



Fakultät für Informatik

Studiengang Informatik

Optische Positionserkennung von elektronischen Bausteinen
in der Prüfautomatisierung

Master Thesis

von

Benedikt Acker

Datum der Abgabe: tt.mm.jjjj

Erstprüfer: Prof. Dr.

Zweitprüfer: Prof. Dr.

EIGENSTÄNDIGKEITSERKLÄRUNG / DECLARATION OF ORIGINALITY

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Rosenheim, den tt.mm.jjjj

Vor- und Zuname

Kurzfassung

Jeder angefertigter elektronischer Baustein muss in Bezug der Funktionsfähigkeit in einer Testroutine überprüft werden. Um diesen Prüfprozess eines Moduls zu automatisieren, ist das Ziel dieser Arbeit die Objekterkennung und Positionserkennung von Modulen in aufgenommenen Bildern. Als grundlegende Technologie verwendet der Autor hierfür Methoden und Filter der Computer Vision und ein zusätzliches KI-Modell sowie einen Polfilter für die Kamera. Die Pipeline für die Detektion ist in der Programmiersprache Python implementiert und verwendet unter anderem für die optischen Filter die Bibliothek OpenCV. Vorausgehend lag der Fokus der bisherigen Arbeiten auf Verwendung von KI-Methoden, um die Position der Module zu detektieren. Dieses Verfahren erzielte eine maximale Genauigkeit von 79%. Die resultierende Pipeline dieser Arbeit erreicht eine Genauigkeit von bis zu 100%.

Schlagworte: Computer Vision, Python, OpenCV, Optische Erkennung, Konturerkennung, Objekterkennung

Inhaltsverzeichnis

1 Ausgangssituation	1
1.1 Aufbau des Prototypen	1
1.1.1 Kamerasystem	1
1.1.2 Tray	2
1.2 Zielsetzung	2
1.3 Aufbau	3
2 Grundlagen zur optischen Erkennung	5
2.1 Polarisationsfilter	5
2.2 Farbraum	5
2.2.1 HSV-Farbraum	5
2.2.2 CIELAB-Farbraum	6
2.3 Schwellenwertverfahren	7
2.4 Grundlagen zur Kantendetektion	8
2.4.1 Detektion der Intensitätsänderung	8
2.4.2 Gradient eines Bildes	9
2.5 Optische Filter	10
2.5.1 Tiefpass-Filter	10
2.5.2 Hochpass-Filter	11
2.5.3 Morphologie	13
2.6 Hough-Transformation	14
3 Pipeline	16
3.1 Ziel	16
3.2 OpenCV	16
3.3 Otsu-Schwellenwertverfahren	17
3.4 Positionsdetektion im Tray	19
3.4.1 Detektion mit OpenCV	19
3.4.2 Detektion mit KI-Modell	21
3.5 Positionsdetektion des Moduls	21
3.5.1 Moduldetektion	21
3.5.2 Detektion mit Hough-Kreise	25
3.5.3 Chipdetektion	25
3.5.4 Messung der Genauigkeit	25
3.6 Evaluation der Genauigkeit	25
4 Ausführung der Pipeline auf einem Embedded Device	26
5 Fazit	27
6 Ausblick	28

A Erstes Kapitel des Anhangs	29
Literaturverzeichnis	31

Abbildungsverzeichnis

1.1 Anzahl der Industrieroboter aus [Gue21]	2
1.2 Abbildung des Prototypen, aus [Nie22]	3
1.3 Bestandteile des Kamerasytems	4
1.4 Distanzdarstellung zwischen Kamera und den Modulen, aus [Nie22]	4
1.5 Beispiel eines Trays mit Modulen	4
2.1 Funktionsweise des Polfilters, aus [Hal23]	6
2.2 Visualisierung der Anwendung des Polfilters, aus [Hal23]	6
2.3 Visualisierung der Anwendung des Polfilters	7
2.4 Visualisierung der Farträume	7
2.5 Visualisierung der Farträume mit einem Beispielbild	8
2.6 Visualisierung der Änderung der Intensität an einem Beispielbild, aus [Gon08, S.694]	9
2.7 Anwendung des Gradienten, aus [Gon08, S.707]	9
2.8 Anwendung von linearen Filtern mit einer Dimension von 3×3	12
2.9 Visualisierung der morphologischen Operationen mit einem 5×5 Quadrat, aus [Sze21, S.137]	13
3.1 DHCorMX6-Modul	17
3.2 Anwendung des Otsu-Verfahrens, aus [Gon08, S.748]	17
3.3 Visualisierung der Resultate der Pipeline	19
3.4 MobileNet V2	20
3.5 Visualisierung der Resultate der Pipeline	22

Tabellenverzeichnis

1 Ausgangssituation

Laut einem Konferenzbericht der IFR aus dem Jahr 2021 hat sich die geschätzte Anzahl der operativen Industrieroboter zwischen 2010 und 2020 nahezu verdreifacht. Dies ist in Abbildung 1.1 dargestellt. Aus dieser Statistik lässt sich entnehmen, dass Industrieroboter in Unternehmen eine große Relevanz auch in Zukunft haben werden. Der Einsatz dieser Roboter ist unter anderem in der Herstellung zur Prozessautomatisierung. Ein Einsatzzweck eines Industrieroboters behandelt das Thema dieser Arbeit. [Gue21]

Das Unternehmen DH electronics GmbH entwickelt Hard- und Software für die Industrie- und Gebäudeautomation und spezialisierte sich auf die Entwicklung von System on Moduls (SoM). Mitarbeiter von DH electronics prüfen manuell die Module, die extern die Firma BMK (BMK Group GmbH & Co.KG) herstellt.

Hierfür positioniert ein Mitarbeiter die Module in einem Prüfstand und startet die Testroutine. Anschließend entnimmt er das getestete Modul und legt es zurück. Das Ziel ist die Automatisierung dieses Prüfprozesses. Dafür setzt das Unternehmen einen bereits bestehenden Prototypen ein.

Das Ziel ist die Ausarbeitung eines Algorithmus, welcher die Position und Rotation des Moduls auf einem aufgenommenen Bild detektiert. Basierend auf den Ergebnissen des Algorithmus, kann ein Roboterarm die Module aufnehmen. Somit würde dieser Roboter den Aufwand des Mitarbeiters reduzieren.

Hierfür wurde in einer vorherigen Bachelorarbeit [Nie22] bereits ein Verfahren entwickelt, welches sich auf KI-Methoden stützt. In der erwähnten Arbeit verwendete der Autor zur Positionsdetektion Key-Point-Detection auf der Grundlage von neuronalen Netzen. Zum Einsatz kam hierbei unter anderem das neuronale Netz *ResNet50V2*. Da die Genauigkeit dieser Software eine maximale Genauigkeit von 79% erzielte, war deren Einsatz in der Praxis zu unpräzise. Aus diesem Grund liegt der Fokus dieser Arbeit auf der Verwendung von optischen Filtern für die Positionsbestimmung.

Im folgenden Kapitel stellt die Arbeit den Aufbau des Prototypen vor.

1.1 Aufbau des Prototypen

Der bereits existierende Prototyp ist in Abbildung 1.2 erkennbar. Dieser Prototyp besitzt ein Kamerasystem, im Bild mit 1 gekennzeichnet, einen Roboterarm, im Bild mit der Zahl 3 erkennbar und eine Anzeige der Kamera, in der Abbildung mit der Zahl 4 zu erkennen. Das sogenannte „Tray“ ist mit der Zahl 2 in der Abbildung markiert.

Der Roboterarm Panda der Firma Franka Emika GmbH ist ein kollaborativer 7-Achsen-Roboter. Die folgenden Kapiteln stellen die weiteren Bestandteile des Prototypen vor.

1.1.1 Kamerasystem

Das Kamerasystem besteht aus einer 12 MP Sony *IMX477R* Kamera mit einem 16 mm Teleobjektiv *RPIZ* und einem Raspberry-Pi-4. Die Bestandteile sind in Abbildung 1.3a

1 Ausgangssituation

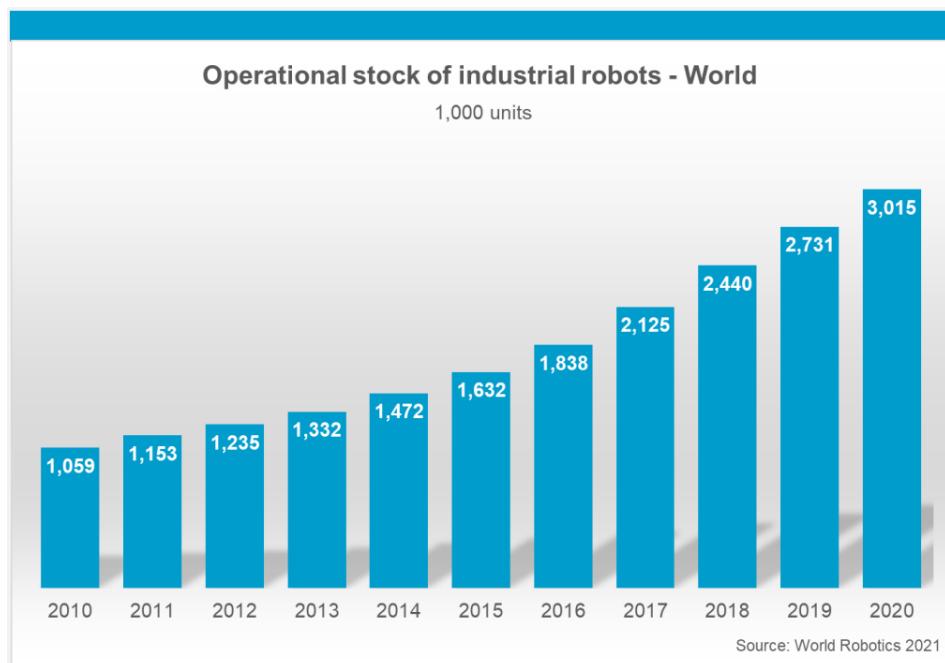


Abbildung 1.1 Anzahl der operativen Industrieroboter von 2010-2020 aus [Gue21]

veranschaulicht. Die Zahl eins kennzeichnet die Kamera, die Zahl zwei den Vakuumgreifer, die Zahl drei den Raspberry-Pi und die Zahl vier den oberen Teil des Roboterarms.

Für die Aufnahme der Bilder von den Modulen im Tray verwendet der Autor zusätzlich einen LED-Ring des Modells WS2812 der Firma YWBL-WH mit einem Durchmesser von 85 mm und einen Polfilter für die Kamera und für den LED-Ring. Der verwendete LED-Ring ist in Abbildung 1.3b abgebildet. Der Polfilter des LED-Rings wurde für die beste Genauigkeit um 80 Grad von der Ursprungsposition gegen den Uhrzeigersinn gedreht. Die Ursprungsposition befindet sich bei 270 Grad mit dem Uhrzeigersinn. Die Funktionsweise eines Polfilter beschreibt die Arbeit im Kapitel 2.1.

In Abbildung 1.4 ist die gewählte Distanz zwischen Kamera und den Modulen abgebildet. Sie beträgt 410 mm von Tray-Unterkante zur Vakuumgreifer-Unterkante. Für die Distanz ist der maximale Roboterarm-Radius und die Auflösung der Kamera entscheidend.

