955
2005
006

Fuzzy logic voor het modelleren van autobestuurdersgedrag: een hiërarchische benadering

Arthur Perton
934542
juni 2005

Begeleiders:
Jos A.G. Nijhuis (RuG)
Marcel Wierda (Vesc BV)
Mattijs van Delden (Vesc BV)

Kunstmatige Intelligentie
Rijksuniversiteit Groningen

955

Fuzzy logic voor het modelleren van autobestuurdersgedrag: een hiërarchische benadering

Afstudeeronderzoek

in opdracht van
het Interfacultair Opleidingsinstituut Kunstmatige Intelligentie aan de Rijksuniversiteit Groningen
en
Virtual Environments Systems and Consultancy (VESC) BV

door

Arthur Perton
geboren op 20 februari 1976 te Veendam

Groningen
2005

Samenvatting

Inleiding (hoofdstuk 1)

In de robotica worden voor autonome navigatie met succes hiërarchische fuzzy logic systemen toegepast. Deze technieken zijn slechts in beperkte mate gebruikt voor het modelleren van de autobestuurderstaak. Eén en ander is voor mij de aanleiding geweest om de toepasbaarheid van hiërarchische fuzzy logic technieken voor het modelleren van autobestuurdersgedrag te onderzoeken. De onderzoeksvraag luidt: 'In hoeverre is vage logica geschikt voor het modelleren van het tactische niveau van de autobestuurderstaak?'. De focus ligt op het tactische niveau, omdat daar de beslissingen worden genomen aangaande de directe interactie tussen de bestuurder en de verkeersomgeving.

Literatuuronderzoek (hoofdstuk 2)

Het type model dat in dit onderzoek centraal onderzocht wordt classificeert men in de literatuur als een functioneel cognitief procesmodel of productiesysteem. Tot de rijtaak worden ten minste de volgende onderdelen gerekend: visuele oriëntatie, snelheidshandhaving, recht rijden en bochten nemen, motorische controle bij het bedienen van de instrumenten, navigatie en coördinatie en multitasking. De rijtaak wordt vaak als hiërarchisch opgebouwd gezien, volgens het populaire paradigma van J.A. Michon, waarbij er drie taakniveau's worden onderscheiden, namelijk van 'hoog' naar 'laag' het strategische, tactische en operationele niveau.
In de literatuur worden tenminste vijf verschillende modelleermethoden aangedragen. Met name zijn dit Soar, ACT-R, Hidden Markov Modellen, kwantitatieve expersystemen en fuzzy logic. Elk methode heeft zijn eigen voor- en nadelen, welke in het literatuuronderzoek op rij worden gezet.
Een hiërarchisch fuzzy logic systeem is in feite een (verzameling van) fuzzy controllers. Een fuzzy controller bestaat uit vier componenten, namelijk een kennisbank, een inferentiemechanisme en fuzzificatie- en defuzzificatiemodules. Deze componenten worden in het literatuuronderzoek verder toegelicht. Een hiërarchisch fuzzy logic systeem werkt met meerdere gedragsmodules, waarvan de uitvoer op intelligente wijze wordt samengenomen.
Tot slot van het literatuuronderzoek wordt er een aantal suggesties aan de hand gedaan hoe de verschillende modellen gewaardeerd en vergeleken kunnen worden.

Onderzoeksmethode (hoofdstuk 3)

Als logisch vervolg op het literatuuronderzoek wordt een deel van de bestuurderstaak geïmplementeerd, in de vorm van een hiërarchisch fuzzy gedragsysteem dat een auto in een simulatie aanstuurt. Aan de hand van deze implementatie kan de geschiktheid van deze methode voor het modelleren van de bestuurderstaak verder bepaald worden. Ook de vergelijking met andere methoden wordt zo beter mogelijk gemaakt. Om de omvang van de implementatie enigzins behapbaar te houden, wordt de te implementeren deel van de rijtaak beperkt tot het rijden op de snelweg.
Het programma fuzzyTECH van het bedrijf INFORM speelt een belangrijke rol in de implementatie. De wijze waarop dit programma wordt aangewend wordt in dit hoofdstuk toegelicht.
Van elk van de in het literatuuronderzoek genoemde onderdelen van de rijtaak wordt aangeduid in hoeverre ze geïmplementeerd worden. Verder worden enkele globale richtlijnen voor de wijze van implementatie gedefinieerd.
De simulatie bestaat uit een 3D-omgeving met een snelweg en één of meerdere voertuigen. De positie en oriëntatie van de voertuigen worden real-time doorgerekend aan de hand van een eenvoudig natuurkundig automodel. Het scenario dat zich in de simulatie afspeelt kan vooraf in termen van uitgangspositie en overige randvoorwaarden worden gedefinieerd.
Het vervaardigde model wordt met diverse middelen geëvalueerd op de punten ontwerpproces, implementatie en prestaties. Bij het ontwerpproces wordt er gekeken naar de ontwerpkosten en de mogelijkheid tot een incrementeel ontwerpproces. De implementatie wordt geëvalueerd op de

algemene architectuur, de variabelen en membershipfuncties, de gedragsmodules, de gedragscoördinatie en het bepalen van de rijstijl via de zogenaamde strategische gewichten. Bij de prestaties wordt gekeken naar de systeemrespons (open loop situatie); het bestuurdersgedrag (closed loop situatie); de observatie van het bestuurdersgedrag door experts; de mogelijkheden tot het differentiëren van verschillende rijstijlen en tot slot de rekenlast.

Resultaten (hoofdstuk 4)

Wat betreft de ontwerpkosten het volgende: De meeste tijd zit in het opzetten van de architectuur. De benodigde expertkennis is goedkoop en alom voorhanden, daar iedereen in die in het bezit is van een rijbewijs als expert beschouwd kan worden. Het invoegen van de expertkennis in het systeem is eenvoudig, aangezien de analogie tussen de kennis in natuurlijke taal en die van de productieregels in het systeem sterk is. De productieregels worden namelijk geformuleerd in termen van linguïstische variabelen. Tot slot leent het type systeem zich uitstekend voor een incrementeel ontwerpproces.
Voor de implementatiedetails worden in dit hoofdstuk in woord en beeld beschreven. Voor een schematische weergave van het systeem, alle productieregels en de gebruikte variabelen wordt de lezer verwezen naar bijlage A.
In dit hoofdstuk wordt eveneens de systeemrespons uitvoerig beschreven, als mede het bestuurdersgedrag in termen van metingen en observaties. Aan de hand van de observaties wordt eveneens vastgesteld dat een viertal verschillende rijstijlen, die aangemaakt zijn door de strategische gewichten te variëren, goed onderscheiden kunnen worden. De rekenlast blijkt beperkt te zijn: op een doorsnee PC kunnen tot 5000 FuzzyDrivers tegelijk worden doorgerekend op de minimaal benodigde doorrekenfrequentie van tien Hertz.

Conclusies en aanbevelingen (hoofdstuk 5)

Het autobestuurdersgedrag kan eenvoudig gemodelleerd worden met een hiërarchisch fuzzy logic systeem. De heuristieken van de taak (de expertkennis) kunnen in bijna natuurlijke taal worden ingevoerd. Dit houdt de applicatie bovendien doorzichtig en eenvoudig aan te passen. Het nauwkeurig afstellen van het systeem is pas na het invoeren van de heuristieken met behulp van een simulatie uit te voeren, via een trial-and-error proces. De hiërarchische ordening van de gedragsmodules komt het overzicht ten goede en maakt het hergebruik van lagere-orde modules mogelijk.
Het systeem is geschikt voor een incrementeel ontwikkelproces. Door het modulaire karakter van het systeem kan het gedragsspectrum uitgebreid worden, zonder dat de balans verstoord wordt en het geheel herschreven moet worden.
Het gebruik van fuzzy logic draagt bij aan natuurlijk ogend gedrag. Het voorkómen van oscillaties en plotselinge, bruuske bewegingen is inherent aan het gebruik van deze vage logica.
Partijen die gebruik willen maken van een hiërarchisch fuzzy systeem voor het simuleren van autobestuurdersgedrag dienen zich eerst af te vragen wat het doel is van de simulatie. Dit doel zal bepaalde eisen stellen aan de simulatie. In dit hoofdstuk wordt beschreven in hoeverre FuzzyDriver aan zulke eisen voldoet. Voorts worden de benodigde middelen beschreven en tot slot de stappen van het ontwikkeltraject.

Suggesties voor verder onderzoek (hoofdstuk 6)

Het verslag wordt afgesloten met enkele suggesties voor verder onderzoek, die uitbreidingen en aanpassingen aan het systeem behelsen, alsmede een empirische vergelijking met de andere mogelijke modelleermethoden die genoemd zijn in het literatuuronderzoek.

Voorwoord

Dit is het verslag van mijn afstudeeronderzoek voor de studie Kunstmatige Intelligentie. Eén van de onderzoeksgebieden van Kunstmatige Intelligentie is autonome systemen. Hierbij wordt gekeken naar geschikte methoden om zelfstandige agents te ontwikkelen die in een onbekende omgeving een taak kunnen uitvoeren en adequaat op onverwachte gebeurtenissen in die omgeving kunnen reageren. Een uitstekend voorbeeld van zo'n agent is een kunstmatige autobestuurder.

