Eine konfigurierbare Systemarchitektur zur geometrisch-topologischen Exploration von Innenräumen*

S. Blum, T. Einsele, A. Hauck, N. O. Stöffler, G. Färber
Lehrstuhl für Realzeit-Computersysteme

T. Schmitt, C. Zierl, B. Radig
Lehrstuhl für Bildverstehen und Wissensbasierte Systeme

Technische Universität München
blum@rcs.ei.tum.de

Kurzfassung. Die robuste, effiziente Exploration von unpräparierten Innenräumen mit mobilen Servicerobotern erfordert eine Systemarchitektur, die die verschiedenen Sensoren, Aktoren und Verarbeitungsmodule flexibel, situations- und missionsabhängig einsetzt. Der Beitrag beschreibt die für das Forschungsprojekt EvI (Exploration von Innenräumen mit optischen Sensoren) entwickelte Architektur, die vorhandene und neue Techniken zur Sensordatenverarbeitung zum Aufbau eines Umgebungsmodells modular integriert. Anhand eines Szenarios wird eine exemplarische Konfiguration der Architektur vorgestellt.

1 Einleitung

Damit ein mobiler Roboter in unpräparierten Innenräumen wie Wohnungen oder Büros Serviceaufgaben durchführen kann, benötigt er ein adäquates Modell seiner Umgebung. Je nach Teilaufgabe und damit verbundenen sensorischen Perzeptionen wurden hierfür bereits eine Vielzahl von unterschiedlichen Modellierungstechniken vorgeschlagen [5, 6, 17, 23]. Zur lokalen Pfadplanung und Hindernisvermeidung werden geometrische Modelle benötigt. Für die Bewegungssteuerung einer mobilen Plattform können 2D- oder $2\frac{1}{2}$D-Karten ausreichend sein; werden Manipulationen wie das Öffnen von Türen oder Schränken notwendig, sind dreidimensionale Beschreibungen von Geometrie und Kinematik erforderlich (CAD-Modelle). Für die globale Planung von Missionen haben sich abstraktere Beschreibungen bewährt, wie z. B. Topologiegraphen zur Darstellung der möglichen Fahrtrouten. Zur Kommandierung des Roboters auf möglichst benutzernahem Abstraktionsniveau („Hole eine Tasse Kaffee aus der Küche“) müssen relevante Modellelemente auch mit

* Die vorliegende Arbeit wurde im Rahmen des Projekts *Exploration von Innenräumen mit optischen Sensoren auf mehreren, aufgabengerechten Abstraktionsebenen* (Förderungsnummer Fa109/14-1) von der Deutschen Forschungsgemeinschaft (DFG) gefördert.

symbolischer Information wie Objektnamen, Raumnummern und funktionellen Attributen verknüpft sein. Da während der eigentlichen Ausführung einer Aufgabe eine autonome Interpretation der aktuellen sensorischen Informationen erforderlich ist, um Störungen und Umgebungsänderungen zu erkennen und zu berücksichtigen, muß die Modellierung auch die Interpretation von Sensordaten unterstützen. Einer der hier möglichen und bewährten Ansätze ist ebenfalls die Verwendung von geometrischen Objekt- und Umgebungsmodellen, die je nach Sensor und Perzeptionsaufgabe um spezielle Merkmale erweitert werden.

Ziel des diesen Arbeiten zugrundeliegenden Projektes EvI („Exploration von Innenräumen mit optischen Sensoren auf mehreren, aufgabengerechten Abstraktionsebenen“) ist es, ein Umgebungsmodell, das all diesen Anforderungen Rechnung trägt, autonom von einem mobilen Roboter aufbauen zu lassen. Kapitel 2 beschreibt die zugrundeliegende Modellstruktur. Damit Information auf allen nötigen Abstraktionsebenen exploriert werden kann, liegt dabei einer der Schwerpunkte auf der Verwendung von Videosensorik. Dazu ist das Zusammenspiel einer Vielzahl von Modulen nötig, die Sensordaten vorverarbeiten, abstrahieren und teilweise parallel nach verschiedenen Kriterien und mit unterschiedlichsten Algorithmen interpretieren. Hierbei wird ein effizientes Management der Datenflüsse und Systemressourcen erforderlich, um Synergien zu nutzen und Konflikte aufzulösen. Auch kann je nach Explorationsphase eine dynamische Umkonfiguration erforderlich werden. Um eine uniforme Einbindung aller beteiligten Verarbeitungsmodule zu ermöglichen, wurde eine generische, CORBA-basierte Systemarchitektur entworfen, die in Kapitel 3 vorgestellt wird. Darauf aufbauend beschreibt Kapitel 4 das Zusammenspiel einer Reihe von Verarbeitungsmodulen beim Aufbau eines exemplarischen Szenenmodells.

