Graz University of Technology

Institute for Computer Graphics and Vision

Masterarbeit

Objekterkennung mit dem Autonomen Serviceroboter FLEA

Joachim Pehserl, BSc.

0230095

Graz, Austria

Juli 2011

Begutachter

Univ.-Prof., Dipl.-Ing. Dr.techn. Horst Bischof

To My Robot ;) ...

Es ist nicht genug zu wissen - man muss auch anwenden. Es ist nicht genug zu wollen - man muss auch tun.

Johann Wolfgang von Goethe

Statutory Declaration

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

Eidesstattliche Erklärung

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

Ort	Datum	Unterschrift

Abstract

The focus of this thesis is on the design and implementation of an object recognition system for the autonomous mobile service robot, FLEA. Focusing on partial planar textured objects, the robot is able to learn robust object models based on stable local features. Those features are obtained by capturing multiple views of the object with a stereo camera system. The evaluation of different feature extraction methods shows that the CenSurE interest point detector combined with a SURF descriptor ensures realtime capability while maintaining a good enough recognition rate. To handle larger object databases, we demonstrate that random KD-trees are the prefered choice to ensure realtime computation while mainting a high enough precision to successfully match objects. Furthermore, it is shown that depth information obtained from a stereo camera can be used to successfully boost the overall robustness of the system.

The proposed object recognition system is optimized for solving the RoboCup@Home Lost and Found Task. We perform Lost and Found tests in a typical home environment. Pre-learned objects are hidden in a home environment and found by the robot within a given amount of time. By combining multiple views and depth information from a stereo camera, the robot is able to successfully find objects with a significantly low false-positive rate. The implemented object recognition system demonstrates the ability to solve the Lost and Found task. Moreover, it makes use of multicore computer architectures and allows flexibility for future extensions.

Kurzfassung

Der Fokus dieser Arbeit lag im Design und der Implementierung eines bildbasierten Objekt-Erkennungssystems für den autonomen mobilen Roboter FLEA. Das System wurde für den RoboCup@Home Lost and Found Task optimiert. Ziel der Arbeit war es, stückweise planare, texturierte Objekte mit Hilfe von lokalen Feature Methoden zu lernen und robust wieder zu finden. Teil der Aufgabe war die Einbindung von verschiedenen Ansichten des Objekts während der Lernphase als auch die Tiefeninformation einer Stereokamera.

Die Evaluierung der einzelnen Methoden zeigte, dass CenSurE Keypoints mit dem SURF Deskriptor die Echtzeitfähigkeit des Systems garantieren. Die Erkennungsrate war dabei völlig ausreichend. Die Einbindung von Randomized KD-Trees anstatt Bruteforce Matching ermöglichte dem System auf großen Objektdatenbanken zu arbeiten. Die Verwendung dieser Datenstruktur ermöglichte die Implementierung eines echtzeitfähigen Systems, wobei durch die Beschleunigung des Matchings nur marginal Präzision verloren ging.

Das implementierte Objekt-Erkennungs-System wurde Tests in einer echten Haushaltsumgebung unterzogen. Das System war in der Lage vorher gelernte Objekte, die in der Umgebung versteckt waren, robust wieder zu finden. Durch die Ausfilterung instabiler Features in der Lernphase und die Einbeziehung von Tiefeninformation kam es zu sehr wenigen falsch positiven Erkennungen.

In dieser Diplomarbeit wurde gezeigt, dass das implementierte Objekt-Erkennungssystem in Verbindung mit dem Roboter-System den RoboCup@Home Lost and Found Test in einer echten Haushaltsumgebung zuverlässig lösen kann. Die Systemarchitektur wurde für Multicore-Computer optimiert und ist erweiterbar.

Keywords. FLEA, object recognition, RoboCup@Home, Lost and Found, CenSurE, SURF, RKD-Trees, MCL, stereo, multiple views, local features, matching

Danksagung

Ich möchte an dieser Stelle allen Personen danken, die mich im Laufe dieser Arbeit beraten, finanziell unterstützt und motiviert haben! Vor allem danke ich all denjenigen, die mich bei der Entwicklung des Roboters FLEA über die vergangenen fünf Jahre aktiv unterstützt haben! Speziell hervorheben möchte ich DI Petra Korica-Pehserl, Rudolf Rösler und meine Eltern! Auch möchte ich mich bei Dr. DI Michael Grabner für die informativen Diskussionen und insbesondere bei Prof. Dr. DI Bischof für die langjährige Projekt-Unterstützung bedanken!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Motivation

Haushaltsroboter erfreuen sich immer größerer Beliebtheit. Im Moment sind diese Roboter zwar nur autonome Staubsauger oder Rasenmäher, doch in Zukunft könnten bald auch schon ganz andere Geräte durch unsere Wohnungen fahren. Roboter, die nicht nur in der Lage sind mit Hilfe von Zufallsmustern einen Raum abzufahren, sondern zu jeder Zeit exakt wissen, wo sie sich befinden, wo ihr Ziel ist und wie sie am effizientesten dorthin gelangen sind aus technologischer Sicht soweit gereift, um in Haushaltsumgebungen eingesetzt werden zu können. Dies schafft die Grundvoraussetzung für mobile Robotik im Haushalt.

Die Fähigkeit autonom von A nach B fahren zu können ohne die Inneneinrichtung zu zerstören ist ausreichend für Roboter-Staubsauger. Soll ein Roboter jedoch mehr können, führt kein Weg am maschinellen Sehen vorbei. Komplexere Aufgaben verlangen ein Verständnis für die Umwelt in der sich der Roboter bewegt.

Ein moderner Haushaltsroboter muss zumindest einige Personen oder Haushalts-Objekte wiedererkennen können. Dies ist die Voraussetzung für komplexere Aufgaben, wie zum Beispiel ein Objekt zu greifen und es zu einer bestimmten Person zu bringen. Doch um Objekte manipulieren zu können, muss der Roboter diese in der Umgebung wiederfinden. Dabei ist entscheidend, dass ein Roboter das richtige Objekt ausfindig macht. Bei der RoboCup@Home Weltmeisterschaft 2009 in Graz wurde genau dieser Aspekt zufällig im Finale beleuchtet: ein Roboter sollte eine Bierdose vom Esstisch zum im Wohnzimmer auf der Couch sitzenden Juror bringen. Der Roboter setzte sich Richtung Esstisch in Bewegung, erkannte ein zylindrisches Objekt, ergriff es und brachte es dem Juror. Leider handelte es sich bei dem Objekt nicht um die zu bringende Bierdose, sondern um eine leere Dose Kartoffelchips. Der Roboter war also nicht in der Lage eine

Bierdose von Kartoffelchips zu unterscheiden. Wie es richtig geht, zeigt die Abbildung 1.1.

Diese Diplomarbeit beschäftigt sich mit der Problematik stückweise planare, texturierte Haushaltsobjekte robust mit dem selbst entwickelten Roboter FLEA zu erkennen. Diese Arbeit soll das Fundament für komplexe Aufgaben wie Objekt-Manipulation schaffen. Das genannte Beispiel verdeutlicht klar die Vision: es reicht nicht aus beliebige Objekte, die sich von der Tischplatte abheben und eine bestimmte Form haben, greifen zu können. Vielmehr ist es notwendig Objekte genau zu betrachten und diese mit einer Datenbank von bekannten Objekten abzugleichen um das richtige Objekt zu selektieren. Ein solcher Vergleich ist mit Hilfe von maschinellem Sehen möglich.

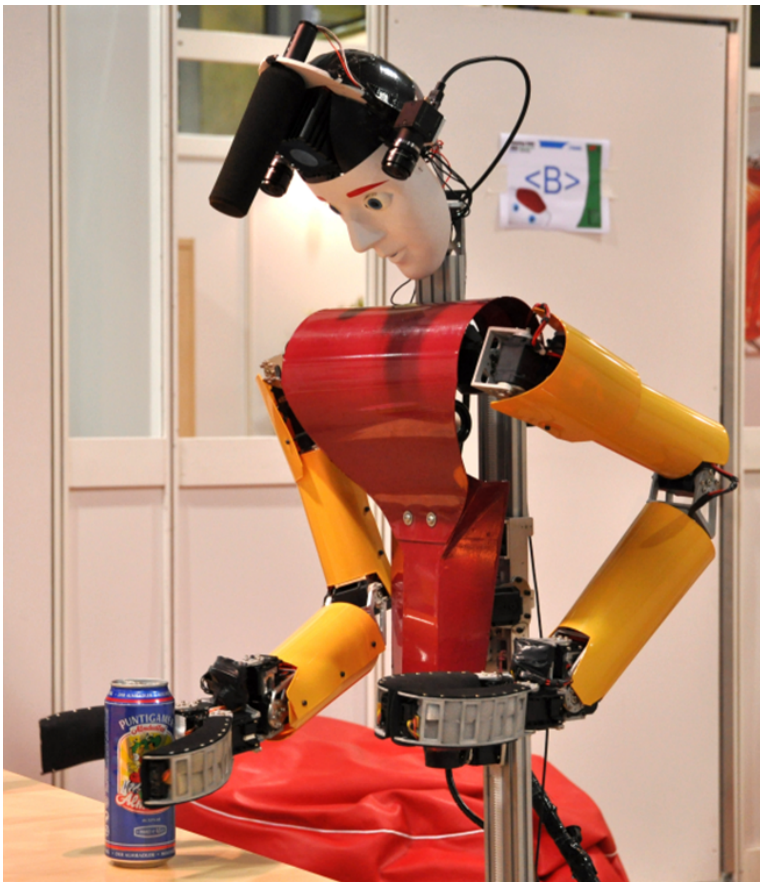

Abbildung 1.1: Ein Roboter der Objektmanipulation bei der RoboCup@Home Weltmeisterschaft 2009 betreibt. Für das Greifen ist Objekterkennung unumgänglich.[23]

1.1 Ziel der Arbeit

Diese Arbeit beschäftigt sich mit der Erkennung von Objekten in einer Haushaltsumgebung mit dem autonomen Roboter FLEA. Abbildung 1.2 zeigt den Roboter FLEA wie er zum Zeitpunkt der Fertigstellung der Diplomarbeit aussieht. Haushaltstypische, stückweise planare Objekte sollen innerhalb kurzer Zeit vom Roboter erlernt werden können. Ziel der Arbeit ist es, die gelernten Objekte in der Haushaltsumgebung robust wiederzufinden. Hierzu werden die Objekte von einem Menschen in der Wohnung platziert. Die Einschränkungen sind, dass die Objekte für den Roboter „fair“ positioniert sind (also über das Kamerasystem wahrnehmbar) und das die verwendeten Objekte stückweise planar und texturiert sein müssen. Typischerweise handelt es sich bei den Objekten um Supermarkt Artikel, wie sie bei den RoboCup@Home Weltmeisterschaften verwendet werden (Reispackung, Babynahrung, etc.). Das zu implementierende Objekterkennungssystem soll ohne CAD Modelle auskommen und die nötige geometrische Information mit Hilfe von Stereoskopie selbst extrahieren, wobei die Tiefeninformation zur Robustheitserhöhung bei der Erkennung eingesetzt werden soll. Es ist nicht Ziel der Arbeit vollständige 3D Modelle automatisch abzuleiten oder 3D Rekonstruktion zu betreiben, sondern vielmehr die Entwicklung eines robusten und echtzeitfähigen Objekterkennungssystems, welches im Wettbewerb eingesetzt werden kann. Es soll untersucht werden, ob mehrere leicht unterschiedliche frontale Ansichten eines Objekts und Stereo-Information die Robustheit der Objekterkennung verbessern können.

1.2 Aufbau der Arbeit

Die Diplomarbeit gibt in Kapitel 2 einen Überblick über die in Wettbewerbs-Haushaltsrobotern aktuell eingesetzte Hardware und Software. Es wird gezeigt, welche Technologien die Roboter an Bord haben und mit welchen Methoden sie Objekterkennung betreiben. In weiterer Folge wird tiefer in den Bereich der Objekterkennung mit lokalen Features Einblick gegeben. Es werden Feature Detektoren, Deskriptoren und Matcher beschrieben die heute in der mobilen Robotik und Computer Vision ihren Einsatz finden. Die Kapitel 3 und 4 befassen sich mit der Umsetzung der visuellen Suche am Roboter. Es wird gezeigt, welche Robotermodule und welche Hardwarekomponenten von FLEA an der Suche beteiligt sind. Es wird beschrieben wie das Objekterkennungssystem funktioniert. In Kapitel 5 werden die vorher beschriebenen Objekterkennungsmethoden auf ihre Einsatzfähigkeit am Roboter FLEA evaluiert. Die Evaluierung zeigt, welche

Abbildung 1.2: Roboter FLEA, Entwickler-Version 2011 (ohne Chassis)

Auswirkungen mehrere Objekt-Ansichten beim Lernen eines Objekts auf die Robustheit der Wiedererkennung haben. In weiterer Folge wird gezeigt wie die Einbindung von Tiefeninformation die Robustheit des Systems beeinflusst. Es werden Experimente zur visuellen Suche beschrieben, welche in einer typischen Haushaltsumgebung durchgeführt werden.

Kapitel 2

State of the Art

Seit einigen Jahren finden regelmäßig Wettbewerbe statt, bei denen Universitäten und Firmen die Fähigkeiten ihrer Roboter unter Beweis stellen können. Einer der größten und bekanntesten Wettbewerbe ist der RoboCup. Beim RoboCup stellen jedes Jahr Teams aus aller Welt die Fähigkeiten ihrer Roboter zur Schau, in dem sie diese in verschiedensten Wettbewerben gegeneinander antreten lassen. Die bekanntesten Ligen im RoboCup beschäftigen sich mit Fußball, daneben gibt es aber noch andere große fußballfremde Wettbewerbe wie zum Beispiel die Rescue-, Simulation- oder die @Home-Liga. Speziell für Haushaltsroboter ist die RoboCup@Home Liga relevant. Sie wies bei der RoboCup Weltmeisterschaft 2009 die höchste Teilnehmerzahl (antretende Teams) neben Fußball auf. Dies zeigt den globalen Trend in Richtung Servicerobotik.

Die RoboCup@Home Liga hat das Ziel die Service- und Assistenz-Robotertechnologie mit hoher Relevanz für zukünftige Haushaltsanwendungen weiterzuentwickeln und zu fördern. Der RoboCup ist der größte, jährlich stattfindende internationale Wettbewerb für autonome Serviceroboter und ist Teil der RoboCup Föderation. Mit Hilfe verschiedener Benchmark-Tests werden die Fähigkeiten und Leistungen der Roboter in realistischen, nicht standardisierten Haushaltsumgebungen evaluiert. Derzeit liegt der Fokus des Wettbewerbs auf folgenden Teilbereichen: Mensch-Roboter-Interaktion und Kooperation, Navigation und Mapping in dynamischen Umgebungen, Computer Vision und Objekterkennung unter natürlichem Licht, Objekt Manipulation, adaptives Verhalten und Verhaltens-Integration, Umgebungs-Intelligenz, Standardisierung und Systemintegration. [6]

Somit deckt RoboCup@Home fast die komplette Bandbreite der praxisrelevanten Haushaltsrobotik ab. Die Einbindung von maschinellem Sehen erfolgt bei RoboCup@Home derzeit hauptsächlich bei den Tests Objekterkennung („Lost and Found“ Test) und Personen-

bzw. Gesichtserkennung („Who is Who“ Test). Kein am RoboCup teilnehmendes Team ist verpflichtet seinen Quellcode zu veröffentlichen. Im Gegenzug dazu gibt es einen weiteren internationalen Wettbewerb, der sich hauptsächlich auf Computer Vision und Objekterkennung beschränkt: die „Semantic Robot Vision Challenge“ (SRVC). Bei diesem Wettbewerb ist die Veröffentlichung des Quellcodes erforderlich.

Der Fokus der SRVC liegt in der Förderung der Entwicklung des Bildverstehens und automatischer Wissensgewinnung unter Zuhilfenahme großer, unstrukturierter Internet-Bilddatenbanken. Bei der SRVC steht die Fusion von Robotik und Computer Vision im Vordergrund: die Roboter bekommen eine Liste von Objekten übergeben, die sie anschließend in einer speziellen Umgebung finden müssen. Dabei ist entscheidend, dass die Objekte zuvor nicht physikalisch verfügbar sind (im Gegensatz zu RoboCup@Home), sondern ausschließlich Bilder aus dem Internet zum Training herangezogen werden dürfen. [20]

Aus historischen Gründen wird in dieser Arbeit auf die Anforderungen von RoboCup@Home eingangen, da der selbst entwickelte Roboter FLEA ursprünglich für diesen Wettbewerb konzipiert wurde.

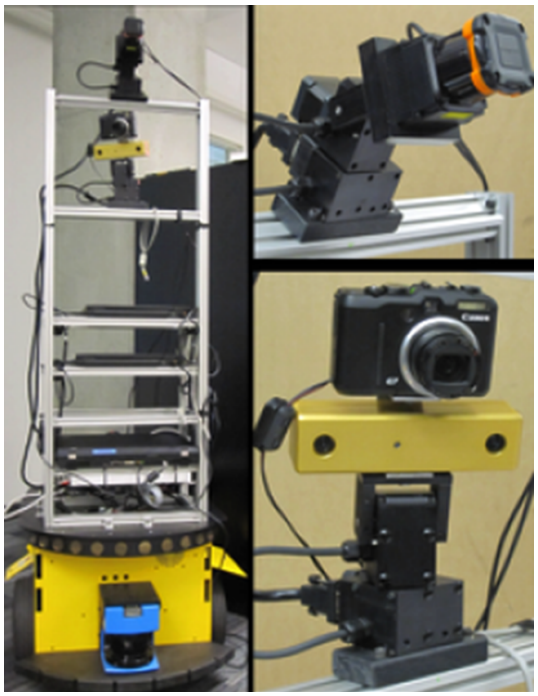

Abbildung 2.1: University of British Columbia - Curious George Hardware [15]

2.1 Roboter Systeme

In diesem Unterkapitel werden die aktuell besten Robotersysteme beider Wettbewerbe vorgestellt. Nachdem bei RoboCup@Home durchschnittlich über 20 Teams antreten, wurden drei Finalisten der letzten beiden Veranstaltungsjahre ausgewählt. Die Semantic Robot Vision Challenge ist nicht so bekannt wie RoboCup@Home und dementsprechend nehmen nicht so viele Teams an der Veranstaltung teil. In diesem Unterkapitel werden aktuelle Wettbewerbs-Robotersysteme beschrieben, die entweder bei der SRVC oder bei RoboCup@Home Weltmeisterschaft ins Finale gekommen sind.

2.1.1 UBC - Curious George

Curious George ist ein Roboter der University of British Columbia (UBC). Er wurde entwickelt um an dem SRVC Wettbewerb teilzunehmen. Mittlerweile befindet sich Curious George in der 3. Generation und konnte in den letzten Jahren der SRVC jeweils den ersten Platz erreichen. Der Roboter ist in der Lage Objekt-Lokalisierung in einer natürlichen Haushaltsumgebung durchzuführen. Dies beinhaltet das autonome Lernen von Klassifikatoren mit Hilfe von Internetbildern, Navigation (Exploration) und die Erkennung von Objekten mittels der zuvor gelernten Klassifikatoren in einer Haushalts-Umgebung. Der Roboter bedient sich vielerlei Software und Hardware um diese Aufgabe zu lösen. Ein Sensorsystem bestehend aus einer Stereokamera und einer digitalen Kompaktkamera, montiert auf einer Pan/Tilt Einheit, ermöglicht es dem Roboter Tiefeninformation mit hochauflösenden Bildern zu verknüpfen. Zusätzlich ist Curious George mit einem kleinen Laserscanner ausgestattet, der auf einer Tilt Einheit sitzt, um möglichst dichte und genaue 3D Informationsdaten zu bekommen. Wie bei der SRVC üblich bedient sich der Roboter an Internet Bilddatenbanken, um die entsprechenden Objekt-Klassifikatoren autonom zu trainieren.

Die mobile Basis-Plattform von Curious George bildet die industrielle Roboterplattform „Powerbot“ der Firma Mobile Robots Inc. Wie bei den meisten derzeit im Einsatz befindlichen Robotern üblich sitzt ein Sick Laser Range Finder (Sick LRF) auf dieser mobilen Basiseinheit. Der Sick LRF dient der Lokalisierung und Navigation. Sechs Computer werden benötigt, um Curious George für die SRVC zu betreiben. Während der SRVC laufen darauf 50 unabhängige Prozesse simultan. Eine Bumblebee Stereokamera von Pointgrey und eine Canon G7 Kompaktkamera liefern die Bilder. Zusätzlich rundet ein lichtgewichtiger Hokuyo Laserscanner auf einer Tilt-Einheit die Sensorfront ab. Die Abbildung 2.1 zeigt die Hardwarekomponenten von Curious George. [15]

Die Kernkomponente für die verteilte Berechnung und Nachrichtenverteilung bildet das "Robot Operating System". ROS ist Open Source und setzt auf einem gewöhnlichen Betriebssystem als sogenanntes Meta-Betriebssystem (Framework) auf. Hardware-Abstraktion, Low-Level Gerätekontrolle, Robotik-Bibliotheken und Nachrichtenverteilung sind integrale Bestandteile von ROS, das seit 2007 entwickelt wird und seinen Ursprung an der Stanford University hat. [19]

Curious George benutzt unterschiedliche Internet-Bilddatenbanken als Quelle für den Aufbau der Klassifikatoren. Dabei gibt es eine Trennung in zwei verschiedene Klassen von Datenbanken. Die erste Klasse stellen annotierte Computer Vision Bilddatenbanken dar, wie zum Beispiel LabelMe und ImageNet. Diese weisen wenig Unsicherheit auf, da diese Datenbanken von Menschen für den Zweck des maschinellen Sehens aufgebaut wurden. Die zweite Datenbankklasse beinhaltet eine hohe Unsicherheit und keine oder nur eine schlechte Annotierung der Bilder. Beispiele sind die Google Bildsuche oder die Produktdatenbank von Walmart. Die visuelle Klassifikation wird in Curious George mit Hilfe von drei verschiedenen Techniken umgesetzt: SIFT, Contour Matching und Deformable Parts Model. Entscheidend ist, dass die drei unterschiedlichen Klassifikatoren nicht kombiniert werden, sondern es wird beim Training für eine Objektkategorie der Klassifikator ausgewählt, der dafür die höchste Erkennungsrate aufweist. Curious George verwendet keine unterschiedlichen Objekt-Ansichten für das Training des Klassifikators. [15]

ROS ermöglicht Curious George mittels Laser basiertem SLAM (GMapping) eine geometrische 2D Karte der Umgebung zu erstellen, welche in der Robotik als „Occupancy Grid“ bezeichnet wird. Zusätzlich bietet ROS einen Laser basierten Tisch- und Sessel-Detektor an, mit dessen Hilfe die 2D Karte mit Zusatzinformationen über diese Gegenstände erweitert wird. Der Roboter benutzt Exploration um zuvor ungesehene Bereiche in der Umgebung einzusehen. Dabei merkt er sich die bereits angesehenen Bereiche, um eine Mehrfachabtastung zu verhindern. In weiterer Folge verwendet der Roboter „Visual Attention“ Methoden um möglichst schnell uninteressante Bilder oder Bildbereiche verwerfen zu können. Interessante Objekte werden bei dieser Implementierung visuell und über die 3D-Struktur (Größe, Kontext) erkannt. [15]

Bei der SRVC 2009 konnte Curious George 13 von 20 Objekten erkennen. Die Abbildung 2.2 zeigt die Umgebung in der die Objekte versteckt wurden.

Abbildung 2.2: Wettbewerbsumgebung bei der SRVC Zitat Curios George Paper [22]

2.1.2 Nimbro - Dynamaid

Dynamaid wurde für die Verwendung bei RoboCup@Home konzipiert und stellt einen der leichtesten Roboter dar, der jemals bei RoboCup@home teilgenommen hat. Emöglicht wird dies durch die Leichtbauweise und Robotis Dynamixel Aktuatoren. Das Fahrwerk besteht aus Dynamixel EX-106 Aktuatoren, die jedes Rad in eine beliebige Rotation und Geschwindigkeit versetzen können. Somit können omnidirektionale Bewegungen ausgeführt werden, die besonders bei Manipulationsaufgaben hilfreich sind. Ein auf der Plattform montierter Sick LRF liefert Daten für Navigation und Lokalisierung. Eine Besonderheit von Dynamaid ist die variable vertikale Basis, die es erlaubt die Höhe des Oberkörpers des Roboters um bis zu einem Meter zu verändern. Somit kann der Roboter Gegenstände vom Boden aufgreifen oder Objekte auf einem bis zu 100cm hohen Tisch manipulieren. Bei einer Gesamthöhe von 180cm im ausgefahrenen Zustand kann er auch Personen direkt in die Augen schauen. Ein am Torso montierter Hokuyo URG-04LX Laser Scanner auf einer Roll-Einheit wird für die Objekterkennung verwendet. Im Kopf befinden sich eine Time-of-Flight (ToF) 3D Kamera von Swissranger und ein Pointgrey Stereokamera-Paar. Im Dynamaid ist das Robot Operating System für die Inter-Prozess Kommunikation verantwortlich. Zur Lokalisierung wird Monte Carlo Lokalisierung, für SLAM wird GMapping verwendet. [23]

Für die Objekterkennung verwendet Dynamaid einen kombinieren Ansatz aus ToF Tiefeninformation und Bildinformation. Mit Hilfe der Tiefeninformation werden Oberflächen

extrahiert auf der sich Objekte befinden können. Danach werden 3D Punkte, die nicht zur Oberfläche gehören zu Objekten gruppiert und somit segmentiert. Auf dieser Objekt-Segmentierung werden geometrische Primitive extrahiert (Sphäre, Zylinder, Ebene). Diese erkannten Primitive werden in den Bildraum projiziert und in den entsprechenden Bereichen werden Farbhistogramme und SURF Features extrahiert. Diese Features werden mit einer zuvor erstellten Objektdatenbank abgeglichen. Anzumerken ist, dass auch Dynamaid unterschiedliche Ansichten eines Objekts während des Trainings unterstützt, um die spätere Erkennung robuster zu machen. [23] In der Abbildung 2.3 ist der Objekterkennungsvorgang bildlich dargestellt.

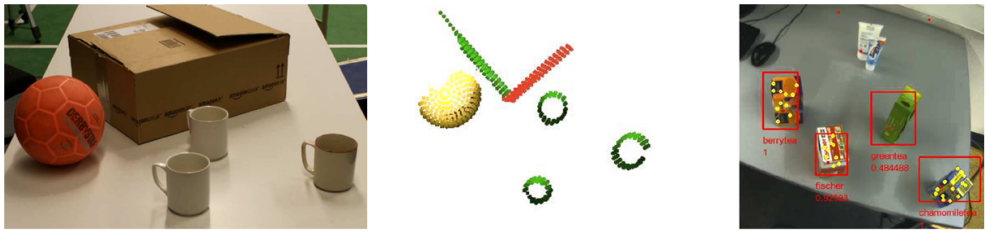

Abbildung 2.3: Dynamaid Objekterkennung. Links: Szene, Mitte: Segmentierung der Tiefeninformation in geometrische Primitive, Rechts: Primitive rückprojiziert ins Bild mit extrahierten SURF Features Nimbro Dynamaid TDP [23]

Abbildung 2.4: Nimbro Dynamaid, Nimbro Dynamaid TDP [23]

2.1.3 Homer - Lisa

Lisa heißt der Roboter des RoboCup@Home Teams der Universität Koblenz. Der Roboter wurde ebenfalls für die Verwendung bei RoboCup@Home konzipiert. Die Basis bildet die Pioneer3-AT Plattform von Mobile Robots Inc. Der Roboter ist mit einem SICK LRF ausgestattet, der wie üblich in Bodennähe montiert ist und für Lokalsierung und Mapping

verwendet wird. Ein Hokuyo Laserscanner ist auf einer Pan/Tilt Einheit montiert. Eine Philips Webcam ist rigid mit dem Hokuyo Laserscanner verbunden und dient der Gesichtserkennung. Eine ToF-Kamera (MESA Swissranger) befindet sich zusätzlich auf der Pan/Tilt Einheit. Für Objekterkennung werden SURF Features vom Kamerabild extrahiert und gegen eine Datenbank verglichen. Das Ergebnis wird mit Hilfe einer Homographie von Ausreißern bereinigt. [9]

Abbildung 2.5: Lisa vom Team Homer der Universität Koblenz Zitat: Homer 2010 TDP [9]

2.1.4 B-it-bots - Johnny Jackanapes

Johnny Jackanapes, vom Team b-it-bots, wurde für die Verwendung bei RoboCup@Home konzipiert. Der Roboter basiert auf der Volksbot Plattform, die vom Fraunhofer Institut für Intelligente Analyse- und Informationssysteme wurde. Abbildung 2.7 zeigt den Roboter, welcher mit einem Manipulator der Firma Neuronics ausgestattet ist. Mit dessen Hilfe lassen sich Gegenstände bis zu einem Gewicht von 500g manipulieren. Ein SICK Laser Range Finder und eine Bumblebee Stereokamera auf einer Pan/Tilt Einheit befinden sich ebenfalls an Bord. Für den Informationsaustausch der einzelnen autonomen Prozesse (Navigation, Lokalisierung, Vision) sorgt die ICE Middleware. ICE stellt ein Framework für die netzwerkbasierte Kommunikation dar. Die Objekterkennung erfolgt bei Johnny Jackanapes in einem zwei Schritte Verfahren. Zuerst wird eine Farbsegmentierung durchgeführt.

