Qualitätsgesicherte Fahrentscheidungsunterstützung für automatisches Fahren auf Schnellstraßen und Autobahnen

Arne Bartels[1], Christian Berger[2], Holger Krahn[2], Bernhard Rumpe[2]

[1]Volkswagen AG - Konzernforschung
Brieffach 1777
38436 Braunschweig
http://www.volkswagen.de

[2]Technische Universität Braunschweig
Institut für Software Systems Engineering
Mühlenpfordtstraße 23
38106 Braunschweig
http://www.sse-tubs.de

Kurzfassung

Das wahrnehmbare Umfeld um ein auf einer Autobahn oder Schnellstraße fahrendes Fahrzeug ist im Vergleich zum urbanen Umfeld relativ stabil und bietet sich daher für automatisch durchgeführte Fahrten an. Die Volkswagen AG und das Institut für Software Systems Engineering haben gemeinsam eine in verschiedenen Kontexten verwendbare, flexible und qualitätsgesicherte Software-Komponente entwickelt, die in Projekten zum automatischen Fahren die Schnittstelle zwischen Sensordatenfusion und Fahrzeugregelung bildet.

1. Einleitung

Der konsequente Schritt der Weiterentwicklung heutiger Fahrerassistenzsysteme besteht maßgeblich darin, die verschiedenartigen Daten der Fahrzeugsensoren zu fusionieren, um so ein verlässliches Abbild der Umgebung zu erhalten, das intelligente Fahrfunktionen über einfaches Einparken oder Spurhalten hinaus ermöglicht. Wie in [1] beschrieben und insbesondere durch die mediale Begleitung von Wettbewerben wie der DARPA Urban Challenge [2], nimmt die Nachfrage und der Anspruch an moderne Fahrzeuge und leistungsstarke Komfortfunktionen kontinuierlich zu. Zunehmend „intelligentere“ Fahrzeuge sind eine für den Wettbewerb essentielle Differenzierung.

Die Aufgaben der DARPA Urban Challenge zur Demonstration autonomen Fahrens in städtischen Umgebungen waren für heutige Sensorik sowie Algorithmen eine deutliche Herausforderung, da sich die Umgebung des eigenen Fahrzeugs nahezu beliebig komplex und relativ unerwartet verändern kann. Die Anzahl der möglichen Fahrsituationen für ein autonomes Fahrzeug ist somit nahezu beliebig unbeschränkt und ihre Entwicklung instabil.

Auf Schnellstraßen und Autobahnen können jedoch eine Reihe von Annahmen getroffen werden, die den Anspruch des Systemkontexts an ein technisches System reduzieren.

[BBKR09] A. Bartels, C. Berger, H. Krahn, B. Rumpe
Qualitätsgesicherte Fahrentscheidungsunterstützung für automatisches Fahren auf Schnellstraßen und Autobahnen.
In: Proceedings des 10. Braunschweiger Symposiums „Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel“, GZVB, Braunschweig, Februar 2009.
www.se-rwth.de/publications

Beispielsweise entfällt die Erkennung einer Lichtzeichensignalanlage, ebenso können Fußgänger auf und neben der Fahrbahn als unwahrscheinlich betrachtet werden.

Zur Darstellung einer Machbarkeitsstudie für den Themenbereich automatisches Fahren, also fahrzeugbestimmtes Fahren mit dem Fahrer als Sicherheitsinstanz im Gegensatz zu vollkommen selbst überlassenen Fahrentscheidungen, wurde der verglichen zur urbanen Umgebung stabilere Systemkontext einer Autobahn gewählt. Hierfür hat die Volkswagen AG gemeinsam mit dem Institut für Software Systems Engineering der Technischen Universität Braunschweig und weiteren regionalen Forschungseinrichtungen und Partnern den Versuchsträger „Intelligent Car“ (Abbildung 1) eingerüstet.

Abbildung 1: Versuchsträger „Intelligent Car“.

2. Gesamtsystemarchitektur

Der gewählte Systemkontext bedingt eine Reihe unausweichlicher Hard- sowie Software-Entscheidungen, die zur Gestaltung einer Gesamtsystemarchitektur führen. Im Detail werden diese in [3] beschrieben, wobei für das weitere Verständnis dieser Arbeit hier nur einzelne Teilaspekte erläutert werden.

Systembeschreibung

Grundsätzlich ist das Gesamtsystem als Verarbeitungskette nach dem Pipes & Filters-Prinzip [4] strukturiert. Hierbei werden die eingehenden Daten auf unterschiedlichen Ebenen verarbeitet und nachgeschalteten Verarbeitungsprozessen asynchron zur Verfügung gestellt. Abbildung 2 zeigt überblicksartig die stromorientierte Architektur.

