Hochschule Reutlingen
Reutlingen University

INF
Informatik

Uwe Kloos, Natividad Martínez, Gabriela Tullius (Hrsg.)

Informatics Inside
connect(IT);

Informatik-Konferenz an der Hochschule Reutlingen
9. Mai 2018

9 783000 586453

Impressum

Anschrift:
Hochschule Reutlingen / Reutlingen University
Fakultät Informatik
Human-Centered Computing
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4042

E-Mail: infoinside@reutlingen-university.de
Internet: http://infoinside.reutlingen-university.de

Organisationskomitee:
Prof. Dr. Gabriela Tullius, Hochschule Reutlingen
Prof. Dr. Natividad Martínez, Hochschule Reutlingen
Prof. Dr. Uwe Kloos, Hochschule Reutlingen

Benjamin Batt
Claudiu Bräuer
Sinem Cicek Celik
Emanuel Geiger
Frauke Grieble
Peter Grupp
Pia Laubacher
Öznur Öner
Katharina Pavic
Ngoc Linh Phan
Claudia Ryniak
Josia Scheytt
Christian Steinmann
Clemens Weißenberg
Vanessa Willenbrock
Steffen Wittig

Copyright: © Hochschule Reutlingen, Reutlingen 2018
Herstellung und Verlag: Hochschule Reutlingen
ISBN: 9 -83000-586453

Hochschule Reutlingen
Reutlingen University

Inhaltsverzeichnis

Longpaper

Konzept einer Motion-Capture basierten Simulationsumgebung zur Untersuchung von Interaktionen zwischen Passanten und autonomen Fahrzeugen*

David Leisten
Reutlingen University
david.leisten@Student.Reutlingen-University.DE

Abstract

Im Rahmen dieser Arbeit wurde eine Software-Architektur entwickelt, mit der sich Interaktionen zwischen autonomen Fahrzeugen und Passanten im Straßenverkehr in einer simulierten Umgebung untersuchen lassen. Hierbei wird das autonome Fahrzeug durch einen externen Fahrsimulator gesteuert. Der Einsatz eines Motion-Capture-Systems ermöglicht dabei die Aufzeichnung und Übertragungen der Bewegungsdaten von Passant und Fahrer in die virtuelle Umgebung. Durch den Einsatz von Head-Mounted Displays sollen Akteure die virtuelle Umgebung möglichst als real empfinden. Auf Basis der entwickelten Software-Architektur wurde eine Simulationsumgebung realisiert, in der Interaktionen zwischen einem Passant und einem autonomem Fahrzeug untersucht werden können. Das Projekt soll das Potential von Motion-Capture gestützten Simulationsumgebungen für die Konzeption und Entwicklung von autonomen Fahrsystemen aufzeigen.

Schlüsselwörter

Motion-Capture, Virtual Reality, Simulation, Interaktion

CR-Kategorien

I.6m [**SIMULATION AND MODELING**]: Miscellaneous; I.3.7 [**COMPUTER GRAPHICS**]: Three-Dimensional Graphics and Realism, Virtual reality

1 Einleitung

Die Entwicklung autonomer Fahrzeuge schreitet aktuell schnell voran. Jedoch ergeben sich für die erfolgreiche Integration autonomer Fahrzeuge in den Straßenverkehr eine Vielzahl von Herausforderungen. Die Verkehrssicherheit der Verkehrsteilnehmer sowohl in urbanen als auch ländlichen Umgebungen steht in einem engen Zusammenhang mit einer erfolgreichen Interaktion zwischen allen Straßenverkehrsteilnehmern. Diese Interaktionen können sehr verschieden ausfallen. So kann sich bspw. die Interaktion zwischen einem Autofahrer und einem Passanten auf einen einfachen Blickkontakt beschränken. Dieser Blickkontakt ist für die Einschätzung der aktuellen Situation beider Verkehrsteilnehmer essentiell [6]. Damit Anforderungen an autonome

*

Betreuer Hochschule: Prof. Dr. Cristobal Curio
Hochschule Reutlingen
Cristobal.Curio@Reutlingen-University.de
Betreuer Hochschule: Dennis Ludl, M.Sc.
Hochschule Reutlingen
Dennis.Ludl@Reutlingen-University.de
Informatics Inside 2018
Wissenschaftliche Vertiefungskonferenz
09. Mai 2018, Hochschule Reutlingen
Copyright 2018 David Leisten

Fahrzeuge im Bezug auf das Verhalten im Straßenverkehr besser formuliert werden können, ist es notwendig, die Interaktionen zwischen diversen Verkehrsteilnehmer in verschiedensten Situationen zu verstehen und die gewonnen Erkenntnisse in geeigneter Weise auf das Verhalten von autonomen Fahrzeugen zu übertragen. Die mögliche Vielfalt an Situationen bringt dabei einige Problematiken mit sich. Zum einen ist es nicht wünschenswert, alle erdenklichen Interaktionssituationen regelbasiert in autonome Fahrzeuge zu übertragen, sondern ein geeignetes Interaktionskonzept zu finden; zum andern ist die Beobachtung und Auswertung vieler verschiedener Interaktionen in realen Umgebungen nach [5] limitiert und aufwändig.