Hierbei wird eine Segmentierung der dominanten Farben mit Hilfe des YUV Farbraums vorgenommen. Danach werden SIFT Features auf den segmentierten Bildbereichen extrahiert und mit Features in einer zuvor erstellten Objektdatenbank verglichen. Abbildung 2.6 veranschaulicht diesen Ansatz anhand einer Ketchup Flasche. [5]

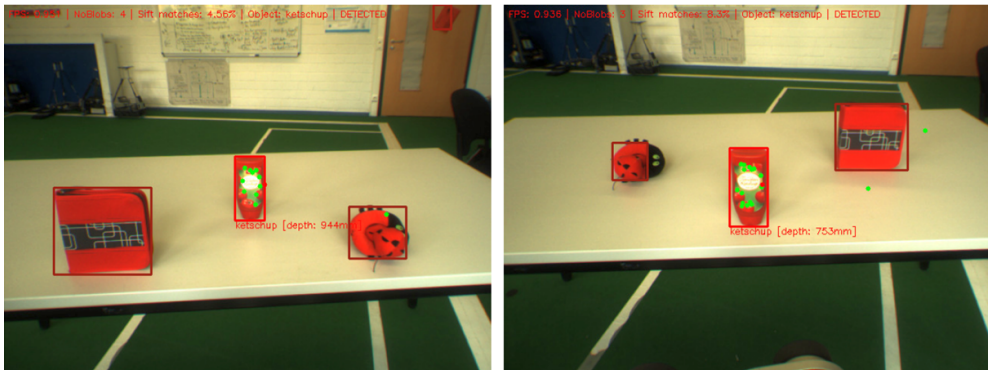

Abbildung 2.6: SIFT Features werden auf farbsegmentierten Bereichen extrahiert. b-it-bots 2010 [5]

Abbildung 2.7: b-it-bots Johnny Jackanapes [5]

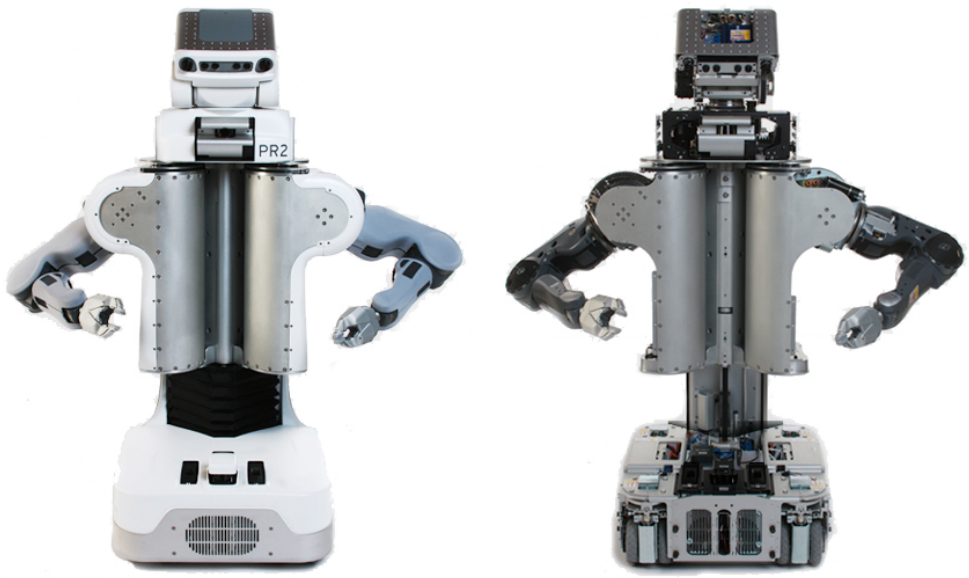

Abbildung 2.8: Willow Garage PR2. Links: Ansicht mit Chassis. Rechts: Einblick in die Innereien des Roboters.[27]

2.1.5 Willow Garage - PR2

Obwohl der Willow Garage PR2 Roboter nicht an den RoboCup oder SRVC Meisterschaften teilgenommen hat, soll er dennoch in dieser Arbeit vorgestellt werden. Der PR2 wurde als Forschungsroboter für Wissenschaftler konzipiert und ist für die Verwendung in Haushaltsumgebungen optimiert. Im Vergleich zu den vorher vorgestellten Robotern stellt er den Gold-Standard in der mobilen Haushalts-Robotik dar. Der PR2 wird in einer kleinen Serie hergestellt und unterscheidet sich somit maßgeblich von den bisher vorgestellten Robotern. Die Abbildung 2.8 zeigt den PR2 mit und ohne Chassis.

Der PR2 verfügt über Standard PC Hardware und ist somit frei programmierbar. Der PR2 beherrbergt zwei QuadCore Recheneinheiten mit je 24GB RAM und insgesamt 2TB Festplattenkapazität. Eine 1.3kWh Batterie kann den Roboter für 2 Stunden mit Energie versorgen. Ein omnidirektionales Fahrwerk und eine vertikale variable Basis bieten dem Roboter eine enorme Bewegungsfreiheit (siehe Abbildung 2.10). Zwei Manipulatoren ermöglichen die Interaktion mit der Umgebung. Die Sensorik umfasst zwei Stereo Farb-Kameras, wobei eine Weitwinkel-Objektive hat. Weiters ist ein Textur-Projektor und eine 5 Megapixel Kamera im Kopf integriert. Alle Kameras sind Global Shutter und somit hervorragend für Robotik Anwendungen geeignet. In den Greifern befinden sich jeweils eine Kamera, in der Basis ist ein Laserscanner für Navigation, im Torso ein schwenkender

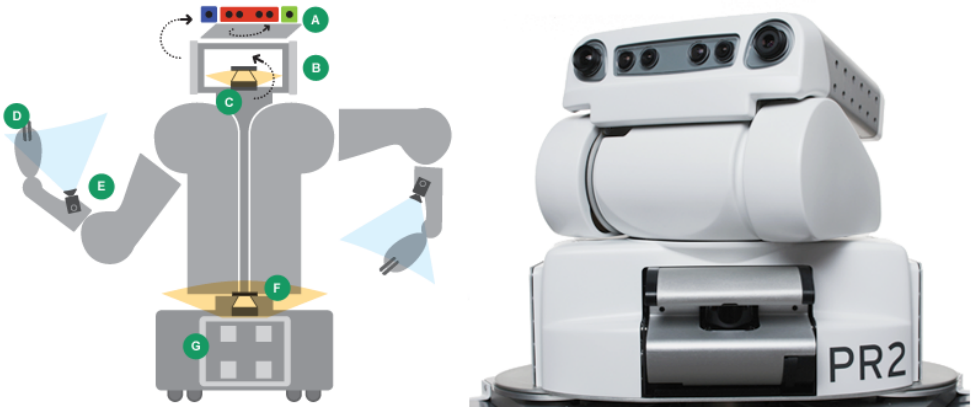

Abbildung 2.9: Willow Garage PR2 Sensoren. (A) Stereo Kameras, Textur-Projektor und 5MP Kamera, (B) Pan/Tilt Einheit, (C) schwenkender Laserscanner, (D) Greifer, (E) Greifer Kamera, (F) horizontaler Laserscanner, (G) Computer. Rechts: Detailaufnahme des Kopfes (A, B und C). [27]

Laserscanner für Objekterkennung und Navigation. Abbildung 2.9 zeigt eine abstrakte Übersicht der Hauptsensoren als auch eine Detailaufnahme des Kopfes mit den Kameras. [27]

Abbildung 2.10: Willow Garage PR2 Bewegungsfreiheit. Links: Omnidirektionales Fahrwerk ermöglicht Bewegungen in alle Richtungen. Rechts: Vertikale variable Basis erweitert den Bewegungsraum des Roboters. [27]

Der PR2 wird über ROS (siehe [19]) angesteuert. Der Herstellter Willow Garage positioniert seine Roboter bei weltweit führenden Universitäten im Bereich der Robotik um

die Weiterentwicklung der „Personal Robotics“ zu fördern.

2.2 Objekterkennung mit lokalen Features

Alle im vorigen Unterkapitel vorgestellten Roboter verwenden lokale Features für die Objekterkennung. Dieses Unterkapitel gibt eine Übersicht über die wichtigsten Feature basierten Methoden die derzeit in der mobilen Robotik Anwendung finden.

Feature Extraktion und Matching ist eines der fundamentalen Probleme in der Low-Level Computer Vision. Dabei wird der Bildinhalt in lokale, invariante Features transformiert. Diese Features sollen möglichst invariant in Bezug auf Translation, Rotation, Skalierung und Radiometrie sein.

Gegeben sei eine Serie an Bildern für ein bestimmtes Objekt. Zuerst wird versucht mittels Feature Extraktion eine Menge an stabilen Bildpunkten mit einer hohen Aussagekraft in jedem einzelnen Bild zu extrahieren. Dabei soll jeder Punkt eine bestimmte Stabilität in Bezug auf Translation, Skalierung und Rotation aufweisen. Die extrahierten Punkte aus jedem Bild werden anschließend mit den Punkten aus den anderen Bildern verglichen (Matching), um Feature zu Feature Korrespondenzen zwischen den einzelnen Bildern herzustellen. Abbildung 2.11 zeigt eine bildliche Darstellung für solche Feature Korrespondenzen. Feature Extraktion und Matching bilden somit das Fundament für übergeordnete Aufgaben wie zum Beispiel Objekterkennung, visuelle Odometrie oder 3D-Rekonstruktion.

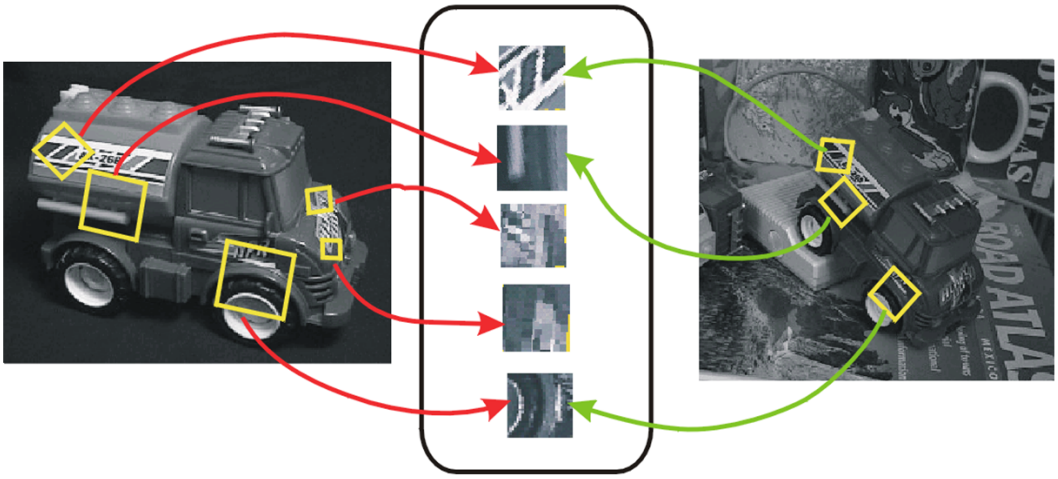

Abbildung 2.11: Beispiel für Feature zu Feature Korrespondenzen zweier Bilder mittels SIFT Features [13]

In diesem Unterkapitel werden die heute in der Robotik und Computer Vision ge-

bräuchlichsten Vertreter der lokalen Feature Extraktoren (Interest Point Detektoren und Deskriptoren) vorgestellt. Bei der Feature Extraktion werden generell zuerst stabile Interest Points (IP) bestimmt, die anschließend mit einem Deskriptor beschrieben werden. Das resultierende Set aus einem Keypoint (IP) und einem mehrdimensionalen Deskriptor wird später als Feature bezeichnet.

2.2.1 Feature Detektion

Die Extraktion von lokalen Features wurde speziell in den letzten zehn Jahren immer populärer. Einer der Hauptgründe für diese Entwicklung war die Platzierung des „Scale Invariant Feature Transform“ (SIFT) als Standardmethode.

2.2.1.1 SIFT Detektor

Der erste Schritt von SIFT besteht darin stabile, lokale Interest Points zu extrahieren. Für diesen Zweck wird der sogenannte Scale Space verwendet. Anstatt auf jeder Ebene einer Bildpyramide Keypoints zu extrahieren und diese zu matchen werden bei SIFT Keypoints extrahiert, die sowohl in Bezug auf Position und Skalierung stabil sind. Ermöglicht wird dies bei SIFT durch einen Scale Space, der mit Hilfe von Difference of Gaussians (DoG) aufgebaut wird. Der DoG approximiert den Laplacian, ist invariant gegenüber Skalierungs- und Rotationsänderungen und kann effizient mit Hilfe einer Bildpyramide berechnet werden. Dabei werden in jeder Ebene (Oktave) Differenzbilder von benachbarten, mit verschieden großen Gaußkernen gefilterten, Bildern erstellt. Der Prozess ist in Abbildung 2.12 grafisch dargestellt.

Um stabile Keypoints zu extrahieren muss der entstandene DoG Scale Space nach lokalen Maxima und Minima abgesucht werden. Dabei wird die rund um den mit „X“ markierten Pixel dargestellte 3x3x3 Umgebung (also die 26 Nachbarn in der 3x3 Umgebung auf angrenzenden Skalierungen) auf ein lokales Maximum bzw. Minimum geprüft. Abbildung 2.13 visualisiert dies.

Nachdem ein Keypoint Kandidat gefunden wurde, muss dieser einer weiteren Prüfung unterzogen werden und seine Position verfeinert werden. Bei lokalen Features ist es nicht wünschenswert Kantenpixel oder Regionen mit wenig Kontrast als Keypoints zu haben, da diese unzuverlässig matchen. Um diese ungünstigen Keypoints zu filtern wird die Hessische Matrix $H = \begin{bmatrix} Dxx & Dxy \\ Dxy & Dyy \end{bmatrix}$ für das Differenzbild D ermittelt und Keypoints entfernt bei welchen $\frac{Spur(H)^2}{Det(H)} > 10$ ist. Um für die verbleibenden Keypoints Rotationsinvarianz zu

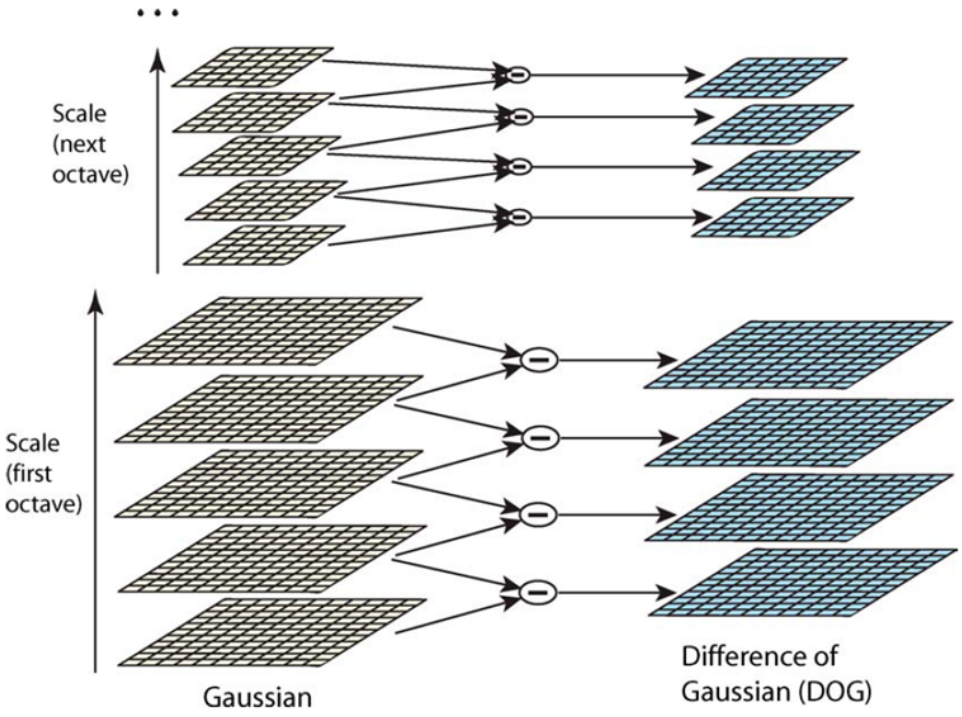

Abbildung 2.12: Aufbau der DoG Scale Space Pyramide [13]

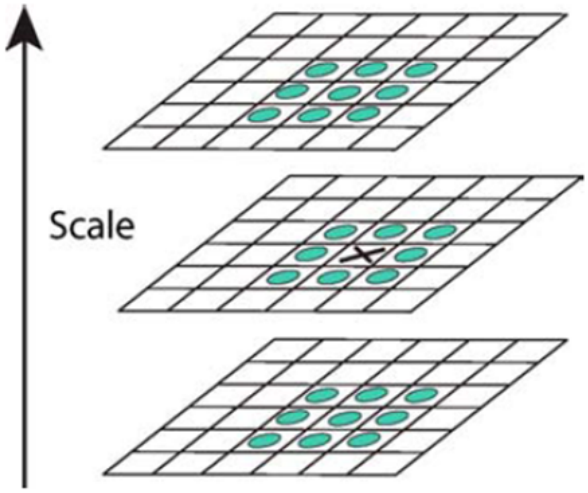

Abbildung 2.13: Maxima und Minima Detektion zur Keypoint Lokalisierung [13]

erreichen, muss die Hauptorientierung des Keypoints bestimmt werden. Dafür wird ein Orientierungshistogramm mit einer Diskretisierung von 10° pro Bin (36 Bin Histogramm) um den Keypoint berechnet. [13]

2.2.1.2 MSER

MSER (Maximally Stable Extremal Regions) wurde 2002 von [14] vorgestellt. Es handelt sich dabei um einen Affine-Invarianten Regionen-Detektor. Dieser Ansatz extrahiert stabile Regionen indem für jede Pixelintensität ein Schwellwertbild vom Eingabebild berechnet wird. Die resultierenden Binärbilder werden anschließend analysiert, indem die zusammenhängenden Konturen (Connected Components) eines Binärbildes mit den nächstgelegenen Binärbildern anderer Intensitätsstufen verglichen werden. Stabile Regionen sind solche, die am langsamsten über die Binärbilder expandieren. Dabei wird die Stabilität primär über den Parameter „Delta“ gesteuert, welcher angibt wie viele Intensitätsstufen erforderlich sind damit eine Region stabil ist. Um Maxima und Minima zu erhalten muss das Bild invertiert werden (MSER+, MSER-). [14]

Die detektierten MSER Regionen können mit verschiedensten Deskriptoren beschrieben werden. [17] passen jeweils einen elliptischen Patch an die detektierte MSER Region an und warpen diesen in einen zirkulären Patch konstanter Größe. Der Patch wird anschließend mit Hilfe Deskriptoren (z.B. SIFT) beschrieben werden.

2.2.1.3 SURF Detektor

SURF (Speeded Up Robust Features) wurde 2006 von [3] vorgestellt. Es handelt sich um einen auf Keypoints basierenden Feature Detektor und Deskriptor, welcher integrale Datenstrukturen zur Berechnungsbeschleunigung nutzt. Eine der bekanntesten integralen Datenstrukturen ist das Integralbild [26], welches in SURF für eine schnelle Boxfilterung benutzt wird. SURF bedient sich der Determinanten der approximierten Hesse-Matrix für die Skalierungsbestimmung. Für die Approximierung der Hesse-Matrix bedient sich SURF der Boxfilter mit denen in konstanter Zeit mit beliebig großen Kernen gefiltert werden kann. Im Gegensatz zu [13] wird nicht das Eingabebild skaliert um den Scale Space zu erstellen, sondern die Filtergröße verändert. Dies ist mit dem Integralbild sehr effizient möglich. Die Maxima Suche funktioniert ähnlich zu SIFT in einer 3x3x3 Umgebung. Die Hauptorientierung des Keypoints wird ebenfalls mit Hilfe von integralen Datenstrukturen (Haar Wavelets) berechnet. Hierzu wird die Umgebung des Keypoint Kandidaten nach x und y abgeleitet und die dominante Orientierung mittels eines Sliding Window Ansatzes

	CenSurE	SIFT	SURF
Spatial Resolution at Scale	Full	Subsampled	Subsampled
Scale Space Approximation	Laplace (Center Surround)	Laplace (DoG)	Hessian (DoB)
Edge Filter	Harris	Hessian	Hessian
Rotational Invariance	Approximate	Yes	No

Tabelle 2.1: Hauptunterschiede Keypoint basierender Detektoren [1]

gesucht.

2.2.1.4 CenSurE

CenSurE (Center Surround Extremas) ist ein neuer Detektor von [1] der SIFT und SURF ähnelt. CenSurE ist für Echtzeitanwendungen konzipiert worden und eignet sich speziell für visuelle Odometrie und Objekterkennung.

Der Hauptunterschied zwischen CenSurE und SURF/SIFT ist, dass CenSurE die Response für alle Pixel aller Skalierungen bestimmt. Im ersten Schritt wird bei CenSurE ähnlich wie bei SURF mit Hilfe von integralen Datenstrukturen eine Approximation des Laplacian of Gaussian durchgeführt. In der Praxis haben sich dafür die Box- und Oktagon-Filterung bewährt. Die Box-Filterung ist die einfachste und schnellste Variante, da hier mit Hilfe des Integralbildes die Summe der Pixelintensitäten einer aufrecht-rechteckigen Region mit vier Additionen bestimmt werden kann. Der Oktagon-Filterkern ist aufwändiger in der Berechnung, kann aber in Primitiva zerlegt werden: zwei Trapezoide und ein Rechteck. Es müssen hier zusätzlich zum Standard Integralbild zwei abgeschrägte Integralbilder verwendet werden. Es braucht drei Additionen um die Pixelintensitäten eines Trapezes zu bestimmen. Die Abbildung 2.14 zeigt die möglichen Filterkerne. Die Center-Surround Features werden für alle Positionen und auf allen Skalierungen berechnet. Im zweiten Schritt werden lokale Extremwerte bestimmt. Diese werden wie üblich in einer lokalen 3x3x3 Nachbarschaft berechnet. Im letzten Schritt werden Kandidaten entfernt die eine schwache Corner-Response nach dem Harris-Maß aufweisen.[1]

Bei SURF und SIFT nimmt die Lokalisierungsgenauigkeit bei höheren Skalierungen ab. Dieser Nachteil kann bei den beiden Verfahren nur durch Subsampling gelöst werden, welches sich aber negativ auf die Verarbeitungsgeschwindigkeit auswirkt. Laut [3] nimmt durch die Approximation der Hesse-Matrix bei SURF die Wiederholbarkeit bei Bildrotationen, die ein Vielfaches von $\pi/4$ aufweisen, ab. Dieser Umstand ist CenSurE hingegen fremd, da durch die Verwendung der Center-Surround Approximation eine Rotationsinvarianz erreicht wird. Abbildung 2.15 stellt die Wiederholbarkeit bei unterschiedlichen

SIFT	SURF	CenSurE (Octagons)	CenSurE (DoB)	Harris
304 ms	75 ms	23 ms	17 ms	10 ms

Tabelle 2.2: Detektor Rechenzeit für ein 512x384 Bild auf einem Intel Pentium-M 2GHz [1]

Rotationen gegenüber. Ein weiterer Vorteil von CenSurE gegenüber Hesse-basierten Systemen ist der Harris Kantenfilter, welcher zu einer besseren Kanteneliminierung führt als dies über die Hesse-Matrix möglich ist. [1]

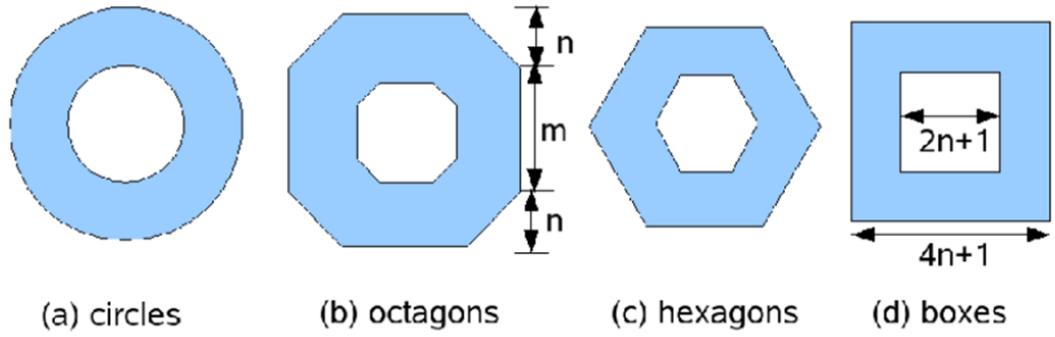

Abbildung 2.14: Center-Surround Filter. (a) Zirkulär symmetrischer Bilevel LoG. (b)(c)(d) stellen mögliche Approximationen von (a) mittels Boxfilter dar. [1]

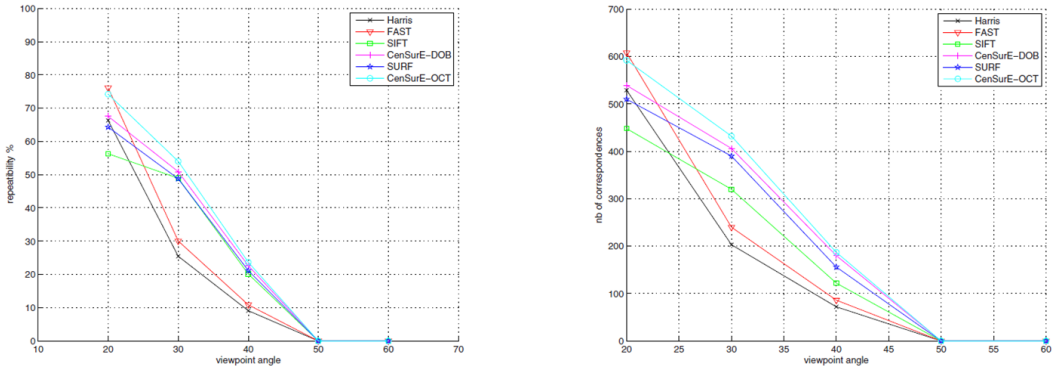

Abbildung 2.15: Gegenüberstellung der Wiederholbarkeit und Korrespondenzen anhand des Mikolajczyk Frameworks. Als Deskriptor für CenSurE wurde ein modifiziert SURF Deskriptor verwendet. [1]

Durch die Verwendung von integralen Datenstrukturen und den entsprechenden Filterkernen ist CenSurE ein sehr schneller Keypoint Detektor. Ein Vergleich ist in Tabelle 2.2 dargestellt. Als Deskriptor für CenSurE bietet sich SURF mit 64D an.

2.2.2 Feature Deskriptoren

Die vorher beschriebenen Methoden liefern Interest Points (IP) die möglichst stabil bei Veränderungen der Kameraposition bleiben. In diesem Unterkapitel werden Methoden gezeigt, mit deren Hilfe sich diese IP beschreiben lassen, um in weiterer Folge IP zu IP Korrespondenzen herstellen zu können.