2 Struktur des zu explorierenden Modells

Basierend auf früheren Arbeiten zur modellgestützten Sensordateninterpretation [10, 20], wurde eine hybride Modellstruktur entwickelt (*GEM - Generalized Environmental Model*), die die folgenden Ebenen enthält:

a) Objektmodelle: Diese Ebene enthält geometrisch-kinematische CAD-Modelle aller missionsrelevanten Objektklassen, d. h. der Objekte, mit denen der Roboter interagieren muß und die ihm namentlich bekannt sind.
b) Geometrische Modellinseln: Beschreibungen missionsrelevanter Bereiche, bei denen eine genaue Kenntnis der Geometrie nötig ist. Hierunter fallen Andockbereiche, Teile funktionaler Räume usw. Diese Teilmodelle enthalten Instanzen bekannter Objektklassen und optional eine polygonale Beschreibung unbenannter bzw. unbekannter Hintergrundelemente, wie z. B. Wände oder sonstige Hindernisse. Jede Insel hat ihr eigenes Bezugskoordinatensystem.
c) Topologiegraph: Knoten markieren missionsrelevante Punkte innerhalb der Inseln, z. B. Abzweigungen in Gängen und Andockpunkte vor Räumen. Die Kanten stellen die Verbindungen dar, die vom Roboter befahren werden

können. Neben Knoten innerhalb derselben Insel können Kanten auch Knoten in unabhängigen Inseln verbinden und fügen so die einzelnen Teilmodelle zu einem Gesamtmodell der Umgebung zusammen. Da in diesem Fall kein kartesischer Bezug zwischen den Knoten vorhanden ist, müssen solche Kanten mit Fahrverhalten attribuiert werden, um ihre Traversierung zu ermöglichen.

Während der Inhalt von Modellebene *a* als a priori gegeben vorausgesetzt wird, sollen *b* und *c* autonom erkundet werden. Abb. 1 stellt eine einfache Gangszene und das zugehörige Umgebungsmodell dar.

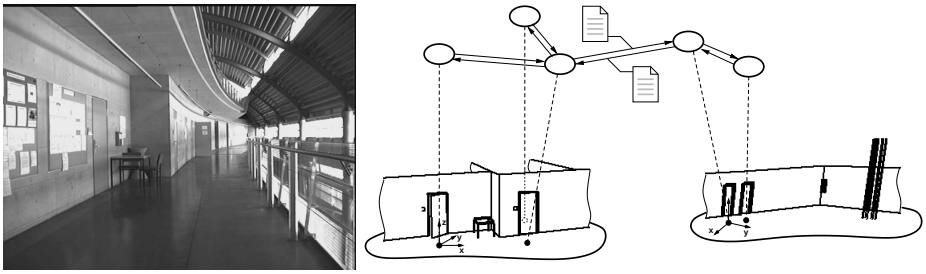

Abb. 1. Exemplarische Einsatzumgebung (links) und Topologiegraph mit Modellinseln (rechts).

3 Die Systemarchitektur OSCAR

Komplexe Aufgabenstellungen wie die Exploration von Innenräumen mit unterschiedlichen Sensoren lassen sich im allgemeinen aufgrund der parallel anfallenden Sensordaten und der damit verbundenden Notwendigkeit von Nebenläufigkeit nicht durch wenige monolithische Verarbeitungsblöcke bewerkstelligen [1, 18]. Vielmehr ist hierbei eine Abbildung der logischen Verarbeitungsschritte in skalierbare Module sinnvoll. Die einfache Austausch- und Wiederverwendbarkeit in verschiedenen Phasen einer Mission macht hierbei die Aufteilung in möglichst schlanke, jedoch in sich abgeschlossene Module notwendig, deren Anzahl entsprechend größer ist. Eine Zielarchitektur muß also die flexible Koordination dieser Module beherrschen, wobei zudem die Möglichkeit zur statischen Erweiterbarkeit sowie zur dynamischen Umkonfiguration der Modulstruktur zur Laufzeit berücksichtigt werden muß. Wünschenswert ist ebenfalls eine Kapselung der Roboterhardware, was eine einfache Portierung auf andere Systeme ermöglicht.

Die für das Projekt EvI entworfene hierarchische Systemarchitektur OSCAR (Operating System for the Control of Autonomous Robots) verwaltet eine Menge von standardisierten Verarbeitungsmodulen. Diese lassen sich gemäß ihrer Teilaufgabe im Gesamtsystem in eine der folgenden drei Schichten einordnen (siehe Abb. 2).