Die in dieser Ausarbeitung vorgestellte Software-Architektur soll durch den Einsatz virtueller Simulationsumgebungen und präzisem Motion-Tracking dazu beitragen, die Bandbreite an Testumgebungen zu vergrößern, kosten für Testdurchführung zu verringern und das Repertoire auswertbarer Daten zu vergrößern.

Nach einem kurzen Überblick über den Stand der Technik (Abschnitt 2), wird in Abschnitt 3 die zentrale Software-Architektur vorgestellt, gefolgt von einer konkreten methodischen Umsetzung in Abschnitt 4, die derzeit im Rahmen des OFP-Projektes[1] (Open Fusion Plattform), durch das Bundesministerium für Bildung und Forschung gefördert, eingesetzt und weiterentwickelt wird. Die Arbeit schließt mit einer Diskussion der umgesetzten Implementierung und zeigt künftige Erweiterungen der Software-Architektur auf.

2 Stand der Technik

Damit Interaktionen und mit ihnen verbundene Handlung im Straßenverkehr beobachtet und analysiert werden können, werden aktuell vorwiegend reale Situationen im Straßenverkehr betrachtet. In einer Studie von Schneemann und Gohl [5] wurde das Interaktionsverhalten von 25 Autofahrern und Fußgängern an Zebrastreifen unter realen Verkehrsbedingungen beobachtet und analysiert. Dazu definierten sie eine Teststrecke auf der die Testfahrer und Passanten aufeinander trafen. Die Teilnehmer wurden anschließend mittels Fragebogen interviewt. Die Studie zeigte, das das Fahrverhalten und Aufmerksamkeit eines Autofahrers, eng mit der Geschwindigkeit zusammen hängt, mit der er sich dem Zebrastreifen nähert. Darüber hinaus zeigten die Ergebnisse der Studie, dass die Fahrzeuggeschwindigkeit einen signifikanten Einfluss, auf die Entscheidung des Passanten die Straße zu überqueren, hat.

Eine weitere Studie, zeigt das Verhalten von Verkehrsteilnehmern auf autonome Fahrzeuge zu untersuchen, wurde im Rahmen der Studie Ghost Driver von Rothenburg *et al* [4] der Universität Stanford durchgeführt. Dazu tarnten sie den Fahrer als Fahrersitz und ließen so das Fahrzeug als autonom erscheinen. Im Verlauf des Feldtests wurde eine Vielzahl von Interaktionen mittels einer Videokamera aufgezeichnet und diese anschließend ausgewertet. Zudem wurden die Passanten befragt, die auf das autonome Fahrzeug trafen. Diese berichteten, dass sie zwar keinen Fahrer im Fahrzeug sehen konnte, sie aber dennoch ohne Schwierigkeiten das Verhalten des Fahrzeuges einschätzen konnten. Ausgenommen, das Fahrzeug verhielt sich entgegen der Erwartungen des Passanten, z.B. durch Anfahren an Zebrastreifen, wenn der Passant im Begriff war, die Straße zu überqueren.

Resultierend aus den Ergebnissen wie sie Schneemann und Gohl liefern, gehen Langström und Lundgren [12] dazu über, konkrete Vorschläge für Interaktionsmechnismen zu entwickeln und diese ebenfalls in realen Umgebungen zu testen. Dazu

[1] Open Fusion Plattform, HELLA GmbH&Co.KGaA. 2016 `http://www.ofp-projekt.de/ofp-project/de/index.html`[8]

montieren sie auf dem Dach ihres Testfahrzeuges eine LED-Leiste, die Passanten auf das aktuelle Vorhaben des Fahrzeuges in Form von visueller Interaktion aufmerksam machten. Für ihre Studie instruierten sie die Passanten über die Bedeutung der verschiedenen visuellen Feedbacks und ließen diese anschließend deren Verständlichkeit und Intuitivität in inszenierten Testsituationen evaluieren.
Der Einsatz von virtuellen Simulationsumgebungen, wie Hartmann *et al* [7] sie in ihrer Arbeit *Pedestrian in the Loop* vorschlagen, kann dazu genutzt werden, verschiedene Szenarien im Straßenverkehr zu simulieren.

3 Software-Architektur

Im folgenden Abschnitt werden die einzelnen Komponenten, die zur Gesamtheit des Konzeptes beitragen, vorgestellt.

