Masterarbeit

Bildgestützte Qualitätsprüfung von Feuerbohnen

Heinz Fleischhacker

Institut für Maschinelles Sehen und Darstellen
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Franz Leberl

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Horst Bischof

Graz, im November 2010

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, im November 2010 Heinz Fleischhacker

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Sortierung von Lebensmitteln, im konkreten wird eine Lösung zur bildgestützten Sortierung von Feuerbohnen präsentiert. Da die Arbeit jedoch nicht den gesamten Bereich der dafür notwendigen Technologien abzudecken vermag, liegt der Fokus auf der Implementierung der Bildverarbeitung. Zu diesem Zweck werden zu Beginn Aufgaben der Lebensmittelsortierung erläutert, kurze Informationen zur speziellen Kultur der Feuerbohne leiten das Projekt ein. Es folgt die Vorstellung von Sortierkonzepten, aus denen die Anforderungen einer Bildverarbeitungssoftware abgeleitet werden, welche Feuerbohnen klassifizieren kann. Danach werden Grundlagen der Bildverarbeitung in den Bereichen Bildgewinnung und Technologien, Vorverarbeitung sowie Merkmalsextraktion behandelt. Weiters werden Elemente des maschinellen Lernens sowie Echtzeitsysteme eingeführt, welche für die weiteren Darstellungen ebenfalls benötigt werden. Methoden zur Erfassung der Bilddaten und dessen Repräsentation leiten im Anschluss die Vorstellung der eingesetzten Bildverarbeitung ein. Diese wird unter Berücksichtigung der Systemanforderungen umgesetzt und argumentiert. Die Bildverarbeitung ist für kontinuierliche Datenverarbeitung ausgelegt. Im Fall der Analyse von Feuerbohnen kommt diese Notwendigkeit durch die Verwendung von Zeilensensoren zu Stande. Die Techniken werden analysiert und an Hand einer konkreten Implementierung im Detail untersucht. Daraus wird die Verwendbarkeit der entwickelten Lösungen argumentiert. Abschließende Bemerkungen geben einen weiteren Ausblick.

Abstract

The presented work deals with the sorting of food. In detail, the sorting of scarlet runners is addressed using an image-based method. Since the work cannot cover all necessary topics, it focuses on the impelmentation of the image processing part. For this purpose, food sorting is introduced in general, some information about scarlet runners and project requirements is the starting point of the work. To get an overview, sorting concepts and software-related issues are discussed. This also helps to define exact specifications for the image processing system. Fundamentals in image processing are introduced afterwards, including image acquisition and technologies, preprocessing and feature extraction. Further needed subjects are machine learning and real-time systems. Methods to implement a suitable software are described and analyzed using complexity. Practical usage of the solutions is proofed via experiments to evaluate performance and timing behavior. The final chapter gives some conclusions and perspectives.

Danksagung

Diese Arbeit entstand im Zuge des Austauschs von aktuellen Erfahrungen im Rahmen der Hochschulausbildung bereits vor einigen Jahren. Immer wieder stellt man sich die Frage nach dem Sinn und dem Nutzen des Erlernten, falls man sich nicht zum Kreis der reinen Theoretiker zählen kann. Und so kam es, dass während einer abendlichen Diskussion der Startschuss für ein, wie ich später feststellen musste, äußerst umfangreiches Projekt fiel. Dass dieses bis zum heutigen Stand umgesetzt werden konnte, verdanke ich in erster Linie Franz Edler, mit welchem die Idee entstand und welcher alle Schwierigkeiten abseits von meinen Programmiertätigkeiten zu lösen versucht. Weiters entscheidend war der Wille und die Erfahrung des Instituts für Maschinelles Sehen und Darstellen (ICG) an der Technischen Universität Graz, welches mich zu Beginn für das Projekt motivierte und in allen organisatorischen und technischen Fragen unterstützte. So wurde ich in die Lage versetzt, meine Arbeiten im Rahmen einer Masterarbeit zu dokumentieren. Gerade die Organisation legt einem bei Projekten diesen Ausmaßes immer wieder große Steine in den Weg. Vordergründig danke ich meinem Betreuer Prof. Horst Bischof, der sich wohl von Anfang an über den Aufwand, der mich erwartete, im Klaren war. Ich bin froh, dass es mir nicht klar war! Weiters danke ich Dr. Mathias Ruether, dem Sekretariat des ICG sowie allen weiteren involvierten Personen. Am Ende sei noch Hrn. Anton Edler für seine organisatorische Unterstützung gedankt. Ich hoffe, mit einem erfolgreichen Abschluss dieses Projekts etwas zurückgeben zu können.

Inhaltsverzeichnis

1 Einleitung **11**
- 1.1 Sortierung von Lebensmitteln . . . 11
- 1.2 Feuerbohnen . . . 13
- 1.3 Sortierung von Feuerbohnen - Projektentstehung . . . 14
- 1.4 Ziele und Inhalte der Arbeit . . . 15

2 Aufbau einer Sortierung **18**
- 2.1 Komponenten einer Sortiermaschine . . . 18
  - 2.1.1 Anforderungen an das Gesamtsystem . . . 20
- 2.2 Optische Hardware . . . 21
- 2.3 Bildverarbeitung für Sortieranwendungen . . . 22

3 Grundkonzepte **25**
- 3.1 Prinzipien der Bildaufnahme . . . 26
  - 3.1.1 Das Rasterbild . . . 26
  - 3.1.2 Bildgebende Sensoren und Peripherie . . . 28
  - 3.1.3 Dimensionierung optischer Systeme . . . 33
- 3.2 Vom Rasterbild zum Featurevektor . . . 35
  - 3.2.1 Farbräume . . . 36
  - 3.2.2 Operationen der Bildvorverarbeitung . . . 39
  - 3.2.3 Objektsegmentierung und Labeling . . . 45
  - 3.2.4 Berechnung von Objektmerkmalen . . . 46
  - 3.2.5 Auswahlverfahren . . . 50
- 3.3 Maschinelles Lernen für Sortieranwendungen . . . 52
  - 3.3.1 Grundlagen des maschinellen Lernens . . . 53
  - 3.3.2 Überwachtes Lernen . . . 55
  - 3.3.3 Unüberwachtes Lernen . . . 61
  - 3.3.4 Klassifikatoren für die Sortierpraxis . . . 62
- 3.4 Echtzeitsysteme . . . 62
  - 3.4.1 Echtzeit Grundlagen . . . 63
  - 3.4.2 Echtzeit Design . . . 65

4 Aufnahme und Repräsentation von Feuerbohnen **67**
- 4.1 Aufnahmesetup . . . 67
  - 4.1.1 Sensorwahl . . . 67
  - 4.1.2 Optische Vorgaben . . . 68

4.1.3 Beleuchtungskonfiguration . . . 68
4.1.4 Bilddatentransport . . . 69
4.2 Repräsentation der Bilddaten . . . 69
4.2.1 Bildpuffer . . . 69
4.2.2 Objekte an Puffergrenzen . . . 70
4.2.3 Benötigte Datenstrukturen . . . 72
4.2.4 Überlappende Objekte . . . 74

5 Klassifikation von Feuerbohnen 75
5.1 Vorverarbeitung . . . 75
5.1.1 Segmentierung der Feuerbohnen . . . 75
5.1.2 Objektlabeling . . . 76
5.1.3 Bewertung der Vorverarbeitung . . . 78
5.2 Berechnung des Merkmalvektors $\vec{f}$ . . . 78
5.2.1 Komponenten von $\vec{f}$ . . . 78
5.2.2 Berechnung von $\vec{f}$ . . . 80
5.2.3 Die Rolle des Flächenschwerpunkts . . . 83
5.2.4 Bewertung der Merkmalsextraktion . . . 84
5.3 Erkennen positiver Proben . . . 85
5.3.1 Lernen des Klassifikators . . . 85
5.3.2 Auswahl des Klassifikators . . . 86
5.3.3 Ablauf der Klassifikation . . . 87
5.4 Steuerung der Sortierung . . . 87

6 Interpretation der Ergebnisse 89
6.1 Der Evaluierungsvorgang . . . 89
6.2 Sortierperformance . . . 91
6.2.1 Evaluierung mit einzelnen Merkmalen . . . 91
6.2.2 Evaluierung des gesamten Merkmalvektors $\vec{f}$ . . . 94
6.2.3 Bemerkungen zur Evaluierung . . . 95
6.3 Analyse der zeitlichen Deterministik . . . 95
6.3.1 Bearbeitungszeit einzelner Bildpuffer . . . 96
6.3.2 Bearbeitungszeiten im Sortierbetrieb . . . 96
6.3.3 Sortierdurchsatz . . . 97
6.4 Bemerkungen zur Praxistauglichkeit . . . 98

7 Schlusswort 102

A Fallverhalten von Feuerbohnen 103
A.1 Aufnahme-Setup und Kalibrierung . . . 103
A.2 Berechnung der Fallzeiten . . . 105

B Wichtige Systemkomponenten 108
B.1 Hardware . . . 108
B.2 Software . . . 108
B.3 Testparameter . . . 109

Literaturverzeichnis **110**

Abbildungsverzeichnis

1.1 Schema einer Kleingutsortierung 12
1.2 Eigenschaften von Sortiergütern 13
1.3 Feuerbohne 14

2.1 Sortiermaschine - Probenfluss 19
2.2 Sortiermaschine - Aufnahmebereich 20
2.3 Positive Proben von Feuerbohnen 24

3.1 Elektromagnetisches Spektrum 27
3.2 Prinzip von Zeilensensoren 30
3.3 Zeilensensor mit Farbmosaik 31
3.4 Optisches Setup 34
3.5 RGB Farbraum 37
3.6 HSV Farbraum 38
3.7 Anwendung von Filtern 41
3.8 Binäre Morphologie 45
3.9 Freeman Code 48
3.10 Merkmalauswahl 51
3.11 Separation von Datensätzen 57
3.12 SVM: Berechnung von Separationsebenen 59
3.13 Echtzeitsoftware 63
3.14 Asymptotische Schranken 64

4.1 Bildpuffer - Beispiel 69
4.2 Repräsentation der Szene durch Bildpuffer 71
4.3 Objekte an Puffergrenzen 72

5.1 Klassifikation von Feuerbohnen 76
5.2 Pufferlabeling - Fallunterscheidung 77
5.3 Fehlstellen auf Feuerbohnen 79
5.4 Fehlinformation der M-Matrix 82
5.5 Aktualisierung des Merkmalvektors 83
5.6 Referenzierung des Flächenschwerpunkts 84

6.1 Verteilung der $\vec{c}$ Vektoren 90
6.2 Merkmale - Farbverteilung 92
6.3 Zeitverhalten der Callback-Funktionen 99

6.4 Zeitparameter - Sortierbetrieb 100
6.5 Rechnerauslastung im Sortierbetrieb 101

A.1 Kalibrierreferenzen 104
A.2 Freifallszene von Feuerbohnen 106
A.3 Fallzeiten von Feuerbohnen 107

Tabellenverzeichnis

1.1 Funktionsgruppen einer Lebensmittelsortierung 16

3.1 Technologien bildgebender Sensoren 28

6.1 Sortierperformance - Flächeninhalt 93
6.2 Sortierperformance - Kontur 93
6.3 Sortierperformance - Globales Farbmittel 94
6.4 Sortierperformance - Grenzbereichsfarben 94
6.5 Sortierperformance - Fokusmaß 95
6.6 Sortierperformance - Gesamter Merkmalvektor 95
6.7 Sortierdurchsatz 97

Kapitel 1

Einleitung

Die Produktion von Lebensmitteln ist, wie viele andere Produktionssparten, mehr und mehr der Industrialisierung und Globalisierung unterworfen. Der Wettbewerb fordert von Betrieben aller Größen (in Bezug auf deren Wirtschaftsleistung) eine Steigerung der Produktionseffizienz, um dem internationalen Preisdruck standhalten zu können. Kleinbetriebe sind davon speziell betroffen, können sie doch stark schwankende Preise mangels fehlender Ressourcen nicht auf längere Zeit kompensieren, wie dies während längerer Durststrecken notwendig wäre. Trotz der harten Realität der freien Marktwirtschaft schaffen sich viele Unternehmen ihre Daseinsberechtigung. Dies gelingt ihnen zum Einen durch die Erschließung von Marktnischen, zum Anderen durch effiziente Produktion. Letzeres wird von technischer Seite aus durch den Einsatz moderner und auch zuverlässiger Technologien erreicht[1]. Die Mechanisierung der Lebensmittelproduktion geht in deren Automatisierung über. Einer der letzten Schritte der Produktionskette stellt die Quailtätskontrolle dar, die sicherstellen muss, dass keine Ausschußware an die Konsumenten weitergereicht wird[2]. Man spricht in diesem Zusammenhang häufig von Sortierung, da die Qualitätsprüfung im Normalfall gleichzeitig die mindere Ware von der Verkaufsware trennt. In dieser Arbeit wird in weiterer Folge ebenso dieser Begriff verwendet, was jedoch den Titel nicht beeinflussen soll, da sie die Anaylse der Ware in den Vordergrund stellt.

1.1 Sortierung von Lebensmitteln

Der größte Anteil von Grundnahrungsmitteln wird heute (noch!) von landwirtschaftlichen Gewerbebetrieben erzeugt. Ackerfrüchte wie Getreide oder Gemüse werden vor deren Weiterverwendung bzw. Verkauf gereinigt und von Fremdkörpern, beschäditger oder minder qualitativer Ware getrennt. Mit welcher Genauigkeit hier vorzugehen ist, hängt stark von der Art der Verwendung (z.B. Mensch oder Tier), sowie den bereits erwähnten Richtlinien ab. Je höher die geforderte Genauigkeit, desto größer ist naturgemäß der Sortieraufwand

[1] Dieser Aspekt ist natürlich nur einer unter vielen! Innovative Produkte, überregionale Vermarktung, ökonomische Betriebsführung usw. tragen ebenso dazu bei. Nicht zu unterschätzen sind auch Subventionen, welche von Staaten bzw. Wirtschaftsbündnissen vergeben werden.

[2] Es sollte in diesem Zusammenhang beachtet werden, wie der Begriff Ausschuß zu verstehen ist. Es gibt per Gesetz vorgeschriebene Qualitätsstandards für jedes Produkt, welche einzuhalten sind. Darüber hinaus werden oft noch weitere, freiwillige, Standards festgelegt um den Wert des Produkts zu steigern (Beispiel: biologische Landwirtschaft).

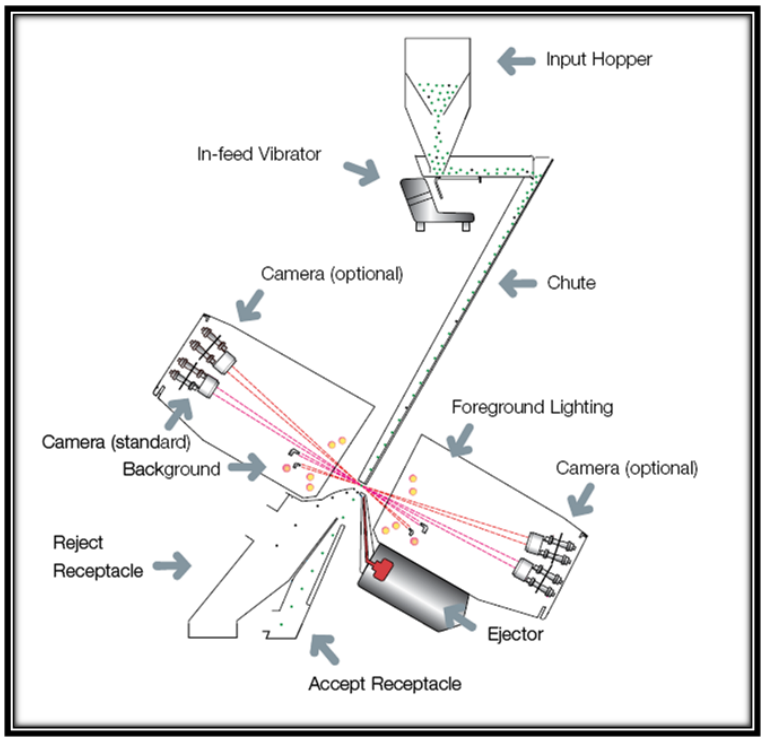

Abbildung 1.1: **Prinzip von Kleingutsortierungen;** Aus einem Behälter wird das Gut übernommen und mittels einer geeigneten mechanischen Struktur verteilt. Die Proben gehen danach in den freien Fall über und werden von einem bzw. mehreren Sensoren erfasst. Nach einer Positiv/Negativ Entscheidung werden die einzelnen Proben getrennt. Hierfür werden z.B. pneumatische Auswerfer eingesetzt. Die Abbildung wurde aus [Bue07] entnommen.

anzusehen[3]. Das traditionelle Handsortieren der Güter wird immer unwirtschaftlicher, da Personalkosten steigen sowie straffe Standards in dieser Weise oft nur schwer zu erfüllen sind. Manuelles Sortieren stellt sich als äußerst monotone Fließbandarbeit dar, die auf der anderen Seite jedoch große Durchsätze zu tragen hat.

Mit dem Aufkommen der Halbleitertechnologie wurden in der zweiten Hälfte des 20. Jahrhunderts erstmals Möglichkeiten geschaffen, diesen Arbeitsschritt zu automatisieren. Man benötigt dazu einen Sensor, welcher gewisse physikalische Größen einer Ware feststellen kann, sowie einen Computer zum Vergleich der Messwerte. Unterscheiden sich diese Größen bei den zu trennenden Produktklassen entsprechend, ist es einer geeigneten mechanischen Struktur möglich, die Waren aufzutrennen. Als Sensor ist jeder beliebige Meßaufnehmer denkbar. Da die Lebensmittel zu früheren Zeiten per Hand sortiert wurden, ist die Verwendung von bildgebenden Sensoren äusserst naheliegend. Bereits in den Achziger Jahren wurden entsprechende Maschinen hergestellt. Diese generierten Aufnahmen von der gesamten Ware, welche im Computer zur Unterscheidung analysiert wurden. Obwohl die damaligen technischen Möglichkeiten im Vergleich zu 2009 noch recht bescheiden waren,

[3]Man bedenke, dass sich durch globalen Einsatz der Güter auch die Standards ständig ändern bzw. verstärken können.

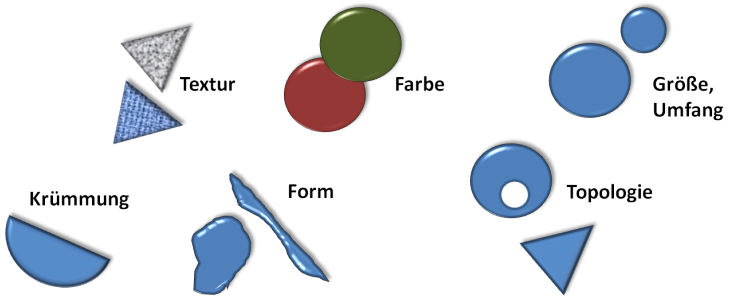

Abbildung 1.2: **Typische Sortierkriterien;** Visuelle Merkmale kommen in der industriellen Bildverarbeitung bevorzugt zum Einsatz. Deren Berechnung ist meist sehr effizient möglich. Ein Auszug möglicher Merkmale wird hier dargestellt. Für speziellere Sortieraufgaben kommen jedoch auch häufig weniger intuitive Merkmale zum Einsatz.

hat sich das Grundprinzip der Sortierung seit damals nicht geändert.
Moderne Sortiermaschinen trennen die verschiedensten Lebensmittel mit hohem Durchsatz in Ausschuß- und Verkaufsware auf. Werden Produkte natürlichen Ursprungs klassifiziert, beinhaltet die Software dieser Maschinen Lernalgorithmen, welche sich dieses Klassifikationsproblem antrainieren können. Für eine solche Problemstellung ist dieser Ansatz einzig sinnvoll, gibt es doch keinerlei Schablonen für Objekte der Natur, welche eindeutige Zuordnungen ermöglichen würden. Da die Lernalgorithmen nicht zwischen der Art der Güter unterscheiden, sondern einzig Daten verarbeiten, sind höchst flexible Anwendungen realisierbar. Ein und dieselbe Softare ist prinzipiell im Stande, jegliche Art von Gütern zu sortieren, sofern diese durch die berechneten Merkmale unterscheidbar sind. Die hohe Flexibilität wird jedoch durch eine geringere Anpassungsfähigkeit erkauft, welche für die Sortierung von Gütern mit spezielleren oder breiter gestreuten Merkmalen notwendig ist. Dies trifft auf die im nächsten Abschnitt beschriebene Feuerbohne zu. Abbildung 1.1 erläutert die Prinzipien einer Sortierung, Abbildung 1.2 zeigt typische Merkmale, welche von Proben extrahiert werden können.

1.2 Feuerbohnen

Die Feuerbohne (bot. *Phaseolus coccineus*, auch Prunkbohne oder Schminkbohne) ist eine einjährige Hülsenfrucht, die in Österreich fast ausschließlich unter dem Namen Käferbohne bekannt ist. Diese Bohnenart ist ursprünglich südamerikanischer Abstammung, wurde im 17. Jahrhundert nach Europa importiert und ab dieser Zeit auch hier angebaut. Während sie damals auf Grund hoher Krankheitsresistenzen und Robustheit sehr großflächig angebaut wurde, ist ihre Bedeutung im Laufe der Zeit wieder gesunken. Genauere Informationen sowohl zur geschichtlichen Entwicklung als auch zu den Eigenschaften der Feuerbohne finden sich in [Fru62]. In Europa ist die Feuerbohne heute hauptsächlich als Zierpflanze bekannt. Es gibt jedoch vereinzelte Anbaugebiete wie zum Beispiel in der österreichischen Südoststeiermark. Daten bzgl. Anbaumengen und -regionen sind nur äussert spärlich vorhanden, da die statistischen Ämter Produkte der selben Gattung (in diesem Fall Hülsenfrüchte bzw Körnerleguminosen) zusammenfassen. Die Anbaufläche in Österreich beträgt nur einige wenige hundert ha (siehe [uML07]), was eine der kleinsten Anbauflächen im

Abbildung 1.3: **Feuerbohne;** (a) zeigt eine Pflanze im Wachstumsstadium, welche nicht hochwachsend ist. Das äußerst verschiedenartige Aussehen von Feuerbohnen wird in (b) deutlich.

Vergleich zu anderen Hülsenfrüchten bedeutet. Eine Nachfrage ist dennoch vorhanden, was auf eine entsprechende Vermarktung als hochqualitatives Produkt zurückzuführen ist. Weiters trägt das variantenreiche Aussehen der Bohne zu ihrer Beliebtheit bei (das Auge isst bekanntlich mit). Da die Feuerbohne (wie viele andere Arten) in den meisten Fällen hochwachsend ist, werden zur Wachstumsunterstützung Stangen oder auch Hilfspflanzen wie Mais eingesetzt. Es gibt jedoch auch Kulturen, welche ohne diese Hilfsmittel auskommen. Anbautechniken und Wachstumsverhalten können in [Koo24] nachgelesen werden. Eine Darstellung der Feuerbohne ist in Abbildung 1.3 gegeben.

1.3 Sortierung von Feuerbohnen - Projektentstehung

Wie bereits angedeutet, werden die erzeugten Lebensmittel einer Qualitätsprüfung unterzogen. Dies gilt selbstverständlich auch für die Feuerbohne. Im Gegensatz zu vielen anderen Produkten wird die Feuerbohne noch heute händisch sortiert[4]. Produzenten von Feuerbohnen führten Versuche mit vorhandenen Sortiermaschinen durch, welche dem heutigen Stand der Technik entsprechen. Der Sortiermaschine wurden Proben verschiedener Anbauflächen zum Trainieren des Klassifikators zugeführt und die erreichbare Sortierperformance evaluiert. Dabei konnten Genauigkeiten von 70% erreicht werden, d. h. in der Verkaufsmenge befanden sich 30% Ausschußware und vice versa. Diese Ergebnisse zeigen, dass die eingesetzten Lernalgorithmen sich nicht im benötigen Ausmaß auf diese Bohnenart einzustellen vermochten. Dies kann mehrere Gründe haben[5]. Die in Abschnitt 1.1 angesprochene Inflexibilität kann ausschlaggebend sein, da eben jegliche Lebensmittel (sofern

[4]Diese Aussge ist nur zum Teil richtig. Die Feuerbohne wird zwar bereits der automatisierten Sortierung zugeführt, die verfügbare Sortiergenauigkeit (in weiterer Folge auch als Sortierperformance bezeichnet) erfordert allerdings eine händische Nachsortierung.

[5]Hier wird mit Absicht von möglichen Gründen gesprochen, da diese Angaben nicht durch Dokumente belegt werden können. Die Sortiertechnologien und das ihnen zugrunde liegende Wissen ist zum Teil sehr speziell und somit nicht öffentlich einsehbar.

sie bzgl. deren Abmessungen handhabbar sind) mit der selben Software analysiert werden. Weiters weisen gewisse, zu entfernende Typen von Feuerbohnen (wie z. B. vertrocknete Bohnen), ein den positiven Proben sehr ähnliches Aussehen auf. Ein breites Spektrum der vorkommenden Farben ist ein weiterer Schwierigkeitsfaktor. Diese Probleme können umgangen werden, wenn zusätzlich zu den bildgebenden Sensoren im visuellen Spektrum $(380 - 700nm)$ weitere Sensoren verwendet werden. Ein Beispiel: Thermographische Sensoren mit 2-dimensionalen Detektoren sind in der Lage, vertrocknete Proben zu erkennen, da diese die von aussen angreifende Temperaturstrahlung in geringerem Maße absorbieren. Nachteilig dabei wäre jedoch eine starke Erhöhung des Preises der gesamten Sortierung. Da also eine brauchbare Lösung bis dato nicht vorhanden war, wurde die Idee der Entwicklung einer Sortiereinrichtung, welche ausschließlich mit Sensoren des visuellen Spektrums diese Schwierigkeiten bewältigen kann, aufgegriffen. Das so entstandene Projekt hat sich zum Ziel gesetzt, einen praxistauglichen Prototypen zu realisieren, welcher unmittelbar danach auch zum Einsatz kommt. Dabei sollen neben der vorausgesetzten Funktion noch weitere Nachteile marktüblicher Maschinen ausgemertzt werden. Dies sind deren hohe Kosten, die teilweise durch Monopolstellungen der Firmen und auch durch die Dimensionierung der Maschinen selbst entstehen. Sortiermaschinen sind im Allgemeinen für höchste Durschsätze ausgelegt. Für Betriebe mit geringeren Produktchargen werden keine kleineren Varianten zur Verfügung geststellt. Dies ist aber von der technischen Seite heute durchaus machbar, da die einst noch astronomisch teure Hardware[6] schon erschwinglich geworden ist. Es sollen nachfolgend die Kerninteressen des Projekts zusammengefasst werden:

- Aufbau einer voll funktionstüchtigen Anlage zur Sortierung von Feuerbohnen
- Genauigkeit der Sortierung von positiven und negativen Proben von zumindest 95% (d.h. maximal 5% Ausschuss in der Verkaufsmenge und maximal 5% Verkaufsware in der Ausschußmenge)
- Auslegung der Anlage für kleinere Durchsätze, ca. 400 kg je Stunde
- Verwendung von kostengünstigen Komponenten, um den Gesamtpreis und somit die Amortisationsdauer gering zu halten

Die bisherigen Erläuterungen machen durchaus deutlich, dass es sich bei Sortiermaschinen um komplexe Systeme mit Funktionsgruppen aus verschiedenen Ingenieursdisziplinen handelt. Nur ein einwandfreies Zusammenspiel der Komponenten erlaubt ein robust arbeitendes Gesamtsystem. Es sollen nun jene Bereiche herausgearbeitet werden, die diese Arbeit abzudecken versucht.

1.4 Ziele und Inhalte der Arbeit

Die wichtigsten Funktionsblöcke der Sortierung sind in Tabelle 1.1 zusammengefasst. Darin wird auch der Fokus dieser Arbeit hervorgehoben. Die Bildverarbeitung ist als das Kern-

[6] Dazu zählt hauptsächlich leistungsfähige Rechnerhardware. Weiters können Beleuchtung und Zeilensensoren, aber auch pneumatische Komponenten, wie Schnellschaltwentile, Düsen und Luftreinigungssysteme genannt werden.

Komponente	Funktionen
Mechanischer Aufbau	Transport der Ware durch die Anlage: Vorreinigung, Vorsortierung, Vereinzelung der Ware;
Pneumatischer Kreislauf	Versorgung aller pneumatischen Komponenten wie Auswerfer und druckluftbasierte Reinigungssysteme
Optisches System	Komponenten der Bildgewinnung: Beleuchtung, Sensoren, Optische Bauelemente;
Bildverarbeitung	Software zur Datenverwaltung, Verarbeitung und Analyse; Ergebnis: eine positiv/negativ Entscheidung je Probe;
Auswerfer	Düsenmatrix zur Entfernung einzelner Proben;
Steuerung und Interface	Ausführung, Überwachung und Manipulation sämtlicher automatisierter Prozesse: Ansteuerung der Auswerfer, Prozessüberwachung, Bildschirmausgaben usw.;

Tabelle 1.1: Funktionsgruppen einer Sortierung

element der gesamten Sortierung anzusehen. Aus den der Software übergebenen Bilddaten sollen die einzelnen Proben extrahiert, deren Merkmale berechnet und diese einem geeigneten Klassifikator zugeführt werden. Dieser dient der Entscheidungsfindung, um für jede einzelne Probe abzuschätzen, ob sie zu entfernen ist oder nicht. Die konkrete Realisierung dieser Software wird in dieser Arbeit eingehend erläutert. Dabei soll immer wieder ein Bezug zum Gesamtsystem hergestellt werden. Dies erleichtert nicht nur das Verständnis der Zusammenhänge, sondern dient auch oftmals dazu, gewisse Systementscheidungen zu argumentieren. Man hat nämlich bei der Entwicklung von Lösungen immer darauf zu achten, dass die Software den späteren Ansprüchen, wie Integration in das Gesamtsystem, Einhaltung von Maschinenparametern uvm. genügt. Nebenbei werden auch einige Blicke auf das optische System gerichtet, da dieses die Bildverarbeitung erst ermöglicht und sie auch stark beeinflusst.
Kapitel 2 gibt eine Einführung in die Funktionsweise von Sortiermaschinen. Auf die Beschreibung der jeweiligen Funktionsgruppen folgt die Auflistung der konkreten, in diesem Projekt zu realisierenden Anforderungen. Abschnitt 2.1 zeigt den Gesamtaufbau einer Sortierung, Kapitel 2.2 wirft einen Blick auf das optische System. Im abschließenden Kapitel 2.3 werden die Aufgaben einer Software zur Sortierung abgesteckt.
Nach der Festlegung der Parameter des Gesamtsystems wird der Fokus auf die Bildverarbeitung gerichtet. Dessen Realisierung bedarf vieler grundlegender Kenntnisse, welche in Kapitel 3 besprochen werden. Abschnitt 3.1 beschreibt Konzepte der Bildgewinnung, die als Basis für jede Bildverarbeitung notwendig sind. Dabei werden hauptsächlich im Rahmen dieser Arbeit benötigte Themen im Gesamtkontext behandelt. Dies gilt übrigens auch für alle weiteren Abschnitte in Kapitel 3. In Abschnitt 3.2 werden Grundlagen zur Bildvorverarbeitung, Segmentierung, Merkmalen und deren Extraktion diskutiert. Wiederum wird nur ein sinnvoller Kontext verwendet, da die Methoden und Techniken äußerst vielfältig sind. Abschnitt 3.3 stellt diverse Lerntechniken vor und vergleicht diese bzgl. ihrer Verwendbarkeit zur Sortierung. Im letzten Teil dieses Kapitels, Abschnitt 3.4, sollen Grundbegriffe für Echtzeitsysteme behandelt werden. Diese Konzepte sind von fundamentaler Bedeutung für die Sortierung, da der Praxisbetrieb nur mit Hilfe einer Echtzeitsoftware

möglich ist. Weiters sind bei der Erstellung einer Implementierung zu jedem Zeitpunkt die Echtzeitanforderungen zu beachten, da sich diese auf die Wahl von Algorithmen, Datenmengen usw. auswirken.
Nachdem das Basiswissen sowie die notwendigen Parameter vermittelt wurden, wird in Kapitel 4 erklährt, wie die Bildaufnahme durchgeführt werden kann. Abschnitt 4.1 stellt ein Aufnahmesetup vor, danach gibt Abschnitt 4.2 eine geeignete Repräsentation für die Bilddaten an. Die für die folgende Bildverarbeitung benötigten Datanstrukturen werden eingeführt.
Methoden zur Implementierung der Bildverarbeitung finden sich in Kapitel 5 im Detail erläutert. Auf die Bildvorverarbeitungstechniken in Abschnitt 5.1 folgt die Festlegung und Berechnung des Merkmalvektors in Abschnitt 5.2. Die Auswahl des Klassifikators sowie der Ablauf des Lernens wird in Abschnitt 5.3 behandelt. Am Ende dieses Kapitels wird auf die Weiterverwendung der Klassifikationsergebnisse eingegangen (Abschnitt 5.4).
Anhand einer konkreten Implementierung werden die erzielten Ergebnisse in Kapitel 6 präsentiert. Um das System auf Praxistauglichkeit zu evaluieren, müssen geeignete Testdaten und eine entsprechende Testumgebung verwendet werden. Damit kann in Abschnitt 6.2 die Funktion, Genauigkeit und Robustheit der Software argumentiert werden. Dem Zeitverhalten wird auf Grund dessen Wichtigkeit ein eigener Abschnitt gewidmet. Schlussfolgerungen für den Betrieb der Sortiersoftware in der Praxis runden dieses Kapitel ab. Einige zusammenfassende Bemerkungen der Arbeit, Schlussfolgerungen, offene Probleme sowie Ausblicke werden in Kapitel 7 gegeben.

Kapitel 2

Aufbau einer Sortierung

Der Betrieb einer Sortiermaschine stellt sich in der Realität als ein komplexes Zusammenspiel diverser Funktionsgruppen dar. Obwohl das Softwaresystem rund um die Bildverarbeitung den Kern einer solchen Anlage ausmacht, spielen Auswurfeinheit, der mechanische Aufbau sowie die Steuerelektronik eine wesentliche Rolle. Eine passende Software kann demnach nur entwickelt werden, wenn bei der Planung die Anforderungen aller übrigen Komponenten bedacht werden. Bevor also in den folgenden Abschnitten im Detail auf die Bildverarbeitung eingegangen wird, soll hier eine Übersicht des Gesamtsystems gegeben werden. Nach einem kurzen Blick auf den generellen Aufbau solcher Systeme werden globale Anforderungen zur Sortierung von Feuerbohnen aufgelistet. Darauf folgend werden die Aufgaben der optischen Hardware diskutiert. Die praktischen Anforderungen an die Bildverarbeitung schließen dieses Kapitel ab. Sie dienen als Grundlage zur weiteren Argumentation der verwendeten Techniken und verdeutlichen die Position der Bildverarbeitung im Gesamtsystem.

2.1 Komponenten einer Sortiermaschine

In der Einführung wurden die Kernanforderungen der Sortierung aufgelistet: Durchsatz, Kosteneffizienz und Sortiergenauigkeit; Diese Punkte zu erfüllen bedarf einer genaueren Spezifizierung. Abbildung 2.1 zeigt mittels einer 3-dimensionalen Darstellung beispielhaft, welche mechanischen, pneumatischen und optischen Komponenten ausgeführt werden müssen. Die verschiedenen Funktionsgruppen sind dabei farblich zu unterscheiden. Über einen Zubringer und Vibrorinnen werden die Proben zur optischen Einheit transportiert, wo die Bildaufnahme während des freien Falls der Proben stattfindet. Eine Vorreinigung entfernt Kleinteile. Pneumatische Auswerfer (Düsen) dienen der Umleitung der Flugbahn der Proben auf verschiedene Rutschbleche. Die gezeigte Anlage ist nur für binäre Sortierung geeignet, also z. B. zur Aufteilung der Proben in eine Verkaufs- und eine Ausschußmenge (gelbes und grünes Blech in Abbildung 2.1). Ein detailierterer Ausschnitt eines solchen Systems findet sich in Abbildung 2.2. Sie zeigt den Bereich, in dem jedes Objekt von mehreren Sensoren erfasst wird. Man erkennt weiters die Montierung der Beleuchtung und die Position der darunterliegenden pneumatischen Auswerfer, die einen gewissen Fallweg für die Proben festlegen.

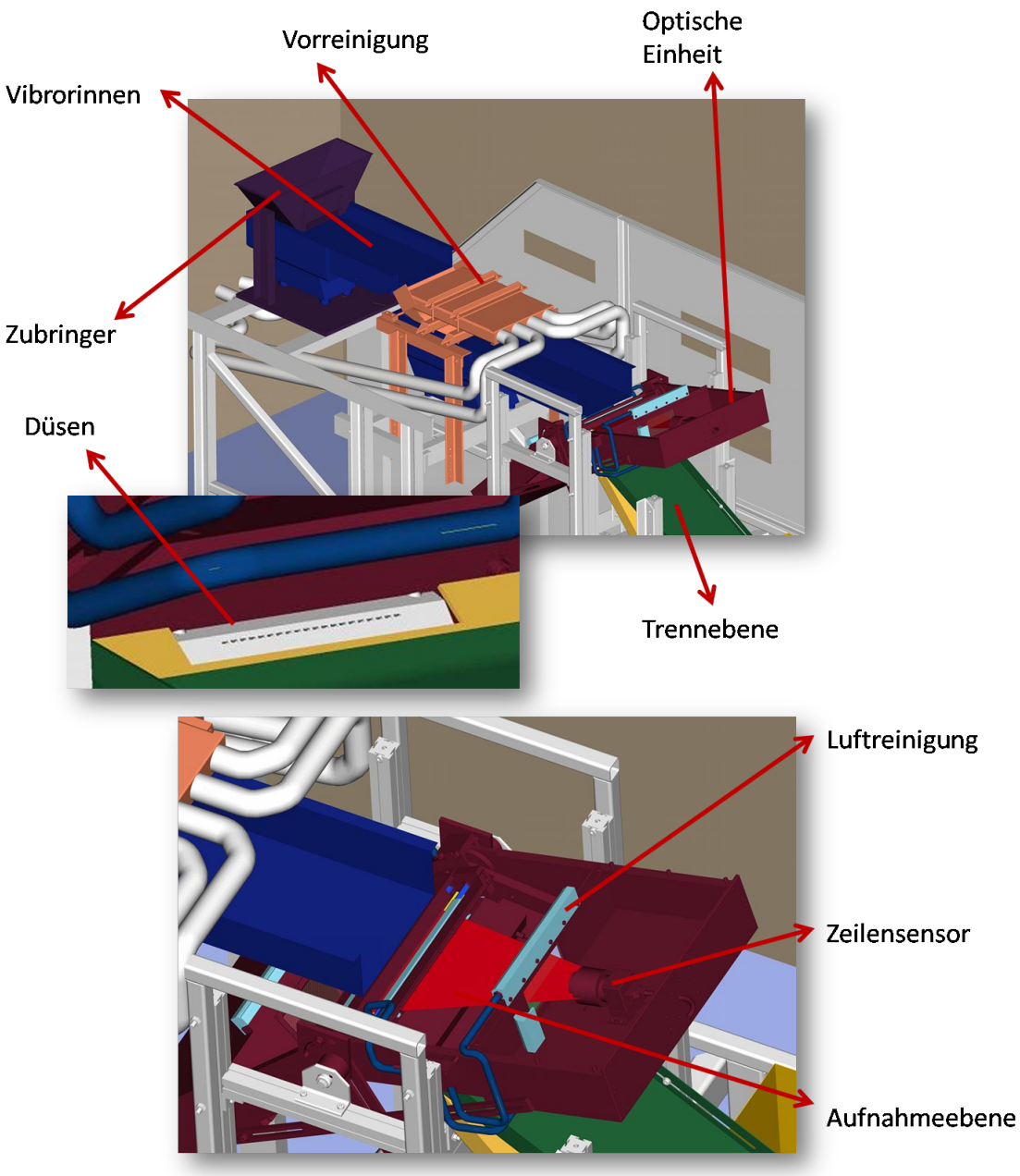

Abbildung 2.1: **Probenfluss durch die Sortierung;** Aus einem Vorbehälter gelangen die Proben auf Vibrorinnen (blau), die zur räumlichen Aufteilung benötigt werden. Weiters erfolgt eine Vorreinigung zur Entfernung von Kleinteilen durch ein Sauggebläse (orange). Die optische Einheit (dunkelrot) erfasst die Proben auf zwei Seiten und übermittelt die Bilddaten an einen Rechner. Unter der optischen Einheit befinden sich die Trennebenen (grün und gelb), auf welchen die Proben durch eine fein gerasterte Düsenleiste aufgeteilt werden. Die optische Einheit ist mit einer eigenen Reinigung ausgestattet, um den Aufnahmebereich sauber zu halten.

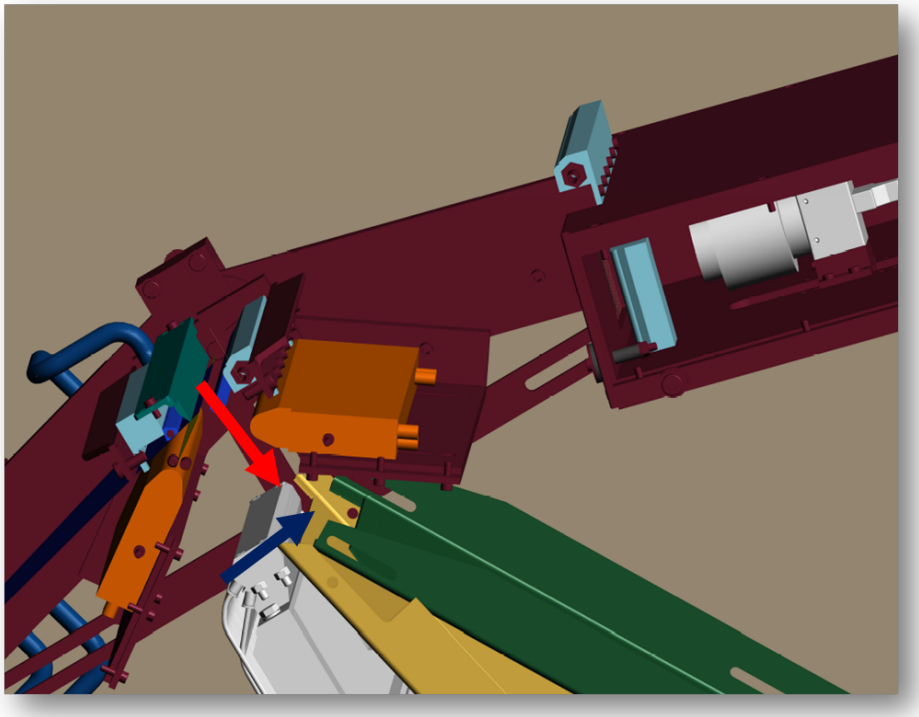

Abbildung 2.2: **Aufnahmebereich der Sortierung;** Die Proben erreichen nach Transport über die Vibrorinnen die Fallkante (dunkelgrün). Dort gehen sie in den freien Fall über (Weg durch den roten Pfeil angedeutet) und erreichen das innere Rutschblech (gelb). Wird eine Probe durch die Auswerfer (grau) in Richtung des blauen Pfeils umgelenkt, so rutscht sie über das äußere Rutschblech (grün) ab. Im Bereich zwischen der Fallkante und den Rutschblechen erfolgt die Bildaufnahme. Die Beleuchtung (orange) ist entsprechend der verwendeten Bildsensoren zu wählen.