Die in dieser Arbeit beschriebene Software-Komponente *CoursePlanner*, die in Abbildung 2 orange dargestellt ist, bildet einerseits die logische Schnittstelle zwischen der Sensordatenaufbereitung und –interpretation und andererseits die technische Schnittstelle zwischen der Software, die in der Programmiersprache C++ umgesetzt wurde und der

Software, die über Matlab/Simulink realisiert wurde. Sie stellt damit einen kritischen Teil der gesamten Verarbeitungskette dar.

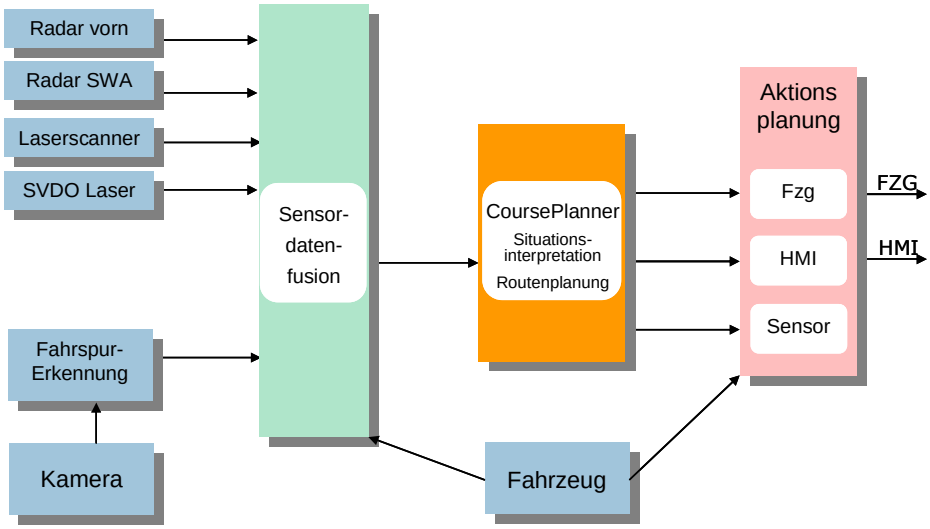

Abbildung 2: Grundsätzliche Systemarchitektur.

Modulanforderungen *CoursePlanner*

Die prozessierten Eingabedaten des Moduls *CoursePlanner* stellen in von konkreten Sensoren abstrahierter Form eine Abbildung des Systemkontexts als objektorientiertes Datenmodell dar. Dieses Datenmodell wird zyklisch nach Durchlauf eines gesamten Verarbeitungsschritts der Sensordatenfusion aktualisiert. Die Aufgabe des Moduls für die gesamte Datenstromverarbeitung besteht darin, das erzeugte Umfeldmodell zu interpretieren und kontinuierlich in ein diskretes Format zur Übertragung an die nachfolgende Verarbeitungsstufe zu überführen. Das diskrete Format dient diesen Prozessen zur Ableitung von konkreten Fahrentscheidungen, beispielsweise ein langsam voraus fahrendes Fahrzeug zu überholen. Die detaillierte Beschreibung hierzu ist in Abschnitt 3 zu finden.

Ergänzend zu dieser taktischen Entscheidungsvorbereitung leistet das Modul darüber hinaus eine strategische Entscheidungsunterstützung, in dem es Daten über zukünftige Charakteristika des geplanten Routenverlaufs zusammenfasst und kontinuierlich an die nachfolgende Verarbeitungsstufe überführt. Dieses Teilmodul weist im Gegensatz zum taktischen Planungsmodul eine ereignisorientierte Schnittstelle auf, über die folgende Verarbeitungsstufen Steuersignale absetzen können. Im Detail wird in Abschnitt 4 auf dieses Modul eingegangen.

Modularchitektur

Das Modul *CoursePlanner* selbst wurde vom Restsystem durch das bekannte Adapter-Entwurfsmuster [5] strukturell getrennt und kann als eigenständige Einheit betrieben werden. Ein wesentlicher Vorteil für diese Design-Entscheidung liegt darin begründet, das Modul lastorientiert auf einem anderen Rechnersystem einsetzen zu können und die benötigten Daten serialisiert [6] bereitzustellen. Darüber hinaus stellt das Software-Modul eine wiederverwendbare Schnittstelle für Nutzer einer Sensordatenfusion bereit. Dabei wurde vom konkret genutzten Datenmodell innerhalb der Sensordatenfusion an den wesentlichen Stellen abstrahiert, so dass die Schnittstelle unabhängig von der genauen Ausgestaltung des Umfelds und den eingesetzten Sensoren verwendet werden kann.

