Universität Hamburg, Fachbereich Informatik

Baccalaureatsarbeit

Entwurf und Implementierung einer Architektur zur Suche von Büchern durch Serviceroboter

Thomas Dorka
Matrikelnr. 5401295

30. September 2005

Betreuer: Prof. Dr. J. Zhang

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit entstand im Rahmen des Hauptstudiumprojekts „Serviceroboter“ des Arbeitsbereichs Technische Aspekte Multimodaler Systeme (AB TAMS) am Fachbereich Informatik der Universität Hamburg. In dieser Arbeit wird ein System entworfen und umgesetzt, das einem Serviceroboter in einer natürlichen Umgebung ermöglicht, von einem Benutzer gewünschte Bücher an verschiedenen, jedoch definierten Orten zu suchen und zu finden.

1.1 Aufgabenstellung

Die Idee dieser Arbeit ist, ein Programm zu entwerfen, das die vielfältigen Möglichkeiten eines autonomen Serviceroboter nutzt, um Menschen in einer natürlichen Umgebung Arbeit abzunehmen, und somit die Fähigkeiten dieses Roboters zu demonstrieren. Am Wichtigsten hierbei ist, dass es sich um eine natürliche Umgebung handelt. Dazu sollen die Anforderungen an die Umgebung möglichst gering sein oder eine Umgebung ohne viel Aufwand zu einer passenden Umgebung umgewandelt werden können.

Folgendes Szenario ist vorstellbar: Während der Erstellung einer Arbeit, eines Berichts oder bei der Implementierung eines Programms stellt der Benutzer fest, dass er ein Buch benötigt. Ist das Buch in der Bücher-Datenbank enthalten, beauftragt er den Roboter, dieses Buch zu suchen. Der Roboter nimmt dem Benutzer die Arbeit ab, verschiedene Regale nach diesem Buch zu durchsuchen, und meldet ihm, an welchem Ort das Buch zu finden ist, sofern es gefunden wurde. Der Benutzer kann während der Suche seine Arbeit fortsetzen und bei einer positiven Rückmeldung des Roboters das Buch an der gemeldeten Stelle abholen.

1.2 Das Projekt

In dem Projekt „Serviceroboter“ ging es darum, Verfahren aus den verschiedenen Bereichen der modernen Robotik kennenzulernen und ausgewählte Probleme der Robotik zu bearbeiten. Die erstellten Programme konnten am Serviceroboter des AB TAMS getestet werden. Für die verschiedenen Aufgaben wurden Gruppen gebildet. Die während des Projektes entstandenen Algorithmen wurden zu einem großen Teil mit dem Mathematikprogramm Matlab realisiert, für das eine umfassende Grafik-Bibiliothek existiert.

Für die vorliegenden Arbeit wurde das gesamte entworfene System, aufbauend auf der Roblet®-Technologie[1] (siehe auch Kapitel 3.4), in Java[2] neu implementiert.

1.3 Aufbau der Arbeit

Um die Problemstellung zu erläutern und eine Lösung vorzustellen, wird folgendermaßen vorgegangen: Nach der Einleitung wird in Kapitel 2 ein Überblick über den Stand der Technik bei gegebenen Ansätze, Techniken und Vorgehensweisen erläutert. Danach werden in Kapitel 3 die Roboterplattform und anschließend in Kapitel 4 die Umgebungsanforderungen vorgestellt, anhand derer das Projekt an der Universität Hamburg umgesetzt und getestet wurde. Darauf folgt in Kapitel 5 die Einführung der verwendete Algorithmen und Hilfsmittel, die in der nachfolgenden detaillierten Ablaufbeschreibung in Kapitel 6 benutzt werden. Die Ablaufbeschreibung selbst legt ausführlich die Vorgehensweise des gesamten Algorithmus dar. Abschließend wird dieser Ansatz in Kapitel 7 mit Rückblick auf die Aufgabenstellung erneut betrachtet und ein Ausblick auf Erweiterungs-, Verfeinerungs- oder Verbesserungsmöglichkeiten gegeben.

2 Stand der Technik

Zum Thema „Bücher von einem Serviceroboter holen lassen“ gibt es bereits einige Ansätze und Ideen. Viele dieser Ideen gehen jedoch davon aus, dass die Bücher geordnet sind oder bekannt ist, wo die Bücher stehen. Dabei sind auch die Maße des Regals, in dem das Buch steht, bekannt. In einer natürlichen Umgebung kann von diesen Voraussetzungen normalerweise nicht ausgegangen werden.

In [13] wird ein Robotersystem beschrieben, das es Bibliotheken ermöglichen soll, Bücher auch an Standorten ausserhalb der Bibliothek selbst zu lagern. Die Notwendigkeit dazu besteht, weil viele Bibiliotheken trotz digitaler Medien noch sehr viele gedruckte Medien vorhalten. Das führt zu Platzmangel innerhalb der Bibliothek. Eine dezentrale Lagerung der Bücher wäre also sehr hilfreich. Dabei muss jedoch die Zugänglichkeit der Medien gewahrt werden. Dazu wurde das Roboterprojekt „Comprehensive Access to Printed Material (CAPM)“ entworfen. Ein Roboter soll in dem dezentralen Lager gewünschte Bücher holen und zur Weiterverarbeitung zur Verfügung stellen. Idealerweise wäre für eine solche Weiterverarbeitung eine Online Anwendung, die es ermöglicht, das Buch aus der Ferne, mittels besonderer Hardware zum Umblättern und einer Kamera zum Aufnehmen der Buchseiten, zu lesen. Alternativ bietet sich noch die Möglichkeit, die Bücher mittels Scanner und Optical Character Recognition-Software (OCR-Software) digital zu speichern und zum Lesen anzubieten.
Dieser Ansatz weist interessante Ideen auf, jedoch wird nur eine Studie ohne konkrete Tests in einer realen Umgebung geboten. Es wird hauptsächlich die Steuerung der Hardware-Module beschrieben, aber nicht auf Büchersuche eingegangen.

In [14] beschreiben Tomizawa et al. einen Roboter, der Bücher aus einem Regal greift, sie für den Benutzer aufschlägt und die gewünschten Seiten abfotografiert. Hier wird sogar ein konkreter Roboter vorgestellt, der eine spezielle Hardware zum Greifen und Aufblättern des Buches besitzt. Die Bücher in einem Regal werden dabei durch Entfernungsmessung voneinander unterschieden. Eine optische Erkennung der Bücher erfolgt nicht. Das Greifen eines Buches scheint jedoch weit entwickelt. Nachteilig ist aber eventuell die speziell benötigte Hardware.

In [15] wird von denselben Autoren ein erweiterter Roboter beschrieben, der aber im Grunde die gleichen Fähigkeiten besitzen soll wie der in [14] beschriebene. Hier wird aber eine kombinierte Technik aus Laser und optischer Erkennung verwendet, um die Bücher zu identifizieren. Die Orientierung von nicht senkrecht stehenden Büchern wird mit Hilfe von Kantenbildern ermittelt. Es wird in beiden Veröffentlichungen kein Buch aktiv gesucht, sondern dem Benutzer nur eine Auswahl an vorhandenen Büchern zur

Wahl gestellt, von denen das Gewünschte daraufhin gegriffen wird. Die Steuerung zu einem gesuchten Buch wird also nicht betrachtet.

Eine sehr weit entwickelte Arbeit wird in [10, 11, 12] beschrieben. Hier wird ein Robotersystem entwickelt, das Bücher suchen und auch greifen kann. Dazu wird mittels Kamera ein Bild der Bücher aufgenommen, die Signaturen mittels OCR extrahiert und mit einer gesuchten Signatur verglichen. Es wird allerdings davon ausgegangen, dass sich die Bücher an einem definierten Ort befinden, z.B. in einem bestimmten Regal auf einer bestimmten Regalbretthöhe. Die Ermittlung der Signaturen erfolgt, ähnlich wie in dieser Arbeit, durch die Erstellung von Schwarz-Weiss-Bildern und einer darauf durchgeführen OCR. Das System soll Sprachbefehle von Benutzern annehmen. Nach der Aufforderung, ein Buch zu bringen, wird in bestimmten Regalen nach der Signatur des Buches gesucht. Ist das gesuchte Buch gefunden, so wird es gegriffen und dem Benutzer gebracht.

3 Der Roboter

Das zur Implementierung benutzte System ist der Service-Roboter der Universität Hamburg (siehe Abb. 3.1, [16]). Er besteht aus mehreren Modulen, von denen einige nötig sind, um Bücher suchen zu lassen. Dazu gehören die Roboterplattform, die es dem Roboter ermöglicht, an verschiedene Orte zu fahren, eine Kamera, um Bilder aufzunehmen, und eine Schwenk-Neige-Einheit (engl. Pan-Tilt-Unit, PTU), mit der die Kamera geschwenkt und geneigt werden kann. Diese Einheiten werden im Folgenden kurz beschrieben. Gesteuert wird das System über die Roblet®-Architektur, die es ermöglicht, den Roboter unabhängig von Hardwaredetails zu programmieren. Auch auf diese wird kurz eingegangen.

3.1 Die Roboterplattform

Die Roboterplattform des Serviceroboters des AB TAMS ist eine modifizierte Version des MP-L655 der Firma NEOBOTIX (GPS GmbH Stuttgart)[17]. Sie dient als Basis des Roboters und ermöglicht seine Bewegung.

Sie ist ca. 150kg schwer und kann 100kg Nutzlast befördern. Die Batterien sorgen für knapp 12 Stunden Autonomie - zusätzliche Geräte verringern diese Zeit. Die Fahrgeschwindigkeit beträgt mehr als 1m/s und die Drehgeschwindigkeit mehr als 90°/s. Weiterhin besitzt die Plattform zwei gefederte Differentialantriebe in der Mitte der Plattform und ein Stützrad hinten sowie zwei vorne, wodurch es möglich ist, auf der Stelle Drehungen auszuführen. Vorhanden sind außerdem ein Präzisions-Gyroskop und je ein Rad-Dreh-Sensor mit 4096 Schritten/Motorumdrehung pro Antriebsrad.

