Realzeitfähige Multiagentenarchitektur für autonome Fahrzeuge

Steffen Görzig[1], Axel Gern[2] und Paul Levi[3]

[1] DaimlerChrysler AG
Forschung und Technologie
Software Architekturen FT3/SA
steffen.goerzig@daimlerchrysler.com
[2] DaimlerChrysler AG
Forschung und Technologie
Bildverstehen FT3/AB
axel.gern@daimlerchrysler.com
[3] Universität Stuttgart
Institut für Parallele und
Verteilte Höchstleistungsrechner
Abteilung Bildverstehen
paul.levi@informatik.uni-stuttgart.de

Zusammenfassung Das Aufgabengebiet des Autonomen Fahrens bringt neben Herausforderungen bei der Bildverarbeitung und den Hardware-Komponenten auch zahlreiche Ansprüche an die Software-Architektur mit sich. Dies gilt umso mehr, je komplexer und umfangreicher diese Systeme aufgebaut sind.
In diesem Beitrag wird eine Multiagentenarchitektur beschrieben, welche in der Lage ist, Software-Module dynamisch zu konfigurieren und untereinander zu vernetzen. Es wird gezeigt, daß die Architektur in der Lage ist, den Ansprüchen komplexer autonomer Fahrsysteme gerecht zu werden. Dazu gehören Punkte wie Echtzeitfähigkeit, Skalierbarkeit, Parallelverarbeitung, Konfigurierbarkeit und Ressourcenverteilung. Als erste Anwendung wurde ein autonomer Stop&Go Betrieb in der Innenstadt verwirklicht.

1 Einleitung

In der Vergangenheit wurden große Anstrengungen unternommen, um Autonomes Fahren auf Autobahnen zu ermöglichen. So demonstrierte Dickmanns bereits 1986 optische Spurhaltung auf Autobahnen [1]. Anläßlich der Abschlußveranstaltung des Europäischen PROMETHEUS Projektes demonstrierte der DaimlerChrysler Versuchsträger VITA II Autonomes Fahren einschließlich der Planung und Durchführung von Überholmanövern [2].

Diese vielversprechenden Ergebnisse haben uns ermutigt, ein weitaus komplexeres Problem in Angriff zu nehmen: Autonomes Fahren in der Innenstadt. In der Innenstadt sind zwar die Geschwindigkeiten gegenüber der Autobahn geringer, die Umgebung ist hingegen vielschichtiger:

- chaotische Spurführung
- unzuverlässige, verschiedenartige Spurmarkierungen und fehlende Fahrspurmodelle
- bunte Umgebung (Werbetafel oder Verkehrsschild?)
- zahlreiche, verschiedenartige Verkehrsteilnehmer
- unterschiedliche Bewegungsrichtungen der Verkehrsteilnehmer
- umfangreiche Infrastruktur (Verkehrszeichen, Ampeln, Bodenpfeile usw.)

Autonomes Fahren in der Innenstadt ist aber nicht nur aus algorithmischer Sicht eine Herausforderung, sondern auch aus Sicht der Systemarchitektur. Die wachsende Komplexität der Autonomen Systeme bedingt Architekturen, welche verschiedenartige Anforderungen erfüllen müssen:

- Integration und Kooperation verschiedener Algorithmen.
- Unterschiedliche Abstraktionsstufen von Aktion, Wahrnehmung und Steuerung.
- Sensorfusion.
- Statische und dynamische Systemrekonfiguration.
- Effiziente Nutzung von Ressourcen.
- Echtzeitverarbeitung. In diesem Zusammenhang verstehen wir unter Echtzeit, "daß der Verarbeitungssyklus so wenig Zeit beansprucht, daß das Gesamtsystem eine Reaktionszeit aufweist, die unter der eines menschlichen Fahrers liegt." [3]. Als Grundlage dient dabei die in der Bildverarbeitung übliche Bildrate von 25 Bildern pro Sekunde.
- Verteiltes Rechnen.
- Skalierbarkeit in Bezug auf Rechner und Systemkomponenten.
- Testumgebungen für Systemkomponenten.

