Jakob ZIEGLER

Optische Kohärenztomographie zur Klassifizierung und Positionsbestimmung von Filmtabletten in einem pharmazeutischen Beschichtungsprozess

Masterarbeit

Institut für Medizintechnik
Technische Universität Graz
Kronesgasse 5, A - 8010 Graz
Vorstand: Univ.-Prof.Dipl.-Ing.Dr.techn. Rudolf Stollberger

Betreuer: Dipl.-Ing. Daniel Markl

Begutachter: Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Hermann Scharfetter

Graz, (März, 2014)

Eidesstattliche Erklärung

Ich erkläre an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

.....	.....
Graz, am	(Unterschrift)

Statutory Declaration

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 eihter literally or by content from the used sources.

.....	.....
date	(signature)

Optische Kohärenztomographie zur Klassifizierung und Positionsbestimmung von Filmtabletten in einem pharmazeutischen Beschichtungsprozess

Zusammenfassung

Ziel Diese Diplomarbeit dient als Basis für einen Algorithmus, der eine in-line Überwachung der Schichtdicke von Filmtabletten während des Beschichtungsvorganges realisieren soll. Als Messverfahren wird die optische Kohärenztomographie eingesetzt. Innerhalb der aufgenommenen Bilddaten sollen die einzelnen Tabletten erkannt und deren relative Position zum Messsystem bestimmt werden.

Methoden Die Algorithmen wurden in der Entwicklungsumgebung MATLAB (Version 8.2, The MathWorks Inc., Natick, Massachusetts/USA) implementiert. Die Daten werden mit den Verfahren „Logistische Regression“ und „Support Vector Machines“, sowie einem „Hidden Markov Modell“ klassifiziert, die Positionsbestimmung erfolgt mit einem geeigneten „Circle-Fitting“-Verfahren.

Ergebnisse Die Klassifizierung der Daten erfolgt mit einer Genauigkeit von fast 96%, bei ca. 75% der gefundenen Tabletten kann eine Positionsbestimmung erfolgreich durchgeführt werden.

Conclusio Sowohl Genauigkeit, als auch Geschwindigkeit des erstellten Algorithmus genügen den Anforderungen an eine spätere in-line Überwachung des Beschichtungsvorganges.

Schlüsselwörter Tablettenbeschichtung, Optische Kohärenztomographie, Support Vector Machines, Hidden Markov Modell, Circle-Fitting

Optical coherence tomography for classification and position evaluation of tablets in a pharmaceutical coating process

Abstract

Objective This master thesis provides the basis for an algorithm realizing in-line monitoring of a pharmaceutical tablet coating process. The used measurement method is optical coherence tomography. The Goal is to detect single tablets within images, which are generated from the measured data. Additionally the position of the tablets in relation to the measurement system should be determined.

Methods The algorithms are implemented in MATLAB (Version 8.2, The MathWorks Inc., Natick, Massachusetts/USA). The data is classified using the methods „logistic regression“, „support vector machines“ and „hidden markov models“ and the position of the tablets is determined by a dedicated „circle fitting“ algorithm.

Results The classification of the data offers an accuracy of nearly 96%, a successful position determination is achieved for approximatly 75% of the detected tablets.

Conclusion Accuracy and speed of the implemented algorithm meet the requirements of an in-line monitoring tool for film coating processes.

Key Words tablet coating, optical coherence tomography, support vector machines, hidden markov model, circle fitting

Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung

1.1 Tablettenbeschichtung

Das Beschichten (engl.: coating) von Filmtabletten ist ein oft angewandter Prozess zur Herstellung von Arzneimitteln. Üblicherweise ist dabei das Auftragen einer dünnen kontinuierlichen Schicht auf ein Objekt, das einen oder mehrere aktive pharmazeutische Wirkstoffe enthält, gemeint. In vielen Fällen wird dieser Prozess aus marketingtechnischen Gründen eingesetzt, wie es beispielsweise visuelle Attraktivität, Geschmacksbildung oder das Markenzeichen des Herstellers sein können. Viel wichtiger allerdings ist das Ziel, die Funktionalität von Tabletten durch Beschichtungen zu verbessern. Diese Funktionalität zeichnet sich durch Merkmale wie Produktstabilität, Haltbarkeit, erleichterte Herstellung oder bessere Kontrolle der Wirkstofffreisetzung aus. Eine Möglichkeit, mit Hilfe derer solche funktionellen Beschichtungen die Wirkstofffreisetzung kontrollieren, ist eine Schicht mit einer Widerstandsfähigkeit gegenüber bestimmten pH-Werten aufzutragen. Dies verleiht der Tablette beispielsweise Resistenz gegenüber Verdauungssäften. Die große Herausforderung bei der Herstellung von Filmtabletten besteht darin, die gewünschte Dicke und Gleichförmigkeit einer Beschichtung zu erreichen. Diesbezügliche Unzulänglichkeiten resultieren etwa in verspäteter oder verfrühter Wirkstofffreisetzung. Beschichtungsdicke und -gleichförmigkeit sind also die kritischen Parameter in einem Beschichtungsprozess, und bedürfen daher einer in-line Überwachung. Bezüglich der Gleichförmigkeit unterscheidet man bei der Prozessüberwachung während der Tablettenbeschichtung folgende zwei Größen [1], [2]:

Intra-tablet coating uniformity Die intra-tablet coating uniformity ist die Schwankung der Beschichtungsdicke innerhalb einer einzelnen Tablette, wie sie vor allem zwischen deren verschiedenen Flächen auftritt. Aber auch Produktionsfehler wie Risse oder (Luft-) Einschlüsse gehen in diese Größe ein.

Inter-tablet coating uniformity Die inter-tablet coating uniformity betrifft die Abweichungen der Beschichtungsdicke zwischen den Tabletten innerhalb einer Charge.

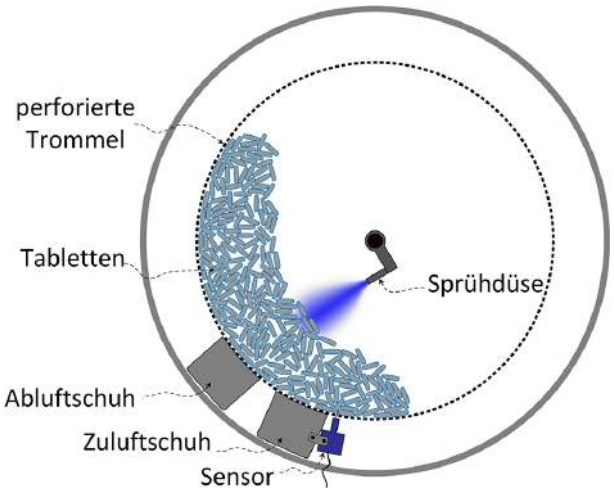
1.2 Beschichtungsverfahren

Die drei wichtigsten Verfahren zur Tablettenbeschichtung sind:

- Aufbringen eines Puders durch elektrostatische Kräfte
- Eintauchen des Tablettenkerns in eine Flüssigkeit
- Aufsprühen einer Flüssigkeit

Das Besprühen des Tablettenkerns mit einer Flüssigkeit ist hierbei die am häufigsten verwendete Methode, um den Beschichtungsfilm auf das Objekt zu applizieren. Abhängig von der Geometrie der zu beschichtenden Objekte erfolgt der Vorgang mittels eines Wirbelschicht- oder eines Trommelcoaters (engl.: drum coater) [3]. Für weitere Informationen bezüglich der Beschichtung von Tabletten siehe [4] und [5].

Der Aufbau eines Trommelbeschichters ist in Abbildung 1 dargestellt.



Die Tabletten werden innerhalb einer perforierten Trommel mit der Sprühdüse besprüht und nach dem Sprühvorgang luftgetrocknet. Für die notwendige Luftzirkulation sorgen der Zuluft- und der Abluftschuh. Dieser Prozess wird solange wiederholt, bis die gewünschte Schichtdicke erreicht ist. Die Überwachung des Beschichtungsvorganges wird in der zukünftigen Realisierung mit einem außerhalb der Trommel angebrachten Sensor erfolgen.

Abbildung 1: Aufsprühen der Beschichtung von Tabletten in einem Trommelbeschichter und Überwachung des Vorganges mittels Sensor

1.3 Optische Kohärenztomographie

1.3.1 Hintergrund

In-line Monitoring erfordert eine schnelle und robuste Methode, die Informationen über die Tablettenbeschichtung liefert. Zerstörende Messmethoden, wie die Rasterelektronenmikroskopie, erfordern einen Schnitt durch die Tablette, was einen erhöhten Zeitaufwand mit sich bringt. Daher sind solche Technologien für die vorliegende Problemstellung nicht geeignet. Verfahren, die eine zerstörungsfreie Messung erlauben, sind beispielsweise Nah-Infrarot- und Raman-Spektroskopie. Ein großer Nachteil dieser Methoden ist allerdings, dass sie keinen Absolutwert der Schichtdicke liefern und, basierend auf vorangegangene Messungen, kalibriert werden müssen. Tomographische Methoden wie Röntgen-Mikrotomographie, Magnetresonanztomographie, Terahertz-Puls-Bildgebung (engl.: terahertz pulse imaging, TPI) und Optische Kohärenztomographie (engl.: optical coherence tomography, OCT) haben diese Nachteile nicht. Die Voraussetzungen für ein in-line Überwachungssystem, wie zum Beispiel eine kurze Akquisitionszeit, hohe räumliche Auflösung und hohe Sensitivität, liefern unter diesen Methoden allerdings nur TPI und OCT [3]. Der Vorteil der Optischen Kohärenztomographie im Vergleich zum TPI ist dabei die schnellere Durchführung der Messungen bei höherer Auflösung. Da ein Großteil der produzierten Menge an Tabletten messtechnisch erfasst werden soll, muss die Datenbeschaffung und -auswertung in sehr kurzer Zeit erfolgen. Ein optimaler Kompromiss zwischen Akquisitionszeit und Datenqualität muss also gefunden werden.

1.3.2 OCT-Verfahren

Die Optische Kohärenztomographie ist ein Verfahren, das zerstörungsfreie, kontaktlose und hochauflösende Bildgebung ermöglicht. Aufgrund dieser Eigenschaften ist ein großes Einsatzgebiet dieser Technologie die Biomedizin, wo sie vor allem im Bereich der Augenheilkunde [6], der Dermatologie [7] und der Zahnmedizin [8] angewandt wird.

Das Prinzip ist dem der Ultraschall-Bildgebung ähnlich und besteht darin, einen Lichtstrahl auf das Messobjekt zu richten und die Intensität, sowie die Echozeitverzögerung des zurückgestreuten Lichtes zu messen. Im Gegensatz zum Ultraschall kann allerdings aufgrund der hohen Geschwindigkeit des Lichtes die Echozeit nicht direkt gemessen werden. Deshalb nutzt man bei der OCT Interferometrietechniken, speziell das Prinzip der Weißlichtinterferometrie. Grundlage dieses Verfahrens sind die beiden Kernbegriffe *Interferenz* und *Kohärenz*.

Interferenz Als Interferenz bezeichnet man die Überlagerung von Wellenzügen nach dem Superpositionsprinzip. Man unterscheidet destruktive Interferenz, bei der sich die Wellen gegenseitig auslöschen, und konstruktive Interferenz, bei der sie sich gegenseitig verstärken. Voraussetzung für eine zeitlich konstante Interferenz ist kohärentes Licht.

Kohärenz Zwei Lichtwellen sind zeitlich kohärent, wenn sie eine feste Phasenbeziehung zueinander, also die selbe Wellenlänge, haben. Dabei schwingen die Wellen nur für eine bestimmte Zeit mit unveränderter Phase. Diese Dauer, in der die Lichtwelle in vorhersagbarer Art und Weise schwingt, wird Kohärenzzeit genannt, die entsprechende Länge heißt Kohärenzlänge. Davon zu unterscheiden ist der Begriff der räumlichen Kohärenz. Zwei Lichtwellen sind räumlich kohärent, wenn sie die gleiche Ausbreitungsrichtung haben. Abbildung 2 veranschaulicht den Unterschied zwischen zeitlicher und räumlicher Kohärenz.

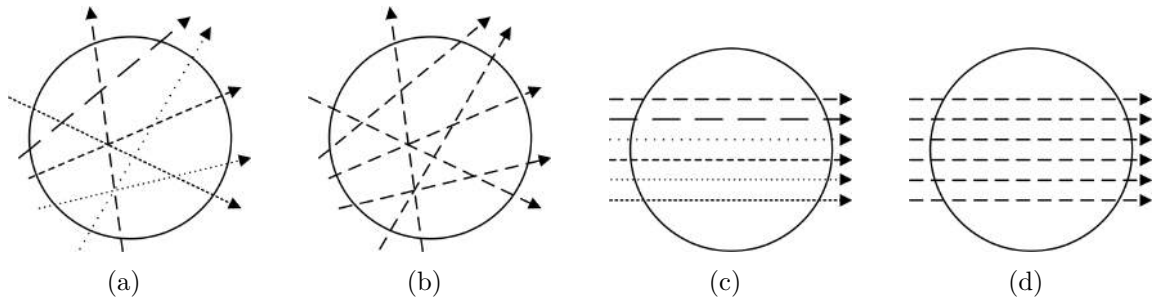


Abbildung 2: Lichtstrahlen auf einem kreisförmigen Objekt, die unterschiedlich strichlierten Linien stellen dabei verschiedene Wellenlängen dar. a) zeitlich und räumlich inkohärent, b) zeitlich kohärent, c) räumlich kohärent, d) zeitlich und räumlich kohärent

Weißlichtinterferometrie Ein Lichtstrahl einer Lichtquelle mit hoher räumlicher und niedriger zeitlicher Kohärenz (hohe axiale Auflösung, Formel 1) wird mittels Strahlteiler in zwei Teilstrahlen aufgeteilt, die auf einen Referenzspiegel und das Messobjekt gelenkt werden. Infolge einer Änderung des Brechungsindexes aufgrund zweier unterschiedlicher Schichten wird der Messstrahl reflektiert und gelangt über den gleichen Strahlteiler zu einem Detektor. Nur wenn die Strahlen innerhalb der Kohärenzzeit am Detektor auftreffen, interferieren sie konstruktiv und es ist ein Messsignal detektierbar. Vom bekannten Abstand des Referenzspiegels kann man somit auf den Abstand der reflektierenden Schicht in der Probe rückschließen. Der prinzipielle Aufbau eines Weißlichtinterferometers ist in Abbildung 3 dargestellt.

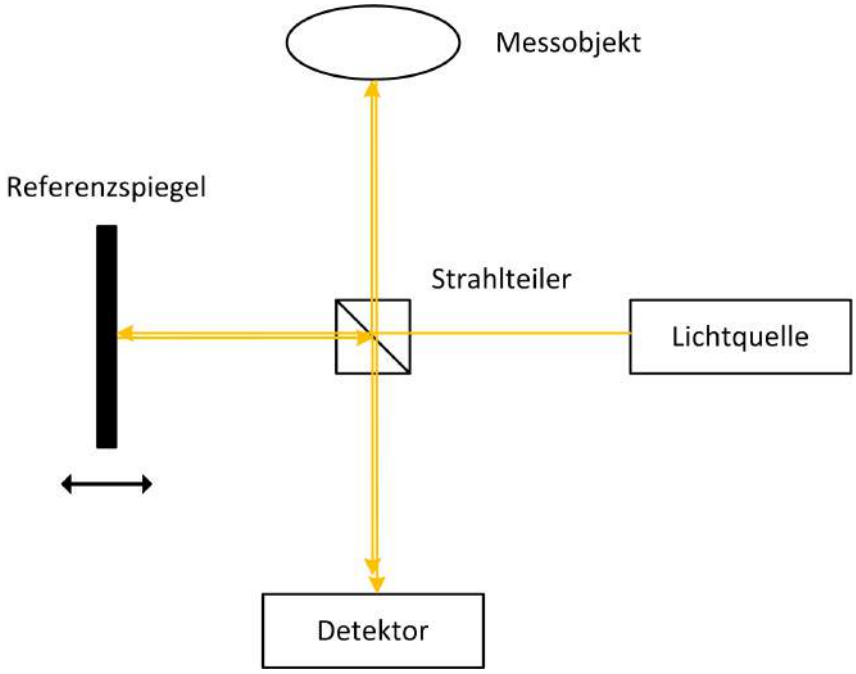
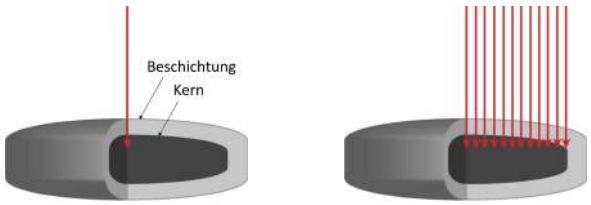


Abbildung 3: Prinzipieller Aufbau eines Weißlicht-Interferometers

Wenn also verschiedene Schichten im Messobjekt erfasst werden sollen, ist es erforderlich, die Position des Referenzspiegels zu verändern. Man erhält Information über die Lichtintensität der Rückstreuung in Abhängigkeit der Position in der Probe und spricht, analog zur Ultraschall-Sonographie, von einem A-Scan.

Genau nach diesem Prinzip erfolgt auch bei der Optischen Kohärenztomographie die Datenaufnahme. Interessant sind dabei allerdings nicht nur die einzelnen A-Scans, sondern vielmehr ein ganzer Querschnitt durch das Testobjekt. Solch ein Querschnittsbild, auch B-Scan genannt, erhält man demnach, wenn die oben beschriebene Messung mit veränderlicher Referenzspiegelposition (z-Richtung) zudem für verschiedene Punkte in x- oder y-Richtung wiederholt wird (siehe Abbildung 4). Damit lässt sich ein großer Vorteil der Optischen Kohärenztomographie erkennen: die axiale Auflösung ist von der lateralen Auflösung unabhängig.



(a) (b)

Abbildung 4: a) Einzelner A-Scan, b) B-Scan, bestehend aus mehreren nebeneinander erfolgenden A-Scans. Als Untersuchungsobjekt ist eine Tablette dargestellt.

Die axiale Auflösung ist durch die zentrale Wellenlänge und die Bandbreite der Lichtquelle limitiert. Üblicherweise ist sie als halbe Kohärenzlänge l_c definiert und gegeben durch [9]:

$$\Delta z = \frac{l_c}{2} = K \frac{\lambda_c^2}{\Delta\lambda} \tag{1}$$

K ist eine Konstante (0.44 für eine Gauß'sche Spektralverteilung), λ_c ist die zentrale Wellenlänge und $\Delta\lambda$ stellt die Halbwertbreite (engl.: full width half maximum, FWHM) des Spektrums dar. Die laterale Auflösung ist nur von der Optik des Messsystems und der zentralen Wellenlänge der Lichtquelle abhängig und lässt sich folgendermaßen berechnen [10]:

$$\Delta x = \frac{4\lambda_c f}{\pi D} \tag{2}$$

f ist hierbei die Fokuslänge der verwendeten Linse, D der Strahldurchmesser und λ_c die zentrale Wellenlänge.

1.3.3 OCT: Zeitbereich versus Frequenzbereich

Das OCT Verfahren kann anhand der Art der Signalerfassung in zwei Klassen unterteilt werden, die OCT im Zeitbereich (engl.: time-domain OCT, TdOCT) und die OCT im Frequenzbereich (engl.: frequency-domain OCT, FdOCT). Eine Aufteilung der verschiedenen OCT Verfahren ist in Abbildung 5 zu sehen. Die Arbeitsweise der TdOCT entspricht der des Weißlichtinterferometers (Abbildung 3), der Referenzspiegel muss also bewegt werden, um verschieden tiefe Schichten in der Probe erfassen zu können. Im Gegensatz zur TdOCT wird bei der FdOCT kein beweglicher, sondern ein fixer Referenzspiegel verwendet, was die Robustheit des Messaufbaus erhöht. Eine Darstellung des prinzipiellen Aufbaus der beiden Verfahren findet sich in Abbildung 6. Die Information über die Position der Reflexionspunkte an den einzelnen Schichten im Untersuchungsobjekt wird bei der FdOCT aus dem Spektrum des gesamten zurückgestreuten Lichtes bezogen. Das oszillatorische Originalsignal wird dabei entsprechend dieser Schichten frequenzmoduliert [9]. Auf das so entstehende Messsignal wird danach eine Diskrete Fourier Transformation (DFT) angewandt. Gegenüber der TdOCT sind bei der FdOCT sowohl die Sensitivität (typischerweise um mehr als 20dB, [11]), als auch die Akquisitionsrate (Faktor 100, [9]) stark erhöht.

Der Singal-Rausch-Abstand ist somit bei der FdOCT im Vergleich zur TdOCT höher, wie Gleichung (3) zeigt [9]. Z_{max} ist hierbei der axiale Bildbereich, Δz die axiale Auflösung.

$$SNR_{FdOCT} = \frac{2\ln 2}{\pi} \cdot \frac{Z_{max}}{\Delta z} \cdot SNR_{TdOCT} \tag{3}$$

Die FdOCT kann anhand der verwendeten Komponenten zusätzlich in zwei Unterklassen aufgeteilt werden. Diese Unterklassen sind einerseits spectral-domain OCT (SdOCT), bei der die Signalerfassung mit einem Spektrometer erfolgt, und andererseits die swept-source OCT (SSOCT), bei der ein durchstimmbarer Hochgeschwindigkeits-Laser als Lichtquelle dient.

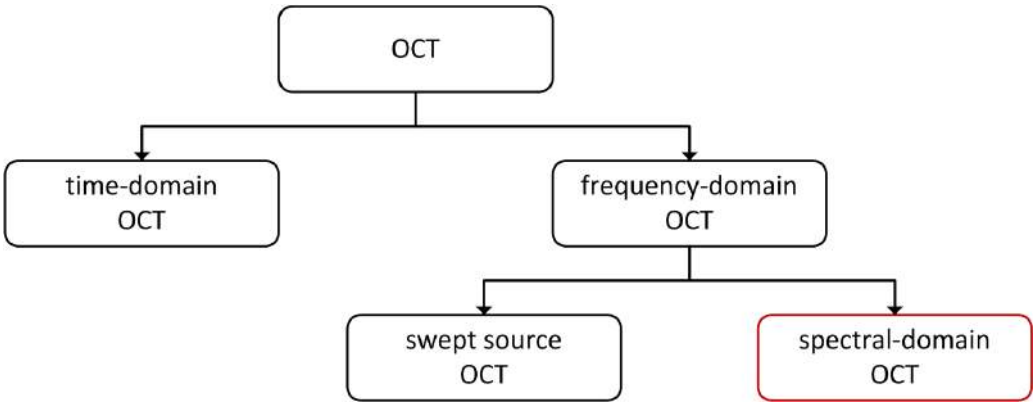
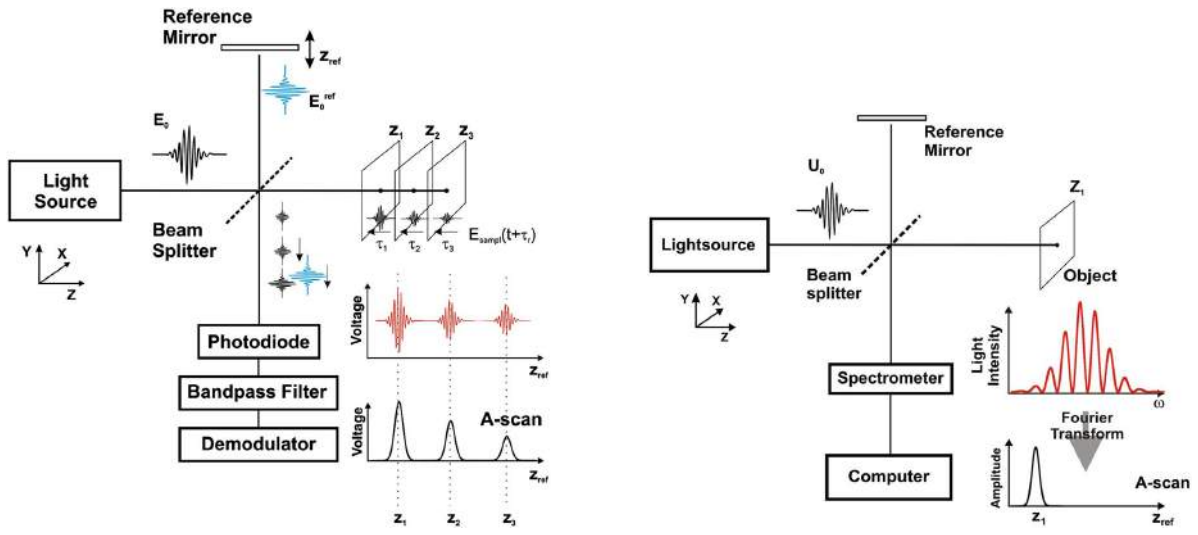


Abbildung 5: Einteilung der wichtigsten Klassen der Optischen Kohärenztomographie



(a) (b)

Abbildung 6: Messaufbau von zwei unterschiedlichen Arten der Optischen Kohärenztomographie [9]. a) time-domain OCT (TdOCT), b) spectral-domain OCT (SdOCT)

Die für diese Arbeit zur Verfügung gestellten Messdaten wurden mittels spectral-domain OCT aufgezeichnet, weshalb an dieser Stelle auf die anderen Verfahren nicht weiter eingegangen wird. Für zusätzliche Informationen bezüglich Optischer Kohärenztomographie siehe [9] und [11].

1.4 Messaufbau

Wie bereits in Abbildung 1 beschrieben, wird bei der in-line Überwachung des Beschichtungsvorganges durch die Löcher der perforierten Trommel hindurch gemessen. Um diese Messumgebung zu simulieren, wurde der nachfolgende Messaufbau verwendet. Die Messobjekte wurden unter einem ebenen Lochblech mit einem Lochdurchmesser von *3 mm* und einem Lochabstand von *5 mm* in verschiedenen Positionen platziert. Danach wurde der Messkopf mit verschiedenen Geschwindigkeiten über dieses Lochblech bewegt und die Messdaten während des gesamten Vorganges aufgezeichnet. Eine schematische Darstellung ist in Abbildung 7, der tatsächliche Messaufbau in Abbildung 8 zu sehen.

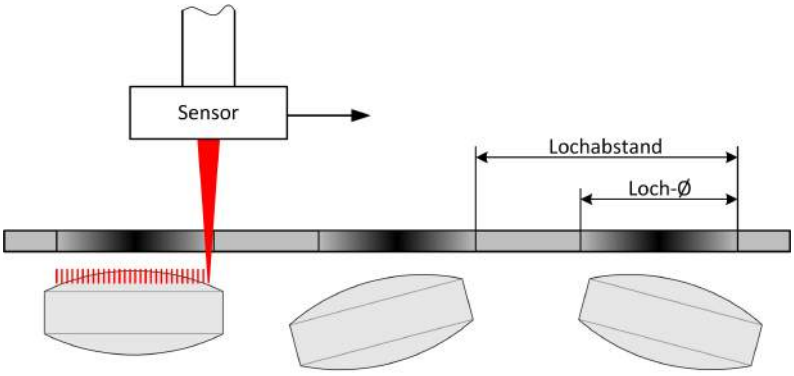
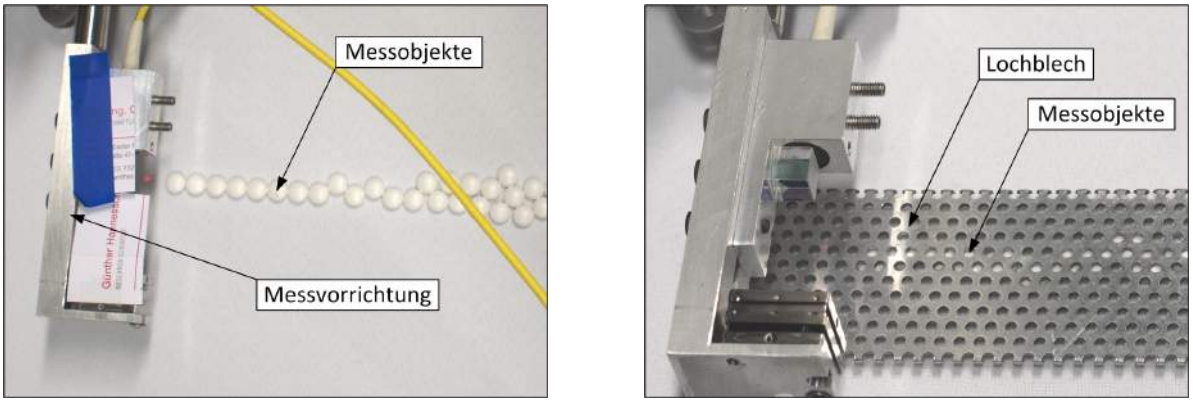


Abbildung 7: Schematische Darstellung des Messaufbaus. Der Sensor wird mit verschiedenen Geschwindigkeiten über das Lochblech und die darunter platzierten Tabletten bewegt und nimmt währenddessen Daten auf.



(a) (b)

Abbildung 8: Abbildung des Messaufbaus. a) Datenaufnahme ohne Lochblech, b) Messung mit dem über den Messobjekten platzierten Lochblech.