3. Taktische Fahrentscheidungsunterstützung

In diesem Abschnitt werden die grundsätzlichen Ideen hinter der taktischen Fahrentscheidungsunterstützung dargestellt. Darüber hinaus wird auf die technische Realisierung eingegangen.

Grundsätzliches Verfahren

Wie bereits im vorangegangenen Abschnitt skizziert wurde, soll das taktische Teilmodul für nachgelagerte Verarbeitungsstufen eine diskrete Entscheidungsvorbereitung durchführen. Wie in Abbildung 3 dargestellt, lässt sich der Systemkontext einer Schnellstraße oder Autobahn hierzu in einen begrenzten Zustandsraum unterteilen, der die Basis der hier dargestellten Lösung bildet. In dieser Abbildung dargestellt ist ein vereinfachter Ausschnitt einer dreispurigen Autobahn. Über der Fahrbahn liegen 25 Zellen einer quadratischen 5x5-Matrix, dessen mittleres Element die Bezeichnung „Ego“ trägt und das eigene Fahrzeug bezeichnet.
Die Matrix bildet die Grundlage zur Diskretisierung des Systemkontexts und ist selbst fahrzeugzentriert. Jede Zelle mit Ausnahme der mittleren hat eine spezifische Bedeutung relativ zum Eigenfahrzeug. Zunächst sind in den Zeilen die einzelnen Fahrspuren nachgebildet. In Fahrtrichtung links vom Eigenfahrzeug liegende Fahrspuren werden aufsteigend eindeutig nummeriert, in Fahrtrichtung rechts liegende Fahrspuren absteigend eindeutig.
In den Spalten werden die um das Eigenfahrzeug befindlichen Objekte des Umfelds nach dem im nächsten Abschnitt beschriebenen Verfahren einsortiert. Vor dem Fahrzeug liegende Zellen werden aufsteigend eindeutig, hinter dem Fahrzeug liegende Zellen absteigend eindeutig nummeriert.

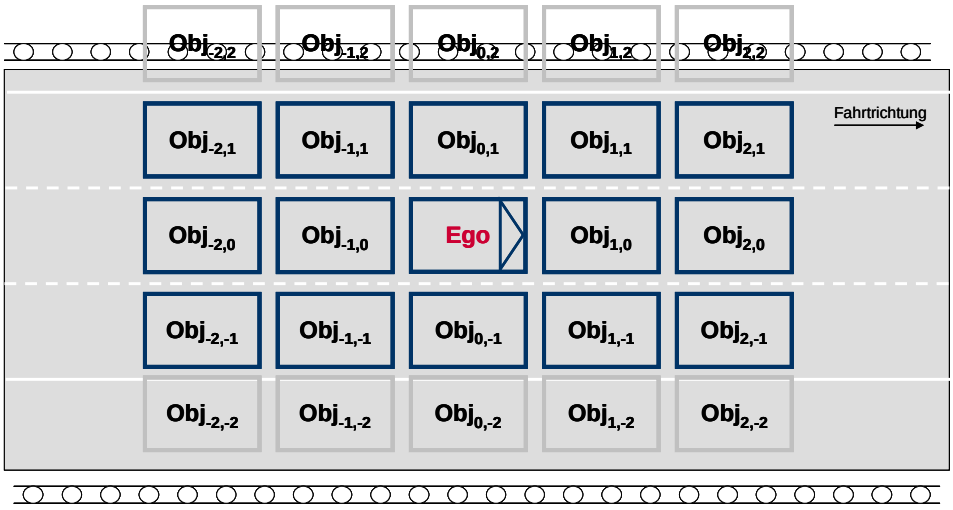

Abbildung 3: Schema zur Diskretisierung des Zustandsraums.

Die Indizierung der einzelnen Zellen erfolgt hierbei derart, dass zunächst die Spalten und anschließend die Zeilen adressiert werden. Das Eigenfahrzeug hat die Adresse (0;0). Zur sequentiellen Verarbeitung wird diese Indizierung zudem eindimensional eindeutig codiert, beginnend bei Objekt $Obj_{2,2}$, das die Nummer 1 trägt, bei Objekt $Obj_{-2,-2}$ mit der Nummer 25 endet und allgemein $Obj_{i,j}$ mit (13-5*i-j) codiert. Das Eigenfahrzeug hat daher den eindimensionalen Index 13. Die eindimensionale Indizierung hat sich bei der stromorientierten Datenübertragung auf dem CAN-Bus [7] bewährt.