Um diesen Anforderungen gerecht zu werden wurden verschiedene Ansätze vorgeschlagen, z. B. in [3] und [4]. In diesem Beitrag beschreiben wir ein Multiagentensystem, welches für Autonomes Fahren bzw. Fahrerassistenzsysteme auf der Autobahn als auch in der Innenstadt eingesetzt werden kann. Als erste Applikation wurde ein autonomer Stop&Go Betrieb in der Innenstadt verwirklicht.

Das "Agent NeTwork System" (ANTS) steuert dabei diverse Bildverarbeitungs-, Regelungs- und Fahrerschnittstellenprozesse. Dies ist weltweit der erste Ansatz, ein autonomes Fahrzeug für die Innenstadt zu entwickeln und erfolgreich einzusetzen.

2 Multiagentenarchitekturen

Multiagentenarchitekturen haben sich mittlerweile zu einem eigenständigen und wandlungsfähigen Konzept entwickelt. Unter dem Terminus "Agent" werden jedoch zur Zeit verschiedenartige Konzepte zusammengefaßt. Aus diesem Grund gibt es keine eindeutige Begriffsdefinition. Als Arbeitsdefinition für Multiagentensysteme (MAS) dient in diesem Beitrag die Definition nach O´Hare/Jennings:

"a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities" [5].

Die kleinste Einheit eines Multiagentensystems ist der Agent. Ein Agent kann als eine Recheneinheit beschrieben werden, die bestimmte Dienste zur Verfügung stellt. Er besitzt darüber hinaus einen gewissen Grad an Weltwissen, Entscheidungsmöglichkeiten, Kooperation und Kommunikation.

Mit diesen Arbeitsdefinitionen ist es leicht zu verstehen, wie Multiagentensysteme für Autonomes Fahren und Fahrerassistenzsysteme eingesetzt werden können. Jedes Bildverarbeitungs- oder Fahrzeugsteuerungmodul stellt Funktionalitäten zur Verfügung, die für Anwendungen in diesem Bereich von Nutzen sind. Im folgenden werden die Dienste eines Agenten als Funktionseinheit bezeichnet. Die Entscheidungsmöglichkeiten und die Kooperation werden von sogenannten Administratoren geregelt. Auf diese Weise können Module untereinander verbunden werden, um Anwendungen wie den autonomen Stop&Go Betrieb in der Innenstadt zu ermöglichen. Die Kommunikation geschieht über eine verteilte Datenbank, in der das Weltwissen abgelegt ist.

3 ANTS Komponenten

ANTS besteht im wesentlichen aus den Komponenten (siehe Abbildung 1):

- Datenbank
- Module/Funktionseinheiten
- Administratoren

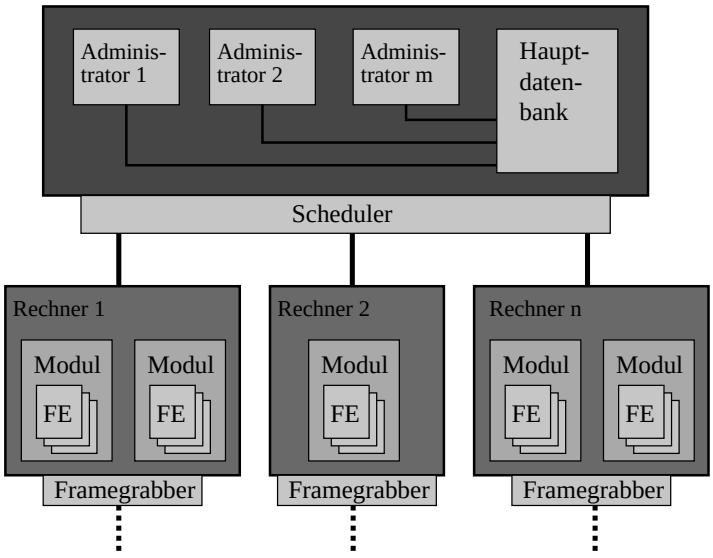

Abbildung1. ANTS Komponenten.