1.1.2 Tray

Das Tray dient als Ablage der Module nach der Herstellung von BMK. Ein Beispiel des verwendeten Trays ist in Abbildung 1.5, zu erkennen. Es besteht aus ESD geschützter Pappe mit den Maßen 550 mm x 350 mm. Der Platz für das Modul besitzt eine Fläche von 45 mm x 45 mm sowie einer Tiefe von 30 mm.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung einer Pipeline in der Programmiersprache Python, die die Rotation, die Eckpunkte und den Mittelpunkt eines Moduls innerhalb eines aufgenommen Bildes berechnet. Ein weiteres Ziel ist die Ausführbarkeit dieser Pipeline auf

1 Ausgangssituation

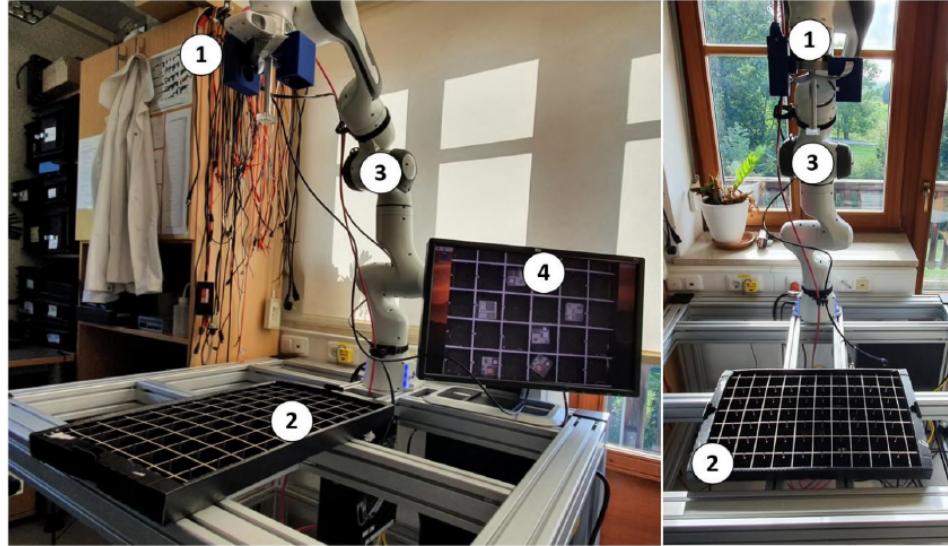


Abbildung 1.2 Abbildung des Prototypen, aus [Nie22]

einem hausinternen Embedded Device von DH electronics. Die Arbeit verwendet für die Positionsdetektion eine Kombination aus einem neuronalen Netz und optische Filter. Für die Anwendung in der Praxis liegt das Ziel der Genauigkeit bei mehr als 85 %. Eine detaillierte Beschreibung der Position beschreibt das Kapitel ??.

1.3 Aufbau

Da das bisherige Kapitel die Ausgangssituation beschrieb, folgt im anschließenden Kapitel eine Vertiefung der Grundlagen. Hier stellt die Arbeit den Polarisationsfilter, die relevanten Farbräume, die Logik eines Schwellenwertverfahrens, die Grundlagen zur Kantendetektion, Variationen von optischen Filter und abschließend die Hough-Transformation.

Anschließend folgt die nähere Beschreibung des Aufbaus der resultierenden Pipeline zur Positionsdetektion. Hierbei unterscheidet die Arbeit auf die Anwendung eines Bildes von einem Tray oder eines Modulbildes. Im Anschluss evaluiert die Arbeit die Pipeline in Bezug auf der Genauigkeit und der Performanz. Dabei dient als Vergleich der Genauigkeit zum bisherigen Ansatz mit **KI-Methoden!**. Abschließend wertet der Autor die Ausführung auf einem Embedded Device aus.

In einem Fazit fasst die Abschlussarbeit die bisherige Projektarbeit zusammen und zeigt Möglichkeiten für Ansatzpunkte in der Zukunft.

1 Ausgangssituation

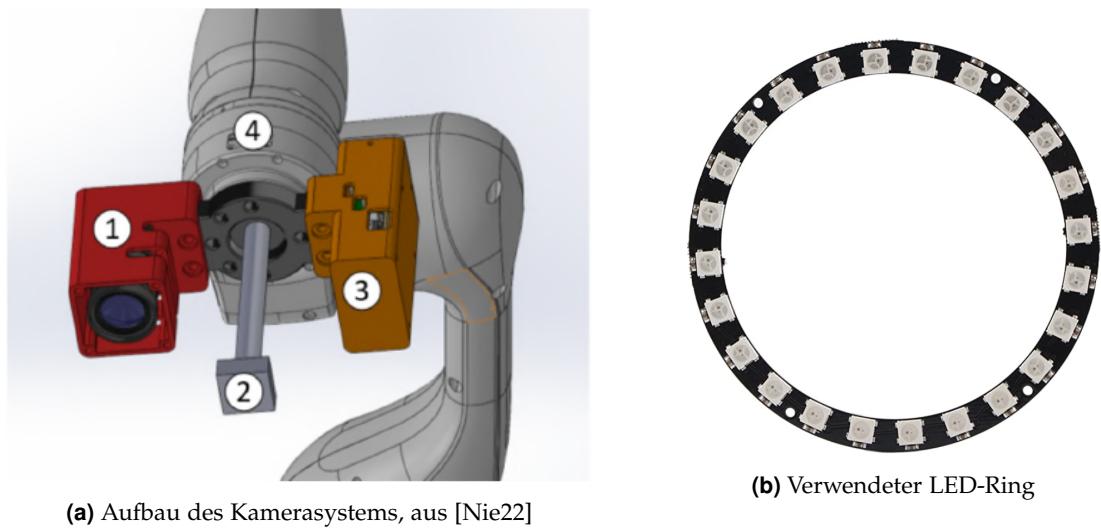


Abbildung 1.3 Bestandteile des Kamerasytems

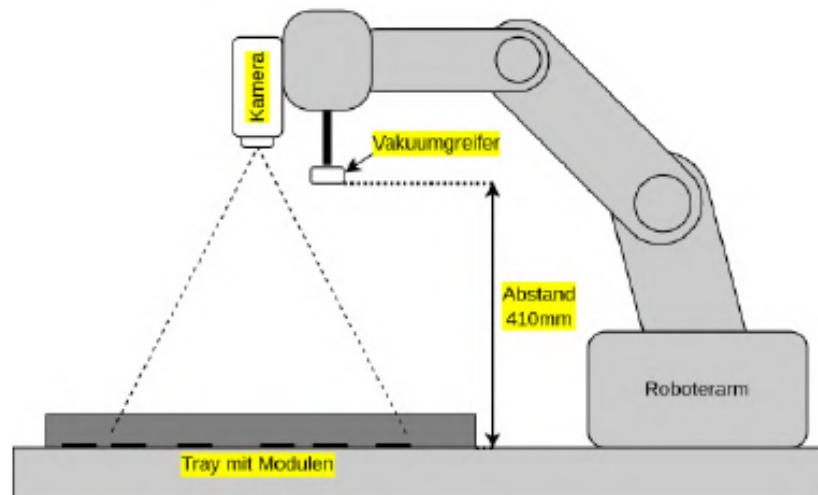


Abbildung 1.4 Distanzdarstellung zwischen Kamera und den Modulen, aus [Nie22]

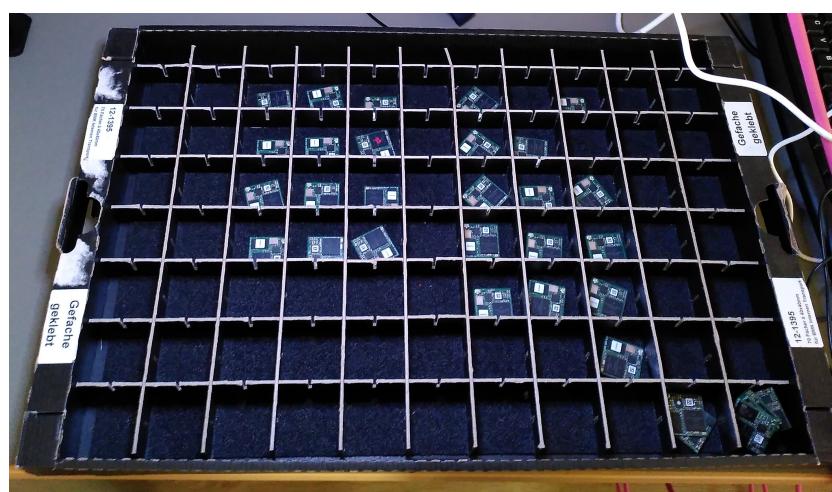


Abbildung 1.5 Beispiel eines Trays mit Modulen

2 Grundlagen zur optischen Erkennung

Für die Entwicklung der Pipeline sind verschiedene Bereiche der optischen Erkennung relevant. Hierzu zählen verschiedene Methoden der Vorbehandlung des Bildes. Die folgenden Kapitel stellen die notwendigen Themen vor.

2.1 Polarisationsfilter

Da in der Vorarbeit in [Nie22] bereits Reflexionen des Lichts im Bild zu Verlusten der Genauigkeit führte, lag der Fokus in der Vorarbeit auf der Suche nach einem externen Filter für die Kamera, welcher die Reflexionen im Bild reduziert. Für diese Aufgabe fiel die Wahl auf den Polarisationsfilter Typ ST-38-20 der Firma ScreenTech, im Folgenden abgekürzt mit Polfilter. Der Polfilter erhöht bei der Bildaufnahme den Dynamikumfang und die Kontraste und intensiviert die Farben, wohingegen Reflexionen im Bild reduziert werden. Die Funktionsweise des Polfilters veranschaulicht hierbei Abbildung 2.1. Das Licht schwingt im dreidimensionalen Raum und der Polfilter filtert je nach Rotation spezifische Lichtwellen.

Das Resultat der Anwendung des Polfilters zeigt die Abbildung 2.2. Nach der Anwendung ist in Abbildung 2.2b die Reduktion der Reflexion auf dem zentralen Felsen im Bild erkennbar. Des Weiteren sind helle Stellen im Wasser gesättigter. [Hal23] [Der23]

Die Auswirkungen auf Traybilder ist in 2.3 veranschaulicht. Bei den Bildern handelt es sich um unterschiedliche Aufnahmen des Trays. In Abbildung 2.3a ist auf den zentralen Modulen klar die Reflexion des LED-Ringes erkennbar. In Abbildung 2.3b ist hingegen nach der Anwendung des Polfilters keine Reflexion des LED-Ringes abgebildet. Die Farben wirken gesättigter und die Module wirken klarer erkennbar.

2.2 Farbraum

Da zur Detektion der Module auf den Bildern die Umwandlung in ein anderen Farbraum notwendig ist, soll dieses Kapitel einen kurzen Überblick über die verschiedenen Farbräume geben.