Charakteristik-basierte Einsortierung

Die Einsortierung von Objekten erfolgt zunächst durch eine eindeutige Zuordnung zu einer Zeile. Innerhalb einer Zeile erfolgt die Sortierung *Charakteristik*-basiert, wobei die zu betrachtende Charakteristik frei definierbar ist. Spalten, die in der Matrix hinter dem Eigenfahrzeug liegen, tragen in ihren Zellen Objekte, die einen rückwärtigen Bezug zum Eigenfahrzeug aufweisen. Spalten, die vor dem eigenen Fahrzeug liegen, weisen einen vorwärts gerichteten Bezug auf.

Eine Charakteristik C ist konstant für eine konkrete Ausprägung $m_{C(t)}$ der Matrix M zu einem gegebenen Zeitpunkt t. Beispielsweise lässt sich der euklidische Abstand d eines Objekts zum Eigenfahrzeug als Charakteristik verwenden, um nur ein offensichtliches Kriterium zu benennen.

Unter dem Aspekt von Entwurfsmustern entspricht die Charakteristik-basierte Einsortierung einer *Strategie* [5]. Damit lassen sich, neben den zum Beispiel in [8] dargestellten Verfahren, je nach konkreter Anwendung weitere mögliche Charakteristika zur Einsortierung verwenden:

- Relativgeschwindigkeit zum Eigenfahrzeug
- Kollisionszeit zu einem gegebenen Objekt
- parametrisierbare Aufenthaltswahrscheinlichkeiten und Verwendung von Sensorunsicherheiten
- virtuelle Fahrzeuge in vehicle-in-the-loop-Simulationen für gefahrlose Fahrzeugtests

Transformation der Matrix bei Fahrspurwechseln

Bei Fahrspurwechseln des Eigenfahrzeugs erhalten die Zellen der Matrix eine andere Bedeutung, wäre sie nicht fahrzeugzentriert. Das bedeutet aber auch, dass die einzelnen Einträge der Zellen in ihrer lateralen Einsortierung nicht modifiziert werden dürfen, da sich ansonsten ein zum Umfeld inkonsistentes Bild ergäbe.

Zur Erhaltung eines konsistenten Umfelds wurde eine lateral rollierende Funktion in die Matrix implementiert, die beispielsweise bei einem Fahrspurwechsel von der mittleren Fahrspur auf die benachbart rechts liegende alle Elemente der Matrix um eine Zeile in die entgegen gesetzte Richtung verschiebt. Die zum Eigenfahrzeug relative Adressierung aller Elemente ist nach der Transformation identisch, so dass das Eigenfahrzeug weiterhin unter dem Index 13 lokalisiert ist, indem seine Daten dorthin übertragen werden. Ein Fahrzeug hingegen, das bislang unter dem eindimensionalen Index 2 zu finden war, wird nach der Transformation unter dem Index 1 adressiert. Diese Form der Transformation setzt nun voraus, dass die Zelle unter dem vormaligen Index 14 unbesetzt ist. Diese Annahme kann allerdings getroffen werden, da eine unbesetzte Zelle mit dem Index 14 der Situation entspricht, dass die neben dem Eigenfahrzeug befindliche Fahrspur für den Fahrspurwechsel frei ist.

Bearbeitung der Matrix am Beispiel euklidischer Distanzen

Die Einsortierung neuer Elemente in die Matrix unterliegt, wie im bestehenden Abschnitt angedeutet, einer je Zeitschritt konstanten Charakteristik. Wie in Abbildung 4 dargestellt, lässt sich die Aufgabe selbst, da eine objektorientierte Programmiersprache verwendet wird, als polymorphes Sortierungsproblem lösen.

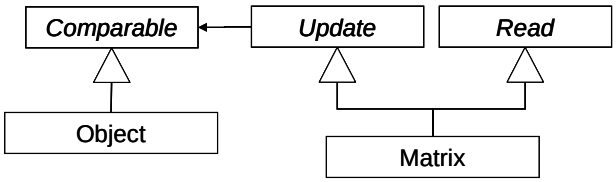

Abbildung 4: Read- und Update-Interfaces der Matrix.