2.1.1 Anforderungen an das Gesamtsystem

Die im Kapitel 1 angegebenen Forderungen[1] sind ein Durchsatz von $400kg$ je Stunde, eine Sortiergenauigkeit von 95% und niederer Gesamtpreis. Mit Ausnahme des Durchsatzes sind diese Forderungen recht subjektiv. Dennoch müssen daraus Kriterien für die Maschine abgeleitet werden, welche sich dieses Ziels annehmen. Essentielle Punkte für einen praktischen Betrieb sind:

- **Vorreinigung:** Vor der Bildaufnahme sollen kleine Partikel vom Sortiergut entfernt werden. Diese würden die Bildanalyse und somit den kontinuierlichen Sortierfluß nur unnötig belasten.
- **Probentransport:** Für den Transport zur Bildaufnahme kommen viele Techniken in Frage. Entscheidend ist der ruhige Übergang der Proben in den freien Fall, damit

[1] Weitere in der Praxis relevante Kriterien wie Abmessungen, Leistungsaufnahme und Ähnliches sind für die Entwicklung eines Prototyps auch zu überlegen, spielen in Bezug auf die Inhalte dieser Arbeit jedoch eine untergeordnete Rolle.

dieser für alle Proben vergleichbar abläuft. Es sollen keine größeren Geschwindigkeitsunterschiede der Proben ermöglicht werden, da diese die Vergleichbarkeit der visuellen Merkmale empfindlich beeinträchtigen können. Weiters würde der Einsatz der Auswerfer erschwert werden, da unterschiedliche Fallzeiten größere Zeitfenster bei der Aktivierung der Auswerfer erfordern.

- **Probenvereinzelung:** Jede Probe muss einzeln analysiert werden. Daher sind Überlappungen während der Bildaufnahme im freien Fall zu vermeiden. Auch sollen die Auswerfer nur einzelne Proben entfernen. Zu geringe Abstände von aufeinander folgenden (bzw. nebeneinander fallenden) Proben sind daher zu vermeiden. Die erlaubte Dichte des Probenflusses hängt von den Parametern der verwendeten Auswerfer, der Fallhöhe und dem Transportsystem ab.

- **Bildaufnahme:** Die Bilddaten jeder Probe müssen alle zur Analyse benötigten Details der Objekte enthalten. Dahingehend sind Sensoren und Zubehör festzulegen. Da Mängel der Proben an beliebiger Stelle und in verschiedenster Größe auftreten können, sind zwei oder mehr Sensoren zu verwenden.

- **Bildanalyse:** Alle denkbaren Fehlstellen und Defekte, die für die Sortierung relevant sind, müssen von der Bildverarbeitung zur Klassifikation der einzelnen Proben verwendet werden. Als Ergebnis wird jede Probe mit negativ bzw. positiv bewertet.

- **Rechner:** Ein Rechner mit entsprechender Software hat die Bilddaten, welche je Sekunde generiert werden, innerhalb einer Sekunde abzuarbeiten.

- **Auswerfer:** Im Falle von negativen Proben sorgen pneumatische Auswerfer für ein Umlenken deren Flugbahn. Entsprechend der Dichte des Probenflusses müssen die (De)aktivierungszeiten der Auswerfer kurz genug sein, um nicht weitere Proben zu entfernen, welche sich nahe der negativ bewerteten Probe befinden. Die Luftversorgung hat von Öl gereinigte Druckluft bereitzustellen. Je nach Anzahl der Auswerfer ist der Druckluftkreis zu dimensionieren.

- **Systemsteuerung:** Aufgaben der Steuerung sind ein flüssiger Sortierbetrieb, die Erkennung von fehlerhaften Betriebszuständen und defekten Komponenten, das Bereitstellen eines Interfaces für den Betreiber und die Steuerung der Auswerfer.

Da sich diese Arbeit ausschließlich dem optischen System und der Bildverarbeitung der Sortierung widmet, werden nun die Anforderungen dieser Funktionsgruppen im Detail aufgelistet.

2.2 Optische Hardware

Die Qualität eines bildverarbeitenden Systems hängt maßgeblich von der Qualität der Bilddaten[2] ab. Die Dimensionierung optischer Hardware beginnt mit der Auswahl des Sensors und der dazugehörigen Optik. Dann kann die Beleuchtung festgelegt werden. Als

[2] Damit sind nicht Bildparameter wie geometrische Auflösung gemeint. Wesentlich ist die Abbildung der interessierenden Information.

Schnittstelle zum Rechner wird ein Framegrabber verwendet, der die Daten der Bildverarbeitung zur Verfügung stellt. Notwendige Eigenschaften des optischen Systems zur Feuerbohnensortierung sind:

- **Bildsensoren:** Die Szene (also der Fallbereich der Proben) muss kontinuierlich aufgenommen werden, da sich jederzeit Proben vorbeibewegen können. Abhängig von der Breite des Fallschachts ist ein ausreichender Abbildungsmaßstab vorzusehen. Abmessungen müssen nicht maßstäblich abgebildet werden, da keine Vermessungsaufgaben durchgeführt werden. Die gesamte Fläche einer jeden Probe ist zu erfassen. Typ und Parameter der Sensoren sind entsprechend der zu analysierenden Objektmerkmale auszulegen.

- **Optik:** Die gesamte Tiefe des Aufnahmebereichs muss im Bereich der Schärfentiefe liegen. Zusammen mit dem gewählten Bildsensor ist der Abbildungsmaßstab festzulegen.

- **Beleuchtung:** Die Proben befinden sich im freien Fall. Die Fallgeschwindigkeiten der Proben legen eine obere Schranke der Belichtungszeit des Sensors fest. Daher ist ein Beleuchtungssetup zu verwenden, welches in dieser Zeit ausreichend Licht zur Verfügung stellt. Da visuelle Merkmale der Proben analysiert werden sollen, sind Reflexionen nach Möglichkeit zu vermeiden.

- **Framegrabber:** Die Schnittstelle zwischen Bildsensor und Rechner soll eine Synchronisation aller verwendeten Sensoren erlauben. Dies erleichtert die Steuerung des Analysevorgangs.

Verfügbare Technologien werden in Kapitel 3.1 diskutiert. Im letzten Teil dieses Kapitels folgen die Anforderungen an die Bildverarbeitung zur Auswertung der Feuerbohnen.

2.3 Bildverarbeitung für Sortieranwendungen

Das optische System kann nicht dimensioniert werden, bevor die Aufgaben der Bildverarbeitung feststehen. Diese hat im Allgemeinen immer zum Ziel, gewisse Eigenschaften der zu prüfenden Proben festzustellen und zu entscheiden, welcher Klasse die Probe zuzuorden ist. Im einfachsten Falle der Qualitätsprüfung gibt es nur zwei Klassen (mangelhaft und einwandfrei). Die Anzahl der Klassen ist jedoch beliebig. So kann z. B. zur Größensortierung von Süßkirschen eine Einteilung in fünf Klassen notwendig sein.
Die Sortierung der Feuerbohne ist ein binäres Klassifikationsproblem, welches mit Hilfe extrahierter Objektmerkmale gelernt und betrieben wird. Die Wahl der Merkmale und deren mathematische Beschreibung implizieren demnach die Anforderungen an die Bildverarbeitung. Die Kriterien für negativ zu bewertende Proben zur Sortierung von Feuerbohnen wurden erstmals in [Fle09] analysiert und sind:

- **Fremdkörper:** Objekte, welche bzgl. der Größe im Bereich der Feuerbohnen liegen und somit nicht durch die Vorsortierung bzw. -reinigung entfernt werden können (Mais, Steine, Bruchstücke).

- **Gebrochene Feuerbohnen:** Im Zuge der Ernte ergeben sich Bruchstücke, während der Lagerung können Bohnen in zwei Hälften zerbrechen.

- **Fremde Kultur:** Bei der Bewirtschaftung der Agrarflächen können vereinzelt andere Sorten vorkommen.
- **Frühreife:** Typischerweise ist der gewählte Erntezeitpunkt nicht für alle Bohnen ideal, da nicht jede Frucht zur selben Zeit reif wird.
- **Vertrocknete Feuerbohnen:** Diese stellen eine minderqualitative Ware dar und dürfen daher nicht in der Verkaufsmenge enthalten sein.
- **Visuell erkennbare Mängel:** Darunter sind alle Arten von Rissen und Beschädigungen der Bohnenoberfläche zu verstehen, deren Größe einen gewissen Schwellwert überschreiten.

Abbildung 2.3 (a-c) zeigt Aufnahmen einiger aufgelisteter Fehlerklassen. Diese Beispiele lassen darauf schließen, dass eine Mixtur von verschiedensten Merkmalen zur Anwendung kommen wird. So können beschädigte Proben sicher mittels Konturanalyse erkannt werden, eingerissene Oberflächen werden hingegen eher mit Farbstatistik zu detektieren sein. Einige positiv zu bewertende Proben sind in Abbildung 2.3 (d) zu sehen. Feuerbohnen weisen verschiedenste Farbmischungen und Größen auf. Anhand der verwendeten Merkmale sollte neben der Feststellung von Defekten die Identifizierung einer Feuerbohne im Gegensatz zu Fremdkörpern in jedem Fall ermöglicht werden.

In diesem Kapitel wurden Einblicke in Aufbau, Funktion und Anforderungen an Sortiermaschinen im Allgemeinen gegeben, weiters wurden Spezifikationen für ein System zur Bewertung von Feuerbohnen angeführt. Nach dieser genaueren Definition der Aufgaben werden im nächsten Kapitel theoretische Grundlagen und relevante Technologien diskutiert, welche sich zur Realisierung von Bildanalyse und Sortierung prinzipiell eignen.

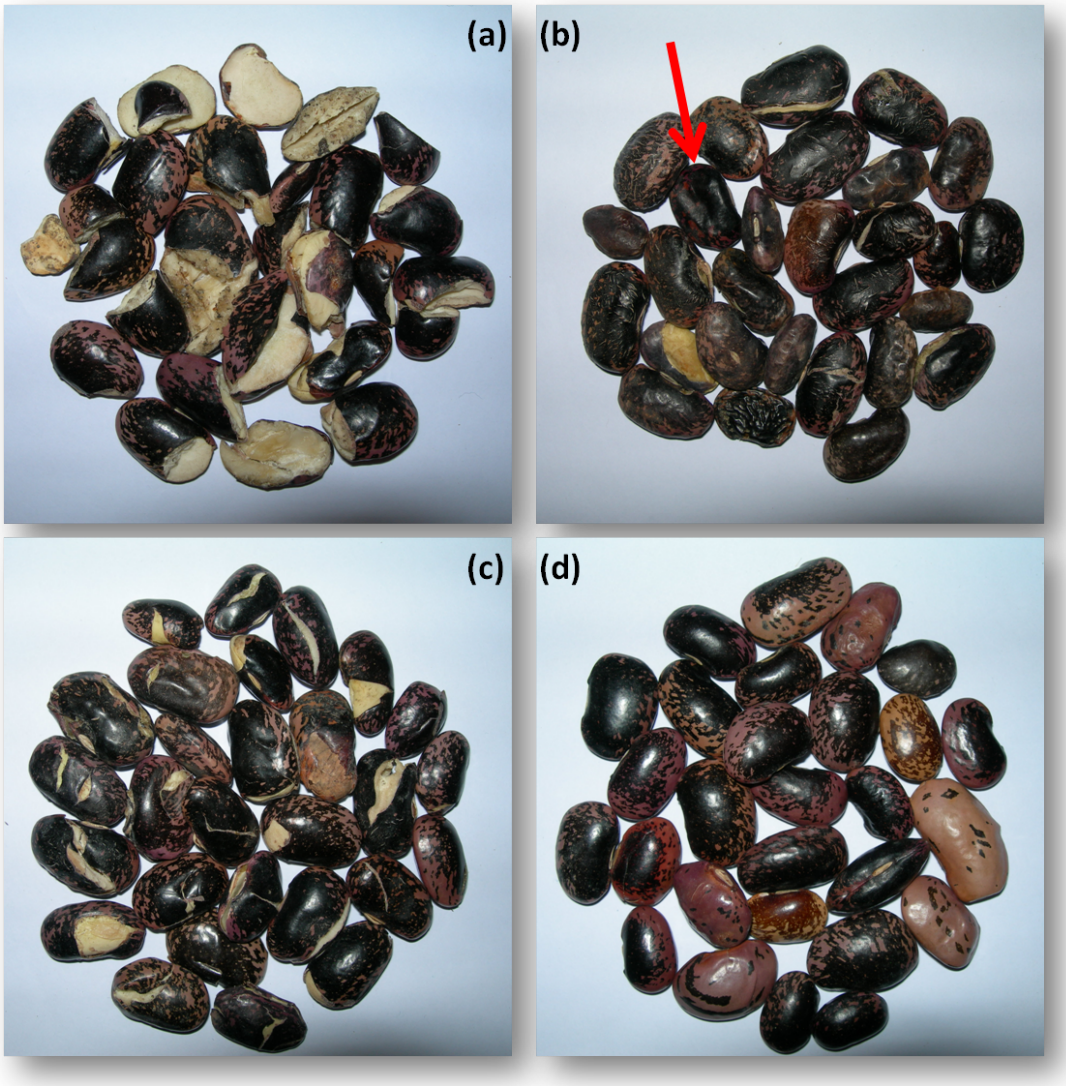

Abbildung 2.3: **Beispielproben von Feuerbohnen;** In (a) werden mechanische Schäden und halbierte Proben gezeigt. Stark qualitative Kriterien wie Trockenheit oder Früreife (durch rötliche Verfärbung erkennbar, siehe Markierung) sind in (b) abgebildet. Weiters stehen visuelle Mängel, welche nichts mit Qualitätsmängeln zu tun haben (c) den einwandfreien Proben (d) gegenüber. In (d) wird deutlich, wie variantenreich die Farbverteilungen vorkommen.

Kapitel 3

Grundkonzepte

Zur Realisierung einer bildgestützten Sortierung[1] sind grundlegende Kenntnisse der Disziplinen Bildverarbeitung, Maschinelles Lernen und Echtzeitsysteme unumgänglich, um geeignete Methodiken auszuwählen, richtig einzusetzen und zu kombinieren. Die Sortierung muß in der Lage sein, Bilddaten zu generieren und zu repräsentieren, Objekte zu segmentieren, festgelegte Objektmerkmale zu berechnen und mit deren Hilfe jedem Objekt eine Klasse zuzuordnen. In den folgenden Unterabschnitten wird jeweils eine Zusammenfassung der verfügbaren Techniken gegeben. Auf Grund des Umfangs können viele Methoden jedoch nur aufgezählt werden, wogegen detailiertere Erläuterungen angeführt werden, falls eine Thematik große Relevanz in Bezug zur Aufgabenstellung aufweist.
Ausgehend von Erzeugung und Repräsentation von Rasterbildern werden für die Sortierung geeignete Halbleitersensoren kurz vorgestellt. Für den Einsatz von bildgebenden Sensoren ist die Auswahl geeigneter Aufnahmehardware entscheidend. Dazu zählen optische Komponenten (Linsen, Filter, usw.), Beleuchtungstechnik sowie Hardware zur Datenübertragung. Da das Aufnahmesetup die Bildqualität festlegt, ist dieses den Anforderungen möglichst gut anzupassen[2].
Aus den Rasterbildern, welche die zu analysierenden Objekte enthalten, sind danach Merkmale zu extrahieren, die mit den Objektfehlern (aber auch mit den einwandfreien Objekten) korrelieren. Es werden (auszugsweise) jene Merkmale erläutert, welche für das Sortierproblem in Frage kommen. Grundlagen zur Berechnung werden ebenso vorgestellt wie Techniken, um aus den Eigenschaften die geeignetsten auszuwählen.
In der Einführung wurde bereits festgestellt, dass Analysesoftware für Sortieranwendungen mit Hilfe von Lernalgorithmen realisiert werden kann. Daher sollen die wichtigsten Lernkonzepte zusammengefasst und verglichen werden. Den Abschluss dieses Kapitels bildet eine kurze Einführung in Echtzeitsysteme, auf deren Basis ein grosser Teil der industriellen Bildverarbeitungssysteme arbeiten. Es ist eingänglich, dass die Aufgabe einer Qualitätsanalyse nur in Echtzeit sinnvoll durchgeführt werden kann.

[1] Aus Sicht der Bildverarbeitung kann man Sortierung mit Klassifikation gleichsetzen.

[2] Je geeigneter die generierten Daten die Informationen repräsentieren, desto leichter fällt es der Bildverarbeitung, diese zu separieren. Unnötig aufwändige Vorverarbeitung kann reduziert werden.

3.1 Prinzipien der Bildaufnahme

Jedes bildgestützte Meßsystem verwendet einen oder mehrere bildgebende Sensoren, welche aus der physikalischen Grösse *Licht* einen Meßwert ableiten. Wie jeder andere Sensor auch, ist ein bildgebender Sensor durch seine typischen Charakteristika bestimmt: Auflösung, Temperaturabhängigkeit, Meßbereich, Sensorgeometrie uvm. Im Allgemeinen gibt ein bildgebender Sensor ein digitales Rasterbild aus, welches die zu bestimmenden Informationen enthält. Vor der näheren Erläuterung sollen noch einige wichtige Begriffe definiert werden.

- **Licht:** Das Licht ist ein spektraler Bereich der elektromagnetischen Strahlung (siehe Abbildung 3.1), welche im Wellenlängenbereich $\lambda = 380 - 770nm$ festgelegt ist. Die Übergänge der Grenzen werden in der Literatur oft etwas abweichend angegeben. Dieses Spektralband kann vom menschlichen Auge als visuelle Information verwertet werden. Dies ist auch der Grund, weshalb man häufig nicht von elektromagnetischer Strahlung, sondern von Licht spricht[3]. Eine ausführliche mathematische Beschreibung gibt [Hec01].

- **Bildgebender Sensor:** Die bildgestützte Messung bezieht sich auf sichtbare Information (entsprechend der Definition von Licht im Spektrum der elektromagnetischen Strahlung).

- **Optischer Sensor:** Ein optischer Sensor kann auch mit anderen Wellenlängen arbeiten, da das Spektrum der optischen Wellenlängen den Bereich zwischen Radiowellen und Röntgenstrahlung (mit Überschneidungen) abdeckt.

Bildgebende Sensoren sind also eine Art von optischen Sensoren, die im Spektrum des Lichts angesiedelt sind und die Messgröße in typischer Form ausgeben. Anzumerken bleibt noch, daß Informationen beliebiger Sensoren in Form von Bildern organisiert werden können. Der Einsatz solcher künstlich generierten Bilder in einem Meßsystem ist also ebenfalls eine Form von bildgestützter Messung. Ein Beispiel: Das Generieren von medizinischer Bildinformation geschieht häufig am Computer, da die Sensoren selbst noch keine Bilder ausgeben. Der Arzt stellt basierend auf den generierten Bildern den Vergleich mit einer Referenzgröße an. Die von bildgebenden Sensoren erzeugten Rasterbilder werden im nächsten Abschnitt eingeführt.

3.1.1 Das Rasterbild

Zur digitalen Auswertung von Lichtinformation ist eine Quantisierung der analogen Meßwerte notwendig. Diese erfolgt zeitlich (über die Integrationszeit des Sensors) und auch örtlich (über die geometrischen Eigenschaften des Sensors sowie das verwendete optische Aufnahmesystem). Die Generierung von digitalen Rasterbildern umfasst eine ganze Kette von Operationen. Durch die Quantisierung des Meßsignals erhält man ein digitales Rasterbild. Detailierte Informationen über Rasterbilder finden sich zB. in [Pin97], die

[3] Die optische Meßtechnik unterscheidet die Radiometrie als Wissenschaft über und zur Messung elektromagnetischer Strahlung von der Fotometrie, die die Augencharakteristik mit einbezieht. Dadurch wird also der Meßbereich eingeschränkt, die gemessenen Grössen stehen in engem Zusammenhang mit dem eigenen Empfinden.

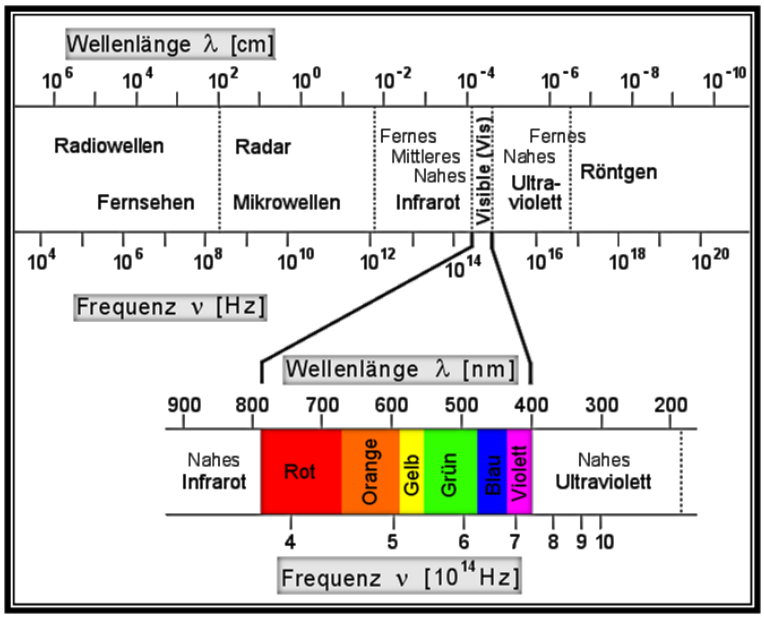

Abbildung 3.1: **Das elektromagnetische Spektrum;** Der sichtbare Bereich im Gesamtkontext des Spektrums. Das Auge nimmt zusätzlich zum visuellen Bereich Infrarot- und Ultraviolettleistung auf, auch wenn sie für uns unsichtbar erscheint.

folgende mathematische Beschreibung ist an [AVO68] angelehnt. Details bezüglich des Quantisierens von Bildinformation führt [Bra95] ein. Grundlagen über optische Systeme liefert [Hec01].
Ein Rasterbild ist eine diskrete Darstellung einer Szene, die demnach als diskrete Funktion $i(x, y)$ angegeben werden kann, sofern es sich um ein 2-dimensionales Bild handelt. Dabei gibt x die Zeilenkoordinate und y die Spaltenkoordinate des Bildes an. Jeder Punkt dieser Funktion (in der Bildverarbeitung mit Pixel bezeichnet) hält einen Wert, welcher in einem Grauwertbild die Intensität widerspiegelt. Behandelt man Farbbilder, so werden Farbräume der Dimension c eingesetzt, daher wird die Funktion zu $i(x, y, c)$ erweitert. Bei Verwendung dieser dritten Koordinate c spricht man auch von Farbkanälen, d. h. c gibt die Anzahl dieser Kanäle an. Es handelt sich also um c Funktionen $i(x, y)$, welche die entsprechenden Kanalinformationen enthalten. Da Rasterbilder immer eine endliche Anzahl an Pixel aufweisen, bezeichnet M die Anzahl der Funktionswerte in y-Richtung (d.h. die Anzahl der Zeilen) und N entsprechend die Anzahl der Spalten eines Bildes. Für ein Rasterbild soll im Weiteren die Notation

$$i(x, y), x = \{0, 1, ..., M - 1\}, y = \{0, 1, ..., N - 1\} \tag{3.1}$$

verwendet werden, wenn es sich um ein Grauwertbild bzw. um ein Kanalbild handelt. Für eine Bildkomposition aus mehreren Kanälen wird $i(x, y, c)$ verwendet, wobei $N, M \in \mathbb{N}$ gelten muss und typischwerweise $c = 3$ gilt. Als Ursprung wird die linke obere Bildecke gewählt, d.h. $i(0, 0)$. Alle im Folgenden besprochenen Themen beziehen sich auf Bilder der Form $i(x, y)$, andernfalls wird explizit darauf hingewiesen.
Ein Rasterbild kann also als eine $M \times N$ Matrix aufgefasst werden. Wichtige Kenngrößen des Rasterbildes sind:

- **Geometrische Auflösung:** Sie gibt die Dimension der Bildmatrix an und bestimmt, welches räumliche Detail einer Szene noch im Rasterbild zu erkennen ist. Eine geeignete Wahl dieser Auflösung ist für die industrielle Bildbearbeitung wesentlich: Die benötigte Information sollte gerade noch erkennbar und separierbar sein. Eine höhere Auflösung enthält nicht unbedingt mehr Information und ist daher redundant.

- **Radiometrische Auflösung:** Diese Größe gibt den Wertebereich der Quantisierung eines Pixel an. Bei einer gängigen Auflösung von 8 Bit kann der Pixelwert demnach im Bereich von $0 - 255$ liegen. 10 Bit und 12 Bit sind ebenfalls oft verwendete Auflösungen.

- **Pixelnachbarschaft:** Dies ist eine wichtige Definition zur Verwendung von Rasterbildern. [uREW02] beschreibt die 4-Nachbarschaft N_4 als die horizontalen und vertikalen Nachbarn eines Pixels. Werden auch die diagonalen Nachbarn einbezogen, spricht man von 8-Nachbarschaft N_8. Die Pixelnachbarschaft ist ein wichtiges Kriterium bei der Implementierung vieler Algorithmen der Bildverarbeitung. Von der Wahl der Nachbarschaft ist beispielsweise abhängig, ob zwischen Pixel eine Verbindung besteht oder nicht.

3.1.2 Bildgebende Sensoren und Peripherie

Sensoren zur Gewinnung von Rasterbildern gibt es in breiter Auswahl. Einige Beispiele listet Tabelle 3.1 auf, eine Zusammenstellung aller verfügbaren Technologien und Anwendungsbereiche sind in [PG99] zu finden. Für die industrielle Bildverarbeitung im visuellen Bereich kommen prinzipiell zwei Halbleitertechnologien zum Einsatz. Charged Coupled

Technologie	**Sensoren**	**Anwendung**
Festkörper	CCD, CMOS	Visuelle Bildverarbeitung
Dünnfilm	TFA Sensor	Anwendungen hoher Dynamik
Multispektral	Multispektralkamera	Spektralanalyse, Fernerkundung
High Dynamic Range	CMOS-HDRC	nichtlineare Abbildung der Intensität

Tabelle 3.1: Sensortechnologien

Devices (CCD's) und Complimentary Metall Oxid Semiconductor Sensoren (CMOS) werden mit ihren wichtigsten Eigenschaften kurz vorgestellt. Beide Technologien werden als monochromatische Sensoren oder als Farbsensoren ausgefürt. Letztere besitzen Pixel unterschiedlicher spektraler Empfindlichkeit, deren Intensitätswerte nach dem Auslesen zu Farbpixel kombiniert werden. Der Beschreibung der Sensoren folgen Betrachtungen zur Sensorgeometrie.

CCD Sensor

CCD Sensoren bedienen sich der Ladungstrennung über eine Kapazität, wobei die dafür nötige Energie von einströmenden Photonen geliefert wird. Die Ladung, die nach dem Auslesen in eine Spannung gewandelt wird, ist proportional der Anzahl der einfallenden Photonen und somit der Intensität. Im Gegensatz zur parallelen Belichtung steht ein serielles Auslesen. Die Auslesezeit ist bei CCD Sensoren ein begrenzender Faktor. Weitere Nachteile sind störende Bildeffekte wie das Blooming (bei Überbelichtung). Dies kann zwar technisch verhindert werden, allerdings wird dann der Zusammenhang zwischen der einfallenden Lichtmenge und der Ausgangsspannung nichtlinear. Weiters ist es nicht möglich, Pixel direkt zu adressieren. Vorteilhafte Eigenschaften des CCD sind eine hohe dynamische Reichweite (also das gleichzeitige Erfassen dunkler und heller Szenenbereiche), die Verfügbarkeit von Sensoren im Infrarot-, Ultraviolett- und Röntgenbereich und nicht zuletzt eine hohe Lichtempfindlichkeit.

CMOS Sensoren

Je nach Technologiegrad integriert der CMOS Sensor immer mehr Funktionalität direkt am Pixel selbst. Dies beginnt bei Verstärkern und reicht bis hin zu Analog-Digital Wandlern (ADC). Aus diversen Gründen setzt sich vor allem im Konsumentenbereich der CMOS Sensor immer mehr durch. Auf der einen Seite werden seine generellen Nachteile, wie die fertigungstechnisch hervorgerufenen Abweichungen der einzelnen Pixelempfindlichkeiten, besser in den Griff bekommen. Auf der anderen Seite weist er sehr prägnante Vorzüge auf. Diese ergeben sich hauptsächlich durch den Umstand, daß jedes Pixel direkt die Ladung in eine Spannung umsetzt. Man erhält dadurch eine direkte Adressierbarkeit der Pixel, hohe Auslesegeschwindigkeiten und paralleles Auslesen. Durch die Integration von mehr Elektronik am Pixel sinkt jedoch dessen Fill Factor (der lichtempfindliche Anteil des Pixel zur gesamten Pixelfläche). Geringere Baugröße und geringerer Stromverbrauch sind ebenso zu nennen wie ein sehr geringer Bloomingeffekt.

Sensorgeometrie

Bildgebende Sensoren sind in verschiedenen Größen (je größer ein Pixel, umso lichtempfindlicher ist er) und in verschiedenen Geometrien erhältich. Der Flächensensor (2-dimensional) wird am häufigsten eingesetzt, dieser Umstand liegt einfach in der Natur der Anwendung. Geht es jedoch darum, kontinuierlich bewegte Szenen bzw. Objekte zu erfassen, zeigen Zeilensensoren ihre Vorteile auf. Diese 1-dimensionalen Sensoren belichten eine einzelne Zeile mit hoher Frequenz. Aus mehreren Zeilen kann ein Bild zusammengesetzt werden, sofern dies notwendig ist. Große Vorteile dieser Sensoren sind die kontinuierliche Datengenerierung, weiters werden Objekte niemals mehrfach aufgenommen. Bei geeigneter Wahl der Zeilenfrequenz und bekannter Objektgeschwindigkeit werden Größenverhältnisse richtig erfasst. Auf Grund der Bauform ergeben sich weitere positive Effekte, z.B. ein Fill Factor von 100%. Neben monochromatischen Zeilensensoren kann man Farbzeilensensoren bezüglich des Aufnahmeprinzips in drei Typen einteilen[4]:

[4]Diese Einteilung gilt unter Beachtung der geometrischen Unterschiede ebenso für Flächensensoren.

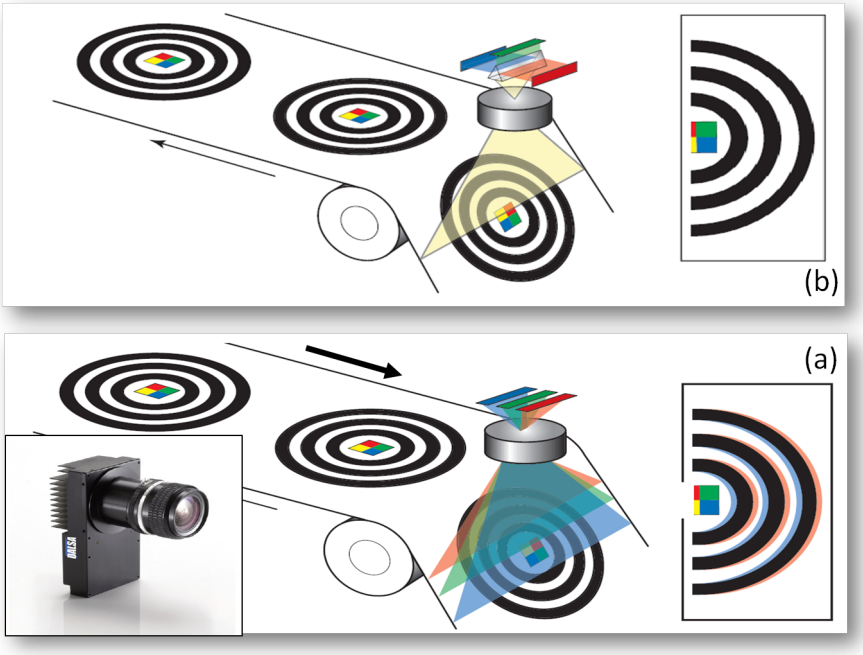

Abbildung 3.2: **Prinzip von Zeilensensoren;** Zeilensensoren erfassen bewegte Szenen (z. B. am Förderband). Der in (a) dargestelle trilineare Sensor verwendet drei räumlich nebeneinander angeordnete Sensoren, durch Software wird aus jeweils drei Pixel ein Farbpixel kombiniert. In gekrümmten Szenen ergeben sich dadurch farbliche Verzerrungen. Dies ist an dem dargestellten Zeilenkomposit rechts in (a) erkennbar. Der Prismensensor (b) umgeht dieses Problem durch den Einsatz einer zusätzlichen Optik. Die Grafik stammt aus [Fir02].

- **Trilinearer Sensor:** Ein Farbpixel entsteht aus Kombination mehrerer monochromatischer Pixel. Solche Sensoren weisen gute Lichtempfindlichkeit auf (vergleichbar mit monochromatischen Sensoren), eignen sich jedoch nur begrenzt für 3-dimensionale Szenen (siehe Abbildung 3.2((a)).
- **Prismensensor:** Ein Prisma spaltet die Spektralbänder auf. Rund um das Prisma sind drei Pixel angeordnet. Dadurch entfällt das Problem bei 3-dimensionalen Szenen, durch die zusätzliche Optik wird jedoch die Lichtempfindlichkeit reduziert (siehe Abbildung 3.2((b)).
- **Zeilensensor mit Farbmosaik:** Solche Sensoren sind eine günstige Alternative und können zur Generierung von Farbbildern oder monochromatischen Bildern verwendet werden. Die Auflösung bei Verwendung als Farbsensor reduziert sich um 2/3, da drei nebeneinander liegende Pixel zu einem Farbpixel kombiniert werden (Prinzip wird in Abbildung 3.3 gezeigt). Für die spektrale Empfindlichkeit entsprechend der Farben kommen Farbfilter zum Einsatz.

Nach der Einführung in Sensortechnologie zur Erstellung digitaler Rasterbilder werden Möglichkeiten zur Szenenbeleuchtung und der Übertragung der Bilddaten aufgelistet.

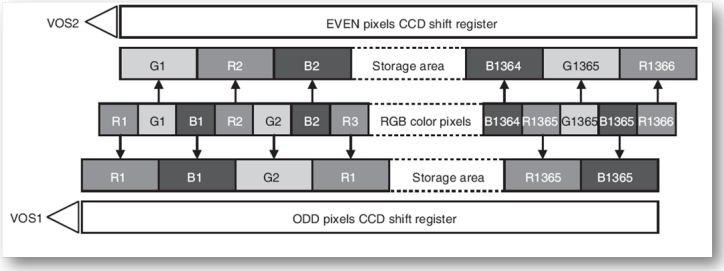

Abbildung 3.3: **Zeilensensor mit Farbmosaik;** Beim Auslesen der generierten Pixelwerte (mittlere Zeile) entstehen zwei Pixelzeilen. Deren Kombination ergibt eine Farbzeile. Alle Pixel werden in einem einzelnen Takt in zwei Schieberegister geschoben (Abbildung aus [Atm05]).

Beleuchtungstechnologien

Die Wahl der richtigen Beleuchtung ist für ein bildgestütztes System ein essentieller Schritt. Stark abhängig von der Anwendung[5] ist die Beleuchtungsgeometrie zu wählen, weiters kommen diverse Typen von Lichtquellen zum Einsatz. Oft verwendete Geometrien sind:

- **Gerichtetes Licht:** Licht fällt direkt von der Quelle auf die Szene. Dabei auftretende Schatten und Reflexionen sind richtungsabhängig.
- **Diffuses Licht:** Licht wird gestreut und verliert dadurch die Richtungsabhängigkeit. Reflexionen und Schatten können unterdrückt werden, jedoch ist bei gleicher Belichtungszeit eine höhere Lichtstärke notwendig, um die gleiche Beleuchtungsstärke zu erzielen wie bei direktem Licht.
- **Diffus axiales Licht:** Licht wird über einen Spiegel gestreut, wodurch ebenfalls Reflexionen und Schatten unterdrückt werden.
- **Ringlicht:** Ringförmige Anordnungen von Lichtquellen sind diffus und gerichtet verügbar.
- **Linienlicht:** Diese Ausführung eignet sich in Kombination mit Zeilensensoren.
- **Domlicht:** Durch eine hohe Zahl von verschiedenen Beleuchtungsrichtungen wird eine hohe Reflexions- und Schattenunterdrückung erreicht.
- **Hintergrundlicht:** Die Ausleuchtung des Szenenhintergrunds wird vor allem zur Konturanalyse von Objekten eingesetzt.
- **Dunkelfeld:** Objekte werden vor dunklem Hintergrund seitlich beleuchtet, damit nur indirektes Licht den Sensor erreicht. Damit kann man Details auflösen, die nur schwer abzubilden sind, z. B. Niveauunterschiede an Oberflächen.

[5] Je nach Art der Szene und der Anwendung sind Überlegungen bezüglich Reflexionen, Farben, Transparenz, bewegter Objeke uvm. zu tätigen.

- **Strukturiertes Licht:** Lichtmuster werden auf Objekte projiziert, um daraus Tiefeninformation zu berechnen. Diese Technik kommt z. B. bei 3-dimensionaler Rekonstruktion zum Einsatz.

Detailierte Infomationen lassen sich in [PG99] nachlesen. Natürlich sind auch diverse Variationen und Kombinationen dieser Geometrien denkbar.
Nach der Wahl der Geometrie ist eine geeignete Beleuchtungsquelle (allgemeiner: Strahlungsquelle) zu finden. Strahlungsquellen werden nach verschiedenen Kriterien unterschieden. Die Strahlungsintensität

$$I_e = \frac{d\Phi_e}{d\Omega} \tag{3.2}$$

ist jene Strahlungsleistung Φ_e, welche eine Strahlungsquelle in ein Raumwinkelelement $d\Omega$ abstrahlt. Die Angabe erfolgt in W/sr. Die Strahlungsleistung

$$\Phi_e = \frac{dQ}{dt} \tag{3.3}$$

ist die von den elektromagnetischen Wellen geführte Energie Q je Zeit in W. Zur Angabe der Strahlungsleistung, welche auf eine Fläche A einfällt, verwendet man die Bestrahlungsstärke, welche durch

$$E_e = \frac{d\Phi_e}{dA} \tag{3.4}$$

gegeben ist. Da E_e als Leistung definiert ist, ergibt sich die Einheit W/m^2. Die drei bisher eingeführten Größen besitzen fotometrische Entsprechungen zur Beschreibung von Lichtquellen, d. h. unter Berücksichtigung der spektralen Empfindlichkeit des Auges. Die Strahlungsleistung Φ_e entspricht dem Lichtstrom Φ_v, welcher in Lumen (lm) angegeben wird. Das Analogon zur Strahlungsintensität I_e ist die Lichtstärke I_v, welche sich auf gleiche Weise berechnen lässt. Die Lichtstärke wird in Candela (cd) angegeben. Die Bestrahlungsstärke E_e wird in der Fotometrie mit der Beleuchtungsstärke E_v angegeben, welche in Lux (lx) auszudrücken ist. Die Einheiten der Fotometrie entsprechen natürlich der selben physikalischen Bedeutung wie die zugehörigen Größen in der Radiometrie. Die bisherigen Ausführungen dienen der Beschreibung von Strahlungs- bzw. Lichtquellen. Geht es um den Vergleich von Lichtquellen untereinander, so eignet sich dafür die Lichtausbeute (fotometrische Größe)

$$\eta = \frac{\Phi_v}{P} \tag{3.5}$$

am Besten. Sie gibt das Verhältnis des abgegebenen Lichtstroms zur aufgenommenen Leistung P in lm/W an und ist daher ein Maß für die Effizienz einer Lichtquelle.
Weitere Kenngrößen von Strahlungsquellen sind das Spektrum der abgegebenen Strahlung und die Strahlungsfrequenz. Bei der industriellen Bildverarbeitung (bzw. für Laborzwecke) im visuellen Spektrum kommen heute hauptsächlich zwei Typen von Lichtquellen zum Einsatz:

- **Halogen- und Xenonlampe:** Hohe Lichtstärken werden durch Fokussierung erzeugt, jedoch ist auch die Leistungsaufnahme hoch. Diese Lampen sind auch in Kaltlichtausführung (gedämpfter Infrarotanteil) erhältlich.

- **Leuchtdiode (LED):** LED Quellen sind für viele Wellenlängenbereiche verfügbar, sehr preiswert und kompakt. Durch ihre geringe Leistungsaufnahme und beliebige Kombinierbarkeit lassen sich eine Vielzahl von Beleuchtungsaufgaben lösen.

Genauere Informationen zu Strahlungsquellen im Allgemeinen stehen in [Hec01]. Lichtquellen für die Bildverarbeitung werden eingehend in [PG99] vorgestellt.

Übertragung von Bilddaten

Zum Transport der Bilddaten zu einem Rechner bzw. Speicher bieten sich mehrere Übertragungsprotokolle an. Ausschlaggebend bei der Wahl des Protokolls sind die zu übertragende Datenmenge, die Distanz, die Signalart (analog oder digital) sowie der Preis. Gebräuchlich sind vor allem USB 2.0 (Spezifikation in [Com00]), Gigabit Ethernet (in [Iee08]), Firewire (in [Tra04]) und CameraLink (in [Aia04]). In Zukunft werden auch USB 3.0 sowie CameraLinkHS eine Rolle spielen. Zur Übernahme der Bilddaten am Rechner können Framegrabber eingesetzt werden. Diese sind speziell für die Pufferung von Bilddaten ausgelegt und je nach Ausführung mit Prozessoren zur Echtzeitvorverarbeitung ausgestattet. Gerade in zeitkritischen Anwendungen mit hohen Datenraten ist der Einsatz eines Framegrabbers sinnvoll. Die grundlegende Funktionsweise dieser Hardware und die verschiedenen Ausführungsformen werden in [PG99] angegeben.

3.1.3 Dimensionierung optischer Systeme

Nachdem ein geeigneter Sensor für die Bildaufnahme festgelegt wurde, ist die zu verwendende Optik zu bestimmen. Zum Einen soll die zu erfassende Szene auf dem Sensor abgebildet werden, zum Anderen ist der gewünschte Detailgrad festzulegen. Das Setup muss im Bereich der sich ergebenden Schärfentiefe liegen. Die angegebene Vorgehensweise basiert auf der geometrischen Optik, deren vereinfachte Betrachtungsweise für diese Anwendungen weitestgehend ausreichend ist. Die Berechnungen richten sich nach [Sch07].
Ausgangspunkt stellt die Linsengleichung

$$\frac{1}{a} + \frac{1}{a'} = \frac{1}{f} \tag{3.6}$$

dar, in der a den Messabstand, a' die Bildweite (Abstand des Sensors vom Objektivzentrum) und f die Brennweite des Objektivs festlegen, siehe auch Abbildung 3.4. Der Abbildungsmaßstab β berechnet sich zu

$$\beta = \frac{L}{S} = \frac{a}{a'}. \tag{3.7}$$

L bezeichnet die Objekt- bzw. Szenenbreite, S die Breite des Sensors. Damit lässt sich die benötigte Brennweite mittels

$$f = \frac{OO'}{\frac{1}{\beta} + \beta + 2} \tag{3.8}$$

angeben. OO' ist dabei der Abstand zwischen Sensor und Szene. Der Objektivauszug $\Delta s'$

$$\Delta s' = \frac{f}{\beta} \tag{3.9}$$

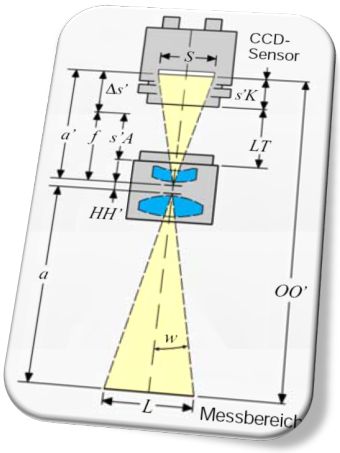

Abbildung 3.4: **Geometrie von Sensor, Objektiv und Szene;** Dieses Schema führt relevante Abmessungen zwischen Sensor und Szene ein, welche zur Objektivbestimmung benötigt werden (Abbildung entnommen aus [Sch07]).

ermöglicht die Berechnung der Tubuslänge LT, die sich aus der Summe Objektivauszug und Auflagemaß des Objektivs $s'A$ abzüglich dem Auflagemaß der Kamera $s'K$ ergibt.

$$LT = s'A + \Delta s' - s'K \tag{3.10}$$

Der Gesamtabstand folgt dann zu

$$OO' = \left(\beta + \frac{1}{\beta} + 2\right) f + HH', \tag{3.11}$$

worin der Hauptpunktabstand HH' enthalten ist. Kommt eine Vorsatzlinse zum Einsatz, die eine Brennweite von f_{VL} aufweist, ergibt sich resultierend eine effektive Brennweite f'.

$$\frac{1}{f'} = \frac{1}{f} + \frac{1}{f_{VL}} \tag{3.12}$$

Damit verkleinert sich auch der Abbildungsmaßstab

$$\beta' = \beta \frac{f'}{f + \beta(f - f')}. \tag{3.13}$$

In die Berechnung der Schärfentiefe z_s geht unter anderem die Blendenzahl K und der Abstand zwischen den Mittelpunkten zweier benachbarter Pixel am Sensor $\Delta y'$ ein.

$$z_s = 2\Delta y' K \beta \,(1 + \beta) \tag{3.14}$$

Dies bedeutet, dass z_s mit der Blendenzahl, dem Abbildungsmaßstab und dem Pixelabstand zunimmt. Weiters ist zu bedenken, dass im Falle $\beta \leq 10$ die Objektivauflösung mit einer effektiven Blendenzahl

$$K' = K\left(1 + \frac{1}{\beta}\right) \tag{3.15}$$

zu rechnen ist. Die Randintensität $R_{\%}$ beschreibt die Abnahme der Lichtintensität zum Rand der Szene hin. Diese tritt selbst bei homogener Beleuchtung auf. Da sie nicht zur Gänze vermieden werden kann, ist sie zu bestimmen und in der Bildaufnahme zu berücksichtigen (zB. durch die Konfiguration von Pixel Offsetwerten am Sensor). Zur Ermittlung wird der Bildwinkel w benötigt, der durch

$$w = \arctan\left(\frac{S}{2f\left(1+\frac{1}{\beta}\right)}\right) \tag{3.16}$$

definiert wird. Damit kann $R_{\%}$ zu

$$R_{\%} = \cos^4(w) \tag{3.17}$$

gefunden werden. Es sollte in jedem Fall $R_{\%} > 70$ gelten, da andernfalls das Bildsignal am Rand deutlich verstärkt werden muss, wodurch der Signal-Rausch Abstand SNR gesenkt wird. Letztendlich wird noch die Grenze der Auflösung angeführt, welche das Objektiv (theoretisch) erreichen kann. Sie tritt auf Grund der Lichtbeugung auf. Zwei benachbarte Punkte am Bild sind noch zu unterscheiden, wenn die Bedingung

$$\Delta y' \geq 2.4\lambda K' \tag{3.18}$$

erfüllt ist. Es kann ohne Weiteres $\lambda = 550nm$ angenommen werden.

