technische universität dortmund

lehrstuhl informatik 12

Lehrstuhl für Förder- und Lagerwesen

Endbericht

Projektgruppe: Camera-assisted Pick-by-feel

29. Mai 2015

Autoren:

Birol Sevim
Christian Matuschek
Daniel Schulze Bisping
Florian Schulz
Jannik Zappe
Malte Alexander Baumann
Matthias Fey
Matthias Neuhaus
Sebastian Kurth
Sigo Rosenkranz
Uthenthira Sivapatham

Betreuer:

Prof. Dr.-Ing. Gernot A. Fink
M. Sc. René Grzeszick
Dipl.-Inform. Sascha Feldhorst
Dipl.-Inform. Christian Mosblech

Technische Universität Dortmund
Fakultät Informatik
Lehrstuhl 12
http://ls12-www.cs.tu-dortmund.de

Technische Universität Dortmund
Fakultät Maschinenbau
Lehrstuhl für Förder- und Lagerwesen
http://www.flw.mb.tu-dortmund.de

Inhaltsverzeichnis

KAPITEL 1

Einleitung

Die Projektgruppe Camera-assisted Pick-by-feel hat sich mit der Entwicklung eines neuartigen Kommissionierleitsystem beschäftigt. Kommissionierleitsysteme dienen als Hilfsmittel für die Kommissionierung, welche ein Aufgabengebiet der Logistik ist. Die Tätigkeit der Kommissionierung ist das Sammeln von Artikeln aus einem Gesamtbestand, z. B. einem Warenlager, auf Basis von Aufträgen. Ein Lagermitarbeiter, der Kommissionierer, erhält eine Liste von Artikeln mit Lagerort und gesuchter Anzahl, z. B. in Form eines Zettels. Seine Aufgabe als Kommissionierer ist es, die Artikel der Reihenfolge nach zu kommissionieren und zu einem definierten Zielgebiet zurückzubringen. Als Kommissionierleitsystem funktioniert in diesem Beispiel der Zettel als Pickliste, die einfachste Ausführung eines solchen Systems. Weitere Systeme sind z. B. Pick-by-light-Systeme, bei denen im Lager jede Entnahmestelle mit Anzeigefeldern ausgestattet ist, um so den Kommissionierer zu führen, oder Pick-by-voice-Systeme, bei denen der Kommissionierer mittels Headset im ständigen Kontakt zu einem Computer steht, welcher die benötigten Informationen mitteilt. Diese und weitere Systeme haben unterschiedliche Vor- und Nachteile. Oftmals sind sie teuer in der Anschaffung, umständlich zu bedienen, haben einen hohen Verwaltungsaufwand oder belasten die Konzentration des Kommissionierers über den ganzen Arbeitstag hinweg.

Das System der Projektgruppe hat zum Ziel bei geringen Kosten und Aufwand den Kommissionierer in seiner Tätigkeit so wenig wie möglich zu beeinträchtigen. Es besteht aus Smartphone, Smartwatch und Server (siehe Abbildung 1.1). Die Smartwatch stellt alle fürs Kommissionieren benötigten Informationen zur Verfügung, z. B. Artikel, Anzahl und Lagerort. Das Smartphone kommuniziert mit dem Server über WLAN und mit der Smartwatch per Bluetooth. Es ist die Schnittstelle zwischen Server und Smartwatch. Das Smartphone wird mit einem Gurt am Körper getragen, mit der Kamera nach außen. Das

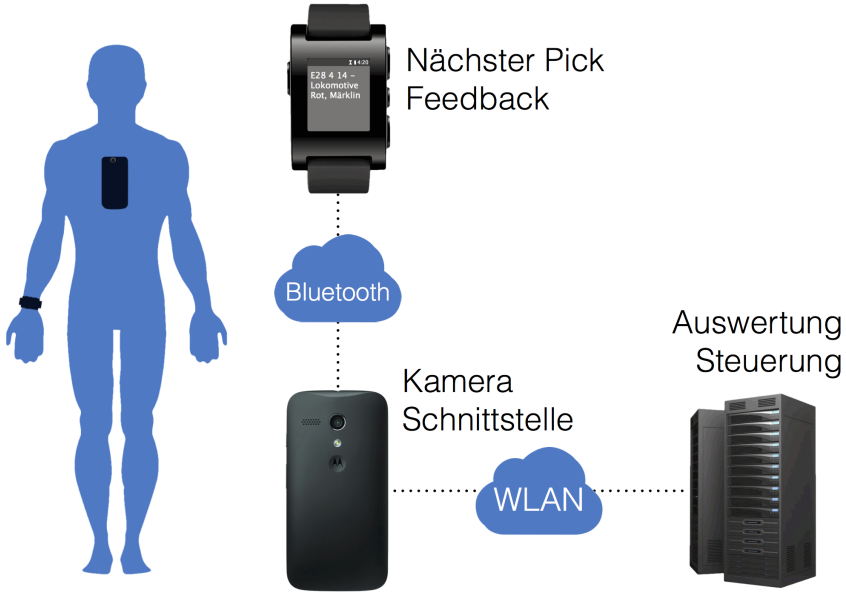


Abbildung 1.1: Übersicht über die Komponenten Smartphone, Smartwatch und Server.

Smartphone sollte auf ähnlicher Höhe wie in Abbildung 1.1 gezeigt getragen werden. Mithilfe des Smartphones können ohne direkte Einwirkung des Kommissionierers Bilder der kommissionierten Artikel gemacht werden. Die Kamera nimmt nicht durchgängig Bilder auf. Es werden erst Bilder aufgenommen, sobald eine Bewegungserkennung feststellt, dass der Kommissionierer stehen geblieben ist. Dies wird dadurch erklärt, dass nur Bilder von den zu kommissionierenden Artikel gemacht werden müssen, und dass der Kommissionierer dann stehen bleibt, wenn er den Lagerplatz dieses Artikels erreicht hat. Die Bilder werden per WLAN an den Server weitergeleitet. Der Server überprüft anhand von Barcode-Detektion, Image Retrieval und Klassifikation, ob das Bild den aktuell gesuchten Artikel zeigt. Wird der entsprechende Barcode gefunden, oder wird auf dem Bild der Artikel erkannt, so wird eine Erfolgsnachricht an die Smartwatch gesendet. Ab sofort wird dieser Artikel als erkannt angezeigt und der Kommissionierer kann zum nächsten Artikel übergehen.

Wird das System beim Kommissionieren benutzt, so müssen Smartwatch und Smartphone am Körper getragen werden. Zudem müssen die Verbindungen zwischen Server, Smartwatch und Smartphone aufgebaut sein. Ein Blick auf die Smartwatch genügt, um zu erfahren, welcher Artikel gesucht wird und wo er sich befindet. Der Kommissionierer sucht dann den Lagerplatz auf. Ist dieser erreicht, bleibt der Kommissionierer stehen. Die Bewegungserkennung erkennt dies, und das Smartphone nimmt Bilder auf. Zur selben Zeit greift der Kommissionierer schon den Artikel und hält ihn in das Blickfeld der Kamera. Die Bilder werden an den Server gesendet, wo sie ausgewertet werden. Hat der Kommissionierer den korrekten Artikel gewählt, so erhält er eine entsprechende Benachrichtigung und

kann sich den nächsten Artikel auf der Smartwatch anzeigen lassen. Er packt den Artikel ein und macht sich auf den Weg zum nächsten Artikel.

Das System der Projektgruppe soll eine kostengünstige Alternative zu bestehenden modernen Kommissioniersystemen bilden. Unter der Annahme, dass ein Server stets vorhanden ist, werden nur Smartphone und Smartwatch zusätzlich benötigt. Dabei handelt es sich um Alltagsgegenstände und vom System her werden keine hohen Anforderungen an diese zwei Gegenstände gestellt. Daher sollte der Preis für Smartphone und Smartwatch wenige hundert Euro betragen. Das entwickelte System bietet eine ergonomische Lösung mit hoher Kommissionierleistung und geringem Handhabungsaufwand. So bleibt der Kommissionierer uneingeschränkt in seiner Tätigkeit und hat stets seine Hände frei. Zusätzlich reduziert das System die Fehlerrate der falsch gewählten Artikel mit Hilfe von Barcode- und Bilderkennung.

Im folgenden Kapitel wird die Organisation und der Ablauf der Projektgruppe beschrieben. Kapitel 3 wiederholt die Ergebnisse der Seminarphase. Kapitel 4 erläutert die Funktionen, die Anwendungsfälle und Betriebsbedingungen. Im Kapitel 5 werden die eingesetzten Bibliotheken aufgezählt. Kapitel 6 erklärt den Aufbau der Software sowohl serverseitig, als auch auf Smartphone und auf Smartwatch. Kapitel 7 widmet sich der Evaluation. Kapitel 8 enthält die Tests des Systems und deren Auswertung. Kapitel 9 schließt den Endbericht mit einer Zusammenfassung und anschließendem Ausblick ab.

KAPITEL 2

Organisation der Projektgruppe

In großen Softwareprojekten entwickeln sich viele organisatorische Aufgaben. In der Projektgruppe waren es bspw. die Leitung wöchentlicher Treffen oder die Wartung eingesetzter Werkzeuge.

2.1 Aufgaben und Werkzeuge

Bei der Durchführung der Projektgruppe fielen organisatorische Aufgaben an, wie die Leitung der regelmäßigen Treffen, die Protokollführung oder die Wartung von Wiki und Versionsverwaltung. In dieser Projektgruppe kam Redmine [6] als Projektmanagementsoftware zum Einsatz. Das darin integrierte Wiki wurde zur Protokollführung verwendet. Weiter wurde das integrierte Ticketsystem zum Festhalten und Nachverfolgen von Fehlern und Aufgaben verwendet. Zur Versionsverwaltung wurden Git-Repositories [7] verwendet, welche in das Redmine-System integriert und auf diese Weise einfach zu verwalten waren. Die Kommunikation der Projektgruppe verlief primär innerhalb der Treffen oder über eine Mailingliste, welche zu diesem Zweck erstellt wurde.

Die Leitung der wöchentlichen Treffen mit allen Mitgliedern der Projektgruppe hat Matthias Neuhaus gleich zu Beginn des ersten Semesters übernommen. Die Protokollführung erfolgte in einem abwechselnden Rhythmus. Die Pflege des Wikis und des Versionsverwaltungssystems haben Sebastian Kurth und Sigo Rosenkranz übernommen. Bei dieser Tätigkeit ging es primär darum, das Wiki übersichtlich zu halten und bei Bedarf weitere Git-Repositories aufzuschalten.

2.2 Arbeitsgruppen

Das im Rahmen der Projektgruppe entwickelte System besteht aus mehreren Komponenten. Die Implementierung der Anwendungen fand dabei auf unterschiedlichen Plattformen statt. So mussten sowohl Serveranwendungen, als auch Anwendungen für Smartphone und Smartwatch geschrieben werden. Weiterhin mussten zwischen diesen Anwendungen Nachrichten ausgetauscht werden. Da sich diese Plattformen gut voneinander abgrenzen lassen, haben wir die Projektgruppe in drei Gruppen aufgeteilt. Passend dazu entstanden so die Gruppen **Server**, **Android** und **Kommunikation**. Die Servergruppe kümmerte sich dabei um die Implementierung der serverseitigen Software, wohingegen sich die Androidgruppe mit der Realisierung der Android-App, der Pebble-App sowie der Kommunikation zwischen diesen beschäftigte. Durch die Kommunikationsgruppe wurde sichergestellt, dass ein reibungsloser Nachrichtenaustausch zwischen Server und Android erfolgte. In gruppeninternen Treffen wurde über die erzielten Fortschritte diskutiert, gemeinsam an Problemen gearbeitet und sich gruppenspezifisch organisiert. In den wöchentlichen Treffen der gesamten Projektgruppe wurden die Fortschritte der einzelnen Gruppen vorgestellt, diskutiert und abgestimmt. Tabelle 2.1 gibt einen Überblick über die drei Arbeitsgruppen sowie deren Mitgliedern.

Die Erarbeitung von schriftlichen Arbeiten wie dem Pflichtenheft, Zwischenbericht oder diesem Endbericht erfolgten in gemeinschaftlicher Arbeit. Kapitel wurden so aufgeteilt, sodass der Aufwand möglichst gleichmäßig auf alle Projektgruppenteilnehmer verteilt wurde. Weiter wurden für alle Kapitel auch andere Projektgruppenteilnehmer als der Autor selbst bestimmt, die diese gegenlesen und im Detail verbessern sollten. Präsentationen wurden i.d.R. von einer kleineren Gruppe erstellt und vorgetragen. Gewählt wurden diese Gruppen von allen Teilnehmern der Projektgruppe.

Arbeitsgruppe	**Mitglieder**
Server	Christian Matuschek Daniel Schulze Bisping Florian Schulz Matthias Fey Matthias Neuhaus Uthenthira Sivapatham
Android	Malte Alexander Baumann Sigo Rosenkranz Birol Sevim
Kommunikation	Sebastian Kurth Jannik Zappe

Tabelle 2.1: Arbeitsgruppen der Projektgruppe und deren Mitglieder

2.3 Erstes Semester

Zu Beginn der Projektgruppe wurde im ersten Semester eine Seminarphase veranstaltet. In dieser war es die Aufgabe eines jeden Teilnehmers der Projektgruppe, eine 40-minütige Präsentation über eines von zwölf Themen zu halten. Diese Themen bestanden aus den vier Hauptthemen Software-Entwicklung, Logistik, Bildverarbeitung sowie Bilderkennung und waren so gewählt, dass sie gezielt zur Lösung eines Teilbereichs der Projektgruppe dienten. Damit wurden die Grundlagen für die Lösung der späteren Aufgaben der Projektgruppe geschaffen.

Im Anschluss wurden in wöchentlichen Treffen ein für die Projektgruppe verbindliches Pflichtenheft angefertigt, welches Punkte wie Produktentwurf, Produkteinsatz, Produktumgebung, Produktfunktionen, Zielbestimmungen, Produktleistungen, die Benutzeroberfläche sowie Testfälle abdeckte.

Nach Einteilung in Kleingruppen (siehe auch Kapitel 2.2) konnten erste Planungen und Implementierungsansätze zur Umsetzung von Teilaufgaben vorangetrieben werden. Zum Ende des ersten Semesters wurde ein Zwischenbericht angefertigt, welcher noch einmal zusammenfassend alle Themen der Seminarphase beinhaltete. Hierfür war es die Aufgabe jedes Teilnehmers eine ca. 10-seitige Ausarbeitung zu seinem eigenen Thema zu erstellen. Zusätzlich wurde in diesem Bericht der aktuelle Entwicklungsstand festgehalten und die Planung für das darauf folgende zweite Semester festgelegt.

2.4 Zweites Semester

Mit dem Start des zweiten Semesters wurde die Implementierungsphase begonnen. Wie in Kapitel 2.2 beschrieben, wurden die Kleingruppen aus dem ersten Semester beibehalten, die Teilaufgaben der Projektgruppe lösten. In Treffen, die zweimal in der Woche stattfanden, wurde zudem in der großen Gruppe mit allen Mitgliedern versucht, die Fortschritte und angefallenen Probleme zu diskutieren. Eine Präsentation wurde als Zwischenpräsentation abgehalten, indem der aktuelle Stand der Entwicklung noch einmal aufgezeigt werden sollte.

Kurz vor Abschluss der Projektgruppe stand die Endpräsentation und das Schreiben des Endberichts im Vordergrund. Im Zuge der Endpräsentation wurde das Produkt in seiner finalen Version ausgiebig mit Einbeziehung aller Features vorgestellt. Es konnte ein kompletter Kommissioniervorgang mit verschiedenen Artikeln aus unterschiedlichen Kategorien durchgeführt werden, wobei auch die Gelegenheit gegeben war, die einzelnen Hardwarekomponenten des Produkts zu zeigen. Insbesondere war zu verzeichnen, dass durch die koordinierte Zusammenarbeit innerhalb der Projektgruppe eine zufriedenstellende Vorstellung erfolgt ist. Die Absprachen zwischen den einzelnen Arbeitsgruppen Server, Android und Kommunikation konnte über die meiste Zeit erfolgreich eingehalten werden, so dass letztendlich bei der Umsetzung nur geringe Verzögerungen entstanden sind.

KAPITEL 3

Grundlagen - Ergebnisse der Seminarphase

In der Seminarphase wurden Themen aus der Informatik und Logistik erarbeitet, die von den Teilnehmern der Projektgruppe vorgestellt worden sind. Die Themen Kommissionierung, Bildverarbeitung und Bilderkennung werden im folgenden Kapitel aufgegriffen, da sie die Grundlage zur weiteren Arbeit bilden.

3.1 Kommissionierung

Die Kommissionierung beschreibt das Einsortieren bzw. Entnehmen von Artikeln in einem Lager. Lager sind dabei nach einem bestimmten Muster aufgebaut und es existieren diverse unterschiedliche Lagermittel, auf welche in Abschnitt 3.1.1 genauer eingegangen wird. Die Arbeit des Einlagerns oder der Entnahme von Artikeln wird i.d.R. von Menschen übernommen, welchen jedoch technische Hilfsmittel zur Verfügung gestellt sind (siehe dazu auch Abschnitt 3.1.2). Diesen Hilfsmitteln sind jedoch unterschiedliche Vor- und Nachteile angelastet. Sie sind oft entweder teuer, statisch, belastend oder störend für den Kommissionierer oder eine Kombination aus diesen.

Die Projektgruppe hatte die Aufgabe ein Kommissionierleitsystem zu entwickeln das keines dieser Nachteile besitzt. Die Wahl ist deshalb auf eine Kombination aus Standardkomponenten gefallen, die leicht im freien Handel zu erwerben sind. Jeder Kommissionierer wird mit einem Smartphone und einer Smartwatch ausgestattet (siehe Kapitel 6). Ausserdem werden die Verteilung der Kommissionieraufträge und Auswertungen der anfallenden Daten von einem Server übernommen (siehe Kapitel 6.1). Die Smartwatch dient als Anzeige für das Smartphone, welches selber als Schnittstelle zum Server fungiert. Die Hauptgründe für diese Komponenten waren die gute Verfügbarkeit, der geringe Preis, das geringe

Gewicht und die Vertrautheit des Kommissionierers zu den Komponenten. Eine genauere Beschreibung ist in Kapitel 7 gegeben.

3.1.1 Kommissioniersysteme - Allgemein

In diesem Abschnitt wird auf die Eigenschaften und Hilfsmittel der Kommissionierung eingegangen.

Lager

Lager gibt es in verschiedenen Bereichen. Sie können vom Regal Zuhause für den kleinen Online-Shop bis hin zu riesigen Lagern von Amazon reichen. Viele Firmen ohne Shop haben ebenfalls Lager, meist für die interne Handhabung von Komponenten. Gemein haben alle Lager, dass Waren nach einem bestimmten Prinzip abgelegt und bei Kommissionieraufträgen entnommen werden.

Was die Lager beinhalten ist dabei ebenso wenig vorgegeben wie der Ort, an dem ein bestimmtes Gut abgelegt wird. Wenn eine Ware einen bestimmten Lagerplatz hat, kann dieser von Kommissionierern schnell gefunden werden und es fällt frühzeitig auf, wenn der Bestand zu Ende geht. Allerdings wird viel Platz verbraucht, wenn das Lager nicht komplett gefüllt ist. Daher setzen vor allem große Lager auf das Prinzip der chaotischen Lagerhaltung. Bei diesem Prinzip kann jeder Platz jede Ware halten. Der Platz wird nur mit der Priorität ausgewählt, dass um diesen herum keine Artikel aus der gleichen Kategorie lagern. Sonst folgt die Platzwahl keinem besonderem Prinzip, daher auch der Name **chaotisch**. Ein Buch würde bspw. neben Spielsachen und Elektrogeräten lagern, und nicht neben anderen Büchern.

Zonen

Ein großes Lager ist typischerweise in Zonen aufgeteilt. Die Einteilung wird meist aus organisatorischen oder technischen Gründen vollzogen. Zum Beispiel können Hochregallager nur mit technischer Unterstützung betrieben werden.

In Abbildung 3.1 ist ein Lager mit vier Zonentypen abgebildet. Am oberen Ende wird ein Auftrag angenommen und in zwei Teilaufträge aufgeteilt. In der orangenen Zone wird der erste Teilauftrag bearbeitet. Der Kommissionierer empfängt den Teilauftrag und läuft die Reihen ab, um alle Gegenstände auf der Pickliste (siehe 3.1.2) aus den Regalen zu nehmen. Am unteren Ende der orangenen Zone übergibt er die kommissionierten Güter an die Zusammenstellungszone. In der gelben Zone wird der zweite Teilauftrag von einem automatischem Kommissioniersystem zusammengestellt. Der Kommissionierer teilt die gelieferten Güter den entsprechenden Aufträgen zu. In der roten Zone werden beide Teilaufträge wieder zu dem Gesamtauftrag vereint. Sobald dies geschehen ist, ist der Auftrag abgearbeitet.

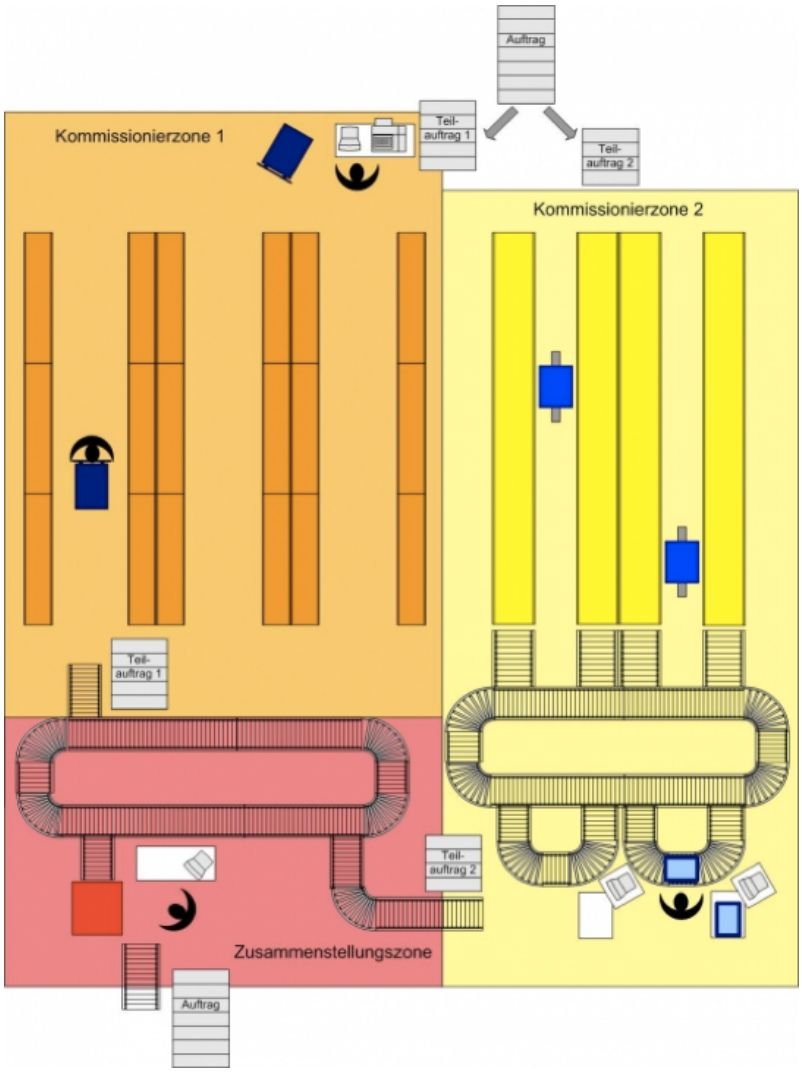


Abbildung 3.1: Ein Lager mit 3 Zonen. Die orangene Zone zeigt das Prinzip von Person zur Ware, die gelbe Zone das Prinzip Ware zur Person und in der roten Zone werden die Teilaufträge konsolidiert (Quelle: http://www.fml.mw.tum.de/fml/index.php?Set_ID=320&letter=M&b_id=3744417B-4442-3530-302D-433545442D34)

Aufträge

Die Anforderung von gelagerten Gütern für die Entnahme wird Auftrag genannt. Ein Auftrag kann beliebig viele Güter enthalten.

Einstufiges Kommissionieren Für die Bearbeitung der Aufträge gibt es verschiedene Verfahren, die drei vorherrschenden werden nun vorgestellt.

Einzelauftrag Ein Kundenauftrag wird in einem einzigen Durchgang durchgeführt. Dabei werden die Güter nacheinander kommissioniert. Diese Vorgehensweise ist sehr über-

sichtlich und schnell erlernbar. Außerdem entfällt die Vereinzelung und die Zusammenstellung.

Auftragsgruppen Die Positionen mehrerer Aufträge werden in Gruppen aufgeteilt, so dass ein Kommissionierer gleichzeitig Teile von unterschiedlichen Aufträgen bearbeitet. Nach der Kommissionierung werden die Positionen einer Auftragsgruppe wieder dem Kundenauftrag zugeordnet und mit den restlichen Teilen des Kundenauftrags vereint. Das gleichzeitige Bearbeiten mehrerer Aufträge erhöht die Effizienz eines Kommissionierers bei der Entnahme. Jedoch fällt mit der Vereinzelung und Zusammenführung ein Mehraufwand an, der nur mit technischer Unterstützung zu bewältigen ist.

Teilaufträge und Konsolidierung Ein Auftrag wird in mehrere Teile aufgeteilt und von mehreren Kommissionierern gleichzeitig bearbeitet. Jeder Kommissionierer bearbeitet allerdings nur einen Teilauftrag gleichzeitig. Diese Technik ist vor allem bei größeren Lagern wichtig, da die langen Wege sonst eine erhebliche Verzögerung erzeugen würden. Die Konsolidierung wird vereinfacht, da die Zuordnung zu den Kundenaufträgen entfällt. Die Effizienz eines Kommissionierers ist allerdings nicht so hoch wie bei den Auftragsgruppen.

Zweistufiges Kommissionieren Das zweistufige Kommissionieren ist vor allem interessant, wenn viele Artikel einer Ware gleichzeitig verkauft werden. Der Kommissionierer bearbeitet in diesem Fall keinen Kundenauftrag direkt, sondern Auftragsstapel oder auch Batches genannt. Diese enthalten die Anzahl der zu entnehmenden Güter, die sich gewöhnlich aus einer Vielzahl von Kundenaufträgen zusammensetzt. Bei der Vereinzelung werden die Teilmengen der entnommenen Güter dann den Kundenaufträgen zugeteilt. Bestellt bspw. Kunde A 10 Feuerzeuge und Kunde B 20 Feuerzeuge, steht auf der Pickliste des Kommissionierers nur 30 Feuerzeuge. Diese Technik erlaubt dem Kommissionierer maximale Effizienz bei der Entnahme, erzeugt aber eine schwierige Vereinzelung.

Position vs. Pick

Für die spätere Erklärung der Abläufe muss zwischen den beiden Begriffen *Position* und *Pick* unterschieden werden.

Position ist eine Zeile in einem Kommissionierauftrag. Sie beschreibt den Standort und die Ware. Siehe 3.1.2

Pick wird das Greifen eines Artikels genannt.

Person und Ware

Es wird grundsätzlich zwischen zwei Bewegungen unterschieden. Entweder bewegt sich die Person oder die Ware. Die Methoden können auch kombiniert werden.

Person zur Ware bedeutet, dass die Ware statisch gelagert ist und die Person sich zu der Ware bewegt. Siehe orangener Kommissionierzone in Abbildung 3.1.

Ware zur Person bedeutet, dass die Ware transportiert wird und die Person an einem bestimmten Platz steht. Siehe gelber Kommissionierzone in Abbildung 3.1.

Technische Komponenten

Die technischen Komponenten werden in vier Kategorien unterteilt.

Lagermittel Es wird zwischen statischen und dynamischen Fördermitteln unterschieden. Sind sowohl das Lagergut als auch das Regal ortsfest, wird das Lagermittel als statisch bezeichnet. Wird die Ladeeinheit nach dem Einlagern bewegt, ist das Lagermittel dynamisch. Siehe Abbildung 3.3 und Abbildung 3.2.

Abbildung 3.2: Statisches Lagermittel: Fachbodenregal (Quelle: http://www.fachbodenregale.com/images/Fachbodenregale_600.jpg)

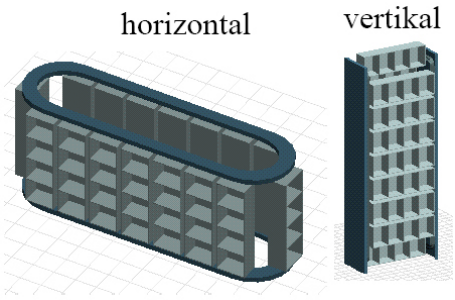


Abbildung 3.3: Dynamisches Lagermittel: Umlaufregal (Quelle: http://www.iwiki.de/wiki/images/4/49/Umlaufregal.jpg)

Fördermittel werden für den Transport von Gütern und Personen verwendet. Zusätzlich können Fördermittel für das Verteilen, Sortieren, Puffern oder Zwischenlagern von Gütern verwendet werden. Zum Beispiel Niederhubwagen, Sorter für Ware zu Person, siehe Abbildung 3.4.

Handhabungsmittel wird für die Sortierung und Verlagerung von Gütern verwendet. Zum Beispiel Industrieroboter und Greifer.

Ladehilfsmittel können Artikel zu Ladeeinheiten bündeln und für den Transport der kommissionierten Artikel verwendet werden. Zum Beispiel Paletten und Tablar.

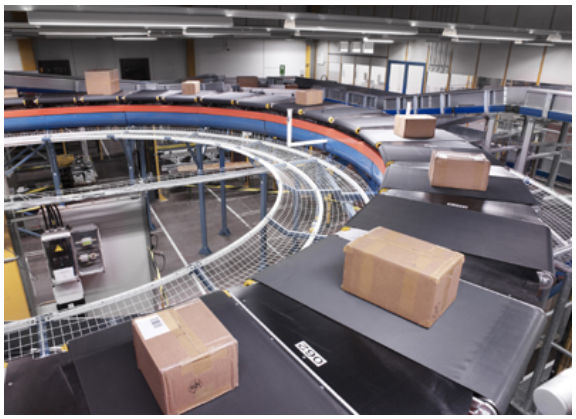


Abbildung 3.4: Quergurtsorter (Quelle: http://www.siemens.com/press/pool/de/pressebilder/2009/mobility/072dpi/imo200901009-01_072dpi.jpg)

3.1.2 Kommissionierleitsysteme

Wenn Informationen über Bestellungen, wie bspw. Menge und Art eines Artikels sowie Ort der Lagerstätte, ermittelt und aufbereitet wurden, werden diese an den Kommissionierer weitergeleitet, damit dieser die aufgestellten Posten einsammeln kann. Optional sind auch Wege, Fehlmengen oder weitere Kundenangaben möglich. Um den Mitarbeiter zu leiten

Abbildung 3.5: Verfahren der Kommissioniererführung (Quelle: [19])

existieren unterschiedliche Ansätze, die so genannten Kommissionierleitsysteme. Abbildung 3.5 zeigt eine Übersicht über diese.

Im Allgemeinen wird zwischen papiergebundenen und beleglosen Verfahren unterschieden. Bei den papiergebundenen Verfahren werden dem Kommissionierer alle für einen Pickvorgang wichtigen Informationen in Papierform zur Verfügung gestellt. Beleglose Verfahren arbeiten i.d.R. komplett digital und erleichtern so häufig nachträgliche Änderungen. In diesem Kapitel wird ein Überblick über alle abgebildeten Verfahren gegeben und Vorteile sowie

Pickliste

Kunden-Nr. / Auftrags-Nr.

06.11.2008 12:04
Seite 1

Kunde: K-006
Auftragsnummer: 1
Auftragsdatum: 06.11.2008
Bemerkung: Tor 10

Regal Z	Säule X	Ebene Y	Packstück-ID	Artikelnummer / Bezeichnung	Menge	Ent.	Erl.
Lagerbereich AL-Ost							
001	001	003	0000000003	4712000000 Fensterheber-Motor	20	Voll	
001	001	005	0000000005	4050300004815 Glühlampe 30 W - E14	10	Teil	
001	002	002	0000000007	4711000000 Scheibenwischer	20	Teil	

Entnahmen insgesamt: 3

Abbildung 3.6: Beispielhafte Pickliste (Quelle: http://www.idr-online.de/Newstour_Light_200/tour_06.htm)

Nachteile von diesen aufgezeigt. Die Reihenfolge sowie der Inhalt richten sich dabei nach [19].

Pickliste

Die Pickliste ist eine der einfachsten Formen der Kommissioniererführung. Es werden Listen mit anstehenden Picks erstellt und ausgedruckt, wie in Abbildung 3.6 abgebildet. Diese Ausdrucke werden anschließend den Kommissionierern zur Verfügung gestellt, damit diese die entsprechenden Artikel einsammeln können. Dieses Verfahren der Kommissioniererführung ist prinzipiell für sämtliche Formen von Kommissionierleitsystemen geeignet. Als positiver Nebeneffekt ergibt sich neben der besonders einfachen und kostengünstigen Umsetzung die Möglichkeit Nebentätigkeiten auszuführen. So kann der Kommissionierer auch Preisschilder anbringen oder Bestände kontrollieren. Um die Effektivität hoch zu halten und Fehler bei den Pickvorgängen zu vermeiden, sollte großer Wert auf die Gestaltung der Picklisten gelegt werden. Ist diese zu unübersichtlich gestaltet vergrößert sich der Totzeitanteil, da durch das notwendige Identifizieren von anstehenden Picks Zeit verloren geht, in der keine sinnvolle Arbeit erledigt werden kann. Weiter weisen Picklisten eine geringe Flexibilität bzgl. Änderungen auf. Sind die Listen erst einmal erstellt und gedruckt können diese nur noch aufwendig geändert und dem Kommissionierer neu ausgestellt werden. Handelt es sich um ein (teil-)automatisiertes System, so erweist sich eine Pickliste auch als nachteilig, da dort im System bereits alle Informationen für eine vollständig digitale Handhabung vorhanden sind (vgl. [19]).

Vorteile:

- Einfache Ausfertigung
- Kostengünstig

Abbildung 3.7: Stationäres Terminal (Quelle: [19])

- Möglichkeit von Nebentätigkeiten des Kommissionierers

Nachteile:

- Geringe Flexibilität bei nachträglichen Änderungen
- Hoher Totzeitanteil

Stationäre Terminals

Terminals gehören zur beleglosen Kommissioniererführung. Sie umfassen ein Anzeigemedium (z. B. Bildschirm) und ein Eingabemedium (z. B. Scanner oder Tastatur), über welche der Kommissionierer interaktiv mit dem Datenverarbeitungssystem arbeiten kann. Unterschieden wird dabei zwischen stationären Terminals, die an einem festen Standort stehen, und mobilen Terminals.

Stationäre Terminals, wie eines in Abbildung 3.7 gezeigt wird, werden meist in zentralen Kommissionierstationen eingesetzt (z. B. Ware-zur-Person Kommissionierung). Dabei werden dem Kommissionierer auf einem fest installierten Monitor Entnahmeinformationen angezeigt. Diese werden bspw. direkt einem Warehousemanagementsystem (WMS) oder einem Enterprise-Ressource-Planning-System (ERP) entnommen. Auf diese Weise sind auch Änderungen, z. B. geänderte Mengen einer zu pickenden Einheit, einfach umsetzbar. Außerdem kann der Mitarbeiter wegen der Handfreiheit effektiv arbeiten. Aufgrund der Technik sind stationäre Terminals i.d.R. relativ teuer und unflexibel bzgl. Ortsänderungen (vgl. [19]). Sie sind daher vor allem auf die Ware-zur-Person Kommissionierung beschränkt.

Abbildung 3.8: Pick-by-light Einheit, welche eine Entnahmestelle markieren und die zu entnehmende Menge Anzeigen kann (Quelle: http://commons.wikimedia.org/wiki/File:Pick-by-Light-Fachanzeigen_PickTermFlexible.JPG)

Vorteile:

- Flexibel bzgl. Änderungen
- Handfreiheit

Nachteile:

- Preis
- Unflexibel bzgl. Ortsänderungen

Pick-by-light

Pick-by-light Systeme zeichnen sich durch Anzeigefelder aus, die ober- bzw. unterhalb von Entnahmestellen angebracht sind. In den meisten Fällen handelt es sich dabei, wie in Abbildung 3.8 dargestellt, um Displays, die die zu entnehmende Menge anzeigen. Ist statt eines Displays nur eine Lampe angebracht, so signalisiert diese durch Aufleuchten die Stelle, an der eine Ware entnommen werden soll. Neben der Anzeige der Entnahmestelle bieten einige Systeme auch noch durch Knöpfe die Möglichkeit einer Korrektur oder Quittierung der Entnahme.

Anwendung finden Pick-by-light Systeme vor allem in Bereichen, in denen es um kleinvolumige Artikel geht, meist in Kombination mit einer geringen Sortimentsbreite, also einer geringen Anzahl an verschiedenen Warengruppen, sowie kurzen Laufwegen. Da Pickaufträge einem Kommissionierer zugeordnet werden müssen, beschränkt sich Pick-by-light meist auf einen Auftrag pro Zone zu einer Zeit. Durch den Einsatz mehrfarbiger Anzeigen, bei

welchen den Kommissionierern unterschiedliche Farben zugeordnet werden, können auch mehrere Kommissionierer innerhalb einer Zone agieren. Da die Entnahmestellen mit dem übergeordneten Leitsystem verbunden sind, können Änderungen auf einfache Weise vorgenommen werden. Weiter ist die direkte Bestätigung und Erfassung des Picks, die Möglichkeit der parallelen Bearbeitung sowie die Reduzierung von Totzeiten durch die Displays von Vorteil. Als großer Nachteil gilt bei diesem System der hohe Anschaffungspreis und Organisationsaufwand, da jede Entnahmestelle mit entsprechenden Displays und Knöpfen ausgestattet und mit einem übergeordneten Leitsystem verbunden werden muss (vgl. [19]).

Vorteile:

- Direkte Erfassung der Picks
- Parallele Bearbeitung
- Reduzierung von Totzeiten

Nachteile:

- Hoher Anschaffungspreis
- Hoher Organisationsaufwand
- Beschränkt auf einen Auftrag pro Zone zu einer Zeit (bei Einsatz von nur einer Farbe)

Put-to-light

Put-to-light funktioniert ähnlich wie Pick-by-light, mit dem Unterschied, dass statt des Entnahmeortes der Abgabeort gekennzeichnet wird (siehe Abbildung 3.9). Dazu scannt der Kommissionierer nacheinander die einzusortierenden Artikel und eine Lampe am jeweiligen Ablagefach, in welches dieser gelegt werden soll, leuchtet auf. Dieses Verfahren wird meist in Kombination mit anderen Verfahren eingesetzt, z. B. um eine auftragsparallele Kommissionierung umzusetzen. So werden mithilfe von Pick-by-light bspw. von einem Picker mehrere Kommissionieraufträge parallel erledigt und diese Picks anschließend unter Zuhilfenahme von Put-to-light wieder in getrennte Aufträge aufgeteilt (vgl. [19]).

Mobile Terminals

Mobile Terminals bestehen meist aus einer Tastatur und einem Anzeigemedium (z. B. LCD Display). Siehe dazu auch Abbildung 3.10. Häufig finden sich auch Zusatzfunktionen wie Labeldrucker oder Barcodescanner wieder. Teilweise sind die Terminals auch in Geräte integriert, die der Identifizierung von Gütern dienen (wie z. B. einer Waage). Bei mobilen Terminals ist grundsätzlich zwischen Online- und Offline-Geräten zu unterscheiden, je nachdem ob die Geräte live mit einem Hintergrundsystem zusammenarbeiten oder nicht (vgl. [19]).

Abbildung 3.9: Put-to-light-System, bei dem die Aufträge in Schächte mit Lämpchen kommissioniert werden (Quelle: http://www.fml.mw.tum.de/fml/index.php?Set_ID=320&letter=P&b_id=3946337B-4431-4430-382D-463133412D34)

Abbildung 3.10: Mobile Terminals im Einsatz (Quelle: http://www.pfb.de/auto-id/mobile-terminals/)

Offline-Geräte Offline-Geräte besitzen im Einsatz keine direkte Verbindung zu einem Leitsystem. Um die Kommissionieraufträge auf diese Geräte zu übertragen, müssen diese regelmäßig z. B. mithilfe einer Dockingstation synchronisiert werden. Einmal synchronisiert kann der Kommissionierer mit dem Gerät die Kommissionierung beginnen (vgl. [19]).

Vorteile:

- Mobilität
- Möglichkeit von Zusatzfunktionen (z. B. Labeldrucker)

Nachteile:

- Geringe Flexibilität bzgl. Änderungen (nur zum Zeitpunkt von Synchronisationen möglich)

Online-Geräte Online-Geräte zeichnen sich dadurch aus, dass sie per Funk mit einem Leitsystem verbunden sind. In der Vergangenheit wurde zu diesem Zweck häufig Infrarot eingesetzt. Diese Technik weist jedoch einige Nachteile auf. So sorgt eine keulenförmige Ausbreitung z. B. dafür, dass Infrarotstrahlen gerichtet ausgesendet werden müssen. Dies erschwert den mobilen Einsatz sehr. Weiter sind bidirektionale Übertragungen schwierig und die Skalierbarkeit schlecht (vgl. [19]).

Aus diesen Gründen sind Infrarotverbindungen heutzutage größtenteils durch Funkverbindungen ersetzt worden (vgl. [19]). Meist beruhen diese Verbindungen auf Wireless Local Area Networks (WLAN), über welche das Gerät dann mit einem Server kommuniziert. Eine kugelförmige Ausbreitung der Signale sorgt dafür, dass der Kommissionierer mehr Freiheiten bei der Kommissionierung hat. Größere Datenraten bei der Übertragung sowie eine einfache bidirektionale Übertragung ermöglichen außerdem weitreichende Anwendungsmöglichkeiten bis hin zu Lösungen, bei denen der Kommissionierer mit dem System kommunizieren kann (Pick-by-voice) (vgl. [19]).

Vorteile:

- Mobilität
- Möglichkeit von Zusatzfunktionen (z. B. Labeldrucker)
- Hohe Flexibilität dank live Kommunikation mit Server

Nachteile:

- Von funktionierendem Netzwerk abhängig

Pick-by-voice

Pick-by-voice Systeme bestehen meist aus einem mobilen Terminal und einem zusätzlichen Akku, der am Gürtel des Kommissionierers befestigt ist. Dieser sorgt dafür, dass der Kommissionierer ausreichend lange mit dem System arbeiten kann. Neben dem mobilen Terminal trägt der Kommissionierer ein Headset, über welches dieser mittels Mikro mit einem Computer kommunizieren kann. So erhält der Kommissionierer über das Headset

Abbildung 3.11: Pick-by-voice im Einsatz (Quelle: http://www.profil-marketing.com/?id=42&tx_lipresscenter_pi1[news]=895&tx_lipresscenter_pi1[folder]=76)

die notwendigen Informationen für einen Pickvorgang. Über eine Spracheingabe kann er diesen auch gleich, z. B. mittels einer Prüfziffer, die bei Erfolg nachgesprochen werden soll, bestätigen. Ein Vorteil dieses Systems liegt in der Handfreiheit, da alle Anweisungen über das Headset ausgetauscht werden können.

Ein wesentlicher Unterschied der Systeme liegt in der Sprecherabhängigkeit/Sprecherunabhängigkeit eines Systems. Sprecherabhängige Systeme sind speziell auf einen Kommissionierer abgestimmt und funktionieren nur mit diesem. Eingerichtet werden die Geräte meist durch das Vorlesen eines standardisierten Textes. Auf dieser Grundlage wird dann ein sprecherabhängiges Profil erstellt. Sprecherunabhängige Systeme hingegen haben eine Vielzahl an Sprachmustern hinterlegt und arbeiten unabhängig vom jeweiligen Kommissionierer. Zum Einsatz kommen in der Praxis auch häufig Mischsysteme, die durch Training bspw. an einen Dialekt gewöhnt werden können (vgl. [19]).