Funktionseinheiten (FE) sind die grundlegenden Berechnungseinheiten des Systems. Sie sind in Funktionsmodulen zusammengefaßt. Der Aufruf der Funktionseinheiten wird durch zugeordnete Administratoren übernommen. Sie entscheiden, welche Funktionseinheiten aufgerufen werden und können diese parametrisieren. Die explizite Unterscheidung zwischen ausführender Einheit und Entscheidungseinheit bietet verschiedene Vorteile:

- Die Komponenten können parallel ausgeführt werden; bereits während der Ausführung einer Entscheidung kann der nächste Schritt geplant werden.
- Bestehende Software kann als Funktionale Einheit wiederverwendet werden.
- Die Modifikation einer Komponente bedingt nicht notwendigerweise die Modifikation einer anderen.

Der Systemaufbau wird im folgenden detailiert beschrieben.

3.1 Datenbank

Das Weltwissen der Agenten in ANTS ist in einer verteilten Datenbank abgelegt (siehe Abbildung 2). Die Datentypen sind frei definierbar, es kann sich also sowohl um symbolische als auch um subsymbolische Daten handeln. Jeder Agent besitzt eine lokale Sicht auf die Datenbank und kann transparent auf die Daten zugreifen. Die Verteilung der Daten auf die jeweiligen Prozesse wird von der Datenbank selbst übernommen. Um Inkonsistenzen zu vermeiden, sind die Daten mit Zugriffssperren versehen. Zwei Agenten können nicht gleichzeitig schreibend auf dieselben Daten zugreifen. Die lokalen Datenbanken der Agenten besitzen nur den Umfang, den diese Agenten benötigen. Dies reduziert den Umfang der lokalen Instantiierungen. In der Regel sind dies Ein- und Ausgabewerte der Agenten. Der Austausch von Daten zwischen Agenten erfolgt ebenfalls über die Datenbank. Benötigt etwa ein Agent die Ergebnisse eines anderen, so kann er über seine lokale Datenbank transparent darauf zugreifen.

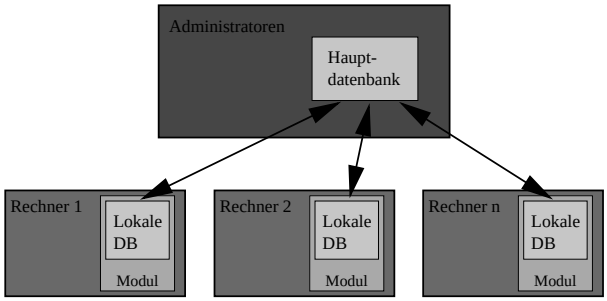

Abbildung2. Die Datenbank.

In der Datenbank sind zudem die Konfigurationsparameter als auch Informationen über die Agenten selbst abgelegt. Die Administratoren können so zur Laufzeit die Agenten parametrisieren und die Zustände der Agenten (aktiv, wartend, ausgefallen, Zeitüberschreitungen) berücksichtigen. Der Aufbau der verteilten Datenbank erfolgt mit Hilfe der Skript-Sprache OCS (Object Creation System), welche für ANTS entwickelt wurde. In Skripten wird festgelegt, welche Daten in den jeweiligen Instanzen zu sehen sind, ebenso können die Daten initialisiert werden.

Die Verteilung der Daten wird von der Datenbank auf eine Nachrichtenbibliothek abgebildet. Die Bibliothek ist in der Lage, Objekte zu versenden und empfangen und basiert auf PVM (Parallel Virtual Machine)[1]. Mittels PVM können heterogene Rechnersysteme transparent miteinander verbunden werden. Dies ermöglicht die Skalierbarkeit und Portierbarkeit des Systems.

3.2 Module und Funktionseinheiten

Funktionseinheiten sind die grundlegenden Berechnungseinheiten des Systems (siehe Abbildung 3). Funktionseinheiten enthalten Algorithmen wie z. B. die Stereoobjekterkennung oder die Spurerkennung, welchen über eine Schnittstelle (IF) mit ANTS verbunden werden. Über die Schnittstelle hat ein Algorithmus Zugriff auf die Datenbank. Die Spurerkennung greift so z. B. auf die Ergebnisse der Stereoobjekterkennung zu, um ihre Spurschätzung zu verbessern.