3.2 Vom Rasterbild zum Featurevektor

Eine bildgestützte Sortierung zu realisieren, ist softwareseitig ein Klassifikationsproblem. Kern eines solchen Systems ist ein Klassifikator, welcher vorerst als Blackbox angesehen werden kann, die Eingänge in Form von Objektmerkmalen und als Ausgang eine Klassifikationsentscheidung aufweist. Die Mathematik zur Berechnung dieser Objektmerkmale wird im Laufe dieses Unterkapitels erläutert.
Um die Merkmale möglichst effizient extrahieren zu können, wird zuerst eine geeignete Repräsentation für die Rasterbilder gesucht. Neben allgemeinen Eigenschaften des im letzten Abschnitt eingeführten Rasterbildes ist bei der Verwendung von Farbbildern die Wahl des Farbraumes zu entscheiden. Die in dieser Weise festgelegten Bilder werden einer Bildvorverarbeitung unterzogen, welche die Daten entsprechend aufbereitet. Aus diesen müssen anschließend die Objekte einzeln segmentiert werden. Das Segmentationsproblem wird mit einigen einfachen Verfahren vorgestellt.
Die Auswahl von Objektmerkmalen, welche für die Klassifikation geeignet sind, stellt ein fundamentales Problem dar. Sie muss für jede Aufgabenstellung der Bildverarbeitung neu gelöst bzw. angepasst werden und ist (bei Vorhandensein von brauchbaren Bilddaten) wohl jene Entscheidung, die sich am Stärksten auf die Performance der Klassifikation auswirkt. Die Auswahl kann manuell oder auch automatisiert erfolgen. Es soll hier ein (recht grober) Überblick über Objektmerkmale gegeben werden. Etwas genauer, wenn auch nur zusammenfassend, werden jene Merkmale beschrieben, die für das konkrete Sortierproblem in Frage kommen.

Zur visuellen Inspektion von Objekten werden die Merkmale häufig manuell zusammengestellt. Dabei sollte jedoch sichergestellt werden, dass die Merkmale in ihrer Anzahl ausreichend, in ihrer Aussagekraft schlüssig und in ihrer Kombination nicht mehrdeutig sind. Dies kann prinzipiell durch eine Performanceanalyse des Klassifikators geschehen, im Vorhinein kann jedoch bereits abgeschätzt werden, inwieweit einzelne Merkmale bzw. deren Kombination als Entscheidungsgrundlage dienen können. Einige Auswahlverfahren werden am Ende des Unterkapitels kurz vorgestellt.

3.2.1 Farbräume

Farben werden vom menschlichen Auge wahrgenommen, da die in der Retina vorhandenen Zäpfchen (von welchen es zumindest drei Typen gibt) unterschiedliche spektrale Absorptionsbereiche aufweisen [Pin97]. Es sollte auch erwähnt werden, dass das menschliche Farbsehen auch von diversen kognitiven Faktoren wie Lernen, Emotionen und Erinnerungen beeinflusst wird. Grundsätzlich sind drei (Primär-)farben ausreichend, um durch Mischung beliebige Farben zu erhalten. Das Mischen von Farben wird durch die Graßmannschen Gesetze beschrieben [Pog02] . Angelehnt an diese Beobachtungen können beliebige Farbräume definiert werden.

RGB Farbraum

Der wohl verbreitetste Farbraum ist der RGB[6] Farbraum, welcher sehr einfach in der Darstellung ist, wenngleich er für den Menschen nicht sehr intuitiv ist[7]. Abbildung 3.5 zeigt den RGB Farbraum, welcher die drei Primärfarben Rot, Grün und Blau linear separiert. Die meisten bildgebenden Sensoren arbeiten auf Basis dieses Farbraumes, da sich die Aufspaltung des visuellen Bereiches in drei Kanäle als technisch relativ einfach gestaltet. Durch Normalisieren der Farbkanäle entsprechend

$$\begin{aligned} r &= \frac{R}{R+G+B} \\ g &= \frac{G}{R+G+B} \\ b &= \frac{B}{R+G+B} \end{aligned} \tag{3.19}$$

ist die starke Beleuchtungsabhängigkeit dieses Farbraumes reduzierbar, dies wurde in [uTP87] gezeigt. Da $b = 1-r$ gilt, kann der RGB Raum durch zwei normalisierte Koordinaten repräsentiert werden. Eine Ableitung dieses Farbraumes stellt der CMYK dar. Dieser verwendet die Sekundärfarben des RGB Raumes, Cyan, Magenta und Gelb als Primärfarben. Allerdings ist seine Beschreibung allgemein gehalten, sodaß prinzipiell auch andere Farben verwendet werden könnten (generativer Farbraum). Der CMYK stellt einen Standard in der Drucktechnik dar. Das in der Abkürzung verwendete K steht für „key plate“, womit eine schwarze Platte gemeint ist, an der die Farbplatten ausgerichtet werden.

[6] Die Abkürzung steht für rot, grün und blau.

[7] Man kann sich unter einer Mischung aus 25% Rot, 30% Grün und 45% Blau nur schwer eine konkrete Farbe ausmalen.

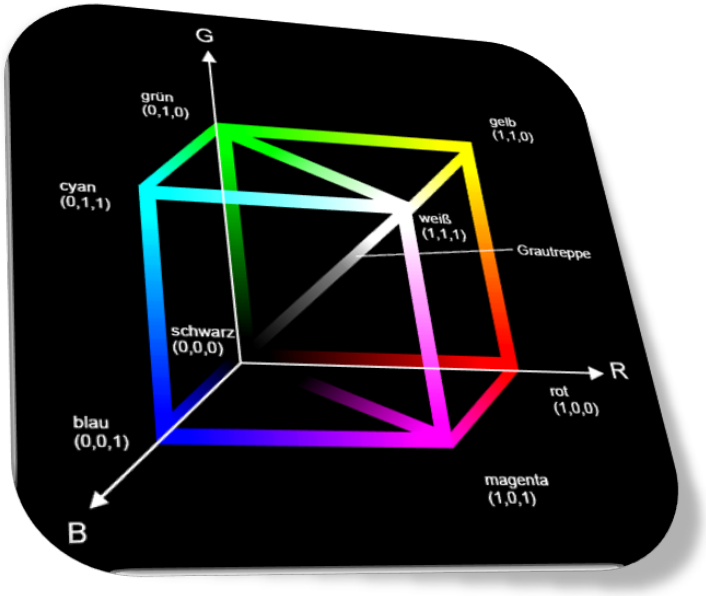

Abbildung 3.5: **RGB Farbraum;** Der Farbraum in karthesischen Koordinaten zeigt die Primär- und Sekundärfarben an den Eckpunkten (volle Sättigung), sowie die Verbindungslinie zwischen Schwarz und Weiss als Diagonale des Würfels. Die Farbe weiß setzt sich aus Maximalwerten aller drei Primärfarben zusammen.

HSV Farbraum

Interessantere Eigenschaften für die industrielle Bildverarbeitung besitzt der HSV Raum. Dieser nichtlineare Farbraum spaltet den Farbton (Hue), die Sättigung (Saturation) und die Intensität (Value) auf. Der Farbraum wird oft auch als HSI (I für Intensität) bezeichnet, wie in weiterer Folge auch in dieser Arbeit. Eine alternative Darstellung liefert der HLS Raum, in dem die Helligkeit (Luminance) zur Anwendung kommt. Der HSI Raum ist in Abbildung 3.6 dargestellt. Entscheidendes Merkmal ist die Kapselung von Farbinformation und Helligkeit, weswegen er sich gut für die Farbsegmentierung eignet.
RGB und HSI Räume bieten sehr vielseitige Einsatzmöglichkeiten in der Industrie, aus diesem Grund kommen sie auch in der Realisierung der Sortieraufgabe zum Einsatz. Eine Umrechnung wird häufig schon am Framegrabber durchgeführt, da dies pixelweise sehr effizient geschehen kann. Um aus einem RGB Bild die HSI Variante zu berechnen, verwendet man

$$H = \begin{cases} \theta, & B \leq G \\ 360 - \theta, & B > G \end{cases} \tag{3.20}$$

für den Farbton, wobei

$$\theta = \arccos\left(\frac{\frac{1}{2}(2R - G - B)}{\sqrt{(R-G)^2 + (R-B)(G-B)}}\right) \tag{3.21}$$

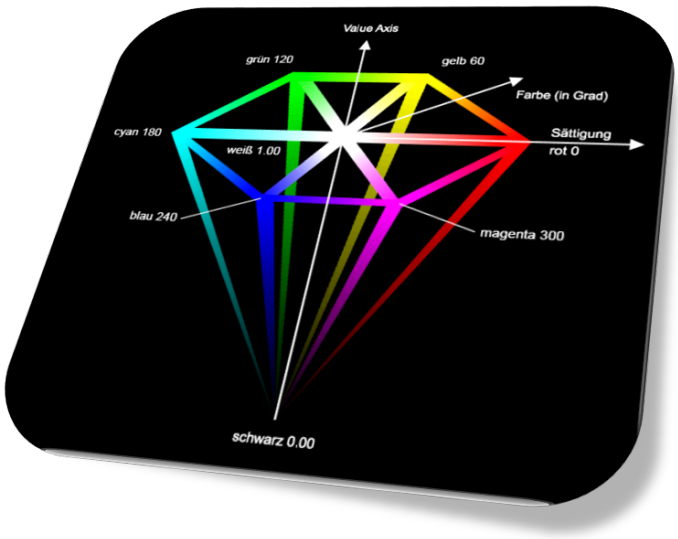

Abbildung 3.6: **HSI Farbraum (HSV Variante);** Der Winkel in der Grundflächenebene entspricht dem Farbton, die Sättigung ist die Länge des Vektors in diese Richtung meist auf Werte von 0 bis 1 normalisiert. Eine Sättigung von 0 bedeutet also weiß, egal, welcher Intensitätswert vorliegt.

gilt. Die Sättigung wird durch

$$S = 1 - \frac{3}{R+G+B} min(R,G,B) \tag{3.22}$$

ausgedrückt und die Intensität folgt zu

$$I = \frac{1}{3}(R+G+B). \tag{3.23}$$

Die Abkürzungen beziehen sich auf die Farbkanäle. Zur Verwendung dieser Formeln ist eine Normierung der Kanäle auf den Bereich $[0,1]$ notwendig. In umgekehrter Weise erhält man die RGB Kanalinformation aus dem HSI Bild. Der Blaukanal wird mittels

$$B = I(1-S) \tag{3.24}$$

bestimmt, Rot entspricht

$$R = I\left(1 + \frac{S\cos H}{\cos(60-H)}\right) \tag{3.25}$$

und der Grünkanal ist

$$G = 3I - (R+B), \tag{3.26}$$

d.h. er kann einfach aus dem Intensitätskanal durch Abzug der beiden anderen Kanäle erhalten werden. Die angeführten Formeln richten sich nach [uREW02], die Herleitung ist z.B. in [Smi78] nachzuvollziehen.

Alternative Farbräume

Kurz erwähnt sollen noch einige weitere interessante Farbräume werden. Der CIE Farbraum beschreibt das Spektrum aller sichtbaren Farben. Die dazugehörige Normtafel wurde durch Versuche an „Normalbeobachtern“erstellt. Details zu diesem Farbraum gibt [Fai98] wieder. Der Cie L* a* b* Farbraum ist ein kugelförmiges Farbmodell und ermöglicht eine geräteunabhängige Farbbeschreibung.

3.2.2 Operationen der Bildvorverarbeitung

Die Vorverarbeitung gestaltet sich entsprechend der vorhandenen Aufgabenstellung sehr unterschiedlich. Geht es bei der Bildverbesserung oder -restaurierung eher darum, den visuellen Eindruck zu verbessern, ist es z.B. in der Medizin oft notwendig, gewisse Bildkomponenten (Objekte) hervorzuheben, um diese gut zu erkennen. Klassifikationsaufgaben werden erleichtert, indem die diskriminanten Merkmale von Objekten verstärkt bzw. die irrelevanten Unterschiede von Objekten (bei gleichzeitiger Erhaltung der diskriminanten Merkmale) reduziert werden. Dies kann durch Konvertierung in geeigente Farbräume geschehen oder durch Filteroperationen, die sehr unterschiedliche Bildkomponenten verstärken bzw. abschwächen. Verständlicherweise gestaltet sich die Vorverarbeitung umso einfacher, je geeigneter die Bilddaten generiert werden. Denn die Vorverarbeitung kann niemals zusätzliche Informationen erzeugen, sondern immer nur Informationen entfernen. Ideale Bilddaten können aber in der Praxis meist nicht zur Verfügung gestellt werden. Die Sensoren und die Optik arbeiten fehlerbehaftet, auch wird die Szene selbst gestört. Viele dieser Störungen sind durch ein Rauschmodell zu beschreiben (siehe folgender Abschnitt). Zur Behandlung dieser und auch anderer Störungen werden verschiedene Operationen verwendet:

- **Lokale Pixeloperation:** Die Operation ist nur vom aktuellen Pixel abhängig. Beispiel: Grauwerttransformationen;
- **Lokale Nachbarschaftsoperation:** In eine Berechnung eines Pixel gehen die Nachbarn des Pixel mit ein. Beispiel: Hoch- bzw. Tiefpassfilter;
- **Globale Operation:** Die Operation an einer Stelle wird von jedem Pixel des Bildes beeinflusst. Beispiel: Fourier Transformation;

Eine Pixeloperation entspricht einer Transformation des Originalpixel $i_o(x, y)$ zu $i(x, y)$. Dies wird formal mit

$$i(x, y) = T(i_o(x, y)) \tag{3.27}$$

dargestellt. Die globalen Operationen werden an dieser Stelle nicht näher betrachtet, da sie in dieser Arbeit nicht zur Anwendung kommen. Die lokalen Nachbarschaftsoperationen spielen jedoch eine große Rolle in der Vorverarbeitung und auch der Merkmalsberechnung. Daher sollen diese nun eingeführt werden. Eine Hauptanwendung liegt in der Filterung von Bildern mit Hilfe der Faltung.

Faltungsoperationen

Wie aus der Fouriertheorie bekannt ist, entspricht eine Multiplikation im Frequenzbereich einer Faltung im Raumbereich. Die Fouriertransformation $I_c(u, v)$ einer kontinuierlichen,

2-dimensionalen Funktion $i_c(x, y)$ ist gegeben durch

$$I_c(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} i_c(x, y) e^{-j2\pi(ux+vy)} dxdy, \tag{3.28}$$

die dazugehörige Inverse wird lt.

$$i_c(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I_c(u, v) e^{j2\pi(ux+vy)} dudv \tag{3.29}$$

bestimmt. Da Bilder jedoch diskreten Funktionen entsprechen, kommt die diskrete Fourier Transformation (DFT) zur Anwendung, welche im 2-dimensionalen Fall mit

$$I(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} i(x, y) e^{-2j\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{3.30}$$

und die Inverse durch

$$i(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} I(u, v) e^{2j\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \tag{3.31}$$

gegeben ist. Bei der verwendeten Formulierung sind x und y die Raumkoordinaten, u und v sind die dazugehörigen Variablen im Frequenzbereich. Die DFT wird für eine diskrete Bildfunktion der Größe $M \times N$ definiert. Diese Transformation entspricht dem Beschreiben einer beliebigen Funktion durch eine Summe periodischer Funktionen. Wichtig ist nun das Faltungstheorem

$$i(x, y) * h(x, y) \Longleftrightarrow I(u, v)H(u, v), \tag{3.32}$$

worin $h(x, y)$ eine Filtermaske (auch Filterkern oder nur Filter) darstellt, sowie $H(u, v)$ die entsprechende Fourier Transformierte $F\{h(x, y)\}$. Das Theorem beschreibt im Raumbereich eine Faltung, welche der Gleichung

$$i(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} h(i, j) i_o(x + i, y + j) \tag{3.33}$$

entspricht. Dabei ist die Größe der Filtermaske durch $m = 2a+1$ und $n = 2b+1$ gegeben, wobei $a, b \in \mathbb{N}^+$ gelten muß. Man sieht hierbei, dass alle Maskenelemente in die Berechnung des aktuellen Pixel miteingehen.
Auf Basis der Faltung können verschiedenste Vorverarbeitungsschritte durchgeführt werden. Wie bereits erwähnt ist das Filtern von Bildern die Hauptanwendung. In Abbildung 3.7(a) wird ein Originalbild dargestellt, welches diversen Filteroperationen unterzogen wird. In Abbildung 3.7(b) kommt ein Mittelwertfilter mit der Maske

$$h_m(i, j) = \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.34}$$

zum Einsatz, Abbildung 3.7(c) zeigt das Ergebnis einer Laplacefilterung mit Maske

$$h_l(i, j) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{3.35}$$

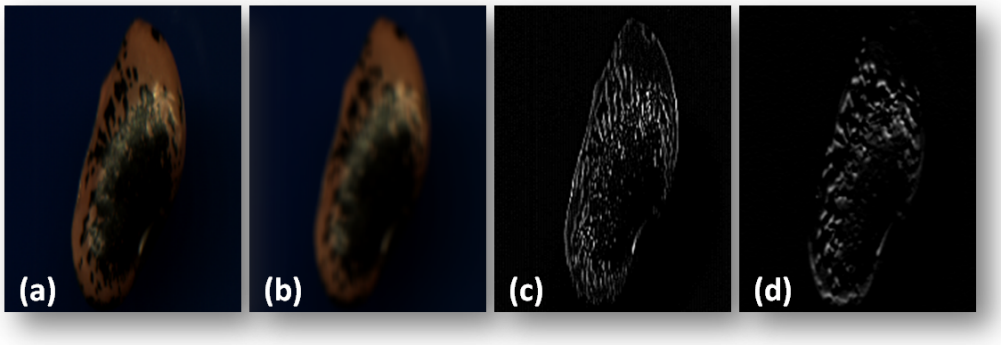

Abbildung 3.7: **Anwendung von Filtern;** Das Originalbild (a) wird mit drei verschiedenen Filtermasken gefaltet. Bild (b) zeigt eine Mittelwertfilterung mit Maske lt. Gleichung 3.34, (c) wendet das Filter aus Gleichung 3.35 an, bei dem es sich um ein Laplacefilter handelt. Bild (d) wurde mit den Sobelfiltern aus den Gleichungen 3.36 bzw. 3.37 erstellt.

Abbildung 3.7(d) wurde mit den Sobelfiltern

$$h_{s,x}(i,j) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{3.36}$$

und

$$h_{s,y}(i,j) = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{3.37}$$

bearbeitet. Wie der Laplacefilter ist auch der Sobelfilter ein Kantenfilter. In Abbildung 3.7 werden damit die Farbkanten (also sprunghafte räumliche Änderungen der Farbe) hervorgehoben, während homogene Bereiche gedämpft werden. Bevor in weiterer Folge eine Einteilung diverser Filter erfolgt, muss noch das Bildrauschen definiert werden, damit der Einsatz der Filter auch mathematisch zu begründen ist.

Bildrauschen

Abhängig von der Aufnahmeumgebung können zufällig verteilte Störungen verschiedenster Ursachen auftreten. Dies reicht von Bildverzerrungen (z.B. durch Luftströmungen in der Szene verursacht) bis hin zu aufgenommenen Partikeln im Pixel- und Subpixelbereich (Salz- und Pfefferrauschen). Im Lebensmittelbereich können Staub, Kleinteile und Bruchstücke auftreten. Ob solche Störobjekte bereits während der Vorverarbeitung entfernt werden, hängt natürlich von deren geometrischen Ausdehnungen ab. Hier sind Objekte mit Größen von wenigen Pixel gemeint, größere Objekte müssen später im Zuge der Klassifikation als Fremdkörper identifiziert werden. Rauschen wird nach Art des Auftretens und dessen Eigenschaften grundsätzlich als additiver oder multiplikativer Term modelliert. Da es sich bei Rauschen um eine nicht vorhersagbare Erscheinung handelt, erfolgt die mathematische Beschreibung mittels Wahrscheinlichkeitsdichtefunktionen (engl. probability density function, pdf). Multiplikatives Rauschen (wie z.B: Specklerauschen, welches durch kohärente Beleuchtung optisch rauher Flächen entsteht) spielt für diese Arbeit keine Rolle. Im Falle des additiven Rauschens erfolgt die Beschreibung anhand des

Terms $\eta(x, y)$, der dem idealen Bild $i_i(x, y)$ überlagert wird. Für jedes Pixel ergibt sich dadurch die Gesamtintensität

$$i_o(x, y) = i_i(x, y) + \eta(x, y). \tag{3.38}$$

Das Gausssche Rauschmodell, dessen 1-dimensionale Dichte mit

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.39}$$

gegeben ist, wird sehr gerne eingesetzt, da es mathematisch im Raumbereich und im Frequenzbereich gut handhabbar und eine gute Anpassung an die Aufgabenstellung möglich ist. Neben vielen weiteren denkbaren Modellen (nachzulesen z.B. in [uREW02]) sei hier noch das Impulsrauschen erwähnt, da dieses Rauschmodell sehr gut zufällig auftretende Pixelausreißer beschreibt, wie dies im Falle von Staub der Fall sein kann. Im Allgemeinen steuern Pixel, die durch Impulsrauschen belastet sind, maximal aus. Sie weisen also den maximal bzw. den minimal möglichen Intensitätswert auf. Die Dichtefunktion ist mit

$$f_X(x) = \begin{cases} F_{min}, & x = a \\ F_{max}, & x = b \\ 0, & sonst \end{cases} \tag{3.40}$$

gegeben, wobei a den kleinsten und b den größten möglichen Intensitätswert darstellt. Das Intervall der Intensitäten wird durch die radiometrische Auflösung bestimmt (Sättigungsintensitäten). Die zugehörigen Wahrscheinlichkeiten F_{min} und F_{max} summieren sich zur Wahrscheinlichkeit von 1 auf.

Lineare Filter

Der erste Schritt der Vorverarbeitung (neben eventuellen Farbraumtransformationen) liegt meist in der Entfernung von Störungen bzw. in der Verbesserung (Aufwertung) der Bildinformation. Bei der Bildanalyse zur Sortierung ist es vor allem notwendig, eine Entfernung von Pixelrauschen bzw. Kleinstpartikeln vorzunehmen. Dies Aufgabe können die zuvor eingeführten Filteroperationen durch Wahl geeigneter Filtermasken übernehmen. An dieser Stelle soll eine Einteilung mit kurzen Erläuterungen vorgenommen werden.
Durch eine Filtermaske repräsentierbare Filter sind lineare Filter. Diese Filter gehorchen den Linearitätsbedingungen und sind gleichermaßen mittels der Faltung einsetzbar. Dieser Umstand ist sehr günstig, können doch beliebige Filter mit ein und derselben Berechnungsvorschrift implementiert werden. Beispiele dafür sind:

- **Tiefpass:** In diese Kategorie fallen Filter, welche sich glättend auf das Bild auswirken. Dazu gehören Mittelwertfilter und Gaußfilter. Mittelwertfilter reduzieren Rauschen, indem ein Pixelwert durch Mittelung aller Werte einer lokalen Umgebung berechnet wird. Dazu existieren eine Vielzahl von Implementierungen, wie z.B. gewichtete Filter und rotierende Masken, welche eine Kantenverschmierung vermeiden. Nachteilig kann evtl. sein, dass neue Pixelwerte entstehen, auch werden Konturen unschärfer.

- **Hochpass:** Diese Filter werden auch Kantenfilter genannt, da sie Intensitätsunterschiede verstärken, homogene Bereche jedoch abdämpfen. Klassische Filtermasken sind Sobel, Prewitt und Roberts (basierend auf der 1. Ableitung der Intensitäten), sowie Laplace und Marr-Hildreth Operatoren (basierend auf der 2. Ableitung).

Kombinationen von Tiefpass und Hochpass ergeben einen Bandpass, welche einen speziellen Bereich von Bildfrequenzen verstärken oder dämpfen können. Letzteres wird zum Entfernen von periodischem Rauschen verwendet.
Bezüglich der Effizienz der Implementierung soll noch die Möglichkeit der Separierung von Filtermasken erwähnt werden. Ist eine 2-dimensionale Filtermaske $h(x, y)$ isotrop, so kann sie durch zwei 1-dimensionale Filtermasken ersetzt werden, die hintereinander zur Anwendung kommen. Die Filteroperation mit einer $N \times N$ Filtermaske benötigt N^2 Multiplikationen sowie $N^2 - 1$ Additionen. Separiert man diese Maske, so reduziert sich der Rechenaufwand auf $2N$ Multiplikationen und $2(N - 2)$ Additionen.

Nichtlineare Filter

Nichtlineare Filter vermeiden den Umstand, neue Inensitätswerte zu generieren, wenn sie jedem Pixel nur vorhandene Werte zuweisen. Diese Filter lassen sich nicht mittels einer Filtermaske ausdrücken, weshalb je nach Typ des Filters eine geeignete Berechnungsvorschrift anzuwenden ist. Die häufig bessere Anpassung der Filter wird also durch den Umstand einer geringeren Effizienz bzgl. des Rechenaufwands erkauft. Eine große Gruppe der nichtlinearen Filter sind die Rangordnungsfilter. Ein Maximumfilter

$$i(x, y) = \max_{(i,j) \in N_{xy}} \{i_o(i, j)\} \tag{3.41}$$

sortiert z. B. alle Pixelwerte unter einer Maske und weist dem aktuellen Pixel den größen Wert zu. N_{xy} ist die Pixelmenge der lokalen Nachbarschaft. Analog funktioniert der Minimumfilter und der Medianfilter, welcher aus der sortierten Pixelmenge den mittleren Wert zuweist. Rangordnungsfilter können nach Belieben iterativ verwendet werden. Die notwendige Sortierung die Intensitätswerte erwirken den höheren Rechenaufwand der Rangordnungsfilter.

Adaptive Filter

Ist ein Filter dazu in der Lage, sich den Bildgegebenheiten anzupassen, spricht man von adaptiven Filtern. Ein adaptiver Mittelwertfilter kann z.B. durch

$$i(x, y) = \overline{i_m} + \frac{\sigma_m^2}{\sigma_m^2 + \sigma_i^2} \left(i_m(x, y) - \overline{i_m} \right) \tag{3.42}$$

angeschrieben werden. Dabei stellt i_m den aktuellen Mittelwert unter der Maske dar. Ist die Grauwertvarianz unter der Maske σ_m^2 hoch im Vergleich zur Bildvarianz σ_i^2, dann befindet man sich im Bereich hoher Bildfrequenzen (z.B. Kanten). Hier wird der Grauwert kaum verändert. Im Bereich niederer Frequenzen wird $i(x, y) \propto \overline{i_m}$. Näheres kann in [Hay96] nachgelesen werden.
Ein wichtiger adaptiver Filter ist der bilaterale Filter, welcher in [uCT98] eingeführt wurde. Dieser Mittelwertfilter passt die Filtermaske an die lokale Bildinformation in der Art an,

dass an Intensitätskanten keine Mittelung erfolgt. Damit erreicht man eine Bildglättung bei gleichzeitiger Kantenerhaltung. Der Rechenaufwand ist jedoch (wie allgemein bei adaptiven Filtern) sehr hoch. Dies begründet sich durch die Nichtlinearität und dem Umstand, dass die Maske ständig variiert. Daher sind sehr effiziente Implementierungen für die Verwendung in Echtzeitsystemen notwendig.

Morphologie

Die Morphologie wird in der Bildvorverarbeitung zum Filtern eingesetzt. Ferner hat diese Disziplin Anwendungen bei der Segmentierung und Objektbeschreibung. Morphologische Operationen haben große Bedeutung bei der Anwendung auf Binärbilder $b(x, y)$, in denen Pixel nur die Werte $\{0, 1\}$ annehmen können. Als Maske wird ein Strukturelement (S_E) verwendet, welches mit dem Bild über Operationen der Mengenlehre, wie Vereinigung oder Schnitt, verknüpft wird. Das Bild selbst kann ebenfalls als Menge verstanden werden. Beispiel: Ein Binärbild besteht aus der Menge aller Hintergrundpixel (schwarz) sowie einer Menge von Objektpixel (weiß). Da Binärbilder sich gut als Objektmasken eignen, stellen sie ein nützliches Werkzeug in der Praxis dar. Hier sollen nur die gängigsten Operationen erwähnt werden. Dazu wird noch die Definition der Reflexion

$$\hat{A} = \{w \mid w = -a, a \in A\} \tag{3.43}$$

und der Translation

$$(A)_p = \{c \mid c = a + b, a \in A\}, p = (p_1, p_2) \tag{3.44}$$

bezüglich einer Menge A benötigt. Die Dilation

$$A \oplus S_E = \{p \mid [(\hat{S}_E)_p \cap A] \subseteq A\} \tag{3.45}$$

weitet ein Objekt um das Strukturelement auf. Dessen Form kann beliebig gewählt und somit den Anforderungen angepasst werden. Da man in der Morphologie von Mengen ausgeht, wird eine entsprechende Schreibweise gewählt. A entspricht der Menge der Objektpixel, p einen Punkt dieser Menge. Sowohl A als auch S_E sind Teilmengen des $\mathbb{Z}^2$. Man kann sich diese Operation durch das Bewegen des Strukturelements entlang der Objektgrenze vorstellen. Die Aufweitung geht mit dem Auffüllen kleiner Lücken einher. Die Erosion

$$A \ominus S_E = \{p \mid (S_E)_p \subseteq A\} \tag{3.46}$$

verkleinert dagegen das Objekt und entfernt Brücken, die eine geringere Ausdehnung als das Strukturelement aufweisen. Diese Operationen können auch kombiniert werden. Durch das Öffnen, welches mit der Gleichung

$$A \circ S_E = (A \ominus S_E) \oplus S_E \tag{3.47}$$

definiert wird, entfernt Objekte, die kleiner als das Strukturelement sind, erhält dabei die Größe von Objekten und glättet deren Kontur (entfernt Entartungen). Dem gegenüber steht das Schließen, welches entsprechend

$$A \bullet S_E = (A \oplus S_E) \ominus S_E \tag{3.48}$$

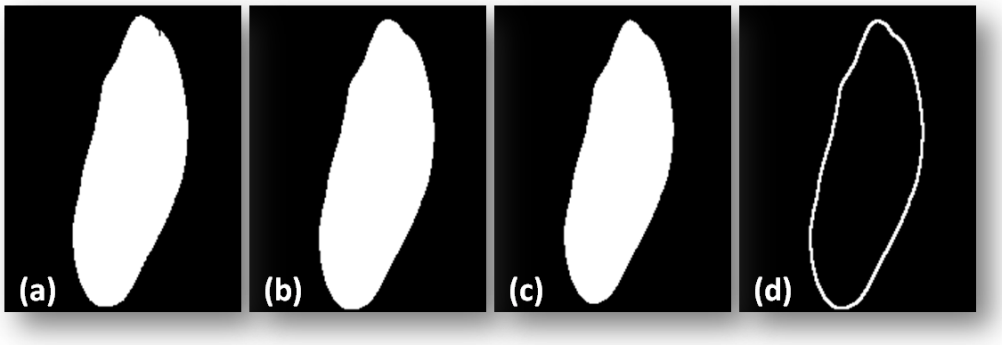

Abbildung 3.8: **Binäre Morphologie, Beispiel;** Eine Maske (a) wird zuerst mit der Schließen Operation gefiltert, um eine glattere Oberfläche zu erhalten (b). Dieses Bild wird nun erodiert (c) und danach von der originalen Maske subtrahiert. Als Ergebnis liegt die extrahierte Kontur des Objekts vor (d).

Brücken zwischen Objekten bildet und Löcher schließt. Wie zuvor werden Objektgrößen erhalten und Konturen geglättet. Mit diesen Methoden lassen sich viele Anwendungen realisieren. Zum Beispiel kann man auf einfache Weise die Kontur einer binären Maske bestimmen, indem man die erodierte von der originalen Maske subtrahiert.

$$C(A) = A - (A \ominus S_E) \tag{3.49}$$

Ein Beispiel zeigt Abbildung 3.8, weitere logische Operationen und deren Anwendungen finden sich in [Ros00], die grundlegenden Arbeiten zu diesem Gebiet können in [Ser82] nachgelesen werden.

3.2.3 Objektsegmentierung und Labeling

Die Segmentierung der Objekte vom Hintergrund bildet nach der Vorverarbeitung der Bilddaten den nächsten Schritt zur Gewinnung der Objetkmerkmale. Ein Bild Ω (diese Notation kann als Menge von Pixel aufgefasst werden) kann als eine Vereinigungsmenge von N Regionen Ω_i lt.

$$\Omega = \bigcup_{i=1}^{N} \Omega_i \tag{3.50}$$

aufgefasst werden. Die Segmentierung teilt Ω in Regionen (Teilmengen) Ω_i ein, für welche

$$\Omega_i \bigcap \Omega_j = \emptyset, \forall i \neq j \tag{3.51}$$

gelten soll. Eine Region soll zusammenhängend sein, wenn alle zugehörigen Pixel ein festgelegtes Homogenitätskriterium

$$H(\Omega_i) = wahr \tag{3.52}$$

erfüllen. Dieses Kriterium kann ein Grauwert, eine Farbe, eine Textur, Kombinationen uvm. sein. Man kann Segmentierungstechniken bzgl. der Extraktion von Regionen und der

Extraktion von Kanten einteilen. Regionsbasierte Verfahren sind z. B. Schwellwertsegmentierung, Watershed Segmentierung, Mean Shift Prozedur und die Mumford-Shah Technik. Kantenbasierte Methoden können Edge Linking, aktive Konturen und die Hough Transformation sein. Auf Grund des Umfangs dieser Thematik und der eingeschränkten Verwendung in dieser Arbeit erfolgt an dieser Stelle keine nähere Beschreibung dieser Verfahren. Jede komplexe Methodik ist nicht zielführend im Sinne von performanter Verarbeitung und bei Bereitstellung entsprechender Bilddaten auch nicht notwendig. Einen umfangreichen Überblick über Verfahren und Anwendungen sind in [uGCS01] und in [Umb98] zu finden. Die Mean Shift Prozedur wird in [uPM02] erläutert, die Mumford-Shah Segmentierung kann in [uJS89] nachgelesen werden.
Die Schwellwertsegmentierung ist das einzige Verfahren, welches in diesem Rahmen betrachtet wird. Durch einen definierten Schwellwert wird aus einem Bild meist ein Binärbild $b(x, y)$ erzeugt, dazu wird die Relation

$$b(x,y) = \begin{cases} 1, & i(x,y) > T \\ 0, & i(x,y) \leq T \end{cases} \tag{3.53}$$

verwendet. Je nach Bedarf kann dieser einfache Mechanismus den Bedürfnissen angepasst werden. Beispielsweise kann eine automatische Bestimmung des Schwellwertes erfolgen, weiters werden statt einem einzigen globalen Schwellwert mehrere regionale Schwellwerte eingesetzt. Eine statische Beleuchtung der Szene ist bei diesem Segmentierungsverfahren vorauszusetzen. Die aus der Segmentierung erhaltenen Binärmasken werden mit einer der zuvor beschriebenen Filtertechniken nachbearbeitet.
Zur praxisgerechten Sortierung ist verständlicherweise ein vorgegebener Durchsatz zu gewährleisten. Dies legt nahe, dass auf einem Bild stets mehrere Objekte vorhanden sein werden. Eine reine Segmentierung vom Hintergrund ist in diesem Fall nicht ausreichend, müssen doch die einzelnen Objekte unabhängig voneinander betrachtet werden. In der Objektmaske ist also noch ein Labeling durchzuführen, welches jedes Objektpixel einem Objekt zuweist. Im einfachsten Fall wird dazu die Maske zeilenweise durchlaufen. Alle Nachbarn eines Objektpixels gehören sinngemäß zum selben Objekt, somit können jedem Pixel des Objekts o eine ID l zugewiesen werden, wobei $l \in \mathbb{N}$ zweckmäßig ist. Vorteil dieses einfachen Ansatzes ist, dass er ohne Einschränkung für Zeilensensoranwendungen geeignet ist. Ein Objekt muss nicht notwendigerweise auf nur einem Bild enthalten sein. Durch Speicherung der Zwischenergebnisse eines Bildes kann dass Labeling beim nächsten Bild fortgesetzt werden. Dieses und weitere Labelingverfahren werden in [Dav90] erläutert.

3.2.4 Berechnung von Objektmerkmalen

Die bisherigen Überlegungen beschreiben, wie man generell vom Erfassen einer Szene bis hin zu segmentierten Objekten kommen kann. Nun muss ein Überblick der berechenbaren Objektmerkmale gegeben werden, welche für die Analyse der Proben in Frage kommen. Dabei kann auf einige Gebiete nicht eingegangen werden. Jedoch gilt es auch hier, die Komplexität der Merkmale in Grenzen zu halten. Die gewählte Einteilung der Merkmale richtet sich im Wesentlichen nach [Fle09].
Die Merkmale einer Objektklasse (oft auch Deskriptoren genannt), welche für eine Klassifikationsaufgabe verwendet werden, sollten signifikante Unterschiede zu den Merkmalen

anderer Objektklassen aufweisen. Zumeist ist die Verwendung mehrerer Merkmale sinnvoll, um die Objektklassen eindeutiger identifizieren zu können. Bei der Verwendung von n Merkmalen f_i spannen diese einen n-dimensionalen Merkmalraum (engl. Feature Space) auf. Ein Objekt kann somit rein durch dessen Merkmale repräsentiert werden. Die n Merkmale bilden den Merkmalvektor (engl. Feature Vector)

$$\vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}, \tag{3.54}$$

welcher für jedes Objekt bestimmt wird und der späteren Klassifikation als Input dient. Die folgenden Unterabschnitte zeigen die Berechnung einer Vielzahl an Merkmalen f_i.

Regionsbasierte Merkmale

Regionsbasierte Merkmale spiegeln Eigenschaften der Objektfläche wieder. Unter diese Kategorie fallen auch topologische Merkmale. Die einfachsten Merkmale sind der Flächeninhalt A, welcher die Anzahl der Objektpixel angibt. Der Umfang p ergibt sich aus der Anzahl der Randpixel, welcher je nach gewählter Pixelnachbarschaft variiert. Die Eulerzahl [Dye80] ist die Differenz zwischen verbundenen Komponenten und Löchern eines Objekts. Ein weiteres Maß ist die Kompaktheit V eines Objekts, welche sich zu

$$V = \frac{p^2}{4\pi A} \tag{3.55}$$

errechnet und ein Verhältnis zwischen Umfang p und Flächeninhalt A beschreibt. Aus der Formel ist erkennbar, dass für einen Kreis $V = 1$ gilt. Ein ähnliches Merkmal ist die Exzentrizität, welche die längste Achse mit der zu ihr orthogonalen Achse vergleicht. Dadurch ist ebenfalls ein Maß für die Ausdehnung eines Objekts gegeben. Andere Ansätze betrachten ein Bild als eine 2-dimensionale Dichtefunktion (pdf) eines Zufallsvektors. Daraus lassen sich verschiedene Momente bestimmen, welche ebenfalls charakteristisch für ein Objekt sind (Beispiele: Entropie, Energie). Momente und deren Eigenschaften werden in [Sav88] beschrieben. Die konvexe Hülle ist die kleinste konvexe Region, die ein Objekt umschließt, Details erläutert [uMIS85]. Erwähnenswert bleiben Regionsdekomposition, welche die Objektregion in elementare Regionen aufteilt [Pav77], die Hough Transformation [uELZ97] und morphologische Operationen [dsrumcf90].

Konturbasierte Merkmale

Informationen über den Konturverlauf dienen nicht nur dem Detektieren von fehlerbehafteten Objekten, sondern auch der Erkennung von Fremdkörpern, da viele dieser Körper einen abweichenden Konturverlauf aufweisen. Weiters lassen sich aus der Kontur unmittelbar weitere elementare Eigenschaften ableiten, wie z.B. Umfang oder (indirekt auch) der Flächeninhalt.
Um konturbasierte Merkmale zu bestimmen, muss zuerst die Kontur selbst extrahiert werden. Dabei haben unterschiedliche Ansätze ihre Berechtigung, in Abbildung 3.8 wurde eine Methode der Morphologie vorgestellt. Prinzipiell ist es jedoch ausreichend, ausgehend von

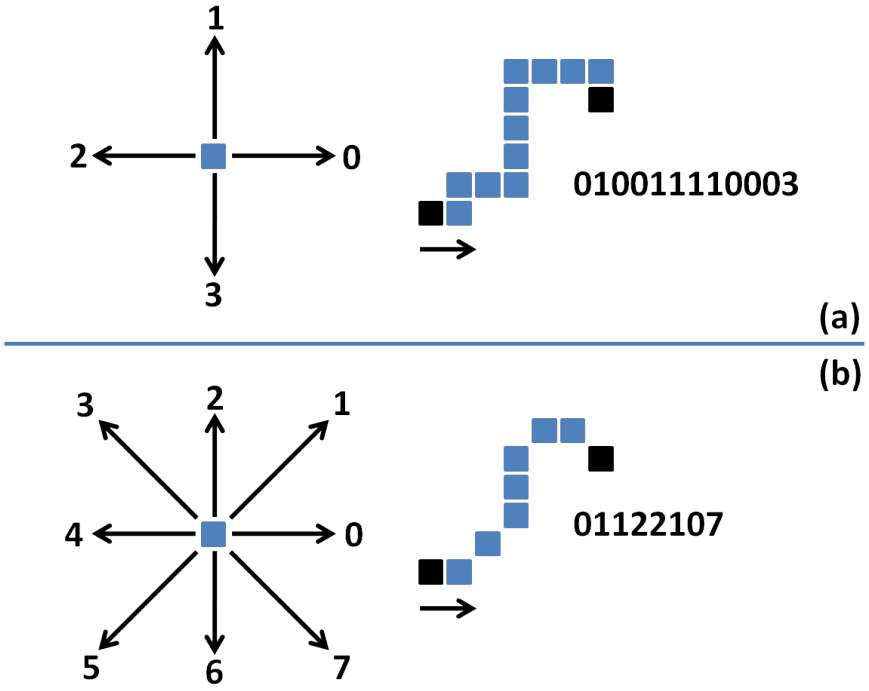

Abbildung 3.9: **Freeman Chain Code;** Zur Bestimmung des Freeman Chain Codes verwendet man ein Codierungsschema, wie dies in (a) für die N_4 bzw. in (b) für die N_8 Pixelnachbarschaft gegeben ist. Ausgehend von einem Startpixel durchläuft man die Kontur und fügt dem Code jenen Zahlenwert hinzu, welcher auf Grund der Lage des nächsten Pixels dem Schema entspricht. Für die beiden Konturbeispiele ist der jeweilige Code angegeben.

einem Startpixel die Kontur zu verfolgen und so das (zuvor segmentierte) Objekt zu umlaufen. Damit erhält man für jeden Punkt der Kontur die Koordinaten in der Bildebene und kann daraus eine Kurve zusammensetzen. Praktischer und platzsparender sind 1-dimensionale Repräsentationen. Der Freeman Chain Code [Fre61] ist die populärste Form der Konturrepräsentation. Die Berechnung folgt einem fixen Schema: abhängig von der Lage des nächsten Konturpixel wird der Code um eine Zahl erweitert. Die Zahlen werden durch ein Nachbarschaftsschema festgelegt. Abbildung 3.9 zeigt zwei solcher Schemata, mit deren Hilfe der Freeman Chain Code für die N_4 und für N_8 Pixelnachbarschaften zu berechen ist und gibt zwei Beispiele. Entsprechend der verwendeten Nachbarschaft ergeben sich natürlich unterschiedliche Codes. Um diese Codes als vergleichbare Merkmale zu verwenden, müssen folgende Nachbearbeitungen für jeden Code $\vec{C}$ der Länge p durchgefürt werden:

- **Rotationsinvarianz:** Aus $\vec{C}$ wird dessen Differenzcode $\vec{C}_d$ bestimmt. Diesen erhält man durch Subtraktion aller jeweils benachbarter Codewerte.
- **Startpunktinvarianz:** Abhängig vom Startpunkt der Codeberechnung erscheinen vergleichbare Codes verschoben. Um diesem Umstand entgegenzuwirken, wird der

Code so lange rotiert, bis die kleinste Zahl (der gesamte Code wird als Zahl betrachtet) gefunden ist.