Auf der Plattform sind andere Einheiten montiert, unter anderem zwei Laserscanner Typ LS 200 der Firma SICK[18], ein omnidirektionales Sichtsystem, bestehend aus einer Sony DFW-SX900[19] und einem hyperbolischen Spiegelsystem[20], ein Stereo-Kamerasystem, bestehend aus einer PTU der Firma Directed Perceptions[21] und zwei Sony DFW-VL500 Kameras[22], und ein Roboterarm Mitsubishi Heavy Industries PA10-6C[23] samt Dreifinger-Roboterhand der Firma Barrett Technologies, Inc.[24] und einer Handkamera[25].

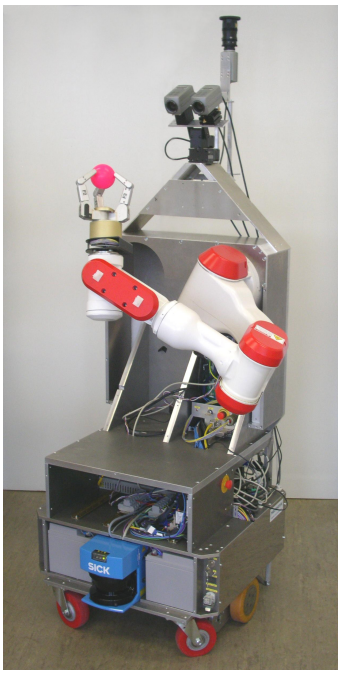


Abbildung 3.1: Der Service-Roboter des AB TAMS am Fachbereich Informatik der Universität Hamburg

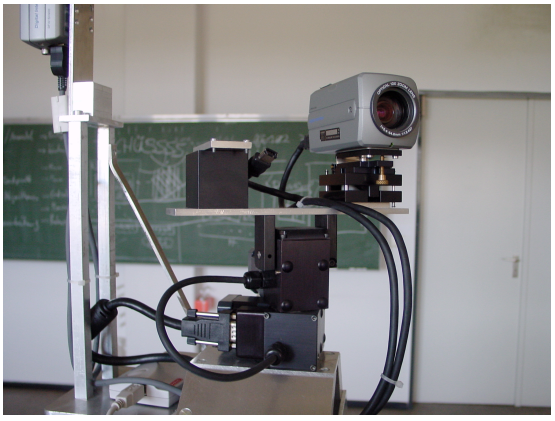


Abbildung 3.2: Die PTU mit einer der beiden Kameras des Roboters. Die PTU erlaubt Drehung und Neigung der Kamera.

3.2 Die Kamera

Als Kamera kommt eine der beiden DFW-VL500 zum Einsatz (siehe auch Abb. 3.2). Die Kamera kann Bilder in VGA-Auflösung (Video Graphics Array, bis zu 640x480 Bildpunkten) bei einer Framerate von 30 Bildern/s machen. Sie besitzt einen 12-fach optischen Zoom, der wie der Fokus und die Iris motorisiert eingestellt werden kann. Der IEEE1394-1995 (Institute of Electrical and Electronics Engineers) Standard wird unterstützt. Die Übertragung der Bilder sowie die Stromversorung erfolgt über die IEEE1394-Schnittstelle. Die Kamera muss nicht kalibriert werden, da sich die unkalibrierten Bilder der Kamera als völlig ausreichend herausgestellt haben.

Positiv an dieser Kamera ist die weitgehend automatische Regelung von Kameraeigenschaften, wie Weißabgleich und Iris-Einstellung, und der hohe Zoomfaktor, der softwareseitig gesteuert werden kann. Die hohe Bildgeschwindigkeit von 30 Bildern/s ist nicht zwingend notwendig, da nur Einzelbilder verarbeitet werden. Die Auflösung ist für dieses Projekt ausreichend. Eine höhere Auflösung wäre aber für die Bildverarbeitung an vielen Stellen von Vorteil.

Nachteilig an dieser Lösung ist, dass die Kamera keine automatische Fokussierung besitzt. Scharfe Bilder sind jedoch für die Bildverarbeitung, die für die Suche nach Büchern eingesetzt wird (z.B. Kantenerkennung, siehe Kapitel 5.1), notwendig. Damit scharfe Bilder aufgenommen werden können, wird in Kapitel 5.2 ein Autofokus-Algorithmus vorgestellt, der per Software implementiert werden kann.

3.3 Die Pan-Tilt-Unit

Die Pan-Tilt-Unit, kurz PTU, ist die Schwenk-Neige-Einheit für das Stereokamera-System (siehe dazu Abb. 3.2). Da nur eine Kamera zum Einsatz kommt, ist also auch nur deren Position wichtig.

Die PTU ermöglicht es, die Kamera in bestimmte Richtungen zu drehen. Sie ist um ca. 159 Grad drehbar in beide Richtungen und kann nach oben ca. 31 Grad und nach unten ca. 47 Grad geneigt werden. Die Auflösung beträgt 0,051428 Grad in alle Richtungen, die maximale Geschwindigkeit 300 Grad/s und die Tragkraft 2,72 Kilogramm. Die Steuerung erfolgt über eine RS232/485 Schnittstelle.

Das Schwenken macht eine zeitaufwendige Bewegung der Roboterplattform überflüssig und beschleunigt dadurch den Ablauf der Büchersuche. Die Neigung der Kamera führt zu einem erweiterten Sichtfeld, da auch Bilder oberhalb bzw. unterhalb der horizontalen Ausrichtung der Kamera möglich sind.

3.4 Die Roblet®-Technologie

Die Roblet®-Technologie, detailliert erläutert in [1], ist eine neuartige Form von verteilten Systemen in der Robotersteuerung. Ziele dieser Architektur sind:

- einfache Benutzbarkeit zur Verfügung stellen,
- tiefergehende Funktionen für Experten verfügbar machen,
- ständige Laufzeit garantieren,
- Robustheit und Fehlertoleranz bieten und
- Arbeit an entfernten, nicht immer erreichbaren Orten ermöglichen.

Um die einfache Benutzung zu gewährleisten, muss das System auch für Benutzer ohne tiefergehende Kenntnisse der Hardware und der Programmierung derselben zugänglich sein und ohne viele neu zu erlernende Programmiertechniken auskommen. In der Roblet®-Technologie wird dies durch verschiedene Konzepte möglich. Zunächst ist sie in Java implementiert und erfordert von Benutzern nur die Kenntnis dieser Programmiersprache. Da Java auf fast allen Systemarchitekturen eingesetzt werden kann, bleibt für den Benutzer das Betriebssystem des Steuerrechners des Roboters verborgen und es kann unabhängig davon programmiert werden. Es wird also von der vorhandenen Hardware abstrahiert. Dies trifft auch auf die zu steuernden Sensoren oder Aktoren zu. Für Benutzer des Systems werden die Hardware-Einheiten des Roboters gekapselt, so dass bei verschiedener Hardware für den Benutzer die Steuerung der Einheiten gleich bleibt. Die Verwendung von bereits bestehenden Bibliotheken für Hardware-Komponenten wird durch das Java Native Interface (JNI) ermöglicht, sie müssen also nicht zwingend neu implementiert werden. Die Netzwerktechnologie wird ebenfalls gekapselt, so dass die Implementierung eines neuen Systems auf einem Client-Rechner erfolgen kann und die Roboter-spezifischen Teile für den Benutzer wie lokale Objekte erscheinen und benutzt werden können. Dem Benutzer wird es auf einfache Weise möglich gemacht, mit einem Roboter zu interagieren, ihn zu steuern, ihn zu überwachen oder ihm Aufgaben zuzuteilen. Dazu besitzt jeder Roboter einen oder mehrere Roblet®-Server, die Aufgaben übernehmen können. Die Client-Anwendung sendet ein Roblet® an einige der Roblet®-Server und wartet dann auf deren erfolgreiche Ausführung oder interagiert mit ihm.

Da ein Roboter ständig einsetzbar sein soll, erlaubt es die Roblet®-Technologie, neue Teile des Servers einzusetzen oder vorhandene Teile zu ersetzen, ohne dass das System heruntergefahren oder anderweitig gewartet werden muss. Das Einsetzen der neuen Server kann aus der Ferne erfolgen, so dass ein Zugang zum Roboter nicht zwingend erforderlich ist. Diese Technik erlaubt auch den Einsatz an Orten, die für den Benutzer zu gefährlich oder unzugänglich sind, z.B. sterile Orte.

Die Roboter-spezifischen Softwareteile werden in der Roblet®-Technologie sicher ausgeführt, auch wenn Fehler auftreten. So kommt es nicht zu einem Dead-Lock der Einheit und eine Wartung ist nicht nötig. Sollte es innerhalb der Ausführung eines solchen Software-Teils zu Fehlern kommen, führt das System die Arbeit, sofern möglich, fort oder setzt sich ansonsten in einen Ausgangszustand zurück, aus dem es wieder neue Aufgaben übernehmen kann. Im Falle eines Fehlers wird eine Meldung an die Client-Anwendung gesendet. Alternativ dazu kann auch eine Fehlerbehandlung innerhalb des Roblets erfolgen. So muss ein aufgetretener Fehler nicht unbedingt das Ende der Ausführung des Programmes bedeuten, sondern ein angemessenes Verhalten bei Fehlern kann im Roblet® vorgegeben werden.