1.5 Materialien

1.5.1 Messsystem

Das Messsystem zur Datenaufnahme wurde von der Firma RECENDT GmbH (Research Center for Non Destructive Testing GmbH, Altenbergerstraße 69 - 4040 Linz) entwickelt. Eine Übersicht über die Messparameter ist in Tabelle 1 zu finden. Die verwendeten Kernkomponenten des Systems sind:

Lichtquelle: Superlum SLD 351-HP (Superlum Diodes Ltd., Irland)

CCD Kamera: AViiVA EM4 CL 2010 OCT Version (Atmel Corporation, Kalifornien/USA)

Strahlteiler: nichtpolarisierend, 700-1000 nm (Thorlabs Inc., New Jersey/USA)

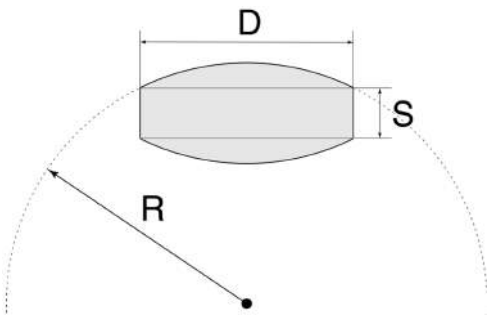
Tabelle 1: Übersicht über die Parameter der verwendeten Messdaten

Parameter	Wert
A-Scans pro B-Scan	1000
Zeit pro A-Scan	67.56 μs
Akquisitionsrate	~15 kHz
Kamerapixel des Spektrometers	2048
untere Wellenlänge	720 nm
obere Wellenlänge	930 nm
axiale Auflösung	7 μm in Luft
laterale Auflösung	ca. 20 μm (abhängig vom Messkopf)

1.5.2 Messobjekte

Filmtabletten: Thrombo ASS 50 mg (G.L. Pharma, Österreich)

Als Wirkstoff enthalten diese Filmtabletten 50 mg Acetylsalicylsäure. Die Geometrie der verwendeten Tabletten ist in Abbildung 9 dargestellt. Der Brechungsindex der Beschichtung beträgt 1.48 (siehe [3]).



D ist der Tablettendurchmesser, S die Höhe des Steges. Der Radius R definiert die kugelsegmentartige Ober- und Unterseite der Tablette und beträgt laut RECENDT GmbH (Research Center for Non Destructive Testing GmbH, Altenbergerstraße 69 - 4040 Linz) für die untersuchten Tabletten ohne Beschichtung 6.94 mm.

Abbildung 9: Geometrie der für die Messdaten verwendeten Tabletten

Die untersuchten Messobjekte wurden zu unterschiedlichen Zeitpunkten während des Beschichtungsvorganges aus der Produktion entnommen. Dadurch ergaben sich fünf verschiedene Beschichtungsdicken. Eine Gegenüberstellung von Beschichtungsdauer und ungefährer Beschichtungsdicke der Testobjekte ist in Tabelle 2 angeführt.

Tabelle 2: Vorkommende Schichtdicken in den vorhandenen Messdaten

Beschichtungsdauer	ungefähre Schichtdicke
10 Minuten	0 μm
37 Minuten	20 μm
70 Minuten	32 μm
96 Minuten	50 μm
127 Minuten	70 μm

1.6 Messdaten

Die verwendeten Messdaten wurden von der Firma RECENDT GmbH (Research Center for Non Destructive Testing GmbH, Altenbergerstraße 69 - 4040 Linz) geliefert, die Datenerstellung entspricht jener in [12]. Dabei wurde der OCT Sensor mit unterschiedlichen Geschwindigkeiten (0.2 m/s, 0.3 m/s, 0.4 m/s, 0.5 m/s und 0.6 m/s) über ein Lochblech und die darunter liegenden Testobjekte bewegt, welche die oben angeführten, fünf verschiedenen Schichtdicken aufwiesen.

Da angenommen wurde, dass die relative Geschwindigkeit zwischen OCT-Messkopf und Tabletten in der Produktion etwa im Bereich von 0.4 m/s liegt, wurden für die Erstellung und das Training des Algorithmus nur Daten verwendet, die mit 0.4 m/s beziehungsweise 0.5 m/s aufgezeichnet wurden. Sämtliche fünf verschiedenen Schichtdicken wurden dabei berücksichtigt. Insgesamt wurden 207000 A-Scans, also 207 Bilder für die Erstellung des Algorithmus herangezogen. Tabelle 3 zeigt, wie diese Daten auf die unterschiedlichen Beschichtungsdicken aufgeteilt waren:

Tabelle 3: Anzahl der vorhandenen A-Scans je Schichtdicke

Schichtdicke	vorhandene A-Scans
0 μm	44000
20 μm	40000
32 μm	42000
50 μm	30000
70 μm	51000

Die Messdaten lagen je B-Scan als Spektraldaten vor, zusätzlich wurde für jede unterschiedliche Beschichtungsdicke und Geschwindigkeit ein Referenzspektrum aufgenommen. Außerdem wurden für die durchgeführten Messungen bereits umgewandelte Bilddaten mitgeliefert. Mit Hilfe dieser Bilddaten konnten die vorhandenen Messungen auf jene zusammengefasst werden, bei denen tatsächlich ein oder mehrere Tabletten erkennbar waren. Bilder ohne sichtbares Messobjekt wurden nicht berücksichtigt.

1.7 Aufgabenstellung

Ziel dieser Arbeit ist es die Basis für einen Algorithmus zur automatischen Bestimmung der Schichtdicke zu schaffen, mit Hilfe derer eine spätere in-line Überwachung eines Beschichtungsvorganges samt Messung der aktuellen Schichtdicke der Tabletten ermöglicht wird. Dazu sollen die bereitgestellten Messdaten klassifiziert werden, um eine Trennung zwischen nützlichen und unbrauchbaren Daten vornehmen zu können. Innerhalb der als *Tablette* bestimmten und damit brauchbaren Daten soll eine Positionsbestimmung durchgeführt werden, um die für die nachfolgenden Schritte erforderliche Lage des Messobjektes zu ermitteln. Die gesamte Datenverarbeitung erfolgt dabei in der Entwicklungsumgebung MATLAB (Version 8.2, The MathWorks Inc., Natick, Massachusetts/USA).

Der erstellte Algorithmus lässt sich, wie in Abbildung 10 dargestellt, grob in vier Teilbereiche aufteilen.

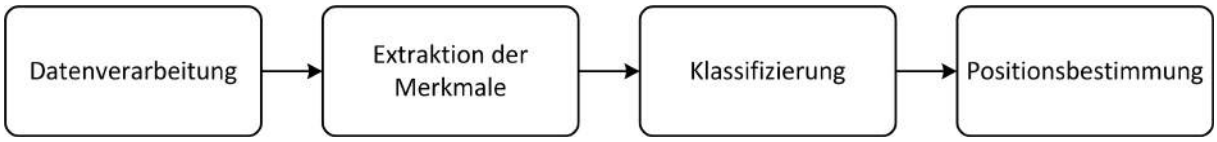


Abbildung 10: Blockdiagramm der einzelnen Teilbereiche des Algorithmus

Datenverarbeitung Es wurde versucht die Klassifizierung direkt mit den spektralen Rohdaten der A-Scans durchzuführen. Hierbei konnten allerdings keine Merkmale gefunden werden, die zufriedenstellende Ergebnisse erlaubten. Aufgrund dessen war eine Umwandlung der Rohdaten in Bilddaten notwendig, aus welchen die Klassifizierungsmerkmale extrahiert und ansprechende Klassifizierungsergebnisse erreicht wurden. Da die Bilddaten für die spätere Positionsbestimmung aber sowieso erforderlich waren, stellte dies keinen Mehraufwand dar.

Extraktion der Merkmale Um eine Klassifizierung durchführen zu können, müssen die einzelnen Klassen anhand bestimmter Merkmale unterschieden werden können. Dieser Teil des Algorithmus extrahiert die aussagekräftigsten Merkmale der Bilddaten (pro A-Scan) und liefert so die Grundlage für die nachfolgende Klassifizierung. Mit dieser wird überprüft, ob die jeweiligen Daten tatsächlich zu einem Messobjekt gehören und somit für weitere Analysen relevant sind, oder nicht.

Klassifizierung Durch den oben beschriebenen Messaufbau befindet sich der Messsensor während der Datenaufnahme entweder über dem Metall des Lochblechs oder er misst durch eines dessen Löcher hindurch. Befindet er sich über einem Loch, so erfasst er entweder die Messdaten einer darunter platzierten Tablette oder erhält als Messsignal nur jenes von Luft. Aus diesen vom Messsystem gelieferten Daten soll daher eine Trennung zwischen den drei möglichen Zuständen *Luft*, *Metall* und *Tablette* erfolgen. Somit können die Tabletten im Bild erfasst und eine Positionsbestimmung durchgeführt werden.

Positionsbestimmung Wurden A-Scan-Daten bei der Klassifizierung als zu einer Tablette gehörend erkannt, kann aus diesen Bilddaten die Tablettenstruktur eruiert werden. Diese Strukturdaten wiederum ermöglichen eine Positionsbestimmung der Tablette im Bild und sind somit Basis für eine nachfolgende Bestimmung der Schichtdicke. Dieser Schritt ist notwendig, da aufgrund der gekrümmten Oberfläche der Tabletten und der Position der Tablette relativ zum Messsensor die Bilddaten der Beschichtung verzerrt dargestellt werden. Um die tatsächliche Dicke der Tablettenbeschichtung bestimmen zu können, muss diese Verzerrung mit rechnerischen Operationen eliminiert werden. Die genaue Kenntnis über die Position des Messobjektes ist dafür Voraussetzung.

2 Datenverarbeitung

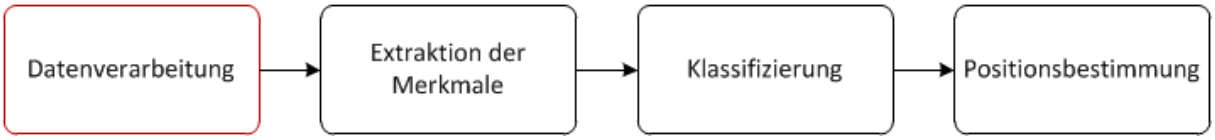


Abbildung 11: Erster Schritt des Gesamt-Algorithmus: Datenverarbeitung

2.1 Theorie

Basis und erster Schritt des Algorithmus ist die Datenverarbeitung (Abbildung 11). Dieser Abschnitt gliedert sich in drei Teilschritte (Abbildung 12): die Subtraktion des Referenzspektrums, die Transformation der Rohdaten vom Wellenlängen- in den Wellenzahlbereich und die Durchführung der diskreten Fouriertransformation.

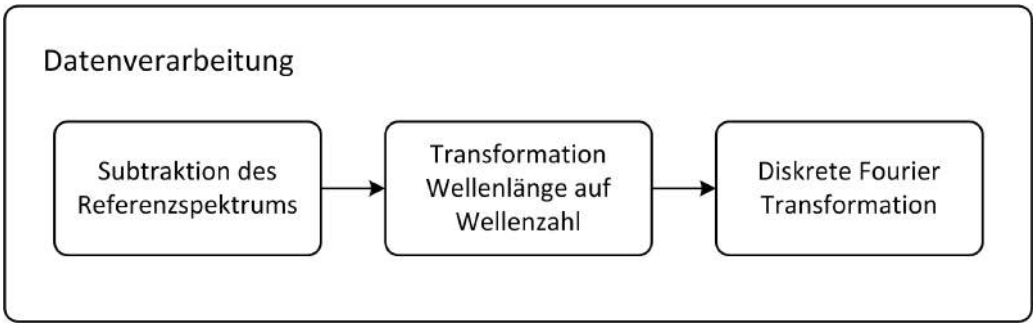


Abbildung 12: Teilschritte der Datenverarbeitung

Die Rohdaten bestehen aus einzelnen B-Scans, die aus jeweils 1000 aneinandergereihten Interferenzspektren zusammengesetzt sind. Jedes dieser Interferenzspektren besteht dabei aus 2048 Datenpunkten, da die Messungen mit einer Zeilenkamera mit 2048 Pixel aufgenommen wurden. Um aus diesen Spektraldaten die erforderlichen Bilddaten zu erhalten, wird eine diskrete Fourier-Transformation (DFT) durchgeführt.

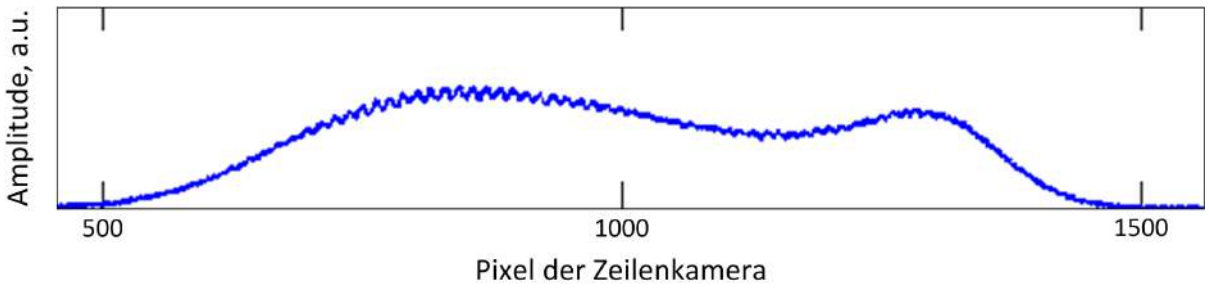
Vor diesem Prozess ist es allerdings sinnvoll die Daten aufzubereiten. Ziel ist ein besserer Signal-Rausch-Abstand, um eine optimierte Grundlage für sämtliche weiteren Arbeitsschritte zu schaffen.

2.2 Methoden

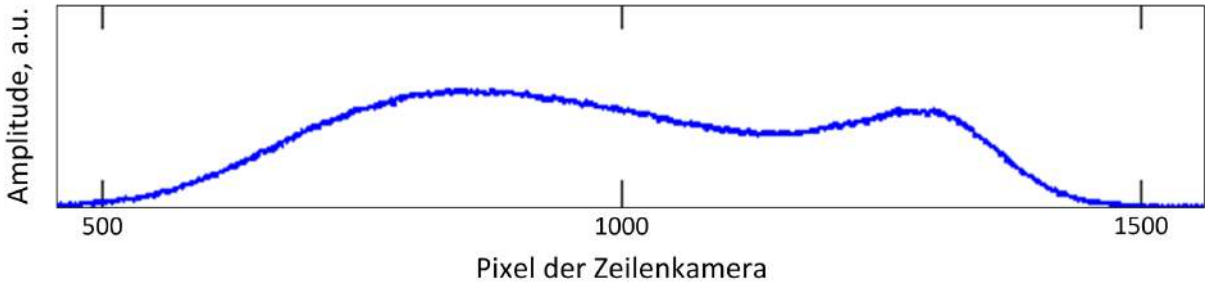
Subtraktion des Referenzspektrums Die Subtraktion eines Referenzspektrums von den spektralen Rohdaten ist eine Methode der Datenaufbereitung. Für dieses Referenzspektrum wird bei abgedecktem Objektpfad des Interferometers eine Messung durchgeführt. Die aufgenommenen Daten enthalten somit nur das Signal des Referenzpfades, weches näherungsweise dem Spektrum der verwendeten Lichtquelle entspricht und von den nachfolgenden Messungen abgezogen wird.

Durch diesen Prozess werden systematische Messfehler eliminiert, wodurch die Daten für die nächsten Arbeitsschritte besser verwertbar werden.

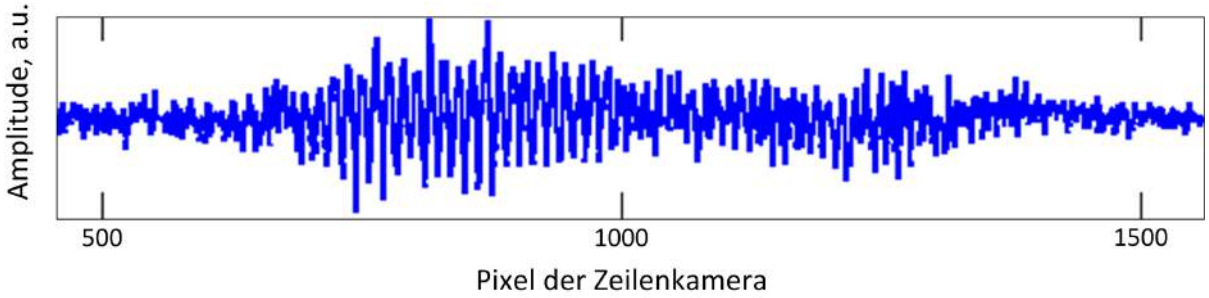
Zu diesem Zweck wurde von der Firma RECENDT GmbH (Research Center for Non Destructive Testing GmbH, Altenbergerstraße 69 - 4040 Linz) zusätzlich zu jedem Datensatz ein entsprechendes Referenzspektrum mitgeliefert. Die Subtraktion des Referenzspektrums von den Rohdaten ist in Abbildung 13 dargestellt. Die Auswirkung dieses Schrittes im Bildbereich ist in Abbildung 14 zu sehen.



(a)

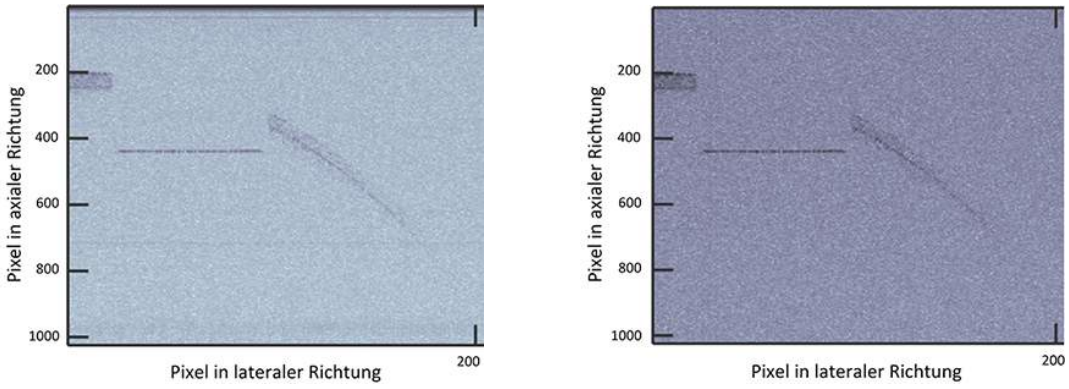


(b)



(c)

Abbildung 13: Vorgang der Datenaufbereitung, dargestellt anhand einer Messung von einer Tablette. a) Rohdaten, b) Datensatz des Referenzspektrums, c) Differenzspektrum, erhalten durch die Subtraktion der Referenz von den Rohdaten. Dargestellt ist hier nur ein Teil der 2048 Pixel der Zeilenkamera, jedes Pixel entspricht dabei einem bestimmten Wellenlängenbereich.



(a) (b)

Abbildung 14: Auswirkungen der Subtraktion des Referenzspektrums im Bildbereich. a) unbehandelte Rohdaten, b) Daten mit subtrahierter Referenz. Es ist deutlich zu erkennen, dass die horizontalen Strukturen und Schlieren (vor allem im obersten Bildbereich) nach diesem Arbeitsschritt eliminiert wurden.

Transformation Wellenlänge auf Wellenzahl Die zur Verfügung stehenden Messdaten wurden mit einem Spektrometer aufgezeichnet, welches linear im Wellenlängenbereich arbeitet. Die Wellenzahl k ist definiert als der mit 2π skalierte Reziprokwert der Wellenlänge λ:

$$k = \frac{2\pi}{\lambda} \approx \frac{1}{\lambda} \tag{4}$$

Diese Rechenoperation ist nichtlinear, daher werden auch die Daten nichtlinear transformiert. Da die Fourier-Transformation aber eine lineare Operation ist, müssen auch die Daten in linearer Art vorliegen. Es ist also erforderlich eine Interpolation durchzuführen, die die Daten in ein Spektrum umwandelt, welches linear im Wellenzahlbereich ist.

Diskrete Fourier-Transformation Um aus den Spektren die gewünschten Informationen über die Tiefenstrukturen der Messobjekte zu erhalten, wird eine inverse diskrete Fourier-Transformation durchgeführt. Dabei werden die Frequenzanteile des Spektrums in die Bilddaten des Zeitbereichs transformiert. Zu beachten ist, dass durch diese Operation die 2048 Kamerapixel des Spektrometers auf die Hälfte, also auf 1024 axiale Bildpunkte reduziert werden.

Wie später in den Abbildungen zu sehen ist, erscheint das Lochblech relativ häufig unterhalb der Tabletten. Aufgrund des in Kapitel 1 besprochenen Messaufbaus kann das allerdings nie der Fall sein. Diese falsche Darstellung des Metalls ist ein Artefakt der Fourier Transformation. Das Signal des Lochblechs befindet sich eigentlich oberhalb des dargestellten Bildbereichs, wird aber durch die Transformation in das Bild hinein gespiegelt.

3 Extraktion der Merkmale

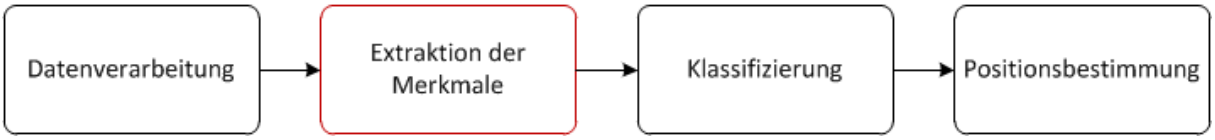
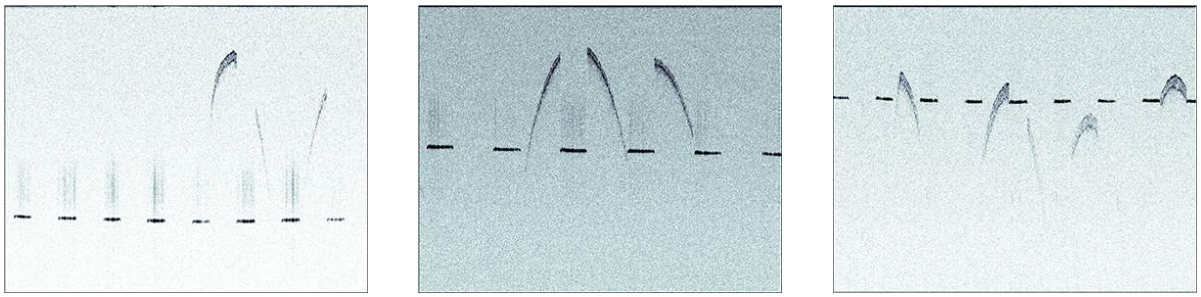


Abbildung 15: Zweiter Schritt des Gesamt-Algorithmus: Extraktion der Merkmale

3.1 Theorie

Eine Klassifizierung der Messdaten, das heißt eine Zuteilung dieser zuvor definierten Klassen, ist nur dann möglich, wenn die einzelnen A-Scans anhand bestimmter Merkmale (engl.: features) unterschieden werden können. Die zweite Hauptaufgabe des Algorithmus (Abbildung 15) ist daher die Erfassung möglichst aussagekräftiger Merkmale.

Wie in Abbildung 16 ersichtlich, konnte bei der Untersuchung der Messwerte festgestellt werden, dass sowohl die Bildparameter (z.B.: Helligkeit und Kontrast), als auch die Position der einzelnen Strukturen (Lochplatte, Tablette) stark schwanken können. Zudem kommen, entsprechend der Dauer des Beschichtungsvorganges, naturgemäß unterschiedliche Schichtdicken vor. Ausgehend von diesen Beobachtungen ergeben sich bestimmte Voraussetzungen für eine sinnvolle Wahl der Merkmale.



(a) (b) (c)

Abbildung 16: Darstellung der unterschiedlich vorkommenden Bildparameter (z.B. Helligkeit und Kontrast), sowie verschiedener Positionen der Strukturen und Schichtdicken. a) helles Bild, Lochplatte unten, Beschichtungsdauer 96 min, b) dunkles Bild, Lochplatte in der Mitte, Beschichtungsdauer 70 min, c) mittlere Helligkeit, Lochplatte oben, Beschichtungsdauer 127 min

Merkmalskriterien:

- invariant bezüglich der absoluten Bildparameter, wie Helligkeit und Kontrast
- invariant bezüglich der Position der Strukturen im Bild
- invariant bezüglich der Schichtdicke

Zusätzlich zu diesen Voraussetzungen ist es notwendig, dass die extrahierten Features aussagekräftig genug sind, um eine akkurate Klassifizierung zu ermöglichen. Um die Rechenleistung gering zu halten ist es außerdem sinnvoll, nur Merkmale zu verwenden, die sich schnell und ohne großen mathematischen Aufwand berechnen lassen.

3.2 Methoden

Alle Merkmale wurden aus den Bilddaten jedes einzelnen A-Scans extrahiert. Dabei wurden sowohl der ganze Datensatz betrachtet, als auch einzelne Ausschnitte daraus mit Hilfe von Fenstern untersucht. Diese Fenster durchlaufen dabei die Bilddaten Pixel für Pixel in vertikaler Richtung (siehe Abbildung 17). Fenstergrößen von 10 Pixeln beziehungsweise 30 Pixeln erwiesen sich bei der Durchführung der Merkmalsextraktion als am geeignetsten. Zehn verschiedene Merkmale wurden hinsichtlich ihrer Relevanz für die Klassifizierung und ihrer Berechnungsdauer untersucht (siehe Tabelle 4). Klassifiziert wurden dabei 15000 A-Scans, bestehend aus den zuvor definierten Klassen *Luft*, *Metall* und *Tablette* zu jeweils 5000 Scans. Zudem wurde dieser Vorgang mit allen Merkmalen gemeinsam durchgeführt. Als Werkzeug für die Klassifizierung selbst wurde die Methode der Logistischen Regression angewandt, die in Kapitel 4 noch genauer beschrieben wird.

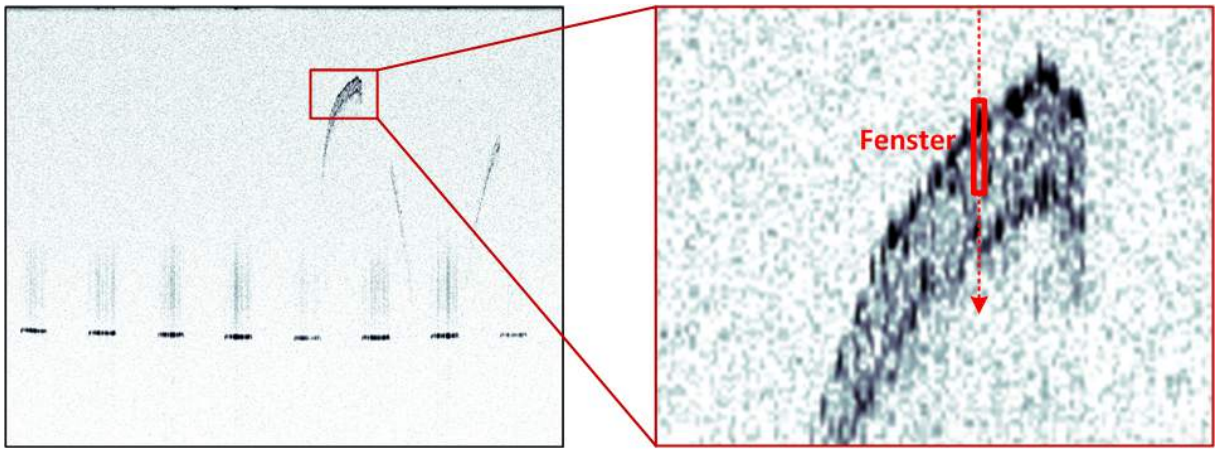


Abbildung 17: Darstellung des vertikalen Durchlaufs eines Fensters für einen A-Scan.

Tabelle 4: Auflistung der untersuchten Klassifizierungs-Merkmale

Nr.	Beschreibung
1	Summe aller Pixelwerte pro A-Scan
2	Position des ersten Fensters, an der die Summe der Fenster-Pixelwerte maximal ist
3	Position des zweiten Fensters, an der die Summe der Fenster-Pixelwerte maximal ist
4	Maximalsumme der Pixel-zu-Pixel-Differenz innerhalb des zweiten Fensters
5	Mittelwert innerhalb des ersten Fensters an Position von Merkmal Nr.2
6	Median innerhalb des ersten Fensters an Position von Merkmal Nr.2
7	Median innerhalb des zweiten Fensters an Position von Merkmal Nr.3
8	Grauwertverteilung innerhalb des ersten Fensters an Position von Merkmal Nr.2
9	Grauwertverteilung innerhalb des zweiten Fensters an Position von Merkmal Nr.3
10	Maximalwert innerhalb des ersten Fensters an Position von Merkmal Nr.2

Aus diesen zehn Merkmalen sollen diejenigen, die ohne hohen Rechenaufwand für eine Klassifizierung am aussagekräftigsten sind, ausgewählt und in einen Vektor gespeichert werden. Hierbei ist anzumerken, dass die Merkmale Nummer *8* und *9* Grauwertverteilungen sind und aus jeweils zehn einzelnen Werten bestehen. Werden die Merkmalsvektoren aneinandergereiht, ergibt sich die Merkmalsmatrix für den entsprechenden B-Scan. Abbildung 18 stellt diesen Vorgang schematisch dar.

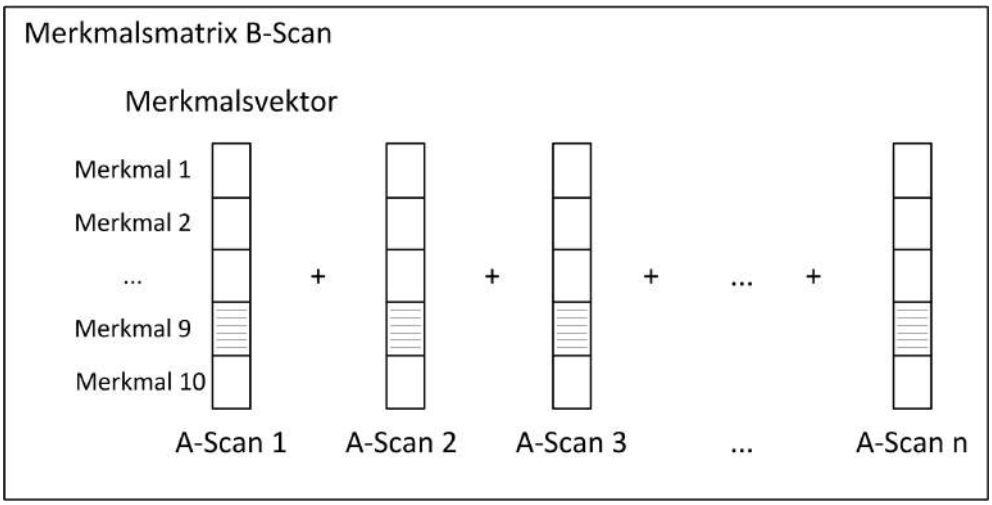


Abbildung 18: Erstellen der Merkmalsmatrix eines B-Scans, bestehend aus n A-Scans

3.3 Ergebnisse

Tabelle 5 zeigt das Ergebnis der Prüfung der Merkmale hinsichtlich der Relevanz für die Klassifizierung und der Dauer der Berechnung. Es ist zu beobachten, dass die einzelnen Merkmale durchaus unterschiedliche Wirkung auf die Klassifizierung der einzelnen Klassen besitzen. Zudem gibt es große Unterschiede im mathematischen Aufwand der Berechnung und damit der Dauer der Merkmalsextraktion.