- **Normalsieren der Codelänge:** Werden von zwei ähnlichen Objekten (im Sinne von ähnlicher Abbildung) $\vec{C_1}$ und $\vec{C_2}$ berechnet, so wird üblicherweise $p_1 \neq p_2$ gelten, obwohl die selbe Information enthalten ist. Durch geeignete Normalisierung (z. B. durch Mittelung über Codeabschnitte kann $p_1 = p_2$ erreicht werden. Gleichzeitig wird dadurch eine Dimensionsreduzierung vorgenommen, da ein Code (betrachtet als Merkmalvektor) oft eine viel zu hohe Dimension aufweist.

Mit Hilfe dieser Betrachtungen kann der Freeman Chain Code als vergleichbares Merkmal eingesetzt werden. Der Konturverlauf kann beliebig ausgewertet werden, oder der Berechnung weiterer Merkmale dienen: Die Krümmung enthält beispielsweise Informationen bezüglich der Winkeländerung einer Kontur [Ros74]. Die Biegeenergie E wird mittels

$$E = \frac{1}{p} \sum_{i=1}^{p} C^2(i) \tag{3.56}$$

berechnet und weist signifikante Werte für verschiedene Verläufe auf [uJV80].
Alternativen zur Konturrepräsentation sind die Annäherung durch Polygone [uJOE91], die Berechnung von Objektsignaturen [uCMB82], die Dekomposition der Kontur in stückweise stetige Funktionen [Pav77] oder der Einsatz von Gerüsten (engl. skeleton) [uTYZ]. Alle diese Techniken besitzen einen Nachteil gegenüber dem Freeman Chain Code: die Berechnung ist aufwendiger. Daher wird auf eine genauere Beschreibung verzichtet. Einen weiteren Typ von Deskriptoren stellen die Fourier Deskriptoren dar, welche basierend auf der Fourier Transformation der Kontur errechnet werden. Details dazu sind der Abhandlung der Deskriptoren [uBSR96] bzw. der Grundlagenarbeit [uCTZ72] zu entnehmen.

Farbbasierte Merkmale

Farben und Farbverteilungen stellen wichtige Deskriptoren dar. Eine weiße Bohne lässt sich z. B. über die Kontur nicht von einer Feuerbohne unterscheiden, über die Farbe ist diese Aufgabe sehr einfach zu lösen. Bei der Verwendung von Farbe sind mehrere Ansätze sinnvoll. Zum einen können globale oder regionale Farbstatistiken eines Objekts ausgewertet werden, beispielsweise über dessen Farbhistogramme. In [Ben85] wird diese Vorgehensweise verfolgt. Eine weitere Möglichkeit bietet sich an, wenn gewisse Farben gesucht werden bzw. auf Objekten nicht vorkommen dürfen. In diesem Fall denkt man sich einen 3-dimensionalen Farbraum, in dem die Farben durch die Koordinaten definiert werden (vergleiche mit dem RGB Farbmodell). In dem durch mehrere Dimensionen aufgespannten Merkmalraum können manuell oder automatisch Bereiche festgelegt werden, die gewünschte oder verbotene Farben enthalten. Mittels eines Distanzmaßes kann dann die Farbe eines Objektpixels mit diesen Farbbereichen verglichen werden. Gängige Distanzmaße sind die bekannte Euklidische Distanz $d_e(\vec{p}, \vec{q})$, die sich lt.

$$d_e(\vec{p}, \vec{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{3.57}$$

berechnet ([uLS98]). In der Gleichung stellt also n die Dimension dar. Eine weitere Variante ist die Mahalanobis Distanz

$$d_m(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \vec{C}^{-1} (\vec{x} - \vec{y})}, \tag{3.58}$$

worin $\vec{x}$ und $\vec{y}$ zwei Zufallsvektoren sind und $\vec{C}$ die zugehörige Kovarianzmatrix ist. In dieses Maß geht also die Korrelation der Variablen (bzw. in diesem Fall der Farben) mit ein [Mah36].

Fokusbasierte Merkmale

Fokusmerkmale messen die Unschärfe einer Objekttextur. Diese Problemstellung stammt aus Autofokusanwendungen, wie sie in Digitalkameras eingesetzt werden. Warum diese Merkmale für die Analyse von Feuerbohnen bedeutend sind, wird in Abschnitt 4 verdeutlicht. Hochpassfilter, welche Kanten hervorheben und homogene Regionen eines Objekts dämpfen, werden zur Erstellung eines Kantenbildes verwendet. Aus diesem kann dann ein Schärfemaß bestimmt werden, was meist durch Aufsummieren der Beträge passiert. Verschmierte Texturen weisen deutlich weniger Kantenmaxima auf als strukturierte Texturen. Als Filter stehen jene zur Auswahl, die auf der 1. Ableitung basieren, wie Sobel [uPH73], Roberts [Rob65] oder auch Prewitt [Pre70]. Basierend auf der 2. Ableitung arbeiten z.B. der Marr Hildreth Operator [Mar82] oder der Laplacian of Gaussian, welcher eine gauss gefilterte Maske verwendet. Als robustes Schärfemaß bietet sich das Tenengrad Fokusmaß an, welches durch

$$TEN = \sum_{x}^{X} \sum_{y}^{Y} [S(x,y)]^2, \bigvee_{S(x,y)^2} > T_h \tag{3.59}$$

definiert wird. Darin ist T_h ein Schwellwert und $S(x, y)$ der Betrag des in x und y Richtung gefilterten Bildes. An der Stelle (x, y) wird der Betrag demnach mit

$$S(x,y) = \sqrt{S_x(x,y)^2 + S_y(x,y)^2}, \quad \forall (x,y) \tag{3.60}$$

ermittlet. Diese Methode stammt aus [uVA07] bzw. [uWBS96]. Weitere Möglichkeiten zur Fokusabschätzung liefern die Methoden Varianz der Gradienten, Hesssche Matrix [uHN88] Strukturtensor [uMS88]. Egal, welcher Mechanismus zur Anwendung kommt, es ist ein globaler oder regionaler (d.h. Bestimmung mehrerer Werte) Einsatz denkbar.

3.2.5 Auswahlverfahren

Die letzten Ausführungen haben gezeigt, dass eine große Anzahl an berechenbaren Merkmalen zur Verügung steht. Gerade in der industriellen Sortierung können durch genaue Problemdefinition und ausreichende Erfahrung bereits eine geeignete Anzahl von Merkmalen gewählt werden, mit welchen vorhandene Objektklassen gut zu repräsentieren sind. Jedoch ist bei der manuellen Auswahl nicht bekannt, ob der verwendete Merkmalvektor optimal ist oder ob bessere Varianten (Kombinationen von Merkmalen) existieren. Die automatisierte Auswahl von Merkmalen kann aus einer gegebenen Menge von Merkmalen F jene Teilmenge F_s bestimmen, welche die verwendeten Daten bestmöglich repräsentiert. Es muss jedoch durch die Bereitstellung geeigneter Daten sichergestellt werden, dass der

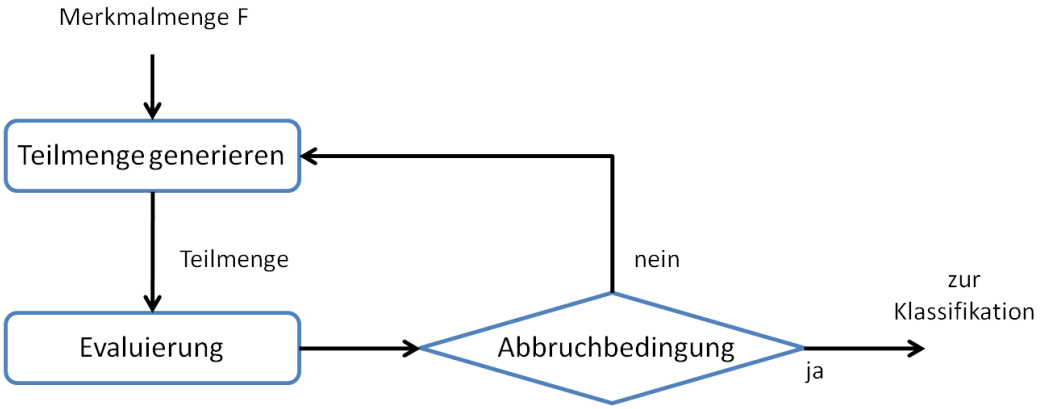

Abbildung 3.10: **Ablauf der Merkmalauswahl;** Jedem Algorithmus zur Merkmalauswahl wird zunächst die Menge aller verfügbaren Merkmale übergeben. Daraus werden iterativ neue Teilmengen gebildet, welche mittels einer Evaluierungsfunktion geprüft werden. Ist eine Teilmenge geeignet, wird sie zur späteren Verwendung weitergereicht, im anderen Fall verworfen.

entstehende Merkmalvektor auch die reale Welt (also alle gegebenen Objektklassen) repräsentieren kann.
Abbildung 3.10 zeigt die einzelnen Schritte eines Auswahlverfahrens. Aus der Menge aller Merkmale F wird in jedem Iterationsschritt eine Teilmenge F_s erzeugt. Diese Teilmenge wird evaluiert, wobei mit Hilfe einer Abbruchbedingung die Relevanz von F_s bezüglich der Klassifikationsaufgabe geprüft wird. Ist eine Menge F_s für diese Aufgabe relevant, kann mit ihr validiert werden. Die Generation von Teilmengen F_s kann auf mehrerlei Arten erfolgen:

- **Vollständig:** Alle möglichen Kombinationen der Merkmale von F werden als Teilmengen generiert. Je nach Dimension des Merkmalraums kann dieses Vorgehen zu aufwendig sein, jedoch werden alle Möglichkeiten geprüft.
- **Heuristisch:** Die Merkmale werden unter gewissen Konditionen kombiniert. Dies führt schneller zu einer nicht zwingend optimalen Lösung.
- **Zufällig:** Merkmale werden ohne definiertes Schema gewählt. Abhängig von der Anzahl der Versuche stellt sich ein anderes Ergebnis ein.

Die Evaluierung kann wie folgt durchgeführt werden:

- **Filter:** Die Merkmalmenge F_s wird über eine Maßzahl hinsichtlich ihrer Eignung zur Klassifikation geprüft. Denkbar sind hierbei Distanzmaße, der Informationsgehalt (Entropie), Korrelation zwischen Merkmalen und Daten, usw.
- **Wrapper:** Als Evaluierungsfunktion wird der Klassifikator selbst verwendet. Als Maßzahl wird hier z.B. die Fehlerrate der Klassifikation herangezogen.

Im Allgemeinen führt der Wrapper-Ansatz zu höherer Genauigkeit der Merkmalauswahl, jedoch auf Kosten der Generalität, wird doch schon ein spezieller Klassifikator zur Evaluierung eingesetzt. Weiters ist der Wrapper-Ansatz meist auch deutlich langsamer. Je nach

Wahl der Art der Teilmengengenerierung sowie der verwendeten Evaluierungsfunktion gibt es verschiedenste Algorithmen. Als Hilfestellung zur Auswahl der geeigneten Methode sind Datentyp, Datenmenge, Anzahl der Klassen und das Vorhandensein von Störungen (z.B. unvollständige Daten oder Ausreißer) in Betracht zu ziehen. Einige Beispiele sollen nun aufgelistet werden:

- **Sequentielle Vorwärtsauswahl:** Beginnend mit einer leeren Merkmalmenge wird zunächst für jedes einzelne Merkmal evaluiert. Jenes mit der besten Performance wird gewählt. Danach wird ein weiteres Merkmal so ergänzt, damit wiederum die beste mögliche Performance der Kombination aus zwei Merkmalen erreicht wird. Dies wird bis zur Abbruchbedingung wiederholt (z.B. in [JK94]).

- **Sequentielle Rückwärtsauswahl:** Hier startet man mit der Gesamtmenge F und entfernt iterativ Merkmale. Dabei wird jeweils das Merkmal entfernt, welches in einem maximalen Performancegewinn resultiert. Möglichkeiten der Implementierung zeigt [uCC00b].

- **Forwärts-Rückwärts-Auswahl:** Eine Kombination der ersten beiden Verfahren kompensiert deren gemeinsamen Nachteil einer nicht vollständigen Suche, da einmal hinzugefügte bzw. verworfene Merkmale nicht mehr entfernt bzw. erneut hinzugefügt werden können. Die Grundidee findet sich in [uAS09]

- **Las Vegas Algorithmus:** Dieser Algorithmus findet optimale Lösungen auch unter Störungseinfluß und verwendet den Ansatz der zufälligen Generation der Teilmenge. Eine Erläuterung gibt [uRS96].

- **Minimal Description Length Method (MDLM):** Mit Hilfe eines MDLM Kriteriums wird festgestellt, welche Teilmenge von Merkmalen ideal ist. Dabei werden alle möglichen Teilmengen geprüft. Die Methodik wurde erstmals in [uJS00] vorgestellt.

Einen Überblick dieser und weiterer Auswahlverfahren, deren Methodik und Implementierungen geben [uHL97] sowie [uHM07].
Am Ende dieses Abschnitts soll noch erwähnt werden, dass neben diesen Auswahlverfahren auch Methoden zur Reduktion der Dimension des Merkmalraumes angewandt werden können. Die Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA) führt eine Eigenwertzerlegung durch, mit deren Hilfe die Daten in einen niederdimensionalen Raum projiziert werden. Die Reduzierung des Informationsgehaltes steht jedoch nicht im direkten Zusammenhang mit der Klassifikation. Details zur PCA sind in [Jol02] ausgearbeitet. Weitere Verfahren zur Dimensionsreduktion sind die Faktorenanalyse (z.B. in [Hat94]), die Multidimensionale Skalierung (Grundlagen sowie Anwendungen gibt [uPG05]) sowie die Diskriminanzanalyse, welche erstmals in [Fis37] vorgestellt wurde.

3.3 Maschinelles Lernen für Sortieranwendungen

Die mathematische Repräsentation von Feuerbohnen (oder anderen Objekten) kann mit Merkmalvektorn $\vec{f}$ erfolgen, wie sie in Gleichung 3.54 definiert wurden. Anhand von $\vec{f}$ muss jedem Objekt eine der vorhandenen Klassen zugewiesen werden. Diese Aufgabe übernimmt

ein Klassifikator. Klassifikatoren bedienen sich der Methodik, ähnliche Beobachtungen zusammenzufassen und abweichende Beobachtungen zu unterscheiden. Das Themengebiet wird häufig auch als Mustererkennung bezeichnet und findet sich mit den meisten seiner Techniken im Standardwerk von [uDGS01]. In diesem Kapitel werden nach der Erläuterung wichtiger Grundbegriffe zwei verbreitete Arten des Lernens vorgestellt, mit deren Hilfe sich Klassifikatoren für Sortieraufgaben finden lassen. Danach werden kurz einige Klassifikatoren bzgl. ihrer Eigenschaften unterschieden.

3.3.1 Grundlagen des maschinellen Lernens

Ein Klassifikator ist eine Funktion Θ, welche den Merkmalraum in Entscheidungsregionen bzw. Klassen einteilt. Eine Klassifikation ist demnach eine Abbildung

$$\Theta : \vec{f} \rightarrow \Omega, \tag{3.61}$$

wobei $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ den Entscheidungsraum mit k Klassen darstellt. Bei der Klassifikation von Objekten der Realität kann Θ nicht direkt aufgestellt werden, falls für die Objektklassen keine Schablonen zur Verfügung stehen. In diesem Fall kann Θ mit einem Lernverfahren gefunden werden. Einem Lernverfahren müssen Daten zugeführt werden, welche alle vorkommenden Objektklassen in Form von Merkmalvektoren beschreiben. Die Menge dieser Trainingsbeispiele T ermöglicht dem Lernverfahren also, die Klassifikatorfunktion Θ an die Aufgabenstellung anzupassen, um im späteren Betrieb selbstständig die Klassenzuweisung durchführen zu können.

Formalisierung des Lernens

Ein Lernverfahren bzw. Lernalgorithmus $\mathcal{A}$ passt einen Klassifikator Θ anhand einer Menge von Trainingsbeispielen T an. Die genaue Definition von T hängt von der Art des Lernens ab (siehe folgende Abschnitte). Dies kann mit

$$\mathcal{A} : T \rightarrow \Theta \tag{3.62}$$

ausgedrückt werden. Je nach Gestalt von Θ ergeben sich verschiedene Typen von Klassifikatoren (z.B. linearer Klassifikator). Die Anpassung von Θ wird unter Zuhilfenahme einer Lernregel durchgefürt. Wird z.B. ein binärer linearer (2 Klassen) Klassifikator

$$\Theta = sgn(\vec{w}^T \vec{f}) \tag{3.63}$$

erlernt, so bedeutet dies die Einstellung der Gewichte $\vec{w}$, welche mit den Merkmalen $\vec{f}$ multipliziert werden. Dabei gilt $\vec{w}, \vec{f} \in \mathbb{R}^n$, wobei n die Dimension des Merkmalraums angibt. Eine Lernregel bestimmt nun, auf welche Weise $\vec{w}$ zu verändern ist. So gibt z.B. die Perceptron Lernregel vor, dass die Gewichte anzupassen sind, wenn ein Merkmalvektor zu einer falschen Klasse zugewiesen wird. Der Lernvorgang selbst läuft meist iterativ ab, d.h. die Parameter des Klassifikators werden solange aktualisiert, bis eine Abbruchbedingung erreicht ist. Eine häufig verwendete Vorgehensweise ist die Minimierung des empirischen Fehlers (Schätzung des wahren Fehlers), welcher lt.

$$error_T = \frac{1}{T} \sum_{i=1}^{t} E(\omega_i, \omega_{i,W}) \tag{3.64}$$

definiert wird. Es handelt sich also um ein Optimierungsproblem. Alle einzelnen Fehlklassifikationen

$$E(\omega, \omega_W) = \begin{cases} 0, & \omega = \omega_W \\ 1, & \omega \neq \omega_W \end{cases} \tag{3.65}$$

werden dazu über die Anzahl der Trainingsbeispiele t gemittelt. Dabei gibt ω das Klassifikationsergebnis und ω_W die tatsächliche Objektklasse an. Diese Methodik geht davon aus, dass jedes Beispiel aus T unabhängig und zufällig generiert wird. Durch eine geeignete Anzahl an Trainingsbeispielen t kann dadurch der wahre Fehler angenähert werden. Eine weitere Möglichkeit besteht in der Minimierung des mittleren quadratischen Fehlers (MSE).
Nach dem Lernen steht Θ zur Klassifikation zur Verfügung und kann mit einer neuen Datenmenge, der Testdatenmenge V evaluiert werden. Dabei muss $T \cap V = \emptyset$ gelten. Mit V kann also die Qualität des Klassifikators geprüft werden.

Optimierungsaufgaben

Sucht man nach einem Maximum bzw. Minimum einer Funktion $i(\vec{x})$ unter der Einhaltung von eventuell gegebenen Nebenbedingungen, so spricht man von Optimierung. Dabei ist zwischen lokaler und globaler Optimierung zu unterscheiden. Erstere sucht nach einem lokalen Extremum, während die Suche nach einem globalen Extremum das Ziel einer globalen Optimierung ist. Die zu optimierende Funktion $i(\vec{x})$ wird Zielfunktion genannt. Liegt eine lineare Zielfunktion

$$i(\vec{x}) = i_0 + c_1 x_1 + ... + c_n x_n \tag{3.66}$$

vor, so spricht man von linearer Optimierung. Hierbei müssen die Nebenbedingungen ebenfalls lineare Gleichungen bzw. Ungleichungen sein. Die Koeffizienten in Gleichung 3.66 sind bekannt und erfüllen $c_k, i_o \in \mathbb{R}$. Die Parameter der Zielfunktion sind x_k. Soll diese Funktion maximiert werden, so kann die Schreibweise

$$\max_x i_0 + \vec{c}^T \vec{x} \tag{3.67}$$

verwendet werden, wobei noch Nebenbedingungen

$$\begin{aligned} \mathbf{A}\vec{x} &\leq \vec{b}, \mathbf{A} \in \mathbb{R}^{m,n}, \vec{b} \in \mathbb{R}^m \\ \vec{b} &\geq \mathbf{o} \\ \vec{x} &\geq \mathbf{o}, \vec{x} \in \mathbb{R}^n \end{aligned} \tag{3.68}$$

zu definieren sind. Die Nebenbedingungen können in der Vektorfunktion $h(\vec{x})$ zusammengefasst werden. Die Schreibweise $\vec{x} \geq \mathbf{o}$ bedeutet, dass für alle Komponenten von $\vec{x}$ $x_i \geq 0$ gelten muss. Jeder Vektor $\vec{x}$ wird als zulässige Lösung bezeichnet, falls dieser alle gegebenen Nebenbedingungen erfüllt. Eine Lösung ist optimal, wenn sie zulässig ist und wenn die Zielfunktion mit dieser Lösung den optimalen (in diesem Beispiel maximalen) Wert annimmt.
Analog ist eine Minimierungsaufgabe festzulegen. Optimierungsaufgaben werden mit verschiedensten Algorithmen gelöst, einen guten Überblick liefert [uJS03], während die Thematik in [uLS09] sehr anwendungsorientiert vermittelt wird. Die gegebene Einführung ist

an [Bar01] angelehnt. Entgegen dem gezeigten Beispiel handelt es sich um eine nicht lineare Optimierung, falls die Zielfunktion nicht linear ist.
Bei Optimierungsaufgaben ist das Prinzip der Dualität von Bedeutung. Mit Dualität bezeichnet man die Tatsache, dass zu einer Optimierung

$$\min_x i(\vec{x}) \tag{3.69}$$

mit Nebenbedingungen $h(\vec{x})$ eine duale Optimierung

$$\max_\lambda \min_x \mathcal{L}(\vec{x}, \vec{\lambda}) \tag{3.70}$$

existiert. Dabei ist

$$\mathcal{L}(\vec{x}, \vec{\lambda}) = i(\vec{x}) + \vec{\lambda}^T h(\vec{x}) \tag{3.71}$$

die zur Optimierung gehörige Lagrange Funktion und $\vec{\lambda}$ der Vektor der Langrange Multiplikatoren (auch: duale Variablen). Der Langrangesatz besagt (z.B. aus [Bar01] zu entnehmen), dass sich das Optimierungsproblem aus Gleichung 3.69 unter $h(\vec{x}) = 0$ nur durch jene Vektoren $\vec{x}$ lösen lässt, an deren Stellen Lagrange Multiplikatoren λ_i existieren, für welche

$$\nabla_x \mathcal{L}(\vec{x}, \vec{\lambda}) = \nabla i(\vec{x}) + \sum_i \lambda_i \nabla h_i(\vec{x}) = 0 \tag{3.72}$$

gilt. Bei der Lösung des dualen Problems erlaubt man die Verletzung von Nebenbedingungen. Im Ausgleich dazu werden der Zielfunktion jedoch Kosten auferlegt. Für die Lösung linearer Probleme ergeben sich damit zwei äquivalente Ansätze.

3.3.2 Überwachtes Lernen

Eine Möglichkeit des Lernens ist es, die Trainingsdaten mit deren richtiger Klassenzugehörigkeit zu versehen. Ein Testdatum kann demnach als ein Tupel $\{\vec{f}, \omega\}$ aufgefasst werden, also der Merkmalvektor plus die Klasse. Der Klassifikator erhält eine Menge von Trainingsdaten $T = \{\{\vec{f_1}, \omega_1\}, ..., \{\vec{f_t}, \omega_t\}\}$ mit bekannten Klassen, um sich zu parametrisieren. Diese Menge enthält t Tupel. Nach dem Lernen werden v Testdaten $V = \{\vec{f_1}, ..., \vec{f_v}\}$ mit unbekannter Klasse verwendet, um die Performance des Klassifikators zu prüfen. Um zu klassifizieren kann auch die A-priori Wahrscheinlichkeit $P(\omega_i)$ einer Klasse ω_i miteinbezogen werden. Mit Hilfe der A-posteriori Wahrscheinlichkeit, die sich zu

$$P(\omega_i \mid \vec{f}) = \frac{p(\vec{f} \mid \omega_i) P(\omega_i)}{p(\vec{f})} \tag{3.73}$$

berechnet, kann bereits ein Klassifikator entwickelt werden. Sie gibt also die Wahrscheinlichkeit an, dass eine Klasse ω_i unter Beobachtung des Vektors $\vec{f}$ vorliegt. In der Formel ist $p(\vec{f})$, die Wahrscheinlichkeit des Auftretens der Beobachtung $\vec{f}$ bezüglich aller n Klassen mit

$$p(\vec{f}) = \sum_{i=1}^{n} p(\vec{f} \mid \omega_i) P(\omega_i) \tag{3.74}$$

definiert. Da die Summe der A-posteriori Wahrscheinlichkeiten aller Klassen immer 1 ergeben muss, kann man einen Klassifikator für zwei Klassen mit

$$\Theta = \begin{cases} \omega_1, & P(\omega_1 \mid \vec{f}) > P(\omega_1 \mid \vec{f}) \\ \omega_2, & sonst \end{cases} \tag{3.75}$$

festlegen. Dabei errechnet sich die Fehlerwahrscheinlichkeit zu

$$P(Fehler \mid \vec{f}) = \begin{cases} P(\omega_1 \mid \vec{f}), & \Theta = \omega_2 \\ P(\omega_2 \mid \vec{f}), & \Theta = \omega_1 \end{cases}. \tag{3.76}$$

Eine große Klasse von Klassifikatoren bilden die Diskriminanzfunktionen. Dabei besteht der Klassifikator zur Einteilung von n Klassen aus n Entscheidungsfunktionen. Eine Beobachtung $\vec{f}$ wird auf jede dieser Funktionen angewandt und jener mit dem höchsten Ergebnis zugewiesen. Für zwei Klassen kommen die Funktionen $g_1(\vec{f})$ und $g_2(\vec{f})$ zum Einsatz, die Entscheidung erfolgt entsprechend

$$\Theta = \begin{cases} \omega_1, & g_1(\vec{f}) > g_2(\vec{f}) \\ \omega_2, & sonst \end{cases}. \tag{3.77}$$

Definiert man die Diskriminanzfunktionen über die Bayesfunktion zu

$$g_i(\vec{f}) \equiv P(\omega_i \mid \vec{f}) = \frac{p(\vec{f} \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{n} p(\vec{f} \mid \omega_j)P(\omega_j)}, \tag{3.78}$$

kann durch Logarithmieren und Streichen des konstanten Nennerterms die Funktion zu

$$g_i(\vec{f}) = \ln(p(\vec{f} \mid \omega_i)) + \ln(P(\omega_i)) \tag{3.79}$$

gefunden werden. Im Falle von zwei Klassen, in welchem der Klassifikator als Dichotomizer bezeichnet wird, folgt

$$g(x) = P(\omega_1 \mid \vec{f}) - P(\omega_2 \mid \vec{f}) = \ln \frac{p(\vec{f} \mid \omega_1)}{p(\vec{f} \mid \omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}, \tag{3.80}$$

woraus sich eine einfache Version des Bayes Klassifikators durch

$$\Theta = \begin{cases} \omega_1, & g(x) > 0 \\ \omega_2, & sonst \end{cases} \tag{3.81}$$

festlegen lässt. Man sieht in der Berechnung dieser Diskriminanzfunktionen, dass sowohl die bedingten Klassenwahrscheinlichkeiten $p(\vec{f} \mid \omega_i)$ als auch die A-priori Wahrscheinlichkeiten $P(\omega_i)$ bekannt sein müssen. Da diese Wahrscheinlichkeiten in der Praxis meist unbekannt sind, müssen sie abgeschätzt werden. Die Techniken dazu können in hier nur aufgezählt werden, finden sich aber alle in [uDGS01] eingehend behandelt wieder. Man teilt hierbei in parametrische Methoden und nicht parametrische Methoden ein. Erstere modellieren Daten mit Hilfe von Wahrscheinlichkeitsmodellen und schätzen deren Parameter ab. Werden Daten zB. mit der Normalverteilung beschrieben, sind Mittelwert und Varianz abzuschätzen. Eine sehr bekannte Methode stellt die Maximum Likelihood Schätzung dar. Wird auf die Verwendung von Wahrscheinlichkeitsverteilungen verzichtet, so kommen Techniken wie der k-Nearest Neighbour Algorithmus oder das Parzen Window zum Einsatz. Diese Ansätze werden bevorzugt, wenn sich die Daten nur schlecht bzw. nicht durch Verteilungsmodelle beschreiben lassen.

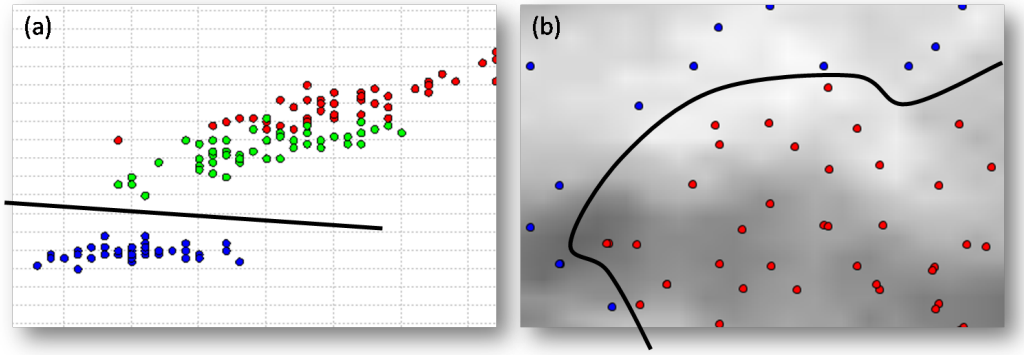

Abbildung 3.11: **Separation von Datensätzen;** In (a) lässt sich der blaue Datensatz durch eine lineare Ebene von den restlichen Klassen trennen. Dabei sind die unterschiedlichsten Ebenen denkbar. Der grüne und der rote Datensatz lassen sich zwar auch linear, jedoch nicht ohne Klassifikationsfehler trennen. Bild (b) gibt ein Beispiel einer nichtlinearen Separationsebene.

Support Vektor Maschine

Einen speziellen Mechanismus von Diskriminanzfunktionen verwenden Support Vektor Maschinen (SVM), welche in der Klassifikation mittlerweile weit verbreitet sind. Die folgende Abhandlung lehnt sich stark an [uCC00a] an, welche einen guten Überblick über die Grundlagen und Anwendungsgebiete gibt. SVM sind sowohl für binäre Klassifikationen (2 Klassen), als auch für die Multiklassenklassifikation geeignet. Da für das Sortierproblem die binäre Klassifikation ausreicht, beschränkt sich die Abhandlung auf diesen Typ.
Da die SVM aus der Familie der Diskriminanzverfahren stammt, ist das Grundprinzip ein Analoges. Gesucht wird eine Trennfunktion, welche die verschiedenen Klassen optimal separiert. Diese stellt im 2-dimensionalen Fall eine Gerade dar und reicht bis zu einer Hyperebene beliebig hoher Dimension. Wird eine lineare Trennfunktion verwendet, spricht man von linearer SVM. Ein Trainingstupel einer binären Klassifikation ist durch $\{\vec{f_i}, y_i\}$ gegeben, wobei $\vec{f_i} \in \mathbb{R}^d$ einen Merkmalvektor und $y_i = \{-1, +1\}$ das Klassenlabel darstellt. Die Trainingsmenge besteht aus n Daten, als Klassifikationsregel wird

$$\Theta(\vec{f}) = sgn(\vec{w} \cdot x + b) \tag{3.82}$$

definiert. Darin bestimmt $\vec{w}$ die Orientierung der Trennebene und b den Offset der Ebene. Die lineare Separierbarkeit von Daten stellt den einfachsten Fall eines Klassifikationsproblems dar, daher soll auch dieser zuerst betrachtet werden. Abbildung 3.11 (a) zeigt einen solchen Fall, aus dem hervorgeht, dass beliebig viele Separationsebenen denkbar sind. Man sucht nun jene Hyperebene, die am Weitesten von allen Klassendaten entfernt liegt. Zwei Ansätze führen dabei zur gleichen Lösung (siehe Abbildung 3.12). Durch Berechnen der konvexen Hülle aller Klassen lassen sich jene Punkte finden, welche sich am Nächsten zu Punkten anderer Klassen befinden. Man findet demnach die ideale Trennebene durch

$$\vec{w} = \vec{d} - \vec{c}, \tag{3.83}$$

wenn $\vec{d}$ und $\vec{c}$ diese Forderung erfüllen. Dies lässt sich als quadratisches Minimierungsprolem

$$\min_{\alpha} \frac{1}{2}\|\vec{c} - \vec{d}\| \tag{3.84}$$

darstellen und berechnen, wobei

$$\sum_{y_i=1} \alpha_i \vec{f_i}, \quad \sum_{y_i=-1} \alpha_i \vec{f_i} \tag{3.85}$$

die Punkte der konvexen Hülle der beiden Klassen repräsentieren und die Minimierung unter den Bedingungen

$$\sum_{y_i=1} \alpha_i = 1, \quad \sum_{y_i=-1} \alpha_i = 1, \quad \alpha_i \geq 0, \quad i = 1, 2, .., m \tag{3.86}$$

durchzuführen ist. Man sieht aus Abbildung 3.12, das für die Lösung nur die Punkte im Grenzbereich (die Support Vektoren) interessant sind, andere Punkte werden nicht benötigt. Ganz gleich verhält es sich mit der zweiten Möglichkeit, die Hyperebene zu finden. Dabei maximiert man den Abstand zweier paralleler Trennebenen. Eine Ebene ist als Trennebene geeignet, wenn alle Punkte einer Klasse auf einer Seite der Ebene liegen und vice versa. Für die Klasse $y = 1$ ist demnach b und $\vec{w}$ so zu bestimmen, dass $\vec{w} \cdot \vec{f_i} - b > k$ sind. Man kann k als den optimalen Wert der Minimierung betrachten und diesen beliebig skalieren. Somit kann man die beiden Ebenen durch

$$\begin{aligned} \vec{w} \cdot \vec{f_i} - b &\geq 1 \\ \vec{w} \cdot \vec{f_i} - b &\leq -1 \end{aligned} \tag{3.87}$$

beschreiben. Diese werden durch die Abstandsmaximierung so weit wie möglich von der eigenen Klasse weggeschoben. Die beiden bestimmten Ebenen $\vec{w} \cdot \vec{f_i} = b+1$ und $\vec{w} \cdot \vec{f_i} = b-1$ liegen dann im Abstand

$$\gamma = \frac{2}{\|\vec{w}\|} \tag{3.88}$$

zueinander. Damit zeigt sich, dass die Minimierung

$$\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \tag{3.89}$$

das Problem auf die gleiche Weise lösen kann. Die notwendigen Bedingungen

$$\begin{aligned} \vec{w} \cdot \vec{f_i} &\geq b + 1, y_i = 1 \\ \vec{w} \cdot \vec{f_i} &\leq b - 1, y_i = -1 \end{aligned} \tag{3.90}$$

können vereinfacht lt.

$$y_i(\vec{w} \cdot \vec{f_i} - b) \geq 1 \tag{3.91}$$

geschrieben werden. Da diese beiden Ansätze zur gleichen Lösung führen, werden sie häufig als duale Probleme bezeichnet. Eine weitere Dualität, die bei der Verwendung von SVM

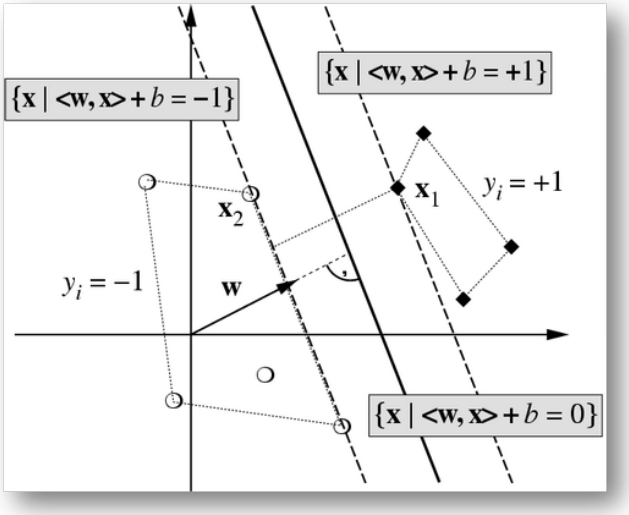

Abbildung 3.12: **Berechnung von Separationseben;** Die konvexen Hüllen der Klassen ∘ und ⧫ sind punktiert eingezeichnet. Die Punkte der Klassen, welche der jeweiligen anderen Klasse am Nächsten liegen, sind $\vec{x}_2$ und $\vec{x}_1$. Damit kann die mittlere Trennebene gefunden werden. Ansatz 2 bestimmt die maximale Distanz γ zwischen den beiden gestrichelten, parallelen Ebenen. Die Abbildung stammt aus [uAJS02].

oft eingesetzt wird, ist die Überführung der Minimierungsaufgabe in eine Lagrange Optimierung (zB. in [Tim98]), welche durch

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \vec{f_i} \cdot \vec{f_j} - \sum_{i=1}^{m} \alpha_i \tag{3.92}$$

gegeben ist. Dabei gibt m wiederum die Anzahl der Trainingsdaten an, α_i sind die Lagrange Multiplikatoren. Diese Schreibweise wird auch hier in weiterer Folge verwendet und kann über die Lagrangeformel hergeleitet werden. Die Minimierungsbedingungen sind

$$\sum_{i=1}^{m} y_i \alpha_i = 0, \quad \alpha_i \geq 0 \quad i = 1, 2, ..., m. \tag{3.93}$$

Egal welcher dieser drei Ansätze verwendet wird, findet man die gesuchte Hyperebene

$$\vec{w} = \sum_{i=1}^{m} y_i \alpha_i \vec{f_i} + b, \tag{3.94}$$

wobei sich der Offset b über die Supportvektoren finden lässt. Somit ist für den Fall der linearen Separierbarkeit die Lösung gefunden.
Sind die Daten nicht linear separierbar (was oft der Fall ist), kann man die Bedingungen der Optimierung etwas lockern. Man erlaubt durch die Einführung von Schlupfvariablen einzelne falsche Klassifikationen und erreicht dadurch, dass dennoch die Fehlerwahrscheinlichkeit minimiert wird. Dazu genügt es, die Minimierungsbedingung aus Gleichung 3.93

zu

$$C \geq \alpha_i \geq 0 \tag{3.95}$$

umzuschreiben. Es wird also durch die Einführung einer Komplexitätskonstante C eine obere Schranke für α_i festgelegt. Mit den bisherigen Formulierungen kann eine separierende Hyperebene beliebiger Dimension gefunden werden. Der Mechanismus wird als lineare SVM bezeichnet.
Sind die Daten im Merkmalraum derart angeordnet, dass keine sinnvolle lineare Separationsebene gefunden werden kann, so kann das Konzept erweitert werden, ohne dabei auf die Vorteile der linearen SVM verzichten zu müssen. Angenommen ein Klassifikationsproblem lässt sich durch eine quadratische Diskriminante lösen, der Merkmalraum selbst sei im $\mathbb{R}^2$. Durch Hinzufügen von weiteren Merkmalen, welche quadratische Funktionen der ursprünglichen Merkmale sind, kann der Merkmalraum entsprechend

$$[r, s] \rightarrow [r, s, rs, r^2, s^2] \tag{3.96}$$

abgebildet werden. Im so entstandenen Raum im $\mathbb{R}^5$ wird eine lineare Diskriminante berechnet, welche natürlich im $\mathbb{R}^2$ nicht mehr linear ist. Diese Abbildung wird formal durch

$$\Phi(\vec{f}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'} \tag{3.97}$$

beschrieben, wobei $d' > d$ gilt. Im konkreten Beispiel ist

$$\Phi(\vec{f}) = [r, s, rs, r^2, s^2] \tag{3.98}$$

wodurch der zugehörige Klassifikator zu

$$\Theta(\vec{f}) = sgn(\vec{w} \cdot \Phi(\vec{f}) - b) = sgn(w_1 r + w_2 s + w_3 rs + w_4 r^2 + w_5 s^2) \tag{3.99}$$

wird. Er ist also linear im abgebildeten Raum und nicht linear im ursprünglichen Merkmalraum, dort ist er quadratisch. Der Nachteil dieser Lösung ist das exponentielle Dimensionswachstum der Abbildung, welches zu Overfitting Problemen führen kann. Weiters ist die Berechnung von $\Phi(\vec{f})$ mitunter außerst komplex. Um diese zu umgehen, führt man die Verwendung eines Kernel ein[8]. Gleichung 3.92 hat mit einem abgebildeten Merkmalraum die Form

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \Phi(\vec{f_i}) \cdot \Phi(\vec{f_j}) - \sum_{i=1}^{m} \alpha_i \tag{3.100}$$

und verwendet die modifizierte Bedingung aus Gleichung 3.95. Nun wird eine Kernelfunktion $K(u, v)$ eingeführt. Das Mercer Theorem besagt, dass für eine Abbildung Φ und zwei Vektoren $\vec{u}$ und $\vec{v}$ das Inprodukt der abgebildeten Punkte $\Phi(\vec{u})$ und $\Phi(\vec{v})$ durch einen Kernel gefunden werden kann, ohne die Abbildungsfunktion selbst zu kennen. Diese Vorgehensweise bei der Verwendung von SVM wird zB. in [uVV95] gezeigt. Es gilt also der Zusammenhang

$$\Phi(\vec{u}) \cdot \Phi(\vec{v}) = K(\vec{u}, \vec{v}). \tag{3.101}$$

Kernelfunktionen gibt es in vielen Varianten, z.B. wird bei einer Abbildung in Form eines Polynoms des Grades p der Kernel

$$K(u, v) = (\vec{u} \cdot \vec{v} + 1)^p \tag{3.102}$$

[8] Dieser Ansatz wird in der Literatur meistens als Kerneltrick bezeichnet.

verwendet. Substituiert man nun in Gleichung 3.100, so erhält man

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(\vec{f_i}, \vec{f_j}) - \sum_{i=1}^{m} \alpha_i \tag{3.103}$$

bei Verwendung der gleichen Optimierungsbedingungen. Durch die Wahl einer geeigneten Kernelfunktion lassen sich beliebige nichtlineare Klassifikatoren erstellen, welche mit unveränderten, robusten Algorithmen der linearen SVM realisiert werden können. Der Klassifikator dieser SVM entspricht nun

$$\Theta(\vec{f}) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i K(\vec{f}, \vec{f_i}) - b\right). \tag{3.104}$$

Mit der vorgestellten Methodik lassen sich nicht nur Klassifikationsprobleme lösen, die SVM wird z.B. auch für Regressionen eingesetzt, wie [uRC01] zeigt. Zur Implementierung müssen nur wenige Parameter eingestellt werden. Wird der Kernel aus Gleichung 3.102 verwendet, so müssen lediglich C und p vorgegeben werden. Die Wahl von C kann zB. durch Cross-Validierung erfolgen. Sehr positiv wirkt sich weiters die Tatsache aus, dass auf Grund der linearen Optimierungsalgorithmen keine Probleme mit lokalen Minima auftreten. Der Algorithmus bleibt also nicht stecken, wie dies bei Alternativen wie Neuronalen Netzen der Fall sein kann. Wie viele Arbeiten zur SVM zeigen, werden sehr stabile, reproduzierbare Ergebnisse erreicht.

