Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Studienarbeit

Webcam basiertes Kamerasystem für autonome Roboter: Erste Konzeption

vorgelegt von
Michael Gottwald
am 1. Juni 2005

Studiengang Softwaretechnik

Betreuender Prüfer: Prof. Kai von Luck

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Webcam basiertes Kamerasystem für autonome Roboter: Erste Konzeption

Stichworte Client/Server-Architekturen, CORBA, .NET, Java-Enterprise, autonome mobile Roboter, Bildverarbeitung, Personal Digital Assistant

Zusammenfassung

In dieser Arbeit erfolgt eine erste Konzeption eines Kameraserver. Das gesamte System besteht hierbei aus einen Roboter, einen Servers, der spezielle Dienste für den Roboter bereitstellt und einen Kamera-Server, welcher über eine Kamera gelieferte Bilder auswertet (Objekterkennung und -Positionierung) und diese Informationen bereitstellt. Insbesondere werden hierbei unterschiedliche für die Entwicklung von verteilten Anwendungen in Frage kommenden Middlewares betrachtet.

Camera-System based on Webcam for autonomous robot: First concept

Keywords client/server-architecture, CORBA, .NET, Java-Enterprise, autonomous mobile robot

Abstract

This report is a first concept of a Camera-Server. The whole System is divided into following parts: A Robot, a Server providing special services to the robot, and a camera-server for image-processing (object-identification and -location). Especially there will be a examination of middlewares suitable for developement distributed systems.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

Diese Studienarbeit baut im wesentlichen auf die Diplomarbeit von Rainer Balzerowski „Realisierung einer Webcam basierten Kamera System für mobile Roboter auf“ (Balzerowski 2002). Ziel dieser Arbeit war es, ein geeignetes Verfahren zur digitalen Bildverarbeitung zu erarbeiten respektive zu realisieren. Hierbei galt es, bestimmte Farbobjekte zur erkennen, und deren Position über eine definierten Schnittstelle an einem Roboter weiterzugeben. Diese Arbeit sollte die bestehende Lösung zur digitalen Bildverarbeitung von LEGO, die LEGO Vision Command Kamera in Zusammenspiel mit LEGO RCX Mikrocomputer, verbessern. Natürlich unterlag diese Arbeit Hardware- und Software Rahmenbedingungen, die sich vornehmlich auf die von Lego angebotenen Software-Toolkits, Frameworks und auch Hardware stützte. Diese Arbeit wird sich primär mit der Kommunikationsschicht zwischen Roboter und Kamera-Server beschäftigen und eine erste Konzeption eines Kameraservers unter Benutzung geeigneter Middleware aufzeigen.

1.1 Motivation

Die eigentliche Motivation rührte von der bereits in der Einleitung beschriebene Diplomarbeit Rainer Balzerowski's her. Obwohl die Arbeit schon eine große Verbesserung eines bestehenden Systems darstellte, waren einige Punkte noch nicht zu vollsten Zufriedenheit, beispielsweise im Kommunikationsbereich oder auch bei der Bildauswertungsbereich, gelöst. Wie könnte also eine Verbesserung aussehen? Wo kann etwas verbessert werden? Sind neue Technologien anwendbar? Diese und andere Fragen haben mich bei der Durchsicht Balzerowskie's Arbeit beschäftigt und waren gewichtige Gründe für den Beginn dieser Arbeit.

1.2 Ziele

Ziel ist es, unter Einbeziehung der Arbeiten von Mirco Gerling (Gerling 2003) und Ilia Revout (Revout 2003) sowie neuer Techniken, eine verbessertes Modell eines Kamera-Servers zu entwickeln. Der Einsatz neuer Techniken (Software, Middleware, Hardware) wird nach sorgfältiger Auswahl, gemessen an bestimmten Kriterien, erfolgen.

1.3 Aufgabenbeschreibung

1.4 Kapitelübersicht

Nachfolgend sei eine kurze Beschreibung der Kapitel gegeben:

- **Kapitel 2: Beschreibung der Teilbereiche**
  Gibt eine kurzen Einblick über die Bereiche, in der sich diese Studienarbeit bewegt.

- **Kapitel 3: Szenario: Robot-Soccer-League**
  Beschreibung des Einsatzbereichs des entstehenden Framework.
- **Kapitel 4: Kommunikationsschicht und Middleware**
  Welche Middleware eignet sich am besten für die zu entwickelnde Anwendung? Beschreibung einzelner in Frage kommender Frameworks werden hier beschrieben.
- Resümee Beschreibt die Ergebnisse der Studienarbeit.

2 Beschreibung der Teilbereiche

Vorerst sollen ein paar grundlegende Begriffe erläutert werden, welche uns im Verlaufe dieser Arbeit immer wieder begegnen werden. Dem jeweiligen Teilgebiet folgt eine Kurzbeschreibung der Arbeiten von Rainer Balzeroswki, Ilia Revout und Micro Gerling. Da Balzeroswki's Arbeit als Ausgangspunkt dient, wird diese dementsprechend detaillierter beschrieben.
Folgende Abbildung zeigt nochmals die Kernthemen der Diplomarbeiten, auf die ich im weiteren Verlaufe dieser Arbeit fortwährend Bezug nehmen werde.

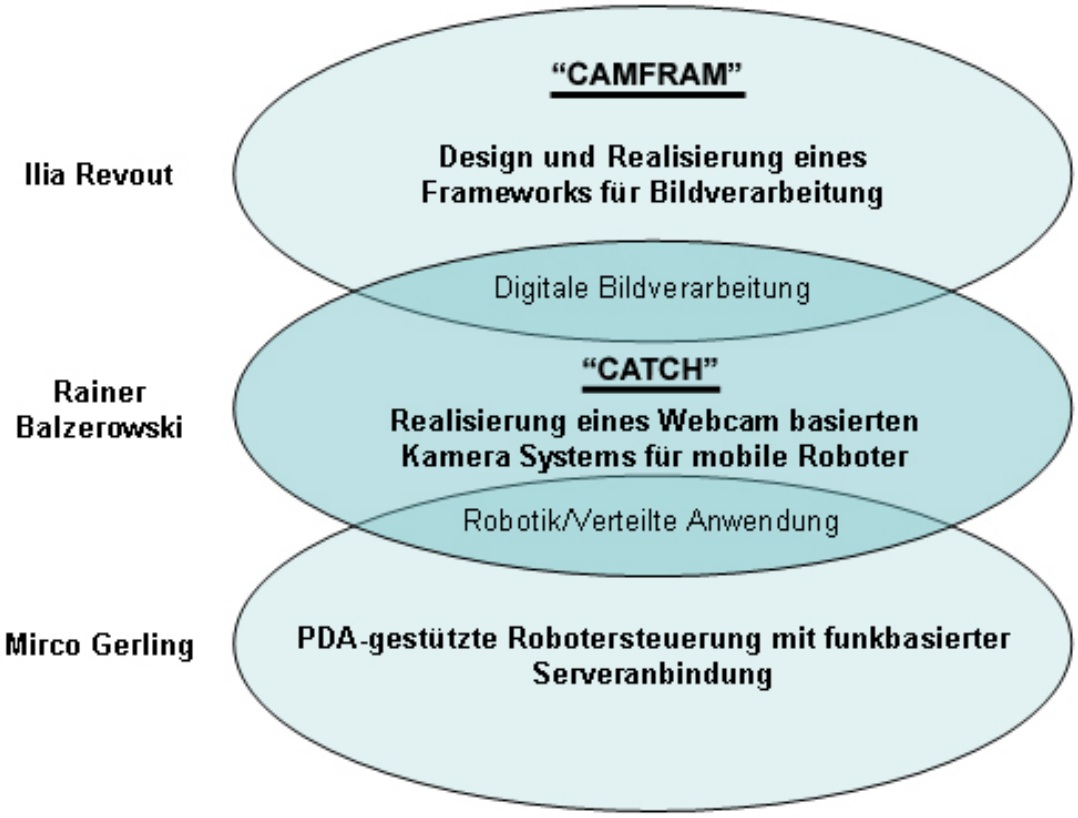


Abbildung 2.1: Involvierte Diplomarbeiten

2.1 Digitale Bildverarbeitung

Ein großer Teil unseres Gehirn ist für die Verarbeitung visueller Informationen zuständig. Täglich werden vom Menschen gerade auf Basis dieser visuellen Information unzählige Entschei-

dungen getroffen. Wie aber kann diese Fähigkeit des Menschen „zu sehen“ nachgebildet respektive auf Maschinen übertragen werden? Hier beginnt das Aufgabengebiet der digitalen Bildverarbeitung. Analog zum Zusammenspiel von Auge und Gehirn beim Menschen, gibt es bei der digitalen Bildverarbeitung eine Kombination aus Bildsensor und Computertechnologie. Dabei sind Anwendungsgebiete der digitalen[1] Bildverarbeitung vielfältig, und es zeichnet sich jetzt schon ab, daß diese noch junge Technologie zur einer Schlüsseltechnologie heranwachsen wird. Nachfolgend seien einige Beispiele für künftige und auch gegenwärtige Einsatzgebiete der digitalen Bildverarbeitung genannt:

Bereich	**Beispiel**
Industrielle Bildverabeitung	Druckuntersuchung Metrologie (Präzisionsmessung); **Robotersteuerung**; Nahrungsuntersuchung
Medizinische Bildverarbeitung	Dreidimensionalen Computer-Tomographie-Bildern; Analyse von Blutzellen
Automatische Identifikation	Lesen von 1D- und 2D-Barcodes; Sortierung von Paketen durch Lesen der Barcodes oder zeichenbasierten Identifizierungscodes
Kriminalistik Überwachung Sicherheit	Spurensuche auf Tatortfotos; Auffinden von Sprengstoffen und Waffen in Gepäck auf Flughäfen; Raumüberwachung
Verarbeitung von Schriftstücken	Briefsortierung in Post-Sortierungs-Zentren; Lesen von Formularen
Analyse dynamischer Prozesse	Botanik: Wachstums von Pflanzen und der Mechanismen, die es steuern;

Tabelle 2.1: Digitale Bildverarbeitung: Anwendungsbeispiele (ADB 2005)

Die digitale Bildverarbeitung ist ein komplexer Prozess und beginnt bei der Bilderfassung, Bildvorverarbeitung und endet beim Bildverstehen. Auf der nachfolgende Seite ist eine Hierarchie von Bildverarbeitungsoperation abgebildet (Springer) - dabei bilden die in den rechteckigen Kästchen abgebildeten Operationen eine eigene Disziplin in der digitalen Bildverarbeitung (selbstverständlich kann in dieser Arbeit nicht auf jede einzelne Disziplin detailliert eingegangen werden, dieses würde bei weitem den Rahmen dieser Ausarbeitung sprengen). Eine Übersicht über die Bildverarbeitung und deren Komplexität gibt die Abbildung 2.2 wieder. Ein Bildverarbeitungssystem läßt sich in folgende Bereiche aufteilen:

- Bildgebung
- Vorverarbeitung

[1] [nach lat. digitus »Finger«, bzw. digitalis »zum Finger gehörig«], Bezeichnung für die Eigenschaft einer Größe, nur eine endliche Zahl von Werten annehmen zu können (daher der Name, denn man kann die Zahl der Werte mit den Fingern abzählen). Im Unterschied zu einer analogen Größe sind keine Zwischenwerte definiert.