Im Umgang mit sprachbasierten Systemen arbeitet ein Kommissionierer effektiver, wenn er in seiner Muttersprache arbeiten kann. Bei der Anschaffung eines Pick-by-voice Systemes sollte daher auch darauf geachtet werden, dass das System mehrere Sprachen anbietet und so dem Kommissionierer bei der Benutzung entgegen kommt. Je nach Funktionsumfang des Systemes ist ein Wortumfang von 100 Wörtern ausreichend. Bei zusätzlichen Funktionen kann der Bedarf an Wörtern auf bis zu 1000 ansteigen. Der Wortumfang steht auch in einem engen Zusammenhang mit der Dialogtiefe eines Systems. Wird ein Kommissionierer bspw. noch in ein System eingearbeitet, so kann es notwendig sein, dass dieser durch möglichst vollständige Sätze umfangreich angeleitet wird. Mit zunehmender Erfahrung des Kommissionierers können die Anweisungen jedoch immer weiter reduziert werden, sodass am Ende lediglich wenige Worte für eine erfolgreiche Anweisung ausreichen. Da es in Lagerhallen häufig zu vielen Störgeräuschen kommt, z. B. Lärm, sollte außerdem ein Filter für Nebengeräusche vorhanden sein. Diese können sowohl hardwareseitig auf Sei-

Abbildung 3.12: Beipsielhafter Einsatz eines Augmented Reality (AR) Systemes zu Anzeige eines gesuchtern Lagerplatzes (Quelle: http://www.salt-solutions.de/blog/index.php/die-neun-vorteile-von-google-glass-der-logistik/)

ten des Mikrofons, aber auch softwareseitig umgesetzt werden (vgl. [19]).

Vorteile:

- Kommissionierer hat Hände frei

Nachteile:

- Konzentration auf Gesprochenes
- Einarbeitung bei Sprecherabhängigkeit

Forschung: Augmented Reality

Im Bereich der Kommissionierung wird laufend an neuen Ansätzen geforscht, wie die Arbeit der Kommissionierer weiter vereinfacht und der Arbeitsablauf optimiert werden kann. Augmented Reality stellt einen relativ neuen Ansatz in der Forschung bzgl. der Kommissioniererführung dar und wird durch Azuma et al. [11] folgendermaßen beschrieben:

> Augmented Reality (AR) bedeutet wörtlich erweiterte Realität. In einem AR-System wird die reale Welt durch im Computer generierte Objekte ergänzt, die scheinbar im selben Raum wie die reale Welt bestehen.

Auf die Festlegung auf ein konkretes Medium wie bspw. Bildschirme wird verzichtet, da AR prinzipiell auch auf andere Sinne angewendet werden kann. AR ist ein Teilgebiet der vermischten Realitäten und reichert die Realität um virtuelle Objekte an. Sie ist nicht zu verwechseln mit der Augmented Virtuality (AV) oder der Virtual Reality (VR), bei welchen der virtuelle Aspekt überwiegt.

Abbildung 3.13: Tracking mithilfe von Marken in AR Systemen (Quelle: http://www.salt-solutions.de/blog/index.php/pick-by-vision-zum-anfassen/)

Tracking Unter dem Einsatz von AR müssen virtuelle und reale Objekte überlagert werden. Die Berechnung der genauen Position von virtuellen Objekten innerhalb eines realen Umfeldes des Kommissionierers wird auch Tracking genannt. Zu diesem Zweck existieren zwei Verfahren:

- **Markerbasiertes Tracking**: Beim markerbasierten Tracking werden mithilfe einer Kamera Marker aufgezeichnet, die sich innerhalb eines Raumes befinden. Mithilfe einer Software können diese dann aus der Umgebung extrahiert und virtuelle Objekte entsprechend platziert werden. Bewegt sich der Kommissionierer nun im Raum, so bewegen sich auch die virtuellen Objekte passend zu den Bewegungen der Marker. Die Videoaufnahmen werden häufig im Rahmen eines hybriden Trackings durch weitere Verfahren, z. B. akustische (Ultraschall), ergänzt.
- **Markerloses Tracking**: Beim markerlosen Tracking wird versucht ohne künstlich in die Umwelt eingebrachte Marker zu arbeiten. Zum Einsatz kommen hier neben der Analyse des Bildes auch Verfahren zur Positionsbestimmung, wie bspw. GPS und Kompass.

Schnittstellen Schnittstellen stellen die Medien dar, mithilfe welcher der Kommissionierer mit dem System in Berührung kommt. So gehören zu dieser Klasse Anzeigemedien, Eingabemedien sowie Geräte zur Positionsbestimmung. Anzeigegeräte können dabei in vier Kategorien eingeteilt werden: Head-Mounted-Displays (HMD), handgeführte Geräte, projizierende Displays sowie fest montierte Displays.

Head-Mounted-Displays bieten dem Kommissionierer eine Videodarstellung oder eine optische Darstellung. Bei Ersterem versperrt ein Bildschirm die vollständige Sicht

Abbildung 3.14: Head-Mounted-Display (HMD) mit einer Videodarstellung (Quelle: http://sensics.com/news-events/press-kit-and-product-photos/)

Abbildung 3.15: HMD mit einer optischen Darstellung (Quelle: http://www.technologiepark-bremen.de/de/techpark-news-detail?sv[id]=297109)

des Kommissionierers (siehe Abbildung 3.14). Die Realität wird dabei angereichert um die künstlichen Elemente auf dem Display dargestellt. Bei der optischen Darstellung (siehe Abbildung 3.15) wird das Sichtfeld des Kommissionierers nicht durch ein Display versperrt. Vielmehr werden die künstlichen Objekte auf das Auge des Kommissionieres projiziert.

Handgeführte Geräte sind Geräte mit kleinen Displays, meist handelsübliche PDAs mit integrierter Kamera (siehe Abbildung 3.16). Der Kommissionierer kann diese Geräte dann wie eine Art Fenster verwenden, durch das er hindurch sehen und die Umwelt um weitere Informationen angereichert sehen kann.

Abbildung 3.16: PDA, welches die Umwelt um Information anreichert (Quelle: http://commons.wikimedia.org/wiki/File:Wikitude3.jpg)

(a) Regler (b) Projektor

Abbildung 3.17: Von einem Projektor projizierte Regler sowie der dazugehörige Projektor (Quelle: [29])

Wenn zusätzliche Informationen direkt auf Gegenstände projiziert werden, spricht man von **projizierenden Displays** (siehe Abbildung 3.17). Projiziert werden die zusätzlichen Informationen dabei von einem fest installierten Projektor oder von einem mobilen Gerät, welches vom Kommissionierer mitgeführt wird.

Auch zum Einsatz kommen (teil-)durchsichtige **fest montierte Displays**, die über einem potenziellen Einsatzgebiet angebracht sind. Durch die Anzeige von Informationen holen diese dann alle notwendigen Informationen in das Blickfeld des Kommissionierers.

Der Einsatz von AR Systemen könnte ähnliche bis weitreichendere Möglichkeiten wie Pick-by-light bieten, welches sehr kostenintensiv ist. Die grafische Unterstützung könnte außerdem um weitere Elemente erweitert werden, wie bspw. einer akustischen Kommis-

sioniererführung. Da es sich um ein aktives Forschungsfeld handelt bestehen noch einige Probleme bei AR Systemen, welche nachfolgend aufgelistet sind (Entnommen aus [19]):

- **Performanz**: das Nachführen der Bilder bei Bewegungen
- **Energieversorgung**: Bei mobilen Systemen verfügen die Akkus nicht über ausreichende Energiereserven
- **Sensoren**: Rauschen bei Bewegungen, Drift, Abschaltung des Trackingsystems (z. B. bei GPS)
- **Daten**: Verfügbarkeit und hohe Komplexität der Daten
- **Visualisierung**: um die Einbettung der virtuellen Szenen in die realen möglichst überzeugend zu leisten
- **Benutzerschnittstellen**: Insbesondere bei mobilen Anwendungen z. B. im Außenbereich ist die Eingabe von Informationen durch Tastatur und Menüsteuerung durch eine Maus umständlich
- **Ergonomie**: Die Systeme sind in der Regel noch zu schwer, um sie dauerhaft am Körper zu tragen

Auch wenn noch Probleme bestehen, so bietet das Feld der AR viel Potenzial, um aktiv in diese Richtung zu forschen.

Systemansatz der Projektgruppe

Die Projektgruppe hat als Ziel die Entwicklung eines neuen Kommissionierleitsystemes *Camera-assisted Pick-by-feel*, welches nicht vollständig in die bestehenden Kategorien eingeordnet werden kann. Der genaue Aufbau des Systems wird in Kapitel 1 und Abschnitt 8.1 besprochen. Das zu entwickelnde System ist dabei für einen mobilen Einsatz vorgesehen, jedoch ohne den Einsatz von mobilen Terminals in der derzeitigen Form. Zur Anwendung sollen erstmals im normalen Handel erhältliche Produkte sein, welche in einem Verbund miteinander funktionieren. Das System setzt demnach nicht auf den Einsatz eines einzelnen speziell entwickelten Gerätes, sondern kombiniert drei unterschiedliche im normalen Handel erhältliche Geräte zum Zwecke der Kommissionierung. Auf diese Weise können Kosten für die Beschaffung von Geräten minimiert und der Austausch dieser erleichtert werden. Als Vorteil ergibt sich, dass der Kommissionierer durch den Einsatz einer Smartwatch die Hände frei hat und aufgrund der automatischen Erkennung der Produkte weniger Arbeit mit der Bestätigung von Picks hat. Weiter kann er die anstehenden Picks einfacher identifizieren als in gängigen mobilen Terminals, da er dazu nur auf seine am Arm befestige Smartwatch schauen muss. Die Smartwatch ersetzt so gängige Lösungen wie mobile Terminals und bietet dem Kommissionierer eine einfache Möglichkeit der Identifizierung neuer

Picks, der erfolgreichen Kommissionierung dieser und das bei einem minimalen Gewicht und Aufwand.

Vorteile:

- Kommissionierer hat Hände frei
- Geringes Gewicht durch eingesetzte Komponenten (insbesondere die Smartwatch belastet den Arm nicht)
- Zeitersparnisse durch ausbleibenden manuellen Scanvorgang eines Barcodes
- Schnelle Einarbeitung neuer Mitarbeiter
- Austausch nur einzelner Komponenten möglich

Nachteile:

- Ausfall einer Komponente führt zu partiellem oder totalem Ausfall des Systems
- Manuelle Eingriffe können vonnöten sein

3.2 Bildverarbeitung

Dieses Kapitel beschäftigt sich mit der Vorverarbeitung von Bildern. Dabei wird in Kapitel 3.2.1 zunächst beschrieben, wie Bilder repräsentiert werden, welche Variationen auftreten und wie diese behoben werden können, um die weitere Verarbeitung zu vereinfachen. Einfache Verfahren zur Vorverarbeitung werden in Kapitel 3.2.2 erläutert. Kapitel 3.2.3 befasst sich mit dem Finden von Kanten in einem Bild. In Kapitel 3.2.4 werden Verfahren zur Merkmalsextraktion beschrieben.

3.2.1 Repräsentation von Bildern

Bilder müssen für die weitere Verarbeitung zunächst digitalisiert und gespeichert werden. Im Folgenden wird beschrieben, wie die Bilder repräsentiert werden und welche Variation in den Bildern auftreten können.

Bilder werden als eine $N \times M$ Matrix dargestellt. Der Wert (x, y) repräsentiert einen bestimmten Pixel. Dabei kann die Matrix z. B. Intensitätswerte (Grauwertbild) oder Werte einens Farbraums beinhalten (Farbbild). Grauwertbilder werden als eindimensional bezeichnet und enthalten Werte von 0 (schwarz) bis 255 (weiß). Farbbilder sind mehrdimensional und bauen auf einem Farbmodell, z. B. RGB (Rot, Grün, Blau), auf.

Variationen

Bilder können verschiedene Variationen aufweisen, welche die weitere Verarbeitung erschweren. Durch zu kurze oder zu lange Belichtungszeiten kann das Bild unscharf werden. Dieser Effekt tritt z. B. auf, wenn sich schnell bewegende Objekte mit einer zu hohen Belichtungszeit aufgenommen werden. d. h., während der Aufnahme ändern die Objekte ihre Position und wirken dadurch unscharf. Des Weiteren kann ein Bild schlechte Kontraste aufweisen. Durch ungleichmäßige Beleuchtung des Bildes entsteht das so genannte Bildrauschen.

3.2.2 Normalisierung & Filter

Bilder, insbesondere von natürlichen Szenen, enthalten oft die zuvor beschriebenen Variationen. Für die weitere Verarbeitung, z. B. das Erkennen von Objekten auf Bildern, ist es häufig von Vorteil, diese zu entfernen. Welche Möglichkeiten es dafür gibt, wird im Folgenden näher erläutert.

Histogrammnormalisierung

Histogramme sind Häufigkeitsverteilungen, siehe Abbildung 3.18. Auf Bilder angewandt, beschreiben sie, wie oft jeder Intensitätswert im Bild vorkommt. Wird z. B. die Helligkeit eines Bildes erhöht, verschiebt sich das Histogramm nach rechts, das heißt, die vorkommenden Intensitätswerte steigen an. Durch eine Kontrasterhöhung wird die Zusammensetzung der Intensitätswerte eines Bilds auf einen größeren Wertebereich verteilt, wobei der Bereich immer noch auf Werte zwischen 0 und 255 beschränkt ist.

Bei der Histogrammnormalisierung wird versucht, den Kontrast eines Graustufenbildes zu erhöhen. Dabei werden die Grauwerte eines Bildes so verändert, dass der gesamte verfügbare Wertebereich abgedeckt wird. Da sich die Veränderung immer nur auf einen Pixel bezieht und nicht auf seine Nachbarn, wird dies als Punktoperation bezeichnet. Für die Normalisierung wird der dunkelste Pixel auf den niedrigsten Intensitätswert abgebildet und der hellste auf den höchsten. Alle dazwischen liegenden Pixel werden linear verteilt (siehe Abbildung 3.19).

Nachteil dieses Verfahrens ist die geringe Robustheit. Einzelne Ausreißer beeinflussen das Ergebnis stark. Daher wird in der Regel ein bestimmter Prozentsatz der Pixel am oberen und unteren Rand des Histogramms auf die Extremwerte abgebildet.

Filter

Filter sind lokale Bildoperationen. Im Gegensatz zu Punktoperationen werden hier auch die Nachbarpixel betrachtet. Dabei können verschiedene Fenstergrößen, also unterschiedlich viele Nachbarpixel, betrachtet werden. Sie werden z. B. verwendet um Merkmale hervor-

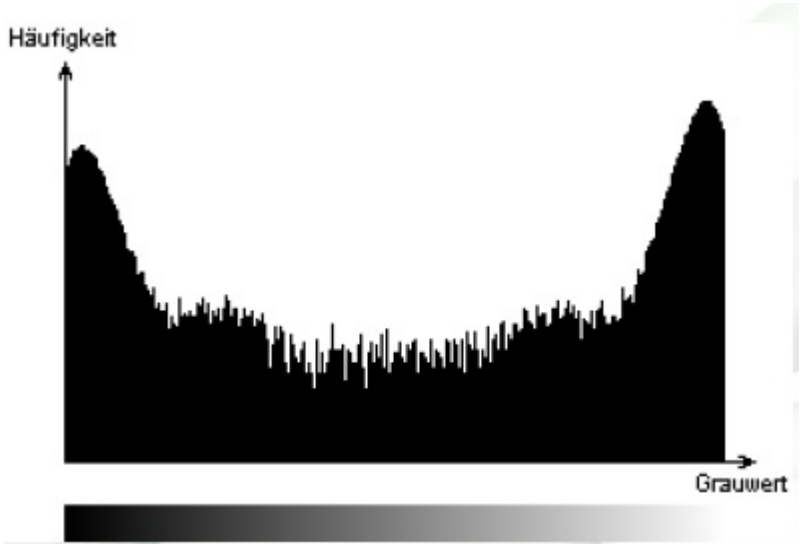


Abbildung 3.18: Beispiel für ein Grauwerthistogramm (Quelle: http://upload.wikimedia.org/wikipedia/commons/2/2b/Rauschbild-Histogramm.png)

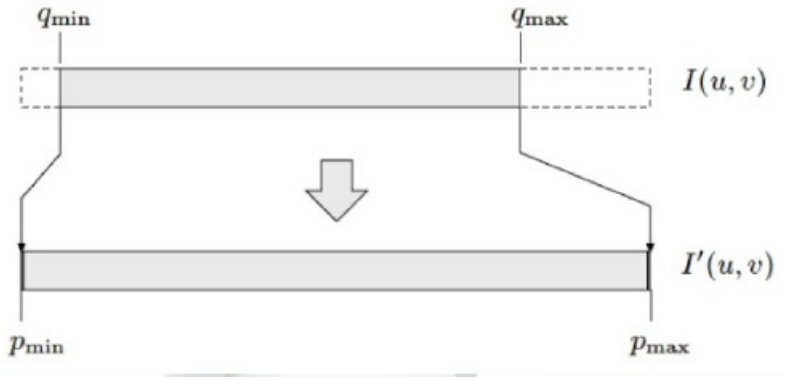


Abbildung 3.19: Histogrammnormalisierung, verwendete Grauwerte werden auf den gesamten Wertebereich abgebildet (Quelle: http://www.cg.tuwien.ac.at/courses/EinfVisComp/Skriptum/SS13/EVC-11%20Point%20Operations.pdf)

zuheben oder um die Bildqualität zu verbessern. Dabei wird zwischen linearen und nicht linearen Filtern unterschieden.

Lineare Filter Bei den linearen Filtern wird eine Filtermatrix auf jedes Pixel angewandt und ein neuer Wert für ihn berechnet. Dazu gehört z. B. der Mittelwertfilter, er berechnet den Mittelwert der umliegenden Pixel und ersetzt den aktuellen Wert damit. Dadurch wird das Bildrauschen geglättet. Allerdings werden auch Kanten weicher. Um Kanten zu verstärken kann u. a. der Sobel Filter angewandt werden [35].

Nicht lineare Filter Bei den nicht linearen Filtern werden die neuen Werte nicht berechnet, sondern anhand sortierter Integer-Wert Listen aus der Nachbarschaft gewählt. Daher

ist der Aufwand häufig höher als bei linearen Filtern. Beim Minimumfilter, auch Erosion genannt, wird bspw. der jeweils kleinste Wert aus einer Bildmaske ausgewählt. Dadurch werden schwarze Flächen größer bzw. weiße Flächen kleiner. Bei einem Maximumfilter (Dilatation) ist dies genau andersherum. Beide Verfahren können auch kombiniert werden. Beim so genannten Closing wird zunächst die Dilatation und dann die Erosion angewandt. Dadurch werden kleine Löcher zunächst geschlossen und anschließend die ursprüngliche Größe des Objektes wieder hergestellt (siehe Abbildung 3.20). Beim Opening hingegen wird zunächst die Erosion und anschließend die Dilatation verwendet. Dadurch werden im ersten Schritt kleine Teilstrukturen entfernt und anschließend die ursprüngliche Größe wieder hergestellt.

Abbildung 3.20: Closing: zunächst Dilatation dann Erosion (Quelle: http://kurse.fh-regensburg.de/cato/module/bildverarbeitung/bv/5_4filter/5_4filter.pdf)

3.2.3 Kanten

Als *Kante* wird der Bereich eines Bildes bezeichnet, in dem sich die Intensität auf kleinem Raum und entlang einer Richtung stark ändert. Für den Menschen reichen einige markante Kanten häufig aus, um ein Objekt zu identifizieren. Kanten sagen also offensichtlich viel über ein Objekt aus (vgl. [23]).

Um Kanten zu finden, wird die Intensität entlang einer Zeile/Spalte betrachtet. Bei starker Änderung liegt vermutlich eine Kante vor. Um diese Veränderung zu erkennen, kann die erste Ableitung verwendet werden. Die Kanten befinden sich dann an den Min- bzw. Maxima. Die Funktion für eine Zeile u wird als f(u) bezeichnet. Das Problem dabei ist, dass f(u) diskret und die Ableitung nicht definiert ist. Allerdings kann die erste Ableitung mit Hilfe der Nachbarpixel wie folgt approximiert werden:

$$\frac{df}{du} \approx \frac{f(u-1) + f(u+1)}{2}$$

Abbildung 3.21 veranschaulicht die Approximation über die Nachbarpixel. Um die erste Ableitung für jeden Bildpunkt zu berechnen, kann das Bild mit den Gradienten H_x zeilen- bzw. H_y spaltenweise durchlaufen werden. Entsprechend der obigen Formel werden diese wie folgt definiert:

$$H_x = \begin{bmatrix} 0.5 & 0 & 0.5 \end{bmatrix} = 0.5 * \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$$

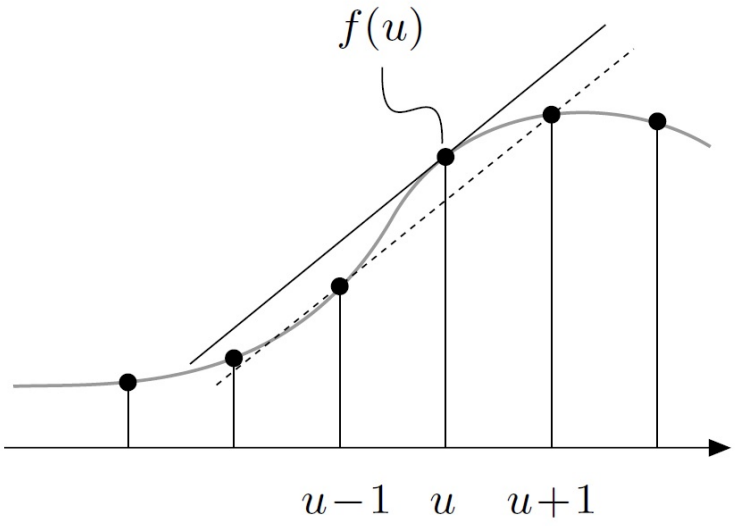


Abbildung 3.21: Approximation der ersten Ableitung mittels Nachbarpixel, (Quelle: [23])

und

$$H_y = \begin{bmatrix} 0.5 \\ 0 \\ 0.5 \end{bmatrix} = 0.5 * \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

Allerdings ist diese Berechnung fehleranfällig, da einzelne fehlerhafte Pixel das Ergebnis stark beeinflussen. Daher werden in der Regel mehrere Zeilen/Spalten betrachtet wie z. B. beim Prewitt Operator

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Oder beim Sobel Operator

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Wobei der Sobel Operator die aktuelle Zeile/Spalte stärker gewichtet. Die Gradienten H_y sind analog (vgl. [23]).

Wird für jeden Pixel eines Bildes mit Hilfe der Gradienten die erste Ableitung in x- bzw. y-Richtung berechnet, so entstehen zwei neue Bilder D_x und D_y. Diese beiden Bilder können zu einem Kantenbild zusammengesetzt werden. Die Stärke der Kante ist dabei der Betrag der Gradienten. Für den Pixel in Zeile u und Spalte v wird die Kantenstärke wie folgt berechnet:

$$E(u,v) = \sqrt{(D_x(u,v))^2 + (D_y(u,v))^2}$$

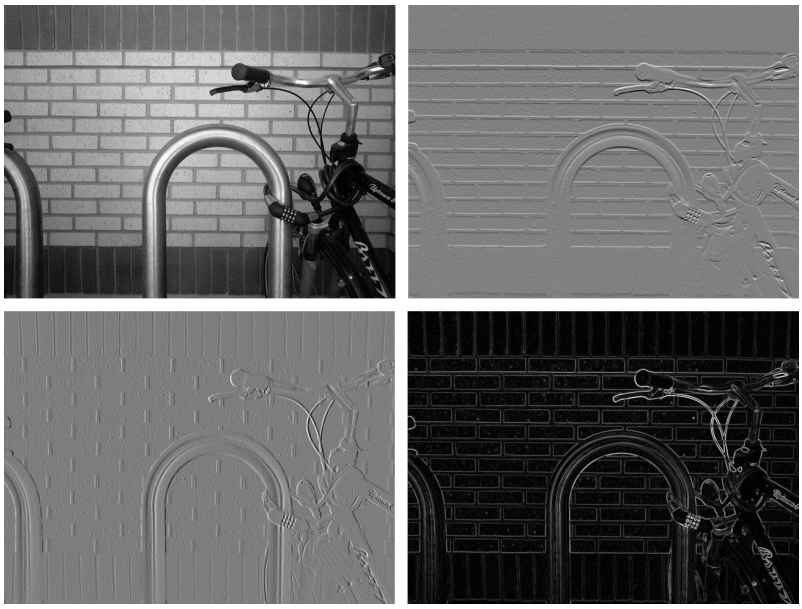


Abbildung 3.22: Kantendetektion mit Sobel Operator. Links oben: original, rechts oben: in y-Richtung, links unten: in x-Richtung, rechts unten: Kantenstärke (Quelle: http://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=620521668)

3.2.4 Merkmale

Merkmale sind mathematische Beschreibungen von Signalen oder Daten, in diesem Kontext von Bildern oder Bildteilen. Dabei sind sie Repräsentationen, die leichter zu unterscheiden sind als reine Pixelwerte, und werden genutzt, um relevante Informationen für bestimmte Aufgaben zu extrahieren. Diese Aufgaben beinhalten unter anderem Objekterkennung, *Robotic Mapping and Navigation*, *Image Stitching/Alignment*, Gestenerkennung und *Video Tracking*. Merkmale können in lokale und globale Merkmale aufgeteilt werden.

Globale Merkmale beschreiben ein Bild in seiner Gesamtheit. So kann z. B. ein ganzes Bild mit nur einem Vektor generalisiert werden. Beispiele für globale Merkmale sind Grauwerthistogramme, Konturrepräsentationen, Formbeschreibungen und Texturmerkmale. Globale Merkmale sind empfindlich gegenüber *Occlusion* und *Clutter* und im Allgemeinen nicht geeignet, um sie in komplexen Anwendungen zu verwenden. Occlusion bedeutet in diesem Fall die Überdeckung eines gesuchten Objekts durch andere Objekte. Clutter ist eine Anhäufung von Objekten in einem Bereich, die die Unterscheidung der Objekte unübersichtlich machen kann.

Lokale Merkmale beschreiben kleine Bildregionen. Sie sind eine mathematische Beschreibung einzigartiger, interessanter Bildbereiche, den *Interest Points*, und ihrer Nachbarschaft. Diese Interest Points werden bei der Merkmalsdetektion bestimmt. Dabei werden Bildbereiche mit hohem visuellen Informationsgehalt und möglichst eindeutig bestimmbarer räumlicher Anordnung identifiziert. Anschließend wird ein Merkmalsvektor zu jedem Interest Point berechnet, der die lokale Bildstruktur beschreibt. Die Menge aller möglichen

Merkmalsvektoren bildet einen Merkmalsraum. Im folgenden Abschnitt wird SIFT als ein Beispiel für lokale Merkmale vorgestellt. Ein Beispiel für globale Merkmale wäre der Gist einer Szene. Darauf basiert das Spatial Envelope model (Quelle: [27]), das Szenen als ganzes räumlich beschreibt.

SIFT

SIFT ([21]) steht für *Scale Invariant Feature Transform* und ist ein Algorithmus zur Berechnung lokaler Merkmale in Bildern. SIFT wurde für *Image Matching* entwickelt und bietet Invarianz gegenüber Skalen, Rotation und partielle Invarianz gegenüber Veränderungen der Beleuchtungen und des Blickpunkts. Als nächstes wird die Funktionsweise von SIFT erläutert, aufgeteilt in vier Schritte. Zunächst werden Extrempunkte des Bildes nach definierten Kriterien in einem Skalenraum bestimmt. Diese Extrempunkte werden danach auf ihre Brauchbarkeit überprüft. Ausgewählte Extrema kommen als *Keypoints* in Betracht. Anschließend wird zu jedem Keypoint eine oder mehrere Orientierungen berechnet. Darauf aufbauend werden die Keypoint-Deskriptoren erstellt, die als Merkmale, d. h. Beschreibungen, dienen.

Erfassung der Extrempunkte Zunächst wird das zu untersuchende Bild im Skalenraum dargestellt. Der Skalenraum des Bildes repräsentiert verschiedene Größenordnungen und Entfernungen des originalen Bildes. Die erste Oktave enthält zunächst das Originalbild I. Anschließend werden weitere Versionen L mit Hilfe eines Gauss-Unschärfe-Filters G erzeugt:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Dabei ist

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

So wird jedes folgende Bild mit konstantem Faktor k unschärfer. Jede weitere Oktave nimmt das vorletzte Bild der vorherigen Oktave mit halbierter Größe als erstes Bild. Das wird aus Effizienzgründen getan, da es schneller ist das Bild zu samplen, als es mit Hilfe von G zu berechnen. Diese Bilder werden ebenfalls mit Faktor k unschärfer gemacht. Nach der Berechnung der Oktaven werden die Extrempunkte des Skalenraums berechnet. Dazu werden zunächst in jeder Oktave *Difference-of-Gaussian* Bilder berechnet, wobei zwei aufeinanderfolgende Bilder der Oktave voneinander subtrahiert werden (siehe Abbildung 3.23). Anschließend werden die Extrempunkte dieser Difference-of-Gaussian Bilder ermittelt, indem jedes Pixel eines Difference-of-Gaussian Bildes mit seinen Nachbarpixeln verglichen wird, wie in Abbildung 3.24 zu sehen ist. Nachbarpixel sind sowohl die angrenzenden acht Pixel innerhalb des selben Bildes wie auch die neun Pixel des Bildes eine Skale höher und die neun Pixel des Bildes eine Skale niedriger. Nur Pixel, die größer als alle Nachbarn oder kleiner als alle Nachbarn sind, kommen als Näherungen der Extrema in Frage.

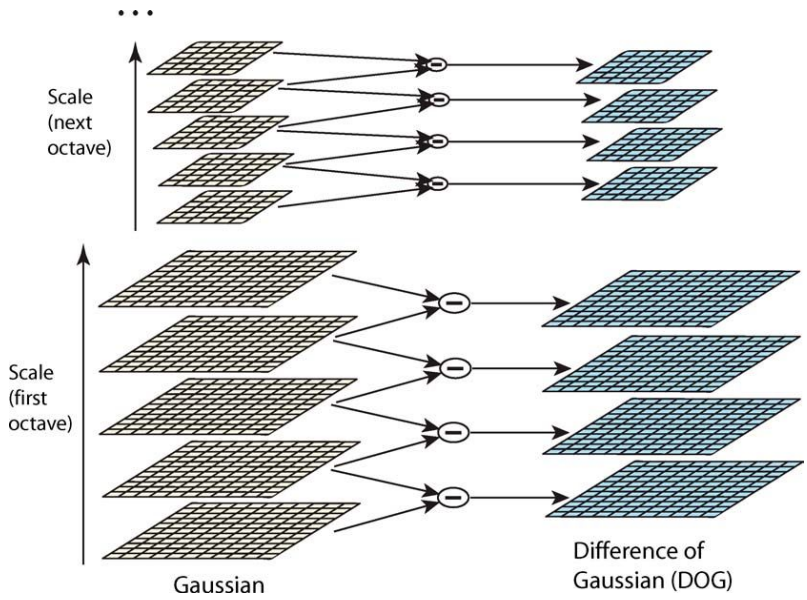


Abbildung 3.23: Berechnung der Difference-of-Gaussian Bilder (Quelle: [21])

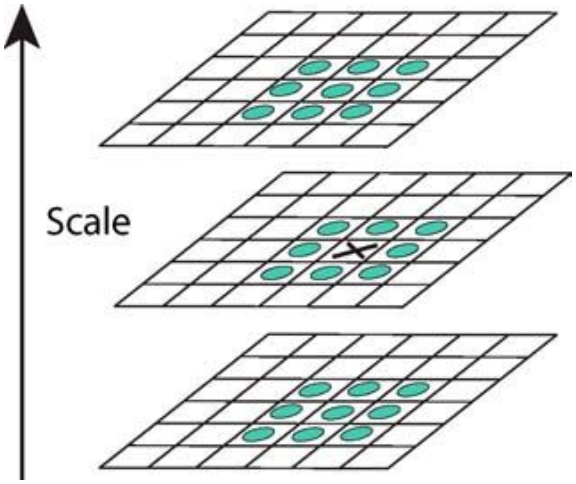


Abbildung 3.24: Das ausgewählte Pixel ist mit einem X markiert. Es wird mit den Nachbarpixeln verglichen, welche in grün gekennzeichnet sind (Quelle: [21])

Lokalisation von Keypoints Die tatsächlichen Subpixelwerte der Extrema werden nun mit Hilfe von Taylorreihenerweiterung bestimmt. Ein Beispiel für ein Extremum, das zwischen den Pixeln liegt, ist in Abbildung 3.25 zu sehen. Als nächster Schritt werden Extrema mit niedrigem Kontrast aussortiert. Dafür wird die Intensität jedes berechnetem Subpixel bestimmt. Liegt die Intensität unter einem Schwellwert, so wird das entsprechende Extremum aussortiert. Des Weiteren wird für jedes verbleibende Extremum bestimmt, ob es auf einer Kante liegt. An jedem Extremum werden Gradienten berechnet, die senkrecht zueinander sind. Sind beide Gradienten klein, handelt es sich um eine flache Region. Ist ein Gradient groß und der andere klein, so befindet sich das Extremum auf einer Kante.

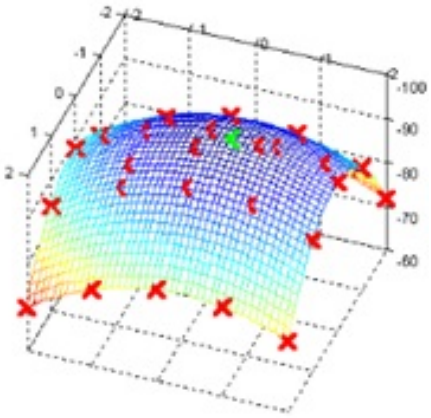


Abbildung 3.25: Die roten Kreuze geben die Pixelpositionen an. Das grüne Kreuz gibt die tatsächliche Position des Extremums an (Quelle: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoints/)

In beiden Fällen wird das Extremum aus der Auswahl entfernt. Sind beide Gradienten groß, so liegt das Extremum wahrscheinlich auf einer Ecke, was wünschenswert ist. Die verbleibenden Extrema werden als Keypoints betrachtet. Da zu jedem Extrempunkt die Skale und die Oktave bekannt sind, so ergibt sich die Skalierungsstufe für jeden Keypoint. Diese ermöglicht den Keypoints Invarianz gegenüber Skalierung.

Orientierungszuweisung Die Orientierungszuweisung erfolgt, um Keypoints invariant gegenüber Bildrotation zu machen. Dabei wird um jeden Keypoint die Größe und Richtung der Gradienten berechnet (siehe Abbildung 3.26). Diese Gradienten werden in einem Orientierungshistogramm mit 36 Bins dargestellt (siehe Abbildung 3.27). Dabei entspricht jeder Bin 10 Grad Orientierung, d.h. 0-9 Grad, 10-19 Grad, etc.. Der Wert eines Bins entspricht den Beträgen der Gradienten, die die selbe Orientierung teilen. Falls es nur einen Peak gibt, so wird dem Keypoint diese Orientierung zugewiesen. Gibt es mehrere Peaks, die mindestens 80% des Betrags der größten Orientierung haben, so werden mehrere Keypoints mit jeweils entsprechender Orientierung erstellt. Da jedem Keypoint eine Orientierung zugewiesen wurde, sind sie rotationsinvariant.

Erstellung des Keypoint-Deskriptors In diesem Schritt wird jedem Keypoint ein Vektor zugewiesen. Dieser soll ein beschreibender Fingerprint sein. Dafür wird um jeden Keypoint ein 4×4 Fenster gelegt. Dies wird in Abbildung 3.28 dargestellt. Auf der linken Seite der Abbildung ergibt sich so ein 16×16 Gitter um den Keypoint, der rot markiert ist. In jedem Feld des Gitters wird der Gradient samt seiner Größe und Ausrichtung berechnet. Von der Orientierung der Gradienten wird die Orientierung des entsprechenden Keypoints abgezogen, um die Rotationsinvarianz umzusetzen. Für jedes Fenster (gekennzeichnet durch dickere Linien in der linken Abbildung) werden die Gradienten in einem

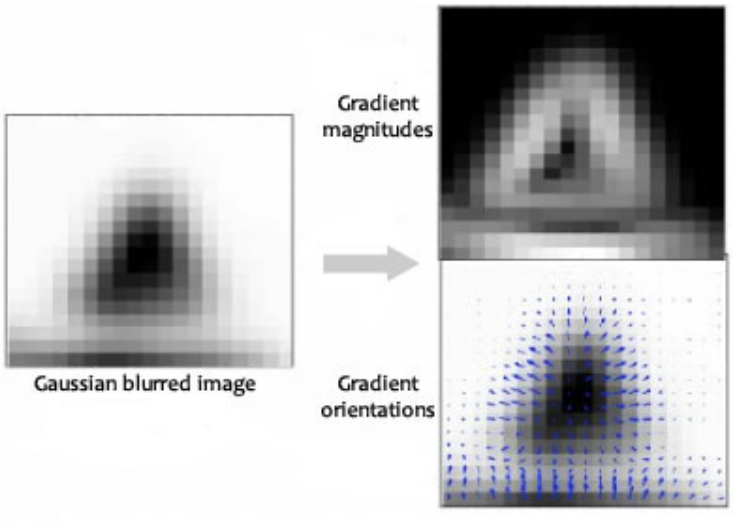


Abbildung 3.26: Zu jedem Pixel des linken Bildes wird die Größe des Gradienten (rechtes Bild oben) und die Richtung des Gradienten (rechtes Bild unten) berechnet (Quelle: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/)

gewichtetem Orientierungshistogramm mit 8 Bins zusammengefasst. Dies ist auf der rechten Seite der Abbildung 3.28 zu sehen. Zu beachten ist, dass die Gewichtung der Beträge der Gradienten abhängig von ihrer Entfernung zum Keypoint ist. Je weiter weg, desto weniger fällt der Gradient ins Gewicht. Jedes Orientierungshistogramm eines der 16 Fenster um einen Keypoint wird dann durch 8 Werte beschrieben. Bei insgesamt 16 Fenstern ergibt sich für jeden Keypoint ein Vektor mit 128 Elementen. Dieser Vektor wird normiert. Um den Vektor robuster gegenüber Veränderungen in der Beleuchtung zu machen, werden Werte, die 0.2 überschreiten, auf 0.2 begrenzt. Es ist zu beachten, dass die Anzahl der Fenster und die Anzahl der Orientierungen im Histogramm variabel sind. 4×4 Fenster und 8 Orientierungen sind jedoch typische Größen.

Diskussion

Weil die Projektgruppe einzelne Objekte in Bildern erkennen möchte, werden lokale Merkmale gegenüber globalen Merkmalen bevorzugt. Da einzelne Objekte erkannt werden sollen und nicht die räumliche Einordnung einer Szene, werden diese globalen Modelle als nicht relevant betrachtet.

Die von SIFT berechneten Merkmale werden sowohl zur Klassifikation (Kapitel 6.1.3) als auch zum Retrieval (Kapitel 6.1.3) genutzt und bilden damit einen wichtigen Bestandteil der Erkennung neben der Barcode-Detektion (Kapitel 6.1.3). SIFT bietet Merkmale, die invariant gegenüber Skalen, Rotation und partiell invariant gegenüber Veränderungen der Beleuchtungen und des Blickpunkts sind. Des Weiteren ist SIFT eine gängige Methode, um zuverlässige Merkmalsdeskriptoren zu berechnen. In [24] wurden SIFT-Desktriptoren

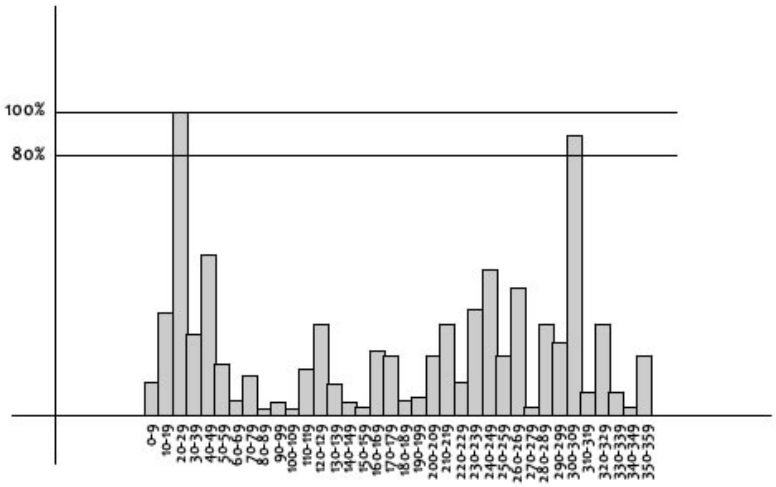


Abbildung 3.27: Orientierungshistogramm mit 36 Bins. In diesem Fall gibt es zwei Peaks. So würden zwei Keypoints mit jeweiliger Orientierung erstellt werden (Quelle: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-keypoint-orientation/)

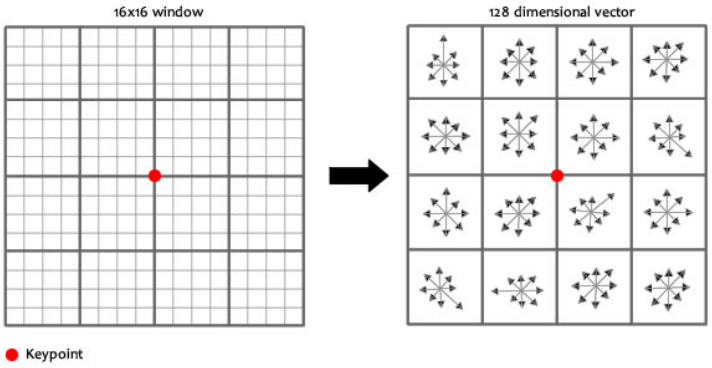


Abbildung 3.28: Im linken Bild wird die Aufstellung eines 16×16 Fenster um den Keypoint gezeigt. Im rechten Bild werden die 16×16 Fenster zu 4×4 Orientierungshistogrammen zusammengefasst (Quelle: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-features/)

mit anderen bekannten Deskriptoren verglichen. So wurden unter anderem auch die Genauigkeiten der einzelnen Deskriptoren bei Skalenänderungen, Änderungen des Blickpunktes und Drehung der Bilder miteinander überprüft. Dabei erzielte SIFT gute Werte für *recall* und *1-precision* unter verschiedenen Skalen, Bilddrehungen, Blickpunktänderungen und Weichzeichnungen. Für eine detaillierte Erklärung von *recall* und *precision* sei auf Kapitel 3.3.1 verwiesen. Die zuverlässigen Werte unter den genannten Änderungen sind hilfreich, da es bei der Kommissionierung vorkommen kann, dass die vom Kommissionierer

aufgenommenen Bilder sich von den vorhandenen Referenzbildern z. B. durch Blickwinkel und Entfernung unterscheiden.

3.3 Bilderkennung

Dieses Kapitel beschäftigt sich mit der Erkennung von Objekten und Strukturen in Bildern. Zunächst wird in Kapitel 3.3.1 beschrieben, wie zu einem gegebenen Referenzbild ähnliche Bilder aus einer Datenmenge ermittelt werden können. Kapitel 3.3.3 beschäftigt sich mit der Textdetektion in Bildern und 3.3.4 mit der Erkennung von Barcodes.

3.3.1 Image-Retrieval

Häufig ist es notwendig, gewünschte Informationen aus einer großen Menge von Daten zu extrahieren bzw. wiederzufinden. Für strukturierte bzw. textuelle Daten wird diese Aufgabe *Information Retrieval* genannt, bei der Anwendung auf Bilder wird der Begriff *Image Retrieval* verwendet.