Hierbei wird die Entscheidung, welches Kriterium für eine konkrete Sortierung verwendet wird, durch eine abstrakte Schnittstelle *Comparable* mit einer Vergleichsmethode `int compareTo(Object &o);` gekapselt. Die Methode liefert einen Wert kleiner als 0 zurück, wenn das betrachtete Objekt vor Objekt o liegt, genau 0, wenn sie keinen Unterschied zueinander aufweisen, und größer als 0, wenn das betrachtete Objekt hinter Objekt o liegt. Ein konkretes Objekt implementiert diese Schnittstelle und bildet entweder selbst den Vergleich ab oder delegiert den Aufruf an eine gewünschte Strategie.
Die Schnittstelle *Update* kapselt eine Methode zum Einfügen neuer Objekte vom Typ *Comparable*. Die Matrix selbst implementiert diese Schnittstelle und kann anhand verschiedener Sortieralgorithmen ein Objekt einfügen. In dieser Arbeit wurde sich für den klassischen Insertion-Sort-Algorithmus [9] entschieden, der insbesondere für kleine, bereits vorsortierte Datenmengen eine ausreichende Effizienz aufweist.
Die einsortierten Objekte selbst enthalten die für die Ableitung konkreter Fahrentscheidungen notwendigen Informationen wie Position relativ zum Eigenfahrzeug, Relativgeschwindigkeiten und Relativbeschleunigungen. Auf die Objekte innerhalb der Matrix kann über die abstrakte Schnittstelle *Read* zugegriffen werden, in der aufgrund der möglichen Operatorenüberladung die intuitive Schnittstelle `Object& operator[](const unsigned int &index);` bereitgestellt wird, die eine eindimensionale Adressierung der Elemente erlaubt, wie in Abschnitt 3.1 bereits erläutert wurde.

4. Strategische Fahrentscheidungsunterstützung

Die Komponente zur taktischen Entscheidungsunterstützung wird ergänzt um eine strategische Komponente zur Beeinflussung langfristiger Fahrentscheidungen. Hierbei spielen dynamische Objekte des Umfelds keine Rolle, sondern Elemente, die von heutigen Navigationsgeräten bekannt sind: Fahrbahnverlauf, Fahrspurenanzahl und vieles mehr.
In Abbildung 5 sind die Elemente, die im strategischen Teilsystem des Moduls verwendet werden, dargestellt:

- Entfernung eines Elements relativ zur derzeitigen Position des Eigenfahrzeugs
- Parameter für den Verlauf der zu Grunde liegenden mathematischen Beschreibung einer Fahrspur
- Anzahl Fahrspuren und Geschwindigkeitsinformationen

- Navigationsinformationen, um beispielsweise Fahrspurwechsel auf linksseitige Spuren kurz vor einer zu erreichenden Abfahrt zu verbieten

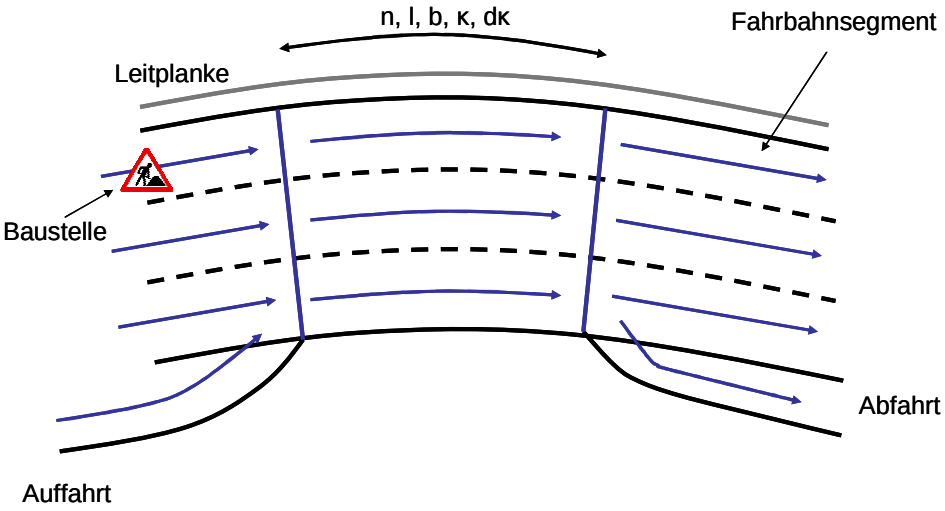

Abbildung 5: Elemente für die strategische Routenplanung.

Grundsätzliches Verfahren

Das Teilsystem bietet eine Vorschau über eine frei definierbare Distanz vor dem Eigenfahrzeug. Hierzu lokalisiert sich das Modul über die derzeitige Eigenposition in eine digitale Straßenkarte in Form eines Knoten-Kanten-Netzwerks, die eine Fahrroute zu einem vorab definierten Zielpunkt enthält.