- Segmentierung
- Merkmalsextraktion
- Klassifikation

2.1.1 Bildgebung

Bilder dienen als Informationsquelle. Dabei könnnen die Bildinformationen die bereitgestellt werden soll, je nach Anwendungsgebiet unterschiedlicher Art sein. Dementsprechend gibt es unterschiedliche bildgebende Verfahren, die je nach angestrebter Informationsart Bilder künstlich erzeugen. Beispiele seien hier:

- Fotografie
- Digitale Fotografie
- Ultraschall (medizinischer Bereich)
- Röntgen/Defraction Enhanced Imaging (med. Bereich)
- Computer Tomographie
- Laserkamera
- Scanner

Bei der Gewinnung von Bildinformationen wird zwischen

- Passiven „Sensoren“ (Fotokamera, Digitalkamer)
- Aktiven „Sensoren“ (Laufl̈angenverfahren, z.B. Ultraschall, Radar, Satellitenbilder; Umgebung wird „aktiv“ abgetastet)

unterschieden. Examplarisch sei hier die funktionsweise der Digitalkamera näher beschrieben - diese wird auch später in der Anwendung als bildgebendes Verfahren eingesetzt: Eine digitale Kamera funktioniert grundsätzlich ähnlich wie ein mechanischer Fotoapparat. Sie unterscheidet sich lediglich in ihrer Speicherart und der damit verbundenen Technologie.

Während ein analoges Modell den Schnappschuss auf einem Zelluloid-Streifen verewigt, wird in der digitalen Kamera das Abbild mit speziellen Sensoren aufgefangen. Hier werden zunächst über einen Bildwandler, den sogenannten CCD- Charge Coupled Devices oder CMOS-Chip Complementary Metal Oxide Semiconductor, Informationen in elektronische Signale umgewandelt. Zum Sichern der digitalen Bilder werden die Daten auf einem Speichermedium abgelegt. Je nach Anspruch und Bedürfnis stehen dabei verschiedene Speichertypen und -volumina zur Verfügung.

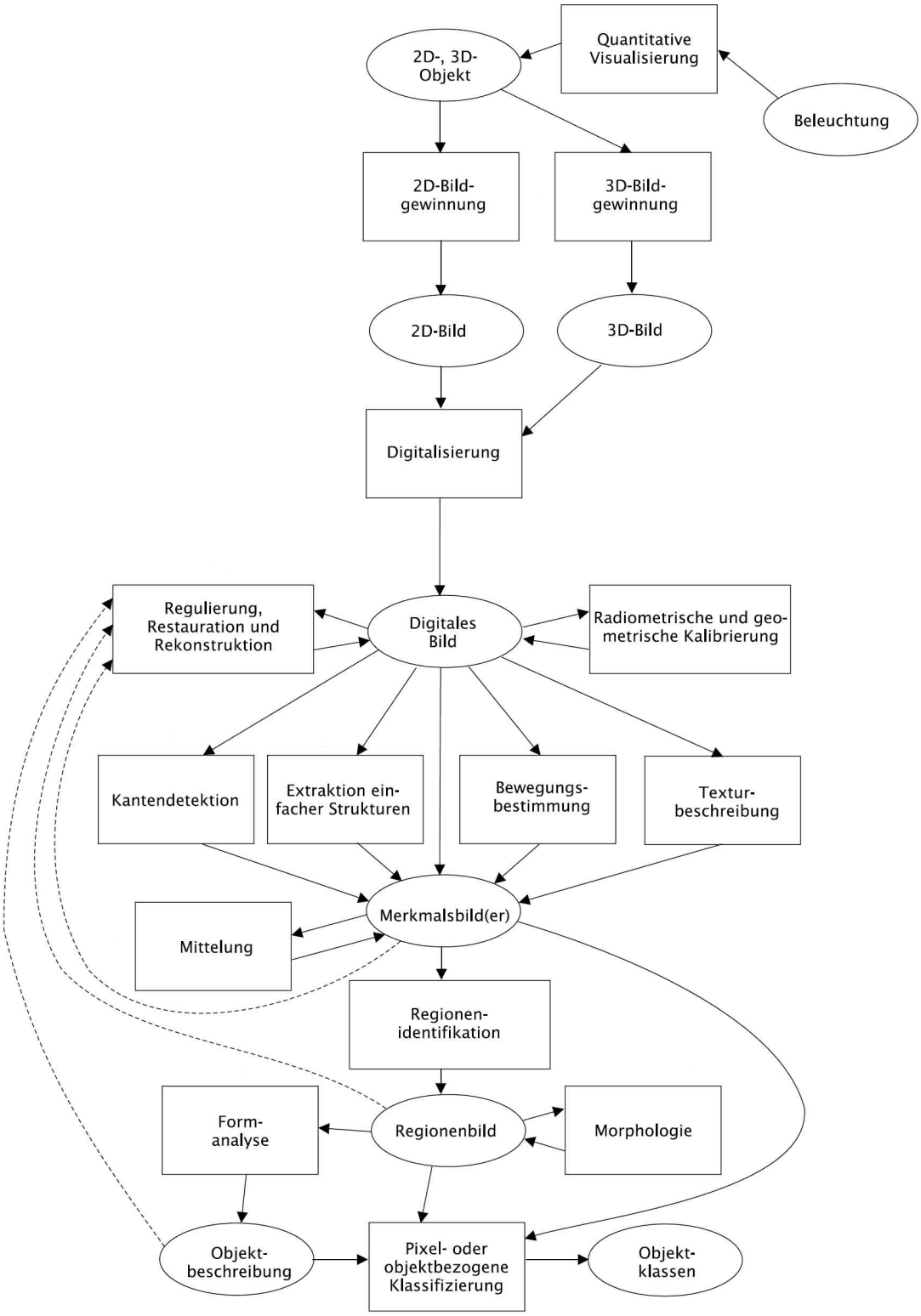


Abbildung 2.2: Hierarchie der Bildverarbeitungsoperationen von der Bildaufnahme bis zum Bildverstehen, (Jähne 2002)

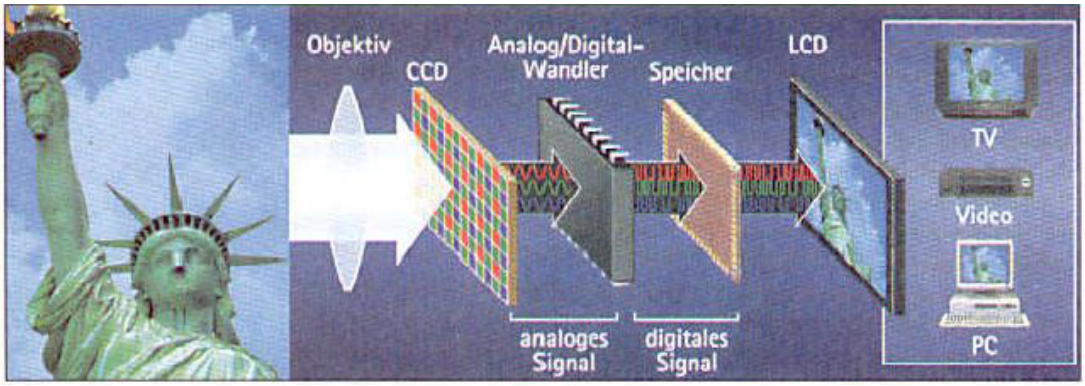


Abbildung 2.3: Funktionsweise der Digitalkamera: Charged Coupled Device (CCD), ein analoger Sensor, welcher proportional zu Lichtintensität Spannung liefert. Farben werden durch den CCD nicht erkannt - die Farben werden zuerst durch Prismen in bestimmte Farben (RGB = Red Green Blue oder CMY = Cyan Magenta Yellow) aufgespalten und dann dem Sensor weitergereicht, (Sanyo-Fisher 2005)

2.1.2 Vorverarbeitung

Bei der Bildvorverarbeitung wird das Bild modifiziert ohne aber den Informationsgehalt wesentlich zu verändern. Es besteht somit eine Bildtransformation mit Eingabe- und Ausgabebild. Ziele der Bildvorverarbeitung sind:

- Beleuchtungskorrekturen (inhomogene Beleuchtung)
- Glättung
- Beseitigung von Digitalisierungsfehlern
- Kontrastverstärkung
- Normierung (Größe, Form, Farbe)
- Beseitigung von Inhomoginitäten der Photoschicht des Aufnahmesystems
- Korrektur von Grauwertverzerrungen
- Filteranpassung an bestimmte Frequenzen

Beispielsweise läßt sich mit der Histogrammebnung eine Grauwertäqualisation und Gleichverteilung der Grauwerte über der Skala erreichen - in der Regel führt dieses zu einer Verbesserung der visuellen Erkennbarkeit sowie zu einer Kontrastverstärkung.

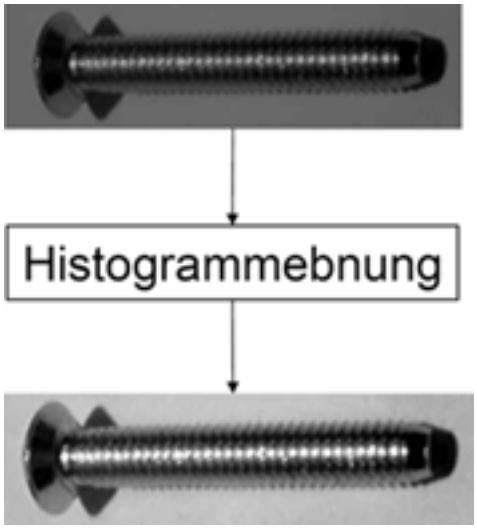


Abbildung 2.4: Beispiel einer Histogrammebnung, aus (Hermes 2004)

2.1.3 Segmentierung

Die Segmentierung versucht:

- die zu untersuchenden Objekte von den „übrigen“ Bildstrukturen zu trennen,
- sich gegenseitig berührende Objekte zu trennen,
- eine Zerlegung in Unterobjekte vorzunehmen,
- eine Klassifikation auf Pixelebene (Objektpixel/Nichtobjektpixel) vorzunehmen,

Eine Art der Segmentierung wäre das punktorientierte Verfahren. Hierbei handelt es sich um einen Schwellwertverfahren, welches durch definierte Maxima (Intensität) Objekte aus einem Bild herauslöst (siehe Abbildung 2.5). Bei der Farbsegmentierung wird das Bild in Regionen zerlegt. Hierbei wird unterschieden zwischen einer

- Untersegmentierung, d. h. das Bild wurde in zuwenige Regionen eingeteilt (Informationsverlust)
- Übersegmentierung, d. h. das Bild wurde in zuviele Regionen eingeteilt (Informationsüberfluß)