Eine der Hauptschwierigkeiten ist, die in den Bildern enthaltenen Informationen automatisch in eine durchsuchbare Form zu bringen. Die zu durchsuchenden Kriterien können in den Metadaten vorhanden sein (Tags o.ä.), alternativ können auch die Inhalte der Bilder betrachtet werden, man spricht dann von *Content Based Image Retrieval* (CBIR). Im Fall von CBIR kann nach einzelnen, benannten Objekten in den Bildern gesucht werden, oder man vergleicht das gesamte Bild mit dem einen Referenzbild.

Grundlagen

Content Based Image Retrieval ist eine Kombination aus Bildverarbeitung und Information Retrieval. Der grundlegende Ablauf ist zweistufig. Eine erste Stufe baut eine Datenbank auf, während in einem zweiten Schritt ein Referenzbild mit den Informationen (Merkmale, vgl. Kap. 3.2.4) aus der Datenbank verglichen wird.

Für den ersten Schritt werden die gewünschten Informationen (Merkmale) aus den Bildern extrahiert, aus welchen anschließend eine Datenbank aufgebaut wird. Um eine effiziente Suche zu ermöglichen, wird über der Datenbank ein Such-Index erstellt. Dieser erste Schritt muss nur durchlaufen werden, wenn neue Bilder zur Datenbasis hinzugefügt werden sollen.

Der zweite Schritt wird für jede Suchanfrage durchgeführt. Es werden die gewünschten Informationen aus dem Suchbild extrahiert, diese werden dann mit den Informationen aus der Datenbank verglichen. Typischerweise werden mehrere Bilder gefunden, welche ähnliche Merkmale wie die des Suchbildes enthalten. Diese Bilder werden nach einem oder mehreren Kriterien sortiert und ausgegeben. Abschließend kann die Sortierung und die Suche allgemein noch mit statistischen Mitteln beurteilt werden, um die Güte der Suche (richtige/falsche Funde usw.) zu ermitteln.

Darstellung des Bild-Inhalts

Die Methoden des Image Retrieval entsprechen im Wesentlichen denen des Information Retrieval, welche jedoch auf textuellen Daten arbeiten. Im Falle des CBIR wird der Bildinhalt durch die Merkmale des Bildes dargestellt. Es werden jedoch nicht die einzelnen Merkmale betrachtet, stattdessen werden sie zu einem beschreibenden Vektor zusammengefasst, oft wird ein Bag of Features (BoF) Ansatz benutzt.

Bag of Features BoF entstand aus dem Ansatz *Bag of Words* aus dem Gebiet der Textklassifikation, bei dem Vorkommen bestimmter Wörter als Merkmalsvektoren zur Klassifikation von Texten genutzt werden. Dazu wird aus den Wörtern der zu klassifizierenden Texte ein Vokabular erstellt. Der Merkmalsvektor eines Textes besteht aus der Häufigkeit des Vorkommens der Wörter aus dem Vokabular im jeweiligen Text. Um diese Methoden auf Bilder anwenden zu können, muss die Darstellung des Bildinhalts so angepasst werden, dass die vorhandenen Methoden angewendet werden können. Es muss eine andere, textuelle Darstellung des Bildinhalts ermittelt werden. Der Ablauf der Ermittlung einer Darstellung des Bildinhalts wird in Abbildung 3.29 skizziert.

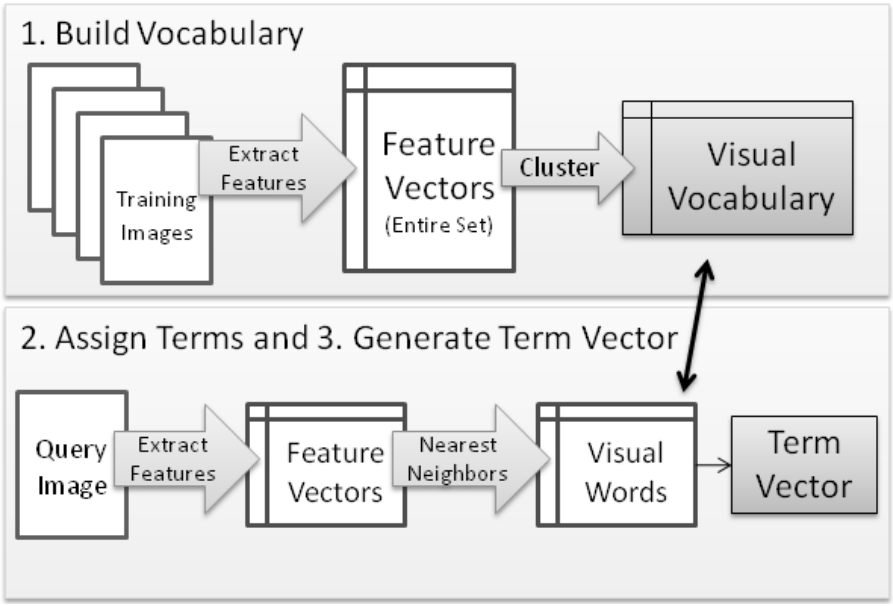


Abbildung 3.29: Ermittlung der Darstellung des Bild-Inhalts mit *Bag of Features* Repräsentation. [26]

Die Grundlage für die Darstellung von Bildern als *Bag of Features* ist, wie auch beim Verarbeiten von Texten, ein bekanntes Vokabular. Dieses wird anhand eines Satzes von Trainingsbildern ermittelt: Die lokalen Merkmale werden aus den Bildern extrahiert (siehe Kapitel 3.2.4), und zu Wörtern zusammengefasst, bspw. durch Clustering (vgl. [26], [28]). Dieses Vokabular wird einmalig festgelegt, und anschließend während der weiteren Verwendung nicht verändert.

Basierend auf diesem Vokabular können jetzt die eigentlichen Merkmalsvektoren aus den Bildern extrahiert und notiert werden. Hierzu wird in jedem Bild nach lokalen Merkmalen gesucht, die gefundenen lokalen Merkmale werden anschließend z. B. mit einer Nächste-Nachbar-Suche den Repräsentanten des Vokabulars zugeordnet. Somit ergibt sich für jedes Bild ein Satz von Repräsentanten aus dem vorgegebenen Vokabular, die den Inhalt des Bildes beschreiben. Der Merkmalsvektor des Bildes besteht aus dem Histogramm (Häufigkeitsverteilung) der vorkommenden Repräsentanten des Vokabulars. Der Inhalt eines Bildes wird somit durch seinen Merkmalsvektor beschrieben.

Merkmalsextraktion Zur Ermittlung der Positionen der Merkmale (*Feature Detection*) gibt es zwei Möglichkeiten: die Ermittlung von relevanten Positionen mit Hilfe von Detektoren oder gleichmäßige Verteilung von Merkmalen über das gesamte Bild mit Hilfe eines Rasters. Anschließend werden die gefundenen Positionen mit Deskriptoren beschrieben (*Feature Description*).

Zur Merkmals-Suche gibt es zwei wichtige Klassen von Detektoren: *Interest Point Operators* und Detektoren basierend auf visueller Auffälligkeit (*Visual Saliency*). Von Interest Point Operatoren werden lokal klar begrenzte Regionen wie Kanten, Kurven oder Flecken erkannt. Die am häufigsten genutzten Interest Point Operatoren sind bspw. SIFT (*Scale-invariant feature transform*, vgl. Kap. 3.2.4), *Harris-Affine Detektoren* oder *Maximally Stable Extremal Regions (MSER) Detektoren*, da diese in vielen Situationen gute Ergebnisse liefern. Eine Alternative sind Detektoren, die die visuelle Auffälligkeiten auswerten. Mit Hilfe von Berechnungs-Modellen werden Regionen erkannt, die visuell auffällig sind, wie Farbunterschiede, auffällige Orientierungen in Linien oder unterschiedliche Helligkeits-Intensitäten(vgl. [26]). Falls keine Detektoren genutzt werden, werden die Positionen der Merkmalspunkte mit einem Raster festgelegt, welches über das Bild gelegt wird. Hierbei werden die Merkmale in regelmäßigen Abständen über das Bild verteilt.

Die so ermittelten Positionen werden mit einem Feature Descriptor beschrieben. Hier wird beim Image Retrieval hauptsächlich der SIFT-Deskriptor genutzt, da dieser sehr robust ist und in vielen Situationen gute Ergebnisse liefert (vgl. [26], Kapitel 3.2).

Datenbank und Suche

Die ermittelten Beschreibungen werden sinnvollerweise in einer Datenbank gespeichert, hier wird für jedes Bild der zusammengefasste Merkmalsvektor gespeichert. Ein Beispiel für die Größe einer Datenbank liefern Sivic und Zisserman ([34]), die von ihnen erstellte *Video Google* Datenbank enthält 10.000 Merkmale (Wörter), 4.000 Bilder und pro Bild 2.000 Merkmale. Eine Suche nach einem Bild nimmt in dieser Datenbank ca. 0,1 Sekunde in Anspruch.

Um die Datenbank nach einem Referenzbild zu durchsuchen, wird der Merkmalsvektor des Referenzbildes berechnet. Dann wird die Datenbank nach den Merkmalen des Referenz-

bildes durchsucht. Alle Bilder, die Merkmale des Referenzbildes enthalten, werden weiter betrachtet. Dies ist besonders effizient, wenn der Merkmalsvektor viele 0-Einträge enthält, also dünn besetzt ist. In diesem Fall können viele Merkmale im Index aus der Betrachtung ausgeschlossen werden. Ausserdem sind die Rechenoperationen auf dünn besetzten Vektoren extrem schnell. Es wird für jedes gefundene Bild eine Ähnlichkeit zum Referenzbild berechnet. Hierzu könnte bspw. die Cosinus-Distanz

$$\cos(\theta) = \frac{a \cdot b}{\|a\|\|b\|}$$

zwischen dem Merkmalsvektor des Referenzbildes (a) und dem des gefundenen Bildes (b) berechnet werden. Die Cosinus-Distanz ist an dieser Stelle sinnvoll, da es sich um ein normierendes Maß handelt. Bilder mit unterschiedlichen Größen (Auflösungen) enthalten unterschiedlich viele Mermale, auch wenn es sich um den gleichen Bildinhalt handelt. Durch ein normierendes Maß lassen sich diese jedoch auch einfach vergleichen. Anhand dieses Ähnlichkeitsmaßes wird ein Ranking erstellt, welches die Bilder der Datenbank nach der Ähnlichkeit mit dem Referenzbild sortiert. Die Ausgabe des Image Retrieval ist somit eine Auswahl von Bildern aus der Datenbank, sortiert nach der Ähnlichkeit mit dem Referenzbild.

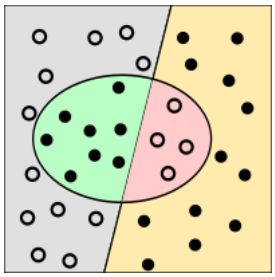
Auswertung der Ergebnisse

Bevor die so gewonnenen Informationen genutzt werden, können und sollten die Suchergebnisse bewertet und verifiziert werden. Wichtige Fragestellungen sind bspw. wann ein Bild noch als ähnlich zum Referenzbild betrachtet wird, und wann es nicht mehr ausreichend ähnlich ist, um in das Suchergebnis aufgenommen zu werden. Eine weitere Fehlerquelle ist die Merkmalsextraktion, auch hier können Artefakte oder besondere Strukturen im Bild Fehler hervorrufen. Somit muss jede Suche, zumindest in einer Evaluationsphase, bewertet werden. Dies kann manuell durch den Benutzer geschehen oder aber automatisch mit statistischen Methoden. Eine manuelle Bewertung bietet sich für den Einführungs- bzw. Erprobungszeitraum eines Suchalgorithmus an, während eine automatische Bewertung auch später stattfinden kann.

Bei der manuellen Bewertung gibt ein Benutzer an, ob ein gefundenes Objekt dem gesuchten entspricht. Anhand dieser Rückmeldung können dann die Suchparameter (wie Abstandsmaß oder gewünschter Ähnlichkeitsschranke) sowie die Ermittlung der Merkmalsvektoren angepasst werden.

Bei der automatischen bzw. statistischen Bewertung werden verschiedene Kriterien betrachtet. Informell sind dies die Anzahl der gefundenen Bilder, das Maß der Übereinstimmung mit dem Referenzbild sowie die Anzahl falscher Funde, die nicht zum Referenzbild passen. Formell sind diese Kriterien mit den statistischen Kriterien *Precision (Genauigkeit)*, welches den Anteil relevanter Bilder im Suchergebnis beschreibt, *Recall (Trefferquote)*, der Anteil gefundener Bilder verglichen mit der Gesamtzahl relevanter Bilder, sowie *Fallout*

(Ausfall), der Anzahl irrelevanter Bilder im Suchergebnis zu beschreiben. Anhand dieser Kriterien lassen sich die Bilder im Suchergebnis in vier Kategorien unterteilen, diese sind in Abbildung 3.30 grafisch dargestellt.



$$precision = \frac{\{\text{relevante Dok.}\} \cap \{\text{gefundene Dok.}\}}{\{\text{gefundene Dokumente}\}}$$

$$recall = \frac{\{\text{relevante Dok.}\} \cap \{\text{gefundene Dok.}\}}{\{\text{relevante Dokumente}\}}$$

	Relevant	Nicht relevant
Gefunden	8	4
Nicht gefunden	12	12

Abbildung 3.30: Darstellung der Klassifikation als Menge (oben) bzw. Wahrheitsmatrix (unten) (Grafiken: Wikipedia)

Bei der Auswertung dieser Werte ist zu beachten, dass die Werte im Allgemeinen voneinander abhängig sind. Als eine Möglichkeit ist zu nennen, dass man die Suchparameter so verändert, dass eine Dimension fixiert und nur eine andere Dimension verändert wird. Somit kann man z. B. auswerten, wie sich Änderungen an den Suchparametern bei einer festen Trefferquote auf die Genauigkeit auswirkt, man spricht dann bspw. von *precision bei einem recall von 0.75*.

Eine weitere Möglichkeit wäre, mehrere Werte zu einem Maß zusammenzufassen. Nennenswert ist hier vor allem das F_1-Maß, welches Precision und Recall in einem gewichteten harmonischen Mittel zusammenführt:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Probleme ergeben sich auch durch nicht bekannte, oder nicht hinreichend bestimmbare Werte. So ist nicht immer bekannt, wie groß die Datenbasis ist, bzw. wie viele relevante Bilder für eine Suche in der Datenbasis enthalten sind. In diesem Fall kann nur die precision ermittelt werden, der recall jedoch nicht. Bei einer bekannten Datenbasis, wie z. B. einem abgeschlossenen Produktspektrum, können diese Werte ermittelt werden. Somit lässt sich im Rahmen der Projektgruppe ermitteln, wie gut die genutzten Verfahren das vorgegebene Produktspektrum (vgl. Pflichtenheft) abdecken (siehe auch Kapitel 6.1.3). Sobald sich jedoch die Datenbasis ändert, müssen diese Werte neu ermittelt werden, auch muss die Verifikation der Suche neu durchgeführt werden. Ein weiteres Problem ist die Fragestellung, wann ein Bild relevant ist. Sucht man ein bestimmtes Produkt in einem Produktkatalog, ist auch diese Frage einfach zu beantworten. Gibt es jedoch ein Produkt in möglicherweise unterschiedlichen Varianten oder Verpackungen, wird die Bestimmung ungleich schwieriger.

Anpassung der Suchparameter

Bei der Wahl der Suchparameter muss sorgfältig abgewogen werden, welche Ausgaben man als Suchergebnis akzeptiert. Zum einen müssen hinreichend viele Bilder gefunden werden, andererseits dürfen auch nicht zu viele falsche Bilder im Suchergebnis erscheinen. Diese Diskrepanz wird durch die Werte *Precision* und *Recall* dargestellt, mittels Evaluation müssen die Suchparameter eingestellt werden. Eine Anpassung ist auch dynamisch möglich, nach der Rückmeldung eines Benutzers über fälschlicherweise ausgegebene Bilder, oder einen zu geringen Umfang der Ergebnisse.

Es ergeben sich mehrere Möglichkeiten zur Beeinflussung des Suchergebnisses. Bei der Implementierung und dem Aufbau der Datenbasis muss ein passender Merkmalsdetektor verwendet werden, auch müssen die verwendeten Merkmale zu der Aufgabe passen. Diese Parameter sind jedoch hinterher nicht mehr zu beeinflussen und müssen daher von Anfang an sinnvoll gewählt und evaluiert werden. Ein recht gut anpassbarer Suchparameter ist das Distanz-Maß, ein typisches Distanz-Maß ist bswp. die Cosinus-Distanz. Ein Distanzmaß bewertet jeden Eintrag im Merkmalsvektor gleich, jedoch sind manche Merkmale evtl. wichtiger als andere. Ein weniger häufiger vorkommendes Merkmal ist aussagekräftiger als ein häufig vorkommendes Merkmal. Um häufig vorkommende und somit wenig aussagekräftige Merkmale zu entfernen, kommen *stop-word-lists* zum Einsatz.

Oft genutzt wird auch die unterschiedliche Gewichtung unterschiedlicher Merkmale. Hier ist vor allem die Gewichtung des Merkmalsvektors mittels TF-IDF (*Term Frequency - Inverse Document Frequency*, dt. Vorkommenshäufigkeit - inverse Dokumenthäufigkeit) zu nennen. Dabei werden jene Merkmale höher gewichtet, die selten in der Datenbasis vorkommen, oder in einem Bild sehr häufig. (vgl. [26], [22])

Wichtig ist auch die höchste zulässige Distanz eines Merkmalsvektors zu einem Vektor der Datenbank, innerhalb derer das Bild als gleich angesehen wird. Hier wäre auch eine Staffelung in mehrere Wahrscheinlichkeitsstufen denkbar.

Eine weitere Möglichkeit zur Verbessung des Suchergebnisses ist der 2^{nd} Nearest Neighbour Test. ([21]) Die Idee ist, dass ähnliche Bilder durch den Vergleich von Merkmalen erkannt werden, und somit alleinstehende Merkmale eine besonders gute Identifikation erlauben. Es wird beim Vergleich der Merkmale getestet, wie eindeutig ein Treffer in der Suche ist. Dazu werden die Distanzen von erstem und zweitem Nachbar eines Merkmals verglichen: $nnThreshold = \frac{1^{st}\text{ NN distance}}{2^{nd}\text{ NN distance}}$. Akzeptiert werden nur die Merkmale, bei denen der zweite Nachbar weit genug entfernt liegt, bspw. $nnThreshold \approx 0.8$.

In vielen Fällen zeigen die Referenzbilder zwar den gleichen Bildinhalt, jedoch aus unterschiedlichen Perspektiven, Skalierungen oder Aufnahmeentfernungen (Abbildung 3.31). Diese Unterschiede lassen sich durch geometrische Transformationen wie Verschiebung, Streckung, Drehung beschreiben. Dies kann man sich zur Hilfe machen, um bei einem Vergleich zweier Bilder die zueinander gehörigen Merkmale besser identifizieren zu können.

Abbildung 3.31: Gleiche Objekte aus unterschiedlichen Perspektiven [28]

Man berechnet für ein Merkmal die Transformation, und wendet diese Transformation auf alle Merkmals-Punkte im Bild an. Anschließend werden nur die Übereinstimmungen zwischen Merkmalen im Referenzbild und im Vergleichsbild akzeptiert, die einen gewählten Abstand voneinander nicht überschreiten. Diese Schritte werden für jedes Merkmal durchgeführt, anschließend wird mit der Transformation, die die meisten Übereinstimmungen produziert, weitergerechnet. (vgl. [36]) Auf diese Weise werden nicht alle Übereinstimmungen willkürlich ausgewertet, sondern nur die Übereinstimmungen, die zur erwarteten geometrischen Transformation passen.

Verwendung im implementierten System

Image Retrieval ist neben der Barcode-Detektion die zweite wichtige, in der Projektgruppe genutzte Technik zur Erkennung des aktuell gegriffenen Artikels. Während für die Barcode-Erkennung recht deutliche Bilder des Barcodes benötigt werden, reichen beim Image Retrieval auch weniger scharfe Bilder aus, um die benötigten Merkmale zu berechnen. Dies führt zu einem relativ robusten Verfahren zur Objektdetektion.

3.3.2 Klassifikation

Klassifikation ist eines der klassischen Probleme des maschinellen Lernens, dessen Ziel es ist, Beobachtungen anhand von Merkmalen einer Klasse einzuordnen. In der Mustererkennung wird Klassifikation genutzt um beispielsweise Gegenstände einer, für den Menschen natürlichen, Kategorie zuzuordnen (siehe Abbildung 3.32). Da vorheriges statistisches Wissen über den Problemkreis als Eingabe zugeführt werden muss, wird die Klassifikation dem Teilgebiet des überwachten Lernens zugeschrieben.
Der gesamte Erkennungsvorgang für das System setzt sich aus folgenden Teilen zusammen:

1. **Vorverarbeitung** - Transformation der Daten mit dem Ziel für die weiteren Schritte besser geeignet zu sein

2. **Merkmalsextraktion** - Reduktion der Daten auf die für die Klassifikation wichtigen Informationen

3. **Klassifikation** - Abbildung eines Merkmalsvektors auf eine Klasse

Da Vorverarbeitung und Merkmalsextraktion bereits in 3.2.2, 3.2.3 und 3.2.4 beschrieben wurden, konzentriert sich der folgende Abschnitt auf den Klassifikationsvorgang.

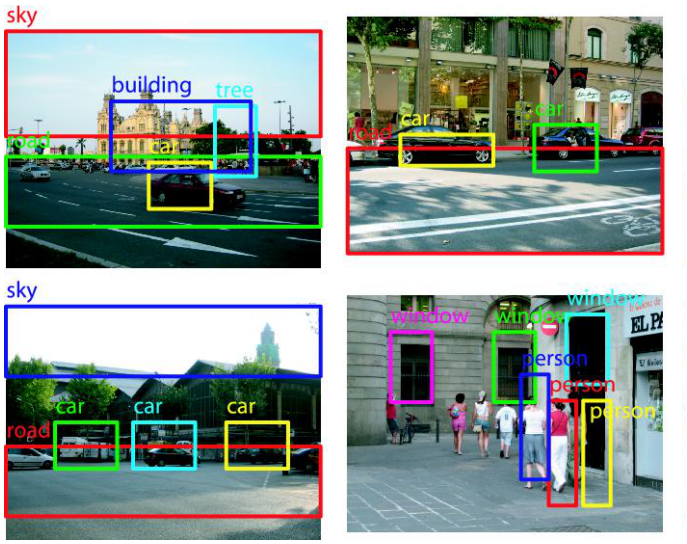


Abbildung 3.32: Fotos auf denen verschiedene Objekte erkannt und klassifiziert wurden. (Quelle: http://www.marcosanbiagio.info/object-detection-and-classification/)

Aufbau

Als Ausgangsmaterial werden neben den zu klassifizierenden Daten noch sogenannte Trainingsdaten benötigt. Trainingsdaten bestehen aus Beobachtungen, welche die selbe Vorverarbeitung durchlaufen und die selben Merkmale wie die späteren zu klassifizierenden Daten besitzen. Außerdem gehört zu jeder Beobachtung das spezielle Merkmal *Label*, welches eine Klassenzugehörigkeit spezifiziert. Die Trainingsdaten dienen dazu dem Klassifikator möglichst gute Kenntniss über die statistischen Eigenschaften der Klassen zu verschaffen. Dieser verarbeitet diese Informationen zu einem statistischen Modell, anhand dessen die Klassifikation anschließend vorgenommen wird.

Die Berechnung des Models wird Trainingsphase, die anschließende Klassifikation von Daten ohne Label wird Testphase genannt. Der komplette Klassifikationsvorgang ist in Abbildung 3.33 schematisch abgebildet.

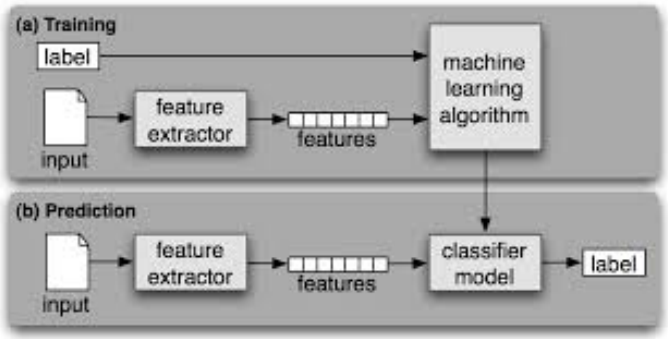


Abbildung 3.33: Schematische Abbildung des Klassifikationsvorganges. Dieser teilt sich auf in (a) Training und (b) Testing. Im Training wird anhand eines gelabelten Datensatzes ein Klassifikationsmodel berechnet, welches in der Testphase angewandt wird, um die Testdaten einem Label (also einer Klasse) zuzuordnen. (Quelle: http://www.nltk.org/book/ch06.html)

Algorithmen

Es gibt viele Algorithmen, die den Prototyp des Klassifikators implementieren. Zu den bekanntesten zählen die Support Vector Maschinen (SVM), die Bayes-Klassifikatoren, k-Nearest-Neighbor (kNN) und künstliche neuronale Netze. Da das entwickelte System eine Support Vector Machine zur Klassifikation nutzt, wird dieser Algorithmus im Folgenden kurz beschrieben.

Support Vector Machine

Ziel der SVM ist es, wie für jeden Klassifikator, jeden Punkt im Merkmalsraum eindeutig einer Klasse zuzuordnen. Diese Zuordnung soll, auf Basis von Trainingsdaten, möglichst genau mit der realen Klassenverteilung übereinstimmen. Dies wird versucht zu erreichen, indem die SVM den Vektorraum mittels einer Hyperebene zwischen den Klassen trennt. Da es evtl. viele Hyperebenen gibt, die beide Klassen trennen, wird stets die Hyperebene mit maximalen Abstand zu den nächsten Trainingsvektoren gesucht (auch *maximum-margin hyperplane* genannt).

So lassen sich mit der SVM scheinbar nur Zwei-Klassen-Probleme lösen. Dies wird umgangen, indem mehrere SVMs für je zwei Klassen berechnet werden. Man unterscheidet zwischen *one-versus-all*, wobei jede Klasse gegen alle anderen Klassen (zusammengefasst zu einer großen Klasse) getrennt wird, und *one-versus-one*, wobei jedes Paar von Klassen getrennt wird. Es gibt zudem noch weitere Methoden zur Multiclass SVM. Abbildung 3.34 zeigt eine maximum-margin Hyperebene einer SVM im zweidimensionalen Merkmalsraum.

3.3.3 Textdetektion

Natürliche Szenen enthalten oftmals Text, der wertvolle Informationen über diese enthalten kann. Es gibt unzählige vorstellbare Anwendungsgebiete für maschinelle Fähigkeit,

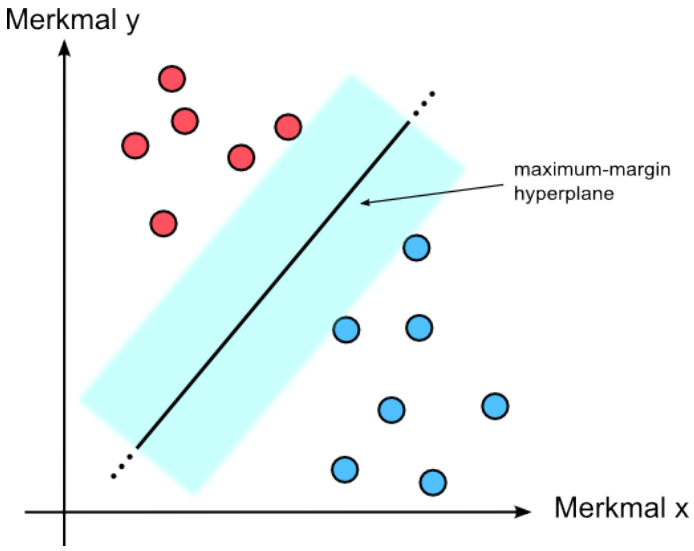


Abbildung 3.34: Einteilung eines zweidimensionalen Merkmalsraumes in zwei Klassen durch Hyperebene der SVM. Abstand (hier blau) von Hyperebene und nächsten Vektoren der Trainingsdaten ist maximiert (maximal-margin hyperplane).

textuelle Informationen aus einem digitalen Bild zu gewinnen. Beispielsweise können damit wortbasierte Suchanfragen auf Bildbeständen geführt werden, die sich den Text im entsprechenden Bild zunutze machen. Ebenso wäre die Texterkennung ein zweifelsohne wichtiges Hilfsmittel für Blinde, um den Betroffenen im Alltag zu helfen.

Grundlagen

Text, der in Bildern zu sehen ist, lässt sich in für gewöhnlich in eine von zwei Untergruppen kategorisieren [17]:

- **Szenentext** (engl. *scene text* oder *graphics text*)
- **künstlich hinzugefügter Text** (engl. *artificial text* oder *caption text*)

Beim künstlich hinzugefügten Text handelt es sich um Text, der in einer späteren Bearbeitungsphase dem Bild hinzugefügt wurde, bspw. der Name von jemandem, der interviewt wird (vgl. Abbildung 3.35(a)). Im Gegensatz dazu ist Szenentext selbst Teil der Szene, z. B. ein Banner oder ein Plakat im Bild (vgl. Abbildung 3.35(b)). Die Detektion von Szenentext ist auf Grund von Bedingungen der Umwelt, wie Schatten oder Perspektive, ein weitaus komplizierteres Problem.

Zieht man das Anwendungsgebiet von Pick by Feel zu Grunde, so wird im weiteren Verlauf dieses Kapitels nur noch die Detektion von Szenentext behandelt.

Die Textdetektion ist in der Regel der erste Schritt in einer Reihe von Verfahren bis zum maschinenlesbaren Text. Sie hat die Aufgabe, Regionen in Bildern zu identifizieren, in

(a) Künstlich hinzugefügter Text (Quelle: Stromberg)

(b) Szenentext (Quelle: [15])

Abbildung 3.35: Unterscheidung zwischen Szenentext und künstlich hinzugefügtem Text

denen Text vorhanden ist. Als fortführender Schritt geschieht die Textlokalisierung, das bedeutet, das gleicher Text in benachbarten Regionen verschmolzen wird. Nachdem der Text lokalisiert wurde, findet die Binarisierung statt, welche die Textpixel und die Hintergrundpixel voneinander separiert. Im letzten Schritt läuft dann eine Zeichenerkennung (engl. *Optical Character Recognition*, OCR), die das Ziel hat, die Textpixel in einen korrespondierenden ASCII Text umzuwandeln. Abbildung 3.36 visualisiert die Abfolge grafisch [17].

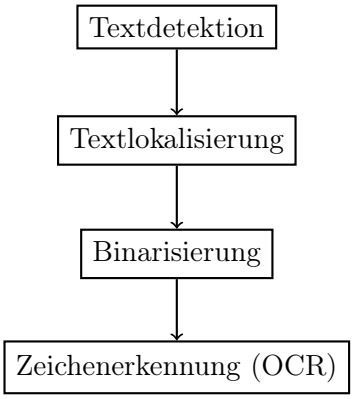


Abbildung 3.36: Allgemeine Vorgehensweise zur Texterkennung

Ein Beispiel für eine erfolgreiche Textdetektion und Lokalisierung liefert Abbildung 3.37. In der Detektion wurden alle Regionen des Bildes, die Text enthalten, erkannt. Die Textlokalisierung unterteilt die gefundenen Regionen dann in 4 Bereiche, die zusammengehörigen Text bzw. Wörter symbolisieren.

Abbildung 3.37: Erfolgreiche Textdetektion und Lokalisierung in einem digitalen Bild (Quelle: [37])

Mögliche Ansätze

Es gibt bereits eine Vielzahl von Ansätzen und Möglichkeiten, Text auf Bildern zu lokalisieren und folglich zu erkennen. So wurde bspw. der Algorithmus von Gao und Yang (vgl. [16]), der chinesische Schrift auf Schildern lokalisiert, betrachtet und ein allgemeinerer Algorithmus, der anwendungsunabhängig über maschinelles Lernen funktioniert (vgl. [15]), untersucht. Der Ansatz von Gao und Yang ist ein Ansatz, der sich auf bestimmte Eigenschaften des Textes beruft. Das hat den Vorteil, dass die Textdetektion für ihr spezielles Anwendungsgebiet relativ gute Erfolgsaussichten bietet und schnell arbeitet, für eine globale Textdetektion in einem beliebigen Anwendungsgebiet jedoch völlig unbrauchbar ist. Ebenso kann der Algorithmus nur schwer mit den vielen Schwierigkeiten in natürlichen Szenen umgehen. Die Textdetektion durch maschinelles Lernen ist ein skalierbares und schnelles Verfahren, um Text mit relativ hohen Erfolgsaussichten in einem Bild zu entdecken. Es ist insofern als schnell anzusehen, da der größte Rechenaufwand beim Trainieren benötigt wird. Für die reine Textdetektion müssen aus dem Bild dann nur noch die Merkmalsvektoren berechnet und klassifiziert werden.

Schwierigkeiten

Bilder mit natürlichen Szenen stellen die Textdetektion in der Regel vor eine komplexe und schwierige Aufgabe, da es kein einheitliches Muster oder etwaige Vorbestimmungen für Text in diesen gibt. Im Folgenden sind einige Schwierigkeiten bei Szenentext aufgelistet, unterteilt in die Schwierigkeiten im Bild, die durch die Szene oder die Kamera gegeben sind, und die Schwierigkeiten im Text selber, der beliebig variabel gehalten werden kann [16]:

1. im Bild:
    - schlechte Qualität/Komprimierung des Bildes

- instabile Kameraführung
- beliebig komplexer/ungleichmäßiger Hintergrund
- perspektivische Verzerrung
- ungleichmäßige Beleuchtung durch Schatten, Reflexionen oder durch den Kamera-Blitz

2. bei der Schrift:
   - Schriftgröße
   - verschnörkelte Schriftarten
   - Schriftfarbe
   - Ausrichtung der Schrift
   - Zeichenabstand

(a) Reflexionen und ungleichmäßiger Hintergrund

(b) Schriftfarbe

(c) Belichtung bzw. Blitz

(d) perspektivische Verzerrung und Schriftgröße

Abbildung 3.38: Schwierigkeiten bei der Textdetektion

Einige der Schwierigkeiten lassen sich leicht an den Beispielen in Abbildung 3.38 zeigen.

So zeigt das Bild 3.38(a) einen Text auf einer Glasscheibe, der durch Reflexionen und den ungleichmäßigen Hintergrund bzw. den schwachen Kontrast selbst für das menschliche Auge schon schwer zu lesen ist. Die „1“ am rechten Rand des Bildes unterscheidet sich farblich und in der Schriftgröße stark von dem Text auf der Glasscheibe - für eine Textdetektion ist es wünschenswert, dass beide Regionen als Text erkannt werden. Das Bild 3.38(b) zeigt eine Plakatwand, die als Schriftfarbe ein Hintergrundbild enthält. Dadurch entstehen auch Kontraste bzw. Kanten innerhalb der Schrift, die für eine Textdetektion als unwichtig und hinderlich einzustufen sind. Bild 3.38(c) zeigt, wie eine Textdetektion durch eine unsachgemäße Kamerahandhabung erschwert wird. Hier hilft der Blitz zwar, die rechte Seite des Textes als solche zu erkennen, leuchtet die linke Seite jedoch nicht genügend aus, sodass die Zeichen links sich kaum noch vom Hintergrund unterscheiden. Lässt man eine Textdetekion auf Bild 3.38(d) laufen, so sollte diese trotz der perspektivischen Verzerrung den Text erfolgreich erkennen können. Es ist ebenso wünschenswert, dass auch der kleine Text auf der Tafel in der Mitte des Bildes als solcher erkannt wird.

Die Projektgruppe hat sich entschieden, die Textdetektion für Pick by Feel nicht weiter zu verfolgen. Diese Entscheidung basiert auf folgenden Gründen:

Der Aufwand zur Implementierung einer globalen Textdetektion steht zu keinem Nutzen für die Objekterkennung. Da die Textdetektion in natürlichen Szenen immer noch ein aktuelles Forschungsthema in der Informatik ist, welche unter den gegebenen Einschränkungen und Einflüssen nur unbekannt „gut“ ist, hat die Projektgruppe Bedenken, zu viele Ressourcen in eine möglicherweise nicht funktionierende Textdetektion zu investieren.

Ebenso kommt zu einer Textdetektion noch eine Zeichenerkennung, die zwar ähnlich über maschinelles Lernen realisiert werden kann (vgl. [15]), aber weiteren Implementierungsaufwand erfordert.

Schlussendlich ist eine Textdetektion für das Anwendungsgebiet der Projektgruppe nicht unbedingt erforderlich. Da wir den zu kommissionierenden Artikel stets kennen, brauchen wir keine Textdetektion, die uns aus einem Bild einen maschinenlesbaren Text liefert, sondern nur eine Überprüfung, ob das aufgenommende Bild zum kommissionierenden Artikel passt. Für dieses Anwendungsgebiet eignen sich Klassifizierung und Image Retrieval weitaus mehr.

3.3.4 Barcodedetektion

Barcodes, auch Strichcode oder Streifencode genannt, dienen der Darstellung von Daten für optische Lesegeräte. Bei den Lesegeräten handelt es sich in der Regel um Kameras oder spezielle Scanner. Die Daten beschreiben meist das Objekt auf dem sich der Barcode befindet. Eines der bekanntesten Beispiele ist ein Artikel im Einzelhandel, dessen Barcode im Kassenbereich gescannt wird und so der korrekte Betrag abgerechnet werden kann. Abbildung 3.39 zeigt einen solchen Barcode auf der Rückseite eines Buches.

Abbildung 3.39: Barcode mit der ISBN und der für Deutschland geltenden Buchpreisbindung auf der Rückseite des Buches „Alles über Wikipedia und...“ (Quelle: http://de.wikipedia.org/wiki/Strichcode)

Typen von Barcodes

Es gibt viele verschiedene Typen von Barcodes, die sich in der Art und Weise, in der die enthaltenden Daten kodiert, sind unterscheiden. Bei Strich- bzw. Barcodes unterscheidet sich bei den unterschiedlichen Typen die Informationsdichte, also die mögliche Anzahl an kodierbaren Ziffern und die Definition, welche Teile des Codes eine Ziffer kodieren. Auch Höhe und Breite der Codes können sich unterscheiden, so werden besonders hohe Codes genutzt, um das Objekt in jeder möglichen Lageposition scannen zu können.

Im Gegensatz zu Texten lassen sich Barcodes auch in verschiedenen Orientierungen erkennen, da diese über spezielle Muster definiert werden oder der Barcode, bspw. der EAN-13 Code, invertiert codiert ist.

Der EAN-13 Barcode

Abbildung 3.40: EAN13-Barcode (Quelle: http://de.wikipedia.org/wiki/European_Article_Number)

Einer der bekanntesten Verteter der Barcodes ist der in Abbildung 3.40 gezeigte EAN-13 bzw. GTIN-13 Barcode. EAN steht für European Article Number und bezeichnet damit bereits den Hauptverwendungszweck des Barcodes, nämlich der Identifikation von Artikeln

im europäischen Einzelhandel. Dabei stellt EAN-13 die veraltete Bezeichnung dar. Heute ist der Strichcode Teil der GTIN (Global Trade Item Number) Familie. Der Code umfasst 13 Stellen, die in Form eines Linienmusters dargestellt werden und in ihrer Breite variieren. Die Bitmuster des EAN-13 Barcodes sind so definiert, dass jeder Barcode auch um 180 Grad gedreht gelesen und dekodiert werden kann ohne die dekodierte Ziffer zu verändern.

Aufbau der Ziffern Die Ziffern des Barcodes sind in drei Bereiche aufgeteilt. Die ersten zwei oder drei bzw. sieben, acht oder neun Ziffern werden hierbei durch die GS1-Gruppe vergeben. Die GS1-Gruppe vergibt dabei Zifferkombinationen, die ihren jeweiligen lokalen Niederlassungen zugeordnet sind [9]. Auf die GS1 Ziffern folgen die durch den jeweiligen Hersteller festgelegten Nummern, also die eigentlichen Artikelnummern. Die letzte Ziffer bildet eine Prüfsumme, die nach Dekodierung eine Prüfung des Ergebnis ermöglicht. Die einzelnen Ziffern des 13-stelligen Codes können jeweils die Ziffern 0-9 kodieren. Aufgrund der Prüfziffer an letzter Stelle können also effektiv 12 Ziffern kodiert werden, wobei einige fest durch die GS1-Gruppe vergeben werden.

Aufbau des Linienmusters Jede Ziffer wird durch insgesamt vier Balken, also zwei schwarze und zwei weiße senkrechte, dargestellt. Dabei wird eine Ziffer aus jeweils sieben schwarzen bzw. weißen Linien aufgebaut. Zusätzlich zu den codierten Ziffern gibt es drei Begrenzungsmuster, die Scannern eine Differenzierung der einzelnen Bereiche des Barcodes und das Festlegen der Linienbreite ermöglichen. Die Begrenzungsmuster bestehen aus zwei äußeren Mustern, die aus drei Linien ein 101 Muster bilden und einem in der Mitte platzierten 01010 Muster.

Diese Begrenzungsmuster trennen den Barcode in einen linken und einen rechten Bereich. Jeder Bereich codiert sechs Ziffern mit je sieben Linien, also insgesamt 42 Linien.

Kodierung der Ziffern In den, durch die Begrenzungsmuster separierten, Blöcken werden die sechs einzelnen Ziffern durch jeweils zwei weiße und zwei schwarze Balken kodiert. Dabei können die einzelnen Balken unterschiedliche Breiten besitzen. Die 1. Ziffer des Barcodes, im Beispiel in Abbildung 3.40 die Ziffer 0, wird durch die Parität der Ziffern im linken Block des Barcodes berechnet. Im Beispiel in Abbildung 3.40 also durch die Zahlen an den Stellen 1-6. Diese entsprechen wiederum einer, von der GS1 vorgegebenen, Parität, was über eine tabellarische Zuordnung eine Ableitung der Ziffer 0 ermöglicht.

Zur Dekodierung eines Barcodes wird zunächst das linke Begrenzungsmuster gesucht. Nach Finden des linken Begrenzungsmusters kann die erste Ziffer dekodiert werden. Das sieben Stellen lange Bitmuster einer Ziffer kodiert dabei die jeweilige Ziffer anhand einer festgelegten Zuordnungstabelle. Anhand einer solchen Tabelle lassen sich also aus den sechs Bitmustern im linken Bereich die jeweiligen Ziffern bestimmen. Dabei legen die Tabellen noch weitere Informationen für die Ziffern im linken Block fest. Jeder Ziffer des linken Blocks wird eine Parität zugewiesen. Anhand der Parität aller Ziffern im linken Block lässt

sich die erste Ziffer des Barcodes dekodieren. Auch diese Zuordnung findet sich in einer durch die GS1 veröffentlichten Tabelle[9]. Mit diesem Verfahren werden, nach Passieren des zentralen Begrenzungsmusters, auch die Ziffern im rechten Block dekodiert.

Für die so erhaltene Ziffernfolge kann nun noch eine Prüfsumme berechnet und mit der letzten Ziffer verglichen werden. Dazu werden alle Ziffern x_n, beginnend mit der vorletzten Ziffer x_{n-1}, alternierend mit 3 und 1 multipliziert und abschließend die Summe der Produkte gebildet:

$$x_{n-1} * 3 + x_{n-2} * 1 + \ldots + x_{n-5} \ldots$$

Die Prüfziffer wird aus der Differenz der Summe zum nächsten Vielfachen von 10 gebildet. Stimmen Prüfziffer und berechnete Prüfziffer überein, wurde der Barcode korrekt dekodiert.