Nu is in het Drentse plaatsje Annen een bedrijf gevestigd, genaamd Vesc, dat zich bezighoudt met 3D-visualisatie en -simulatie. Het onderwerp van de kunstmatig intelligente weggebruiker is hier al vaker aan de orde gekomen. In zowel producten voor de zakelijke markt, zoals een rijsimulator, als PC (race)games is een belangrijke rol weggelegd voor 'intelligent verkeer'. Na de kennismaking tussen het bedrijf en mij – in verband met een broodnodige aanvulling op de studiebeurs heb ik een aantal web-applicaties voor het bedrijf gemaakt – besloten we aan onze gezamenlijke interesses en belangen tegemoet te komen: ik heb bij dit bedrijf mijn afstudeer-onderzoek gedaan naar het ontwikkelen van een kunstmatige autobestuurder met behulp van fuzzy-logictechnieken uit de robotica, die ik in mijn studie ben tegengekomen.
In dit rapport doe ik verslag van dit onderzoek. Daarbij ga er vanuit dat de lezers, in de eerste instantie de beoordelaars van mijn werk, enige voorkennis hebben op het gebied van kunstmatige intelligentie.

Tijdens mijn afstudeeronderzoek en mijn werkzaamheden bij Vesc heb ik een groot enthousiasme gekregen voor de 3D-visualisatie en -simulatie en het ontwikkelen van systemen voor autonome voertuigbesturing in het bijzonder. Zo werkte ik voor Vesc en haar dochter Bumblebeast aan een padensysteem met autonome voertuigen en voetgangers voor het simuleren van stadsverkeer, alsmede een systeem voor de autonome besturing van racewagens, voor gebruik in de video games die daar in ontwikkeling waren.

Tot slot van dit voorwoord wil ik een aantal mensen bedanken: Mijn dank voor de goede begeleiding gaat uit naar mijn mentor Jos Nijhuis, verbonden aan de universiteit. Verder dank ik Vesc, voor het beschikbaar stellen van haar visualisatie- en simulatiekernel 4Space, en met name Mattijs van Delden voor het mij bijbrengen van de fijne kneepjes van het programmeren in Visual C++; Marcel Wierda (oud-onderzoeksmedewerker aan het Verkeerskundig Studiecentrum van de Rijksuniversiteit Groningen) voor de zeer bruikbare adviezen en literatuur voor het schrijven van dit onderzoeksverslag; Rik Teerling voor het modelleren van de gesimuleerde taakomgeving voor 'FuzzyDriver' en Patrick Schoenmaker voor het maken van de tekening die nu de omslag van dit verslag siert. Tot slot wil ik mijn vader en moeder bedanken voor hun onvoorwaardelijke steun en mijn broer Emile voor het redigeren van het stuk en voor de vele praktische tips voor het schrijven.
Bedankt!

Arthur Perton
Groningen, april 2005

Inhoudsopgave

Samenvatting ..... **iv**
Voorwoord ..... **vi**
Inhoudsopgave ..... **vii**
1 Inleiding **1**
1.1 Motivatie 1
1.2 Doelstelling 1
1.3 Onderzoeksvraag 1
1.3.1 Autobestuurderstaak 1
1.3.2 Tactisch niveau 2
1.3.3 Vage logica 2
1.3.4 Geschiktheid meten 2
1.4 Opbouw van het rapport 3
2 Literatuuronderzoek **4**
2.1 Modellen van bestuurdersgedrag 4
2.1.1 Typen modellen 4
2.1.2 De rijtaak in het algemeen 6
2.1.3 De rijtaak als hiërarchische taak 6
2.1.4 Bekende architecturen voor het modelleren van bestuurdersgedrag 7
2.2 Fuzzy controllers (FCs) 10
2.2.1 Knowledge base 11
2.2.2 Fuzzificatie 12
2.2.3 Inferentie 12
2.2.4 Defuzzificatie 14
2.3 Hiërarchische vage gedragsystemen 16
2.3.1 Systemen gebaseerd op gedragsmodules 16
2.3.2 Coördinatie van de gedragsmodules: arbitrage en combinatie 16
2.3.3 Conflict resolutie 17
2.4 Waarderen en vergelijken van modellen 18
3 Onderzoeksmethode **19**
3.1 Inleiding 19
3.1.1 Motivatie voor de implementatie 19
3.1.2 Resumptie van de literatuurstudie 19
3.1.3 Afbakening van de te implementeren taak 19
3.2 Gebruik van het programma fuzzyTECH 19
3.2.1 Inleiding 20
3.2.2 FuzzyTECH als server 20
3.2.3 Gegenereerde code 21
3.3 Te implementeren onderdelen van de rijtaak 21
3.3.1 Inleiding 21
3.3.2 Visuele oriëntatie 21
3.3.3 Snelheidshandhaving 22
3.3.4 Recht rijden en bochten nemen 22
3.3.5 Rijstrook kiezen 22
3.3.6 Motorische controle bij het bedienen van instrumenten 23
3.3.7 Navigatie 23
3.3.8 Coördinatie en multitasking 23
3.4 Richtlijnen voor het gedragsysteem 23

3.4.1 Inleiding 23
3.4.2 Gedragsmodules 23
3.4.3 Sequentieel gedrag 24
3.5 Simulatie van voertuig en taakomgeving 25
3.5.1 Inleiding 25
3.5.2 Natuurkundig automodel 25
3.5.3 Toestand & Perceptie 26
3.5.4 Scenario's 27
3.6 Evaluatie 28
3.6.1 Inleiding 28
3.6.2 Ontwerpproces 28
3.6.3 Implementatiedetails 29
3.6.4 Prestaties 32
4 Resultaten 36
4.1 Inleiding 36
4.2 Ontwerpproces 36
4.2.1 Inleiding 36
4.2.2 Ontwerpkosten 36
4.2.3 Incrementeel ontwerpproces 37
4.3 Implementatie 37
4.3.1 Algemene architectuur 37
4.3.2 Variabelen en membership functies 38
4.3.3 Primitieve gedragsmodules 38
4.3.4 Gedragscoördinatie 39
4.3.5 Strategische gewichten en rijstijl 41
4.4 Prestaties 42
4.4.1 Inleiding 42
4.4.2 Beschrijving van de systeemrespons (open loop) 42
4.4.3 Beschrijving van het bestuurdersgedrag (closed loop) 48
4.4.4 Observaties van de experts 66
4.4.5 Differentiatie van rijstijlen 67
4.4.6 Rekenlast 67
5 Conclusies en aanbevelingen 69
5.1 Conclusies 69
5.1.1 Samenvatting 69
5.1.2 Ontwerpproces 69
5.1.3 Implementatie 69
5.1.4 Prestaties 70
5.2 Aanbevelingen 70
5.2.1 Doel van de simulatie 70
5.2.2 Beschikbare middelen 71
5.2.3 Ontwikkeltraject 71
6 Suggesties voor verder onderzoek 73
6.1 Inleiding 73
6.2 Uitbreiding van het systeem 73
6.2.1 Inleiding 73
6.2.2 Uitbreiding van het mogelijk gedrag 73
6.2.3 Integratie van perceptie in het hiërarchische fuzzy logic systeem 74
6.3 Aanpassingen aan het systeem 74
6.3.1 Inleiding 74
6.3.2 Voorstellen voor aanpassingen 74
6.4 Vergelijking met andere modelleermethoden 75

Literatuurlijst **76**
Bijlage A – Beschrijving van het fuzzy systeem **78**

1 Inleiding

1.1 Motivatie

Sinds de publicatie van Lotfi Zadeh's 'Fuzzy sets' in 1965 [20] is fuzzy logic onderwerp van wetenschappelijk onderzoek. Fuzzy logic betekent in goed Nederlandse 'vage logica'. De term 'vaag' houdt overigens niet in dat deze techniek niet serieus genomen moet worden: de vage logica werd met name in het laatste decennium van de vorige eeuw al op grote schaal toegepast in consumentenproducten. Dit heeft te maken met de succesvolle toepassing van vage logica in regelsystemen, waardoor het ook uitstekend geschikt is voor industriële toepassingen.
In ditzelfde decennium begonnen ook bewegingen vanuit de robotica om de toepassing van fuzzy logic boven het niveau van eerste-orde regelsystemen uit te tillen. Fuzzy logic bleek ook succesvol gebruikt te kunnen worden voor het coördineren van primitieve gedragsmodules, 'behaviors' genaamd, waardoor een hiërarchisch systeem ontstaat dat mobiele robots tot complex gedrag in staat stelt. Planmatig en reactief handelen kunnen hierbij hand in hand gaan [12].
In de literatuur blijkt niet – of slechts in beperkte mate [5] – dat deze technieken ook geprobeerd zijn bij het modelleren en simuleren van autobestuurders. Wel zijn er vele andere benaderingen geprobeerd (zie paragraaf 2.1.4).
De hiërarchische fuzzy logic technieken hebben echter wel potentieel. Het is niet moeilijk de overeenkomsten te zien tussen de taak van een mobiele robot en de autobestuurderstaak. En hiërarchische fuzzy systemen zijn in mobiele robots succesvol toegepast. Redenen genoeg om de toepasbaarheid van fuzzy logic in kunstmatige autobestuurders aan een onderzoek te onderwerpen.