Der Ablauf könnte bei einer Anwendung für einen Roboter z.B. so aussehen: Mittels eines Verzeichnisdienstes wird ein verfügbarer Roboter gesucht, der die für die Aufgabe benötigten Fähigkeiten besitzt. An dessen Roblet®-Server wird ein Roblet® über das Netzwerk gesendet. Der Roboter führt das Roblet® auch bei auftretenden Fehlern zu Ende und versetzt sich danach, unabhängig davon, ob Fehler auftraten oder nicht, wieder in einen Zustand, in dem er für andere Benutzer bzw. Client-Anwendungen verfügbar ist.

4 Die Umgebung

Da für die Implementierung des Systems einige wenige Einschränkungen für die Umgebung notwendig sind, werden diese hier vorgestellt. Die Beschränkungen sollen dabei möglichst gering sein, damit das System in fast jeder Umgebung leicht eingesetzt werden kann.

4.1 Die Regale

An die Regale müssen einige Anforderungen gestellt werden. Zunächst muss das Regal natürlich zugänglich für den Roboter sein. Das schließt ein, dass der Weg dorthin für den Roboter passierbar und das Regal erreichbar ist. Hinzu kommt, dass vor dem Regal genügend Platz zur Verfügung steht, damit der Roboter sich in einiger Entfernung positionieren kann und auch noch eventuelle Bewegungen parallel zum Regal möglich sind (siehe dazu Kapitel 6.4). Die Entfernung des Roboters zum Regal hat Einfluss auf das Sichtfeld des Roboters, eine Entfernung von ca. 150 cm hat sich als gut erwiesen. Bei dieser Entfernung kann der Roboter einen sehr großen Bereich des Regals erfassen. Er steht jedoch noch nah genug an den Büchern, damit der Zoom der Kamera eine gute Sicht auf die Signaturaufkleber ermöglicht. Ausserdem sind auch gute Lichtverhältnisse nötig, da ein großer Teil des Systems auf Bildverarbeitung beruht. Sollte ein Regal sehr breit sein, kann es eventuell notwendig sein, es in Abschnitte einzuteilen, damit der Roboter diese sequentiell durchsuchen kann. Dies ist deswegen notwendig, da der Roboter durch seine beschränkte Kamerasicht eventuell nicht das ganze Regal erfassen kann. Ebenfalls durch die Kamerasicht beschränkt ist die Anzahl der Böden, die innerhalb des Regals benutzt werden können. Es werden nur die Buchreihen erkannt, die vollständig innerhalb des Sichtfeldes des Roboters sind. Dabei ist die Höhe der Regalböden uninteressant, ebenso wie der Abstand zwischen ihnen. Wichtig ist nur die Höhe der Buchreihe. Das Durchsuchen von z.B. CD- oder DVD-Regalen wäre also prinzipiell möglich. Die Belegung der Regalböden mit Objekten ist frei von Beschränkungen. Zusätzliche Objekte im Regal können den Vorgang der Suche verlängern, verhindern aber nicht das Finden eine Buches.

Für den Test des Systems wurde ein einfaches zweiteiliges Metallregal mit mehreren Einlegeböden verwendet (siehe Abb. 4.1(a)). Die Aufstellung und Belegung erfolgte nach den oben genannten Kriterien. Für die Belegung wurden neben den Büchern z.B. verschiedene Arten von Flaschen und einige Aktenordner benutzt. Das Regalsystem wurde in zwei Teile eingeteilt, die sich zu einem geringen Teil überlappen.

(a) Bild des Regalsystems mit einer Beispiel-Belegung der Regalböden.

(b) Bild der Bücher, die für den Test des Systems genutzt wurden.

Abbildung 4.1: Das Regalsystem und eine Auswahl von Büchern, die für den Test und den Entwurf des Systems benutzt wurden.

4.2 Die Bücher

Auch an die Bücher bzw. die Buchreihen innerhalb eines Regals müssen einige Anforderungen gestellt werden. Zunächst müssen pro Buchreihe mehrere Bücher vorhanden sein. Diese Beschränkung ergibt sich daraus, dass die Buchreihen anhand einer Häufung von Kanten detektiert werden (siehe hierzu Kapitel 6.3). Die Reihen sollten bündig mit der Regalfront abschließen. Stehen die Bücher zu weit hinten oder unregelmäßig nach vorne und hinten verteilt, kann es vorkommen, dass die Kamera die Signaturaufkleber nicht mehr erfassen kann. Bücher mit einfarbigem Buchrücken sind wünschenswert, da Bücher mit texturierten Buchrücken die Erkennung der Signaturaufkleber erschweren (siehe auch Kapitel 6.7). Für die OCR müssen die Aufkleber mit den Signaturen der Bücher gut und vollständig zu erkennen sein, so dass die Bücher also auch einen entsprechend breiten Buchrücken benötigen. Die Bücher müssen möglichst aufrecht im Regal stehen, da ein schräg stehendes Buch auch eine schräg stehende Signatur bedeutet, was wiederum für die Erkennung der Signaturen Nachteile hat.

Für die Tests des Systems wurden aktuelle Bücher aus der Bibliothek des Fachbereichs Informatik benutzt (Abb. 4.1(b)). Die Auswahl erfolgte zufällig, jedoch mit Rücksicht auf die oben genannten Kriterien, sowie die Kriterien für die Signaturaufkleber im folgenden Absatz. Auch Bücher mit sehr ähnlichen Signaturen wurden ausgewählt (nur ein Zeichen Unterschied), um die Robustheit der Erkennung zu testen. Einige Bücher hatten einen sehr schmalen Buchrücken, die der Signaturaufkleber in der Breite komplett überdeckte.

Bei diesen Büchern kam es in Einzelfällen zu Fehlern, die auf die Erkennung der Aufkleber zurückzuführen sind.

4.3 Die Signaturaufkleber

Wie schon in Kapitel 4.2 erwähnt, müssen die Signaturaufkleber auf den Buchrücken vollständig zu sehen sein. Ein nur unvollständig sichtbarer Aufkleber erlaubt keine korrekte Erkennung der Signatur. Zusätzlich ist es wichtig, dass sie nicht durchscheinend sind, da sie sonst nur schwer durch die Bildverarbeitung von dem Buchrücken unterschieden werden können.

Alle Aufkleber auf den zufällig ausgewählten Büchern entsprachen diesen Kriterien. Zwischenzeitlich wurden auch Bücher mit leicht durchscheinenden Signaturaufklebern mit Erfolg getestet.

5 Algorithmen und Hilfsmittel

In diesem Kapitel werden Algorithmen und Hilfsmittel vorgestellt, die für das Projekt von tragender Rolle sind. Neben dem Canny-Kantenfilter wird auch der benutzte Autofokus-Algorithmus, die verwendete OCR-Software und die Levenshtein Distance kurz vorgestellt werden.

5.1 Der Canny-Algorithmus

In seiner Arbeit [4] beschreibt Canny einen Kantenerkennungsalgorithmus. Dieser Algorithmus ist in der Bildverarbeitung weit verbreitet. Es wird kurz erläutert, wie dieser Algorithmus Kanten in einem Bild findet. Da der Algorithmus nur auf Grauwert-Bildern arbeitet, werden auch nur diese betrachtet.

In Grauwert-Bildern sind Kanten durch stark verschiedene Helligkeitswerte zweier benachbarter Pixel charakterisiert. Solche Unterschiede können jedoch auch auftreten, wenn bei der Aufnahme der Bilder ein Rauschen aufgetreten ist. Um dieses Problem zu umgehen, sollte das Bild vorher mit einem Gaußschen Weichzeichner bearbeitet werden, damit solche Fehler minimiert werden. Dafür wird eine Maske erstellt, die die Gaußsche Normalverteilung annähernd darstellt. Das Bild wird mit dieser Maske gefaltet und man erhält ein weichgezeichnetes Bild.

Auf diesem Bild werden nun die Gradienten der Kanten mittels eines linearen Filters jeweils der horizontalen und vertikalen Richtung ermittelt und der Winkel des Anstiegs einer potentiellen Kante errechnet. Da hierbei alle Pixel einen Winkel zugeordnet bekommen, muss gefiltert werden, wo sich wirkliche Kanten befinden. Dazu benutzt Canny einen Algorithmus, den er „Non-maximum Suppression“ nennt. Dabei wird für jeden Pixel anhand der acht umliegenden Pixel und des Anstiegswinkels der Kante errechnet, ob dieser Pixel das Maximum dieser Kante darstellt. Ist dies der Fall, so wird er als Kantenpixel, andernfalls als Nicht-Kantenpixel markiert. Anschließend werden mittels eines Hysterese-Verfahrens die Kanten verfolgt. Dabei werden zwei Schwellwerte K und S mit $K > S$ benötigt, anhand derer die Kanten verfolgt werden. Ist ein Pixelwert größer oder gleich K, so ist er automatisch ein Kantenpixel. Jeder Pixel, dessen Pixelwert größer oder gleich S ist, wird nur dann als Kante gezählt, wenn einer seiner acht Nachbarpixel ein Kantenpixel ist. Dabei ist es unabhängig davon ob dieser Pixel ein direkter Kantenpixel ist, weil sein Pixelwert größer oder gleich K ist. Er kann ebenso ein indirekter Kantenpixel sein, weil er eine Verbindung zu einem anderen Kantepixel hat. Es muss also

von jedem Pixel mit Pixelwert größer oder gleich S einen Pfad über eben solche Pixel zu einem Pixel geben, dessen Pixelwert größer oder gleich K ist, damit dieser Pixel ein Kantenpixel ist. Insgesamt entsteht ein Kantenbild, das für jeden Pixel die Information enthält, ob er ein Kantenpixel ist oder nicht.