Tabelle 5: Klassifizierungsfehler und Berechnungsdauer der einzelnen Merkmale. Der Klassifizierungsfehler gibt das Verhältnis von den falsch klassifizierten zu den gesamten A-Scans je Klasse an.

Merkmal	Klassifizierungsfehler			Dauer der Berechnung
	Metall	Tablette	Luft	
1	26,00%	33,00%	21,00%	0,382s
2	9,25%	14,60%	6,87%	0,737s
3	19,97%	24,95%	8,38%	0,721s
4	49,01%	20,53%	47,40%	7,683s
5	9,25%	14,60%	6,87%	0,721s
6	8,58%	14,59%	7,51%	1,038s
7	33,75%	24,01%	17,26%	1,025s
8	21,78%	27,32%	26,20%	2,433s
9	6,87%	26,27%	23,36%	2,503s
10	7,72%	14,76%	8,42%	0,752s

Ausgehend von diesen Ergebnissen wurden die Merkmale *2,3,6,9* und *10* als hinreichend aussagekräftig (Anteil an Falschklassifizierungen ist für mindestens eine Klasse unter 10%) erachtet. Mit einer Kombination dieser Merkmale sollen dann alle Klassen zuverlässig klassifiziert werden können.

Das Ergebnis der Klassifizierung bei Kombination der ausgewählten Merkmale wurde mit jenem bei Kombination aller angeführten Merkmale verglichen. Diese Gegenüberstellung ist in Tabelle 6 dargestellt. Dabei fällt auf, dass die Klassifizierung mit den ausgewählten Merkmalen nicht nur genauer ist, sondern auch viel schneller durchgeführt wird.

Tabelle 6: Gegenüberstellung: Klassifizierung bei Kombination alle Merkmale versus Klassifizierung bei Kombination der ausgewählten Merkmale

Merkmal	Klassifizierungsfehler			Dauer der Berechnung
	Metall	Tablette	Luft	
alle	3,81%	8,52%	6,95%	12,528s
2,3,6,9,10	3,83%	8,17%	6,66%	3,187s

Zur besseren Veranschaulichung der Wirksamkeit der Merkmale ist in Abbildung 19 eine Gegenüberstellung der Features Nummer *2* und *3* aus Tabelle 4 dargestellt.

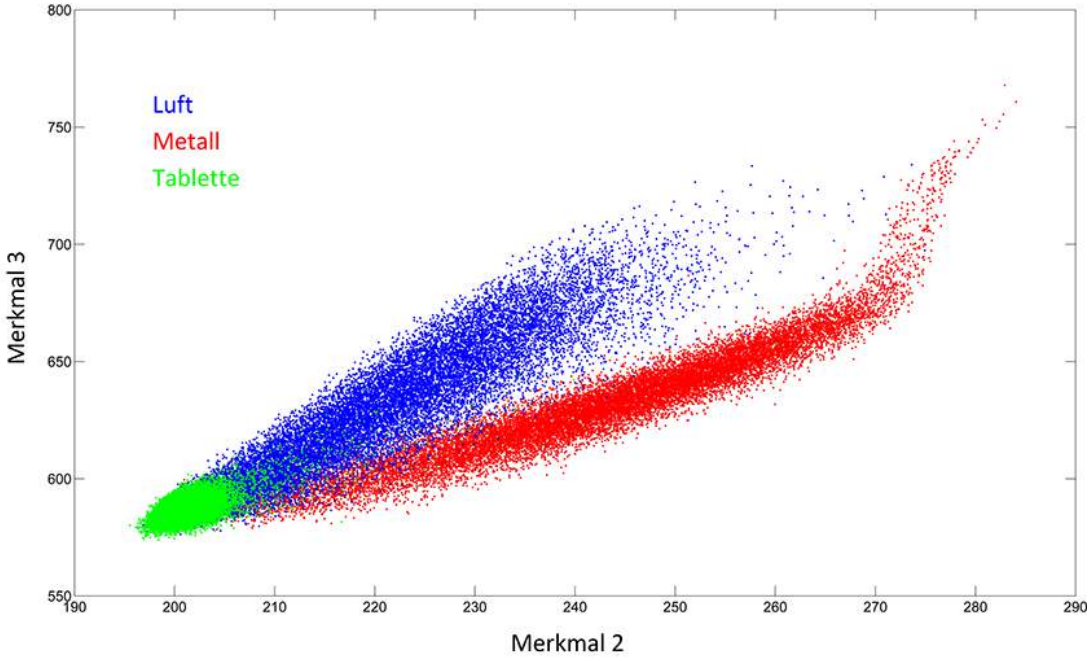


Abbildung 19: Darstellung der einzelnen Klassen durch Gegenüberstellung der Merkmale Nummer *2* und *3* aus Tabelle 4. Es ist bereits mit nur diesen zwei Merkmalen gut zu erkennen, dass sich die Klassen voneinander abzugrenzen beginnen und die Grundlage für eine erfolgreiche Klassifizierung somit gegeben ist.

4 Klassifizierung

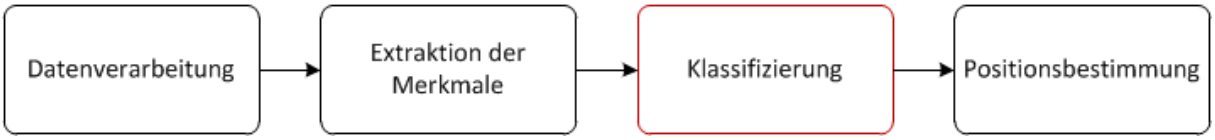


Abbildung 20: Dritter Schritt des Gesamt-Algorithmus: Klassifizierung

4.1 Theorie

Der nächste Schritt des Algorithmus ist die Einteilung der Messdaten in Klassen anhand der in Kapitel 3 ermittelten Merkmale (Abbildung 20). Grundvoraussetzung einer sogenannten überwachten Klassifizierung (engl.: supervised classification) ist das Erstellen eines Klassifizierungsmodells. Dieses Modell wird anhand von Trainingsdaten generiert, deren zugehörige Klassen im Voraus bekannt sein müssen. Im Gegensatz dazu ist dieses Vorwissen bei einer unüberwachten Klassifizierung (engl.: unsupervised classification) nicht notwendig. In dieser Arbeit wurde der Prozess der überwachten Klassifizierung realisiert, welcher aus drei Arbeitsschritten besteht (siehe Abbildung 21).

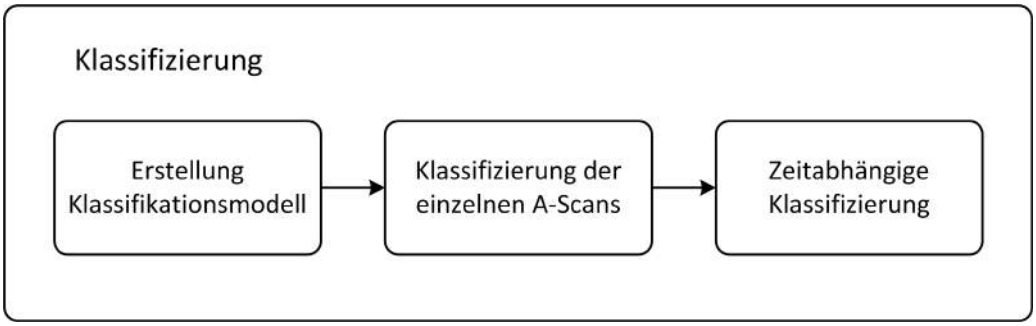


Abbildung 21: Arbeitsschritte der Klassifizierung

Erstellung des Klassifizierungsmodells Basis jeder Klassifizierung ist ein Klassifizierungsmodell, das mittels der vorhandenen Daten erstellt werden muss. Es gibt verschiedene Ansätze dieser Modellerstellung um eine Klassifizierung durchzuführen. Zwei weit verbreitete Methoden sind dabei die *Logistische Regression* und *Support Vector Machines*. Ausführliche mathematische Beschreibungen dieser Verfahren sind unter [13] zu finden.

Allgemein müssen für jede Klassifizierungsmethode folgende Unterpunkte durchgeführt werden:

- Aufteilen der vorhandenen Daten in einen Trainings- und einen Validierungsteil
- Definition der unterschiedlichen Klassen
- Erstellen des Klassifizierungsmodells mit den Trainingsdaten

Klassifizierung der einzelnen A-Scans Ist die Methode gewählt und ein entsprechendes Modell erstellt, kann der Datensatz klassifiziert werden. Die Validierung der Klassifizierungsmethode erfolgt dabei mit B-Scan Bildern aus einem Validierungsdatensatz, deren A-Scans einzeln klassifiziert werden. Wenn das Ergebnis nicht zufriedenstellend ist muss das Modell optimiert oder ein anderer Ansatz gewählt werden.

Zeitabhängige Klassifizierung Da die wiederkehrenden Strukturen sich in den hier gegebenen Daten stark ähneln, sind die Klassen der einzelnen A-Scans in gewisser Weise vorhersehbar. Es besteht beispielsweise also eine bestimmte Wahrscheinlichkeit, dass einem A-Scan der Klasse *Metall* noch einige weiter Scans der selben Klasse folgen. Um die Klassifizierung noch zu verbessern ist es daher sinnvoll, diese zeitliche Komponente in den Algorithmus einfließen zu lassen. Eine sich anbietende Möglichkeit dafür ist das *Hidden Markov Modell*.

4.1.1 Logistische Regression

Die Logistische Regression (engl.: logistic regression, LR) ist ein statistisches Analyseverfahren, dessen Ziel es ist, die Abhängigkeit der Wahrscheinlichkeit eines bestimmten Ereignisses (abhängige Variable) von einer oder mehreren unabhängigen Eingangsvariablen zu beschreiben. Es wird davon ausgegangen, dass die abhängige Variable aufgrund individueller Unterschiede in den unabhängigen Variablen nicht perfekt vorhergesagt werden kann. Diese Unsicherheit wird beim Verfahren der logistischen Regression dadurch berücksichtigt, dass hierbei die Wahrscheinlichkeit eines bestimmten Wertes der abhängigen Variable betrachtet wird.

In einem einfachen Beispiel wird also die Wahrscheinlichkeit P berechnet, dass eine abhängige Variable Y den Wert 0 beziehungsweise 1 annimmt, wenn die Eingangsvariable X den Wert x besitzt. Die Werte von Y stellen dabei die zwei definierten Klassen dieses Beispiels dar. Bei der Logistischen Regression schließen die beiden Klassen einander aus, daher ist $P(Y = 1|X = x)$ die Gegenwahrscheinlichkeit zu $P(Y = 0|X = x)$ und es reicht, eine der beiden Wahrscheinlichkeiten zu berechnen:

$$P(Y = 1|X = x) = 1 - p(Y = 0|X = x) \tag{5}$$

Würde eine lineare Regression verwendet werden, so könnte die Wahrscheinlichkeitsfunktion den Wertebereich von 0 bis 1 für bestimmte Werte von X verlassen, was theoretisch nicht möglich ist. Daher bedient man sich bei der Logistischen Regression der sogenannten logistischen Funktion. Das heißt, die Funktion ist so definiert, dass sie für Werte $X \to -\infty$ gegen Null und für Werte $X \to +\infty$ gegen Eins strebt:

$$P(Y = 1|X = x) = \frac{e^x}{1 + e^x} \tag{6}$$

Abbildung 22 zeigt den Unterschied zwischen einer linearen und einer logistischen Regressionskurve.

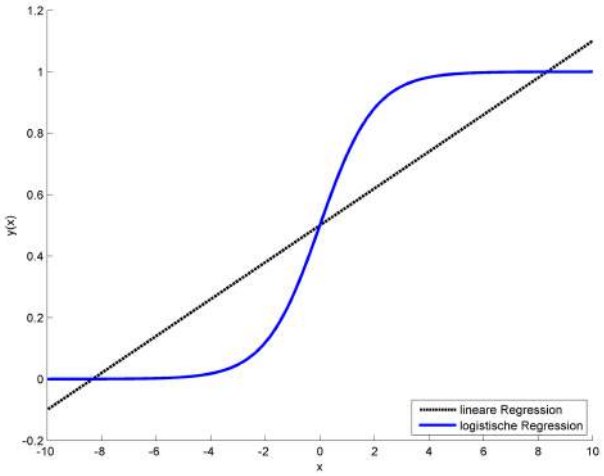


Abbildung 22: Lineare und logistische Regressionskurven. Im Gegensatz zur logistischen verlässt die lineare Regressionskurve für Werte von $x < -10$ und $x > 10$ den zulässigen Wertebereich zwischen Null und Eins.

Da die abhängige Variable (für einen A-Scan i) nicht wie in dem oben beschriebenen Beispiel von nur einer unabhängigen Variablen X abhängt, sondern von den in Kapitel 3 beschriebenen Merkmalen $X_{i,j,j=1...M}$, wird eine neue Variable z_i eingeführt:

$$z_i = b_{i,0} + b_{i,1}X_{i,1} + b_{i,2}X_{i,2} + ... + b_{i,j}X_{i,j} \tag{7}$$

Der Wert z_i wird dabei auch als *Logit*, die Gewichte $b_{i,j,j=1...M}$ als *Logit-Koeffizienten* bezeichnet. $b_{i,0}$ beeinflusst die horizontale Position der Kurve, die anderen Gewichte definieren deren Steigung.

Entsprechend Gleichung 6 erhält man demnach:

$$P_i(Y_i = 1|X = \mathbf{x}_i) = \frac{e^{z_i}}{1 + e^{z_i}} \tag{8}$$

und

$$P_i(Y_i = 0|X = \mathbf{x}_i) = 1 - \frac{e^{z_i}}{1 + e^{z_i}} \tag{9}$$

Diese beiden Gleichungen lassen sich zusammenfassen zu:

$$P_i(Y_i) = \left(\frac{e^{z_i}}{1 + e^{z_i}}\right)^{Y_i} \cdot \left(1 - \frac{e^{z_i}}{1 + e^{z_i}}\right)^{1-Y_i} \tag{10}$$

Um ein funktionierendes Klassifizierungsmodell zu erhalten, müssen nun die Gewichte $b_{i,j}$ so geschätzt werden, dass die sogenannte Likelihood (zu deutsch etwa: Plausibilität) L für alle Beobachtungen (A-Scans) i ein Maximum erreicht. Grundlage dieser Parameterschätzung ist das sogenannte *Maximum-Likelihood-Verfahren.*

$$L = \prod_{i=1}^{N} \left(\frac{e_i^z}{1 + e_i^z} \right)^{Y_i} \cdot \left(1 - \frac{e_i^z}{1 + e_i^z} \right)^{1-Y_i} \longrightarrow max! \tag{11}$$

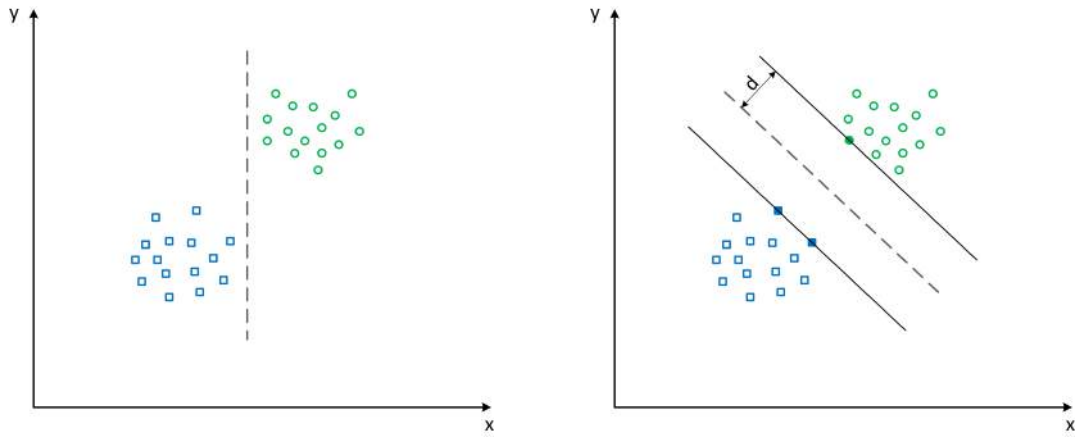
Multinomiale Logistische Regression Kann die abhängige Variable in mehr als zwei Klassen eingeteilt werden, also mehr als zwei Zustände annehmen, spricht man von einer multinomialen Logistischen Regression. Solch ein Modell wird aus mehreren binären Logistischen Regressionen aufgebaut. Für eine Problemstellung, bei der Y_i die Werte $c = 1...n$ annehmen kann, gilt:

$$P_i(Y_i = c|X = \mathbf{x}_i) = \frac{e^{\mathbf{x}_i b_c}}{1 + \sum_{s=1}^{n} e^{\mathbf{x}_i b_s}} \tag{12}$$

Dabei sind die Gewichte b_c klassenspezifisch. Beispiele für die Anwendung von multinomialer Logistischer Regression sind unter [14], [15] und [16] zu finden.

4.1.2 Support Vector Machines

Eine Support Vector Machine (SVM, Übersetzung: Stützvektormaschine, Stützvektormethode) ist ein Klassifikator. Die Grundidee dieses Verfahrens besteht darin, die Grenze zwischen den einzelnen Klassen so zu wählen, dass zwischen den Klassen ein möglichst breiter Bereich entsteht, der frei von zu klassifizierenden Objekten ist. Das SVM-Verfahren ist also ein sogenannter *Max-Margin-Classifier*, da der Abstand der Objekte, die der Klassengrenze (Hyperebene) am nächsten liegen, maximiert wird (siehe Abbildung 23). Diese Abstände werden durch Vektoren dargestellt. Eine Eigenschaft des Verfahrens ist, dass die Freiheitsgrade reduziert werden und somit die Gefahr der Überanpassung (engl.: overfitting, zu viele erklärende Variablen) vermieden wird. Die trennende Hyperebene ist nur von den ihr am nächsten liegenden Vektoren abhängig. Diese Stützvektoren reichen aus, um das Modell mathematisch exakt zu beschreiben.



(a) (b)

Abbildung 23: Graphische Darstellung des Prozesses der Klassifizierung mit Support Vector Machines. a) Willkürliche Trennung zweier Klassen (dargestellt durch die farbigen Symbole) ohne Optimierung, b) Klassentrennung mittels SVM. Der Abstand d zwischen der gezogenen Klassengrenze (dargestellt durch die strichlierte Linie) und den nächsten Klassenelementen wird maximiert. Nur diejenigen Elemente, die direkt auf der aufgespannten Hyperebene liegen (ausgefüllte Symbole), werden verwendet.

Eine Anzahl von N Trainingsdaten $\{\mathbf{x}_i, y_i\}$ ist in zwei Klassen aufgeteilt. $\mathbf{x}_i$ stellen dabei die Datenpunkte, y_i die definierten Klassen $(+1, -1)$ dar. Die Hyperebene ist definiert durch:

$$H = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \tag{13}$$

$\mathbf{w}$ ist der Normalenvektor der Klassengrenze, b die Verschiebung. Diese Parameter müssen nun so gewählt werden, dass die Hyperebene die beiden Klassen trennt.

Da aber eine Skalierung mit der Konstante a, also $(a\mathbf{w}, ab)$, die gleiche Hyperebene beschreibt wie $(\mathbf{w}, b)$, ist diese nicht eindeutig beschrieben. Es wird eine Skalierung relativ zu den Trainingsdaten vorgenommen, sodass nun kein Punkt mehr auf der ursprünglichen Hyperebene liegt:

$$\min_{i=1\ldots N} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| = 1 \tag{14}$$

Eine in dieser Weise skalierte Hyperebene wird auch *kanonische Hyperebene* genannt. Für Punkte direkt am Rand der beiden neu definierten kanonischen Ebenen erfolgt die Klassifizierung nun durch $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = +1$ für die Klasse $y = +1$ und $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = -1$ für die Klasse $y = -1$ (Abbildung 24).

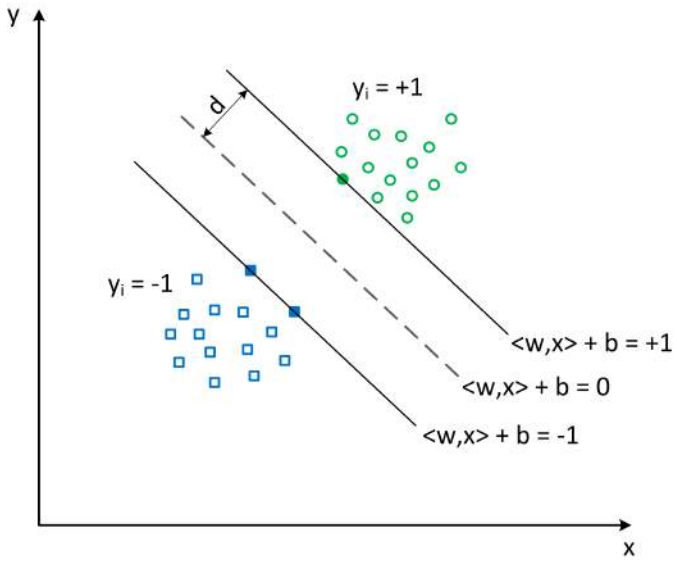


Abbildung 24: Graphische Darstellung der mathematischen Beschreibung der Klassifizierung.

Der Abstand der Hyperebene zum nächst gelegenen Klassenelement wird als Rand (engl.: margin) d bezeichnet und hat den Wert:

$$d = \frac{1}{\|\mathbf{w}\|} \tag{15}$$

Die Trennspanne zu maximieren ist demnach gleichbedeutend mit der Minimierung von $\|\mathbf{w}\|$ (was einer Minimierung von $\frac{1}{2}\|\mathbf{w}\|^2$ entspricht). Bei korrekter Klassifizierung ist der Term $(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ durch die Multiplikation mit y_i immer positiv, woraus sich folgende Nebenbedingung ableiten lässt:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \tag{16}$$

Das Optimierungsproblem kann mit Hilfe der sogenannten *Lagrange-Funktion* gelöst werden. Dazu werden die Lagrange-Multiplikatoren $\boldsymbol{\alpha}_i \geq 0$ eingeführt und die Optimierungsfunktion dargestellt als:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \boldsymbol{\alpha}_i (y_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) - 1) \tag{17}$$

$L(\mathbf{w}, b, \boldsymbol{\alpha})$ ist nun hinsichtlich $\mathbf{w}$ und b zu minimieren und hinsichtlich der $\boldsymbol{\alpha}_i$ zu maximieren, das heißt:

$$\begin{aligned} \tfrac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad &\Longrightarrow \quad \mathbf{w} = \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i \mathbf{x}_i \\ \tfrac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = 0 \quad &\Longrightarrow \quad \sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i = 0 \end{aligned}$$

Diese Ergebnisse ergeben angewandt auf Gleichung 17 das duale Problem:

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{N} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{18}$$

Durch die Lösung dieses Problems erhält man die $\boldsymbol{\alpha}_i$, die $W(\boldsymbol{\alpha})$ maximieren. Somit können der Normalenvektor $\mathbf{w}$ und die Verschiebung b berechnet werden und die Hyperebene mit der maximalen Trennspanne ist gefunden.

Die Entscheidungsfunktion f für neue Klassenelemente $\mathbf{x}_{neu}$ ist:

$$f(\mathbf{x}_{neu}) = \operatorname{sign}(\langle \mathbf{w}, \mathbf{x}_{neu} \rangle + b) = \operatorname{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i \langle \mathbf{x}_{neu}, \mathbf{x}_i \rangle + b \right) \tag{19}$$

Nur die Support Vektoren, also die Punkte, die sich am nächsten zur Hyperebene befinden, haben Einfluss auf deren Lage ($\boldsymbol{\alpha}_i \neq 0$) und tragen somit zur Klassifizierung bei. Für alle anderen Punkte i gilt $\boldsymbol{\alpha}_i = 0$.

Nichtlineare Klassifizierung Die oben angeführten Berechnungen gelten nur für die Tatsache, dass die Klassenelemente linear trennbar sind. Ist dies nicht der Fall, kann keine trennende Hyperebene konstruiert werden. Für nichtlinear trennbare Daten behilft man sich dabei mit dem sogenannten *Kernel-Trick*. Grundidee des Kernel-Tricks ist es, die Daten mit einer Funktion Φ in einen Raum höherer Dimension (auch Feature-Raum genannt) zu überführen, in welchem sie sich linear trennen lassen. Zur Veranschaulichung dieses Vorganges siehe Abbildung 25.

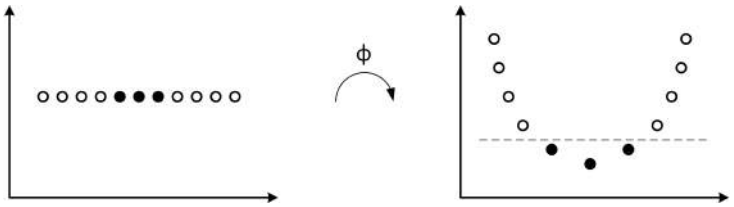


Abbildung 25: Graphische Darstellung des Kernel-Tricks. Die ursprünglichen Daten im linken Bild sind nicht linear trennbar. Durch die Transformation mit der Funktion Φ werden sie in den höher dimensionalen Feature-Raum überführt, in dem sie linear trennbar werden und eine Hyperebene erstellt werden kann (rechtes Bild).

Durch diese Transformation kann theoretisch jeder beliebige Datensatz in einen linear trennbaren umgewandelt werden, solange die Dimension hoch genug ist. Das Problem dabei ist allerdings, dass nun nicht mehr nur Skalarprodukte der Art $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ wie in Formel 19, sondern $\langle \Phi(x_i), \Phi(x_j) \rangle$ zu berechnen sind. Bei hohen Dimensionen steigt dabei die Komplexität und damit die Rechenlast sehr stark an.

Der eigentliche Trick dieser Methode besteht nun darin, für die Transformation eine Kern-Funktion (engl.: kernel function) k zu verwenden, die sich im Feature-Raum wie ein Skalarprodukt verhält.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \tag{20}$$

Die Lösung des Problems ergibt sich damit durch Ersetzen des ursprünglichen Skalarprodukts durch die Kern-Funktion:

$$f(\mathbf{x}_{neu}) = \operatorname{sign}\left(\sum_{i=1}^{N} \boldsymbol{\alpha}_i y_i k(\mathbf{x}_{neu}, \mathbf{x}_i) + b \right) \tag{21}$$

Eine Kern-Funktion k muss symmetrisch und positiv definit sein. In Verbindung mit Support Vector Machines häufig verwendete Kern-Funktionen sind:

Polynomiell vom Grad d:	$k(\mathbf{x}_i, \mathbf{x}_j) = (c + \langle \mathbf{x}_i, \mathbf{x}_j \rangle)^d$	für c konstant
Radial Basis:	$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2}$	für $\gamma > 0$
Neuronales Netzwerk:	$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta)$	für $\kappa > 0$ und $\theta \in \mathbb{R}$

Soft Margin Hyperebene Oftmals ist es so, dass ein Datensatz zwar grundsätzlich in einem Feature-Raum mit nicht allzu hoher Dimension linear trennbar wäre, wenn nicht ein paar „Ausreißer“ eine viel höhere Dimension für eine exakte Trennung erfordern würden. Es drängt sich daher die Überlegung auf, die Komplexität und den damit verbundenen Rechenaufwand auf Kosten einiger weniger falscher Klassifizierungen gering zu halten.

Die Grundidee dieser Methode ist es, die Hyperebene mit einem „weichen Rand“ (engl.: soft margin) zu versehen. Das bedeutet, dass die Randbedingungen abgeschwächt und eine bestimmte Anzahl an Missklassifizierungen zwar erlaubt, aber auch entsprechend bestraft werden. Um diesen Ansatz verwirklichen zu können, werden sogenannte Schlupfvariablen (engl.: slack variables) $\xi_i \geq 0$ eingeführt.

Diese sind so definiert, dass sich Datenpunkte mit $\xi_i = 0$ innerhalb der Klassengrenze befinden. Datenpunkte mit $\xi_i = 1$ liegen direkt auf der Entscheidungsgrenze und Punkte mit $\xi_i > 1$ wurden missklassifiziert (siehe Abbildung 26).

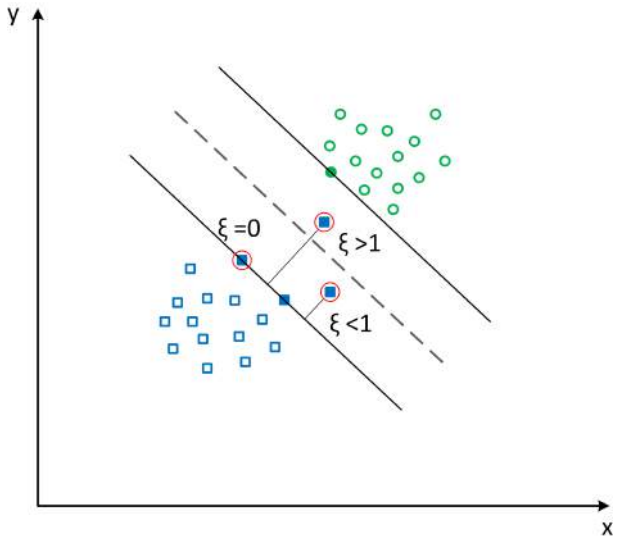


Abbildung 26: Definition der Schlupfvariablen ξ. Ausreißer, die die Trennspanne verkleinern würden (was eine geringere Generalisierung zur Folge hat), haben $0 < \xi \leq 1$, Missklassifizierungen haben $\xi > 1$

Als Strafe für die Ausreißer wird der Kostenterm $C\xi_i$ definiert, wobei $C > 0$ als Fehlergewicht interpretiert werden kann. Je weiter der Abstand des Datenpunktes zur Klassengrenze ist, desto höher ist die Strafe.