1.2 Doelstelling

Het doel van dit onderzoek is om de praktische waarde van fuzzy logic te bepalen bij het maken van een kunstmatige autobestuurder. Deze waarde kunnen we bepalen door het gedrag van een autobestuurder (deels) te simuleren met een hiërarchisch fuzzy gedragssysteem en het ontwerpproces en het eindresultaat te beoordelen (zie paragraaf 1.3.4).

1.3 Onderzoeksvraag

De onderzoeksvraag luidt:

> *In hoeverre is vage logica geschikt voor het modelleren van het tactische niveau van de autobestuurderstaak?*

In de volgende drie paragrafen worden de termen uit deze vraag verder toegelicht.

1.3.1 Autobestuurderstaak

Een automobilist moet voortdurend beslissingen maken aan de hand van de verkeerssituatie. Hierbij wordt hij geleid door een aantal doelen dat hij nastreeft, zoals het (snel) bereiken van het reisdoel, het volgen van de verkeersregels en het voorkomen van schade en ongelukken. In dit onderzoek wordt de autobestuurderstaak in enge zin opgevat als de beslissingen die een bestuurder van een auto tijdens het manoeuvreren op de snelweg moet nemen in het algemeen en in een aantal nauwkeurig beschreven scenario's in het bijzonder.

1.3.2 Tactisch niveau

De verkeerstaak, dus ook de taak van een automobilist, wordt vaak om pragmatische redenen opgesplitst in drie taakniveaus [8]. Op het *operationele niveau* worden de mentale processen geplaatst die direct verantwoordelijk zijn voor de voertuigbeheersing in enge zin, zoals gas geven, sturen en remmen. Op het *tactische niveau* bevinden zich de processen voor de interactie tussen de deelnemer en de directe verkeersomgeving, dat wil zegen de infrastructuur en andere weggebruikers. Op dit niveau wordt bijvoorbeeld besloten in te halen of een afslag te nemen. Op het *strategische* niveau wordt bepaald hoe van A naar B te komen. Hier worden keuzes gemaakt over de route, het vervoermiddel en tijd.

Het tactische niveau vormt het hart van deze bestuurderstaak. Operationele processen hebben meer het karakter van 'simpele' eerste orde regelsystemen, terwijl op het strategische niveau de randvoorwaarden worden gezet voor de taak, die vaak onveranderd blijven.

1.3.3 Vage logica

Voor het beschrijven van gedrag worden productieregels gebruikt, die gebruik maken van logica. Een voorbeeld van zo'n regel is: *als* het regent *dan* neem ik een paraplu mee. Of, om bij het onderwerp te blijven: *als* ik te dicht achter iemand rijd *dan* moet ik afremmen.
In de klassieke logica kan een bewering slechts waar of onwaar kan zijn, daartussen is er niks. Voor het beschrijven van menselijk gedrag is dit problematisch. Zo wordt de uitspraak 'ik rijd te dicht achter iemand' niet plotsklaps waar als de afstand tussen mij en mijn voorganger onder een bepaalde arbitraire grens – bijvoorbeeld 17,4 meter – komt, waarop ik vol de rem in moet trappen.
Vage logica is de logica van de zachte grenzen. Met vage logica wordt de relatie tussen een getalsmaat (zoals de afstand in meters) en de waarheid van een bewering beschreven met een functie, die de membership functie wordt genoemd. Hierdoor kan een uitspraak als 'ik rijd te dicht achter iemand' *in zekere mate* waar zijn, en een consequentie als 'ik moet remmen' *in gerelateerde mate* worden uitgevoerd.
Het redeneren met vage logica weerspiegelt in veel grotere mate het menselijk redeneren dan de klassieke logica. Voor een verdere uitwijding wordt voorlopig verwezen naar werken die gewijd zijn aan de vage logica [20] en regelsystemen met vage logica [11].

1.3.4 Geschiktheid meten

De vraag in hoeverre vage logica voor het gestelde doel geschikt is werpt meteen een volgende vraag op: hoe kan deze geschiktheid gemeten worden? Het antwoord valt in twee delen uiteen. Enerzijds moeten we kijken naar het ontwerp- en implementatieproces en anderzijds naar het resultaat, dit is de simulatie van een autobestuurder (zie figuur 1.1).
Het ontwerpproces moet bij voorkeur lage kosten met zich meebrengen. De term 'kosten' dient hier breed opgevat te worden als de benodigde investeringen in tijd, kennis en geld. Een proces wordt duur als:

- er veel expertkennis nodig is;
- het systeem uitvoerig getraind moet worden;
- een uitbreiding betekend dat het systeem deels of geheel opnieuw gemaakt moet worden.

Uiteraard moet de toepassing van vage logica ook tot het gewenste eindresultaat leiden. Maar wat maakt een simulatie van een autobestuurder 'goed'? Allereerst moet de bestuurder zijn taak goed kunnen uitvoeren, dus er moet veilig en efficiënt gereden kunnen worden. Vervolgens moet het

gedrag voor de toeschouwer ook natuurlijk ogen. Deze karakteristieken zijn niet direct meetbaar of getalsmatig uit te drukken. Een panel van experts (in dit geval automobilisten) kan hier echter wel een oordeel over vellen. Tot slot moet een goed systeem met eenvoudige aanpassingen verschillende rijstijlen kunnen produceren, zoals agressief of juist offensief, roekeloos of juist veilig.

	Wat meten	Hoe meten
Proces	kosten van het proces	(hoeveel) expertkennis is nodig? (hoeveel) training is nodig? incrementeel ontwerpproces mogelijk?
Resultaat	natuurlijk gedrag/ goede taakuitvoering	gedrag wordt beoordeeld door panel van experts
	wordt er veilig en efficiënt gereden?	gedrag wordt beoordeeld door panel van experts
	is het gedrag doorzichtig?	simulatie stopzetten en proberen de besluitvorming te achterhalen (trace)
	kunnen met het model eenvoudig verschillende rijstijlen worden gegenereerd?	gedrag wordt beoordeeld door panel van experts

Figuur 1.1 – Methoden voor het meten van de geschiktheid van een modelleermethode
Bij het ontwikkelen van een systeem volgens een bepaalde modelleermethode kunnen zowel het proces als het resultaat bekeken worden, tenelnde de geschiktheid van de methode te bepalen. Deze tabel geeft aan wat er gemeten kan worden en hoe.

1.4 Opbouw van het rapport

Hoofdstuk 2 bevat het verslag van het literatuuronderzoek. Hierin wordt eerst, in paragraaf 2.1, beschreven wat voor modellen voor autobestuurders er al zijn en welke juist in dit onderzoek gebruikt wordt. Vervolgens wordt de rijtaak in het algemeen beschouwd. De bestuurderstaak is op te vatten als een hiërarchisch gedragsysteem. Dit wordt ondersteund door onderzoek van Michon [8] die laat zien dat de rijtaak een drie-lagige hiërarchische structuur heeft. Voorts worden de ervaringen met verschillende, reeds bestaande modellen en architecturen beschreven die gebruikt zijn om autobestuurdersgedrag te simuleren (de concurrentie van de fuzzy aanpak). In paragraaf 2.2 worden de hiërarchische fuzzy gedragssystemen beschreven zoals die in de robotica worden gebruikt. Nadruk wordt gelegd op de verschillen in uitvoering van de verschillende systemen, met name a) op het gebied van de keuze uit en/of het samennemen van uitvoer van deelgedragingen (behaviors) en b) op het gebied van prioriteren van doelen en het modelleren van doelen die elkaar opvolgen. Als laatste in hoofdstuk 2 een kort stuk theorie over het waarderen en vergelijken van gedragsmodellen.
In hoofdstuk 3 wordt de onderzoeksmethode uiteengezet. Hierin wordt de eisen aan het systeem beschreven, alsmede de middelen en methoden om het systeem te implementeren. Tot slot wordt de evaluatiemethode uiteengezet. De testresultaten worden in hoofdstuk 4 samengevat.
Het verslag wordt afgesloten met de conclusies en aanbevelingen in hoofdstuk 5 en een aanzet tot discussie en verder onderzoek in hoofdstuk 6.
Na het verslag volgt een genummerde lijst met gebruikte literatuur. Naar deze literatuur wordt in de tekst verwezen met het corresponderende nummer tussen vierkante haken.

2 Literatuuronderzoek

2.1 Modellen van bestuurdersgedrag

2.1.1 Typen modellen

Een taak is te beschrijven met verschillende soorten modellen. In figuur 2.1 zien we een indeling van typen modellen in de verkeerskunde volgens het tweedimensionale stelsel van Michon [8]. Hierbij wordt er enerzijds onderscheid gemaakt tussen taxonomische en functionele modellen, en anderzijds tussen gedragsmatige en cognitieve modellen. Voor het eerste onderscheid wordt gekeken of de componenten van het model dynamische interacties vertonen. Als dit niet het geval is spreekt men van taxonomische modellen en anders van functionele modellen. Het laatste onderscheid, dat tussen cognitieve modellen en gedragsmatige modellen, hangt vooral samen met de vraag of in het model al dan niet aannamen worden gemaakt over mentale processen die zich in het brein van de weggebruiker afspelen. Als dit niet het geval is spreekt men van gedragsmatige modellen en anders van cognitieve modellen. In de komende alinea's noem ik de typen modellen die in deze classificatie zijn opgenomen, en geef verdere toelichting bij de modellen die dat behoeven.