5.2 Der Autofokus-Algorithmus

Da die vorhandene Kamera keine automatische Scharfstellung besitzt, muss auf Softwareebene ein Autofokus implementiert werden. Fokussierung der Bilder ist wichtig, da Bildverarbeitung auf unscharfen Bildern oft schwerer ist, wie z.B. bei der Kantenerkennung (siehe Kapitel 5.1). Unter den heutigen Autofokus-Techniken gibt es aktive und passive Methoden (siehe dazu [8]). Aktive Ansätze, wie z.B. Infrarotsensoren fallen für eine Softwareimplementierung aus, da dazu neue Hardware nötig wäre, die aktiv die Entfernung zu Hindernissen erfasst. So bleibt also noch eine passive Autofokus-Technik. Dazu gibt es heute verschiedene Techniken, die teilweise noch zusätzliche Hardware benötigen. Auf Methoden, die neue Hardware benötigen, wird nicht weiter eingegangen. Interessant bleibt aber "depth from focus", allgemein auch als „Autofokus“ oder „Softwarefokus“ bekannt. In [5] wird beschrieben, dass ein idealer Autofokus-Algorithmus nur ein Maximum hat, monoton ist und nur dann ein Maximum erreicht, wenn das Bild scharf eingestellt ist. Ein solcher Algorithmus ist unter vielen Bildstörungen u.ä. nicht zu finden. Es muss also eine erweiterte Suche für die beste Fokuseinstellung benutzt werden, um eine genaue Einstellung zu erreichen. Da für dieses Projekt eine sehr genaue Fokussierung nicht nötig ist, wird nur eine sehr einfache Suche implementiert, wie später im Kapitel erläutert.

Um generell ein Maß für die Schärfe zu erzeugen, reicht eine Gradientenfunktion über das Bild aus. Xiong und Shafer benutzten in [5] dazu einen Sobel Operator. Auch Krotov betrachtet in [8] eine Gradientenfunktion mit Schwellwert und benutzt einen Kantendetektor. Es wird erwähnt, dass diese Methode nach mehreren Tests die besten Ergebnisse lieferte. Hierbei ergibt sich leider ebenfalls keine Kurve mit nur einem Maximum.

Um den Aufwand für den Autofokus gering zu halten und möglichst schon mit vorhandenen Techniken zu arbeiten, wird der Einfachheit halber der Canny-Kantenfilter benutzt, um die Schärfe eines Bildes zu beurteilen (siehe auch Kapitel 5.1). Dazu wird für ein aufgenommenes Bild das zugehörige Canny-Kantenbild errechnet und die Anzahl der Kantenpixel ermittelt. Je höher diese Anzahl, desto mehr Kanten, also auch ein schärferes Bild. Um den besten Fokus-Wert herauszufinden müsste man für viele Fokus-Einstellungen ein Bild aufnehmen, das Kantenbild erstellen und dann die Anzahl der Kantenpixel aufsummieren. Die Suche nach dem besten Fokuswert müsste mit einer komplizierten Suchfunktion realisiert werden. Dieser Vorgang würde sehr lange dauern, da die ganze Berechnung sowie die Suche zeitaufwendig ist. Um diese Zeit etwas

zu verkürzen, wird ein etwas anderes Verfahren angewendet. Es wird eine Schrittweite festgelegt, in der Fokus-Stichproben gemacht werden, z.B. 20 Einheiten[1]. Die reale Schrittweite richtet sich nach dem Bereich der Fokuseinstellungen und sollte nicht zu klein gewählt werden, da eine lange Suche nach dem besten Fokus das Ergebnis wäre. So werden bei den Fokuseinstellungen 0, 20, 40, ... die Kantenpixelsummen errechnet und man erhält ein vorläufiges Maximum, z.B. bei 80. Um diesen Fokus-Wert herum werden wieder mit bestimmter Schrittweite Stichproben genommen, also z.B. von 60 bis 100 in Schritten der Schrittweite 5. Man erhält wieder ein Maximum und setzt den Algorithmus mit diesem Wert fort, während man die Schrittweite succesive verringert. So kann ein guter Fokuswert etwas schneller errechnet werden. Der eingestellte Fokuswert reicht für die weitere Bildverarbeitung gut aus.

Auf der vorhandenen Roboterplattform dauert ein Durchgang dieses Autofokus-Algorithmus ca. 40-60 Sekunden. Für eine schnelle Verarbeitung ist dieser Algorithmus also nicht optimal, aber er ist ausreichend für den Prototypen, der mit dieser Arbeit entwickelt wurde.

5.3 Die Optical Character Recognition-Software

Eine Optical Character Recognition (OCR) Software wird benötigt, um aus den extrahierten Signaturaufklebern der Bücher die Signatur zu erkennen. Dabei wird versucht, in einem Bild Schriftzeichen zu erkennen.

Für dieses Projekt wird die Open Source Software GOCR[26] benutzt, da sie die Anforderungen nach einigen Tests gut erfüllte. Die Verfügbarkeit des Source-Codes bei Open Source Projekten hat den Vorteil, dass die Software für das Projekt plattformunabhängig bleibt. Der Sourcecode läßt sich so auf annähernd jede Plattform übertragen und dort zu einer ausführbaren Datei umwandeln. Benutzt wird diese Software als kompilierte, ausführbare Datei, so dass sie leicht durch die jeweils zum System passende Datei ersetzt werden kann.

Auf dem Markt sind auch einige kommerzielle Produkte erhältlich, jedoch bieten nicht alle eine solche Plattformunabhängigkeit und Kostenfreiheit.

5.4 Levenshtein Distance (Edit Distance)

Die Levenshtein Distance oder Edit Distance geht auf V.I. Levenshteins Arbeit [6] zurück, in der er Transmissionen von binären Codes auf Kanälen betrachtet, die ersetzen, einfügen und löschen können. Er beweist auch, wann ein Code s Löschungen, Ersetzungen oder Einfügungen korrigieren kann. Sie wird in dieser Arbeit benutzt, um

[1] Eine Umrechnung der Einheiten in z.B. mm wird durch das Datenblatt der Kamera nicht gegeben.

zwei Zeichenketten miteinander zu vergleichen und dadurch ein Maß der Ähnlichkeit zu erhalten.

Laut Definition in [7] ist die Levenshtein Distance „die kleinste Anzahl an Einfügungen, Löschungen und Ersetzungen, die benötigt werden, um eine Zeichenkette oder einen Baum in eine andere(n) zu verändern“. Dabei gelten folgende Regeln mit s und t als Zeichenketten. S soll in t umgewandelt werden: Der Startwert der Distanz ist 0. Muss ein Buchstabe in s eingefügt oder gelöscht werden, so erhöht sich die Distanz jeweils um 1. Wird ein Buchstabe durch einen anderen ersetzt, so erhöht sich die Distanz ebenfalls um 1. Hier ein paar Beispiele, $LD()$ bezeichnet hier die Levenshtein Distance:

- s = test, t = testing ⇒ LD(s,t) = 3, da 3 Zeichen zu s hinzugefügt werden müssen.
- s = test, t = tent ⇒ LD(s,t) = 1, da 1 Zeichen in s ersetzt werden muss.
- s = test, t = tes ⇒ LD(s,t) = 1, da 1 Zeichen aus s entfernt werden muss.
- s = test, t = tents ⇒ LD(s,t) = 2, da 1 Zeichen in s ersetzt und 1 Zeichen hinzugefügt werden muss.

Vorgemerkt sei hier, dass aus den erkannten Zeichenketten der OCR alle Zeichen herausgefiltert wurden, die nicht in einer Signatur vorkommen können. Für die Zeichen, die bei der Levenshtein Distance ersetzt worden wären, ändert sich hierbei nichts. Ersetzungen werden jetzt zu Einfügungen und verändern so die Levenshtein Distance nicht. Anders ist dies allerdings bei den Zeichen, die bei der Levenshtein Distance gelöscht worden wären. Bei diesen Zeichen wird davon ausgegangen, dass die OCR-Software fehlerhafte Zeichen aus Pixelfragmenten erkannt hat, die für die weitere Betrachtung uninteressant sind. Da diese Zeichen theoretisch in sehr großer Anzahl auftauchen können, werden sie einfach eleminiert. Bei der Betrachtung der Distance brauchen diese Fehlerkennungen dann nicht mehr betrachtet zu werden. Die Anzahl an Einfügungen, Löschungen und Ersetzungen wird in diesem Projekt verwendet, um die Unterschiede zwischen der erkannten und der gesuchten Signatur zu finden. Oft wird bei der OCR z.B. ein „O“ als eine Null erkannt. Ein solcher Fehler soll natürlich nicht gleich dazu führen, dass eine Signatur nicht erkannt wird. Ebenso soll es möglich sein, dass der erkannte String zusätzliche Zeichen zu den in der Signatur enthaltenen beinhalten kann. Da aber bei der Levenshtein Distance auch die Länge ein Rolle spielt (zusätzliche Zeichen werden einfach eingefügt, zählen also zur Gesamtanzahl), muss die Anzahl der Zeichenoperationen relativ zur Längendifferenz der beiden Zeichenketten betrachtet werden. In diesem Projekt wurde folgende einfache Rechnung verwendet. Subtrahiert man von der Levenshtein Distance die Differenz der Längen der Zeichenketten A und B, so erhält man die Anzahl der Buchstaben in Zeichenkette A, die nicht in B vorhanden waren (wenn B die längere Zeichenkette ist). Zu dieser Anzahl addiert man dann noch einmal die Levenshtein Distance und erhält so ein Maß, das auch das zufällige Einfügen von zusätzlichen Zeichen berücksichtigt.