Der Kostenterm muss im Minimierungsproblem berücksichtigt werden, somit gilt:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \longrightarrow min! \tag{22}$$

unter Erfüllung der Bedingung:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \tag{23}$$

Die Konstante C stellt in den weiteren Berechnungen eine zusätzliche Beschränkung der Lagrange-Multiplikatoren $\boldsymbol{\alpha}_i$ dar, ansonsten wird analog zu oben vorgegangen. Die Schwierigkeit bei der Verwendung des Kostenterms besteht also darin, einen Mittelweg zwischen einem relativ großen Rand und mehr Ausreißern und einem kleineren Rand und weniger Ausreißern zu finden. Bestimmt wird dieser Zusammenhang von C.

Parameteroptimierung Abhängig von der Wahl der Kernfunktion und je nach dem, ob die Soft Margin Variante gewählt wurde oder nicht, sind verschiedene Parameter zu optimieren, um das bestmögliche Klassifizierungsergebnis zu erreichen. Im Falle von Soft Margin SVM mit Radial-Basis-Funktion (RBF) als Kern wären die zu optimierenden Konstanten beispielsweise das Fehlergewicht C, sowie der Parameter γ (RBF). Eine Optimierung dieser Werte erfolgt zumeist per Kreuzvalidierung oder Rastersuche (engl.: grid search).

Multiklassen SVMs Grundsätzlich sind Support Vector Machines binäre Klassifikatoren. Allerdings können mehrere solche Zwei-Klassen-Klassifikatoren zu einem Multiklassen-Klassifikator zusammengesetzt werden. Zwei häufig verwendete Ansätze sind:

- *Einer-gegen-Alle* (engl.: one-versus-all, OVA)
- *Jeder-gegen-Jeden* (engl.: one-versus-one, OVO)

Bei der **OVA-Methode** werden für M verschiedene Klassen M verschiedene binäre Klassifikatoren benötigt. Die Entscheidungsfunktion für die Klasse j sieht dabei folgendermaßen aus:

$$f_j(x) = \begin{cases} +1 & \text{wenn x zur Klasse j gehört} \\ -1 & \text{sonst.} \end{cases}$$

Ein Problem dieser Methode ist, dass für beispielsweise 10 verschiedene Klassen mit gleicher Anzahl an Datenpunkten jeder der binären Klassifikatoren für die Unterscheidung von 90% negativen und nur 10% positiven Elementen trainiert wird. Die Symmetrie des Originalproblems geht dadurch verloren.

Die **OVO-Methode** benötigt für M verschiedene Klassen $M\frac{M-1}{2}$ binäre Klassifikatoren. Die Entscheidungsfunktion für jedes Paar an Klassen (i, j) lautet dabei:

$$f_{i,j}(x) = \begin{cases} +1 & \text{wenn x zur Klasse i gehört} \\ -1 & \text{wenn x zur Klasse j gehört} \end{cases}$$

Für jede Klasse werden hier mehrere Klassifikatoren angewandt. Jede Entscheidung für eine Klasse wird dabei numerisch (beispielsweise mit 1) bewertet, die Bewertungen werden aufsummiert. Die Entscheidung fällt dann auf die Klasse mit den meisten Votings.

Ein Vergleich dieser beiden Methoden, sowie weiteren Ansätzen ist unter [17], [18] und [19] zu finden.

4.1.3 Hidden Markov Modell

Ein Hidden Markov Modell (engl.: hidden markov model, HMM) ist eine statistische Modellierung eines Systems, das zwar bestimmten Grundregeln folgt, dessen Verhalten aber nichtdeterministisch (nicht eindeutig vorhersagbar) ist. Wichtige Anwendungsbeispiele sind die Spracherkennung [20], die Erkennung von Handschriften [21] oder die Klassifizierung von Proteinsequenzen [22].

Markov Modell Ausgangspunkt ist das zu beobachtende System, das eine bestimmte Anzahl von definierten Zuständen annehmen kann. Jeder dieser Zustände tritt mit einer bestimmten Häufigkeit (Wahrscheinlichkeit) auf. Zu diskreten Zeitpunkten geht dieses System mit einer gewissen Wahrscheinlichkeit vom einen Zustand in einen anderen (oder den gleichen) über. Diese Übergangswahrscheinlichkeiten werden auf Basis von Trainingsdaten bestimmt. Sie sind daher im Vorhinein bekannt und werden in einer Matrix zusammengefasst. Ein oft zur besseren Veranschaulichung herangezogenes Beispiel ist folgendes:

- System: Wetter
- Zustände: Sonne, Wolken, Regen
- Zustandswahrscheinlichkeit: bekannt
- diskrete Zeitpunkte: Tage
- Übergangswahrscheinlichkeiten: Sonne - Sonne, Sonne - Wolken, Sonne - Regen, Wolken - Sonne, ...

Ausgehend vom aktuellen Zustand (z.B. Sonne), der Wahrscheinlichkeit der einzelnen Zustände und den bekannten Übergangswahrscheinlichkeiten kann nun ein Markov Modell erstellt werden, welches dieses System abbildet. Auf Basis dieses Markov Modells kann nun der wahrscheinlichste Wetterverlauf für die nächsten Tage berechnet werden (z.B. Sonne - Sonne - Wolken - Regen - Sonne).

Hidden Markov Modell Im Gegensatz zum Markov Modell sind die einzelnen Zustände des Systems beim Hidden Markov Modell nicht direkt beobachtbar, sondern nur deren Auswirkungen (Emission) auf die Umgebung. Das Modell muss also nun um die Wahrscheinlichkeit erweitert werden, dass die beobachtbare Emission bei einem bestimmten Zeitpunkt und zugehörigem Zustand auftritt, sowie die Häufigkeit, mit der die Emission zu beobachten ist. Ein Beispiel hierfür wäre, dass man sich in einem fensterlosen Zimmer befindet und daher die Zustände des Wetter nicht direkt beobachtbar sind. Einzigen Rückschluss auf das Wetter bietet die Beobachtung, ob der tägliche Besucher einen Hut trägt oder nicht. Das Beispiel wird erweitert um:

- Emission: Hut
- Emissionswahrscheinlichkeit: Sonne + Hut, Wolken + Hut, Regen + Hut

Zusammengefasst besteht ein Hidden Markov Modell demnach aus folgenden Größen:

$\mathbf{S} = \{s_1, s_2, ..., s_{N_s}\}$	mögliche Zustände, die das System annehmen kann
$\mathbf{V} = \{v_1, v_2, ..., v_M\}$	mögliche Beobachtungen
$\mathbf{Q}^T = \{q_1, q_2, ..., q_N\}$	Sequenz der hintereinander angenommenen Zustände, $q_n \in \mathbf{S}$
$\mathbf{X}^T = \{x_1, x_2, ..., x_N\}$	Sequenz der dazugehörenden Beobachtungen, $x_n \in \mathbf{V}$
$\mathbf{A} \in \mathbb{R}^{N_s x N_s}$	Übergangsw'keiten a_{ij} von einem Zustand in den anderen
$\mathbf{B} \in \mathbb{R}^{NxM}$	Emmissionsw'keiten b_{ij} der Beobachtung v_j in Zustand $q_n = s_i$
$\boldsymbol{\pi} \in \mathbb{R}^N$	Anfangswertverteilung: W'keit von s_i als Startzustand

Die Wahrscheinlichkeit eines HMM mit den Parametern $\{\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}\} = \boldsymbol{\Theta}$, dass N Zustände in der Reihenfolge $\mathbf{Q}^T = \{q_1, q_2, ..., q_N\}$ auftreten, berechnet sich als Produkt der Übergangswahrscheinlichkeiten zwischen den Zuständen:

$$P(\mathbf{Q}|\boldsymbol{\Theta}) = \pi_{q_1} \prod_{n=1}^{N-1} a_{q_n, q_{n+1}} = \pi_{q_1} \cdot a_{q_1,q_2} \cdot a_{q_2,q_3} \cdot ... \cdot a_{q_{N-1},q_N} \tag{24}$$

Bei oben gegebener Zustandssequenz $\mathbf{Q}^T$ ist die Wahrscheinlichkeit der entsprechenden Beobachtungssequenz $\mathbf{X}^T = \{x_1, x_2, ..., x_N\}$ (gleicher Länge):

$$P(\mathbf{X}|\mathbf{Q}, \boldsymbol{\Theta}) = \prod_{n=1}^{N} P(x_n|q_n, \boldsymbol{\Theta}) = b_{q_1,x_1} \cdot b_{q_2,x_2} \cdot ... \cdot b_{q_N,x_N} \tag{25}$$

Die Wahrscheinlichkeit, dass eine bestimmte Sequenz $\mathbf{X}^T$ bei einer Zustandssequenz $\mathbf{Q}^T$ beobachtet wird, also die gemeinsame Wahrscheinlichkeit von $\mathbf{Q}^T$ und $\mathbf{X}^T$, ist gegeben durch:

$$P(\mathbf{X}, \mathbf{Q}|\boldsymbol{\Theta}) = P(\mathbf{X}|\mathbf{Q}, \boldsymbol{\Theta}) \cdot P(\mathbf{Q}|\boldsymbol{\Theta}) \tag{26}$$

Um nun zu eruieren, mit welcher Wahrscheinlichkeit ein Hidden Markov Modell mit den Parametern $\boldsymbol{\Theta}$ eine bestimmte Beobachtungssequenz $\mathbf{V}^T$ erzeugt, könnte man nun die Summe aller möglichen Wahrscheinlichkeiten dieser Sequenz entlang sämtlicher zulässiger Pfade berechnen:

$$P(\mathbf{X}|\boldsymbol{\Theta}) = \sum_{\mathbf{Q}} P(\mathbf{X}, \mathbf{Q}|\boldsymbol{\Theta}) \tag{27}$$

In der Praxis stellt diese Aufgabe aufgrund des extremen Rechenaufwandes allerdings in den meisten Fällen ein zu rechenintensives Problem dar. Daher muss auf alternative Lösungswege zurückgegriffen werden. Ein möglicher Ansatz dafür ist der sogenannte *Viterbi-Algorithmus*.

Viterbi-Algorithmus Dieser Algorithmus bestimmt den wahrscheinlichsten Pfad in einem Hidden Markov Modell zu einem Zustand an einem bestimmten Zeitpunkt hin. Er arbeitet rekursiv und berücksichtigt eine definierte Anzahl an vorangegangenen Zuständen. Abbildung 27 stellt den Vorgang graphisch dar.

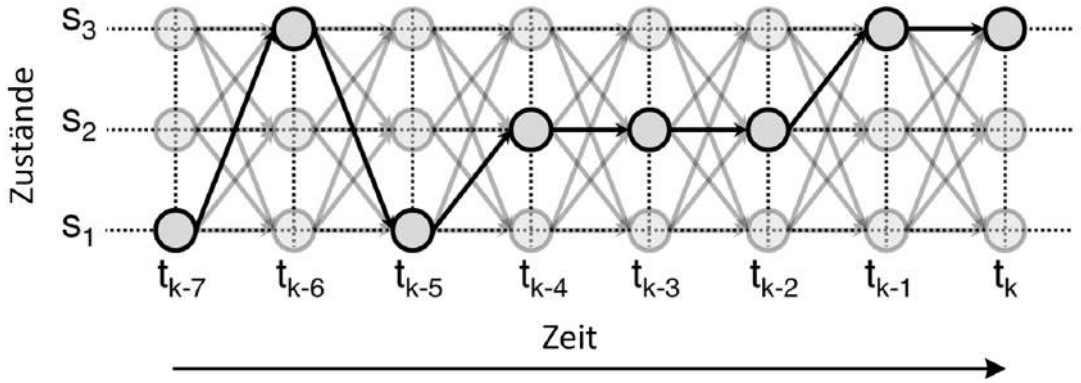


Abbildung 27: Darstellung des wahrscheinlichsten Verlaufes der Zustände eines Systems. Diese Zustandssequenz wird aufgrund eines Hidden Markov Modells berechnet

Die Grundidee dieses Lösungsansatzes ist es, den wahrscheinlichsten Pfad vom Start- zum Endzustand Schritt für Schritt für alle Zwischenzustände zu bestimmen. Wenn sich das System zum Zeitpunkt t in Zustand i befindet, so ist die größte Wahrscheinlichkeit $\delta_i(t)$ eines einzelnen unter allen möglichen Pfaden genau die Sequenz V^T erzeugt zu haben:

$$\delta_i(t) = \max\left(P(q_1, q_2, ..., q_n = s_i, x_1, x_2, ..., x_n|\mathbf{\Theta})\right) \tag{28}$$

Für jeden Schritt wird also der Teilpfad mit der höchsten Wahrscheinlichkeit gesucht. Die Sequenz der Zustände wird in einer Variablen $\psi_i(t)$ gespeichert. Die Komplexität des Algorithmus steigt dabei linear mit der Länge der Sequenz an.

Weitere Informationen zu Hidden Markov Modellen sind beispielsweise unter [13] zu finden.

4.1.4 Hybridmodelle

Um zeitlich veränderliche Daten effektiver klassifizieren zu können, werden Verfahren wie Logistische Regression oder Support Vector Machines gerne mit der Modellbildung durch Hidden Markov Modelle verknüpft ([23], [24], [25], [26], [27]).

Bezogen auf diese Arbeit bedeutet dies im Speziellen, dass die Ausgangsgrößen von Logistischer Regression oder Support Vector Machines, also die Wahrscheinlichkeiten eines A-Scans zu den jeweiligen Klassen zu gehören, einem Hidden Markov Modell als Eingangsgrößen (Emissionswahrscheinlichkeiten) dienen.

Das Verfahren der Logistischen Regression liefert diese Wahrscheinlichkeiten direkt, bei Verwendung der SVM-Methode müssen die Werte erst berechnet werden. Eine Abschätzung der Wahrscheinlichkeiten kann mittels einer Sigmoid-Funktion durchgeführt werden [23]:

$$P_i(Y_i = +1|X = \mathbf{x}_i) = \frac{1}{1 + e^{u_1 h(\mathbf{x}_i) + u_2}} \tag{29}$$

$P_i(Y_i = +1|X = \mathbf{x}_i)$ ist dabei die Wahrscheinlichkeit des Elements X (A-Scan) zur Klasse $Y_i = +1$ zu gehören und $h(\mathbf{x}_i)$ der Abstand des Datenpunktes zur trennenden Hyperebenen. Die Parameter u_1 und u_2 werden mittels Maximum-Likelihood-Methode aus einem Trainingsdatensatz ermittelt. Dazu kann der gleiche Trainingsdatensatz verwendet werden, der zuvor schon für die Erstellung des SVM Modells herangezogen wurde.

4.2 Methoden

4.2.1 Datenaufteilung

Dieser Schritt ist notwendig, um eine zuverlässige Klassifizierung zu erhalten und overfitting zu vermeiden. Wird das Modell mit den selben Daten validiert, mit Hilfe derer es auch erstellt worden ist, kann nicht sichergestellt werden, dass die Klassifizierung auch für beliebige andere Daten funktioniert. Tabelle 7 zeigt, wie die zur Verfügung stehenden 207000 A-Scans aufgeteilt wurden:

Tabelle 7: Aufteilung der Messdaten in einen Trainings- und einen Validierungsteil

Beschichtungsdauer Minuten	Geschwindigkeit m/s	A-Scans Training -	A-Scans Validierung -
10	0.4	12000	12000
	0.5	10000	10000
37	0.4	11000	10000
	0.5	9000	9000
70	0.4	13000	13000
	0.5	8000	8000
96	0.4	8000	8000
	0.5	7000	7000
127	0.4	14000	13000
	0.5	12000	12000

4.2.2 Klasseneinteilung

Folgende drei Klassen wurden für die Klassifizierung der einzelnen A-Scans definiert:

- Tablette
- Metall
- Luft

Ausgehend von dieser Klasseneinteilung und oben definiertem Trainingsdatensatz ist es nun möglich, die Übergangswahrscheinlichkeiten von einer Klasse in die nächste zu bestimmen.

4.2.3 Funktionen

`logregPredict.m` Mit Hilfe dieser Funktion kann eine Klassifizierung durch Logistische Regression implementiert werden. Das Klassifizierungsmodell muss davor mit einem Trainingsdatensatz und der Funktion **`logregFit.m`** erstellt werden. Für Code und Dokumentation dieser Funktionen siehe [28].

`svmpredict.m` Diese vorgefertigte Funktion führt eine Klassifizierung mittels Support Vector Machines durch. Die Modellerstellung erfolgt mit der Funktion **`svmtrain.m`**. Der Code sowie eine genaue Beschreibung der Funktionseigenschaften und -parameter sind unter [29] zu finden.

`getPath.m` Innerhalb des Codes dieser Funktion wird die Unterfunktion **`viterbi_path.m`** aufgerufen, welche mit Hilfe der Klassifizierungsergebnisse den wahrscheinlichsten Verlauf der Klassen innerhalb eines B-Scans berechnet.

4.2.4 Parameteroptimierung

Für die Soft Margin SVM Klassifizierung mit RBF Kernfunktion wurden die Parameter C (Fehlergewicht) und γ (Konstante der RBF) mit Hilfe einer Rastersuche optimiert. Als Startwerte wurden $C = 1$ und $\gamma = 0.01$ gewählt und schrittweise verändert. Für jedes Wertepaar wurde die Genauigkeit der Klassifizierung eruiert. Das beste Ergebnis konnte mit folgenden Werten erzielt werden:

$$C = 0.8$$
$$\gamma = 0.003$$

4.3 Ergebnisse

In diesem Kapitel sind die Ergebnisse der Klassifizierung mit den unterschiedlichen Methoden angeführt. Um einen besseren Vergleich schaffen zu können, ist in den Abbildungen immer derselbe B-Scan dargestellt. Zu beachten ist dabei, dass rot eingefärbte A-Scans das Klassifizierungsergebnis *Metall*, grün eingefärbte die Klasse *Tablette* und blau eingefärbte die Klasse *Luft* darstellen.

Klassifizierungsgenauigkeit Die Genauigkeit der Klassifizierung bezieht sich in den nachfolgenden Daten nur auf die Klasse *Tablette*, da speziell diese für die Effektivität des Algorithmus aussagekräftig ist. Dieser Wert stellt das Verhältnis von den richtig als *Tablette* klassifizierten A-Scans und der Gesamtanzahl der A-Scans, die als *Tablette* definiert wurden, dar. Die Klassifizierungsgenauigkeit wird dabei jeweils für den Trainings-, beziehungsweise den Validierungsdatensatz getrennt berechnet.

4.3.1 Logistische Regression

In Tabelle 8 ist das Ergebnis der Klassifizierung mittels Logistischer Regression samt verwendeter Anzahl an A-Scans pro Klasse für Training und Validierung, sowie der benötigten Rechendauer aufgelistet. Abbildung 28 ist die graphische Darstellung der erreichten Klassifizierungsgenauigkeit (Verhältnis von richtig klassifizierten A-Scans zur Gesamtanzahl der A-Scans).

Tabelle 8: Ergebnis der Klassifizierung mit Logistischer Regression

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	91.35	91.83	0.90
3x7500	91.75	91.90	1.29
3x10000	91.82	92.03	1.67
3x12500	92.10	92.16	2.07
3x15000	92.38	92.03	2.37
3x17500	92.28	91.97	2.88
3x20000	92.12	91.98	3.31

4.3.2 Support Vector Machines

In Tabelle 9 und Tabelle 10 ist das Ergebnis der Klassifizierung mit Support Vector Machines mit linearem und mit RBF Kernel aufgelistet. Abbildung 29 ist wiederum die graphische Darstellung der erreichten Klassifizierungsgenauigkeit.

Tabelle 9: Ergebnis der Klassifizierung mit Support Vector Machines mit linearem Kernel

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	91.27	91.81	19.6
3x7500	91.79	91.93	42.9
3x10000	91.88	91.98	77.5
3x12500	92.05	92.12	128
3x15000	92.30	92.07	184
3x17500	92.26	91.98	262
3x20000	92.08	91.98	364

Tabelle 10: Ergebnis der Klassifizierung mit Support Vector Machines mit RBF Kernel

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	94.59	91.58	40.5
3x7500	94.63	92.03	95.4
3x10000	94.40	92.28	165
3x12500	94.52	92.59	262
3x15000	94.66	92.67	377
3x17500	94.51	92.66	513
3x20000	94.29	92.61	675

4.3.3 Hybrid Modelle

In Tabelle 11 und Tabelle 12 ist das Ergebnis der Klassifizierung mit Logistischer Regression beziehungsweise Support Vector Machines (RBF Kernel) in Verbindung mit einem Hidden Markov Modell aufgelistet. Abbildung 30 und Abbildung 31 erlauben erneut die graphische Interpretation der erreichten Klassifizierungsgenauigkeit.

Tabelle 11: Ergebnis der Klassifizierung mit Logistischer Regression und Hidden Markov Modell

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	94.43	95.06	4.757
3x7500	94.83	95.15	6.237
3x10000	94.56	95.16	7.819
3x12500	94.97	95.17	9.240
3x15000	95.17	95.17	10.79
3x17500	95.15	95.18	12.38
3x20000	95.08	95.18	14.06

Tabelle 12: Ergebnis der Klassifizierung mit Support Vector Machines (RBF Kernel) und Hidden Markov Modell

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	97.81	95.57	114
3x7500	97.83	95.48	337
3x10000	97.36	95.50	624
3x12500	97.40	95.55	1046
3x15000	97.42	95.49	1569
3x17500	—	—	—
3x20000	—	—	—

Abschließend ist in Tabelle 13 und Abbildung 32 das beste Ergebnis, erzielt mit Support Vector Machines mit RBF Kernel und optimierten Parameter und Hidden Markov Modell, zu sehen.

Tabelle 13: Ergebnis der Klassifizierung mit Support Vector Machines (RBF Kernel, optimierte Parameter) und Hidden Markov Modell

A-Scans -	Genauigkeit Training %	Genauigkeit Validierung %	Rechendauer s
3x5000	95.55	95.85	31.43
3x7500	95.91	95.84	64.18
3x10000	95.44	95.85	114
3x12500	95.75	95.84	183
3x15000	95.84	95.77	276
3x17500	95.85	95.80	395
3x20000	95.81	95.83	534

4.3.4 Abbildungen

In den nachfolgenden Abbildungen sind rot eingefärbte A-Scans als Klassifizierungsergebnis *Metall*, grün eingefärbte als Klasse *Tablette* und blau eingefärbte als Klasse *Luft* dargestellt.

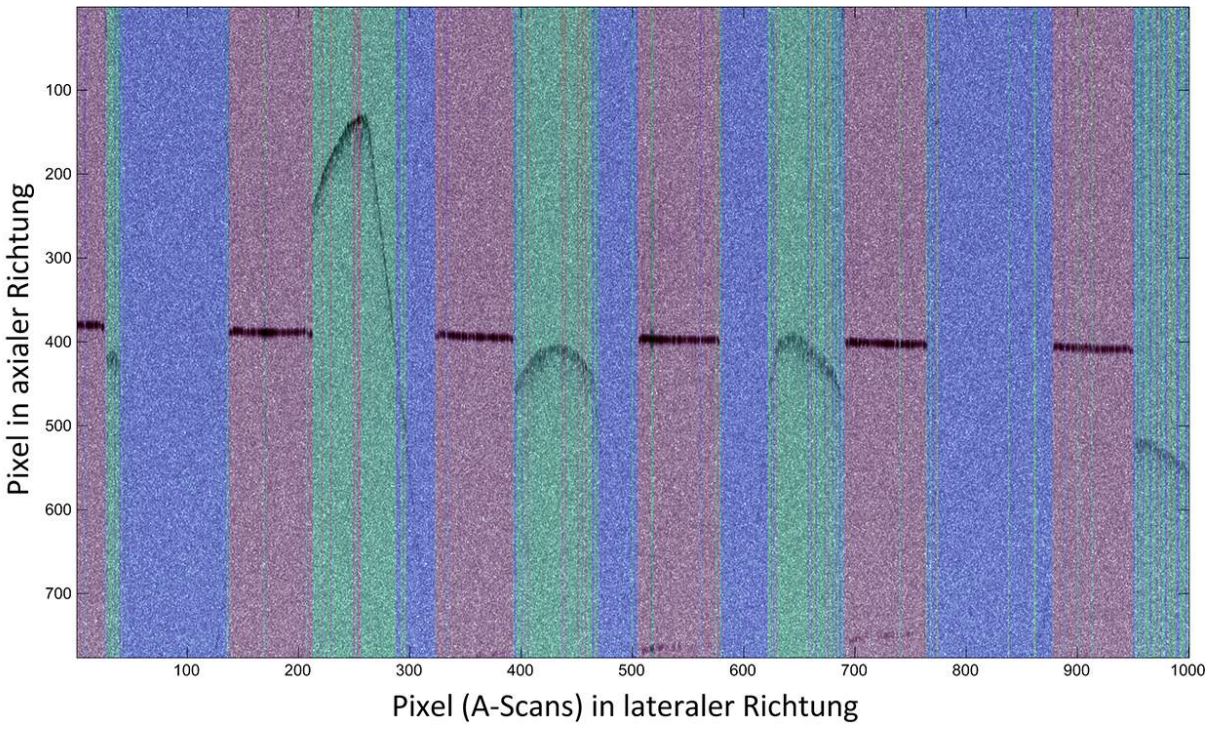


Abbildung 28: Klassifizierung mit Logistischer Regression.