Die Input-Bilder sind aus dem BGR-Farbraum. Hierbei handelt es sich um eine Umkehrung des RGB-Farbraums. Im RGB-Farbraum besteht eine Farbe aus der Kombination der drei Grundfarben, rot, grün und blau. Jede Farbe besitzt somit für diese drei Farben einen Wert von 0 bis 1. Aus diesem Grund kann man diesen Farbraum auch als dreidimensionales, kartesisches Koordinatensystem visualisieren.

2.2.1 HSV-Farbraum

Im Gegensatz zu dem RGB-Farbmodell besteht das sogenannte HSV-Farbmodell aus drei verschiedenen Variablen und verwendet eine Darstellung in Zylinderkoordinaten. Abbildung

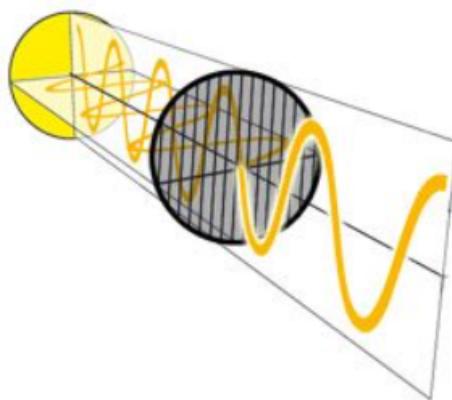


Abbildung 2.1 Funktionsweise des Polfilters, aus [Hal23]



(a) Originalbild

(b) Anwendung des Polfilters

Abbildung 2.2 Visualisierung der Anwendung des Polfilters, aus [Hal23]

2.4a visualisiert diesen Farbraum. In diesem Farbraum repräsentiert die Variable H die Färbung, S die Sättigung und V den Hellwert.

2.2.2 CIELAB-Farbraum

Der CIELAB-Farbraum ist in Abbildung 2.4b visualisiert. In diesem Farbmodell steht der Wert der Variable L für die Helligkeit. Somit repräsentiert dieser Wert den Grauwert im Bild. Senkrecht zu dieser Achse verläuft die Koordinatenachse der Variable a. Hier visualisieren positive Werte rote und negative Werte grüne Farbtöne. Anschließend befindet sich die Koordinatenachse der Variable b ebenfalls senkrecht zu den bisherigen Koordinatenachsen. Positive Werte dieser Variable resultieren in gelben und negative Werte in blauen Farbtönen. Abschließend repräsentieren die Polarkoordinaten C den Radius und h den Farbtonwinkel. Der Vorteil dieses Farbraumes ist die bessere Unterscheidungsmöglichkeit der einzelnen Farben, da in diesem Farbraum eine Distanzmetrik vorhanden ist. [Nis12]

Als Vergleich der unterschiedlichen Farbräume dient die Abbildung 2.5. Dabei repräsentiert Abbildung 2.5a das Originalbild und Abbildung 2.5b und 2.5c das Resultat der Transformation in den HSV- und in den CIELAB-Farbraum. In dieser Abbildung wird ersichtlich, dass im Gegensatz zur Abbildung 2.5b die Abbildung 2.5c die Kanten des Moduls deutlich

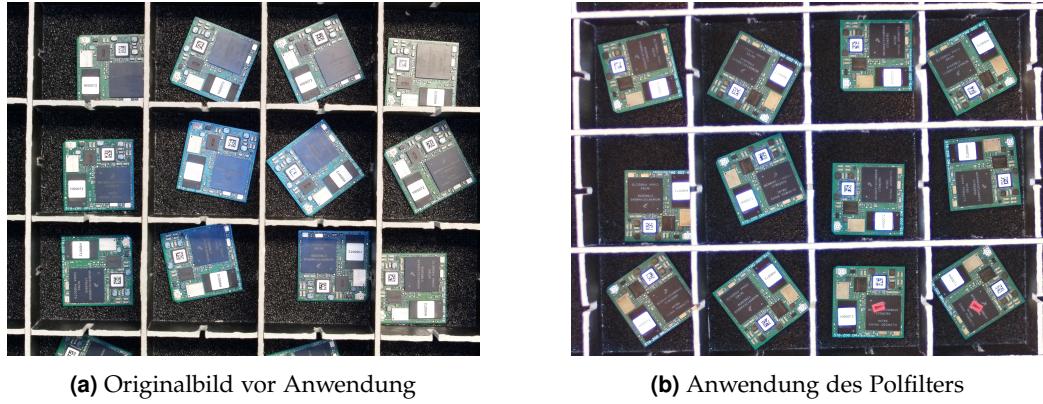


Abbildung 2.3 Visualisierung der Anwendung des Polffilters

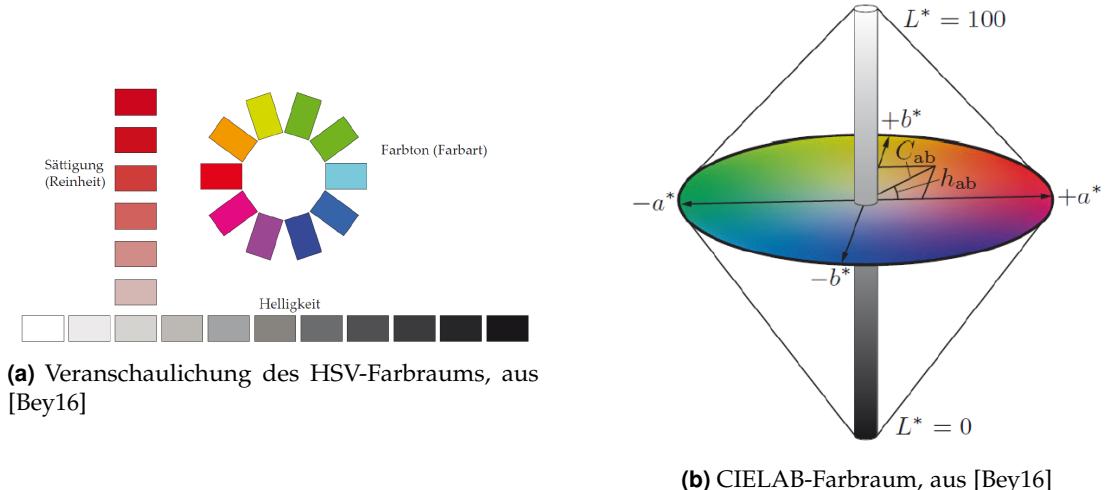


Abbildung 2.4 Visualisierung der Farträume

klarer zeigt und somit für die Detektion des Moduls relevanter ist. Dazu kommt, dass in Abbildung 2.5b einzelne Punkte im Hintergrund des Bildes ein Rauschen hervorrufen, z.B. auf der unteren rechten Seite.

2.3 Schwellenwertverfahren

Das Schwellenwertverfahren ist eine Methode der Umwandlung eines Bildes in Graustufen in ein Binärbild. Dies ist ein notwendiger Schritt der Pipeline vor der Kantenerkennung. Dieses Verfahren lässt sich durch Formel 2.1 allgemein beschreiben. Hier steht f für den Grauwert des jeweiligen Pixels und t für den Schwellenwert.

$$\theta(f, t) = \begin{cases} 1 & \text{für } f \geq t, \\ 0 & \text{sonst,} \end{cases} \quad (2.1)$$

Das Ziel der Anwendung dient einer besseren Differenzierung der verschiedenen Objekte innerhalb eines Bildes in Kontrast zum Hintergrund. Falls ein Verfahren den Wert von t für ein komplettes Bild konstant wählt, zählt das Verfahren als globales Schwellenwertverfahren. Wenn sich das jedoch der Wert von t über das Bild ändert, handelt es sich um ein

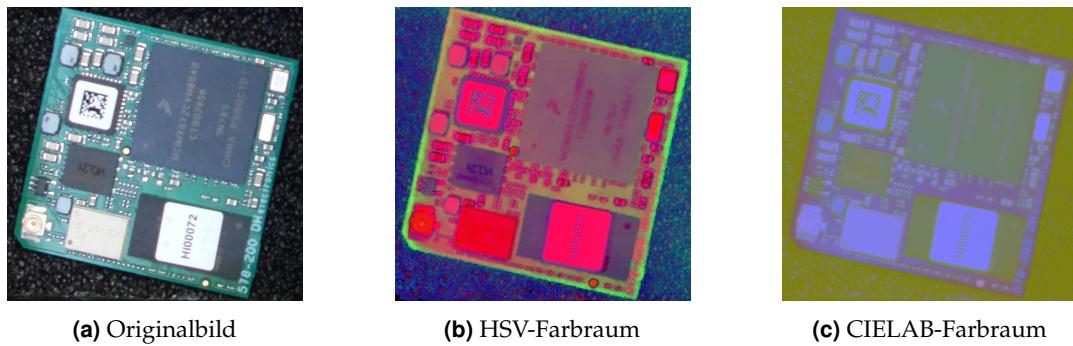


Abbildung 2.5 Visualisierung der Farträume mit einem Beispielbild

variables Schwellenwertverfahren. Zu diesen zählen unter anderem regionale und adaptive Schwellenwertverfahren. In einem regionalen Verfahren ist der Wert von t abhängig von der Umgebung des betrachteten Pixels, zum Beispiel die Durchschnittsintensität. Bei einem adaptiven Verfahren ändert sich der Wert t abhängig von den spezifischen Koordinaten.[Gon08, S.738-739][Sze21]

2.4 Grundlagen zur Kantendetektion

Da Kantendetektion für die Detektion der Module und des Chips innerhalb eines Bildes ein relevanter Bestandteil der Pipeline ist, stellt dieses Kapitel die Grundlagen der Detektion von Kanten vor. Hierfür stellt das folgende Kapitel vor, wie verschiedene Intensitäten und Kanten innerhalb eines Bildes detektierbar sind. Anschließend stellt die Arbeit die Eigenschaften eines Gradienten in einem Bild vor. Als Grundlage dienen im folgenden Kapitel Binärbilder.

2.4.1 Detektion der Intensitätsänderung

Kanten lassen sich zu Beginn unterteilen in Kantenpixel. Diese Kantenpixel sind Pixel in einem Bild, in welchem sich die Intensität einer Bildfunktion abrupt ändert. Um abrupte Änderungen der Intensität zu berechnen, ist die Anwendung der ersten und der zweiten Ableitung eine passende Wahl zur Detektion dieser Änderungen.