6 Suche nach Büchern

Der Ablauf sieht folgendermaßen aus (ein detaillierterer Ablaufplan kann Abb. 6.1 entnommen werden): Zunächst wird die gewünschte Signatur eines Buches als Auswahl aus einer Liste eingegeben. Dann fährt der Roboter einen definierten Punkt vor einem Regal an, in dem er nach dem gewünschten Buch sucht. In dem Regal wird die Position möglicher Buchreihen gefunden, damit diese Bereiche dann konkret mit dem Roboter angefahren werden können. Die Buchreihen werden danach sequentiell durchsucht, bis das gewünschte Buch gefunden wurde. Dazu wird die PTU, auf der die Kamera zur Erstellung von Bildern positioniert ist, und der Zoom der Kamera so eingestellt, dass die Signaturen auf den Büchern mit einer OCR-Software erkannt werden können. Nach einer Ähnlichkeitsanalyse der erkannten Signatur mit der gesuchten Signatur wird entschieden, ob das richtige Buch gefunden wurde. Ist das Buch in dem durchsuchten Regal nicht gefunden worden, kann im nächsten Regal weitergesucht werden.

6.1 Auswahl der Signatur

Die Auswahl der Signatur kann über das implementierte Java-Interface ausgewählt werden, in Abb. 6.2 auf einem Apple-System zu sehen. Die Signaturen der Bücher, die gesucht werden können, sind bisher fest in das Programm integriert, können aber leicht abgeändert werden. Die Auswahl erfolgt über den Titel des Buches. Mehrfachnennungen des gleichen Buches kommen vor. Für jedes Buch wird die jeweilige Signatur ausgewählt. Denkbar wäre hierbei auch nur eine Auswahl des Titels und die automatische Suche nach einem Exemplar mit diesem Titel, unabhängig von der konkreten Signatur eines bestimmten Buches.

6.2 Anfahren des Regals

Um nach Büchern suchen zu können, müssen Orte vorgegeben werden, an denen nach dem gewünschten Buch gesucht werden soll. Diese Orte sind praktischerweise allerdings nicht die Standorte der Regale, sonder die Punkte, die der Roboter anfahren muss, um eine gute Sicht auf das Regal zu haben. Je näher der Roboter am Regal steht, desto kleiner wird sein Sichtfeld. Eine zu große Entfernung macht allerdings später die Erkennung der Signaturen schwieriger. Mit der Roboterplattform des AB TAMS hat sich eine Entfernung der Kamera von 150 cm zum Regal als gut erwiesen.

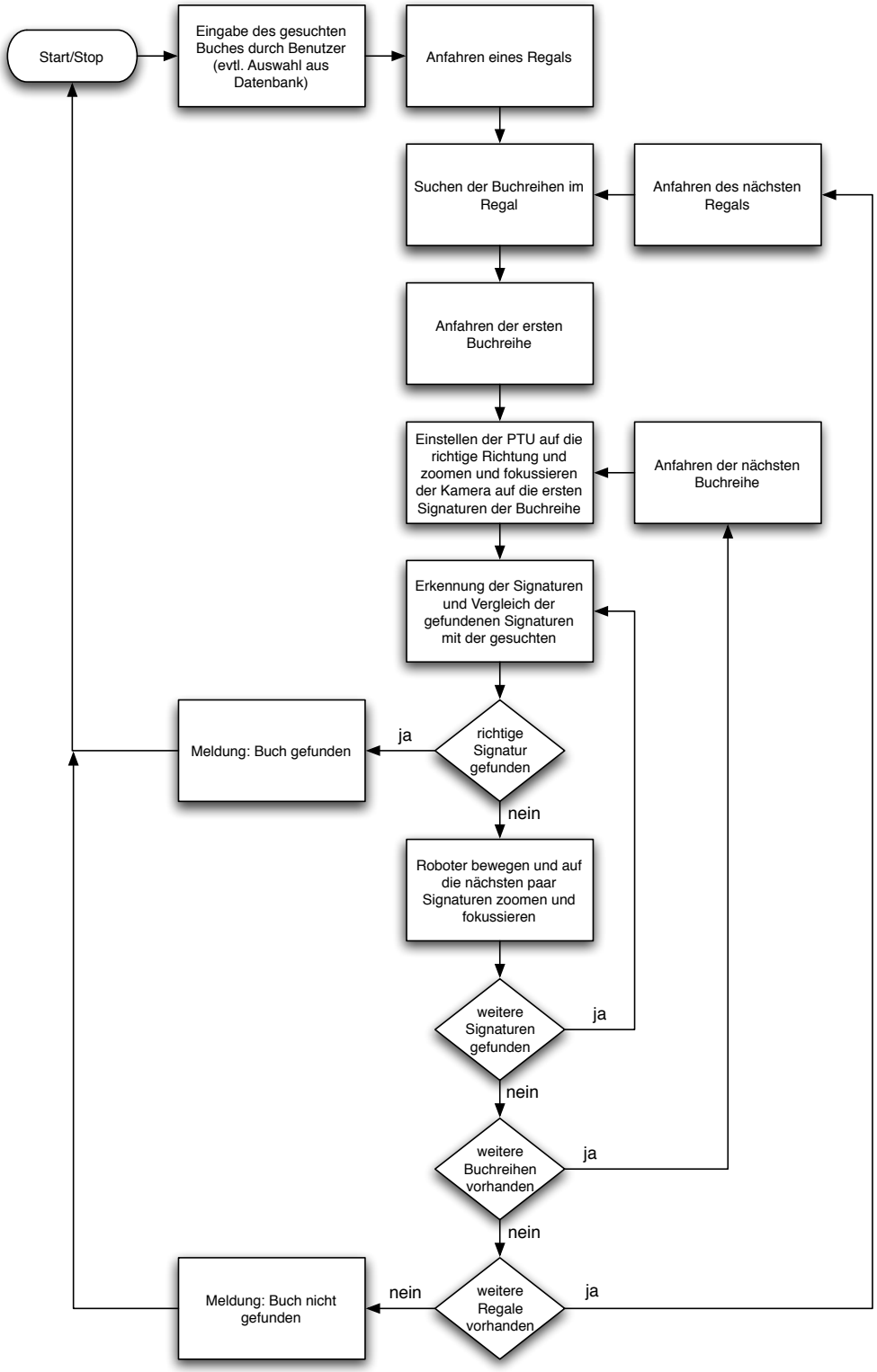


Abbildung 6.1: Ablaufdiagramm für die Suche eines Buches mit einer gewünschten Signatur

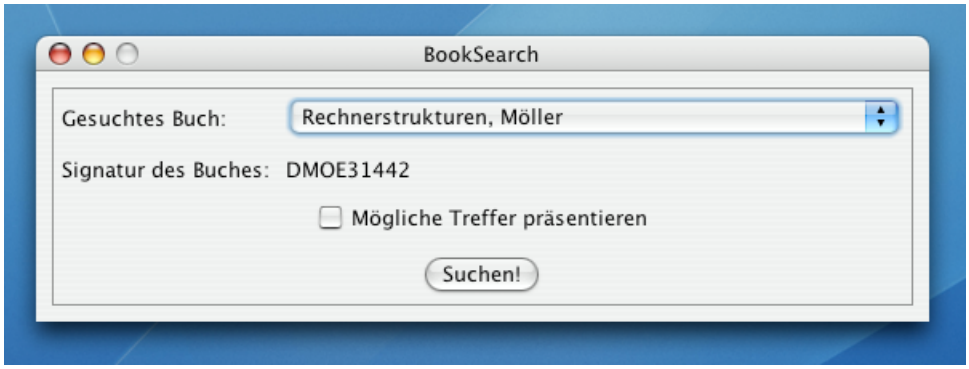


Abbildung 6.2: Interface zur Selektion des zu suchenden Buches. Gezeigt wird ebenfalls die Option, mögliche Treffer sofort zu präsentieren.

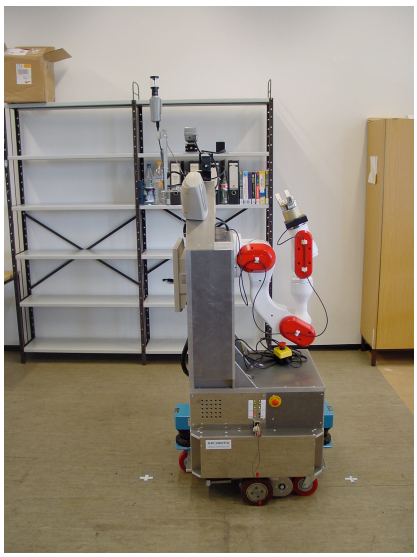


Abbildung 6.3: Die Ausrichtung des Roboters erfolg parallel zum Regal, das sich auf der linken Seite befinden soll. Andere Ausrichtungen sind leicht umzusetzen.

Die Anfahrt an diese Punkte selbst wird von der Roboterplattform vorgegeben und wird hier nicht näher betrachtet. Im Grunde kommen verschiedene Methoden in Frage, um den Roboter an den Zielpunkt zu bewegen. Beispiele wären eine dynamische Fahrtplanung oder eine festgelegte Route zu bestimmten Orten. Wichtig im Allgemeinen ist natürlich, dass die Punkte anfahrbar sind, also dass Kollosionsfreiheit sowohl auf dem Weg zu den Punkten wie auch an den Punkten selbst garantiert wird. Die Orientierung des Roboters zum Regal spielt im Allgemeinen keine Rolle, jedoch ist für den weiteren Verlauf der Suche ein Orientierung parallel zum Regal von Vorteil. In diesem Projekt wird davon ausgegangen, dass der Roboter parallel zu dem Regal steht. Das Regal befindet sich auf seiner linken Seite (siehe auch Abb. 6.3). Andere Orientierungen verhalten sich im Grunde analog. Vorwärtsbewegungen des Roboters, die in diesem Projekt vollzogen werden, werden in Bewegungen nach links umgewandelt, sollte der Roboter frontal zum Regal stehen, oder in Bewegungen nach hinten, sollte er parallel zum Regal stehen, das Regal rechts von sich.