	Taxonomisch	Functioneel
Gedragsmatige modellen	taakanalyses	mechanistische modellen adaptieve regelmodellen - regelkringen - stroomdiagrammen
Cognitieve modellen	factormodellen	risicoperceptiemodellen cognitieve procesmodellen - scenario's - productiesystemen

Figuur 2.1 – Classificatie van modellen in de verkeerskunde (naar Michon [8])
Voor het classificeren worden twee dimensies gehanteerd. Een model wordt gekenmerkt als taxonomisch of functioneel en gedragsmatig of cognitief.

Taxonomische modellen
Een taxonomisch model is een soort opsomming, een feitenbestand. Verschillende typen relaties tussen die feiten worden beschreven in termen van boven-, neven- en onderschikking, identiteit, volgorde, frequentie, waarschijnlijkheid en afstand. Taxonomische modellen zijn niet primitiever of minder bruikbaar dan andere typen modellen, maar ze hebben wel een aantal fundamentele beperkingen. Met een taxonomisch model kan men niet goed dynamische, tijdafhankelijke relaties uitdrukken. Bovendien zijn de verbanden die met behulp van empirisch onderzoek kan vaststellen hoogstens correlatief; dat wil zeggen, men kan vaststellen dat er een verband is, maar niet wat de oorzaak is en wat gevolg.

- *taakanalyses (taxonomisch, gedragsmatig)*
  Een mooi voorbeeld van een taxonomisch model is de taakanalyse van McKnight en Adams [7]. Hierin wordt een uitputtende inventarisatie gegeven van alle deeltaken waaruit de rijtaak is opgebouwd, dat wil zeggen 45 subtaken en meer dan 1700 elementaire deeltaken totaal.

- *factormodellen (taxonomisch, cognitief)*
  Er is een groot aantal vaardigheden in kaart gebracht met behulp van factormodellen, waaronder ook het autorijden. De gedachte die aan dit model ten grondslag ligt is dat alle perceptieve, cognitieve en motorische vaardigheden zijn opgebouwd uit gewogen combinaties van een klein aantal elementaire componenten – vermogens of trekken ('traits') genoemd – zoals reactiesnelheid, ruimtelijk oriëntatievermogen of stressgevoeligheid. Als gevolg van leren of van natuurlijke ontwikkeling kunnen deze combinaties veranderingen ondergaan. De theoretische betekenis van een factormodel is gering: ze leren ons niet hoe de informatieverwerking tijdens het besturen van een auto in zijn werk gaat, maar alleen in welke mate bepaalde vermogens bijdragen aan geobserveerd gedrag.

Functionele modellen
In de wetenschap van kunstmatige intelligentie zijn vooral die modellen interessant die gedrag en/of cognitie beschrijven in termen van informatiestromen. Dit is ook het type model dat in dit onderzoek centraal staat. Deze modellen worden functionele modellen genoemd.

- *mechanistische modellen*
  Het uitgangspunt voor deze mechanistische modellen is de analogie met fysische processen, waarvoor men een mathematische vergelijking kan opstellen. Verkeerstromen worden bijvoorbeeld vergeleken met stroperige vloeistoffen. Bij deze modellen wordt over het algemeen niet beoogd het individuele gedrag van de automobilist te beschrijven.

- *adaptieve regelmodellen – regelkringen*
  De rijtaak kan opgevat worden als een continue of 'intermitterende' regeltaak. Vanuit beide gezichtspunten kan men modellen ontwikkelen voor de basisvaardigheden, zoals het handhaven van koers en snelheid op een rechte of gebogen weg of het ontwijken van obstakels op de weg. Zulke regelmodellen maken gebruik van invoer zoals de zijdelingse positie van het voertuig in de rijstrook (compensatiegedrag, bijsturen), de kromming van de weg of de afstand tot de voorganger (volggedrag). Overbrengingsverhoudingen die de dynamische eigenschappen van zowel het voertuig als de bestuurder representeren vormen de componenten van dergelijke modellen. Zij omvatten parameters voor de tijdconstanten in het systeem waarmee ondermeer anticipaties en reactiesnelheden van een bestuurder gekwantificeerd worden.

- *cognitieve procesmodellen – productiesystemen*
  Een productiesysteem is in zijn eenvoudigste vorm een stelsel van ALS-DAN regels ofwel productieregels. Een voorbeeld van zo'n productieregel:

  ALS het verkeerslicht rood is
  DAN laat gas los *en*
  druk het rempedaal in *en*
  stop voor kruispunt

  De gedachte die aan een dergelijke regel ten grondslag ligt, is dat de interne informatie-verwerkingsprocessen in het menselijk brein zodanig gestructureerd zijn dat voor de toeschouwer lijkt alsof deze regel expliciet in het brein is verankerd. Zo'n regel als in het bovenstaande voorbeeld moeten we opvatten als het product, of een externe beschrijving, van een proces dat zich afspeelt in een 'black box'. De regels die binnen die 'black box' gelden hoeven niet dezelfde te zijn als die we observeren.

Het model dat vervaardigd is voor dit onderzoek valt in deze laatste categorie, waarbij de componenten in de onderste (controle)laag op te vatten als regelkringen.

2.1.2 De rijtaak in het algemeen

Alvorens in te gaan op de hiërarchische structuur van de rijtaak, is het goed om kort stil te staan bij wat er in algemene zin bij de taak komt kijken. Aasman [1] geeft een globale taxonomie van de rijtaak weer, waarbij hij zes belangrijkste deeltaken onderscheidt:

Visuele oriëntatie. Een automobilist heeft de continue taak om op het juiste moment naar de juiste plek kijken, om de voor de verkeerstaak relevante objecten en gebeurtenissen in de omgeving waar te nemen. Voor een inspectie van alles wat zich in de omgeving bevindt en afspeelt is in de regel geen tijd; er moeten beslissingen genomen worden.

Snelheidshandhaving. De bestuurder moet ook steeds in kritische situaties de snelheid aanpassen aan de taakomgeving, bijvoorbeeld bij het naderen van een kruispunt of bij het volgen van iemand op de snelweg. Deze taak (snelheidshandhaving) is sterk gerelateerd aan de visuele oriëntatie- en de motorische controletaak.

Recht rijden en bochten nemen. Ook moet de bestuurder steeds zorgen dat hij het voertuig binnen de rijstrook houdt. Bij bochten moet bepaald worden wanneer de bocht wordt ingezet en welke stuurhoek vereist is. Deze taak is eveneens sterk gerelateerd aan de visuele oriëntatie- en de motorische controletaak.

Motorische controle bij het bedienen van instrumenten. Om te sturen of van snelheid te veranderen, moeten de instrumenten worden bediend. Gas geven, remmen, sturen, schakelen en dergelijke zijn voor de geoefende rijder geheel geautomatiseerde taken.

Navigatie. Vervolgens is er het probleem van bepalen waar men heen moet bij een afslag of kruispunt. Dit vergt een stuk planning vooraf, maar ook aanpassen aan de omgeving in het geval van afzettingen, omleidingen, files, etcetera.

Coördinatie en multitasking. Als laatste moet de juiste actie op het juiste moment, en soms meerdere tegelijk, worden uitgevoerd.

2.1.3 De rijtaak als hiërarchische taak

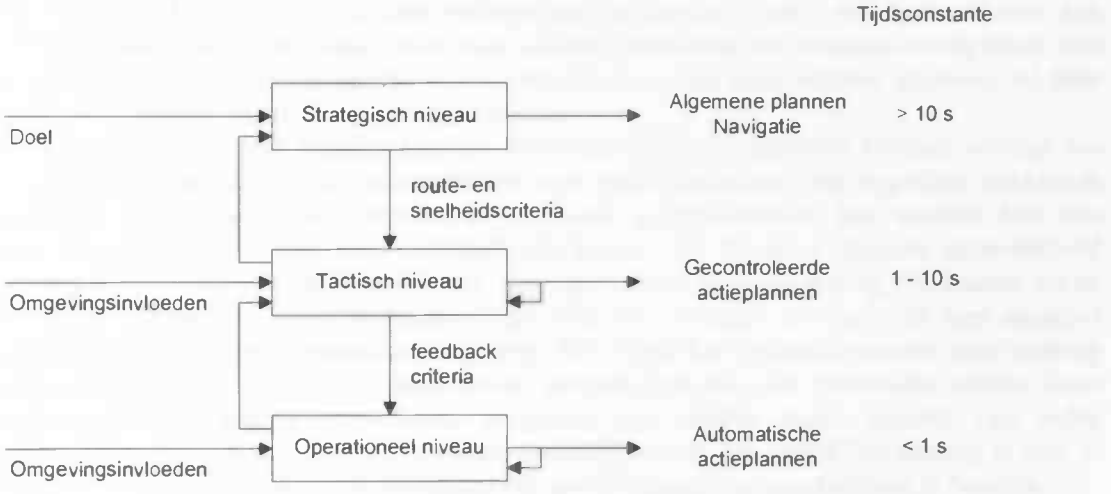
De verkeerstaak wordt vaak om pragmatische redenen opgesplitst in drie taakniveaus. Deze niveaus en hun onderlinge samenhang zijn weergegeven in figuur 2.2. Op het *operationele niveau* worden de mentale processen geplaatst die direct verantwoordelijk zijn voor de voertuigbeheersing in enge zin. Men noemt dit ook wel het controle niveau. De processen op dit niveau duren zo'n 10 tot 500 milliseconden. Op het *tactische niveau* bevinden zich de processen voor de interactie tussen de deelnemer en de directe verkeersomgeving, dat wil zegen de infrastructuur en andere weggebruikers. Aan dit niveau wordt ook wel gerefereerd met de term 'manoeuvreniveau'. Deze processen duren 500 milliseconden tot enkele tientallen seconden. Op het *strategische* niveau wordt bepaald hoe van A naar B te komen. Hier worden keuzes gemaakt over de route, het vervoermiddel en tijd [19].

