c})^2 + (y_\mathrm{i} - y_\mathrm{c})^2}. \tag{3.11}$$



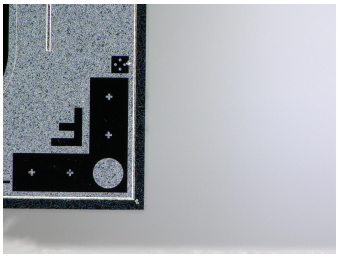
**(a)** Obere linke Ecke.



**(b)** Obere rechte Ecke.



**(c)** Untere linke Ecke.



**(d)** Untere rechte Ecke.

**Abbildung 3.9:** Vier Bilder der Ecken eines Sensors.

Die Berechnung des Schnittkantenwinkels erfordert die Bestimmung der Abstände $d_1$ und $d_2$ bei den an die Schnittkante anliegenden Ecken sowie die Umrechnung des Abstandes $a$ in µm zwischen den beiden kreisförmigen Markierungen in Einheiten von Pixeln. Diese Umrechnung ist erforderlich, damit der Abstand in der Konfigurationsdatei in µm angegeben werden kann. Die Abstände der Markierungen unterscheiden sich je nachdem, ob die Schnittkantenvermessung an der Oberkante oder den seitlichen Kanten durchgeführt wird. Aus diesen Größen ergibt sich der Winkel gemäß

$$\alpha = \arctan \frac{d_2 - d_1}{a}. \tag{3.12}$$

Das Verfahren ist in Abbildung 3.10 am Beispiel der in Abbildung 3.9b gezeigten Ecke dargestellt. Für die auf dieser Abbildung gezeigte Vergrößerung (150-fach) beträgt die maximale Auflösung der Schnittkantenvermessung ungefähr 1,36 µm (13,6 µrad auf 10 cm), was der Größe eines Pixels entspricht. Für die geringere Vergrößerung des Bildes in Abbildung 3.15 (100-fach) ergibt sich hingegen eine maximale Auflösung von rund 2,05 µm (20,5 µrad auf 10 cm). Damit lassen sich Abweichungen der Schnittkante ermitteln, welche kleiner als die Mindestanforderung von 5 µm (50 µrad auf 10 cm) sind.

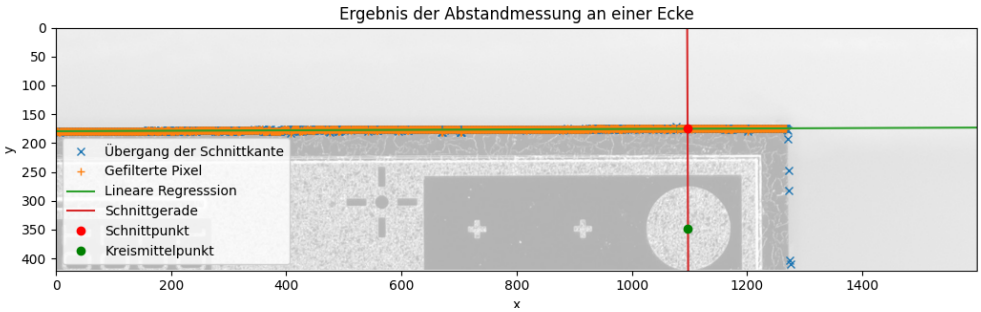


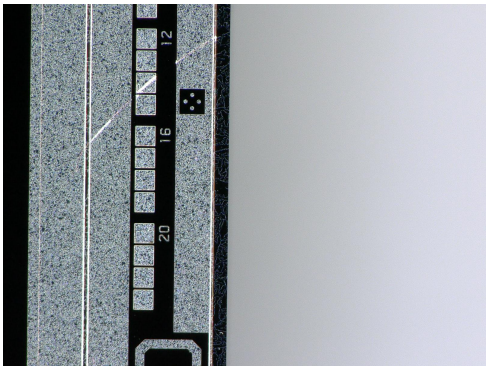

**Abbildung 3.10:** Ergebnis des Verfahrens zur Abstandmessung bei der in Abbildung 3.9b dargestellten Ecke.

### 3.2.6 Erkennung der Sensornummer

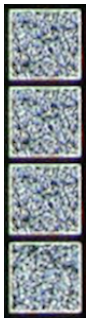
Die Erkennung der Sensornummer erfordert das Finden der entsprechenden sechs Markierungen (Scratchpads) im Randbereich des Sensors, welche in Abbildung 3.11a und 3.11b dargestellt sind. Diese ergeben jeweils eine mittels 4-Bit binär kodierte Ziffer zwischen 0 und 9. Die Sensornummer ergibt sich aus diesen sechs Ziffern. Zum Finden der Markierungen wird mittels des in Abbildung 3.11c gezeigten Templates ein Template Matching-Verfahren ausgeführt, wobei nur die Bilder betrachtet werden, welche am rechten Rand des Sensors liegen. Dies führt zu einem Ergebnis wie in Abbildung 3.11e und 3.11f. Die sich hierbei ergebenden Pixel, welche über dem Schwellwert des Template Matching liegen, werden anhand der Differenz ihrer $y$-Koordinate gruppiert. Die Koordinaten des Pixels jeder Gruppe, welcher sich am weitesten oben links befindet, dienen als obere linke Ecke zum Zuschnitt eines Bildes pro Gruppe auf Größe des Templates gemäß Abbildung 3.12.

Im Anschluss wird für jedes der sechs Bilder die zugehörige Ziffer bestimmt. Hierfür wird mit dem in Abbildung 3.11d gezeigten Template Template Matching durchgeführt. Die sich hierbei ergebenden Pixel, welche oberhalb des Schwellwerts liegen, werden analog zum ersten Teil des Prozesses gruppiert. Das Ergebnis ist in Abbildung 3.13 dargestellt. Die entsprechende Ziffer wird durch die Prüfung, in welchen Vierteln des Bildes sich die $y$-Koordinaten der Marker befinden, ermittelt. Diese binäre Darstellung wird in eine Zehnerdarstellung umgerechnet und für alle sechs Bilder zur Sensornummer hintereinandergereiht. Werden mittels Template Matching nicht alle sechs Marker gefunden, ist die Sensornummer manuell einzugeben.

**(a)** Obere Markierungen der Sensornummer.

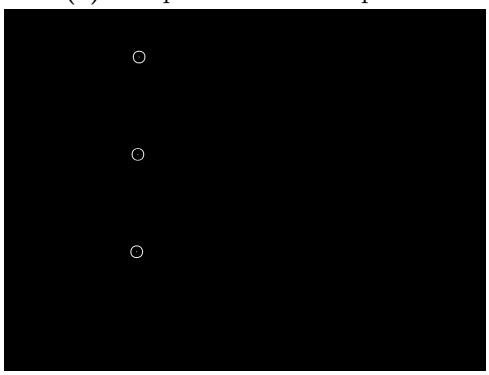


**(b)** Untere Markierungen der Sensornummer.

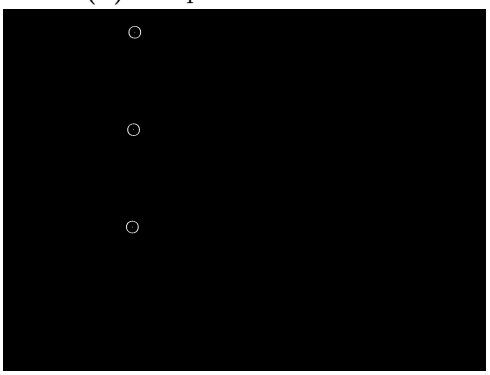


**(c)** Template der Scratchpads.

**(d)** Template des Markers.

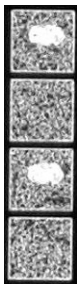


**(e)** Template Matching für (a).

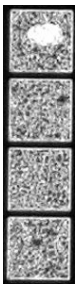


**(f)** Template Matching für (b).

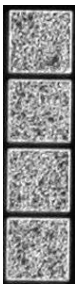
**Abbildung 3.11:** Markierungen der Sensornummer (a,b), Template eines Scratchpads (c), Template des Markers (d) und Ergebnis des Template Matching für die Scratchpads in (a) und (b) (e,f). Zur besseren Erkennbarkeit sind die Punkte mit Kreisen markiert.