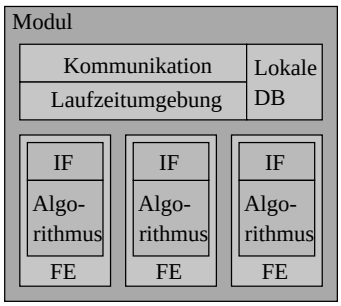

Abbildung3. Der Modulaufbau.

Die Übergabe aller Ein- und Ausgangsdaten, der Konfigurationsparameter sowie der Aufruf von Algorithmen sind in der Schnittstelle gekapselt. Dies ist in mehrerer Hinsicht von Nutzen:

– Existierende Algorithmen können wiederverwendet werden.

[1] Eine überarbeitete Fassung der C++ Nachrichtenbibliothek ist unter dem Namen CPPvm verfügbar [6].

- Der Entwickler eines Algorithmuses muß sich nicht um ANTS Konzepte wie Administratoren oder Nachrichtenbibliotheken kümmern.
- Der Benutzer eines Algorithmus kann sich ebenfalls auf die Schnittstelle beschränken; bei der Verwendung eines Algorithmus wird keine Detailkenntnis über diesen benötigt.
- Diese Trennung erlaubt die Verwendung von Testumgebungen, wie sie weiter später beschrieben werden.

Die Funktionseinheiten werden in Modulen zusammengefaßt. Ein Modul beinhaltet die Laufzeitumgebung der Funktionseinheiten und übernimmt ihre Aufrufe (einmal, n-mal, permanent). Die Anweisungen dazu erhält sie über die Kommunikationsschnittstelle von den Administratoren.

3.3 Administrator

Die Administratoren steuern eine Menge von Modulen. Sie treffen die Entscheidung, welche Funktionseinheiten als nächstes auszuführen sind und übergeben diese Entscheidung an den Scheduler. Der Scheduler leitet die Entscheidungen an die Module weiter und überprüft, ob die Aufträge ausgeführt wurden.

Der Weg bis zur Entscheidungsfindung ist in Abbildung 4 dargestellt. Die Beschreibung der zu einem Administrator gehörenden Module und Funktionseinheiten wird aus der Datenbank gelesen und dem ersten Filter übergeben. Filter sind dazu da, die Menge der auszuführenden Module und Funktionseinheiten einzuschränken, um die spätere Entscheidungsfindung zu beschleunigen. In der Fahrphase "Autobahn" können so z. B. die Funktionseinheiten "Ampelerkennung" und "Bodenpfeilerkennung" bereits von der Ausführung ausgenommen werden. Die Filter erhalten einheitlich eine Menge an Modulen/Funktionseinheiten als Eingabe und geben eine (Unter-) Menge davon weiter. Auf diese Weise kann eine beliebige Anzahl von Filtern hintereinandergeschaltet werden.

Der Entscheider erhält als Eingabe die übrig gebliebene Menge. Er bestimmt nun, welche Funktionseinheiten in welchen Modulen mit welchen Konfigurationsparametern und Daten ausgeführt werden. So existieren beispielsweise verschiedene Spurerkenner für Autobahn- und Innenstadtbetrieb, zwischen denen umgeschaltet wird. Ein weiteres Beispiel ist die Stereoobjekterkennung, welche im Innenstadtszenario auf halbe Bildauflösung parametrisiert wird. Dies erhöht die Verarbeitungsgeschwindigkeit und bietet für den dort interessierenden Nahbereich ausreichende Genauigkeit. Die Entscheidung wird an den Scheduler weitergegeben, welcher für eine konkrete Umsetzung zuständig ist.