3.3.3 Unüberwachtes Lernen

Das überwachte Lernen bringt eine große Einschränkung mit sich. Trainingsdaten müssen (meist händisch) mit den korrekten Klassen versehen werden. Je nach Art der Problemstellung kann dies zu einem unrealistischen Aufwand ausarten. Dem kann man entgegenwirken, indem man auf die Klassenzuweisung verzichtet und die Einteilung der Daten ganz dem Lernalgorithmus überlässt. Der Merkmalraum wird selbstständig in Klassenregionen, sogenannte Cluster unterteilt. Nach dem Trainieren entscheidet man dann über ein Distanzmaß, zu welcher Klasse eine neue Beobachtung $\vec{f}$ zuzuweisen ist. Wiederum wird ein Ansatz entsprechend der im vorhinein bekannten Parameter gewählt. Beispielsweise kann die Anzahl der Cluster (also Klassen) festgelegt werden oder nicht. Weiters können die Cluster wiederum durch Wahrscheinlichkeitsmodelle parametrisiert werden oder eine unparametrische Form annehmen. Das hierachische Clustern (in [uDGS01]) nimmt zu Beginn des Lernens jeden Punkt im Merkmalsraum als einen eigenen Cluster an und verschmelzt schrittweise Punkte entsprechend einem Distanzmaß (Bottom-up Version). Das k-means Clustering Verfahren [uLS03] setzt zufällig gewählte Clusterzentren im Merkmalraum und sucht die n nächsten Nachbarn. Daraufhin werden die Zentren (d.h. die Mittelwerte) neu berechnet. Dies erfolgt bis zum Erfüllen einer Abbruchbedingung. Man sieht ein, dass die Wahl der Startzentren hier in das Endergebniss miteingeht. Erwähnenswert ist an dieser Stelle auch das Mean-Shift Verfahren, ein sehr breit anwendbares Clusteringverfahren, welches auf Dichteschätzungen basiert. Die Arbeit von [uPM02] führt in diese Thematik ein.

3.3.4 Klassifikatoren für die Sortierpraxis

Nach dem kurzen Überblick über Lernverfahren sollen nun einige Eigenschaften verschiedener Klassifikatoren aufgelistet werden, um die Auswahl zu erleichtern. Die Angaben sind sehr allgemein gehalten, da sich auf Grund hoher Anwendungsabhängigkeit nur globale Tendenzen ableiten lassen.

- **Neuronale Netze (NN):** In neuronalen Netzen werden aus simplen Elementen komplexe Systeme aufgebaut. Lernen erfolgt durch Verändern der Gewichte der Verbindungen zwischen den Netzwerkelementen. Es wird eine verzweigte und parallele Berechnung erreicht, wodurch sich im Betrieb ein effizientes Verhalten bzgl. der Rechenzeit und des Speicherbedarfs ergibt. Auch ist ein solches System robust gegenüber fehlerhaften Daten. Die Lernphase dauert relativ lange. NN sind auf Grund teilweise hoher Komplexität schwer interpretierbar. Die Praxis zeigt, dass NN bei Klassifikationsaufgaben häufig schlechter abschneiden als die SVM.

- **Nearest Neighbor Klassifikator:** Die Klassifikation geschieht auf Basis der bereits vorhandenen Merkmalvektoren. Ein neuer Merkmalvektor wird jener Klasse zugeordnet, deren k Datenvektoren den geringsten Abstand zum neuen Vektor aufweisen. Durch diese Vorgehensweise wird praktisch keine Lernzeit benötigt, jedoch ist die Klassifikation vieler Merkmalvektoren entsprechend aufwändig. Da alle Vektoren zu speichern sind, ist das Verfahren speicherineffizient. Bei ausreichend vorhandenen Trainingsdaten zeigen sich gute Klassifikationsergebnisse.

- **Entscheidungsbäume:** Bäume kommen im Speziellen bei nominalen Merkmalen zum Einsatz. Der Baum wird im Zuge des Lernens aufgebaut. Für kontinuierliche Klassifikationsfunktionen können keine Bäume verwendet werden. Das Lernen und Testen ist sehr aufwendig, hingegen ist der Speicherbedarf gering. Entscheidungsbäume sind sehr gut zu interpretieren.

- **SVM:** Die SVM wurde bereits ausführlich besprochen. Zusammengefasst zeigt sich eine häufig gute Klassifikationsperformance, ein hoher Lernaufwand, dafür jedoch ein rechenextensiver Klassifikationsbetrieb. Es wird nur sehr wenig Speicher benötigt.

3.4 Echtzeitsysteme

In den bisherigen Ausführungen wurden Prinzipien der bildgestützen Analyse vorgestellt. Um ein praxistaugliches System zur Sortierung von Feuerbohnen zu entwickeln, muss zum Einen eine korrekte Klassifikation der Proben erreicht werden. Wie aus Kapitel 1 hervorgeht, werden die Proben im freien Fall durch Zeilensensoren erfasst und je nach Ergebnis der Klassifikation von pneumatischen Auswerfern auf mehrere Behältnisse verteilt. Dies muss ohne Frage nach einer genauen zeitlichen Deterministik erfolgen. Je nach räumlichem Abstand von Sensoren und Auswerfern gilt es ein Zeitfenster einzuhalten, innerhalb dessen die Klassifikation zu erfolgen hat. Dieses Beispiel zeigt, dass es sich um ein Echtzeitsystem handelt, welches alle Aufgaben in definierten Zeiten zu erfüllen hat. In diesem Abschnitt sollen die Grundbegriffe von Echtzeitsystemen angeschnitten werden, weiters sollen Überlegungen zur Realisierung einer deterministischen Software für die Qualitätsanalyse getätigt werden.

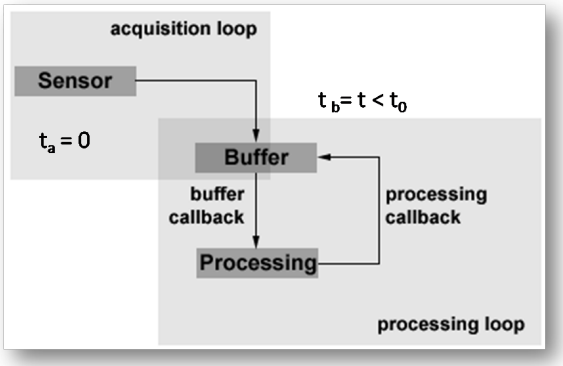

Abbildung 3.13: **Echtzeitsoftware;** Wird von einer Bildverarbeitungssoftware ein Echtzeitverhalten verlangt, so ist ein Schleifendurchlauf innerhalb einer gegebenen Zeit t_0 abzuarbeiten. Diese Schleife inkludiert die Speicherung der Bilddaten in einem Puffer und die anschließende Bearbeitung $t_a \rightarrow t_b$.

3.4.1 Echtzeit Grundlagen

Der Begriff Echtzeit sagt nichts darüber aus, wie schnell eine Aufgabe auszuführen ist bzw. ausgeführt wird. Wenn eine Steuerung einmal pro Stunde Daten an einen Aktuator senden muss, um die Funktion eines Gerätes zu gewährleisten, ist dies ebenfalls ein Echtzeitsystem. D.h. es existiert eine Vorschrift bzgl. einer zeitlichen Deterministik. Je nach Systemverhalten bei Mißachtung dieser Vorgaben teilt man Echtzeitsysteme ein. Ein System ist ein hartes Echtzeitsystem, wenn eine Verletzung der Zeitvorgaben eine korrekte Ausführung verhindert bzw. sogar zu Schäden führen kann. Dem gegenüber stehen weiche Echtzeitsysteme, die zwar aus Performancegründen ein Echtzeitverhalten benötigen, jedoch bei Mißachtung keine Gefährdung für das Systemverhalten ergeben. Beispiele von Echtzeitanforderungen sind zyklisch auszuführende Aufgaben (ein Task ist alle $10ms$ auszuführen), und Aufgaben, welche bis zu einer Deadline erledigt sein müssen.
Eine Sortiermaschine hat viele Echtzeitvorgaben, dabei ist das Zusammenspiel von Bilderfassung und Auswerfern das Kritischste. Angenommen, ein Probeobjekt benötigt im freien Fall vom Sensor bis zum Auswerfer die Zeit t_0[9], dann sind innerhalb einer Zeit $t \leq t_0$ folgende Aufgaben zu erledigen:

- Erfassung durch den Sensor
- Transport der Daten zum Computer
- Zwischenspeichern der Daten
- Analyse der Daten, Berechnung der Objektmerkmale
- Entscheidungsfindung des Klassifikators
- Gegebenenfalls: Aktivierung der pneumatischen Auswerfer und Warten auf die Ventilöffnung

[9]Die Bestimmung der Fallzeit t_0 wird in Anhang A erläutert.

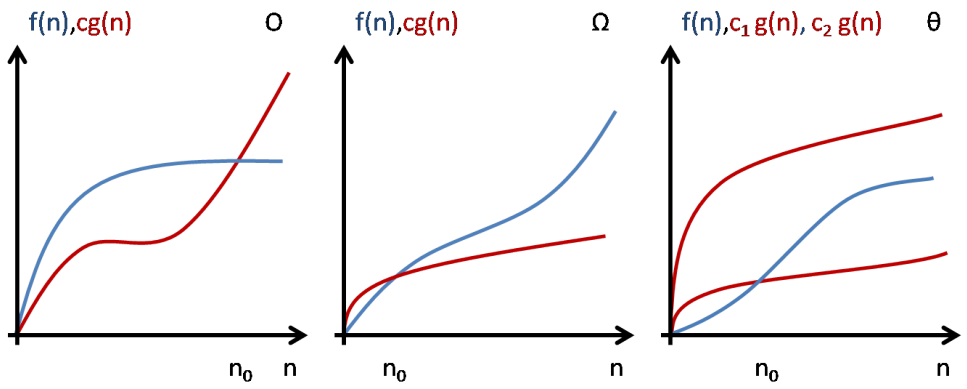

Abbildung 3.14: **Asymptotische Schranken;** Die Abbildung zeigt Beispielfunktionen der $\mathcal{O}$, Ω und Θ Notation. Die Schrankenbedingung ist dabei jeweils erfüllt, sobald die Anzahl der Inputdaten n einen konstanten Wert n_0 übersteigt. Weitere Konstanten sind c, c_1 und c_2.

Weiters sind oft mehrere Proben zur gleichen Zeit im Bereich der Aufnahme, wodurch die Auslastung des Rechners weiter steigt. Abbildung 3.13 stellt den Ablauf einer solchen Software und die zeitlichen Verhältnisse dar. Kommt es zu einer Überlastung des Rechners, kann es ohne entsprechende Maßnahmen zu einem Ausfall kommen. Dagegen kann man sich absichern, indem man in einem solchen Fall einen Teil der Daten verwirft und so die Auslastung reduziert. Dadurch kommt man jedoch in die Lage, die Sortiergenauigkeit nicht mehr erreichen zu können. Obwohl es also nur zu einer fehlerhaften Ausführung ohne Gefährdung für das System kommt, müssen dennoch harte Echtzeitanforderungen an die Software gestellt werden. Nur dadurch kann ein flüssiger Systembetrieb gewährleistet werden.
In komplexen Softwaresystemen ist es bekanntlich schwierig, konkrete Aussagen zur Laufzeit zu tätigen. In der Bewertung von Algorithmen wird dazu der Begriff der Komplexität verwendet, bei welcher asymptotische Schranken bzgl. Laufzeit und Speicherbedarf bestimmt werden. Sind solche Schranken gefunden, kann man sich darauf verlassen, dass diese auch eingehalten werden. Eine Prüfung auf die Einhaltung der Echtzeitanforderungen wird dadurch möglich. Drei Notationen sind dabei in Verwendung. Diese werden an dieser Stelle definiert und in weiterer Folge immer wieder zur Argumentation des Laufzeitverhaltens der Implementierungen herangezogen. Zu betonen bleibt noch, dass diese Schranken nur das Verhalten eines Algorithmus (genauer: Zeitkomplexität und Speicherkomplexität) beschreiben. Dabei geht das verwendete Rechnermodell nicht mitein, d.h. das Verhalten des Algorithmus kann in der Realität je nach eingesetzem Rechner variieren.
Zur Definition der Schranken werden die Funktionen $f(n)$ und $g(n)$ eingeführt, welche einer Abbildung $\mathbb{N} \to \mathbb{R}^+$ entsprechen. Ein Algorithmus weist ein Laufzeit- oder Speicherverhalten $f(n)$ entsprechend der $\mathcal{O}$ Notation auf, wenn

$$f(n) = \mathcal{O}(g(n)) \longleftrightarrow \exists c \in \mathbb{R}, \quad c > 0, n_0 \in \mathbb{N} : f(n) \leq cg(n) \quad \forall n \geq n_0 \tag{3.105}$$

gilt. Mit steigender Menge der Inputdaten bleibt die Laufzeit bzw. der Speicherbedarf also ab einem n_0 immer unter $cg(n)$, wobei n_0 und c Konstanten sind. Diese Schranke gibt mit

$cg(n)$ den Worst-Case Fall an, welcher von einem Algorithmus nicht überschritten wird. Im Vergleich dazu gibt die Ω Notation

$$f(n) = \Omega(g(n)) \longleftrightarrow \exists c \in \mathbb{R}, \quad c > 0, n_0 \in \mathbb{N} : f(n) \geq cg(n) \quad \forall n \geq n_0 \tag{3.106}$$

den Mindestbedarf an Laufzeit bzw. Speicher ab einem n_0 an Inputdaten an. Zur Angabe des mittleren Bedarfs wird die Θ Notation lt.

$$\begin{gathered} f(n) = \Theta(g(n)) \longleftrightarrow \exists c_1, c_2 \in \mathbb{R}, \\ c_1, c_2 > 0, n_0 \in \mathbb{N} : c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0 \end{gathered} \tag{3.107}$$

verwendet. Abbildung 3.14 stellt die Schranken, deren Notation auch als Landau Symbole bezeichnet wird, graphisch dar. Weist ein Algorithmus eine Ausführungszeit von $\mathcal{O}(n)$ auf, so wächst die Laufzeit linear zu einer steigenden Anzahl von Inputdaten. Ein $\mathcal{O}(1)$ Algorithmus hat dagegen eine konstante Ausführungszeit, unabhängig vom Input. Weiterführende Informationen zur Analyse und Bewertung von Algorithmen sind in [Ski08] und in [uKL07] zu finden.

3.4.2 Echtzeit Design

Um ein echtzeitfähiges Gesamtsystem zu schaffen, müssen alle relevanten Komponenten die Anforderungen erfüllen. Genauer gesagt müssen die Hardware und die Software echtzeitfähig sein. In der Auflistung des letzten Abschnittes können die Bildaufnahme, die Datenübertragung (inklusive Pufferung) und die Aktivierung der Auswerfer mit deren Verzögerungszeiten in die Kategorie der hardwareseitigen Echtzeitanforderungen gelegt werden. Die Bildaufnahme wird durch die Belichtungszeit und die Sensorauslesezeit bestimmt, diese Zeiten können mit t_s und t_r benannt werden. Die Auswerferansteuerung plus Öffnungs- und Schließverzögerung sei mit t_e gegeben. Die softwareseitigen Zeiten ergeben sich aus Berechnungszeit für die Merkmale t_f und der Klassifikationszeit t_c. Um ein korrektes Auswerfen zu ermöglichen, ist also

$$t_s + t_r + t_e + t_f + t_c = t \leq t_0 \tag{3.108}$$

einzuhalten. Die hardwarebedingten Zeiten werden durch Datenblätter gegeben[10] und sind nicht änderbar.

Man hat hier ein entsprechendes Setup zu wählen, welches möglichst geringe Verzögerungen mit sich bringt. Dies gilt vor allem für die Auswahl des Übertragungsprotokolls und die anzusteuernden Magnetventile.
Softwareseitig ist der Spielraum deutlich größer. Da bei Geschwindigkeitsaspekten von Programmen immer die gewählten Algorithmen im Vordergrund stehen und nicht die verwendete Rechnerhardware, entscheidet deren Auswahl und Implementierung über den erzielten Erfolg. Für beinahe alle Algorithmen liegen auch Bewertungen ihrer Komplexität vor. Zielführend ist die Wahl von Algorithmen linearer Komplexität, falls vorhanden. Berechnungen, die nur sporadisch auftreten, können notfalls auch mit komplexeren Algorithmen durchgeführt werden. Bei der Zusammenführung von Algorithmen zu einer Applikation sollten weiters jegliche Konstrukte vermieden werden, die zu Schwankungen in der

[10] Die Angaben sollten jedoch immer durch Messung überprüft werden, da man sich bei kritischen Anforderungen nicht rein auf die Angaben verlassen sollte.

Ausführungszeit führen können (Beispiele: dynamischer Speicher, rekursive Algorithmen mit unbekannter Inputmenge). In jedem Fall ist jedoch eine Messung der Ausführungszeit durchzuführen, um die konkreten Verhältnisse zu evaluieren. Natürlich darf auch nicht auf das Betriebssystem vergessen werden. Ideal ist hierbei der Einsatz von Echtzeitbetriebssystemen (RTOS), welche nicht nur eine Deterministik garantieren, sondern auch meist Überwachungsfunktionalität zur Verfügung stellen. Soll ein Echtzeitsystem ohne ein solches Betriebssystem erstellt werden, sind beträchtliche Zeitverzögerungen als Sicherheit einzuplanen. Weiters soll entsprechend der Prozessprioritäten verhindert werden, dass andere Prozesse, z.B. Systemprozesse die eigentliche Applikation behindern.

Die in diesem Kapitel gezeigten Grundlagen machen deutlich, dass es viele Aspekte bei der Realisierung einer Sortierung zu bedenken gibt. Die in Kapitel 4 und 5 vorgestellten Vorgehensweisen müssen nicht nur bzgl. korrekter Funktion, sondern auch bzgl. des zeitlichen Verhaltens diskutiert werden.

Kapitel 4

Aufnahme und Repräsentation von Feuerbohnen

Bevor eine Implementierung der Bildverarbeitung zur Analyse und Klassifikation von Feuerbohnen durchgeführt und evaluiert werden kann, ist die Erstellung eines optischen Systems zur Aufnahme von Bilddaten notwendig. Die dazu verwendete Vorgehensweise stellt den Inhalt dieses Kapitels dar. Zunächst werden Kriterien für ein geeignetes Aufnahmesetup erläutert. Die Wahl von optischen Sensoren, Beleuchtung und Peripherie wird hierfür argumentiert. Danach werden geeignete Repräsentationen der Bilddaten eingeführt, welche von der eigentlichen Bildverarbeitung zu verwenden sind.

4.1 Aufnahmesetup

Die Anforderungen an die Bildaufnahme wurden in Abschnitt 2.2 zusammengestellt. Entsprechend dieser Punkte und den generellen Vorgaben an die Sortierung aus Abschnitt 2.1 kann ein Setup zusammengestellt werden. Die Hauptmerkmale sind ein Durchsatz von 400 kg sowie die Realisierung einer kosteneffektiven Hardwarelösung. Auch wenn das Setup allgemein gehalten werden kann, ist sicherzustellen, dass diesen Forderungen Rechnung getragen wird.

4.1.1 Sensorwahl

In Abschnitt 3.1 wurden gängige bildgebende Sensoren aufgelistet, auch wurden verfügbare Sensorgeometrien gezeigt. Nun soll daraus ein geeigneter Sensor selektiert werden. Im Folgenden werden vorherrschende Aufnahmebedingungen beschrieben und daraus eine geeignete Technologie abgeleitet:

- **Probenfluss:** Die Objekte erreichen die Szene in zufälliger Anzahl. Das Auftreten überlappender Objekte wird zwar durch entsprechende Zuführung verhindert, können jedoch nicht ausgeschlossen werden.
- **Aufnahmeszene:** Die Szene zeichnet sich durch bewegte (frei fallende) Objekte aus. Eine kontinuierliche Aufnahme ist sicherzustellen, weiters muss man entsprechend der Fallgeschwindigkeiten kurze Belichtungszeiten berücksichtigen. Zur zielführenden

Inspektion der Objekte sind Farbbilder zu generieren. Der Hintergrund ist in einer homogenen, auf den Objekten im Normalfall nicht vorhandenen Farbe gehalten.

- **Sensoreigenschaften:** Der Sensor muss über eine hohe Lichtempfindlichkeit verfügen. Ein hoher Dynamikumfang wird nicht gefordert. Je nach verwendeter Auflösung ist für rasches Auslesen der Bilddaten Sorge zu tragen.

Auf Grund dieser konkreten Anforderungen stellt ein Zeilensensor die geeignetste Wahl dar. Mit bekannter Fallgeschwindigkeit der Objekte kann die Zeilenfrequenz f_z so gewählt werden, dass sich die Objekte in Originalgröße abbilden lassen. Durch zeilenweise Bildgewinnung ist eine kontinuierliche Aufnahme sowie ein kontinuierlicher Datentransport sichergestellt. Der Fill Factor beträgt 100%, somit wird die Sensorfläche bestmöglich ausgenützt. Zur Erhaltung der Kosteneffizienz kann ein monochromatischer Sensor mit Farbmosaik verwendet werden.

4.1.2 Optische Vorgaben

Durch die Vorgabe des Durchsatzes lässt sich errechnen, wie viele Objekte die Bildverarbeitung je Sekunde analysieren muss. Geht man dabei von 400 kg aus und berücksichtigt, dass Feuerbohnen ein Tausendkorngewicht von etwa 1200 g aufweisen, so ergeben sich sekündlich bereits 93 zu analysierdende Objekte! Eine mehrseitige Aufnahme der Szene ist in Praxis zwar notwendig, spielt jedoch für diese Arbeit keine Rolle. Jede Aufnahmeseite kann mit der selben Konfiguration ausgeführt werden, was zu äquivalenten Bilddaten bei gleicher Datenrate führt. Auch die Bildverarbeitung wird auf die Bilddaten aller Sensoren gleichermaßen angewendet. Läuft die Bildaufnahme der Sensoren auch synchron, so kann jede Aufnahmeeinheit (d.h. Sensor und Verarbeitung) als eigenständig aufgefasst werden. Es bleibt nur zu bedenken, dass bei der Verwendung mehrerer Sensoren die Gesamtdatenmenge steigt, wodurch die Anforderungen an die Rechenleistung linear wachsen. Diesem Umstand kann jedoch mit Mehrkern- bzw. Mehrprozessorlösungen auf sinnvolle Weise entgegengetreten werden.
Zur Ausstattung des Sensors mit einem Objektiv sind die Berechnungsvorgaben aus Gleichungen 3.7 bis 3.18 heranzuziehen. Zum Zwecke der Aufnahme von 3-dimensionalen Objekten im freien Fall (d.h. die Szene ist nicht flach) muss berücksichtigt werden, dass der Abstand von Objekt zu Sensor leichten Schwankungen unterworfen ist. Beim Festlegen der Schärfentiefe ist dies zu bedenken.

4.1.3 Beleuchtungskonfiguration

Bei Verwendung von Zeilensensoren ist eine Lichtquelle mit hoher Beleuchtungsstärke, bzgl. Kosteneffizienz mit hoher Lichtausbeute einzusetzen. Wird für die Zeilenfrequenz z.B. $f_z = 10\ kHz$ festgelegt, so liegt die Belichtungszeit bei 100 μs. Als geometrische Anordnung kommt das Linienlicht oder auch das Domlicht in Frage. Mit Linienlichtern lässt sich zwar genau die Aufnahmezeile ausleuchten, durch gerichtetes Licht ist bei hoher Beleuchtungsstärke jedoch mit Reflexionen zu rechnen. Eine Reduktion ist durch die Verwendung von Diffusorfolien zu erreichen, welche das Licht etwas aufstreuen. Bei gleicher Beleuchtungsstärke (in der Zeile) ist das Domlicht bei weitem teurer, auch der Einbau im Setup erweist sich (speziell bei Verwendung mehrerer Sensoren) aus Platzgründen als problematisch. Das Linienlicht ist als beste Kompromisslösung anzusehen.

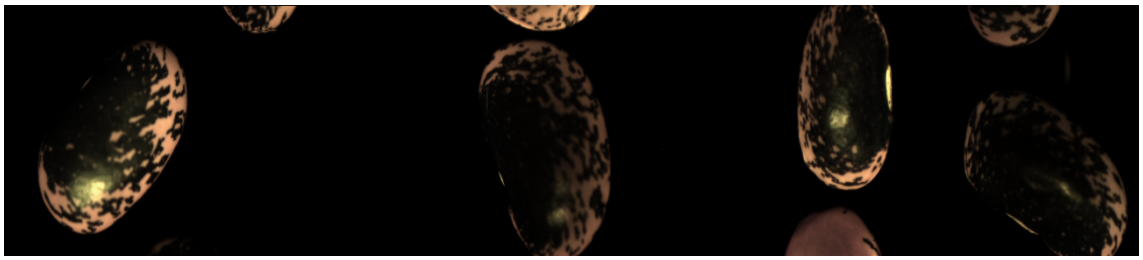

Abbildung 4.1: **Bildpuffer - Beispiel;** Das Bild wurde mit einer geometrischen Auflösung von 4096 Pixel (RGB Mosaik) aufgenommen, indem ein Puffer mit 285 Zeilen zur Verfügung gestellt wurde (Radiometrische Auflösung: 12 Bit).

4.1.4 Bilddatentransport

Bilddaten eines Zeilensensors können zeilenweise zum Rechner übertragen werden, wo sie in Bildpuffern mit wählbarer Zeilenanzahl zusammenzufassen sind. Dabei ist von sehr hohen Datenraten auszugehen. Beispiel: Ein Farbzeilensensor mit 4096 Pixel wird mit $f_z = 10\ kHz$ betrieben. Damit kommt man auf $40,96$ Megapixel je Sekunde, bei einer radiometrischen Auflösung von 10 Bit kommt es bereits zu einer Datenmenge von $409,6$ Megabyte je Sekunde. Damit fallen bereits die Protokolle USB 2.0 (bis 50 Megabyte / Sekunde), Gigabit Ethernet (bis 125 Megabyte / Sekunde) und Firewire der Spezifikation 2008 (400 Megabyte / Sekunde) aus. Daher ist eindeutig das Camera Link Protokoll (bis 680 Megabyte / Sekunde) einzusetzen. In Zukunft wird USB 3.0 ebenfalls über ausreichende Kapazitäten verfügen.

4.2 Repräsentation der Bilddaten

Am Rechner bzw. am Framegrabber werden die Bilddaten des Sensors entgegengenommen und für die Bildverarbeitung in Puffern abgelegt. Abbildung 4.1 zeigt einen Bildpuffer mit 285 Zeilen. Die abgebildeten Feuerbohnen wurden im freien Fall aufgenommen, der Puffer zeigt einen kurzen Ausschnitt des Probenflusses.

4.2.1 Bildpuffer

Je nach Aufnahmeeinstellung des Sensors bzgl. Farbraum, radiometrischer und geometrischer Auflösung und Zeilenfrequenz können Farbpuffer als Bilder $i(x, y, c)$ lt. Definition in Gleichung 3.1 aufgefasst werden. Für RGB und auch HSV Farbraum gilt wegen jeweils dreier Farbkanäle $c = 3$. Bei dieser $M \times N \times 3$ Matrix $i(x, y, 3)$ ist M nun die festgelegte Zeilenanzahl des Puffers, N die Pixelanzahl des Zeilensensors. Ein solches Bild kann auch durch drei Puffer $i(x, y)$ dargestellt werden. Wenn für jeden Puffer beide Farbräume zur Bearbeitung zur Verfüng stehen sollen, werden demnach sechs gleichartige Puffer benötigt[1]. Die Kanalbilder der RGB Repräsentation werden mit $i_r(x, y)$, $i_g(x, y)$ sowie $i_g(x, y)$ bezeichnet. Gleichermaßen sind die Kanalbilder im HSV Farbraum mit $i_h(x, y)$, $i_s(x, y)$ und

[1]Die Verwendung mehrerer Farbräume ermöglicht eine flexiblere Bildverarbeitung wegen der spezifischen Farbraumcharakteristika. Detailiertere Erläuterungen werden im Laufe von Abschnitt 5 gegeben.

$i_v(x, y)$ benannt. Die sechs Puffer werden zur besseren Identifizierung im weiteren als Kanalpuffer bezeichnet, wogegen mit Puffer die aktuelle Szene (also alle sechs Kanalpuffer gemeinsam) gemeint ist. Dies vereinfacht die Beschreibung zeitlich aufeinander folgender Puffer, z.B. Puffer i und Puffer $i + 1$. Abbildung 4.2 verdeutlicht diese Definitionen.
Ein vollständig (vom Sensor) gefüllter Puffer i wird analysiert, danach wird er von den neuen Bilddaten überschrieben. Nach der Neubefüllung steht der Puffer $i + 1$ zur Analyse bereit. Dieses Prinzip ist auch aus Abbildung 3.13 zu entnehmen.
Neben der Pufferung der originalen Bilddaten werden abhängig von der Bildverarbeitung weitere Puffer für die Speicherung von Binärmasken, Labelergebnissen usw. benötigt. Diese werden analog repräsentiert. Wo benötigt, werden diese in den Abschnitten der Bildverarbeitung eingeführt.
Bis auf die Wahl der Zeilenzahl für die Puffer sind alle festzulegenden Parameter durch die Kameraparameter (siehe letzter Absatz) vorgegeben. Harte Echtzeitbedingungen verlangen, dass alle sekündlich entstehenden Zeilen auch innerhalb einer Sekunde abzuarbeiten sind. Die Zeilenzahl (oder auch Pufferhöhe) ist auf Basis mehrerer Aspekte festzulegen:

- **Objektgröße:** Objekte sollten prinzipiell in einem Puffer Platz finden, um nicht unnötig viele Puffer zur Darstellung eines Objekts zu benötigen (der Aufand von Rechenleistung und Speicherbedarf steigt dadurch an).
- **Wartungsaufwand:** Software zur Bereitstellung und Wartung von Puffern (z.B. Löschen, Kopieren uvm.) wird ineffizienter, je häufiger diese Wartungsintervalle auftreten. Dieser Effekt stellt sich vordergründig bei sehr kleinen Pufferhöhen ein.
- **Datentransport:** Wird eine zu große Pufferhöhe eingesetzt, so leidet die kontinuierliche Abarbeitung. Da die Befüllung des Puffers i die Auswertung des Puffers $i-1$ abwarten muss, können bzgl. des Rechenaufwands Auslastungsschwankungen enstehen. Für eine ideal verteilte Auslastung wäre eine Pufferhöhe von 1 zu verwenden. Dies steht jedoch im direkten Widerspruch zu den restlichen Überlegungen.

Bei der Pufferhöhe M ist wiederum eine Kompromisslösung naheliegend. An dieser Stelle soll die Wahl von M durch

$$M = \frac{f_z}{15} \tag{4.1}$$

für eine festgelegte Zeilenfrequenz f_z vorgeschlagen werden. Dies ist jedoch nur ein Richtwert und muss für andere Anwendungen noch überprüft werden.

4.2.2 Objekte an Puffergrenzen

Da die Bildaufnahme kontinuierlich geschieht, gibt es keine überlappende Aufnahme der Szene. Wie in Abbildung 4.1 zu sehen ist, wird je nach Puffergröße immer ein kurzer zeitlicher Ausschnitt der Szene erfasst. Daher sind Objekte häufig nur teilweise auf einem einzelnen Puffer abgebildet. Da ein Puffer i durch den direkt darauf folgenden Szenenausschnitt (Puffer $i + 1$) überschrieben wird, sind auch die teilweise abgebildeten Objekte zu analysieren. Deren Bewertung ist anschließend im Folgepuffer fortzusetzten, die Speicherung von Zwischenergebnissen ist daher notwendig. Abbildung 4.3 stellt zwei hintereinander folgende Puffer $i - 1$ bzw. i dar. Daraus ist ersichtlich, dass Objekte die Puffergrenzen berühren, überschreiten und auch einschließen können. Natürlich kann ein Objekt auch die

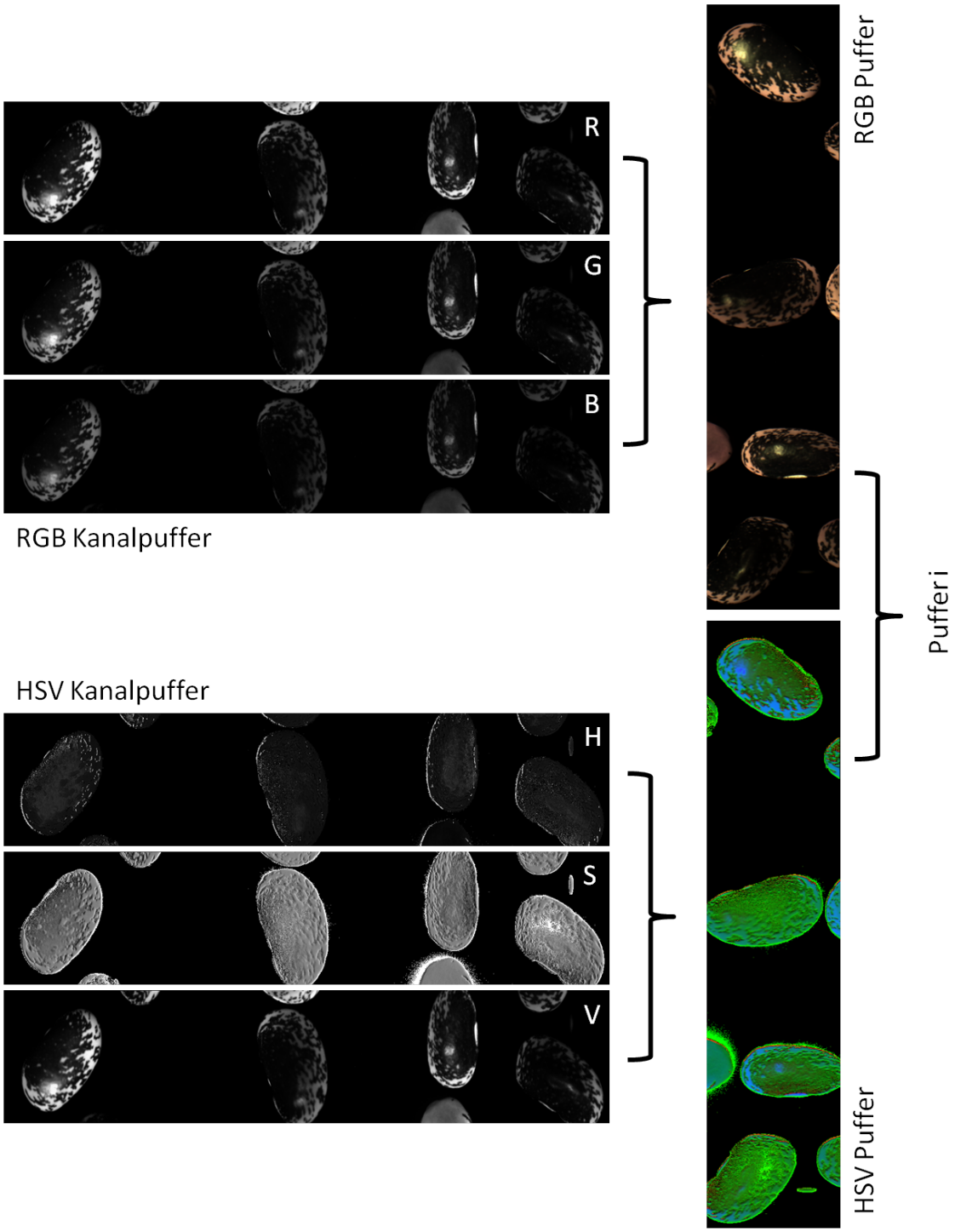

Abbildung 4.2: **Repräsentation der Szene durch Bildpuffer;** Der zu einem Zeitpunkt zur Verfüng stehende Puffer i wird durch sechs Kanalpuffer entsprechend RGB und HSV Farbraum repräsentiert.

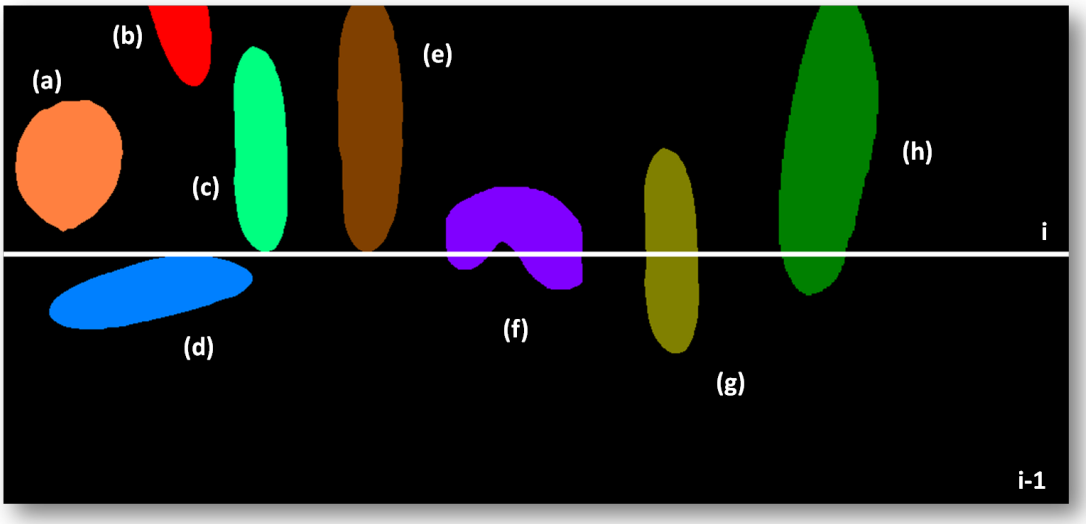

Abbildung 4.3: **Objekte an Puffergrenzen;** Die Abbildung stellt verschiedene Positionierungen von Objekten (a) bis (h) auf zwei aufeinander folgenden Puffern $i-1$ und i dar. Im Idealfall ist das gesamte Objekt auf einem Puffer abgebildet (z.B. (a)). Hier ist keine Randbehandlung notwendig. Objekte können auch die Puffergrenzen berühren wie dies für (c), (d) und (e) der Fall ist. Auch Einschlüsse (Objekt (h)) sind zu erwarten, wenn auch nur selten.

obere und untere Puffergrenze überschreiten, siehe Abbildung 4.3(h). Diese Verhältnisse sind bei der Diskussion der Bildverarbeitung speziell zu berücksichtigen.

4.2.3 Benötigte Datenstrukturen

Wie in Abschnitt 4.2 erklärt wurde, steht jeweils ein Szenenausschnitt zur Verfügung, welcher analysiert werden soll (Puffer i). Bevor der Puffer $i+1$ zu bewerten ist, wird Puffer i gelöscht. Daher muss die Information aller aktuellen Objekte in einer Datenstruktur, der Eigenschaftsmatrix $\mathbf{F_i}$, verwaltet werden.

Eigenschaftsmatrix $\mathbf{F_i}$

Die Eigenschaftsmatrix beinhaltet sämtliche Informationen der aktuellen Objekte und wird durch die Bearbeitung jedes Puffers aktualisiert. Der Index i deutet an, dass die Matrix durch die Analyse von Puffer i aktualisiert wurde. Jedes zu einem Zeitpunkt existierende (d.h. Objekt ist in Bearbeitung) Objekt belegt eine Zeile von $\mathbf{F_i}$, welche durch

$$\mathbf{F_i} = \begin{pmatrix} ID_1 & \vec{f}_1 & \mathbf{M_1} & \vec{C}_1 & \vec{s}_1 & State_1 \\ ID_2 & \vec{f}_2 & \mathbf{M_2} & \vec{C}_2 & \vec{s}_2 & State_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ ID_o & \vec{f}_o & \mathbf{M_o} & \vec{C}_o & \vec{s}_o & State_o \end{pmatrix}^T \tag{4.2}$$

gegeben ist. Diese Matrix besitzt eine statische Anzahl von o Zeilen, d.h. zu einem Zeitpunkt können o Objekte verwaltet werden[2]. Die Einträge einer Zeile von $\mathbf{F_i}$ sind:

- ID: Label des Objekts
- $\vec{f}$: Aktueller Merkmalvektor
- M: Maskenmatrix (wird im nächsten Unterabschnitt erklärt)
- $\vec{C}$: Konturkoordinaten des Objekts
- $\vec{s}$: Aktueller Flächenschwerpunkt
- *State*: Objektzustand

Jedes Objekt besitzt demnach einen Identifier, einen Merkmalvektor, eine Maskenmatrix, die Koordinaten der Kontur und des Schwerpunkts sowie einen Zustand. Da sich Objekte über mehrere Puffer erstrecken können, sind die Einträge von $\vec{f}$, $\vec{C}$ und $\vec{s}$ erst nach der vollständigen Erfassung eines Objekts gültig. Die ID ist ein Skalar zur Objektidentifizierung, $\vec{f}$ wird in Abschnitt 5.2 beschrieben. Die Konturkoordinaten eines Objekts sind lt.

$$\vec{C} = (x_1, y_1, x_2, y_2, ..., x_p, y_p)^T \tag{4.3}$$

organisiert, dessen Flächenschwerpunkt ist mit

$$\vec{s} = (x_s, y_s)^T \tag{4.4}$$

definiert. Mit dem Objektzustand *State* können Objekte gekennzeichnet werden, welche vollständig erfasst wurden und daher für die Klassifikation bereit stehen.

Maskenmatrix M

Ähnlich der objektweisen Verwaltung der Information ist auch eine objektweise Merkmalsextraktion notwendig. Durch Labeling werden Pufferpixel einem Objekt zugewiesen. Dadurch enthält der Puffer die Objektmasken aller vorhandenen Objekte. Ein solcher Objektpuffer kann prinzipiell für die Berechnungen verwendet werden, eine effizientere Methode stellt jedoch der Einsatz einer Maskenmatrix $\mathbf{M}$ dar. $\mathbf{M}$ ist lt.

$$\mathbf{M} = \begin{pmatrix} top(row) & low(row) \\ right(col) & left(col) \\ y_{colstart,1} & \sharp rowpixel_1 \\ \vdots & \vdots \\ y_{colstart,n} & \sharp rowpixel_n \\ \vdots & \vdots \end{pmatrix} \tag{4.5}$$

gegeben und ist für jedes Objekt vorhanden (siehe Gleichung 4.2). Die Maskenmatrix besitzt die Dimension $M+2\times 2$, wobei M wiederum die Pufferhöhe darstellt. In der ersten

[2]Die Zeilenzahl von $\mathbf{F_i}$ muss natürlich ausreichend groß sein, um in jeder Situation alle Objekte verwalten zu können. Eine statische Speicherung ist jedoch im Sinne einer effizienten Bildverarbeitung der dynamischen Speicherung vorzuziehen.