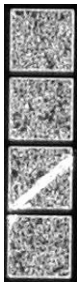
**(a)** Markierung 0.



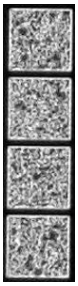
**(b)** Markierung 4.



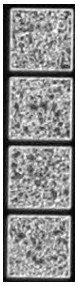
**(c)** Markierung 8.



**(d)** Markierung 12.



**(e)** Markierung 16.



**(f)** Markierung 20.

**Abbildung 3.12:** Mittels Template Matching ermittelte und zugeschnittene Markierungen, welche die Sensornummer 15 darstellen.

**(a)** Marker auf Markierung 0.

**(b)** Marker auf Markierung 4.

**(c)** Marker auf Markierung 12.

**Abbildung 3.13:** Mittels Template Matching ermittelte Positionen der Marker auf den Abbildungen 3.12a (a), 3.12b (b) sowie 3.12d (c). Zur besseren Erkennbarkeit sind die Punkte mit Kreisen markiert. (c) zeigt, dass Kratzer wie auf 3.12d nicht als Marker erkannt werden.

### 3.2.7 Grafische Benutzeroberfläche

Die grafische Benutzeroberfläche besteht aus Komponenten zum Einlesen der Ergebnisse einer Defekterkennung, der Anzeige dieser Ergebnisse sowie einem Fenster zur Einstellung der Scan-Parameter. Beim Start des Skriptes ist die Eingabe des Ordners, in welchem die Resultate gespeichert sind, sowie die Auswahl, ob es sich um einen 2S-Sensor oder PS-s-Sensor handelt, erforderlich. Das Einlesen der Daten aus **data.txt** erfolgt in eine Instanz der Datenklasse *data*. Das in Abbildung 3.14 anhand eines Beispiels dargestellte Hauptfenster der grafischen Benutzeroberfläche zeigt die Ergebnisse eines Scans. Der rechte Teil des Fensters gibt die Ergebnisse der Schnittkantenvermessung und die Sensornummer wieder, wobei bei der Darstellung der Schnittkantenwinkel die in der Konfigurationsdatei angegebene Zahl der Nachkommastellen berücksichtigt wird. Die Legende dient der Deutung des Schaubildes. Hier stellen rote und blaue Quadrate Bilder mit Defekten dar, für welche eine weitere Untersuchung in Frage kommt. Die blaue Farbe markiert hierbei, dass sich der Defekt nicht auf den Streifen befindet. Die grauen Quadrate stellen die Bilder dar, auf welchen ein Fehler lokalisiert, jedoch nicht als Defekt charakterisiert wurde. Auf den Bildern, welche durch die schwarzen Quadrate gekennzeichnet sind, wurde bei der Detektion kein Fehler lokalisiert. Durch Mausklick auf eines der Quadrate öffnet sich das zugehörige Bild mit Spalte $c$, Zeile $l$ und dem berechneten Bereich an Streifennummern $s$. Diese berechnet sich mittels

$$s(\text{oben}) = \left[2n - \left(c\frac{n}{w-2} + \frac{n}{2w}\right) - 2, 2n - \left(c\frac{n}{w-2} - \frac{n}{2w}\right) + 2\right] \tag{3.13}$$

$$s(\text{unten}) = \left[\left(c\frac{n}{w-2} - \frac{n}{2w}\right) - 2, \left(c\frac{n}{w-2} + \frac{n}{2w}\right) + 2\right], \tag{3.14}$$

wobei $n$ die Anzahl der Streifen auf der gesamten Breite des Sensors und $w$ die Breite des Sensors in Einheiten von Bildern beschreibt. Für Bilder am linken und rechten Rand des Sensors entfällt die Anzeige der Streifennummer. Da die Bilder des Sensors nicht zwangsläufig gerade sind, weist die Berechnung eine Abweichung von wenigen Streifen vom tatsächlichen Wert auf, wie Abbildung 3.15 zeigt. Um die Überlappung nebeneinander liegender Bilder zu berücksichtigen, überschneiden sich die Streifennummer-Bereiche an den beiden äußersten Streifen. Die fortlaufende Nummer $n$ des zu öffnenden Bildes hängt aufgrund der Aufnahme der Bilder in einer Mäanderstruktur davon ab, in welcher Richtung die Bilder dieser Zeile aufgenommen sind. Für eine Aufnahme von links nach rechts ergibt sich diese zu

$$n = lw + c, \tag{3.15}$$

für eine Aufnahme von rechts nach links hingegen zu

$$n = (l+1)w - (c+1). \tag{3.16}$$

Für einen PS-s-Sensor, welcher sich aus einer Aufnahme zusammensetzt, gilt für Streifen gerader Zeilennummer Gleichung 3.15 und für Streifen ungerader Zeilennummer Gleichung 3.16. Bei 2S-Sensoren hängt die Zuordnung zusätzlich davon ab, ob die zweite Aufnahme auf der selben Seite beginnt, auf der die erste Aufnahme endet. Ist dies nicht der Fall, hat der Sensor eine in Einheiten von Bildern ungerade Höhe, was dazu führt, dass ab der Hälfte der Bilder die umgekehrte Zuordnung gilt. Beim Öffnen eines Bildes mit Defekt wird dieses aus dem Verzeichnis mit den Resultaten geladen, da die Defekte auf den Bildern in diesem Verzeichnis mit grünen Kreisen markiert sind. Für andere Bilder wird hingegen auf den Pfad zurückgegriffen, über welchen die Bilder beim Scan geladen wurden. Ist dieser Pfad nicht verfügbar, wird eine Fehlermeldung ausgegeben.

Über die Schaltfläche „Edit scan parameters“ öffnet sich das Fenster zur Einstellung der Scan-Parameter, welches in Abbildung 3.16 dargestellt ist. Dieses ermöglicht die Einstellung aller in der Konfigurationsdatei definierten Parameter. Der erlaubte Datentyp und Wertebereich ist hierbei mit angegeben. Die Schaltfläche „Save“ dient dem Überschreiben der Konfigurationsdatei mit den aktuell eingetragenen Werten. Das Schließen des Fensters bewirkt ein Abbrechen des Bearbeitungsprozesses, die eingetragenen Werte bleiben jedoch bis zum Neustart des Programms erhalten.