Von dieser Position aus wird entlang der Kanten eine Queue bis zu einer maximalen Entfernung Φ vor dem Eigenfahrzeug mit den für die langfristige Entscheidungsvorbereitung notwendigen Daten befüllt. Die in die Queue eingefüllten Elemente enthalten alle die relative Entfernung zum Eigenfahrzeug sowie im Speziellen die ab der jeweiligen Entfernung gültigen Umfeldbedingungen, wie beispielsweise Geschwindigkeitsbeschränkungen oder –aufhebungen.

Die Elemente der Queue werden sequentiell an die nachfolgenden Verarbeitungsstufen übertragen, bis alle notwendigen Informationen übermittelt wurden. Anschließend werden nur noch neu eingefüllte Einträge weitergeleitet.

Verwaltung der Einträge der Vorschau

Die Kanten der digitalen Straßenkarte werden im Versuchsträger durch Klothoiden [10] modelliert. Zur Bestimmung der Entfernung einzelner Elemente ist hierzu die Bogenlänge zu

berechnen. Hierbei wird nach dem Simpson-Verfahren [11] approximiert, dessen Fehler bei Anwendung unterhalb von 0,05m für eine Bogenlänge bis 200m für maximal 20 Schritte liegt. Dieser Fehler ist für den adressierten Systemkontext hinreichend genau.
Sobald sich das Eigenfahrzeug bewegt, wird seine Positionsveränderung linear interpoliert. Aufgrund der zyklischen Aktualisierung des Teilsystems von etwa 10Hz ist dieses Vorgehen hinreichend genau. Das errechnete Veränderung δ wird von allen Elementen in der Queue subtrahiert. Sinkt die Entfernung des letzten und somit am weitesten vom Eigenfahrzeug entfernten Elements unter einen frei definierbaren Schwellenwert η ab, wird die Queue mit neuen Daten aus der digitalen Straßenkarte solange gefüllt, bis die Queue den Schwellenwert η wieder überschreitet. Somit wird gewährleistet, dass die Queue ausreichend viele Elemente enthält. Im Intelligent Car beträgt $\eta = 3.000$m.
Vom Fahrzeug passierte Elemente, deren Distanzen aufgrund der Aktualisierung um das errechnete δ negativ werden, werden nach einem speziellen Verfahren aus der Queue entfernt: Elemente, die Angaben über Krümmungen und Krümmungsänderungen enthalten, werden direkt entfernt, da sie keine Auswirkung auf das derzeitige Fahrverhalten haben. Werden hingegen Elemente passiert, die beispielsweise Angaben über geltende Geschwindigkeitsbeschränkungen enthalten, werden sie erst dann aus der Queue entfernt, sobald das semantisch inverse Element ebenfalls passiert wurde.

Ereignisorientierter Datenaustausch

Wie bereits in Abschnitt 2.2 angedeutet, weist dieses Teilsystem für die Routenplanung des Gesamtmoduls *CoursePlanner* eine ereignisorientierte Schnittstelle auf. Über diese Schnittstelle kann eine nachgelagerte Verarbeitungsstufe signalisieren, ob sie initial über alle Einträge der Queue oder nur über neue Elemente, die nach Einfügen aufgrund der Unterschreitung eines Schwellenwertes vorliegen, informiert werden möchte.
Dieses Ereignissystem vermindert die technische Abhängigkeit des Moduls von nachfolgenden Verarbeitungsstufen, da sie keine Zustandsüberwachung der nachgelagerten Module durchführen muss. Es überträgt sequentiell bei Modulstart einmalig alle Einträge und dann nur die neu hinzugekommenen Elemente ohne notwendiges Wissen nachgelagerter Verarbeitungsstufen. Benötigt hingegen eine nachgelagerte Verarbeitungsstufe erneut alle erforderlichen Informationen, kann sie diese über ein definiertes Signal über die Steuersignalschnittstelle anfordern.

5. Qualitätssicherung der entworfenen Algorithmen

Die entworfenen Algorithmen werden durch ein speziell auf den Projektkontext angepasstes Unit-Testframework [12] qualitätsgesichert. Hierzu wurden für beide Algorithmen jeweils umfangreiche Testsuites definiert, die unterschiedliche Aspekte der Algorithmen testfallspezifisch absichern.

Die Qualitätssicherung der taktischen Komponente gliedert sich insbesondere in die Absicherung der Einsortierungskriterien. Hierbei werden zunächst die verwendeten Datenstrukturen separat von ihrer Verwendung getestet. Durch Nutzung von Templates der Programmiersprache C++ konnte die Matrix mit primitiven Datentypen für die kritischen Funktionen der Transformation abgesichert werden, da die durchzuführende Transformation unabhängig von dem konkret enthaltenen Datentypen ist.