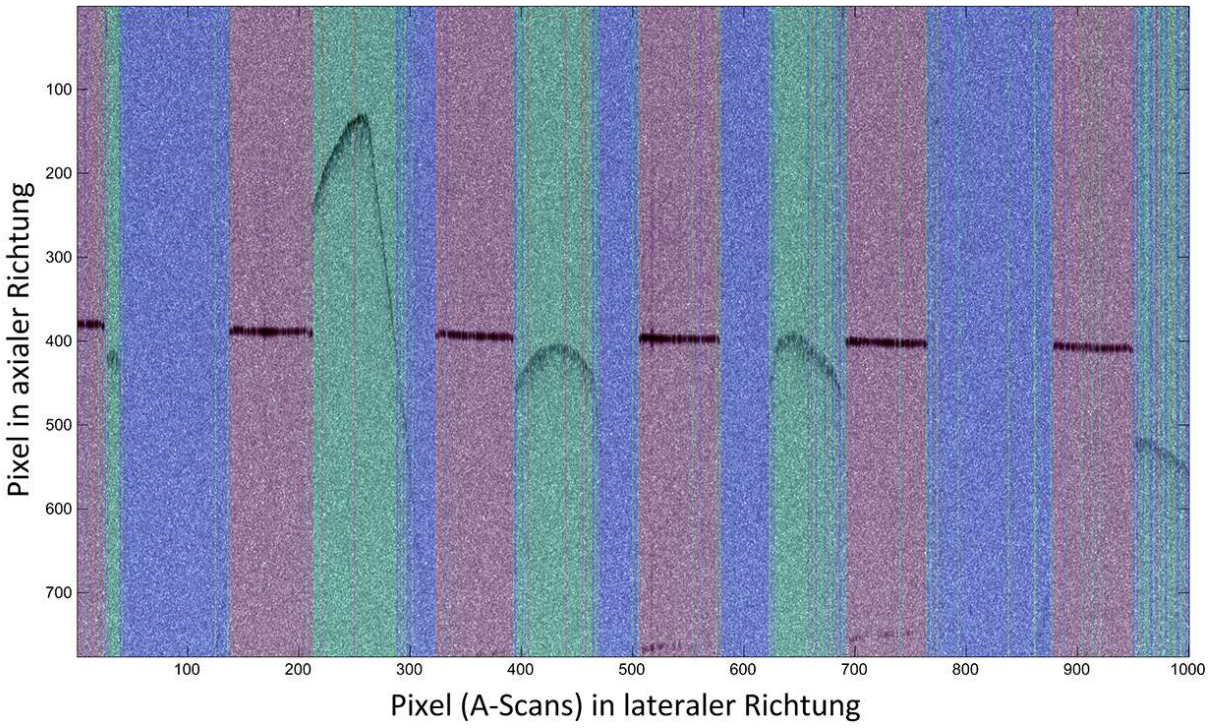


Abbildung 29: Klassifizierung mit Support Vector Machines (RBF Kernel)

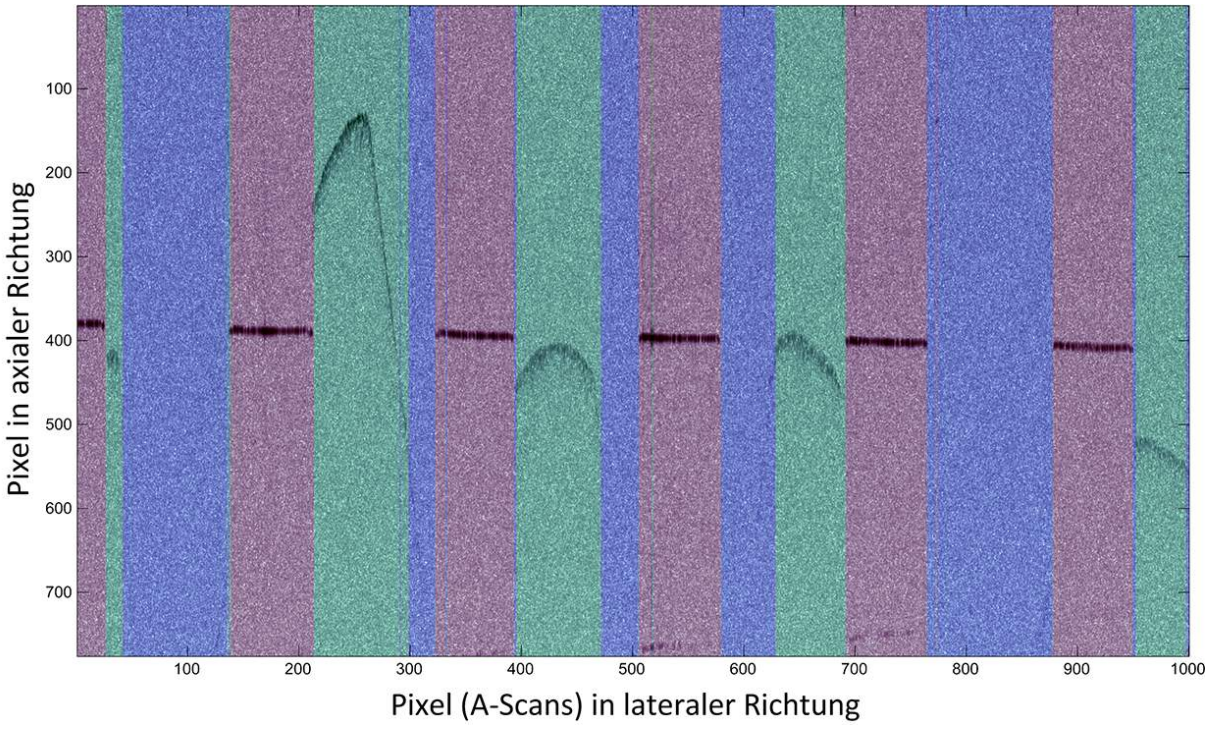


Abbildung 30: Klassifizierung mit Logistischer Regression und Hidden Markov Modell

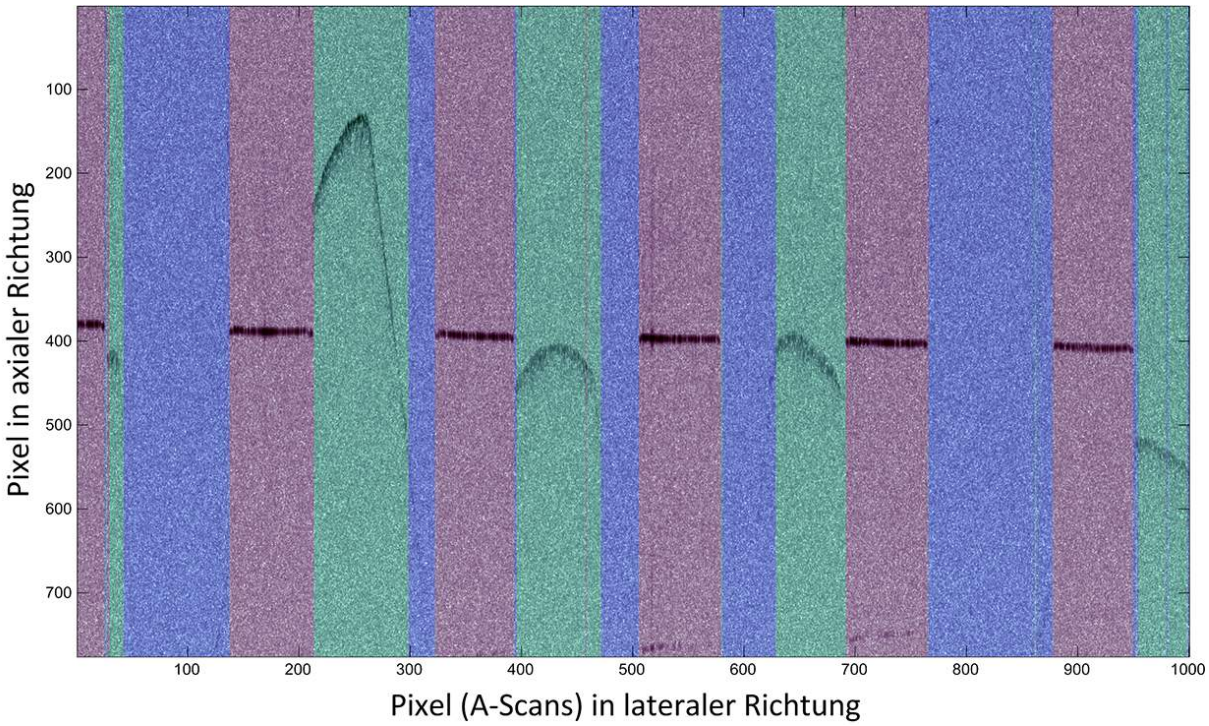


Abbildung 31: Klassifizierung mit Support Vector Machines (RBF Kernel) und Hidden Markov Modell

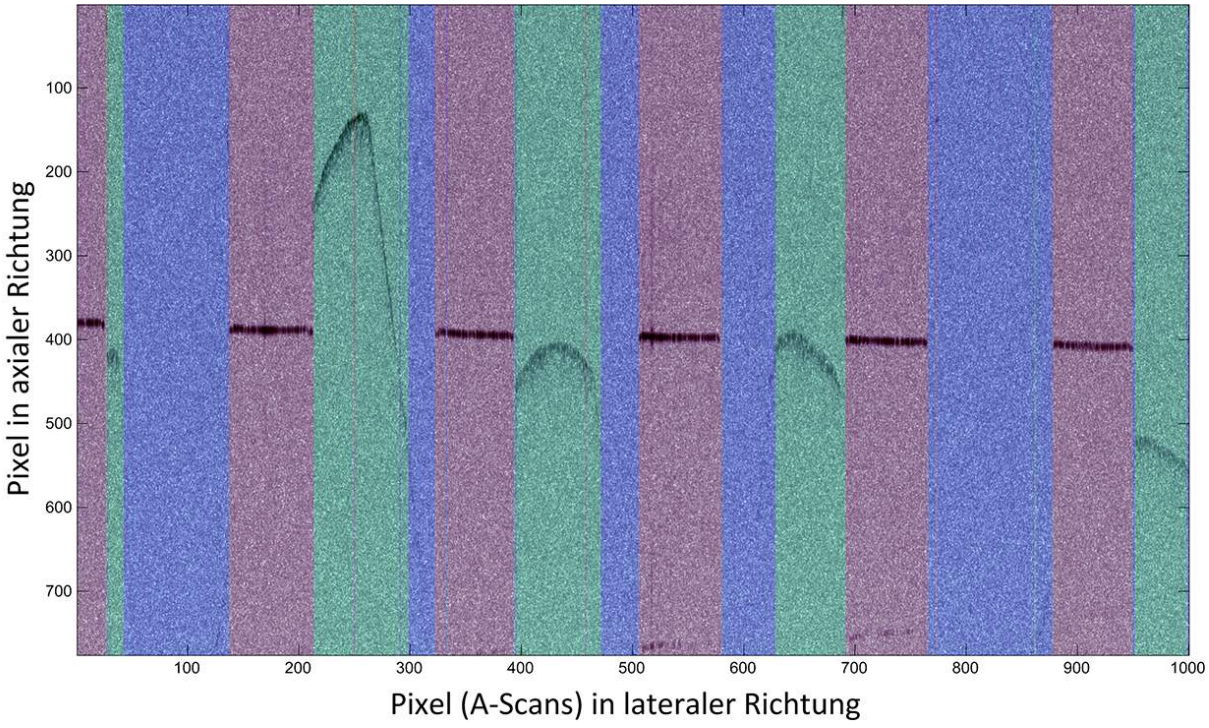


Abbildung 32: Klassifizierung mit Support Vector Machines (RBF Kernel, optimierte Parameter) und Hidden Markov Modell

4.3.5 Zusammenfassung

Wie zu erwarten war eine Klassifizierung mittels Logistischer Regression mit geringerer Rechenzeit durchzuführen, als eine Klassifizierung mit Support Vector Machines. Auch die Verwendung einer Kernfunktion resultierte durch die höhere Komplexität des Algorithmus wie erwartet in einer längeren Berechnungsdauer, wobei keine Verbesserung des Ergebnisses erzielt werden konnte.

Ein deutlicher Anstieg der Klassifizierungsgenauigkeit von etwa 4% wurde allerdings durch die Einbringung der zeitlichen Komponente mit Hilfe des Hidden Markov Modells erreicht. Diese Verbesserung ist auch in den Abbildungen leicht ersichtlich. Man erkennt deutlich, dass die Fehlklassifizierungen innerhalb der einzelnen Bereiche (*Metall*, *Tablette*, *Luft*) stark abgenommen haben. Das bessere Ergebnis der beiden Hybrid Modelle erzielte hierbei zwar Hidden Markov in Verbindung mit Support Vector Machines, allerdings zeigt ein Vergleich der Rechenzeiten, dass dieser leichte Vorteil betreffend der Klassifizierungsgenauigkeit nur mit großem rechnerischen Aufwand möglich ist.

Mit einer zuvor durchgeführten Optimierung der Parameter (C, γ) kann dieser Rechenaufwand aber wieder deutlich reduziert werden. Zudem wurde bei dieser Methode der absolut beste Wert des Klassifizierungsergebnisses, eine Genauigkeit von 95.85%, erzielt.

Ausgehend von diesen Erkenntnissen wurde für die weitere Arbeit als Klassifizierungsmethode eine Kombination aus Soft Margin Support Vector Machines mit RBF-Kern und optimierten Parametern, sowie nachfolgendem Hidden Markov Modell gewählt. Hierbei wurde das Verhältnis aus (Rechen-)Aufwand und Nutzen als am besten erachtet.

5 Positionsbestimmung

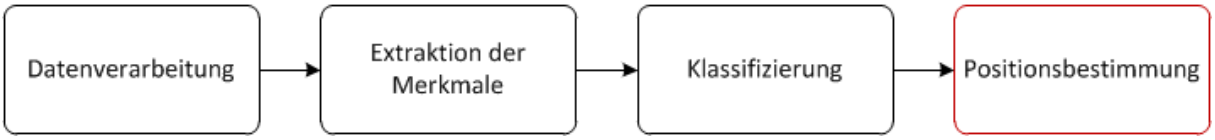


Abbildung 33: Vierter und letzter Schritt des Gesamt-Algorithmus: Positionsbestimmung

5.1 Theorie

Die Bestimmung der Position der Tablette innerhalb des Bildes ist der letzte Schritt des in dieser Arbeit erstellten Algorithmus (Abbildung 33) und die Grundlage für nachfolgende Untersuchungen, wie etwa die Messung der Dicke oder der Gleichförmigkeit der Beschichtung. Dieser Vorgang kann, wie in Abbildung 34 zu sehen, in drei Teilschritte unterteilt werden: die Detektion der Tablettenoberfläche, eine Schätzung des Mittelpunktes des einzupassenden Kreises und schließlich die Kreisanpassung selbst.

Jene Daten, die wie in Kapitel 4 beschrieben als *Tablette* erkannt wurden, werden nun für die Positionsbestimmung herangezogen. Dazu wird aus den Bilddaten die Struktur der Tablettenoberfläche mit einem einfachen generischen Algorithmus detektiert.

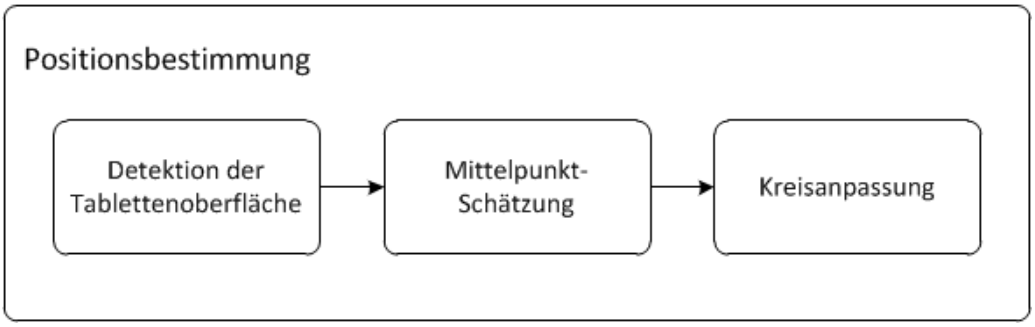


Abbildung 34: Teilschritte der Positionsbestimmung

Die Anpassung eines Kreises (engl.: circle fitting) an vorhandene Datenpunkte ist ein Problem, das in der Wissenschaft häufig auftritt (siehe zum Beispiel [30], [31], [32]). Grundsätzlich können die Lösungsansätze in zwei Gruppen unterteilt werden, die geometrischen und die arithmetischen Verfahren. Geometrische Verfahren arbeiten iterativ und minimieren eine definierte Kostenfunktion (zum Beispiel die Abstände der Datenpunkte zum Kreisumfang). Arithmetische Verfahren hingegen berechnen (ebenfalls ausgehend von einer Kostenfunktion) die Kreisparameter direkt. Im Allgemeinen gelten geometrische Circle-Fitting-Verfahren als genauer, aber rechenaufwändiger. Zudem besteht bei ihnen die Gefahr der Divergenz. Als geometrische Lösungsansätze können beispielsweise die Newton-Raphson-Methode [33], die Gauß-Newton-Methode [34] oder der Levenberg-Marquardt-Algorithmus [35] verwendet werden. Arithmetische Verfahren wurden unter anderem von Kåsa [36], Pratt [37] oder Taubin [38] entwickelt.

Wichtige Grundlagen des Circle-Fittings im Allgemeinen sind der Maximum-Likelihood-Ansatz und die Least-Square-Error-Methode ([39], [40]).

Maximum Likelihood Ansatz Bei diesem Verfahren werden Mittelpunkt und Radius desjenigen Kreises ermittelt, bei dem die Wahrscheinlichkeit, dass die Datenpunkte zu diesem gehören, ein Maximum darstellt.

Least Square Error Methode Dieses Verfahren berechnet Mittelpunkt und Radius des Kreises, bei dem die Abweichung der Datenpunkte zu den modellierten Kreispunkten im quadratischen Mittel minimal wird.

5.2 Methoden

5.2.1 Detektion der Tablettenoberfläche

Die Oberfläche der Strukturen innerhalb der Bilddaten des B-Scans wird anhand der Fensterposition, die aus der Extraktion der Merkmale (Kapitel 3) hervorgeht, bestimmt. Wie in Abbildung 35 schön zu sehen ist, werden dadurch sowohl die Datensegmente des Lochblechs, als auch die der Tabletten deutlich charakterisiert.

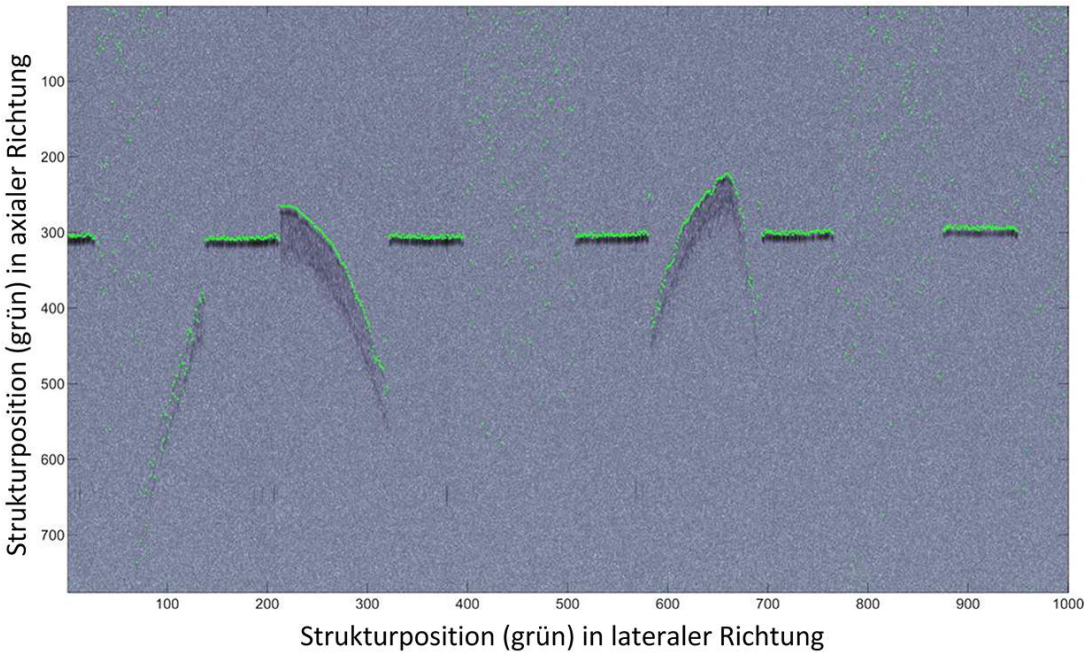
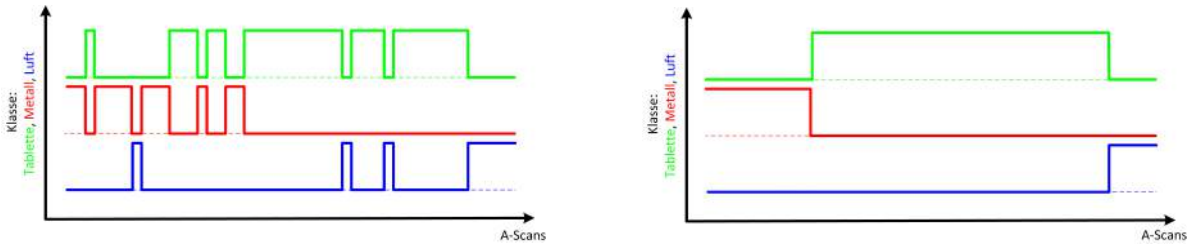


Abbildung 35: Detektion der Strukturen innerhalb des Bildes. Während bei den Klassen *Metall* und *Tablette* die Oberfläche schön getroffen wird, ist bei der Klasse *Luft* wie erwartet nur Rauschen zu erkennen.

Optimierung der Daten Da die vorhandenen Scans jeweils nur einen kleinen Teil der Tabletten erfassen, steht für das Fitting-Verfahren pro Tablette nur eine kleine Menge an Datenpunkten zur Verfügung. Daraus resultiert, dass sich diese Daten auf einem entsprechend kleinen Kreissektor befinden. Da sich das Einpassen des Kreises mit Abnahme der Sektorgröße erschwert [**?**], wurden für die weiteren Berechnungen nur Datenpakete verwendet, die aus mindestens 50 zusammenhängenden A-Scans bestehen.

Es kann trotz Aufbereitung und Verbesserung des Klassifizierungsergebnisses mittels Hidden Markov Modell vorkommen, dass einzelne A-Scans in einem Paket der Klasse *Tablette* eine offensichtlich falsche Klasse aufweisen. Daher werden die Klassenvektoren vor der Detektion der Oberfläche optimiert, sodass das entsprechende Datenpaket aufgrund der zu geringen Datenmenge nicht im Vorhinein schon als unbrauchbar erachtet wird. Dieser Vorgang ist in Abbildung 36 graphisch dargestellt. „Löcher“ innerhalb der Klassenpakete mit der Länge von bis zu zwei A-Scans werden dabei korrigiert, da es sich dabei mit hoher Wahrscheinlichkeit um falsch klassifizierte Zustände handelt.



(a) (b)

Abbildung 36: Graphische Darstellung der Optimierung der Klassenvektoren. Das obere Level der Kurven bedeutet, dass diese Klasse dem A-Scan zugeordnet wurde. a) Klassenvektoren vor der Optimierung, b) Klassenvektoren nach der Optimierung. Einzelne Ausreißer mit der Länge von 1 oder 2 A-Scans werden unterdrückt.

Innerhalb der nun verwendeten Oberflächendaten wurde ebenfalls eine Optimierung vorgenommen, um auch hier etwaige Ausreißer zu eliminieren. Im Zuge dessen wurde außerdem ein Algorithmus verwendet, der den eventuell am Bild vorhandenen Steg der Tablette detektiert. Dieser war bereits vorhanden und musste nur mehr in das Programm eingebunden werden. Die Bestimmung der Position des Tablettensteges erfolgt dabei über eine Untersuchung des Steigungsverlaufes zwischen den Datenpunkten der Tablettenoberfläche.

Datentransformation Abhängig von der Relativgeschindigkeit zwischen Sensor und Messobjekt variiert die laterale Ausdehnung, also die Breite der A-Scans. Bei gleichbleibender Anzahl von A-Scans, ändert sich also der abgedeckte Bildbereich des B-Scans. Die Bildhöhe der Scans hängt hingegen nur vom verwendeten Messsystem ab.

Entsprechend der Geschwindigkeit erscheint die Tablettenoberfläche nicht kreisförmig, sondern mehr oder weniger gestaucht. Für das Circle-Fitting ist es also notwendig, die elliptische Form der Daten auf die eigentliche Kreisform zu transformieren. Um dies durchführen zu können, ist es zuvor notwendig die Relativgeschwindigkeit aus den Bilddaten zu

ermitteln. Mit dieser Information, sowie der bekannten Geometrie des Lochblechs und der benötigten Dauer für einen A-Scan (siehe Kapitel 1, Tabelle 1), kann die Transformation der Daten erfolgen.

Bestimmung der Geschwindigkeit Um die Relativgeschwindigkeit zwischen Sensor und Messobjekten zu ermitteln, wurden mit Hilfe des Klassenvektors die Übergänge des Zustandes *Nicht-Metall*, also *Luft* oder *Tablette*, auf den Zustand *Metall* gesucht. Der mittlere Abstand dieser Zustandsübergänge (Transitionen) entspricht dem Lochabstand des Lochblechs. Um Fehlern vorzubeugen, wurden Transitionen mit einem Abstand zum vorhergegangenen Zustandsübergang *Nicht-Metall* auf *Metall* von weniger als 100 A-Scans ignoriert. Abbildung 37 zeigt ein Beispiel der so gefundenen Übergänge.

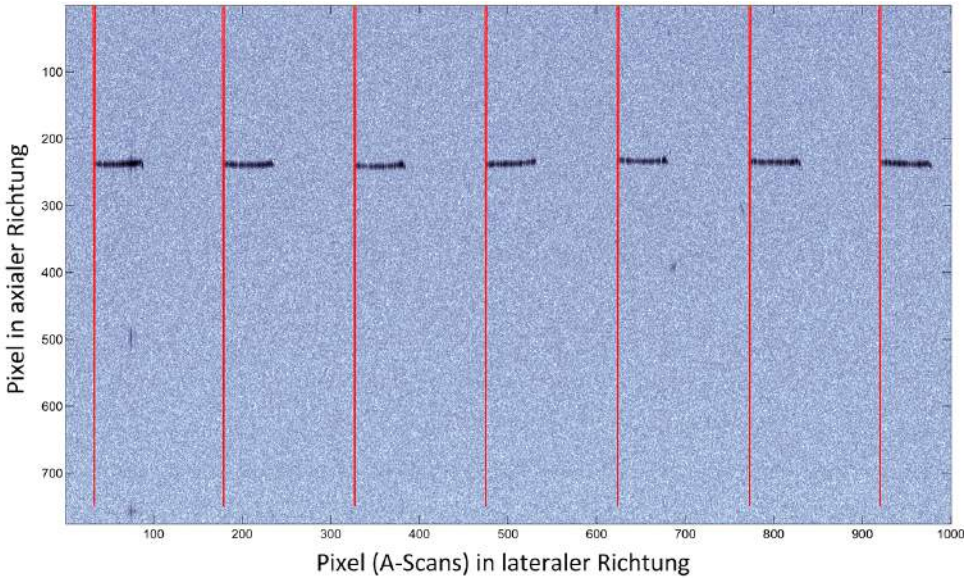


Abbildung 37: Gefundene Übergänge von *Nicht-Metall* auf *Metall*, dargestellt durch die roten Linien. Der mittlere Abstand dieser Linien entpricht dem Lochabstand des Lochblechs.

Die Geschwindigkeit v_{rel} ergibt sich demnach aus dem Lochabstand (engl.: hole center distance) x_{hcd} in Metern, dem mittleren Abstand der Transitionen x_{trans} in Pixel (A-Scans) und der benötigten Scanzeit t_{scan} pro A-Scan in Sekunden.

$$v_{rel} = \frac{x_{hcd}}{x_{trans} \cdot t_{scan}} \tag{30}$$

Damit lässt sich die Breite des abgedeckten Bildbereichs pro A-Scan x_{ascan} in Metern folgendermaßen berechnen:

$$x_{ascan} = v_{rel} \cdot t_{scan} \tag{31}$$

Für die einzelnen relativen Geschwindigkeiten zwischen Messsystem und Messobjekten und einer benötigten Zeit von 67.56 μs pro A-Scan ergibt sich demnach folgender numerische Zusammenhang (Tabelle 14):

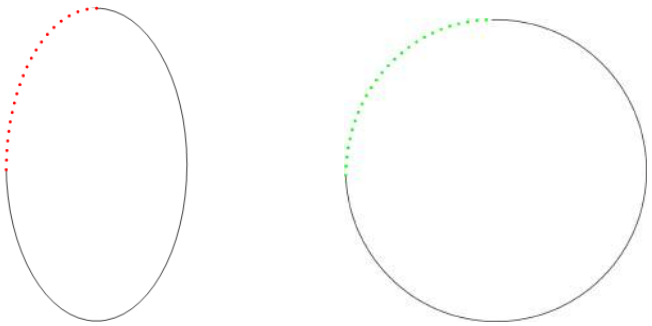
Tabelle 14: Numerischer Zusammenhang zwischen der Relativgeschwindigkeit Sensor - Messobjekt und der Bildbreite für 1000 A-Scans

Geschwindigkeit m/s	Bildbreite mm
0.1	6.76
0.2	13.51
0.3	20.27
0.4	27.02
0.5	33.78
0.6	40.54
0.7	47.29

Der axiale Bildbereich in Metern ergibt sich aus der Anzahl der Pixel in axialer Richtung $n_{px,axial}$, sowie der axialen Auflösung des Messsystems Δ_z:

$$x_{axial} = n_{px,axial} \cdot \Delta_z \tag{32}$$

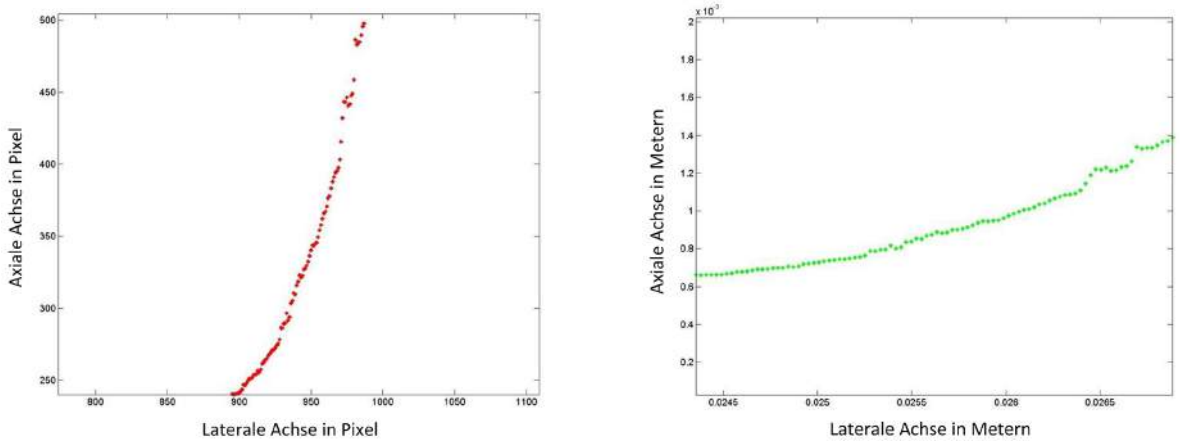
Somit sind Breite und Höhe der Bildpunkte bekannt und die Datentransformation kann erfolgen. Dieser Vorgang ist in Abbildung 38 graphisch dargestellt.



(a) (b)

Abbildung 38: Graphische Darstellung der Datentransformation vor dem Circle-Fitting, die farbigen Punkte symbolisieren hierbei die Datenpunkte der Tablettenoberfläche. a) Die Daten vor der Transformation sind entsprechend der Relativgeschwindigkeit verzerrt, b) die Daten nach der Transformation ergeben die tatsächliche Kreisform der Tablettenoberfläche und können für das nachfolgende Circle-Fitting verwendet werden.

Abbildung 39 zeigt die Transformation mit tatsächlich verwendeten Daten. Zu beachten ist hierbei, dass die Datenpunkte im Vergleich zu Abbildung 38 um die horizontale Achse gespiegelt erscheinen. Dies ist dadurch zu erklären, dass die vertikale Achse entsprechend des Scanvorganges von oben nach unten verläuft. Demnach ist im Bild der oberste Bildpunkt Pixel Nummer 1, der unterste ist Pixel Nummer 1024. In Abbildung 39 verläuft die Achse hingegen von unten nach oben, wodurch sich diese Spiegelung ergibt.



(a) (b)

Abbildung 39: Graphische Darstellung der Datentransformation mit realen Daten. a) Die Daten vor der Transformation sind entsprechend der Relativgeschwindigkeit verzerrt, b) die Daten nach der Transformation ergeben die tatsächliche Kreisform der Tablettenoberfläche und können für das nachfolgende Circle-Fitting verwendet werden.

Circle-Fitting mit fixem Radius Der größte Teil der publizierten Fitting-Verfahren ermittelt sowohl den Mittelpunkt, als auch den Radius des für die jeweilige Aufgabenstellung optimalen Kreises. Ein Merkmal dieser Algorithmen ist, dass sie umso effektiver arbeiten, je eher die Datenpunkte kreisförmig verteilt sind [**?**]. Da sich die in dieser Arbeit verwendeten Daten nur auf einem sehr kleinen Kreissektor befinden (siehe Abbildung 39), variiert der Radius der gefitteten Kreise sehr stark. Um ein sinnvolles Fitting-Ergebnis zu erhalten, muss daher der Radius des anzupassenden Kreises fix vorgegeben werden, sodass nur mehr der optimale Mittelpunkt gefunden werden muss.

Für weitere Informationen zu Circle-Fitting-Verfahren siehe beispielsweise [41] und [42].

5.2.2 Mittelpunktschätzung

Iterative Circle-Fitting-Verfahren erfordern einen initialen Wert von Radius und Kreismittelpunkt. Ausgehend von diesen Startwerten werden die Parameter solange verändert und optimiert, bis ein gewünschtes Kriterium erfüllt ist. Je näher die Anfangsparameter an den tatsächlichen Werten liegen, desto schneller und effektiver arbeitet der Circle-Fitting-Algorithmus. Zudem wird durch eine gute Schätzung der Startwerte das Risiko minimiert, beim Fitting-Verfahren ein lokales Minimum zu treffen.