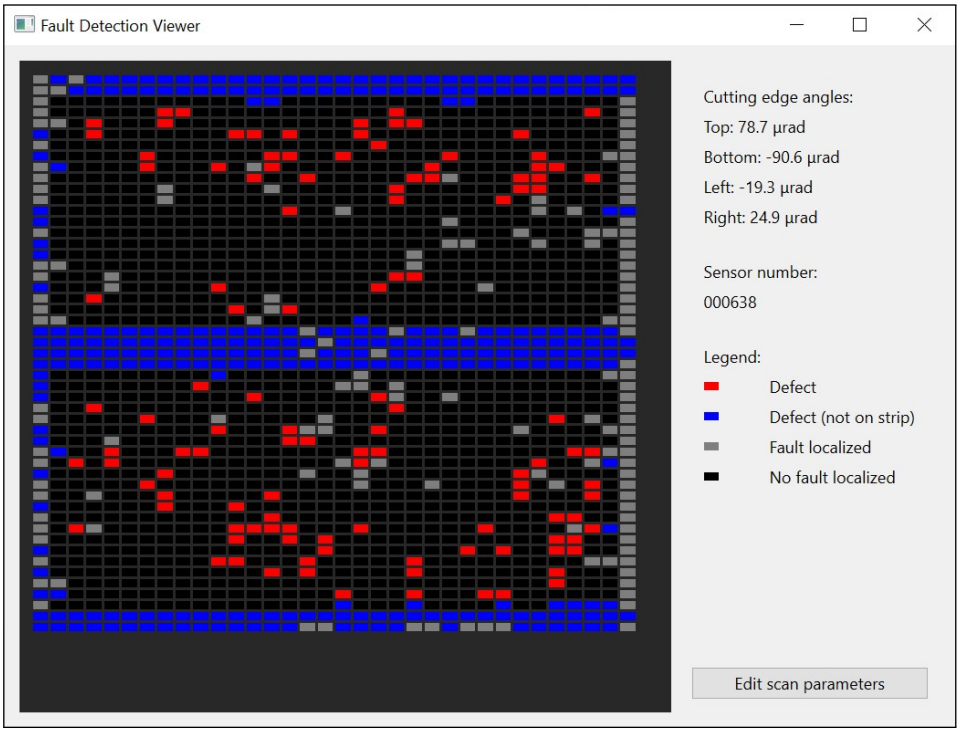



**Abbildung 3.14:** Darstellung der Ergebnisse in der grafischen Benutzeroberfläche. Die Laufzeit der Lokalisierung und Charakterisierung des abgebildeten Scans betrug bei Parallelisierung auf acht Kernen insgesamt ungefähr eine Stunde und sechs Minuten (Intel Core i7 9700F, 16 GB RAM).

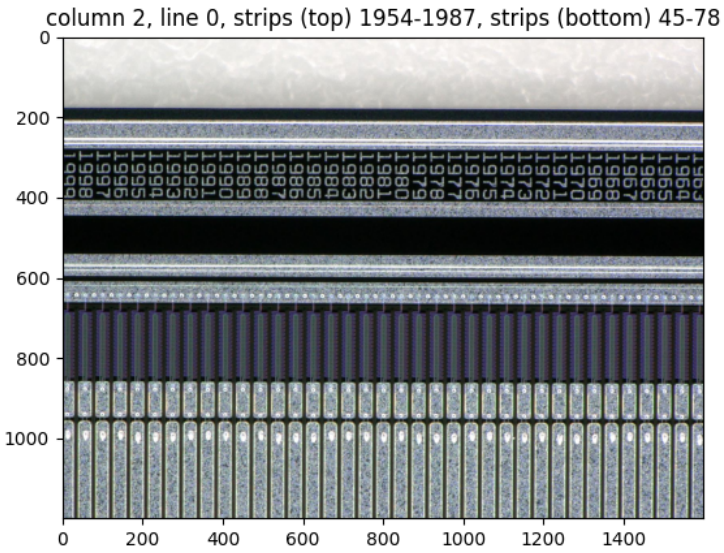



**Abbildung 3.15:** Darstellung eines Bildes in der grafischen Benutzeroberfläche.

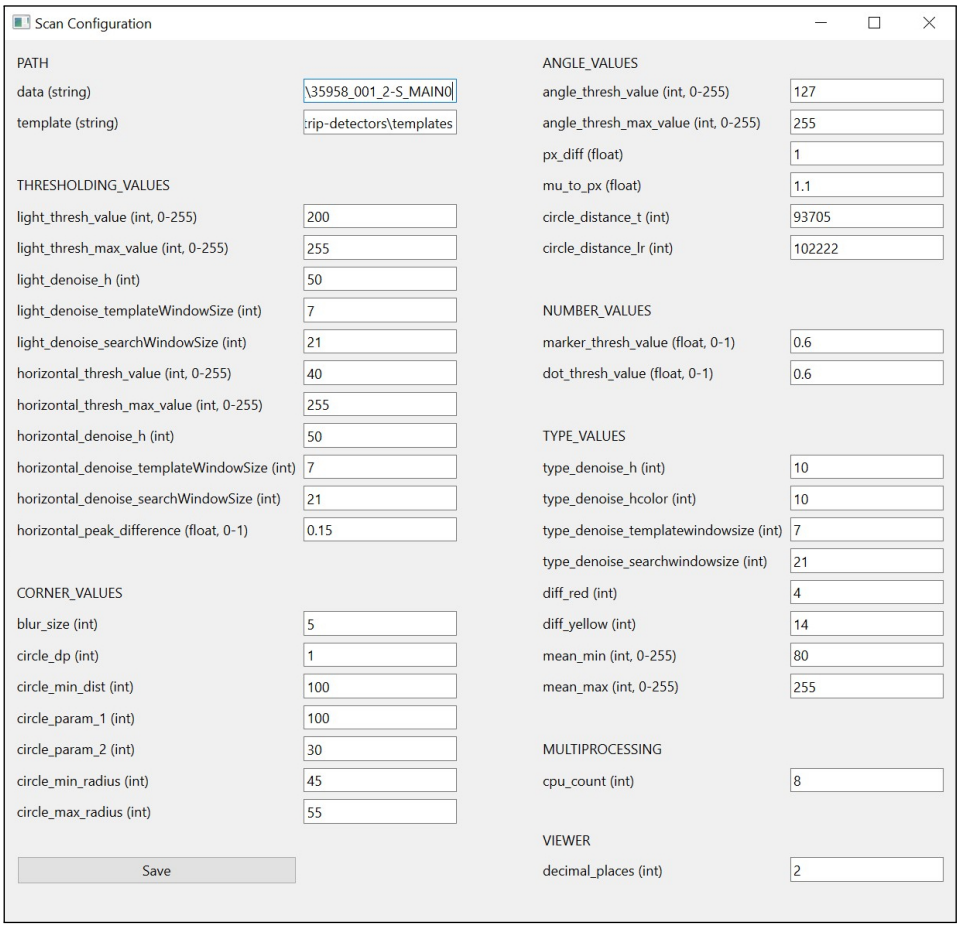



**Abbildung 3.16:** Fenster zur Einstellung der Scan-Parameter.

# 4

# Ermittlung der Scanparameter

Ideale Scanparameter zeichnen sich durch die klare Unterscheidung von Kratzern gegenüber Verunreinigungen sowie Markierungen aus. Im realen Anwendungsfall zeigt die Defekterkennung hier jedoch eine Überlappung beider Bereiche bei verschiedenen Arten der zuvor genannten Defekte. Dies resultiert aus den großen Unterschieden in der Manifestation von Kratzern und Verunreinigungen auf dem Sensor. Die Ermittlung von Scanparametern, welche das Kriterium der klaren Unterscheidung möglichst gut erfüllen, basiert auf der Durchführung der Defekterkennung für verschiedene Grenzwerte der Farbabweichung $A$. Diese ist in Ungleichung 3.1 für die roten Farbanteile und in Ungleichung 3.2 für die gelben Farbanteile definiert.

## 4.1 Ergebnisse der Defekterkennung für verschiedene Einstellungen

Als Parameter zur Ausführung der Defekterkennung dienen hier die ganzzahligen Werte innerhalb des Intervalls $A_\mathrm{r}, A_\mathrm{y} \in [0, 20]$.