2.2.2.1 SIFT Deskriptor

Der SIFT Deskriptor wird erzeugt, indem der Gradient für jeden Pixel in einem 16x16 Pixel Fenster (bestehend aus 4 Subregionen) um den detektierten Keypoint berechnet wird. Die Gewichtung der Gradienten-Magnitude erfolgt dabei über eine Gaußfunktion, damit sich Ungenauigkeiten in der Keypoint Positionsbestimmung (Fehlregistrierung) nicht drastisch auf den Deskriptor auswirken. Der Einfluss von Gradienten, die weiter vom Zentrum weg sind, ist mit Hilfe dieser Maßnahme nicht so stark. In weiterer Folge werden für jeden 4x4 Pixel Quadranten einer Subregion ein Gradienten-Histogramm mit acht diskreten Intervallen (Bins) erstellt. Um auch hier den Einfluss einer möglichen Fehlregistrierung zu minimieren wird jede der 256 gewichteten Gradienten-Magnituden mittels trilinearer Interpolation zu einem 2x2x2 Histogramm aufsummiert. Der resultierende Deskriptor (Feature-Vektor) mit 128 Werten (Dimensionen) wird anschließend noch auf die Einheitslänge normalisiert, um negative Effekte von Kontrast und Beleuchtung zu reduzieren. Anschließend werden die Werte im Deskriptor noch nach oben hin beschränkt (0.2) und der resultierende Vektor noch einmal auf die Einheitslänge normalisiert. Abbildung 2.16 skizziert den Prozess für die Erstellung des Deskriptors. [13]

2.2.2.2 SURF Deskriptor

Der SURF Deskriptor ähnelt dem SIFT Deskriptor. Im Unterschied enthält ein SURF Deskriptor nur 64 Dimensionen und wird über integrale Datenstrukturen berechnet. Er basiert auf der Summe der Haar Wavelet Werte innerhalb der Deskriptor Region, gesplittet in 4x4 Subregionen à 4 Dimensionen. Der 4D Vektor der Subregion beschreibt die zugrundeliegende Intensitätsstruktur mit $v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$. Die 4D Vektoren der 16 Subregionen werden zu einem 64D Deskriptor konkateniert und anschließend auf die Einheitslänge normalisiert. Eine weitere Zusatzinformation gibt die Spur der Hesse-Matrix über den mit dem Deskriptor beschriebenen Patch: das Vorzeichen des Laplace Operators. Diese Information beschreibt dunkle Regionen auf weißem Hintergrund bzw. umgekehrt.

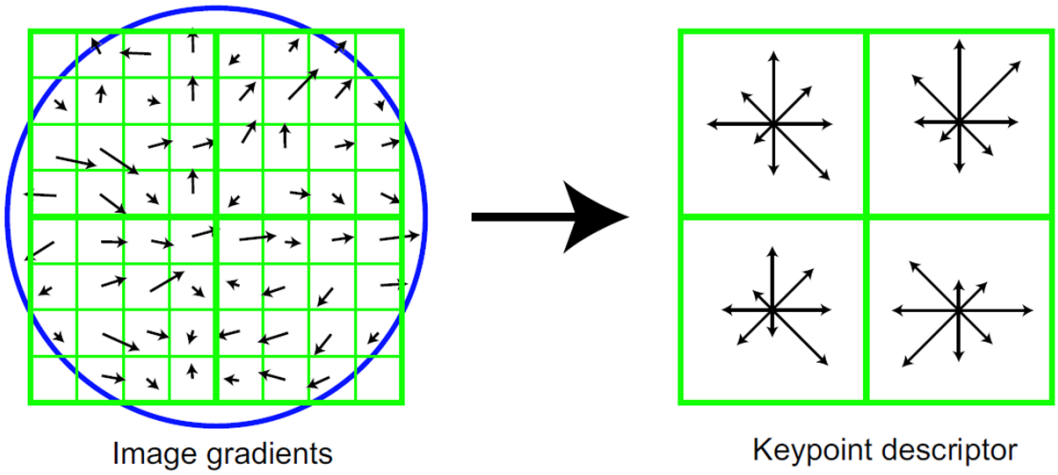

Abbildung 2.16: Schematische Darstellung vom SIFT Deskriptor. Die Orientierungen und Magnituden der Gradienten werden für jeden Pixel im Fenster berechnet und mit einer Gaußfunktion (blauer Kreis) gewichtet. Ein gewichtetes Gradienten-Histogramm wird für jede Subregion berechnet und zu einem Keypoint Deskriptor konkateniert.[13]

Mit Hilfe des Vorzeichens kann das nachfolgende Matching beschleunigen, da nur Blobs mit ähnlichem Kontrast verglichen werden müssen.

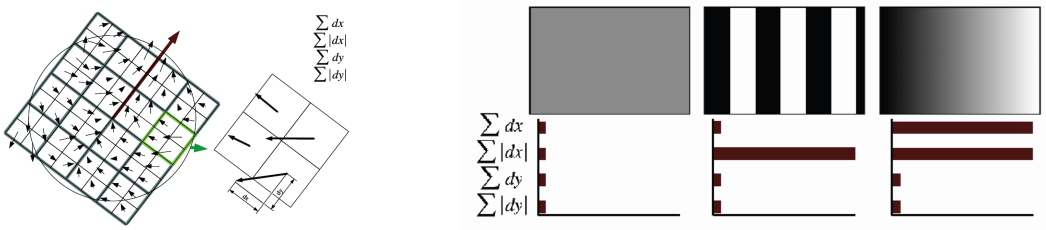

Abbildung 2.17: Aufbau des 64D Deskriptors bestehend aus 4x4 Subregionen à 4D Vektoren. [3]

In den letzten Jahren hat SURF vor allem in der mobilen Robotik immer stärker Einzug erhalten. Dies ist darauf zurückzuführen, dass die Berechnungszeit deutlich kürzer als SIFT ist und die Robustheit gut genug für die derzeitigen Anwendungen ist. SURF kann auf heutigen Computern in Echtzeit ausgeführt werden.

2.2.3 Feature Matching

In der Computer Vision ist das Thema „Feature Matching“ von besonderer Bedeutung, da die Deskriptoren von lokalen Features oft 64 Dimensionen (Standard SURF) oder 128 Dimensionen (Standard SIFT) aufweisen. Das Absuchen von hochdimensionalen Räumen

zur Feature-Vektor Korrespondenzfindung ist demnach ein sehr rechenintensiver Vorgang und lässt sich als Nächster-Nachbar (NN) Suchproblem definieren. Die Ermittlung des NN ist eines der wichtigsten Suchprobleme in der Bildverarbeitung basierend auf lokalen Features. Die Problematik besteht im schnellen Wiederfinden der besten Korrespondenzen von lokalen Features in großen Datenbanken. Das Suchproblem bekommt in Verbindung mit lokalen Features einen besonderen Stellenwert, da durch die Verwendung von hochdimensionalen Deskriptoren das Problem erschwert wird. In [13] wird beschrieben, dass sehr gute Such-Algorithmen wie der KD-Tree (siehe [7]) ab ca. 10 Dimensionen gegenüber der Brute-Force-Suche keinen Geschwindigkeitsvorteil bei der Ermittlung des exakten NN bringen: „Es gibt keinen bekannten Algorithmus der den exakten NN von Punkten in hochdimensionalen Räumen effizienter finden kann als eine Brute-Force-Suche.“ [13]

Die Brute-Force-Suche ist die naivste und rechenaufwändigste aller Methoden um den exakten NN zu ermitteln. Die Ermittlung des exakten NN stellt aber für viele Objekterkennungsanwendungen einen unnötigen Flaschenhals dar, da auf Grund von vielen extrahierten Features nicht immer der exakte NN ermittelt werden muss (Redundanz). Diese Unschärfe rechtfertigt die approximierte NN-Suche in hochdimensionalen Räumen, für die es mittlerweile verschiedenste Lösungsansätze gibt. Die Verwendung von approximierten NN-Suchstrategien beschleunigt den Vorgang drastisch, aber auf Kosten der Präzession. So kommt es in wenig Fällen vor, dass der exakte NN nicht ermittelt werden kann. Dieser Genauigkeitsverlust ist aber marginal und kann für die meisten Anwendungen wegen der hohen Redundanz vernachlässigt werden. Viele Arbeiten auf diesem Gebiet befassen sich mit der Thematik der approximierten NN Suche in großen Datenbanken. In diesem Unterkapitel werden die derzeit effizientesten und beliebtesten Methoden vorgestellt: „Best Bin First“, „Hierachical K-Means Trees“ und „Randomized KD-Trees“.

2.2.3.1 Best-Bin-First

Der „Best-Bin-First“ (BBF) Algorithmus [4] liefert eine approximierte Lösung für das NN-Suchproblem (ANN) in hochdimensionalen Räumen. Es handelt sich dabei um eine Modifikation des KD-Tree Such-Algorithmus von [7]. Der klassische kd-tree Algorithmus ist effizient in niedrig dimensionalen Räumen, aber je höher die Dimension desto schlechter ist seine Geschwindigkeit. BBF ist ein approximierende Variation des kd-tree Algorithmus, welcher den optimalen NN in den meisten Fällen findet. In den verbleibenden Fällen liefert BBF nicht den optimalen NN, sondern einen Nachbarn der sehr nahe am optimalen NN liegt. In [13] wird eine Zuverlässigkeit von 95% bei der approximierten NN Suche bezogen

auf 128 Dimensionen angegeben. Gegenüber der Brute-Force-Suche wird die Suchzeit um zwei Größenordnungen beschleunigt. [4]

Der klassische kd-tree partitioniert eine Menge an Punkten N in der Dimension i, in welcher die Daten die größte Varianz aufweisen. Die Trennung (Knotenpunkt) erfolgt beim Median Wert m dieser Dimension, sodass eine gleiche Anzahl an Punkten auf die eine und andere Seite des Knotens im Baum fallen. Der Knotenpunkt speichert m und i, danach beginnt der Trennungsvorgang für die beiden entstandenen Teilmengen von vorne und wiederholt sich solange bis die gesamte Menge partitioniert wurde. Es entsteht ein balancierter Binärbaum der Tiefe log^N bei dem die Blätter die komplette Menge N partitionieren. Um den NN für einen gesuchten Punkt q im kd-tree zu finden wird zuerst der gesamte Baum von oben nach unten einmal durchlaufen. Das erreichte Blatt wird in log^N Zeit erreicht und stellt schon eine gute Annäherung an den wahren NN dar. Um den optimalen NN zu finden wird Backtracking verwendet. Dabei werden ganze Teilbäume ausgelassen bei denen am Knoten die Distanz q zu dem bisherigen NN größer ist. Die Suche bricht ab, wenn keine noch nicht besuchten Verzweigungen mehr existieren. [4]

Da der klassische kd-tree nur in niedrig dimensionalen Räumen effizient funktioniert, wird die Suchreihenfolge modifiziert um eine Geschwindigkeitssteigerung auch für hochdimensionale Räume zu erreichen (zB 128D für SIFT). Ermöglicht wird dies durch die Einführung einer Prioritätsschlange für die effiziente Bestimmung der Suchreihenfolge (Backtracking Schritt) und einer Abbruchbedingung basiert auf einer konstanten Anzahl an zu besuchenden Knotenpunkten bevor der approximierte NN zurückgeliefert wird. [4]

[13] verwendet als Abbruchbedingung den Besuch von 200 NN Kandidaten. Für eine Datenbank mit 100000 Feature-Vektoren mit 128 Dimensionen zeigt Lowe, dass eine Geschwindigkeitssteigerung von zwei Größenordnungen zur exakten NN-Suche möglich ist. Nur 5% der korrekten Korrespondenzen gehen auf Kosten der Approximierung verloren.

2.2.3.2 Hierarchical k-means Tree

Schon 1975 haben [8] vorgeschlagen, dass NN Suchproblem mit Hilfe einer Baumstruktur und k-means Clustering zu lösen. In [17] wurde der hierarchische k-means Baum für die Computer Vision erfolgreich umgesetzt („Vokabelbaum“).

Der Vokabelbaum definiert eine hierarchische Quantisierung, welche durch k-means Clustering gebildet wird. Dabei definiert k nicht die schlussendliche Anzahl der Cluster, sondern den Verzweigungsfaktor für einen Knoten (Anzahl der Kinder pro Knoten). Der Vorgang findet auf den gesamten Deskriptor-Vektoren aller zu lernenden Objekte statt.

Zuerst wird k-means Clustering auf den gesamten Trainingsdaten angewendet um die ersten k Cluster Zentren zu finden. Die Trainingsdaten werden dann in k Gruppen partitioniert, wobei jede Gruppe aus den Feature-Vektoren mit dem geringsten Abstand zum jeweiligen Clusterzentrum besteht. Der Vorgang wird solange rekursiv per Gruppe wiederholt bis das Abbruchkriterium erfüllt ist. Die Rekursion bricht ab, wenn die Anzahl der Feature-Vektoren einer Gruppe kleiner als k ist. Das Training des Vokabelbaums mittels hierarchischem k-means Clustering findet zuerst Offline statt. In der Online-Phase können neue Daten zur Laufzeit (on-the-fly) zum Vokabelbaum hinzugefügt werden. Für die Suche im Vokabelbaum zur Laufzeit wird der Feature-Vektor von der Wurzel des Baums aus zu den Knoten weiterpropagiert. Dort wird der Abstand des Feature-Vektors zu den Clusterzentren der Kinder des Knoten berechnet und der Knoten mit dem minimalen Abstand ausgewählt. Dann wiederholt sich der Vorgang bis ein kinderloser Knoten erreicht wurde. Abbildung 2.18 stellt den Aufbau des Vokalbaums mit k=3 grafisch dar. Abbildung 2.20 zeigt die Traversierung und Bewertung einer Suche im Baum. [17]

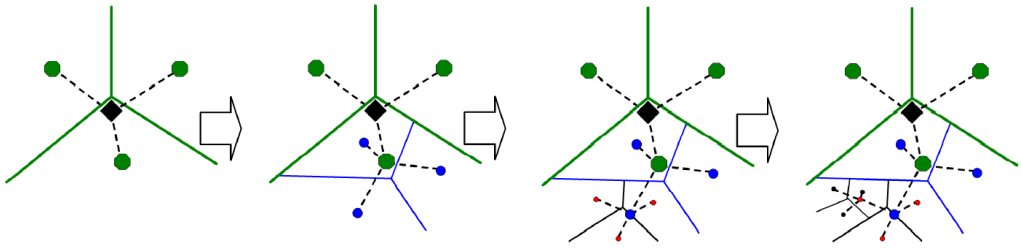

Abbildung 2.18: Grafische Darstellung des Aufbaus vom Vocabulary Tree mittels k-means (k=3)[17]

Neben der Methode von Nister und Stewenius gibt es eine weitere Variante von Muja und Lowe. In [16] wird die schon auf KD-Trees erfolgreich angewandte „Best-Bin-First“ Methode auf dem k-means Baum angewendet. Zuerst durchläuft der Algorithmus den Baum einmal und fügt alle nicht besuchten Verzweigungen zu einer Prioritätsschlange hinzu. Dann wird aus der Prioritätsschlange die Verzweigung gewählt, deren Clusterzentrum am nächsten zum Such-Feature-Vektor liegt. Danach wird die Suche von diesem Knotenpunkt aus neu gestartet und solange wiederholt bis das Abbruchkriterium erfüllt wurde. Je nach Approximationsgrad ist das Abbruchkriterium durch Early-Stopping geprägt. Das bedeutet, dass ein Abbruch nach n Knotenpunkten stattfindet. Der Parameter n wird während des Trainings ermittelt und ist abhängig von der gewünschten Präzession. [16]

Die Abbildung 2.19 zeigt eine grafische Darstellung für unterschiedliche hierarchische

k-means Bäume mit verschiedenen Verzweigungstiefen k.

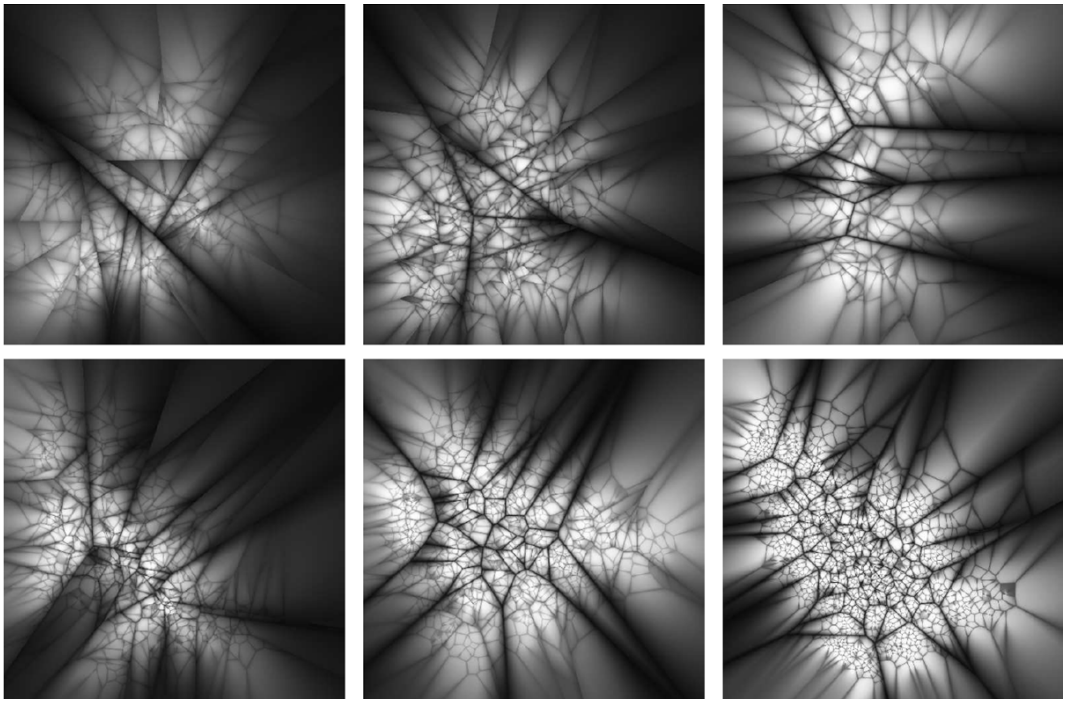

Abbildung 2.19: Grafische Darstellung verschiedener Verzweigungstiefen (2,4,8,16,32,128) beim hierarchischen k-means Baum. Die Grauwerte repräsentierten das Verhältnis zwischen den Distanzen zu den nächsten und zweitnächsten Cluster Zentren pro Baumebene. Dunkle Bereiche repräsentieren ein geringes Verhältnis und grenzen die einzelnen k-means Gruppen voneinander ab. [16]

2.2.3.3 Randomized KD-Trees

Das Konzept der randomisierten kd-trees (RKD-tree) wurde in [21] vorgestellt. Sie unterstellen, dass der Ansatz der Splittung beim kd-tree, nämlich in der Dimension mit der höchsten Datenvarianz, in der Realität nicht viele Auswirkungen hat. Somit macht es eigentlich keinen großen Unterschied in welcher Dimension die Splittung vorgenommen wird und könnte zufällig erfolgen. Diese Beobachtung bildet die Grundlage für RKD-trees, indem die Trennung bei einer zufällig gewählten Dimension innerhalb der ersten D Dimensionen auf jedem Level gemacht wird. In [16] werden typischerweise für D fünf Dimensionen gewählt. Bei der Erstellung von RKD-trees werden somit m Bäume unter Verwendung von unterschiedlicher Randomisierung aufgebaut. Bei der Suche in RKD-trees wird eine gemeinsame Prioritätsschlange für alle m Bäume verwendet. Die Suche läuft gleichzeitig

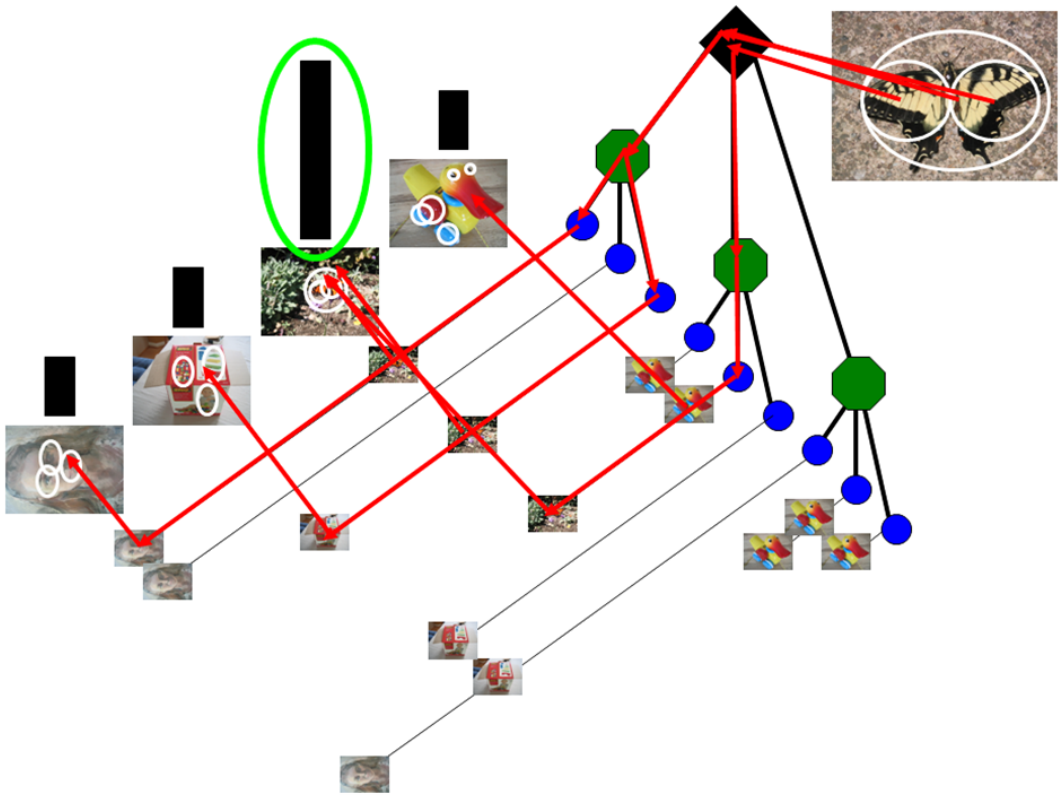

Abbildung 2.20: Traversierung und Bewertung des Nister Vokabelbaums [17]

auf allen m Bäumen. Zuerst wird der Initiale NN in allen Bäumen gesucht. Hierzu wird jeder Baum einmal durchlaufen und der beste NN jedes Baumes ermittelt. Unter Verwendung der gemeinsamen Prioritätsschlange wird ein Ranking aller Knoten aller Bäume vorgenommen. Dies resultiert darin, dass alle Bäume in der Reihenfolge ihrer Distanz zum gesuchten Punkt q simultan abgesucht werden. Wie auch schon bei der approximierten kd-tree Suche wird das Suchlimit von n Knoten verwendet. Dadurch lässt sich auch bei RKD-trees der Grad der Präzession steuern. Die Abbildung 2.21 stellt die in diesem Unterkapitel beschriebenen Verfahren der Approximierten-Nächster-Nachbar Suche gegenüber. Die approximierte NN mittels einem kd-tree wird als ANN in Abbildung 2.21(a) gezeigt. In [16] wird ein Framework präsentiert, dass je nach gegebenen Datensatz automatisch den optimalen Algorithmus (k-means-tree oder rkd-trees) und die optimalen Parameter für die gewünschte Präzession, Speicherbedarf und Trainingszeit auswählt.

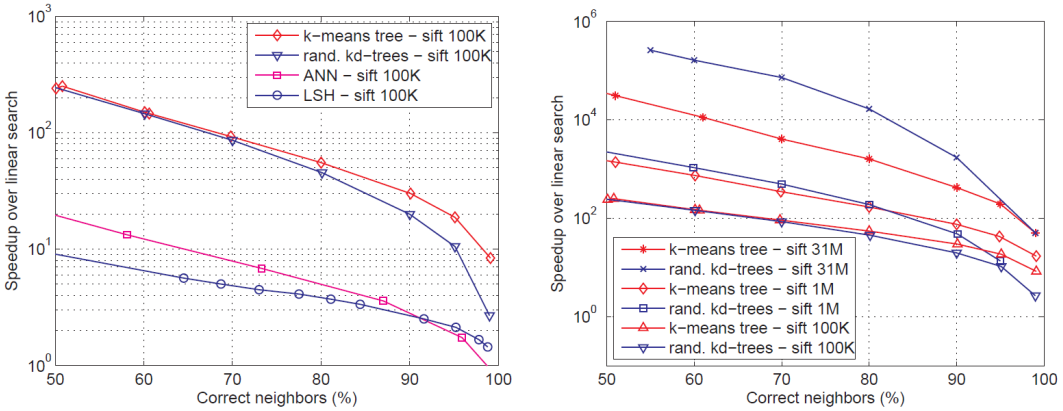

Abbildung 2.21: Vergleich der beschriebenen Algorithmen untereinander (b) Vergleich von k-mean-tree und rkd-trees in Bezug auf unterschiedliche SIFT Datensatzgrößen [16]

2.3 Bewertung der Ansätze

In dem Kapitel wurde gezeigt dass SIFT noch immer der Gold-Standard unter den Feature Extraktoren ist, an dem sich andere Ansätze messen müssen. Durch die Verwendung von integralen Datenstrukturen ist es mit Methoden wie SURF oder CenSurE möglich ähnlich gute Ergebnisse zu erzielen wie mit SIFT, aber in kürzerer Zeit. Der Vorteil von SURF gegenüber SIFT ist, dass SURF einen Faktor 4 schneller berechnet als SIFT (gezeigt in Tabelle 2.2). Der neue Feature Detektor CenSurE ermöglicht gegenüber SURF eine weitere Geschwindigkeitssteigerung bei einer Verbesserung der Wiederholbarkeit (siehe Abbildung 2.15). Somit erscheint CenSurE die optimale Wahl als Keypoint Detektor zu sein. Der SIFT als auch der SURF Deskriptor sind rotationsinvariant und eignen sich für das Objekterkennungssystem. Die Berechnungszeit von SURF ist geringer, die Genauigkeit nur marginal schlechter. Für die Konzipierung eines Echtzeit Systems ist der SURF Deskriptor somit bestens geeignet.

Die extrahierten Features können mit allen beschriebenen Matching Verfahren verglichen werden. Das Bruteforce Matching ist genau, eignet sich aber auf Grund der Komplexität nicht für große Objekt-Datenbanken. Für große Datenbanken eignen sich baumartige Strukturen wie RKD-Trees oder der k-means Tree. Die RKD-Trees arbeiten auf großen Objekt-Datenbanken schneller als der k-means Tree. Weiters lässt sich beim RKD-Tree der Grad der Präzession steuern. In Bezug auf ein skalierbares, echtzeitfähiges System zeichnen sich die RKD-Trees als Favorit ab.

Kapitel 3

FLEA Roboter System

Im Jahr 2006 wurde mit der Entwicklung des "Friendly Learning Electronic Assistant"(FLEA) an der Technischen Universität Graz (TUG) begonnen. FLEA wird seitdem kontinuierlich weiter entwickelt und wurde von Anfang an für die Verwendung bei RoboCup@Home konzipiert. Bis zum Jahr 2008 war FLEA gleichzeitig das RoboCup@Home Team der TUG und konnte einige Erfolge verbuchen. So wurde FLEA 2007 Vize-Europameister bei RoboCup@Home! Die Abbildung 3.1 zeigt einige Impressionen aus dieser Zeit.

Abbildung 3.1: FLEA Generation 2 (Mr. FLEA) und 3 (Nova), RoboCup@Home Vize Europameister 2007 (Teamfoto)

3.1 FLEA PR-G5

Seit 2008 ist das Projekt nicht mehr an der TUG, sondern wurde vom Autor privat weiter betrieben. Dieser Schritt war vor allem durch Veränderungen im Team beeinflusst. Aufgrund dieser Veränderungen musste der Roboter 2008 von Grund auf neu konzipiert werden. Es entstand der Roboter der 4. Generation, der aber sehr schnell durch FLEA

der fünften Generation (PR-G5) ersetzt wurde. PR-G5 wurde in den Jahren 2008 und 2009 komplett eigenständig entwickelt. Auslöser war, dass die gesamte Hardware der Vorgängermodelle nicht übernommen werden konnte. FLEA PR-G5 (Finalist bei der RoboCup@Home Weltmeisterschaft 2009) ist in Abbildung 3.2 gezeigt.

Abbildung 3.2: FLEA, Projekt Generation 5 (PR-G5), Stand Weltmeisterschaft 2009

Die Hardware von PR-G5 besteht aus drei Kernkomponenten: Fahrwerk, Torso und Kopf. Entscheidend für diese Trennung ist die Möglichkeit, die einzelnen Module separat entwickeln zu können. Auch ist es mit Hilfe der Modulbauweise einfacher später Komponenten auszutauschen.

Das Fahrwerk von PR-G5 wurde aus Aluminium mit leistungsstarken, bürstenlosen Gleichstrom-Elektromotoren gebaut. Die Motoren beinhalten die gesamte Steuerungselektronik und werden über das aus der Automobilindustrie bekannte CAN-Bus Protokoll gesteuert. Im Fahrwerk sind auch die Akkumulatoren untergebracht. Aus finanziellen Gründen wurden Bleigel-Akkus mit einer Gesamtleistung von 768Wh im Fahrwerk verbaut. Dies ist ausreichend um den Roboter für drei bis sechs Stunden zu betreiben. An der Frontseite befinden sich Ladekontakte um an der Ladestation andocken zu können. Das Fahrwerk bildet eine abgeschlossene Einheit. Der komplette Kabelbaum sowie Sicherun-

gen und Schalter sind fix in die Plattform integriert. Ein weiteres Merkmal ist die fixe Integration des Laserscanners für Lokalisierung und Navigation in das Fahrwerk. Der Laserscanner scannt Objekte auf einer Ebene mit einer Höhe von 48cm. Auch die klare Sichtbarkeit und Zugänglichkeit der Not-Aus Knöpfe ist von entscheidender Bedeutung, da diese Sicherheitsmaßnahme zwingend erforderlich für jeden Wettbewerb ist.