In dieser Arbeit wurde ein simples geometrisches Schätzverfahren implementiert und als hinreichend genau befunden. Von dem aktuellen Satz an Oberflächendaten einer Tablette wurde sowohl von den ersten drei, als auch von den letzten drei Datenpunkten der Mittelwert gebildet. Diese beiden errechneten Punkte wurden anschließend mit einer Linie verbunden. Orthogonal zum Mittelpunkt dieser Linie wurde im Abstand des vorher festgelegten Radius (in Scanrichtung) der Initialwert des Mittelpunktes definiert.

Der Schätzalgorithmus wurde an einem künstlich generierten Datensatz gestestet, die Parameter dieses Datensatzes sind in Tabelle 15 aufgelistet.

Tabelle 15: Parameter des für den Test des Schätzalgorithmus generierten Datensatzes

Parameter	Wert
Mittelpunkt	[5,5]
Radius	1
Standardabweichung Radius	0.001, 0.01, 0.1
Datenpunkte	30
Kreissektor	30°

Abbildung 40, Abbildung 41 und Abbildung 42 zeigen die Ergebnisse der Mittelpunktschätzung für Datensätze mit verschiedenen Standardabweichungen des Radius. Der wahre Mittelpunkt mit den Koordinaten [5,5], sowie der dazu gehörende Kreis mit einem Radius von 1 sind in den Graphiken grau dargestellt. Die künstlich generierten und vom Kreisumfang mit der entsprechenden Standardabweichung abweichenden Datenpunkte sind grün eingefärbt. Der geschätzte Mittelpunkt und Ausgangswert für das danach erfolgende Fitting-Verfahren ist in den Graphiken als kleiner roter Kreis dargestellt und wird wie oben beschrieben berechnet (schwarze Kreise, rote Linien).

Anhand der Abbildungen und der darin beschriebenen mittleren Abweichung der geschätzten von den wahren Mittelpunkt-Koordinaten erkennt man, dass auch bei suboptimalen Daten ein brauchbarer Ausgangswert für das Fitting-Verfahren gefunden wird.

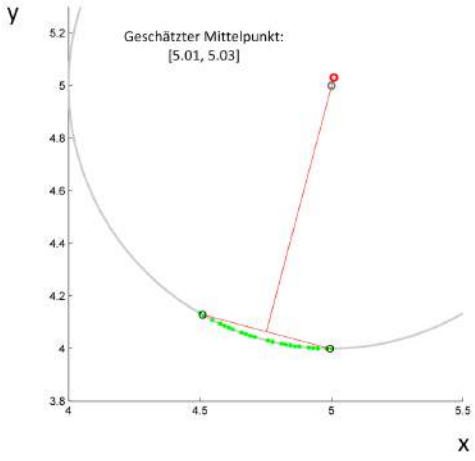


Abbildung 40: Geschätzter Mittelpunkt (rot) des Datensatzes bei einer Standardabweichung von $\sigma = 0.001$. Mittlere Abweichung der Koordinaten: 0.4%

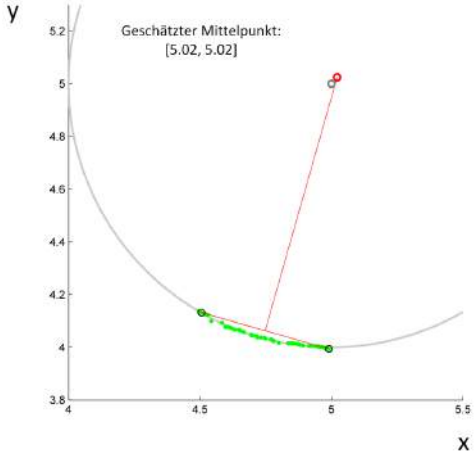


Abbildung 41: Geschätzter Mittelpunkt (rot) des Datensatzes bei einer Standardabweichung von $\sigma = 0.01$. Mittlere Abweichung der Koordinaten: 0.4%

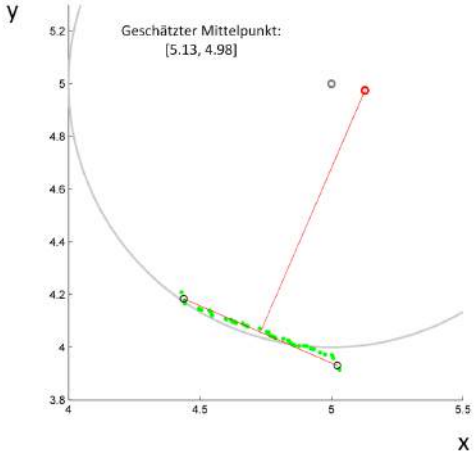


Abbildung 42: Geschätzter Mittelpunkt (rot) des Datensatzes bei einer Standardabweichung von $\sigma = 0.1$. Mittlere Abweichung der Koordinaten: 1.1%

5.2.3 Kreisanpassung

Die arithmetischen Verfahren von Kåsa [36], Pratt [37] und Taubin [38] berechnen sowohl Mittelpunkt, als auch Radius des Kreises. Dies ergab mit den zur Verfügung stehenden Daten stark variierende Radien und somit unzureichende Ergebnisse, weswegen in dieser Arbeit auf geometrische Verfahren zurückgegriffen wurde.

Für das Circle-Fitting-Verfahren wurden zwei verschiedene Algorithmen gestestet. Ein von Li et al. publizierter Algorithmus [43] verwendet das Prinzip der Maximum-Likelihood-Estimation (MLE) als Grundlage, der zweite Algorithmus wurde selbst entworfen und basiert auf der Newton-Raphson-Methode und dem Least-Square-Error-Ansatz (LSE). Die Newton-Raphson-Methode gilt als schneller konvergierend als die Levenberg-Marquardt-Methode [44], allerdings besteht dabei die Gefahr des Divergierens.

Beide implementierten Verfahren definieren eine Kostenfunktion, die mit Hilfe eines Update-Terms iterativ minimiert wird.

Circle-Fitting mittels Maximum-Likelihood-Estimation Der Update-Term des von Li et al. publizierten Algorithmus ergibt sich nach einigen Rechenschritten, Annahmen und Umformungen zu:

$$c_n = \frac{1}{N} \sum_{i=1}^{N} \left(a_i + r \cdot \frac{c_{n-1} - a_i}{||c_{n-1} - a_i||} \right) \tag{33}$$

c_n ist hierbei der aktuell zu berechnende Kreismittelpunkt bestehend aus seinen zwei Koordinaten; N ist die Anzahl der Datenpunkte; a_i sind die Datenpunkte, bestehend aus ihren zwei Koordinaten; r ist der festgelegte Radius und c_{n-1} ist der in der vorherigen Iteration berechnete Kreismittelpunkt. Die Überlegungen zu diesem Algorithmus und der gesamte Rechenvorgang sind unter [43] nachzulesen.

Circle-Fitting mittels Least-Square-Error-Ansatz Für die Punkte $\boldsymbol{p}_i = (p_{i,x},p_{i,y})$ auf einem Kreis mit dem Mittelpunkt $\boldsymbol{c} = (c_x,c_y)$ und dem Radius r gilt folgende Gleichung:

$$(p_{i,x} - c_x)^2 + (p_{i,y} - c_y)^2 = r^2 \tag{34}$$

Der quadratische Fehler (engl.: square error) E der Datenpunkte $\boldsymbol{a}_i$ zum tatsächlichen Kreis ergibt sich zu:

$$E = \sum_{i=1}^{N} (r^2 - (a_{i,x} - c_x)^2 - (a_{i,y} - c_y)^2)^2 \tag{35}$$

Ziel des Fitting-Verfahrens ist es also denjenigen Kreis zu finden, bei dem dieser Fehler minimal wird ($E = min$). Das Minimum einer Funktion erhält man durch das Nullsetzen deren erster Ableitung. Dabei ist nicht sichergestellt, dass damit auch das globale Minimum gefunden wird, es besteht die Möglichkeit, ein lokales Minimum zu treffen (siehe Abbildung 43). Aus diesem Grund ist die zuvor beschriebene Schätzung des Kreismittelpunktes unbedingt durchzuführen.

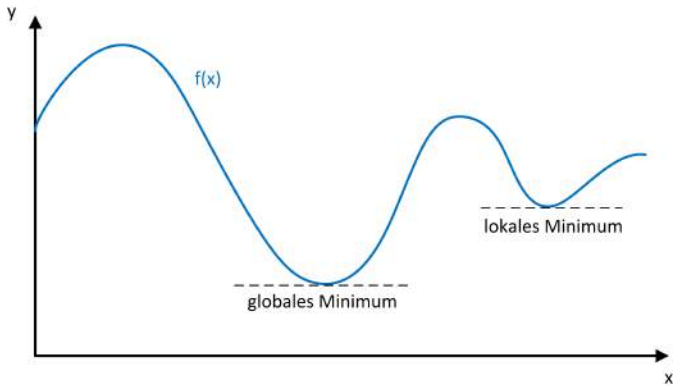


Abbildung 43: Globales und lokales Minimum einer Funktion $f(x)$

Newton-Raphson-Methode Die Newton-Raphson-Methode [33], oft auch nur Newton-Methode genannt, ist ein Verfahren, mit dem Näherungswerte zu Lösungen der nichtlinearen Gleichung $f(x) = 0$ berechnet werden können.

Grundlage dieser Annäherung ist die Taylorreihen-Entwicklung, mit der sich eine Funktion $f(x)$ ausdrücken lässt durch:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + ... + \frac{f^(n)(x_0)}{n!}(x - x_0)^n \tag{36}$$

x_0 ist hierbei der Startwert, der im Vorhinein bekannt sein muss. Eine einfache Annäherung von $f(x) = 0$ lässt sich so mit den ersten beiden Termen aus Gleichung (36) realisieren.

$$f(x) = 0 \approx f(x_0) + f'(x_0)(x - x_0) \tag{37}$$

Durch Umformen und einer wiederholten Anwendung dieser Gleichung erhält man:

$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$,
$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$,
$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$, ...

wodurch sich der Newton-Raphson-Ansatz ergibt:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{38}$$

Dieser Schritt wird so lange ausgeführt, bis eine vorher definierte Fehlerschwelle unterschritten, oder eine gewisse Anzahl Iterationsschritte erreicht wird.

Mehrdimensionaler Fall Das Newton-Raphson-Verfahren kann, wie in dieser Arbeit benötigt, auch im Falle einer mehrdimensionalen Funktion angewandt werden. Dabei werden der Gradient und die Hesse-Matrix $\boldsymbol{H}$ der zu minimierenden Funktion E benötigt. Die Hesse-Matrix bestehet aus den zweiten partiellen Ableitungen dieser Funktion. Da die Fehlerfunktion in der hier auftretenden Problemstellung zwei Variablen, nämlich die beiden Koordinaten des Kreismittelpunktes beinhaltet, gilt:

$$\nabla E = \left(\frac{\partial E}{\partial c_x}, \frac{\partial E}{\partial c_y} \right)^T \tag{39}$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial c_x \partial c_x} & \frac{\partial^2 E}{\partial c_x \partial c_y} \\ \frac{\partial^2 E}{\partial c_y \partial c_x} & \frac{\partial^2 E}{\partial c_y \partial c_y} \end{bmatrix} \tag{40}$$

Analog zum eindimensionalen Fall erhält man für die hier beschriebene Problemstellung, das Ermitteln des Kreismittelpunktes $\boldsymbol{c}$, folgenden Update-Formalismus:

$$\mathbf{c}_{n+1} = \mathbf{c}_n - \mathbf{H}(\mathbf{c}_n)^{-1} \nabla E(\mathbf{c}_n) \tag{41}$$

5.3 Ergebnisse

5.3.1 MLE vs. Newton-Raphson mit Mittelpunktschätzung

Die Abbildungen 44 und 45 zeigen das Ergebnis der beiden Circle-Fitting-Verfahren, angewandt auf einen Testdatensatz und mit zuvor durchgeführter Mittelpunktschätzung.

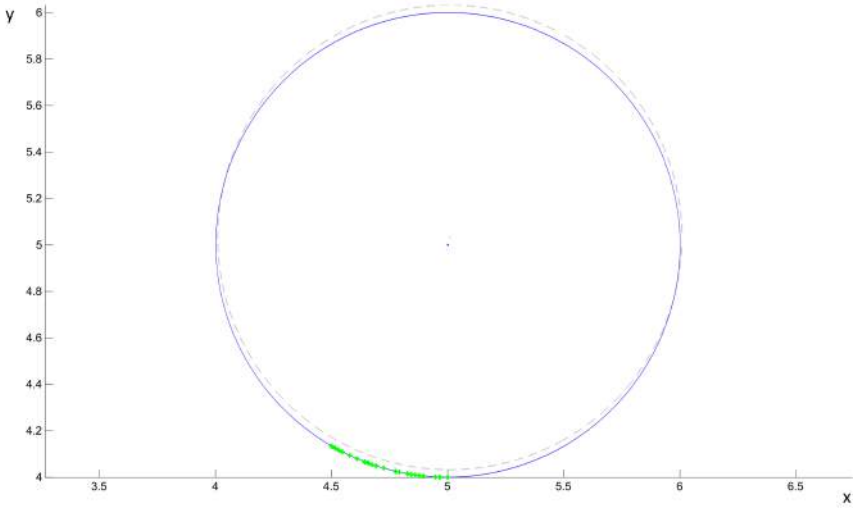


Abbildung 44: Circle-Fitting mittels Newton-Raphson auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.001$. Der graue, strichlierte Kreis ist das Ergebnis der Mittelpunktschätzung und stellt den Startwert des Verfahrens dar. Der dunkelste blaue Kreis ist das Ergebnis des Circle-Fittings und wurde nach drei Iterationen erreicht.

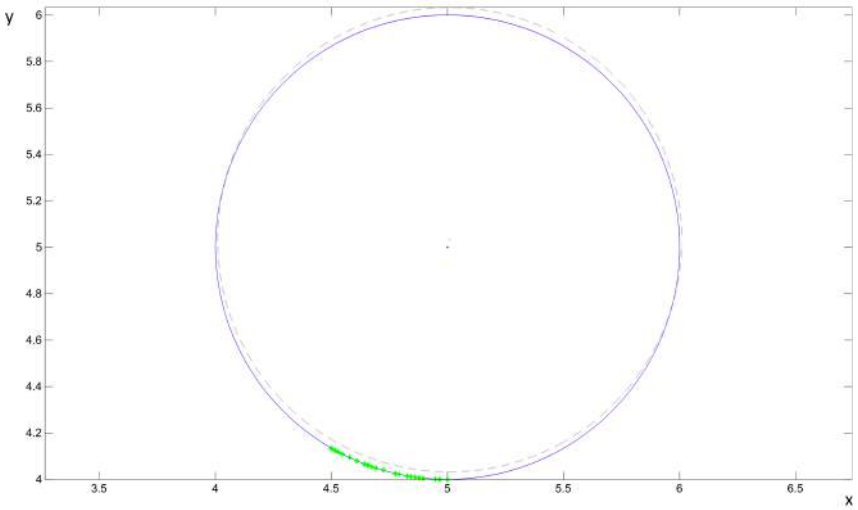


Abbildung 45: Circle-Fitting mittels Maximum-Likelihood-Estimation auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.001$. Der graue, strichlierte Kreis ist das Ergebnis der Mittelpunktschätzung und stellt den Startwert des Verfahrens dar. Der dunkelste blaue Kreis ist das Ergebnis des Circle-Fittings und wurde nach drei Iterationen erreicht.

5.3.2 MLE vs. Newton-Raphson ohne Mittelpunktschätzung

Die nachfolgenden Bilder (Abbildungen 46, 47, 48 und 49) zeigen die Ergebnisse der Circle-Fitting-Verfahren ohne zuvor durchgeführte Mittelpunktschätzung. Der Startwert wurde dabei bewusst schlecht gewählt, um die Stabilität der Algorithmen zu beobachten.

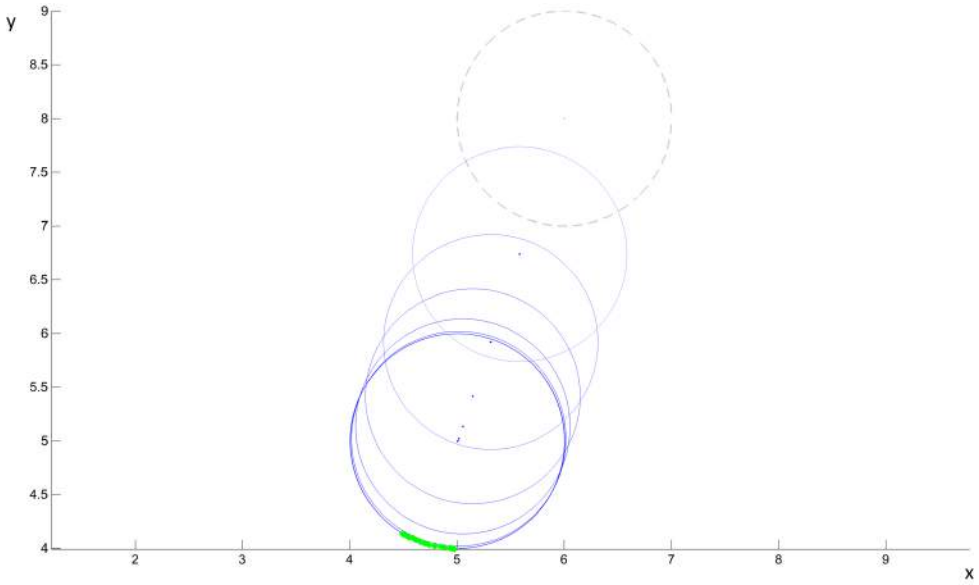


Abbildung 46: Circle-Fitting mittels Newton-Raphson auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.01$. Der graue, strichlierte Kreis stellt den Startwert [6,8] des Verfahrens dar. Der dunkelste blaue Kreis ist das Ergebnis des Circle-Fittings und wurde nach acht Iterationen erreicht.

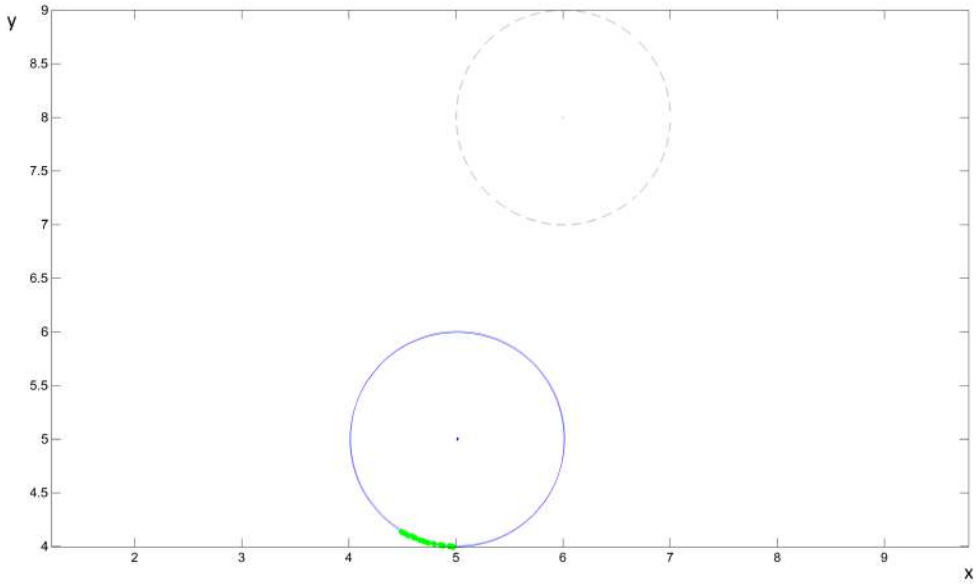


Abbildung 47: Circle-Fitting mittels MLE auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.01$. Der graue, strichlierte Kreis stellt den Startwert [6,8] des Verfahrens dar. Der dunkelste blaue Kreis ist das Ergebnis des Circle-Fittings und wurde nach drei Iterationen erreicht.

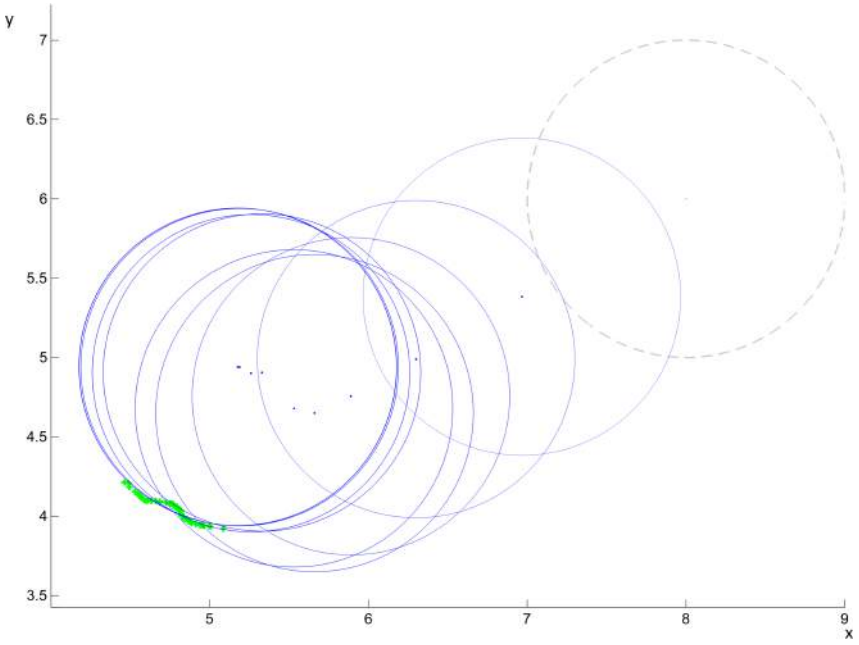


Abbildung 48: Circle-Fitting mittels Newton-Raphson auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.1$. Der graue, strichlierte Kreis stellt den Startwert [8,6] des Verfahrens dar. Der dunkelste blaue Kreis ist das Ergebnis des Circle-Fittings und wurde nach zehn Iterationen erreicht.

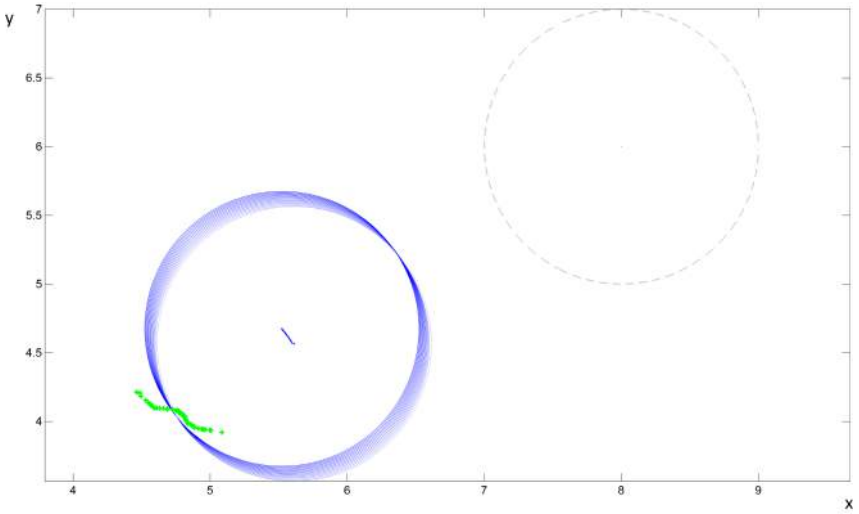


Abbildung 49: Circle-Fitting mittels MLE auf einen Testdatensatz mit einer Standardabweichung von $\sigma = 0.1$. Der graue, strichlierte Kreis stellt den Startwert [8,6] des Verfahrens dar. Nach den maximal zugelassenen 100 Iterationen wurde kein brauchbares Fitting-Ergebnis gefunden.

5.3.3 Numerische Gegenüberstellung der beiden Verfahren

In Tabelle 16 ist ein numerischer Vergleich der beiden Circle-Fitting-Verfahren angeführt. Als Grundlage der Bewertung der Qualität des Circle-Fittings wurde der mittlere quadratische Fehler (engl.: mean square error) der Kreispunkte zu den Datenpunkten verwendet. Dieser Wert berechnet sich zu:

$$E_{ms} = 1/N \sum_{i=1}^{N} (r^2 - (a_{i,x} - c_x)^2 - (a_{i,y} - c_y)^2)^2 \tag{42}$$

Zusätzlich zu diesem Parameter wurden der Zeitaufwand für die Mittelpunktschätzung, sowie die Dauer der beiden Algorithmen und deren benötigten Iterationen aufgezeichnet. Als Testdatensatz wurde derselbe wie schon für die Mittelpunktschätzung verwendet, die Parameter dieser Daten sind in Tabelle 15 aufgelistet. Für den Testfall ohne Mittelpunktschätzung wurde der Startwert der Fitting-Verfahren für den Kreismittelpunkt auf [6,8] gesetzt.

Tabelle 16: Vergleich der Circle-Fitting-Algorithmen

Standardabweichung Rauschen	Startwert-Schätzung	Zeit Schätzverfahren (in ms)	Algorithmus	Zeit Algorithmus (in ms)	Mean Square Error	Iterationen	Gesamtzeit (in ms)
0,001	nein	0,00	Maximum Likelihood	5,19	0,0064	3	5,19
			Newton Raphson	6,18	0,0008	8	6,18
	ja	4,86	Maximum Likelihood	5,12	0,0009	2	9,98
			Newton Raphson	4,86	0,0009	3	9,72
0,01	nein	0,00	Maximum Likelihood	5,71	0,0146	3	5,71
			Newton Raphson	5,73	0,0062	8	5,73
	ja	5,06	Maximum Likelihood	5,20	0,0066	2	10,26
			Newton Raphson	4,61	0,0057	3	9,67
0,1	nein	0,00	Maximum Likelihood	5,59	0,0384	3	5,59
			Newton Raphson	6,23	0,0207	9	6,23
	ja	4,83	Maximum Likelihood	5,09	0,0229	2	9,93
			Newton Raphson	4,61	0,0219	3	9,44

5.3.4 Bewertung der Ergebnisse

Sowohl die Newton-Raphson-Methode, als auch der Algorithmus mit dem Maximum-Likelihood-Ansatz arbeiten schnell (wenige Iterationen) und effektiv. Es lässt sich in obiger Tabelle aber erkennen, dass die Methode nach Newton-Raphson bei vorher durchgeführter Mittelpunktschätzung sowohl betreffend der Geschwindigkeit, als auch der Genauigkeit, ein geringfügig besseres Ergebnis liefert. Zudem scheint dieser Algorithmus stabiler bezüglich des Findens des globalen Minimums zu sein und wurde daher schlussendlich für weitere Schritte ausgewählt.

6 Darstellung der Ergebnisse

In diesem Kapitel wird die Überprüfung des Gesamtalgorithmus auf seine Funktionsfähigkeit angeführt. Insgesamt wurde der Algorithmus auf 250 B-Scans (Zehn Bilder je Schichtdicke und Geschwindigkeit) angewandt. Wie bereits in Kapitel 4.1 angeführt, wurde in dieser Arbeit das Verfahren der „supervised classification“ umgesetzt. Grundlage dieses Verfahrens ist eine Vorab-Klassifizierung der Daten, auf Basis derer die entsprechenden Modelle und Algorithmen erstellt werden. Da für die in dieser Arbeit verwendeten Daten keine definierte Klassenzuteilung existierte, mussten die einzelnen A-Scans im Vorfeld manuell (optisch) klassifiziert werden.

Im Laufe der nachfolgend durchgeführten Datenverarbeitung (z.B. Kreisanpassung) musste festgestellt werden, dass nicht alle Daten, die bei der manuellen Klassifizierung als *Tablette* eingestuft wurden, nun auch für die Weiterverarbeitung verwendet werden können. Es war deshalb erforderlich, eine erneute manuelle Klassifizierung durchzuführen bei der nun auch die Weiterverarbeitung berücksichtigt wurde. Dabei wurde nur mehr zwischen „aussagekräftigen Daten“ und „nicht aussagekräftigen Daten“ unterschieden.

Ein Vergleich zwischen den manuellen Einstufungen „aussagekräftig“ und „nicht aussagekräftig“ ist in Abbildung 50 zu sehen. Grundlage dieser Unterscheidung war einerseits die Erkennbarkeit der Tablettenbeschichtung, andererseits die grob abgeschätzte Länge des Objektes von ungefähr 50 A-Scans.