Erkennung eines EAN-13 Barcodes

Das folgende Kapitel beschreibt eine mögliche Variante, um in einem aufgenommenen Bild einen Barcode zu identifizieren, freizustellen und, im Falle eines EAN-13 Barcodes, zu dekodieren. Für diese Problemstellung gibt es viele verschiedene Ansätze, wobei hier allerdings nur auf die in [14] vorgestellte Variante eingegangen wird.

Vorverarbeitung Zur effizienten Anwendung des Barcode-Detektierungsverfahrens wird das Bild, in dem der Barcode erkannt werden soll, vorverarbeitet. Diese Bearbeitung beschränkt sich auf die Konvertierung in Graustufen und, falls nötig, eine Beschneidung des Bildes. Das Verfahren bearbeitet das Bild in Blöcken von 32x32 Pixeln, deshalb wird das Bild gleichmäßig auf allen Seiten beschnitten, bis eine Aufteilung in die genannte Blockgröße möglich ist.

Lokalisierung des Barcodes Zur Erkennung des eigentlichen Barcodes innerhalb des Bildes, wird dieses in Blöcke aufgeteilt und die einzelnen Blöcke so behandelt, dass eine Identifizierung der enthaltenen Strukturen möglich wird. Die verschiedenen Zwischenschritte für einen Block lassen sich an Abbildung 3.41 nachvollziehen.

Um eine Erkennung der Strukturen zu ermöglichen, werden zunächst die einzelnen Grauwerte der Pixel in Binärwerte konvertiert. Dies geschieht mit *Otsus thresholding technique*[8]. Dabei wird das normalisierte, bimodale Histogramm der Grauwerte der im Bild enthaltenen Pixel untersucht und ein optimaler Schwellwert berechnet. Dieser Schwellwert wird so berechnet, dass eine minimale Intraklassenvarianz erreicht wird, die Klassen also so kompakt wie möglich gehalten werden.

Die so erhaltenen Binärdaten werden skelettiert[12]. Dieses Verfahren verkleinert zusammenhängende Flächen soweit, bis sie nur noch durch eine minimale Fläche repräsentiert werden. Eine mögliche Definition der Skelettierung ist laut Blum das gleichzeitige Anzünden einer trockenen Grasfläche an allen Kanten. Die Stellen, an denen mindestens zwei

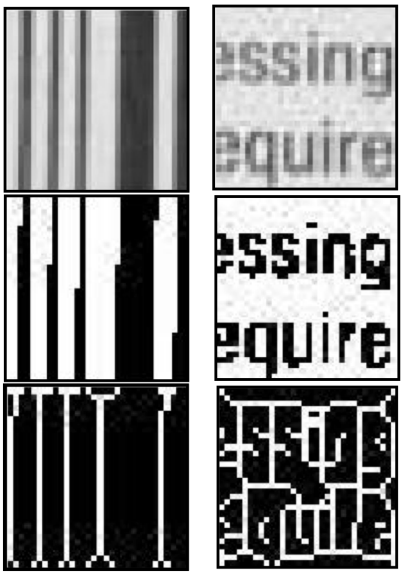


Abbildung 3.41: Ausgeschnittene, in Binärdaten konvertierte und skelettierte Blöcke (Quelle: [14])

Brandfronten aufeinandertreffen, bilden dann das Skelett. Das Bilden des Skeletts kann hierbei auf verschiedene Weisen erfolgen. So wäre eine Möglichkeit das gleichmäßige Entfernen von am Rand gelegenen Pixeln, bis die verbleibenden Pixel einzelne Linien bilden.

Aus den so gewonnenen Skeletten wird nun die dominierende Orientierung berechnet. Ziel ist es hierbei, parallele Linienmuster, nämlich die Striche des Barcodes, zu identifizieren. Dazu werden die einzelnen Regionen, die durch die Skelettierung innerhalb eines Blocks freigestellt wurden, benannt. Für jede Region wird der Winkel zur X-Achse berechnet und der Durchschnitt über alle Regionen gebildet. Ist die Orientierung aller Regionen innerhalb eines Blockes ähnlich, kann angenommen werden, dass der Block ein paralleles Linienmuster enthält. Die Orientierung des Linienmusters entspricht dabei dem Durchschnitt der Orientierung aller Regionen.

Wird eine Gruppe von Blöcken mit parallelen Linienmuster in annähernd gleicher Richtung gefunden, kann von einem Barcode im jeweiligen Bildbereich ausgegangen werden. Dieser Bildbereich wird extrahiert und im nächsten Schritt kann dann der Barcode dekodiert werden.

Dekodierung des Barcodes Über den gefundenen Bildbereich wird eine horizontale Datenreihe gebildet. Eine solche Reihe ist in Abbildung 3.42 dargestellt. Da zur Dekodierung eine Unterscheidung in schwarze und weiße Bereiche ausreicht, kann hier mit den vorhandenen / vorher berechneten Binärbildern gearbeitet werden.

Um aus der binären Datenreihe eine Repräsentation der im Barcode vorhandenen Striche zu erhalten, wird die Laufweite einer einzelnen Linie berechnet. Hierbei ermöglichen die

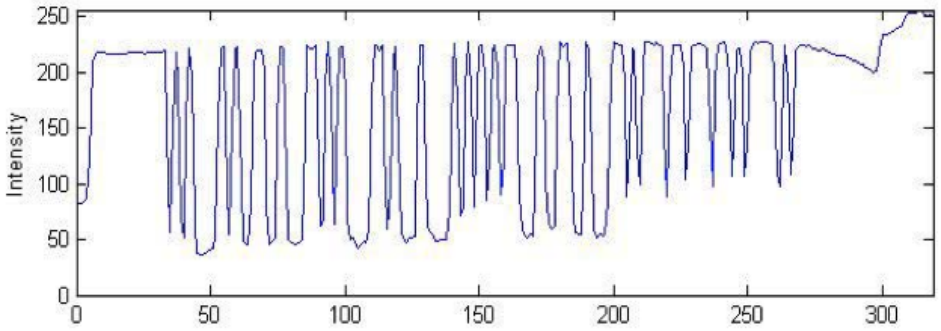


Abbildung 3.42: Datenreihe über extrahiertem Bildbereich mit Barcode (Quelle: [14])

Begrenzungsmuster, die einzelne Linien in festen Mustern bereitstellen, die Identifikation der Breite einer einzelnen Linie.

Die so erhaltene binäre Darstellung des Barcodes, kann nun nach dem in Kapitel 3.3.4 beschriebenen Verfahren zur Ziffernfolge dekodiert werden.

Einsatz in der Projektgruppe

Die Erkennung und Dekodierung von Barcodes ist insgesamt weniger komplex, als bspw. die in Kapitel 3.3.3 vorgestellte Textdetektion. Die Barcodes sind für eine schnelle und effiziente Erkennung konzipiert und heute in Smartphones verwendete Kamerasensoren sind hochauflösend genug, um parallele Linienmuster mit dem in diesem Kapitel beschriebenen Verfahren zu erkennen.

Auch die in Kapitel 3.3.3 beschriebenen Problematiken wie farbige Texte, schlechte Beleuchtung oder Verzerrung sind weniger kritisch. Barcodes liegen ohnehin meist zweifarbig vor und heutige Schwellwertverfahren können diese Unterscheidung sehr gut vornehmen.

Die Projektgruppe hat die Barcodedetektion zur Erkennung des gehaltenen Artikels deshalb aus folgenden Gründen verwendet:

- Heutzutage sind fast alle, im Einzelhandel erhältlichen, Artikel mit EAN-13 bzw. GTIN-13 Barcodes ausgestattet.
- Die Detektion von Barcodes ist weit verbreitet und erfordert verhältnismäßig wenig Rechenleistung und kann effizient durchgeführt werden.
- Eine erfolgreiche Dekodierung eines Barcodes ermöglicht eine eindeutige Identifikation eines Artikels und macht weitere Analysen überflüssig. Sollte eine solche Dekodierung also bereits zu Beginn der Analyse erfolgreich stattgefunden haben, kann der weitere Analysevorgang abgebrochen und eine Erfolgsmeldung zurückgegeben werden.

Aufgrund der hohen Zuverlässigkeit von Barcodedetektion hat sich die Projektgruppe zusätzlich dazu entschlossen bei Schwierigkeiten während des Erkennungsvorgangs eine

Scan-Aufforderung an den Kommissionierer zu übermitteln. Diese fordert den Kommissionierer nach fünf gescheiterten Erkennungsversuchen auf, einen auf dem Artikel angebrachten Barcode dem Smartphone zuzuwenden. Dadurch steigt die Chance einer erfolgreichen Barcodedetektion und damit einer erfolgreichen Identifizierung des Artikels.

Die verwendete Technologie zur Barcodedetektion wird in Kapitel 5.2.4 vorgestellt und eine ausführliche Darstellung der Implementierung befindet sich in Kapitel 6.1.3.

KAPITEL 4

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das zu entwickelnde System beschrieben, sowie auf die geforderten Produktfunktionalitäten eingegangen und herausgestellt, wie diese umgesetzt werden. Außerdem werden die Betriebsbedingungen benannt. Es wird für jede unterstützte Funktion des Systems beschrieben, wie sie im Anwendungsfall abläuft und unter welchen Bedingungen sie aufgerufen wird. Dazu werden zunächst im Abschnitt 4.1 die Anforderungen an das System erläutert, die nötigen Betriebsbedingungen werden in Abschnitt 4.2 genannt. Die Anforderungen werden durch Anwendungsfälle konkretisiert, diese werden im Abschnitt 4.3 erläutert. In Abschnitt 4.4 folgen weitere Produktfunktionen, die keinem Anwendungsfall direkt zugeordnet werden können.

4.1 Anforderungen

Ziel ist die Entwicklung eines Hilfssystems für die Arbeit des Kommissionierers beim Zusammenstellen (Picken) von Aufträgen. Die Hauptschnittstelle des Systems zum Kommissionierer ist hierbei eine Smartwatch, die der Anzeige von Picklisten, der Rückmeldung über erfolgte Picks sowie eine Eingabeschnittstelle für Bedieneingaben des Kommissionierers vorsieht. Ein weiterer Hauptbestandteil ist ein Smartphone, welches zum einen ein Kamerabild für die Bildverarbeitung liefert, als auch als Bindeglied zwischen der Smartwatch und dem Steuer- bzw. Auswerteserver dient. Der Server sendet Kommissionieranweisungen und Statusmeldungen über das Smartphone bzw. die Smartwatch an den Kommissionierer und wertet das Kamerabild des Smartphones aus, um eine automatische Erkennung der gepickten Artikel zu ermöglichen.

4.1.1 Muss-Kriterien

Erkennung und Identifizierung der gegriffenen Objekte. Zunächst gilt es zu erkennen, wann ein Greifvorgang stattfindet. Dies könnte bspw. durch die Trennung von Hand und Objekt mit Hilfe von Segmentierung stattfinden. Schlägt eine Handerkennung fehl, kann alternativ durch die Auswertung der Sensordaten des Smartphones (Gyro-Sensoren, Beschleunigungssensoren) versucht werden den Greifvorgang zu detektieren. Durch die erhaltenen Werte der Sensoren kann auf dem Smartphone klassifiziert werden, ob der Kommissionierer geht oder steht. So würde die Erkennung eines Objekts z. B. erst starten, wenn festgestellt wurde, dass der Kommissionierer zuerst gegangen und anschließend stehen geblieben ist. Das gegriffene Objekt muss identifiziert werden, indem Barcodes in der Nähe der Hand erkannt und dekodiert werden. Sofern dies nicht möglich ist (z. B. da der Barcode nicht lesbar ist oder kein Barcode vorhanden ist) gilt es das Objekt mit einer Reihe von Referenzobjekten zu vergleichen und das ähnlichste Objekt als Treffer auszugeben. Dies kann mit einer mindestens zu erreichenden Anzahl von gleichen Merkmalen nach unten abgegrenzt werden, sodass eine ausreichende Ähnlichkeit vorhanden sein muss, um als Treffer gewertet zu werden. Im Anschluss daran wird das Objekt noch mit Vergleichsbildern bzw. Merkmalen aus einer Datenbank abgeglichen, um somit die Objektkategorie zu ermitteln. Diese Kategorie wird mit der Kategorie des zu greifenden Artikels verglichen, um somit eine Aussage über die Richtigkeit des Picks treffen zu können. Falls eine Erkennung nicht möglich ist, wird dem Kommissionierer dies mitgeteilt, es besteht dann die Möglichkeit entweder einen Barcode manuell vor die Kamera zu halten und damit scannen zu lassen, oder den Pick manuell als richtig zu bestätigen. Aufgrund der Evaluationsergebnisse aus Kapitel 7.1 hat es die Objektklassifikation allerdings nicht geschafft ein Bestandteil des Systems zu werden.

Kommunikation. Um die verschiedenen Kompomenten des Systems zu verbinden, ist eine Kommunikationsstruktur notwendig. Vom Smartphone muss ein Videostream sowie Einzelbilder von der Kamera an den Server geschickt werden. Der Server sendet Antworten über den Status des Picks an das Smartphone, z. B. richtiger Pick, falscher Pick. Ebenso können weitere Nachrichten an das Smartphone gesendet werden, wie Informationen über den nächsten Pick oder weitere Anweisungen an den Kommissionierer. Das Smartphone sendet Benutzereingaben und Statusnachrichten an den Server, wie die manuelle Bestätigung eines Picks oder die Unterbrechung der Kommissionierung in Pausen.

Auch zwischen Smartphone und Smartwatch findet Kommunikation statt. Nachrichten vom Server werden über das Smartphone an die Smartwatch weitergeleitet, dort werden sie entsprechend verarbeitet und angezeigt. Ebenso wird die Rückmeldung des Servers über den erfolgten Pick an die Smartwatch weitergeleitet, dort erfolgt die Benachrichtigung des Kommissioniers durch Vibration. Einfache Eingaben wie die manuelle Bestätigung eines

Picks werden von der Smartwatch über das Smartphone an den Server geleitet.

Hardware. Es muss eine Halterung entwickelt werden, um das Smartphone an einer geeigneten Position am Kommissionierer zu fixieren, ohne diesen in seiner Tätigkeit zu beeinträchtigen. Gleichzeitig muss gewährleistet sein, dass die Kamera des Smartphones den Arbeitsbereich des Kommissionierers aufzeichnen kann.

Pickliste. Es muss eine Möglichkeit geschaffen werden um Picklisten zu erstellen, die Inhalte der Picklisten an die Mobilgeräte (Smartphone/Smartwatch) zu senden sowie die erfassten Artikel mit der Pickliste abzugleichen und den Status des Picks in der Pickliste zu speichern.

Testdaten. Ein Satz an Testdaten muss erstellt werden. Die Testdaten bestehen aus Objekten verschiedener, möglichst unterschiedlicher Kategorien, um ein breit gefächertes Produktspektrum zu simulieren. Anhand der Testdaten wird das System evaluiert, parametrisiert und verifiziert. Ebenfalls werden die Grenzen des Systems mit diesem Testdatensatz ermittelt.

Demonstration des Systems. Das System wird an einem Testcase demonstriert, welcher an die im Kapitel 8.1 definierte Anwendungsumgebung angelehnt ist.

4.1.2 Soll-Kriterien

Verhalten bei einem Kommunikationsfehler. Da das System aus mehreren Komponenten besteht, welche mit unterschiedlichen Funktechnologien verbunden sind, besteht immer die Möglichkeit eines Verbindungsabbruchs. Eine solche Unterbrechung sollte erkannt und dem Benutzer mitgeteilt werden, ebenso soll die Unterbrechung auf dem Server in einem Protokoll vermerkt werden.

Verbesserung der Bildverarbeitung. Es können Filter (bspw. Schärfen, Kontrasterhöhung) eingesetzt werden, um bei der Bildverarbeitung bessere Resultate zu erzielen.

Mehrbenutzer-Fähigkeit. Da das System später in einem Lager eingesetzt werden soll, in dem mutmaßlich mehr als ein Kommissionierer tätig ist, soll die Möglichkeit zur Mehrbenutzerfähigkeit vorgesehen werden. Eine konkrete Umsetzung einer Mehrbenutzerumgebung mit Anmeldung und Verwaltung mehrerer Nutzer ist jedoch nicht erforderlich.

Test der Funktionalität. Das System soll unter realitätsnahen Bedingungen (siehe Kapitel 4.2) getestet werden. Ebenso sollen die in Kapitel 8.2 genannten Testfälle getestet

werden. Dies schließt sowohl Tests im vorgesehenen Ablauf als auch Tests mit Fehlerfällen mit ein.

4.1.3 Kann-Kriterien

Implementierung einer Mehrbenutzer-Umgebung. Die Implementierung einer Mehrbenutzerumgebung inklusive einer Benutzerverwaltung und Anmeldung ist möglich. Dies ist jedoch nicht zwingend notwendig, da oftmals eine Einbindung schon vorhandener Benutzerkonten gewünscht ist.

Automatische Fehlerbehebung. Ebenfalls wünschenswert wäre die automatische Erkennung und Behebung von Verbindungsfehlern.

Energiesparmaßnahmen. Bisher nicht genau abschätzbar ist das Potential von Energieeinsparungen und Laufzeitverlängerungen. Dies kann untersucht werden, mögliche Energiespar-Mechanismen, z. B. Abschalten der Kamera bei Nichtgebrauch, Drosselung der Datenrate können noch implementiert werden.

Weitere Erkennungsmethoden. Eine weitere Möglichkeit zur Identifizierung des gegriffenen Objektes ist die Erkennung von Text auf dem Objekt. Das Problem der Textdetektion in natürlichen Umgebungen ist jedoch im Vergleich zu den anderen, gewählten Verfahren ungleich schwieriger, und bietet somit geringere Erfolgsaussichten. Aus diesem Grund wurde entschieden, dieses Verfahren im Rahmen der Projektgruppe nicht zu implementieren.

Automatische Parametrisierung. Im Rahmen der Evaluation und während der Tests des Systems soll der Benutzer angeben können, ob die Antwort des Systems auf einen Pick richtig oder falsch war. Mit Hilfe dieser Rückmeldungen können Fehler in der Objektklassifikation erkannt werden, außerdem kann die Rückmeldung zur weiteren Parametrisierung des Systems genutzt werden.

Automatisches Lernen. Die Rückmeldungen des Benutzers zu den Antworten des Systems können automatisch ausgewertet werden, um die Identifikationsalgorithmen durch interaktives Lernen während des laufenden Betriebes zu verbessern.

Auslagerung der Vorverarbeitung. Der grundlegende Entwurf des Systems sieht vor, möglichst alle Verarbeitungsschritte auf dem Server auszuführen, damit die verwendete Peripherie (Smartphone und Smartwatch) möglichst gegen Geräte mit anderen Architekturen und Betriebssystemen austauschbar ist. Es ist jedoch auch denkbar, vorbereitende Erkennungsschritte (wie Barcode-Detektion oder Bewegungserkennung für Energiespar-

mechanismen) auf das Smartphone auszulagern, um die Serverlast zu verringern, und die vorhandene Rechenkapazität des Smartphones besser auszunutzen.

4.2 Betriebsbedingungen

Um die vorgestellten Anforderungen erfüllen zu können, werden einige Bedingungen an die Betriebsumgebung gestellt. Diese werden im Folgenden vorgestellt. Für den einwandfreien Einsatz des Systems wird eine Umgebung vorausgesetzt, in der schlechte Lichtverhältnisse durch ausreichende und gleichmäßige Beleuchtung der Hallen vermieden und die Einflüsse zur Blendung der Kameralinse minimiert werden. Weiterhin muss eine unterbrechungsfreie drahtlose Datenverbindung (WLAN) mit dem Server ermöglicht werden. Eine zusätzliche Besonderheit in diesem Umfeld bildet die chaotische Lagerhaltung. Sie wird für eine optimierte Objektklassifikation vorausgesetzt und bietet die Grundlage der Funktionalitäten des Kommissionierungssystems, welches in der Praxis zur Anwendung kommt. Zudem werden für die Erkennung der Produkte Beispielaufnahmen benötigt. Diese können als eigene Aufnahmen oder auch in Form externer Datenbanken vorliegen.

4.3 Anwendungsfälle

Die Anwendungsfälle sind in Abbildung 4.1 grafisch dargestellt und werden im Folgenden näher erläutert. Die einzelnen Funktionen werden dabei nach folgendem Schema durchnummeriert: Die Darstellung erfolgt nach dem Schema "/F.../". Dabei ist der Buchstabe F die Kennzeichnung für eine Produktfunktion und die darauf folgende Zahlenkombination stellt die Funktionsnummer dar. Mit einem optionalen Zusatz S oder K wird dabei auf die Soll- bzw. Kann-Kriterien aus den Zielbestimmungen verwiesen. Ohne Zusatz beschriebene Funktionen sind daher Muss-Kriterien.

Der zeitliche Ablauf und Zusammenhang zwischen den einzelnen Anwendungsfällen wird im Ablaufdiagramm 7.1 des Pflichtenheftes erläutert.

/**F001K**/ *Benutzer anmelden*: Der Benutzer kann sich mit einer Kennung anmelden und wird so im System zugeordnet. Bei erfolgreicher Anmeldung wird eine Pickliste geladen und der erste Pick kann erkannt werden. Bei fehlgeschlagener Anmeldung wird der Benutzer darüber informiert.

/**F002K**/ *Benutzer abmelden*: Der angemeldete Benutzer kann sich aus dem System abmelden, wenn seine Arbeit getan ist.

/**F003**/ *Erkennung pausieren*: Der Benutzer pausiert die Erkennung. Die Verbindung zum Server bleibt bestehen, aber die Übertragung bzw. Erkennung wird unterbrochen.

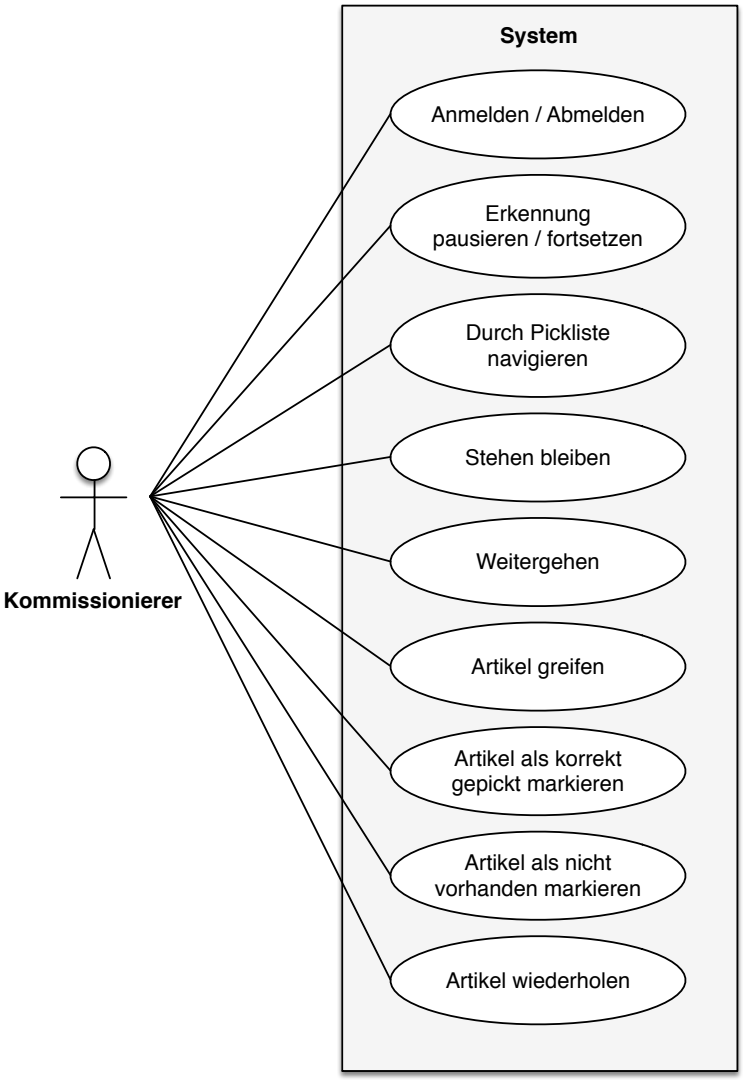


Abbildung 4.1: Anwendungsfalldiagramm

/**F004**/ *Erkennung fortsetzen*: Der Benutzer setzt eine pausierte Sitzung fort. Die Erkennung wird wieder aktiviert.

/**F005K**/ *Durch Pickliste navigieren*: Der Benutzer kann durch die Pickliste bis zu dem aktuellen Pick vor- und zurücknavigieren, um auch noch zu einem späteren Zeitpunkt fehlerhafte Picks zu korrigieren.

/**F006**/ *Bewegungserkennung*: Bleibt der Benutzer stehen, so beginnt die Erkennungsphase des aktuellen Picks. Dabei ist die Voraussetzung, dass zuvor ein "Gehen" vom Smartphone klassifiziert worden ist und die Erkennungsphase noch nicht läuft. Das Stehenbleiben des Benutzers ist dabei Ausgangspunkt für das Greifen eines Artikels und verhindert damit die dauerhafte Übertragung des Kamerabildes an den Server. Wird ein Artikel in irgendeiner Form erkannt, so stoppt die Übertragung der Bilder

an den Server. Erst durch eine erneute Detektion des Wechsels zwischen gehen und stehen wird die nächste Erkennungsphase gestartet.

/**F007**/ *Artikel greifen*: Ein wichtiger Zwischenschritt in einer Erkennungsphase ist das Greifen des entsprechenden Artikels vom Benutzer. Erst nach der Erkennung eines Greifvorgangs versucht das System, den Artikel zu klassifizieren bzw. zu scannen.

/**F008**/ *Artikel als korrekt gepickt markieren*: Ist ein Artikel vom System nicht erkennbar und der Barcode ist nicht lesbar oder nicht vorhanden, so kann der Benutzer diesen auf der Smartwatch manuell als korrekt gepickt bestätigen.

/**F009**/ *Artikel wiederholen*: Wurde ein Artikel laut System erfolgreich erkannt, aber der Benutzer stellt fest, dass er einen falschen Artikel gepickt hat, so kann er diesen auf der Smartwatch zu einem späteren Zeitpunkt zurücksetzen und die Erkennung wiederholen.

/**F010K**/ *Artikel als nicht vorhanden markieren*: Ist ein Artikel für den Benutzer nicht auffindbar, so kann er diesen manuell über Knopfdruck auf der Smartwatch überspringen. Der Artikel wird entsprechend markiert und an das Ende der Pickliste verschoben.

4.4 Weitere Produktfunktionen

/**F101S**/ *Signal bei Verbindungsabbruch*: Der Benutzer wird über einen Verbindungsabbruch vom Server oder der Smartwatch durch ein entsprechendes Signal auf dem Smartphone informiert.

/**F102S**/ *Verbindung wieder aufnehmen*: Der Benutzer kann auf dem Smartphone eine fehlgeschlagene Verbindung zum Server oder der Smartwatch wieder aufnehmen.

/**F103**/ *Anzeige der Artikelbezeichnung*: Der Benutzer erhält auf der Smartwatch eine identifizierende Beschreibung über den zu kommissionierenden Artikel wie z. B. den Namen.

/**F104**/ *Anzeige des Lagerplatzes*: Der Benutzer erhält auf der Smartwatch detaillierte Informationen über den Lagerplatz des zu kommissionierenden Artikels.

/**F105**/ *Anzeige/Signal bei erfolgreicher Erkennung*: Der Benutzer wird über ein entsprechendes Signal und der Anzeige auf der Smartwatch bei erfolgreicher Erkennung eines Artikels informiert und erhält ein positives Feedback.

/**F106**/ *Anzeige/Signal bei Erkennung eines falschen Artikels*: Der Benutzer wird über ein entsprechendes Signal und der Anzeige auf der Smartwatch bei Erkennung eines Artikels informiert, welcher nicht dem auf der Pickliste stehendem Pick entspricht.

/**F107**/ *Anzeige/Signal bei fehlgeschlagener Erkennung*: Der Benutzer wird bei einer fehlerhaften oder fehlgeschlagenen Erkennung über ein entsprechendes Signal und der Anzeige auf der Smartwatch informiert und kann den Artikel in seiner Hand neu ausrichten, bspw. in dem er den Barcode des Artikels explizit versucht einzuscannen.

KAPITEL 5

Systementwurf und Technologien

Dieses Kapitel stellt den Entwurf sowie einen Überblick über die Architektur des entwickelten Systems vor. Des weiteren werden die Technologien erläutert, die im Rahmen der Projektgruppe für die Entwicklung der einzelnen Software-Komponenten zum Einsatz gekommen sind.

5.1 Systementwurf

Das von der Projektgruppe entwickelte System besteht aus drei eigenständigen, miteinander verbundenen Software-Komponenten. Die einzelnen Komponenten und ihr Zusammenhang sind in Abbildung 5.1 dargestellt.

Die Haupt-Schnittstelle zum Bediener, dem Kommissionierer, stellt die Smartwatch dar. Auf dieser läuft ein Programm (*Picker-Watch* in Abbildung 5.1), welches Informationen bereitstellt und Eingaben vom Bediener entgegen nimmt. Die Smartwatch kommuniziert mit einem Smartphone, welches der Kommissionierer immer mit sich trägt.

Das Smartphone hat zwei Aufgaben: zum einen verbindet es die Smartwatch mit dem steuernden Server, zum anderen liefert die Kamera des Smartphones die zur Identifizierung der Artikel nötigen Bilder. Dazu läuft auf dem Smartphone eine App (*Picker*), welche diese Aufgaben ausführt.

Die Steuerung des Prozesses sowie die Erkennung der Artikel wird auf einem Server durchgeführt. Der Server ist per WLAN mit dem Smartphone verbunden und tauscht mit diesem Nachrichten aus. Auf dem Server läuft ein Programm, welches die Benutzer verwaltet, die Kommunikation mit den anderen Komponenten steuert sowie die Erkennung der Artikel ausführt (*Picker-Server*).

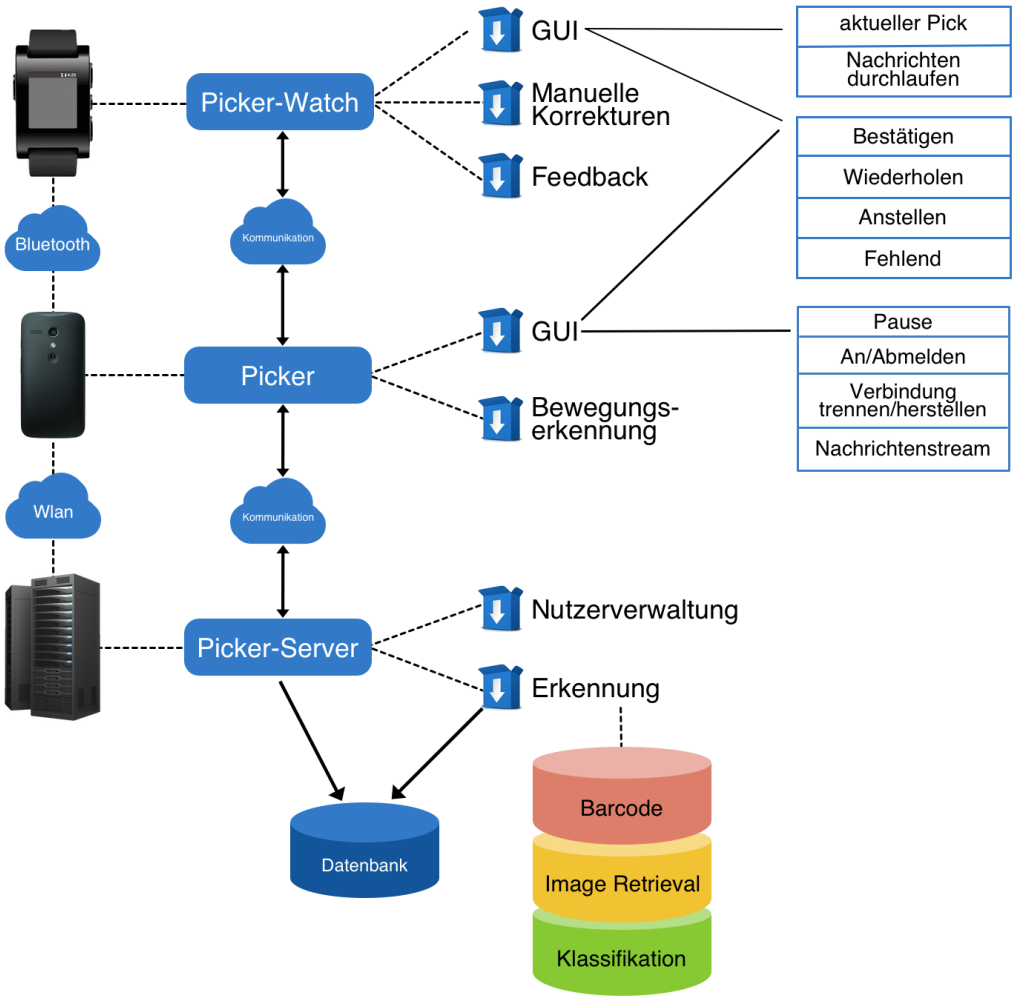


Abbildung 5.1: Eingesetzte Hardware-Komponenten des Systems, zu entwickelnde Software sowie von dieser verwendeten Techniken (zur Kommunikation und Realisierung der Funktionalität)

5.2 Eingesetzte Technologien

Hier werden die zur Realisierung des Systems eingesetzten Bibliotheken, Frameworks usw. vorgestellt. Die konkrete Implementierung des Systems sowie die Einbindung der Technologien wird in Kapitel 6 erläutert.

5.2.1 OpenCV

Ein wesentlicher Teil für die Erfüllung der Anforderungen an die Projektgruppe ist die Umsetzung der Objekterkennung auf Bildern. Hierfür wird im Rahmen der Projektgruppe auf die Open-Source Bibliothek **OpenCV** (***Open Source Computer Vision Library***) [3] zurückgegriffen.

Heute wird OpenCV hauptsächlich von **Willow Garage** weitergepflegt. Als freie Open-Source Bibliothek steht sie unter den Bedingungen der BSD-Lizenz und ist somit frei zugänglich für die akademische sowie auch kommerzielle Nutzung.

Grundlagen und Wahl der Programmiersprache

OpenCV ist plattformübergreifend einsetzbar und unterstützt daher die Betriebssysteme Windows, Linux, Mac OS, iOS und Android. Es sind zudem Schnittstellen für die Programmiersprachen C++, C, Python und Java vorhanden. OpenCV beinhaltet eine Bibliothek mit Algorithmen für maschinelles Lernen. So wird maschinelles Lernen mit einem reichen Funktionsumfang wie Lernen eines Entscheidungsbaumes, Nächste-Nachbarn-Klassifikation, dem kmeans-Algoritmus, SVM etc. unterstützt. Für die Bildverarbeitung stehen fertige Algorithmen zur Verfügung, die z. B. für die Identifikation und Verfolgung von Objekten, das Klassifizieren von Bewegungen in Videos, das Extrahieren von Objekten als 3D-Modelle oder für das Finden von ähnlichen Bildern in einer Datenbank genutzt werden können.

Im Rahmen der Projektgruppe ist die Wahl der Programmiersprache auf C++ gefallen. Begründet wird dies dadurch, dass OpenCV, direkt in dieser Sprache entwickelt wurde, und zu allen weiteren Sprachmöglichkeiten jeweils nur Wrapper vorhanden sind. Hierdurch erhoffte sich die Projektgruppe die optimale Leistung aus der zur Verfügung stehenden Bibliothek schöpfen zu können. Die damit einhergehende Reduzierung der Berechnungszeit wurde von der Projektgruppe als Hauptleistungsmerkmal des Endproduktes ausgemacht. Dies ist auch einer der Gründe, weswegen die Auswertung der Objekterkennung nicht direkt auf dem Smartphone, sondern auf dem Server durchgeführt wurde.

5.2.2 Android-Plattform

Die Entwicklung auf der Android-Plattform erfolgt im Verlauf der Projektgruppe für das Smartphone und die Smartwatch Komponenten des Kommissioniersystems **Pick-by-feel**. Nachfolgend werden Informationen zur genutzten Android Version und den damit verbundenen Neuerungen beschrieben, um dann den Aufbau einer Android Applikation aufzuzeigen. Im darauffolgenden Unterkapitel 5.2.2 wird auf die Projektstruktur einer solchen Applikation eingegangen und die Entwicklungsumgebung vorgestellt. Der Lebenszyklus einer Android-App wird in Unterkapitel 5.2.2 erläutert, bevor zum Abschluss das Thema Smartwatches aufgegriffen wird.

Grundlagen

Die Mindestanforderungen des Projekts verlangen für das mobile Betriebssystem eine Android Version ab 4.0. Eine typische Android Anwendung unterteilt sich nach den Richtlinien im Wesentlichen in die Action-Bar, der oberen Leiste auf dem User Interface, die

einen schnellen Zugriff auf die wichtigsten Funktionen bietet, der Top-Bar zur Implementierung von bspw. Reitern und einer weiteren Bottom Bar für weitere Funktionsbuttons. Diese Reiter (engl. *Tabs*) in der dargestellten Variante, wie sie in Abbildung 5.2 zu sehen ist, ermöglichen mit einer Wischbewegung nach links oder rechts den Übergang zu anderen Elementen der Benutzungsoberfläche. Eine wichtige Eigenschaft für die Gestaltung und Anordnung der UI-Elemente eines Bildschirms wird durch die Auswahl des geeigneten Layouts und deren hierarchischen Zusammenstellung gewährleistet. Hierfür stehen für die Entwickler bereits einige Gerüste zur Verfügung, von denen eine kleine Auswahl in Abbildung 5.3 dargestellt ist. Während das Linear Layout die lineare Anordnung von UI Elementen auf dem Bildschirm ermöglicht, werden innerhalb eines Relative Layouts die Elemente immer in Relation zueinander gestellt. Eine sinnvolle Verschachtlung solcher Layouts ist eine der Aufgaben im User Interface Design.

Entwicklungsumgebung und Projekte

Die zur Auswahl stehenden Entwicklungsumgebungen sind neben der **Eclipse IDE** mit **Android Developer Tools (ADT) Plugin**, das auf dem **IntelliJ IDEA** aufbauende **Android Studio** [1]. Auch wenn innerhalb der beiden Umgebungen die wichtigsten Funktionalitäten gleichermaßen zur Verfügung stehen, geht der Trend in Richtung Android Studio, da dieser eine bessere Unterstützung für die wachsenden Android Technologien, wie die Android Wearables, gibt. Die Projektansichten, wie sie in Abbildung 5.4 auch zu erkennen sind, weisen keine großen Unterschiede auf. Jede Android App besitzt eine Manifest Datei, welche zum Ausführen benötigt wird und u. a. die Berechtigungen einer Applikation definiert. Testen lassen sich schließlich Android Apps entweder auf echter Hardware oder

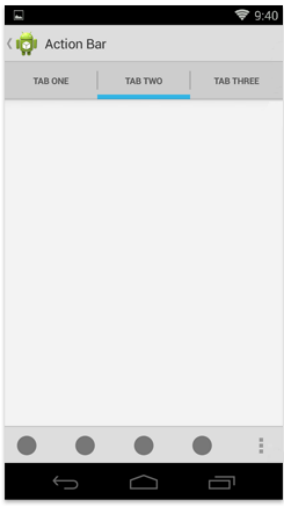


Abbildung 5.2: Aufbau der Benutzungsoberfläche einer Android App (Quelle: https://developer.android.com/design/patterns/actionbar.html)

mithilfe des Android Virtual Devices Managers (AVD) auf vielen verschiedenen Geräteemulatoren mit diversen Möglichkeiten zur Individualisierung [33].

App Lebenszyklus

Eine Android Applikation befindet sich zur Laufzeit in verschiedenen Zuständen, die über die Methodenaufrufe `onCreate()`, `onStart()`, `onResume()` und `onPause()` angesprochen werden (vgl. Abbildung 5.5). Die `onCreate()` Methode wird unmittelbar nach dem Öffnen einer Applikation ausgeführt, bei der typischerweise das Layout aufgebaut und initiale Konfigurationen durchgeführt werden. Sobald die erste Activity auf dem Bildschirm erscheint, wird die Methode `onStart()` aufgerufen. Die beiden Methoden `onResume()` und `onPause()` werden nur dann umgeschaltet, wenn die aktive App unterbrochen wird. Wechselt der Nutzer zum Beispiel zu einer anderen App als die zuvor ausgeführte, wird dafür die `onPause()` Methode aufgerufen bis der Fokus wieder auf diese App gelegt und die `onResume()` Methode aufgerufen wird. Diese Methoden können überschrieben werden, um bestimmte Funktionen in den jeweiligen Zuständen zu implementieren. Eine mögliche Modellierung wäre bspw. die Löschung gespeicherter Historien bei der Beendigung einer App.

5.2.3 Pebble-SDK / PebbleKit

Zur Programmierung einer Anwendung für die Pebble-Smartwatch werden zwei Komponenten benötigt: Das Pebble-Watchapp-SDK und PebbleKit-Android. Ersteres bildet dabei die Grundlage für die Anwendung auf der Smartwatch und letzteres die Komponente auf Seiten des Smartphones, die für eine Kommunikation mit der Smartphone-App benötigt wird.

Pebble-Watchapp-SDK Das Pebble-Watchapp-SDK [4] ist verfügbar für Mac OS X oder Linux. Für Windows-Nutzer wird empfohlen zum Programmieren CloudPebble zu verwenden. CloudPebble ist die offizielle Online-IDE zur Entwicklung von Pebble-Anwendungen und wurde in dieser Projektgruppe zur Entwicklung verwendet.

(a) Linear Layout (b) Relative Layout (c) List View

Abbildung 5.3: Layouts einer Android App (Quelle: http://developer.android.com/guide/topics/ui/declaring-layout.html)

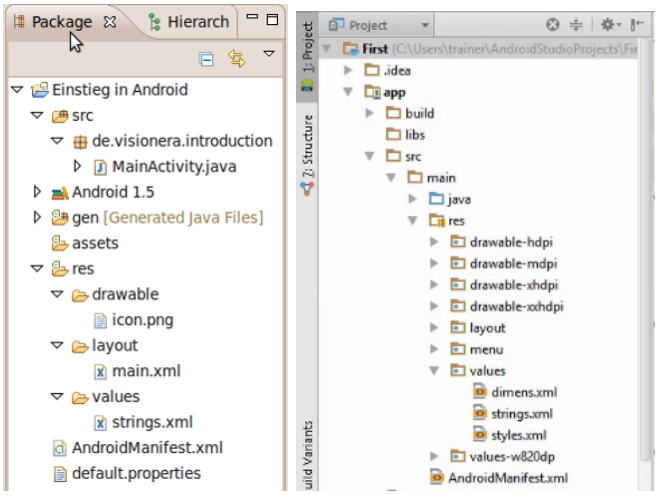
Anwendungen, die mit dem Pebble-Watchapp-SDK entwickelt werden, müssen in C programmiert werden. Pebble-Anwendungen bestehen aus einem oder mehreren `Windows`, auf welchen mittels `Layern` Grafiken oder Texte angeordnet werden können. Das SDK stellt dabei viele Methoden bereit, welche zur Umsetzung der gewünschten Funktionen verwendet werden müssen.

Das Framework der Pebble kümmert sich nicht um die verbrauchten Ressourcen, der Entwickler muss daher alle Objekte selbstständig anlegen und auch wieder zerstören, wenn es nicht mehr benötigt wird. Aufgrund der begrenzten Ressourcen auf der Pebble ist dies auch notwendig.

PebbleKit-Android PebbleKit-Android [5] ist ein zur Verfügung gestelltes Modul, welches in die eigene Android-App eingebunden werden kann. Mithilfe dieses Moduls kann die App mit der Pebble kommunizieren und anders herum. Weiter kann durch den Einsatz dieses Moduls bspw. der Speicher des Android-Smartphones von der Pebble als Speicher verwendet werden, um dort Programmdaten abzulegen.