Der Torso besteht aus einem Computer und einigen Sensoren. PR-G5 beinhaltet nur einen einzigen, dafür aber leistungsfähigen Rechner. Eine QuadCore CPU mit je 2.83GHz pro Kern und 12MB Cache, 4GB RAM und eine CUDA fähige nVidia GPGPU sorgen für die nötige Rechenkapazität. Ein auf der Rückseite befindliches Touch-TFT Panel erlaubt die komfortable Bedienung des Roboters. Für Entwicklungstätigkeiten sehr hilfreich ist auch die Integration einer ausziehbaren Lade mit einer Tastatur/Touchpad Kombination und USB Stecker. An der Frontseite des Torsos befinden sich ein schräg nach oben gerichteter Infrarot-Sensor und ein nach vorne gerichteter Ultraschall-Sensor. Diese Sensoren dienen der Absicherung gegenüber hoher Tischkanten und zur Verifizierung ob eine Person vor dem Roboter steht. An der Oberkante des Torsos befindet sich eine kleine Stereokamera, die schräg nach unten Richtung Boden gerichtet ist. Diese dient der Hinderniserkennung, da Objekte unter 48cm nicht vom Laserscanner erfasst werden (zB: Schuhe). Die schlechte geometrische und radiometrische Auflösung der kleinen Stereokamera macht sie unbrauchbar für Objekterkennungsaufgaben am Roboter.

Für die RoboCup@Home Weltmeisterschaft 2009 wurde ein neuer Kopf entwickelt. Dieser sollte vor allem funktional, aber keinesfalls menschenähnlich sein. Aus Kostengründen konnte zum damaligen Zeitpunkt nur eine Logitech Webcam verbaut werden. Der Kopf („Orb“) kann rotieren (Tilt Unit) und beinhaltet auch das Mikrofon für die Mensch-Roboter Interaktion über natürliche Sprache. Für diese Diplomarbeit wurde PR-G5 mit einer Stereokamera erweitert. Diese sitzt auf dem Torso und kann Objekte auf Tischhöhe einsehen. Das Stereosystem ist eine Bumblebee XB3 von Pointgrey. Die Stereokamera liefert monochrome Bilder mit einer Auflösung von 1280x960 Pixel. Die Baseline zwischen den Kameras beträgt 12cm, wobei eine dritte Kamera die Baseline auf 24cm erweitern kann. Diese variable Baseline ist weniger wichtig für die Objekterkennung im Zuge der Diplomarbeit, kann aber für Anwendungen (zB Visual SLAM) von Vorteil sein. Die aktuelle Version von FLEA ist in Abbildung 1.2 dargestellt.

3.2 Hardware Komponenten

Die Aufgabenstellung stellt besondere Anforderungen an die Hardware und Software des Roboters. FLEA muss in der Lage sein eine kleine Haushalts-Umgebung ($20m^2$ bis $50m^2$) kontrolliert (ohne in Hindernisse zu fahren) und möglichst effizient abzusuchen. Hierfür stehen nur wenige Minuten Zeit zur Verfügung. Es ist also zwecknotwendig in diesem Kapitel die unmittelbar mit der Lösung der Aufgabe in Zusammenhang stehenden Roboterkomponenten zu beleuchten. Diese sind neben der Hardware des Roboters auch Software-Komponenten wie Navigation, Lokalisierung und Planung. Für die visuelle Suche benötigt ein Roboter neben einer Kamera und einem Objekterkennungssystem eine Vielzahl von Subsystemen die ihm die Interaktion mit der realen Welt ermöglichen. Dieses Unterkaptitel beschreibt die Architektur und die benötigten Systemkomponenten des Roboters.

Das FLEA Roboter System ist ein großes, komplexes Software- und Hardwaresystem. Für die Aufgabenstellung werden nicht alle verfügbaren Subsysteme benötigt. Die Abbildung 3.3 skizziert die wichtigsten Komponenten, die an der visuellen Suche beteiligt sind. Zuerst wird kurz auf die verwendete Hardware eingegangen, danach werden die Methoden und Softwarekomponenten beschrieben. Die Software des Roboters besteht aus mehreren Schichten, läuft aber innerhalb desselben Prozesses in mehreren Threads. Dabei besitzt jedes Hardwaremodul einen eigenen Thread. Selbiges gilt für die implementierten Softwaremodule wie Navigation, Planung, Lokalisierung, Scheduler und Sprache. Komplexer ist der Aufbau des Vision Systems. Diese System wurde im Zuge der Diplomarbeit neu entwickelt und wird im Kapitel 4 beschrieben.

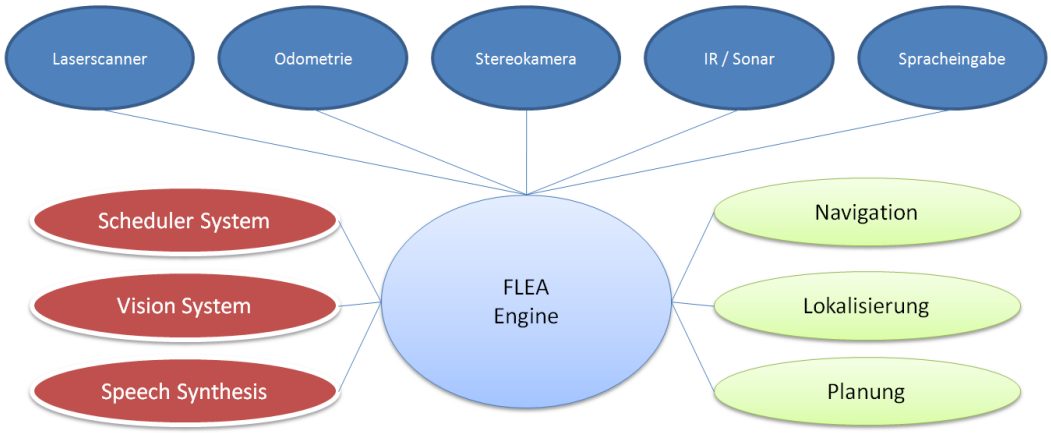

Abbildung 3.3: FLEA Systemübersicht

Das Robotersystem verfügt über Sensoren und Aktuatoren die eine Abtastung und

Interaktion mit der Umgebung erlauben.

3.2.1 Aktuatoren

Für die Fortbewegung verwendet FLEA das in der Robotik weit verbreitete Differential-Drive. Dieses Fahrwerk zeichnet sich vor allem durch seine einfache Steuerbarkeit und Bauweise aus. Zwei von Motoren direkt angetriebene Räder und ein frei bewegliches Stützrad sind dafür notwendig. Bei dieser Art von Fahrwerk handelt es sich um ein nicht-holonomes System, welches in der Ausführung von Bewegungen beschränkt ist. Mit einem Differential-Drive ist der Roboter nur in der Lage sich vorwärts, rückwärts und in Kurvenbahn zu bewegen, aber nicht seitwärts. Hilfreich ist, dass der Roboter sich mit Hilfe dieses Fahrwerks auch am Stand drehen kann. Dazu müssen die Motoren gegengleich angetrieben werden.

Die Vorgängerversion von FLEA hatte auch einen Greifarm an Bord. Zum gegenwärtigen Zeitpunkt ist der Roboter aber nicht mit einem solchen ausgestattet. Die Entwicklung eines Greifers ist eine mögliche Weiterentwicklung des Roboters FLEA und nicht Teil dieser Arbeit.

3.2.2 Sensoren

Zur Abtastung der Umgebung benutzt FLEA eine Vielzahl von verschiedenen Sensoren. Jeder Sensor wird mit unterschiedlichen Frequenzen betrieben und liefert unterschiedliche Daten.

3.2.2.1 Laserscanner

FLEA verwendet einen Leuze Rod4 Laserscanner (siehe Abbildung 3.4) als primären Sensor zur Lokalisierung und Navigation.

Abbildung 3.4: Leuze Rod4. Laserscanner. [12]

Der Laserscanner liefert 25 Updates pro Sekunde mit einer maximalen Reichweite von 65m und einer Winkelauflösung von 0,36°. Der Scanbereich von 190° ist in Abbildung 3.5 veranschaulicht. Bei FLEA ist der Laserscanner direkt über dem Fahrwerk montiert und liefert einen horizontalen 2D Scan der Umgebung in der Höhe von 48cm über dem Boden. Somit können mit Hilfe des horizontalen Lasers (und der richtigen Software) Wände, Menschen oder Möbelstücke erkannt werden. Der Laserscanner arbeitet auf dem Time-of-Flight (ToF) Prinzip und hat eine Genauigkeit von bis zu 15mm. Der Laserscanner wird über eine Ethernet Schnittstelle angesprochen.

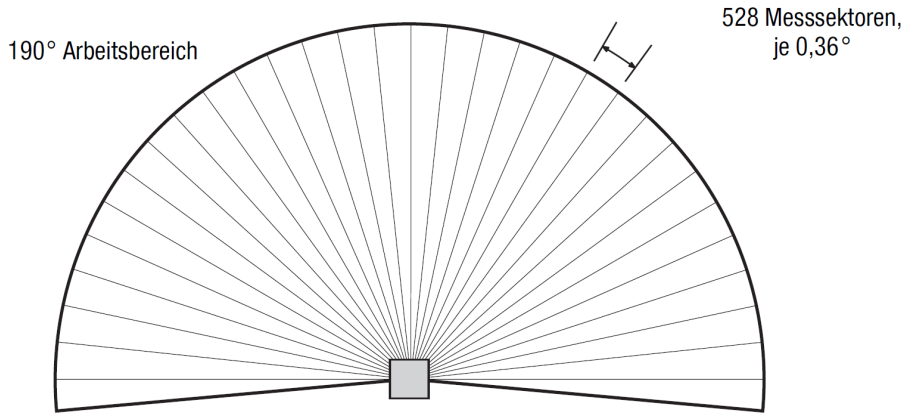

Abbildung 3.5: Rod4 Scanbereich. [12]

3.2.2.2 Kamerasystem

Für die Diplomarbeit wurde FLEA um die Pointgrey Bumblebee XB3 Stereokamera erweitert um einerseits hochauflösende Bilder für die Feature-Extraktion zu erhalten und andererseits für die Gewinnung von Tiefeninformation. Die in FLEA verwendete Bumblebee XB3 liefert Bilddaten mit einer Auflösung von 1280x960 Pixel und bis zu 15 Bilder pro Sekunde. Die minimale, hardwarebedingte Latenzzeit beträgt bei dem verwendeten Bussystem 67ms.

Die verwendete Bumblebee Stereo Kamera ist mit einem 3.8mm Objektiv ausgestattet und es wird eine Stereo Baseline von 12cm verwendet. Das 3.8mm Objektiv ergibt in Kombination mit dem 1/3“ Sensor einen horizontalen Öffnungswinkel von 66°. Die Kamera ist mit monochromen CCD Sensoren (ohne Bayer Filter) ausgestattet.

Pointgrey liefert zu der Kamera auch einen SDK mit entsprechenden Treibern. Die Stereo Kamera ist ab Werk kalibriert. Mit Hilfe vom SDK können Bilder effizient aufgenommen und entzerrt werden. Zusätzlich inkludiert Pointgrey einen Stereo-Densematcher

Abbildung 3.6: Pointgrey Bumblebee XB3. 1280x960 Pixel. Monochrom. [18]

für die Gewinnung von Tiefeninformation.

3.3 Software Komponenten

Dieses Unterkapitel gibt eine Übersicht über die wichtigsten Softwaremodule des Roboters FLEA die für die Durchführung der visuellen Suche notwendig sind. Moderne mobile Roboter arbeiten nach dem Prinzip der „Probabilistic Robotics“, welches im gleichnamigen Buch von [25] beschrieben ist. Diese wahrscheinlichkeitsbasierte Herangehensweise kann sehr gut mit Unsicherheiten umgehen, mit denen ein Roboter permanent konfrontiert ist. Man stelle sich zum Beispiel eine einfache Karte einer Haushaltsumgebung vor in der sich der Roboter lokalisieren soll. Diese Karte wird einmal erstellt und ist im Regelfall nach einiger Zeit veraltet, da sich zB Objekte im Zimmer verschieben oder Objekte nach einiger Zeit entfernt bzw. hinzugefügt werden. Hinzu kommt, dass Sensoren nicht absolut fehlerfrei arbeiten und sich öfter Meßfehler einschleichen können. Noch schlimmer wird es für einen Roboter, wenn sich Objekte in dieser Umgebung bewegen. Mit Hilfe von Methoden der „Probabilistic Robotics“ gelingt es mit dieser Unsicherheit umgehen zu können.

3.3.1 Lokalisierung

Um die visuelle Suche effizient durchführen zu können, muss FLEA zu jeder Zeit die genaue Position in der Wohnung bestimmen können. Die Indoor Robotik bedient sich an Methoden die auf Umgebungskarten basieren. Eine Umgebungskarte stellt eine kompakte, abstrahierte Repräsentation der Umgebung dar.

Der gebräuchlichste Vertreter von Umgebungskarten in der Robotik ist die Oc-

> cupancy Grid Map. Dabei handelt es sich um eine fein aufgelöste Rasterkarte. Die Occupancy Grid Map zeigt einen 2D Raumplan, welcher durch einen horizontalen Schnitt mit der 3D Umgebung entstanden ist. In den meisten Fällen befindet sich die Schnittebene auf Höhe des Laserscanners. Die 2D Occupancy Grid Map ist die bevorzugte Repräsentation für Umgebungen in denen sich der Roboter planar fortbewegt. S. 284, 285 [25]

Die Erstellung einer Occupancy Grid Map kann entweder manuell geschehen (z.B.: CAD Programm) oder automatisch durch den Roboter. Der automatische Ansatz wird als SLAM (Simultaneous Localization and Mapping) bezeichnet. FLEA verwendet einen auf dem Partikelfilter aufbauenden SLAM Ansatz um die Gridmap mittels Odometrie und Laserscanner Daten zu erstellten. Die Occupancy Grid Map kann bei FLEA als Bild interpretiert werden, wobei Pixel die freie Bereiche repräsentieren den Wert 0 haben und Wände oder andere vom Laser erfasste Hindernisse mit 255 vermerkt werden. Zusätzlich könnten auch noch Bereiche definiert werden, die der Roboter nicht befahren darf (zB Tische, Sofa, Pflanzen). Diese Bereiche werden als „Offlimit“ bezeichnet. Abbildung 3.8 zeigt die Occupancy Grid Map von FLEA mit den Offlimit-Bereichen. Abbildung 3.7 zeigt die Rohdaten, die für die automatische Kartengenerierung verwendet werden und das Ergebnis einer Occupancy Grid Map Erstellung via SLAM auf diesen Rohdaten.

Für die zu implementierende visuelle Suche wird vorausgesetzt, dass dem Roboter bereits eine Karte für die Navigation, Lokalisierung und Planung zur Verfügung steht. Ein autonomer mobiler Roboter muss zu jeder Zeit wissen, wo er sich in Bezug auf die Karte befindet. Dies geschieht über das Lokalisierungsmodul, welches bei FLEA die Monte Carlo Lokalisierung implementiert.

> Monte Carlo Lokalisierung (MCL) ist eine der populärsten Lokalisierungsmethoden für mobile Roboter. Abbildung 3.9 verdeutlicht die Funktionsweise mit einem einfachen eindimensionalen Beispiel bei dem ein Roboter einen Gang entlang fährt. Zuerst wird die globale Unsicherheit der Lokalsierung bestimmt, indem zufällig Partikel (mögliche Roboter Posen) überall eingestreut werden (siehe a). Wenn der Roboter eine Türe erkennt, wird die Wichtigkeit eines jeden Partikels neu bewertet (siehe b). Die Höhe eines Partikels repräsentiert in diesem Beispiel die Wichtigkeit des Partikels (Partikelgewichtung). Der Roboter bewegt sich vorwärts und die Partikel werden an der neu berechneten Position in Zusammenhang mit der zuvor bestimmten Partikelgewichtung neu

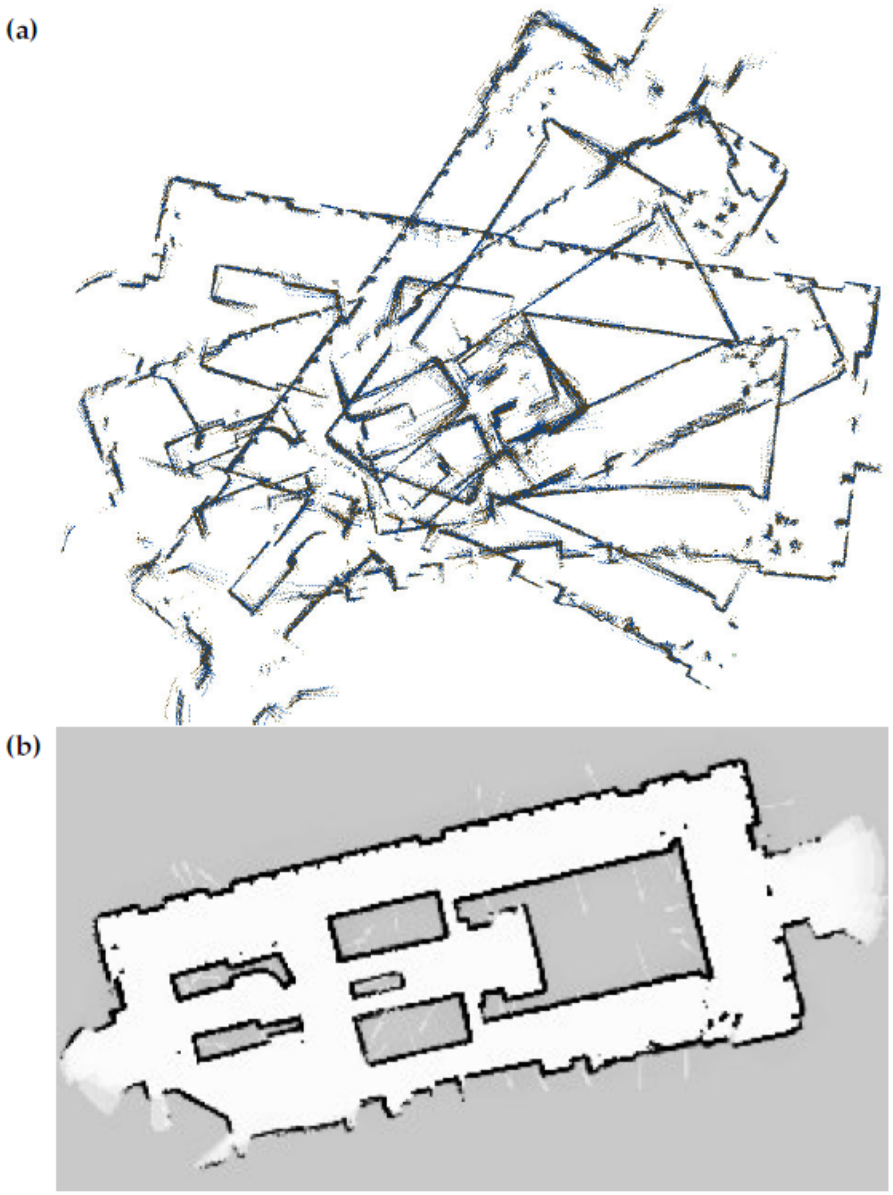

Abbildung 3.7: (a) Karte erstellt basierend auf den Rohdaten (Laser, Odometrie) ohne SLAM (b) SLAM basierte Karte auf den selben Rohdaten. [25] S. 283

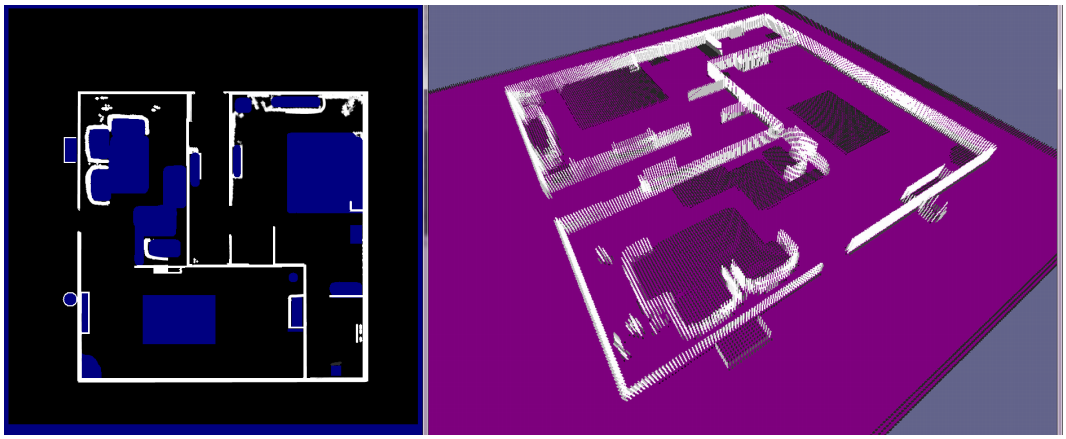

Abbildung 3.8: Occupancy Grid Map für Lokalisierung und Navigation. Links: Occupancy Grid Map von FLEA (RoboCup@Home Weltmeisterschaft 2009). Rechts: Visualisierung in 3D, befahrbare Bereiche sind purpur eingefärbt. Vom Laser erkennbare Orientierungshilfen (Wände, Mistkübel,...) sind weiß eingefärbt. Offlimits Bereiche (Tischplatte, räumliche Abgrenzung, etc.) sind schwarz eingefärbt.

> eingestreut (Resampling). Dies führt zu einer neuen Partikelverteilung mit uniformen Partikelgewichtungen mit Häufungen bei den drei möglichen Positionen (siehe c). Danach wird wieder eine Türe erkannt und die Partikelgewichtungen neu bestimmt (siehe d). Der Großteil der Partikel befindet sich jetzt bei der zweiten Türe, welche auch die wahrscheinlichste Position ist. Weitere Vorwärtsbewegung führt zu weiterem Resampling (siehe e). (S. 250) [25]

Mit Hilfe der Occupancy Grid Map und MCL berechnet FLEA die externe Orientierung des Roboters zehn Mal pro Sekunde neu. Das Lokalisierungs-Modul bedient sich dabei ausschließlich der aktuellen Odometrie und Laserscanner Daten.

3.3.2 Navigation

Der Roboter weiß zu jeder Zeit, wo er sich in Bezug auf die Karte befindet. Die Karte stellt gleichzeitig auch unüberwindbare (statische) Hindernisse dar, mit der der Roboter nicht kollidieren darf. Um die Umgebung nach Objekten absuchen zu können, muss der Roboter eine Fahrplanung durchführen. Dabei ist zu beachten, dass FLEA sich in einer dynamischen Umgebung kontrolliert fortbewegen soll ohne mit Hindernissen zu kollidieren. Eine oftmalige Neuplanung ist somit unabdingbar, da sich die Umgebung ständig ändern kann (Personen, bewegte Möbelstücke, etc.).

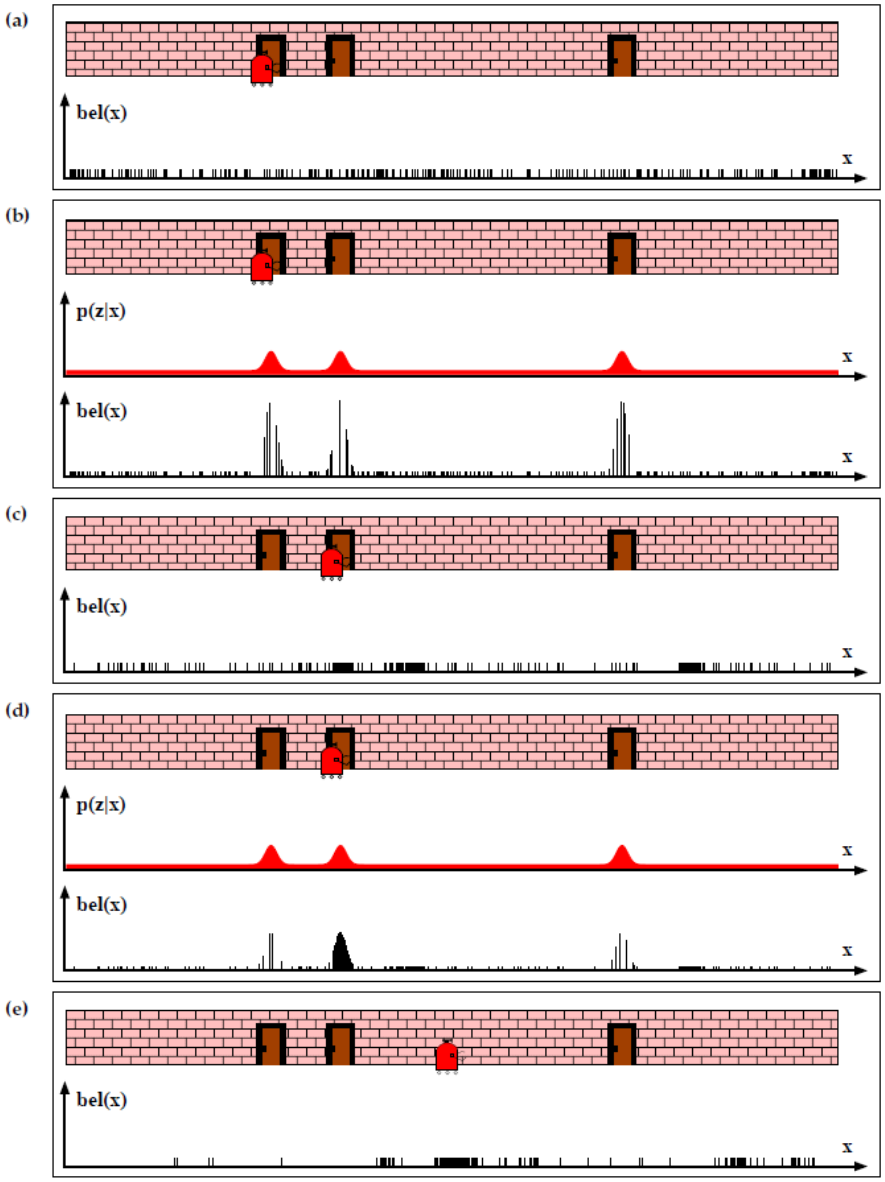

Abbildung 3.9: Monte Carlo Lokalisierung 1D. [25] S. 251

In der Robotik gibt es viele verschiedene Methoden die für die Pfad- und Bewegungsplanung verwendet werden können. Diese reichen von Potentialfeld-Methoden über D* (siehe [24]) bis hin zu „Rapidly Exploring Random Trees“ (siehe [10]). Die Navigationsplanung in FLEA erfolgt über den Planungsalgorithmus D*. Für die sichere Navigation werden die Kosten von Zellen rund um Hindernisse erhöht. Mit Hilfe dieser Kostenerhöhung lässt sich eine unfallfreie Navigation von A nach B sicherstellen, sofern die Hindernisse vom

Laserscanner erkannt werden und die Lokalisierung nicht fehlerhaft ist.

3.3.3 Planung und Ausführung

Der Roboter soll Objekte in einer Wohnung wieder finden. Um diese Aufgabe zu lösen muss FLEA einen Suchplan entwerfen. Dieser Suchplan besteht aus eine Serie von externen Orientierungen (EO). Jede EO ist eine Position mit einem Rotationswinkel, da sich der Roboter nur auf einer planaren Oberfläche fortbewegen kann. Die Anforderung an den Roboter, Objekte in einer Wohnung wiederzufinden kann ohne a-priori Wissen dazu führen, dass entsprechend viele EO geplant, befahren und abgesucht werden müssen. Die Aufgabenstellung hat die Anforderung die Wohnung innerhalb einer bestimmten Zeitspanne abzusuchen. Eine vollständige Suche würde sehr lange dauern. Um die gesamte Wohnung nach den Objekten abzusuchen, muss der Suchraum möglichst effizient partitioniert werden. Nachdem bei RoboCup@Home die Karte bereits vor dem Wettkampf zur Verfügung steht und frei manipuliert werden darf, kann ein Suchplan mit entsprechenden EO's a priori vorgegeben werden.

Der Planer entwirft einen Suchplan basierend auf einer Liste von vorgegebenen EO's. Diese EO's werden in einem manuellen Schritt bestimmt. Die Planung erfolgt so, dass beginnend von einem beliebigen Startpunkt die Suche über alle vorgegeben EO's läuft. Der Roboter fährt nacheinander alle EO's ab, dreht dort am Stand (unter Berücksichtigung des Öffnungswinkels der Kamera) und sucht solange bis entweder alle Positionen abgesucht worden sind oder eine Zeitgrenze überschritten wurde. Der Scheduler ist für die Koordinierung des Suchplans in FLEA zuständig und führt die geplanten Schritte aus. Basierend auf Prioritäten ähnelt der Scheduler einer Warteschlange, wobei mehrere Aufgaben (Navigation, Objekte erkennen, Spracherkennung, Logging, etc.) parallelisiert oder in Serie ablaufen können.