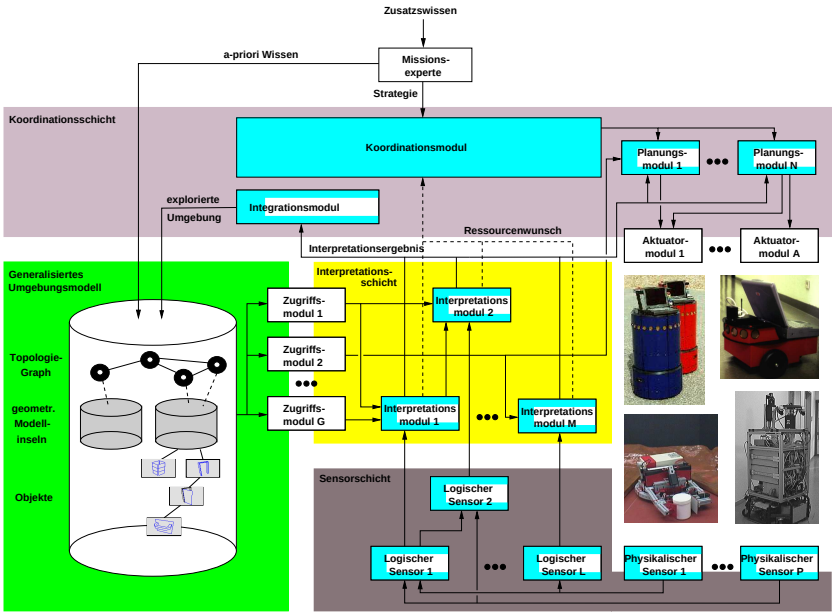

Abb. 2. Hierarchischer Aufbau der OSCAR-Systemarchitektur

- In der *Sensorschicht* befinden sich physikalische und logische Sensoren. Die physikalischen Sensoren haben die Aufgabe, aufgenommene Sensordaten in standardisierte OSCAR-Datentypen (siehe Abschnitt 3.3) zu konvertieren. Der Austausch dieser Komponenten ermöglicht den Einsatz auf verschiedenen Plattformen. Logische Sensoren können somit plattformunabhängig aus anfallenden Sensorrohdaten Merkmale extrahieren und diese weiter abstrahieren.
- Die *Interpretationsschicht* beinhaltet sog. Interpretationsmodule, die die von den logischen Sensoren bereitgestellten Merkmale auf verschiedenen Abstraktionsebenen interpretieren. Im Gegensatz zu den logischen Sensoren wird hierfür Modellwissen benötigt, das durch Zugriffsmodule vom Umgebungsmodell bereitgestellt wird. Als Ergebnis entstehen hierbei Objekt- und Ortshypothesen.
- Die *Koordinationsschicht* beinhaltet verschiedene Integrationsmodule, die die Hypothesen auf Konsistenz prüfen und ggf. einen Eintrag ins Modell vornehmen. Neben dem zentralen Koordinationsmodul (siehe Abschnitt 3.4) beinhaltet diese Schicht auch Planungsmodule, die für die Umsetzung einer gegebenen Strategie zuständig sind. An die Roboterhardware gerichtete Kommandos werden wiederum durch sog. Aktuatormodule gekapselt, um Plattformunabhängigkeit zu gewährleisten.

3.1 Kommunikation

Zentrale Voraussetzung einer modularen Systemarchitektur ist die Möglichkeit des effektiven Datenaustausches der evtl. auch auf unterschiedlichen Rechnern verteilt arbeitenden Module. In den vergangenen Jahren haben sich hierfür unterschiedliche Technologien wie z. B. Socket-Programmierung, Remote Procedure Call (RPC) oder HTTP/CGI, aber auch proprietäre Implementierungen wie ACE [19] herausgebildet. Die Common Object Request Broker Architecture (CORBA) [15] hebt die Problemstellung verteilten Programmierens auf ein höheres Abstraktionsniveau, indem sie einerseits objektorientierte Designparadigmen wie Kapselung, Vererbung und Polymorphismus einsetzt und andererseits plattform- und sprachenunabhängig ist. CORBA ist in dieser Eigenschaft ein als Spezifikation inzwischen etablierter Industriestandard der OMG.

Im Rahmen von OSCAR wird CORBA als Middleware eingesetzt, wobei die konkrete Implementierung ORBacus [16] verwendet wird, die eine Abbildung u. a. auf C++ ermöglicht. Als Spezifikation für Datentypen und Schnittstellen dient OMG IDL, mit der sämtliche in OSCAR übertragenen Datentypen und Komponenten-Schnittstellen definiert sind.

3.2 Das Verarbeitungsmodul

Ein Verarbeitungsmodul (siehe Abb. 3) wird durch zwei Prozesse realisiert: Der Rahmen ist für jedes Modul identisch und stellt einen reaktiven Server dar, der standardisierte Konfigurationsaufrufe und den Datenaustausch der Module untereinander abwickelt. Der Verarbeitungskern hingegen besteht im wesentlichen aus einer mit der Applikation überladbaren C++-Klasse. Kernstück dieser Klasse ist die Methode *step*, die durch einen *get*-Aufruf zunächst Daten anfordert, diese verarbeitet und dann mittels *supply*-Aufruf weitergibt. Die Ablaufsteuerung des Kerns ist durch einen Zustandsautomaten realisiert, der die Methode *step* entweder kontinuierlich (kontinuierlicher Modus) oder durch ein externes Ereignis getriggert (Einzelschrittmodus) aufruft. Daneben existiert noch ein dritter Zustand, der das Modul vorübergehend deaktiviert. Der Rahmen nimmt die *get*- und *supply*-Aufrufe des Kerns auf und leitet sie seinerseits