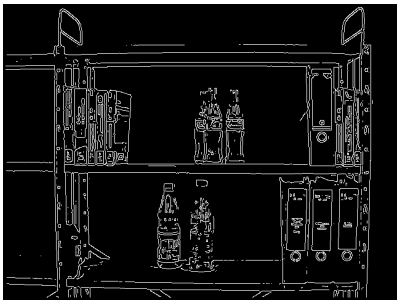
6.3 Suchen der Buchreihen im Regal

Für das Suchen von Buchhreihen in einem Regal wird ein Algorithmus eingesetzt, der sich hauptsächlich auf Kantenbilder und Berechnungen darauf stützt. Die richtige Position des Roboters vor dem Regal wird vorausgesetzt. Bilder zu den einzelnen Verarbeitungsschritten finden sich in Abb. 6.4.

Zunächst muss die Kamera des Roboters auf das Regal ausgerichtet werden. Da der Roboter in einer bestimmten Orientierung vor dem Regal steht, kann die Ausrichtung der PTU fest gespeichert werden. Die PTU muss so ausgerichtet sein, dass sie das Regal als senkrechte Ebene vor sich hat. Dann wird möglichst weit herausgezoomt und das Bild fokussiert, damit ein Bild von dem Regal aufgenommen werden kann (Abb. 6.4(a)).

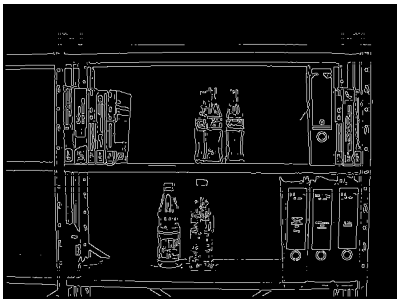
Für die nächsten Schritte wird ausgenutzt, dass Bücher in einem Regal viele vertikale Kanten bilden. Eine Häufung solcher Kanten ist also ein Indikator für eine Reihe von Büchern. Damit dieser Indikator greifen kann ist es nötig, das mehrere Bücher nebeneinander stehen. So wird also von dem aufgenommenen Bild wird ein Kantenbild nach Canny erzeugt (Abb. 6.4(b)), wie in Kapitel 5.1 beschrieben. Der Canny-Algorithmus eignet sich gut für diese Aufgabe, da er klare, ein Pixel breite Kanten im Kantenbild erzeug und auch Kanten im Ursprungsbild nachverfolgt. Für das entstandene Kantenbild werden für jede Zeile die Pixel aufsummiert, die eine Kante darstellen (also im Kantenbild weiß dargestellt sind). Überschreitet diese Summe einen bestimmten Schwellwert, so besteht die Möglichkeit, dass eine Buchreihe in dieser Zeile vorhanden ist. So erhält man also eine Reihe von Zeilen, die eine Buchreihe enthalten könnten (Abb. 6.4(c), 6.4(d)). Aus diesen Zeilen werden jetzt Zeilenregionen erzeugt, da einzelne Zeilen nur schlecht ausgewertet werden können. Dazu werden die Zeilen zusammengefasst, zwischen denen maximal vier Zeilen liegen, die keine Buchreihe enthalten. Danach werden die Bereiche

(a) Das aufgenommene Bild des fokussierten Regals



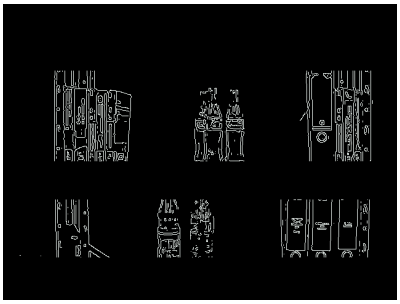
(b) Canny-Kantenbild zu Abb. 6.4(a)

(c) Ausschnitte der Zeilen mit hoher Kantenpixelanzahl

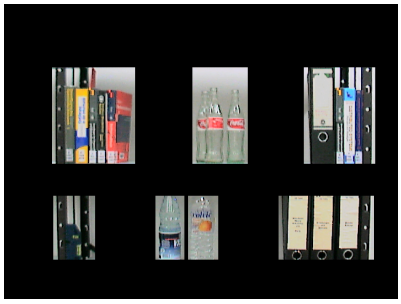


(d) Canny-Kantenbild zu Abb. 6.4(c)

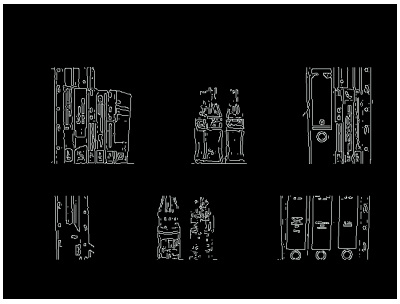
(e) Auschnitte der erkannten Buchreihen-Zeilen



(f) Auschnitte des Canny-Kantenbild zu Abb. 6.4(e)



(g) Auschnitte der erkannten Buchreihen



(h) Auschnitte des Canny-Kantenbild zu Abb. 6.4(g)

Abbildung 6.4: Ablauf der Erkennung der Buchreihen. Gezeigt werden das z.T. beschnittene Kamerabild und die zugehörigen Kantenbilder.

gelöscht, deren Regionshöhe weniger als 80 Zeilen beträgt, da diese wahrscheinlich zu klein sind, um eine Buchreihe zu enthalten. Diese Höhe ist prinzipiell frei wählbar. Bei kleineren Büchern (oder CD's, DVD's) ist ein kleiner Wert besser, könnte aber zur Suche zusätzliche irrelevante Bereiche hinzufügen und den Zeitaufwand dadurch erhöhen. Ein guter Schwellwert ist ein Wert, der etwas unterhalb der Pixelhöhe des kleinsten Buches im Regalbild liegt. So werden keine Bücher übersehen und nur wenige irrelevante Regionen durchsucht.

Die so entstandenen Zeilenbereiche sind also letztendlich interessant, da sie genügend viele vertikale Kanten enthalten und groß genug sind, um Bücher zu enthalten. Das Ergebnis ist in Abb. 6.4(e) und 6.4(f) zu sehen.

Innerhalb dieser Zeilen müssen jetzt noch die Bereiche identifiziert werden, in denen sich Bücher befinden könnten. Dazu wird noch einmal das Kantenbild zur Hilfe genommen. Für jeden identifizierten Zeilenbereich wird nun die Summe der Kanten pro Spalte errechnet. Hohe Werte werden erreicht, wenn z.B. eine vertikale Kante in der Spalte vorhanden ist. Ein Schwellwert identifiziert auch hier wieder die interessanten Spalten. Spalten, in deren Umfeld von 5 Pixeln keine weiteren interessanten Spalten sind, werden nicht weiter betrachtet. Um jede der verbleibenden Spalten wird ein Rahmen von 5 Pixeln gelegt und die sich überschneidenden Rahmen zu einem Rahmen gruppiert. Da sich im Bereich von Büchern solche interessanten Spalten häufen, entsteht um diese Bücher ein Rahmen (Abb. 6.4(g), 6.4(h)).

Diese Rahmen werden in Pixelkoordinaten für die weitere Verarbeitung abgespeichert.

6.4 Anfahren der ersten Buchreihe

Nachdem alle Bereiche, die Bücher enthalten könnten, identifiziert worden sind, wird einer dieser Bereiche angefahren werden. Während des ganzen Vorgangs der Büchersuche in einem Regal werden nach und nach die Bereiche angefahren, bis das richtige Buch gefunden wurde. Sehr große Bereiche werden in kleinere Bereiche eingeteilt.

Die Pixelkoordinaten helfen, den Roboter an die richtige Position zu fahren, um die weitere Verarbeitung vorzunehmen. Da die Entfernung des Roboters zum Regal vorgegeben ist, kann sehr leicht manuell abgemessen werden, wie breit das aufgenommene Bild auf Ebene des Regals ist. Ein ungefährer Wert, der einmalig als Konstante im Programm gespeichert wird, ist hier ausreichend. Anhand dieses Werts kann bestimmt werden, wie weit nach rechts bzw. links sich der Roboter bewegen muss, damit er zentriert vor einem Bücherbereich zum stehen kommt. Dazu wird die horizontale Entfernung des Mittelpunkts der zu bearbeitenden Buchregion vom Ursprungsbildmittelpunkt berechnet. Zu diese Entfernung muss der Roboter nach vorne bzw. hinten fahren, je nachdem, ob die zu durchsuchende Region rechts oder links im Bild ist.

6.5 Einstellen der Pan-Tilt-Unit

Bei der Einstellung der PTU wird ein ähnliche Verfahren wie in Kapitel 6.4 benutzt. Dazu muss die Höhe des aufgenommenen Bildes auf Regalebene einmalig manuell ungefähr ermittelt und dem Programm als Konstante mitgeteilt werden. Diesmal wird die vertikale Entfernung des Buchregionsmittelpunkts zum Bildmittelpunkt in Zentimetern bestimmt. Da die Entfernung der Kamera bekannt ist, kann mit einfacher Trigonometrie der Winkel berechnet werden, um den sich die Kamera noch oben bzw. nach unten bewegen muss, um auf die unterste Zeile der Buchregion ausgerichtet zu sein.

Zusätzlich muss die Kamera auch nach rechts und links geschwenkt werden, um alle Signaturaufkleber einer Buchregion zu erfassen. Angemerkt sei hierbei, dass die Kamera bei der gleichzeitigen Drehung und Neigung der PTU nicht eine Gerade, sondern einen Bogen auf der Regalebene ablichtet. Dieses Problem wird einfach dadurch umgangen, dass sehr breite Buchregionen in schmalere Regionen aufgeteilt werden. Der Winkel, um den die Kamera nach rechts gedreht werden muss, errechnet sich wiederum durch Trigonometrie. Die Hälfte der Breite der Buchregion in Pixeln ist bekannt und kann mittels der Breite des Originalbildes in Zentimeter umgerechnet werden. Die halbe Breite wird deswegen vewendet, da sich der Roboter mittig vor der Buchregion befindet. Mit der Entfernung der Kamera und des Tangens ergibt sich der zu drehende Winkel α. Um diesen Winkel wird die Kamera nach links geschwenkt und ein Bild aufgenommen. In 2-Grad-Schritten wird die Kamera nun nach rechts gedreht und jeweils ein Bild aufgenommen, bis die Kamera um 2α gedreht wurde. Damit wurde die komplette Buchreihe aufgenommen und die Bilder können verarbeitet werden. Die sehr kleine Schrittweite stellt sicher, dass jeder Buchsignaturaufkleber mindestens einmal vollständig im Bild zu sehen ist, besser jedoch zweimal.