Het verdient enige nadruk dat deze opdeling niet per se betekent dat de onderverdeelde processen in hun functionele karakter of lokalisatie in het brein verschillen. Deze driedeling beschrijft echter wel de typische structuur van allerlei taken (en in het bijzonder de rijtaak) in de zin van de organisatie van beslissingen en informatiestromen. In deze zin is de indeling juist erg rigide.

De driedeling is hiërarchisch. In een hiërarchisch model worden lagere orde niveaus beheerst door hogere orde doelen [4]. Als bijvoorbeeld op het tactische niveau het subdoel 'inhalen' actief is, dan

pas worden de geassocieerde gedragsmodules op het operationele niveau (in de spiegels kijken, richting aangeven, gas geven en naar de linkerbaan gaan) geactiveerd. En het reisdoel, zoals bepaald op het strategisch niveau, zorgt voor doelen en gedrag op het tactische niveau (de volgende afslag nemen).

De uitkomst op een niveau bepaalt de *probleemruimte* voor het onderliggende niveau [19]. Het gedrag op dat onderliggende niveau wordt dus niet in deterministische zin gedicteerd [4]. Hogere orde doelen specificeren slechts een globaal plan of strategie in plaats van specifiek gedrag; de werkelijke acties hangen in grote mate af van de omstandigheden waarin uiteindelijk geopereerd wordt. De doelen moeten eigenlijk gelieerd zijn (of: worden) aan de taakomstandigheden. Dit toont de noodzaak aan voor bottom-up controle of feedback.



Figuur 2.2 – De hiërarchische structuur van de rijtaak [9]
De rijtaak is opgedeeld in drie taakniveaus, namelijk het strategische, tactische en operationele niveau. Het strategisch niveau staat onder invloed van de gestelde doelen. Op het tactische en operationele niveau worden de processen bepaald door omgevingsinvloeden en criteria gesteld door het bovenliggende niveau. Elk niveau creëert een eigen categorie plannen. Des te hoger het niveau, des te langer duren de processen die met dat niveau geassocieerd worden.

2.1.4 Bekende architecturen voor het modelleren van bestuurdersgedrag

Het modelleren van bestuurdersgedrag is een populaire bezigheid in de verkeerskunde. In dit hoofdstuk worden een aantal beproefde architecturen en methoden genoemd, alsmede de geclaimde resultaten. Eén en ander is getabelleerd in figuur 2.3.

Soar
Soar, een bekend model voor menselijk probleemoplossen, is door Aasman [1] gebruikt om bestuurdersgedrag te modelleren. Dit leverde een aantal interessante bevindingen op. Hij laat zien dat het mogelijk is Soar te gebruiken voor een redelijk functioneel model van rijgedrag, waarbij de belangrijkste subtaken van de rijtaak vertaald kunnen worden naar Soar. Eén (technische) tekortkoming was dat Soar computationeel zeer inefficiënt is. Andere, meer fundamentele problemen waren: het werken met kwantitatieve gegevens; de afwezigheid van een timing mechanisme; het onvermogen om dingen te vergeten en om slechte dingen af te leren. Tot slot laat de gebruikersvriendelijkheid van Soar volgens Aasman nogal te wensen over.

ACT-R

De kunstmatige architectuur voor cognitie ACT-R werd door Salvucci e.a. [14] gebruikt om een autobestuurdersmodel te maken. Hierin werden verschillende aspecten van bestuurdersgedrag vertegenwoordigd waaronder het sturen en oogbewegingen. Het model stelt de continue taakuitvoering telkens na een bepaald tijdsinterval bij. Het houdt actief de omgeving in de gaten, en maakt besluiten aan de hand van het zo gevormde 'mentale model van de omgeving' van dat moment. Hiervoor wordt een hiërarchische structuur van doelen gebruikt, alsmede verschillende gedragslagen met hoger-niveau gedrag (bijvoorbeeld besluitvorming) die de initiatie en het management verzorgen van lagere-niveau gedrag (controle). Het model produceert gedrag dat in kwalitatieve en kwantitatieve zin goed overeenkomt met menselijk gedrag, zoals dat verzameld is in een studie met een simulator. De eventuele nadelen van een dergelijke implementatie worden in het artikel niet vermeld.

Hidden Markov Modellen

Een Hidden Markov Model (HMM) is een verzameling toestanden waarin een systeem zich kan bevinden. De toestanden zijn met elkaar en/of met zichzelf verbonden via toestandsovergangen met een bepaalde kans. In een aantal toestanden wordt, met een bepaalde kans, uitvoer gegeven. Al deze kansen kunnen door middel van training worden bepaald.

Bayarri e.a. [2] beschrijven hoe Hidden Markov Modellen (HMMs) gebruikt kunnen worden om bestuurdersgedrag te modelleren. Ze redeneren dat een aantal aspecten van rijgedrag voldoende stereotypisch zijn om het mogelijk te maken dat een actie gerepresenteerd kan worden door een omschrijving van de toestand, met een voorspellende kracht. De rij-acties kunnen gemodelleerd worden als een reeks mentale toestanden, elk met zijn eigen karakteristieke gedrag. De kansen op de overgangen tussen de verschillende toestanden zijn bepaald, alsmede de kans dat een toestand resulteert in het met die toestand geassocieerde gedrag. Het model kan gemaakt worden door training aan de hand van statistische empirische data, zonder tussenkomst van een menselijke expert. Deze methode zou natuurlijk gedrag garanderen, aangezien het model heeft geleerd van echte bestuurderacties. Een probleem vormt het kostbare trainingsproces. De HMM-benadering is nog in ontwikkeling en er zijn nog onvoldoende resultaten om goed de mate van geschiktheid te bepalen.

Kwantitatieve expertsystemen

Dit is een klassieke benadering voor het besturen van voortuigen in een simulatie of spel. Elk voertuig heeft een set parameters, die de huidige toestand definieert. Ze hebben ook elk een stel vooraf bepaalde doelen, die worden bereikt gedurende de route door een wegennetwerk. De toestand van het voertuig en de doelen samen met de perceptie van de externe verkeersparameters, bijvoorbeeld afstand tot een voertuig, toestand van de verkeerslichten en de obstakels op de rijstrook zijn de basis voor het evalueren van de regels in het systeem. De basis-subset van regels die gewoonlijk worden gerepresenteerd in een implementatie van zo'n expertsysteem zijn 'auto volgen', 'rijstrook kiezen' en 'kruispunt conflict resolutie'. De drie subsets zijn nogal aan elkaar gerelateerd omdat sommige delen van de voortgang van het voertuig meestal meerdere van die subsystemen betrekken. Daarom is er een inferentiesysteem nodig dat de regels toebehorende aan het subsysteem coördineert en dat eventuele conflicten oplost tussen de beslissingen die door de subsystemen gemaakt worden. Die beslissingen die geïmplementeerd zijn in de set van regels bevatten wiskundige en kinematische kwantitatieve modelisaties van de voortgang van het voertuig. Als voorbeeld is een dergelijke formule uit [2] voor het bepalen van de juiste snelheid voor het behouden van een veilige volgafstand hieronder opgenomen:

2.1) $v_n(t+\tau) = \min(\alpha, \beta)$

met: $\alpha = v_n(t) + 2.5a_n\tau\left(1 - v_n(t)V_n\right)^{1/2}$

$\beta = b_n\tau + \left(b_n{}^2\tau^2 - b_n\left(2\left(x_{n-1}(t) - s_{n-1} - x_n(t)\right) - v_n(t)\tau - v_{n-1}(t)^2 / b^{\wedge}\right)\right)^{1/2}$

De snelheid voor het behouden van een veilige volgafstand in het volgende frame $v_n(t+\tau)$ wordt berekend uit onder andere de tijd tussen twee frames τ; de tijd t; de snelheid $v_n(t)$; de positie van het voortuig $x_n(t)$ en de positie van het voertuig ervoor $x_{n-1}(t)$, de maximum rem b_n en de geschatte rem van het voertuig ervoor $\hat{b}$.

Fuzzy logic
Fuzzy logic (vage logica) heeft vooral haar nut bewezen in regelsystemen. De resultaten laten zich kenmerken door een adequaat, glad, regelmatig en oscillatievrij verloop. De rijtaak is echter niet op te vatten als een simpel regelsysteem [8]. Wel zijn er succesvolle resultaten uit de robotica waarbij robots die zich moeten handhaven in een kantoor- of fabrieksomgeving uitgerust werden met een hiërarchische gedragsgebaseerde fuzzy systeem [3][5][13][16][17][18]. Een dergelijk systeem, zoals die in het volgende hoofdstuk wordt beschreven, past vanwege zijn hiërarchische karakter beter bij de rijtaak. Of het ook geschikt is voor het modelleren van deze taak, zal uit het onderzoek blijken.