Zeile befindet sich der Zeilenindex von der obersten und untersten Zeile des Puffers, auf der das Objekt vorkommt, analoges wird in der zweiten Zeile für die Spalten eingetragen. Ist das Objekt über die gesamte Pufferhöhe ausgedehnt, so steht in der ersten Zeile $(1, M)$. Die weiteren Zeilen enthalten jeweils die y-Koordinate des ersten Objektpixel dieser Zeile (also jenes mit dem kleinsten x-Wert, sowie die Anzahl der in der Zeile befindlichen Objektpixel. Durch die **M** Repräsentation werden alle Objektpixel kompakt gespeichert. Die Matrix kann für alle objektbezogenen Operationen verwendet werden, bei welchen die Objektpixel zu durchlaufen sind. Somit ist nicht ein ganzer Puffer für die Objektmasken notwendig, je Objekt werden nur $M + 2 \times 2$ Speicherplätze benötigt.

4.2.4 Überlappende Objekte

Zum Abschluss dieses Kapitels soll noch kurz ein Problem angerissen werden, welches in Abbildung 5.2(d) aufgezeigt wird - Überlappung von Objekten. Generell ist ein Aufbau vorzusehen, der solche Konstellationen durch geeignete Auftrennung des Probenflusses verhindert. Dennoch kann man sich nicht zur Gänze darauf verlassen, sodass vereinzelt mit Überlappungen gerechnet werden muss. Um überlappende Objekte sinnvoll zu trennen sind diverse Methoden denkbar. Im System zur Sortierung von Feuerbohnen ergeben sich jedoch zwei entscheidende Nachteile, wenn eine solche Berechnung durchgeführt wird: Zum Einen steigt der Rechenaufwand nicht unerheblich. Es müssen nicht nur Objekte getrennt werden, sondern es muss auch entschieden werden, wann eine Überlappung vorliegt. Zum Anderen liefert selbst eine erfolgreiche Auflösung einer Überlappung keine deutliche Verbesserung der Sortierperformance, da die pneumatischen Auswerfer so nahe aneinander liegende Objekte nicht einzeln entfernen können. Es würden alle entfernt. Die einzig realistische Lösung dieses Problems liegt darin, überlappende Objekte als ein Objekt zu labeln, es aber auf Grund diverser Eigenschaften wie Farbverteilung dennoch als Bohnenart zu klassifizieren. Die Auswerfer können diese Objekte dann in einen eigenen Kanal zuführen, welcher wieder zum Start der Anlage zurückführt, um die Objekte erneut durch die Bildverarbeitung zu schleusen.

Bis zu diesem Punkt wurde festgehalten, wie eine Bildaufnahme mit Hauptaugenmerk auf die Aufnahme von Feuerbohnen funktionieren kann. Auch wurde beschrieben, wie Bilddaten transportiert, gespeichert und repräsentiert werden. Zusätzliche Datenstrukturen erlauben die Speicherung von Zwischenergebnissen, da die Pufferinformationen bei Einlesen neuer Bilddaten verloren gehen. Nun liegt eine Repräsentation zur Analyse von Feuerbohnen vor, welche das Kernthema des nächsten Kapitels darstellt.

Kapitel 5

Klassifikation von Feuerbohnen

Aus Bilddaten von Feuerbohnen in Form von Bildpuffern können die Objekte vom Hintergrund segmentiert und gelabelt werden. Für jedes einzelne Objekt wird dann der Merkmalvektor berechnet, um diesen dem Klassifikator zur Klassenzuordnung zu übergeben. Diese Schritte werden von der Bildverarbeitung durchgeführt, Abbildung 5.1 zeigt den Ablauf. Um einen Klassifikator zu trainineren, werden Bildpuffer dem Trainingsmodul zugeführt, welcher die gleichen Schritte bis hin zur Merkmalsextraktion anwendet. Beim Lernen wird jedem Merkmalvektor die Klassenzugehörigkeit zugewiesen. Als Ergebnis des Lernens steht ein Klassifikator zur Verfügung, welcher im Betrieb eingesetzt werden kann. Die einzelnen Module aus Abbildung 5.1 werden in diesem Abschnitt im Detail erläutert. Dazu werden Methoden zur Umsetzung jedes Moduls vorgeschlagen, mit Alternativen verglichen und argumentiert. Die Auswahl der Methoden erfolgt auf Grund von Funktionalität sowie Laufzeit- und Speicherkomplexität. Das Komplexitätsverhalten von Algorithmen ist ein entscheidendes Auswahlkriterium, sollen doch die Techniken nach deren Implementierung harten Echtzeitanforderungen genügen. Diese Forderungen wurden in Kaptitel 3.4 dargestellt. Eine Implementierung zur Analyse und Bewertung von vereinzelten Feuerbohnen wurde erstmals in [Fle09] vorgestellt. Diese wird verallgemeinert und auf das Vorhandensein von mehreren Objekten sowie Objekten an den Puffergrenzen erweitert. Für die Bildanalyse steht jeweils ein Puffer i zur Verfügung, die Eigenschaftsmatrix $\mathbf{F_{i-1}}$ ist zu jeder Zeit vorhanden (siehe dazu Abschnitt 4.2). Den Anfang macht die Vorverarbeitung der Bildpuffer.

5.1 Vorverarbeitung

Die Vorverarbeitung der Bildpuffer umfasst die Segmentierung der Objekte vom Hintergrund, das Entfernen von Störungen durch Filterung sowie die Unterscheidung (Trennung) der vorhandenen Objekte durch Labeling.

5.1.1 Segmentierung der Feuerbohnen

Zunächst erfolgt die Segmentierung der Objekte vom Hintergrund. Dieser entspricht einer Farbe, welche auf den natürlichen Objekten nicht vorkommt (sollte dies dennoch auftreten, wird das Objekt in jedem Fall als Fremdkörper detektiert werden). Somit ist es völlig

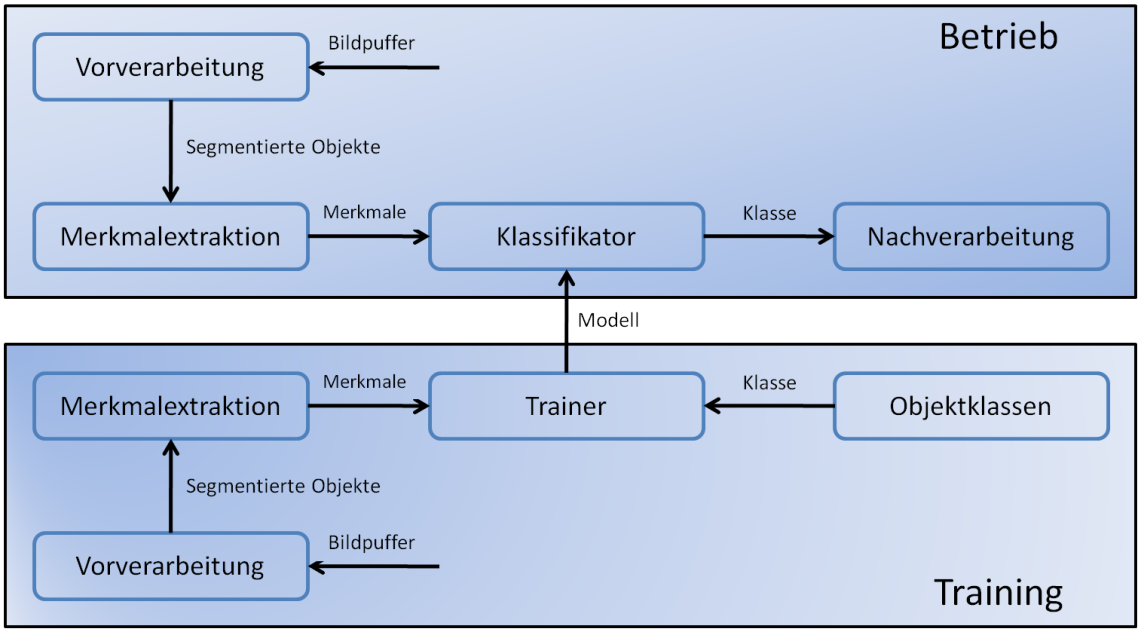

Abbildung 5.1: **Klassifikation von Feuerbohnen;** Das System zur Bildverarbeitung umfasst Vorverarbeitung, Berechnung der Merkmale und die Klassifikation. Wird der Klassifikator trainiert, wird ihm die Klassenzugehörigkeit der Objekte als zusätzlicher Input zur Verfügung gestellt.

ausreichend, eine Schwellwertsegmentierung aus Gleichung 3.53 anzuwenden. Der Hue Kanalpuffer ist für die Segmentierung am geeignetsten, da er den Farbton von der Sättigung sowie von der Beleuchtungsstärke trennt. Im Zuge dieser Segmentierung wird eine binäre Maske des Bildpuffers erstellt, welche nach Bearbeitung gelabelt wird.
Vor dem Labeling müssen drei Erscheinungen auf den Puffern korrigiert werden. Von der Schwellwertsegmentierung sind verrauschte Ergebnisse zu erwarten. Durch Lichtstreuung und Reflexionen an den Objektoberflächen können einzelne Pixel in den Huebereich des Hintergrunds fallen. Dadurch entsteht eine löchrige Maske. Die Kontur der Objekte kann noch stärker verrauscht sein, da dort die Pixelfarben in den Hintergrund übergehen. Dieser Effekt tritt vermehrt bei der Verwendung von Zeilensensoren mit Farbmosaik auf, da drei korrespondierende Pixel räumlich versetzt sind. Des weiteren ist die Kontur von Feuerbohnen oft durch kleine Unregelmäßigkeiten wie wegstehende Schalenteilchen o.Ä. gekennzeichnet. Zu guter letzt können trotz entsprechender Gegenmaßnahmen Kleinstpartikel auftreten, die sich am Bild durch verrauschte Pixel zeigen.
Zusammengefasst sollen Löcher entfernt und die Kontur geglättet und Kleinpartikel entfernt werden. Hierfür eignen sich morphologische Filter, sehr gute Ergebnisse liefert jedoch der Median Filter mit einer $[5 \times 5]$ Filtermaske.

5.1.2 Objektlabeling

Beim Labeln eines Puffers werden zusammenhängende Regionen der gleichen ID zugewiesen. Prinzipiell kann hierfür beliebig aus der Palette der entwickelten Algorithmen

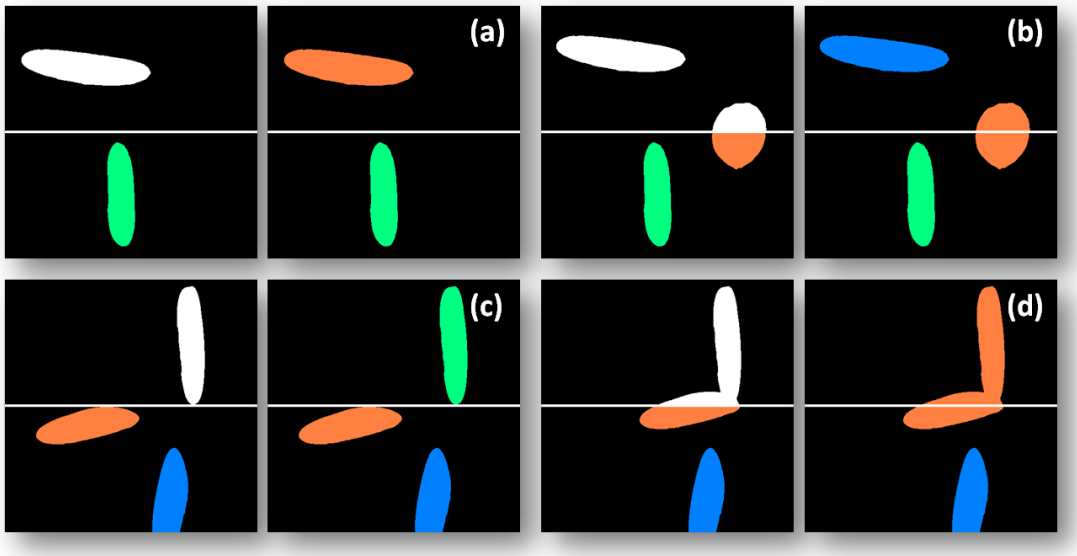

Abbildung 5.2: **Pufferlabeling - Fallunterscheidung;** Die Darstellung zeigt jeweils zwei aufeinander folgende Puffer $i-1$, i. In (a) gibt es keine Überschneidung mit den Puffergrenzen. Szene (b) zeigt die Kombination eines Labels nach Labeling des oberen Puffers. In (c) werden zwei Grenzfälle dargestellt. Sowohl für das grüne, als auch für das orange Label ist eine Überprüfung zur eventuellen Kombination notwendig, da beide Objekte Pixel an den Puffergrenzen aufweisen. Das Problem von Überlappungen wird in (d) durch ein falsches Labeling gekennzeichnet.

ausgewählt werden. Zur Effizienzsteigerung ist zu überlegen, ob während des Labelings bereits weitere Aufgaben mit bearbeitet werden können. Der Einsatz von konturbasierten Algorithmen ist zu empfehlen, da diese gleichzeitig zu den Objektlabeln die Konturkoordinaten bestimmen. Vorgeschlagen wird ein Algorithmus, welcher in [uCJL03] entwickelt wurde. Dieser kann in der Art modifiziert werden, dass er für jedes im Puffer enthaltene Objekt die Liste der Konturkoordinaten $\vec{C}$ und die jeweilige Startkoordinate (kleinster Wert der x Koordinate, dazu die passende kleinste y Koordinate) in $\vec{l}$ zurückgibt. Mit den gelabelten Objekten und den Konturinformationen stehen alle benötigten Informationen bereit, um für jedes Objekt dessen Maskenmatrix $\mathbf{M}$ zu beschreiben. Der verwendete Labelingalgorithmus besitzt ein äußerst effizientes Laufzeitverhalten und kann auch innere Konturen (Löcher) extrahieren. Ein Beispiel von zwei gelabelten Puffern ist in Abbildung 4.3 zu sehen, wobei die einzelnen ID's farblich zu unterscheiden sind.
Im Bild ist bereits zu erkennen, dass jeder Labelingalgorithmus modifiziert werden muss, um die zeitliche Abfolge der Bildpuffer worauf dieser zu berücksichtigen, wenn Objekte über mehrere Puffer ausgedehnt sind. Diverse Fallunterscheidungen sind dabei möglich. Abbildung 5.2 zeigt den Vorgang des Labeling farblich und stellt die Verhältnisse an den Puffergrenzen dar. In Abbildung 5.2(d) wird außerdem gezeigt, wie durch Überlappung von Objekten aus mehrerern Objekten fälschlicherweise eine einzelne Objektmaske entstehen kann.

5.1.3 Bewertung der Vorverarbeitung

Die Vorverarbeitung aktualisiert die Eigenschaftsmatrix durch Einträge neuer Objekte und Bestimmen einiger Informationen. Diese Aktualisierung kann durch

$$\mathbf{F_{i-1}}(1,\forall) = \begin{pmatrix} ID_n & \vec{f}_n & \mathbf{M_n} & \vec{C}_n & \vec{s}_n & State_n \end{pmatrix}^T \tag{5.1}$$

ausgedrückt werden (neue Werte in rot). Der Index $i-1$ der Eigenschaftsmatrix bleibt erhalten, bis der Puffer komplett analysiert wurde.
Die Bewertung der Komplexität gestaltet sich relativ einfach, kann aber auch nur grob angegeben werden, da keine detaillierte Implementierung angegeben wird. Hier und auch in weiterer Folge wird immer der Worst-Case betrachtet, um auf der sicheren Seite zu bleiben. Da ausschließlich statischer Speicher verwendet wird, gilt für die Speicherkomplexität $\mathcal{O}(1)$. Zur Schwellwertsegmentierung muss der Puffer einmal durchlaufen werden ($\mathcal{O}(P)$). Dabei gilt $P = M \times N$. Implementierungen für den Medianfilter gibt es sogar mit $\mathcal{O}(1)$ Verhalten ([uSP07]), andere mit $\mathcal{O}(logP)$ ([Wei]). Letzterer wird in diesem System vorgeschlagen und verwendet. Für einen Puffer und dem linearen Labelingalgorithmus gilt $\mathcal{O}(P)$. Damit erreicht die Vorverarbeitung bzgl. der Laufzeit eine Komplexität von

$$\mathcal{O}_t = \mathcal{O}(P) \tag{5.2}$$

bzw eine Speicherkomplexität von

$$\mathcal{O}_m = \mathcal{O}(1). \tag{5.3}$$

5.2 Berechnung des Merkmalvektors $\vec{f}$

Zur Klassifikation wird jedes Objekt durch einen Merkmalvektor $\vec{f}$ beschrieben. Da im Zuge der Vorverarbeitung die Objekte des Bildpuffers bereits identifiziert wurden, kann die Berechnung der Merkmale objektweise erfolgen. Von jedem Objekt liegen die zugehörigen Konturkoordinaten, sowie die Maskenmatrix $\mathbf{M}$ vor. Zuallererst wird ein geeigneter Merkmalvektor für Feuerbohnen eingeführt, danach erfolgt die Berechnung der einzelnen Komponenten von $\vec{f}$. Als zusätzliches Merkmal wird der nicht in $\vec{f}$ enthaltene Flächenschwerpunkt $\vec{s}$ berechnet.

5.2.1 Komponenten von $\vec{f}$

Die Sortierkriterien für Feuerbohnen wurden in Abschnitt 2.3 aufgelistet. An Hand dieser Kriterien ist nun zu entscheiden, welche Merkmale zur positiv/negativ Separation eingesetzt werden sollen.
Fremdkörper weisen entweder andere Konturformen auf, besitzen möglicherweise Löcher und zeigen mit hoher Wahrscheinlichkeit von Feuerbohnen abweichende Farbcharakteristika. Wenn Eigenschaften zur Repräsentation von Feuerbohnen selbst zur Verfügung stehen, dann können mit diesen auch Fremdkörper erkannt werden. Bruchstücke weisen bestimmt eine andere Kontur auf und sind eventuell in Umfang oder Flächeninhalt deutlich kleiner. Sortenfremde Bohnen sind allein über deren Farbverteilung zu unterscheiden, damit jedoch ziemlich eindeutig. Über eine Farbanalyse lassen sich im Übrigen auch frühreife Exemplare detektieren. Feuerbohnen weisen nämlich eine (je nach Typ) vom Reifestadium abhängige

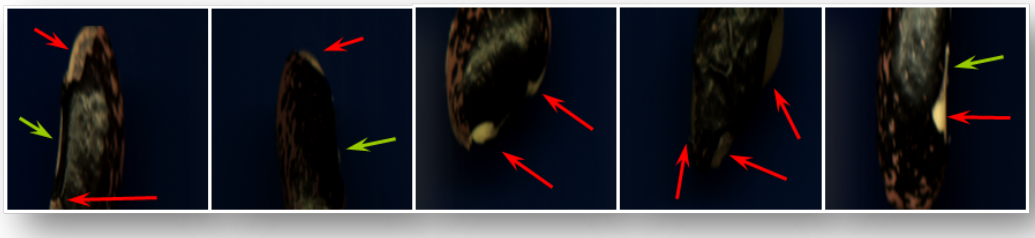

Abbildung 5.3: **Fehlstellen auf Feuerbohnen;** Risse entstehen immer in der Randregion und weiten sich nach innen aus (rot). Auch die Keimlingsansätze finden sich in der Grenzregion (grün).

Farbverteilung auf. Diese ändert sich auch während der Lagermonate weiter. Zusätzliche Merkmale sind zur Behandlung von vertrockneten Proben notwendig. Diese sind weder durch Farbverteilungen, noch durch Formabweichungen stabil zu erkennen. Hierfür kommen die in Abschnitt 3.2 eingeführten Fokusmerkmale zum tragen. Vertrocknete Proben zeichnen sich durch eine verschmierte Struktur der Farben aus, klar strukturierte Flecken sind je nach Stadium der Austrocknung nicht mehr vorhanden. Jedoch bleibt zu bedenken, dass auch einwandfreie Proben manchmal fast kein Fleckenmuster aufweisen. Visuelle Mängel dagegen sind besser zu erkennen, da das Bohneninnere eine sehr helle Gelbverteilung besitzt und sich somit stark von den Oberflächenfarben unterscheidet.
Die durch visuelle Inspektion gewonnenen Überlegungen können durch eine Kombination von aussagekräftigen Merkmalen in den Merkmalvektor geschrieben werden. Als geometrische Merkmale kommen Umfang p und Flächeninhalt A zum Einsatz. Diese können sofort aus dem Freeman Chain Code der Objektkontur gewonnen werden und stellen einfache, aber robuste Merkmale dar. Der Freeman Chain Code wird nach einigen Umrechnungen als normiertes Konturmerkmal $\vec{c}$ verwendet. Bei den Farbeigenschaften wird zuerst eine globale Größe eingeführt, da diese bereits Fremdkörper und große Fehlstellen feststellen kann. Für jedes Objekt wird dafür ein RGB Mittelwert $\vec{n}_{[3\times1]}$ bestimmt. Kleinere Fehlstellen und Risse benötigen dagegen feinere Merkmale. Die intuitive Lösung wäre das Aufsuchen von Regionen mit falschen Farben. Diese Vorgehensweise ist jedoch nicht sehr robust: Die Keimlinge weisen gleiche Farbverteilungen wie Problemstellen auf, das Finden, Zählen und flächenmäßige Erfassen der Fehlstellen ist recht aufwändig. Genauere Betrachtungen der Feuerbohnen eröffnen jedoch einen besseren Weg. Anstatt die gesamte Objektfläche auf Fehlstellen zu untersuchen, beschränkt man diese auf die Grenzregion. Risse entwickeln sich nämlich stets von der Grenzregion der flachen Proben zur Mitte hin. Abbildung 5.3 zeigt dieses Verhalten anhand einiger Beispiele. Natürlich findet sich auch der Keimlingsansatz in dieser Region, dieser wird während dem Lernvorgang miteinbezogen und in weiterer Folge vom Klassifikator akzeptiert. Die gemittelte Farbe dieser Grenzregion wird als $\vec{d}$ geschrieben. Der Vektor ist von der Dimension 4, er beinhaltet die Mittel der drei RGB Farbkanäle und das Verhältnis von Schwarz zu anderen Farben in dieser Region ($\overline{s_g}$). Dieses Farbmerkmal kann während der Berechnung von $\vec{n}_{[3\times1]}$ bestimmt werden und bedarf nur während der Vorverarbeitung einiger zusätzlicher Operationen. Für weitere Flexibilität werden zwei Farbschwellwerte r_b und y_b in den Merkmalvektor aufgenom-

men, um bei Bedarf über Distanzmaße gewisse Farbspektren zu detektieren. Ein globales Fokusmaß t, das Tenengradmaß, komplettiert den Vektor. Dessen Berechnung detektiert nebenbei diverse homogene Fremdkörper und auch halbierte Proben. Der Merkmalvektor kann abschließend zu

$$\vec{f} = \begin{pmatrix} \vec{c}^T & p & A & \vec{n}^T & \vec{d}^T & r_b & y_b & t \end{pmatrix}^T \tag{5.4}$$

geschrieben werden. Darin sind die Farbmittelvektoren entsprechend

$$\vec{n} = \begin{pmatrix} \overline{r} & \overline{g} & \overline{b} \end{pmatrix}^T \tag{5.5}$$

und

$$\vec{d} = \begin{pmatrix} \overline{r_g} & \overline{g_g} & \overline{b_g} & \overline{s_g} \end{pmatrix}^T \tag{5.6}$$

definiert. Die Definition von $\vec{c}$ findet sich in Gleichung 5.12. Es folgt nun die Berechnung der Komponenten.

5.2.2 Berechnung von $\vec{f}$

Der Merkmalvektor aus Gleichung 5.4 wird nach dem Typ der Merkmale in drei Komponenten eingeteilt, um die Beschreibung übersichtlicher zu gestalten. Die geometrischen Merkmale werden in

$$\vec{f}_A = \begin{pmatrix} \vec{c}^T & p & A \end{pmatrix}^T \tag{5.7}$$

geschrieben, der Vektor der Farbmerkmale ist mit

$$\vec{f}_B = \begin{pmatrix} \vec{n}^T & \vec{d}^T & r_b & y_b \end{pmatrix}^T \tag{5.8}$$

gegeben. Es bleibt noch das Fokusmerkmal, welches durch

$$f_C = t \tag{5.9}$$

ausgedrückt wird. Es muss an dieser Stelle noch einmal wiederholt werden, dass die Merkmale der Objekte nur komplett berechenbar sind, wenn das gesamte Objekt in einem Puffer abgebildet vorliegt. Ansonsten wird in $\vec{f}$ nur ein Zwischenergebnis geschrieben. Es wird daher von der Aktualisierung des Merkmalvektors gesprochen, nicht von dessen Berechnung.

Geometrische Merkmale - $\vec{f}_A$

Die geometrischen Merkmale sind aus Gleichung 5.7 zu entnehmen. Während A und p direkt aus dem zu bestimmenden Freeman Chain Code ableitbar sind[1], muss $\vec{c}$ durch einige Operationen auf den Freeman Chain Code gewonnen werden.
Es kann an diesem Punkt die Frage entstehen, warum für die Verwaltung der Konturinformation zwei Strukturen, nämlich $\mathbf{M}$ und $\vec{C}$, zur Anwendung kommen. $\mathbf{M}$ wird als Maske für diverse Bearbeitungsschritte benötigt, die dadurch sehr effizient durchgeführt werden können. Der Freeman Chain Code kann ebenfalls mittels $\mathbf{M}$ bestimmt werden,

[1] Da auch die Maskenmatrix $\mathbf{M}$ verfügbar ist, lässt sich A daraus trivial bestimmen.

über die Konturinformation ist er jedoch einfacher zu gewinnen. Manchmal ist eine Mehrfachkombination von Konturstücken notwendig, wenn Objektkonturen die Puffergrenzen einschließen (siehe dazu Abschnitt 4.2). In solchen Fällen erweist sich $\vec{C}$ als deutlich vorteilhafter und effizienter.
Die Aktualisierung von A erfolgt durch das Addieren der Anzahl der Objektpixel des aktuellen Puffers. Dazu werden die Einträge von $\mathbf{M}$ durchlaufen, die ja die Objektpixel je Zeile enthalten. Das Konturmerkmal $\vec{c}$ und der Objektumfang p werden nach Erhalt des kompletten Freeman Codes berechnet. Die Berechnungsvorschrift für den Freeman Chain Code findet sich in Abschnitt 3.2. In der Eigenschaftsmatrix wird durch *State* signalisiert, dass das Objektende erreicht wurde (die Abbildung des Objekts endet im aktuellen Puffer). Der Freeman Chain Code (in 8-Nachbarschaft) kann direkt aus der Objektkontur $\vec{C}$ (Gleichung 4.3) ausgelesen werden. Aus der Länge des Codes l kann p gefunden werden. Dabei wird p um 1 erhöht, wenn der nächste Pixel horizontal oder vertikal angrenzt, im anderen Fall um $\sqrt{2}$.
Nun muss noch das Konturmerkmal $\vec{c}$ aus dem Freeman Chain Code gewonnen werden. Dazu wird der Code zunächst gemäß den Angaben in Abschnitt 3.2 überarbeitet, um Varianzen bezüglich gewähltem Startpunkt und Objektrotation zu entfernen. Durch die Normalisierung der Codes können Konturen unterschiedlicher Längen l verglichen werden. Eine Repräsentation als stückweise glatte Funktion (vergleichbar mit einer Polynomapproximation) eignet sich für Vergleichszwecke sehr gut. Eine einfache Möglichkeit zur Normalisierung ist, den Freeman Chain Code in k Blöcke zu unterteilen. Ein Block besitzt dann

$$b = round\left(\frac{l}{k}\right) \tag{5.10}$$

Codewerte $X = \{x_1, x_2, ..., x_b\}$. Die Codewerte werden mit

$$c_i = \frac{1}{b}\sum_{j=1}^{b} X_j \tag{5.11}$$

aufsummiert und gemittelt. Führt man dies für alle b Blöcke aus, erhält man k Mittelwerte. Diese stellen die Komponenten des gesuchten Konturmerkmals $\vec{c}$ dar, welches nun mit

$$\vec{c} = \begin{pmatrix} c_1 & c_2 & \cdots & c_k \end{pmatrix}^T \tag{5.12}$$

dargestellt und in $\vec{f}$ eingetragen werden kann. Die Dimension von $\vec{f}$ wird also durch die Dimension von $\vec{c}$ mitbestimmt, für welche $dim(\vec{c}) = k$ gilt. Abhängig von der Anzahl der gewählten Stützpunkte k kann die Auflösung der Kontur verändert werden. Die Wahl von k ist stark von der durchschnittlichen Codelänge abhängig. In dieser Arbeit wird die Verwendung

$$k = 20 \tag{5.13}$$

vorgeschlagen. Damit wird die Kontur ausreichend genau reprästentiert, kleine Krümmungen und Unregelmäßigkeiten werden ignoriert.

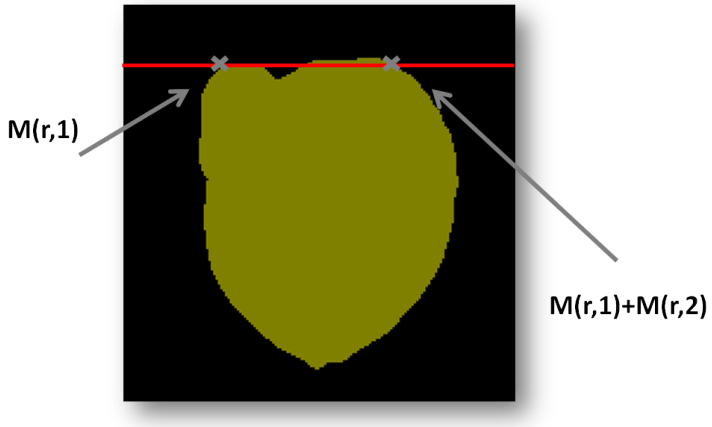

Abbildung 5.4: **Fehlinformation der M-Matrix;** Die Darstellung markiert die beiden Einträge von **M** einer gewissen Zeile. Da nur Start- bzw. Endpixel gespeichert werden, werden fälschlicherweise Hintergrundpixel miteinbezogen. Diese müssen daher in den Algorithmen noch separiert werden.

Farbmerkmale - $\vec{f}_B$

In diesem Abschnitt wird der globale Farbmittelwert $\vec{n}$, das Farbmittel der Objektgrenzregion $\vec{d}$, sowie die beiden Farbgrenzwerte r_b und y_b berechnet, um fehlerhafte Farbspektren auszugrenzen. Die Maskenmatrix **M** wird als Objektmaske verwendet, da sie Information über alle Objektpixel des Puffers bereitstellt. Über die Einträge von **M** werden Start- und Endzeile sowie Start- und Endspalte verifiziert. Damit können alle Farbwerte für $\vec{n}$ aufsummiert werden. Falls in einer dieser Zeilen Hintergrundpixel zwischen Objekten auftreten, was durchaus vorkommen kann (siehe Abbildung 5.4), dürfen diese Pixel nicht in die Berechnung miteinfließen; Zur Aktualisierung von $\vec{d}$ ist jeweils zu prüfen, ob sich ein Pixel in der Grenzregion des Objekts befindet. Dazu muss dessen Breite g festgelegt werden. Zunächst wird festgestellt, ob die Anzahl der Objektpixel nicht ohnehin geringer ist als die Grenzregion. In diesem Fall dürfen nämlich alle Pixel für die Grenzregion aufsummiert werden. Andernfalls werden nur die Farbwerte der Grenzregionspixel zu $\vec{d}$ hinzugezählt. Für die frei wählbaren Farbschwellwerte r_b und y_b wird für jeden Pixel der euklidische Abstand zu zwei Farbzentren berechnet. Liegen diese Werte innerhalb der verwendeten Schwellwerte, werden die entsprechenden Farbgrenzwerte inkrementiert. Sind Objekte auf Grund des Erreichens des Objektendes zu finalisieren, werden die Farbwerte durch A gemittelt. Darin ist zu diesem Zeitpunkt bereits der fertig berechnete Flächeninhalt enthalten. In $\vec{d}$ wird noch das Verhältnis von Farb- zu Schwarzpixel eingetragen.

Fokusmerkmal - f_C

Zur Komplettierung von $\vec{f}$ fehlt nur noch das Fokusmaß t. Zur Berechnung können die Gleichungen 3.59 bwz. 3.60 herangezogen werden. Um t zu finden, wird der Rotkanalpuffer mit einer separierten Sobelmaske gefiltert. Dazu wird die Maske mittels den Pixelinformationen aus **M** über das Objekt geschoben, wobei die äußersten Randpixel (gegeben durch

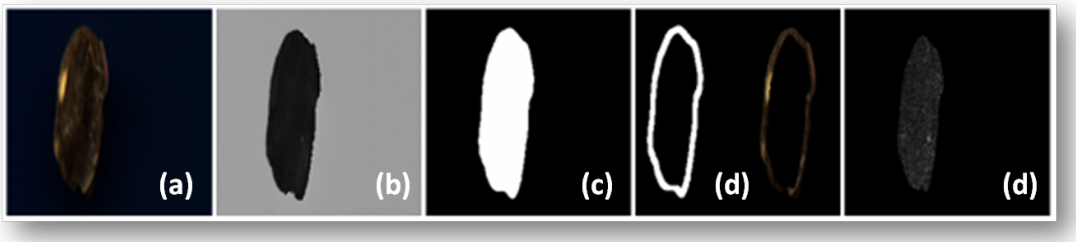

Abbildung 5.5: **Aktualisierung des Merkmalvektors;** Die Bildreihe stellt einige Zwischenschritte bei der (Neu)berechnung von $\vec{f}$ dar. Vom Originalbild (a) wird der Huekanal (b) berechnet, daraus eine gefilterte Maske (c), aus welcher **M** gefunden wird. Farb-eigenschaften der Grenzregion werden den Grenzregionspixel (d) entnommen, zur Fokusmaßberechnung wird ein sobel-gefiltertes Objektbild eingesetzt.

die Größe der Maske) nicht in die Berechnung mit eingehen, um Falschwerte im Randbereich zu vermeiden. Für jede Zeile bzw. jede Spalte wird geprüft, ob die Pixelkoordinate auch im gültigen Bereich liegt. Nun werden die Beträge des gefilterten Bildes in t aufsummiert, falls sie über einem fest eingestellten Schwellwert T_m liegen. Kleine Beträge treten nämlich auf allen Objekten auf, deren Berücksichtigung schwächt also die Performance unnötig ab. Am Ende folgt noch eine Mittelung über A, falls das Objektende erreicht ist. Zwischenschritte der Berechnungen sind unter anderen in Abbildung 5.5 dargestellt.

5.2.3 Die Rolle des Flächenschwerpunkts

Der in der Eigenschaftsmatrix $\mathbf{F_{i-1}}$ einzutragende Flächenschwerpunkt eines Objekts $\vec{s}$ wurde in Gleichung 4.4 definiert. Der Schwerpunkt wird zwar nicht zur Klassifikation der Feuerbohnen benötigt, jedoch wird er von der Steuerung der Sortiermaschine zum Aktivieren der entsprechenden Auswerfer eingesetzt. Um diese korrekt ansteuern zu können, muss die Position des Schwerpunkts eines Objekts über die Puffer mitverfolgt werden. Bei der Bearbeitung des letzten Puffers, auf dem das Objekt enthalten ist, wird der Objektschwerpunkt, referenziert zum ersten Pixel der Aufnahmezeile gespeichert (Abbildung 5.6 zeigt das Prinzip).
Für jeden Puffer i wird zunächst der Schwerpunkt des Objektabschnitts im Puffer (x_s, y_s) berechnet. Der Zeilenschwerpunkt x_s wird vorerst nur durch die Anzahl der Objektzeilen festgehalten, der Spaltenschwerpunkt y_s entspricht der Mittelung von maximaler und minimaler Objektspalte. Dann wird mittles *State* geprüft, ob das Objekt bereits im Puffer $i-1$ enthalten war. In diesem Fall liegen bereits Schwerpunktskoordinaten vor und eine Kombination der Werte wird notwendig. Für x_s wird die Gesamtanzahl der Objektzeilen aktualisiert. Im Falle von y_s wird ein Offset von altem zu neuem Schwerpunktswert errechnet und addiert. Handelt es sich um ein neues Objekt, so werden die zuvor errechneten Koordinaten gespeichert. Ist das jeweilige Objekt im aktuellen Puffer abzuschließen, so ist die Zeilenkoordinate des Objektschwerpunktes fertig zu berechnen (Spaltenkoordinate wird mit jedem Puffer richtig aktualisiert). Durch Halbieren der Anzahl der Objektzeilen erhält man den Zeilenschwerpunkt. Da der Schwerpunkt auf die Aufnahmezeile zu referenzieren ist, ist noch der Abstand des Objekts zum oberen Pufferrand o zu addieren.

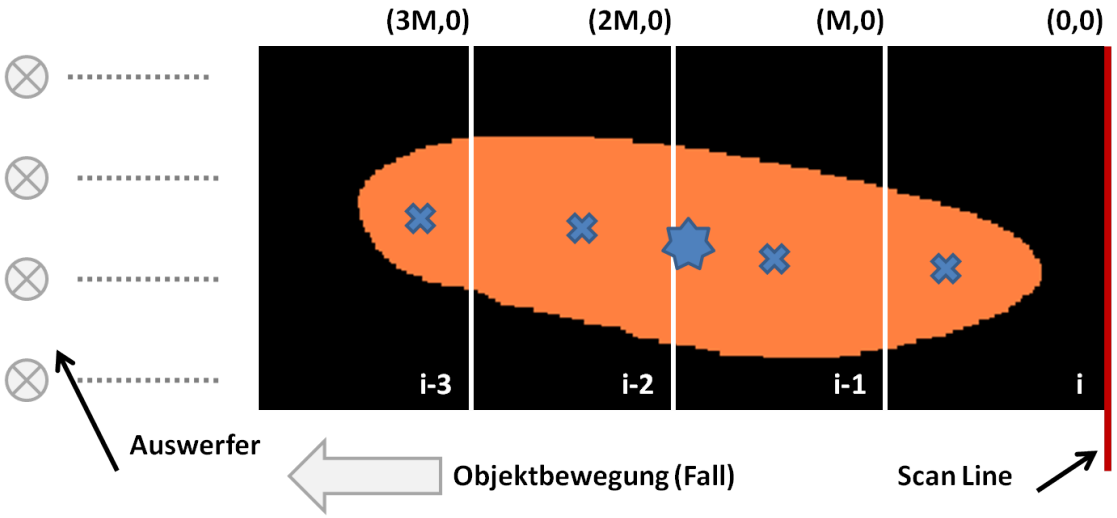

Abbildung 5.6: **Referenzierung des Flächenschwerpunkts;** Vier Puffer (diesmal horizontal dargestellt) beinhalten ein und dasselbe Objekt. Die Aufnahmezeile (Scan Line) hat den aktuellen Puffer i gefüllt, in dem das Objektende enthalten ist. Der Objektschwerpunkt (Stern) wird auf den Ursprung bei $(0, 0)$ referenziert. Der Zeilenabstand von Schwerpunkt zu Scan Line entspricht der halben Anzahl der Objektzeilen plus Offset o. Nachdem mittels Puffer i die letzte bekannte Position dieses Objekts bestimmt werden kann, muss diese Information bei Bedarf den Auswerfern bereitgestellt werden. Die Kreuze markieren die Objektschwerpunkte je Puffer.

5.2.4 Bewertung der Merkmalsextraktion

Durch das Vorgehen entsprechend der letzten Abschnitte wird die Eigenschaftsmatrix lt.

$$\mathbf{F_{i-1}}(1, \forall) = \left(ID_n \quad \vec{f}_n \quad \mathbf{M_n} \quad \vec{C}_n \quad \vec{s}_n \quad State_n \right)^T \tag{5.14}$$

aktualisiert. Einige Zwischenschritte werden exemplarisch in Abbildung 5.5 gezeigt. Darin finden sich der Huekanal, eine binäre Maske, die Randbereichsmaske zur Berechnung von $\vec{d}$ sowie ein sobel-gefiltertes Bild im Zuge der Bestimmung von t wieder.
Zuletzt muss wiederum eine Bewertung der Komplexität erfolgen.

- $\vec{f}_A$**:** Die zeitliche Analyse beginnt mit der Berechnung von A, welche nie länger als $\mathcal{O}(M)$ dauert. Zur Extraktion des Freeman Chain Codes werden die Koordinaten einmal durchlaufen, dies dauert $\mathcal{O}(L)$, wenn L die Anzahl der Konturkoordinaten darstellt. Zum Bereitstellen der Invarianz bzgl. des Startpunktes wird im schlimmsten Fall (welcher nur selten eintritt) $\mathcal{O}(L^2)$ benötigt. Nach $\mathcal{O}(b)$, Die Berechnung wird immer schneller als $\mathcal{O}(L^2)$ abgearbeitet.

- $\vec{f}_B$**:** Die vorgeschlagene Berechnung der Farbmerkmale ist ein gutes Beispiel einer zeilenbasierten und kontinuierlichen Verarbeitung. Der Rechenaufwand ist zur Anzahl der am Puffer auftretenden Objektpixel linear proportional, dies entspricht $\mathcal{O}(P)$.

- f_C**:** Das zeitliche Verhalten dieses Algorithmus wird durch die Filterimplementierung geprägt. Ein separiertes Filter bringt hier entscheidende Vorteile, weiters zeigt sich

aus Experimenten, dass das Filtern in eine Richtung (z.B. nach x) ausreichend ist. Filtern und die Mittelwertberechnung benötigen jeweils $\mathcal{O}(P)$.

- $\vec{s}$: Alle Operationen erfolgen in konstanter Zeit, da auch immer ein konstanter Input erfolgt. Daher ist diese Berechnung mit $\mathcal{O}(1)$ zu bewerten.

Die Gesamtkomplexität der Laufzeit ergibt sich unter der realistischen Annahme $P \sim L^2$ zu

$$\mathcal{O}_t = \sharp F \mathcal{O}(P), \tag{5.15}$$

die Speicherkomplexität ist wiederum mit

$$\mathcal{O}_m = \mathcal{O}(1) \tag{5.16}$$

zu bewerten. In Gleichung 5.15 gibt $\sharp F$ die Anzahl der Objekte des Puffers an. Da die Merkmalberechnungen für jedes Objekt erfolgen, ist die Komplexität natürlich von dieser Zahl abhängig.

5.3 Erkennen positiver Proben

Mit der vollständigen Berechnung des Merkmalvektors steht der Input für einen Klassifikator bereit. Bei der Sortierung von Feuerbohnen wird nur zwischen Verkaufs- und Ausschussware unterschieden, d.h. es gibt zwei Klassen. Ein Klassifikator der Struktur

$$\Theta(\vec{f}) \rightarrow \{-1, +1\} \tag{5.17}$$

ist zu wählen. Nach der Entscheidung, welches Lernverfahren zum Einsatz kommt, wird ein Klassifikatortyp festgelegt. Danach wird der Ablauf der Klassifikation im vorliegenden System erörtert.

5.3.1 Lernen des Klassifikators

In Abschnitt 3.3 wurde das überwachte dem unüberwachten Lernen gegenübergestellt. Der Hauptunterschied liegt darin, dass für das überwachte Lernen die Trainingsdaten mit der Klassenzugehörigkeit zu versehen sind. Dies bringt einen nicht unerheblichen Aufwand mit sich, da auf Grund der großen Variation von negativen und positiven Proben eine ausreichende Anzahl an Trainingsdaten bereitzustellen ist. Vorteil dieses Ansatzes ist jedoch eine weitaus genauere Definitionsmöglichkeit der Klassen. Zwar kann man mittels unüberwachten Lernens ebenso binäre Klassifikatoren erstellen, die Klassenzuteilung erfolgt dann jedoch automatisch. Es muss dann anhand von Evaluierungen überprüft werden, ob die gewollte Klassenzuordnung auch erreicht wurde. Zum Einhalten einer geforderten Sortiergenauigkeit sind unter Umständen sehr viele Iterationsschritte notwendig. Dabei ist nach wie vor nicht garantiert, dass die gewünschten Ergebnisse auch erzielt werden.
Dies ist natürlich auch bei überwachtem Lernen nicht garantiert. Jedoch zeigt die Evaluierung solcher Klassifikatoren die Separationsmöglichkeiten der Klassifikatorfunktionen, die Klassenzugehörigkeit ist zu jeder Zeit den Vorgaben entsprechend. Aus diesem Grund soll überwachtes Lernen zur Bestimmung eines Klassifikators zum Einsatz kommen.