Jeder Filter und Entscheider hat Zugriff auf die Datenbank. Sie enthält das Weltwissen und liefert damit grundlegende Auswahl- und Entscheidungskriterien. Die einheitlichen Schnittstellen der Komponenten erlauben die Implementierung verschiedener Verfahren. Der Ablauf für ANTS bleibt dabei der gleiche. Für die Stop&Go Anwendung wurden zwei Entscheidungsverfahren für die Bildverarbeitungsmodule eingesetzt:

- Auswahl nach Priorität

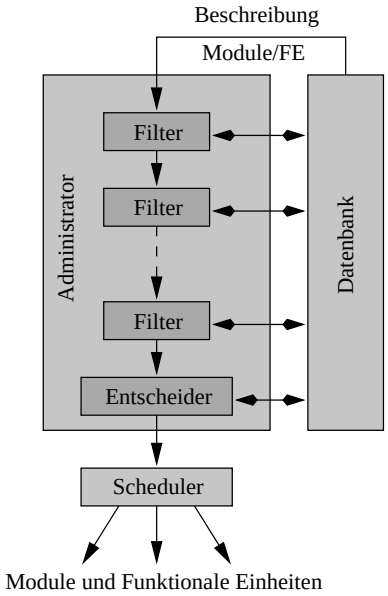

Abbildung4. Aufbau eines Administrators.

– Contract-Net basiertes Verhandlungsprotokoll [7]

ANTS kann beliebig viele Administratoren verwalten, welche jeweils eine Untermenge an Modulen kontrollieren. So gibt es bei der Stop&Go Anwendung beispielsweise Administratoren für die Bildverarbeitung und die Visualisierung. Diese Aufteilung erleichtert den Umgang mit dem Gesamtsystem sowie die Entscheidungsfindung der einzelnen Administratoren. Entscheidungen können so für thematisch geordnete Mengen von Modulen getroffen werden.

Die Administratoren sind für die dynamische Konfiguration der Module und Funktionseinheiten zur Laufzeit zuständig. Sie ermöglichen damit z. B. die dynamische Rekonfiguration einer Anwendung "Automatische Spurhaltung" auf Autobahnen hin zur Anwendung "Stop&Go" in der Innenstadt.

Die statische Konfiguration in der Initialisierungsphase von ANTS wird ebenso wie die Initialisierung der Datenbank durch die Skript-Sprache OCS übernommen. Mittels OCS können Module und Funktionseinheiten auf bestehende Rechner verteilt werden. Wird dem System etwa ein neuer Rechner zur Verfügung gestellt, so kann ANTS über OCS ohne Neuübersetzung daran angepaßt werden.

3.4 Testumgebung

Komplexe parallele Systeme tragen ein hohes Fehlerpotential in sich. Fehler können sowohl durch die einzelnen Komponenten als auch durch das Zusammenspiel

der Komponenten untereinander verursacht werden. Deshalb sollte bereits bei dem Systementwurf auf Testmöglichkeiten geachtet werden.

Die Schnittstellen der Funktionseinheiten (siehe Abbildung 3) bieten die Möglichkeit, Administratoren und Algorithmen getrennt voneinander zu testen. So kann ein Modul per Skript-Sprache auf Standalone-Betrieb umgeschaltet werden. In diesem Modus werden in einer Endlos-Schleife alle Funktionseinheiten des Moduls durchlaufen. Dies ist nicht nur für die Fehlersuche bei Modulen von Nutzen, sondern hat sich auch bei der Integration neuer Funktionseinheiten bewährt. Nach dem Entwurf der Schnittstelle wird eine neue Einheit erst in einer Standalone-Version getestet. Erst wenn sich diese Version als stabil erwiesen hat, wird sie per Skript in das Gesamtsystem integriert.

Es hat sich in der Praxis gezeigt, daß nur wenige Modifikationen nötig sind, um vorhandene Software in ANTS aufzunehmen. Auf diese Weise können verschiedene Versionen vermieden werden, was die Wartung der Software deutlich erleichtert.

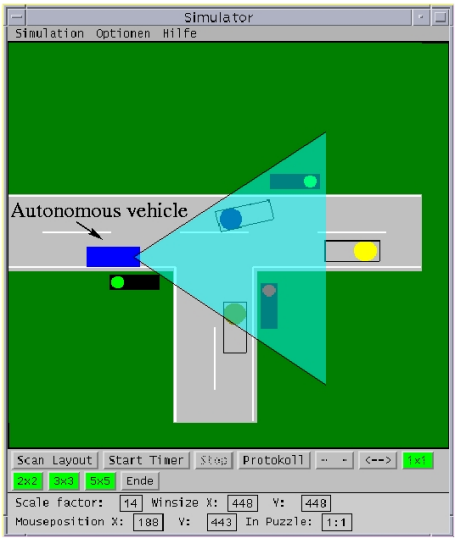

Abbildung5. Innenstadt-Simulator.