5.2.4 ZBar

Zur Barcode-Detektion wird die ZBar-Bibliothek [13] eingesetzt. Dies ist eine C-Bibliothek, die mit C++-Wrappern versehen ist. Die Erkennung basiert auf der Funktionsweise eines Linienscanners, eine grafische Darstellung der Erkennungs-Pipeline bietet Abbildung 5.6. Als Eingaben erwartet die Bibliothek einen Videostream oder ein Bild. Im Falle eines Videostreams wird dieser in Frames zerlegt und die Einzelbilder werden weiter verarbeitet (*image stream*). Für das Bild wird im *image scanner* für jede Zeile ein Histogramm erstellt. Dieses wird vom *linear scanner* weiter verarbeitet, dieser berechnet aus der Abfolge der



(a) Eclipse IDE mit ADT (b) Android Studio

Abbildung 5.4: Ansicht des Projektexplorers

Intensitäten des Histogramms die Breiten der möglichen Barcode-Blöcke. Anschließend werden diese Abfolgen von Barcode-Blöcken in den *detector* gegeben, dieser dekodiert einen möglicherweise vorhandenen Barcode. Die Ausgabe des Algorithmus ist, falls vorhanden, der gelesene Barcode als Zeichenfolge.

Da der Algorithmus das Bild zeilenweise durchläuft werden nur annähernd waagerecht liegende Barcodes erkannt. Da dies im vorliegenden Szenario nicht gewährleistet werden kann, ist eine Vorverarbeitung der Bilder mit Hilfe von OpenCV notwendig. Ein weiterer Grund für die Vorverarbeitung ist die mit der Bildgröße steigende Laufzeit des Algorithmus. Die Einbindung der ZBar-Bibliothek sowie die Vorverarbeitung wird im Kapitel 6.1.3 beschrieben.

5.2.5 Protocol Buffers

Zum Austauschen der Nachrichten zwischen Server und Smartphone wurde nach einer Lösung gesucht, die flexibel und leicht zu verstehen ist. Da die Objekte zwischen Java und C++ ausgetauscht werden, können diese nicht mit den nativen Funktionen der jeweiligen Sprache serialisiert und deserialisiert werden. Eine Möglichkeit die Informationen zu übertragen, ist diese in JSON oder XML zu kodieren. Dies hätte zur Folge gehabt, dass mindestens das Generieren nativer Objekte von Hand erledigt werden müsste.

Daher ist die Wahl auf die Google Protocol Buffer [2] gefallen. Diese Bibliothek bietet die Möglichkeit Objekte auf einer abstrakten Ebene in einem einfachen Schema zu definieren und aus den Definitionen automatisch Klassen für die Sprachen Java und C++ zu erzeugen. Der generierte Quellcode enthält neben Getter und Setter auch Checker für jede Variable, sowie Serialisierer und Deserialisierer für die Objekte. Dies hat zur Folge, dass ein Objekt, welches auf dem Server erzeugt wurde, mit einem Funktionsaufruf serialisiert werden kann, das Ergebnis per TCP auf das Smartphone übertragen wird und dort mit ebenfalls einem Funktionsaufruf ein exakt gleiches Objekt deserialisiert wird. Dies funktioniert auch in die andere Richtung.

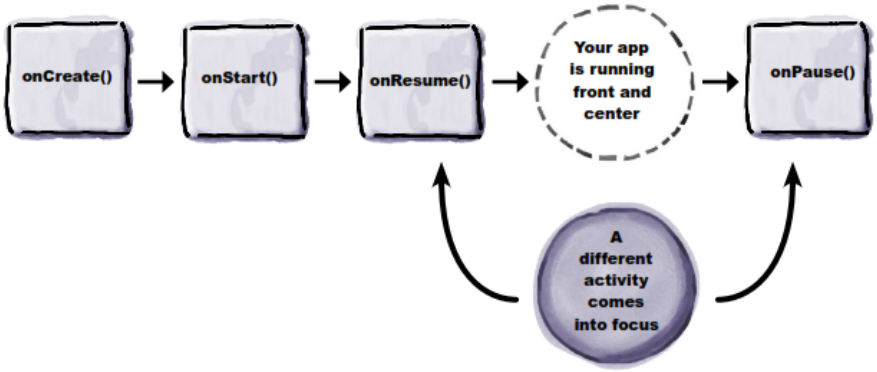


Abbildung 5.5: Der Lebenszyklus einer Activity (Quelle: [33])

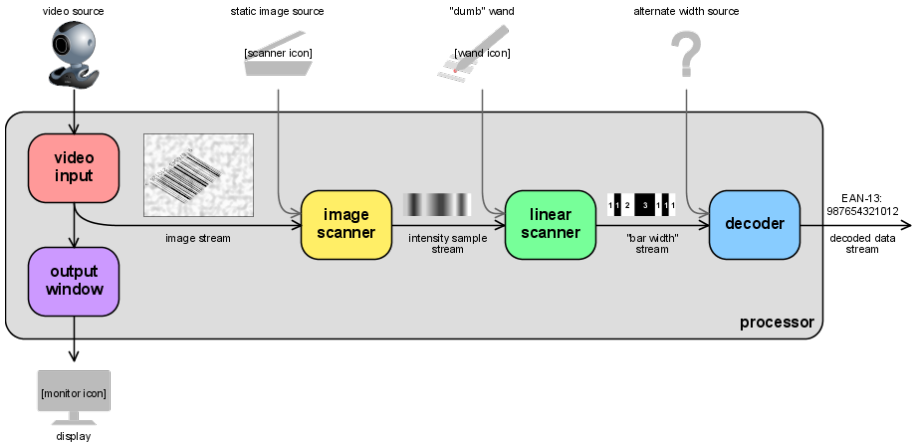


Abbildung 5.6: Die ZBar Erkennungs-Pipeline (Quelle: http://zbar.sourceforge.net/about.html)

KAPITEL 6

Architektur und Implementierung

In diesem Kapitel werden unter anderem die Funktionalität und Oberfläche der einzelnen Software-Komponenten vorgestellt. Weiterhin wird erläutert, welche Technologien eingesetzt werden.

Zur effizienten Bearbeitung der einzelnen Teilbereiche wurden die Teilnehmer der Projektgruppe in einzelne Arbeitsgruppen eingeteilt. Die Aufteilung und die zugeordneten Mitglieder wurden in Kapitel 2.2 vorgestellt. Das entwickelte System wird in den folgenden Abschnitten vorgestellt. Die Gliederung erfolgt dabei gemäß den Arbeitsgruppen **Android**, **Server** und **Kommunikation**.

6.1 Server

Der Server ist das Herzstück des Systems. Mit einer einfach gehaltenen Kommunikationsstruktur können sich Benutzer über WLAN im System an- und abmelden, Picklisten empfangen und Bilder von Artikeln senden, die dann auf dem Server analysiert und Artikeln im Lager zugeordnet werden. Intern verwaltet der Server das Hinzufügen, Löschen und Bearbeiten von Benutzern, Artikeln, Standorten, Picks und Picklisten.

In diesem Kapitel soll die genaue Architektur des Servers vorgestellt werden. Zuerst wird die Oberfläche des Servers und dessen vorhandenen Aktionsmöglichkeiten beschrieben. Darauffolgend wird die Implementierung bzw. die Umsetzung der Funktionen des Servers näher erläutert. Anschließend werden die Umsetzung der Objekt-Erkennung von gesendeten Bildern und die Kommunikation mit dem Smartphone vorgestellt.

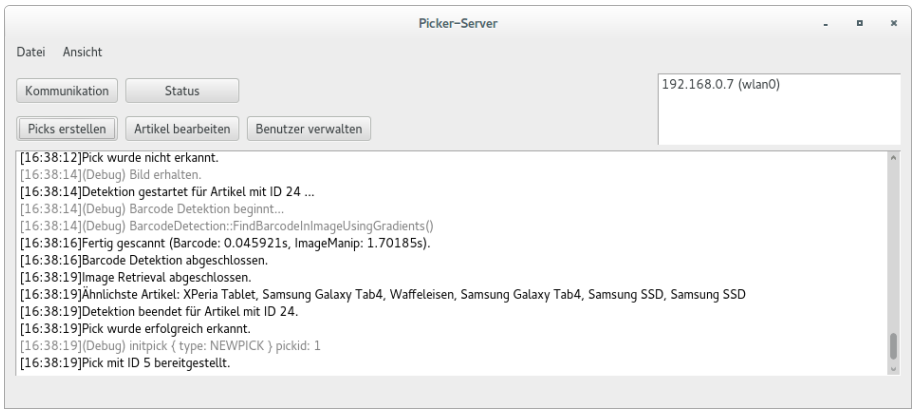


Abbildung 6.1: Das Hauptfenster des Servers.

6.1.1 Bedienung & GUI

Wird der Server gestartet, so öffnet sich zunächst das Fenster **Picker-Server** (siehe Abbildung 6.1).

Picker-Server Das Fenster **Picker-Server** ist das Hauptfenster und von dort aus lassen sich alle weiteren Funktionen aufrufen (siehe Abbildung 6.1). Zentral sticht das große Nachrichtenfenster heraus, in welchem Debug- und Servernachrichten geloggt werden. Das kleinere weiße Fenster zeigt die aktiven Netzwerkadressen des Servers an. Der Menüpunkt **Datei** ermöglicht es das Programm zu schließen oder Einstellungen aufzurufen. In den Einstellungen lassen sich z. B. die Pfade für die Datenbank oder den relativen Bilderordner anpassen. Mit dem Menüpunkt **Ansicht** können weitere Buttons angezeigt werden. Diese werden hier vernachlässigt, weil sie nur zum Testen gedacht sind. Die fünf Buttons in Abbildung 6.1 werden im Weiteren erläutert.

Kommunikation Dieser Button öffnet das Fenster **Verlauf Kommunikation** (siehe Abbildung 6.2). Dies ist ein Logfenster, welches die Kommunikation zwischen Smartphone und Server wiedergibt.

Status (siehe Abbildung 6.3). Das Statusfenster zeigt den Status der Pickliste eines beliebigen, angemeldeten Benutzers an. Unten rechts lässt sich der Benutzer auswählen, zu dem der Status abgerufen werden soll. Oben rechts wird die Pickliste angezeigt. Es wird angezeigt, ob Artikel schon erkannt oder als fehlend markiert wurden. Der zur Zeit gesuchte Artikel ist grau unterlegt. Informationen zu diesem Artikel werden unten links aufgelistet. Oben links wird das vom entsprechendem Smartphone zuletzt gesendete Bild angezeigt. Zwischen Benutzerauswahl und Pickliste befindet sich ein Text, der den Betrachter wissen lässt, ob die Erkennung zur Zeit läuft oder für das Bild abgeschlossen ist.

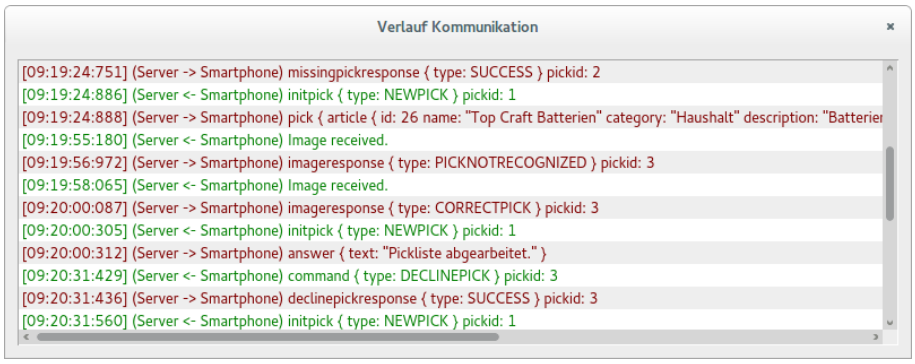


Abbildung 6.2: Logfenster, welches die Nachrichten aufzeichnet, welche zwischen Server und Smartphone ausgetauscht werden. Rote Nachrichten gehen dabei vom Server aus, grüne Nachrichten entstammen vom Smartphone.

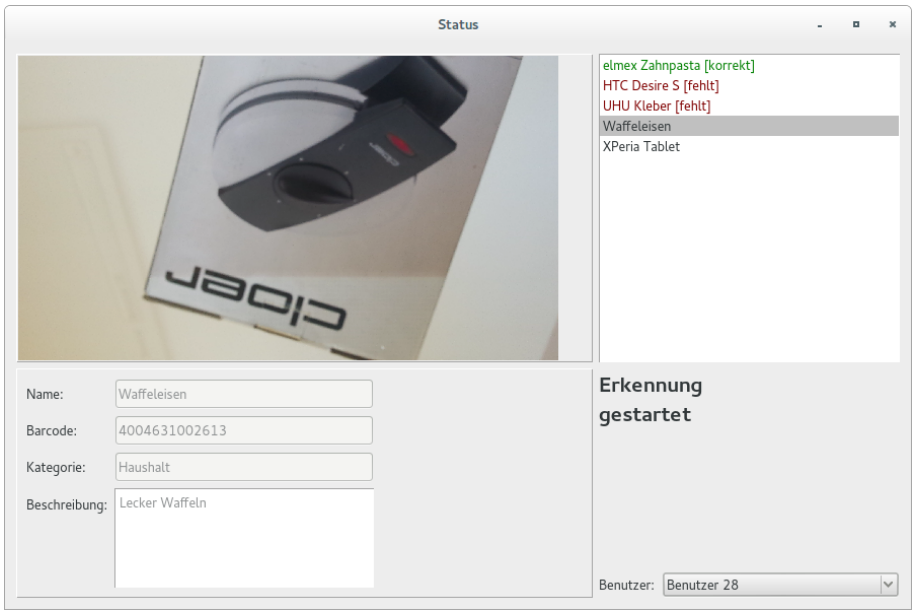


Abbildung 6.3: Das Statusfenster zeigt den Status der Pickliste eines beliebigen Benutzers an, zusätzlich mit der letzten Aufnahme.

Picks erstellen Dort können Picks und Lagerplätze bearbeitet werden (siehe Abbildung 6.4). Soll ein neuer Pick erstellt werden, so muss ein Lagerplatz aus der Liste gewählt werden. Des Weiteren muss auf der rechten Seite die Anzahl angegeben werden und auf der selben Seite der Benutzer ausgewählt werden. Um einen neuen Lagerplatz anzulegen muss auf der rechten Seite Ebene, Fach und Regal ausgefüllt werden. Die Liste auf der linken Seite zeigt alle Picks an, die dem Benutzer zugewiesen sind, der im Dropdown-Feld unter dieser Liste ausgewählt ist. Unter der Liste der Picks werden Informationen zu dem Artikel und dem Lagerplatz angezeigt, falls ein Pick ausgewählt ist. Als weitere Funktion

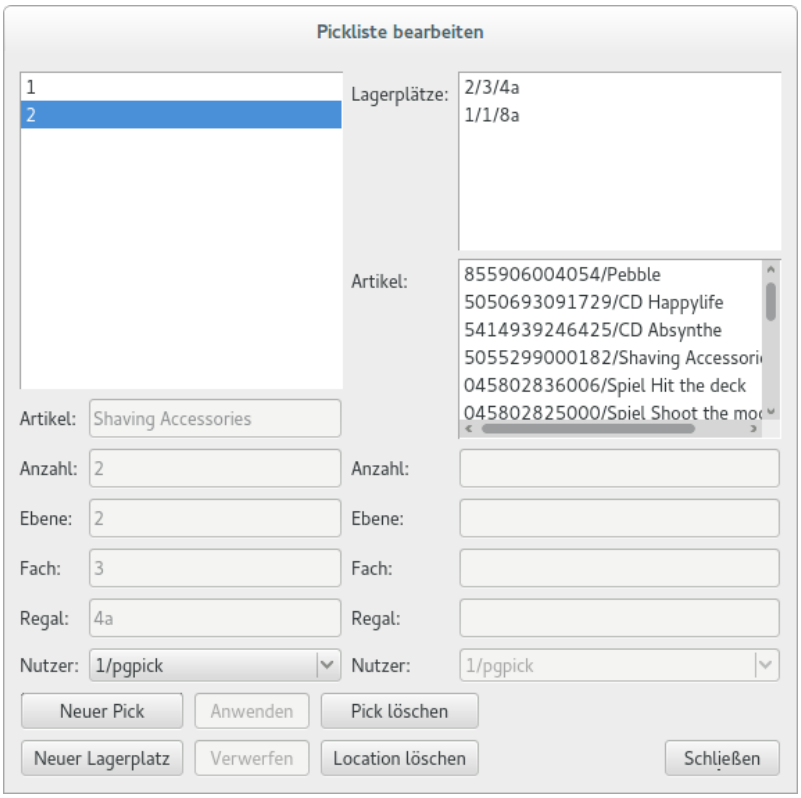


Abbildung 6.4: Das Fenster **Pickliste bearbeiten** bietet die Möglichkeit Lagerplätze und Picks zu erstellen oder zu löschen.

ist es möglich, Lagerplätze oder Picks zu löschen. Wird ein Lagerplatz entfernt, so werden zugeordnete Picks automatisch gelöscht, falls diese existieren.

Artikel bearbeiten Bei einem Klick auf den Button **Artikel bearbeiten**(siehe Abbildung 6.5). Die Liste auf der linken Hälfte zeigt alle bisher vorhandenen Artikel an. Die Artikel werden mit ihrem EAN-Barcode und Namen aufgelistet. Soll ein neuer Artikel angelegt werden, so muss der Name, der EAN-Barcode, die Kategorie und eine Beschreibung angegeben werden. Optional können Bilderpfade für den entsprechenden Artikel ausgewählt werden. Das Label wird für die Klassifikation benötigt, und wird automatisch gesetzt. Werden Bildpfade angegeben und die Checkbox **Merkmale berechnen** angetickt, so wird zu jedem Bild ein Merkmalsvektor erstellt, der ermöglicht, den Artikel durch Image-Retrieval zu erkennen. Wurde ein falscher Bildpfad ausgewählt, so lässt sich dieser nachträglich löschen. Um einen Artikel zu ändern, muss zunächst dieser Artikel aus der Liste gewählt werden. Gleiches gilt für das Löschen. Wird ein Artikel gelöscht, so werden Picks, die diesen Artikel beinhalten automatisch entfernt.

Benutzer verwalten Der Button **Benutzer verwalten** öffnet das Fenster für die Benutzerverwaltung (siehe Abbildung 6.6). Die Liste enthält alle vorhandenen Benutzer. Um

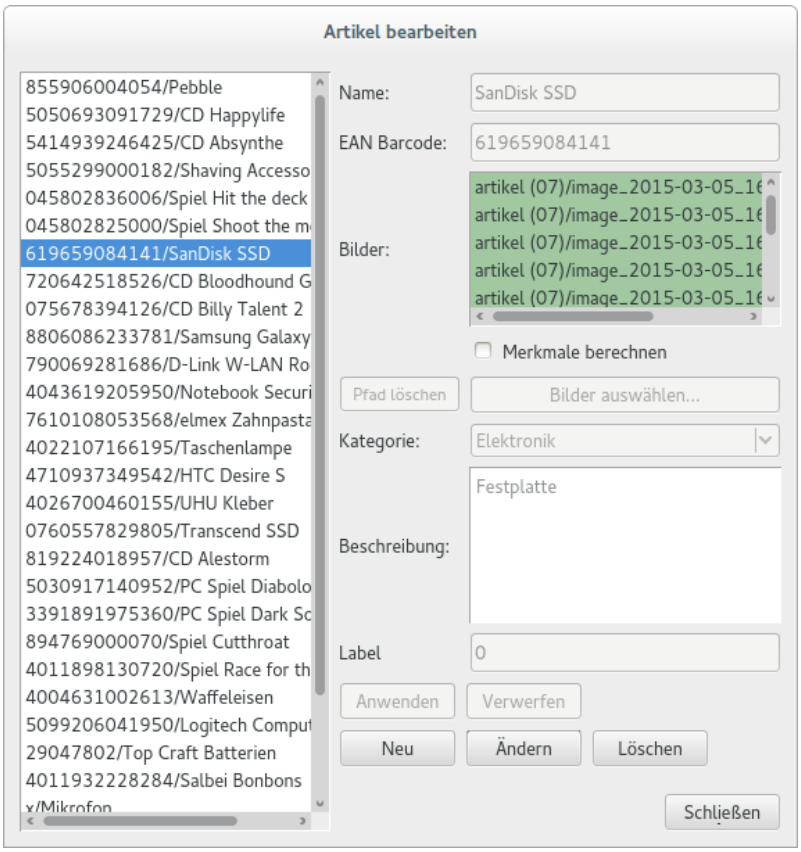


Abbildung 6.5: Das Fenster **Artikel bearbeiten** bietet die Möglichkeit Artikel anzulegen, zu bearbeiten oder zu löschen.

einen Benutzer zu löschen oder sein Passwort zu ändern, muss dieser zuerst ausgewählt werden. Wird ein Benutzer gelöscht, so werden seine Picks ebenfalls gelöscht. Es ist auch möglich neue Benutzer hinzuzufügen.

6.1.2 Implementierung

Der Server wurde in `C++` erstellt. Mittels **Google Protocol Buffers** kommuniziert der Server mit einem oder mehreren Smartphones. Abbildung 6.7 zeigt einen groben Überblick über die Klassen der Serverarchitektur.

Einstiegspunkt ist die Klasse `Server`, die auf eingehende Verbindungen wartet. Kommuniziert ein Smartphone bzw. ein Benutzer mit dem Server, so wird für diese Verbindung ein neuer Thread `UserThread` erstellt, der Anfragen vom Benutzer erhält, diese an die `Communication`-Klasse weiterleitet und Antworten an den Benutzer zurückliefert. Näheres zur Kommunikation findet sich in Kapitel 6.1.4.

Die Klasse `Communication` ist die Schnittstelle zwischen dem TCP-Server und der eigentlichen Verwaltung. Für jeden eingeloggten Benutzer wird ein `UserController` erstellt, der den aktuellen Benutzer speichert und die aktuelle Pickliste verwaltet. Ebenso erschafft

Abbildung 6.6: Das Fenster **Benutzer verwalten** bietet die Möglichkeit Benutzer anzulegen, zu löschen oder Passwörter zu ändern.

jeder `UserController` einen `DetectionController`, der sich um die Erkennung der gesendeten Bilder kümmert und sich die Ergebnisse der Detektor-Klassen `BarcodeDetection`, `ImageRetrieval` und `Classification` zu Nutze macht. (vgl. Kapitel 6.1.3).

Model-Entitäten

Die Serveranwendung bedient sich einer recht übersichtlichen Model-Struktur, um den Status der Anwendung persistent zu halten. Abbildung 6.8 zeigt die Datenhaltungsklassen und ihre Verbindungen.

User Ein User stellt einen Benutzer im System dar. Ein User besitzt einen Benutzernamen und ein Passwort. Ihm wird ebenso eine Pickliste zugeteilt.

Article Repräsentiert einen Artikel. Ein Artikel besteht aus einer ID, einem Namen, einem Barcode, einer Beschreibung und einer visuellen Kategorie (Label). Ebenso besitzt jeder Artikel eine Anzahl von Merkmalsvektoren und Bildern, die den Artikel zeigen.

Location Beschreibt einen Lagerplatz bzw. Standort. Einem Standort ist eine ID, ein Regal, eine Reihe und eine Spalte zugeordnet.

Pick Fügt einen Artikel und einen Lagerplatz zu einem Pick zusammen. Ebenso beschreibt ein Pick die Anzahl der Artikel, die eingesammelt werden müssen.

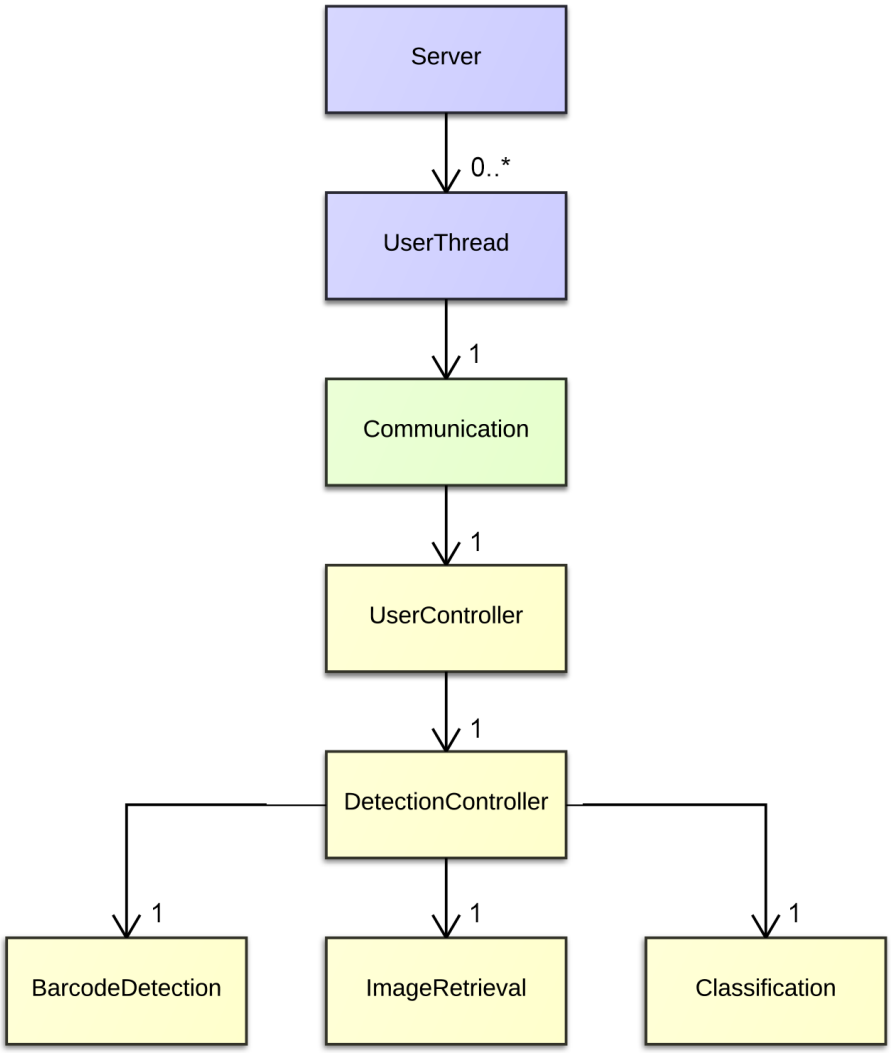


Abbildung 6.7: Aufbau und Hierachie der Server-Klassen

PickInList Setzt sich aus einem Pick und dessen aktuellen Status im Kommissioniervorgang zusammen. Ein Pick kann akzeptiert, als fehlend oder als übersprungen markiert sein.

Picklist Jeder Benutzer besitzt eine Pickliste, in diesem Fall eine geordnete Liste von `PickInList`-Objekten.

Datenbank

Die Datenbank beruht auf SQLite. Sie besteht aus Tabellen für Benutzer, Artikel, Picks, Lagerplätze und Merkmale. Dabei orientiert sie sich an der Struktur der Model-Entitäten.

Die Benutzertabelle enthält Spalten für die ID, den Benutzernamen und das Passwort. Dabei können Benutzer eindeutig an ihrer ID oder ihrem Benutzernamen erkannt werden. Diese Tabelle enthält alle Benutzer, die sich im System anmelden können. Möchte sich ein

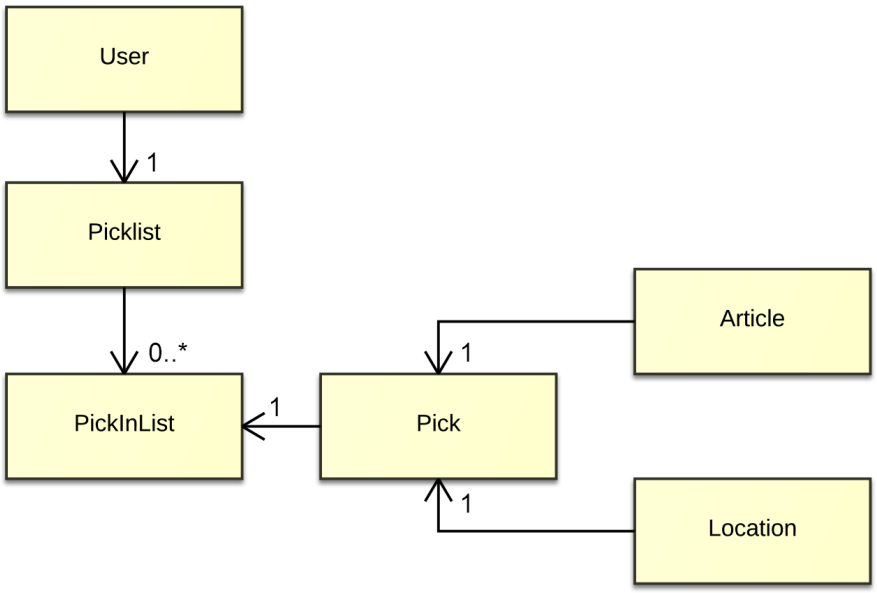


Abbildung 6.8: Klassendiagramm der Model-Entitäten

Benutzer im System anmelden, so wird verglichen, ob dieser Benutzer mit dem angegebenen Passwort in der Datenbank existiert. Über eine Maske können neue Benutzer hinzugefügt werden (siehe Abbildung 6.6).

Die Artikeltabelle enthält Spalten für die ID, Artikelbezeichnung, EAN-Barcode, Kategorie, Beschreibung und Label. EAN-Barcodes sind einzigartig in der Datenbank. In der Artikeltabelle sind alle Artikel hinterlegt, die im Lager existieren. Über eine Maske können neue Artikel hinzugefügt werden (siehe Abbildung 6.5).

Die Merkmalstabelle enthält Spalten für die ID, Artikel-ID, Bildpfad und Merkmalsvektor. Das heißt, jeder Spalte in der Merkmalstabelle ist ein Artikel aus der Artikeltabelle zugeordnet. Artikel können mehrere Einträge in der Merkmalstabelle besitzen, da zu jedem Artikel mehrere Bildpfade angegeben werden können. Die Merkmalsvektoren werden beim Image-Retrieval und der Klassifikation verwendet. Merkmalsvektoren für die Bilder eines Artikels werden nur berechnet, wenn beim entsprechenden Artikel **Merkmale berechnen** im **Artikel bearbeiten**-Fenster (Abbildung 6.5) gesetzt ist.

Die Lagerplatztabelle enthält Spalten für die ID, Ebene, Fach und Regal. Sie enthält die vorhandenen Lagerplätze.

Die Picktabelle enthält Spalten für die ID, Lagerplatz-ID, Artikel-ID, Benutzer-ID und Anzahl. Also werden jedem Eintrag in der Picktabelle ein Artikel und ein Lagerplatz zugeordnet. Eine Pickliste für einen Benutzer mit der ID **id** sind in diesem Fall alle Picks mit der Benutzer-ID **id**. Picks und Lagerplätze können im Programm gelöscht oder hinzugefügt werden (siehe Abbildung 6.4).

Kommunikationsmethoden

Die `Communication`-Klasse bietet neun unterschiedliche Funktionen, die das Smartphone anfragen kann und welche eine Antwort an das Smartphone senden.

```
1 bool login(string username, string password);
2 bool loggedIn();
3 bool logout();
4 Pick* giveActPick();
5 int pictureReceived(int pickId, cv::Mat picture);
6 bool pickAccepted(int pickId);
7 bool pickMissing(int pickId);
8 bool pickSkipped(int pickId);
9 bool pickRepeated(int pickId);
```

Listing 6.1: Kommunikationsmethoden der `Communication`-Klasse

Je nach Anfrage ruft der `UserThread` die jeweilige Methode in der `Communication`-Klasse auf (vgl. 6.1.4). Die neun Methoden sind in Listing 6.1 abgebildet.

Die ersten drei Methoden sind nötige Anfragen zur Mehrbenutzerfähigkeit des Systems. Bevor ein Benutzer andere Anfragen an den Server stellen kann, muss sich dieser zuerst einloggen. War der Login auf dem Server erfolgreich, wird er über ein **SUCCESS** darüber informiert. Andernfalls erhält er die Nachricht, dass seine Benutzerdaten falsch eingegeben wurden. Über die Methode `loggedIn` kann der Benutzer erfragen, ob er erfolgreich eingeloggt ist. Hat er seine Arbeit erledigt, so kann über ein `logout` das System darüber informieren und die bestehende Verbindung wird getrennt.

Ist der Benutzer erfolgreich eingeloggt, so erhält er über die Funktion `giveActPick`, den aktuell zu bearbeitenden Pick. Wurde ein Pick erfolgreich bearbeitet, so liefert diese Methode einen neuen Pick, andernfalls wird immer der gleiche Pick gesendet.

Hat der Benutzer einen aktuellen Pick erhalten, so wird über `pictureReceived` das Kamerabild des Smartphones mit dem aktuellen Pick an den Server übertragen und die Erkennung wird auf dem Server gestartet. Der Ablauf der Erkennung wird in Kapitel 6.1.2 und die einzelnen Erkennungsmethoden in Kapitel 6.1.3 detailliert beschrieben. Der Kommissionierer erhält eine von vier möglichen Antwortmöglichkeiten über das Ergebnis der Erkennung, die wie folgt codiert werden:

- `-1` = **ERROR**: Es ist ein Fehler aufgetreten. Möglicherweise ist das Bild ungültig oder die ID des Picks ist nicht gültig.
- `0` = **CORRECKTPICK**: Der Artikel auf dem Bild stimmt mit dem Artikel des Picks mit der mitgesendeten ID überein.

- `1` = **WRONGPICK**: Der Artikel auf dem Bild kann über die Barcode-Detektion eindeutig einem Artikel in der Datenbank zugeordnet werden, allerdings stimmt dieser nicht mit dem Artikel des Picks mit der mitgesendeten ID überein.
- `2` = **PICKNOTRECOGNIZED**: Der Artikel auf dem Bild konnte nicht erkannt werden.

Ebenso stehen dem Benutzer für einen Pick vier Auswahlmöglichkeiten zur Verfügung, die er benutzen kann, falls die Erkennung fehlschlägt.

- `pickAccepted`: Falls die Erkennung wiederholt den Artikel nicht erkennen kann, der Benutzer aber sicher ist, den richtigen Artikel in der Hand zu halten, so kann er über diese Methode den Pick manuell akzeptieren. Diese Methode liefert ein **FAILURE**, falls der Pick bereits richtig erkannt wurde, ansonsten **SUCCESS**.
- `pickMissing`: Ist ein Artikel für den Kommissionierer nicht auffindbar, so kann er dies dem System melden und der Pick wird in der Pickliste als fehlend markiert. Liefert ein **FAILURE**, falls der Artikel bereits akzeptiert wurde, sonst **SUCCESS**.
- `pickSkipped`: Will ein Benutzer den aktuellen Pick überspringen, z. B. weil das Regal von einem anderen Kommissionierer besetzt wird, so kann er dies über diese Anfrage tun. Liefert ein **FAILURE**, falls der Artikel bereits akzeptiert, als fehlend markiert oder übersprungen wurde.
- `pickRepeated`: Wurde ein Pick fehlerhaft akzeptiert oder die Überspringung oder die Markierung des Fehlens des Picks soll zurückgesetzt werden, so kann dies über diese Anfrage geschehen. Liefert immer ein **SUCCESS**, außer der Benutzer ist nicht eingeloggt oder die ID des Picks ist fehlerhaft.

Erkennungsprozess

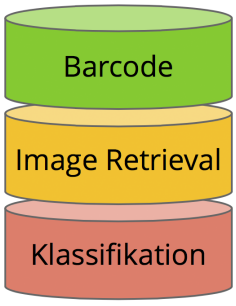


Abbildung 6.9: Erkennungsmethoden zur Identifizierung eines Artikels

Wurde ein Bild an den Server gesendet, so wird der `DetectionController` darüber informiert. Dieser aktiviert drei Erkennungsmethoden aus Abbildung 6.9. Zum derzeitigen

Punkt ist die Klassifikation aus technischen Problemen abgeschaltet, sie soll hier jedoch trotzdem erwähnt werden.

Barcode Die Barcode-Detektion sucht nach Barcodes auf dem empfangenen Bild und liefert eine Liste der gefundenen Barcodes. Dies wird im Kapitel 6.1.3 weiter beschrieben.

Image-Retrieval Das Image-Retrieval ermittelt die ähnlichsten Artikel zum gesendeten Bild. Dabei eignen sich für Pick by Feel die ähnlichten sechs Artikel. Dies wird in Kapitel 6.1.3 näher beschrieben.

Klassifikation Die Klassifikation stuft das Bild in eine visuelle Kategorie ein und wird ebenfalls in Kapitel 6.1.3 näher erläutert.

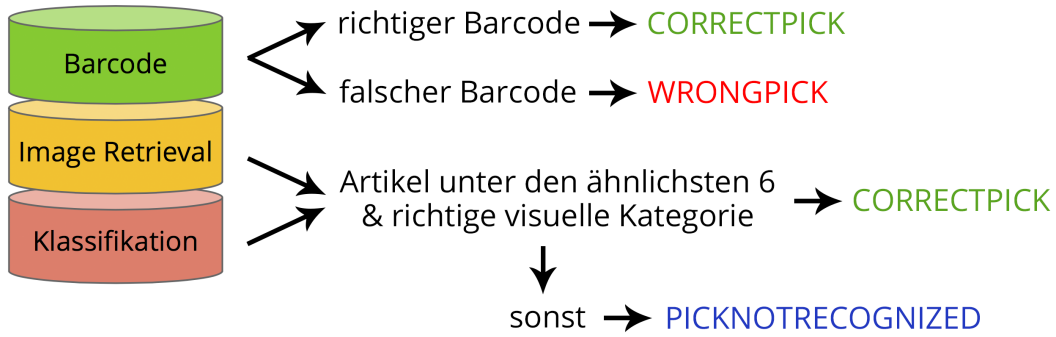


Abbildung 6.10: Erkennungsprozess zur Identifizierung eines Artikels

Die einzelnen Erkennungsmethoden werden im `DetectionController` parallel mit Hilfe von Threading gestartet. Über `signal`- und `slot`-Anweisungen können diese Threads dann mit dem `DetectionController` kommunizieren. Beispielsweise sendet die Barcode-Detektion bei jedem gefundenen Barcode diesen an den Controller und sendet ebenso eine abschließende Nachricht, wenn die Detektion beendet ist.

Der Ablauf der Erkennung (vgl. Abbildung 6.10) kann wie folgt beschrieben werden:

- **richtigen Barcode gefunden**: Der Server antwortet mit **CORRECKTPICK**
- **falschen Barcode gefunden**: Der Server antwortet mit **WRONGPICK**
- **Artikel unter den ähnlichsten sechs Artikeln im Image-Retrieval und richtige visuelle Kategorie**: Der Server antwortet mit **SUCCESS**
- **sonst**: Der Server antwortet mit **PICKNOTRECOGNIZED**

Es wird deutlich, dass ein Artikel nur dann als **WRONGPICK** akzeptiert wird, wenn ein falscher Barcode im Bild gefunden wurde. Die Kombination Retrieval und Klassifikation antwortet niemals mit **WRONGPICK**, sondern immer mit **PICKNOTRECOGNIZED**, falls der Artikel nicht erfolgreich zugeordnet wurde.

Da die Detektoren unabhängig voneinander arbeiten, kann der Erkennungsprozess ohne Probleme erweitert werden. Entscheidet man sich bspw. für die Implementierung einer Textdetektion, so muss diese nur zusätzlich im `DetectionController` aufgerufen werden. Diese erhält dann das gesendete Bild und würde daraufhin bspw. eine Aufzählung von Text zurückschicken, der auf dem Bild erkannt wurde. Die Logik, die entscheidet, ob ein Artikel erfolgreich erkannt wurde, muss bzgl. der Textdetektion angepasst werden.

6.1.3 Objekt-Erkennung

Barcode-Detektion

Ein Teil der Objekterkennung ist die Barcode-Detektion. Der auf dem Produkt vorhandene EAN-Barcode soll hierzu erkannt und decodiert werden. Wie im Kapitel 5.2.4 bereits beschrieben, werden die Bilder mit Hilfe der ZBar-Bibliothek auf Barcodes gescannt. Zusätzlich sind Vorverarbeitungs-Schritte notwendig. Die Implementierung von Vorverarbeitung und Erkennung wird im Folgenden beschrieben.

In der Vorverarbeitung sollen Regionen aus dem Bild extrahiert werden, die mutmaßlich einen Barcode enthalten. Diese Bereiche enthalten nur einen kleinen Ausschnitt des Bildes und lassen sich somit schneller von der ZBar-Bibliothek verarbeiten. Die Resultate der einzelnen Schritte sind in Abbildung 6.11 zu sehen, im Folgenden werden die notwendigen Schritte beschrieben. Da Barcodes eine Abfolge von hellen und dunklen Balken darstellen, sind in diesen Regionen starke Gradienten in der Helligkeitsverteilung zu erwarten. Das empfangene Bild enthält Farben, diese sind jedoch für die Barcode-Erkennung unnötig, das Bild wird in ein Graustufenbild umgewandelt. Im ersten Schritt werden die Gradientenbilder in horizontaler und vertikaler Richtung mit Hilfe eines Scharr-Operators berechnet. Dieser wirkt wie der Sobel-Operator (vgl. Kapitel 3.2.3), berücksichtigt Rotationssymmetrieen jedoch besser. Der Scharr-Operator hat daher die Form

$$H_x = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \quad \text{bzw.} \quad H_y = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}.$$

Die so ermittelten Bilder werden voneinander subtrahiert, die Absolutwerte werden gebildet. Anschließend wird eine Binarisierung durchgeführt. Dazu werden alle Bildpunkte mit einem Helligkeitswert kleiner als 225 auf 0 gesetzt, alle anderen Bildpunkte auf den Wert 255. In der Erprobung hat sich hier der Grenzwert von 225 als sinnvoll erwiesen, prinzipiell ist der Wert abhängig von der Belichtung des Bildes. Somit ergibt sich ein schwarz/weiß-Bild, in dem nur Regionen mit einem starken Gradienten weiß gefärbt sind. Um die Region des Barcodes als ganzes zu erhalten, werden die kleineren Regionen verschmolzen. Dazu wird eine Reihe von Closing und Opening Operationen durchgeführt. Somit werden zu schmale bzw. punktförmige Regionen eliminiert, während die vorhandenen großen Bereiche verschmolzen werden. (vgl. [32], Kapitel 3.2.2) Abschließend werden die jenigen Bereiche

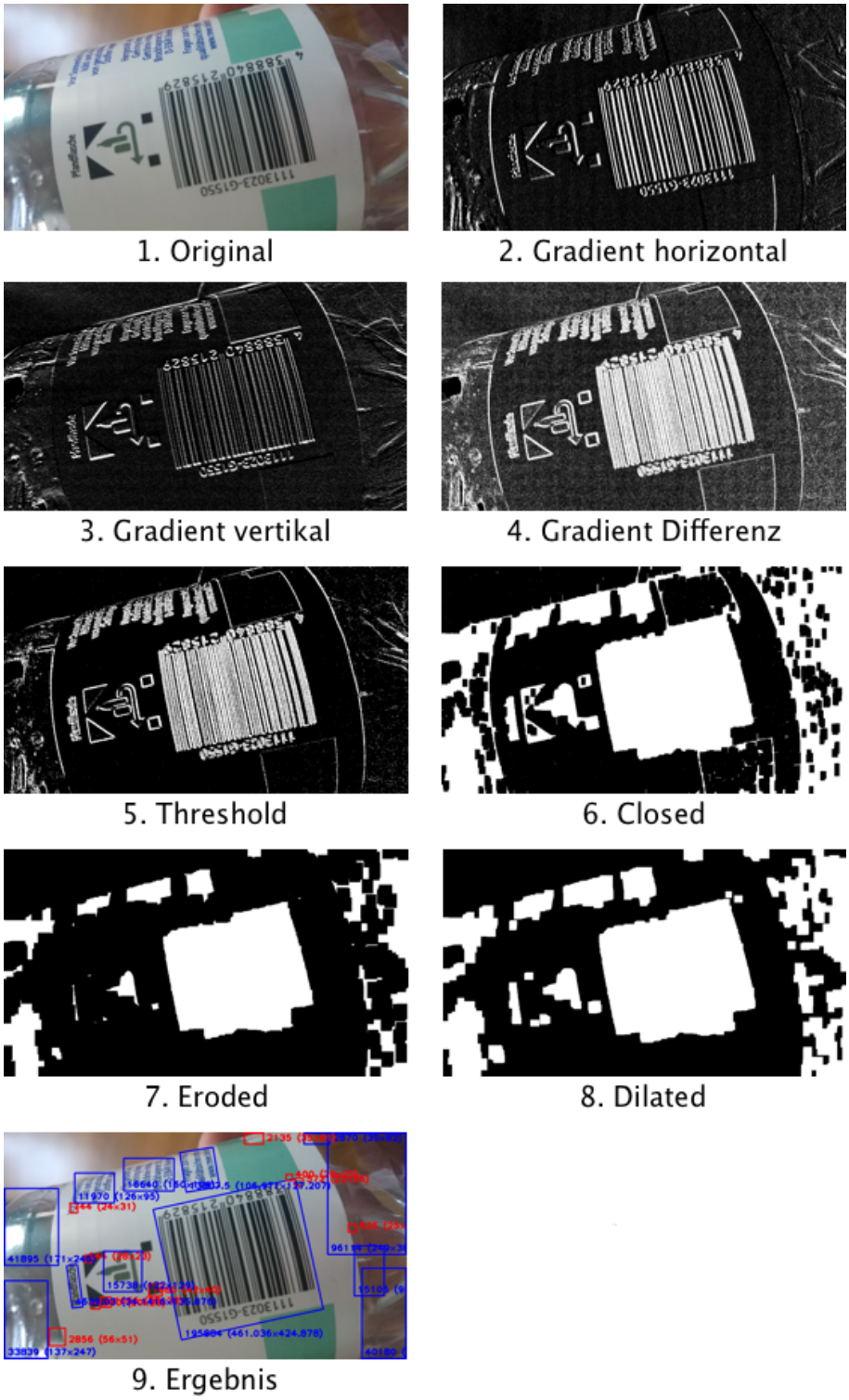


Abbildung 6.11: Bearbeitungsschritte zur Erkennung der Regionen mit Barcodes

für die Barcode-Dekodierung ausgewählt, die eine ausreichende Größe aufweisen. In der Praxis hat sich hier eine minimale Höhe von 50 Pixel und eine minimale Breite von 150 Pixel als sinnvoll erwiesen (vgl. Evaluation in Kapitel 7).