Hierfür gelten folgende Punkte für die zweite Ableitung bei der Detektion von Kanten innerhalb des Bildes:

- Bei konstanter Intensität: $= 0$
- Am Anfang und am Ende einer Änderung der Intensität: $\neq 0$
- Innerhalb einer konstanten Änderung der Intensität: $= 0$

Die Abbildung 2.6 veranschaulicht hierbei die vorgestellten Prinzipien. In dieser Abbildung ist das Originalbild oben links. Das Originalbild zeigt verschiedene Objekte, einen einzelnen Punkt sowie eine vertikale Linie in verschiedenen Stufen der Intensität. Die graphische Abbildung der Werte Intensität befindet sich oben rechts. Dabei zeigt der Graph die Werte auf einer zentralen Linie durch den einzelnen Punkt und durch das Zentrum des Originalbildes. Eine genauere Betrachtung des Graphen und Zuordnung der Werte zeigt die Abbildung mittig und im unteren Teil listet eine Tabelle vereinfacht die Werte der Intensität und deren

2 Grundlagen zur optischen Erkennung

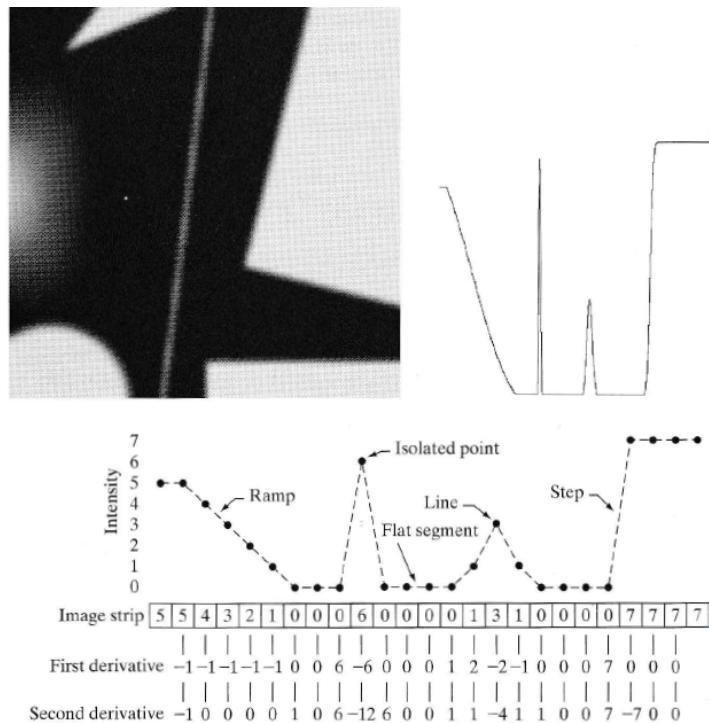


Abbildung 2.6 Visualisierung der Änderung der Intensität an einem Beispielbild, aus [Gon08, S.694]

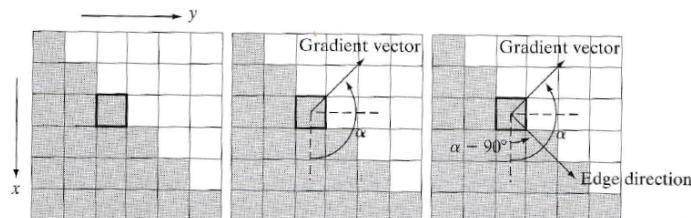


Abbildung 2.7 Anwendung des Gradienten, aus [Gon08, S.707]

erste und zweite Ableitung auf. Dabei beträgt die Spannbreite eines Wertes der Intensität vier Pixel im Bild. Es existieren zur Vereinfachung lediglich acht verschiedene Intensitätsstufen. Auf dem Graphen ist bereits ersichtlich, dass es zwei Kategorien der Änderung der Intensität gibt. Es existieren die konstanten und die abrupten Änderungen.

Zu Beginn ändern sich die Werte der Intensität konstant bis sie den Wert 0 erreichen. Der Wert 0 bleibt für 3 Einträge gleich. Dabei ist erwähnenswert, dass die Werte der ersten Ableitung bei der Reduzierung der Werte einen konstanten negativen Wert besitzen, wohingegen die Werte der zweiten Ableitung am Anfang und Ende der Änderung ungleich 0 sind und während dieser gleich 0 sind.

Anschließend folgt der einzelne helle Punkt und der Wert der Intensität steigt sprunghaft auf 6 und sinkt danach auf 0 herunter. Hier besitzt die erste Ableitung vor dem Anstieg den Wert 6, bei dem maximalen Intensitätswert den Wert -6 und nach dem Anstieg den Wert 0. Die zweite Ableitung hingegen hat vor und nach dem Anstieg den Wert 6 und auf dem höchsten Punkt den Wert -12. Man erkennt, dass die zweite Ableitung auf die Änderung mit einem größeren Wert reagiert. Dies ist auch bei den nachfolgenden An- und Abstiegen der Intensität erkennbar. Des Weiteren lässt sich durch den Vorzeichenwechsels der zweiten

Ableitung ablesen, ob die aktuelle Farbe schwarz oder weiß ist. [Gon08, S.692-695]

2.4.2 Gradient eines Bildes

Um die Stärke und die Richtung einer Kante in einem Koordinatensystem zu detektieren, ist die Berechnung der ersten Ableitung, im Folgenden als Gradient bezeichnet, ein passende Methode. Dabei wird der Gradient folgendermaßen als Vektor definiert:

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (2.2)$$

Dieser Vektor besitzt die Eigenschaft, in die Richtung des maximalen Unterschiedes der Funktion f basierend auf einem zweidimensionalen Koordinatensystem. Die Länge $M(x,y)$ des Gradienten lässt sich mit folgender Formel definieren:

$$M(x,y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \quad (2.3)$$

Dabei beschreibt dieser Wert die Änderungsrate in Richtung des Gradientenvektors. Die anschließende Formel beschreibt die Berechnung der Richtung des Vektors des Gradienten anhand der x-Achse:

$$\alpha(x,y) = \tan^{-1} \left[\frac{g_x}{g_y} \right] \quad (2.4)$$

Wichtig zu dieser Formel ist anzumerken, dass die Richtung einer Kante immer orthogonal zur Richtung des Gradientenvektors ist. Zur Veranschaulichung dieser drei Formeln dienen die Abbildungen in 2.7. Dabei zeigt die linke Abbildung einen hinein gezoomten Bereich eines Bildes mit einer geraden Kante. Jedes Quadrat symbolisiert einen Pixel im Bild. Die Aufgabe besteht in der Berechnung der Länge und der Richtung der Kante an der Stelle der markierten Box. Zur Vereinfachung besitzen die grauen Pixel den Wert 0 und die weißen Pixel den Wert 1. Im Folgenden zeigt dieses Kapitel einen Rechenansatz zur Berechnung der Ableitung in x- und y-Richtung mit einer 3×3 Umgebung um den markierten Punkt.

Diese Berechnung der Ableitung in x-Richtung basiert auf der Subtraktion der Werte der Pixel in der unteren Reihe von der oberen Reihe. Für die y-Richtung der Ableitung gilt die Subtraktion der Werte der rechten Seite von der linken Seite der Umgebung. Daraus resultieren folgende Werte für $\frac{\partial f}{\partial x} = -2$ und für $\frac{\partial f}{\partial y} = 2$. Somit ergibt sich für den Gradienten folgenden Vektor:

$$\nabla f = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \quad (2.5)$$

Für die Länge gilt aus Formel 2.3 $M(x,y) = 2\sqrt{2}$. Wiederum daraus resultiert aus Formel 2.4 für die Richtung des Gradienten $\alpha(x,y) = -45$. Dieser Wert ist äquivalent zu 135 und die mittlere Abbildung 2.7 veranschaulicht dieses Ergebnis. Aus der Regel der Orthogonalität lässt sich in der rechten Abbildung von 2.7 die Richtung der Kante berechnen. [Gon08, S.706-707]

Im Folgenden stellt die Arbeit verschiedene Varianten von optischen Filtern vor.

2.5 Optische Filter

Da verschiedene optische Filter für die Funktionsweise der Pipeline relevant sind, stellt dieses Kapitel die relevanten Filter vor. Zu Beginn des Kapitels muss jedoch zwischen

linearen und nicht-linearen Filtern unterschieden werden. Der Unterschied bezieht sich auf die Spezifikation der Anwendung. Ein linearer Filter lässt sich durch eine Gleichung beschreiben, wohingegen es für nicht-lineare Filter keine Gleichung für deren Anwendung gibt. Daraus resultiert für nicht-lineare Filter, dass sich basierend aus dem Resultat der Anwendung und dem verwendeten Filter nicht das Eingangsbild rekonstruieren lässt.

2.5.1 Tiefpass-Filter

Zu den linearen Filtern zählen unter anderem die Tiefpass-Filter. Diese Filter berechnen für jeden Pixel basierend auf dessen Umgebung einen Wert. Das Resultat dieser Filter ist unter anderem Rauschreduktion in dem Originalbild.

Ein Beispiel hierfür ist die Mittelwertmethode. Hierfür berechnet sich ein Pixel basierend dem Durchschnitt der Werte der Umgebung des Pixels. Im Folgenden ist eine Filtermaske K der Mittelwertmethode mit einer Dimension von 3×3 dargestellt:

$$K = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.6)$$

Als Resultat erhält man je nach Wahl der Dimension des Filters ein im Vergleich zum Originalbild unschärferes Bild. Des Weiteren stellt das Resultat auch Kanten von Objekten unscharf dar.

Im Kontrast ist der Gaußsche Unschärfe-Filter eine gewichtete Mittelwertmethode. Die Grundidee dieses Filters ist, dass die Werte der Pixel, die sich näher dem betrachteten Pixel befinden, eine höheres Gewicht besitzen als andere Pixel in der Umgebung. Als Vorbild dient die Gaußsche Normalverteilung. Der resultierende Wert des jeweiligen Pixel G lässt sich durch folgende Formel beschreiben:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.7)$$

In der Formel 2.7 steht x und y für die horizontale und vertikale Distanz zum Zentrum des Kernels. σ ist die Standardabweichung und e steht für die Eulersche-Zahl. Im Vergleich mit der Mittelwertmethode zeigt das Resultat dieses Filters das Bild weniger unscharf und erhaltet mehr Kanten. [Gon08, S.152-156]

2.5.2 Hochpass-Filter

Das Ziel der Anwendung eines Hochpass-Filters ist im Kontrast zum Tiefpass-Filter, Kanten innerhalb des Bildes hervorzuheben. Dieser Filter detektiert die Unterschiede in der Umgebung eines Pixels und maximiert diese Unterschiede anschließend. Zwei Beispiele hierfür ist der sogenannte Sobel- und Laplace-Filter. Der wesentliche Unterschied dieser beiden Filter ist die Anwendung der Ableitung. Der Sobel-Filter verwendet die erste Ableitung wohingegen der Laplace-Filter die zweite Ableitung zur Kantendetektion anwendet.