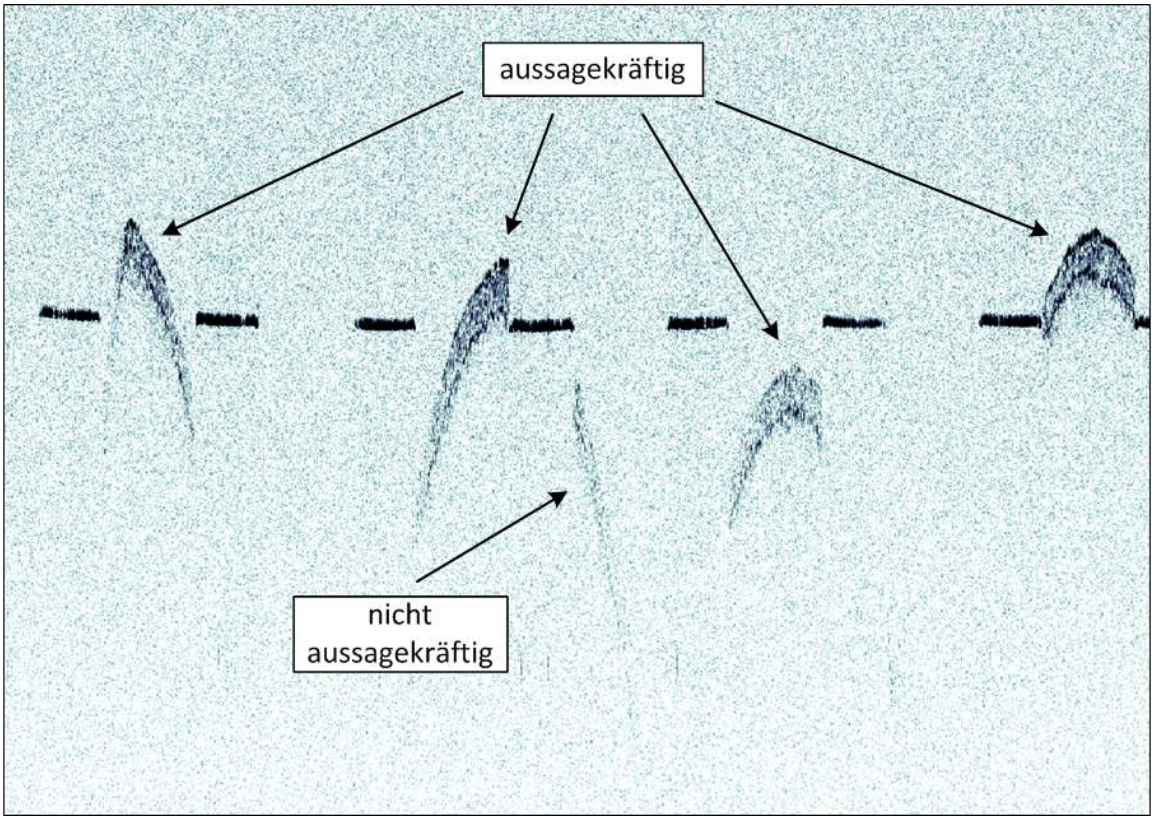


Abbildung 50: Vergleich zwischen manuell als „aussagekräftig“ bzw. als „nicht aussagekräftig“ eingestuften Tablettenaufnahmen.

Rund 20% der als aussagekräftige Messobjekte eingestuften Tabletten waren so unter dem Messsystem platziert, dass der Tablettensteg auf dem Bild sichtbar war. Die Detektion dieses Steges war nicht Teil dieser Arbeit und bedarf weiterer Optimierung. Etwa zwei Drittel der misslungenen Tabletten-Fits des Gesamtergebnisses resultierten aus einem falsch oder gar nicht detektierten Tablettensteg. Eine Übersicht über die durchgeführte Abschlussprüfung des Algorithmus ist in Tabelle 17 dargestellt.

Die Stegdetektion ist derzeit so implementiert, dass über die Berechnung der Steigung zwischen den Datenpunkten der Tablettenoberfläche versucht wird, die Kante des Steges zu ermitteln. An diesem Punkt werden die Daten aufgeteilt und auf beiden Seiten eine Gerade eingepasst. Auf der Seite mit dem geringeren mittleren quadratischen Fehler zwischen Datenpunkten und der Geraden befindet sich der Steg. Bei Tabletten, bei denen ein Steg detektiert wird, werden nur diejenigen Daten vom Algorithmus weiterverarbeitet, die nicht als Tablettensteg erkannt wurden.

Tabelle 17: Daten und Ergebnisse der Abschlussüberprüfung des Algorithmus

Tabletten gesamt	558
Tabletten gefunden	516
davon mit Steg	107
Fit in Ordnung	383
davon mit Steg	33
Fit nicht in Ordnung	133
davon wenn Steg nicht erkannt	51
wenn Steg falsch erkannt	37
wenn kein Steg	45

Beurteilung des Fitting-Ergebnisses Da zum Zeitpunkt der Erstbeurteilung der Ergebnisse nach dem Durchlauf der 250 B-Scans keine weiteren Anhaltspunkte gegeben waren, wurde diese Bewertung manuell durchgeführt. Es wurde optisch überprüft, ob das Ergebnis des Circle-Fitting-Verfahrens dem Verlauf der jeweiligen Tablettenoberfläche entspricht.

Diese Vorgehensweise ist für den realen Einsatz dieses Algorithmus natürlich nicht geeignet, allerdings wurde dadurch eine Grundlage für weitere Überlegungen geschaffen. Ausgehend von den so gewonnenen Erkenntnissen wurden weitere Schritte, wie in Kapitel 6.3 beschrieben, implementiert.

In den nachfolgenden Abbildungen symbolisiert die gelbe Linie das Ergebnis des Circle-Fitting-Verfahrens, das für die jeweilige auf dem Bild erkannte Tablette nach der Klassifizierung durchgeführt wurde.

Wie bereits erwähnt, wurde dieses Fitting-Ergebnis bei der Erstbeurteilung des Algorithmus manuell als „in Ordnung“ oder „nicht in Ordnung“ bewertet. Dabei wurde beurteilet, ob die Linie des gefitteten Kreises mit der im Bild erkennbaren Tablettenoberfläche übereinstimmt.

6.1 Erfolgreiche Tablettendetektion

Abbildung 51 zeigt die erfolgreiche Tablettendetektion für eine geringe Beschichtungsdicke. Für die letzte Tablette im Bild wurden zu wenige Datenpunkte erkannt, sie wird daher vom Algorithmus ignoriert.

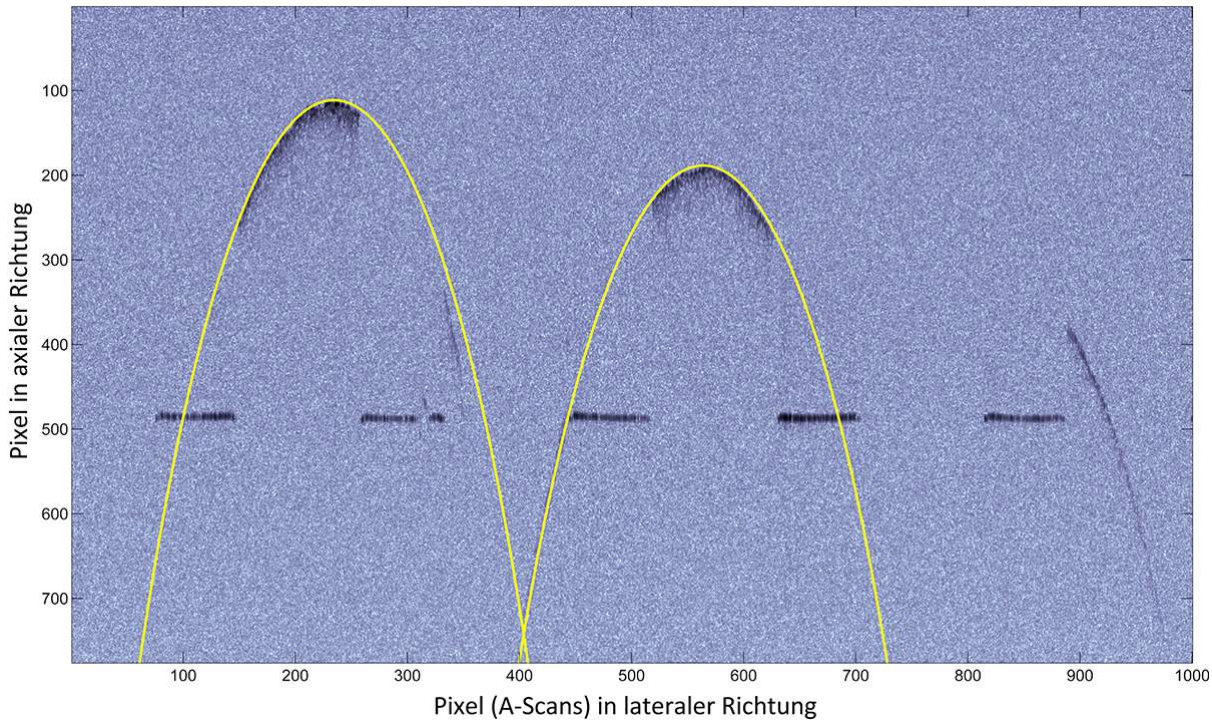


Abbildung 51: Tablettendetektion bei einer Beschichtungsdauer von 10 Minuten. Für die letzte Tablette im Bild wurden zu wenige Datenpunkte erkannt, sie wird daher vom Algorithmus ignoriert.

In Abbildung 52 und 53 ist die Detektion der Tabletten zu sehen, wobei manche der Tabletten so positioniert sind, dass der Steg im Bild erkennbar ist. In diesen Fällen hat die Stegerkennung funktioniert, was somit auch bei diesen Tabletten ein zufriedenstellendes Circle-Fitting ermöglicht.

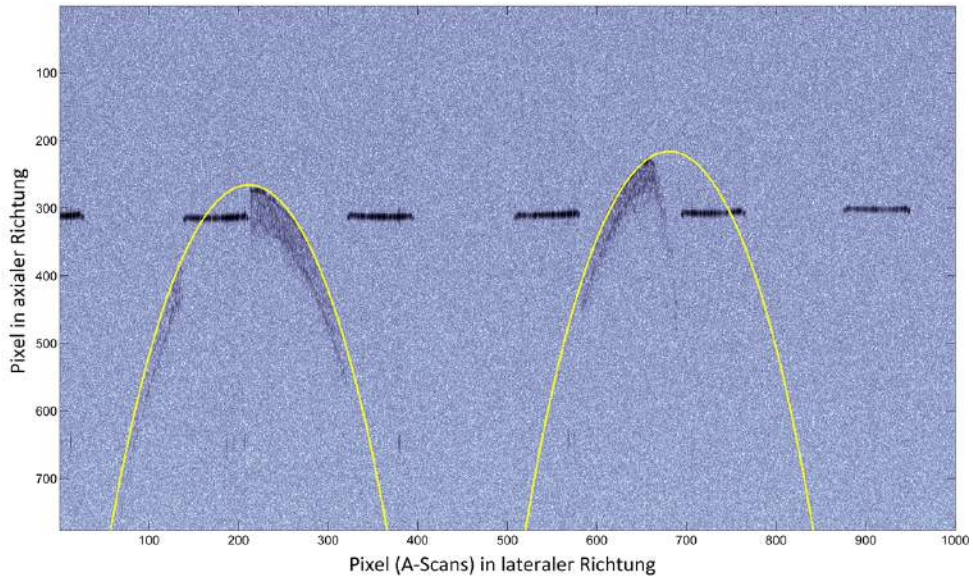


Abbildung 52: Erfolgreiche Tablettendetektion bei einer Beschichtungsdauer von 127 Minuten, die rechte der beiden Tabletten weist einen Steg auf. Dieser wurde richtig erkannt, sodass das Circle-Fitting und somit die Positionsbestimmung mit positivem Ergebnis durchgeführt werden konnte.

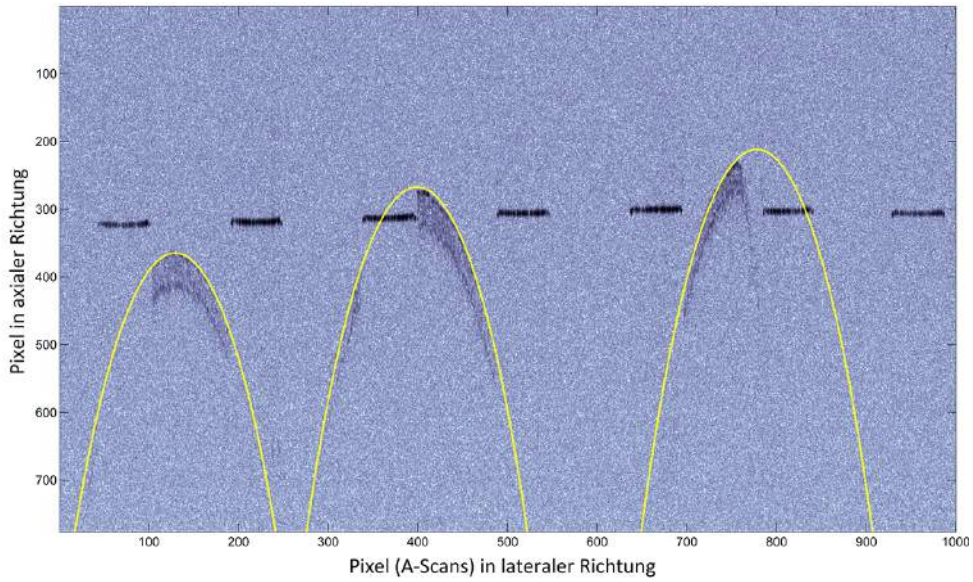


Abbildung 53: Erfolgreiche Tablettendetektion bei einer Beschichtungsdauer von 127 Minuten, die rechte der drei Tabletten weist einen Steg auf. Dieser wurde richtig erkannt, sodass das Circle-Fitting und somit die Positionsbestimmung mit positivem Ergebnis durchgeführt werden konnte.

Wie bereits erwähnt, werden während dem Optimierungsvorgang bei der Eruierung der Datenpunkte der Tablettenoberfläche Objekte mit weniger als 50 zusammenhängenden A-Scans ausgesiebt. Bei diesen wird in weiterer Folge kein Circle-Fitting und somit keine Tablettendetektion vorgenommen, was in den nachfolgenden Abbildungen 54 und 55 erkennbar ist.

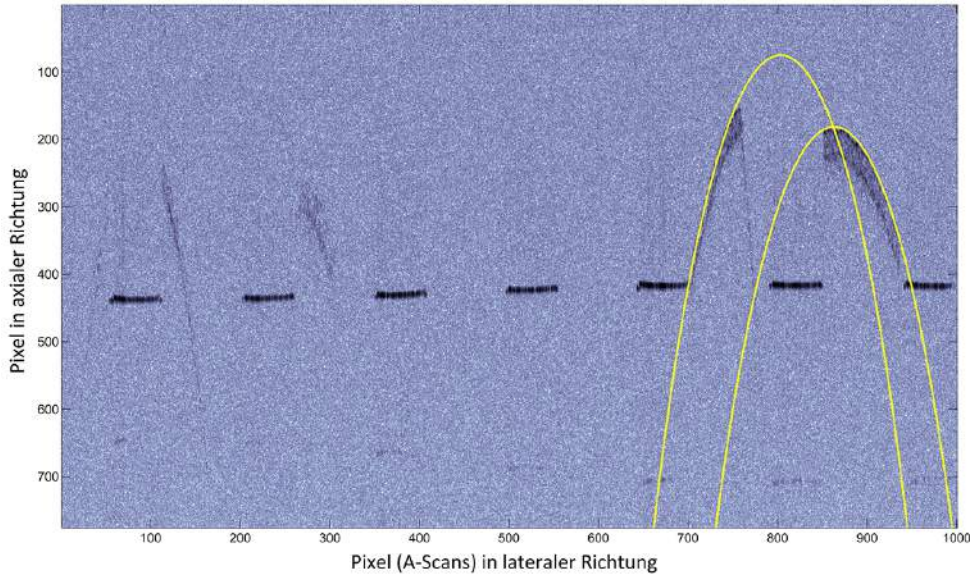


Abbildung 54: Erfolgreiche Tablettendetektion bei einer Beschichtungsdauer von 96 Minuten, die Tabletten im linken Bildbereich erfüllen die Kriterien für das Cirlce-Fitting nicht und werden daher ignoriert.

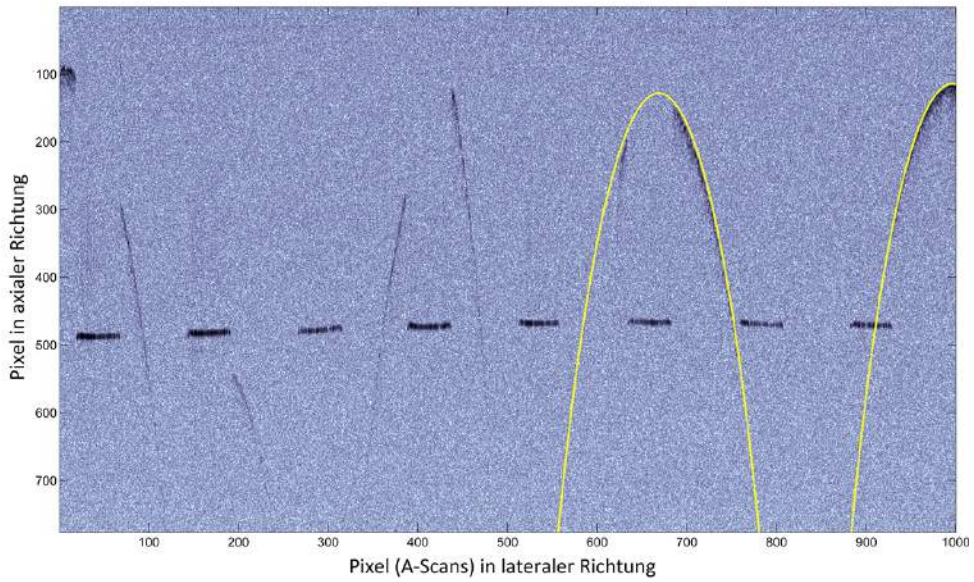


Abbildung 55: Erfolgreiche Tablettendetektion bei einer Beschichtungsdauer von 37 Minuten, die Tabletten im linken Bildbereich erfüllen die Kriterien für das Cirlce-Fitting nicht und werden daher ignoriert.

6.2 Fehlerhafte Tablettendetektion

Eine positive Lagebestimmung ist nur mit entsprechend guten Daten möglich. Haben diese Daten eine niedrige Qualität, ist es für den Algorithmus schwierig die Tablettenoberfläche zu erkennen, wodurch in weiterer Folge das Circle-Fitting suboptimal verläuft (siehe Abbildung 56 und Abbildung 57).

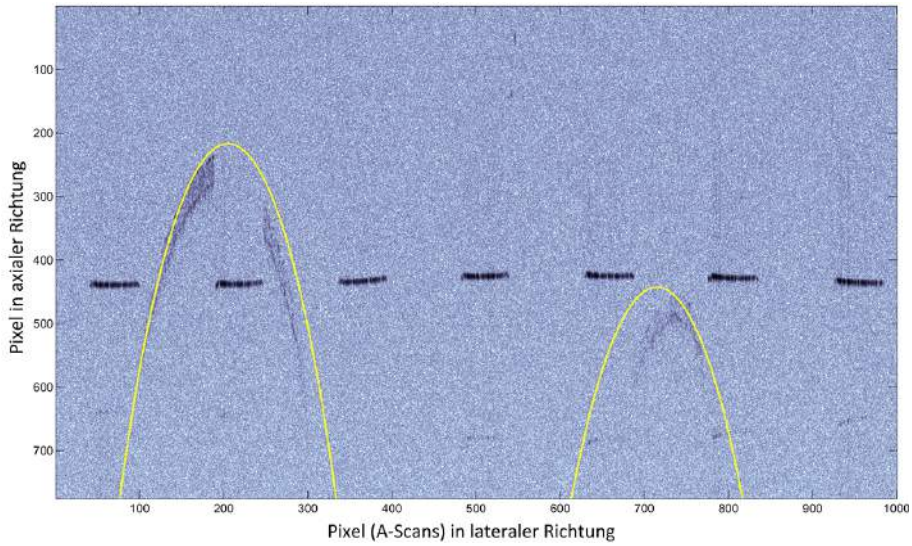


Abbildung 56: Schlechte Lagebestimmung der Tabletten, verursacht durch die niedrige Datenqualität der Scans. Die Tablettenoberfläche kann nicht optimal bestimmt werden, wodurch das Circle-Fitting ein unzureichendes Ergebnis liefert.

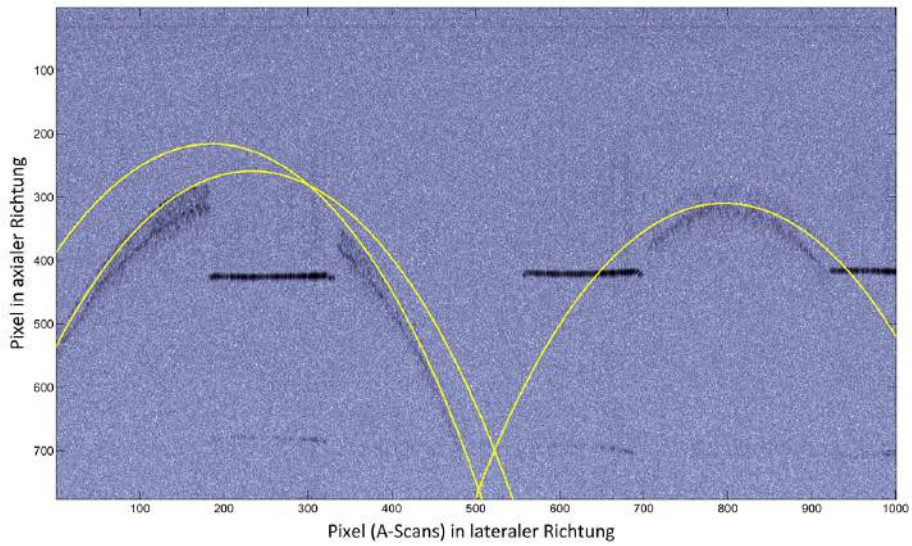


Abbildung 57: Schlechte Lagebestimmung der Tabletten, verursacht durch die niedrige Datenqualität der Scans. Die Tablettenoberfläche kann nicht optimal bestimmt werden, wodurch das Circle-Fitting ein unzureichendes Ergebnis liefert. Zudem ist in der Darstellung leicht zu erkennen, dass die Messung bei geringerer Geschwindigkeit durchgeführt wurde.

Ein weiteres, derzeit noch häufiges Problem ist eine fehlerhafte oder gar nicht vorhandene Erkennung des Tablettensteges. Dabei werden die Datenpunkte des Steges in das Circle-Fitting-Verfahren miteinbezogen, wodurch der Kreis niemals optimal eingepasst werden kann. Diese Problematik ist in den Abbildungen 58 und 59 dargestellt.

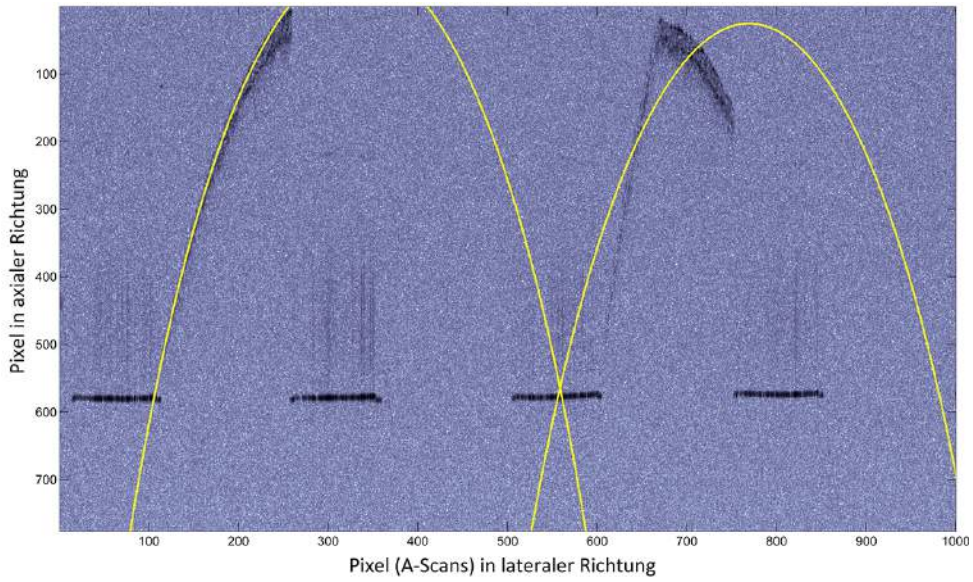


Abbildung 58: Unbrauchbare Tablettendetektion bei der rechten der beiden Tabletten, verursacht durch eine fehlerhafte oder nicht vorhandene Stegerkennung. Die Datenpunkte des Tablettensteges werden in die Oberflächendaten des Circle-Fittings miteinbezogen, wodurch kein optimales Ergebnis erzielt werden kann.

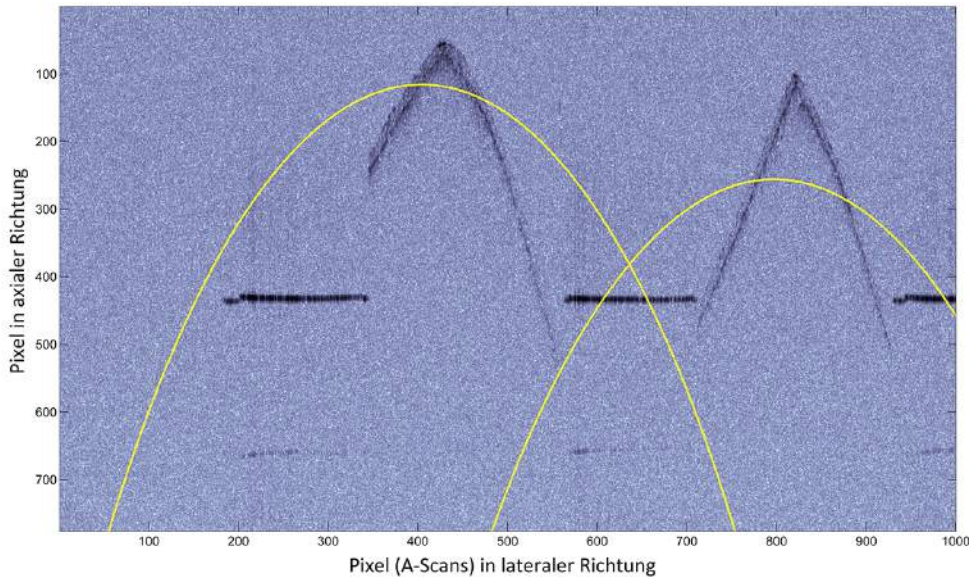


Abbildung 59: Unbrauchbare Tablettendetektion, verursacht durch eine fehlerhafte oder nicht vorhandene Stegerkennung. Die Datenpunkte des Tablettensteges werden in die Oberflächendaten des Circle-Fittings miteinbezogen, wodurch kein optimales Ergebnis erzielt werden kann.

Wie in Kapitel 5 beschrieben, besteht bei der Durchführung des Circle-Fittings die Gefahr, dass nicht das globale, sondern das lokale Minimum der Kostenfunktion gefunden wird. Trotz des relativ stabilen Circle-Fitting-Algorithmus ist dieser Fall während den Tests vereinzelt eingetroffen. Selbstverständlich ist auch hier das Ergebnis der Lagebestimmung nicht zu gebrauchen, wie in den Abbildungen 60 und 61 zu sehen ist.

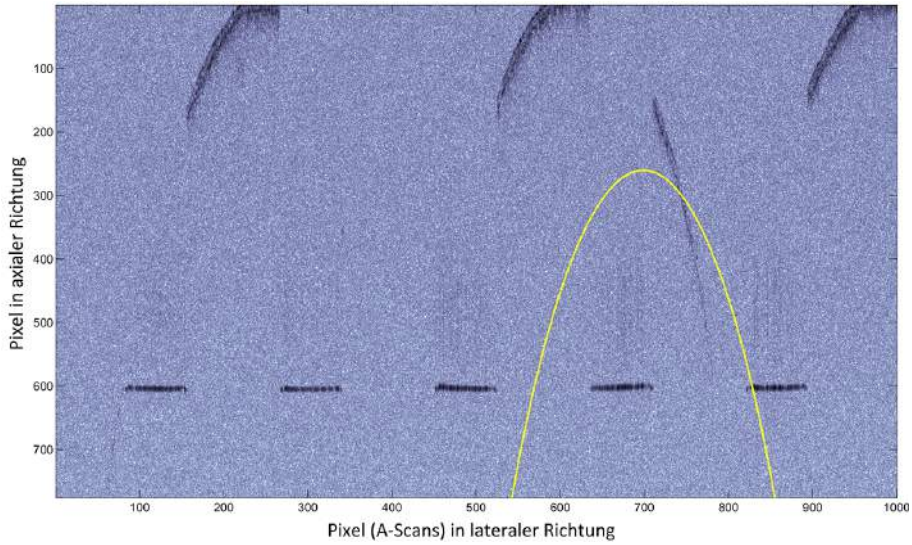


Abbildung 60: Unzureichende Lagebestimmung der dritten Tablette im Bild. Ursache ist hier der Circle-Fitting-Algorithmus, der den Kreis entsprechend einem lokalen Minimum der Kostenfunktion positioniert. Die Detektion der übrigen Tabletten im Bild ist in diesem Fall der Übersichtlichkeit wegen ausgeblendet.

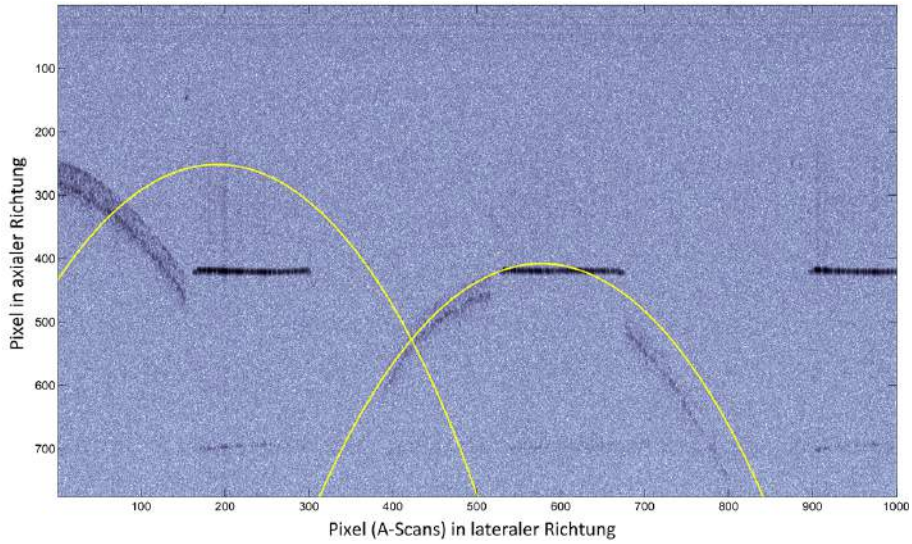


Abbildung 61: Unzureichende Lagebestimmung der ersten Tablette im Bild. Ursache ist hier wiederum der Circle-Fitting-Algorithmus, der den Kreis entsprechend einem lokalen Minimum der Kostenfunktion positioniert. Auch die Lage der zweiten Tablette wird nicht optimal erkannt, allerdings ist in diesem Fall die schlechte Datenqualität ausschlaggebend.