Die so ermittelten Bildausschnitte werden jetzt noch in einem Schärfungsprozess bearbeitet, um die Linien des Barcodes besser hervor zu heben. Dazu wird ein Gauß-Filter auf das Bild angewendet, um eine unscharfe Version des Bildes zu erzeugen. Ein Gauß-Filter ist ein linearer Filter (vgl. Kapitel 3.2.2), der zur Gewichtung der Nachbarpixel eine Gaußsche Glockenkurve nutzt. Anschließend wird diese unscharfe Version vom ursprünglichen Bild subtrahiert. Durch diese Operationen werden Kanten hervorgehoben und eher gleichmäßige Bereiche des Bildes unterdrückt. Die Filtermatrix des Gauß-Filters hat eine Größe von 17x17 Pixel, mit einer Standardabweichung in der verwendeten Gauß-Kurve von 23. Bei der anschließenden Subtraktion wird das ursprüngliche Bild mit einem Faktor von 2,5 gewichtet, das unscharfe Bild mit einem Faktor von 1,5. Diese Werte haben sich aus Versuchen während der Evaluationsphase ergeben.

Anschließend werden die Abschnitte an die ZBar-Bibliothek übergeben. Hier wird die `ImageScanner`-Klasse der Bibliothek genutzt. Es wird ausschließlich nach EAN-Barcodes gesucht, da nur diese in die Datenbank aufgenommen wurden. Prinzipiell ist hier jedoch auch die Dekodierung von weiteren Barcode-Typen möglich. Als Rückgabe liefert die Barcode-Detektion eine Liste der gefundenen Barcodes, diese können anschließend mit dem Barcode des zu kommissionierenden Artikels verglichen werden.

Visuelle Erkennung

Bei der visuellen Erkennung wird versucht, den Inhalt eines Bildes zu erkennen. Dafür ist es im Allgemeinen wichtig, Informationen über den Problemkreis zu sammeln. Aus diesem Grund wurden von jedem Artikel im Lager Beispielbilder aufgenommen. Die Bilder werden dann verwendet, um verschiedene Erkennungsverfahren zu trainieren. Im Folgenden werden die Verfahren, sowie notwendige Vorverarbeitungsschritte näher dargestellt.

Merkmale Damit eine visuelle Erkennung angewendet werden kann, ist es notwendig die Bilder durch einen Merkmalsvektor zu beschreiben. Um einen solchen Vektor für ein Bild erstellen zu können, wird zunächst eine Bag of Features (BoF) erstellt (vgl. Kapitel 3.3.1). Der schematische Ablauf für diesen Schritt ist in Abbildung 6.12 dargestellt. Zunächst wird für alle Beispielbilder eine Keypoint-Detektion mit dem SIFT-Detektor durchgeführt und für jeden gefundenen Keypoint ein 128 dimensionaler Deskriptor berechnet. Das Vorgehen des SIFT-Detektors und die Berechnung eines Deskriptors wurde bereits in Kapitel 3.2.4 beschrieben. Die extrahierten Deskriptoren aller Bilder werden anschließend mit dem Lloyd-Algorithmus [20] in Cluster unterteilt. Die Anzahl der Clusterzentren muss dabei vorgegeben werden und stellt ein mögliches Optimierungskriterium dar (siehe Kapitel 6.1.3 Abschnitt Image-Retrieval). Die so gefundenen Clusterzentren entsprechen

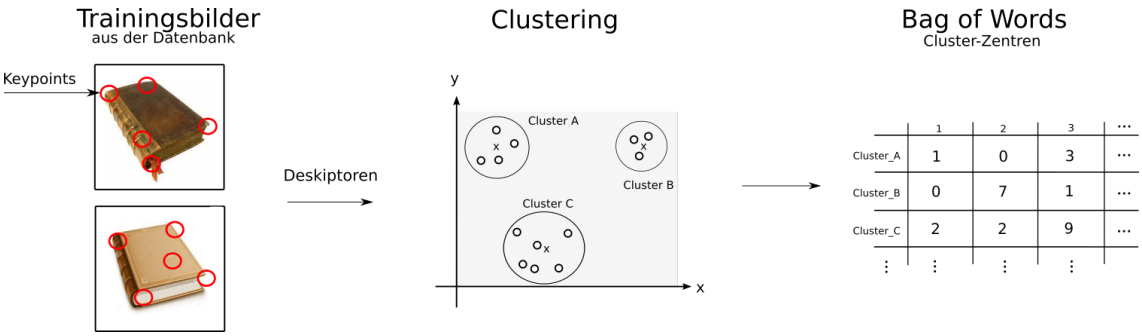


Abbildung 6.12: Schematischer Ablauf für die Berechnung eines Bag of Features

dem Vokabular der Bag of Features. Es kann als eine Matrix mit 128 Spalten und einer Zeile pro Clusterzentrum repräsentiert werden.

Mit Hilfe des Vokabulars kann für jedes Bild ein Merkmalsvektor erstellt werden. Um einen Merkmalsvektor für ein Bild zu berechnen, werden zunächst wieder die Keypoints bzw. Deskriptoren für das Bild extrahiert. Jeder Deskriptor wird anschließend dem Clusterzentrum zugeordnet, dem er am nächsten liegt. Anschließend wird gezählt, wie viele Deskriptoren jedem Clusterzentrum zugeordnet wurden. Diese Werte werden in ein Histogramm geschrieben und ergeben so den Merkmalsvektor für das Bild. Die Dimensionalität des Merkmalsvektors entspricht damit der Anzahl der Clusterzentren.

Diese Schritte werden für alle Bilder im Trainingsdatensatz durchgeführt. Die berechneten Merkmalsvektoren werden anschließend in einer Datenbank auf dem Server gespeichert. Für die Bilder, welche das Smartphone zur Auswertung an den Server sendet, werden die Merkmalsvektoren analog berechnet. Mit den so berechneten Merkmalsvektoren können die im Folgenden beschriebenen Erkennungsverfahren angewendet werden.

Image-Retrieval Die Aufgabe des Image-Retrieval ist es, zu einem gegebenen Bild ähnliche Bilder zu finden (vgl. Kapitel 3.3.1). Für den konkreten Anwendungsfall bedeutet dies, dass zu dem Bild, welches vom Smartphone gesendet wird, die ähnlichsten Bilder aus den Beispielbildern ermittelt werden. Anhand der ähnlichsten Bilder kann dann entschieden werden, ob der richtige Artikel gegriffen wurde. Dies ist genau dann der Fall, wenn unter einer Menge der ähnlichsten Bildern mindestens ein Bild vom richtigen Artikel ist. Die Ähnlichkeit zweier Bilder wird über den Abstand ihrer Merkmalsvektoren berechnet. Für zwei Vektoren x, y kann der Abstand d unter anderem wie folgt berechnet werden (vgl. Abbildung 6.13):

- Euklidische-Norm, ist der direkte Abstand zwischen zwei Punkten im Raum:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan-Norm, ist die Summe der Differenzen jeder Koordinate:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Maximums-Norm, ist das Maximum der Differenzen jeder Koordinate:

$$d(x, y) = \max_i(|x_i - y_i|)$$

- Cosinus-Ähnlichkeit, ist der Winkel zwischen zwei Vektoren:

$$d(x, y) = 1 - \frac{x \cdot y}{||x|| * ||y||}$$

Mit diesem Vorgehen lässt sich bereits ein Image-Retrieval implementieren. Um die Fehlerrate zu reduzieren, kann das Image-Retrieval auf verschiedene Weisen angepasst werden. Hier können folgende Parameter variiert werden (vgl. Kapitel 7.1):

- Anzahl der Merkmale
- Kriterium für die Ähnlichkeit zweier Vektoren
- Anzahl der ähnlichsten Bilder für die Entscheidung

Klassifikation Ein weiteres Erkennungsmodul der Serversoftware bildet die Klassifikation. Ziel der Klassifikation, Artikel in Klassen einzuteilen und bei Eingabe von Bildern der Artikel, die Klasse des Artikels korrekt anzugeben. Diese Erkenntnis fließt als zusätzliche Variable in den Erkennungsprozess des Servers ein. Im Unterschied zur Barcodeerkennung und dem Image Retrieval wird kein konkreter Artikel erkannt, sondern nur eine Klasse, welche sich aus mehreren Artikeln bildet.

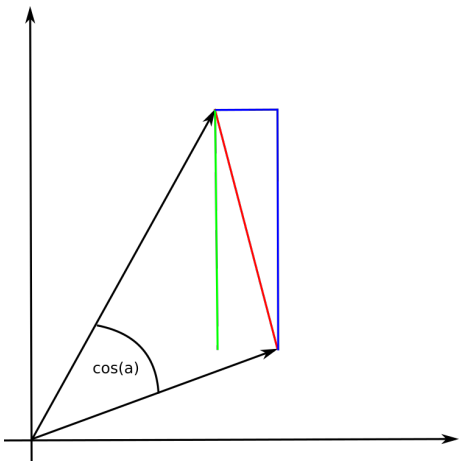


Abbildung 6.13: Ähnlichkeit zweier Vektoren: Euklidische-Norm (rot), Manhattan-Norm (blau), Maximums-Norm (grün) und Cosinus-Ähnlichkeit (schwarz)

Es stellt sich zunächst die Frage, welche Artikel zusammen eine Klasse bilden. So könnte man Artikel einer Warengruppe (z.B. Haushaltswaren oder Bücher) zu einer Klasse zusammenfassen. Jedoch würde dies für die visuelle Erkennung keinerlei Belang haben, da Artikel einer Warengruppe sehr unterschiedlich aussehen können. Beispielsweise können Bleistifte und Druckerpapier eindeutig zur Artikelgruppe Schreibwaren gezuordnet werden, ähneln sich visuell aber sehr wenig. Das hätte zur Folge, dass die Definition der Klasse im Raum der visuellen Merkmale stark diffus ausfällt und eine Zuordnung unmöglich wird. Deshalb werden zur Klasseneinteilung visuelle Merkmale genutzt, welche im Kapitel 3.2.4 bereits behandelt wurden.

Die Klasseneinteilung selbst wird durch die OpenCV-Implementierung des Lloyd-Algorithmus durchgeführt. Der Lloyd-Algorithmus sieht einige Parameterangaben durch den Benutzer vor: k beschreibt die Anzahl der Klassen, in die der Datensatz aufgeteilt werden soll. Weiterhin muss eine Methode zur Wahl der initialen Clusterzentren gewählt werden. Für unsere Testläufe wurden zufällige Clusterzentren und die Initialisierung von k-Means++ nach [10] gewählt. Die benutzte Abstandsmetrik des Algorithmus, die L^2-Metrik, kann in der OpenCV-Implementierung nicht geändert werden.

Da jeder Artikel durch mehrere Bilder beschrieben wird, kommt es bei der Klasseneinteilung vor, dass Bilder des gleichen Artikels verschiedenen Klassen zugeteilt werden. Dies begründet sich dadurch, dass die Ansicht des Artikels auf den Bildern variiert. Beispielsweise zeigen einige Bilder die Vorderseite, während weitere Bilder die Rückseite eines Artikels zeigen. Diese müssen sich nicht zwingend visuell ähnlich sein. Um diese Problem zu lösen wurden pro Artikel nicht nur ein, sondern alle vorkommenden Klassenlabels gespeichert. Durch dieses nötige Vorgehen verringert sich die Information, die durch das berechnete Klassenlabel repräsentiert wird. Um dem entgegen zu wirken kann die Anzahl der Klassen erhöht werden, so dass weniger Artikel einer Klasse zugeordnet werden.

Nach der Vergabe der Klassenlabels für jedes Bild, berechnet der Klassifikator ein statistisches Model, anhand dessen er im Erkennungsbetrieb seine Entscheidungen trifft. Die Wahl des Klassifikationalgorithmus fiel auf die Support Vector Machine (SVM) der OpenCV Library.

6.1.4 Kommunikation

Jede Nachricht ist in ein Google Protocol Buffer Objekt verpackt, siehe Abschnitt 5.2.5.

Für die Kommunikation wird ein TCP-Server gestartet an dem sich das Smartphone anmelden kann. Daher muss die erste Nachricht auch den Benutzernamen und das Passwort des Smartphone-Benutzers enthalten. Sind die Daten fehlerhaft, wird die Verbindung umgehend abgebrochen. War die Anmeldung erfolgreich, fragt das Smartphone den ersten Pick ab. Diese Nachrichten sind relativ klein und passen damit in ein einziges TCP-Paket. Sobald die Bewegungserkennung dem Smartphone mitteilt, dass der Kommissionierer steht, werden Bilder gesendet. Die versendeten Bilder müssen auf mehrere Pakete aufgeteilt wer-

den, weshalb alle eingehenden Nachrichten in dem TCP-Server in einen Buffer geschrieben werden, bis eine fest definierte Zeichenkette empfangen wird, die das Ende der Nachricht beschreibt. Dann wird der Inhalt des Buffers verwendet um die Protocol Buffer Nachricht zu deserialisieren. Danach liegt auf dem Server ein Objekt vor, das genau dem vor dem Versand entspricht.

6.2 Android-App

Die Android-App dient als Schnittstelle zwischen dem Server und der verbundenen Smartwatch. Hier werden die Vorgänge, die im laufenden Betrieb ablaufen, koordiniert und die Kamera des Smartphones zur Aufnahme der Bilder genutzt. Die App bietet die Möglichkeit eine Konfiguration der Verbindung zum Server vorzunehmen und die Kamera an die gegebenen Verhältnisse anzupassen. Zusätzlich erkennt die Android-App den aktuellen Bewegungszustand des Kommissionierers. Hierbei wird zwischen Gehen und Stehen unterschieden und der laufende Kommissioniervorgang und die Kamera den entsprechenden Verhältnissen angepasst. Die Umsetzung dieser Bewegungserkennung wird in Kapitel 6.2.3 vorgestellt. Dieses Kapitel bietet zuerst einen Überblick über die Oberfläche der Android-App und geht danach auf die dahinter liegende Implementierung ein.

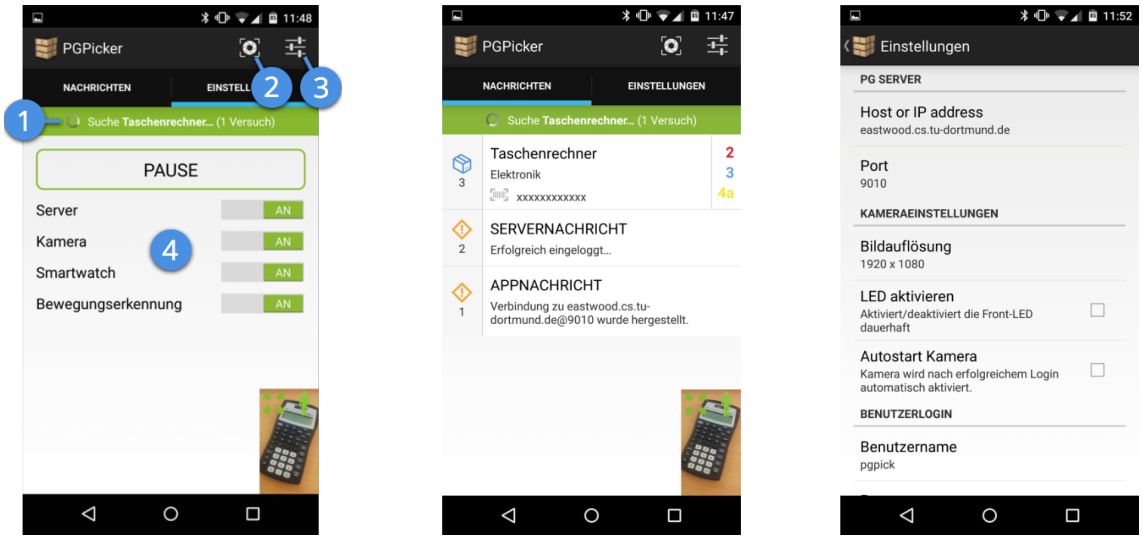
6.2.1 Bedienung & GUI

Die Oberfläche der App bietet insgesamt drei verschiedene Ansichten, die jeweils unterschiedliche Bereiche der Konfiguration und Darstellung des laufenden Prozesses abdecken. Die einzelnen Ansichten sind in Abbildung 6.14 dargestellt. Oberhalb der Fenster zu den Einstellungen und Nachrichten finden sich eine Statusleiste und einzelne Icons für weitere Funktionen.

Die Statusanzeige ((1) in Abbildung 6.14) ermöglicht es dem Kommissionierer den aktuellen Status der Erkennung abzulesen. Dabei sind die in Tabelle 6.1 aufgeführten Statusmeldungen möglich. Bei aktivierter Kamera und vorhandenem Pick erscheint innerhalb der Statusanzeige zusätzlich ein Button, der einen manuellen Start der Erkennung ermöglicht. Ist zusätzlich noch die Bewegungserkennung aktiviert, erscheint an dieser Stelle ein Hinweis, dass ein Stehenbleiben die Erkennung automatisch auslöst.

Zustand	**Anzeige**	**Farbe**
Kein Pick vorhanden	Analyse inaktiv	Grau
Pick erhalten, Kamera inaktiv	Kamera muss aktiviert werden	Orange
Pick erhalten, Kamera aktiv	Erkennung kann gestartet werden	Grün
Analyse aktiv	Gesuchter Artikel, Anzahl Übertragungen	Grün

Tabelle 6.1: Zustände der Statusanzeige in der Android-App



(a) Einstellungen (b) Nachrichten (c) Konfiguration

Abbildung 6.14: Die drei Teilbereiche der Android-App

Über das Fokus-Icon ((2) in Abbildung 6.14) in der Leiste am oberen Bildschirmrand kann zusätzlich eine Steuerung des Autofokus erfolgen. Bei aktivierter Kamera öffnet sich hier der in Abbildung 6.15 dargestellte Dialog. Über diesen Dialog lässt sich der Autofokus der Kamera feststellen. Der Kommissionierer kann dadurch den Fokusbereich der Kamera auf seine eigene Armlänge feststellen und unbeabsichtigtes Fokussieren auf im Hintergrund liegende Bereiche vermeiden. Das ganz rechts positionierte Konfigurations-Icon ((3) in Abbildung 6.14) öffnet die Konfigurationsansicht.

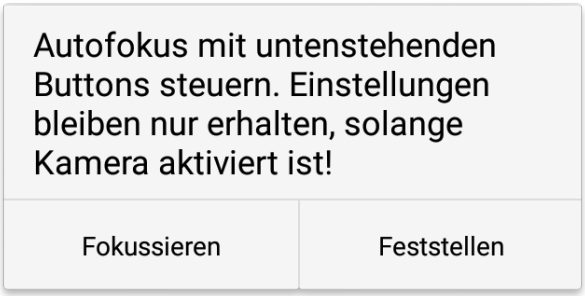


Abbildung 6.15: Dialog zur Konfiguration des Autofokus

Einstellungen

Die Einstellungsansicht ((4) in Abbildung 6.14) erlaubt die Steuerung der einzelnen Verbindungen, sowie der Kamera. Beim Aktivieren des Server-Schalters wird versucht eine Verbindung zur hinterlegten Adresse herzustellen. Ist diese Verbindung erfolgreich, wird

ein Anmeldeversuch mit den eingegebenen Benutzerdaten gestartet. Wurde die Anmeldung vom Server bestätigt, erfolgt ein Abruf der ersten Pickdaten vom Server.

Über den Kamera-Schalter lässt sich die Kamera de- bzw. aktivieren. Dadurch kann der Kommissionierer diese auch während einer laufenden Erkennung abschalten.

Der Schalter für die Verbindung zur Smartwatch erlaubt es dem Kommissionierer den aktuellen Verbindungsstatus zu überprüfen. Ist eine Pebble-Smartwatch verbunden, so ist der Schalter aktiviert. Eine Trennung der Verbindung zur Smartwatch ist hier allerdings technisch nicht möglich. Diese Trennung müsste über die separat installierte Pebble-App erfolgen.

Der Schalter zur Bewegungserkennung aktiviert/deaktiviert diese. Ist der Schalter aktiviert werden Bewegungsdaten gesammelt, ausgewertet und die entsprechenden Komponenten der App über Änderungen des Zustands informiert.

Ein zusätzlicher, zentral positionierter Pause-Button erlaubt die vorübergehende Pausierung der Kommissionierung. Dabei wird die Verbindung zum Server gehalten, jedoch die Kamera abgeschaltet und die Bewegungserkennung gestoppt. Der Kommissionierer kann durch erneutes Betätigen des Buttons die Tätigkeit wieder aufnehmen und setzt seinen Kommissioniervorgang fort.

Nachrichten

Die Nachrichtenansicht listet alle vom Server erhaltenen Nachrichten, sowie weitere in der App entstandenen Ereignisse auf. Die Nachrichten werden aufsteigend nummeriert und es wird zwischen Picknachrichten und Systemnachrichten unterschieden. Picknachrichten beinhalten den Namen des zugehörigen Artikels, den hinterlegten Barcode und die Position aufgeschlüsselt nach Regal, Ebene und Reihe. Systemnachrichten enthalten die eigentliche Nachricht, sowie deren Ursprung, also Server, Android-App oder Smartwatch.

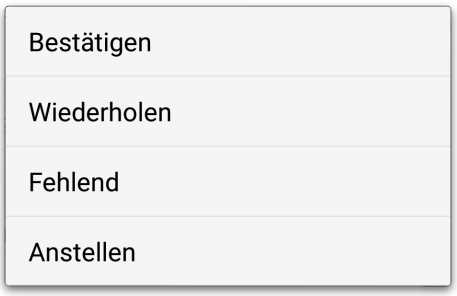


Abbildung 6.16: Kontextmenü zu Picknachrichten

Bei einem langen Drücken auf Picknachrichten öffnet sich das in Abbildung 6.16 gezeigte Kontextmenü. Dieses bietet die gleichen Interaktionsmöglichkeiten, wie das entsprechende Menü auf der Smartwatch, welches in Kapitel 6.3.1 vorgestellt wird.

Konfiguration

Über die Konfigurationsseite lassen sich grundlegende Einstellungen, die zum Betrieb der Android-App nötig sind, vornehmen. Die Einstellungen sind in einzelne Kategorien aufgeteilt um dem Benutzer einen einfacheren Überblick zu ermöglichen.

- **PG Server**
  - **Host/IP-Adresse**: Hostname unter dem der Server im Netzwerk erreichbar ist.
  - **Port**: Der zum Host gehörige Port.
- **Kameraeinstellungen**
  - **Bildauflösung**: Hier wird eine Auswahl der gewünschten Bildauflösung ermöglicht. Die Auswahlmöglichkeiten entsprechen den auf dem Gerät möglichen Auflösungen. Die gewählte Auflösung wird zur späteren Übertragung an den Server verwendet. Hier ist zu beachten, dass stark voneinander abweichende Auflösungen zwischen Testdaten und während der Erkennung verwendeten Bildern zu schlechteren Ergebnissen führen können. Auch besonders geringe Auflösungen können aufgrund weniger übertragenen Details zu schlechteren Ergebnissen führen.
  - **Kamera-LED**: Aktiviert bzw. deaktiviert das Blitzlicht der Kamera als permanente Beleuchtung.
  - **Autostart Kamera**: Wenn aktiviert, wird die Kamera nach erfolgreichem Login automatisch aktiviert.
- **Benutzerlogin**
  - **Benutzername**: Der Benutzername des Kommissionierers.
  - **Passwort**: Das Passwort des Kommissionierers.
- **Bewegungserkennung**
  - ***Preprocessing***: Wenn aktiviert, werden die gesammelten Daten auf dem Gerät lokal gespeichert.
  - **Aktivität**: Der Zustand, der den beim *Preprocessing* gesammelten Daten zugewiesen wird.

6.2.2 Implementierung

Die Android-App lässt sich in vier Bereiche aufteilen. Abbildung 6.17 zeigt alle Bereiche und die zugehörigen Klassen. Die Kommunikationsebene verwaltet die Verbindungen zum Server und zur Smartwatch, wohingegen die Logikebene den Erkennungsvorgang verwaltet

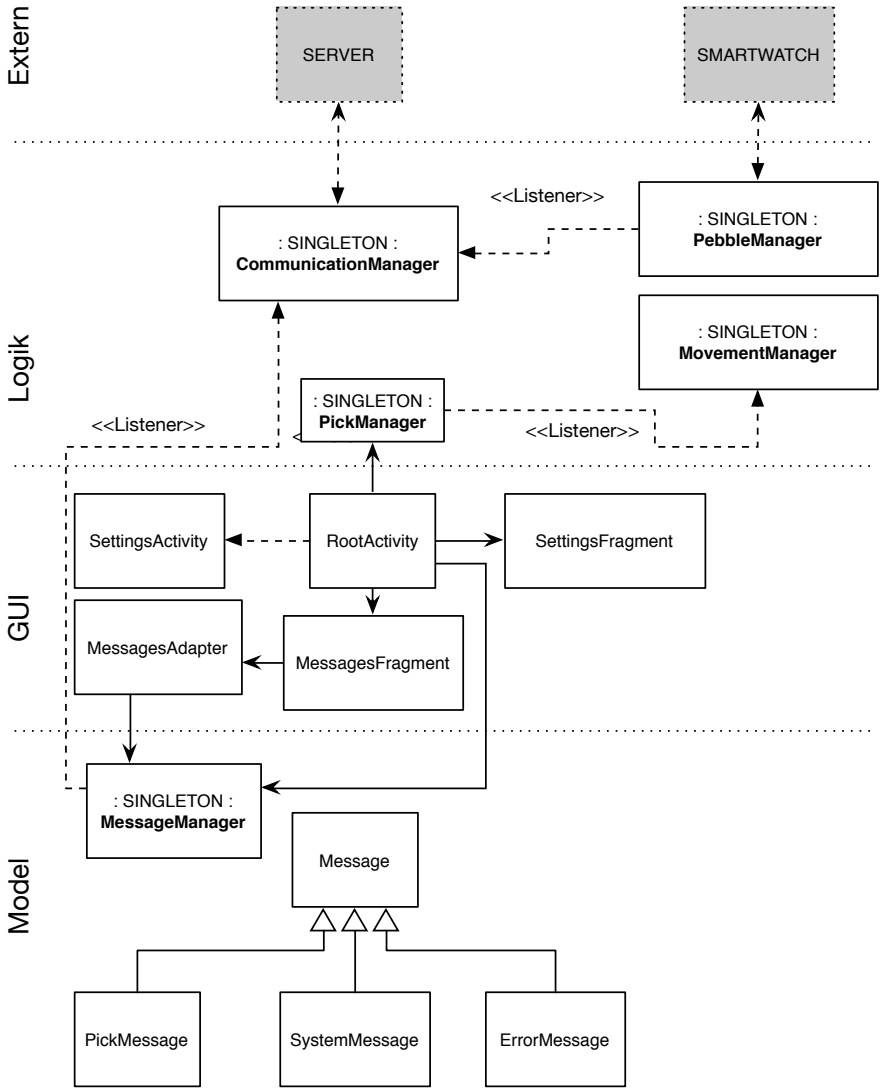


Abbildung 6.17: Klassen und Beziehungen unterteilt nach Ebene

und die Bilder sammelt, die zur Erkennung des Artikels genutzt werden. Mit Hilfe der GUI-Ebene kann der aktuelle Status der Erkennung dem Benutzer präsentiert werden und es wird eine Konfiguration einiger Parameter für die Logikebene erlaubt. Die Model-Ebene beschränkt sich auf eine eigene Nachrichten-Klasse, die innerhalb der Android-App verwendet wird und die von den *Google Protocol Buffern* generierten Klassen.

Kommunikation

Die Kommunikation zwischen Server und App wird zentral im `CommunicationManager` verwaltet. Die beteiligten Klassen sind in Abbildung 6.17 im oberen Logik Bereich zu sehen. Hier wird eine Socketverbindung zum Server gehalten. Zu diesem Zweck ist der `CommunicationManager` als *Singleton* implementiert und die zentrale Instanz lässt sich über

die `getInstance()`-Methode abrufen. Der `CommunicationManager` stellt externen Komponenten über öffentliche Methoden verschiedene Funktionen bereit. Dazu gehören der Login, das Ausführen der über das Kontextmenü und das Menü der Smartwatch erreichbaren Funktionen und der Logout. Innerhalb dieser Funktionen werden die der Anfrage entsprechenden *Google Protocol Buffer* Nachrichten instanziiert und auf den `OutputStream` der Socketverbindung serialisiert. Einen solchen Vorgang kann man anhand der in Listing 6.2 dargestellten login-Methode nachvollziehen.

```
1  public void login() {
2      // Create a user object
3      PickProtos.User user = PickProtos.User.newBuilder()
4          .setUsername(username)
5          .setPassword(password)
6          .build();
7
8      // Create a generic message object and
9      // add the given user
10     PickProtos.Message msg = PickProtos.Message.newBuilder()
11         .setUser(user)
12         .build();
13
14     // Send the object via tcp socket
15     sendProtocolBufferMessage( msg );
16 }
```

Listing 6.2: Login-Methode des CommunicationManagers

Um über eingehende Nachrichten benachrichtigt zu werden, können sich andere Komponenten als *Listener* beim `CommunicationManager` registrieren. Dazu müssen diese das `CommunicationManagerListener`-*Interface* implementieren, welches die vom `CommunicationManager` aufgerufenen Methoden definiert. Dabei entspricht jeder der Methoden einem anderen Nachrichtentyp. Das *Interface* ist in Listing 6.3 aufgeführt.

```
1  public interface CommunicationManagerListener {
2      // A new pick was received
3      public void onPickMessageReceived(PickMessage message);
4
5      // A a generic message was received
6      public void onSystemMessageReceived(SystemMessage
           message);
7      public void onErrorMessageReceived(ErrorMessage message);
8
9      // An image was analyzed and the corresponding
```

```
10      // status was returned (right, wrong, unknown or error)
11      public void onPickStatusMessageReceived(PickStatusMessage
            message);
12
13      // A manual action, started by the user was
14      // executed on the server side. Eg. a pick
15      // was skipped on the smartwatch and the
16      // server has finished processing this action
17      public void onPickActionResponseMessageReceived();
18  }
```

Listing 6.3: CommunicationManagerListener-Interface

Die Kommunikation mit der Pebble-Smartwatch erfolgt über den `PebbleManager` und wird in Kapitel 6.3.4 im Detail vorgestellt.

Logik

Zur Verwaltung des Ablaufs der Erkennung wird eine Instanz des `PickManagers` als Instanzvariable der `RootActivity` angelegt. Auch der `PickManager` ist in Abbildung 6.17 im Logik Bereich angesiedelt und verbindet GUI und die Kommunikation. Diese reagiert auf Ereignisse, die die Erkennung beeinflussen und verwaltet abhängig vom aktuellen Zustand die Zustandsanzeige sowie die Kamera. Die Verteilung der Ereignisse innerhalb der App erfolgt über die greenrobots *EventBus*-Bibliothek [18]. Dadurch ist keine feste Koppelung des `PickManagers` mit allen anderen Komponenten nötig und die Behandlung der Ereignisse kann durch die Bibliothek direkt auf dem *Main-Thread* erfolgen. Von jeder Komponente aus können dazu generische *Event*-Objekte auf den *EventBus* gesendet werden und jede Komponente, die eine entsprechende Methode implementiert hat, wird über dieses Event benachrichtigt. Durch die gezielte Benachrichtigung auf dem *Main-Thread* ist auch eine Veränderung der GUI als Reaktion auf ein Ereignis direkt in der Behandlungsmethode möglich. Bei Modifikation der GUI auf einem anderen *Thread* würde die Android-Umgebung einen Absturz der App auslösen.

Der gesamte Ablauf kann in Abbildung 6.18 nachvollzogen werden. Auslöser des Vorgangs ist das Empfangen eines neuen Picks im `CommunicationManager`. Zu diesem Zeitpunkt wird der Pick über den *EventBus* verschickt und der `PickManager` wechselt in den "Pick erhalten" Zustand. Ist die Kamera nicht aktiviert, wird der Benutzer durch eine angepasste Statusanzeige auf diesen Umstand hingewiesen. Steht die Kamera bereits zur Verfügung, kann die Erkennung gestartet werden. Um diese auszulösen existieren drei verschiedene Möglichkeiten. Zum einen kann ein zusätzlicher Button unterhalb der Statusanzeige gedrückt werden oder zum anderen kann ein langer Druck auf den **SELECT**-Button der Smartwatch die Erkennung auslösen. Ideal ist jedoch das Auslösen über die Bewegungserkennung. Erkennt diese einen "Stehen"-Zustand, so löst der PickManager selbst die

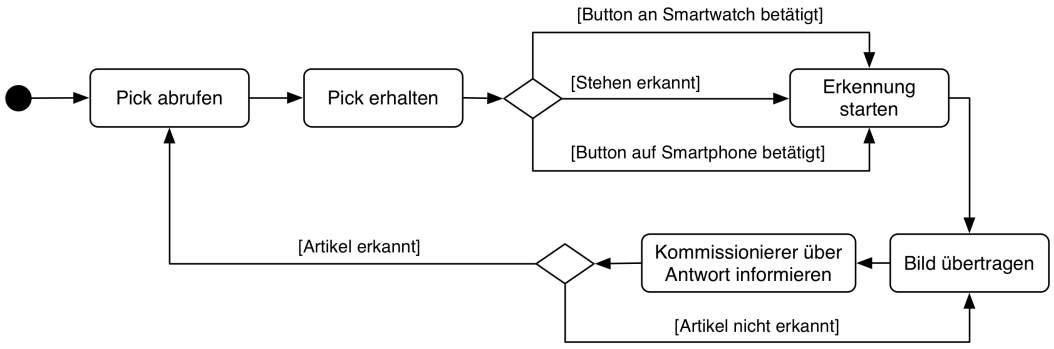


Abbildung 6.18: Ablauf der Erkennung im PickManager

Erkennung aus, indem das erste Bild an den Server übertragen wird. Die Startmöglichkeiten über die Buttons dienen in erster Linie als optionale Möglichkeit, falls der automatische Start über die Bewegungserkennung fehlschlägt.

Alle für die Erkennung verwendeten Bilder werden aus der Schnittstelle zur Kameravorschau extrahiert. Dazu werden die Rohdaten, die von der Kamera auf die permanent sichtbare Kameravorschau-Fläche gezeichnet werden, in einen Ringpuffer geschrieben. Dieser Ringpuffer hält ständig die letzten drei Bilder vor. Der Vorteil gegenüber der Aufnahme von Bildern über die Auslösung der Kamera liegt hier im stark reduzierten Speicherverbrauch und der Vermeidung von Artefakten durch die JPEG-Kompression. Soll ein solches Bild an den Server übertragen werden, wird das aktuellste Bild aus dem Ringpuffer entnommen und in das *Google Protocol Buffer*-Format kodiert. Die so erhaltene Nachricht wird daraufhin über den `CommunicationManager` versendet.

Zu jedem versendeten Bild erhält die App ein `ImageResponse`-Objekt vom Server. Dieses enthält den Status der durchgeführten Erkennung. Die möglichen Zustände sind **Korrekter Artikel**, **Falscher Artikel**, **Nicht erkannt** und **Fehler**. Im ersten Fall, also einem korrekt erkannten Artikel, beendet der `PickManager` den Pickvorgang und ruft einen neuen Pick ab. Der Erkennungsprozess startet neu. Da die Kamera bereits aktiviert ist, springt der Ablauf dabei direkt zum Wartezustand, der den Start der Erkennung durch Buttons oder die Bewegungserkennung abwartet. In den anderen Fällen wird ein weiteres Bild aus dem Ringpuffer entnommen und übertragen. Es wird also ein weiterer Versuch der Erkennung unternommen. Nach **5** fehlgeschlagenen Erkennungsversuchen wird eine Systemnachricht generiert, die den Benutzer dazu auffordert den Barcode des Artikels näher vor die Kamera zu halten. Parallel zur Behandlung der `ImageResponse`-Objekte wird der Benutzer über die Pebble-Smartwatch über die Vorgänge informiert.

GUI

Die Oberfläche der Android-App nutzt die vom Android-SDK bereitgestellten UI-Elemente. Alle beteiligten Klassen und Objekte sind in Abbildung 6.17 im GUI Bereich abgebildet.

Die Bereiche Nachrichten und Einstellungen aus Abbildung 6.14 sind als `Fragments` angelegt und werden innerhalb der `RootActivity` von einem `FragmentAdapter` innerhalb eines `ViewPagers` dargestellt. Dies erlaubt die Navigation über die Tab-Schaltflächen am oberen Bildschirmrand und entspricht einem üblichen Bedienungsmuster auf Android-Geräten.

Das Nachrichten-`Fragment` beinhaltet einen `ListView`. Dieser wird von einem `MessagesAdapter` mit Zeilen gefüllt und bildet die `TreeMap` des `MessageManager`s ab. Diese `TreeMap` nimmt jede im System erstelle Nachricht auf und dient auch als Quelle für die auf der Smartwatch dargestellten Inhalte. Auf diese Weise wird sichergestellt, dass die Anzeigen auf der Smartwatch und in der Android-App stets dem gleichen Stand entsprechen. Zu diesem Zweck kommt auch beim `MessageManager` wieder das *Singleton*-Muster zum Einsatz.

Das Einstellungen-`Fragment` bietet über die Schalter die Möglichkeit zur Steuerung der einzelnen Komponenten. Dabei rufen die einzelnen Schalter auf den vorhandenen Instanzen lediglich `start()` bzw. `stop()` Methoden auf. Die dadurch ausgelösten Ereignisse führen dann zu einer Statusänderung der Schalter. So ist ein direktes Feedback über erfolgreich durchgeführte Aktionen für den Kommissionierer gegeben.

Die über das oben rechts platzierte Icon erreichbare und in Abbildung 6.14 rechts zu sehende Konfiguration ist als eigenständige `Activity` implementiert und erbt dazu von der systemeigenen `PreferenceActivity`. Dadurch musste zur Auflistung der Konfigurationsmöglichkeiten nur noch eine XML-Datei erstellt werden, die die Optionen vorgibt und Konstanten zuweist. Alle hinterlegten Einstellungen werden über die vom System bereitgestellten `SharedPreferences` gesichert und verwaltet.

Model

Die Model-Ebene beschränkt sich auf die `Message`-Klasse und ihre Varianten, die in Abbildung 6.17 im entsprechenden Bereich abgebildet sind. Je nach Nachrichtentyp existieren verschiedene Attribute. Die `PickMessage`-Klasse erlaubt beispielsweise den Zugriff auf den jeweiligen Pick und dessen Standort. Alle `Message`-Objekte werden im `MessageManager` in einer `TreeMap` abgelegt und über ihren Schlüssel innerhalb dieser `TreeMap` zugeordnet. Der `MessageManager` bietet hierbei eine zentrale Verwaltung der im System befindlichen Objekte, die von höher liegenden Ebenen genutzt wird. Dazu existiert ein globaler Nachrichtenzähler, der bei Initialisierung einer neuen Nachricht als Schlüssel für diese Nachricht verwendet und dann erhöht wird. Dies erlaubt es der Smartwatch Aktionen für Nachrichten gezielt auszulösen. Dabei werden nach erfolgreicher Durchführung von destruktiven Aktionen, wie das erneute Anstellen eines Picks ans Ende der Pickliste, die zugehörigen Nachrichten aus der Liste entfernt. Dies verhindert das wiederholte Durchführen der Aktionen.

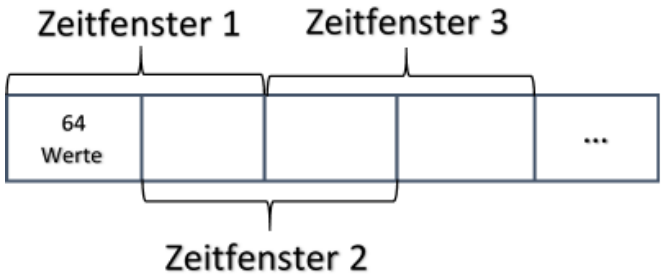


Abbildung 6.19: Modell der Zeitfenster zur Speicherung von Sensordaten

6.2.3 Bewegungserkennung

Im Rahmen der Android App wurde eine Bewegungserkennung implementiert, die dann eine Erkennung startet, wenn der Kommissionierer vor einem Regal im Lager steht und dabei ist einen Pick durchzuführen. Damit wird gewährleistet, dass Bilder für die Bilderkennung auf dem Server nur dann bereitgestellt werden, wenn ein Kommissioniervorgang erfolgen soll. Dies hat auch zur Folge, dass die Arbeitslast auf dem Server insgesamt geringer ausfällt, da unbrauchbare Aufnahmen vermieden werden. Die Bewegungserkennung arbeitet mit den Sensordaten des Beschleunigungssensors und wird damit auch den Anforderungen gerecht den Stromverbrauch so gering wie möglich zu halten und keine hohe Auslastung für die Applikation darzustellen. Das Vorhaben zur Implementierung einer Handdetektion ist dadurch nicht mehr notwendig, da die Bewegungserkennung es ermöglicht die Kommissioniervorgänge von Laufbewegungen zuverlässig zu differenzieren. Ein weiteres Argument, was für diese Umsetzung spricht, ist dass der Kommissionierer von sehr hoher Erkennungsgeschwindigkeit profitiert.