Weiterhin wurden verschiedene Verkehrssituationen für zweispurige und dreispurige Fahrbahnen in Form von Testfällen modelliert, in denen die longitudinale Einsortierung der Elemente für unterschiedliche Kriterien abgesichert wurde. Dieses Vorgehen ermöglichte die zeitnahe Inbetriebnahme am Versuchsträger.

Die Qualitätssicherung der strategischen Komponente erforderte wie zur taktischen Komponente vergleichbare Maßnahmen; insbesondere die Qualitätssicherung des ereignisorientierten Teilsystems. Zunächst wurde die im vorhergehenden Abschnitt beschriebene Befüllung der Queue mit unterschiedlichen Daten getestet. Hierbei wurde auch die Qualität der gewählten Approximationen an das zu Grunde liegende mathematische Modell durch Analyse des Restterms durchgeführt.

Darüber hinaus war der Zustandswechsel nachfolgender Verarbeitungsstufen notwendiges Testkriterium. Hierzu wurden die Übergänge von sequentieller Übertragung zu vollständiger Übertragung bei bereits passierten semantischen Paaren wie etwa Geschwindigkeitsbeschränkungen betrachtet. Auch hier konnte durch Softwaretests, die ohne Nutzung eines Versuchsträgers ausgeführt werden können, zeitnah eine Software-Qualität erreicht werden, die eine rasche Inbetriebnahme am Versuchsträger ermöglichte.

6. Verwandte Arbeiten

Eine thematisch stark verwandte Arbeit ist das vom Institut für Software Systems Engineering geführte und von der Volkswagen AG geförderte CarOLO–Projekt [13, 14]. Die Zielsetzung dieses Projekts lag komplementär zum Intelligent Car in der Einrüstung eines Versuchsträgers

für urbane Umgebungen für die Teilnahme an der DARPA Urban Challenge 2007 [2]. Dort werden ebenfalls Ansätze zur Erzeugung von Belegungsgittern diskutiert, die im Gegensatz zu dem hier beschriebenen Verfahren jedoch stark auf den Sensorrohdaten beruhen.

Der Schwerpunkt der Arbeiten in [15, 16] lag in der integrierten, mehrstufigen Qualitätssicherung für das CarOLO–Projekt. Ergebnisse dieser Arbeit nahmen Einfluss auf [12] und somit rückwirkend auch auf das Intelligent Car–Projekt. Vorarbeiten für ausgewählte Aspekte einer programmtechnischen Qualitätssicherung wurden bereits in [17] dargelegt.

In [8] diskutieren die Autoren verschiedene Verfahren zur qualitativen Bewertung von Verkehrssituationen. Wie bereits im Abschnitt 3.2 angedeutet, lassen sich Ergebnisse dieser Arbeit in zukünftige Algorithmen zur Vorbereitung und Ableitung von Fahrentscheidungen einbeziehen.

Vielversprechend sind auch Ergebnisse der Arbeit von [18]. Dort entwerfen die Autoren ein zweigeteiltes Verfahren zur Makro- und Mikro-Simulation von Verkehrsströmen. Damit ließe sich die Qualitätssicherung der Algorithmen dieser Arbeit verbessern.

Die Arbeiten in [19] und [20] beschreiben unabhängige, gewichtete Verfahren, die ein Fahrzeug prinzipiell fortwährend in Bewegung halten können. Im Rahmen der DARPA Grand Challenge 2005 war dieser Ansatz sehr vielversprechend, da es sich um ein per Definition statisches Umfeld handelte. In dynamischen Umgebungen wie Autobahnen sind diese Ansätze jedoch nur unzureichend.

7. Zusammenfassung

Die Volkswagen AG hat gemeinsam mit dem Institut für Software Systems Engineering und weiteren Partnern einen Versuchsträger eingerüstet, der in der Lage ist, auf Schnellstraßen und Autobahnen automatisch zu fahren. In dieser Arbeit wurde die Komponente *CoursePlanner* dargestellt, die taktische und strategische Fahrentscheidungen vorbereitet und somit ein essentieller Teil eines datenstromorientierten Verarbeitungskonzepts ist.

Die taktische und strategische Vorbereitung der Fahrentscheidungen wurde in der Software-Komponente gleichfalls abgebildet. Dazu wertet die Komponente das durch die Sensorfusion erzeugte abstrakte Umfeldmodell aus und leitet daraus für die Vorbereitung taktischer Fahrentscheidungen eine fahrzeugzentrierte, abstrakte Matrix ab, die alle notwendigen Informationen für die Fahrzeugregelung enthält. Zur Vorbereitung strategischer Fahrentscheidungen wird eine vorausschauende Routenplanung aus den Daten der

Sensorfusion generiert, dessen Daten ereignisorientiert und unter Berücksichtigung technischer Einschränkungen zur Fahrzeugregelung übertragen werden.