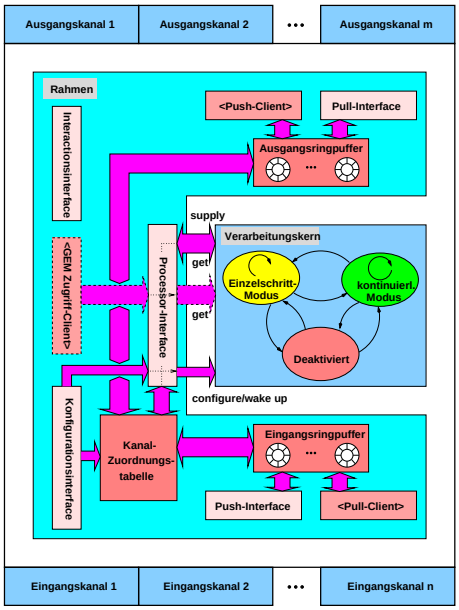

Abb. 3. Verarbeitungsmodul

an andere Verarbeitungsmodule weiter. Grundsätzlich werden alle Eingangs- und Ausgangsdaten in konfigurierbaren Ringpuffern zwischengespeichert. Dies ermöglicht eine zeitlich begrenzte Bewahrung der Historie, womit auch ein effizienterer Datenzugriff möglich ist. Die Eingangsringpuffer haben dabei die Funktion eines lokalen Caches. Dateneingangs- und -ausgangsschnittstelle können aus mehreren Kanälen bestehen, wobei der Datentyp eines Kanals festgelegt ist.

3.3 Datentypen und Transportmechanismus

Der Datenfluß zwischen den Verarbeitungsmodulen basiert auf dem generischen Datentyp *Any* der OMG IDL, der mit beliebigen IDL-Datentypen, die sowohl Sequenzen als auch Basis-Typen sein können, überladbar ist. Eine Dateneinheit besteht außerdem aus einer Sequenz von Zeitstempeln, die von den physikalischen Sensoren vorbelegt werden. Bei Modulen, die Daten fusionieren, besteht somit die Möglichkeit, die Einträge in der Zeitstempelsequenz zu erweitern. Beim derzeitigen Stand der Implementierung fordert grundsätzlich das hierarchisch höher angesiedelte Modul Daten aktiv mittels *pull*-Aufruf an. Neben der Übertragung einzelner Datentypen ist auch der effektivere Aufruf von ganzen Blöcken aus dem Ausgangsringpuffer möglich. Außerdem können durch den Zeitstempel indizierte Dateneinheiten gezielt angefordert werden.

3.4 Konfiguration und Verbindungsmechanismen

Das Koordinationsmodul bestimmt den Rahmen für die zeitliche Abfolge der Datenverarbeitung. Es beinhaltet einen Zustandsautomaten, dessen Zustände genau je einen Datenflußgraphen repräsentieren. Ein Datenflußgraph besitzt als Knoten die jeweils für den aktuellen Zustand aktiven Module; die Kanten repräsentieren den Datenfluß zwischen den Modulen. Bei einem Zustandswechsel werden die Datenflußverbindungen zum Teil neu konfiguriert. Dies wird durch den Austausch der Objektreferenzen der jeweiligen Schnittstellen der Verarbeitungsmodule bewerkstelligt. Ziel hierbei ist, vorhandene Ressourcen effektiv zu nutzen und nur jene Module in Betrieb zu halten, die auch tatsächlich benötigt werden. Das Koordinationsmodul überwacht aber auch die Funktion des gesamten Systems und startet ggf. selbständig Module neu.

4 Beispielhaftes Experimentalszenario

Ziel des im folgenden beschriebenen Experiments ist der Aufbau eines Umgebungsmodells (*GEM*) für die in Abb. 1 dargestellte Gangszene. Das Modell soll durch eine autonom ausgeführte Explorationsfahrt der Multisensorikplattform *MARVIN* [2] erstellt werden und die Plattform im Rahmen späterer Missionen befähigen, bei Vorgabe einer Raumnummer die entsprechende Tür gezielt und auf kürzestem Wege anzufahren. Eine dergestalte Anforderung ergibt sich typischerweise an einen Serviceroboter, der als Bürobote agieren soll.