Beschleunigungssensor

Der Beschleunigungssensor des Smartphones misst die momentane Beschleunigung in m/s^2, wobei das Koordinatensystem relativ zum Gerät abgebildet wird. Es wird insbesondere der lineare Beschleunigungssensor verwendet, der bis auf die herausgerechnete Schwerkraft, dieselben Werte liefert wie der normale Beschleunigungssensor. Eine Einschränkung der Android API zum aktuellen Zeitpunkt ist, dass eine genaue Abtastrate der Sensoren weder eingestellt noch garantiert werden kann. Es kann lediglich aus einer Auswahl von festen Konfigurationen gewählt werden, die sich in *SENSOR_DELAY_NORMAL*, *SENSOR_DELAY_UI* und *SENSOR_DELAY_FASTEST* unterteilen. In der vorliegenden Implementierung wird Letzteres verwendet, um größere Schwankungen zu vermeiden.

Ablauf der Bewegungserkennung

Der Ablauf der Bewegungserkennung sieht es grundsätzlich vor, dass die Sensordaten vom linearen Beschleunigungssensor erfasst und einer Vorverarbeitung unterzogen werden, be-

vor diese mit dem Klassifizierungsalgorithmus weiterverarbeitet werden können. Für die Vorverarbeitung, aber auch für die spätere Verwendung der Bewegungserkennung werden die erfassten Daten in Zeitfenster unterteilt (Abbildung 6.19). Ein Zeitfenster wird geschlossen, sobald genau 128 Werte vom Sensor erfasst wurden. Diese Anzahl entspricht ungefähr, bei einer Abtastrate von ca. 60 Hz, einem Zwei-Sekunden Ausschnitt aus den Sensordaten. Die optimale Länge des Zeitfensters hängt von der durchgeführten Bewegung ab und wurde dementsprechend ausgehend davon, dass für einen Bewegungsschritt ungefähr eine Sekunde ausreichend ist, dieser Wert bestimmt [25]. Auf diesen Rohdaten werden die Mittelwertberechnung, der Vektorbetrag und anschließend die Standardabweichung auf allen Achsen berechnet, um dem Klassifizierungsalgorithmus ausdrucksstarke Daten liefern zu können. Dieser Prozess wird auch als *Merkmalsextraktion* bezeichnet. Er ist von sehr hoher Bedeutung, da die Rohdaten keine ausreichenden Informationen über die durchgeführte Bewegung an sich liefern. In jedem Zeitfenster werden mittels der durchgeführten statistischen Berechnungen die relevanten Daten gefiltert. Auch werden die Zeitfenster zu 50% überlappt, um die Übergänge von zwei verschiedenen Bewegungsformen akkurat zu behandeln. Im Training werden die gefilterten und bereits gelabelten Daten durch einen

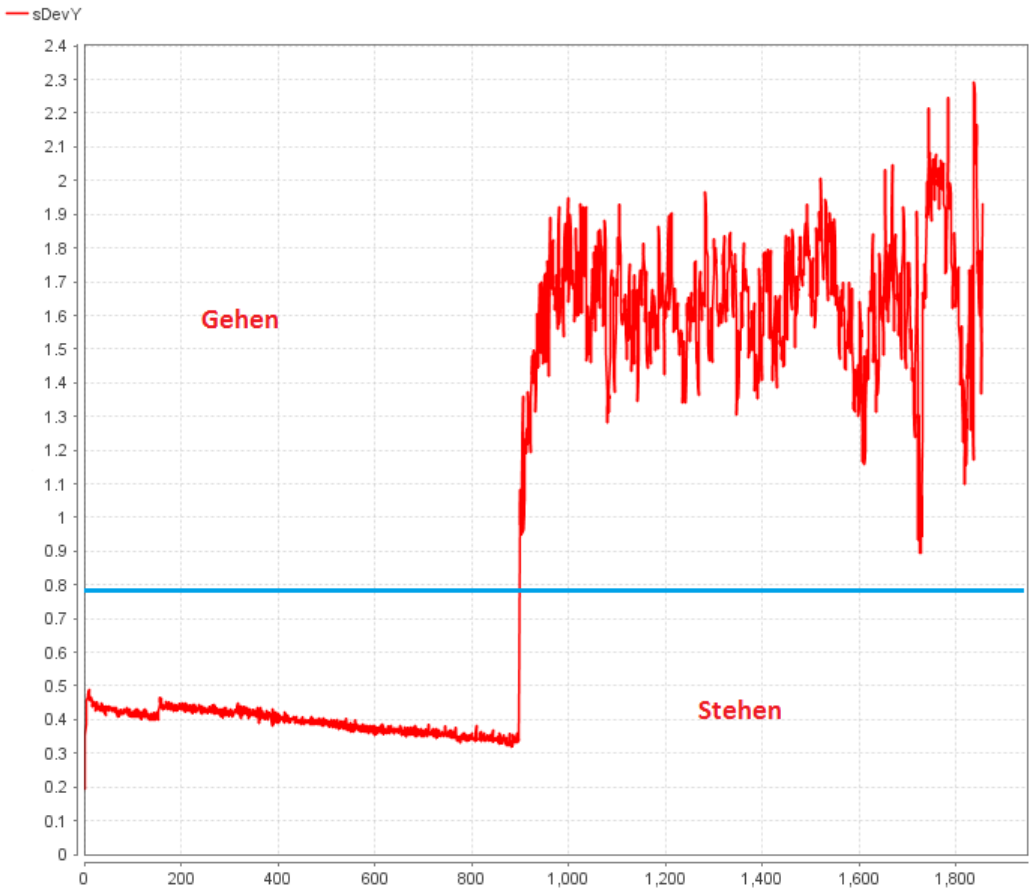


Abbildung 6.20: Streuung des Bewegungsprofils anhand der Standardabweichung auf der y-Achse

Klassifizierungsalgorithmus ausgewertet, sodass zwischen Gehen und Stehen differenziert werden kann. Dieser Klassifizierungsalgorithmus arbeitet mit einem Entscheidungsbaum. Im Training konnte eine ausschließlich gehende Bewegung, bei unterschiedlichem Tempo, klar von einer stehenden Bewegungsform unterschieden werden. Die Auswertung der Daten durch den Klassifizierungsalgorithmus erfolgt hierbei über die Data Mining Software RapidMiner Studio. Dieser liefert u. a. in dem Plot zum Bewegungsprofil (siehe Abbildung 6.20) wichtige Erkenntnisse über die ermittelten Werte zur Unterscheidung der beiden Bewegungsformen. Hierbei können alle Werte die oberhalb der eingezeichneten blauen Linie zu der Bewegung *Gehen* und alle Werte unterhalb dieses Schwellwertes der Bewegung *Stehen* zugeordnet werden. Befindet sich schließlich die Bewegungserkennung im Einsatz, werden anhand der durch diesen Vorverarbeitungsschritt ermittelten Schwellwerte des Entscheidungsbaumes die Bewegung zur Ausführungszeit bestimmt.

6.2.4 Kommunikation Android zu Pebble

Zwischen Android-App und Pebble findet ein Nachrichtenaustausch über Bluetooth statt, welcher genau festgelegt ist und von beiden Seiten eingehalten werden muss. Die Kommunikation soll sicher stellen, dass das Smartphone folgende Inhalte an die Pebble senden kann:

- Daten über einen neuen Pick
- Statusnachrichten vom Server
- Systemnachrichten der Android-App

Nachrichten, die von der Android-App an die Pebble gesendet werden, bestehen immer aus Tupeln in Form von Schlüssel/Wert-Paaren. In einem `PebbleDictionary`, welches in einem Sendevorgang an die Pebble gesendet wird, werden immer nur zwei Tupel gespeichert. Das erste Tupel, welches ein solches `PebbleDictionary` enthält, besteht aus einem `MESSAGE_TYPE` als Schlüssel und einem weiteren Element als Wert. `MESSAGE_TYPE` ist dabei einer der folgenden Strings als Wert zugewiesen:

- **pick**: Daten eines Picks
- **system**: Systemnachricht (z. B. bei Verbindungsfehlern)
- **error**: Fehlernachricht
- **status**: Der Status eines Picks (z. B. `WRONGPICK` bei einem falsch ergriffenem Artikel)
- **count**: Zähler für Erkennungsversuche des Servers

Weitere mögliche Schlüssel für Tupel: `MESSAGE_ID, DESCRIPTION, CATEGORY, BARCODE, SHELF, ROW, COLUMN, NAME, COUNT` und `STATUS`. Diesen Schlüsseln, welche intern als Integer gehandhabt werden, werden innerhalb der Tupel Werte entsprechend ihrer Bezeichnung zugewiesen.

6.3 Pebble

Die Pebble-Smartwatch stellt das zentrale Bedienelement des Kommissionierers dar. Sie besteht aus einem visuellen Part der Benutzung und Vibrationen in Form bestimmter Muster. In diesem Abschnitt wird daher zunächst auf diese beiden Punkte eingegangen, um im Anschluss genauer auf die Implementierung einzugehen.

6.3.1 Bedienung & GUI

Die Pebble wird mithilfe von vier Buttons bedient (siehe Abbildung 6.21). **UP** und **DOWN** ermöglichen eine Navigation innerhalb von Listen oder Ansichten, die sich über die Größe eines Fensters erstrecken. In der Picker-Watch Anwendung haben diese Buttons außerdem die Funktion durch die auf dem Smartphone gespeicherte Pickliste navigieren zu können.

Abbildung 6.21: Steuerung der Pebble über vier Buttons

Ist die Picker-Watch App noch nicht gestartet, so kann diese mit einem **SELECT**-Klick auf Picker-Watch (Abbildung 6.22) gestartet werden.

Abbildung 6.22: Starter im Hauptmenü der Pebble, der zum Starten der Anwendung verwendet wird

Ist die Picker-Watch Anwendung gestartet, so sieht der Benutzer, sobald ein Pick eingetroffen ist, die in Abbildung 6.23 gezeigte Übersicht. Diese Anzeige beschränkt sich auf ein Minimum und zeigt lediglich den Namen des zu pickenden Artikels, einen Zähler für nicht erfolgreiche Erkennungsversuche sowie den Lagerort des Artikels an. Der Lagerort gliedert sich in (von links nach rechts): Regal, Ebene und Fach.

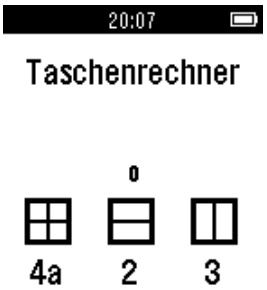


Abbildung 6.23: Anzeige eines Picks auf der Pebble Smartwatch

(a) Details

(b) Bestätigen

(c) Wiederholen

(d) Fehlend

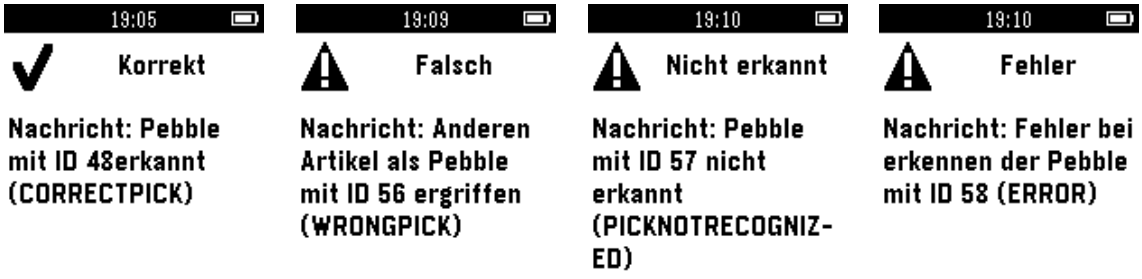
(e) Anstellen

Abbildung 6.24: Menü zu einem aktuellen Pick, in welchem ein Pick bestätigt, als fehlend markiert oder angestellt werden kann. Weiter Können weitere Details angezeigt sowie der Pick wiederholt werden.

Reichen einem diese Informationen nicht aus oder möchte der Kommissionierer manuell in den aktuellen Pick eingreifen, so hat er in einem Menü, welches sich mit einem kurzen **SELECT**-Klick auf der Übersichtsseite öffnet, die in Abbildung 6.24 dargestellten Möglichkeiten. **Details** führt zu einer Übersichtsseite, auf welche im nächsten Abschnitt eingegangen wird. **Bestätigen** bestätigt den aktuellen Pick. Durch diesen Eintrag hat der Kommissionierer die Möglichkeit den Artikel manuell zu bestätigen, falls er sich sicher ist, dass er den richtigen Artikel gegriffen hat. Liefert die Erkennung ein falsches Ergebnis (z. B. **falscher Artikel** bei korrekt gegriffenem Artikel), so kann dies durch **Wiederholen** zurückgesetzt werden. Der Kommissionierer kann einen Artikel auch als **Fehlend** markieren

(a) Details (b) Details Pt. 1 (c) Details Pt. 2 (d) Details Pt. 3

Abbildung 6.25: Detailanzeige zu dem aktuellen Pick, in welchem der Name, der Lagerort, die Kategorie, eine Beschreibung sowie der Barcode des zu pickenden Artikels angezeigt werden.



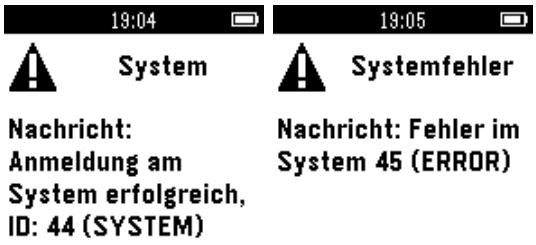
(a) Korrekt (b) Falsch (c) Nicht erkannt (d) Fehler

Abbildung 6.26: Nachrichten zum Status des aktuellen Picks

oder ihn **Anstellen**. Ersteres dient dazu einen Artikel zu kennzeichnen, der nicht an dem vorgesehenen Lagerort zu finden ist. Dies wird dem Server durch die Auswahl von **Fehlend** mitgeteilt und dort entsprechend behandelt. Mittels **Anstellen** kann der Kommissionierer dem Server mitteilen, dass er den aktuellen Pick nicht direkt sondern erst später behandeln möchte. Der Pick wird in diesem Fall an das Ende seiner persönlichen Pickliste gesetzt.

In den Pickdetails finden sich alle Informationen, die dem Smartphone von dem Server bzgl. eines Picks mitgeteilt wurden. Diese werden in einem Fenster, welches über die Grenzen der Pebble hinausragt, angezeigt. Mithilfe der **UP**- und **DOWN**-Tasten kann der Kommissionierer in diesem Fenster navigieren und so alle Informationen erfahren. So kann er bspw. den Barcode des in dem Pick enthaltenen Artikels mit dem des ergriffenen Artikels abgleichen, falls dies über die Erkennung fehlschlägt. Aber auch detailliertere Beschreibungen sind in den Details verfügbar, wodurch er einen Artikel ggf. einfacher identifizieren/finden kann.

Neben der Anzeige des aktuellen Picks sowie den Operationen auf diesem existieren auch noch Fenster für **Statusnachrichten** (siehe Abbildung 6.26). Die Statusnachrichten kommen von Seiten des Servers und werden von dem Smartphone an die Smartwatch weitergeleitet. Statusnachrichten stellen die Reaktionen des Servers auf gesendete Inhal-



(a) Systemnachricht (b) Systemfehler

Abbildung 6.27: Statusnachrichten des Systems (Server/Android), die nicht direkt mit einem Pick in Verbindung stehen

te dar, bspw. dem erfolgreichen Erkennen eines Artikels von Seiten des Servers oder der Bestätigung bei einem manuellen Akzeptieren eines Picks. Diese Fenster überlagern das zuvor angezeigte Fenster und werden (mit Ausnahme einer Fehlernachricht) nach 10 Sekunden wieder geschlossen. Die Fenster können auch, wenn der Kommissionierer nicht warten möchte, mithilfe des **BACK**-Buttons sofort geschlossen werden. Im Falle des Schließens kehrt die Smartwatch zum zuvor angezeigten Bildschirm zurück.

Neben den Statusnachrichten, die von dem Server stammen, existieren auch noch Systemnachrichten, welche von Android produziert und an die Smartwatch gesendet werden (siehe Abbildung 6.27). Diese Nachrichten werden genauso wie Statusnachrichten behandelt und auch in diesem Fall werden Fehlernachrichten nicht automatisch geschlossen.

6.3.2 Vibration

Zusätzlich zu der GUI der App existieren Vibrationsmuster, die eine weitreichende Verwendung der Smartwatch ohne die Verwendung des Displays ermöglichen sollen. Der Kommissionierer weiß durch ein bestimmtes Muster direkt, ob ein Pick vom Server bestätigt wurde oder ob ein Fehler aufgetreten ist. In der App wird von folgenden Mustern Gebrauch gemacht (die Benamungen entsprechen denen in Abbildung 6.26 und Abbildung 6.27):

6.3.3 Implementierung

Die Pebble-App besteht aus mehreren Komponenten. Abbildung 6.28 gibt einen Überblick über diese und deren Beziehungen zueinander. Die Komponenten werden nachfolgend genauer beschrieben.

main Dies ist die Hauptklasse der Anwendung. Hier werden bspw. eingehende Nachrichten behandelt und ein Pick angezeigt.

Ereignis	Muster (Zeit im ms)
Korrekt	400:an, 100:aus, 100:an, 100:aus, 100:an
Falsch	100:an, 100:aus, 400:an, 100:aus, 100:an
Fehler	400:an, 100:aus, 400:an, 100:aus, 400:an
Systemnachricht	100:an, 200:aus, 100:an
Systemfehler	800:an, 200:aus, 800:an, 200:aus, 800:an
Neuer Pick	400:an
Bildersenden beginnen (Bestätigung)	100:an

Tabelle 6.2: Vibrationsmuster in der Pebble-App (in der Form **Millisekunden:Vibrationsstatus**)

PickData `PickData` hält in der App die Daten eines Picks vor. Wird von innerhalb der `main`-Klasse ein neuer Pick empfangen, so werden die in dieser Klasse gespeicherten Werte durch diese ersetzt. Auf der Pebble wird demnach immer nur ein Pick vorgehalten. Die Möglichkeit der Navigation innerhalb der Pick-Historie wird durch eine Kommunikation mit der Android-App realisiert, welche eine Historie über alle Picks bereithält. Diese Klasse ermöglicht außerdem die Ausgabe eines Picks in Form eines einzigen langen Strings, welcher von `PickDetails` zur Anzeige aller Details verwendet wird. Weiter werden in dieser Klasse diverse Puffer bereitgestellt, welche innerhalb der gesamten App ausgelesen werden können und bspw. zum Speichern der Texte auf `TextLayer`n verwendet werden.

PickDetails Erzeugt einen `ScrollLayer` innerhalb eines neuen Fensters, welches bei einem Klick auf Details im `PickMenu` geöffnet wird. Liest einen String aller Details aus der Klasse `PickData` aus und stellt diesen dar.

PickMenu Erzeugt einen `MenuLayer` mit einem darin enthaltenen Menü, welches die Optionen für Bestätigen, Wiederholen, Fehlend, Anstellen (siehe Abschnitt 6.3.4) sowie Details bereitstellt. Bei einem Klick auf Details wird das Fenster von `PickDetails` erzeugt. Die anderen Optionen sind mit einem Sendevorgang an die Android-App verbunden.

PickStatus `PickStatus` stellt Statusnachrichten sowie Systemnachrichten der Android-App dar und besitzt zu diesem Zweck einen Timer, welcher, wenn aktiviert, das Fenster nach 10 Sekunden schließt.

SharedData Enthält Werte, die bspw. für die Kommunikation wichtig und genau definiert sind.

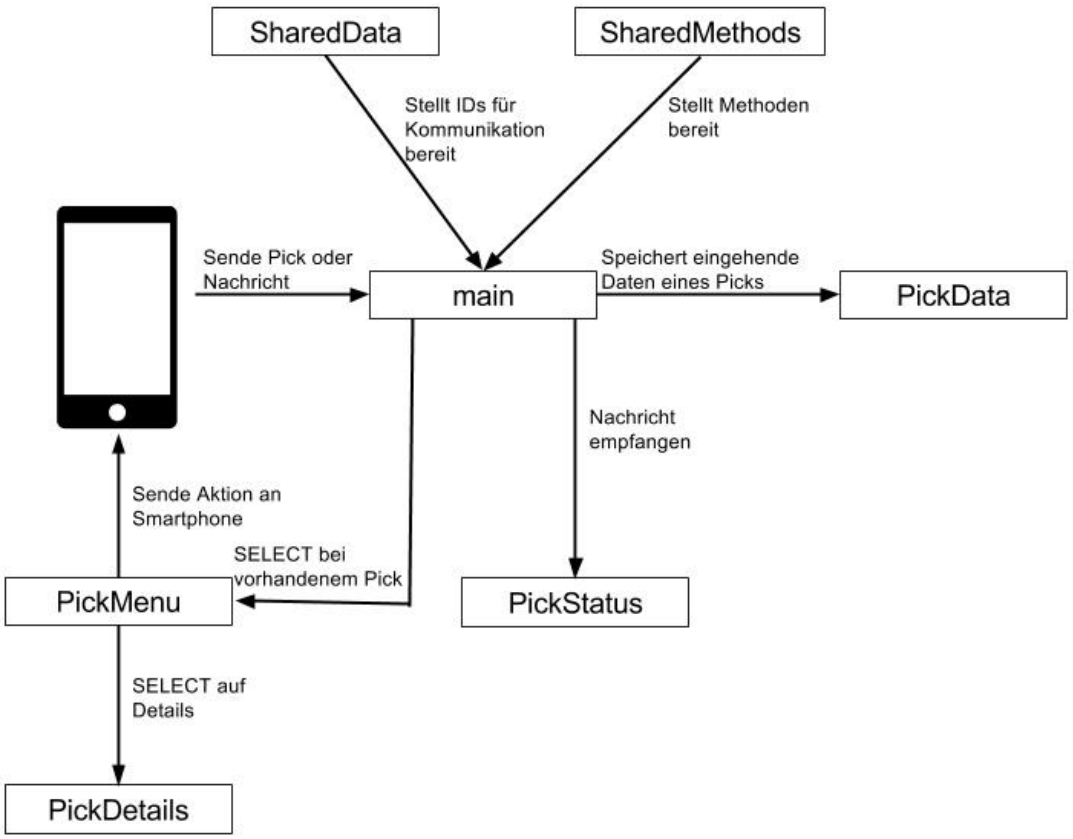


Abbildung 6.28: Elemente der Pebble-App auf Ebene der Programmierung und Beziehungen unter diesen

SharedMethods Enthält Methoden, welche an mehreren Stellen innerhalb der App, insbesondere in unterschiedlichen Klassen, verwendet werden. Die Methode, die Vibrationsmuster bereitstellt und Vibrationen auslöst befindet sich in dieser Klasse.

6.3.4 Kommunikation Pebble zu Android

Zwischen Pebble und Android-App findet ein Nachrichtenaustausch über Bluetooth statt, welcher genau festgelegt ist und von beiden Seiten eingehalten werden muss. Die Kommunikation soll sicher stellen, dass von Pebble folgende Aktionen ausgeführt und an die Android-App gesendet werden können:

- **Bestätigen**: Es wurde der richtige Artikel ergriffen
- **Wiederholen**: Der Artikel wurde falsch erkannt
- **Fehlend**: Artikel ist nicht vorhanden
- **Anstellen**: Pick wird zurück gestellt

In dieser Kommunikationsrichtung wird vor allem das primäre Format "MESSAGE_TYPE / MESSAGE_ID" verwendet. `MESSAGE_TYPE` ist jedoch keiner der in der Gegenrichtung verwendeten Strings, sondern einer von folgenden Integern:

- Bestätigen = 1
- Wiederholen = 2
- Fehlend = 3
- Anstellen = 4

`MESSAGE_ID` ist die ID des Picks, auf welchen sich die aktuelle Operation bezieht. Neben dem primären Format kann auch noch das Schlüssel/Wert-Paar "INIT_IMAGE_SENDING / MESSAGE_ID" an die Android-App gesendet werden. Durch dieses Paar wird der Android-App mitgeteilt, dass der Nutzer die Erkennung manuell starten möchte. Diese Funktion wird bei einem langen Klick auf die SELECT-Taste der Pebble ausgelöst.

KAPITEL 7

Evaluation

Auf allen von der Projektgruppe verwendeten Geräten mussten gewisse Entscheidungen bzgl. des Softwaredesigns getroffen werden. In diesem Kapitel werden die getroffenen Entscheidungen, einige aufgetretene Probleme sowie Problemlösungen erläutert.

7.1 Server

Als Server wurde im Rahmen der Projektgruppe ein Lenovo ThinkPad X200 mit 4GB Arbeitsspeicher und einem Intel Pentium Core2Duo P8400 sowie einer SSD ausgewählt. Eine solche Zusammenstellung kann zum jetzigen Zeitpunkt als mittlerer Standard auf dem bestehenden Markt bezeichnet werden. Als Betriebssystem dient Ubuntu 14.10 64-Bit, die verwendeten Bibliotheken und Tools sind in folgenden Versionen vorhanden: Qt 5.3.0, OpenCV 2.4.9, ZBar 0.10, CMake 2.8.12, gcc 4.9.1. Die Rechenleistung der in dem Server verbauten Komponenten ist stark genug, um unter anderem die verschiedenen Erkennungsmethodiken ausreichend schnell ausführen zu können. Dennoch sind bei der Entwicklung des Servers einige Probleme aufgetreten.

Barcode-Detektion Da die ZBar-Bibliothek das gegebene Bild zeilenweise durchläuft, ist hier eine starke Abhängigkeit der Laufzeit von der Größe des übermittelten Bildes festzustellen. Die Laufzeit zum Scannen eines Bildes in Full-HD-Auflösung beträgt im Durchschnitt ca. 2 Sekunden auf dem o.g. System. Um diese Zeit zu verringen ist es notwendig, nur Teile des Bildes zu scannen. Mit der im Kapitel 6.1.3 beschriebenen Vorverarbeitung beträgt die gesamte Laufzeit nur noch 0,3 Sekunden (Vorverarbeitung + Scannen). Diese Zeit könnte noch verbessert werden, indem die Teilbereiche des Bildes in mehreren Threads

Verfahren	Erkennungsrate	Laufzeit
Ohne Schärfung	78%	0,26s
Mit Schärfung	89%	0,31s

Tabelle 7.1: Erkennungsraten und Laufzeiten der Barcode-Detektion

parallel auf Barcodes gescannt werden, aktuell wird dies für alle Teile nacheinander durchgeführt.

Um die Erkennungsquote der Implementierung bewerten zu können, wurde ein Test auf einem Satz aus Artikelbildern durchgeführt. Das Ergebnis ist in Tabelle 7.1 zusammengefasst und wird im Folgenden weiter erläutert. Der Datensatz bestand aus 322 Bildern, von denen 85 Bilder einen Barcode enthalten. Auf 35 Bildern wurde ein Barcode erkannt, dies entspricht 41% der möglichen Bilder. Im Folgenden wurde untersucht, wieso eine Erkennung der restlichen Bilder nicht möglich war. Dabei zeigte sich, dass die Linien der Barcodes auf 40 der nicht erkannten Bilder teilweise verschwommen waren. Da die Codierung eines Barcodes auf den unterschiedlichen Strichbreiten innerhalb des Barcodes basiert, müssen diese Unterschiede klar erkennbar sein. Wertet man diese Bilder auch als für den Algorithmus nicht erkennbar, liegt die Erkennungsrate bei 78%. Um die Erkennungsquote weiter zu steigern, wurde ein weiterer Verarbeitungsschritt eingefügt. Die Bildausschnitte mit den Barcodes wurden einem Schärfungsprozess (s. Kapitel 6.1.3) unterzogen. Mit diesem Verarbeitungsschritt wurden jetzt Barcodes auf 40 Bildern erfolgreich erkannt, dies entspricht einer Erkennungsrate von 89%, während die Laufzeit um 0,05s (13%) stieg. Des weiteren zeigte sich, dass eine gewisse Mindestgröße der Barcodes im Bild nicht unterschritten werden darf, diese Größe liegt bei 150 x 50 Pixeln.

Eine weitere Verbesserung der Erkennungsrate ist vermutlich durch eine Steigerung der Abbildungsqualität der Kamera möglich. Dies kann von der eingesetzten Smartphone-Kamera jedoch nicht geleistet werden. Bewegungsunschärfe durch eine hohe Belichtungszeit sowie fehlendes optisches Auflösungsvermögen der eingesetzten Kamera beschränken hauptsächlich die Erfolgsrate der Barcode-Detektion.

Image-Retrieval Wie in Kapitel 6.1.3 bereits erwähnt, kann das Image-Retrieval durch Anpassung der Parameter noch weiter optimiert werden. Im Folgenden wird der Einfluss des Abstandsmaßes und der Anzahl der Merkmale genauer betrachtet. Um die Fehlerrate zu bestimmen wurde die Precision, der Recall und der F1-Score berechnet (vgl. Kapitel 3.3.1).

Für die Auswertung wurden Bilder von allen Artikeln aufgenommen. Dabei wurden insgesamt 237 Trainingsbilder der Artikeln aus verschiedenen Perspektiven erstellt. Des Weiteren wurden 171 Testbilder erstellt, auf denen die Artikel in der Hand gehalten wur-

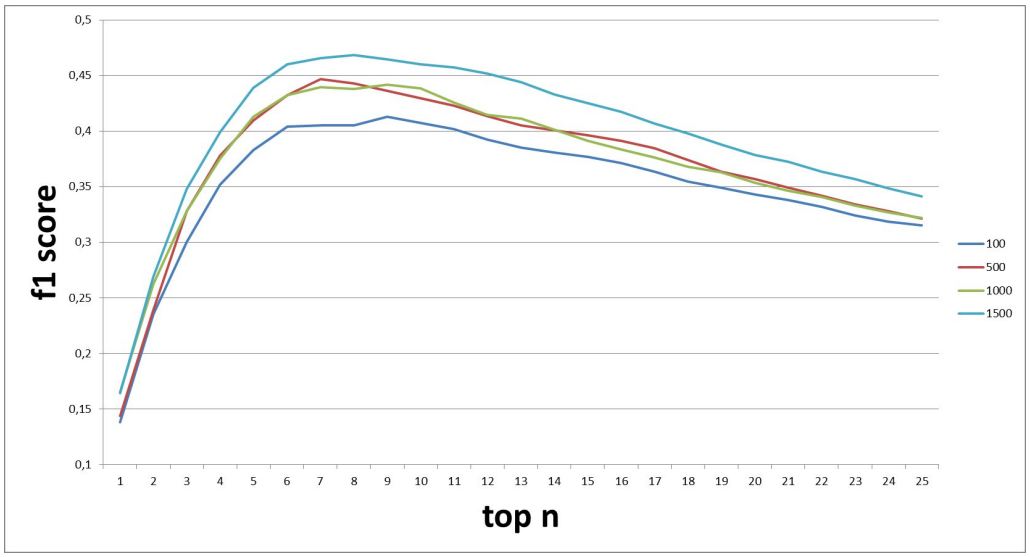


Abbildung 7.1: Einfluss der Anzahl der Merkmale auf die Fehlerrate

den[1]. Im Folgenden wird der Einfluss der Parameter auf die Fehlerrate beschrieben und ein optimaler Wert bestimmt. Dabei wurde so vorgegangen, dass für jedes Testbild eine Liste mit den ähnlichsten Bildern erstellt wurde (Top-N). Die Länge dieser Liste wurde variiert und für jede Anzahl Precision, Recall und F1-Score ermittelt. Anschließend wurden die drei Werte über alle Bilder gemittelt.

Die Anzahl der Merkmale, sprich die Dimensionalität der Merkmalsvektoren ist ein entscheidender Faktor. Je mehr Merkmale verwendet werden, desto feinere Informationen können repräsentiert werden. Bei zu vielen Merkmalen steigt die Fehlerrate allerdings wieder, da die Merkmale dann zu speziell sind und die Auswertung dauert aufgrund der höheren Dimension länger.

In Abbildung 7.1 wird der F1-Score in Abhängigkeit der N ähnlichsten Artikel dargestellt. Als Ähnlichkeitsmaß wurde die Cosinus-Distanz verwendet. Wie bereits zuvor vermutet, wird hier auch deutlich, dass die Anzahl der Merkmale die Fehlerrate beeinflusst. Den besten Wert erreicht das Retrieval, wenn 1500 Merkmale verwendet werden.

Ein weiteres Optimierungskriterium ist die Wahl des Abstandsmaßes. Hier wurden vier verschiedene Metriken getestet (vgl. Kapitel 6.1.3):

- Euklidische Norm
- Manhattan-Norm
- Maximums-Norm
- Cosinus-Ähnlichkeit

[1]Im Verlauf der Projektgruppe wurden zwei Datensätze erstellt, da der erste zu viele unscharfe Bilder enthielt. Der hier verwendete Datensatz stand bei der Auswertung der Barcode-Detektion noch nicht zur Verfügung, weshalb dort der erste verwendet wurde.

In Abbildung 7.2 wird die Leistungsfähigkeit dieser Metriken deutlich (verwendet wurden 1500 Merkmale). Hier zeigt sich, dass die Cosinus-Ähnlichkeit deutlich bessere Ergebnisse liefert als die drei anderen Metriken. Des Weiteren zeigt sich, dass der beste F1-Score bei $N = 6$ erreicht wird. Aus diesem Grund verwendet das Image-Retrieval 1500 Merkmale zur Beschreibung der Bilder und die Cosinus-Ähnlichkeit als Ähnlichkeitsmaß zweier Merkmalsvektoren. Der gegriffene Artikel ist genau dann korrekt, wenn unter den sechs ähnlichsten Bilder der richtige Artikel enthalten ist.

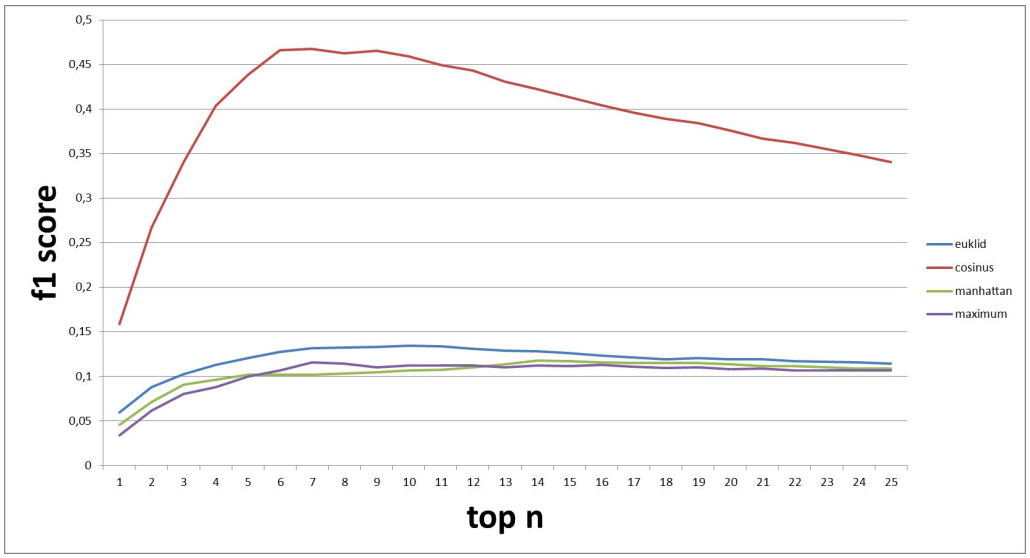


Abbildung 7.2: Einfluss der Ähnlichkeitsmaße auf die Fehlerrate

Um die Leistung des Image-Retrieval vergleichen zu können, wurde ein Test auf einem öffentlichen Datensatz durchgeführt. Dafür wurde der *SHORT-30* [31] Datensatz verwendet. Dieser enthält Bilder von Alltagsprodukten, wie z. B. Shampoo oder Getränke. Von jedem Artikel existieren 36 Trainingsbilder aus unterschiedlichen Perspektiven. Des Weiteren enthält der Datensatz Testbilder. Auf den Testbildern werden die Artikel vor verschiedenem Hintergrund in der Hand gehalten. Dieser Datensatz kommt dem Anwendungsfall im Rahmen der Projektgruppe sehr nahe.

In [30] werden verschiedene Image-Retrieval Verfahren auf dem Datensatz getestet. Dabei kommen State-of-the-art Verfahren zum Einsatz. In dem Test wurde nur die Precision für das jeweils ähnlichste Bild berechnet. In Tabelle 7.2 werden die erreichten Werte gegenübergestellt. Erwartungsgemäß dominieren die State-of-the-art Verfahren das der Projektgruppe. Allerdings ist der Unterschied in einigen Fällen nur gering. Dies zeigt, dass die Erkennung durch Encoding, wie z. B. *locality-constrained linear coding* (LLC) oder Fisher-Vektoren (FV) noch verbessert werden kann.

Das eingesetzte Image-Retrieval scheint demnach schon eine gute Leistung zu erzielen. Dieser Eindruck bestätigt sich auch im Live-Test. Eine Verbesserung ist dennoch möglich. Hier könnte zum einen getestet werden, ob noch mehr Merkmale bessere Ergebnisse erzie-

Verfahren	Precision
LLC-S8-4000	17,89
FV-S8-256	18,38
PG-Pick	15,00

Tabelle 7.2: Vergleich verschiedener Retrieval-Verfahren

len. Eine weitere Möglichkeit, die wahrscheinlich eine Verbesserung mit sich bringt wäre das Ausschneiden der Objekte auf dem Bild. Bei den Trainingsbildern hätte man dadurch den Vorteil, dass die berechneten Merkmale nur das Objekt beschreiben und keine Artefakte aus dem Hintergrund mit aufgenommen werden. Gleiches gilt für die Testbilder bzw. den Live-Betrieb. Insbesondere beim Einsatz im Lager könnte dies Vorteile bringen, da hier im Hintergrund häufig Regale zu sehen sind. Bei einer hohen Tiefenschärfe würden hier viele nicht relevante Keypoints gefunden werden. Um die Objekte auszuschneiden, wäre z. B. einen Detektion der Hand im Bild denkbar.

Klassifikation Die Evaluation der Objektklassifikation kann in drei Teile gegliedert werden: Auswahl der Merkmale, Parametrisierung des Clusterings und des Klassifikationsalgorithmus.

Zur Auswahl stehen **Merkmale** die durch das Verfahren SIFT oder durch ein Raster berechnet wurden. Als Größe für das Bag of Features wurde 1000 und 1500 gewählt, da die benötige Lautzeit zur Berechnung dieser im Rahmen blieb.

Für das **Clustering** sollten zwei Größen parametrisiert werden. Zum einen die Wahl der Initialisierungsmethode der Clusterzentren. Dafür steht eine zufällige Initialisierung und die Initialisierung durch die k-Means-Variante *k-Means++* [10] zur Verfügung. Weiterhin sollten mehrere Werte für die Anzahl der zu findenden Cluster k getestet werden. Ein größeres k, etwa im Verhältnis 1/1 zur Anzahl der Trainingsbilder, würde die Anzahl der Bilder pro Klasse niedrig halten und somit die Aussagekraft einer Klasse erhöhen. Allerdings ist von dem Klassifikationsalgorithmus nicht mit einer solchen Genauigkeit zu rechnen. Für ein niedriges k, etwa 3 Klassen, wird erwartet, dass sich die Genauigkeit erhöht, da die Klassenzentren weit auseinander liegen und so eine Differenzierung erleichtern. Dies geschieht allerdings auf Kosten der Aussagekraft einer Klasse, welche bei nur 3 Klassen sehr niedrig ausfallen sollte. Gesucht wird also ein k welches eine möglichst hohe Genauigkeit beim Klassifizieren bei möglichst hoher Klassenanzahl liefert. Als konkrete Werte für k wurden 3, 5, 10, 30, 100 und 230 gewählt, wobei 30 etwa der Anzahl der Artikel und 230s etwa der Anzahl der Trainingsbilder entspricht. Für den Klassifikationsalgorithmus, in unserem Fall eine Support Vector Machine, bietet OpenCV Varianten mit verschiedenen Kernels mit jeweils verschiedenen Parametern:

- Linear Kernel (keine Parameter)

- Polynomial Kernel
- Radial Basis Function Kernel
- Sigmoid Kernel

Die Ergebnisse des Clusterings fielen schlecht aus, da sich stark ähnelnde Bilder oft verschiedenen Clustern zugewiesen wurden. Auch wurden visuell unähnliche Bilder zum selben Cluster gezählt. Die Nutzung des Clusterings zur Berechnung der Klassenlabels ist unter diesen Umständen unzweckmäßig. Ein Einblick in die Ergebnisse gibt Abbildung 7.1.

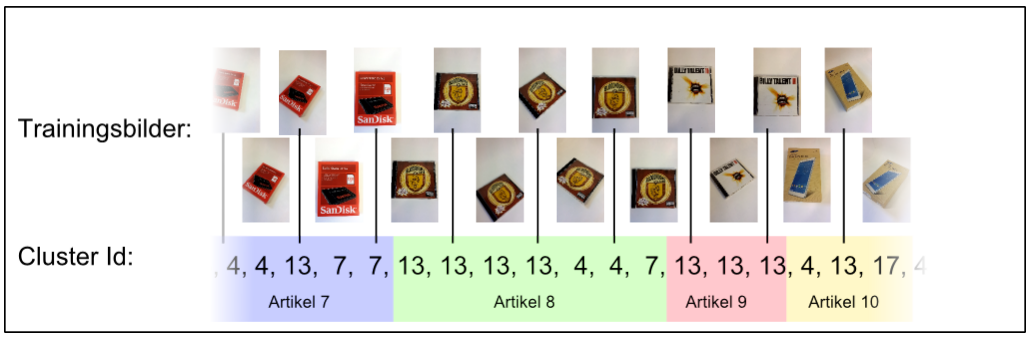


Abbildung 7.3: Trainingsbilder und ihre Clusterzuordnung, welche als Klassenlabels genutzt werden sollen. Visuell stark ähnliche Bilder werden teils mehreren Clustern zugeordnet. Für diesen Versuch wurden folgende Parameter gewählt: $k = 25$, Clusteringalgorithmus = KMeans, Clusterinitialisierung = KMeans++, Anzahl der Itertionen = 100, Versuche = 10. Die Anzahl der Artikel betrug 30, die der Trainingsbilder 232.

Bei der Ausführung des Klassifikators fiel auf, dass dessen Ergebnisse extrem schlecht ausfielen. In allen getesteten Fällen ordnete der Klassifikator jedem Testbild die selbe Klasse zu. Die verwendeten Merkmale, deren Menge und die Anzahl der Klassen hatten keinen Einfluss auf das Ergebnis. Das beste Ergebnis ordnet über 90 Prozent der Testbilder der selben Klasse zu. Anzahl der Klassen k bei diesem Test betrug 10.

Da mit der Klassifikation keine brauchbaren Ergebnisse erzielt wurden, entschied sich die Projektgruppe die Klassifikation aus dem Endprodukt herauszunehmen. Da das Image Retrieval brauchbare Ergebnisse erzielen konnte, können die Trainings- und Testdaten als Ursache vorerst ausgeschlossen werden. Eine mögliche Ursache hierfür könnte das verwendete Distanzmaß des Clusteringalgorithmus sein. OpenCV bietet für den Loyd-Algorithmus ausschließlich das euklidische Distanzmaß an. Dieses führte bereits beim Image Retrieval zu schlechten Ergebnissen. Eine mögliche Lösung wäre, die OpenCV-Implementierung des Clusteringalgorithmus gegen eine Implementierung des *Spherical KMeans* Algorithmus auszutauschen [38]. Dieser nutze die Cosinusdistanz zur Berechnung der Ähnlichkeit zweier Vektoren. Dazu wird jeder Vektor normiert. Der Abstand zwischen zwei Vektoren wird als Abstand auf dem Einheitskreis definiert.