De fuzzy logic benadering is gebaseerd op het idee dat de beschrijving van hoe een bestuurder in het algemeen omgaat met zijn omgeving, informatie uitwisselt en beslissingen neemt, uitgevoerd wordt op een kwalitatieve manier. Het lijkt redelijk om de beslissingen uit de autobestuurderstaak te beschrijven met behulp van kwalitatieve variabelen die eveneens kwalitatieve waarden kunnen aannemen. Een kwalitatieve variabele zou bijvoorbeeld kunnen zijn 'snelheid' die de waarden 'langzaam', 'snel' en 'te snel' zou kunnen aannemen. Regels zouden er kunnen uitzien als: 'Ik ga te snel en de auto voor mij komt dichterbij, dus moet ik mijn snelheid een beetje laten afnemen.'

De meest gebruikelijke methode om dit soort vaag gedefinieerde problemen te benaderen is fuzzy logic. De manier waarop zo'n model geïmplementeerd wordt is als volgt: De set subsystemen is vaak hetzelfde als gegeven bij kwantitatieve expertsystemen. Voor ieder subsysteem worden de in- en uitvoervariabelen aangewezen en aan elke variabele wordt een set van kwalitatieve waarden (linguïstische labels) toegekend. Met behulp van zogenaamde membershipfuncties wordt de relatie tussen de getalsmatige (kwantitatieve) in- en uitvoer en de kwalitatieve waarden beschreven. Met de kwalitatieve waarden worden de regels geformuleerd die beschrijven hoe bepaalde besluiten moeten worden genomen.

Voor een autovolgsysteem, bijvoorbeeld, zijn de variabelen waarmee men rekening zou moeten houden snelheid, afstand tot de voorligger, maximale acceleratie, maximale deceleratie en de veilige afstand. Voor een variabele zoals de afstand tot de voorligger zou je waarden kunnen nemen als 'dichtbij' maar ook modificatoren als 'erg' en 'niet', om de nauwkeurigheid verder te verfijnen.

Kwantitatieve expertsystemen versus fuzzy logic
De reguliere kwantitatieve expertsystemen hebben als voornaamste voordeel de computationele efficiëntie. Het is echter heel moeilijk om de formules en de waarden te bepalen die opgenomen moeten worden in de kwantitatieve regels. Een blik op hierboven gegeven formule voor het bepalen van de veilige volgafstand bevestigt dit. Daarvoor is een grote hoeveelheid kennis nodig en eventueel resultaten van experimentele studies. De fuzzy benadering geeft een gemakkelijke manier om het verkeersgedrag te specificeren, doordat de wijze waarop de regels beschreven worden wat dichter bij natuurlijke taal staat. Zo kun je bijvoorbeeld de belangrijkheid van een variabele bepalen door de bovengenoemde modificatoren zoals 'erg' en 'niet' te gebruiken, in plaats van door het veranderen van kwantitatieve waarden. Er is meer (reken)tijd benodigd in het fuzzificatie- en defuzzificatie-proces, dat wil zeggen de vertaling van getalswaarden naar de interne linguïstische representatie en vice versa. De overgangen tussen verschillend gedrag zijn soepeler dan die van reguliere kwantitatieve expertsystemen die gebruik maken van drempelwaarden om de toepassing van bepaalde regels te bepalen. Het gebruik van membershipfuncties maakt het progressief toepassen van regels mogelijk. De uitvoer van de regels kan op verschillende manieren gecombineerd worden, waarover later meer.

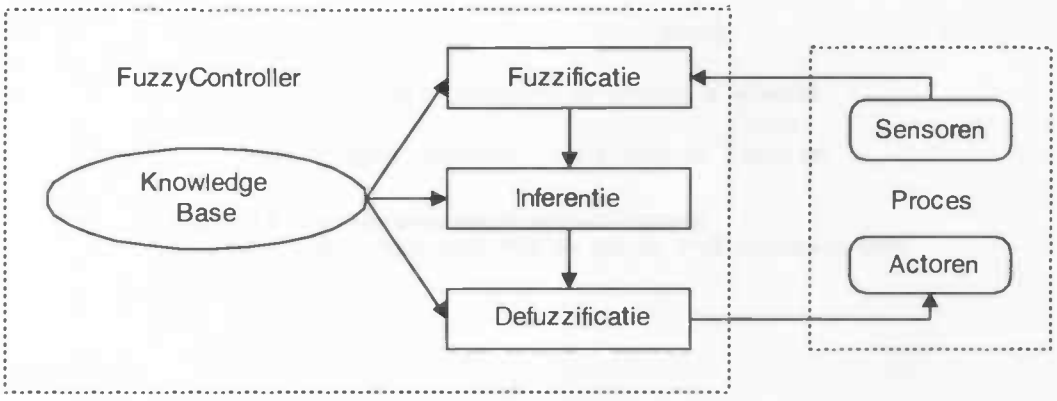
	Voordelen	Nadelen
Soar	- Architectuur voor probleemoplossen bestaat reeds; - Geeft een redelijk functioneel model van rijgedrag	- Computationeel inefficiënt; - Werken met kwantitatieve gegevens problematisch; - Afwezigheid van een timingsmechanisme; - Onvermogen om dingen te vergeten en om slechte dingen af te leren; - Soar is niet gebruikersvriendelijk
ACT-R	- ACT-R is een goede simulator van menselijke cognitie; - Gelaagde, hiërarchische structuur van doelen en gedrag; - Model produceert gedrag dat in kwalitatieve en kwantitatieve zin goed overeenkomt met menselijk gedrag	- Onbekend
HMMs	- Model kan gemaakt worden aan de hand van statistische empirische data, zonder tussenkomst van een menselijke expert	- Trainingsproces zeer kostbaar; - Non-deterministisch; - Methode is nog onvoldoende onderzocht
Kwantitatieve expertsystemen	- Computationeel efficiënt	- Harde grenzen; - Abrupte gedragsovergangen
Fuzzy logic	- Expertkennis op te geven in natuurlijke taal; - Vloeiende overgangen; - Hiërarchische structuur goed mogelijk; - Meerdere doelen tegelijkertijd volgen: inherent aan deze methode is een verscheidenheid aan manieren om uitvoer van subsystemen samen te nemen	- Computationeel minder efficiënt[1]; - Methode is nog onvoldoende onderzocht

Figuur 2.3 – Voor- en nadelen van verschillende modellen van bestuurdersgedrag
In dit overzicht staan de voor- en nadelen van de verschillende modelleermethoden voor autobestuurdersgedrag, zoals die bekend zijn in de literatuur. Opvallend is een negatief punt van de fuzzy logic methode: 'Methode is nog onvoldoende onderzocht'.

2.2 Fuzzy controllers (FCs)

Een fuzzy controller (zie figuur 2.4) bestaat uit vier componenten: een *knowledge base* (kennisbank), een *inferentie*mechanisme, en *fuzzificatie*- en *defuzzificatie*modules. Deze componenten worden in de volgende vier paragrafen verder toegelicht.

[1] Met de vermeende computationele inefficiëntie valt het in de praktijk mee. Dit zal in de onderzoeksresultaten (hoofdstuk 4) verduidelijkt worden.

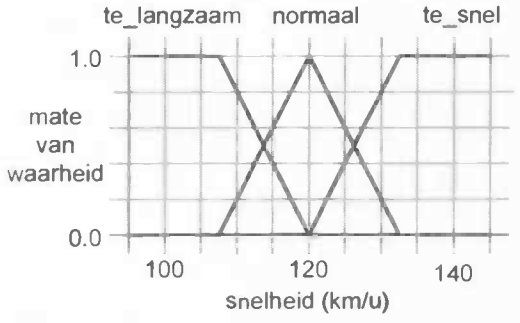


Figuur 2.4 – De architectuur van een fuzzy controller (naar [5])
Een fuzzy controller interacteert via sensoren en actoren met een proces. De informatie die via dé sensoren binnenkomt wordt middels fuzzificatie bruikbaar gemaakt voor het inferentieproces. De uitvoer die uit dit proces voortkomt wordt gedefuzzificeerd en aangeboden aan de actoren, om zodoende het proces te beïnvloeden. De wijze van (de-)fuzzificatie en de regels voor de Inferentie liggen vast in de 'knowledge base' (kennisbank).

2.2.1 Knowledge base

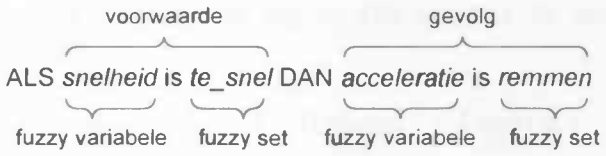
De knowledge base bevat de fuzzy variabelen en regels. Deze vormen als het ware de feitelijke en procedurele kennis van de fuzzy controler.