3.1 Komponenten

Das entwickelte Konzept basiert auf den folgenden vier Komponenten.

1. Das Motion-Capturing-System der Firma VICON [14]
2. Die Head Mounted Displays (HMD) Oculus Rift und HTC Vive
3. Ein Fahrsimulator mit einem G29 Driving Force-Rennlenkrad der Firma Logitech [9]
4. Die Unity3D Spiel-Engine

Das optische Motion-Capturing System von VICON bietet ein präzises und robustes Motion-Tracking und kann, abhängig vom Aufbau ein für den Zweck dieser Software-Architektur ausreichend großes Tracking-Volumen abdecken. Darüber hinaus verfügt das VICON System über eine TCP/IP-Schnittstelle, über die die Trackingdaten an eine beliebige Anzahl Clients übertragen werden können.

Die HTC Vive wurde als HMD Komponente ausgewählt, da sich das für das Tracking verwendete Lighthouse-System von Valve gut für weiträumige Tracking-Volumen eignet. Prinzipiell kann dieses System durch weitere Tracking-Basisstationen vergrößert werden. Somit ist gewährleistet, dass sich die getrackte Person im gesamten Tracking-Volumen bewegen kann. Der Einsatz der Oculus Rift ist im Rahmen der Software-Architektur nicht bindend und kann durchaus durch den Einsatz einer weiteren HTC-Vive umgesetzt werden. Allerdings ist der mögliche Tracking-Volumen der mit der Oculus Rift abgedeckt werden kann, deutlich kleiner als das der HTC-Vive.

Der Einsatz eines Fahrsimulators bietet die Möglichkeit, Informationen über Lenkwinkel oder Stellung der Gas- und Bremspedale direkt in die Simulation einzubringen. Darüber hinaus erlaubt das eingebaute Force-Feedback System ein situationsabhängiges Feedback an den Benutzer des Fahrsimulators. Die Unity3D Spiel-Engine wird als Komponente für die Konstruktion und die Ausführung der Virtuellen Realität eingesetzt. Zu den ausgeführten Komponenten existieren bereits frei zugängliche Modelle und Implementierungen die mittels des Unity Asset Stores in die Simulationsumgebung (in Unity3D *Szenen* genannt) integriert werden können.

3.2 Gesamtsystem

Das konzipierte System orientiert sich am Network Systems Concepts der Dokumentation der Unity3D Spiel-Engine [13]. Das Network Systems Concepts beinhaltet die grundlegenden Funktionalitäten und Konzepte, für die Erstellung netzwerkbasierter Anwendungen in Unity3D. Die Software-Architektur wurde gewählt, da sich in der Implementierung (Abschnitt 4) gezeigt hat, dass die Verwendung zweier HMDs am selben Rechner und in der selben Laufzeitumgebung nicht umsetzbar war. Abbildung 1 zeigt den Aufbau des Gesamtsystems und wurde mittels des Fundamental Modeling Concepts (FMC) modelliert. Hierbei bildet der Agent *Server* die zentrale Schnittstelle

zwischen den verbundenen Clients. Die Clients existieren wie auch der Server in eigenständigen Laufzeitumgebungen. Da diese in Unity3D als Szenen verstanden werden, wird im folgenden der Begriff Szenen verwendet. In der hier vorgestellten Software-

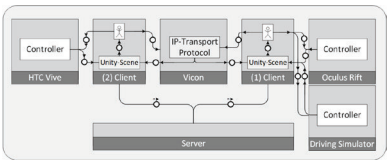

Abbildung 1: Darstellung des Gesamtsystem als Block Diagramm nach FMC [3]

Architektur werden zwei Typen von Clients unterschieden. Der (1) Client (siehe Abbildung 1) stellt den Fahrer dar. Der (2) Client den Passanten. Beide Clients kommunizieren über den Server mittels *Request-Responce*-Kanäle. Der (1) Client interagiert mit der Szene mittels des Fahrsimulators. Die Informationen aus dem Fahrsimulator werden dabei mittels des Agenten *Controller* an die Szene in Unity3D übertragen und ermöglichen so die Steuerung des Fahrzeuges in der virtuellen Umgebung. Die Bewegungen der Person in der Rolle des Fahrers können mittels der Komponente Vicon durch den Agenten *IP-Transport Protocol* ebenfalls an die Szene in einen beliebigen Avatar übertragen werden. Die Person in der Rolle des Fahrers kann so nicht nur das Fahrzeug durch die Szenen steuern, sondern kann ebenso ihre eigenen Bewegungen in Gestalt eines Avatars wahrnehmen. Die Komponente der Oculus Rift überträgt auf Basis der Kopfbewegung des Fahrers das Kamerabild der virtuellen Realität zurück an den Fahrer. Der (2) Client kann sich in der Szene mittel der Komponente Vicon mit seinem Avatar frei bewegen.