Gesucht wird natürlich ein geeignetest Mittelmaß. -

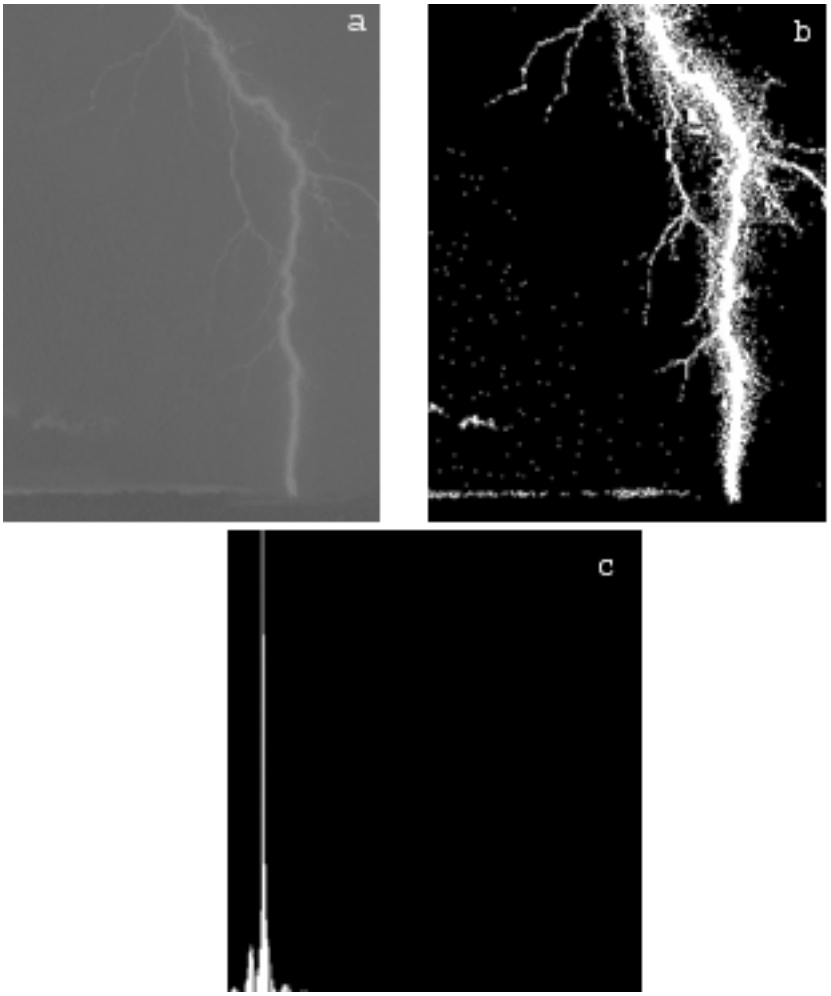


Abbildung 2.5: Schwellwertverfahren: a)Original, b)Ergebnis, c) Maxima, (Hermes 2004)

2.1.4 Merkmalsextraktion

Bei der Merkmalextraktion wird versucht bestimmte Merkmale aus einem Bild herauszulösen. Die Merkmale lassen sich in folgenden Gruppen unterteilen: Dabei wird, je nach angestrebter Merkmalextraktion, das digitale vorverarbeitete Bild in ein sogenanntes Merkmalbild überführt, welches wiederum der Objekterkennung dient. Z.B. können Objekte nicht nur durch den Unterschied ihres mittleren Grauwertes erkannt werden, welcher sich von dem des Hintergrundbildes unterscheidet - es kann hierfür auch die Orientierung oder Größe des Musters herangezogen werden.

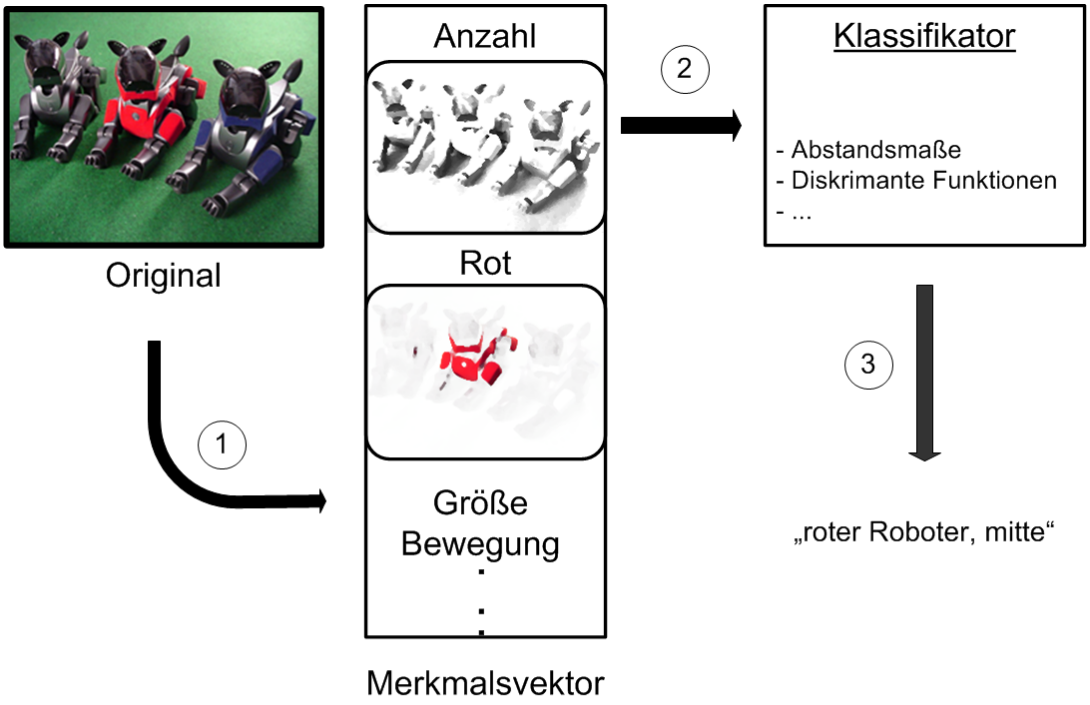
Flächen	**Kontur**
- Intensitätswert/Farbe	- Kettencode
- Umfang, Größe	- Polygon
- Schwerpunkt	- Länge
- Hauptachsen	- Orientierung
- Polyeder	- ...
- ...	
Stat. Merkmale	**Textur**
- Informationsgehalt	- Statistische Merkmale
- Momente	- Visuelle Eigenschaften
- Anzahl der Pixel	- ...
- ...	

Tabelle 2.2: Digitale Bildverarbeitung: Merkmalgruppen

Abbildung 2.6: Objekterkennung durch Unterschiede des Grauwertes, der Orientierung, der Größe eines Musters (Jähne 2002)

2.1.5 Klassifikation

In der Praxis ist ein Merkmalbild für eine korrekte Objekterkennung nicht ausreichend. Vielmehr werden mehrere unterschiedliche Merkmalbilder (z.B. Merkmalbidler für Farbe, Größe, Position etc.) erstellt und diese zu einem Merkmalsvektor zusammengefaßt. Die Klassifikation erfolgt durch einen sogenannten Klassifikator, welcher den Merkmalsvektor als Eingabe nimmt. Als Ausgabe erfolgt eine Abbildung auf eine Klasse, oder auch eine textuelle Beschreibung.



1. Vorverarbeitung, Segmentierung, Merkmalsextraktion
2. Merkmalsvektor als Eingabe für den Klassifikator
3. Ausgabe: Abbildung auf Klasse, o. textuelle Beschreibung

Abbildung 2.7: Klassifikation: Vom Orginalbild, zu Objekterkennung

2.1.6 Ilia Revout: „Design und Realisierung eines Frameworks für Bildverarbeitung“ - CamFram

Diese Diplomarbeit beschäftigt sich mit dem Design und der Implementierung eines bildverarbeitendes Frameworks. Dabei wurden folgende Anforderungen an das Framework gestellt:

1. Imagegrabbing: Das Framework stellt die über eine Kamera aufgenommenen Bilder digitalisiert zur Verfügung
2. Filterkombination: Es sollen unterschiedliche Filter zur Verfügung gestellt und diese nach belieben miteinander kombiniert werden können - daraus ergeben sich dann neue Filter (Kombifilter) etc..
3. Testen der Filter: Einzeln Filter sollen „sichtbar“ getestet werden können.
4. Wiederverwendbarkeit: Das Framework soll in beliebigen Anwendung eingebunden werden können.

5. Persistente Filter: Kombifilter sollen gespeichert und geladen werden können. Hierbei spielt die Reihenfolge der angewendeten Filter eine Rolle, welche nach Speicherung/Laden bestehen bleiben soll.

Je nach Filtertyp werden unterschiedliche Information aus einem digitalisierten Bild gewonnen. Das Ergebnis einer Filteroperation ist stets wieder ein Bild und den vom Filtertyp bereitgestellten neuen Informationen (Beispiel: Filter mit Farbschwerpunkterkennung, welcher die X- und Y-Koordinaten der Schwerpunkte bereitstellt). Das Framework unterteilt sich sich in zwei unabhängigen (d. h. strikte Trennung durch Model-View-Controller-Muster) Frameworks :

1. Framework-Kernel: Deckt die gesamte Funktionalität des Frameworks ab. Stellt Klassen zu Framework-Erweiterung bereit (neue Filter mit den dazugehörigen Dialogen). Filtermanagement etc.

2. Framework-Gui: Stellt die visuelle Filtertestumgebung bereit. Vorhandene und neu entworfene Filter können hier getestet werden - dieses setzt eine Erweiterungsfähigkeit der Gui vorraus, da neue Filter mit den dazugörigen neuen Informationen dargestellt werden müssen.

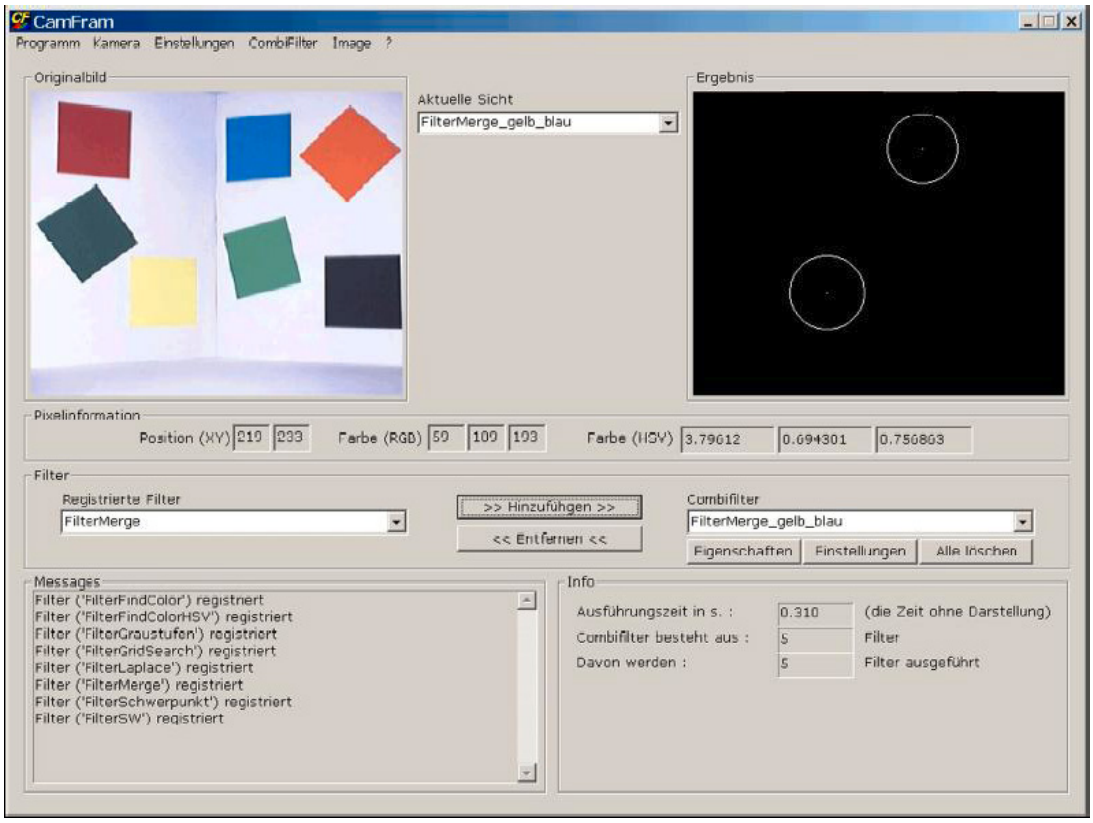


Abbildung 2.8: CamFram-Gui: Oberfläche zum Testen von Filtern - hier: Schwerpunktbestimmung der Farben Blau und Gelb

Detaillierte Informationen sind Revout (2003) zu entnehmen.