Alternativ gibt es Methoden, den Suchplan und die Aktionen dynamisch zu berechnen. Dies ist jedoch nicht Ziel dieser Arbeit und es sei auf [2] verwiesen.

Kapitel 4

FLEA Object Recognition System

In diesem Kapitel wird das für den Roboter FLEA zu implementierende Objekt-Erkennungs-System zur visuellen Suche beschrieben (ORS). Das ORS soll FLEA das automatische Lernen von neuen Objekten und das Wiederfinden dieser in einer Haushalts-Umgebung ermöglichen. Die Komponenten die in den vorigen Kapiteln beschrieben wurden, sind erforderlich, um überhaupt die in diesem Kapitel beschriebene Objekterkennung betreiben zu können. Das System soll sich der im Kapitel 2 vorgestellten Computer Vision Methoden bedienen.

Ein modulares Design des ORS soll die Integration zukünftiger Entwicklungen erleichtern. Es soll gezeigt werden, dass die Einbindung von 3D Tiefeninformation und mehrerer Ansichten die Erkennung von stückweise planaren Objekten verbessern kann. Das ORS gliedert sich in zwei Phasen: Objekte lernen und Objekte erkennen. In diesem Kapitel werden die Anforderungen und der Aufbau beider Phasen beschrieben und implizit die Software-Architektur erläutert.

Beim Design der Architektur wird Wert darauf gelegt, dass beide Phasen von ORS und auch andere Module des Roboters (z.B. Gesichtserkennung oder SLAM) die extrahierte Informationen wiederverwenden können. Aus diesem Grund wird ein Objekt „Vision Event“ (VE) eingeführt. Ein VE beinhaltet prozessierte Sensor-Daten. Ein VE enthält alle für visuelle Aufgaben benötigten Informationen, wie zum Beispiel die extrahierten Feature Vektoren, 3D Tiefeninformation und die rektifizierten Bilder der Stereokamera.

Das Design des ORS ist stark an die gegebene FLEA Hardware gebunden. Der derzeit verwendete QuadCore Prozessor des Roboters ermöglicht die effiziente Abarbeitung mehrerer Threads. Dies soll in dem Design von ORS eine tragende Rolle spielen. Parallelisierbare Aufgaben sollen zeitgleich in der Feature Extraktion Pipeline abgearbeitet

werden.

Die Pipeline zur Feature Extraktion ist in Abbildung 4.1 schematisch dargestellt. Zuerst wird ein Stereobild mit einer Auflösung von 1280x960 Pixel aufgenommen. Danach werden das linke und rechte Bild entzerrt. Das entzerrte Stereobild wird parallelisiert analysiert. Drei Threads kümmern sich um die Prozessierung der Daten. Die ersten beiden Threads extrahieren Features in der vollen Auflösung aus dem Stereobild. Der dritte Thread berechnet in reduzierter Auflösung (320x240) einen Stereo-Densematch und die entsprechenden 3D Koordinaten pro Pixel mit Tiefeninformation. Die Informationen werden anschließend auf Konsistenz überprüft. Das bedeutet, das extrahierte Features nur dann beibehalten werden, wenn es für ihre Position auch Tiefeninformation aus dem Stereo-Densematch gibt. Es wird unterstellt, das detektierte Features nur in texturierten Bereichen auftreten, welche durch das Stereo-Densematching mit Tiefeninformation ausgestattet werden können. Dies reduziert den Suchraum und verwirft automatisch Features, die zu geringe oder zu große Distanz von der Kamera aufweisen (Visual Attention). Die verbleibenden Features aus dem Referenzbild (rechte Stereo-Kamera), werden mit den Features aus dem linken Kamerabild verglichen und zusätzlich wird jedes verbleibende Feature mit einem Epipolar-Geometrie Check überprüft.

4.1 Objekte lernen

Das Lernen von Objekten soll für den Menschen so komfortabel wie möglich gestaltet werden. Um dies zu ermöglichen, ist es notwendig dass der Roboter Objekte weitgehend automatisch lernen kann. Der Roboter soll nur bei einem Fehler die Hilfe des Menschen in Anspruch nehmen müssen.

Im Zuge dieser Arbeit hat sich folgender Ansatz bewährt: Zuerst werden die Objekte mit einer Handykamera (oder Kompaktkamera) frontal fotografiert. Danach werden die Bilder manuell zugeschnitten, mit einem Namen versehen und dem Roboter übergeben. Diese vom Menschen vorverarbeiteten Bilder werden in Folge als Prior bezeichnet. Vorteil dieser Methode ist, dass speziell im Hinblick auf die RoboCup@Home Weltmeisterschaft (bei welcher bis zu 24 Teams gleichzeitig um die Objekte ringen) schnell und komfortabel eine Prior-Datenbank erstellt werden kann. So kann mittels Kompaktkamera ein Schnappschuss gemacht werden, ohne andere Teams zu blockieren.

Dieser manuelle Schritt ist notwendig, um den Roboter mitzuteilen, was er eigentlich lernen soll. Bei der SRVC werden die Bilder nicht manuell erstellt, sondern über das Internet geladen. Dieses Verfahren ist für diese Arbeit aber nicht notwendig, da bei

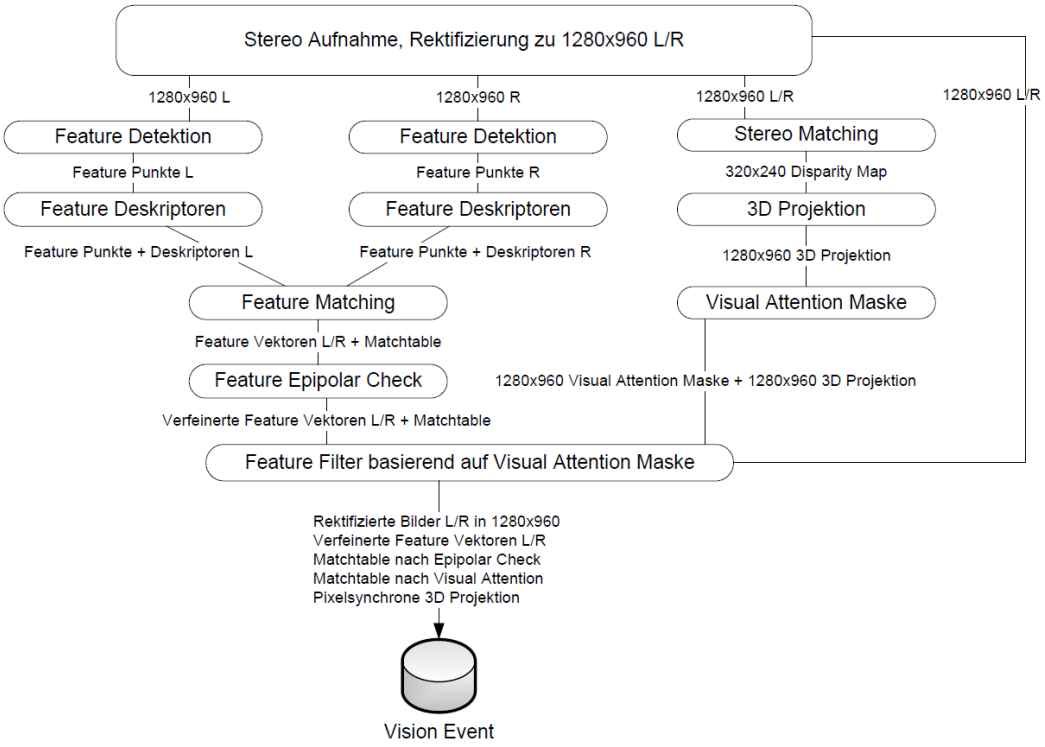

Abbildung 4.1: Feature Extraktion Pipeline. Die Pipeline verwendet Multithreading für die Berechnung der Feature Vektoren und Stereo Densematching. Die Vision Events (VE) sind das Resultat der Pipeline. Die VE werden für Objekt-Suche und Objekt-Lernen verwendet.

RoboCup@Home die Objekte schon vor dem Wettbewerb bekannt gegeben werden und physikalisch verfügbar sind.

Der Roboter soll unter Zuhilfenahme der Prior-Daten automatisch ein 2.5D Modell vom Objekt ableiten. Dieses Modell beinhaltet robuste 2D Features für die Objekterkennung als auch 3D Information (z.B. Größe) über das Objekt. Die 3D Information soll die Robustheit erhöhen und die Objektmanipulation in zukünftigen Anwendungen unterstützen.

Für das automatische Lernen der Objekte muss dem Roboter ein dedizierter Platz zur Verfügung stehen, der auch einen gewissen Freiraum für seine Bewegungen bietet. Die Erfahrungen bisherigen Meisterschaften haben dazu geführt den Team-Tisch als Lernumgebung zu nutzen, da dieser einer der wenigen Fixpunkte bei jeder Meisterschaft darstellt. Auch bieten die meisten Veranstaltungsorte genug Freiraum im Teambereich, damit der Roboter sich ein wenig bewegen kann.

Diese Arbeit setzt ein Vorhandensein vom Prior eines Objekts voraus. Gibt es den Prior nicht, meldet der Roboter, dass er das Objekt nicht lernen kann. Dieser Ansatz wurde

Abbildung 4.2: Mit einer Kompaktkamera (Canon IXUS) aufgenommene und beschnittene Prior Bilder für die Objekt-Datenbank dienen als Grundlage für das autonome einlernen der Objekte.

gewählt, da Objekte unter Wettkampf Bedingungen robust und ohne Risiko erlernbar sein sollen.

4.1.1 Schematischer Ablauf

Um ein Objekt vom Roboter lernen zu lassen, muss sich ein entsprechender Prior in der Datenbank befinden. Der Mensch muss dafür sorgen, dass das zu lernende Objekt an einer bestimmten Position am Tisch steht, damit der Roboter mit einer hohen Wahrscheinlichkeit das Objekt erkennen kann. Dem Roboter muss nur mitgeteilt werden, welches Objekt aus der Datenbank er nun lernen soll. Nachdem der Roboter den Lern-Befehl (z.B.: Lerne Objekt „Langkornreis“) vom Menschen erhalten hat, fährt dieser vordefinierte Positionen an. Von diesen Positionen aus lässt sich das Objekt aus verschiedenen Blickwinkeln und Abständen betrachten. An jeder Position versucht der Roboter das Objekt zu erkennen und sein Wissen über das Objekt zu erweitern. Abbildung 4.3 stellt den Ablauf und die Lernpositionen grafisch dar.

Nachdem der Roboter den Lern-Befehl vom Menschen erhalten hat, fährt dieser zuerst automatisch die vordefinierten Positionen „Pose 1“ bis „Pose 9“ an (siehe Abbildung 4.3). An jeder Position speichert der Roboter die Stereo-Bilder. An Position „Pose 1“ hat der Roboter einen mittleren Abstand zum Objekt (ca. 0,7m) und einen frontalen Blickwinkel. Die Lernposition 1 ist die vielversprechendste Position um das Prior-Bild mit dem Roboter Stereo-Bild abzugleichen und das Objekt in das metrische Roboterkoordinatensystem zu transformieren.

4.1.1.1 View 0

Der Lernvorgang kann am Beispiel des Objekts „Langkornreis“ erörtert werden: Die Abbildung 4.2 zeigt das Prior-Bild der Reispackung, welches mit einer Kompaktkamera auf-

genommen wurde und mit Hilfe eines einfachen Bildbearbeitungsprogramms auf den tatsächlichen Objektbereich zugeschnitten, skaliert und verlustfrei abgespeichert wurde.

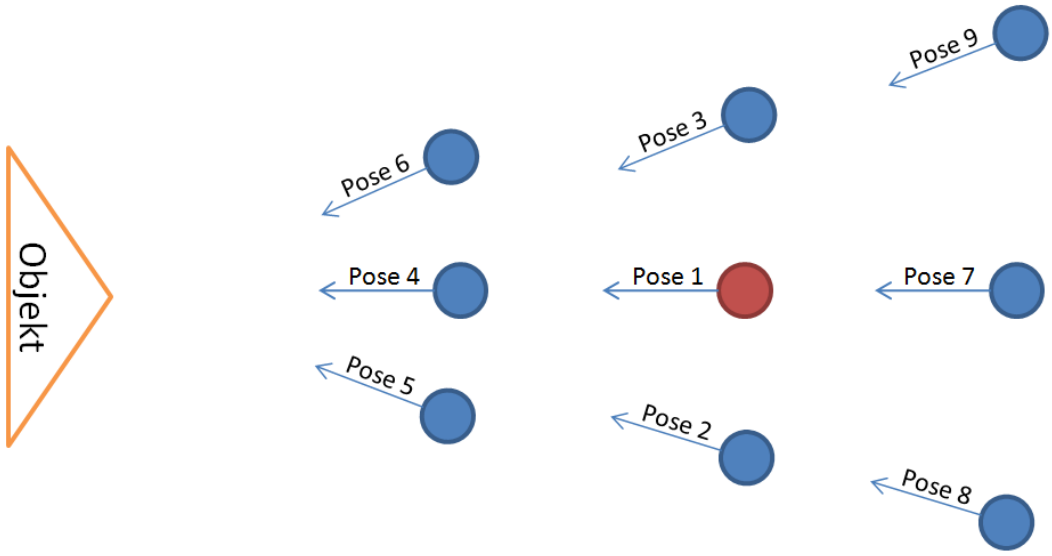

Abbildung 4.3: Lernpositionen die der Roboter nacheinander abfährt. Pose 1 bis 9 sind die Positionen an denen Stereo-Aufnahmen vom Objekt gemacht werden. Bei jeder Aufnahme hat das Objekt eine unterschiedliche Größe und Orientierung

Zuerst wird der „Langkornreis“ Prior vom Roboter geladen und es werden lokale Features extrahiert. Um sicherzustellen, dass genügend Features im Randbereich extrahiert werden, wird der Bildbereich rund um den Prior vergrößert und mit uniformen Rauschen gefüllt. [11] verwenden eine ähnliche Methode bei der Generierung von künstlichen Ansichten des zu lernenden Objekts. Die Vergrößerung des Randbereiches hat zur Folge, dass auch Features die sich am Rand des Objekts befinden extrahiert werden können.

Abbildung 4.4: Referenzbild (Links), Disparity Map (Mitte), Aufmerksamkeits-Maske (Rechts)

Nachdem sich der Prior mit extrahierten Features im Speicher befindet, werden die

Stereo-Bildpaare analysiert. Die Stereo-Bildpaare werden zuerst rektifiziert. Danach wird mittels Stereo-Densematching eine Disparity-Map errechnet, welche mit Hilfe der Kameraparameter in 3D Koordinaten umgerechnet wird (siehe 5.2). Es wird eine binäre Aufmerksamkeits-Maske („Visual Attention Maske“) erstellt, die den Lernbereich einschränkt. Mit Hilfe dieser Maske lassen sich effizient Features selektieren, die sich in einem bestimmten Bereich (0.3m bis 1.4m) vor der Referenzkamera (Stereosystem) befinden. Die entstandene Aufmerksamkeits-Maske ist in Abbildung 4.4 dargestellt.

Im rechten und linken rektifizierten Bild der Stereo-Kamera werden lokale Features extrahiert. Selbstverständlich wird das gleiche Verfahren wie beim Prior-Bild angewendet. Mittels Nearest-Neighbour-Matching werden korrespondierende Features zwischen beiden Bildern ermittelt. Die gematchten Features werden danach einem Epipolar-Geometrie-Check unterzogen um mögliche Outlier zwischen dem linken und rechten Bild zu eliminieren. Danach wird die Aufmerksamkeits-Maske auf die Features angewendet. Nur Features im Bereich der Maske überleben diesen Schritt. Eine grafische Übersicht der Pipeline zur Featureextraktion ist in Abbildung 4.1 ersichtlich.

Im nächsten Schritt werden Feature Korrespondenzen zwischen dem Referenzbild und dem Prior ermittelt. Durch die Verwendung eines Stereo-Kamerasystems ist es möglich für alle korrespondierenden Featurepunkte einen Tiefenwert zu bestimmen. Hier kann das Stereo-System wieder seine Stärken ausspielen, da durch die Verwendung von Tiefeninformation eine Reduktion der Falsch-Positiven Korrespondenzen erzielt werden kann. Zuerst wird der Median-Tiefenwert des Objekts über alle Korrespondenzen ermittelt. Features, deren Tiefenwerte eine zu hohe Varianz zum Median aufweisen, werden verworfen. Abbildung 4.5 zeigt einen Vergleich zwischen gefilterten und ungefilterten Korrespondenzen.

Abbildung 4.5: Vergleich der Outlier mit und ohne 3D Tiefenfilter. Der 3D Tiefenfilter reduziert die Outlier-Rate drastisch. Links: mit 3D Tiefenfilter. Rechts: ohne 3D Tiefenfilter

Da alle 9 Ansichten bereits vorprozessiert im Speicher sind, wird für View0 die Aufnahme an der Pose verwendet an der die stabilste Homographie zum Prior bestimmt werden kann. Dazu werden für alle aufgenommenen 9 Ansichten die Homographien auf das Prior Bild bestimmt. Es wird der View ausgewählt, der am meisten korrespondierende Features zum Prior aufweisen kann und als neuer View0 deklariert. Somit ist die Homographie vom Prior-Bild auf das Referenz-Bild ermittelt. Die Ermittlung der Homographie ist der entscheidende Schritt. Versagt die Homographie-Berechnung muss der Lernvorgang abgebrochen werden oder gegebenenfalls von vorne neu gestartet werden. Ein Versagen der Homographieberechnung ist jedoch selten, da in den meisten Fällen genügend Korrespondenzen vorhanden sind.

Die Qualität der Homographie muss unbedingt geprüft werden. Die Beobachtung der metrischen Objektgröße hat sich bei den Experimenten bewährt. Die Berechnung der Homographie hat versagt, wenn der resultierende Objektbereich in metrischen Einheiten keinen Sinn macht. Das Objekt ist entweder viel zu klein oder zu groß. Für die Experimente wurde definiert, dass ein Objekt größer als 3x3cm und kleiner als 100x100cm sein muss. Selbstverständlich handelt es sich hierbei um frei konfigurierbare Parameter.

Abbildung 4.6: V.l.n.r.: Prior Bild, extrahierte Features von Prior Bild, Eingabebild, Region im Eingabebild in der der Prior erkannt wurde, extrahierte Features aus der Region, Visualisierung der Features von View 0

Wenn die Homographie erfolgreich ermittelt werden kann und in 3D plausibel ist, wird eine auf der Homographie basierende Objektmaske im Referenzbild angefertigt. Der Bildinhalt wird ausgeschnitten und wieder von einem verrauschten Rahmen umgeben. Anschließend werden lokale Features aus dem Objektbereich extrahiert und noch einmal mit dem Prior-Bild verglichen. Diese Menge der Features stellt die Referenz-Objekt Feature-Menge dar und wird in Folge als View0 bezeichnet. Abbildung 4.6 zeigt den Vorgang am Beispiel des Objekts „Fruchtschnitten“.

4.1.1.2 View 1..n

Nachdem die Homographie erfolgreich bestimmt und ein Basis-Modell erzeugt wurde (View0), werden die Bilder mit anderen Blickwinkel zum Objekt analysiert. Ziel ist durch möglichst viele leicht unterschiedliche Ansichten die stabilsten m Features über alle n Views auszuwählen.

View1 bis n werden nacheinander mit View0 abgeglichen. Gelingt eine Homographie und liefert der 3D Check entsprechend korrespondierende Werte zu der bereits zuvor erhaltenen Objektgröße, kann der View0 um Features aus dem aktuellen View erweitert werden.

Wiedererkannte Features aus anderen Blickwinkeln werden in den View0 projiziert. Sollte die projizierte Position mit denen der Korrespondenz aus View0 übereinstimmen, wird die Repeatability-Score für dieses Feature im Objektmodell (das auf View0 basiert) inkrementiert. Um auch neue, zuvor unbekannte Features zum Objekt-Modell hinzuzufügen werden diese, wenn genügend robuste Korrespondenzen gefunden wurden, zum Objektmodell hinzugefügt. Dabei wird der Deskriptor des Features nicht verändert, sondern nur die Position des neuen Features in den View0 mittels der Homographie projiziert. Neue Features werden mit einer Score von 1 initialisiert.

Dieser Vorgang wird solange wiederholt bis alle Views analysiert worden sind. Bei jedem View wächst die Menge der Objekt-Features von View0. Nachdem alle 9 Views analysiert und die Features transferiert wurden, werden die Features im Objektmodell nach ihrer Score aufsteigend sortiert.

Es besteht somit die Möglichkeit nur die x robustesten Features auszuwählen. Dabei muss beachtet werden, dass auf Grund der Sortierung einzigartige Features (aus anderen Blickwinkeln) verloren gehen können. Eine Mindestmenge von 200 Features hat sich in den Tests als ausreichend herausgestellt. Abbildung 4.7 zeigt die Auswirkungen einer Ausdünnung der Objekt-Modell Features.

Abbildung 4.7: Objekt-Modell Features mit unterschiedlicher Ausdünnung. Von links nach rechts sind die 25, 50, 100 und 200 stabilsten Features visualisiert.

4.2 Objekte suchen

Die visuelle Suche nach Objekten hat das Ziel, die vorher gelernten Objekte in der Haushaltsumgebung robust wieder zu erkennen. Die Objekte werden von einem Menschen in der Wohnung platziert, sodass sie zumindest von einer bestimmten Position aus von der Kamera des Roboters gesehen werden können. Die Anzahl der falsch positiven Erkennungen ist so gering wie möglich zu halten, da dies zu Punkteabzügen während des Wettbewerbs führt.

Generell ist eine visuelle Suche sehr zeitaufwändig, da die gesamte Umgebung abgefahren und analysiert werden muss. Zwei Faktoren beeinflussen die visuelle Suche maßgebend: die Winkelauflösung der Kamera und die Prozessierungsgeschwindigkeit des ORS.

Da die Bumblebee XB3 Stereo Kamera mit Weitwinkel-Objektiven ausgestattet ist, beträgt die Winkelauflösung pro Pixel lediglich 0.052°. Zum Vergleich besitzen aktuelle Smartphones eine Winkelauflösung von 0.023° pro Pixel (5MP Kamera und 60° HFOV). Um die Kurzsichtigkeit des Roboters zu kompensieren, sollen möglichst viele verschiedene Posen angefahren werden.

Eine Evaluierung der Geschwindigkeit der in Kapitel 2 vorgestellten Computer Vision Methoden soll eine Entscheidung für eine Methode ermöglichen. Dabei liegt das Entscheidungskriterium in der Echtzeitfähigkeit der Methode. In unserem Fall bedeutet dies, dass die gesamte bildbasierte Analyse während der Objekt-Suchen Phase unter einer Sekunde stattfinden muss. Die Genauigkeit der Methode muss ausreichend sein, um das Objekt robust erkennen zu können. Für die Evaluierung siehe Kapitel 5.

Beim Start der visuellen Suche werden alle zu suchenden Objektmodelle geladen und in einem Random Forest gespeichert.

4.2.1 Integration ins Gesamtsystem

Die Objekt Suche erfordert eine nahtlose Integration des ORS mit dem Robotersystem. Um bei den schwierigen Lichtverhältnissen im Innenbereich noch gute Bilder aufnehmen zu können, muss die Belichtungszeit der Kamera entsprechend hoch gewählt werden (max. 50ms). Die Wahrscheinlichkeit, dass Motion Blur entsteht, sobald der Roboter auch nur minimale Bewegungen durchführt, ist hoch. Somit erfordert die visuelle Suche, dass der Roboter zum vollkommenen Stillstand gekommen ist, bevor ein Bild aufgenommen wird. In weiterer Folge ist zu beachten, dass die Belichtungsregelung der Kamera sich den ständig veränderten Bedingungen anpassen muss. Aus diesem Grund läuft die Kamera im FreeRun Modus und regelt die Belichtung automatisch. Die Kamera nimmt 15 Bilder pro Sekunde

auf und der Roboter wartet, sobald er still steht auf das nächste Bild von der Kamera. So kann Motion Blur erfolgreich bei statischen Szenen vermieden werden. Dies ist akzeptabel, da unterstellt wird, dass sich die zu suchenden Objekte nicht bewegen.

Mit Hilfe der Umgebungskarte des Roboters kann der Suchraum partitioniert werden. Da bei RoboCup@Home die Umgebung im Voraus bekannt ist, ist es nicht notwendig eine automatische Auswahl der Suchposition zu implementieren. Es genügt eine Liste an Suchpositionen, die in einem manuellen Schritt vom Menschen erstellt wird. Der Planer des Roboters orientiert sich anhand dieser Liste und sucht in der Listen-Reihenfolge die Umgebung ab. Der Benutzer hat die Möglichkeit für jede Position bestimmte Suchmuster festzulegen. So gibt es Positionen (z.B. Mitte eines Raumes) bei denen der Roboter sich um 360° drehen soll und alle 30° das Kamerabild analysiert. Suchpositionen die sich nahe einer Wand befinden werden meist mit weniger Rotationen auskommen. Die implementierte manuelle Methode kann in zukünftigen Versionen durch eine automatische Suchpositions-Erkennung ersetzt werden. Die Integration eines Verfahrens, wie in [15] beschrieben kann durchgeführt werden, da der Roboter eine solche Software-Flexibilität aufweist. Im Zuge dieser Arbeit wird aber darauf verzichtet, da dieses Unterfangen mit einem großen Implementierungs- und Testaufwand verbunden ist und es den Rahmen der Diplomarbeit sprengen würde.

Bei beiden Phasen läuft die Kamera frei mit 15Hz und kann je nach Bedarf Bilder zur Analyse an das ORS weiterleiten. Die Weiterleitung der Bilder an das Visionsystem wird vom Scheduler getriggert. Der Scheduler ist konzipiert, bestimme Aufgaben („Job“) und eingebettete Aufgaben („Tasks“) parallel oder in Serie zu erledigen. Der Job „Objekt Lernen“ besteht zum Beispiel aus den Tasks Navigation, Feature Extraktion Pipeline und Objekt Modell Generierung.

4.2.2 Objekt Erkennung

Die Objekt-Erkennung während der Test Phase benötigt als Eingabedaten ein Stereo-Bild und die bereits trainierten Objekt-Modelle. Beim Start des Jobs werden die zu suchenden Objekte von der SSD des Roboters geladen und eine RKF-Tree Struktur trainiert. Nachdem der Roboter eine Such-Position angefahren hat, wird ein Stereo-Bild aufgenommen und nach dem Schema in Grafik 4.1 abgearbeitet. Zuerst werden die Bilder Stereo rektifiziert und anschließend parallel in Threads abgearbeitet. Es werden Features vom linken und rechten Bild extrahiert und verglichen sowie ein Stereo-Densematch berechnet. Basierend auf den 3D Daten wird eine Maske erstellt, die Bereiche im visuellen Wahrneh-

mungsbereich des Roboters maskiert. Features, die im linken und rechten Bild vorkommen, werden geprüft ob sie auf einer Epipolarlinie liegen. Das entstandene Vision Event (VE) beinhaltet alle Informationen, die für die Anfrage an die Objektdatenbank benötigt werden. Zuerst werden die Features des VE, welche die Epipolar Prüfung im maskierten Bereich überlebten, gegen den beim Start trainierten RKD-Tree abgeglichen. Dieser Abgleich liefert Hypothesen über die möglichen Objekte, die sich im Bild befinden. Basierend auf den vom VE zum Objektmodell gematchten Feature zu Feature Korrespondenzen wird eine Homographie bestimmt, wenn sich genug Inlier gefunden haben. Über diese Homographie lässt sich die Größe des Objekts bestimmen. Stimmt die Objekt-Dimension mit der in der Objektdatenbank befindlichen Information überein, wird das Objekt als gefunden gemeldet.

Kapitel 5

Evaluierung

Die Evaluierung fokussiert sich auf robuste Objekterkennung und Navigation in einer normalen Haushaltsumgebung. Die im Kapitel 3 beschriebenen Module der visuellen Suche werden in diesem Kapitel integriert und einem Systemtest unterzogen. Ziel ist es, das Robotersystem mit einem RoboCup@Home „Lost and Found“ ähnlichen Test zu evaluieren. Das Ziel der Evaluierung ist herauszufinden ob das Objekterkennungssystem in der Lage ist Objekte innerhalb einer vorgegebenen Zeit richtig wiederzufinden. Weiters soll untersucht werden, welche Auswirkungen die MultiView Variante gegenüber der SingleView Variante in Bezug auf die Robustheit der Objekterkennung hat.