6.3 Abhilfe bei suboptimaler Lagebestimmung

Die unbrauchbaren Ergebnisse der Lagebestimmung können für die nachfolgenden Berechnungen (zum Beispiel die Bestimmung der Beschichtungsdicke) nicht weiterverwendet werden und müssen daher eliminiert werden. Da bei jedem durchgeführten Circle-Fitting-Prozess auch der mittlere quadratische Fehler der Datenpunkte zum Kreis ermittelt wird, liegt es auf der Hand, dass dieser Wert als entsprechendes Kriterium herangezogen wird.

6.3.1 Bestimmung des Fehlerschwellwertes mittels Konfidenzintervall

Für jedes untersuchte Bild wurde eine Statistik erstellt, die auflistet, ob ein Circle-Fit optisch zufriedenstellend oder unzureichend ist. Zusätzlich dazu wurde für jeden Fall der jeweilige Fehlerwert protokolliert. Ziel dieses Arbeitsschrittes war es, einen Schwellwert für den mittleren quadratischen Fehler zu bestimmen. Wird dieser überschritten, so gilt die Tablettendetektion als nicht gut genug und das Ergebnis wird verworfen.

Abbildung 62 zeigt, stellvertretend für die restlichen Schichtdicken, das Ergebnis dieser Statistik bei einer Beschichtungsdauer von 37 Minuten, Abbildung 63 beinhaltet alle getesteten Bilder.

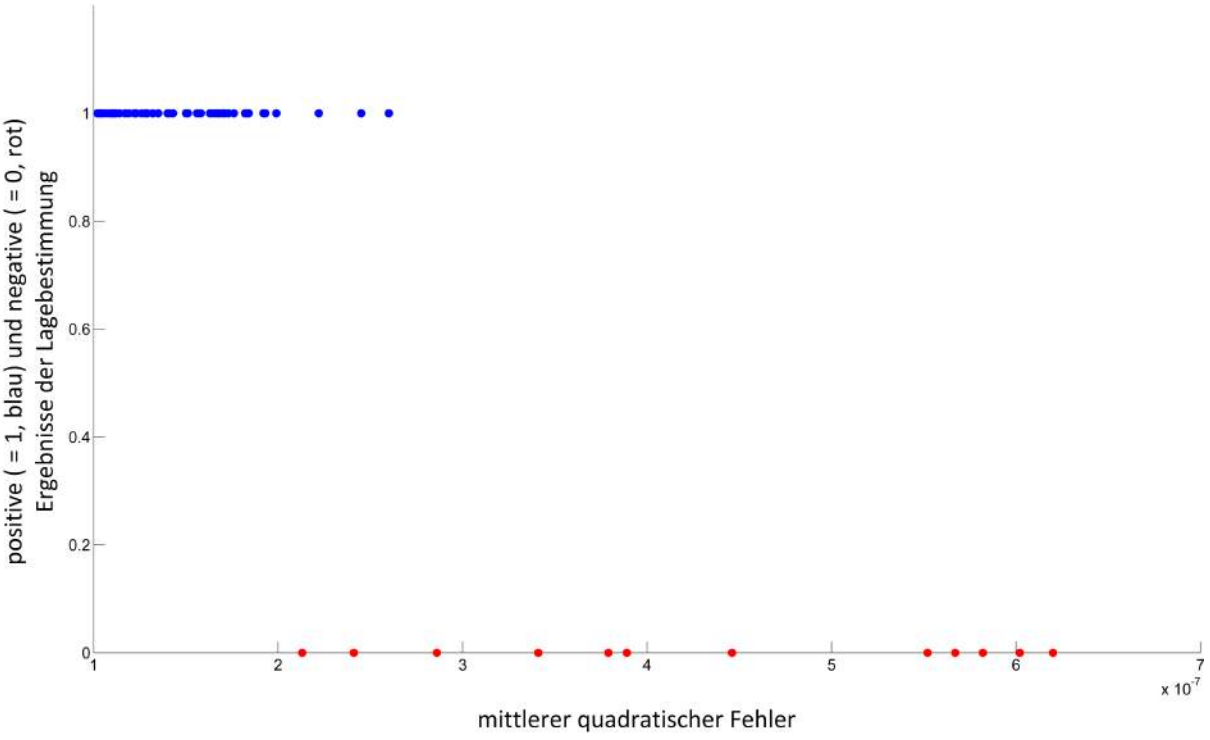


Abbildung 62: Statistik der als positiv (= 1) und negativ (= 0) bewerteten Ergebnisse des Circle-Fittings und des jeweiligen Fehlerwertes für alle getesteten Tabletten bei einer Beschichtungsdauer von 37 Minuten.

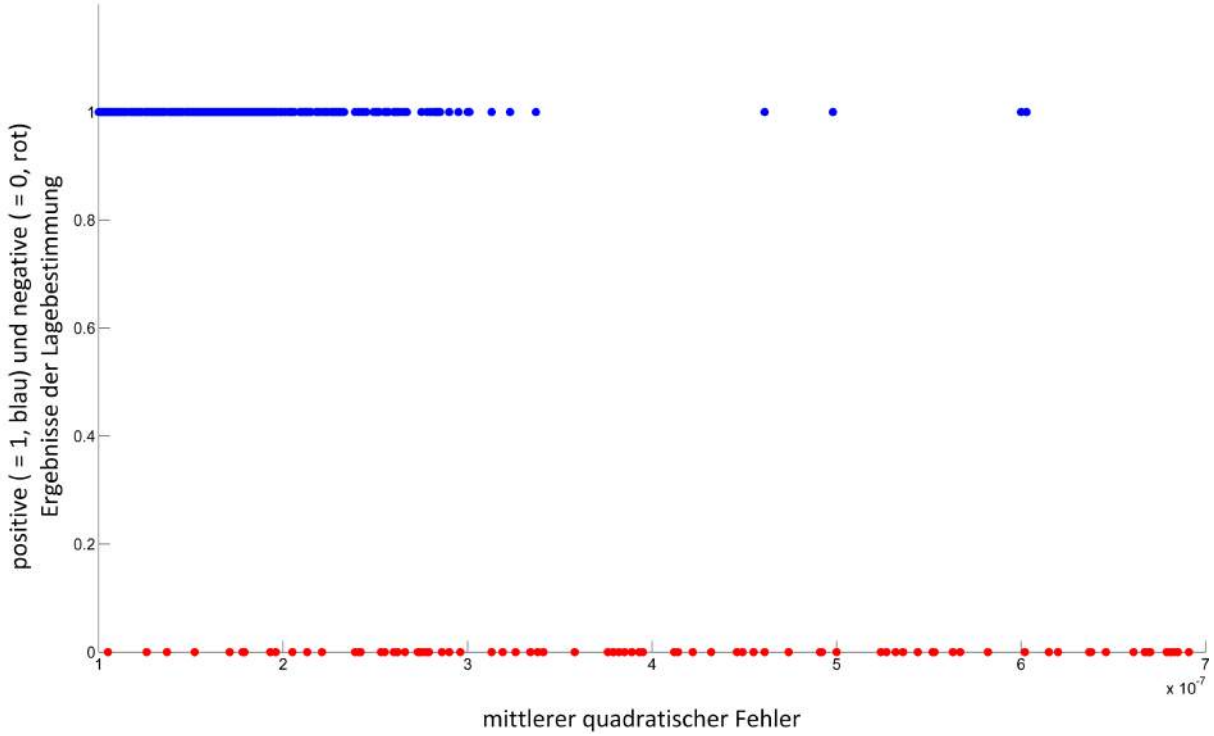


Abbildung 63: Statistik der als positiv (= 1) und negativ (= 0) bewerteten Ergebnisse des Circle-Fittings und des jeweiligen Fehlerwertes für alle getesteten Tabletten.

Für die Bestimmung des Fehlerschwellwertes wurden sämtliche Fehlerwerte sortiert und ein 90%-Konfidenzintervall bestimmt:

- 90% der akzeptablen Circle-Fits haben einen mittleren quadratischen Fehler von unter $2.51 \cdot 10^{-7}$
- 90% der unbrauchbaren Circle-Fits haben einen mittleren quadratischen Fehler von über $2.53 \cdot 10^{-7}$

Ausgehend von diesen Beobachtungen wurde der Schwellwert für den mittleren quadratischen Fehler der Datenpunkte der Tablettenoberfläche zum Kreis des Circle-Fittings mit:

$$E_{ms,max} = 2.5 \cdot 10^{-7}$$

gewählt.

Der Vorgang des Eliminierens der mit Hilfe dieses Schwellwertes vom Algorithmus als unbrauchbar eingestuften Tablettendetektionen ist in den Abbildungen 64 und 65 zu sehen.

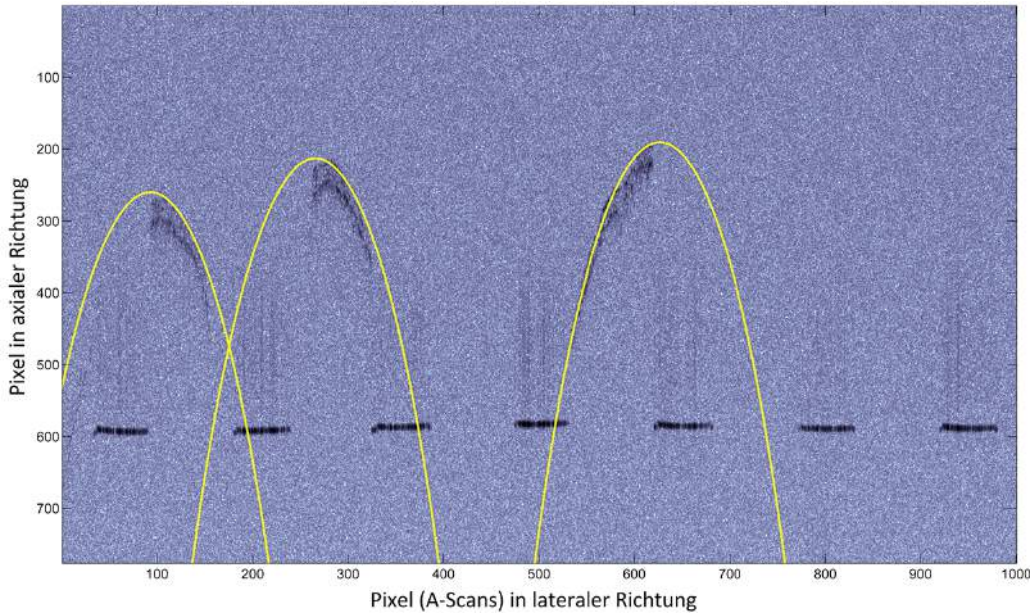


Abbildung 64: Detektierte Tabletten im B-Scan ohne Berücksichtigung des mittleren quadratischen Fehlers der gefitteten Kreise. Es ist zu erkennen, dass der Fit für die rechte der Tabletten nicht optimal passt und für weitere Berechnungen nicht verwendet werden sollte.

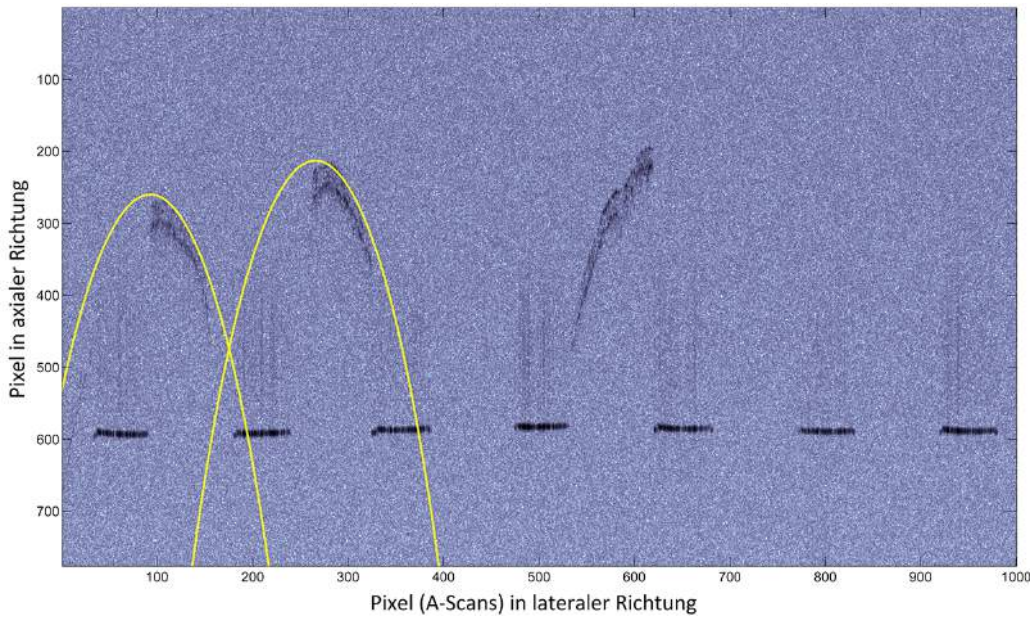


Abbildung 65: Detektierte Tabletten im B-Scan nach Überprüfung des Fehlerschwellwertes. Der Schwellwert des Fits der rechten Tablette hat den maximalen Fehler überschritten und wird daher vom Algorithmus eliminiert. Grund dafür dürfte in diesem Fall eine Deformation der Tablette sein.

6.3.2 Automatische Bestimmung des Fehlerschwellwertes

Die oben beschriebene Methode zur Bestimmung des Fehlerschwellwertes funktioniert für die in dieser Arbeit verwendeten Daten zwar, sie setzt allerdings eine manuelle, und damit subjektive Beurteilung des Fitting-Ergebnisses voraus. Eine automatische Eruierung dieses Wertes anhand von quantitativen Kriterien ist daher zu bevorzugen.

Eine Möglichkeit hierfür wäre, den mittleren quadratischen Fehler des Circle-Fittings in Zusammenhang mit dem vorhandenen Datenrauschen zu bringen. Liegt der Fehler E_{ms} in der Größenordnung (z.B. Faktor $k \leq 4$) der Standardabweichung σ_{noise} des Rauschens, so war die durchgeführte Lagebestimmung erfolgreich:

$$E_{ms.max} = k \cdot \sigma_{noise} \tag{43}$$

Eine weitere Variante zu automatischen Bestimmung des Fehlerschwellwertes bietet die Bestimmung der „intra-tablet coating uniformity“. Diese beschreibt, wie bereits in Kapitel 1.1 angeführt, die Schwankung der Beschichtungsdicke innerhalb einer einzelnen Tablette. Liegt der Fehler E_{ms} analog zur vorherigen Ausführung in der Größenordnung (z.B. Faktor $k \leq 4$) der „intra-tablet coating uniformity“ $\sigma_{cu,intra}$, so war die durchgeführte Lagebestimmung erfolgreich:

$$E_{ms.max} = k \cdot \sigma_{cu,intra} \tag{44}$$

Zum Zeitpunkt der Durchführung dieser Arbeit waren keine Informationen bezüglich des Datenrauschens vorhanden. Auf eine Implementierung eines Verfahrens zur Schätzung der Rauschparameter wurde aufgrund Zeitmangels verzichtet. Der Algorithmus zur Berechnung der Beschichtungsdicke war damals ebenfalls noch nicht ausgereift genug, um die „intra-tablet coating uniformity“ zuverlässig bestimmen zu können. Dennoch sollten die beiden hier angeführten Methoden zur automatischen Bestimmung des Fehlerschwellwertes des Circle-Fitting-Verfahrens für zukünftig erfolgende Optimierungsschritte in Betracht gezogen werden.

7 Conclusio

Das nachfolgende Kapitel fasst die gesammelten Ergebnisse und Erkenntnisse zusammen und ist in die einzelnen Abschnitte des Algorithmus unterteilt. Zusätzlich wird auf die erreichten Ziele und das Potential für zukünftige Verbesserungen eingegangen.

7.1 Datenverarbeitung

Die Aufbereitung der Daten funktioniert einwandfrei und ohne großen Rechenaufwand. Ausgehend von den zukünftigen Daten könnten neben der Subtraktion des Referenzspektrums von den Rohdaten noch weitere SNR verbessernde Schritte unternommen werden, um die Datenqualität zusätzlich zu erhöhen.

7.2 Merkmalsextraktion

Es wurde während dieser Arbeit keine Möglichkeit gefunden, die Spektraldaten direkt zu klassifizieren, da die Aussagekraft der extrahierten Merkmale nicht ausreichend war. Aufgrund dessen müssen nach derzeitigem Stand die Spektraldaten vor der Merkmalsextraktion in Bilddaten transformiert werden.
Die Umwandlung in Bilddaten kann allerdings nie vollständig umgangen werden, da diese in weiterer Folge für die Positionsbestimmung benötigt werden. Trotzdem kann mit zukünftigen, qualitativ höherwertigen Rohdaten erneut eine Merkmalsextraktion im Spektralbereich versucht werden, um dadurch eventuell eine weitere Minimierung des Rechenaufwandes erzielen zu können.

7.3 Klassifizierung

Wie bereits oben angeführt, wurde mit den verwendeten Methoden eine Klassifizierungsgenauigkeit von fast 96% erreicht. Es kann davon ausgegangen werden, dass mit besserer Datenqualität dieser Wert noch weiter verbessert werden kann. Als Grundlage für eine in-line Überwachung des Beschichtungsvorganges stellt die erzielte Genauigkeit unter Berücksichtigung des dabei nötigen Rechenaufwandes aber bereits einen zufriedenstellenden Wert dar.

7.4 Positionsbestimmung

Mit der in dieser Arbeit implementierten Methode der Positionsbestimmung werden in etwa 86% Tabletten in den Bilddaten detektiert und deren Position korrekt bestimmt. Vorraussetzung dafür ist allerdings, dass die Tabletten so unter dem Sensor liegen, dass der Tablettensteg nicht sichtbar ist. Bei Bilddaten mit aufscheinendem Steg sinkt der Wert der korrekten Detektion samt Positionsbestimmung auf rund 31%, bei gemischten Daten auf ca. 75%.
Es besteht daher akuter Verbessungsbedarf bei der Stegerkennung, sowie in weiterer Folge bei der Positionsbestimmung mit Steg.

7.5 Gesamtalgorithmus

Mit dem in dieser Arbeit beschriebenen Gesamtalgorithmus können derzeit, implementiert in MATLAB (Version 8.2, The MathWorks Inc., Natick, Massachusetts/USA), ca. 1000 A-Scans pro Sekunde verarbeitet und die entsprechende Schichtdicke der Tabletten berechnet werden. Somit wird bei einer angenommenen, durchschnittlichen Anzahl von 4 Tabletten pro aufgenommenem Bild (bestehend aus 1000 A-Scans) und der oben angegebenen Rate von 75% erfolgreichen Detektionen und Positionsbestimmungen nach derzeitigem Stand bei 3 Tabletten pro Sekunde die Schichtdicke ermittelt (die Bestimmung der Schichtdicke war nicht mehr Teil dieser Arbeit). Dieser Wert erlaubt eine in-line Überwachung des Beschichtungsvorganges.

7.6 Ausblick

Zurzeit erfolgt eine Neuimplementierung des Algorithmus in der Entwicklungsumgebung CUDA (compute unified device architecture, Version 5.5, Nvidia Corporation, Santa Clara, Kalifornien/USA), wodurch der Algorithmus auf der Grafikkarte ausgeführt werden kann. Die anfallenden Daten können daher zukünftig mit einer noch höheren Geschwindigkeit verarbeitet werden.
Notwendig wird diese schnellere Datenverarbeitung auch, weil das gesamte Messsystem derzeit überarbeitet wird. Eine Rate von bis zu 50000 A-Scans pro Sekunde bei zusätzlich höherer Datenqualität wird angestrebt.

Literatur

[1] Klaus Knop and Peter Kleinebudde. Pat-tools for process control in pharmaceutical film coating applications. International journal of pharmaceutics, 457(2):527–536, 2013.

[2] Daniele Suzzi, Gregor Toschkoff, Stefan Radl, Daniel Machold, Simon D Fraser, Benjamin J Glasser, and Johannes G Khinast. Dem simulation of continuous tablet coating: Effects of tablet shape and fill level on inter-tablet coating variability. Chemical Engineering Science, 69(1):107–121, 2012.

[3] DM Koller, G Hannesschläger, M Leitner, and JG Khinast. Non-destructive analysis of tablet coatings with optical coherence tomography. European Journal of Pharmaceutical Sciences, 44(1):142–148, 2011.

[4] S Porter, G Sackett, and L Liu. Development, optimization, and scale-up of process parameters: pan coating. In Yihong Qiu, Yisheng Chen, Geoff GZ Zhang, Lirong Liu, and William Porter, editors, Developing Solid Oral Dosage Forms-Pharmaceutical Theory and Practice., chapter 33. Academic Press, 2009.

[5] Daniele Suzzi, Stefan Radl, and Johannes G Khinast. Local analysis of the tablet coating process: Impact of operation conditions on film quality. Chemical Engineering Science, 65(21):5699–5715, 2010.

[6] David Huang, Eric A Swanson, Charles P Lin, Joel S Schuman, William G Stinson, Warren Chang, Michael R Hee, Thomas Flotte, Kenton Gregory, Carmen A Puliafito, et al. Optical coherence tomography. Science, 254(5035):1178–1181, 1991.

[7] Adrian Gh Podoleanu, John A Rogers, David A Jackson, Shane Dunne, et al. Three dimensional oct images from retina and skin. Optics Express, 7(9):292–298, 2000.

[8] Bill Colston, Ujwal Sathyam, Luiz DaSilva, Matthew Everett, Pieter Stroeve, and L Otis. Dental oct. Optics express, 3(6):230–238, 1998.

[9] Maciej Wojtkowski. High-speed optical coherence tomography: basics and applications. Applied Optics, 49(16):D30–D61, 2010.

[10] Wolfgang Drexler and James G Fujimoto. Optical coherence tomography: technology and applications. Springer, 2008.

[11] D Stifter. Beyond biomedicine: a review of alternative applications and developments for optical coherence tomography. Applied Physics B, 88(3):337–357, 2007.

[12] Daniel Markl, Günther Hannesschläger, Stephan Sacher, Johannes G Khinast, and Michael Leitner. Optical coherence tomography for non-destructive analysis of coatings in pharmaceutical tablets. In SPIE Optical Metrology 2013, pages 879202–879202. International Society for Optics and Photonics, 2013.

[13] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 1. Springer New York, 2006.

[14] Chao-Ying Joanne Peng and Rebecca Naegle Nichols. Using multinomial logistic models to predict adolescent behavioral risk. Journal of Modern Applied Statistical Methods, 2(1):1–13, 2003.

[15] Woo-Yong Hyun and Robert B Ditton. Using multinomial logistic regression analysis to understand anglers willingness to substitute other fishing locations. In Burns R, Robinson K (eds) Proceedings of the 2006 northeastern recreation research symposium, US Forest Service, Northern Research Station, Pennsylvania, USA, pages 248–255. USDA, 2006.

[16] Balaji Krishnapuram, Lawrence Carin, Mario AT Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(6):957–968, 2005.

[17] Kai-Bo Duan and S Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In Kittler Josef Roli Fabio Nikunj C. Oza, Polikar Robi, editor, Multiple Classifier Systems, pages 278–285. Springer, 2005.

[18] Jonathan Milgram, Mohamed Cheriet, Robert Sabourin, et al. „one against one“ or „one against all“: Which one is better for handwriting recognition with svms? In Tenth International Workshop on Frontiers in Handwriting Recognition, 2006.

[19] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on, 13(2):415–425, 2002.

[20] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989.

[21] Mou-Yen Chen, Amlan Kundu, and Jian Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 16(5):481–496, 1994.

[22] Anders Krogh, Björn Larsson, Gunnar Von Heijne, and Erik LL Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. Journal of molecular biology, 305(3):567–580, 2001.

[23] Michel F Valstar and Maja Pantic. Combined support vector machines and hidden markov models for modeling facial action temporal dynamics. In Abascal J. Junqueira Barbosa S.D. Baranauskas C., Palanque P., editor, Human–Computer Interaction, pages 118–127. Springer, 2007.

[24] Yasemin Altun, Ioannis Tsochantaridis, Thomas Hofmann, et al. Hidden markov support vector machines. In Proceedings of the Twentieth International Conference on Machine Learning (ICML), volume 3, pages 3–10, 2003.

[25] Xin He and Xian-Zhong Zhou. Audio classification by hybrid support vector machine/hidden markov model. World Journal of Modeling and Simulation, 1(1):56–59, 2005.

[26] J Stadermann and G Rigoll. A hybrid svm/hmm acoustic modeling approach to automatic speech recognition. In Interspeech 2004 - ICSLP, 8th International Conference on Spoken Language Processing, 2004.

[27] Wen-Han Yu, Hedda Høvik, and Tsute Chen. A hidden markov support vector machine framework incorporating profile geometry learning for identifying microbial rna in tiling array data. Bioinformatics, 26(11):1423–1430, 2010.

[28] Matt Dunham and Kevin Murphy. PMTK: probabilistic modeling toolkit (version 3). 2010. Software unter `https://github.com/probml/pmtk3`.

[29] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines (version 3.17). ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27, 2011. Software unter `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[30] Veikko Karimäki. Effective circle fitting for particle trajectories. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 305(1):187–191, 1991.

[31] Daw-Tung Lin and Chen-Ming Yang. Real-time eye detection using face-circle fitting and dark-pixel filtering. In Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on, volume 2, pages 1167–1170. IEEE, 2004.

[32] Harvey Motulsky et al. Fitting Models to Biological Data Using Linear and Nonlinear Regression: A Practical Guide to Curve Fitting: A Practical Guide to Curve Fitting. Oxford University Press, 2004.

[33] Tjalling J Ypma. Historical development of the newton-raphson method. SIAM review, 37(4):531–551, 1995.

[34] Robert WM Wedderburn. Quasi-likelihood functions, generalized linear models, and the gauss-newton method. Biometrika, 61(3):439–447, 1974.

[35] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In Watson G.A., editor, Numerical analysis, pages 105–116. Springer, 1978.

[36] I Kasa. A circle fitting procedure and its error analysis. Instrumentation and Measurement, IEEE Transactions on, 1001(1):8–14, 1976.

[37] Vaughan Pratt. Direct least-squares fitting of algebraic surfaces. In ACM SIGGRAPH Computer Graphics, volume 21, pages 145–152. ACM, 1987.

[38] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(11):1115–1138, 1991.

[39] YT Chan, BH Lee, and SM Thomas. Approximate maximum likelihood estimation of circle parameters. Journal of optimization theory and applications, 125(3):723–734, 2005.

[40] Walter Gander, Gene H Golub, and Rolf Strebel. Least-squares fitting of circles and ellipses. BIT Numerical Mathematics, 34(4):558–578, 1994.

[41] Dale Umbach and Kerry N Jones. A few methods for fitting circles to data. Instrumentation and Measurement, IEEE Transactions on, 52(6):1881–1885, 2003.

[42] Ali Al-Sharadqah, Nikolai Chernov, et al. Error analysis for circle fitting algorithms. Electronic Journal of Statistics, 3:886–911, 2009.

[43] Wei Li, Jing Zhong, T Aaron Gulliver, Bo Rong, Rose Qingyang Hu, and Yi Qian. Fitting noisy data to a circle: A simple iterative maximum likelihood approach. In Communications (ICC), 2011 IEEE International Conference on, pages 1–5. IEEE, 2011.

[44] Hajime Tamura, Takeshi Sasaki, Hideki Hashimoto, and Fumihiro Inoue. Circle fitting based position measurement system using laser range finder in construction fields. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 209–214. IEEE, 2010.

8 Anhang

Tabelle 18 stellt eine Auflistung der implementierten Funktionen inklusive einer kurzen Beschreibung dar.

Tabelle 18: Implementierte Funktionen

Bezeichnung	Beschreibung
`init.m`	Definition von Messparametern, Speicherpfaden, etc.
`getRawFiles.m`	Einlesen von Rohdaten und Referenzspektrum
`uigetFile.m`	Öffnet Standard-Dialog-Box zur Auswahl der Dateien
`generateImageData.m`	Datenaufbereitung, Bilderzeugung mittels FFT
`getFeatureVector.m`	Merkmalsextraktion aus den A-Scans der Bilddaten
`logregFit.m`	Erstellung des Klassifizierungsmodells für die logistische Regression anhand der Trainingsdaten
`logregPredict.m`	Klassifizierung der Testdaten mittels LR
`svmtrain.m`	Erstellung des Klassifizierungsmodells für Support Vector Machines anhand der Trainingsdaten
`svmpredict.m`	Klassifizierung der Testdaten mittels SVM
`getPath.m`	Verbesserung der Klassifizierung durch Berechnung des Wahrscheinlichsten Zustandpfades mit Hilfe eines Hidden Markov Modells
`viterbi_path.m`	Implementierung des Viterbi-Algorithmus
`optimizeTargetVector.m`	Optimierung des Pfades, Korrektur von Ausreißern
`optimizeInterfaceVectors.m`	Optimierung der Datenpunkte der Oberfläche der detektierten Tabletten, Erkennung von Tablettenstegen
`evaluateMeasureParam.m`	Berechnung der relativen Geschwindigkeit zwischen Messobjekten und Sensor, sowie der lateralen Ausdehnung pro A-Scan im Bildbereich
`CircleFit.m`	Ermitteln der relativen Position der einzelnen Tabletten zum Messsystem mit einem Circle-Fitting-Verfahren, Überprüfung des Fehlerschwellwertes der einzelnen Fits
`estimateCircleCenterRough.m`	Schätzen des Kreismittelpunktes als Initialwert für das Circle-Fitting-Verfahren
`CircleFitMLE.m`	Circle-Fitting mittels Maximum Likelihood Estimation
`CircleFitNewtonRaph.m`	Circle-Fitting mittels Newton-Raphson-Methode