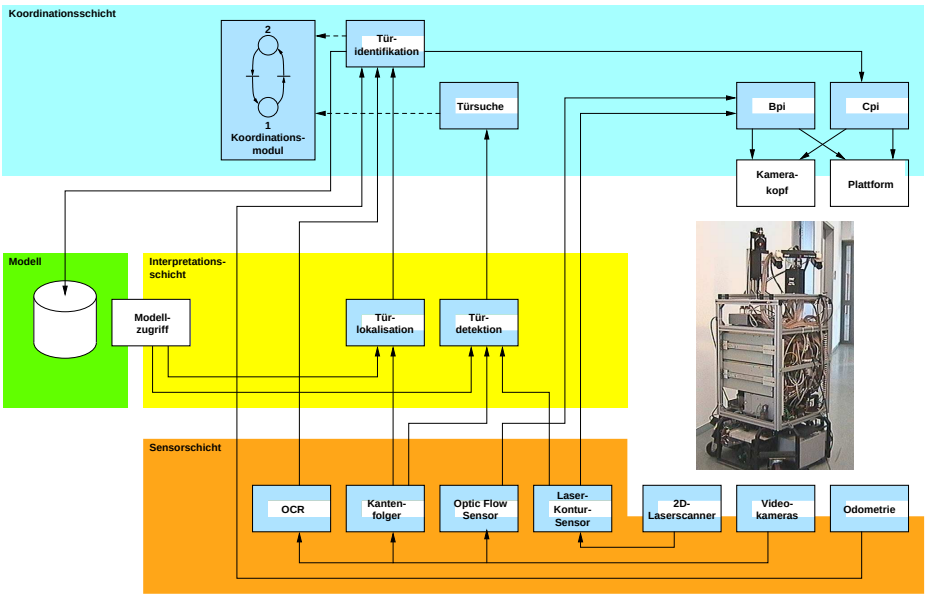

Abb. 4. Gesamtüberblick der im Einsatz befindlichen Module

Diese Aufgabenstellung impliziert, daß die Positionen der Türen entlang des Ganges sowie die Türschilder die für das Modell relevante Information darstellen. Gemäß dem in Kapitel 2 beschriebenen Aufbau von *GEM* wird daher eine geometrische Beschreibung der Objektklasse *Tür* als a priori bekannt vorausgesetzt, da dieser im Sinne der Aufgabenstellung unmittelbare Missionsrelevanz zukommt. Ansonsten enthält das Modell zu Beginn der Explorationsfahrt keinerlei Information über die Einsatzumgebung, so daß *MARVIN* beim Auffinden, Lokalisieren und Identifizieren der Türen sowie beim Aufbau des Modells vollständig auf eigene Sensorik, Aktorik und Informationsverarbeitung angewiesen ist.

4.1 Beteiligte Module

Im folgenden werden die zur Bewältigung der Aufgabenstellung eingesetzten Module vorgestellt.

Sensorik und Aktorik

Die *MARVIN* zur Verfügung stehenden Sensorik- und Aktorikkomponenten sind als OSCAR-Module in Form von physikalischen Sensoren und Aktuatormodulen gekapselt. Auf der Sensorikseite sind dies zwei Standard-*Videokameras* in Stereoanordnung, ein entfernungsgebender, parallel zum Fußboden abtastender *2D-Laserscanner* sowie *Odometrie*. Die Aktorik besteht aus einer *Plattform* mit Panzerkinematik sowie einem Schwenk-Neige-Kopf für die Kameraanordnung (*Kamerakopf*).

Logische Sensoren

Schneller Kantenfolger. Zur Merkmalsextraktion aus Grauwertbildern werden aus Helligkeitssprüngen schritthaltend Konturen und durch anschließende Geradenapproximation Kanten gewonnen [13]. Die hierfür benötigte Rechenzeit im vorliegenden Szenario beträgt bei max. 200 Kanten ca. 0.15 s.

Optic Flow Sensor. Die videogestützte Detektion von Hindernissen in der Fahrtrasse des Roboters basiert auf der Rekonstruktion von Entfernungen aus dem optischen Fluß. Die Berechnung des Flußvektor-Feldes in Realzeit wird durch die Verwendung eines MPEG-Korrelationsprozessors ermöglicht [22]. Je nach Strukturierung der Szene können bis zu 525 3D-Punkte pro Videotakt bestimmt werden. Auch eine Aussage, ob sich bewegte Objekte wie z.B. Personen in der Szene befinden, kann mit Hilfe des optischen Flusses während der Fahrt getroffen werden [21].

Türschilderkennung. Zur Identifikation von Raumnummern auf Türschildern wird ein auf der Bildverarbeitungsbibliothek HALCON [14] basierendes Modul eingesetzt, das nach der Segmentierung des Türschildes aus einem Vollbild mit Hilfe der aus der Optical Character Recognition (OCR) bekannten Technik des Maskenvergleichs robust Raumnummern erkennen kann.

Laser-Kontur-Sensor. Aufgabe dieses Moduls ist es, einen aus einer Vielzahl einzelner Meßpunkte bestehenden Rundum-Scan des 2D-Laserscanners durch eine geordnete Liste möglichst langer Strecken zu approximieren. Die resultierende Beschreibung der Umgebung als eine Sequenz von geradlinigen Konturen, vergleichbar einem Grundriß, kann unmittelbar zur Lokalisation und Navigation herangezogen werden [8].