2.2 Robotik

„Roboter“ (tchechisch „robotnick“) = Leibeigener, Sklave

Abbildung 2.9: Mechanische Ente von Jacques de Vaucanson

Robotik ist die Wissenschaft von der Entwicklung und der Steuerung von automatischen Maschinen (Roboter). Die Robotik wird nicht nur in der Informatik behandelt, sondern auch in Teilgebieten der Elektrotechnik und Maschinenbau. Der tschechische Romanschriftsteller Karel Capek (1890 - 1938) verwendete das Wort „Roboter“ erstmalig in seinem 1920 erschienenen Theaterstück «Rossum's Universal Robot's». Das Stück handelt von humanoide Maschinen, die nur automatische, mechanische Tätigkeiten ausführen können und dem menschen vollendends diensbar sind. Im weiteren Verlauf des Stückes, eignen sich die «seelenlosen» Roboter immer weitere Fähigkeiten an und töten letztendlich ihre Erbauer (durch das Fehlen einer Seele?) - ein besonderes Thema in der Science-Fiction-Literatur (siehe Isaac Asimovs Gesetze der Robotik[2]). Roboter stellen eine Unterkategorie von Automaten dar. Man unterscheidet im wesentlichen zwischen modellbasierten und verhaltensbasierten Roboter.

- modellbasierte Roboter
  kennen ihre Umgebung, d. h. die Welt, in der diese Roboter agieren, ist bereits modelliert.

- verhaltenbasierte Roboter
  reagieren direkt auf äußere Einflüsse. Hier werden Verhaltensweisen vom Menschen oder Tier (animal behaviour) modelliert. Der Roboter «lernt» seine Welt kennen.

Roboter sind somit programmgesteuerte Maschinen, die, im Sinne ihrer Nutzer, in einer realen Umgebung selbständig interagieren (ähnlich verhält es sich mit (Software)Agenten, d. h. autonome Programme, die selbständig mit ihrer Umgebung interagieren, z. B. Suchmaschinen im Internet).

[2] Isaac Asmimov (1941): Three Laws of Robotics (Drei Gesetze der Robotik)

1. A robot may not harm a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
3. A robot must protect its own existence, as long as such protection does not conflict with the First or Second Law.

Abbildung 2.10: NASA (National Aeronautics and Space Administration) Roboter Dante

2.2.1 Zielsetzung

Aus ingenieurtechnsicher Sicht besteht die zu lösende Aufgabe darin, die verfügbaren Mittel, in einer dynamischen, unstrukturierten, unsicheren, und nur teilweise erfahrbaren Umgebung für einen gewünschten Zweck möglichst effizient einzusetzen. Dabei zeichnet die Umwelt folgende Charakteristika auf:

- Dynamik: Es bestehen schnelle und unerwartete Änderungen
- Keine Struktur: Es gibt keine vorgegeben Konstruktion
- Unsicherheit: Es sind teilweise unzuverlässige Daten bzw. verrauschte Daten vorhanden
- Nur teilweise Erfahrbarkeit: Nicht alle Informationen sind zugänglich

Ursachen für die Einschränkung die sich dem Roboter stellen, können grundsätzliche (unzugängliche Informationen) oder komplexitätbedingte (zu hoher Informationsbeschaffungs-Aufwand) sein. In diesem Zusammenhang spricht man auch von „Beschränkte Rationalität“.

2.2.2 Roboterkomponten: Ein kurzer Abriß

Abschließend seien noch kurz, nicht allzu detalliert, die Komponenten eines Roboters erwähnt.

- Sensoren: Erfassung der Umwelt, Erfassung interner Daten

  - mechanische (Prellkontakte, Bumper, Fühler), optische Sensoren (Infrarot, Ultraschall, Video/Bilderkennung) mit direkter Rückkopplung, Sensor-Aktor-Kopplung oder Regelung

- Prozessor(en): Verantwortlich für Wahrnehmung, Motoransteuerung, Planung, ...
  - Lokale Prozessoren, externe Prozessoren (Schwierigkeit der Kommunikation): Kriterien
    * Minimierung des Gewichts- und Energiebedarfs
    * Leistungsfähiges (einfach) zu bedienendes Interface
    * Realzeitanforderung
    * Hohe Rechenleistung (benötigt beispielsweise bei künstlicher Intelligenz)
    * Schutz vor Beschädigung (Kostenfaktor-, Sicherheits-, Zuverlässigkeitsfaktor)

- Energieversorgung
  - Externe oder interne Energieversorgung (Bewegungsfreiheit, Ladezeiten, Eigengewicht)

- Kommunikation
  - drahtlos (Funk, Infrarot, ...), Problem der Bandbreite, Störungen, ..
  - Kabel, Problem der Einschränkung der Mobilität

2.2.3 Rainer Balzerowski: „Realisierung eines Webcam basierten Kamera Systems für mobile Roboter“

In dieser Arbeit wurde ein Kamera-System realisert, welches autonome Roboter über eine Kamera mit Daten versorgt. Im Wesentlichen sollte eine schon bestehende - wenn auch schnelle, dafür aber teure - Hardware-Lösung zur digitalen Bildbearbeitung in eine software-technischen Lösung umgewandelt werden.
Hierbei wurden folgende Rahmenbedingungen festgelegt:

- Hardware-Rahmenbedingungen:
  - Das System besteht aus Komponenten des LEGO MINDSTORM Robotics Invention System (RIS),
  - RCX-Roboterboard sowie
  - LEGO Vision Command Kamera
  - Kommunikation Kamer-Server/Roboter: Infrarot-Schnittstelle

- Software-Rahmenbedingungen:
  - Server: Betriebssystem Windows 98
  - Kommunikation: LEGO Spirit Control

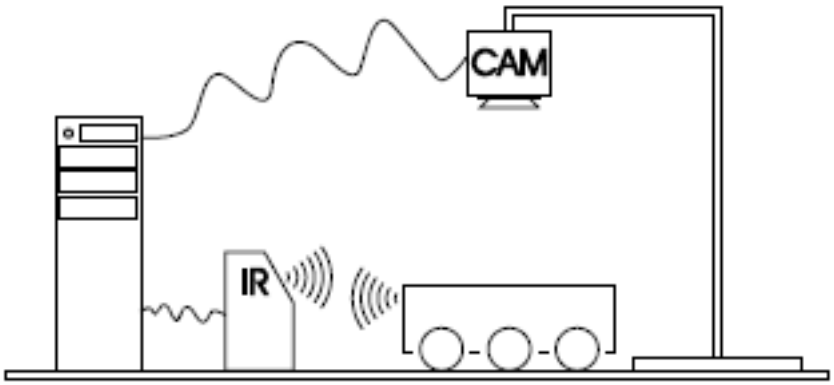


Abbildung 2.11: Konzeption

- Videodaten: Logitech Software Developement Kit
- RCX-Board: LEGO-Firmware

Die hierbei entwickelte Software „CATCH“ (Balzerowski 2002) ist in der Lage, jeweils ein Objekt einer bestimmten Farbe zu beobachten. Die Objekte können eine von den sechs Farben (sechs Farben: Rot, Grün, Blau, Gelb, Cyan, Magenta) annehmen. Die Koordinatenermittlung erfolgt über eine Schwerpunktberechnung einer Farbe - dabei wird immer das Objekt mit der größten Farbfläche einer Farbe gewählt, da pro Farbe nur ein Koordinatenpaar zur Verfügung steht. Die Übermittlung des Koordinatenpaares erfolgt über die Infrarotschnittstelle an das RCX-Board - welches die Daten gegebenfalls weiter auswerten kann.

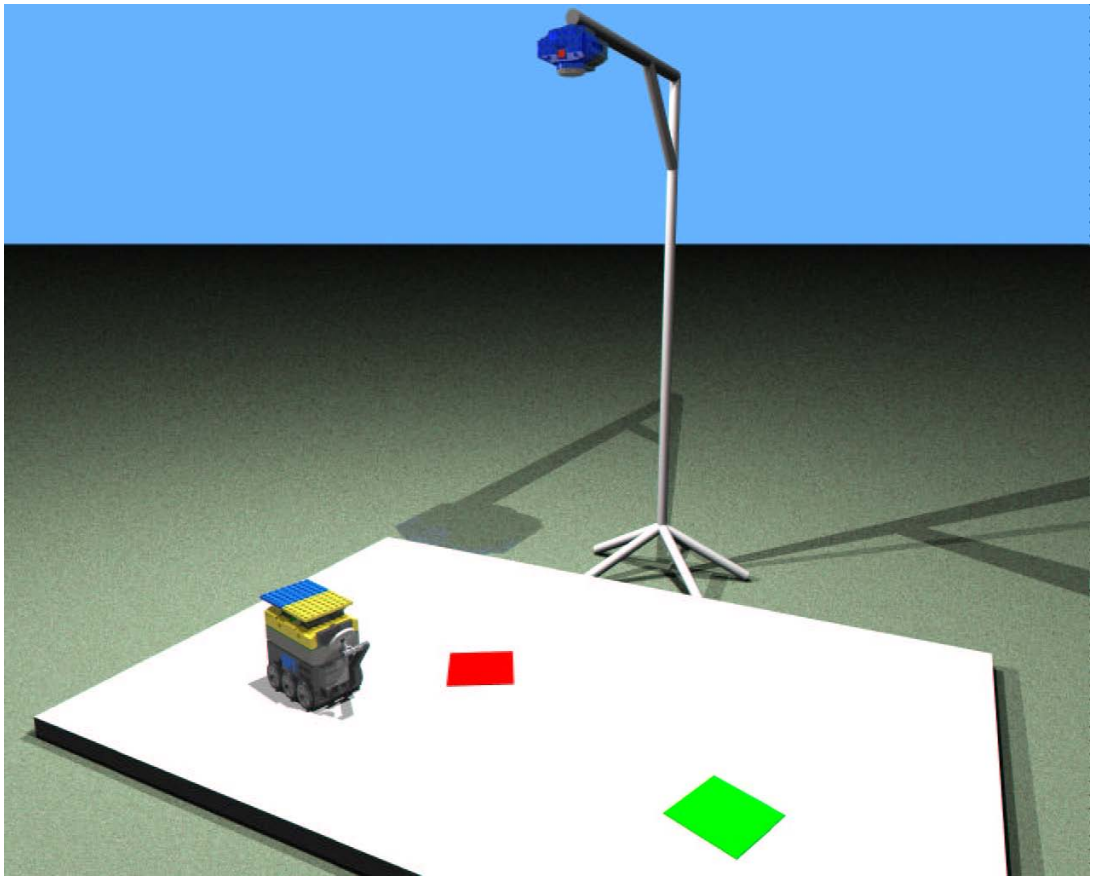


Abbildung 2.12: Versuchsaufbau

2.2.4 Mirco Gerling: „PDA-gestützte Robotersteuerung mit funkbasierter Serveranbindung“

Gegenstand dieser Arbeit ist das Design und die Realisierung eines hybriden Roboters in einer 3-Schicht-Architektur. Der Roboter soll sich in einer definierten Umgebung (Szenario: Ein Parcour mit zwei Tankstellen und Auftragserteilung) zurechtfinden und agieren können. Dabei gibt es einen verhaltenbasierten Anteil (Behandlung von Kollisionen), modellbasierten Anteil (Roboter besitzt innere Karte von der Umgebung, kennt seine eigene Position) und eine planerische Ebene (Server wickelt Auftragsverwaltung ab, stellt Informationen über den weiteren Verlauf der Fahrt des Roboters bereit).
Die Rahmenbedingungen hierfür waren:

- Hardware-Rahmenbedingungen:
  - Roboter: MIT-6.270-Board (RCX-Board),
  - PDA[3]: Compaq iPAQ H3970, mit Bluetooth-Modul für Serververbindung,

[3] **P**ersonal **D**igital **A**ssistant

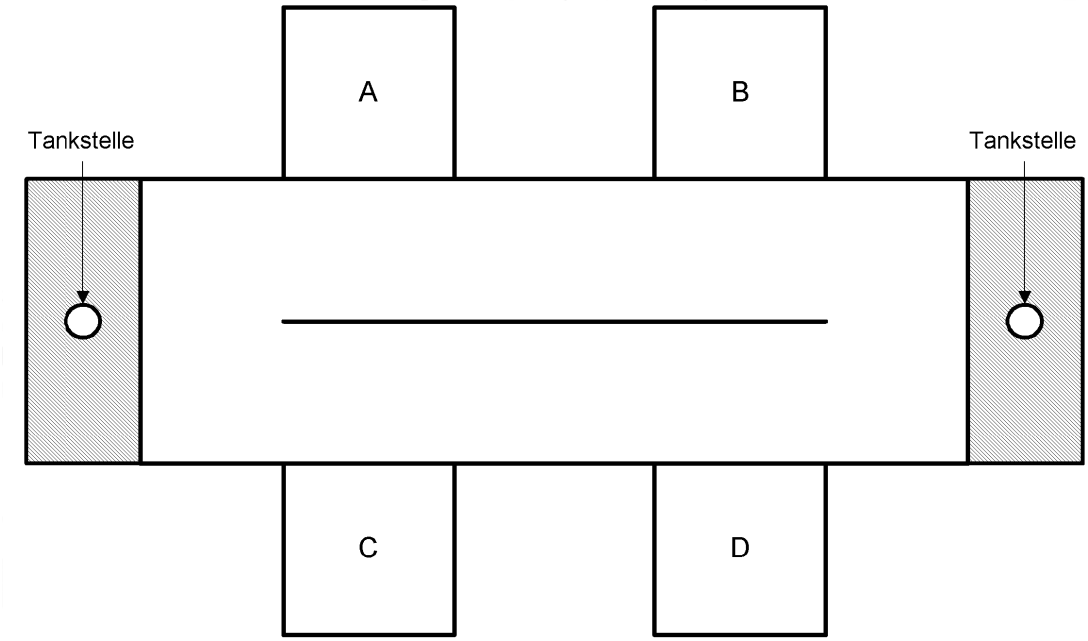


Abbildung 2.13: Szenarion: Parcour mit zwei Tankstellen und vier Zielplätze

  - Server: Blootooth-Adapter (USB-Variante)

- Software-Rahmenbedingungen:
  - Microft Visual C++ (für Serverkomponenten),
  - Microsoft Embedded Visual C++ 3.0 (für PDA),
  - Interactive C Version 3.2 von den Newton Research Lab für das RCX-Board

Die Realisierung des Gesamtsystems ist auf drei Komponenten aufgeteilt (siehe Abbildung 2.14):

1. Controller-Programm: Läuft auf dem Controller-Board (Verhaltenbasierte Logik),
2. Planning-Programm: Läuft auf dem PDA,
3. Server-Programm: Läuft auf einem PC.

Die Kommunikation zwischen Controller und PDA vollzieht sich über eine serielle Schnittstelle, die Kommunikation zwischen PDA und Server über TCP/IP (über Blootooth).

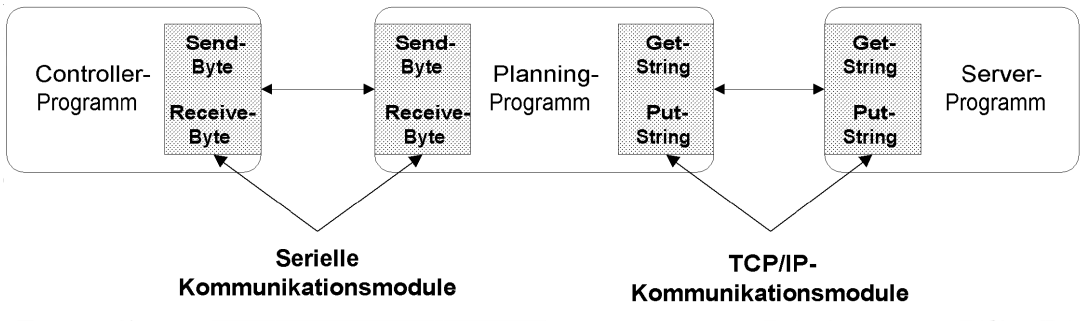


Abbildung 2.14: Die einzelnen Komponenten

3 Szenario: Robot-Soccer-League

Das Szenario spielt im Bereich der Robot-Soccer-League. Es soll eine Client/Server-Architektur für einen Kamera-Server entwickelt werden, welcher autonome Roboter mit zusätzlichen Informationen über deren Umgebung versorgt (ähnlich der Arbeit Balzeroswki Balzerowski (2002)). Welche Informationen bereitgestellt werden sollen, wird im späteren Kapiteln ausführlich erörtert. Als Ausgangspunkt dient die Soccer-Umgebung des Robo-Labs im HAW-Softwarelabor, welches im Rahmen von Robotik/AI[1]-Projekten bereits Anwendung fand.

Um einen Überblick über die Problemstellung zu geben, die sich mit der für die Soccer-Umgebung neuen Kamera-Komponente einstellt, sei nachfolgend der derzeitige und der gewünschte Zustand dargestellt.

3.1 Die derzeitige Soccer-Umgebung

Die Soccer-Umgebung besteht aus einem (x mal x) großen Spielfeld. Dieses Spielefeld hat, wie im Fußball üblich, zwei Tore und besitzt zusätzlich als Sicherheitsabgrenzung eine dem Spielfeld umgebende Wand. Mitspieler hierbei sind zwei oder n^2 autonome Roboter. Die Robotor können ihre Umgebung durch Tast-Sensoren und Infrarot-Sensoren wahrnehmen. Der Ball selbst besteht aus roten LEDs, welche in allen Richtungen Signale sendet. Ziel des Spieles ist es, eine höhere Toranzahl zu erzielen als der Mitspieler.
Die derzeitigen Rahmenbedingungen sind:

- Hardware:
  - Roboter: RCX-Board, Sensoren, Aktuatoren
  - PDA (HP iPAQ Pocket PC H5500)
  - Kommunikation: Infrarot-Schnittstelle, Bluetooth, Wireless LAN
  - Server: Blootooth-Adapter (USB-Variante), Wireless LAN
- Software-Rahmenbedingungen:
  - Microft Visual C++.2003 (für Serverkomponenten),
  - Microsoft Embedded Visual C++ 3.0 (für PDA),
  - Interactive C Version 3.2 von den Newton Research Lab für das RCX-Board
  - Kommunikation: Über Sockets

[1] Artificial Intelligenz - künstliche Intelligenz

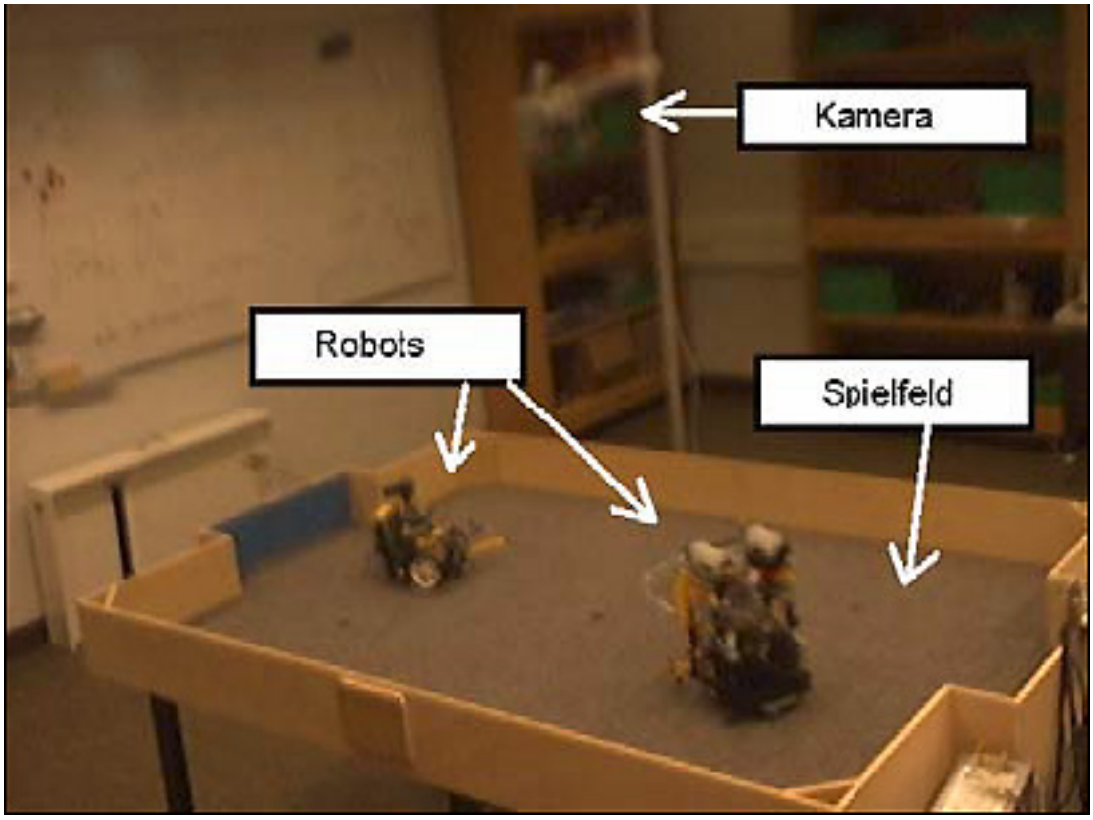


Abbildung 3.1: Soccer-Umgebung (Schmidt 2005)

Für die Robotsteuerung respekive Implementierung der künstlichen Intelligenz (KI) sind es mehrere Alternativen vorstellbar:

1. Die Steuerung erfolgt direkt über das RCX-Board. Modelbasierte und/oder verhaltensbasierten Komponenten befindet sich auf dem Board. Eine Kommunikation mit einem Server wäre dementsprechend nicht notwendig.
2. Die Steuerung erfolgt über einen PDA, d. h. die verhaltenbasierte Komponente befindet auf dem RCX-Board, die modellbasierte Komponente ist auf dem PDA realisiert.
3. Auslagerung der Planung, der KI insgesamt auf einem Server - Kommunikation zwischen Roboter/Server erfolgt über Infrarot oder TCP/IP (Bluetooth, siehe Gerling (2003)).