Die Schnittstellen der Funktionseinheiten erlauben andererseits den Einsatz "virtueller" Algorithmen. Dies kann verwendet werden, um das Verhalten von Administratoren zu testen. Zu diesem Zweck wurde ein Innenstadtsimulator entwickelt (siehe Abbildung 5). Der Simulator kann typische Innenstadt-Szenen generieren mit Autos, Fußgängern, Fahrradfahrern und Kreuzungssituationen. Ein ausgezeichnetes Fahrzeug mit der Dynamik eines S-Klasse Mercedes kann mit dieser Umgebung mittels virtueller Sensoren und Aktuatoren interagieren. Die

Sensoren können dazu verwendet werden, virtuelle Bildverarbeitung zu betreiben. Dabei werden echte Bildverarbeitungsalgorithmen durch virtuelle ersetzt, welche ihre beobachtbare Leistung simulieren.

Auf diese Weise können reproduzierbare Situationen geschaffen werden, um Entscheidungsstrategien der Administratoren zu evaluieren. Darüberhinaus können Fragen erörtert werden, wie z. B.: Wie verhält sich das System bei geringerer Sichtweite? Kann das System von einem neuen, noch nicht existenten Bildverarbeitungsmodul profitieren? Welche Anwendungen werden durch einen weiteren Sensor – z. B. ein Radar – ermöglicht?

4 Ergebnisse und Ausblick

Als Versuchträger dient derzeit das S-Klasse Fahrzeug UTA (Urban Traffic Assistant). UTA besitzt ein Stereokamerasystem und eine Farbkamera. Das Nachfolgefahrzeug UTA II verfügt darüber hinaus über einen Radarsensor.

In ANTS wurden bisher folgende Komponenten integriert:

- Bildverarbeitungsadministrator
- Visualisierungsadministrator
- Fahrzeugsteuerungsadministrator
- Stereo-Objekterkennung [8]
- Verschiedene Ansätze zur Fußgängererkennung:
  - Time delay neural network (TDNN) auf Bildsequenzen [9]
  - Neuronales Netz auf Einzelbildern [10]
  - Chamfer matching [11]
  - Support vector machine [12]
- Zwei Spurerkenner (Innenstadt [13] und Autobahn [14])
- Bodenpfeilerkennung [15]
- Fußgängerüberwegerkennung [13]
- Ampelerkennung [11]
- Zwei Verkehrszeichenerkenner (Farbe [16] und Schwarz/Weiß [11])
- 2D/3D Visualisierung
- Fahrzeugregelung

Die Abbildung 6 zeigt eine Kameraaufnahme von UTA in einer typischen innerstädtischen Kreuzungssituation. Die Visualisierung stellt die in dieser Situation von der Bildverarbeitung erkannten Objekte dar (Fahrzeug, Fußgänger, Spur, Ampel und Verkehrsschild).

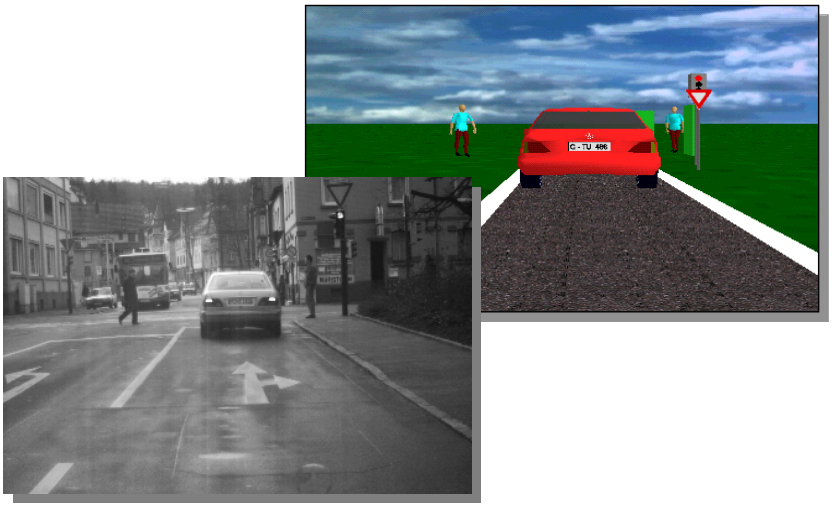

Abbildung6. Kreuzungssituation und die dazugehörige Visualisierung der Bildverarbeitungsergebnisse.