3.3 Server-Client Kommunikation

Abbildung 2 zeigt den Aufbau der Server-Client Kommunikation. Zu Beginn stellt der Server über den Agenten *Communication Service* mit den verfügbaren Clients über einen Request-Responce Channel eine Verbindung zu den verfügbaren Clients her und erfragt deren Rolle (Passant oder Fahrer). Anschließend erstellt der Server mittels des Agenten *Scene Configuration* über einen Lesezugriff aus dem Speicher (Files) die Konfiguration der Szene. Damit die beiden Akteure (Passant und Fahrer) innerhalb der virtuellen Umgebung interagieren können, müssen Informationen über Position und Bewegung zwischen den einzelnen Akteuren übertragen werden. Den Inhalt der Szene defi-

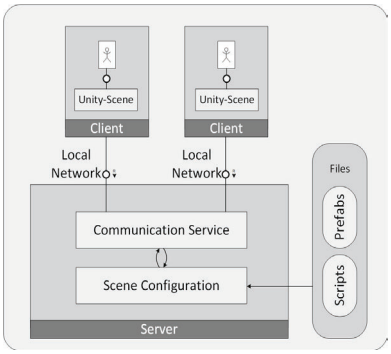

Abbildung 2: Server Client Kommunikation Block Diagramm nach FMC [3]

niert zudem die Rolle des verbundenen Clients. Über den Agenten *Communication Service* werden zur Laufzeit der Simulation alle relevanten Informationen zwischen den Clients ausgetauscht. Dazu gehört in erster Linie das Verhalten (Rotation, Position und Lenkwinkel) des Fahrzeuges in der virtuellen Umgebung, das vom Client in der Rolle des Fahrers gesteuert wird. Die Übertragung der Bewegungen der Avatare beider Clients durch Vicon muss nicht über den Server zwischen den Akteuren kommuniziert werden. Da die Komponenten Vicon eine eigenständige Server-Client Kommunikation zur Verfügung stellt (siehe Abschnitt 4.2) und einen Broadcast aller im Trackingvolumen aktiven Akteure durchführt, können sich beide Clients die Bewegungen aller Avatare direkt

über diesen Broadcast abgreifen. So kann die ausgetauschte Menge an Daten zwischen den Clients reduziert werden und eine aufwändige Interpolation der Bewegung beider Akteure in der Szene umgangen werden. Interpolation wird in der Spielindustrie vor allem im Online-Multiplayer Bereich eingesetzt. Da nicht alle Bewegungsaktionen, aller Spieler in einem Netzwerk übertragen werden können, ohne das dies die Spielqualität drastisch beeinflusst, werden bspw. Charakterbewegungen nur in bestimmten Zeitintervallen zwischen den Spielern ausgetauscht. An dieser Stelle kommt Interpolation zum Einsatz, um bspw. den Pfad einer Bewegungen eines Objektes, aus einer begrenzten Anzahl Positionsdaten zu rekonstruieren und die Bewegung mit möglichst geringer zeitlicher Verzögerung, flüssig erscheinen zu lassen.

3.4 Konzeption des Clients: Fahrer

Der Client *Fahrer* (abgebildet als menschlicher Akteur in Abbildung 3) besteht aus vier Hauptkomponenten.

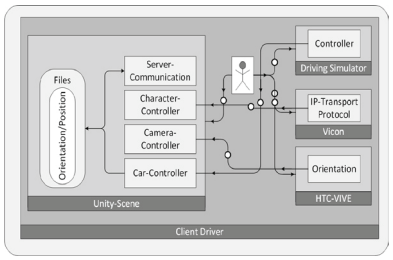

Abbildung 3: Konzept des Clients in der Rolle des Fahrers als Block Diagramm nach FMC

1. *Driving Simulator:* Die Komponente wird aktiv durch den menschlichen Akteur beeinflusst, indem dieser den Fahrsimulator bedient. Der Akteur *Controller* verarbeitet diese Eingaben und überträgt diese an den Agenten *Car-Controller* der Komponente *Unity-Scene*.

2. *Vicon:* Zur besseren Veranschaulichung überträgt der Akteur in dieser Komponente seine Körperbewegungen an das Motion-Capture-System. Streng genommen verfügt diese Komponente nur über die Eigenschaft, die durch das Motion-Capturing aufgezeichneten Bewegungsdaten des Fahrers aus dem Vicon System zu erhalten. Diese Bewegungsdaten werden an den Agenten *Character-Controller* der Komponente *Unity-Scene* übergeben.