3.2 Die neue Soccer-Umgebung

Die Hardware- und Softwarekomponenten sind die Gleichen, nur wird hier zusätzlich eine Kamera über das Spielfeld gehängt, welche mit dem Kamera-Server verbunden ist. Der Kamera-Server verarbeitet ausschließlich Farbinformation [2]; dementsprechend müssen die Roboter re-

[2] filter

spektive Objekte farblich markiert werden. Die Roboter haben somit die Möglichkeit über den Kamera-Server zusätzliche Informationen, wie z. B. Positionsangaben von Mitspielern, Tornähe, Ballposition zu erhalten und diese auszuwerten. Die Kommunikation zwischen Kamera-Server und Roboter ist folglich komplexer und anspruchsvoller geworden.

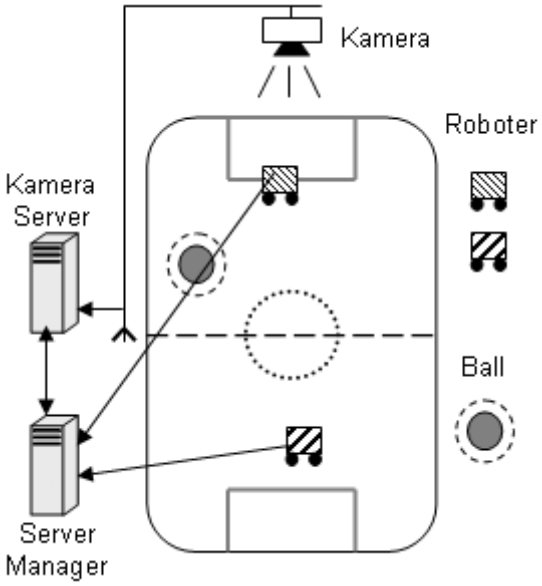


Abbildung 3.2: Robotor-Fussball-Umgebung: Kamera-Server und farbige Roboter (hier schraffiert dargestellt)

Die Rahmenbedingungen haben sich ein wenig geändert.

- Hardware:
  - Roboter: RCX-Board, Sensoren, Aktuatoren
  - Kommunikation: WLan (wireless LAN)
  - PDA (HP iPAQ) mit WLan-Komponente
  - Kamera: Geeignete Kamera, d. h. ausreichend große Bildrate.
  - Server: Ausreichend große Leistungsreserven für die Verarbeitung der Bilder
- Software-Rahmenbedingungen:
  - Microft Visual C++.2003 (für Serverkomponenten),
  - Microsoft Embedded Visual C++ 3.0 (für PDA),
  - Interactive C Version 3.2 von den Newton Research Lab für das RCX-Board
  - Bildverarbeitung: Leistungsstarkes Bildverarbeitungsprogramm, welches die von der Kamera gelieferten Bilder auswertet

  - Kommunikation: Geeignete Middleware für verteilte Anwendung wie Corba, .Net, JEEE

Durch die neuen Komponente ergeben sich u. a. folgende neue Aspekte, die eine genauere Untersuchung bedürfen:

- Anforderungen des Anwenders: Welche Information soll bereitgestellt werden.
- Kommunikationsanforderung: Welcher Art ist die Kommunikation zwischen Roboter und Server. Verschiedene Middlewares stehen hier zur Verfügung (.NET, CORBA, J2EE). Soll die Kommunikation ereignis- oder anfragegesteuert erfolgen.
- Serverarchitektur: Der Server muss ein großes Bildvolumen verarbeiten. Welche Lösungsansätze wären geeignet? Beispiel Grid-Computing, Distributed Computing.

Dieses stellt natürlich nur eine kleinen Überblick dar.

3.3 Neue Anforderungen

Das Projekt teilt sich in vier Bereiche auf:

1. Server-Manager-Bereich, verantwortlich für Client-Anfragen, Organisiert und stellt Dienst (Services) bereit.
2. Client-Roboter-Bereich, Kommunikationsschnittstelle zu dem Server-Manager
3. Server-Bildbearbeitungsbereich, Aufgabenverteilung der Bildverarbeitung (z. B. bei Grid-Computing)
4. Client-Bildbearbeitungsbereich, Auswertung der Bilder (wird benötigt, wenn die Bildverarbeitung und somit die Last auf verschieden PC verteilt werden soll)

Das Zusammenspiel dieser Bereiche könnte sich wie folgt darstellen:
Der Roboter/Client meldet sich bei dem Server-Manager für einen bestimmten Service an - beispielsweise gegnerische Positionsabfrage - und wir dementsprechend registriert. Der Client-Roboter stellt eine Positionsanfrage an den Server-Manager (alternativ ist auch eine ereignisgesteuerte Positionsvergabe, vom Server-Manager zum Robot denkbar). Der Server-Manager stellt seinerseits eine Anfrage an den Kamera-Server, welcher aktuelle Positionsdaten einzelner Objekte bereitstellt (auch hier wäre eine ereignisbasierende Benachrichtung - Observer Pattern[3] - denkbar. Simultan verarbeitet der Kamera-Server fortwährend die von der Kamera geliefert Bilder. Wie der Abbildung zu entnehmen ist, wird ein Server benötigt, hier der Server-Manager, welcher Dienste und Daten bereitstellt sowie sämtliche angemeldete Roboter-Clients organisiert. Zwischen den verschiedenen Bereichen entstehen also unterschiedliche Kommunikationswege:

- Kommunikation zwischen Roboter und Server-Manager

[3] Oberver Pattern

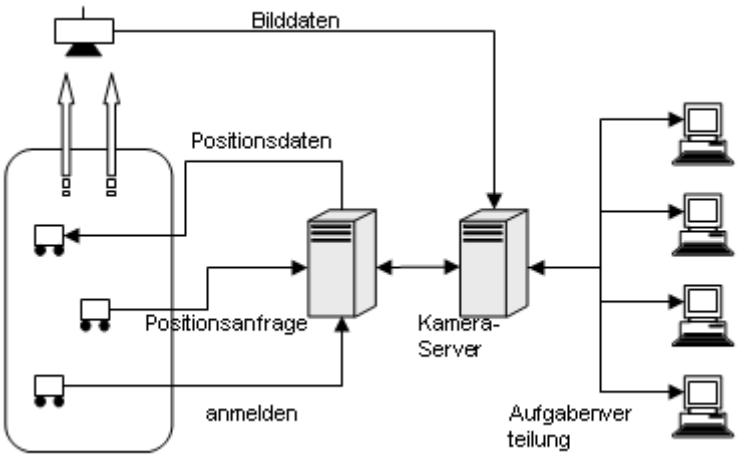


Abbildung 3.3: Roboter-Fussball-Umgebung

- Server-Manager und Server-Bildbearbeitung
- Server- und Client-Bildbearbeitung, d. h. Unterteilung und Verteilung der Bildverarbeitung an z. B. anderen PC's im Netzwerk

Die Anforderungen beziehen sich sowohl auf die zur Verfügung stehenden Hardware (Kamera, PDA-Robots, PC-Anzahl, spezielle Hardware für Bildbearbeitung z. B. Video-Capture-Karte) als auch Software (Kommunikations-Framework, Programmiersprache, usw.). z. B. bedeutet eine leistungsstarke Kamera mit hoher Bildfrequenz einen dementsprechend hohen Leistungsanspruch an die Video-Capture-Karte. Software-Seitig sei hier als Beispiel eine gewünschte Programmiersprachenunabhängigkeit des Client-Roboter zu nennen.

Diese Studienarbeit beschäftigt sich im weiteren Verlauf hauptsächlich mit der Kommunikation zwischen den einzelnen Komponenten und geht der Frage nach, welche Frameworks dieses am besten realisieren.

4 Kommunikationsschicht und Middleware

Für die Kommunikation zwischen Roboter/Server-Manager, Server-Manager/Kamera-Server stehen unterschiedliche Frameworks zur Verfügung. Mit Hilfe dieser Frameworks soll eine geeignete Middleware konzipiert werden, die gewissen Anforderungen und Kriterien genügt. Wir bewegen uns in dieser Studienarbeit im Bereich der „Verteilten Anwendung“. Die in Frage kommenden zu untersuchenden Frameworks wären:

- .Net-Framework (Teile)
- Corba
- Java 2 Enterprise Edition
- Adaptive Communication Enviroment (ACE)

Welche Anforderung muß an der Middleware gestellt werden? Im nächsten Abschnitt werden einige Kriterien und mögliche Anforderungen, jeweils für die beiden oben angegebenen Kommunikationswege, aufgezeigt. Danach werden die zur Verfügung stehenden Frameworks beschrieben - in kleineren Rahmen.

4.1 Kriterien und Anforderung an die Middleware

Um vernünftige Anforderung definieren zu können, müssen vorerst noch die Komponenten, Hardware wie Software, genannt werden, die hier Anwendung finden werden. In Abbildung 3.3 wurde bereits ein abstrakte Konzeption vorgestellt. Eine konkretere, mit ersten Anforderungen, ist in Abbildung 4.1 dargestellt.
Es existieren also drei Akteure:

1. Roboter ,welcher Dienste des Server-Manager in Anspruch nehmen möchte
2. Server-Manager, stellt Dienste bereit und bezieht Informationen vom Kamera-Server
3. Kamera-Server, stellt Informationen bereit und verteilt Aufgaben an Worker-Clients (PC-Pool)

Die einzelnen Komponente haben unterschiedliche Anforderungen an die Teilsysteme (Aufteilung in OSI-Schichten[1] erfolgt vorerst nicht):
Wie aus der Tabelle zu ersehen ist, soll eine möglichst große Plattform- und Programmier-

[1] engl. **O**pen **S**ystems **I**nterconnection **R**eference **M**odel ist ein offenes Schichtenmodell, das seit den 70er Jahren entwickelt und standardisiert wurde. Es teilt die verschiedenen Problembereiche der Netzwerkkommunikation-kommunikation in sieben Schichten auf, die aufeinander aufsetzen

Komponente	Hardware/Software	Anforderung
Roboter	PDA	frei wählbar
	Betriebssystem	frei wählbar
	Kommunikation	WLan für schnellere Kommunikation
	Client-Programmierung	Anwender kann Programmiersprache frei wählen
Server-Manager	PC-Leistung	CPU[2]: mittel, RAM[3]: mittel
	Betriebssystem	frei wählbar
	Kommunikation	WLan (Client-Seitig)
	Server-Programmierung	Implementierung in C++ oder Java
Kamera-Server	PC-Leistung	CPU: hoch, RAM: groß
	Betriebssystem	Windows XP/2000
	Frameworks	CamFramRevout (2003)
	Programmiersprache	C++

Tabelle 4.1: Anforderungen der einzelnen Komponenten

sprachenunabhängigkeit erzielt werden. Für den Roboter kann jedes PDA-Modell mit frei wählbaren Betriebssystem benutzt werden. Des Weiteren soll der Entwickler der Roboterkomponente eine Programmiersprache seiner Wahl einsetzen dürfen. Die klare Trennung zwischen Server-Manager und Kamera-Server soll die Unabhängigkeit der beiden Komponenten hervorheben - der Austausch beider Komponente soll jeder Zeit gewährleistet sein. In der Tabelle wurde beiden Server jeweils ein PC respektive Workstation zugeordnet. Natürlich ist auch eine Lauffähigkeit beider Server auf einer Workstation/PC denkbar - die erste Konzeption geht aber von einer hohen Resourcenanforderung des Kamera-Servers aus. Der Server-Manager ist als Dienstleister für den Roboter anzusehen und ist, im weiteren Sinne, eine Kommmunikationsschnittstelle zwischen Roboter und Kamera-Server. Wie schon erwähnt, soll auch eine möglichst lose Kopplung zwischen Server-Manager und Kamera-Server erreicht werden. Die einzige größere Restriktion, die uns hier begegnet, ist die Plattform- und Programmiersprachenabhängigkeit von CamFram, dem FrameWork zur Bildverarbeitung - hierfür ist WinXP und Visual C++ (MFC[4]) notwendig. Spezifischere Abhängigkeiten wie zum Beispiel Kompilierungsart in Visual C++ (statische, nicht statische etc.) werden erst einmal nicht berücksichtigt.