Ablauf des Lernens

Zur einfachen Bereitstellung von Trainingsdaten kann man wie folgt vorgehen: Man teilt eine Menge von unsortierten Proben manuell (d.h. durch visuelle Inspektion) in die beiden Klassen ein. Die beiden Mengen sollen eine ausgewogene Anzahl an Beispielobjekten enthalten. Man kann nun die Mengen hintereinander durch das Aufnahmesystem schicken, um Bilddaten zu erstellen und die Merkmalvektoren der Objekte zu berechnen. Da in jeder Menge nur Objekte einer Klasse vorkommen, können alle Merkmalvektoren mit dieser Klasse assoziiert werden. Danach kann man die Objekte zu einer Trainingsmenge $T = \{\{\vec{f}_1, \omega_1\}, ..., \{\vec{f}_t, \omega_t\}\}$ zusammenfassen und zum Erlernen des Klassifikators verwenden. Zur Erstellung von Testdaten $V = \{\vec{f}_1, ..., \vec{f}_v\}$ werden einfach die Merkmalvektoren einer unsortierten Probemenge berechnet. Auch dies kann mit dem vorliegenden System vollautomatisch erfolgen.
Mit T und V stehen alle benötigten Inputdaten zum Lernen bereit. Nach Wahl des Klassifikators kann dieser gelernt und evaluiert werden. Als Lernkriterium ist z.B. der empirische Fehler aus Gleichung 3.64 geeignet.

5.3.2 Auswahl des Klassifikators

Dass ein sehr breites Spektrum an Klassifikatoren bekannt ist, wurde in Abschnitt 3.3 durch die Auflistung diverser Beispiele verdeutlicht. Für jede Aufgabenstellung sind gewisse Klassifikatoren geeigneter als andere. Natürlich ist es möglich, eine großere Anzahl mit den Trainings- und Testdaten zu evaluieren und so eine Entscheidung zu finden. Jedoch ist der Aufwand dieser Vorgehensweise beträchtlich. Man kann auch vorab einen Klassifikator auf Basis notwendiger Eigenschaften des Systems heranziehen und damit überprüfen, ob Genauigkeiten entsprechend der Vorgaben zu erreichen sind. Wichtige Kriterien für diese Entscheidung sind:

- **Rechenaufwand des Lernens:** Das Lernen des Klassifikators für Feuerbohnen ist nicht zeitkritisch, da es nicht in Echtzeit erfolgen muss.
- **Rechenaufwand im Betrieb:** Die Klassifikation muss im Betrieb für sehr viele Proben in Echtzeit erfolgen. Daher sollte die Klassenzuordnung möglichst effizient vonstatten gehen.
- **Speicherbedarf:** Der Speicherbedarf ist, sofern statisch, kein kritischer Faktor des Systems.
- **Interpretierbarkeit:** Im Zuge der Parametrisierung eines Klassifikators ist dessen Interpretierbarkeit eine nicht unerhebliche Hilfe. Für die Produktion von Ergebnissen ist sie jedoch nicht relevant.
- **Anpassungsfähigkeit:** Im Sinne von flexibler Verwendung eines Klassifikators ist es vorteilhaft, durch Parameterwahl im Zuge des Lernens eine gewisse Anpassung an die Aufgabe zu ermöglichen. Dadurch wird die Lösung generischer und eignet sich zur Adaption für ähnliche Aufgaben.

Legt man einen Klassifikator anhand dieser Kriterien fest, so zeigt sich (mit Abschnitt 3.3) das die Support Vektor Maschine (SVM) verwendet werden sollte. Sie erfüllt insbesondere

die Vorgaben bzgl. des Rechenaufwands, welche entscheidend für einen stabilen Sortierbetrieb sind. Ob die erreichbaren Ergebnisse den Genauigkeitsvorgaben gerecht werden, ist natürlich noch zu evaluieren.

5.3.3 Ablauf der Klassifikation

Ist ein Klassifikator gewählt und trainiert, kann der eigentliche Sortierbetrieb beginnen. Die im Abschnitt 5.2 gezeigten Techniken liefern die Merkmalvektoren als Input. Durch den *State* Eintrag der Eigenschaftsmatrix $\mathbf{F_i}$ sind jene Objekte gekennzeichnet, welche im aktuellen Puffer abgeschlossen werden. D.h. der jeweilige Merkmalvektor wurde vollständig bestimmt. Für jedes fertige Objekt wird die Klassifikation ausgeführt, das Ergebnis wird zur Ansteuerung der Auswerfer gespeichert.

5.4 Steuerung der Sortierung

Im letzten Teil dieses Kapitels soll noch die Verwendung der Klassifikationsergebnisse besprochen werden. Der Einsatz von Auswerfern zur Sortierung von Feuerbohnen ist der einzige Aspekt der Steuerungsaufgaben, auf welchen in dieser Arbeit eingegangen wird, da alle weiteren Funktionalitäten der Anlage nicht in direktem Zusammenhang mit den Ergebnissen der Bildverarbeitung stehen.
Im Falle einer negativen Klassifikationsentscheidung (-1) soll der entsprechende Auswerfer der Auswerferzeile zum richtigen Zeitpunkt aktiviert bzw. deaktiviert werden. Diese Zeiten (t_{on} und t_{off}) lassen sich mittels Kenntnis der Fallzeiten von Aufnahmezeile zu Auswerferzeile (siehe Anhang A) und dem vertikalen Abstand des Objekts zur Aufnahmezeile finden. Erstere werden mit den Schranken t_{min} und t_{max} definiert. Für einen Puffer mit M Zeilen kann mittels derselben Fallzeitanalyse für jede Zeile eine Fallzeit (Aufnahmezeile bis zu gewählter Zeile) abgeschätzt werden. Eine Lookup Tabelle $LUT_{\vec{s}}$ mit M Einträgen kann über den Zeilenschwerpunkt des Objekts indiziert werden. Dadurch wird der korrekte Modifikator der Fallzeit ausgelesen. Mit diesen Informationen wird die Ein- und Ausschaltzeit für den Auswerfer berechnet. Danach liest man die Zeilenkoordinate des Objektschwerpunkts aus und selektiert den entsprechenden Auswerfer. Es ist weiters zu überprüfen, ob sich das Objekt räumlich an der Grenze zwischen zwei Auswerfern befindet. Ist dies der Fall, sollen beide angrenzenden Auswerfer aktiviert werden. Die Auswerfer sind in einer geeigneten Datenstruktur, der Auswerfermatrix $\mathbf{E}$ zu verwalten, deren Struktur nun erläutert werden soll.
Die Auswerfermatrix besizt $\sharp E_j + 1$ Spalten sowie 7 Zeilen und weist die Form

$$\mathbf{E} = \begin{pmatrix} on_1 & on_2 & \dots & on_{\sharp E_j} & \\ t_{on,1} & t_{on,2} & \dots & t_{on,\sharp E_j} & flag_1 \\ t_{off,1} & t_{off,2} & \dots & t_{off,\sharp E_j} & \\ t_{on,1} & t_{on,2} & \dots & t_{on,\sharp E_j} & flag_2 \\ t_{off,1} & t_{off,2} & \dots & t_{off,\sharp E_j} & \\ t_{on,1} & t_{on,2} & \dots & t_{on,\sharp E_j} & flag_3 \\ t_{off,1} & t_{off,2} & \dots & t_{off,\sharp E_j} & \end{pmatrix} \tag{5.18}$$

auf. Dabei ist $\sharp E_j$ die Anzahl der eingesetzten Auswerfer in der Auswerferzeile. Jede Spalte mit Ausnahme der letzten repräsentiert einen Auswerfer. Zeile 1 gibt an, ob der Auswerfer

aktiviert ist oder nicht. Jeweils zwei weitere Zeilen enthalten geplante Ein- und Ausschaltzeiten. Verständlicherweise muss diese Matrix in fixen Zeitabschnitten aktualisiert werden. Die Zeitabschnitte der Aktualisierung muss entsprechend fein gewählt werden, um den Fall der Objekte mit der notwendigen zeitlichen Auflösung verfolgen zu können. Eine ideale Auflösung ist durch Versuche zu finden.
Die angegebene Matrix stellt also für jeden Auswerfer drei Ein- bzw. Ausschaltslots bereit. Nach Bedarf können natürlich auch zusätzliche Slots verwendet werden. Die Prüfwerte $flag_x$ werden gesetzt, falls sich in dieser Zeile gültige Zeiteinträge befinden. Ein Eintrag von -1 bedeutet, dass diese Stelle unbenützt ist.

In diesem Kapitel wurden alle notwendigen Techniken vorgestellt, um mit Hilfe von Bilddaten Feuerbohnen zu analysieren, zu klassifizieren sowie die Ergebnisse zu verwenden. Dabei zeigte sich, dass mit bekannten Methoden effiziente Implementierungen mit linearer Laufzeitkomplexität gefunden werden können. Es bleibt noch festzustellen, ob mit der vorgeschlagenen Vorgehensweise die Sortierziele zu erreichen sind. Dieser Frage wird im nächsten Kapitel an Hand einer konkreten Implementierung nachgegangen.

Kapitel 6

Interpretation der Ergebnisse

Ein Sortiersystem wie die vorliegende Anlage hat den harten Anforderungen der Praxis gerecht zu werden. Ob eine gewählte Methodik diese umzusetzen vermag, ist nur durch umfangreiche Tests, welche einen großen Anteil des Entwicklungsaufwands darstellen, verifizierbar. Dieses abschließende Kapitel muss also feststellen, ob die Sortierperformance erreicht werden kann (per Festlegung 95%). Bezüglich der Tauglichkeit als Echtzeitsystem soll auch die Rechenauslastung an Hand einer konkreten Implementierung in den Mittelpunkt gerückt werden. Dabei spielt der Klassifikationserfolg selbst keine Rolle. Die Ergebnisse beider zu prüfender Komponenten hängen stark von der Implementierung ab. Gerade die Geschwindigkeit der Abarbeitung steht in enger Relation zur effizienten Programmierung. Bei solch großen Datenmengen ist jedoch die geeignete Wahl der Hardware ebenfalls wichtig. Der einführende Abschnitt des Kapitels beschreibt das zur Evaluierung verwendete Setup und dessen Einschränkungen. Da nicht mit einem tatsächlichen Praxissetup evaluiert wird, werden am Ende des Kapitels Nachbemerkungen angestellt, welche den Übergang in die Praxis herstellen sollen.

6.1 Der Evaluierungsvorgang

Zu Beginn ist klar zu stellen, dass nur ein Langzeittest (z.B. über eine Sortiersaison) mit einem komplett aufgebauten Prototyp endgültige Aussagen zur Eignung des Systems zulässt. Ein solcher Prototyp umfasst den praxisgerechten mechanischen Aufbau samt Auswurfsystem, eine zweiseitige Bildaufnahme und umfangreiche Sensorik zur Sicherstellung des reibungslosen Betriebs. Zum Zeitpunkt des Verfassens dieser Arbeit stand ein kompletter Aufbau jedoch noch nicht zur Verfügung, da dessen Entwicklung einen großen Aufwand bedeutet und noch in Bearbeitung war. Eine Evaluierung, welche recht nah an diese Vorstellung heranreicht, ist trotzdem möglich. Das verwendete Rechnersystem, Softwarekomponenten und Angaben zur implementierten Algorithmik und deren Parameter sind in Anhang B aufgelistet. Die sich aus den Laborbedingungen ergebenden Einschränkungen sind:

- **Bildaufnahme:** Diese erfolgt nur von einer Seite. Dadurch wird der Rechenaufwand um 50% reduziert. Ein zweiter synchroner Sensor liefert die gleiche Menge an Daten, somit schränkt die Betrachtung eines Sensors die Allgemeinheit nicht ein. Allerdings steigt der Evaluierungsaufwand enorm, da Fehler an Proben einseitig auf-

treten können. Somit ist es nicht möglich, eine korrekte Klassifikation im Nachhinein zu überprüfen. Die manuelle Bereitstellung von positiven und negativen Proben ist daher notwendig.

- **Auswurfsystem:** Eine pneumatische Anlage zum Trennen der Proben ist nicht vorhanden. Aus gerade genannten Gründen würde eine solche jedoch die Tests nicht erleichtern. Da die korrekte Funktion der Auswerfer nicht im Kontext dieser Arbeit steht, kann sie hier als gegeben angenommen werden, da keine Beeinflussung der Bildverarbeitung auftritt.
- **Laborumgebung:** Die Tests werden im Labor durchgeführt und sind keinen Bedingungen ausgesetzt, welche bei Langzeitbetrieb auftreten. Gewisse Probleme der Praxis sind dadurch nicht feststellbar. Für das zu prüfende Softwaresystem sind solche Probleme zwar eher zweitrangig, dürfen aber nicht unterschätzt werden.
- **Testmenge:** Feuerbohnen zu sortieren bedeutet, dass jährlich viele Tonnen an Sortiergut durch die Anlage gefördert werden. Durch die eingeschränkten Möglichkeiten der Bildaufnahme des Testsystems muss mit einer deutlich kleineren Menge evaluiert werden (händische Vorsortierung notwendig!). Um trotzdem an realistische Zahlen heranzukommen, werden vier Chargen A, B, C und D aus verschiedenen Anbauflächen verwendet. Jede Charge entspricht einer Menge von 20 kg an Probegut, welches direkt aus dem Erntegerät stammt. Diese 80 kg enthalten immerhin ca. 80.000 Objekte.

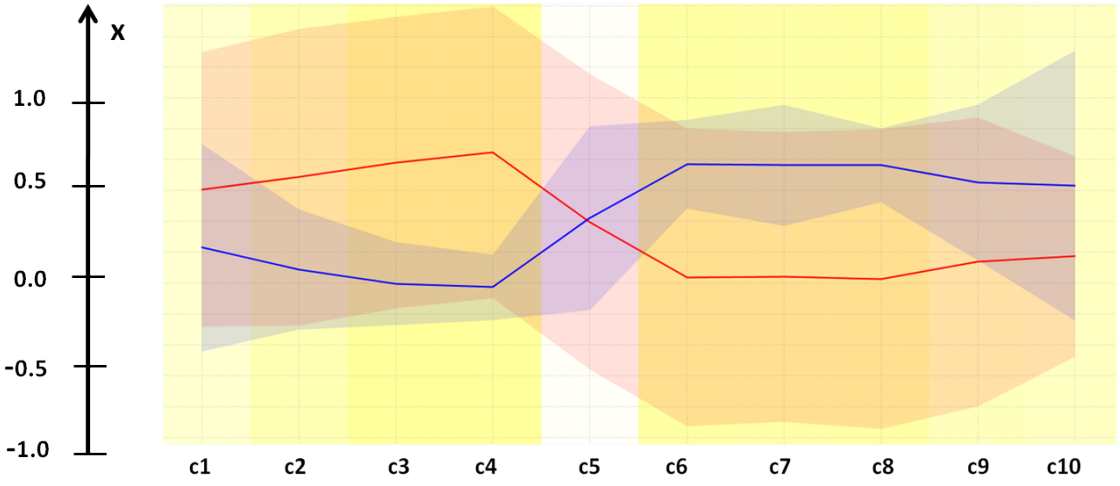

Abbildung 6.1: **Verteilung der $\vec{c}$ Vektoren;** Die Verteilung gibt die Zahlenwerte der $\vec{c}$ Vektoren an. Dabei entsprechen c_1 bis c_9 den berechneten Stützstellen der Konturannäherung durch eine stückweise glatte Funktion. Der Umfang der Objekte wird durch c_{10} repräsentiert. Die y-Achse trägt normalisierte Zahlenwerte der einzelnen Komponenten von $\vec{c}$ auf. Der blau hinterlegte Bereich zeigt den Verlaufsbereich von Konturen positiver Proben, analoges gilt für den rot hinterlegten Bereich und negative Proben. Die starken Linien zeigen für alle Komponenten von $\vec{c}$ die Mittelwerte aller Objekte.

6.2 Sortierperformance

Geht es um das Bestimmen der Sortiergenauigkeit, kann die Geschwindigkeit der Berechnungen zunächst einmal vernachlässigt werden. Viel wichtiger ist es, den Klassifikator mit einer realistischen Probemenge zu trainieren und eine alle Klassen enthaltende Testmenge zu verwenden.

6.2.1 Evaluierung mit einzelnen Merkmalen

In diesen Versuchen werden einzelne Merkmale von $\vec{f}$ evaluiert. Dadurch kann die Sinnhaftigkeit der Merkmale überprüft werden. Für die Evaluierung wird der Klassifikator (lineare SVM) selbst verwendet, d.h. es handelt sich um ein Auswahlverfahren nach dem Wrapper Ansatz (Details sind in Abschnitt 3.2 nachzulesen). Allerdings werden hier die einzelnen Merkmale nur überprüft, die Entfernung von Merkmalen ist nicht angedacht.
Der prinzipielle Ablauf dieser Versuche ist einfach (wenn auch in der Umsetzung aufwendig): Alle Proben der vier Chargen werden mit dem bestehenden, einseitigen Aufnahmesetup erfasst. Die Merkmalvektoren der Proben werden berechnet, auch die jeweilige Klassenzugehörigkeit wird hinzugefügt. Damit dies automatisch geschehen kann, werden alle Proben zunächst händisch sortiert. Um einzelne Merkmale evaluieren zu können, reicht eine händische Aussortierung von positiven und negativen Proben nicht aus. Die negativen Proben müssen weiter in Untermengen unterteilt werden. Jede dieser Untermengen enthält Proben, welche durch eine der Komponenten aus $\vec{f}$ als negativ bewertet werden sollten. Ein Beispiel: Alle Proben mit zu Feuerbohnen abweichendem Flächeninhalt (zu groß oder zu klein) kommen in die Untermenge für den Test von Merkmal A. Alle sich farblich von den Feuerbohnen stark unterscheidenden Proben kommen in die Untermenge für den Test von Farbmerkmal $\vec{n}$. Die gesamte Probenmenge D enthält 80.000 Objekte. Diese teilt sich entsprechend der Angaben lt.

$$D = D_p \cup D_n \tag{6.1}$$

in die Menge der positiven Proben D_p und die Menge der negativen Proben D_n ein. Letztere ist eine Vereinigung von Untermengen, wie durch

$$D_n = D_{n,\vec{c}} \cup D_{n,A} \cup D_{n,\vec{n}} \cup D_{n,\vec{d}} \cup D_{n,t} \tag{6.2}$$

festgehalten werden kann. Die $\vec{f}$ Komponenten p, r_b sowie y_b sind nicht in dieser Einteilung zu finden. Der Umfang p wird zusammen mit $\vec{c}$ ausgewertet und findet sich daher in $D_{n,\vec{c}}$ wieder, die beiden Farbschwellwerte dienen nur der Flexibilität und werden nicht eigenständig evaluiert.
Aus D können nun Proben für die einzelnen Versuche gezogen werden. Dafür wird ein (sehr kleiner) Teil der Daten für das Training verwendet, der Rest steht dann für die Evaluierung bereit. Als bewährtes Verfahren wird die Kreuzvalidierung angewandt (siehe z.B. [uJK82]), wobei die Datenmenge in 200 Teilmengen aufgeteilt wird. Dadurch ergeben sich bei 80.000 Proben Trainingsmengen mit ca. 400 Einträgen. Die Mengen werden zwar nach zufälligem Muster zusammengestellt, für eine Ausgewogenheit des Vorkommens der möglichen Klassen wird jedoch Sorge getragen. Der Klassifikator (lineare SVM) wird jeweils mit der Trainingsmenge gelernt, danach mit der Testmenge evaluiert.

Bevor die gesamte Datenmenge klassifiziert wird, wird der Klassifikator mit jeder Eigenschaft aus $\vec{f}$ einzeln trainiert und getestet. Dazu wird jeweils eine Datenmenge aus positiven sowie negativen Proben aus der entsprechenden Untermenge verwendet. Jede dieser Datenmengen wird mit 10.000 Objekten (davon 50% positiv) generiert. Die Ergebnisse werden mit Hilfe von Tabellen und Grafiken erläutert.

Versuch 1 - Flächeninhalt A

Evaluiert man A zur Klassifikation von Feuerbohnen im Vergleich zu Fremdkörpern bzw. Bruchstücken, welche in der Erntemenge vorkommen, so erhält man Ergebnisse, die in Tabelle 6.1 angegeben sind. Mit einer Falschklassifikation von 1066 Objekten ergibt sich eine Genauigkeit von 89.43%. Klarerweise ist die Grenze von A zwischen negativen und positiven Proben allein nicht sehr aussagekräftig. Eindeutig zu kleine bzw. zu große Objekte

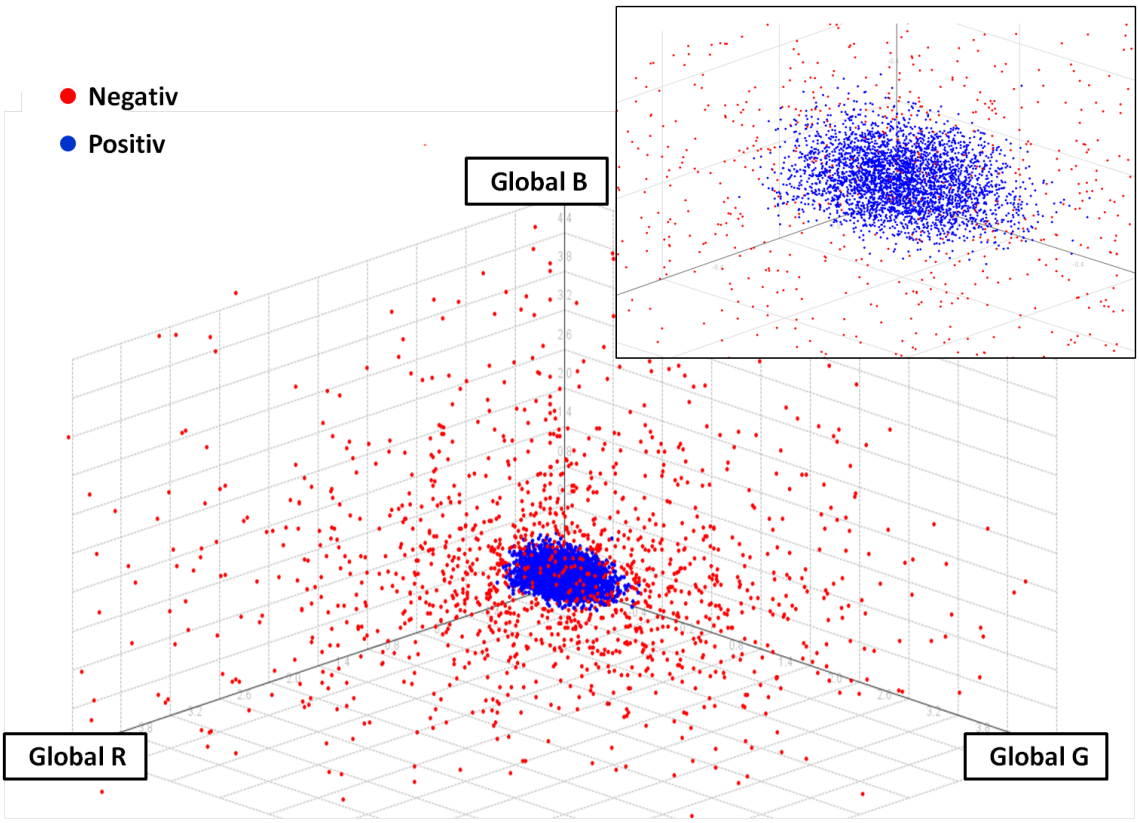

Abbildung 6.2: **Merkmale - Farbverteilung;** Geht man davon aus, dass diverse Fremdkörper die Bildanalyse durchlaufen, so ist eine breite Streuung der Farbverteilungen dieser Objekte zu erwarten. Die Markierung von Proben im RGB Farbraum bestätigt dies. Positive Proben finden sich konzentriert in einer kompakten Wolke (Werte normalisiert).

werden damit jedoch bereits zufriedenstellend bewertet. Es ist noch darauf hinzuweisen, dass die händische Einteilung in die einzelnen Probemengen visuell und daher teilweise subjektiv erfolgt. Manche Objekte befinden sich im Grenzbereich zwischen negativer und positiver Klasse. Allein dieser Umstand ergibt, dass niemals eine Sortiergenauigkeit von

100% zu erreichen ist.

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
A	89.43	+/- 1.05
Label tatsächlich →	positiv	negativ
positiv	4150	207
negativ	859	4793

Tabelle 6.1: Sortierperformance - Flächeninhalt

Versuch 2 - Konturmerkmal $\vec{c}$

Das Konturmerkmal ist für die Feststellung der von der Feuerbohne abweichenden Konturverläufe zuständig. In Abbildung 6.1 wird das Konturmerkmal $\vec{c}$ von positiven Proben (in blau) und von negativen Proben (rot) dargestellt. Die y-Achse gibt normalisierte Zahlenwerte der Komponenten von $\vec{c}$ an. Die starken Linien geben die korrespondierenden Durchschnittswerte an. So besitzt Komponente c_4 für positive Proben im Durchschnitt den Wert 0, bei negativen Proben ergibt sich 0.6. In Bereichen, in denen sich rote und blaue Verläufe nicht überdecken, ist eine eindeutige Klassenzuweisung möglich. Weist ein Objekt z.B. für die Komponente c_4 einen Wert > 0.1 bzw. < -0.1 auf, so handelt es sich um die Kontur eines negativ zu bewertenden Objekts. Klassifikationsergebnisse sind in Tabelle 6.6 angeführt. Diese belegen, dass mit dem in Abschnitt 5.2 vorgeschlagenen

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
$\vec{c}$	94.44	+/- 0.68
Label tatsächlich →	positiv	negativ
positiv	4885	441
negativ	115	4559

Tabelle 6.2: Sortierperformance - Kontur

Konturmekmal $\vec{c}$ schon 94.44% der Testdaten richtig klassifiziert werden können.

Versuch 3 - Globales Farbmittel $\vec{n}$

Proben, die eine von Feuerbohnen abweichende Farbmischung aufweisen, können durch das globale Farbmittel erkannt werden. In diese Kategorie fallen z.B. auch halbierte Feuerbohnen, welche also hüllenlos sind. Generell ist verständlich, dass die Klasse der positiven Feuerbohnen global gesehen eine sehr ähnliche Farbmischung aufweisen wird. In Abbildung 6.2 wird dieser Umstand verdeutlicht. Das globale Farbmittel $\vec{n}$ ist mit dessen drei Komponenten für die Beipielproben auf drei Achsen aufgetragen. Negative Proben (rote Punkte) weisen eine starke Streuung auf, während positive Proben (blaue Punkte) in einer Region konzentriert sind. Die Klassifikationsergebnisse sind in Tabelle 6.3 angeführt. Bereits aus Abbildung 6.2 ließ sich vermuten, dass die deutliche Unterscheidbarkeit durch dieses

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
$\vec{n}$	99.08	+/- 0.23
Label tatsächlich →	positiv	negativ
positiv	4997	89
negativ	3	4911

Tabelle 6.3: Sortierperformance - Globales Farbmittel

Merkmal eine hohe Klassifikationsgenauigkeit ergeben muss. Eine Falschklassifikation von nur 92 Objekten von 10.000 bestätigt dies.

Versuch 4 - Grenzbereichsfarbverteilung $\vec{d}$

Ein Merkmal für die Farbverteilung im Grenzbereich wurde aus der Tatsache heraus eingeführt, dass Risse auf Feuerbohnen fast von den Schmalseiten ausgehen. Beispiele dafür wurden in Abbildung 5.3 dargestellt. Da der Keimlingsansatz der Feuerbohnen ähnliches Aussehen besitzen kann wie Fehlstellen in dieser Region, ist keine sehr hohe Klassifikationsgenauigkeit zu erwarten. In vielen Fällen kann dieses Merkmal jedoch im Alleingang negative Proben detektieren. Die Ergebnisse sind in Tabelle 6.4 einzusehen. Die Überle-

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
$\vec{d}$	90.33	+/- 0.77
Label tatsächlich →	positiv	negativ
positiv	4778	745
negativ	222	4255

Tabelle 6.4: Sortierperformance - Grenzbereichsfarben

gungen bestätigen sich mit einer Sortiergenauigkeit von nur 90.33%.

Versuch 5 - Fokusmaß t

Homogene Strukturen, sowie vertrocknete Feuerbohnen sind mit dem Fokusmaß t festzustellen. Hier ist die visuelle Einteilung der Proben äußerst schwierig, da z.B. Feuerbohnen ohne Fleckenstruktur ebenfalls sehr homogene Texturen besitzen. Die Ergebnisse sind bei Verwendung als einzelnes Merkmal lt. Tabelle 6.5 gegeben. Es zeigt sich auch hier, dass eine Kombination mit anderen Merkmalen sinnvoll ist.

6.2.2 Evaluierung des gesamten Merkmalvektors $\vec{f}$

Nach Versuchen mit einzelnen Merkmalen sollen alle Merkmale zum gesamten Merkmalvektor zusammengefasst und auf die gleiche Weise evaluiert werden. Jede Komponente von $\vec{f}$ dient der Detektion verschiedener Sortierkriterien. Durch Kombination mehrerer dieser Komponenten können sich eventuell Korrelationen ergeben, gewisse Klassenmerkmale werden möglicherweise von mehreren Merkmalen erkannt. Im Gegensatz zu den bisherigen

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
t	93.41	+/- 1.06
Label tatsächlich →	positiv	negativ
positiv	4372	31
negativ	628	4969

Tabelle 6.5: Sortierperformance - Fokusmaß

Versuchen wird in diesem Fall mit der gesamten zur Verfügung stehenden Probenmenge D evaluiert. Die Ergebnisse in Tabelle 6.6 zeigen, dass bei Verwendung des gesamten Merk-

Eigenschaft(en)	**Genauigkeit** [%]	**Abweichung** [%]
$\vec{f}$	99.02	+/- 0.09
Label tatsächlich →	positiv	negativ
positiv	54938	725
negativ	62	24275

Tabelle 6.6: Sortierperformance - Gesamter Merkmalvektor

malvektors eine deutliche Verbesserung der Klassifikationsgenauigkeit erreicht wird. Diese ergibt sich zu 99.02% und erfüllt damit in jedem Fall die Forderung von 95%. Wie aus den Bemerkungen zu den bisherigen Versuchen hervorgeht, ist das Erreichen von einer Genauigkeit von 100% nur theoretisch möglich. Daher sind die erhaltenen Ergebnisse durchaus zufriedenstellend.

6.2.3 Bemerkungen zur Evaluierung

Wie eingangs bemerkt, wurden diese Ergebnisse unter Laborbedingungen erzielt. Die Probemenge wurde jedoch direkt aus verschiedenen Erntemengen gezogen, wodurch dieser Punkt realistisch untersucht wurde. Weiters ist festzuhalten, dass der verwendete Teststand zur Entfernung von kleinen Partikeln und Staub ausgestattet ist, wie dies auch am fertigen Prototyp der Fall sein wird. Auch diesem Problem wurde somit realistisch entgegnet.

6.3 Analyse der zeitlichen Deterministik

Die Auslastung des Rechners während des Betriebs der Bildverarbeitungssoftware ist neben der Performance selbst der entscheidendste Parameter des Systems, da bei Überlastung die Sortierkriterien sicher nicht mehr erfüllbar sind. Die Hardware ist für diese Versuche fix vorgegeben (Anhang B.1), die Implementierung selbst bedient sich aller Vorteile der Tools in Anhang B.2. Insbesondere die API des Framegrabbers stellt bereits das Handling von Callback-Funktionen zur Verfügung. Dabei handelt es sich um Funktionen, die auf Grund eines gewissen Ereignisses ausgelöst werden (z.B. ein Puffer wurde befüllt, eine Zeit ist abgelaufen usw.).

Hardware und Software liegen demnach vor, jedoch bleiben noch einige justierbare Parameter, welche die Bearbeitungszeit maßgeblich beeinflussen. Die folgenden Ergebnisse zeigen neben der Darstellung dieser Parameter auch die Grenzen der vorliegenden Implementierung auf. Die Realisierung der Zeitmessungen erfolgte mit den API Echtzeitfunktionen der $Sapera++$ Bibliothek.

6.3.1 Bearbeitungszeit einzelner Bildpuffer

Ein zeitliches Kriterium ist die Bearbeitungszeit von Bildpuffern. Je nach Pufferhöhe M werden sekündlich eine gewisse Anzahl von Puffern gefüllt, welche auch innerhalb einer Sekunde abzuarbeiten sind. Nach der Befüllung eines Puffers wird durch die API des Framegrabbers eine Callback-Funktion ausgeführt, welche den Puffer abarbeitet. Die Callback-Funktion zur Pufferbearbeitung wird mit $processBuffer()$ bezeichnet. In Abbildung 6.3 werden diese Callback-Funktionen analysiert. Abbildung 6.3(a) zeigt, wie lange die Analyse eines einzelnen Puffers dauert. Diese Zeit steigt natürlich mit der Pufferhöhe an. Man erkennt weiters, dass Pufferhöhen unter 200 Zeilen länger brauchen als erlaubt. Ein Puffer mit 100 Zeilen benötigt im Schnitt 17 ms zur Bearbeitung. Bei einer Zeilenfrequenz von 10 kHz werden je Sekunde 100 Puffer gefüllt. Bei dieser Verarbeitungsgeschwindigkeit werden demnach für sekündlich entstehende Daten 1700 ms benötigt. Natürlich hängt die Bearbeitungszeit der Puffer maßgeblich von der Anzahl der vorkommenden Objekte ab. Für diesen Versuch wird eine gleichmäßige Auslastung im Sinne von 4 Objekten je Puffer erzwungen. Abbildung 6.3(b) variiert die Anzahl der Objekte je Puffer bei fixer Pufferhöhe von 400 Zeilen. Es lässt sich ableiten, dass bei dieser Pufferhöhe nicht mehr als sechs Objekte auf einem Puffer analysiert werden können. Bei 25 solcher Puffer je Sekunde ergibt dies 150 Objekte als theoretische Obergrenze, wodurch je Stunde 540000 Objekte, also etwa 540 kg Probegut, analysierbar wären. Mit diesem Setup sind jedoch nur 2 bis 4 Proben je Puffer sinnvoll zu analysieren.
Nicht nur die Pufferbearbeitung, sondern auch die Aktualisierungsroutinen für die Auswerfer leisten einen erheblichen Beitrag zur Rechnerauslastung. Wie in Abschnitt 5.4 erläutert, ist die Aktualisierung der Auswerfer mit sehr hoher Frequenz vorzunehmen, z.B. mit $1ms$. In diesem Fall sind dies 1000 Ausführungen je Sekunde. Die Aktualisierungsroutine ist ebenfalls als Callback-Funktion implementiert und mit $msCallback()$ bezeichnet. Sie wird zu festen Zeitpunkten ausgeführt. Wählt man eine Taktung je ms, so ergibt sich über eine Sekunde eine Auslastung entsprechend Abbildung 6.3(c). Je ms werden also etwas 0.4 ms verbraucht, wodurch die Gesamtauslastung natürlich drastisch gesteigert wird. Überlegenswert ist hierbei, ob eine zeitliche Auflösung von 1 ms unbedingt benötigt wird, oder ob z.B. auch ein Aufruf alle 2 ms ausreichend ist. Dies kann jedoch nur mit vorhandenem Auswerfersystem festgestellt werden.

6.3.2 Bearbeitungszeiten im Sortierbetrieb

Im Sortierbetrieb laufen die Callback-Funktionen $processBuffer()$ sowie $msCallback()$ simultan. Abbildung 6.4 stellt den sich somit ergebenen Rechenaufwand dar. Zur kontinuierlichen Abarbeitung der Puffer selbst addiert sich der Aufwand der zyklischen Wartungsroutine $msCallback()$. Je nach gewählter Zykluszeit müssen demnach gewisse Mindestpufferhöhen gewählt werden, um einen stabilen Programmfluss zu erreichen. Dies sind

mindestens 600 Zeilen bei einem 1 ms Callback bzw. 300 Zeilen bei 2 ms Callback. Dies folgt aus den Diagrammen in Abbildung 6.4(a) bzw. 6.4(c), welche mit durchschnittlich vier Objekten je Puffer erstellt wurden. Fixiert man die Pufferhöhe mit 400 Zeilen, so dürfen je Puffer bis zu zwei (b) bzw. sechs (d) Objekte vorkommen. Erneut können diese Diagramme gefunden werden, wenn die Abarbeitungszeiten unter statischen Parametern über 1000 Puffer gemittelt werden. Für (a) und (c) kommen vier Objekte je Puffer zur Anwendung. Verständlicherweise findet man bei Wiederholung der Versuche immer leicht abweichende Ergebnisse, da sich der Probenfluss durch die Bildverarbeitung nicht exakt nachstellen lässt.
Das Laufzeitverhalten während des Sortierbetriebs wird in Abbildung 6.5 deutlicher erkennbar. Hierbei wird die Auslastung nicht je Puffer, sondern über die Zeit dargestellt. Die dargestellte Prozesszeit ist direkt als prozentuelle Auslastung zu verstehen, wobei 1000 ms 100 % bedeuten.

6.3.3 Sortierdurchsatz

Aus der Auswertung des Rechenaufwands lässt sich die Menge an analysierbaren Objekten für ein gewähltes Setup und eine gewisse Sortierzeit ermitteln, die folgende Tabelle fasst diese Ergebnisse zusammen. Die Daten der Tabelle zeigen bei entsprechender Konfigura-

Callback Zyklus $[ms]$	**Pufferhöhe** $[Zeilen]$	**Proben je Puffer** [1]	**Proben je Sek.** $[1s^{-1}]$	**Proben je St.** $[kgh^{-1}]$
1	400	2	50	180
	500	2	40	144
	600	2	33	118
2	400	6	150	540
	500	6	120	432
	600	7	116	417
3	400	7	175	630
	500	7	140	504
	600	8	133	478
4	400	10	250	900
	500	10	200	720
	600	10	167	601

Tabelle 6.7: Sortierdurchsatz

tion recht beträchtliche Durchsätze, diese unterliegen allerdings einigen Einschränkungen. Neben den in der Kapiteleinführung aufgelisteten Bemerkungen (z.B. einseitige Bildgewinnung, dadurch halbe Datenmenge) spielt eine Zykluszeit größer 2 in der Praxis evtl. eine untergeordnete Rolle. Weiters geht man bei der Anzahl der Proben je Puffer von separaten Objekten aus. Kommt ein Objekt auf mehreren Puffern vor, so erhöht sich diese Anzahl. Der Durchsatz wird dadurch natürlich nicht gesteigert.

6.4 Bemerkungen zur Praxistauglichkeit

Die in diesem Kapitel präsentierten Ergebnisse werden nun zu einigen Kernaussagen zusammengefasst. Die Performancetests zeigen, dass die gewählte Methodik durchaus in der Lage ist, den Anforderungen der Bildverarbeitung gerecht zu werden. Eine Genauigkeit von 99 % bei geforderten 95 % ist ein sehr gutes Ergebnis, dennoch ist damit noch keine Garantie für den erfolgreichen Praxiseinsatz gegeben. Viele Faktoren, welche in der Praxis von Bedeutung sind, können unter Laborbedingungen nicht berücksichtigt werden. Die Analyse der hier zur Verfügung stehenden Testdaten lässt jedoch aufzeigen, dass die angewandte Methodik eine gute Basis für die weitere Verwendung darstellt.
Es darf weiters nicht vergessen werden, dass jede Komponente des Sortiersystems den erreichten Wert der Sortiergenauigkeit reduziert. Wird z.B. nicht jede negative Probe von den Auswerfern korrekt erfasst, so wird die Gesamtgenauigkeit vermindert. Erst umfangreiche Tests mit einem kompletten Prototypen ermöglichen also eine Aussage zur Gesamtgenauigkeit. Die Aufgabe der Bildverarbeitung scheint durch die Implementierung zu Genüge erfüllt zu werden.
Nicht nur die korrekte Funktion sondern auch die Geschwindigkeit der Bildverarbeitung ist für die Praxis unabdingbar. Die Zeitmessungen machen eines deutlich: Mit dem gewählten Setup befindet man sich auf jeden Fall in einem Grenzbereich der praktischen Tauglichkeit. Darüber hinaus kommt bei zweiseitiger Bildgewinnung die doppelte Datenmenge zu Stande, wodurch zumindest ein zweiter, mit gleicher Ausstattung versehener Rechner benötigt wird. Bei Echtzeitsystemen sollten immer Sicherheiten bzgl. der Auslastung einbehalten werden. So sollte je nach gefordertem Durchsatz die Auslastung der bzw. des Rechners unter 70 % gehalten werden. Da die für die vorliegende Arbeit eingesetzte Hardware bereits durch effizientere Komponenten ersetzbar ist, kann höchstwahrscheinlich eine weitaus niederere Auslastung erreicht werden. Andernfalls ist die Implementierung zu optimieren.
Angemerkt sollte noch werden, dass diese Arbeit, wie in der Einleitung bereits festgehalten, nur Teile des Gesamtsystems Sortierung abdecken kann. Aus anderen Komponenten kann und wird die Notwendigkeit nach weiteren Softwarekomponenten entstehen, die hier nicht berücksichtigt wurden.

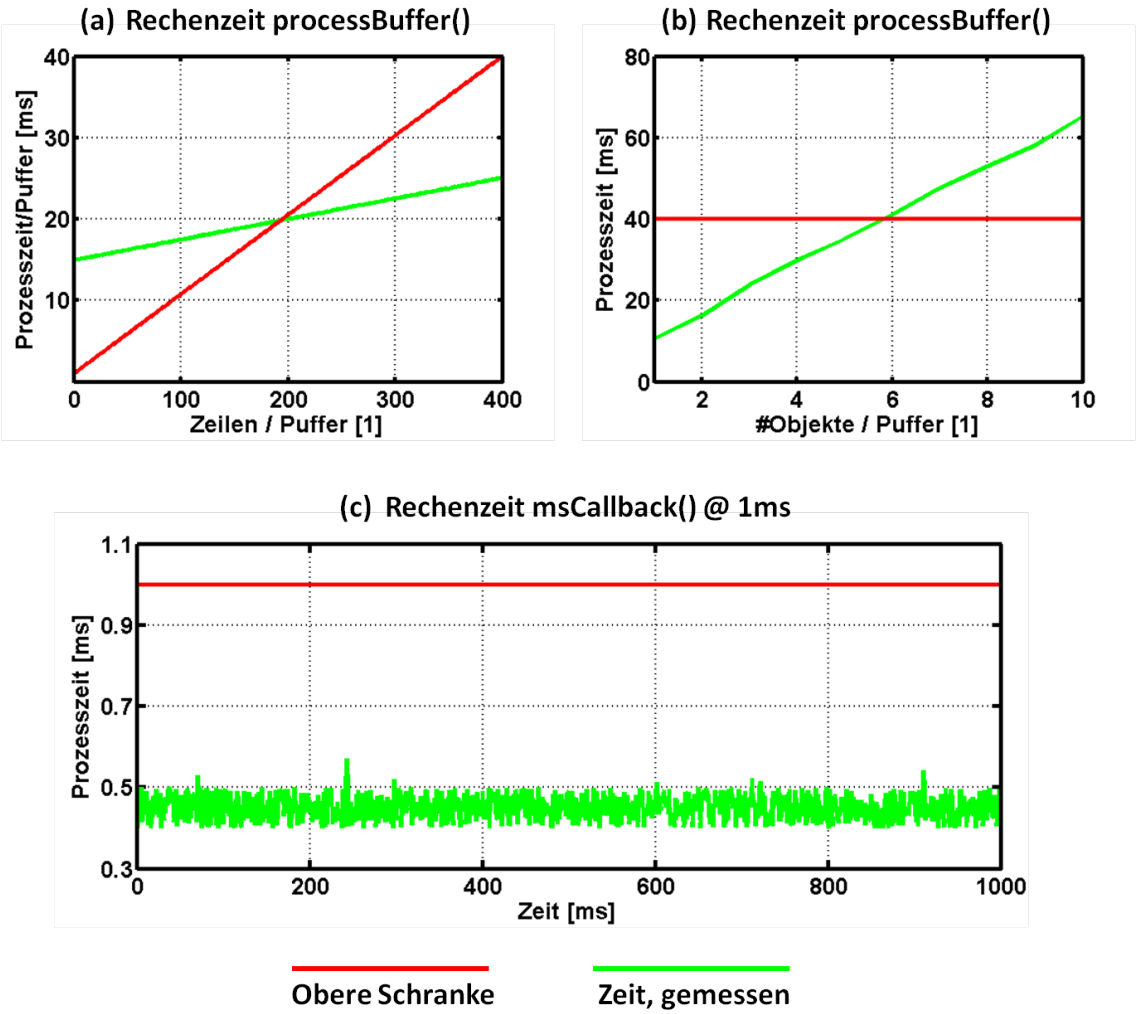

Abbildung 6.3: **Zeitverhalten der Callback-Funktionen;** Die Diagramme zeigen mittlere Bearbeitungszeiten für einzelne Bildpuffer. Die Ergebnisse werden durch Mittelung von 1000 Puffern der selben Größe erhalten. Die jeweils rote Linie markiert das Zeitlimit, welches in der jeweiligen Konfiguration nicht überschritten werden darf, sollen alle Daten abgearbeitet werden. Die grüne Linie stellt die Bearbeitungszeiten dar. Diagramm (a) zeigt Bearbeitungszeiten von $puf ferCallback()$, welche linear mit der Pufferhöhe steigen. Die Puffer dieser Messung enthalten im Durchschnitt 3 Objekte. Dies kann für diese recht kurzen Messungen (1000 Puffer entsprechen nur einigen Sekunden) durch entsprechende Zubringung gewährleistet werden. In Diagramm (b) wird die Pufferhöhe mit 400 Zeilen festgelegt und die Anzahl der Objekte verändert. Auch hier zeigt sich ein fast linearer Verlauf bei steigender Objektanzahl. Diagramm (c) verdeutlicht den Aufwand der zeitlich getriggerten Wartungsroutine $msCallback()$. Ein Aufruf dauert im Mittel 0.4 ms, d.h. über eine Sekunde werden nur von dieser Funktion bereits 400 ms verbraucht! Zur Erstellung dieses Diagramms wurden alle anderen Funktionen deaktiviert.