3. *HTC-Vive:* Die Kopfbewegungen des Akteurs werden mittels des Lighthouse-Trackings der HTC-Vive berechnet und in Form lokaler Rotations- und Positionsdaten an den Agenten *Camera Controller* der Komponente *Unity Scene* übergeben.

4. *Unity-Scene:* Diese Komponente erhält wie bereits beschrieben die Eingaben aller anderen Komponenten und verarbeitet diese mittels ihrer eigenen Agenten. Sie ist somit dafür zuständig die Szene gemäß den vorhandenen Daten zu erstellen und zu aktualisieren. Der Agent *Car-Controller* berechnet aus den eingehenden Lenkbewegungen und Gas- bzw. Bremsintensitäten des *Driving Simulators* Eingaben für das in der virtuellen Realität vorhandene Fahrzeug. Dazu gehören Einschlagwinkel der Räder, Beschleunigungs- und Bremsverhalten. Der Agent *Car-Controller* überträgt anschließend die globalen Bewegungsdaten sowie den Einschlagwinkel per Schreibzugriff auf einen Speicher, aus dem wiederum der Agent *Server-Communication* lesend zugreifen kann. Je nach Anfrage seitens des Servers können so die Bewegungsdaten des Fahrzeuges an den Client (Passant) übertragen werden.

   Zuständig für die Darstellung der virtuellen Umgebung ist der *Camera-Controller*. Dieser errechnet aus den

Kopfbewegungen des Akteurs das entsprechende Kamerabild. Dazu kann in Unity3D die nativ vorhandene Virtual Reality Funktionalität verwendet werden. Der *Character-Controller* erhält seinen Input vom Akteur *IP-Transport Protocol*. Das Vicon Systems baut eine Serververbindung auf, auf die sich Clients registrieren können. Das *IP-Transport Protocol* kann abhängig von der gewünschten Framezahl eine Anfrage an den Server senden und erhält die Bewegungsinformationen aus dem Livetracking des Akteurs in einem definierten Skelett Modell zurück. Damit diese Informationen korrekt in der virtuellen Umgebung dargestellt werden, muss das Charaktermodell in Unity, auf das die Bewegungsdaten übertragen werden sollen, die selbe Konfiguration besitzen wie das definierte Skelett-Modell.

3.5 Konzeption des Clients: Passant

Prinzipiell ist die Funktionsweise des Clients für den Passanten fast identisch zu dem des Fahrers. In Abbildung 4 ist dieser Unterschied in der Übertragung der Informationen des *Car-Controllers* sichtbar. Hier wer-

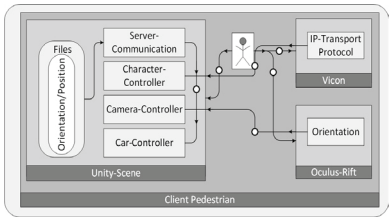

Abbildung 4: Aufbau des Client *Pedestrian*

den die Bewegungsdaten für das in der Szene vorhandene Fahrzeug aus dem Agenten *Server-Communication* bezogen. Wie bereits in Abschnitt 3.5 vorgestellt, werden die Fahrzeugbewegungen über die Server-Client Kommunikation ausgetauscht. Somit kann der Client des Passanten auf Anfrage an den Server die aktuellen Eigenschaften des Fahrzeugs in der Szene aktualisieren. Die Bewegungsdaten für den Avatar werden direkt mittels des *IP-Transport Protocol* abgerufen und in der Szene aktualisiert.

4 Umsetzung

Im folgenden Abschnitt wird die Simulationsumgebung beschrieben, die im Rahmen dieses Projektes umgesetzt wurde. In der Umsetzung wurde aus Zeitgründen keine Server-Client Komponente umgesetzt, sowie die Komponente des Fahrsimulators durch die eines autonomen Fahrzeuges ersetzt. Zwar wurde die Funktionalität des Fahrsimulators, darunter Ansteuerung durch eine Person und Übertragung der Daten aus dem G29 Driving Force-Rennlenkrad realisiert, jedoch aufgrund der fehlenden Server-Client Komponente, nicht in das Projekt integriert. Somit ist die Simulationsumgebung derzeit auf einen menschlichen Akteur beschränkt. Abbildung 5 zeigt, welche Komponenten umgesetzt wurden.