Sobel-Filter

Als Grundlage für den Sobel-Filter dienen die Berechnungen für einen Pixel aus Kapitel 2.4.2. Um die Berechnung auf das gesamte Bild auszuweiten, muss für jeden Pixel die partielle

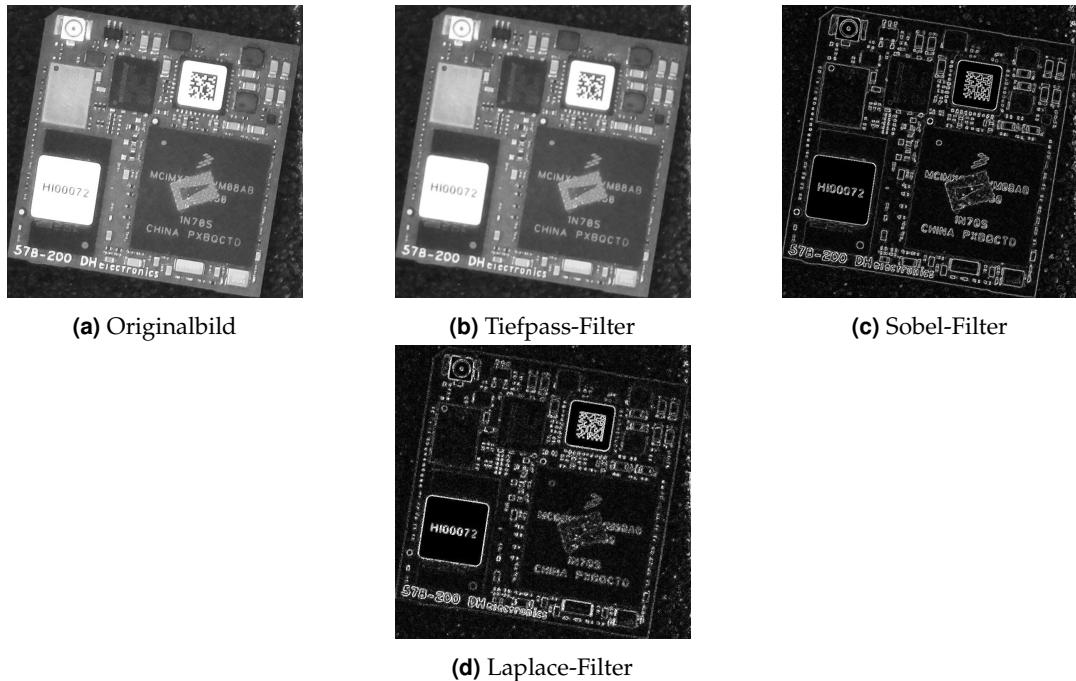


Abbildung 2.8 Anwendung von linearen Filtern mit einer Dimension von 3×3

Ableitung $\frac{\partial f}{\partial x}$ und $\frac{\partial f}{\partial y}$ berechnet werden. Eine 3×3 Umgebung in einem Bild lässt sich durch folgende Matrix veranschaulichen:

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \quad (2.8)$$

Der Sobel-Filter verwendet für die x-Richtung und y-Richtung folgende Masken:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2.9)$$

Somit resultiert für die Ableitung:

$$\begin{aligned} g_x &= \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \\ g_y &= \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \end{aligned} \quad (2.10)$$

Basierend auf der Verwendung der 2 im Zentrum der Maske resultiert eine Bildglättung. [Gon08, S.708-710]

Laplace-Filter

Der Laplace-Filter wendet im Gegensatz zum Sobel-Filter die zweite Ableitung an. Für die zweite Ableitung gilt, dass Kanten auf einem Bild allgemein Nulldurchgänge sind. Dabei

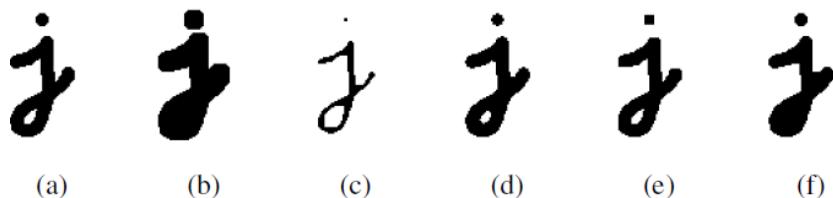


Abbildung 2.9 Visualisierung der morphologischen Operationen mit einem 5×5 Quadrat, aus [Sze21, S.137]

verwendet der Filter folgende Maske:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (2.11)$$

Die zweite Ableitung ist basierend auf der verwendeten Maske durch folgende Formel veranschaulicht:

$$\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y) \quad (2.12)$$

Diese Maske ist in der Lage Änderungen der Intensität auf einem Bild hervorzuheben. Die Unterschiede der beschriebenen Filter sind hierbei in 2.8 erkennbar. Dabei zeigt Abbildung 2.8a das Originalbild. Ein Resultat des Gaußschen Tiefpass-Filters mit einer Dimension von 3×3 ist in Abbildung 2.8b abgebildet. Auffallend ist hier die gute Erhaltung der Kanten des Moduls. Das Resultat der Anwendung eines Sobel-Filters ist in Abbildung 2.8c veranschaulicht und zeigt, dass die Kanten der Aufkleber auf den Chipsätzen auffallend ist. Im Vergleich zur Anwendung des Sobel-Filters in Abbildung 2.8c fällt auf, dass das Resultat des Laplace-Filters deutlich unschärfer wirkt. [Gon08, S.160-162]

2.5.3 Morphologie

Im Kontrast zu den bisher vorgestellten Filtern gelten morphologische Filter als nicht-lineare Filter. Morphologische Operationen sind dabei in der Lage, geometrische Formen in einem Binärbild zu modifizieren. Hierfür dient ein sogenanntes strukturierendes Element als Filter. In diesem Kapitel werden die morphologischen Operationen vorgestellt. Die Resultate der vorgestellten Methoden ist mit Abbildung 2.9 veranschaulicht. (a) ist hier das Originalbild. Lediglich das Resultat oberhalb des Buchstabens (d) wird nicht näher erläutert. Die Resultate erfolgten mit einem Quadrat der Dimension von 5×5 als strukturierendes Element.

Erosion

Die Operation Erosion lässt sich durch folgende Formel spezifizieren:

$$A \ominus B = \{z | (B)_z \subseteq A\} \quad (2.13)$$

Hier steht A für das Binärbild und B für das strukturierende Element. z repräsentiert hier für die resultierenden Punkte, sodass B in A enthalten ist. Resultierend bleiben beim Eingangsbild A die Pixel nur erhalten, wenn alle Werte innerhalb von B > 0 sind.

Das Resultat einer Erosion ist in Abbildung 2.9 oberhalb des Buchstabens (c) abgebildet.

Dabei ist zu erkennen, dass eine Erosion die Größe oder die Dicke des Objektes in einem Binärbild reduziert. Allgemein filtert eine Erosion Objekte, deren Dimension kleiner als die des strukturierendes Elements sind.

Dilation

Die folgende Formel beschreibt die morphologische Operation Dilation:

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\} \quad (2.14)$$

Auch in dieser Formel ist A das Binärbild und B das strukturierende Element. Dabei resultiert die Dilatation von A durch B aus der Menge aller Verschiebungen, so dass \hat{B} und A sich um mindestens ein Element überlappen. \hat{B} ergibt sich aus einer Reflexion aus B.

Das Ergebnis einer Dilation ist in Abbildung 2.9 über dem Buchstaben (b) veranschaulicht. Basierend auf der Gleichung und der Abbildung resultiert aus einer Dilation eine Steigerung der Dicke und der Größe Objekte im Vordergrund des Binärbildes.

Öffnung und Schließung

Dieses Kapitel stellt nun 2 Kombinationen der bisher vorgestellten morphologischen Methoden vor.

Die Öffnung kombiniert die Erosion mit einer danach folgenden Dilation. Ein Resultat ist in Abbildung 2.9 oberhalb des Buchstabens (e) dargestellt. Eine Öffnung glättet die Kontur eines Objektes. Interessant ist bei dem Resultat, dass eine Öffnung den Punkt in dem Buchstaben j zu einem Quadrat formt. Allgemein wirkt das Ergebnis wie eine Maximierung der Dicke aus (c).

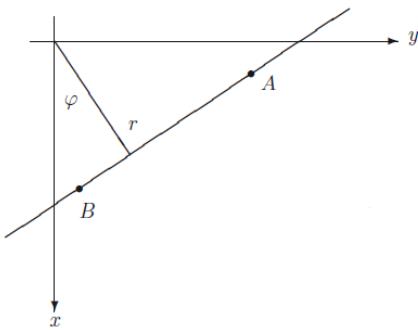
Eine Schließung ist das Gegenteil der Öffnung und beschreibt die Abfolge der Dilation und der anschließenden Erosion. Das Ergebnis einer Schließung ist in Abbildung 2.9 oberhalb des Buchstabens (f) erkennbar. Allgemein kombiniert eine Schließung enge Stellen miteinander. Dies ist auch in der Abbildung durch den unteren Teil des Buchstabens j veranschaulicht. Dabei wirkt das Bild wie eine Reduzierung der Dicke des Ergebnisses aus (b). Allgemein lässt sich bei beiden Kombinationen feststellen, dass jeweils die erste durchgeführte morphologische Operation den größeren Einfluss auf das Resultat besitzt. [Gon08, S. 630-640]

2.6 Hough-Transformation

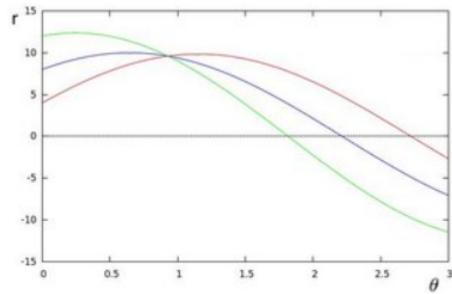
Als letzte Grundlage stellt dieses Kapitel die Hough-Transformation vor. Das Ziel dieser Transformation ist die Detektion geometrischer Figuren in einem Bild. Geometrische Figuren können hier Geraden, Kreise, Ellipsen oder jede andere beliebige Form.

Hierfür lässt sich eine Gerade im Bildbereich basierend auf zwei Variablen definieren. Das kartesische Koordinatensystem verwendet hierfür die Parameter m und b. Im Kontrast besteht das Polar-Koordinatensystem aus den Variablen r und φ . Diese Variablen veranschaulicht die Abbildung 2.10a. Daraus resultiert folgende Geradengleichung für das Polar-Koordinatensystem:

$$y = \left(-\frac{\cos(\varphi)}{\sin(\varphi)} \right) x + \left(\frac{r}{\sin(\varphi)} \right) \Rightarrow r = x\cos(\varphi) + y\sin(\varphi) \quad (2.15)$$



(a) Abbildung einer Geraden, aus [Nis12, S.197]



(b) Visualisierung verschiedener Geraden in Polarkoordinaten, aus [Ope08]

Aus der Gleichung 2.15 folgt, dass für jeden Punkt eine beliebige Anzahl an Geraden existieren, die diesen Punkt schneiden. Abbildung 2.10b zeigt die Visualisierung von drei Geraden. Dabei basiert die rote Gerade auf den Punkten $x_0 = 4$ und $y_0 = 9$, die blaue Gerade auf den Punkten $x_1 = 8$ und $y_1 = 6$ und die grüne Gerade auf den Punkten $x_2 = 12$ und $y_2 = 3$. Wenn sich nun Geraden zweier verschiedener Punkte in einem Punkt schneiden, bedeutet es, dass sie sich beide auf einer Linie in dem Bild befinden. Die drei Geraden schneiden sich in einem einzelnen Punkt (0.925, 9.6).

Daraus folgt, dass eine Linie in dem Bild mithilfe der Anzahl der Überschneidungen der einzelnen Geraden detektierbar ist. Wenn die Anzahl der Punkte auf einer Linie steigt, steigt auch die Anzahl der Überschneidungen der Geraden. Daraus resultiert ein Schwellenwert für eine Mindestanzahl an Überschneidungen für die Detektion einer Linie.