Interpretationsmodule

Türdetektion. Die aus den Laserkonturen gewonnene und auf den Boden projizierte Wandkante wird im Kamerabild prädiziert, um einen Suchraum für die Wand-Boden-Kante vorzugeben. Durch die Berücksichtigung des Kameramodells lassen sich somit vom *Kantenfolger* gelieferte Merkmale bereits als Kandidaten für senkrecht verlaufende Kanten des Türrahmens klassifizieren. Findet dieser Algorithmus den Türabmessungen entsprechende senkrechte Kantenpaare, so meldet das Modul die Hypothese *„Tür detektiert“*.

Türlokalisation. Mit Hilfe des Objekterkennungssystems *MORAL* [12] wird in einem aus horizontaler Blickrichtung aufgenommenen Bild eine Tür identifiziert und ihre Position und Lage relativ zur Kamera bestimmt [11]. Hierzu werden vom *Kantenfolger* die aktuellen Bildkanten angefordert. Als Ergebnis eines Verarbeitungsschritts werden der Klassenname und die relative Position und Lage der identifizierten Tür zur Verfügung gestellt.

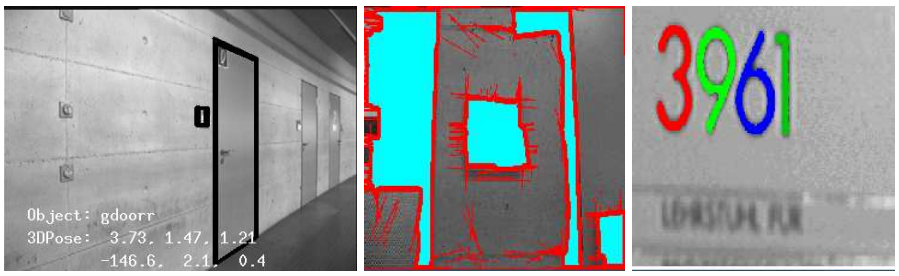

Abb. 5. Lokalisierte Tür (links), segmentiertes Türschild (mitte) und segmentierte Einzelziffern der Raumnummer (rechts).

Planungsmodule

Die für eine Explorationsfahrt notwendigen Kommandos zur Ansteuerung der mobilen Plattform und des Kamerakopfs werden von zwei Planungsmodulen zur Verfügung gestellt, die in unterschiedlichen Phasen der Exploration zum Einsatz kommen (siehe Abschnitt 4.2).

Das Modul *Bpi* (*Behaviour-based pilot*) führt zunächst einen Kameraschwenk nach unten aus, um dem *Optic Flow Sensor* eine Überwachung der Fahrtrasse zu ermöglichen. Ausgestattet mit einer gerasterten, lokalen Hinderniskarte, die mit Hilfe aktueller Sensordaten aufgebaut wird, generiert es dann mit einem Verfahren ähnlich des *Dynamic Window Approach* [9] kreisförmige Fahrwege für die Plattform [3]. Der Radius dieser Trajektorien wird dabei fortwährend so angepaßt, daß ein in der Karte vorgegebener Zielpunkt unter Vermeidung der vermerkten Hindernisse angesteuert wird.

Wird eine Tür detektiert, errechnet das Modul *Cpi* (*Cartesian pilot*) die lokalen Fahrmanöver und Kopfbewegungen, um die Kameras möglichst günstig auf Tür und Türschild auszurichten.

4.2 Exemplarischer Ablauf

Das Zusammenspiel der genannten Module sowie der Aufbau von *GEM* werden von dem in Abschnitt 3.4 beschriebenen Koordinationsmodul kommandiert. Der sich daraus ergebende Ablauf einer beispielhaften Explorationsfahrt wird im folgenden erläutert.

Zunächst befindet sich der dem Koordinationsmodul zugrundeliegende Zustandsautomat im Zustand 1. Das Modul *Türsuche* fragt in die Ebene der Interpretationsmodule beim Modul *Türdetektion* an, ob eine Tür entdeckt wurde. Um diese Entscheidung zu treffen, benötigt die *Türdetektion* einerseits Daten aus der Ebene der Sensoren, in diesem Fall von den logischen Sensoren *Kantenfolger* und *Laser-Kontur-Sensor*; andererseits wird eine Anfrage an *GEM* gerichtet, um das Modell der Objektklasse *Tür* abzurufen. Der *Kantenfolger* seinerseits fordert Grauwertbilder beim physikalischen Sensor *Videokamera* an; der

Laser-Kontur-Sensor triggert im *2D-Laserscanner* die Aufnahme von Rundum-Scans. Vor dem Hintergrund der Exploration ist jedoch die bloße Detektion der Tür nicht ausreichend, sondern muß um eine Lokomotion der Plattform ergänzt werden. Diese wird vom Modul *Bpi* geleistet, das in die Ebene der Sensoren beim *Optic Flow Sensor* und beim *Laser-Kontur-Sensor* anfragt, um seine lokale Hinderniskarte aufzubauen. Durch eine geeignete Wahl des Zielpunktes in der Hinderniskarte ist *Bpi* in der Lage, *MARVIN* autonom eine Wand ansteuern zu lassen und ihn entlang dieser Wand zu führen, wobei Hindernisse wie z. B. Mauervorsprünge oder Personen umfahren werden.