4.2 Frameworks

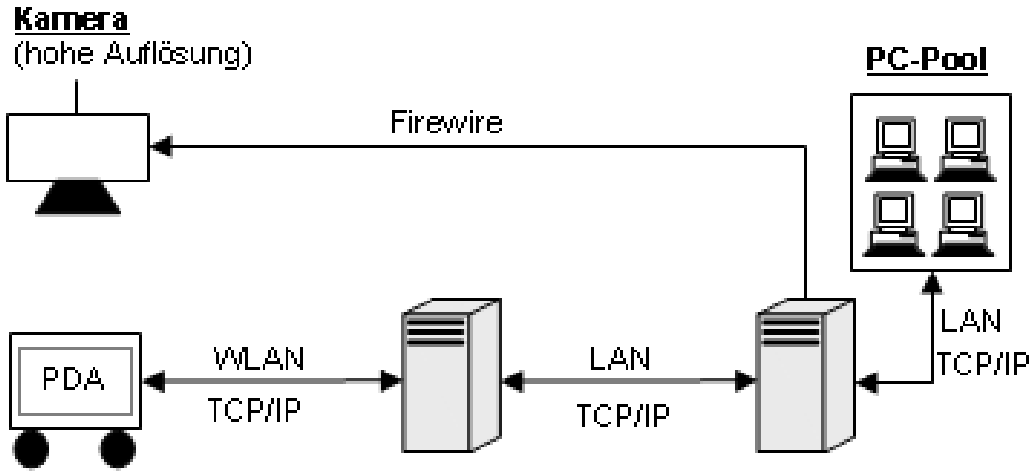
Nachfolgend sei eine kurze Beschreibung der Frameworks gebeben, welche für die Anwendung in Frage kämen.

4.2.1 Microsoft .NET

Folgende Intention stehen hinter .Net:

- Eine globale (Internet mit Inbegriffen) Zusammenarbeit der Applikationen

[4] **M**icrosoft **F**oundation **C**lass - spezielle Microsoft Klassen-Bibliotheken speziell für eine einfachere Anwendungsentwicklung



Roboter	Server-Manager	Kamera-Server
▪ PDA ▪ RCX-Board	▪ PC	▪ leistungsstarker PC ▪ Alternativ: Verteilung
▪ LINUX ▪ Windows ▪ Apple	▪ Windows ▪ Linux	▪ WinXP
▪ C++ ▪ Java ▪ ...	▪ C++ ▪ Java ▪ ...	▪ CamFram (Revout 2003) ▪ C++

Abbildung 4.1: Erste Konzeption: Hardware, Betriebssystem, Software

- Zur Verfügungstellung eines offenen Systems durch Standards

.Net besteht im wesentlichen aus:

- einen .NET-Framework: Einen allen Sprachen gemeinsamen Laufzeitsystem (Typsystem, Just-In-Time-Compiler[5], Garbage Collector[6] und Bibliotheken
- .NET Web Services
- .NET Web Server

Das Laufzeitsystem setzt unmittelbar auf das Betriebssystem auf. Bei jeder .NET-Sprache (VB.NET, C-Sharp.NET, C++.NET u. a.) wird, ähnlich dem Java-Byte-Code, vorerst ein Zwischen-Code (Intermediate Language Code) erzeugt. Dieser Zwischencode ist in jedem Laufzeitsystem, welches für ein Betriebssysteme zur Verfügung stehen muß, lauffähig. Das Laufzeitsystem übersetzt den Zwischencode in Maschinencode und ist u. a. verantwortlich für Speicherverwaltung, Sicherheit, Starten/Stopen von Prozessen etc.. Ein wichtiges Konzept ist „.NET Web

[5] Compiler, welcher den Code (byte-Code) bei einem ersten Programmaufruf einmalig übersetzt
[6] verantwortlich für Speichmanagement, Löschen von nicht mehr benötigten Objekte etc.

Service“. Web Services sind Dienste, die über das Internet an jedes Gerät geliefert werden, auf offene Standards wie HTTP (**H**yperText **T**ransfer **P**rotocol), SOAP (**S**imple **O**bject **A**ccess **P**rotocol), XML (e**x**tensible **M**arkup **L**anguage) basieren und keine Bindungen an Programmiersprachen haben. .NET stellt also eine Strategie dar, welche die Entwicklung, Pflege und Einsatz von (globalen) Anwendungen erleichtern soll. Für die zu entwickelnde Anwendung ist in erste Linie das Konzept der Web-Services interessant (Server-Manager als Web-Service-Anbieter für die Roboter).

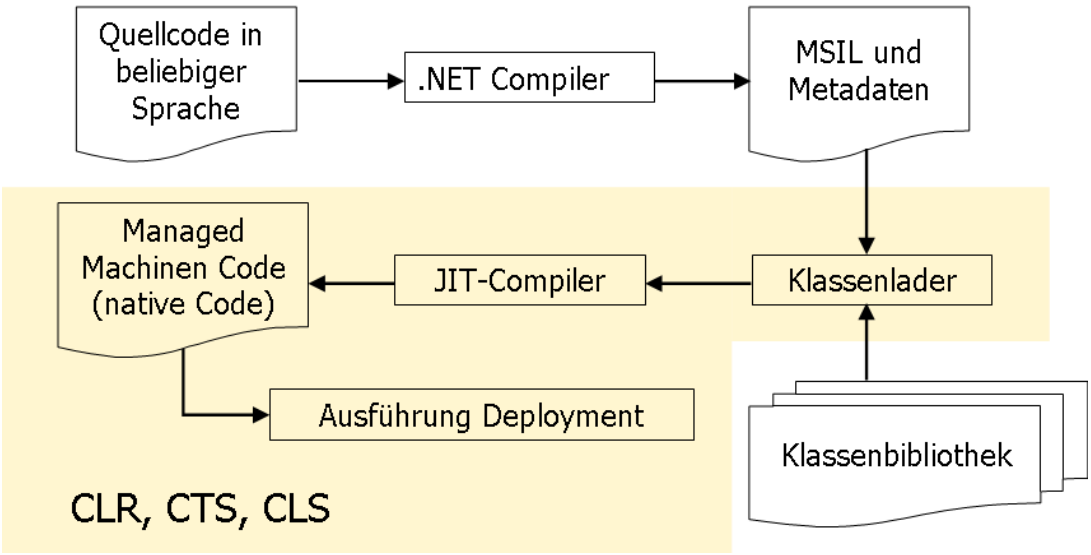


Abbildung 4.2: Entwicklen mit .NET: CLR (Common Language Runtime, CTS (Common Type System), CLS (Common Language Specification)

4.2.2 Java 2 Enterprise Edition, Sun ONE

Sun’s **O**ne **N**et **E**nvironment ist .NET sehr ähnlich und ist für die schnelle Entwicklung von Services ausgelegt. Sun ONE basiert auf Standards wie HTTP, SOAP, XML und verwendet als Haupttechnologie JAVA.

Für die Anwendung selbst ist auch, neben dem abstrakteren SUN ONE-Framework, JAVA selbst und dessen Network-Framework sowie RMI[7] (**R**emote **m**ethod **i**nvocation) interessant.

4.2.3 CORBA

CORBA steht für **C**ommon **O**bject **R**equest **B**roker **A**rchitecture und ist ein Industriestandard, der die Architektur verteilter objektorientierter Anwendungen vereinheitlich. Basis ist die von

[7] engl. Remote Method Invocation (Entfernter Methodenaufruf): Das Objekt, welches die aufzurufende Methode besitzt, muß sich nicht auf der gleichen Maschine befinden, sondern kann irgendwo im Netzwerk (in einer virtuellen Maschine) existieren.

OMG[8] spezifizierte Interface Definition Language (IDL) und das Internet Inter-ORB-Protocol. CORBA:

- löst das Problem der heterogenen Plattformabhängigkeiten (wie Programmiersprache, Betriebssystem, Netzwerkprotokolle, Hardware), welche sich Applikationen gegenüberstehen.
- ist eine Familie von Spezifikation (OMG, siehe Abbildung4.3)
- definiert Schnittschstellen (Interfaces), keine Implementation
- vereinfacht die Entwicklung von verteilten Anwendungen durch Automatisierung/Kapselung von:
  - Objektlokation
  - Verbindungs- und Speicherverwaltung
  - Parameter (de)marshaling, d. h. einpacken und auspacken von Objekten (Parameter)
  - Event- und Requestmultiplexing
  - Fehlerbehandlung, Fehlertoleranz
  - Objekt/Serveraktivierung
  - Nebenläufigkeit
  - Sicherheit

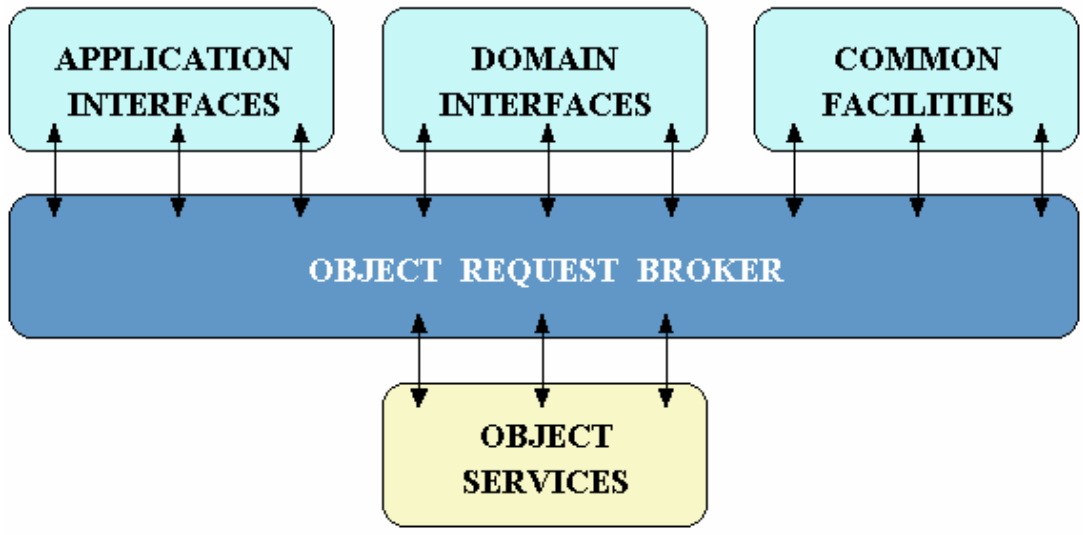


Abbildung 4.3: OMG Referenz Modell Architektur (Schmidt 2003)