Die erste Anwendung von ANTS im Bereich Autonomes Fahren und Fahrerassistenzsysteme ist der autonome Stop&Go Betrieb in der Innenstadt. Sobald ein vorausfahrendes Fahrzeug erkannt wurde, kann der Fahrer das System aktivieren. Das eigene Fahrzeug folgt von nun an autonom diesem Fahrzeug, d. h. die Regelung von Lenkung, Gas und Bremse wird durch das System übernommen. Im Stop&Go Betrieb wurden bereits mehrere hundert Kilometer autonomer Fahrt zurückgelegt. Die Stop&Go Anwendung zeigt, daß ANTS obige Definition für Echtzeitverarbeitung erfüllt. Die Ansteuerung der Fahrzeugaktuatorik erfolgt im 40ms Takt.

ANTS wird kontinuierlich um neue Funktionseinheiten erweitert. Neben dem autonomen Stop&Go Betrieb sind noch weitere Anwendungen vorgesehen:

Geschwindigkeitsassistent Dem Fahrer wird die jeweils aktuell gültige Höchstgeschwindigkeit angezeigt.

Einschlafwarner Wird auf der Autobahn die Spur verlassen, ohne daß ein Spurwechsel angezeigt wurde, so wird der Fahrer akustisch gewarnt.

Optische Spurhaltung Automatische Querregelung auf Autobahnen.

Abstandsregeltempomat Kann die per Tempomat vorgegebene Wunschgeschwindigkeit aufgrund eines langsameren, vorausfahrenden Fahrzeuges nicht gehalten werden, so wird das eigene Fahrzeug abgebremst. Ist die Spur frei, so wird wieder auf die Wunschgeschwindigkeit beschleunigt.

Autonomes Fahren auf der Autobahn Eine Kombination aus optischer Spurhaltung und Abstandsregeltempomat.

5 Zusammenfassung

Autonome Fahrsysteme und Fahrerassistenzsysteme werden zunehmend für immer komplexere Anwendungen eingesetzt. Die wachsende Komplexität dieser Systeme erfordert Software-Architekturen, die den bestehenden als auch zukünftigen Anforderungen gerecht werden.

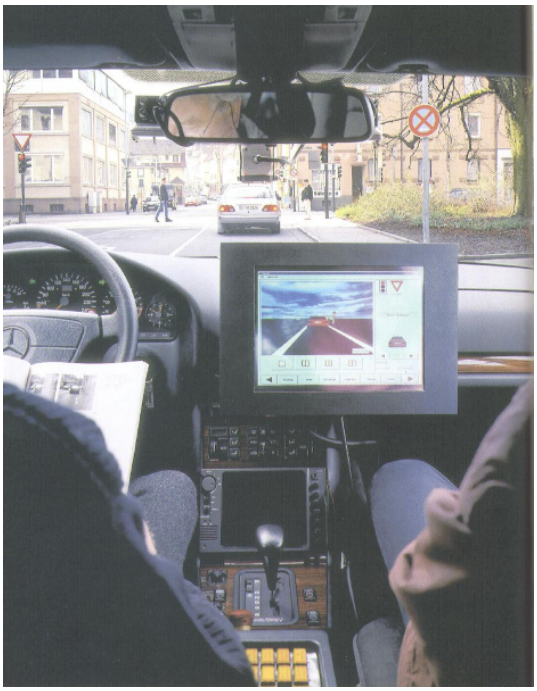

Abbildung7. Kreuzungssituation im Versuchsträger UTA.