Bei RoboCup@Home „Lost and Found“ werden aus einer bekannten Menge von Objekten von Schiedsrichtern (Teamleiter anderer Teams) drei Objekte ausgewählt, die anschließend in der Umgebung versteckt werden. Wichtig ist, dass die Objekte fair versteckt werden. Das bedeutet, dass der Roboter die Möglichkeit haben muss die Objekte zumindest aus einer für ihn erreichbaren Position einsehen zu können. Auch muss die Montagehöhe der Kamera berücksichtigt werden. Im Falle von FLEA (Kamera befindet sich 90cm über dem Boden) bedeutet dies, dass Objekte nicht am Boden liegen oder zu hoch in Regalen platziert sein dürfen. Diese drei Objekte werden eine Stunde bevor der Test beginnt bekanntgegeben. Es bleibt somit genug Zeit um den Roboter entsprechend auf den Test vorzubereiten und die zu suchenden Objekte zu spezifieren.

5.1 Einzeltests

In den folgenden Unterkapiteln soll untersucht werden, welche Auswirkungen verschiedene Methoden auf Genauigkeit und Geschwindigkeit der bildbasierten Analyse haben. Die

Detektor:	SIFT	SURF-CPU	CenSurE (+SURF)	MSER (+SURF)
Zeit(ms)	4800 ms	1329 ms	328 ms	606 ms
Features	1286	2622	1441	740

Tabelle 5.1: Evaluierung der Berechnungszeit von Feature Extraktoren (mit Deskriptor) auf einem 1280x960 Bild

in der Abbildung 4.1 dargestellten Module werden analysiert. Autonome Robotersysteme müssen in Echtzeit funktionieren und es ist aus diesem Grund notwendig die Geschwindigkeit der einzelnen Module auf der Zielplattform zu evaluieren.

5.1.1 Feature Extraktion

Die im Kapitel 2.2.1 vorgestellten Feature Extraktoren werden auf ihre Echtzeitfähigkeit evaluiert. Als Testbild wird ein entzerrtes Kamerabild mit einer Auflösung von 1280x960 verwendet, das als Grundlage für die Geschwindigkeitsermittlung dienen soll. Das Bild zeigt eine Szene mit mehreren Objekten, wie es auch bei der Suchphase des Roboters vorkommt.

SIFT, SURF, MSER und CenSurE werden gegeneinander verglichen. Für den Vergleich wurde die Lowe Implementierung von SIFT und die OpenCV Implementierungen von SURF, MSER und CenSurE herangezogen. Bei SURF, MSER und CenSurE wird der SURF Deskriptor mit 64D verwendet. Das Ergebnis der Evaluierung ist in Tabelle 5.1 ersichtlich. Eine Visualisierung der extrahierten Features ist in 5.1 ersichtlich.

Die Evaluierung zeigt, dass SIFT und SURF auf der Zielplattform nicht eingesetzt werden können, da ihre Berechnungszeiten über einer Sekunde liegen. MSER benötigt fast die doppelte Berechnungszeit von CenSurE und liefert nur die Hälfte an Features. Mit einer Berechnungszeit von 328ms für ein 1280x960 Bild ist CenSurE der klare Gewinner in diesem Einzeltest. Da die Geschwindigkeit der Extraktion das Hauptentscheidungskriterium ist, fällt die Entscheidung somit auf den CenSurE Feature Extraktor.

Als weitere Motivation für CenSurE steht die Wiederverwendung der Features für visuelle Odometrie oder SLAM. Die geringe Berechnungszeit wirkt sich positiv auf die gesamte Pipeline aus.

5.1.2 Feature Matching

Die extrahierten Featurevektoren werden beim Matching mit zuvor gespeicherten Featurevektoren verglichen. Wie in Kapitel 2.2.3 beschrieben, gibt es verschiedene Methoden um den Vergleich effizient zu berechnen. Die von [16] entwickelte Methode wird in einem Test

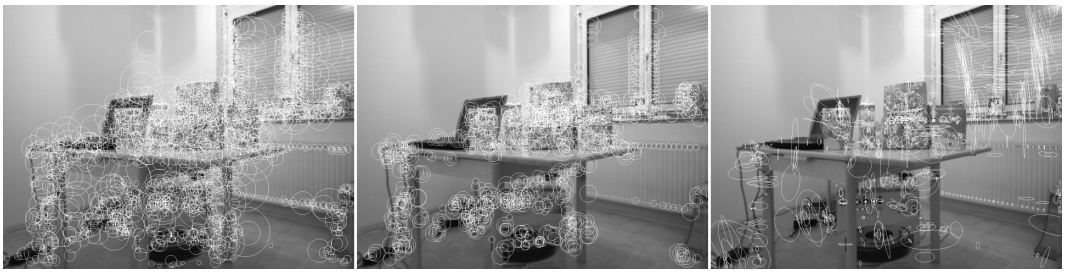

Abbildung 5.1: Visualisierung von SURF, CenCurE und MSER Features

	Deskriptoren	Brute Force	Random Forest
1 Objekt	200	37,5 ms	29,2 ms
2 Objekte	389	60,4 ms	34,3 ms
3 Objekte	589	85,4 ms	33,3 ms
4 Objekte	798	130,7 ms	29,8 ms
5 Objekte	989	156,6 ms	29,8 ms
6 Objekte	1189	196,8 ms	29,3 ms
7 Objekte	1389	218,5 ms	27,3 ms
8 Objekte	1580	278,4 ms	26,3 ms

Tabelle 5.2: Evaluierung der Berechnungszeit von Feature Matching (Brute Force versus Random Forst). Die Objekt Datenbank wird auf 191 Eingabe-Deskriptoren geprüft. Als Random Forest wurde die OpenCV Implementierung von [16] herangezogen. Die Forest Konfiguration ist: 16 Trees, 128 Suchtiefe

mit der Bruteforce Methode verglichen. Der Test wird auf der Roboter Entwicklungsplattform durchgeführt. Der Test soll zeigen welche Geschwindigkeitsvorteile die Verwendung der approximierten NN Suche gegenüber der Brute Force Methode hat.

Die in der Tabelle 5.2 dargestellten Ergebnisse zeigen deutlich, dass die Verwendung der ANN Suche deutliche Geschwindigkeitsvorteile bietet. Die Geschwindigkeit wurde auf einer zuvor trainierten Objektdatenbank ermittelt. Die Ergebnisse zeigen einen direkten Vergleich der Evalierungszeit der Deskriptoren. Die ANN Suche bringt eine massive Geschwindigkeitssteigerung gegenüber der Brute Force Suche.

Da bei der Brute Force Suche kein vorheriges Training erforderlich ist, muss die (einmalige) Aufbauzeit für die ANN Datenstruktur berücksichtigt werden. Da der Aufbau der Datenstruktur während des Objektlernens passiert, ist nur wichtig, dass dieser Prozess in einer für den Menschen akzeptablen Zeit (zB Sekunden) passiert. Die Tabelle 5.3 zeigt die Berechnungszeit für den Aufbau der ANN Datenstruktur mit unterschiedlich vielen Objekten und Deskriptoren. Das Training des Klassifikators ist mit wenigen Objekten sogar in Echtzeit möglich.

	Deskriptoren	Trainingszeit
1 Objekt	200	130,1 ms
2 Objekte	389	144,5 ms
3 Objekte	589	174,3 ms
4 Objekte	798	230,6 ms
5 Objekte	989	274,0 ms
6 Objekte	1189	334,6 ms
7 Objekte	1389	332,3 ms
8 Objekte	1580	457,8 ms

Tabelle 5.3: Evaluierung der Trainingszeit von Random Forests auf der Objektdatenbank. Als Random Forest wurde die OpenCV Implementierung von [16] herangezogen. Die Forest Konfiguration ist: 16 Trees, 128 Suchtiefe

5.1.3 Stereo-Densematch

Die Erstellung einer pixelsynchronen Tiefenmaske liefert dem System implizit Informationen über Bildbereiche die Texturinformation enthalten. Diese Eigenschaft ist für das Objekterkennungssystem besonders gut geeignet, da sich damit der Suchraum im Bild erheblich einschränken lässt. Der effektive Erkennungsbereich des Systems beträgt 0,3 bis 1,4 Meter. Um den Suchraum einzuschränken soll ein Stereo-Densematch texturierte Bildbereiche identifizieren und diese mit metrischer Tiefe zurückliefern. Das Ergbnis des Densematch ist eine Disparity Map, ein zum Eingabebild pixelsynchrones Tiefenbild. Diese Disparity Map kann mit Hilfe der Kamera und Systemkalibrierung in metrische Werte (x,y,z) umgerechnet werden. Anschließend kann ein Binärbild erstellt werden, bei dem Bereiche ausmaskiert sind die ausserhalb des Erkennungsbereiches liegen. Das erhaltene Binärbild wird als Aufmerksamkeits-Maske bezeichnet.

Abbildung 5.2: Rechtes Eingabebild, Disparity Map, 3D Rekonstruktion

Um die Berechnungszeit für den Densematch zu minimieren, werden nicht die hochauflösenden Bilder der Kamera verwendet. Die rektifizierten Stereo-Eingabebilder werden

Szene	Objektdistanz	Orientierung	MultiView Inlier	SingleView Inlier
Szene 1	0.66m	-1.82°	59	59
Szene 2	0.71m	9.97°	60	48
Szene 3	0.62m	-9.97°	53	35
Szene 4	0.45m	3.02°	14	11
Szene 5	0.41m	13.31°	48	21
Szene 6	0.34m	-13.02°	24	13
Szene 7	0.97m	-1.79°	37	30
Szene 8	1.25m	18.35°	22	10
Szene 9	1.05m	-18.31°	35	27

Tabelle 5.4: Vergleich zwischen SingleView und MultiView Variante. Die MultiView Variante ist deutlich robuster als die SingleView Variante.

auf eine Auflösung von 320x240 Pixel verkleinert. Die Verminderung der Auflösung bringt Geschwindigkeitsvorteile und die Genauigkeit der Rekonstruktion ist ausreichend um dem Systemzweck zu genügen. Die Berechnungszeit des Stereo-Densematch benötigt lediglich 31 ms mit dem PointGrey Triclops Algorithmus. Die Abbildung 5.2 zeigt das rechte Eingabebild (320x240), die zugehörige Disparity Map und die metrische 3D Rekonstruktion. Der PointGrey Algorithmus erfüllt die Echtzeit Anforderung und liefert ausreichende Ergebnisse für die Erstellung einer Aufmerksamkeits-Maske.

5.1.4 Single versus Multiple Views

Die Einbindung mehrerer verschiedener Ansichten eines Objekts („Multiple Views“) soll die Robustheit der Wiedererkennung steigern. Im Zuge der Diplomarbeit wurde ein Experiment durchgeführt bei dem zuerst nur eine Ansicht („SingleView“ oder View0) für das Objekt-Modell verwendet wird. Danach werden 9 Test-Positionen angefahren um die Robustheit zu testen. Um eine Gegenüberstellung zu der MultiView Variante zu ermöglichen, wird das Experiment anschließend wiederholt und alle 9 Ansichten für das Objekt-Modell verwendet. Nachdem das Modell erstellt wurde werden die gleichen 9 Test-Positionen wie bei dem SingeView Test angefahren und ausgewertet. Die Ergebnisse in Tabelle 5.4 sprechen klar für die Verwendung der MultiView Variante. Die Abbildung 5.3 stellt das Ergebnis noch einmal grafisch zur besseren Übersicht dar.

5.2 Gesamtsystem-Evaluierung

Der Suchbereich der im Rahmen dieser Arbeit durchgeführten Tests muss aus Platzgründen auf einen Raum mit ca. $25m^2$ Fläche eingeschränkt werden (siehe Abbildung 5.4).

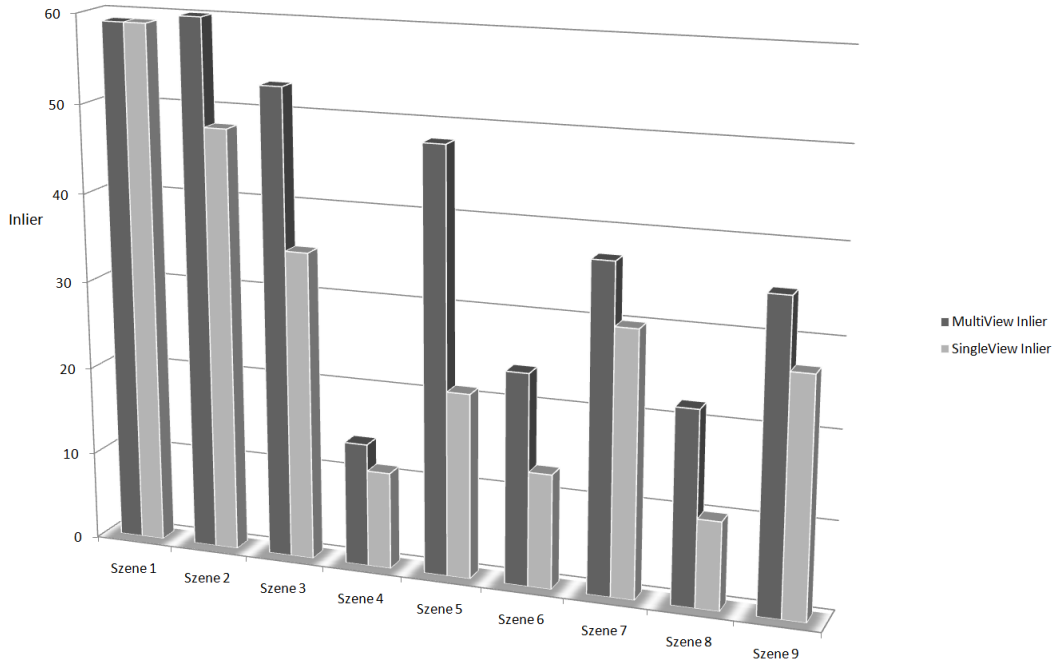

Abbildung 5.3: Grafische Darstellung der Inlier von MultiView und SingleView Variante. Die MultiView Variante ist deutlich robuster als die SingleView Variante.

Abbildung 5.4: Umgebung in der die Evaluierung durchgeführt wird.

Diese Umgebung ist somit in etwa um die Hälfte kleiner als die Umgebung bei der RoboCup@Home Weltmeisterschaft. Um die kleinere Fläche zu kompensieren, werden statt wie bei RoboCup@Home üblich nicht drei Objekte gesucht, sondern sechs Objekte. Für den Test werden die Objekte „Superplus“, „Obstriegel“, „Medisana“, „Fruchtschnitten“, „Langkornreis“ und Medolino“ in der Versuchsumgebung platziert (siehe Abbildung 5.6). Die Laufzeit des Tests, also die Zeit die der Roboter benötigt bis alle Objekte gefunden wurden, muss unter fünf Minuten liegen. Die Objekte müssen vom Menschen so positioniert werden, dass der Roboter zumindest aus einer bestimmten Position einen uneingeschränkten Blick auf das Objekt hat. Der Roboter greift im Zuge der Evaluierung auf die schon in Unterkapitel 4.1 trainierten Objektmodelle zu. Der Roboter arbeitet fehlerfrei, wenn er

alle sechs Objekte in der Testumgebung wiederfinden kann.

Der Roboter startet an einer vordefinierten Startposition. Grund hierfür ist, dass bei der Weltmeisterschaft immer zwei Teams simultan den Test absolvieren müssen. Der Roboter, der zuerst alle Objekte korrekt identifizieren kann und innerhalb von fünf Minuten wieder an die Startposition zurückkehrt, gewinnt den Test. Im Zuge der Diplomarbeit war es nicht möglich den Test „competitive“ zu gestalten, da derzeit kein anderer Roboter als Gegner zur Verfügung steht.

Für die Evaluierung ist besonders die Anzahl der richtig ausgemachten Objekte sowie die Anzahl der falsch erkannten Objekte wichtig. Es soll untersucht werden, wie viele Inlier bei der Erkennung eines Objekts vorhanden sind. Dies wird die Frage klären, ob eine Robustheitssteigerung durch MutipleViews in einer realen Testumgebung erreicht werden kann.

Wie zuvor in Unterkapitel 3.3.1 beschrieben, verfügt der Roboter über eine metrische Karte (GSD: 1px = 0.02m) der Umgebung. Die Karte ist als sogenannte „Occupancy Grid Map“ organisiert. Die Karte von FLEA enthält hohe Werte in Bereichen wo Hindernisse, wie zum Beispiel Wände und Tischbeine, zu erwarten sind. Abbildung 5.5 zeigt solche Bereiche in weißer Farbe, schwarz sind befahrbare Regionen. Auch Bereiche die vom Roboter gemieden werden sollen, können bestimmt werden (blaue Markierungen in Abbildung 5.5).

Ausgehend von der vordefinierten Startposition fährt der Roboter Positionen an, von denen aus er mit hoher Wahrscheinlichkeit ein Objekt sehen kann. Eine Position („Pose“) ist das Tripel $\{x, y, \phi\}$, bestehend aus der 2D Position $Pos = \{x, y\}$ in Bezug zur „Occupancy Grid Map“ mit Orientierung ϕ. Um die in Abbildung 5.4 gezeigte Umgebung abzusuchen sind 5 Positionen (siehe Abbildung 5.5) anzufahren und auszuwerten. An jeder Position rotiert der Roboter um einige Grad um so möglichst viel von der Umgebung zu erfassen.

Ein Objekt gilt bei RoboCup@Home als gefunden, wenn der Roboter sich dem Objekt auf einen halben Meter annähert und den Objektnamen über eine Sprachausgabe verlautbart. Für die Diplomarbeit wurde diese Regelung geändert, da der Roboter im „Debug“ Modus betrieben wird und in diesem Modus Informationen protokolliert. Während des Tests erstellt der Roboter eine Liste der Objekte, die er während der Suche erkennt. Die Liste enthält Informationen über die Roboter-Position an der das Objekt erkannt wurde, um welches Objekt es sich handelt, welche Entfernung es vom Roboter aufweist und wie viele Inlier (korrekte Feature Korrespondenzen) in Bezug auf das Objektmodell gefunden wurden. Zusätzlich speichert FLEA Debug-Bilder mit visualisierten Feature Korrespon-

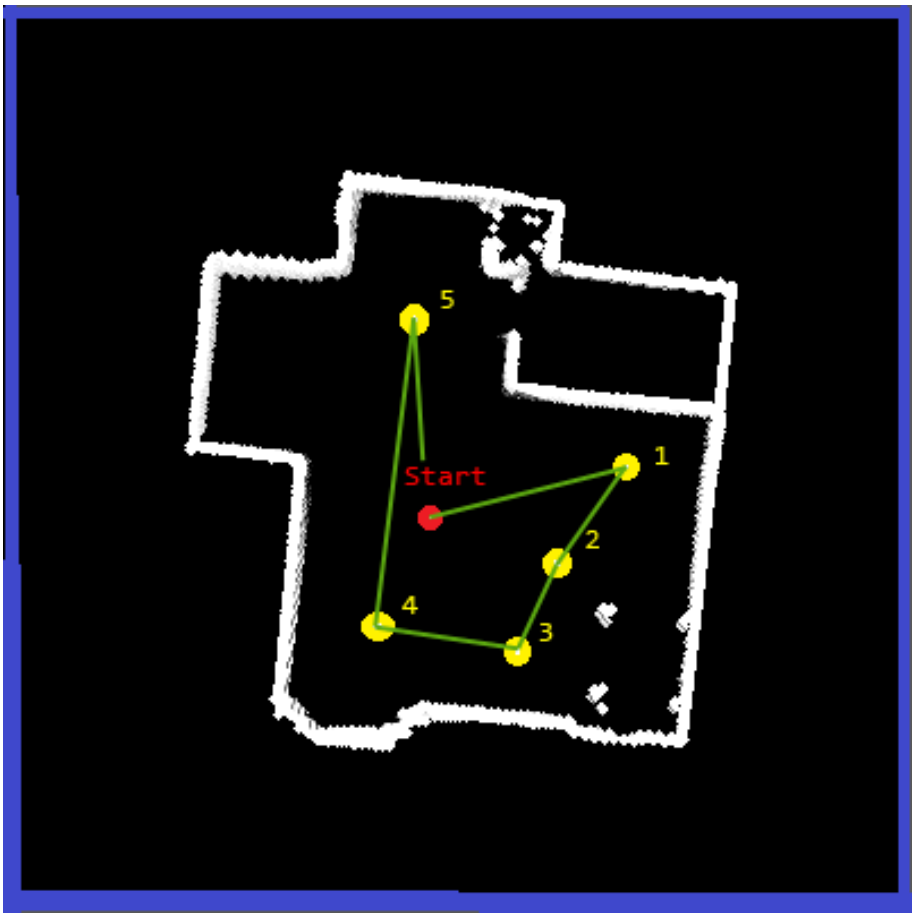

Abbildung 5.5: 2D Karte („Occupancy Grid Map“) die zur globalen Orientierung benutzt wird. Beginnend bei der Startposition (rot) verfolgt der Roboter einen Pfad (grün) der ihn über die fünf weiteren Suchpositionen (gelb) führt.

denzen.

Zu Beginn des Tests befindet sich der Roboter an einer vordefinierten Startposition (zu der er auf jeden Fall zurückkehren muss) und bekommt den Befehl die visuelle Suche zu starten. Die Objekte befinden sich bereits in der Umgebung und der Roboter hat ab jetzt maximal fünf Minuten Zeit die Umgebung abzusuchen.

Die Evaluierung besteht aus zwei Versuchen, wobei beim zweiten Versuch das Objekt „Medisana“ leicht in seiner Position verändert wird. Zwei Durchläufe sollen zeigen wie gut die Wiederholbarkeit ist und wie sich nur minimale Positionsänderungen auf die Objekterkennung auswirken. Ein Versuch besteht in sich aus zwei Wiederholungen, da einmal die Suche mit MultiView Objektmodellen durchgeführt wird und bei der zweiten Wiederholung mit der SingleView Variante.

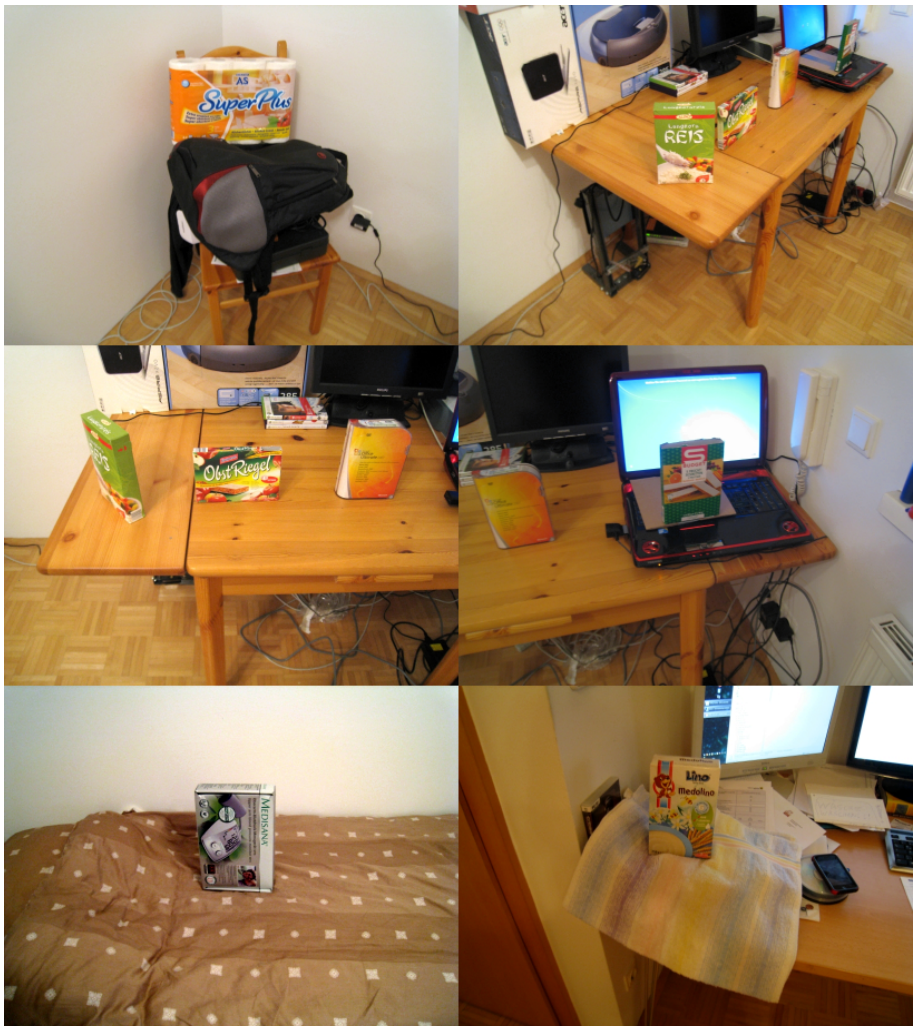

Abbildung 5.6: Sechs Objekte für die visuelle Suche: „Superplus“, „Obstriegel“, „Medisana“, „Langkornreis“, „Fruchtschnitten“ und Medolino“

5.2.1 Versuch 1

Bei jedem Versuch wird die visuelle Suche zwei Mal durchgeführt. Zuerst wird die MultiView Variante gestartet. Der Roboter fährt alle Suchpositionen an, rotiert an jeder Position und analysiert die Bilder mit dem Objekterkennungssystem. Innerhalb von 3 Minuten und 29 Sekunden findet der Roboter alle Objekte in der Umgebung. Es kommt zu keiner falsch positiven Erkennung. Weiters werden Objekte teilweise von mehreren Positionen aus erkannt.

Bei der Wiederholung werden anstatt der MultiView Objektmodelle die SingleView Objektmodelle geladen. Der Roboter benötigt 3 Minuten 30 Sekunden für die visuelle

Suche. Es kommt auch hier zu keiner falsch positiven Erkennung. Allerdings erkennt die SingleView Variante nur fünf der sechs Objekte. Die Feature Korrespondenzen für das Objekt „Medisana“ können nicht richtig zugeordnet werden. Abbildung 5.10 zeigt die falschen Zuordnungen zwischen Objektmodell und Kamerabild. Das Objekterkennungssystem kann in diesem Fall die Homographie nicht richtig bestimmen und retourniert mit einem negativen 3D Plausibilitäts-Check.

5.2.1.1 Position 1

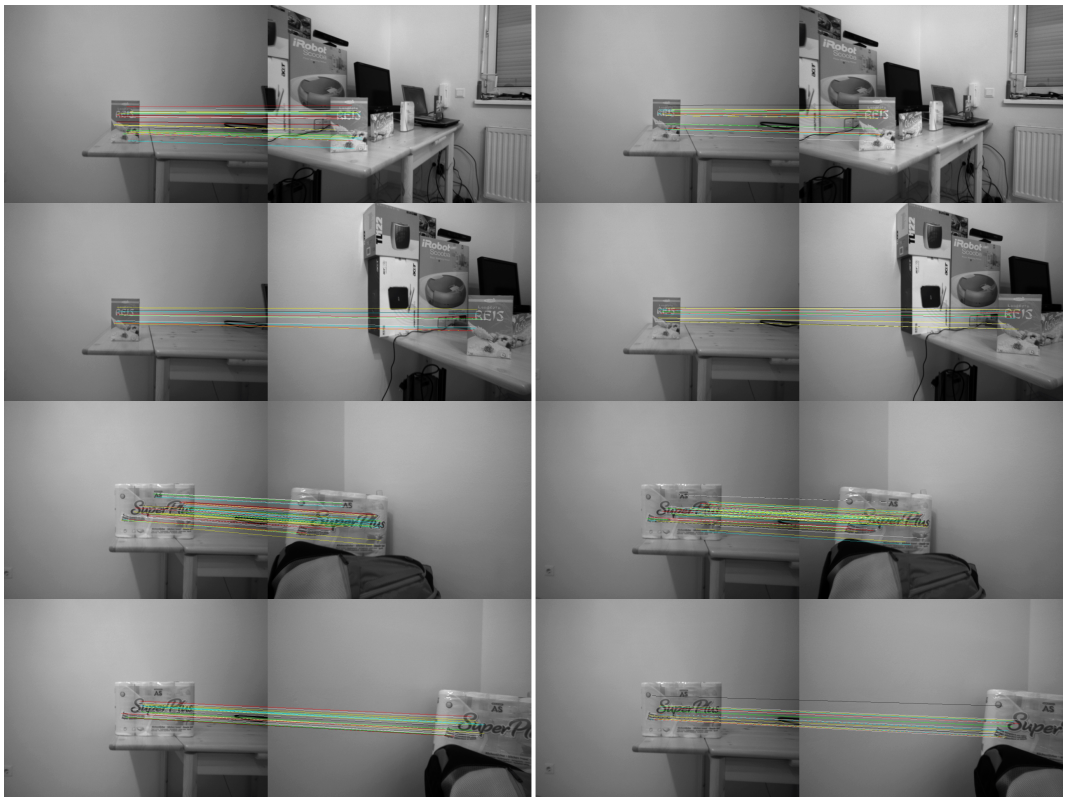

Abbildung 5.7: Versuch 1. Position 1. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante.