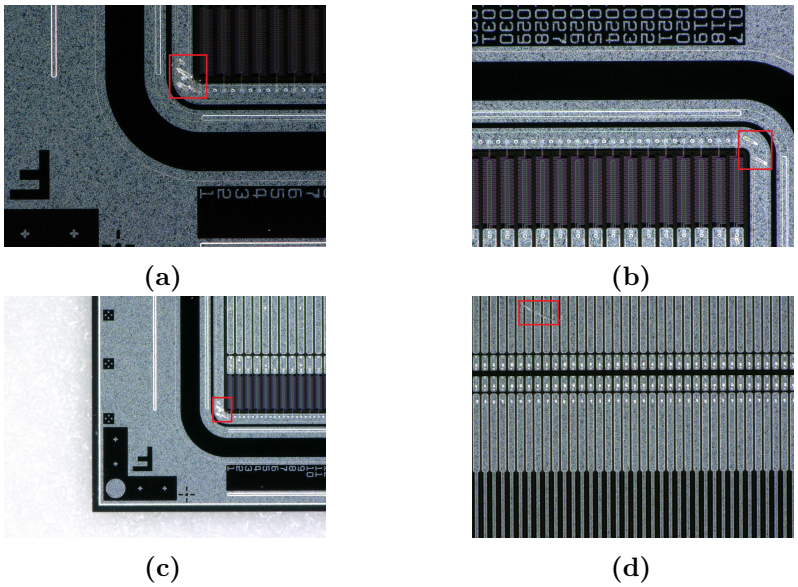
Für die in Abbildung 4.1 dargestellten Bilder, welche Kratzer aufweisen, ergeben sich die in Abbildung 4.2 gezeigten Ergebnisse. Der farbig eingefasste Bereich markiert hierbei die Parameter, für welche ein Fehler detektiert wird. Die maximale Überdeckung schließt den Bereich $A_\mathrm{r} \in [0, 5], A_\mathrm{y} \in [0, 10]$ ein. Bei einer signifikant erhöhten Abweichung des roten Farbanteils werden lediglich die Fehler auf Abbildung 4.1a und 4.1b erkannt. Eine mögliche Begründung dieses Verhaltens liegt in der Belichtung der jeweiligen Bilder sowie der Art des Kratzers. Abbildung 4.1c ist stärker belichtet als die anderen Bilder, wodurch der Kratzer stärker reflektiert. Dies lässt den Defekt auf dem Bild größtenteils weiß erscheinen, was die Bestimmung der Abweichung der Farbwerte beeinträchtigt. Für den Kratzer auf Abbildung 4.1d schlägt die Erkennung für hohe Abweichungen hingegen fehl, da dieser Kratzer im Vergleich zu allen anderen oberflächlicher erscheint. Hierdurch wird der Effekt der Farbabweichung verringert, welcher zur Detektion genutzt wird. Aus der Betrachtung der Ergebnisse dieser Scans zeigt sich, dass die Erkennung von Kratzern eine starke Abhängigkeit von den Scaneinstellungen zeigt.

Auf den in Abbildung 4.3 gezeigten Bildern sind Unterbrechungen des Streifenmusters zu sehen, bei welchen es sich jedoch nicht um einen Kratzer handelt. Diese entstehen durch Staub, Fasern oder andere Verunreinigungen, welche sich durch Verpackung oder Handhabung des Sensors ergeben. Auffällig bei den in Abbildung 4.4 dargestellten Ergebnissen sind die Plots für Abbildung 4.3a und 4.3b. Für diese Bilder werden bis in hohe Bereiche der Parameter Defekte erkannt, welche an den Stellen detektiert werden, an welchen goldfarbene Verunreinigungen im Bild zu sehen sind. Für die häufiger auftretenden Verunreinigungen silbriger Farbe sind die Parameter für Detektion eines Fehlers deutlich geringer. Die mit $A_\mathrm{r} = 1$ und $A_\mathrm{y} = 5$ geringsten Parameter treten bei dem in Abbildung 4.3a dargestellten Bild auf, welches mit der Einstellung für Glanzminimierung und Beleuchtung durch einen Teilring aufgenommen ist. Eine Verringerung der Belichtung bei der Durchführung der Scans dient daher sowohl der besseren Detektion von Fehlern als auch der Vermeidung der Detektion von Verunreinigungen.

Aufgrund der Markierungen, hellen Streifen und Widerstände in den Randbereichen des Sensors sind diese besonders anfällig für eine fehlerhafte Detektion von Kratzern. Dies zeigt

sich an den in Abbildung 4.6 dargestellten Ergebnissen zu den Bildern in Abbildung 4.5. Der Scan für Abbildung 4.5c wurde aufgrund hohen Ressourcenverbrauchs nur im Intervall $A_\mathrm{r}, A_\mathrm{y} \in [8, 20]$ ausgeführt. Im Plot sticht dieses Bild besonders hervor, da hier für wesentlich höhere Parameter die Erkennung eines Defektes erfolgt. Dies liegt daran, dass die Ränder der in Abbildung 4.5c dargestellten Pads annähernd dieselben Farbabweichungen aufweisen wie Kratzer, was sich erst mittels starkem Denoising aufheben lässt. Hierdurch werden jedoch auch eventuelle Kratzer nicht mehr zuverlässig detektiert. Wie die Ergebnisse der Abbildungen 4.5b und 4.5d zeigen, besteht eine weitere Möglichkeit der Verringerung dieses Effekts in der Wahl einer anderen Belichtung. Abbildung 4.5b stellt hier einen Sonderfall dar, bei welchem die Erkennung eines Defektes durch die Belichtungs- und Scaneinstellungen bereits bei niedrigen Parametern vermieden wird. Dies lässt sich bereits an der Abbildung erkennen, da diese im Vergleich zu den anderen Bildern leicht bläulich wirkt.

Abbildung 4.7 stellt Bilder von Streifen dar, welche weder eine Beschädigung noch eine Verunreinigung aufweisen. Die in Abbildung 4.8 gezeigten zugehörigen Ergebnisse weisen nur erkannte Defekte für Abbildung 4.7b auf. Diese werden bis zu hohen Parametern an den Übergängen der Streifen detektiert. Dieser Effekt ist in Abbildung 4.7a aufgrund der durch die Belichtung bläulich wirkenden Darstellung minimiert.



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

**Abbildung 4.1:** Bilder von Sensoren, welche Kratzer aufweisen (rot markierter Bereich).

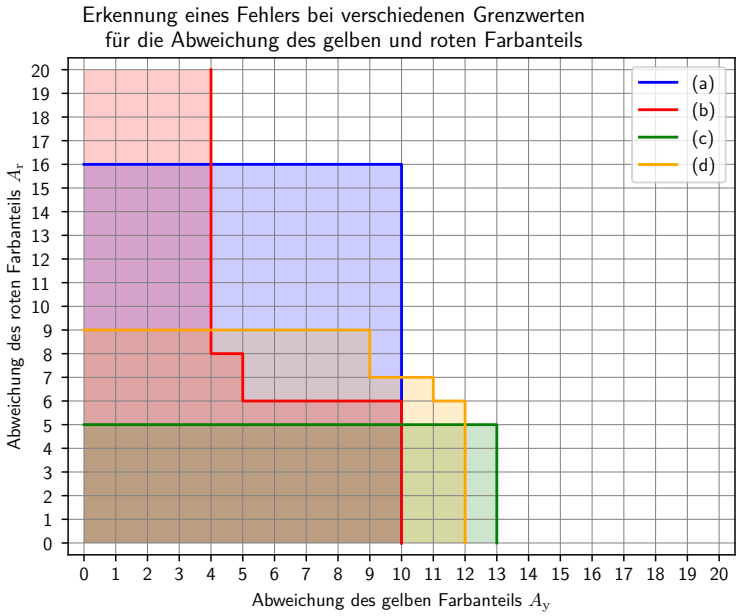



**Abbildung 4.2:** Darstellung der Fehlererkennung bei verschiedenen Werten der Abweichung des gelben und roten Farbanteils für die Bilder aus Abbildung 4.1. Für die farbig eingefassten Werte detekiert die Charakterisierung einen Defekt.