Im Gegensatz zu Linien, benötigt die Detektion von Kreisen drei Parameter: x_{center} , y_{center} , r . Dabei beschreibt r den Radius des Kreises und (x_{center}, y_{center}) die Position des Zentrums. Die Detektion selbst läuft in zwei Schritten ab. Der erste Schritt ist die Kantendetektion und das Detektieren möglicher Zentren der Kreise. Der zweite Schritt ist das Finden des besten Radius für die möglichen Zentren.[Bra08]

3 Pipeline

Dieses Kapitel stellt den Aufbau und die Funktionsweise der in der Programmiersprache Python implementierten Pipeline vor. Hierfür stellt das folgende Kapitel das Ziel der Pipeline vor. Anschließend beschreibt die Arbeit kurz die verwendete Bibliothek OpenCV. Im Anschluss beschreibt ein Kapitel das Otsu-Schwellenwertverfahren. Darauffolgend differenziert die Arbeit Ansätze zur Moduldetektion in einem Tray und in einem bereits zugeschnittenem Bild. Danach veranschaulicht die Arbeit das Verfahren zur Chipdetektion. Zum Abschluss des Kapitels stellt die Arbeit eine Möglichkeit der Messung der Genauigkeit vor und evaluiert die Genauigkeit der Pipeline.

3.1 Ziel

Das Ziel der Pipeline ist mithilfe von optischer Erkennung und einem KI-Modell die Positionsdetektion eines Moduls von DH-electronics in einem Bild. Dabei nimmt das in Kapitel 1.1.1 beschriebene Kamerasystem eine Aufnahme des Trays und die Pipeline verarbeitet dieses Bild. Die Position ist hierbei definiert mit der Angabe des Mittelpunktes und der Rotation des Moduls. Das zu detektierende Modul ist in Abbildung 3.1 erkennbar. Auf der linken Seite ist die Vorderseite und auf der rechten Seite die Hinterseite des Moduls abgebildet. Das Modul selbst hat die Maße 29 mm × 29mm. Der zur Messung der Rotation zu detektierende Modul Chip befindet sich auf der Vorderseite rechts und besitzt die Maße 14 mm × 14 mm. Für die optische Erkennung verwendet die Pipeline Methoden aus der Bibliothek OpenCV. Das folgende Kapitel stellt die Bibliothek vor.

3.2 OpenCV

OpenCV ist eine Open-Source Computer Vision Bibliothek, die im Jahr 1999 veröffentlicht wurde. Sie ist implementiert in C und C++ und ist ausführbar mit Linux, Windows und Mac OS X. Des Weiteren existieren Interfaces für die Programmiersprachen Python, Ruby und Matlab. Das Hauptziel von OpenCV ist dem Benutzer, eine einfach zu benutzende Computer Vision Infrastruktur zu bieten. Dabei beinhaltet die Bibliothek mehr als 500 Methoden, die sich in 5 Kategorien differenzieren lassen.

Die erste Kategorie ist die Computer Vision. Diese Kategorie beinhaltet Methoden zur Behandlung mit Bildern und Algorithmen der Computer Vision. Die nächste Kategorie ist die Machine Learning Bibliothek und beinhaltet statistische Klassifizierer und Werkzeuge für Clustering. Die dritte Komponente ist HighGUI und beinhaltet I/O-Routinen und Funktionen, um Bilder und Videos zu laden und zu speichern. CXCore beinhaltet dabei die Hauptstruktur der Bibliothek und Algorithmen und Zeichenfunktionen. Abschließend besitzt CvAux experimentelle Methoden, um den Vorder- und Hintergrund zu segmentieren. Für die Arbeit verwendet die Pipeline hauptsächlich Methoden der ersten Kategorie. [Bra08]

3 Pipeline



Abbildung 3.1 DHCorMX6-Modul

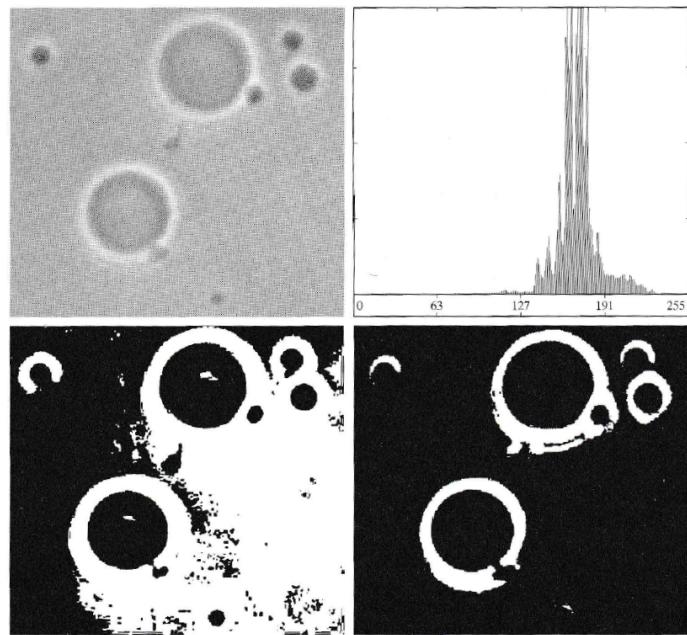


Abbildung 3.2 Anwendung des Otsu-Verfahrens, aus [Gon08, S.748]

3.3 Otsu-Schwellenwertverfahren

Die implementierte Pipeline verwendet das Schwellenwertverfahren von Otsu verwendet. Diese Variation optimiert dabei das globale Schwellenwertverfahren. Das Ziel eines Schwellenwertverfahrens ist die Reduzierung des Durchschnittsfehler bei der Zuweisung der Pixel zu einer oder mehreren Klassen. Hierfür sind zwei Parameter relevant: die Wahrscheinlichkeitsdichtefunktion (WDF) der verschiedenen Intensitätsstufen der einzelnen Klassen und die Wahrscheinlichkeit, dass jede Klasse innerhalb eines Bildes existiert. Dabei ist das Ziel der Otsu-Methode die Maximierung der Varianz zwischen den einzelnen Klassen. Dabei fundieren die Berechnung komplett dem Histogramm des Bildes.

Die verschiedenen Stufen der Intensität in einem digitalem Bild der Größe $M \times N$ Pixel beschreibt $L : \{0, 1, 2, \dots, L - 1\}$. n_i veranschaulicht die Anzahl der Pixel der Intensität i . Nun wird ein Schwellenwertverfahren durchgeführt mit $T(k) = k, 0 < k < L - 1$, das Eingabebild in zwei Klassen C_1 und C_2 differenziert. C_1 besteht aus den Pixel mit den Intensitätswerten aus dem Bereich $[0, k]$ und C_2 aus $[k + 1, L - 1]$. $P_1(k)$ beschreibt die Wahrscheinlichkeit, dass

3 Pipeline

ein Pixel der Klasse C_1 zugewiesen wurde und lässt sich durch folgende beschreiben:

$$P_1(k) = \sum_{i=0}^k p_i \quad (3.1)$$

Dabei steht p_i für die Wahrscheinlichkeit der jeweiligen Intensitätsstufe lässt sich berechnen mit $p_i = \frac{n_i}{MN}$. Aus Formel 3.1 folgt für $P_2(k)$:

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k) \quad (3.2)$$

Der kumulative Durchschnitt der Intensität bis zur Stufe k lässt sich mit folgender Formel berechnen:

$$m(k) = \sum_{i=0}^k ip_i \quad (3.3)$$

Der Durchschnitt der Intensität für das gesamte Bild lässt sich mit folgender Formel veranschaulichen:

$$m_G = \sum_{i=0}^{L-1} ip_i \quad (3.4)$$

Dabei lässt sich die Qualität des Verfahrens mit folgender Formel messen:

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} \quad (3.5)$$

mit $\sigma_B^2(k)$ als Varianz zwischen den Klassen:

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \quad (3.6)$$

und mit σ_G^2 als globaler Varianz

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i \quad (3.7)$$

Für die Wahl des optimalen Schwellenwertes ist es das k , welches den maximalen Wert für $\sigma_B^2(k)$ besitzt. Dabei dient das Resultat aus Formel 3.5 als Messung der Differenzierbarkeit der Klassen innerhalb des Bildes. Somit lässt sich der Algorithmus zusammenfassen:

1. Berechnen des normalisierten Histogramms und $p_i, i = 0, 1, 2, \dots, L - 1$
2. Berechnen der kumulierten Summen für $k = 0, 1, 2, \dots, L - 1$ mit Formel 3.1
3. Berechnen der kumulierten Mittelwerte für $k = 0, 1, 2, \dots, L - 1$ mit Formel 3.3
4. Berechnen des Mittelwertes der globalen Intensität mit Formel 3.4
5. Berechnen der Varianz zwischen den Klassen für $k = 0, 1, 2, \dots, L - 1$ mit Formel 3.6
6. Auswahl des Schwellenwertes mit der maximalen Varianz

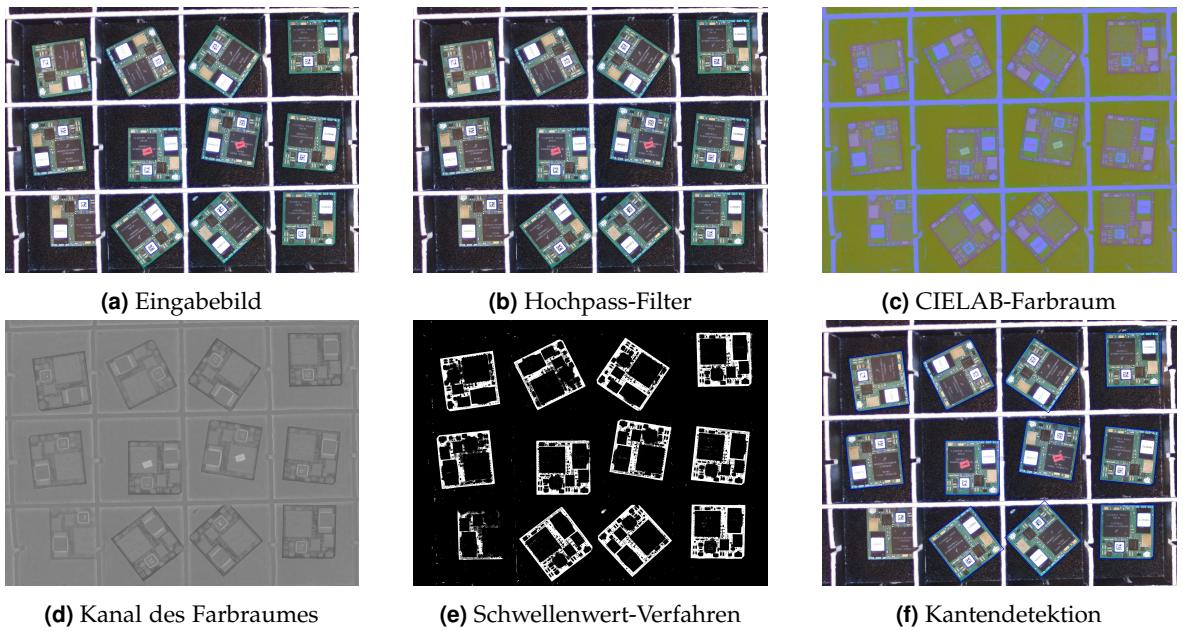


Abbildung 3.3 Visualisierung der Resultate der Pipeline

7. Messen der Qualität der Unterscheidung mit Formel 3.5

Als Veranschaulichung des Verfahrens, dient Abbildung 3.2. In dieser Abbildung ist oben links das Eingangsbild von Polymersomenzellen, oben rechts das Histogramm des Eingangbildes erkennbar. Dabei ist das Ziel des Schwellenwertverfahrens, die dargestellten Moleküle vom Hintergrund zu differenzieren. Unten links in der Abbildung ist das Resultat eines globalen Schwellenwertverfahrens und unten rechts das Ergebnis der Otsu-Methode veranschaulicht. Dabei verwendet das globale Verfahren den Schwellenwert 169 und das Otsu-Verfahren 181. Da der Unterschied der Intensität der Objekte im Vergleich zum Hintergrund klein ist, zeigt die globale Methode Schwächen in der Differenzierung. Der Wert 181 ist dabei weitaus näher an den hellen Regionen im Bild und deswegen zeigt das Verfahren mit diesem Schwellenwert die bessere Unterscheidung. Der Wert für die Messung der Möglichkeit der Unterscheidung η aus der Formel 3.5 beträgt 0.467. [?, S.742-747]

3.4 Positionsdetektion im Tray

Der erste Ansatz der Pipeline war das Ziel der Detektion eines Moduls innerhalb eines Trays basierend auf der Bibliothek OpenCV. Ein Beispiel für ein Input-Bild für die Pipeline zeigt die Abbildung 2.3b.