Die erstellten Teilkomponenten werden durch ein Unit-Testframework qualitätsgesichert. Dazu wurden verschiedene Verkehrsszenarien definiert, die sowohl die Vorbereitung taktischer als auch strategischer Fahrentscheidungen automatisiert und kontinuierlich testen.

Literatur

[1] A. Bartels: *Roadmap Automatisches Fahren*. In: 9. Braunschweiger Symposium für Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel, 2008.

[2] DARPA Urban Challenge, http://www.darpa.mil/grandchallenge/index.asp, 2007.

[3] A. Bartels, S. Steinmeyer, A. Weiser: *Intelligent Car – Teilautomatisches Fahren auf der Autobahn*. In: 10. Braunschweiger Symposium für Automatisierungssysteme, Assistenzsysteme und eingebettete Systeme für Transportmittel, 2009.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: *Pattenr-Oriented Software Architecture*. John Wiley & Sons, 1996.

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns*. Addison-Wesley, 1996.

[6] D. Riehle, W. Siberski, D. Bäumer, D. Megert, H. Züllighoven: *Serializer*. In: Pattern Languages of Program Design 3, Addison-Wesley, 1998.

[7] ISO International Organization for Standardization: *ISO 11898: Controller area network (CAN)*, 1993.

[8] J. Gehrke, A. Lattner, O. Herzog: *Qualitative Mapping of Sensory Data for Intelligent Vehicles*. Workshop on Agents in Real-Time and Dynamic Environments at the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, July 30, 2005, pp. 51-60.

[9] D. Knuth: *The Art of Computer Programming*. Vol. 3: Sorting and Searching. Addison–Wesley, 1999.

[10] H. Kasper, W. Schürba, H. Lorenz: *Die Klothoide als Trassierungselement*. Dümmler, 1968.

[11] G. Merziger, G. Mühlbach, D. Wille, T. Wirth: *Formeln + Hilfen zur Höheren Mathematik*. Binomi, 1996.

[12] C. Basarke: *Anwendung von Testkonzepten in einem automobilen Forschungsprojekt*. Diplomarbeit, Technische Universität Braunschweig, 2006.

[13] C. Basarke, C. Berger, K. Berger, K. Cornelsen, M. Doering, J. Effertz, T. Form, T. Gülke, F. Graefe, P. Hecker, K. Homeier, F. Klose, C. Lipski, M. Magnor, J. Morgenroth, T. Nothdurft, S. Ohl, F. Rauskolb, B. Rumpe, W. Schumacher, J. Wille, L. Wolf: *Team CarOLO – Technical Paper*. Informatik-Bericht 2008-07, Technische Universität Braunschweig, 2008.

[14] F. Rauskolb, K. Berger, C. Lipksi, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher J-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier J. Morgenroth, L. Wolf, C. Basarke, C. Berger, T. Gülke, F. Klose, B. Rumpe: *Caroline: An Autonomoulsy Driving Vehicle for Urban Environments*. Journal of Field Robotics, Wiley Periodicals, Vol. 25, Nr. 9, S. 674-724, September 2008.

[15] C. Basarke, C. Berger, B. Rumpe: *Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence*. Journal of Aerospace Computing, Information, and Communication (JACIC) Vol. 4, Nr. 12, S. 1158-1174, Oktober 2007.

[16] C. Basarke, C. Berger, K. Homeier, B. Rumpe: *Design and quality assurance of intelligent vehicle functions in the "virtual vehicle"*. 11th Automotive Technology Conference, Virtual Vehicle Creation 2007, Vieweg Technologie Forum, 2007.

[17] B. Rumpe, C. Berger, H. Krahn: *Softwaretechnische Absicherung intelligenter Systeme im Fahrzeug*. Proceedings der 21. VDI/VW-Gemeinschaftstagung – Integrierte Sicherheit und Fahrerassistenzsysteme, Oktober 2006.

[18] J. Olstam, J. Lundgren, M. Adlers, P. Matstoms: *A Framework for Simulation of Surrounding Vehicles in Driving Simulators*, ACM Transaction on Modeling and Computer Simulation, 2008.

[19] J. Rosenblatt: *DAMN: Distributed Architecture for Mobile Navigation*. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1997.

[20] H. Kjellander: *Arbiter and Simulation for Team Caltech in DARPA Grand Challenge*. Technischer ericht, Department of Automatic Control, Lund Institute of Technology, 2004.