[8] Object Management Group

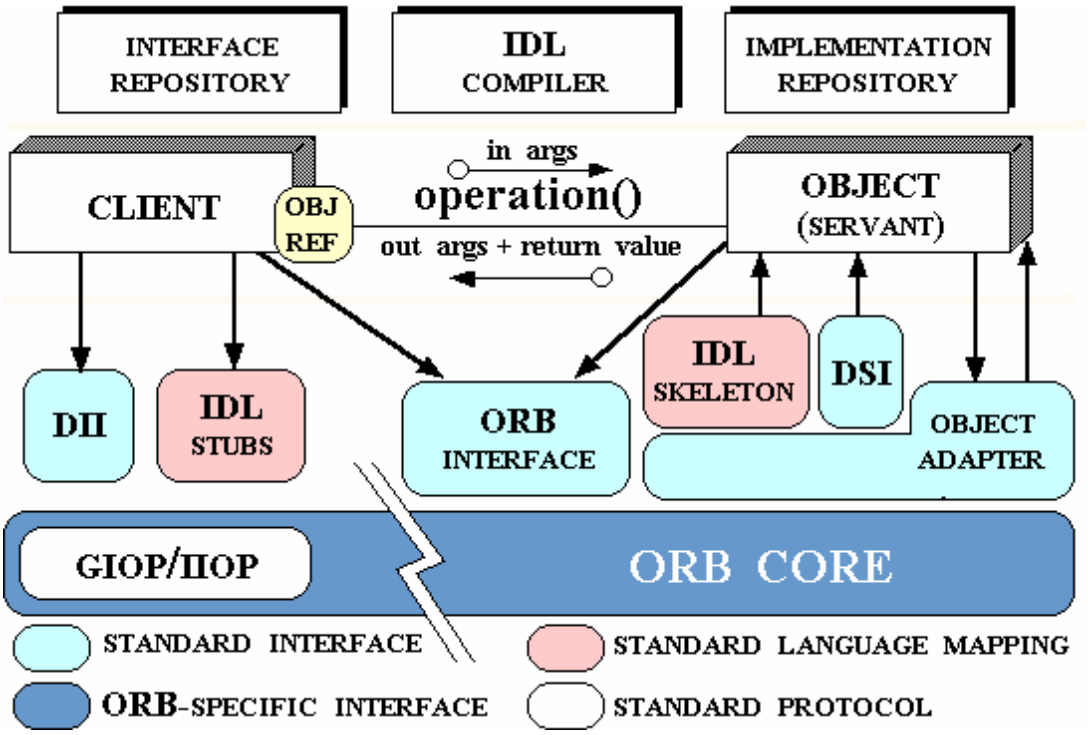


Abbildung 4.4: Corba-Architektur (Schmidt 2003)

Des Weiteren enthält Corba eine Sammlung von Diensten, die sogenannten „Corba Services“, welche die Entwicklung von Corba-Anwendungen unterstützen sollen. Derzeit hat die OMG 15 Standardobjektdienste spezifiert: Tatsächlich haben bisher die wenigsten Corba-Distributionen

1.	Life Cycle	9.	Persistence
2.	Naming	10.	Event
3.	Concurrency Control	11.	Transaction
4.	Relationship	12.	Externalization
5.	Query	13.	Licensing
6.	Properties	14.	Time
7.	Security	15.	Trader
8.	Collection		

Tabelle 4.2: Corba-Services

alle Dienste implementiert. Interessant für die zu entwickelnde Anwendung wäre der Naming-Service (z. B. Registrierung von Robot-Clients), Event-Service (Observer-Pattern: Roboter werden automatisch benachrichtig, wenn sich ein beobachtendes Objekt bewegt) und Security-Service.

4.2.4 Network-Framework in C++

Eine weitere Altenative wäre die Benutzung eines C++-Network-Framework - ähnlich Java-Network-Framework. Als Beispiel ist hier das Adaptive[9] Communication Enviroment (ACE (Schmidt und Huston 2001, 2002) zu nennen. ACE baut im wesentlichen auf native C-Methoden (Ansi-C und Betriebssystemspezifisch Bibliotheken z. B. von unix, windows) auf. Sämtliche „Netzwerk-Methoden“ (Sockets-aufbau u. a.) werden hierbei durch Wrapper typisiert und, in Bezug auf Unix, Windows als weiterverbreiteste Betriebssysteme, standardisiert. Wie in Abbildung 4.5 zu sehen ist, gibt es bei ACE mehrere Schichten:

- OS Adaptation Layer: Vereinheitlich die unterschiedlichen C-Bibliotheken (z. B. besitzen UNIX und Windows unterschiedliche Prozessverwaltung).
- C++ Wrapper Facade Layer, typisiert den OS Adaption Layer und vereinfacht die Entwicklung von Netzwerkapplikation. Durch die Typisierung werden viele Fehlerquellen, die sich aus einer C-Applikationsentwicklung ergeben können, beseitigt (Beispiel: Speicherverwaltung). Quellcode, der z. B. unter Windows entwickelt wird, läßt sich auch auf einer UNIX-Maschine kompilieren
- Framework-Layer: Verschieden Frameworks vereinfachen nochmals die Entwicklung von Netzwerk-Applikation.

Der Vorteil bei ACE liegt in der Geschwindigkeit der damit entworfenen Applikationen - da hier letztendlich native C-Routinen benutzt werden. ACE verbreitet sich immer mehr und wurde bereits adaptiert für: Windows, Unix, Linux, OpenBSD, Macintosh OS X, div. Solaris. Ein Nachteil ist hierbei, dass für die Entwicklung von Applikation nur C benutzt werden kann, d. h. eine Programmiersprachenabhängigkeit besteht.

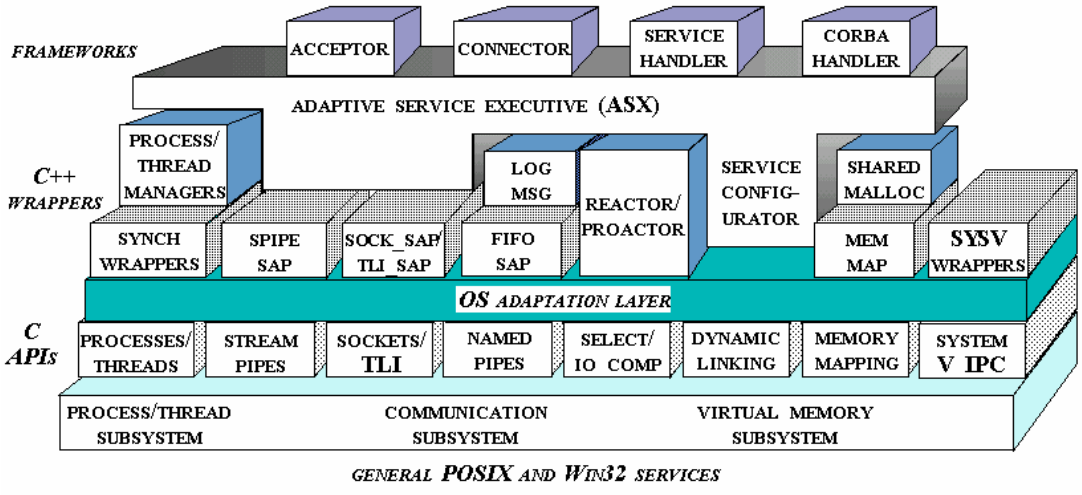


Abbildung 4.5: Adaption Communication Environment: Architektur

[9] engl. anpassungsfähig

5 Resümee

Eine Hauptanforderung an die Anwendung ist die Programmiersprachen- und Plattformunabhängigkeit - insbesondere soll für die Roboterkomponente jede PDA-Plattform und auch Programmiersprache benutzt werden können . Eine weitere Anforderung ist die Geschwindigkeit der Kommunikation einzelner Komponenten untereinander. Für den Kamera-Server muss für Kommunikationsschicht C++ benutzt werden, da dieser mit Visual C (Micosoft) für Windows entwickelt wurde.

Zur Zeit bietet Corba die größte Programmiersprachen- und Plattformunabhängigkeit der aufgezeigten Möglichkeiten und ist somit die erste Wahl. Eine .Net-Variante wurde bereits von Oliver Schmidt (Schmidt 2003) konzipiert. Je nach Anforderung an die Geschwindigeit kann ein Java-basiertes Corba oder C++-basiertes-Corba genommen werden. Entschieden habe ich mich für JacOrb (frei verfügbar, open source) und The ACE Orb (TAO). TAO basiert auf das weiter oben beschriebene ACE-Framework und ist ein Real Time CORBA. Realtime CORBA ist eine Erweiterung des CORBA-Standards, um verteilte Applikationen mit vorhersagbarem Echtzeitverhalten implementieren zu können. Besonders interessant ist diese Spezifikation für die Übertragung von Sprach- und Bilddaten, aber auch für Automatisierungslösungen, welche hartes Echtzeitverhalten erfordern.

Literaturverzeichnis

ADB 2005 ADB, Arbeitskreis Digitale B.: *Aufgabenbereiche*. 2005. – URL http://www-in.fh-swf.de/forschung/ak-bv/ak-bv.htm

Balzerowski 2002 BALZEROWSKI, Rainer: Realisierung eines Webcam basierten Kamera Systems für mobile Roboter. (2002), Januar. – URL http://users.informatik.haw-hamburg.de/~lego/Projekte/Balzerowski/diplomarbeit-www.pdf

Gerling 2003 GERLING, Mirco: PDA-gestützte Robotersteuerung mit funkbasierter Serveranbindung. (2003), Dezember. – URL http://users.informatik.haw-hamburg.de/~kvl/gerling/diplom.pdf

Hermes 2004 HERMES, Thorsten: Einführung in die digitale Bildverarbeitung. (2004), Juni. – URL http://www.informatik.uni-bremen.de/~hermes/home.html

Jähne 2002 JÄHNE, Prof. Dr. B.: *Digitale Bildverarbeitung*. Bd. 1. 5. Auflage. Heidelberg : Springer-Verlag Berlin Heidelberg New York, 2002. – URL http://www.springer.de. – ISBN 3-540-41260-3

Revout 2003 REVOUT, Ilia: Design und Realisierung eines Frameworks für Bildverarbeitung. (2003), Dezember. – URL http://users.informatik.haw-hamburg.de/~kvl/revout/diplomarbeit.pdf

Sanyo-Fisher 2005 SANYO-FISHER: *LCD-Verfahren*. 2005. – URL http://www.sanyo.de/

Schmidt 2003 SCHMIDT, Douglas C.: Developing Distributed Object Computing Application with Corba. (2003), Dezember. – URL http://www.cs.wustl.edu/~schmidt/corba-overview.html

Schmidt und Huston 2001 SCHMIDT, Douglas C. ; HUSTON, Stephen D.: *C++ Network Programming, Volume 1: Mastering Complexity with ACE and Patterns*. Bd. 1. 1. Auflage. Boston : Addison Wesley, Dezember 2001. – ISBN 0-201-60464-7

Schmidt und Huston 2002 SCHMIDT, Douglas C. ; HUSTON, Stephen D.: *C++ Network Programming, Volume 2: Systematic Reuse with ACE and Frameworks*. Bd. 2. 1. Auflage. Boston : Addison Wesley, Oktober 2002. – ISBN 0-201-79525-6

Schmidt 2005 SCHMIDT, Oliver: Studienarbeit: Entwicklung und Aufbau eines Kamera-Service basiert auf .net-Technologie. (2005), Februar. – URL

http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/studien/schmidt.pdf