3.4.1 Detektion mit OpenCV

Die Abbildung 3.3 zeigt die einzelnen Resultate der Anwendungen der Vorbehandlung. Der erste Schritt der Pipeline ist die Anwendung eines Hochpass-Filters auf das Eingabebild, um die Kanten der Module zu verstärken. Hierbei handelt es sich um eine Abwandlung, des in

3 Pipeline

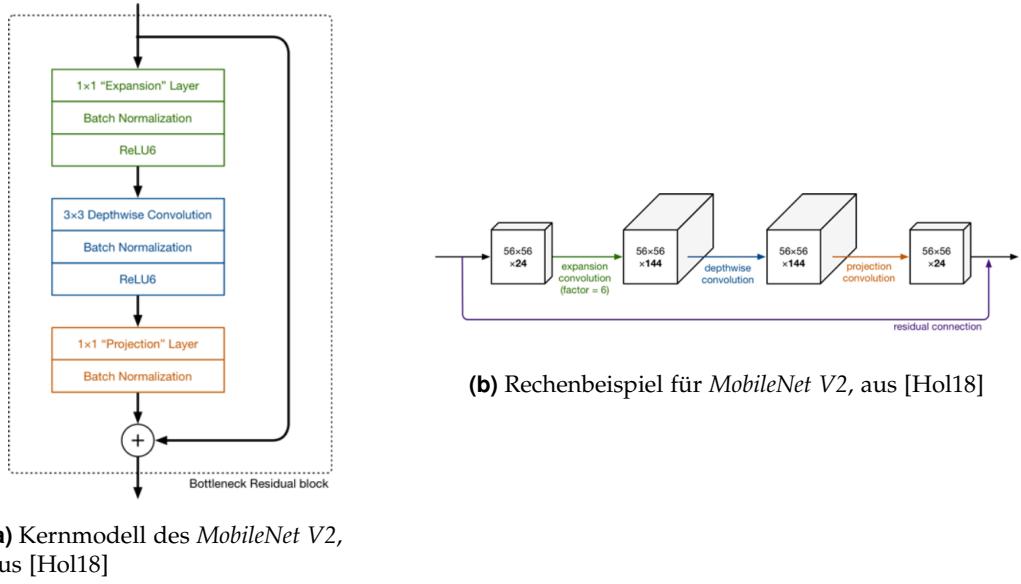


Abbildung 3.4 MobileNet V2

Kapitel 2.5.2 vorgestellten, Laplace-Filters. Die Pipeline verwendet folgende Maske:

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (3.8)$$

Das Resultat des Filters ist in Abbildung 3.3b visualisiert. Im Gegensatz zur vorgestellten Maske in Abbildung 2.8d bleibt die Farbe des Eingabebildes nach der Anwendung der Maske erhalten. Anschließend konvertiert die Pipeline das Bild in den CIELAB-Farbraum und benutzt den Kanal in Graustufen dieses Farbraumes. Die Ergebnisse sind in Abbildung 3.3c und 3.3d visualisiert. Basierend auf diesem Resultat wendet die Pipeline das Otsu-Schwellenwertverfahren an. Das resultierende Bild ist in Abbildung 3.3e abgebildet. Auf diesem Bild ist bereits erkennbar, dass das Schwellenwertverfahren Probleme mit abgeschnittenen Modulen aufweist, erkennbar bei dem Modul unten links im Tray.

Anschließend führt die Pipeline eine Methode der Kantendetektion aus der Bibliothek OpenCV aus. Das verwendete Verfahren basiert auf [Suz85]. Dabei verwendet der Ansatz zwei Algorithmen. Der erste dient zur Detektion der Relationen zwischen den Rändern des binären Bildes. Das Resultat repräsentiert das binäre Bild. Der zweite Algorithmus folgt anschließend nur den Rändern. Das endgültige Ergebnis ist in Abbildung 3.3f visualisiert. In dieser Abbildung umrundet die Pipeline die detektierten Module mit blauer Farbe. Das genaue Verfahren der Pipeline zur Kantendetektion stellt die Arbeit im Kapitel 3.5 vor. Wichtig ist bei diesem Ansatz festzuhalten, dass dieser keine Klassifizierung unterschiedlicher Module durchführen kann und dass es Probleme bei abgeschnittenen Modulen gibt. Aus diesem Grund verwendet die Pipeline ein bisher existierendes KI-Modell, da dieses Modell eine bessere Genauigkeit aufweist und eine Klassifizierung der Module durchführen kann. Dieses KI-Modell stellt das folgende Kapitel vor.

3.4.2 Detektion mit KI-Modell

Als Modell dient das *SSD MobileNet V2 FPNLite*. SSD steht abgekürzt für Single Shot Detector. FPN steht hierbei für Feature Pyramid Network und verwendet hochauflösende Merkmale aus flachen Schichten und niedrig aufgelöste Merkmale aus tiefen Schichten. Die lite-Version stellt eine leichtgewichtige Version dar. Das resultierende Modell verwendet das KI-Modell *MobileNet V2* mit einem anschließenden SSD und einem FPN. Das *SSD MobileNet V2 FPNLite* selbst besteht aus 267 Layer und 15 Millionen trainierbaren Parameter. Der Kernbestandteil ist in Abbildung 3.4a visualisiert und ein Rechenbeispiel mit den Dimensionen eines Eingabebildes befindet sich in Abbildung 3.4b. Dieser Bestandteil des KI-Modells besitzt einen Layer für die Expansion, eine Faltungsschicht und ein Layer der Projektion. Das Modell selbst besteht aus 17 dieser Kernmodelle und anschließender 1×1 Faltungssicht, einer Poolingschicht und ein Layer der Klassifikation.

Dabei dient der erste Layer zur Vergrößerung der Anzahl der Kanäle in den Bildern. Als Default-Einstellung gilt der Faktor 6. Somit ergibt sich aus den 24 Kanälen im Eingangsbild in Abbildung 3.4b das Ergebnis 144 Kanäle. Danach filtert die Faltungsschicht die Eingabedaten. Abschließend komprimiert die nächste Schicht basierend der Projektion die Anzahl der Kanäle. Im Rechenbeispiel reduziert sich die Anzahl von 144 auf 24. [Hol18]

Das bereits trainierte resultierende KI-Modell stammt aus der Arbeit von [?]. Die Anzahl der Trainings- und des Validierungsdatensatzes beträgt 2000 und 500 Bilder. Hinzu kommt, dass das Modell eine Klassifizierung zwischen zwei verschiedenen Kategorien von Modulen durchführen kann. Die Genauigkeit des Modells beträgt 86%. Dieses Modell verwendet die Arbeit, um die Module aus Tray-Bildern, wie in Abbildung 2.2b, zu detektieren. Die resultierenden Bilder der ausgeschnittenen Bilder, wie in Abbildung 2.5a, dient als Eingabe für die Pipeline der Moduldetektion im folgenden Kapitel.

3.5 Positionsdetektion des Moduls

Dieses Kapitel stellt die Methodik der Positionsbestimmung eines einzelnen Moduls vor. Dabei dient als Eingabe ein Bild eines ausgeschnittenes Modul. Die Pipeline ermittelt anschließend den Grad der Rotation und den Mittelpunkt des Moduls. Hierfür ist in einem Schritt die Detektion des Moduls notwendig. Da die Seiten des Moduls quadratisch sind, ist eine Detektion des Chips auf dem Modul notwendig. Anschließend erfolgt eine Messung der Genauigkeit der Berechnungen und abschließend evaluiert das Kapitel die Genauigkeit der Pipeline.

3.5.1 Moduldetektion

Der erste Schritt der Pipeline ist die Detektion des äußeren Moduls. Die einzelnen Schritte der Pipeline visualisiert die Abbildung 3.5. Der Ablauf zur Moduldetektion ist identisch zum Ablauf der Pipeline zur Detektion im Tray in Kapitel 3.4.1. Dabei zeigt Abbildung 3.5a das Input-Bild der Pipeline. Der erste Schritt ist die Anwendung des Schärfe-Filters, Resultat in Abbildung 3.5b. Darauf folgt die Konvertierung in den CIELAB-Farbraum, Abbildung 3.5c. Anschließend reduziert die Pipeline den Farbraum auf ein Kanal, Ergebnis in Abbildung 3.5d. Danach wendet die Pipeline das Otsu-Schwellenwertverfahren an. Das Resultat ist in Abbildung 3.5e. Als letzten Schritt folgt die Kantendetektion auf einem Binärbild. Das Resultat dieser Kantendetektion ist in blauer Farbe in Abbildung 3.5f. Dabei ist bei der Detektion des Moduls erkennbar, dass auch wenn die obere Ecke des Moduls im Bild nicht