An der Suchposition 1 findet der Roboter alle Objekte mit hoher Zuverlässigkeit. Dies gilt sowohl für die MultiView als auch für die Singleview Variante. Abbildung 5.7 zeigt die Feature Korrespondenzen der erkannten Objekte.

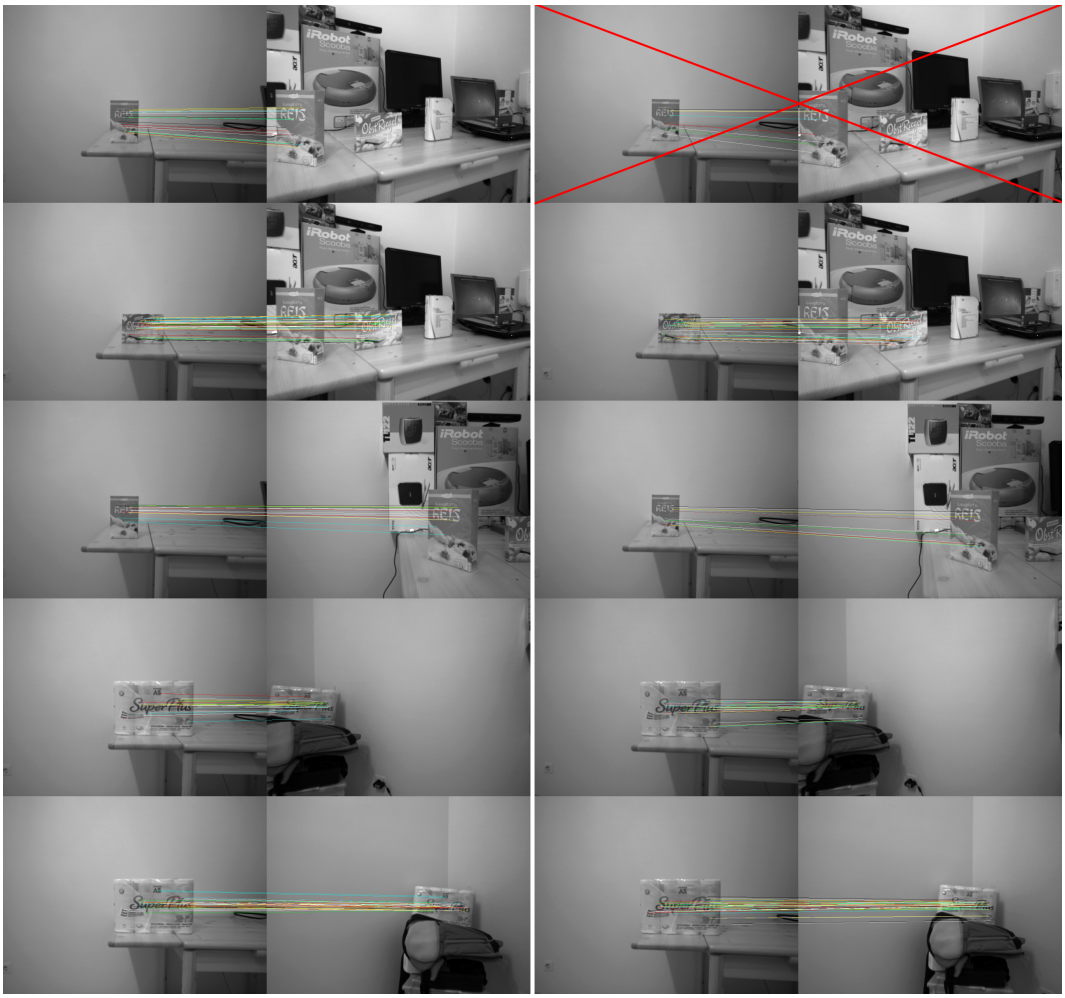

Abbildung 5.8: Versuch 1. Position 2. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante. Das nicht erkannte Objekt ist rot durchgestrichen.

5.2.1.2 Position 2

An der Suchposition 2 findet der Roboter alle Objekte. Die MultiView Variante findet bei einer Ansicht ein Objekt mehr und arbeitet generell mit hoher Zuverlässigkeit. Die SingleView Variante kann an dieser Position alle Objekte lokalisieren, funktioniert aber ein wenig schlechter als die MultiView Variante. Abbildung 5.8 zeigt die Feature Korrespondenzen der erkannten Objekte. Das Objekt „Langkornreis“ wurde in einer Ansicht der SingleView Variante nicht erfolgreich erkannt und ist rot durchgestrichen.

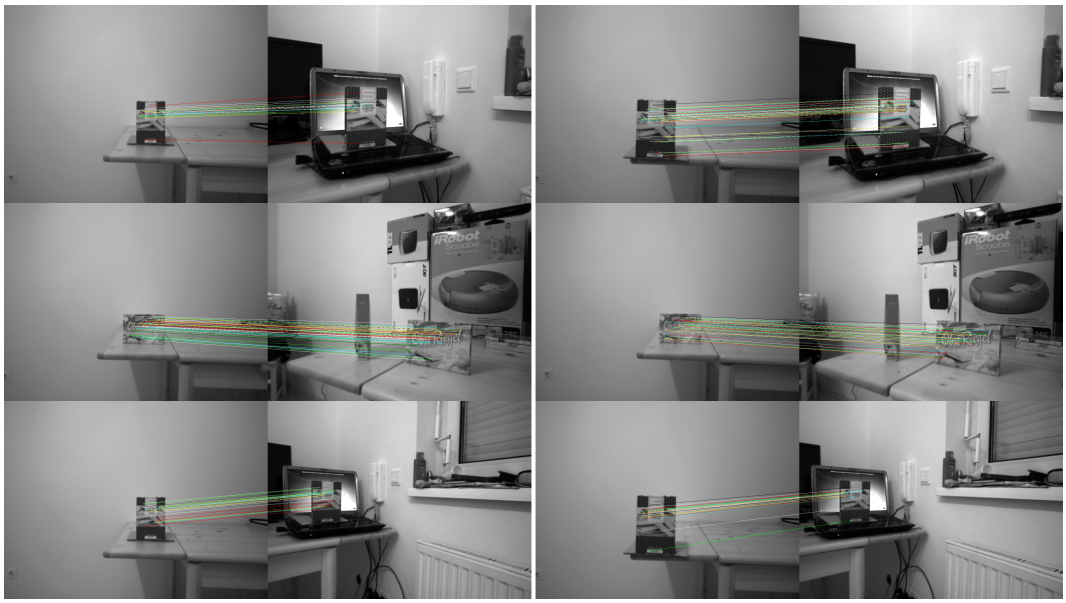

Abbildung 5.9: Versuch 1. Position 3. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante.

5.2.1.3 Position 3

An der Suchposition 3 findet der Roboter alle Objekte mit hoher Zuverlässigkeit. Dies gilt sowohl für die MultiView als auch für die Singleview Variante. Abbildung 5.9 zeigt die Feature Korrespondenzen der erkannten Objekte.

5.2.1.4 Position 4 und 5

Suchposition 4 ist eine Herausforderung für den Roboter. Die Multiview Variante kann das Objekt „Medisana“ richtig lokalisieren (siehe Abbildung 5.10). Bei der SingleView Variante sieht das Ergebnis anders aus: die Objekterkennung versagt schon bei den Feature Korrespondenzen. Die Distanz zum Objekt (1.33m) ist zu hoch für die SingleView Variante und macht somit eine robuste Erkennung unmöglich.

An der Suchposition 5 finden beide Varianten das Objekt. Abbildung 5.10 zeigt die Feature Korrespondenzen der Objekte.

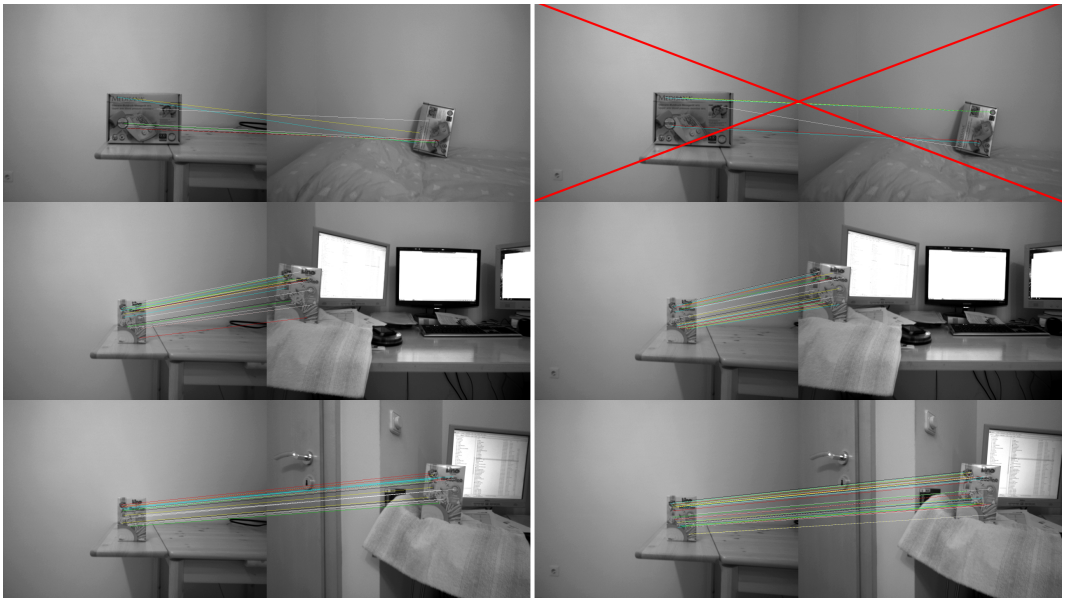

Abbildung 5.10: Versuch 1. Position 4 und 5. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante. Das nicht erkannte Objekt ist rot durchgestrichen.

Position	Objekt	Distanz	Orientierung	MultiView Inlier	SingleView Inlier
1	Langkornreis	0.75m	61.99°	39	24
1	Langkornreis	0.67m	31.85°	11	15
1	Superplus	0.79m	-28.03°	51	64
1	Superplus	0.71m	-56.62°	24	18
2	Langkornreis	0.54m	32.25°	19	not found
2	Obstriegel	0.72m	32.25°	48	42
2	Langkornreis	0.54m	1.39°	14	11
2	Superplus	1.27m	-27.19°	25	28
2	Superplus	1.27m	-57.15°	42	50
3	Fruchtschnitten	0.59m	32.47°	20	45
3	Obstriegel	0.57m	-28.17°	43	33
3	Fruchtschnitten	0.89m	42.99°	25	19
4	Medisana	1.33m	176.91°	8	not found
5	Medolino	0.67m	-87.93°	19	23
5	Medolino	0.66m	-117.41°	22	36

Tabelle 5.5: Versuch 1: Vergleich zwischen SingleView und MultiView Variante. Die MultiView Variante ist robuster als die SingleView Variante und findet mehr Objekte.

5.2.1.5 Auswertung: MultiView versus SingleView

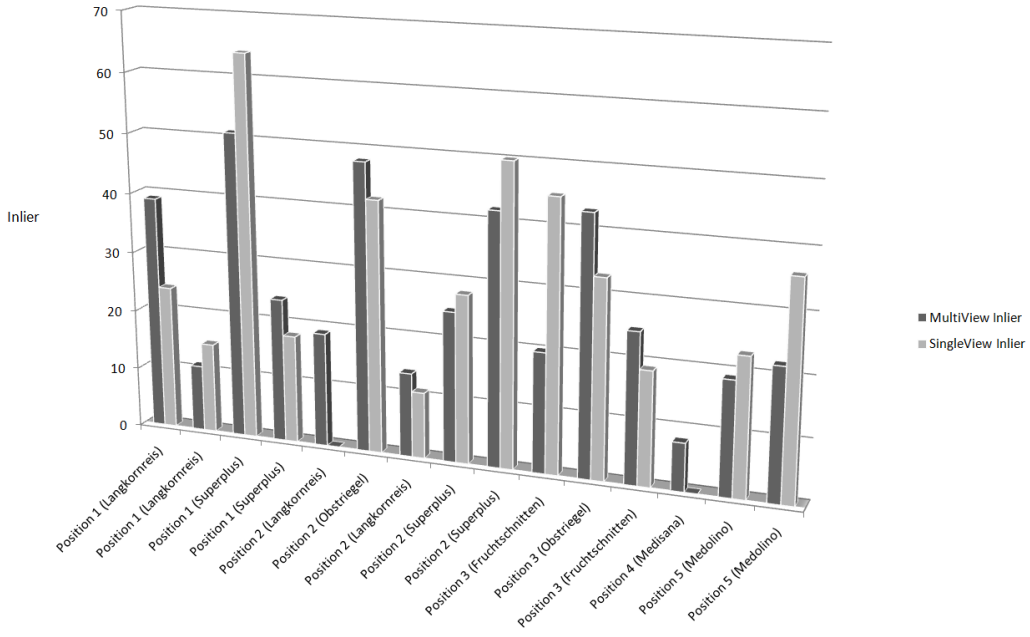

Abbildung 5.11: Versuch 1: Grafische Darstellung der Inlier von MultiView und SingleView Variante. Die MultiView Variante ist deutlich robuster als die SingleView Variante.

Zusammenfassend liefert die MultiView Variante robustere Ergebnisse als die SingleView Variante. Die Ergebnisse beider Varianten sind in Tabelle 5.5 gegenüber gestellt. Abbildung 5.11 stellt die Ergebnisse der Tabelle für eine bessere Übersicht grafisch dar.

5.2.2 Versuch 2

Beim ersten Versuch versagt die Erkennung des Objekts „Medisana“ bei der SingleView Variante auf Grund einer zu hohen Distanz zur Kamera. Für den zweiten Versuch wird dieses Objekt in seiner Position verändert und um ca. 30cm näher zum Roboter gestellt. Der Roboter soll nun den exakt gleichen Ablauf von Versuch 1 wiederholen.

5.2.2.1 Position 1

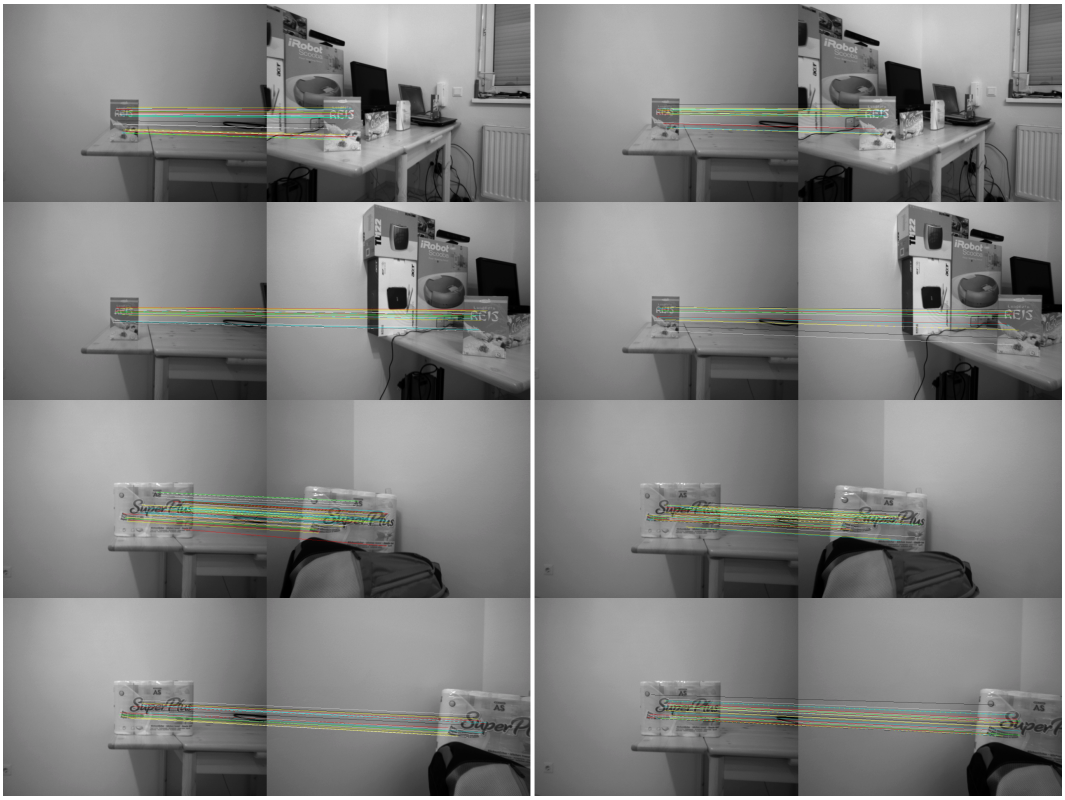

Abbildung 5.12: Versuch 2. Position 1. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante.

An der Suchposition 1 kann der Roboter alle Objekte erkennen. Die Inlier beider Wiederholungen sind ähnlich wie beim 1. Versuch. Abbildung 5.12 zeigt die Feature Korrespondenzen der Objekte.

5.2.2.2 Position 2

An der Suchposition 2 kann der Roboter bei beiden Wiederholungen alle Objekte lokalisieren. Im Gegensatz zum 1. Versuch findet die SingleView Variante das Objekt „Langkornreis“ zwei Mal. Die Robustheit der SingleView Variante liegt aber deutlich unter der der MultiView Variante. Abbildung 5.13 zeigt die Feature Korrespondenzen der Objekte.

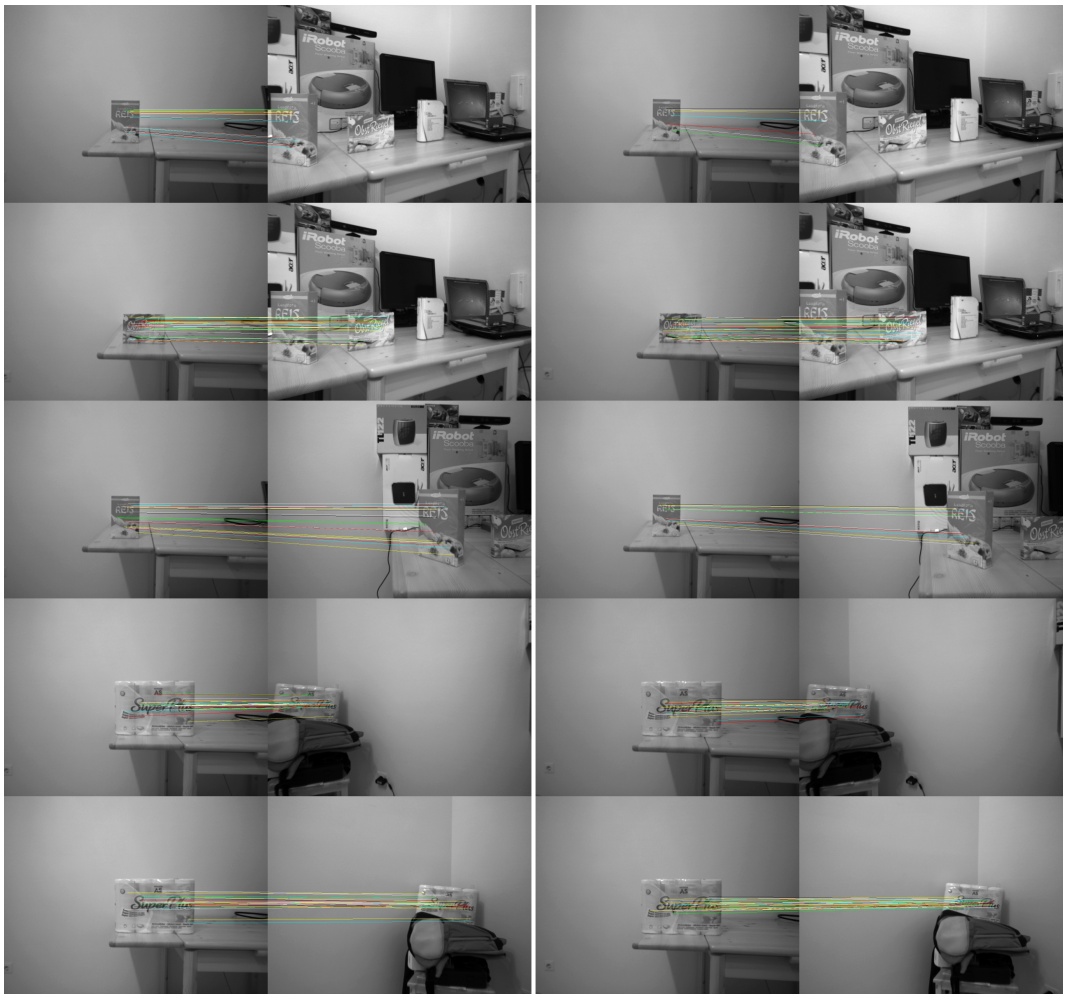

Abbildung 5.13: Versuch 2. Position 2. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante.

5.2.2.3 Position 3

An der Suchposition 3 kann der Roboter bei beiden Wiederholungen alle Objekte lokalisieren. Im Gegensatz zum 1. Versuch findet die SingleView Variante das Objekt „Fruchtschnitten“ nur ein Mal. Auch das Objekt „Obstriegel“ wird nicht von der SingleView Variante in allen Ansichten erkannt. In beiden Fällen kann die Homographie nicht richtig berechnet werden und der 3D Plausibilitäts-Check retourniert negativ. Auch an dieser Suchposition ist die Robustheit der SingleView Variante schlechter als die der MultiView Variante. Abbildung 5.14 zeigt die Feature Korrespondenzen der Objekte. Die Ansichten in denen das Objekt nicht erkannt wurde sind rot durchgestrichen.

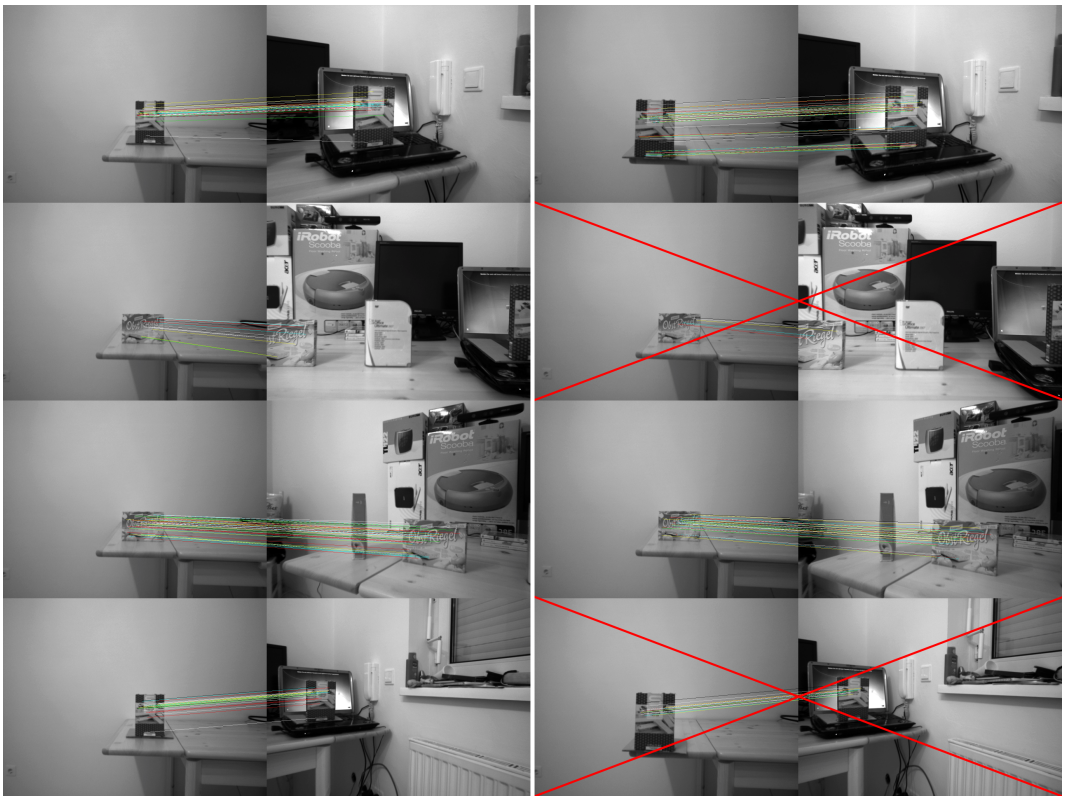

Abbildung 5.14: Versuch 2. Position 3. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante. Ansichten von nicht erkannten Objekten sind rot durchgestrichen.

5.2.2.4 Position 4 und 5.

Im Gegensatz zum 1. Versuch erkennt der Roboter an der Suchposition 4 das Objekt „Medisana“ robust. Die Multiview Variante als auch die SingleView Variante können das Objekt richtig lokalisieren. Auch an der Suchposition 5 finden beide Varianten das Objekt. Abbildung 5.15 zeigt die Feature Korrespondenzen der Objekte.

Abbildung 5.15: Versuch 2. Position 4 und 5. Linke Spalte: MultiView Variante. Rechte Spalte: SingleView Variante.

Position	Objekt	Distanz	Orientierung	MultiView Inlier	SingleView Inlier
1	Langkornreis	0.76m	62.01°	31	24
1	Langkornreis	0.68m	32.62°	20	14
1	Superplus	0.80m	-28.26°	47	52
1	Superplus	0.73m	-56.71°	22	25
2	Langkornreis	0.57m	32.43°	18	7
2	Obstriegel	0.75m	32.43°	51	42
2	Langkornreis	0.56m	2.21°	25	12
2	Superplus	1.29m	-27.47°	37	29
2	Superplus	1.30m	-57.12°	39	43
3	Fruchtschnitten	0.60m	31.56°	33	47
3	Obstriegel	0.56m	2.78°	13	not found
3	Obstriegel	0.57m	-27.67°	46	35
3	Fruchtschnitten	0.92m	42.41°	25	not found
4	Medisana	1.05m	176.16°	17	20
5	Medolino	0.65m	-87.42°	19	21
5	Medolino	0.65m	-117.43°	25	34

Tabelle 5.6: Versuch 2: Vergleich zwischen SingleView und MultiView Variante. Die MultiView Variante ist deutlich robuster als die SingleView Variante.

5.2.3 Auswertung: MultiView versus SingleView

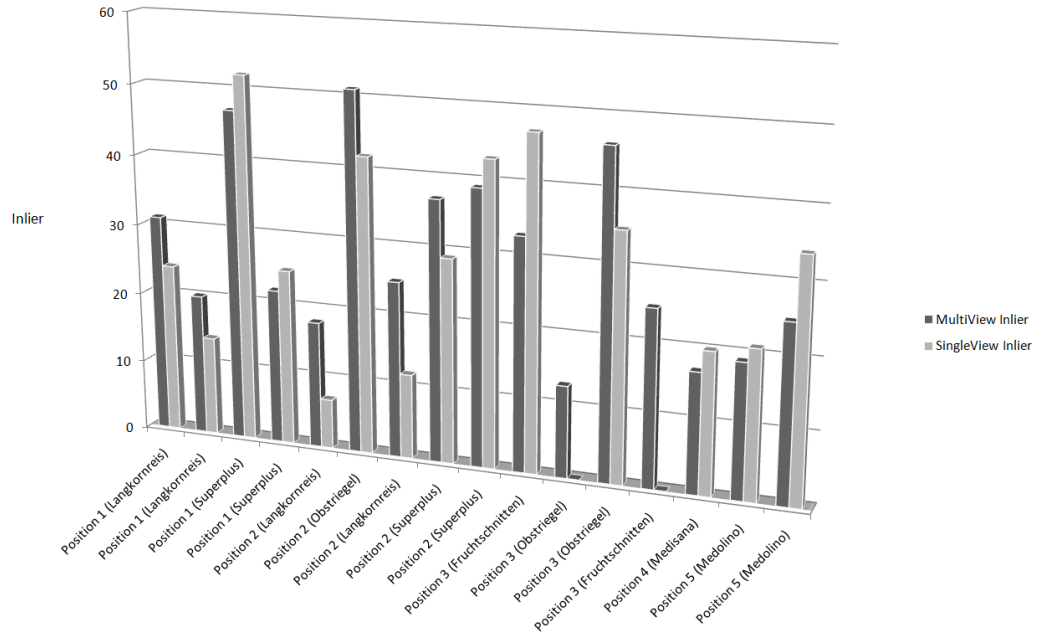

Abbildung 5.16: Versuch 2: Grafische Darstellung der Inlier von MultiView und SingleView Variante. Die MultiView Variante ist deutlich robuster als die SingleView Variante.

Die MultiView Variante liefert robustere Ergebnisse als die SingleView Variante. Die Ergebnisse beider Varianten sind in Tabelle 5.6 gegenüber gestellt. Die Abbildung 5.16 stellt die Ergebnisse der Tabelle für eine bessere Übersicht grafisch dar.