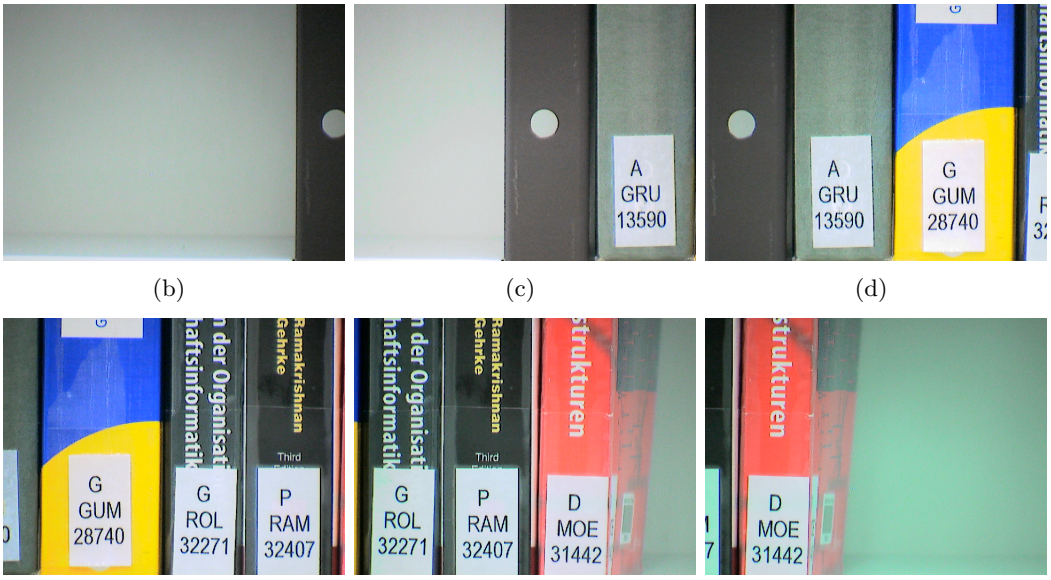
Es hat sich herausgestellt, dass eine Nachregelung des Fokus nicht nötig ist, sobald er einmalig gut auf die Buchrücken eingestellt wurde. Ein kompletter Schwenk über eine Buchregion ist in Abb. 6.5 zu sehen.

6.6 Zoomen der Kamera auf die ersten Signaturen der Buchreihe

Dieser Schritt ermöglicht es der Kamera, Bilder zu liefern, die mittels der OCR auswertbar sind. Da die Auflösung der Kamera beschränkt ist, ist ein stark vergößertes Bild der Signatur vorteilhaft.

Zu beachten ist, das im Projekt dieser Schritt vor der Drehung der PTU ausgeführt wurde. Dieses war deswegen notwendig, weil der Softwarefokus noch ausgeführt werden musste. Die PTU wird also geneigt, die Kamera auf maximalen Zoom eingestellt und das Bild fokussiert. Daraufhin wird die PTU-Drehung ausgeführt. Die Kamera hat nun die erste Signatur im Blickfeld.

(a) Ausschnitt der Buchregion aus dem Bild zur Buchregionserkennung (Abb. 6.4)

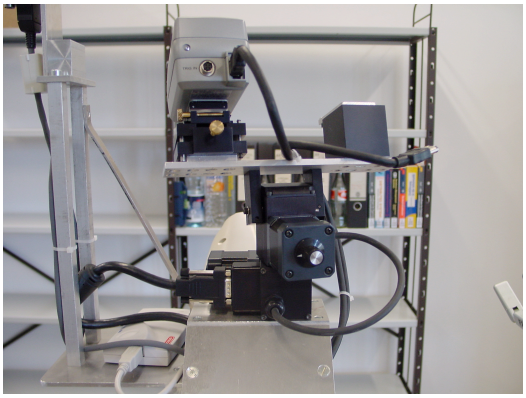


(b) (c) (d)

(e) (f) (g)

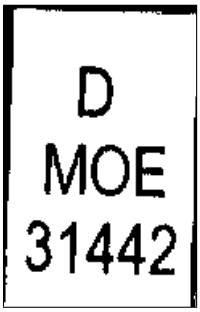
Abbildung 6.5: Bilder eines Schwenks der Kamera durch die PTU über eine Buchregion (a).

(a) Bild der automatisch geneigten PTU mit Kamera.



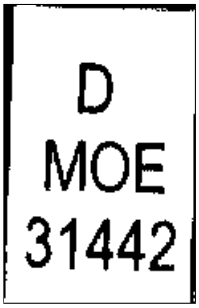
(b) Bilder der automatisch geneigten und gedrehten PTU mit Kamera

Abbildung 6.6: Bild der Neigung und Drehung der PTU. Die jeweiligen Winkel werden aus dem Bild zur Erkennung der Buchregionen errechnet.



(a) Unbeschnittenes Bild 1

(b) Beschnittenes Bild zu 6.7(a)



(c) Unbeschnittenes Bild 2

(d) Beschnittenes Bild zu 6.7(c)

Abbildung 6.7: Ein Beispiel für die vier Bilder, die bei der Extraktion der Signaturaufkleber erstellt werden. Nicht immer sind alle verwertbar.

6.7 Erkennung der Signaturen

Für die Erkennung der Signaturen wird zunächst ein Bild aufgenommen und anschließend in zwei Schwarz-Weiß-Bilder umgewandelt. Diese Umwandlung erfolgt durch einen einfachen Operator der Java-Advanced-Imaging-Bibliothek (JAI, [3]). Als am besten dafür geeignet haben sich die beiden Schwellwerte „AutoMaxVariance“ und „AutoMinFuzziness“ erwiesen, die von der Bibliothek wärend der Umwandlung automatisch berechnet werden. Beide werden benutzt, da die Ergebnisse mal bei dem einen, mal bei dem anderen besser waren.

Um die verschiedenen Signaturregionen zu erhalten, werden nun die sogenannten „Connected Components“ ermittelt (siehe dazu [9]) und die kleinsten Rechtecke (mit den Seiten

parallel zu den Bildseiten, sog. „Bounding Box“) errechnet, die alle Punkte einer Komponente enthalten. Diese Regionen werden aus dem original Farbbild kopiert und wieder in zwei Schwarz-Weiss-Bilder umgewandelt. Dieser Vorgang hat bessere OCR-Ergebnisse geliefert, als wenn die Regionen direkt aus den Schwarz-Weiss-Bildern herauskopiert werden. Diese Bilder werden an die OCR gesendet.

Zusätzlich werden bei den erhaltenen Schwarz-Weiss-Regionen die Randzeilen und -spalten entfernt, die über mehr als 2/3 schwarze Pixel enthalten. Die so gewonnen verkleinerten Regionen werden wieder aus dem Ursprungsbild kopiert, in zwei Schwarz-Weiss-Bilder umgewandelt und an die OCR gesendet.

Durch diesen mehrstufigen Vorgang wird die Trefferwahrscheinlichkeit erhöht.Oft kommt es vor, dass nur eins der vier (zwei beschnittene Bilder, zwei unbeschnittene, Abb. 6.7) erstellten Bilder von der OCR erkannt wird.

6.8 Vergleich der gefundenen Signaturen mit der gesuchten Signatur

Die von der OCR-Software erkannte Zeichenkette muss nun mit der Gesuchten verglichen werden. Dazu wurde die schon in Kapitel 5.4 erläuterte Levenshtein Distance benutzt. Distanzen, die kleiner als 4 sind, werden im Projekt als mögliche Treffer gewertet. Da die Distanz für jedes mögliche Ergebnis gespeichert wird, kann am Ende der Suche der Treffer mit der kleinsten und damit besten Distanz ausgegeben werden.

6.9 Präsentation der Ergebnisse

Die Präsentation der Ergebnisse kann auf zwei Arten erfolgen. Zunächst ist es möglich, den kompletten Suchvorgang des Roboters abzuwarten, um daraufhin den wahrscheinlichsten Treffer präsentiert zu bekommen. Gemeldet wird das Regal, in dem sich das Buch höchstwahrscheinlich befindet. Alternative Orte werden ebenfalls ausgegeben, falls mehrere ähnliche Signaturen gefunden wurden.

Die zweit Möglichkeit der Anzeige ist die direkte Präsentation der gefundenen Treffer. So kann der Benutzer mit dem Roboter interagieren, ihn überwachen und die Suche so möglicherweise verkürzen. Dabei stoppt der Roboter seine Arbeit, sobald er einen möglichen Treffer gefunden hat. Das Bild des Signaturaufklebers und die erkannte Signatur werden dem Benutzer als Ergebnis präsentiert und dieser kann entscheiden, ob das gesuchte Buch gefunden wurde oder der Roboter weitersuchen soll.