3 Pipeline

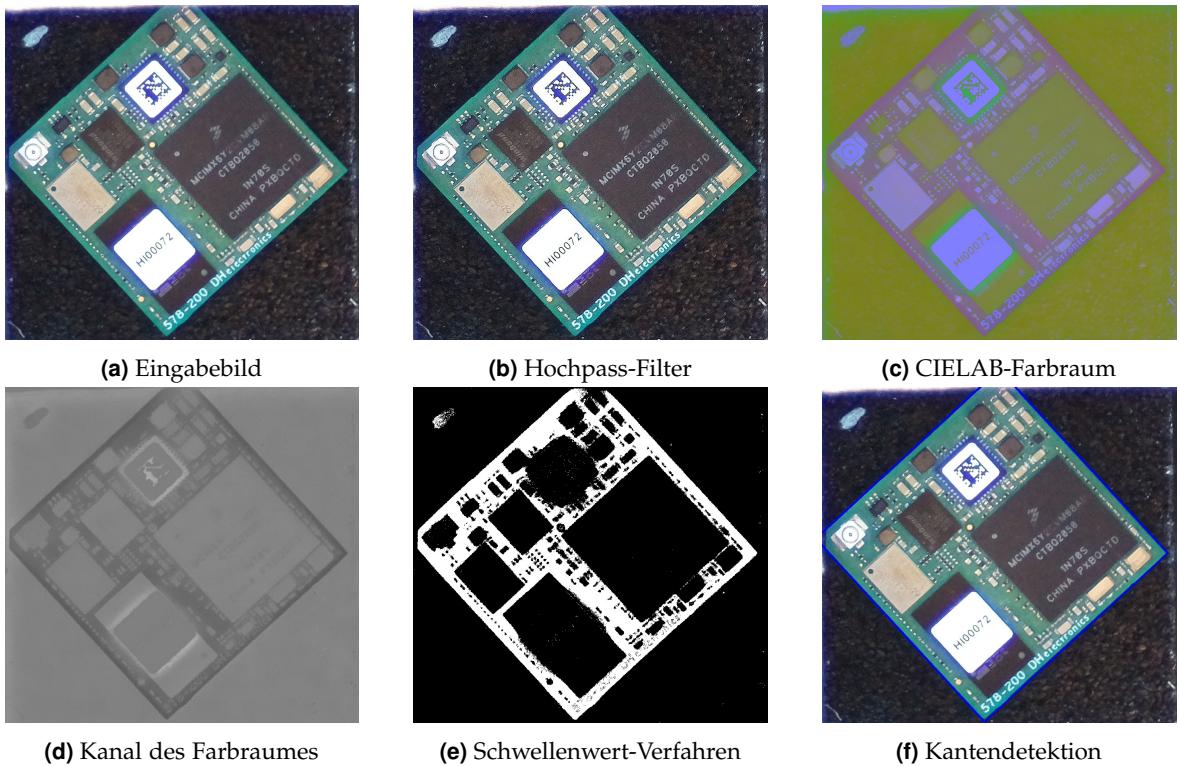


Abbildung 3.5 Visualisierung der Resultate der Pipeline

mehr abgebildet ist, eine Detektion des gesamten Moduls möglich ist. Dieses Kapitel stellt den Quellcode zur Moduldetektion vor. Der Quellcode befindet sich in Listing 3.1 und die Pipeline führt diesen zwischen Abbildung 3.5e und 3.5f aus.

Listing 3.1 Ausschnitt aus der Pipeline zur Moduldetektion

```

1 def findRectangle(image, boxSOM, edgeLengthOfSOM = 0, wholeSomFlag = False,
2                   coloredDetection = False):
3
4     rectangleSOMs = []
5
6     lborder = 0
7     diffValue = 0
8
9     contours, hierarchy = cv2.findContours(image, cv2.RETR_TREE, cv2.
10                                              CHAIN_APPROX_NONE)
11
12     if wholeSomFlag:
13         diffValue = 80
14
15         lborder = 570
16         uborder = 800
17
18     elif coloredDetection:
19         diffValue = 80
20
21         lborder = edgeLengthOfSOM / 2 - 5
22         uborder = edgeLengthOfSOM / 2 + 70

```

3 Pipeline

```
22     else:
23         diffValue = 60
24
25         lborder = edgeLengthOfSOM / 2 - 5
26         uborder = edgeLengthOfSOM / 2 + 40
27
28
29     for c in contours:
30         rect = cv2.minAreaRect(c)
31
32         # w and h < 5?
33         if any(x < 10 for x in rect[1]):
34             continue
35
36         center = rect[0]
37         angle = rect[2]
38
39         box = cv2.boxPoints(rect)
40         box = np.int0(np.around(box))
41
42         a1 = box[1][0] - box[0][0]
43         b1 = box[1][1] - box[0][1]
44         a2 = box[2][0] - box[1][0]
45         b2 = box[2][1] - box[1][1]
46         a3 = box[3][0] - box[2][0]
47         b3 = box[3][1] - box[2][1]
48         a4 = box[0][0] - box[3][0]
49         b4 = box[0][1] - box[3][1]
50
51     # Berechnung der Kantenlängen
52     c1 = round(math.sqrt(a1 ** 2 + b1 ** 2))
53     c2 = round(math.sqrt(a2 ** 2 + b2 ** 2))
54     c3 = round(math.sqrt(a3 ** 2 + b3 ** 2))
55     c4 = round(math.sqrt(a4 ** 2 + b4 ** 2))
56     rect_lengths = np.array([c1, c2, c3, c4])
57
58     mean_val = rect_lengths.sum() / rect_lengths.shape[0]
59     mean_val = round(mean_val)
60
61     diff = abs(c1-c2)
62
63     if mean_val < 250:
64         continue
65
66
67     if diff < diffValue and mean_val > lborder and mean_val < uborder:
68
69         #check if chip rectangle is in SOM
70         if not wholeSomFlag:
71
72             rectangleSOM = cv2.minAreaRect(boxSOM)
73             centerNew = np.array((rectangleSOM[0][0], rectangleSOM[0][1]))
74             lengths = np.array((rectangleSOM[1][0]+15, rectangleSOM[1][1]+15))
75
76             rectangleNew = (centerNew, lengths, rectangleSOM[2])
77             boxNew = cv2.boxPoints(rectangleNew)
78             boxNew = np.int0(np.round(boxNew))
79
80             polygon = Polygon(boxNew)
```

3 Pipeline

```
81         ret_val = True
82         for point in box:
83             p = Point(point)
84             if polygon.contains(p) == False:
85                 ret_val = False
86
87             if not ret_val:
88                 continue
89
90
91
92
93             print(mean_val)
94
95             tuple_val = (box, mean_val, angle, center)
96
97             return tuple_val
```

Die aufgerufene Methode *findRectangle* besitzt folgende Parameter: Das Eingabebild, die Koordinaten und die Kantenlänge des detektierten Moduls, die erst in Kapitel 3.5.3 relevant sind, sowie Flags, die aussagen, welche Detektion notwendig ist. Das Flag *coloredDetection* wird auch in Kapitel 3.5.3 relevant. Diese Methode ruft die Pipeline zur Moduldetektion und zur Chipdetektion, die die Arbeit in Kapitel 3.5.3 näher betrachtet

Zu Beginn der Detektion ruft die Methode in Zeile 8 die Funktion *findContours* aus der Bibliothek OpenCV auf. Zur Kantendetektion verwendet die Bibliothek das Verfahren aus [Suz85], bereits beschrieben in Kapitel 3.4.1. Dabei stehen die Parameter für das Eingabebild, den Abrufmodus und Methode der Annäherung der Kontur. Die Wahl des Abrufmodus legt fest, welche Konturen innerhalb des Bildes zurückgegeben werden. Dabei legt der Modus *RETR_TREE* fest, dass alle Konturen zurückgegeben werden. Der Parameter der Annäherung legt die Anzahl der Punkte fest, die die Kontur beschreiben. Mit dem Wert *CHAIN_APPROX_NONE* speichert die Methode alle Punkte der Kontur ab. Je nach Detektionsmodus setzt die Methode in den Zeilen von 10-26 die Ober- und Untergrenze zur Detektion, sowie einen Wert für die Differenz der jeweiligen Seite des Rechtecks, wenn eine Seite länger ist als eine andere.

Anschließend iteriert die Funktion ab Zeile 29 über jede detektierte Kontur im Bild. Die Methode in Zeile 30 bildet aus der Kontur ein Rechteck beziehungsweise ein Quadrat aus der Kontur. Neben der Höhe und der Länge erhält man nach dem Aufruf, das die Koordinaten des Zentrums sowie den Grad der Rotation. Im Anschluss ist der Rückgabewert der Funktion in Zeile 39 die Koordinaten der Eckpunkte. In den Zeilen 42-56 erfolgt die Kantenermittlung der vier Seiten. Anschließend berechnet die Funktion den Mittelwert aller Seiten und berechnet die Abweichung der Länge der zwei Seiten. Danach folgt in Zeile 67 ein Vergleich der Abweichung und der durchschnittlichen Kantenlänge mit konstanten Werten. Nach diesem Vergleich folgt in den Zeilen 72-89 ein Anwendungsfall für die im Kapitel 3.5.3 vorgestellte Chipdetektion. Hier prüft die Funktion, ob sich alle vier Eckpunkte des detektierten Chips innerhalb des bereits detektierten Moduls befinden. Dies erfolgt mit der Bibliothek *shapely*.

3 Pipeline

3.5.2 Detektion mit Hough-Kreise

3.5.3 Chipdetektion

OpenCV

KI-Modell

3.5.4 Messung der Genauigkeit

3.6 Evaluation der Genauigkeit

4 Ausführung der Pipeline auf einem Embedded Device

5 Fazit

6 Ausblick

A Erstes Kapitel des Anhangs

Wenn Sie keinen Anhang benötigen, dann bitte einfach rausnehmen.

Abkürzungsverzeichnis

ARM Advanced RISC Machines

FPN Feature Pyramid Network

IFR International Federation of Robotics

KI Künstliche Intelligenz

LED Leuchtdiode

MP Megapixel

mm Millimeter

NPU Neural Processing Unit

SoM System on Moduls

SSD Single Shot Detector

WDF Wahrscheinlichkeitsdichtefunktion

Literaturverzeichnis

- [Bey16] J. Beyerer, F. P. León und C. Frese. *Automatische Sichtprüfung*. Springer Vieweg Berlin, Heidelberg, 2016.
- [Bra08] G. Bradski und A. Kaehler. *Learning OpenCV*. O'Reilly Media, Inc., 2008.
- [Der23] D. Derbis. Polfilter Effekt, Anwendung und Vergleich. <https://leidenschaft-landschaftsfotografie.de/polfilter/>, 2023. Aufgerufen am 25.02.2023.
- [Gon08] R. Gonzales und R. Woods. *Digital Image Processing*. 2008.
- [Gue21] M. Guerry, C. Müller, W. Kraus und S. Bieller. World Robotics 2021, IFR. https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf, 2021. Aufgerufen am 25.02.2023.
- [Hal23] M. Haltenhof. Polfilter Test, Anwendung und Wissenswertes. <https://www.matthiashaltenhof.de/tutorials/polfilter/>, 2023. Aufgerufen am 25.02.2023.
- [Hol18] M. Hollemans. MobileNet version 2. <https://machinethink.net/blog/mobilenet-v2/>, 2018. Aufgerufen am 25.02.2023.
- [Nie22] A. Niedl. Optische Erkennung elektronischer Komponenten in der Prüfautomatisierung mithilfe von KI-Methoden. 2022.
- [Nis12] A. Nischwitz, M. Fischer, P. Haberäcker und G. Socher. *Computergrafik und Bildverarbeitung*. Vieweg+Teubner Verlag Wiesbaden, 2012.
- [Ope08] Hough Line Transform. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html, 2008. Aufgerufen am 25.02.2023.
- [Suz85] S. Suzuki und K. Abe. Topological Structural Analysis of Digitized Binary Images by Border Following. 1985.
- [Sze21] R. Szeliski. *Computer Vision: Algorithms and Applications*. 2021.