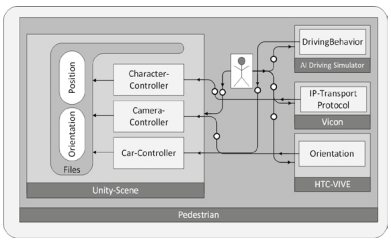

Abbildung 5: Umgesetzter Client *Pedestrian*

4.1 KI-Fahrzeugsteuerung

In der Umsetzung wurde eine KI gesteuerte Variante eines Fahrzeuges implementiert. Diese ersetzt in der aktuellen Umsetzung den Fahrsimulator. Das verwendete Fahrzeug wurde so konzipiert, dass es nur Eingaben für Lenkwinkel, Beschleunigungs- und Bremsintensität erwartet und damit sowohl durch eine KI-Steuerung als auch durch den Fahrsimulator steuerbar ist. Ab-

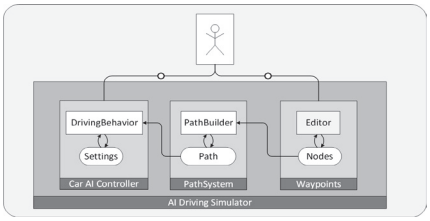

Abbildung 6: Aufbau des AI Car Controllers

bildung 6 zeigt die Funktionalität der KI-Fahrzeugsteuerung. Vor dem Start der Simulation muss festgelegt werden, welchem Pfad das autonome Fahrzeug folgen soll. Dazu müssen vorab Wegpunkte definiert werden. Diese sind in der Komponente *Waypoints* gespeichert. Das *PathSystem* verwendet diese Wegpunkte, um daraus mittels einen zusammenhängenden Pfad zu erstellen. Abbildung 7 veranschaulicht die Erstellung des Pfades. Hierzu kann die in Unity3D bereits vorhandene Klasse *Handles* mit der Funktion *Handles.DrawBezier* verwendet werden.

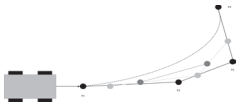

Abbildung 7: Pfadapproximation für ein glatteres Fahrverhalten

Im nächsten Schritt wird mit der Komponente *Car AI Controller* der konstruierte Pfad für das Fahrverhalten aufgegriffen. Die Komponente *DrivingBehavior* navigiert dabei entlang des Pfades und übergibt dazu Lenkwinkel und Beschleunigungs- bzw. Bremsverhalten entsprechend der Kurvenschärfe an das Fahrzeug. Wie die Abbildung 6 zeigt, können hierbei durch den Benutzer zusätzlich Parameter für das Verhalten des Fahrzeuges bestimmt werden. Anpassbare Eigenschaften sind hierbei: Kurvengeschwindigkeit, erlaubte Abweichung vom eigentlichen Pfad und der Schwellenwert für die Abbremsdistanz bei Erreichen des Ziels. Abbildung 8 zeigt das Pfadsystem in der Simulationsumgebung sowie das autonome Fahrzeug, das dem vorgeschriebenen Pfad folgt.

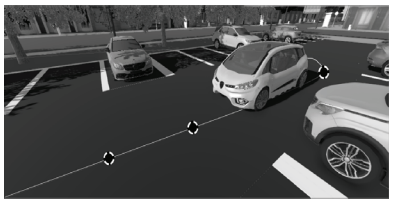

Abbildung 8: Umsetzung des Pfadsystems in der Simulationsumgebung

4.2 Streaming: VICON zu Unity3D

Die Livedaten aus dem Motion-Capturing System werden seitens des VICON Systems an eine frei wählbare IP-Adresse über einen beliebigen Port gestreamt. Die Motion-Capture Daten werden hierzu in ein *.bvh* Format übersetzt, sodass nur Rotationen und Translation der Gelenke eines definierten Skelettes über das Streaming übertragen werden. Abbildung 9 zeigt die Konfiguration des definierten Skelettes. Diese Konfiguration muss jedem Modell zu Grunde liegen, damit es durch das Vicon System in einer virtuellen Umgebung gesteuert werden kann. Um die Skelettdaten in Unity3D zu

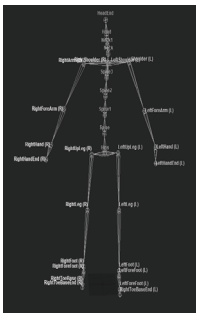

Abbildung 9: VICON Skelettdefinition

übertragen, kommt das *Vicon Pegasus SDK* zum Einsatz. Das SDK wird von VICON zur Verfügung gestellt. Die Komponenten *Vi-*

con in Abbildung 5 greift auf die Funktionen des SDKs zu. Die *Vicon* Komponente baut über das SDK eine Verbindung zum Server des VICON Systems auf und kann, abhängig von der gewünschten Updaterate, die aktuellen Motion-Capture Daten anfordern. Diese müssen anschließend in ein Skelett übertragen werden, welches der Konfiguration des *.bvh* Formates entspricht. Abbildung 10 zeigt

Abbildung 10: Übertragung der Motion-Capture Daten auf einen Avatar

die Visualisierung der Motion-Capture Daten in der Simulationsumgebung. Die Avatare sind hier frei wählbar, mit der Einschränkung, dass sie dem im *.bvh* Format definierten Skelett entsprechen.