7 Zusammenfassung und Ausblick

Um ein vom Benutzer gewünschtes Buch zu finden, wird zunächst vom Roboter ein Regal angefahren, indem er dann nach Regionen sucht, in denen sich Bücher befinden können. Sind diese Regionen identifiziert, positioniert sich der Roboter mittig vor eine dieser Regionen. Die Signaturaufkleber werden durch die maximal gezoomte Kamera aufgenommen, die von der PTU über die Buchregion geschwenkt wird. Die erhaltenen Bilder werden nach Signaturaufklebern durchsucht und die Gefundenen durch eine OCR-Software erkannt. Wenn eine Signatur gefunden wurde, wird sie mit der Gesuchten verglichen. Sind die Signaturen gleich, so wird gespeichert, wo das Buch gefunden wurde, oder dem Benutzer die Möglichkeit gegeben, das gefundene Buch zu akzeptieren und damit die Suche abzubrechen. Wurde das Buch in der Region nicht gefunden, wird mit den nächsten Regionen fortgefahren. Sind alle Regionen aus einem Regal abgearbeitet, wird im nächsten Regal weitergesucht, bis kein Regal mehr vorhanden ist. Sind alle Regale durchsucht wird entweder gemeldet, wo das gesuchte Buch steht, oder dass das Buch nicht gefunden wurde.

Abschließend hat sich gezeigt, dass die Buchsuche für einen Roboter in natürlichen Umgebung lösbar ist. Die Suche dauert teilweise sehr lange und findet nur in wenigen Fällen ein gesuchtes Buch nicht, obwohl es im Regal vorhanden ist. Eine Entfernung des Autofokus-Algorithmus bringt hier sehr viel Geschwindigkeit, weil dieser am zeitaufwendigstens im ganzen Verfahren ist. Eine reelle Anwendung mit Hardware-Autofokus ist also möglich, da die restliche Verarbeitung schnell ausgeführt wird. Die Robustheit der Büchererkennung ist relativ gut. Zwar werden auch viele Gegenstände ausser Büchern erfasst, die Verarbeitung stört dies aber nicht. Die Umsetzung des in Matlab implementierten Systems in Java warf keine Probleme auf. Die Roblet®-Technologie ermöglichte eine einfache und leicht zu erlernende Programmierung des Roboters.

Verbesserungs- und Erweiterungsmöglichkeiten bieten sich hier viele. Zunächst wäre ein besserer, wenn möglich hardwaregestützter Autofokus interessant. Ein direkter Anschluss einer Autofokus-Hardware an den Firewire-Bus wäre hierbei denkbar. Alternativ könnte auch durch ein Stereo-Kamerasystem die Entfernung der Kamera vom Regal errechnet und dadurch ein Fokuswert bestimmt werden. Auch eine noch genauere Detektion der Buchreihen wäre wirkungsvoll. Es werden zwar viele Buchreihen erkannt, jedoch stellen einzelne Bücher ein Problem dar. Auch fehlt ein gutes Beurteilungskriterium, ob in einer gefundenen Buchreihe wirklich Bücher enthalten sind oder ob die gefundenen Kanten eventuell von anderen Gegenständen erzeugt wurden. Eine Idee hierfür wäre z.B. die Suche nach Signaturaufklebern in einer Buchreihe auf dem zuerst aufgenommenen Bild des Regals, ohne die Kamera zoomen zu lassen oder den Roboter zu bewegen. So

könnte der Suchraum erheblich verkleinert werden. Interessant und relativ leicht realisierbar wäre auch die Suche nach DVD's oder CD's, da die Grundstruktur den Büchern gleicht. Es müssten selbstverständlich lesbare Signaturen auf den Medien zu finden sein. Allgemein, allerdings auch in diesem Kontext, wäre auch eine Erkennung von Barcodes interessant, die sich durch neue Hardware (z.B. Laser) oder auch durch OCR erkennen lassen. Auf der Homepage der in diesem Projekt verwendeten OCR-Software GOCR findet sich ein Hinweis darauf, dass auch Barcodes mit der Software erkannt werden können. Ebenfalls wäre es denkbar, Bücher suchen zu lassen, die nicht aufrecht, sondern auch schräg im Regal stehen können. Z.B. könnte das Bild für die Signaturaufkleberer-kennung soweit gedreht werden, bis das entsprechende Buch im Bild vertikal steht. Die Signatur-Extraktion für die danach folgende OCR könnte eventuell mit anderen Verfahren verglichen und verbessert werden, um sie noch robuster zu machen. Ein spezialisierter Roboter könnte die Bücher vielleicht schneller und genauer absuchen. Hier wäre auch das Greifen des Buches denkbar. Das herausgegriffene Buch könnte dann entweder dem Benutzer gebracht werden, an einer Leihstelle bereitgestellt werden oder es könnten sogar gewünschte Seiten abfotografiert oder eingescannt werden, wie schon in einige Ansätzen in Kapitel 2 erwähnt. Die Erprobung von nicht spezialisierten Buch-Greifern würde sich anbieten.

Abkürzungsverzeichnis

AB TAMS	Arbeitsbereich Technische Aspekte Multimodaler Systeme am Fachbereich Informatik an der Universität Hamburg
IEEE	Institute of Electrical and Electronics Engineers - größter technischer Berufsverband von Ingenieuren aus Elektrotechnik und Informatik der Welt
OCR	Optical Character Recognition - Erkennung von Buchstaben und Zeichen in Bildern durch ein Computerprogramm
PTU	Pan-Tilt-Unit - Schwenk-Neige-Einheit, mit der z.B. eine Kamera gedreht und geneigt werden kann
VGA	Video Graphics Array - Ein Standard, der Auflösungen bis 640x480 Bildpunkten erlaubt

Literaturverzeichnis

[1] *D. Westhoff, J. Zhang, H. Stanek, T. Scherer, A. Knoll:* Mobile Manipulatoren und ihre aufgabenorientierte Programmierung. atp - Automatisierungstechnische Praxis 10/2004, Oldenbourg Industrieverlag GmbH, Munich, Germany. ISSN 0178-2320
http://www.oldenbourg.de/verlag/at-technik/rot-atp1.html
http://www.roblet.org

[2] *Sun Microsystems, Inc.:* Java Technology,
http://java.sun.com

[3] *Sun Microsystems, Inc.:* Java Advanced Imaging (JAI) API,
http://java.sun.com/products/java-media/jai/t

[4] *J. F. Canny:* Finding edges and lines in images. Technical Report 720, MIT Artificial Intelligence Laboratory, Dept. of Electrical Engineering and Computer Science, Cambridge, MA, 1983.

[5] *Y. Xiong , S. Shafer:* Depth from Focusing and Defocusing. Technical report CMU-RI-TR-93-07, Robotics Institute, Carnegie Mellon University, March, 1993

[6] *V. I. Levenshtein:* Binary codes capable of correcting deletions, insertions, and reversals, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966.10, 1966.

[7] Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, "Levenshtein distance", from Dictionary of Algorithms and Data Structures, Paul E. Black, ed., NIST.
http://www.nist.gov/dads/HTML/Levenshtein.html

[8] *E. Krotov:* Active Computer Vision by Cooperative Focus and Stereo, Berlin, Springer Verlag, 1989

[9] *M. Jankowski, J.-P. Kuska:* Connected components labeling - algorithms in Mathematica, Java, C++ and C-sharp. 2004, In: P. Mitic, Ch. Jacob, J. Crane (Eds.) New Ideas in Symbolic Computation: Proceedings of the 6th Mathematica Symposium, Positive Corporation Ltd. Hampshire, UK, 2004. ISBN 0-9547810-0-7

[10] *R. Ramos-Garijo, M. Prats, P.J. Sanz, A.P. Del Pobil:* An Autonomous Assistant Robot for Book Manipulation in a Library. In proc. of IEEE International Conference on Systems, Man & Cybernetics (ISBN 0-7803-7953-5), pp. 3912-3917, Washington D.C., USA, 2003.

[11] *M. Prats, R. Ramos-Garijo,P.J. Sanz, A.P. del Pobil:* Autonomous Localization and Extraction of Books in a Library. Intelligent Autonomous Systems, edited by F. Groen et al., IOS Press. pp. 1138-1145 (ISBN 1-58603-414-6). Amsterdam, 2004.

[12] *M. Prats, R. Ramos-Garijo, P.J. Sanz, A.P. Del Pobil:* Recent Progress in the UJI Librarian Robot. In Proc. of IEEE International Conference on Systems, Man & Cybernetics (ISBN 0-7803-8566-7), 2004.

[13] *J. Suthakorn, S. Lee, Y. Zhou, R. Thomas, G. S. Choudhury, G. S. Chirikjian:* A Robotic Library System for an Off-Site Shelving Facility. Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA), 2002

[14] *T. Tomizawa, A. Ohya, S. Yuta:* Book Browsing System Using an Autonomous Mobile Robot Teleoperated via the Internet. Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2002), 2002

[15] *T. Tomizawa, A. Ohya, S. Yuta:* Object Posture Recognition for Remote Book Browsing Robot System. Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2003), 2003

[16] Weitere Informationen zum Service-Roboter der Universität Hamburg
http://tams-www.informatik.uni-hamburg.de/personal
/westhoff/de/robotik/robotik.html

[17] NEOBOTIX (GPS GmbH Stuttgart)
http://www.neobotix.de

[18] SICK Laserscanner
http://www.sick.de

[19] Sony DFW-SX900
http://www.sony.net/Products/ISP/products/interface/DFWSX.html

[20] Panorama Eye, hyperbolische Spiegel
http://www.accowle.com/english/index.html

[21] PTU, Directed Perceptions
http://www.dperception.com/products.html

[22] Sony DFW-VL500 Kamera
http://www.sony.net/Products/ISP/products/interface/DFWV500.html

[23] Roboterarm Mitsubishi Heavy Industries PA10-6C
http://www.robot-arm.com

[24] Dreifinger-Roboterhand der Firma Barrett Technologies, Inc.
http://www.barretttechnology.com/robot/index.htm

[25] JAI - The Mechademic Company, Handkamera
http://www.jai.com

[26] GOCR - Open Source Optical Character Recognition
http://www.gocr.de oder http://jocr.sourceforge.net