TCP-Socket Zur Serialisierung von den zu versendenden Daten zwischen Server und Smartphone werden die Google Protocol Buffers eingesetzt, die ähnlich wie XML konzipiert aber viel kleiner, schneller und einfacher sind als diese. Dadurch, dass die Nachrichten im Vergleich zu XML als Binärformat konzipiert sind ist die Idee, dass diese auch schneller über einen TCP-Socket versendet werden können. Bei kurzen Nachrichten funktioniert dieser Gedanke gut. Werden jedoch Bilder vom Smartphone verschickt, bekommen selbst die Google Protocol Buffer Nachrichten eine beachtliche Größe zustande. Die versendeten Nachrichten werden von dem TCP-Socket zerstückelt und kommen am Server in mehreren Teilen an. Dies ist insofern ein Problem, da das Ende der Nachricht nicht eindeutig identifizierbar ist.

Um festzustellen, wann eine Nachricht zu Ende ist wird deshalb hinter jeder ein Delimiter angefügt, der dies signalisiert. Ein Algorithmus schreibt daher zunächst solange alle eingehenden Nachrichten in einen Buffer, bis erkannt wird, dass ein Delimiter darin enthalten ist. Dieser wird dann wieder aus der Nachricht herausgefiltert und das Google Protocol Buffer kann verarbeitet werden.

7.2 Smartphone

Das Samsung Galaxy S4 wurde als solide Basis zu einem günstigen Preis als Smartphone für die Projektgruppe ausgewählt. Bei der Entwicklung hat sich gezeigt, dass die vorhandenen Bibliotheken und Funktionen der Android-Plattform den Anforderungen entsprechen. Die gewünschten Funktionen konnten zum Großteil in der Android-App umgesetzt und aufeinander abgestimmt werden, ohne dass es zu größeren Schwierigkeiten kam. Das einzige große Problemfeld waren die Android-Schnittstellen zum Kamerasystem bzw. die Anpassung des Samsung Galaxy S4 durch Samsung selbst an diese Schnittstellen.

Kamera Die anfängliche Idee des Aufnehmens ganzer Bilder über den herkömmlichen Kameramechanismus wurde verworfen, als klar wurde, dass so keine Rohdaten erhalten werden können. Mit dem Abgreifen der Vorschaudaten der Kamera wurde aber ein mehr als ausreichender Ersatz gefunden, welcher durch die reduzierbare Auflösung noch weitere Vorteile bietet. So lässt sich die Kamera nun in der App an die gegeben Licht- und Netzwerkverhältnisse anpassen und die Netzwerklast gegebenenfalls reduzieren.

Die größten Schwierigkeiten traten im Zusammenhang mit dem Autofokus der Kamera des Smartphones auf. Durch die kleine Bauweise und die kleine Blende der verwendeten Bauteile verfügen diese über eine sehr große Tiefenschärfe. Dies hat zur Folge, dass der Fokuspunkt für scharfe Aufnahmen der Artikel möglichst genau platziert werden muss. Leider erlaubt jedoch das Galaxy S4 diese genaue Positionierung nicht. Die entsprechenden Schnittstellen werden zwar angeboten und auch vom System aufgerufen, allerdings werden immer Platzhalter anstatt richtiger Werte zurückgegeben, so dass eine automatische Optimierung des Fokusbereichs nicht möglich war. Als Alternative wurde der in

Kapitel 6.2.1 vorgestellte Fokusdialog eingeführt, der eine manuelle Steuerung des Autofokus ermöglicht. Insgesamt ist die Android-App als Grundgerüst für die Erkennung gut geeignet, sollte allerdings mit einer besseren, mehr auf den Einsatz abgestimmten Kamera versehen werden.

7.3 Pebble

Die Pebble wurde als Smartwatch ausgewählt, da zu Beginn der Projektgruppe nur wenige Alternativen vorhanden waren und diese am erfolgversprechendsten galt. In einigen Bereichen hat die Pebble unbestreitbare Vorteile, die auch während der Projektgruppe bestätigt wurden. Der größte Vorteil ist die Akkulaufzeit, die mit mehreren Tagen, durchaus bis zu sieben Tage, bei moderater Nutzung den Smartwatches der Konkurrenz um einiges voraus ist. Auch unter einer hohen Auslastung innerhalb eines Betriebes ist es wahrscheinlich, dass die Pebble mindestens einen Tag mit einer Akkuladung auskommt. Da sie über Nacht geladen werden kann ist demnach eine unterbrechungsfreie Kommissioniererführung durch die Pebble möglich.

Bei der Entwicklung der Pebble-App und der Anwendung der Pebble sind jedoch auch Probleme zum Vorschein gekommen, die zuvor nicht bekannt waren oder nicht ausreichend beachtet wurden.

Bildschirm Der Bildschirm der Pebble besteht aus einem E-Paper-Display, wodurch sich die Pebble von der Konkurrenz absetzt und eine lange Akkulaufzeit erst ermöglicht wird. Der Bildschirm löst lediglich mit 144 mal 168 Pixeln auf, wodurch die Auflösung sehr gering ist und Elemente nicht zu klein dargestellt werden dürfen. Dies ist insbesondere ein Problem, da das Display auch nur ca. 2 cm x 2,5 cm misst. Informationen dürften so nicht für jeden einfach abzulesen sein, vor allem, wenn bspw. Beeinträchtigungen des Sehvermögens vorliegen. Die Anzeige von Informationen wurde aus diesem Grund auch auf ein Minimum reduziert, um diese Informationen größtmöglich darstellen zu können. Neben der geringen Auflösung schränkt auch das monochrome Display die Entwicklung ein.

Kommunikation Die Pebble kommuniziert mittels Bluetooth mit dem Smartphone, mit welchem es verbunden ist. Nachrichten, die von der Pebble zum Smartphone gesendet werden dürfen dabei wesentlich größer sein als Nachrichten, die von dem Smartphone zur Pebble gesendet werden. Eingehende Nachrichten dürfen eine Größe von 126 Byte besitzen, während ausgehende Nachrichten eine Größe von 656 Byte haben dürfen. Da in dieser Projektgruppe jedoch die Kommunikationsrichtung Smartphone → Pebble im Vordergrund stand gestaltete es sich schwer eine stabile Kommunikation gewährleisten zu können. In einem empfangenen Paket von maximal 126 Byte können bspw. nicht nur Nutzdaten gepackt werden. Diese müssen in einer Dictionary-Struktur untergebracht werden, welche ihrerseits auch Speicher benötigt. Außerdem wurden zur Zuordnung Schlüsselwerte benö-

tigt, welche auch Speicher in dieser Struktur einnahmen. Da für vollständige Picks, die an die Smartwatch gesendet werden, diese Größe nicht ausreicht, müssen diese in Teilen gesendet werden. Hier kommt jedoch zu tragen, dass die Architektur der Datenübertragung an die Pebble nicht für ein hohes Aufkommen an Nachrichten entworfen wurde. In der Realität kommt es daher je nach Menge an Kommunikation zu Paketverlusten. Am gravierendsten sind fallen gelassene Pakete auf der Pebble, über welche PebbleKit in der Android-App nicht benachrichtigt wird. Dieser Mechanismus musste daher nachgerüstet werden, was jedoch wiederum zu vermehrter Kommunikation führte.

Als weitere und im realen Einsatz evtl. größte Nachteile bei der Kommunikation ergeben sich aus den Latenzen, die beim Senden und Empfangen auftreten. Da die Kommunikation auf Energiesparen ausgerichtet ist werden Nachrichten nicht immer sofort ausgeliefert. Wird ein Pick an die Smartwatch gesendet und zuvor hat länger keine Kommunikation stattgefunden, so kann dieser Vorgang unter Umständen länger als eine Sekunde dauern. Eine Verzögerung, in der der Kommissionierer keine Möglichkeit hat Informationen zu seinem nächsten Pick zu erhalten. Die Beeinflussung durch diesen Umstand müsste jedoch in der Realität erprobt werden, da lange Kommunikationspausen in einem Lager eher selten vorkommen dürften.

7.4 Bewegungserkennung

Bei der Implementierung der Bewegungserkennung wurden verschiedene statistische Berechnungsverfahren zur Merkmalsextraktion aus Rohdaten der Sensoren verwendet. Diese Verfahren beinhalten u. a. die Methoden der Mittelwertberechnung, des Vektorbetrages und der Standardabweichung über alle Achsen des Beschleunigungssensors. Bei anfänglichen Versuchen allein mit dem Vektorbetrag und der Mittelwerte eine genaue Klassifizierung der Bewegung zu erreichen wurde keine zufriedenstellende Lösung erreicht. Es stellte sich dann heraus, dass die Standardabweichung ein sehr solides Verfahren zur Differenzierung der von uns benötigten Bewegungsformen darstellte. Der Schwellwert, der sich aus der Berechnung der Standardabweichung auf der y-Achse ergibt, wird in der finalen Version der Applikation im laufenden Betrieb dazu verwendet die Bewegungsformen Stehen und Gehen zu erkennen. Bei der Klassifizierung bzw. der Auswertung der gesammelten Daten mittels RapidMiner konnte, wie bereits in Kapitel 6.2.3 erläutert, eine sehr genaue Differenzierung der Bewegung erreicht werden (siehe Tabelle 7.3). Die genannte Konfusionsmatrix beruht

	true Gehen	**true Stehen**	**class precision**
pred. Gehen	828	4	99.52%
pred. Stehen	1	558	99.82%

Tabelle 7.3: Konfusionsmatrix zu einem Testdatensatz zur Bestimmung der Performanz des gewählten Schwellwertes

auf einem Datensatz mit zwei Sequenzen, dem Gehen und Stehen. Sie wurde nach einer Aufzeichnungsdauer von ungefähr 20-25 Minuten in Summe für die beiden Sequenzen ermittelt. Die Berechnung der Matrix erfolgte mit demselben Prozess in Rapidminer Studio, der auch zuvor für die Bestimmung des Schwellwertes genutzt wurde. Die geringe Fehlerrate bei der Vorhersage der Bewegung kann darauf zurückgeführt werden, dass insbesondere manche Greifvorgänge beim Stehen zu ruckartigen Bewegungen des Smartphones führen und somit in einer falschen Klassifizierung der Aktivität resultieren, wie auch in der Tabelle 7.3 zu sehen ist. Das Gehen hingegen wird ohne nennenswerte Schwierigkeiten stets korrekt klassifiziert. Eine weitere Fragestellung ergab sich bei der Implementierung der Bewegungserkennung daraus, ob zur Aufzeichnung von Sensordaten zur weiteren Verwendung in einer Data-Mining Software eine externe Applikation zur Verwendung kommen sollte. Um jedoch eine in sich abgeschlossene Applikation zur Verfügung zu stellen und mögliche Fehlerquellen zu vermeiden wurde eine eigene Lösung in die Applikation eingebaut, die ein *Preprocesssing* ermöglicht und sich über die Einstellungen der App aktivieren bzw. deaktivieren lässt.

KAPITEL 8

Verifikation

Dieses Kapitel gibt einen Überblick über das Testverfahren der in dem Pflichtenheft spezifizierten Testfälle. Für diesen Zweck existiert ein Testsystem mit einem choreografierten Testlauf, der die Testfälle des Pflichtenheftes umfasst. Die Tests sollen primär prüfen, ob das System die Anforderungen erfüllt und Aufschluss über die Leistungsfähigkeit des Systems geben.

8.1 Testsystem

Als Testsystem stand, wie in Kap. 7.1, 7.2 und 7.3 genauer beschrieben, ein Server, ein Smartphone und eine Smartwatch zur Verfügung. Für die Kommunikation zwischen dem Server und dem Smartphone war zusätzlich ein Wlan-Router erforderlich, sodass hierfür ein Apple AirPort Extreme 802.11n zum Einsatz kam. Auf dem Server wurde eine Tabelle mit 67 Produkten erstellt. Sie wurden so zusammengestellt, dass sie zu 9 unterschiedlichen Produktkategorien (Elektronik, Computer-Peripherie, PC Komponenten, Tablet, CD/DVD/PC-Spiel, Spiel, Buch, Haushalt und Lebensmittel) zugeordnet werden konnten. Für jedes dieser Artikel gab es mehrere Trainings- sowie Testbilder. Zu den Artikeln, an denen noch Barcodes erkennbar waren, wurde dieser auch in die Datenbank übernommen. Um die Erkennung in einem Lager abbilden zu können, wurde ein Regal eingerichtet, auf dem die zu erkennenden Produkte angeordnet werden konnten.

8.2 Testfälle

Im Folgenden werden die Tests beschrieben, die die korrekte Implementierung der in Kapitel 4.3 beschriebenen Anwendungsfälle überprüfen sollen. Für die Tests T001 bis T011 gelten die folgenden Vorbedingungen.

- WLAN am Smartphone ist eingeschaltet und Smartphone ist mit einem Zugangspunkt verbunden
- Smartphone App ist gestartet
- Server ist gestartet
- Server ist mit dem Netzwerk verbunden

Sofern weitere Vorbedingungen nötig sind, werden diese bei dem entsprechenden Test explizit angegeben.

/T001/ Benutzer anmelden

Dieser Test dient der Überprüfung, ob sich ein Kommissionierer am Server anmelden kann, um mit seiner Arbeit zu beginnen. Der Kommissionierer kann sich mit einer Kennung anmelden und wird so im System zugeordnet. Bei erfolgreicher Anmeldung wird eine Pickliste geladen und der erste Pick kann erkannt werden. Bei fehlgeschlagener Anmeldung wird der Benutzer darüber informiert.

Vorbedingung:

- Der Kommissionierer hat eine gültige Benutzerkennung, die dem Server bekannt ist

Durchzuführende Schritte:

- Der Benutzer drückt auf den Login Button
- Durch die in den Einstellungen der App hinterlegten Daten (Nutzername, Passwort und Serveradresse) wird der Benutzer beim Server angemeldet

Erwartete Ausgabe:

- Der Benutzer ist im System eingeloggt und in der App sind nun alle Funktionen nutzbar
- Ein Pick wird auf das Smartphone geladen
- Bei Kopplung mit Smartwatch: Nächster Pick wird an die Smartwatch gesendet

Erwartete Nachbedingungen:

- Der Benutzer ist im System eingeloggt
- Es besteht eine dauerhafte Verbindung mit dem Server

Prüfungsanweisungen:

- Prüfe den Verbindungsstatus
- Teste mithilfe einer bekannten Pickliste, ob alle Daten vollständig gesendet wurden

/T002/ Benutzer abmelden

Dieser Test dient zur Überprüfung, ob sich der Kommissionierer vom Server abmelden kann. Nach der Abmeldung soll die Verbindung zum Server getrennt werden.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server

Durchzuführende Schritte:

- Der Benutzer drückt auf den Logout Button

Erwartete Ausgabe:

- Der Benutzer ist aus dem System ausgeloggt und in der App sind nicht mehr alle Funktionen nutzbar

Erwartete Nachbedingungen:

- Der Benutzer ist aus dem System ausgeloggt
- Es besteht keine Verbindung mehr mit dem Server

Prüfungsanweisungen:

- Teste, ob Verbindung getrennt wurde
- Prüfe, ob die Pickliste nicht mehr angezeigt wird

/T003/ Erkennung pausieren

Dieser Test dient zur Überprüfung, ob der Kommissionierer die Erkennung, zum Beispiel während der Mittagspause, pausieren kann. Wurde die Erkennung angehalten, darf das Smartphone keinen Videostream mehr an den Server senden und demzufolge wird auch keine Objekterkennung durch den Server ausgeführt. Die Verbindung zum Server bleibt allerdings bestehen und der Kommissionierer angemeldet.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server

Durchzuführende Schritte:

- Der Benutzer drückt auf den Pause Button
- Der Benutzer äußert in einem Bestätigungsdialog den Wunsch, die Erkennung zu pausieren

Erwartete Ausgabe:

- Das Unterbrechen der Erkennung wird von der App angezeigt

Erwartete Nachbedingungen:

- Es wird kein Stream mehr an den Server gesendet
- Der Benutzer bleibt angemeldet

Prüfungsanweisungen:

- Teste, ob beim Stehenbleiben und Greifen eines Artikels Bilder an den Server gesendet werden
- Teste, ob der Benutzer sich abmelden kann

/T004/ Erkennung fortsetzen

Dieser Test dient zur Überprüfung, ob der Kommissionierer die Erkennung fortsetzen kann, nachdem sie zuvor pausiert wurde.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat die Erkennung zuvor pausiert

Durchzuführende Schritte:

- Der Benutzer drückt auf den Fortsetzen Button
- Der Benutzer äußert in einem Bestätigungsdialog den Wunsch, die Erkennung fortzusetzen

Erwartete Ausgabe:

- Das Fortsetzen der Erkennung wird von der App angezeigt

Erwartete Nachbedingungen:

- Es wird bei Bedarf wieder ein Bild an den Server gesendet

Prüfungsanweisungen:

- Teste, ob beim Stehenbleiben und Greifen eines Artikels ein Bild an den Server gesendet wird

/T005/ Durch Pickliste navigieren

Dieser Test dient zur Überprüfung, ob der Kommissionierer sich auf der Smartwatch bereits abgearbeitete Picks ansehen kann. Das Navigieren durch die Liste erfolgt durch Betätigen der Tasten der Smartwatch. Die vorherigen Picks werden dann auf der Smartwatch angezeigt.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat bereits mindestens einen Pick abgearbeitet und bekommt den nächsten Pick angezeigt

Durchzuführende Schritte:

- Der Benutzer drückt an der Smartwatch den Button, um in der Liste vor/zurück zu navigieren

Erwartete Ausgabe:

- Es wird auf der Smartwatch der vorherige/nächste Pick angezeigt

Prüfungsanweisungen:

- Teste, ob vorherige/nächste Picks angezeigt werden

/T006/ Stehen bleiben

Dieser Test dient der Überprüfung, ob das Smartphone ein Stehenbleiben des Kommissionierers erkennt. Bleibt der Kommissionierer stehen, werden Einzelbilder an den Server gesendet und die Objekterkennung startet.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat den Auftrag einen Artikel zu holen

Durchzuführende Schritte:

- Der Benutzer bleibt vor dem Regal stehen

Erwartete Ausgabe:

- Die App sendet ein Bild an den Server

Prüfungsanweisungen:

- Teste, ob beim Stehenbleiben ein Bild an den Server gesendet wird

/T007/ Weitergehen

Dieser Test dient zur Überprüfung, ob das Smartphone erkennt, dass der Kommissionierer nach dem Pick weitergeht. Beim Weitergehen sollen keine Bilder zum Server gesendet werden, da der Kommissionierer während er geht keinen Pick durchführen kann und somit keine Objekterkennung durchgeführt werden muss.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Ein Pick wurde erfolgreich erkannt

Durchzuführende Schritte:

- Der Benutzer läuft los

Erwartete Ausgabe:

- Es werden keine Bilder an den Server geschickt

Prüfungsanweisungen:

- Teste, ob keine Bilder versendet werden

/T008/ Artikel greifen

Dieser Test dient zur Überprüfung, ob der Greifvorgang des Kommissionierers erkannt wird. Erst nach dem Erkennen des Greifvorgangs versucht das System den Artikel zu erkennen.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer ist stehen geblieben und dies wurde vom System erkannt
- Es wird ein Videostream an den Server gesendet

Durchzuführende Schritte:

- Der Benutzer greift nach einem Artikel und nimmt diesen aus dem Regal

Erwartete Nachbedingungen:

- Das System erkennt den Vorgang und beginnt die Objekterkennung

Prüfungsanweisungen:

- Teste, ob das System versucht den gegriffenen Artikel zu erkennen

/T008a/ Artikel greifen/Signal bei erfolgreicher Erkennung

Dieser Test dient zur Überprüfung, ob ein korrekter Pick vom System erkannt und dies dem Kommissionierer signalisiert wird. Dabei soll festgestellt werden, ob der Server einen korrekten Pick auch als solchen erkennt und ob die Smartwatch ein entsprechendes Feedback gibt.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag

Durchzuführende Schritte:

- Der Benutzer nimmt den korrekten Artikel aus dem Regal

Erwartete Ausgabe:

- Dem Benutzer wird ein positives Feedback gegeben

- Der Artikel wird in der Pickliste als korrekt gepickt markiert

Erwartete Nachbedingungen:

- Der nächste Pick wird angezeigt

Prüfungsanweisungen:

- Prüfe, ob die Pickliste aktualisiert wird
- Prüfe, ob ein positives Feedback erfolgt

/T008b/ Artikel greifen/Signal bei Erkennung eines falschen Artikels

Dieser Test dient zur Überprüfung, ob ein fehlerhafter Pick vom System erkannt und dies dem Kommissionierer signalisiert wird.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag

Durchzuführende Schritte:

- Der Benutzer nimmt einen falschen Artikel aus dem Regal und hält dessen Barcode vor die Kamera

Erwartete Ausgabe:

- Dem Benutzer wird ein negatives Feedback gegeben
- Der Auftrag wird weiterhin in der Pickliste angezeigt

Prüfungsanweisungen:

- Prüfe, ob der Auftrag weiterhin angezeigt wird
- Prüfe, ob ein negatives Feedback erfolgt

/T008c/ Artikel greifen/Signal bei fehlgeschlagener Erkennung

Dieser Test dient zur Überprüfung, ob dem Kommissionierer eine fehlgeschlagene Erkennung angezeigt wird, das heißt, der Server kann den Pick weder bestätigen noch zurückweisen. In diesem Fall ist ein manuelles Eingeben des Barcodes oder eine Bestätigung durch den Benutzer notwendig.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag

Durchzuführende Schritte:

- Der Benutzer nimmt einen Artikel, der nicht im System registriert ist

Erwartete Ausgabe:

- Der Benutzer wird aufgefordert, den Artikel manuell zu bestätigen

Prüfungsanweisungen:

- Prüfe, ob der Benutzer zur manuellen Bestätigung aufgefordert wird

/T009/ Artikel als korrekt gepickt markieren

Dieser Test dient zur Überprüfung, ob der Kommissionierer einen als falsch klassifizierten Artikel manuell bestätigen kann.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag
- Der Benutzer hat einen Artikel gegriffen und dieser wurde als falsch erkannt

Durchzuführende Schritte:

- Um sicher zu gehen, dass der Server den Pick als falsch erkennt, greift der Benutzer einen falschen Artikel
- Der Benutzer bestätigt durch manuelle Eingabe den Pick

Erwartete Ausgabe:

- Der Pick wird in der Liste als korrekt markiert

Erwartete Nachbedingungen:

- Der nächste Pick wird angezeigt

Prüfungsanweisungen:

- Überprüfe den Eintrag in der Pickliste auf der Smartwatch
- Überprüfe, ob ein neuer Pick angezeigt wird

/T010/ Artikel wiederholen

Dieser Test dient zur Überprüfung, ob der Kommissionierer einen als korrekt markierten Artikel manuell zurücksetzen und den Pick später wiederholen kann.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat zuvor einen Pick durchgeführt, der vom Server als korrekt erkannt wurde

Durchzuführende Schritte:

- Der Benutzer navigiert in der Pickliste zu einem als korrekt markierten Artikel und setzt diesen über das Menü der Smartwatch zurück

Erwartete Ausgabe:

- Der Pick wird in der Liste nicht mehr als korrekt angezeigt

Erwartete Nachbedingungen:

- Der Pick wird dem Kommissionierer erneut angezeigt

Prüfungsanweisungen:

- Prüfe, ob der Eintrag in der Pickliste aktualisiert wird
- Prüfe, ob der Kommissionierer den Pick erneut durchführen muss

/T011/ Artikel als nicht vorhanden markieren

Dieser Test dient zur Überprüfung, ob der Kommissionierer einen Artikel manuell als nicht vorhanden markieren kann, zum Beispiel wenn er diesen nicht findet.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag

Durchzuführende Schritte:

- Der Benutzer markiert den Artikel über das Menü der Smartwatch als nicht vorhanden

Erwartete Ausgabe:

- Der Artikel wird in der Pickliste als nicht vorhanden angezeigt

Erwartete Nachbedingungen:

- Der nächste Pick wird angezeigt

Prüfungsanweisungen:

- Prüfe, ob die Pickliste aktualisiert wird
- Prüfe, ob der nächste Pick angezeigt wird

8.3 Weitere Tests

Im Folgenden werden die Tests beschrieben, die die korrekte Implementierung der in Kapitel 4.4 beschriebenen Produktfunktionen überprüfen sollen. Für die Tests T101 bis T103 gelten die folgenden Vorbedingungen.

- WLAN am Smartphone ist eingeschaltet und Smartphone ist mit einem Zugangspunkt verbunden
- Smartphone App ist gestartet
- Server ist gestartet
- Server ist mit dem Netzwerk verbunden

Sofern weitere Vorbedingungen nötig sind, werden diese bei dem entsprechenden Test explizit angegeben.

/T101/ Signal bei Verbindungsabbruch

Mit diesem Test soll überprüft werden, ob der Kommissionierer über einen Verbindungsabbruch zum Server oder zur Smartwatch informiert wird.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Das Smartphone ist mit einer Smartwatch verbunden

Durchzuführende Schritte:

- Der Benutzer trennt die WLAN-Verbindung
- Der Benutzer trennt die Bluetooth-Verbindung

Erwartete Ausgabe:

- Nach dem Trennen der einzelnen Verbindungen wird dem Benutzer ein Feedback gegeben
- In der App wird die Verbindung zum Server bzw. zur Smartwatch als "off" angezeigt

Erwartete Nachbedingungen:

- Es besteht keine Verbindung zum Server und zur Smartwatch

Prüfungsanweisungen:

- Prüfe, ob das Trennen der Verbindung dem Benutzer singalisiert wird

/T102/ Verbindung wieder aufnehemen

Dieser Test dient zur Überprüfung, ob eine zuvor unterbrochene Verbindung wieder aufgenommen werden kann.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Die Verbindung zum Server und/oder zur Smartwatch wurde zuvor unterbrochen

Durchzuführende Schritte:

- Der Benutzer schaltet die WLAN-Verbindung wieder ein
- Der Benutzer schaltet die Bluetooth-Verbindung wieder ein

Erwartete Ausgabe:

- In der App wird die Verbindung zum Server bzw. zur Smartwatch als “on” angezeigt

Erwartete Nachbedingungen:

- Es besteht eine Verbindung zum Server und zur Smartwatch

Prüfungsanweisungen:

- Prüfe, ob die Verbindung zum Server und zur Smartwatch wiederhergestellt wurde
- Prüfe, ob die Verbindungen in der App korrekt angezeigt werden

/T103/ Anzeigen der Artikeldetails

Dieser Test soll überprüfen, ob dem Kommissionierer detaillierte Informationen über den nächsten Pick angezeigt werden. Er bezieht sich auf die Produktfunktionen F103 und F104. Die Details zu einem Artikel sind zum Beispiel Artikelbezeichnung und der Lagerplatz.

Vorbedingung:

- Der Benutzer ist im System eingeloggt und es besteht eine aktive Verbindung zwischen App und Server
- Der Benutzer hat einen aktuellen Pick-Auftrag

Prüfungsanweisungen:

- Prüfe, ob auf der Smartwatch Details zum Pick, wie zum Beispiel Artikelbezeichnung und Lagerplatz angezeigt werden

8.4 Testauswertung

Anhand des Testsystems soll die Korrektheit und Vollständigkeit der Implementierung belegt werden. In der Tabelle 8.1 ist dazu die Choreographie, durch die die Testfälle veranschaulicht werden sollen, aufgelistet. Die volle Funktionsfähigkeit der erstellten Softwarekomponenten impliziert ihre Abnahmefähigkeit und wird anhand von Testfällen (siehe Abschnitt 8.1) definiert, welche die zugesicherten Anwendungsfälle aus Abschnitt 4.3 abdecken. Im Folgenden wird daher gezeigt, dass jeder Testfall durch die beschriebene Choreographie repräsentiert wird.

So wird durch Schritt 0 vorausgesetzt, dass die drei entwickelten Systemkomponenten gestartet und einsatzfähig sind. Der Test T001 korrespondiert zu Schritt 1 - der Anmeldung des Benutzer beim Server. Anschließend wird die abzuarbeitende Pickliste für den Kommissionierer bereitgestellt. Nachdem der erste Artikel erhalten wurde, macht sich der

	Anweisung
0.	Vorbereitung (Starten des Servers, der Android-App sowie Pebble-App)
1.	Benutzer meldet sich beim Server an
2.	Pickliste wird vom Server bereitgestellt
3.	Kommissionierer sucht ersten Artikel
4.	Kommissionierer greift nach dem gefundenen Artikel
5.	Artikel wurde erfolgreich erkannt
6.	Nächster Artikel der Pickliste wird bereitgestellt
7.	Kommissionierer sucht Artikel
8.	Kommissionierer greift nach dem gefundenen Artikel
9.	Artikel wurde erfolgreich erkannt
10.	Nächster Artikel der Pickliste wird bereitgestellt
11.	Kommissionierer sucht Artikel
12.	Kommissionierer greift falschen Artikel
13.	Erkennung eines falschen Artikels
14.	Artikel wird aus der Pickliste entfernt
15.	Pickliste ist abgearbeitet
16.	Historie der Pickliste wird sich angeschaut
17.	Benutzer meldet sich beim Server ab

Tabelle 8.1: Die einzelnen Choreographieschritte

Kommissionierer auf den Weg dorthin. Die Test T006 und T007 sollen durch diesen Vorgang abgedeckt werden. Gelangt der Kommissionierer an sein Ziel, bleibt er stehen, die Erkennung startet und er greift nach dem Artikel. “Artikel greifen” ist für unser Testszenario irrelevant geworden, da der Overhead zur Erkennung des Greifvorgangs zu viel Zeit beansprucht. Daher startet die Erkennung der Artikel schon beim Stehenbleiben. In Schritt 5 wird der Artikel erfolgreich erkannt. Dies entspricht dem Testfall T008a. Danach wird gemäß T103 der nächste Artikel der Pickliste abgerufen und der Kommissionierer begibt sich zu dem angegebenen Ort, der in den Artikeldetails zu finden ist. Der gegriffene Artikel wird ebenfalls erfolgreich erkannt. Anschließend wird erneut ein Artikel aus der Pickliste bezogen. Nachdem der Kommissionierer beim Regal des erhaltenen Artikels angekommen ist startet nach dem Stehenbleiben die Erkennung. Allerdings greift er diesmal den falschen Artikel und der Artikel wird gemäß T008b als falsch klassifiziert. Der Kommissionierer hat nach T010 die Möglichkeit den Artikel wiederholt zu scannen (falls er sich sicher ist, dass er den richtigen Artikel gegriffen hat) oder ihn aus der Pickliste zu entfernen. Durch T009 und T011 wird dieses Szenario abgedeckt. Nachdem einer der beiden Fälle eingetreten ist, hat der Kommissionierer die Pickliste abgearbeitet. Bei Bedarf kann er sich durch beendete Picks - entsprechend T005 - navigieren. Der Test T002 korrespondiert zu Schritt 18 und der Benutzer meldet sich wieder beim Server ab.

Im Rahmen der Abschlusspräsentation der Projektgruppe wurde die festgelegte Choreographie komplett durchgespielt. Der fehlerfreie Durchlauf dieser hat die Korrektheit und Vollständigkeit der Implementierung des Testsystems belegt.

KAPITEL 9

Zusammenfassung und Ausblick

Kommissioniererführung ist heute geprägt von Systemen, die verschiedene Vor- und Nachteile besitzen. Beispielhaft kann die Kommissionierung mithilfe von Picklisten in Papierform kostengünstig auf einfache Weise geleistet werden. Nachteilig sind in diesem Fall die nachträgliche Eingabe in digitalen Systemen und somit der zusätzliche Zeitaufwand. Neuere Systeme, wie Pick-by-Voice oder Pick-by-Light, erleichtern die Kommissionierung, da der Mitarbeiter nicht mehr gezwungen ist, die Pickliste ständig bei sich zu tragen. Nachteile der neueren Systeme sind das dauerhafte Tragen der Kopfhörer und Entgegennehmen von Anweisungen durch eine Computerstimme sowie hohe Investitionskosten.

Mit der Entwicklung eines gänzlich neuartiges Systemes **Camera-assisted Pick-by-feel** wurde während dieser Projektgruppe versucht, eine zukunftsträchtige Kommissioniererführung zu entwickeln, welche viele Vorteile vereint. So kann das entwickelte System durch kostengünstige Komponenten für einen Server, ein Smartphone sowie eine Smartwatch zusammengestellt werden. Der Kommissionierer erhält die Smartwatch, auf der der Lagerort des nächsten Picks angezeigt wird. Zusätzlich wird mithilfe eines Brustgurtes das Smartphone auf seiner Brust befestigt, sodass der Kommissionierer beide Hände für die Kommissionierung verwenden kann. Für den Kommissioniervorgang wird der gesuchte Pick in die Kamera des Smartphones gehalten. Die fotografierten Bilder werden an den Server weitergeleitet und ausgewertet, sodass bestimmt werden kann, ob das vor die Kamera gehaltene Objekt der gesuchte Pick ist. Hierfür kommen auf dem Server eine Barcode Erkennung sowie Image Retrieval zum Einsatz. Damit nicht dauerhaft Bilder erkannt werden müssen, kann optional eine Lauferkennung auf dem Smartphone aktiviert werden. Durch diese Bewegungserkennung wird die Auswertung erst gestartet, wenn der Kommissionierer stehen bleibt. Mithilfe der entwickelten Serversoftware können zusätzliche Picks angelegt

und Picklisten erstellt werden. Die implementierte Mehrbenutzerfähigkeit bietet zudem die Möglichkeit, dass sich mehrere Kommissionierer mit ihren Smartphones beim Server anmelden und jeweils ihre eigenen Picklisten abarbeiten können.

Ausblickend bleibt zu untersuchen, ob durch den Einsatz von Industrie- statt Consumer-Hardware die Ergebnisse weiter verbessert werden können. So kann durch Verwendung einer angepassten Kamera die vergleichsweise schlechten Eigenschaften einer Smartphonekamera bzgl. Fokussierung, Lichtempfindlichkeit und Tiefenschärfe verringert werden. Die Lesbarkeit der Smartwatch könnte mithilfe eines Geräts mit größerem Display untersucht werden. Berücksichtigt werden muss jedoch, dass der Einsatz von Industriehardware wiederum die Kosten steigen lassen. Die Zeit für die Erkennung von Picks könnte durch Verwendung verschiedener Möglichkeiten optimiert werden. Da die Übertragung der Bilder vom Smartphone zum Server der größter Zeitfaktor ist, sind Lösungen wie Komprimierung der Bilder als auch die direkte Auswertung der Bilder auf dem Smartphone denkbar.

Zusammenfassend bleibt festzuhalten, dass das Endprodukt den gestellten Anforderungen gerecht wird. Verbesserungspotenzial besteht darin, dass Bilder der zu erkennenden Picks im Voraus benötigt werden, damit der Server trainiert werden kann. Zusätzlich müsste der Aspekt des Datenschutzes betrachtet werden, da durch Veränderung der Implementierung es möglich wäre, Bilder durchgehend zum Server zur Auswertung zu schicken. Die Vorteile des entwickelten Produkts sind eine einfache Verwendung und die Tatsache, dass sehr geringe Einschränkungen bezüglich der Arbeit des Kommissionierers geboten wird. Auch die anfallenden Kosten für den Aufbau des Systems ist als gering einzuschätzen und vor allem im Vergleich zu bestehenden Kommissionierleitsystemen ein großer Pluspunkt.

ANHANG A

Glossar

Im Folgenden werden die wichtigsten im Endbericht genutzten Fachbegriffe erläutert.

Barcode Auch Strichcode, Balkencode, Streifencode genannt. Maschinenlesbare, optische Schrift zur Speicherung und Darstellung von Daten, meist numerischen IDs.

Bluetooth Funktechnologie zur Kommunikation zwischen elektronischen Geräten über kurze Distanzen.

Chaotische Lagerhaltung Ein Lagerhaltungsprinzip, in dem die größtmögliche räumliche Verteilung von Waren, die sich entweder in Form oder Zugehörigkeit ähneln, die höchste Priorität besitzt.

Client Hard- oder Softwareeinheit in einem System, die Informationen von einem Server erhält oder an diesen sendet.

Filter Algorithmus zur Veränderung bzw. Verbesserung von Daten, hier meist ein Filter zur Veränderung von Bilddaten. Beispiele für Filter sind Schärfen oder Aufhellen von Bildern.

Kommissionierung Zusammenstellung der zu einem Auftrag gehörenden Güter im Lager.

Objekterkennung Automatische Identifizierung eines Objektes, hier im Speziellen aus Bild- oder Videodaten.

Objektklassifizierung Einordnung eines Objektes in eine von mehreren vorgegebenen Kategorien. Im Gegensatz zur Objekterkennung wird nicht versucht ein konkretes Objekt im Vergleichsdatensatz zu finden, sondern es erfolgt nur die Einordnung in eine der durch maschinelles Lernen bestimmten Kategorien.

OpenCV Open Computer Vision, eine Programmbibliothek mit Funktionen zum maschinellen Sehen und für die Verarbeitung von Bildern. Ebenfalls sind in der Bibliothek Funktionen zum maschinellen Lernen vorhanden.

Pick Die Tätigkeit, in einem Lagerbetrieb Ware von einer Lagereinheit zu entnehmen.

Pick-by-Light Ist ein belegloses Kommissionierverfahren. Durch Anzeigen an den jeweiligen Fächern, werden die zu kommissionierenden Waren und die benötigte Anzahl dem Lagerarbeiter mitgeteilt.

Pick-by-vision Ist ein belegloses Kommissionierverfahren. Mithilfe einer Datenbrille werden dem Kommissionierer fortwährend Informationen zu dem gesuchten Pick angezeigt.

Pick-by-voice Ist ein belegloses Kommissionierverfahren. Informationen zu dem Lagerort und den zu kommissionierenden Waren werden in diesem Fall über Sprache übermittelt.

Pickliste Eine Liste, die alle zu kommissionierenden Waren und ihre Lagerorte enthält.

Put-to-light Wie Pick-by-Light, jedoch mit umgekehrter Arbeitsweise. In diesem Fall werden die Waren in Fächer eingeordnet statt entnommen.

SDK Software Development Kit, Bezeichnung für eine Sammlung von Werkzeugen und Bibliotheken zur Entwicklung von Software.

Server Die Hardware, auf der zentrale Komponenten eines Softwaresystems laufen, wird als Server bezeichnet. Ebenso wird die Software, die den Clients Dienste bereitstellt, als Server bezeichnet.

Smartphone Die Weiterentwicklung des Handys, worin durch das Betriebssystem die Möglichkeit bereitgestellt wird, Systemfunktionen durch benutzerspezifische Funktionen zu erweitern.

Smartwatch Elektronisches Gerät, welches am Armband wie eine Armbanduhr getragen werden kann. Es besitzt die Fähigkeit der Kommunikation mit einem Smartphone, sowie die Möglichkeit Informationen anzuzeigen und Benutzereingaben entgegenzunehmen.

(Video-)Stream Datenstrom zur kontinuierlichen Übertragung von Daten, hier von Videobildern.

Visual Computing Akquisition, Analyse und Synthese von Bildmaterial.

WLAN Wireless Local Area Network, eine Funktechnologie zum Aufbau eines (Computer-) Netzwerks.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Android. http://developer.android.com/.

[2] Google Protocol Buffer. https://developers.google.com/protocol-buffers/.

[3] OpenCV. http://docs.opencv.org/.

[4] Pebble-SDK. http://developer.getpebble.com/sdk/.

[5] PebbleKit-Android. https://github.com/pebble/pebble-android-sdk/.

[6] Projektmanagementsoftware Redmine. http://www.redmine.org/.

[7] Versionsverwaltung Git. http://git-scm.com/.

[8] A Threshold Selection Method from Gray-Level Histograms. Systems, Man and Cybernetics, IEEE Transactions on, 9(1):62–66, Jan 1979.

[9] AISBL, GS1: GS1 General Specifications, Januar 2014. Version 14, Jan-2014.

[10] Arthur, David und Sergei Vassilvitskii: K-means++: The Advantages of Careful Seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, Seiten 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[11] Azuma, Ronald, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier und Blair MacIntyre: Recent Advances in Augmented Reality. IEEE Comput. Graph. Appl., 21(6):34–47, November 2001.

[12] Blum, Harry: Biological shape and visual science (part I). Journal of Theoretical Biology, 38(2):205–287, Februar 1973.

[13] Brown, Jeff: ZBar bar code reader. http://zbar.sourceforge.net/.

[14] Chai, D. und F. Hock: Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. In: Information, Communications and Signal Processing, 2005 Fifth International Conference on, Seiten 1595–1599, 2005.

[15] Coates, Adam, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu und Andrew Y Ng: Text detection and character recognition in scene images with unsupervised feature learning. Document Analysis and Recognition (ICDAR), 2011 International Conference on, Seiten 440–445, 2011.

[16] Gäth, Markus: Low Resolution Text Recognition: Text Localization in Natural Scenes. 2007.

[17] Gllavata, Julinda: Extracting Textual Information from Images and Videos for Automatic Content-Based Annotation and Retrieval. Doktorarbeit, Philipps-Universität Marburg, 2007.

[18] greenrobot: greenrobot/EventBus.

[19] Hompel, Michael, Volker Sadowsky und Maria Beck: Kommissionierung - Materialflusssysteme 2 - Planung und Berechnung der Kommissionierung in der Logistik. Springer, 2011.

[20] Lloyd, S.: Least Squares Quantization in PCM. IEEE Trans. Inf. Theor., 28(2):129–137, September 2006.

[21] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vision, 60(2):91–110, November 2004.

[22] Manning, Christopher D., Prabhakar Raghavan und Hinrich Schütze: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA, 2008.

[23] Mark James Burge, Wilhelm Burger und: Digitale Bildverarbeitung - Eine Einfuhrung mit Java und ImageJ. Springer, 2002.

[24] Mikolajczyk, Krystian und Cordelia Schmid: A performance evaluation of local descriptors. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(10):1615–1630, 2005.

[25] Milker, Sven: Bewegungserkennung mit Smartphones mittels deren Sensoren. Bachelorarbeit, Universität Koblenz-Landau, 2012.

[26] O'Hara, Stephen und Bruce A. Draper: Introduction to the Bag of Features Paradigm for Image Classification and Retrieval. CoRR, abs/1101.3354, 2011.

[27] Oliva, Aude und Antonio Torralba: Building the gist of a scene: The role of global image features in recognition. Progress in brain research, 155:23–36, 2006.

[28] Philbin, J., O. Chum, M. Isard, J. Sivic und A. Zisserman: Object Retrieval with Large Vocabularies and Fast Spatial Matching. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[29] Pinhanez, Claudio: Augmenting Reality with Projected Interactive Displays. In: in Proc. VAA, 2001.

[30] Rivera-Rubio, J., S. Idrees, I. Alexiou, L. Hadjilucas und A.A. Bharath: A dataset for Hand-Held Object Recognition. In: Image Processing (ICIP), 2014 IEEE International Conference on, Seiten 5881–5885, Oct 2014.

[31] Rivera-Rubio, J., S. Idrees, I. Alexiou, L. Hadjilucas und A.A. Bharath: Small Hand-held Object Recognition Test (SHORT). In: Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on, Seiten 524–531, March 2014.

[32] Rosebrock, Adrian: Detecting Barcodes in Images with Python and OpenCV. http://www.pyimagesearch.com/2014/11/24/detecting-barcodes-images-python-opencv/.

[33] Simon, Jonathan: Head First Android Development -. O'Reilly Media, Incorporated, Sebastopol, California, 1. Aufl. Auflage, 2012.

[34] Sivic, Josef und Andrew Zisserman: Video Google: A Text Retrieval Approach to Object Matching in Videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2, ICCV '03, Seiten 1470–, Washington, DC, USA, 2003. IEEE Computer Society.

[35] Szeliski, Richard: Computer vision: algorithms and applications. Springer, 2010.

[36] Vedaldi, Andrea und Andrew Zisserman: VGG Computer Vision Practicals. https://sites.google.com/site/vggpracticals/.

[37] Wolf, Christian und Jean-Michel Jolion: Model based text detection in images and videos: a learning approach. Technischer Bericht, LIRIS INSA de Lyon, 2004.

[38] Zhong, Shi: Efficient online spherical k-means clustering. In: Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on, Band 5, Seiten 3180–3185 vol. 5, July 2005.