5 Ausblick

Das vorgestellte Projekt bietet eine Vielzahl möglicher Erweiterungen. So sollte in erster Linie die Umsetzung der Server-Client Funktionalität erfolgen. Die Erweiterung erlaubt bspw. Wizard-of-Oz-Experimente. Diese verfolgen das Ziel, die Kommunikation zwischen Menschen und scheinbar autonomen Systemen zu untersuchen [1]. Mit dieser Methode können Studien wie die Ghost Driver Studie [4] auf virtuelle Umgebung übertragen und kontrollierter analysiert werden.

Des weiteren können die Daten sowohl aus dem Motion-Capturing als auch die Verfolgung der Blickrichtung von Passant und Fahrer ermittelt werden. Das Eye-Tracking ist mit einer Eye-Tracking Erweiterung [11], der in Abschnitt 3.1 vorgestellten HMDs möglich. Eye-Tracking ermöglicht die Blickposition der Benutzer in der virtuellen Umgebung zu berechnen. Diese Daten können bspw. zur Messung der Aufmerksamkeit und Blickrichtung des Fahrers und des Passanten verwendet werden. Der Einsatz des Motion-Capturing-Systems und der damit gewonnenen Daten ermöglicht eine hohe Auswertungsgenauigkeit der Bewegungen aller Beteiligten. Mit Hilfe dieser Daten können Systeme entwickelt werden, die neben einfacher Objekterkennung auch gezielt Bewegungsabläufe zur Detektion und Verfolgung von Verkehrsteilnehmern einsetzen können.

Abbildung 11: Eye-Tracking in VR

Die Simulationsumgebung kann auch dazu verwendet werden, Bildmaterial zu generieren, mit dem autonome Fahr- und Assistenzsysteme trainiert und getestet werden können. Abbildung 12 demonstriert den

Abbildung 12: Objekt Detektion in virtuellen Umgebungen

Einsatz des aufgezeichneten Bildmaterials für das Training von Objektklassifikationsalgorithmen. Es können sowohl Passanten, Fahrzeuge und andere Objekte detektiert werden. Hierbei haben virtuelle Umgebungen den Vorteil, dass Szenerien beliebig modelliert und unter diversen Beleuchtungs- und Wetterbedingungen ausgeführt werden können.

Die vorgestellte Software-Architektur sowie die Umsetzung sollen in weiteren Arbeiten verwendet werden. Hierbei kann das Fahrzeug in der virtuellen Umgebung um weitere Komponenten erweitert werden. Der Schwerpunkt kann dabei auf visuelle Interaktionsmechanismen gelegt werden, wie sie bspw. von Mercedes-Benz im *Mercedes-Benz F 015* Prototypen [10] und Lanström et. al [12] entwickelt wurden. Diese Mechanismen können in Zukunft die Interaktion zwischen autonomen Fahrzeugen und Passanten erleichtern. Das Ziel dieser Interaktionskonzepte ist es, die Verkehrssicherheit für alle Verkehrsteilnehmer trotz der Beteiligung autonomer Fahrzeuge zu gewährleisten.

6 Diskussion

Die Performance moderner Computersysteme, leistungsfähiger Game Engines wie Unity3D und hoch präzise Motion-Capture-Systeme, machen Technologieentwicklungen und empirische Untersuchungen des autonomen Fahrens im urbanen Umfeld erst möglich.

Das vorgestellte Konzept sieht den Einsatz von HMDs wie bswp. die HTC-Vive vor, sodass die beteiligten Testpersonen die virtuelle Testumgebung so realistisch wie möglich wahrnehmen. In der Simulation wird die Kopfbewegung der Person in der virtuellen Umgebung mit dem Tracking der HTC-Vive realisiert und der Avatar mit dem Motion-Tracking bewegt. Problematisch ist hierbei vor allem die Synchronisation der Bewegungen einer Person zwischen dem Motion-Tracking und dem Light-House Tracking der HTC-Vive. Beide Systeme verwenden unterschiedliche Tracking-Verfahren und Einheiten, die sich nicht ohne weiteres verbinden oder ineinander umrechnen lassen. Im Test der Simulation zeigte sich, dass diese Differenzen mitunter dazu führen, dass sich der Körper des Avatars, bei einer Drehung um die eigene Achse, weiter rotiert als die Kopfposition und dem daraus resultierenden Kamerabild der HTC-Vive. Diese Differenzen beeinflussen die Immersivität der virtuellen Umgebung.