5.3 Skalierbarkeit des Systems

Dieses Unterkapitel widmet sich der Evaluierung der Erkennungsleistung und Geschwindigkeit des Systems. Für die Evaluierung wurde ein aussagekräftiger Testdatensatz aufgenommen und dem Erkennungssystem in einer Simulation präsentiert. Die Eingabedaten sind aufgezeichnete Sensordaten. Somit lässt sich die Wiederholbarkeit garantieren. Um die Skalierbarkeit des Systems zu testen, wurde die Objekte-Datenbank auf insgesamt 25 Objekte erweitert.

	Deskriptoren	Brute Force	Random Forest
1 Objekt	780	56,5 ms	4,5 ms
2 Objekte	1189	90,1 ms	5,25 ms
3 Objekte	1560	119,0 ms	5,8 ms
4 Objekte	2238	162,8 ms	7,1 ms
5 Objekte	2770	208,0 ms	8,2 ms
6 Objekte	3210	256,9 ms	9,0 ms
7 Objekte	3645	291,4 ms	9,8 ms
8 Objekte	4497	329,1 ms	11,3 ms
9 Objekte	4870	353,7 ms	11,9 ms
10 Objekte	5128	382,4 ms	12,5 ms
11 Objekte	5958	421,7 ms	13,9 ms
12 Objekte	6158	444,5 ms	14,4 ms
13 Objekte	6491	469,5 ms	15,0 ms
14 Objekte	6704	486,1 ms	15,2 ms
15 Objekte	6929	505,4 ms	15,5 ms
16 Objekte	6937	517,2 ms	15,6 ms
17 Objekte	7255	540,7 ms	16,2 ms
18 Objekte	7455	551,8 ms	16,3 ms
19 Objekte	7655	572,4 ms	16,6 ms
20 Objekte	7846	593,5 ms	16,8 ms
21 Objekte	8046	615,3 ms	17,3 ms
22 Objekte	8246	638,0 ms	17,5 ms
23 Objekte	8730	663,6 ms	18,5 ms
24 Objekte	8930	683,4 ms	18,8 ms
25 Objekte	9263	713,0 ms	19,4 ms

Tabelle 5.7: Gegenüberstellung Brute Force Matching und Random Forst Matching. Die Objekt Datenbank umfasst 25 Objekte. Als Random Forest wurde die OpenCV Implementierung von [16] herangezogen. Die Forest Konfiguration ist: 4 Trees, 32 Suchtiefe

5.3.1 Geschwindigkeit

Für Geschwindigkeits-Tests wurde die Anzahl der maximal erlaubten Features pro Objekt von 200 auf 1000 angehoben um die Feature-Datenbank möglichst groß zu machen. Die Tabelle 5.7 zeigt die gemessenen Werte für das Matching während der Evaluierung. Es wurden 50 Experimente durchgeführt (25 mit Brute Force Matching, 25 mit Random Forest). Bei jedem Experiment wurde der Random Forest neu trainiert. Die Evaluierung fand auf einem aufgezeichnetem Datensatz statt um die Wiederholbarkeit sicherzustellen.

Die Auswertung in Tabelle 5.7 zeigt, dass das Brute Force Matching nur bis zu einer geringen Objektanzahl echtzeitfähig ist. Das Random Forest Matching hingegen skaliert

und ist somit sehr gut für den Echtzeit Einsatz geeignet.

5.3.2 Robustheit: 2D versus 3D

Der 3D Filter wurde eingeführt um das System robuster gegenüber Störungen zu machen. In diesem Unterkapitel soll evaluiert werden wie sich die Zuschaltung des Filters auf Recall und Precision der Objekterkennung auswirkt. Der aufgezeichnete Datensatz wurde mit einer Groundtruth versehen. Für jede Szene gibt es eine Liste von Objekten die in der Szene erkannt werden können. In den Experimenten wird Recall als die Anzahl der richtig positiven Erkennungen (TP) über den maximal zu erkennenden Objekten pro Szene (nP) angegeben ($Recall = \frac{TP}{nP}$). Die Precision ist definiert als die Anzahl der richtig positiven Erkennungen pro Szene über den richtig und falsch-positiven (FP) Erkennungen pro Szene ($Precision = \frac{TP}{TP+FP}$).

In dem aufgezeichneten Datensatz befinden sich sieben Objekte über verschiedene Szenen verteilt. Es soll untersucht werden wie sich die Größe der Objektdatenbank auf Recall und Precision auswirkt.

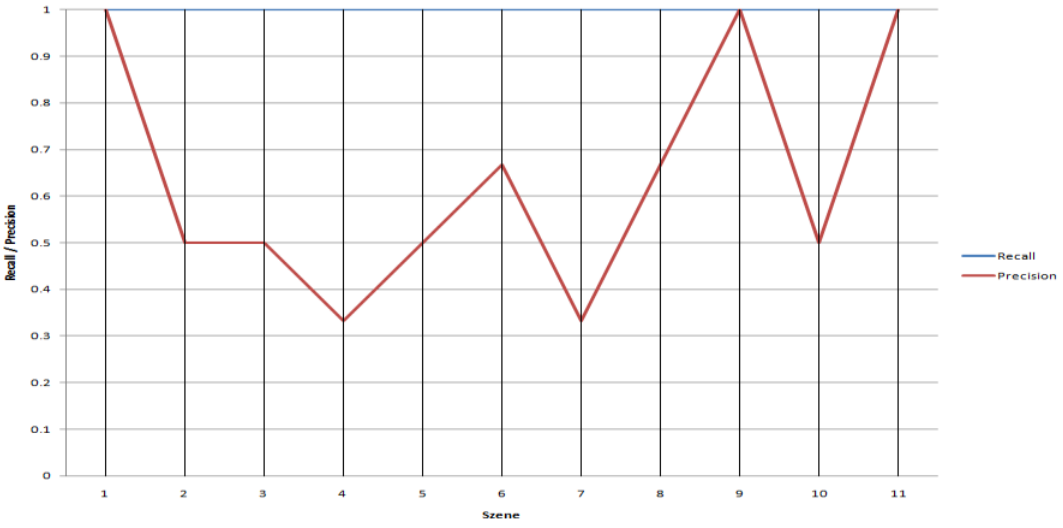

Abbildung 5.17: Ohne 3D Filter: Grafische Darstellung von Recall und Precision bei 7 Objekten in der Datenbank. Es kommt zu 10 falsch positiven Erkennungen.

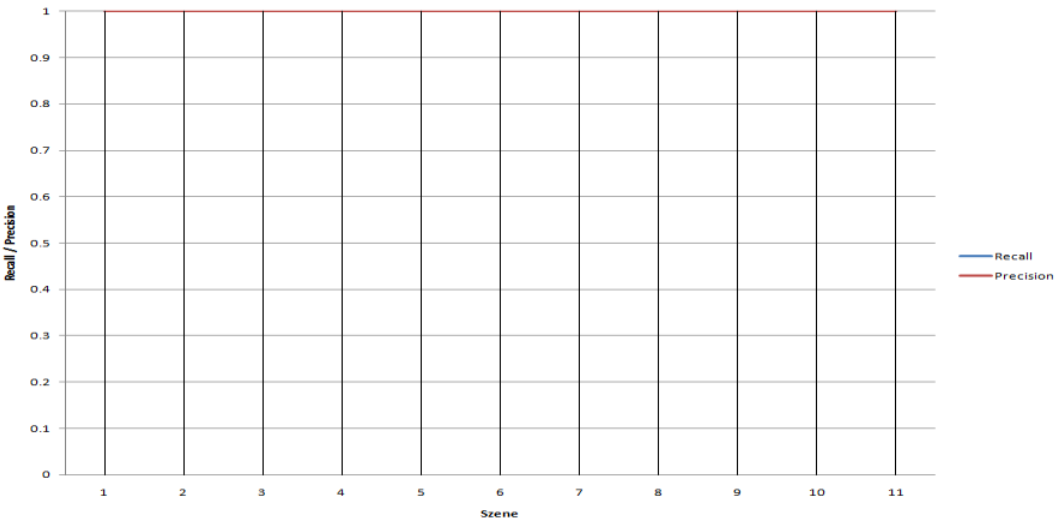

Abbildung 5.18: Mit 3D Filter: Grafische Darstellung von Recall und Precision bei 7 Objekten in der Datenbank. Es gibt keine falsch positiven Erkennungen.

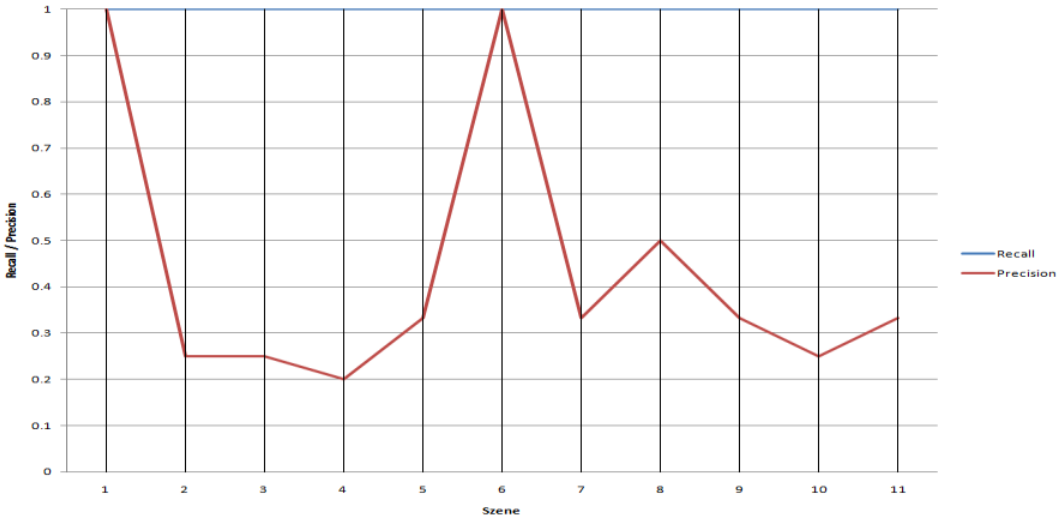

Abbildung 5.19: Ohne 3D Filter: Grafische Darstellung von Recall und Precision bei 25 Objekten in der Datenbank. Es kommt zu 23 falsch positiven Erkennungen.

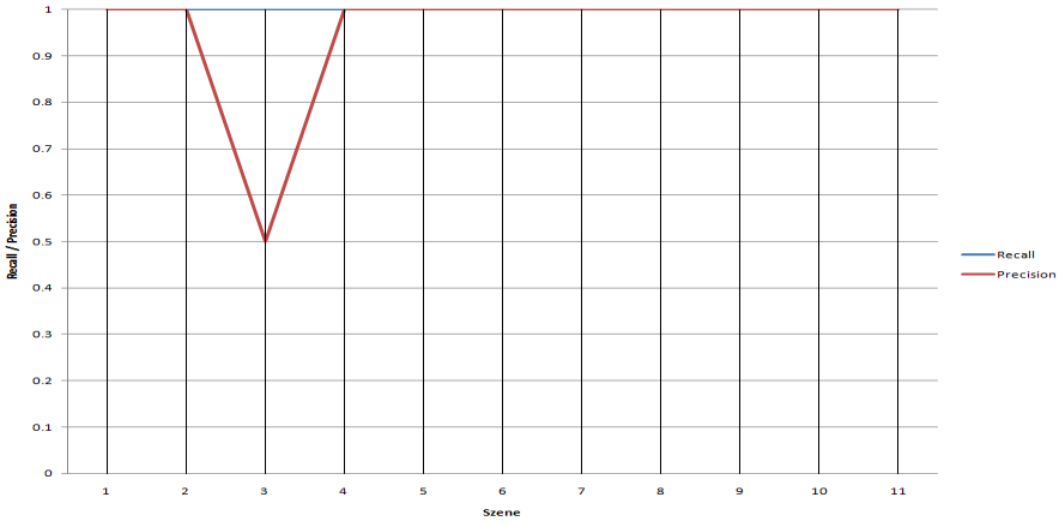

Abbildung 5.20: Mit 3D Filter: Grafische Darstellung von Recall und Precision bei 25 Objekten in der Datenbank. Es kommt zu einer falsch positiven Erkennung.

Bei den durchgeführten Experimenten kann der 3D Filter die Anzahl der falsch positiven Erkennungen reduzieren. In den meisten Fällen kommt es zu keiner einzigen falsch positiven Erkennung. Nur in einem Fall versagt der 3D Filter und lässt eine falsch positive Erkennung zu. Dieses Verhalten ist darauf zurückzuführen, dass das Objekt zufällig mit einem anderen ähnlich großen Objekt verwechselt wird. Eine Wiederholung mit Brute Force Matching zeigt, dass der Fehler vom Random Forest Matching ausgeht. Dieser produziert zu viele falsch positive Inlier. Diese FP Inlier ermöglichen eine Homographie die für die Größenbestimmung verwendet wird. Diese passt zufällig mit dem zu suchendem Objekt zusammen und verursacht die falsch positive Erkennung.

5.4 Stellungnahme

In den beiden Experimenten wird gezeigt, dass das in Kapitel 4 beschriebene und im Zuge der Diplomarbeit implementierte Objekterkennungssystem robust auf den Testdatensätzen arbeitet. Die Einbindung von verschiedenen Ansichten bei der Erstellung des Objektmodells erhöht die Erkennungshäufigkeit und Robustheit. In den beiden durchgeführten Versuchen kann der Roboter mit Hilfe der MultiView Objekt-Modelle alle zu suchenden Objekte erkennen und lokalisieren. Die Ergebnisse sind in den Abbildungen 5.21, 5.22 und 5.23 dargestellt. Die Grafiken stellen die Wiederholbarkeit der Erkennung in Bezug auf die

gefundenen Objekt-Inlier dar. Die MultiView Variante arbeitet robuster als die SingleView Variante.

Die Experimente zeigen, dass der 3D Filter eine unverzichtbare Maßnahme zur Reduktion von falsch positiven Erkennungen ist. Die Genauigkeit der Objekt-Lokalisierung ist dabei ausreichend um später Manipulationsaufgaben durchführen zu können. Probleme können auftreten, wenn das Objekt zu weit von der Kamera entfernt ist oder der Winkel zur Kamera zu hoch ist.

Die Suchzeit ist für alle durchgeführten Experimente unter fünf Minuten und somit unter dem vorgegeben Zeitlimit. Die meiste Zeit benötigt der Roboter für die Navigation. Vor allem die Drehbewegungen der Roboterplattform kosten viel Zeit, da der Roboter komplett still stehen muss bevor ein Bild ausgewertet werden kann. Die Zeit, die das Objekterkennungssystem für die komplette visuelle Analyse an einer Position in Anspruch nimmt, beträgt durchschnittlich 590ms. Dies inkludiert die Bildaufnahme, Übermittlung, Rektifizierung, Stereo Berechnung, Feature Extraktion, Feature Matching und Plausibilitäts-Checks.

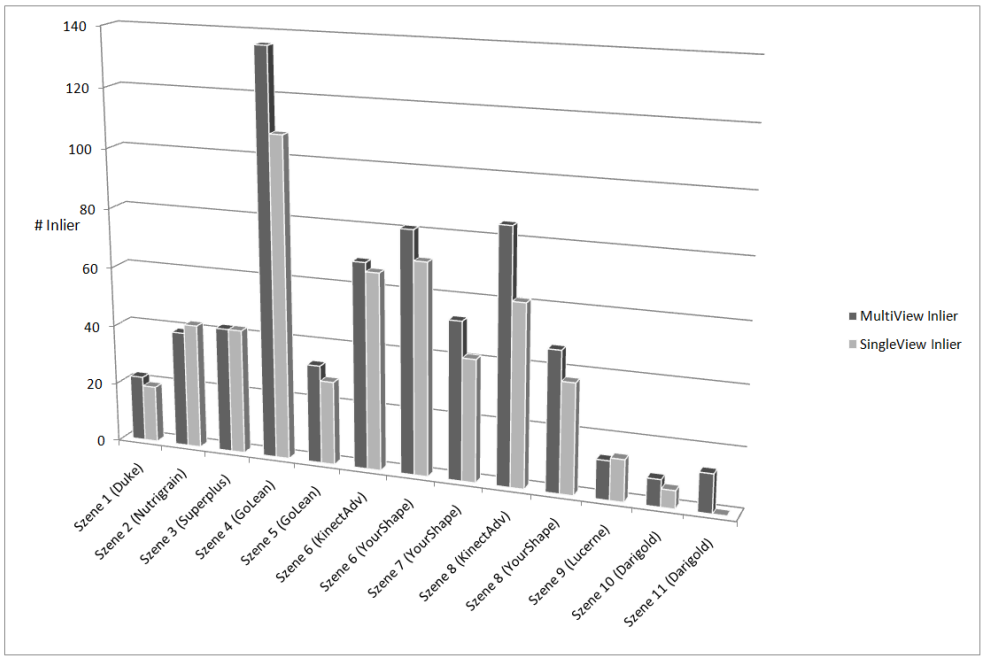

Abbildung 5.21: Vergleich der Anzahl der Inlier von MultiView und SingleView Variante. Die Objekte sind je mit einem View oder neun Views trainiert worden. Die Evaluierung fand auf dem aufgenommenen Datensatz statt.

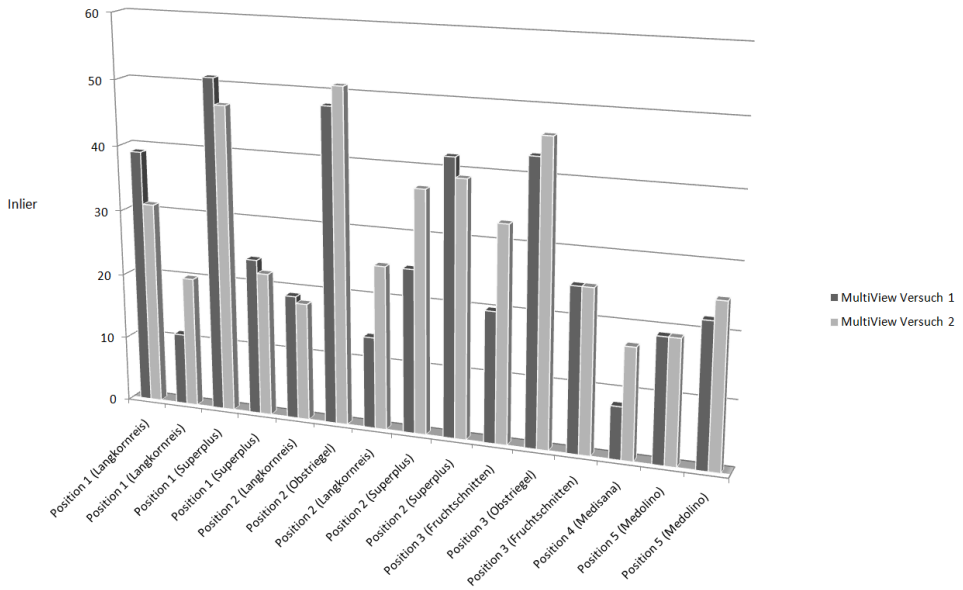

Abbildung 5.22: Wiederholbarkeit MultiView Variante. Die Inlier pro Objekt sind je für Versuch 1 und 2 gezeigt.

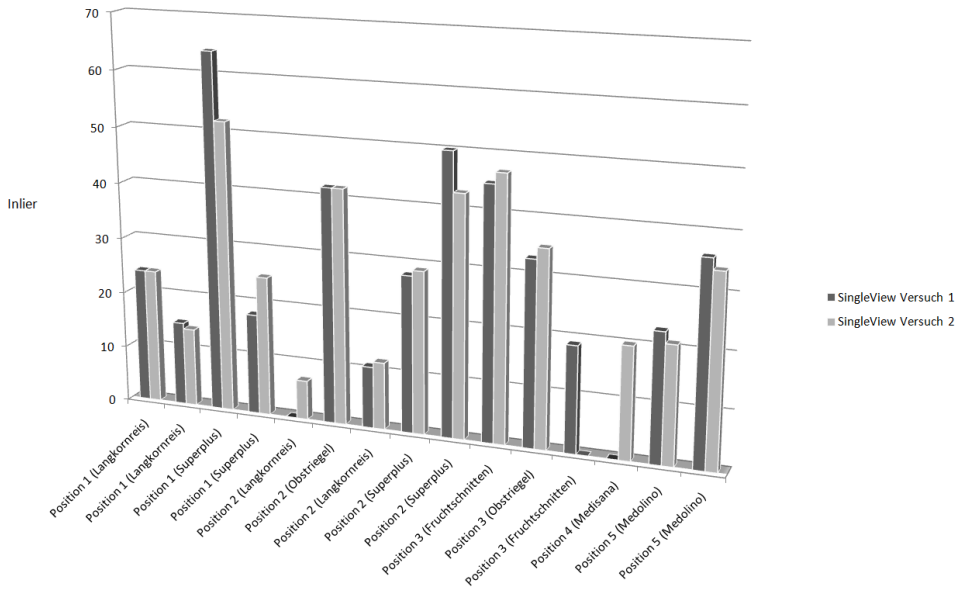

Abbildung 5.23: Wiederholbarkeit SingleView Variante. Die Inlier pro Objekt sind je für Versuch 1 und 2 gezeigt.

Kapitel 6

Zusammenfassung und Ausblick

Ziel der Diplomarbeit war es ein Objekterkennungssystem für den mobilen Roboter FLEA zu entwickeln. Die Erkennung sollte sich auf haushaltstypische, stückweise planare und texturierte Objekte beschränken. Für die Arbeit war es entscheidend, dass Objekte mit wenig menschlichem Aufwand vom Roboter gelernt werden können. Der Mensch übergibt dem Roboter ein Foto vom zu lernenden Objekt, platziert das Objekt auf einem Tisch und gibt dem Roboter den Befehl das Objekt zu lernen. Der Roboter fährt verschiedene Positionen vor dem Objekt an, um verschiedene Ansichten von dem Objekt aufzunehmen. Danach baut das entwickelte Objekterkennungssystem ein Objektmodell auf.

Für die visuelle Erkennung wurden lokale Features verwendet, welche über CenSurE detektiert wurden. Für die Feature Beschreibung wurde ein 64D SURF Deskriptor verwendet. Der Roboter FLEA wurde im Zuge dieser Arbeit um eine Stereo Kamera erweitert. Mit Hilfe von Stereo konnte der visuelle Suchbereich auf den optimalen Arbeitsbereich der Kamera eingeschränkt werden. Die Einbindung von Tiefeninformation zur Objekterkennung diente zusätzlich der Beseitigung von Ausreißern bei der Feature Korrespondenz Findung als auch um die Plausibilität der Erkennung zu prüfen. Verschiedene Ansichten wurden benutzt um die stabilsten Objekt-Features auszuwählen. Für das Matching wurde ein Random Forest eingesetzt, da dieser sehr schnell und zuverlässig arbeitet. Der Roboter lernt beim Training auch die metrische Größe des Objekts und kann somit Fehldetektionen erfolgreich erkennen. Jede Objekt Hypothese wird auf ihre Plausibilität in 3D überprüft.

Das implementierte System wurde mit einem RoboCup@Home „Lost and Found“ ähnlichen Tests geprüft. Bei diesen Tests musste der Roboter innerhalb von fünf Minuten eine Haushaltsumgebung nach zuvor gelernten Objekten absuchen. Das System konnte in den Experimenten die Objekte innerhalb dieser Zeitspanne finden. Es kam zu wenigen bis kei-

nen falsch positiven Erkennungen. Das Experiment wurde in unterschiedlichen Umgebungen wiederholt - mit dem selben Ergebnis. Die Verwendung von verschiedenen Ansichten für die Erstellung des Objektmodells trägt zur Robustheit des Gesamtsystems bei. Die Experimente wurden mit Objektmodellen, die nur mit einer Ansicht gelernt wurden wiederholt. Die Robustheit war bei allen Experimenten geringer als bei der Verwendung von mehreren Ansichten.

Das implementierte Objekterkennungssystem stellt das Fundament für zukünftige Aufgaben dar. Der Roboter ist nun in der Lage ein bekanntes Objekt in einer Umgebung wieder zu finden und die Position des Objekts im Raum zu bestimmen. Der logische nächste Schritt ist die Erweiterung des Roboters um einen Greifarm um Objektmanipulation betreiben zu können.

Literaturverzeichnis

[1] Agrawal, M., Konolige, K., and Blas, M. (2008). Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision (ECCV)*, volume 5305, pages 102–115.

[2] Aydemir, A., Sjöö, K., Folkesson, J., Pronobis, A., and Jensfelt, P. (2011). Search in the real world: Active visual object search based on spatial relations. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA'11)*, Shanghai, China.

[3] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110:346–359.

[4] Beis, J. and Lowe, D. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1000–1006.

[5] Breuer, T., Giorgana, G., Hegger, F., Müller, C., Jin, Z., Reckhaus, M., Paulus, J., Hochgeschwender, N., Awaad, I., Hartanto, R., Ploeger, P., and Kraetzschmar, G. (2010). The b-it-bots robocup@home 2010 team description paper. b-it-bots Website. Online verfügbar unter `http://www.b-it-bots.de/Publications_files/b-it-bots-tdp-2010.pdf`; Stand vom 7. Oktober 2010.

[6] Federation, R. (2010). Robocup@home rulebook 2010. RoboCup@Home Website. Online verfügbar unter `http://www.ai.rug.nl/robocupathome/documents/rulebook2010_FINAL_VERSION.pdf`; Stand vom 5. Oktober 2010.

[7] Friedman, J., Bentley, J., and Finkel, R. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematics Software*, 3(3):209–226.

[8] Fukunaga, K. and Narendra, P. (1975). A branch and bound algorithms for computing k-nearest neighbors. pages 750–753.

[9] Gossow, D., Wojke, N., Bing, R., Buchholz, U., Schrage, R., Mützel, A., Read, K., Thierfelder, S., V. S., and Paulus, D. (2010). Robocup 2010 - homer@unikoblenz. Homer Website. Online verfügbar unter `http://userpages.uni-koblenz.de/~robbie/fileadmin/pdfs/tdp_homer_2010.pdf`; Stand vom 6. Oktober 2010.

[10] Lavalle, S. (1998). Rapidly-exploring random trees: A new tool for path planning.

[11] Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479.

[12] Leuze (2010). Rod4 datenblatt. Leuze Website. Online verfügbar unter `http://www.leuze.de/downloads/los/db/28_brods/DS_ROD4plus_de.pdf`; Stand vom 16. Dezember 2010.

[13] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[14] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, volume 1, pages 384–393.

[15] Meger, D., Muja, M., Helmer, S., Gupta, A., Gamroth, C., Hoffman, T., Baumann, M., Southey, T., Fazli, P., Wohlkinger, W., V. P., Little, J., Lowe, D., and Orwell, J. (2010). Curious george: An integrated visual search platform. UBC Website. Online verfügbar unter `http://www.cs.ubc.ca/labs/lci/curious_george/pubs/meger_srvc_crv_2010.pdf`; Stand vom 10. November 2010.

[16] Muja, M. and Lowe, D. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. INSTICC Press.

[17] Nistér, D. and Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168.

[18] Pointgrey (2010). Bumblebee xb3. Pointgrey Website. Online verfügbar unter `http://www.ptgrey.com/products/bbxb3/bumblebeeXB3_stereo_camera.asp`; Stand vom 16. Dezember 2010.

[19] ROS (2010). Robot operating system. ROS Website. Online verfügbar unter `http://www.ros.org/`; Stand vom 23. Dezember 2010.

[20] Rybski, P. and Efros, A. (2007). Overview of the 2007 semantic robot vision challenge competition. In *Proceedings of the AAAI'07 Mobile Robot Competition and Exhibition Workshop*.

[21] Silpa-Anan, C. and Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

[22] SRVC (2011). Semantic robot vision challenge arena. SRVC Website. Online verfügbar unter `http://1.bp.blogspot.com/_fxRR_bT3LgA/S0OM_lDhePI/AAAAAAAACEc/vBKcg9wAdAE/s1600-h/arena2.jpg`; Stand vom 3. Jänner 2011.

[23] Stückler, J., Dröschel, D., Gräve, K., Holz, D., Schreiber, M., and Behnke, S. (2010). Nimbro@home team description paper 2010. NimbRo Website. Online verfügbar unter `http://www.ais.uni-bonn.de/nimbro/@Home/papers/NimbRo@Home_TDP_2010.pdf`; Stand vom 5. Oktober 2010.

[24] Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3310 – 3317.

[25] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press.

[26] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, Hawaii.

[27] WillowGarage (2011). Willow garage pr2. Willow Garage PR2 Website. Online verfügbar unter `http://www.willowgarage.com/pages/pr2/design`; Stand vom 23. Juli 2011.