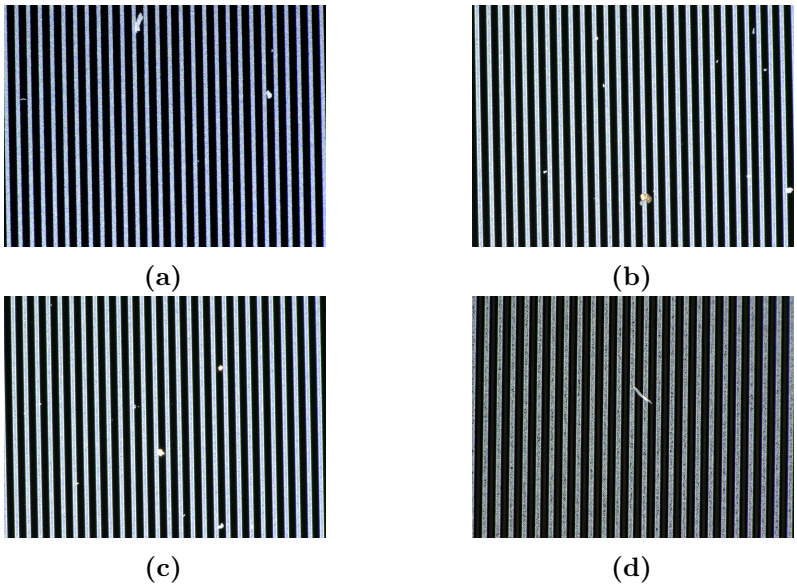



**Abbildung 4.3:** Bilder von Sensoren, welche Verunreinigungen aufweisen.

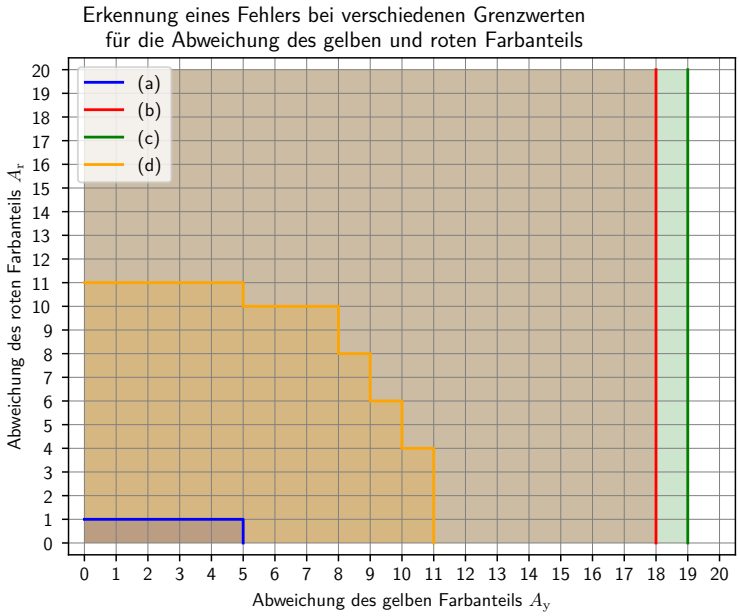



**Abbildung 4.4:** Darstellung der Fehlererkennung bei verschiedenen Werten der Abweichung des gelben und roten Farbanteils für die Bilder aus Abbildung 4.3. Für die farbig eingefassten Werte detekiert die Charakterisierung einen Defekt.

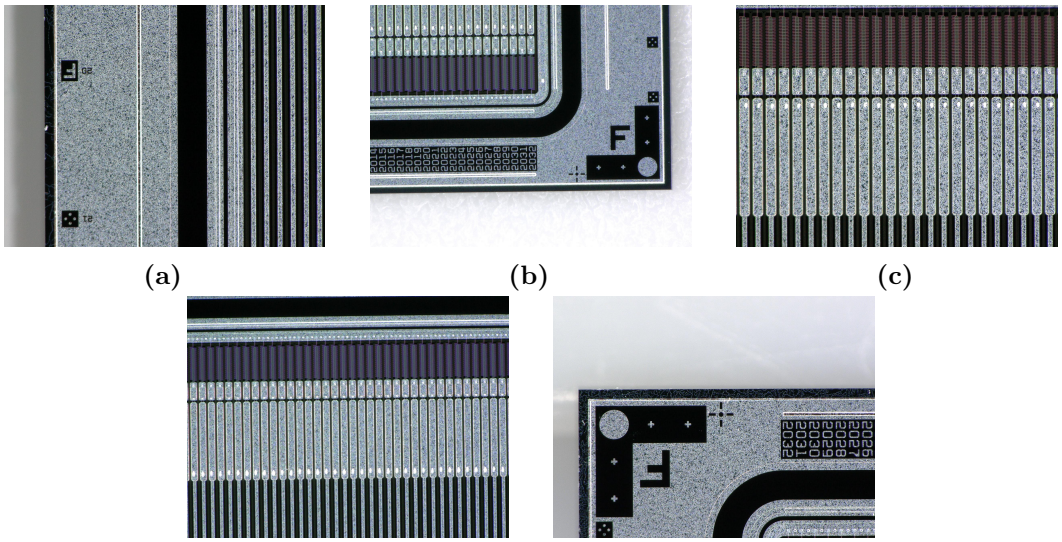



**Abbildung 4.5:** Bilder aus dem Randbereich von Sensoren ohne Beschädigung und Verunreinigungen. Für Abbildung 4.5c ist eine Detektion aufgrund von hohem Ressourcenverbrauch erst ab einem höheren Wertebereich möglich.

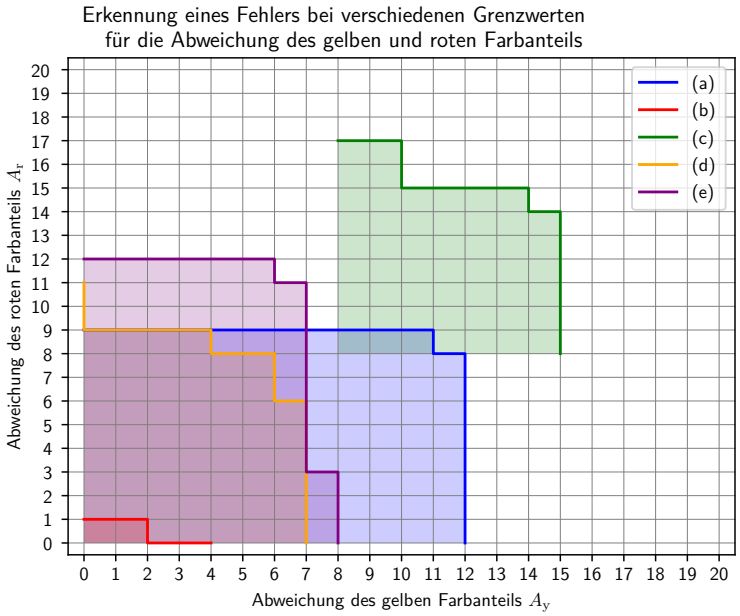
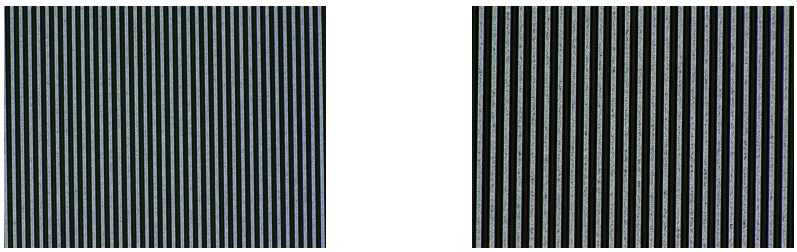



**Abbildung 4.6:** Darstellung der Fehlererkennung bei verschiedenen Werten der Abweichung des gelben und roten Farbanteils für die Bilder aus Abbildung 4.5. Für die farbig eingefassten Werte detekiert die Charakterisierung einen Defekt.