Die beschriebene Multiagentenarchitektur bietet die benötigten Komponenten für derartige Anwendungen. ANTS erlaubt die kontinuierliche Erweiterung des Systems durch neue Module und Rechner. Die Struktur der Funktionseinheiten erlauben es, vorhandene Software wiederzuverwenden. Die Architektur vereinfacht ebenfalls die Kooperation und Fusion der Funktionseinheiten untereinander. Diese können dann in einer Art "Baukastensystem" für verschiedene Anwendungen herangezogen werden.

Der autonome Stop&Go Betrieb in der Innenstadt zeigt, daß Multiagentensysteme durchaus für realzeitfähige Anwendungen eingesetzt werden können (siehe Abbildung 7). Durch die Wandlungs- und Erweiterungsfähigkeit solcher Systeme sind sie auch für zukünftige Anforderungen gerüstet.

Literatur

1. E. D. Dickmanns and A. Zapp, "A curvature-based scheme for improving road vehicle guidance by computer vision," in *SPIE Conference on Mobile Robots*, 1986, vol. 727, pp. 161–167.
2. Berthold Ulmer, "VITA II – Active Collision Avoidance in Real Traffic," in *Proceedings of the Intelligent Vehicles '94 Symposium*, Oct. 1994, pp. 1–6.
3. Dirk Reichardt, *Kontinuierliche Verhaltenssteuerung eines autonomen Fahrzeugs in dynamischer Umgebung*, Ph.D. thesis, Universität Kaiserslautern, Jan. 1996, Forschung F1M/IA Daimler–Benz.
4. M. Maurer and E. D. Dickmanns, "A SYSTEM ARCHITECTURE FOR AUTONOMOUS VISUAL ROAD VEHICLE GUIDANCE," in *IEEE Conference on Intelligent Transportation Systems*, Nov. 1997, pp. 578–583.
5. G. O'Hare and N. Jennings, Eds., *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.
6. S. Görzig, *CPPvm: C++ Interface to PVM (Parallel Virtual Machine)*, 1999, http://www.informatik.uni-stuttgart.de/ipvr/bv/cppvm.
7. Reid G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," in *IEEE Transaction on Computers*, 1980.
8. U. Franke and I. Kutzbach, "Fast Stereo based Object Detection for Stop&Go Traffic," in *IEEE Conference on Intelligent Transportation Systems, Tokyo*, Oct. 1996, pp. 339–344.
9. C. Wöhler and J. K. Anlauf, "An Adaptable Time Delay Neural Network Algorithm for Image Sequence Analysis," Nov. 1999.
10. C. Wöhler, U. Kressel, J. Schürmann, and J. K. Anlauf, "Dimensionality Reduction by Local Processing," in *European Symposium on Artificial Neural Networks*, 1999, pp. 237–244.
11. U. Franke, D. Gavrila, S. Görzig, F. Lindner, F. Paetzold, and C. Wöhler, "Autonomous Driving Goes Downtown," *IEEE Intelligent Systems & their applications*, vol. 13, no. 6, pp. 40–48, 1998.
12. C. Papageorgiou, T. Evgeniou, and T. Poggio, "A Trainable Pedestrian Detection System," in *IEEE Conference on Intelligent Vehicles*, Oct. 1998, pp. 241–246.
13. F. Paetzold and U. Franke, "Road Recognition in Urban Environment," in *IEEE Conference on Intelligent Vehicles*, Oct. 1998.
14. Uwe Franke, "Real time 3D-Road Modeling for autonomous vehicle guidance," in *Selected Papers of the 7th Scandinavian Conference on Image Analysis*, P. Johnason and S. Olsen, Eds., pp. 277–284. World Scientific Publishing Company, 1992.
15. U. Franke, S. Görzig, F. Lindner, D. Mehren, and F. Paetzold, "STEPS TOWARDS AN INTELLIGENT VISION SYSTEM FOR DRIVER ASSISTANCE IN URBAN TRAFFIC," in *IEEE Conference on Intelligent Transportation Systems*, Nov. 1997, pp. 601–606.
16. W. Ritter, F. Stein, and R. Janssen, "Traffic Sign Recognition Using Colour Information," in *Math. Comput. Modelling, No. 4-7*, 1995, vol. 22, pp. 149–161.