Fuzzy variabelen
Iedere *fuzzy variabele* wordt geassocieerd met een aantal termen, *fuzzy sets* genoemd. Een voorbeeld: De variabele 'snelheid' wordt geassocieerd met de drie termen 'te langzaam', 'normaal' en 'te snel'. Voor elke term is er een zogenaamde *membershipfunctie*, die het verband aangeeft tussen de snelheid als getalswaarde – bijvoorbeeld in kilometers per uur – en de *mate van waarheid* van de term bij die snelheid. Deze mate van waardheid of *geldigheid* wordt aangegeven door een waarde tussen 0 (volledig niet waar) en 1 (volledig waar). In figuur 2.5 staat een voorbeeld van hoe de membership-functies voor de drie genoemde termen eruit zouden kunnen zien.



Figuur 2.5 – Voorbeeld van membershipfuncties
Het is gebruikelijk alle membershipfuncties voor één variabele in dezelfde grafiek weer te geven. De functies – drie in dit voorbeeld – bepalen elk het verband tussen de snelheid en de mate van waarheid van de bijbehorende term.

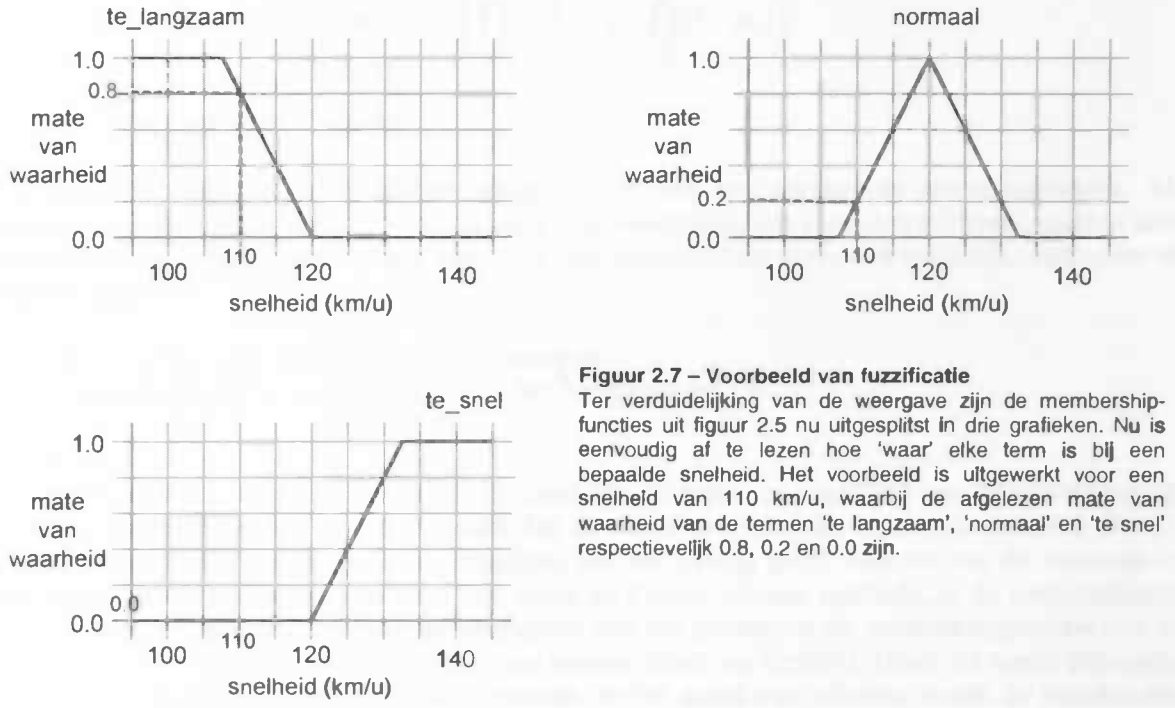
Fuzzy regels
De gebruikte *fuzzy regels* zijn van het type 'als *voorwaarde* dan *gevolg*' waarbij het *gevolg* een propositie is in de vorm '*fuzzy variabele* is *fuzzy set*' en de *voorwaarde* een conjunctie of disjunctie van dergelijke proposities (zie figuur 2.6). Aan elke regel kan een gewicht toegekend worden. Soms wordt dat gewicht de 'degree of support' van een regel genoemd.



Figuur 2.6 – Voorbeeld van een fuzzy productieregel
Een fuzzy productieregel heeft altijd de opbouw 'ALS voorwaarde DAN gevolg'.

2.2.2 Fuzzificatie

Tijdens het fuzzificatieproces wordt de mate van waarheid of geldigheid voor iedere fuzzy set van een fuzzy variabele bepaald. Dit gebeurt aan de hand van de membershipfuncties en de huidige getalswaarde van de grootheid die geassocieerd wordt met de fuzzy variabele. Zo wordt in ons voorbeeld bij een snelheid van 110 kilometer per uur de term 'te langzaam' voor 80% waar, 'normaal' voor 20% waar en 'te snel' 0% waar (zie figuur 2.7).



Figuur 2.7 – Voorbeeld van fuzzificatie
Ter verduidelijking van de weergave zijn de membership-functies uit figuur 2.5 nu uitgesplitst in drie grafieken. Nu is eenvoudig af te lezen hoe 'waar' elke term is bij een bepaalde snelheid. Het voorbeeld is uitgewerkt voor een snelheid van 110 km/u, waarbij de afgelezen mate van waarheid van de termen 'te langzaam', 'normaal' en 'te snel' respectievelijk 0.8, 0.2 en 0.0 zijn.

2.2.3 Inferentie

Het doorrekenen van fuzzy regels wordt *inferentie* genoemd. Het inferentieproces bestaat uit twee stappen, namelijk *aggregatie* en *compositie* (zie figuur 2.8).

Aggregatie
In de aggregatiestap worden eerst per regel het activatieniveau bepaald, oftewel de mate van waarheid van het gehele als-gedeelte van een regel. Dit gebeurt door het samennemen van de mate van waarheid van alle fuzzy sets in de voorwaarde. De wijze van samennemen is afhankelijk van de definitie van de conjuctie- en disjunctieoperatoren. Doorgaans wordt met de AND operator de

minimum membership waarde genomen en met de OR operator de maximum waarde. Ook een mengeling van beide is mogelijk:

2.2) minimum-maximum mix: $\mu = (1-\lambda)\min_{i=1..n}(\mu_i) + \lambda \max_{i=1..n}(\mu_i)$

$\lambda = 0.0$: minimum (AND),
$\lambda = 1.0$: maximum (OR),
$0.0 < \lambda < 1.0$: gemengd.

Hiernaast bestaan er nog twee aggregatiemethoden, die iets minder voor de hand liggen:

2.3) minimum-gemiddelde mix: $\mu = (1-\lambda)\min_{i=1..n}(\mu_i) + \lambda \sum_{i=1}^{n} \mu_i / n$

$\lambda = 1.0$: gemiddelde.

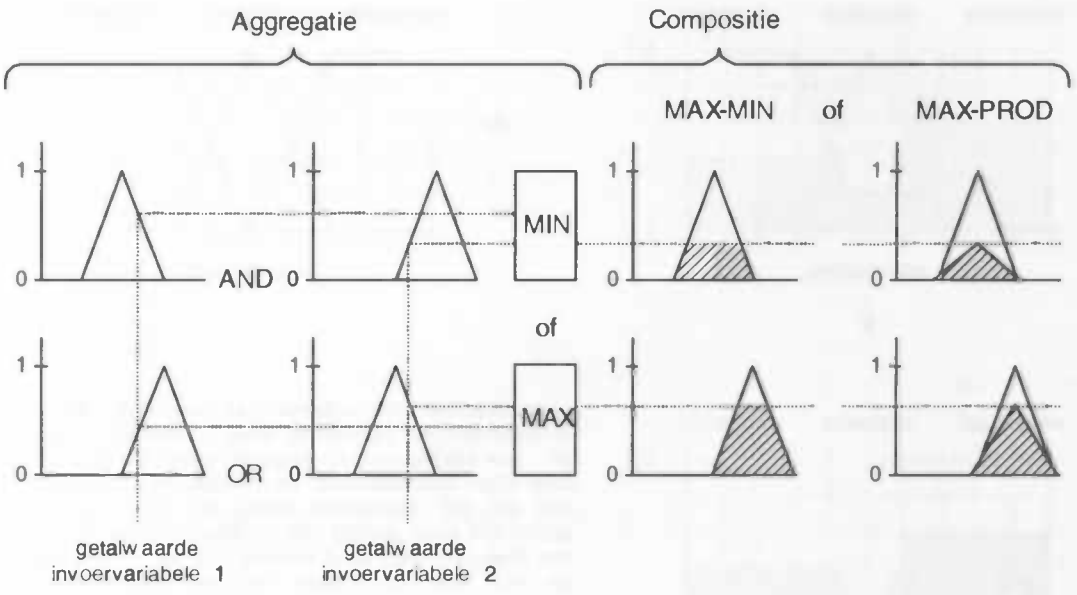
2.4) produkt-complement mix: $\left(\prod_{i=1}^{n} \mu_i\right)^{1-\gamma} \left(1 - \prod_{i=1}^{n} (1-\mu_i)\right)^{\gamma}$

$\gamma = 0.0$: gewoon product.