**(a)** **(b)**

**Abbildung 4.7:** Bilder aus dem Streifenbereich von Sensoren ohne Beschädigung und Verunreinigungen.

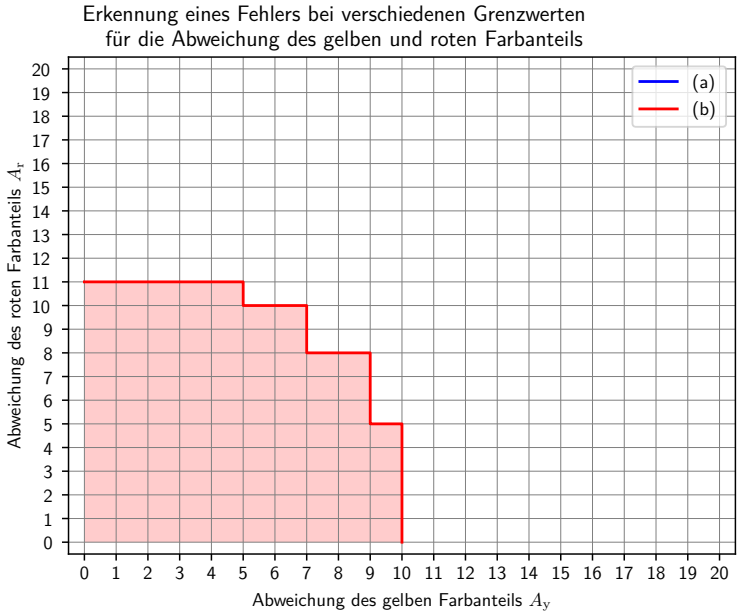



**Abbildung 4.8:** Darstellung der Fehlererkennung bei verschiedenen Werten der Abweichung des gelben und roten Farbanteils für die Bilder aus Abbildung 4.7. Für die farbig eingefassten Werte detekiert die Charakterisierung einen Defekt.

## 4.2 Bestimmung der idealen Parameter

Aus den Betrachtungen im vorigen Abschnitt geht hervor, dass die Erkennung von Defekten in hohem Maß abhängig von den Scaneinstellungen ist. Insbesondere eine ideale Belichtung des Scans trägt zur korrekten Charakterisierung möglicher Defekte bei. Eine Pauschalisierung der Parameter führt daher nicht zu den bestmöglichen Ergebnissen. Daher ist es erforderlich, die Parameter bei jeder Änderung der Belichtung neu zu ermitteln.

Zur Vereinfachung dieses Prozesses enthält die Software ein von den anderen Bestandteilen unabhängiges Skript **parameters.py**. Zur Ermittlung der idealen Parameter sind zunächst zwei Ordner mit Bildern zu erstellen, welche sich darin unterscheiden, ob die enthaltenen Bilder von der Defekterkennung erkannt werden müssen oder nicht. Diese Ordner werden dem Skript übergeben, welches zusätzlich einen Bereich für $A_\mathrm{r}$ und $A_\mathrm{y}$ sowie jeweils einen Wert für die minimale und die maximale Helligkeit der zu erkennenden Defekte benötigt. Das Skript ermittelt für jede Kombination aus $A_\mathrm{r}$ und $A_\mathrm{y}$ den relativen Anteil der Bilder, welche korrekt erkannt werden. Die idealen Parameter werden nach Abschluss des Verfahrens aus der maximalen Anzahl korrekt erkannter Bilder ermittelt und wie in Abbildung 4.9 gezeigt dargestellt. Die Abbildung zeigt, dass die maximale Anzahl korrekt erkannter Bilder vergleichsweise niedrig ist. Dies liegt daran, dass sich Kratzer und Verunreinigungen auf viele verschiedene Arten manifestieren. Daher ist eine Überschneidung der Parameter, für welche Kratzer und Verunreinigungen falsch charakterisiert werden, nicht zu vermeiden.

Die Genauigkeit dieses Verfahrens nimmt mit der Menge an verwendeten Bildern zu, ebenso steigt jedoch die Laufzeit. Letztgenannter Effekt ist jedoch vernachlässigbar, da der Prozess nach Finden einer geeigneten Belichtung nicht wiederholt werden muss, bis die Belichtung geändert wird.

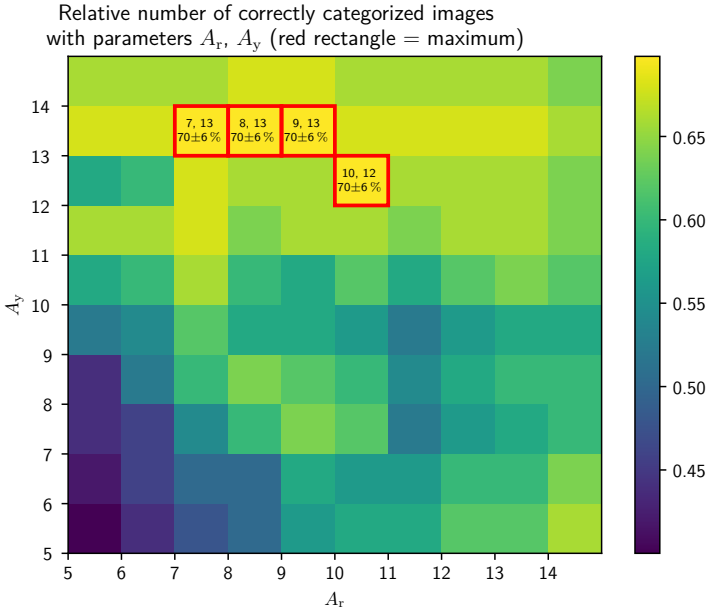



**Abbildung 4.9:** Relative Menge der korrekt erkannten Bilder bei verschiedenen Werten der Abweichung des gelben und roten Farbanteils. Hier wurde ein Datensatz bestehend aus 20 Bildern mit Defekt und 30 Bildern ohne Defekt verwendet. Die Helligkeit der Defekte wurde dabei nicht eingeschränkt. Die maximale Menge an korrekten Bildern ist mit roten Rechtecken markiert.