Studien zu Immersion in virtuellen Umgebungen könnten dazu beitragen, die Testumgebung weiter zu verbessern. Der Einsatz der HMDs bringt in virtuellen Umgebung meist die Problematik der Motion-Sickness mit sich. Motion-Sickness kann auftreten, wenn der visuelle Input aus der virtuellen Umgebung nicht der eigenen Körperwahrnehmung entspricht [2]. In dem Zusammenhang fiel auch das Fehlen der Beschleunigungskräfte bei der Fahrt im Fahrsimulator in virtuellen Umgebungen auf.

7 Fazit

In dieser Arbeit wurde ein Konzept entwickelt, das für diverse Testszenarien die für Untersuchung der Interaktion zwischen Passanten und autonomen Fahrsystemen ermöglicht. Das in dieser Arbeit entwickelte Konzept verwendet das VICON Motion-Capture System für das Motion-Tracking eines Passanten und eines Fahrers, um Daten zur Untersuchung der Interaktionen zu liefern. Ferner soll ein Fahrsimulator eingesetzt werden, der die Steuerung eines autonomen virtuellen Fahrzeuges oder eines herkömmlichen Fahrzeuges ermöglicht. Die Steuerung autonomer Fahrzeuge kann so auf Basis von Wizard-Of-Oz-Experimenten dazu genutzt werden, diverse Verhaltensweisen von menschlichen Akteuren im Bezug auf autonome Fahrzeuge zu untersuchen.

Auf Basis des Konzepts wurde eine Simulationsumgebung realisiert, in der Interaktionen zwischen Passant und autonomem Fahrzeug untersucht werden können. Die Simulationsumgebung wird künftig in einer Masterarbeit sowie im OPF-Projekt [8] weiterentwickelt.

Literatur

[1] U. C. Bass Leonard J., Gornostaev Juri. Human-computer interaction: Third international conference, ewhci '93, moscow, russia, august 3 - 7, 1993 ; selected papers. Lecture Notes in Computer Science, Berlin u.a., 1993. Springer. DOI:10.1007/3-540-57433-6.

[2] Y. Chen, H. Peng, and J. W. Grizzle. Fast trajectory planning and robust trajectory tracking for pedestrian avoidance. volume 5, pages 9304–9317. DOI:10.1109/ACCESS.2017.2707322.

[3] F. M. Consortium. Fmc - notation reference. URL:http://www.fmc-modeling.org/notation_reference Besucht am 13.02.2018".

[4] Dirk Rothenbücher, Jamy Li, David Sirkin, Brian Mok, Wendy Ju. Ghost driver: A field study investigating the interaction between pedestrians and driverless vehicles: August 26 to august 31, 2016, teachers college, columbia university, new york, u.s.a. DOI:10.1109/ROMAN.2016.7745210.

[5] I. G. Friederike Schneemann. Analyzing driver-pedestrian interaction at crosswalks: A contribution to autonomous driving in urban environments: 19-22 june 2016. DOI:10.1109/IVS.2016.7535361.

[6] Gregor M. R. Johannsen. *AutoXpress*. Medien. Diplomica GmbH, Hamburg, 2002. ISBN: 978-3832472535.

[7] M. Hartmann, M. Viehweger, W. Desmet, M. Stolz, and D. Watzenig. "pedestrian in the loop": An approach using virtual reality. In *ICAT 2017*, pages 1–8, Piscataway, NJ, October 2017. IEEE. DOI:10.1109/ICAT.2017.8171601.

[8] HELLA GmbH & Co. KGaA. Open fusion platform: Offene fusionsplattform, 2016. URL: http://www.ofp-projekt.de/ofp-project/de/index.html Besucht am 14.02.2018.

[9] Logitech. Logitech g29 driving force-lenkrad. URL: https://www.logitechg.com/de-de/product/g29-driving-force Besucht am 13.02.2018.

[10] Mercedes-Benz International. Der mercedes-benz f 015 luxury in motion., 2015. URL: https://tinyurl.com/ya7545wo Besucht am 16.02.2018.

[11] Pupil Labs GmbH. Pupil platform: We develop open source software and build accessible hardware for eye tracking, 2016. URL: https://pupil-labs.com/ Besucht am 16.02.2018.

[12] Tobias Lagström, Victor Malmsten Lundgren. *AVIP - Autonomous vehicles interaction with pedestrians: An investigation of pedestrian-driver communication and development of a vehicle external interface*. Master of science thesis in the master degree program industrial design engineering, CHALMERS UNIVERSITY OF TECHNOLOGY, Gothenborg, 2015.

[13] Unity Technologies. Unity - manual: Network system concepts: Server and host, 2018. URL: https://docs.unity3d.com/Manual/UNetConcepts.html Besucht am 13.02.2018.

[14] VICON. Motion capture for object tracking and robotics. URL: https://www.vicon.com/motion-capture/engineering Besucht am 13.02.2018.