Alle aufgeführten Module werden vom Koordinationsmodul im Zustand 1 aktiviert und in der beschriebenen Weise verschaltet. Die Module *Türsuche*, *Bpi* und *Optic Flow Sensor* werden dabei im *kontinuierlichen Modus* betrieben, alle anderen Module laufen im *Einzelschritt-Modus*, benötigen also nur dann Rechenzeit, wenn sie angefragt werden. Diese Betriebsart wirkt sich insbesondere dann entlastend aus, wenn innerhalb der Module rechenintensive Operationen durchgeführt werden, wie dies häufig bei den logischen und physikalischen Sensoren gegeben ist.

Während dieser Phase der Türsuche in unbekanntem Gebiet wird im Topologiegraph eine neue Kante generiert, wobei es zu diesem Zeitpunkt noch offen ist, ob die Kante innerhalb der aktuellen Modellinsel verläuft oder eine Verbindung zu einer neuen, unabhängigen Insel darstellt. Daher wird die neue Kante attribuiert mit einem Fahrverhalten, in diesem Fall der von *Bpi* ausgewählten, angesteuerten und verfolgten Wand.

Wird eine Tür detektiert, geht das Koordinationsmodul in den Zustand 2, *MARVIN* wird angehalten, und die nicht mehr benötigten Module werden deaktiviert. Das Koordinationsmodul startet dann die für die neue Aufgabe der Türidentifikation zusätzlich noch erforderlichen Module und verschaltet diese entsprechend.

Im Zustand 2 wird *MARVIN* zunächst mit Hilfe des Moduls *Cpi* vor der Tür in Position gefahren und der Kamerakopf in eine horizontale Lage geschwenkt. Im folgenden kontaktiert das Modul *Türidentifikation* das Modul *Türlokalisation*, welches zum einen das Türmodell aus *GEM*, zum anderen aus der Ebene der Sensoren Daten anfordert. Im Anschluß wird zur Erkennung der Raumnummer das Sensormodul *OCR* angefragt, wobei vorher mittels des *Cpi* die Kameras wieder entsprechend ausgerichtet werden. Auf Basis der von der *Türlokalisation* gelieferten Daten und unter Einbeziehung von Odometrieinformation erfolgt nun die Erweiterung des Modells.

Der Topologiegraph wird um einen mit der Raumnummer und der Türposition attribuierten Knoten ergänzt. Ist die Entfernung der Plattform zum Ursprung des aktuellen Koordinatensystems klein genug, daß die Odometriedaten als ausreichend genau angenommen werden können, wird keine neue Modellinsel generiert, d. h. der neue Knoten verweist auf die aktuelle Modellinsel und die Position der gefundenen Tür bezieht sich auf das bisherige Koordinatensystem. Bei zu großer Entfernung vom Ursprung und demzufolge unzuverlässiger Odometrieinformation wird eine neue Modellinsel instanziiert, auf die der neue

Knoten verweist und die die gefundene Tür im Ursprung ihres Koordinatensystems trägt.

Nach Abschluß von Türidentifikation und Modellerweiterung geht das Koordinationsmodul wieder zurück in den Zustand 1 und die Türsuche wird fortgesetzt.

Können auf den von *Bpi* ausgewählten Fahrwegen keine Türen mehr gefunden werden, kehrt *MARVIN* unter Nutzung der während der Explorationsfahrt gewonnenen Umgebungsinformation zu seinem Ausgangspunkt zurück.

5 Ausblick

Der derzeitige Stand der Implementierung von OSCAR erlaubt die Realisierung einfacher Explorationsszenarien. Künftige Aufgaben werden sich auf die zeitliche Optimierung beziehen. Der Einsatz von Threads soll dabei einen effektiveren Datentransport ermöglichen, wobei auch das automatische Liefern von Daten im Sinne eines *push*-Mechanismus [18] angedacht ist. Darüberhinaus sollen in OSCAR Verfahren wie Load Balancing und Scheduling zur optimalen Nutzung vorhandener Ressourcen integriert werden. Ein wichtiger Teilaspekt der Autonomie ist in dieser Hinsicht die selbständige Konfiguration des Systems anhand einer gegebenen Aufgabenstellung.

Ferner soll der Aufbau des Umgebungsmodells sukzessive um weitere Merkmale wie Farbe und qualitative Relationen erweitert werden. Darüberhinaus ist geplant, bestehende Verfahren der 3D-Rekonstruktion [4, 7] zu integrieren. Die Systemarchitektur OSCAR ist für diese Erweiterungen durch die Möglichkeit der flexiblen Konfiguration bereits gerüstet.