# 5

# Zusammenfassung und mögliche Verbesserungen der Defekterkennung

Im Rahmen des Phase-2-Upgrades des CMS-Spurdetektors werden in diesem unter anderem neue Siliziumstreifensensoren verbaut. Diese Arbeit hat das Hauptziel der Entwicklung eines Programms zur automatischen Lokalisierung und Charakterisierung von Defekten an solchen Siliziumstreifensensoren. Außerdem bestehen die Ziele der automatischen Ermittlung der Schnittkantenwinkel und der Sensornummer. Dies wird durch die Auswertung von Mikroskopbildern mittels des Programms realisiert. Diese Berechnungen erfordern die einfache Einstellung der hierfür verwendeten Parameter sowie eine anschauliche Visualisierung der Ergebnisse. In den vorigen Kapiteln wurde die Implementierung eines Programms vorgestellt, welches die genannten Anforderungen erfüllt. Hierzu werden Funktionen für Simple Thresholding im Rahmen der Lokalisierung von Defekten und zur klaren Darstellungen von Kontrasten bei der Schnittkantenvermessung verwendet. Weiterhin erfolgt der Einsatz von Non-Local Means Denoising- und Median Blurring-Verfahren zur Glättung der Bilder, um zu vermeiden, dass Punkte in der Größe von wenigen Pixeln als Defekte erkannt werden. Zur Erkennung von Markierungen auf den Sensoren dienen Template Matching und die Anwendung von Hough-Transformationen.

In Kapitel 4 wurde ermittelt, dass das Verfahren zur Charakterisierung der Defekte stark abhängig von der Belichtung bei der Erstellung der Bilder abhängt. Hieraus folgt eine große Menge an ausgegebenen möglichen Defekten, wobei es sich zu einem großen Teil um falsch positive Ergebnisse handelt. Dies ist mit dem momentanen Verfahren notwendig, um alle Kratzer zuverlässig detektieren zu können. Die Schnittkantenvermessung sowie die Erkennung der Sensornummer weisen ebenfalls Grenzfälle auf, welche die korrekte Ausführung der entsprechenden Methoden beschränken. Diese Grenzfälle sowie mögliche Verbesserungen der Funktionen, insbesondere zur Verringerung der falsch erkannten Defekte, werden im Folgenden diskutiert.

Die Schnittkantenvermessung gibt die entsprechenden Winkel für alle Sensorkanten in µrad zurück. Die Grenzen des Verfahrens zeigen sich in solchen Fällen, in welchen sich die kreisförmigen Markierungen und die Schnittkante nicht auf demselben Bild befinden. Dies lässt sich jedoch durch bessere Einstellungen bei der Durchführung des Scans zum Erstellen der Bilder vermeiden. Ein Verfahren, welches ohne solche Einstellungen funktioniert, basiert auf dem automatischen Zusammenfügen der Bilder der Ecken mit den umliegenden Bildern, dem sogenannten Stitching. Hierbei wird garantiert, dass die Markierung und die Schnittkante auf demselben Bild liegen. Ein Nachteil hierbei besteht darin, dass die Bilder beim Stitching eventuell nicht ideal zusammengefügt werden, wodurch leichte Abweichungen bei der Schnittkantenvermessung entstehen.

Die Erkennung der Sensornummer erfordert insbesondere eine Belichtung des Scans, welche nah an der Belichtung der Templates liegt, welche zum Template Matching verwendet werden. Hiermit ist eine möglichst zuverlässige Erkennung der Scratchpads und der entsprechenden Markierungen gewährleistet. Die Rotation des Sensors auf den Bildern sowie die eingestellte Vergrößerung sind ebenso essentiell für die Detektion der Templates. Um eine breite Abdeckung

dieser Variablen zu ermöglichen, lässt sich das Template Matching für mehrere Templates mit verschiedenen Einstellungen durchführen, was jedoch zu einer erheblichen Verlängerung der Laufzeit führen kann.

Bei der Verbesserung der Defekterkennung liegt hauptsächlich die Charakterisierung der Defekte im Vordergrund, da das aktuelle Verfahren einige falsch positive Ergebnisse liefert. Diese treten insbesondere an Kanten auf, an welchen sich Farbwerte ähnlich zu solchen Farbwerten ergeben, die der Detektion von Kratzern dienen. Eine optimale Belichtung minimiert diesen Effekt, wodurch sich die Abgrenzung von Kratzern gegenüber Kanteneffekten erheblich vereinfacht. Aufgrund der begrenzten Möglichkeiten dieser Verbesserung ist es vorteilhaft, ein weiteres Verfahren zur Charakterisierung der Defekte zu implementieren, welches sich nicht ausschließlich auf Farbwerte einzelner Pixel stützt.

Ein möglicher Ansatz für ein solches Verfahren basiert darauf, zusätzlich zur Defekterkennung über Abweichungen der Farbwerte die Struktur der Defekte zu berücksichtigen. Hierbei lässt sich die durch den Abstand zu einem Referenzpunkt gewichtete Verteilung der erkannten Defekte innerhalb eines kleinen Bildbereiches auswerten, um zu bestimmen, ob es sich um eine durchgängige Kante oder viele kleine Stellen mit Defekten handelt. Letzteres ist für tatsächliche Kratzer oft zutreffender als für Stellen, an welchen sich falsch positive Ergebnisse zeigen.

Mittels einer auf Template Matching basierenden Verbesserung lässt sich bereits die Erkennung von falsch positiven Ergebnissen vermeiden. Hierzu dienen Templates der wiederkehrenden Stellen wie Pads, Widerstände oder Markierungen. Nach Detektion der entsprechenden Stellen werden diese durch gegenüber der Charakterisierung neutrale Pixel mit geringem Rot- und Grünanteil im Farbwert ersetzt. Diese Ersetzung findet lediglich dann statt, wenn die Stelle unterhalb einer maximalen Abweichung zum Template liegt, um die Ersetzung tatsächlicher Defekte auszuschließen. Dieses Verfahren führt somit zu einem präventiven Ausschluss von falsch positiven Ergebnissen.

Kombiniert lässt sich die Zuverlässigkeit der Charakterisierung mit den zuvor beschriebenen Verfahren deutlich erhöhen. Weiterhin ist jedoch anzumerken, dass sich die Laufzeit durch zusätzliche Schritte bei der Defekterkennung voraussichtlich stark erhöht. Bei einer Verringerung der Falsch-Positiv-Rate der Defekterkennung bleibt die Rentabilität des Programms dennoch erhalten, da der Aufwand der manuellen Kontrolle hiermit stark reduziert werden kann. Weiterhin ist zu beachten, dass die Defekterkennung insbesondere während der Produktion der Sensoren, wobei mehrere Sensoren pro Woche optisch inspiziert werden müssen, auch außerhalb regulärer Arbeitszeiten durchgeführt werden kann.

# Instructions on using the automatic fault detection for silicon strip sensors

## 1 Required packages

The software runs on Python 3 and requires the installation of the following Python packages, which can be installed using the file **requirements.txt**:

- argparse
- configparser
- dataclasses
- glob
- matplotlib
- multiprocessing
- numpy
- opencv-python
- os
- PyQt6
- re
- sys
- time

## 2 Using the scan software

The scan procedure can be started by executing the file **fault_detection/fault_detection.py**. The optional flags `-c` and `-l` can be used to either skip the fault characterization (`-c`) or both the fault characterization and localization (`-l`).

The program then requires the confirmation of the path set in **config.py** or entering an alternative path. This path should end with the directory containing one or two folders with the images. If there are more than two folders in the directory, selecting one or two folders is required. The same goes for the path to save the results to. Normally, this path should be the same as the path to the images.

After pressing the Enter key to start the fault detection, the program will begin detecting the corners. If all corners are detected, the program asks if the scan ends on the bottom left corner, which is required to save the corners in the right order. If fewer than four corners are detected, a manual input of the corner images is required. In this case, the cutting edge measurement can not be executed. If the images are separated into two folders, the image numbers of the images in the second folder have to be entered as "length of the first folder + image number in the second folder". For more than four detected corners, a manual selection of the correct images is required.