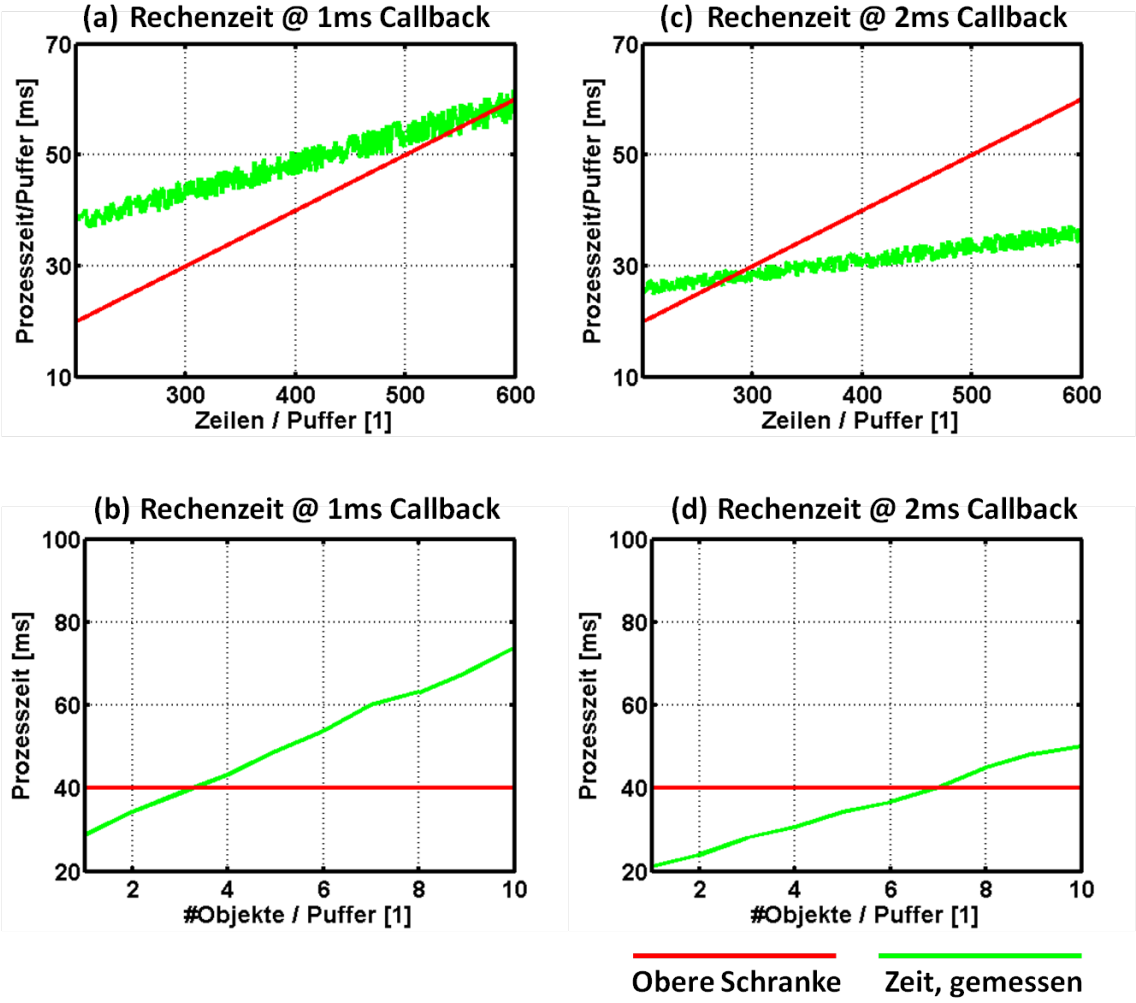

Abbildung 6.4: **Zeitparameter - Sortierbetrieb;** Im Sortierbetrieb sind $pufferCallback()$ und $msCallback()$ aktiviert. Bei Verwendung von 1 ms Callback wird erst über einer Pufferhöhe von 600 Zeilen die Berechnungszeitschranke unterschritten (a). Zu große Puffer führen jedoch wieder zu einem unkontinuierlichen Datenfluss. Wählt man ein 2 ms Callback (c), so erreicht man dies bereits bei Puffern mit 300 Zeilen. Dieses Verhalten zeigt sich analog auch bei Betrachtung der Pufferauslastung mit Objekten. Bei 1 ms Callback sind zwei Objekte je Puffer realistisch, bei 2 ms sind bereits sechs Objekte möglich. Für (b) und (d) sind wiederum 400 Zeilen Puffer in Verwendung.

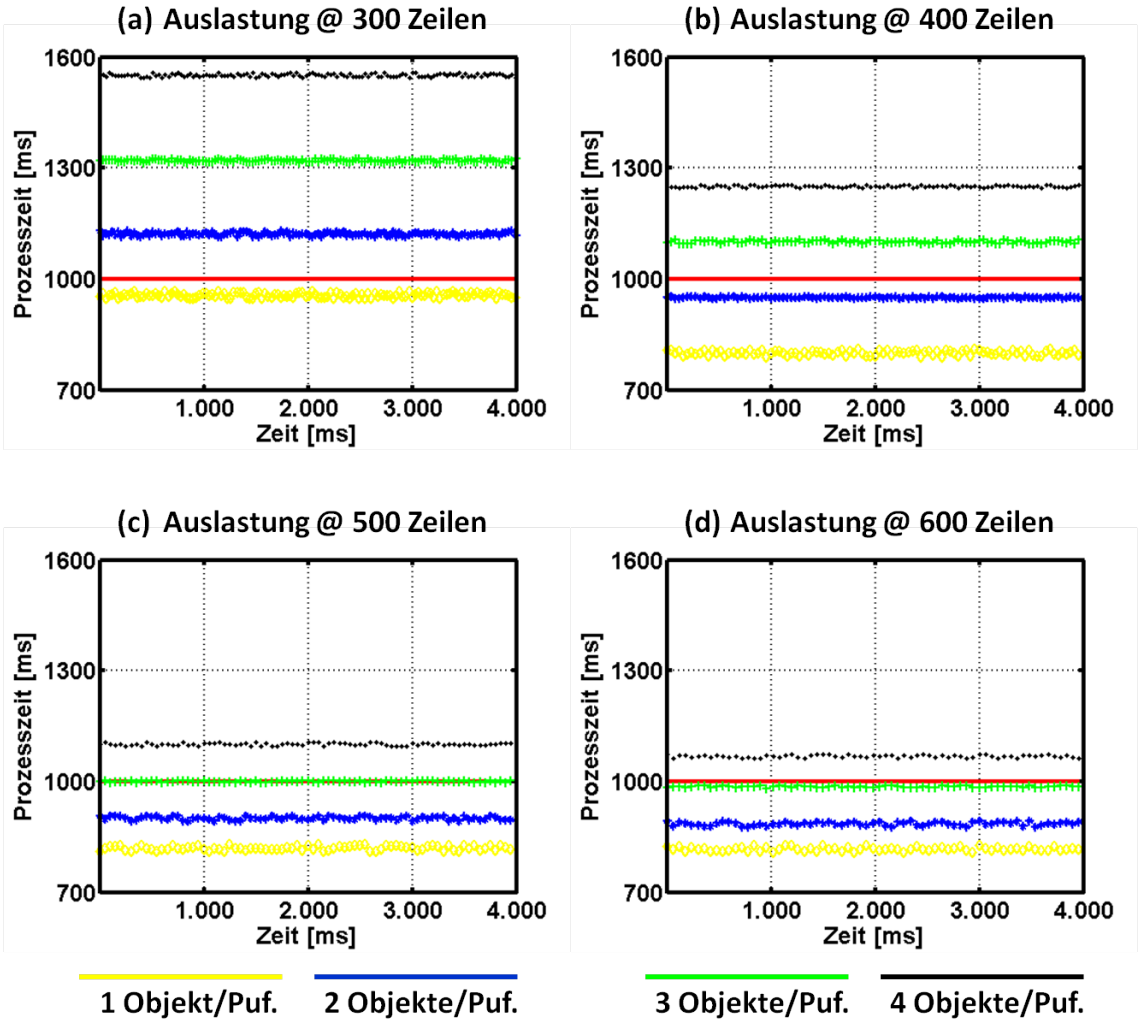

Abbildung 6.5: **Rechnerauslastung im Sortierbetrieb;** Je nach gewählter Pufferhöhe zeigen die Diagramme die Anzahl der praktisch analysierbaren Objekte. Die Auslastung liegt bei 100 %, wenn die 1000 *ms* Marke erreicht wird. Wird diese überschritten, so liegt die Abarbeitungszeit der sekündlich entstehenden Daten darüber, ein kontinuierlicher Betrieb ist nicht mehr möglich. Bei dieser Konfiguration mit einer Zykluszeit von 1 *ms* muss eine Mindestpufferhöhe von 400 Zeilen gewählt werden (b), um zumindest zwei Objekte je Puffer zu erlauben. Allerdings ist auch in diesem Fall die Auslastung bereits über 80 %, daher sind 500 (c) bzw. 600 (d) Zeilen vorteilhaft, um eventuelle Prozessschwankungen abzufangen. Die Abarbeitungszeit stellt sich sehr kontinuierlich dar, dies ist jedoch auf eine sehr genau eingestellte Zuführung der Objekte zurückzuführen. Diese schwanken in der Praxis mehr, deshalb sollte man sich so weit wie möglich unter der 1000 *ms* Marke einpendeln können.

Kapitel 7

Schlusswort

Diese Arbeit stellt ein komplexes, bildgestütztes Verfahren zur Sortierung von Feuerbohnen vor. Ein solches System war vor wenigen Jahren noch unmöglich durch Standardkomponenten aufzubauen. Man bedenke dazu z.B. die generierten Datenmengen, die aufwendige Beleuchtung (bzgl. der erforderlichen Beleuchtungsstärken) und die Verfügbarkeit von Rechenleistung am PC. Das System zur Sortierung von Feuerbohnen zeigt an einem einzelnen, sehr konkreten Beispiel, welch großes Anwendungsspektrum die Bildverarbeitung für die Industrie bereit zu stellen im Stande ist. Zum einen werden immer effizientere Algorithmen gefunden, welche aufwendige Berechnungen schon auf einem Heimrechner ermöglicht. Die Entwicklung neuer, spezieller Hardware für Bildverarbeitung und Computergrafik reizt die aktuellen technischen Möglichkeiten immer besser aus. Erst nach und nach kann die Industrie neue Lösungen erkennen und neue Verfahren erschließen.
Die vorliegende Arbeit vertraut auf rein bildgestützte Techniken. Wie bei den meisten technischen Problemstellungen der Fall, führen auch andere Wege zum Ziel. Die Untersuchung in anderen Bereichen des Spektrums bringt andere Vor- und Nachteile und mit Sicherheit einige Verbesserungen. Es war aber von Anfang an das Ziel dieses Projekts, ein System zu realisieren, welches nicht nur funktionell, sondern auch in hohem Maße kosteneffizient aufzubauen ist. Und es gibt z.Z. keine günstigeren Komponenten der Bildverarbeitung, als jene im visuellen Spektrum.
Die Sortierung von Feuerbohnen stützt sich auf ein bewährtes Prinzip und setzt auf aktuelle Komponenten. Das Potential dieser Thematik wird durch diese Arbeit jedoch noch bei weitem nicht ausgereizt. Zur konkreten Realisierung einer funktionstüchtigen Maschine sind noch viele weitere Arbeitsschritte notwendig. Diese Schritte werden z.Z. auch tatsächlich gesetzt. Durch Verwendung eines Prototyps in der Praxis lassen sich die erarbeiteten Techniken im Detail untersuchen, die Erfahrung wird dann deren Wiederverwendbarkeitswert zeigen. Viele interessante Fragen sollen so beantwortet werden: Welche anderen Analyseobjekte sind ohne Modifikation untersuchbar? Wie effizient arbeitet die Software im Vergleich zu gängigen Analyseprogrammen? Wie sieht die Skalierbarkeit für größere Anwendungen aus? Das Beschäftigungsfeld ist also groß. Dabei wurde noch nicht einmal über neue Technologien nachgedacht. So eröffnet z.B. die Multispektralanalyse wiederum neue Wege. Aber so rasch die Technologien sich auch entwickeln, werden sie durch die ebenfalls immer höher werdenden Anforderungen immer wieder auszureizen sein.

Anhang A

Fallverhalten von Feuerbohnen

Bei der Sortierung von Feuerbohnen ist die Fallzeit der Proben zwischen den Zeilensensoren und den Auswerfern ein entscheidender Faktor. Eine Analyse des Fallverhaltens dient jedoch nicht nur der Bestimmung dieser Fallzeit, sondern auch der Prüfung des Fallverhaltens selbst. Die einzelnen Proben im Probenfluss müssen vertikale Mindestabstände einhalten, weiters ist eine Untersuchung bzgl. horizontalem Versatz und Probenrotation notwendig. Diese Faktoren sind dem mechanischen Aufbau der Maschine unterworfen und für die Software eher unrelevant. Es gilt einzig ein gleichmäßiges, stabiles Fallverhalten sicherzustellen. In diesem Abschnitt wird die Bestimmung der Fallzeiten erörtert. Diese dienen der Festlegung einer Zeitschranke für die Bearbeitungszeit der Bildanalyse und legen fest, zu welchen Zeitpunkten die Auswerfer nach Bedarf zu aktivieren sind. Dazu werden die Proben mit einer Serie von Bildaufnahmen während des freien Falls verfolgt. Aus dieser Sequenz lassen sich die benötigten Informationen extrahieren. Wird diese Berechnung auf eine ausreichend große Menge von Proben angewandt, können obere und untere Fallzeitschranken der Proben gefunden werden. Alle erwarteten Probentypen sind miteinzubeziehen. Nach Auswahl eines geeigneten Sensor Setups und dessen Kalibrierung erfolgt die notwendige Auswertung der Bilddaten.

A.1 Aufnahme-Setup und Kalibrierung

Zur Datengewinnung wird der Flächensensor Firefly MV verwendet, welcher einen $1/3''$ CMOS Chip besitzt. Dieser verfügt über eine maximale Sensorauflösung von 640×480 Pixel bei 60 fps. Die Aufnahmebreite (Szene) beträgt 10 cm. Die aufzunehmende Höhe hängt vom Abstand der Sensoren zu den Auswerfern ab. Dieser wiederum muss berücksichtigen, dass die Analysesoftware für jede Probe eine gewisse Entscheidungszeit benötigt, welche hier mit $t_{SW} = 20\ ms$ festgelegt wird. Bis zum Aktivwerden der Auswerfer vergeht eine weitere Zeit t_{HW}, welche 2 ms ausmacht. Dies ergibt eine Beschleunigungszeit der Proben zwischen Sensor und Auswerfern von

$$t_b = t_{SW} + t_{HW} = 0.02 + 0.002 = 22ms. \tag{A.1}$$

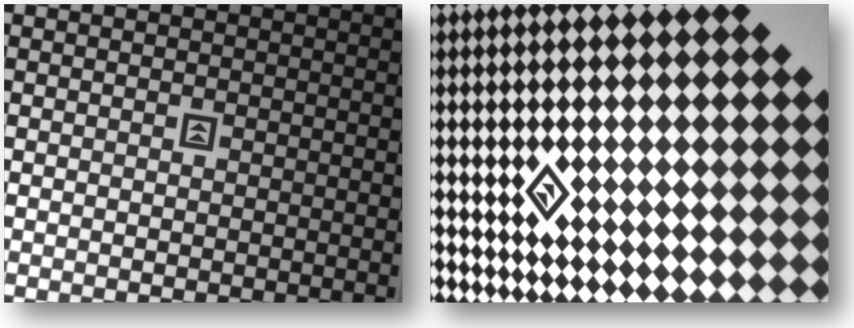

Abbildung A.1: **Kalibrierreferenzen;** Eine solche 2-dimensionale Referenz stellt eine Kalibriermöglichkeit dar. Es sollte ein Muster verwendet werden, welches vom Kalibrieralgorithmus für das Finden bzw. Matchen von Keypoints geeignet ist. Aus diesen kann der Algorithmus die benötigten Transformationsmatrizen berechen.

Die Proben gehen jedoch bereits vor dem Erreichen der Zeilensensoren in den freien Fall über, wodurch sich für einen Beschleunigungsweg von 3 cm eine Beschleunigungszeit von

$$t_a = \frac{v}{a} = \frac{0.767}{9.81} = 78.15ms \tag{A.2}$$

ergibt. Die gesamte Beschleunigungszeit der Proben liegt demnach bei

$$t = t_a + t_b = 22 + 78.15 = 100.15ms. \tag{A.3}$$

Der Fallweg

$$s = \frac{at^2}{2} = \frac{9.81 \cdot 0.1^2}{2} = 49.2mm \tag{A.4}$$

entspricht der minimalen Aufnahmehöhe des Sensors. Diese wird mit 145 mm gewählt, um veränderte Abmessungen gleichsam analysieren zu können. Daraus folgt eine modifizierte Aufnahmebreite von 193.5 mm. Ein Objektiv mit einer Brennweite von $f = 8.5$ mm und die Chipgröße von 4.8×3.6 mm führen zu einem erforderlichen Arbeitsabstand von 430 mm. Der Bereich der Schärfentiefe kann mit Hilfe der Gleichung 3.14 geprüft werden. Eine Verschmierung der Proben durch deren Bewegung während der Aufnahme einzelner Bilder soll vermieden werden. Dazu wird eine Integrationszeit von 0.55 ms verwendet. Die Auflösung wird mit 640×280 gewählt, wodurch eine Bildrate von 117 fps erreicht werden kann.
Da Abstände aus den Bildern gemessen werden sollen, ist eine Kalibrierung des Sensors notwendig. Dafür werden Referenzbilder herangezogen, wie sie in Abbildung A.1 gezeigt werden. Diese werden unter verschiedenen Abständen und Rotationen zum Sensor aufgenommen. Mit Hilfe dieser Referenzdaten und den realen Abmessungen des Setups können die intrinsischen Kameraparameter lt.

$$K_i = \begin{pmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{pmatrix} \tag{A.5}$$

sowie die extrinsischen Parameter

$$K_e = \begin{pmatrix} R & -RC \\ 0 & 1 \end{pmatrix} \tag{A.6}$$

gefunden werden. Mit Hilfe dieser Kalibrierung lässt sich ein Punkt X im Weltkoordinatensystem mittels

$$x = PX = K\{I \mid 0\}X_{CAM} = \begin{pmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{pmatrix} \{I \mid 0\} \begin{pmatrix} R & -RC \\ 0 & 1 \end{pmatrix} X \tag{A.7}$$

auf einen Bildpunkt x abbilden. Darin stellt X_{CAM} einen Punkt im Kamerakoordinatensystem dar. Detailierte Informationen rund um die Kamerageometrie gibt [uAZ04].

A.2 Berechnung der Fallzeiten

Ein Beispielbild einer Aufnahmesession zeigt Abbildung A.2. Der unregelmäßige Verlauf der Intensität der Hintergrundpixel ergibt sich aus den kurzen Belichtungszeiten und der Beleuchtungsgeometrie. Diese kann zur Erleichterung der Bildanalyse durch ein leeres Referenzbild herausgerechnet werden. Die eigentliche Analyse bearbeitet alle aufgenommenen Bilder der Reihe nach in einer Schleife. Zuerst werden die besagten Offsetwerte aus dem Referenzbild addiert. Danach erfolgt eine Konvertierung in Binärbilder, welche morphologisch nachbearbeitet werden. Für jede Probe wird auf jedem Bild Schwerpunkt und Rotationswinkel bzgl. der X-Achse des Bildes berechnet (siehe Abbildung A.2). Je Bild werden die Daten in einer Matrix gespeichert, welche nach Beendigung der Schleife in ein Array zusammengefasst werden. Dieses enthält also Daten je Bild. Eine Konvertierung dieser Daten in objektweise Matrizen der Form

$$\begin{pmatrix} ID & 0 & 0 & 0 \\ Frame_1 & x_{s1} & x_{s2} & \angle_1 \\ \vdots & \vdots & \vdots & \vdots \\ Frame_n & x_{sn} & y_{sn} & \angle_n \end{pmatrix} \tag{A.8}$$

ermöglicht die Fallanalyse einzelner Proben. Darin ist ID eine Probenlaufnummer, die Zeilen $2 \rightarrow n$ enthalten Daten dieser Probe aus einem Bild. Eine Matrix mit $n+1$ Zeilen besagt also, dass diese Probe auf n Bildern enthalten ist. Je Zeile sind Schwerpunktskoordinaten x_{sn} und y_{sn} sowie der Rotationswinkel $\angle_n$ vorhanden. Zur Erstellung dieser Matrizen müssen auf nachfolgenden Bildern die Proben gefunden werden. Dies geschieht mittels Abstandsmaß. Nach Beendigung stehen also so viele Matrizen zur Verfügung, wie Proben auf der Bildsequenz vorhanden sind.
Aus den Bildkoordinaten können nun anhand der Kalibrierparameter beliebige reale Distanzen berechnet werden. Für den Abstand Sensor zu Auswerfer von 49.2 mm wird für jede Probe die Fallzeit über diese Distanzen ermittelt. Dazu wird die Bildrate miteinbezogen. Eine Bildrate von 117 fps entspricht einer zeitlichen Auflösung von $t_{frame} = 8.54\ ms$,

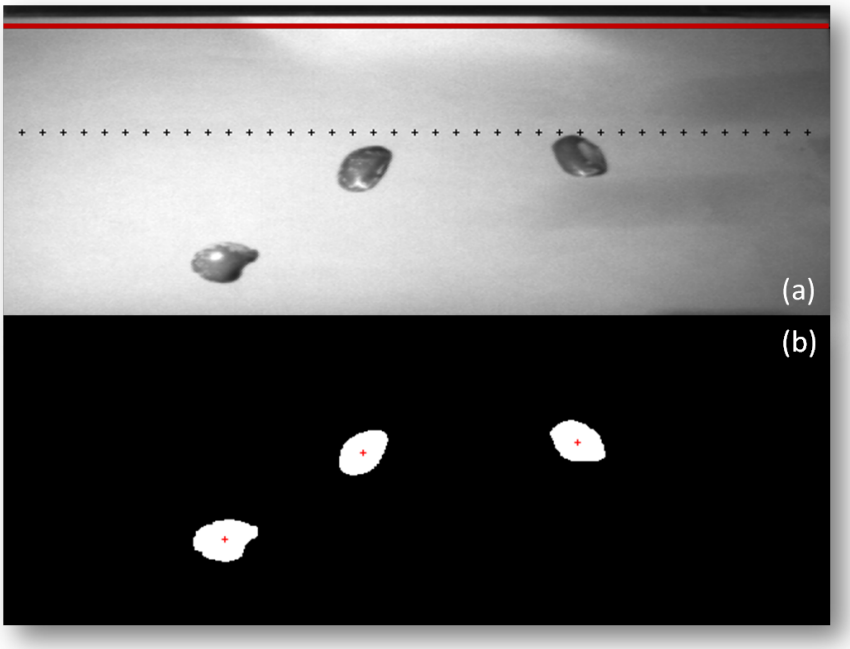

Abbildung A.2: **Freifallszene von Feuerbohnen;** Bild (a) zeigt ein Einzelbild der Aufnahmesequenz. Zusätzlich ist die Aufnahmezeile der Sensoren (rot) sowie die Position der Auswerfer (+) angedeutet. In (b) wird das Ergebnis der Vorverarbeitung dargestellt, wobei die Proben segmentiert und deren Schwerpunkte berechnet wurden.

zwischen den einzelnen Einträgen wird interpoliert. Eine Beispielmatrix für eine Probe ist

$$P_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 157 & 304 & \underline{19} & -6 \\ 158 & 301 & 25 & -13 \\ 159 & 300 & 35 & -16 \\ 160 & 301 & 48 & -16 \\ 161 & 301 & 63 & -17 \\ 162 & 301 & 79 & -17 \\ 163 & 302 & 97 & -18 \\ 164 & 302 & \underline{117} & -19 \\ 165 & 302 & 138 & -19 \\ 166 & 303 & 161 & -19 \\ 167 & 305 & 187 & -16 \\ 168 & 307 & 215 & -13 \end{pmatrix} \tag{A.9}$$

Die Aufnahmezeile des Sensors befindet sich auf Zeile $r_s = 18$, die Auswurfslinie auf Zeile $r_e = 99$. Der Fall wird durch die Bilder mit den Indizes 157 bis 164 beschrieben, was einer Bildanzahl von $\sharp_{frames} = 7$ entspricht. Die Fallzeiten zwischen Zeilen 18 und 19 bzw. Zeilen

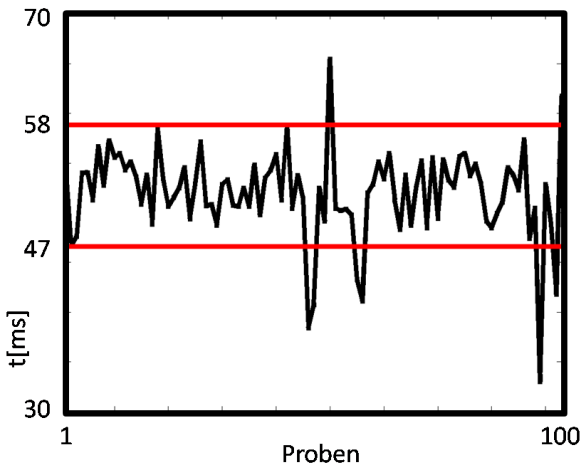

Abbildung A.3: **Fallzeiten von Feuerbohnen;** Die Statistik gibt die Ergebnisse der Fallzeitanalyse für 100 Proben wieder. Die Fallzeit von Zeilensensor bis Auswerfer beträgt $47 - 58ms$. Ausreißer wurden bewusst in den Test eingebracht, diese über- bzw. unterschreiten die Schranken.

99 und 117 werden interpoliert. Damit ergibt sich für diese Probe eine Gesamtfallzeit von

$$\begin{aligned} t_i &= \sharp_{frames} \cdot t_{frame} - \frac{t_{frame}(P_i(9,3) - r_e)}{P_i(9,3) - P_i(8,3)} + \frac{t_{frame}(P_i(2,3) - r_s)}{P_i(3,3) - P_i(2,3)} \\ &= 7 \cdot 8.54 - \frac{8.54(117 - 99)}{117 - 97} + \frac{8.54(19 - 18)}{25 - 19} = 53.51ms. \end{aligned} \tag{A.10}$$

Wird diese Berechnung für alle Proben einer Aufnahmesequenz durchgeführt, finden sich die Fallzeitschranken t_{min} und t_{max}. Abbildung A.3 zeigt eine Ergebnisstatistik von rund 100 Proben.

Anhang B

Wichtige Systemkomponenten

An dieser Stelle wird eine Auflistung wichtiger Komponenten angeführt, welche bei der praktischen Realisierung dieser Arbeit zur Anwendung kamen (und kommen). Zur Betrachtung der Ergebnisse aus Kapitel 6.3 ist die eingesetzte Rechnerhardware am interessantesten. Auch werden Bauteile des Systems zur Bildgewinnung aufgelistet. Softwareseitig werden alle unterstützenden Tools und Bibliotheken angegeben, die entweder in der Echtzeitsoftware inkludiert wurden bzw. zu deren Entwicklung beigetragen haben. Weiters werden performancebestimmende Parameter in der Art festgehalten, in der sie im Zuge der Evaluierung eingestellt wurden.

B.1 Hardware

- **Computer:** Intel Pentium Core2Duo, $2 \times 2.4\ GHz$, 8 $GByte$ RAM;
- **Framegrabber:** Dasla X64-CL Dual;
- **Zeilensensor:** Atmel AViiVA SC2 CL;
- **Objektiv:** Rodenstock ROD APO-RODAGON-N;
- **Beleuchtung:** VCubed VLX2 Linienlicht;

B.2 Software

- **libSVM:** SVM Implementierung, u.A. fr C++;
- **Shark:** C++ Bibliothek mit diversen Lernalgorithmen;
- **Torch:** C++ Bibliothek mit diversen Lernalgorithmen;
- **RapidMiner:** Software zur modularen Entwicklung von Systemen des Maschinellen Lernens, Data Mining sowie umfangreiche Vor- und Nachverarbeitung, mit GUI;
- **Intel Integrated Performance Primitives:** Bibliothek zur Optimierung der Datenverarbeitung für eine gewählte Prozessorarchitektur, Mehrkernoptimierung;

- **Dalsa Sapera LT:** Umfangreiche Bildverarbeitungsbibliothek, 3 Layer;
- **Dalsa Sapera++:** API für Sapera LT in Hochsprache;
- **CommCam:** Sensor zu PC Kommunikationstool;

B.3 Testparameter

- **Zeilenfrequenz:** 10 kHz;
- **Radiometrische Auflösung:** 8 Bit;
- **Zeilenpixel:** 4096, Bayes Farbmosaik;
- **SVM Implementierung:** Algorithmus von John Platt, siehe [Pla98];
- **SVM Kernel:** Linearer Kernel;
- **SVM Komplexitt:** $C = 5$;

Literaturverzeichnis

[Aia04] Specifiactions of the Camera Link. Automated Imaging Association, P.O. BOX 3724, Ann Arbor, MI 48106, USA, 2004.

[Atm05] AVIIVA SC2 CL, Datasheet. Atmel Corporation, 2325 Orchard Parkway, San Jose, CA 95131, USA, 2005.

[AVO68] R. W. Schafer und T. G. Stockham Jr. A. V. Oppenheim. Nonlinear Filtering of Multiplied and Convolved Signals. In *Proc. IEEE, vol 56, no.8*, pages 1264–1291, 1968.

[Bar01] H. J. Bartsch. Taschenbuch Mathematischer Formeln, 19. Auflage. Fachbuchverlag Leipzig, 2001.

[Ben85] K. B. Benson. Television engineering handbook. McGraw-Hill, New York, 1985.

[Bra95] R. N. Bracewell. Two-Dimensional Imaging. Prentice Hall, Upper Saddle River, N.J., 1995.

[Bue07] Sortex Z+ General Brochure 2007. Bühler Sortex Ltd., 20 Atlantis Avenue, London E16 2BF, United Kingdom, 2007.

[Com00] Universal Serial Bus Specification, Revision 2.0. Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V., 2000.

[Dav90] E. Davies. Machine Vision: Theory, Algorithms and Practicalities. Academic Press, New York, 1990.

[dsrumcf90] Two dimensional shape representation using morphological correlation functions. Fourier descriptors for plane closed curves. In *Proc. of the 1990 International Conference of Acoustics, Speech and Signal Processing*, pages 2165–2168, 1990.

[Dye80] C. R. Dyer. Computing the euler number of an image from its quadtree. In *Computer Graphics and Image Processing*, pages 270–276, 1980.

[Fai98] M. D. Fairchild. Color Appearance Models. Addison-Wesley, Reading, Mass, 1998.

[Fir02] Scary Linescan Part Two. Colour Linescan. Firstsight Vision Ltd., The Old Barn, Grange Court, Tongsham, Surry, GU10 1DW, 2002.

[Fis37] R. A. Fisher. The use of multiple measurements in taxonomic problems. In *Annals Eugen, Vol. 7*, pages 179–188, 1937.

[Fle09] H. Fleischhacker. On the Road to Real-Time Feature Extraction from Scarlet Runners. Techn. University, Graz, 2009.

[Fre61] H. Freeman. On the encoding of arbitrary geometric configurations. In *IRE Trans. on Electr. Comp.*, pages 247–255, 1961.

[Fru62] C. Fruwirth. Fisole. In *Handbuch der landwirtschaftlichen Pflanzenzüchtung 6, 2. Auflage*, pages 369–407, Paul Parey Verlag, Berlin, 1962.

[Hat94] L. Hatcher. A step-by-step approach to using the SAS system for factor analysis and structural equation modeling. SAS Institute Press, Cary, NC, 1994.

[Hay96] S. Haykin. Adaptive Filter Theory. Prentice Hall, Upper Saddle River, N.J., 1996.

[Hec01] E. Hecht. Optics, 4th edition. pages 177–192, Addison Wesley, 2001.

[Iee08] IEEE 802.3-2008 – Section One. IEEE, 2008, 2008.

[JK94] J. Novovicova und P. Pudil J. Kittler. Floating search methods in feature selection. In *Pattern Recognition Letters, 15(11)*, pages 1119–1125, 1994.

[Jol02] I. T. Jolliffe. Principal Component Analysis 2nd edition. Springer Verlag, Berlin, 2002.

[Koo24] I. E. Kooistra. Bohnen. In *Handbuch der Pflanzenzüchtung 3, 5. Auflage*, pages 177–192, Paul Parey Verlag, Berlin, 1924.

[Mah36] P. C. Mahalanobis. On the generalised distance in statistics. In *Pro. of the National Institute of Science of India*, pages 49–55, 1936.

[Mar82] D. Marr. Vision. Freeman, San Francisco, 1982.

[Pav77] T. Pavlidis. Structural Pattern Recognition. Springer Verlag, Berlin, 1977.

[PG99] H. Haußecker und B. Jähne P. Geißler. Handbook of Computer Vision and Applications Volume 1 - Sensors and Imaging. Academic Press, 1999.

[Pin97] A. Pinz. Über das Sehen. In *Bildverstehen*, Springer Verlag KG, 1997.

[Pla98] J. C. Platt. A Fast Algorithm for Training Support Vector Machines. In *Technical Report MSR-TR-98-14*, Microsoft Research, 1998.

[Pog02] J.C. Poggendorff. Zur Theorie der Farbmischung. In *Poggendorff's Annalen der Physik und Chemie (89)*, pages 69–84, 2002.

[Pre70] J. M. S. Previtt. Object enhancement and extraction. In *Picture Processing and Psychopictorics*, Academic Press, New York, 1970.

[Rob65] L. G. Roberts. Machine perception of three-dimensional solids. In *Optical and Electro-Optical Information Processing*, MIT Press Cambridge Mass, 1965.

[Ros74] A. Rosenfeld. Digital Straight Line Segments. In *IEEE Trans. on Computers*, pages 1264–1269, 1974.

[Ros00] A. Rosenfeld. Image Analysis and Computer Vision: 1999. In *Computer Vision and Image Understanding, Vol. 78, Nr. 2*, pages 222–302, 2000.

[Sav88] M. Savini. Moments in image analysis. In *Alta Frequenca*, pages 145–152, 1988.

[Sch07] CCD-Zeilenkameras. Schäfter und Kirchhoff, Kieler Strasse 212, D-22525 Hamburg, 2007.

[Ser82] J. Serra. Image Analysis and Mathematical Morphology. Academic Press, New York, 1982.

[Ski08] S. S. Skiena. The Algorithm Design Manual, 2nd edt. Springer, Berlin, 2008.

[Smi78] A. R. Smith. Color Gamut Transform Pairs. In *Proc. SIGGRAPH 78, Computer Graphics, Vol. 12, Nr. 3*, pages 12–19, 1978.

[Tim98] S. Timmann. Repetitorium der Analysis Teil 2. Binomi, Springe, 1998.

[Tra04] TA Document 2003017, IIDC 1394-based Digital Camera Specification Ver.1.31. 1394 Trade Association, 1111 South Main Street, Suite 100, Grapevine, TX 76051, USA, 2004.

[uAJS02] B. Schölkopf und A. J. Smola. Learning with Kernels. MIT Press, 2002.

[uAS09] A. Guillen A. Lendasse I. Rojas G. Rubio und A. Sorjamaa. Mutal Information Based Initialization of Forward-Backward Search for Feature Selection in Regression Problems. In *LNCS - Artificial Neuronal Networks Part 1, Vol. 5768*, pages 1–9, 2009.

[uAZ04] R. Hartley und A. Zisserman. Multiple View Geometry in Computer Vision. 2nd edt. Cambridge University Press, 2004.

[uBSR96] B. N. Chatterji und B. S. Reddy. A fft-based technique for translation, rotation and scale invariant image registration. In *IEEE Trans. on Image Processing*, pages 1266–1271, 1996.

[uCC00a] K. P. Bennett und C. Campbell. Support Vector Machines: Hype or Hallelujah? In *SIGKDD Explorations Vol. 2*, 2000.

[uCC00b] Y. Bresler und C. Couvreur. On the optimality of the backward greedy algorithm for the subset selection problem. In *SIAM Journal on Matrix Analysis and Applications*, 2000.

[uCJL03] F. Chang Chun-Jen Chen und Chi-Jen Lu. A linear-time component-labeling algorithm using contour tracing technique. In *Computer Vision and Image Understanding*, Institute of Information Sciences, Nankang, Taipei, 2003.

[uCMB82] D. H. Ballard und C. M. Brown. Computer Vision. Prentice Hall, Englewood Cliffs, N.J., 1982.

[uCT98] R. Manduchi und C. Tomasi. Bilateral Filtering for Gray and Color Images. In *Proceedings of the Sixth International Conference on Computer Vision*, Washington DC, 1998.

[uCTZ72] R. Z. Roskies und C. T. Zahn. Fourier descriptors for plane closed curves. pages 269–281, Prentice Hall, Englewood Cliffs, N.J., 1972.

[uDGS01] R. O. Duda P. E. Hart und D. G. Stork. Pattern Classification. John Wiley and Sons, New York, 2001.

[uELZ97] N. Guil J. Villalba und E. L. Zapata. A fast hough transform for segment detection. In *IEEE Trans. on Image Processing*, pages 1744–1792, 1997.

[uGCS01] L. G. Shapiro und G. C. Stockman. Computer Vision. Prentice Hall, Upper Saddle River, N.J., 2001.

[uHL97] M. Dash und H. Liu. Feature selection for classification. Intelligent Data Analysis 1. 1997.

[uHM07] H. Liu und H. Motoda. Computational Methods of Feature Selection. Chapman and Hall, 2007.

[uHN88] J. R. Magnus und H. Neudecker. Matrix differential calculus whith applications in statistics and econometrics. John Wiley and Sons, 1988.

[uJK82] P. A. Devijver und J. Kittler. Pattern Recognition: A Statistical Approach. Prentice-Hall, London, 1982.

[uJOE91] A. Bengtsson und J. O. Eklundh. Shape Representation by Multiscale Contour Approximation. In *IEEE Trans. Pattern Anal. Machine Intell., Vol. 13, Nr. 1*, pages 85–93, 1991.

[uJS89] D. Mumford und J. Shah. Optimal Approximations by Piecewise Smooth Functions and Associated Varational Problems. In *Communications on Pure and Applied Mathematics, Vol. XLII*, pages 577–685, John Wiley and Sons Inc., 1989.

[uJS00] B. Dom W. Niblack und J. Sheinvald. A modeling approach to feature selection. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 535–539, IBM, San Jose, CA, 2000.

[uJS03] F. Jarre und J. Stoer. Optimierung. Springer Berlin, 2003.

[uJV80] C. J. Cornelisse J. S. Ploem A. W. M. Smeulders A. M. Vossepoel und J. Vrolijk. Some shape parameters for cell recognition. In *Proc. of Pattern Recognition in Practice*, 131-142, 1980.

[uKL07] T. H. Cormen C. E. Leiserson R. L. Rivest C. Stein P. Molitor M. Krieger-Hauwede und K. Lippert. Algorithmen - Eine Einführung. Oldenbourg, 2007.

[uLS98] J. Kittler M. Petrou und L. Shafarenko. Histogram-based segmentation in a perceptually uniform color space. In *IEEE Trans. on Image Processing*, pages 1345–1358, 1998.

[uLS03] W. Haerdle und L. Simar. Applied Multivariate Statistical Analysis. Springer, New York, 2003.

[uLS09] T. Mellouli und L. Stuhl. Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen. Springer Berlin, 2009.

[Umb98] S. E. Umbaugh. Computer Vision and Image Processing: A Practical Approach using CVIPtools. Prentice Hall, Upper Saddle River, N.J., 1998.

[uMIS85] F. P. Preparata und M. I. Shamos. Computational Geometry: An introduction. Springer Verlag, New York, 1985.

[uML07] K. Brier R. Fehrer F. Göttl O. Hofer und M. Langer. Grüner Bericht 2007. Technical report, Bundesministerium für Land- und Forstwirtschaft, Umwelt und Wasserwirtschaft, Abteilung II 5, Wien, Österreich, 2007.

[uMS88] C. Harris und M. Stephens. A combined corner and edge detector. In *Proc. of the 4th ALVEY Vision Conference*, pages 147–151, 1988.

[uPG05] I. Borg und P. Groenen. Modern Multidimensional Scaling: Theory and Applications, 2nd edition. Springer, New York, 2005.

[uPH73] R. Duda und P. Hart. A 3x3 isotropic gradient operator for image processing. In *Pattern Classification and Scene Analysis from John Wiley and Sons*, pages 271–283, 1973.

[uPM02] D. Comaniciu und P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. In *IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24, Nr. 5*, 2002.

[uRC01] S. Bengio und R. Collobert. SVMTorch: Support Vector Machines for Large-Scale Regression Problems. In *Journal of Machine Learning Research*, 2001.

[uREW02] R. C. Gonzalez und R. E. Woods. Digital Image Fundamentals. In *Digital Image Processing, 2nd edition*, pages 34–74, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2002.

[uRS96] H. Liu und R. Setiono. A probabilistic Approach to Feature Selection - A Filter Solution. In *Proceedings of the International Conference on Machine Learning*, pages 319–327, 1996.

[uSP07] P. Hebert und S. Perreault. Median filtering in constant time. In *IEEE Trans. on Image Processing*, pages 2389–2394, 2007.

[uTP87] H. Ailisto und T. Piironen. Evaluation of color representation methods in a practical vision system. In *Proceedings of 5th SCIA*, Stockholm, 1987.

[uTYZ] C. Y. Suen und T. Y. Zhang. A Fast Parallel Algorithm for Thinning Digital Patterns. In *Comm. ACM, Vol. 27, Nr. 3*, pages 236–239.

[uVA07] D. T. Pham und V. Aslantas. Depth from automatic defocusing. In *Optic Express 1011*, 2007.

[uVV95] C. Cortes und V. Vapnik. Support vector networks. In *Machine Learning 20*, pages 273–297, 1995.

[uWBS96] S. Dutta und W. B. Seals. Everywhere-in-focus image fusion using controllable cameras. In *Proc. of Sensor Fusion and Distributed Robotic Agents*, 1996.

[Wei] B. Weiss. Fast median and bilateral filtering. Shell and Slate Software Corp.