Literatur

1. J. S. Albus and A. M. Meystel. A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems. *International Journal of Intelligent Control and Systems*, 1(1):15–30, 1996.
2. S. Blum, D. Burschka, C. Eberst, T. Einsele, A. Hauck, N. O. Stöffler, and G. Färber. Autonome Exploration von Innenräumen mit der Multisensorik-Plattform MARVIN. In *Autonome Mobile Systeme*, Informatik aktuell, pages 138–147. Springer-Verlag, 1998.
3. T. Burkert. Lokale Navigation mit einem Optic Flow Sensor. Master's thesis, TU München, Fakultät für Elektro- und Informationstechnik, Dec. 1998.
4. D. Burschka. *Videobasierte Umgebungsexploration am Beispiel eines binokularen Stereo-Kamerasystems.* PhD thesis, TU München, Fakultät für Elektro- und Informationstechnik, 1998.
5. W. Daxwanger, E. Ettelt, C. Fischer, F. Freyberger, U. Hanebeck, and G. Schmidt. ROMAN: Ein mobiler Serviceroboter als persönlicher Assistent in belebten Innenräumen. In G. Schmidt and F. Freyberger, editors, *Autonome Mobile Systeme*, Informatik aktuell. Springer-Verlag, 1996.
6. G. L. Dudek. Environment Representation Using Multiple Abstraction Levels. *Proc. IEEE*, 84(11):1684–1705, Nov. 1996.

7. C. Eberst. *Incorporation of Recognition Strategies in Sensory Exploration.* PhD thesis, TU München. submitted.
8. T. Einsele. Real-Time Self-Localization in Unknown Indoor Environments using a Panorama Laser Range Finder. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97)*, pages 697–703, Grenoble, France, Sept. 1997.
9. D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Trans. on Robotics and Automation*, 4(1), Mar. 1997.
10. A. Hauck and N. O. Stöffler. A Hierarchical World Model with Sensor- and Task-Specific Features. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'96)*, pages 1614–1621, 1996.
11. S. Lanser. *Modellbasierte Lokalisation gestützt auf monokulare Videobilder.* PhD thesis, TU München, Fakultät für Informatik, 1997.
12. S. Lanser, C. Zierl, O. Munkelt, and B. Radig. MORAL – A Vision-based Object Recognition System for Autonomous Mobile Systems. In *Computer Analysis of Images and Patterns (CAIP)*, number 1296 in Lecture Notes in Computer Science, pages 33–41. Springer-Verlag, 1997.
13. G. Magin and C. Robl. A Single Processor Real-Time Edge-Line Extraction System for Feature Tracking. In *IAPR Workshop on Machine Vision Applications (IAPR MVA'96)*, 1996.
14. MVTec Software GmbH. *HALCON – The Software Solution for Machine Vision Applications.* http://www.mvtec.com/halcon/.
15. Object Management Group (OMG). CORBA/IIOP 2.3 specification. http://www.omg.org/corba, 1998.
16. Object Oriented Concepts. *Orbacus.* http://www.ooc.com/ob/.
17. J. Ponce, A. Zisserman, and M. Hebert, editors. *International Workshop on Object Representation in Computer Vision II, Cambridge, U.K.*, volume 1144 of *Lecture Notes in Computer Science*. Springer-Verlag, Apr. 1996.
18. C. Schlegel and R. Wörz. Der Softwarerahmen *SmartSoft* zur Implementierung sensomotorischer Systeme. In *Autonome Mobile Systeme*, Informatik aktuell, pages 208–217. Springer-Verlag, 1998.
19. D. C. Schmidt. An architectural overview of the ACE framework. *USENIX login magazine, Tools special issue*, 1998.
20. N. O. Stöffler, A. Hauck, and G. Färber. Ein geometrisch-symbolisches Umgebungsmodell zur Unterstützung verschiedener Perzeptionsaufgaben autonomer, mobiler Systeme. In G. Schmidt and F. Freyberger, editors, *Autonome Mobile Systeme*, Informatik aktuell, pages 108–117. Springer-Verlag, 1996.
21. N. O. Stöffler and Z. Schnepf. An MPEG-Processor-based Robot Vision System for Real-Time Detection of Moving Objects by a Moving Observer. In *Proc. 14th Int. Conf. on Pattern Recognition*, pages 477–481, Brisbane, Australia, Aug. 1998.
22. N. O. Stöffler and G. Färber. An Image Processing Board with an MPEG Processor and Additional Confidence Calculation for Fast and Robust Optic Flow Generation in Real Environments. In *Proc. Int. Conf. on Advanced Robotics (ICAR'97)*, pages 845–850, Monterey, California, USA, July 1997.
23. S. Thrun and A. Brücken. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, Portland, Oregon, Aug. 1996.