Afterwards, the cutting edge angles are measured and printed as [top, bottom, left, right] in µrad.

The detection of the sensor number is executed on the right edge of the sensor. If the program cannot detect all six markers, manual input of the sensor number is required.

If not skipped, the localization and characterization are then executed in parallelized threads.

The end of the scan procedure is marked by the message "Finished fault detection."

# 3 Using the GUI

The GUI can be launched by executing the file
**fault_detection/fault_detection_viewer.py**. At first, the path to the directory with the results folder has to be chosen. The next two inputs, asking whether the scan resembles a PS-s-Sensor and whether the lower half of the sensor should be swapped, determine the order of the images. The second argument depends on whether the first half of the scan ends at the right or at the left side of the sensor.

Then the main window showing the sensor layout and the other results opens. By clicking on one of the rectangles resembling an image, the corresponding image with strip numbers is displayed in another window. Potential defects are marked as green dots with a green circle surrounding them.

The GUI also allows for editing of the configuration file **config.ini** by clicking on "Edit scan parameters". By pressing "Save", the configuration file is overwritten with the entered parameters.

# 4 Configuration file

Apart from the GUI, the configuration file can be edited by replacing parameters in **config.py** and executing the script. The configuration file contains the following parameters:

| | |
|---|---|
| PATH | path values |
| data | path to the directories containing the folders with the images |
| template | path to the templates for template matching |
| THRESHOLDING_VALUES | localization values |
| light_thresh_value | thresholding value to separate defects by brightness |
| light_thresh_max_value | brightness to set the defects to |
| light_denoise_h | See: [1] |
| light_denoise_templatewindowsize | See: [1] |
| light_denoise_searchwindowsize | See: [1] |
| horizontal_thresh_value | thresholding value to separate light from dark strips |
| horizontal_thresh_max_value | brightness to set the light strips to |
| horizontal_denoise_h | See: [1] |
| horizontal_denoise_templatewindowsize | See: [1] |
| horizontal_denoise_searchwindowsize | See: [1] |
| horizontal_peak_difference | maximum relative deviation of brightness peaks in fault localization by interrupted strips |
| CORNER_VALUES | corner detection values |
| blur_size | size for applying medium blur [2] |
| circle_dp | See: [3] |
| circle_min_dist | See: [3] |
| circle_param_1 | See: [3] |
| circle_param_2 | See: [3] |
| circle_min_radius | See: [3] |
| circle_max_radius | See: [3] |

[1] `https://docs.opencv.org/3.4/d1/d79/group__photo__denoise.html`
[2] `https://docs.opencv.org/3.4/d4/d13/tutorial_py_filtering.html`
[3] `https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html`

| | |
|---|---|
| ANGLE_VALUES | cutting edge measurement values |
| angle_thresh_value | thresholding value to separate the cutting edge from the background |
| angle_thresh_max_value | brightness to set the background to |
| px_diff | maximum difference of pixels next to each other to not be discarded |
| mu_to_px | constant to convert µm to pixels |
| circle_distance_t | distance of opposite corners (top edge) |
| circle_distance_lr | distance of opposite corners (left/right edge) |
| NUMBER_VALUES | sensor number detection values |
| marker_thresh_value | thresholding value to filter the results of the scratchpad template matching |
| dot_thresh_value | thresholding value to filter the results of the marker template matching |
| TYPE_VALUES | characterization values |
| type_denoise_h | See: [1] |
| type_denoise_hcolor | See: [1] |
| type_denoise_templatewindowsize | See: [1] |
| type_denoise_searchwindowsize | See: [1] |
| diff_red | minimum red deviation to detect a scratch |
| diff_yellow | minimum yellow deviation to detect a scratch |
| mean_min | minimum brightness to detect a scratch |
| mean_max | maximum brightness to detect a scratch |
| MULTIPROCESSING | multiprocessing values |
| cpu_count | number of threads used for multiprocessing |
| VIEWER | viewer values |
| decimal_places | number of decimal places for displaying the edge angles |

# 5 Determining the ideal parameters

To determine the ideal scan parameters, it is required to fill two folders with sensor images. One folder contains the images that should be detected, while the other folder contains the images that should not be detected. More images in each folder lead to better parameters. To start the process of determining the parameters, the program **parameters.py** needs to be executed. The paths to the images mentioned previously are then passed to the program. **parameters.py** then requires minimum and maximum values for the red and yellow deviation as an area to search for ideal values in. Additionally, the minimum and maximum brightness of faults have to be specified. The program then evaluates every image for every combination of parameters and determines the fraction of correctly characterized images. After **parameters.py** has finished evaluating the images, a visual output displaying the ideal parameters and the relative number of correctly characterized images for all parameter combinations is shown.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literatur


[Bua11] Buades, A. et al. *Non-Local Means Denoising*. In: Image Processing On Line 1 (Sep. 2011), P208–P212. DOI: 10.5201/ipol.2011.bcm_nlm (siehe S. 11).

[Con15] Contardo, D. et al. *Technical Proposal for the Phase-II Upgrade of the CMS Detector*. 2015. ISBN: 978-9-290-83416-8. URL: http://cds.cern.ch/record/2020886/?ln=de (siehe S. 1).

[Har17] Hartmann, F. M. *Evolution of Silicon Sensor Technology in Particle Physics*. 2. Aufl. Springer International Publishing AG, 2017. ISBN: 978-3-319-64436-3. URL: https://link.springer.com/book/10.1007/978-3-319-64436-3 (siehe S. 1, 6, 8, 9).

[Kae17] Kaehler, A. und Bradski, G. *Learning OpenCV 3 : Computer vision in C++ with the OpenCV library*. O'Reilly Beijing Boston, Mass. Farnham Sebastopol Tokyo, 2017. ISBN: 978-1-4919-3796-9. URL: https://primo.bibliothek.kit.edu/primo-explore/fulldisplay?docid=KITSRC48975838X&context=L&vid=KIT&lang=de_DE&search_scope=KIT&adaptor=Local%20Search%20Engine&tab=kit&query=any,contains,learning%20opencv%203&sortby=date&mode=simple (siehe S. 15).

[Kle14] Klette, R. *Concise Computer Vision : An Introduction into Theory and Algorithms*. Springer London Heidelberg New York Dordrecht, 2014. ISBN: 978-1-4471-6319-0. URL: https://link.springer.com/book/10.1007%2F978-1-4471-6320-6 (siehe S. 12–14).

[Kol16] Kolanoski, H. und Wermes, N. *Teilchendetektoren: Grundlagen und Anwendungen*. Springer Spektrum Berlin Heidelberg, 2016. ISBN: 978-3-662-45350-6. URL: https://link.springer.com/book/10.1007%2F978-3-662-45350-6 (siehe S. 3–7).

[Opea] OpenCV. *Hough Line Transform*. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html (siehe S. 12, 13).

[Opeb] OpenCV. *Miscellaneous Image Transformations*. URL: https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57 (siehe S. 10).

[Opec] OpenCV. *OpenCV*. URL: https://opencv.org/ (siehe S. 1, 15).

[Oped] OpenCV. *Template Matching*. URL: https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html (siehe S. 14).

[Opee] OpenCV. *Template Matching*. URL: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html (siehe S. 14).

[Sha20] Shamshad, A. *Building Computer Vision Applications Using Artificial Neural Networks : With Step-by-Step Examples in OpenCV and TensorFlow with Python*. Apress New York, 2020. ISBN: 978-1-4842-5886-6. URL: https://link.springer.com/book/10.1007%2F978-1-4842-5887-3 (siehe S. 10, 11).