Naast de hierboven beschreven invoeraggregatie is er ook nog sprake van uitvoeraggregatie. Als namelijk meerdere regels in de uitvoer een mate van waarheid geven voor *dezelfde term*, moeten deze waarden ook samengenomen worden. Hiervoor zijn verschillende methoden mogelijk, waaronder de volgende twee:

1. maximum – het maximum van alle waarden;
2. bounded sum – de som van alle waarden, tot maximaal 1.0.

Compositie
In de tweede stap, compositie, wordt aan de hand van de mate van waarheid van de voorwaarde de mate van waarheid van het gevolg bepaald. Bij de meest gebruikelijke inferentiemethoden (MAX-MIN en MAX-PROD) is de mate van waarheid van het gevolg gelijk aan die van de voorwaarde. Voor bepaalde defuzzificatiemethoden, met name de Center-of-Area methode, is de compositiestap nog uitgebreider. Aan de hand van de geldigheid van het gevolg en de membershipfunctie van de fuzzy set van de uitvoervariabele wordt er een nieuwe fuzzy set bepaald. Hiervoor wordt afsnijding (MAX-MIN) of schaling (MAX-PROD) gebruikt. In het geval van schaling wordt de membership functie van de output fuzzy set vermenigvuldigd met de geldigheid van de conclusie. Bij afsnijding vormt de oppervlakte van de membershipfunctie onder de geldigheid van de conclusie de resultaat-fuzzy set. Beide methoden zijn grafisch weergegeven in figuur 2.8. Uit de praktijk is overigens gebleken dat deze inferentiemethoden min of meer door elkaar gebruikt kunnen worden.



Figuur 2.8 – Grafische weergave van fuzzy inferentie
Fuzzy inferentie is opgebouwd uit twee stappen, namelijk aggregatie en compositie. Voor belde stappen zijn twee alternatieven methoden uitgebeeld.

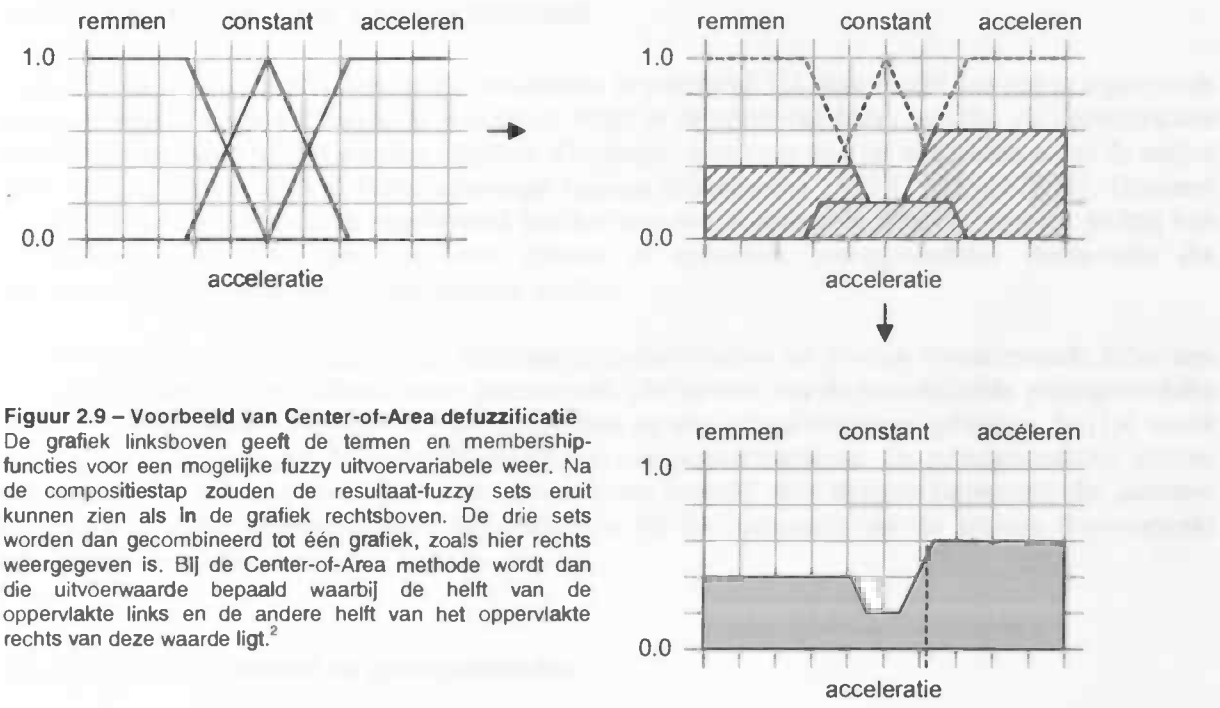
2.2.4 Defuzzificatie

De defuzzificatiestap is de laatste stap van het doorrekenen van een fuzzy logic systeem. In deze stap worden de getalswaarden bij de fuzzy uitvoervariabelen bepaald. Er zijn meerdere defuzzificatiemethoden met elk hun eigen voor- en nadelen. De Center-of-Maximum (CoM) en Center-of-Area (CoA) defuzzificatiemethoden resulteren in het 'beste compromis'; de Mean-of-Maximum (MoM) methode resulteert in de 'meest plausibele oplossing', aldus de documentatie bij fuzzyTECH. In besturingssystemen wordt CoM het meest gebruikt omdat de uitvoerwaarde het beste compromis geeft van alle geïnfereerde resultaten. MoM wordt vaak gebruikt bij patroonherkenning en classificatietoepassingen aangezien een plausibele oplossingen hier het meest gepast is. De drie genoemde methoden worden hieronder toegelicht.

CoA defuzzificatie
Center-of-Area (CoA) is de prevalerende methode voor defuzzificatie. Hij wordt ook wel 'centroid' of 'center of gravity' methode genoemd. Deze methode is relatief rekenintensief, aangezien er integralen berekend moeten worden. Dit is iets om rekening mee te houden in real-time toepassingen. De berekening is als volgt [11]:

$$2.5) \quad x_{uit} = \frac{\int \mu \cdot x \, dx}{\int \mu \, dx}$$

Een voorbeeld van CoA defuzificatie is uitgewerkt in figuur 2.9.



Figuur 2.9 – Voorbeeld van Center-of-Area defuzzificatie
De grafiek linksboven geeft de termen en membership-functies voor een mogelijke fuzzy uitvoervariabele weer. Na de compositiestap zouden de resultaat-fuzzy sets eruit kunnen zien als in de grafiek rechtsboven. De drie sets worden dan gecombineerd tot één grafiek, zoals hier rechts weergegeven is. Bij de Center-of-Area methode wordt dan die uitvoerwaarde bepaald waarbij de helft van de oppervlakte links en de andere helft van het oppervlakte rechts van deze waarde ligt.[2]

CoM defuzzificatie(mean-max-membership)
Deze methode wordt ook wel 'mean-max-membership' genoemd. Als er meer dan één uitvoerterm als (deels) waar kan worden geëvalueerd, bepaalt deze defuzzificatiemethode een compromis tussen de verschillende resultaten. De Center-of-Maximum methode berekent de getalsmatige uitvoer als gewogen gemiddelde van de maxima van de membership functies, gewogen door de inferentieresultaten. Als het maximum voor een term overigens niet uniek is – bijvoorbeeld bij trapezoïde membershipfuncties – wordt hier als maximum het gemiddelde van het maximuminterval genomen. In formulevorm [11]:

$$2.6) \quad x_{uit} = \frac{\sum_{i=1}^{n} \mu_i \cdot \bar{x}_i}{\sum_{i=1}^{n} \mu_i}$$

MoM defuzzificatie(max-membership)
De Mean-of-Maximum (MoM) methode berekent enkel een uitvoer voor de term met de hoogste resulterende mate van geldigheid. De uitvoer is dan het gemiddelde van de membershipfunctie van die term. Dus:

$$2.7) \quad x_{uit} = \bar{x}_{i\max} \text{ waarbij voor } imax \text{ geldt dat } \mu_{i\max} = \max_{i=1..n}(\mu_i)$$

In de literatuur zijn vele (variaties op deze) defuzzificatiemethoden te vinden.

[2] Helemaal correct (maar nog rekenintensiever) is het om de overlappende oppervlaktes dubbel te tellen in het berekenen van de center-of-area.

2.3 Hiërarchische vage gedragsystemen

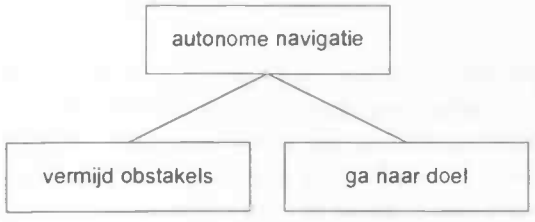
De traditionele fuzzy controllers, zoals beschreven in paragraaf 2.2, zijn vooral van nut in lagere-orde regelsystemen. De uitvoer van zo'n systeem is altijd in de vorm van fuzzy sets die via defuzzificatie naar kwantitatieve waarden worden omgezet. Dergelijke systemen zijn uit te breiden zodat de output van regel(set)s weer andere fuzzy systemen kunnen beïnvloeden [3][5][13][16][17][18]. Daarmee kunnen systemen hiërarchisch opgebouwd worden met een zelfgekozen diepte. Complex gedrag kan zo ontleed worden via een top-down proces in simpelere gedragsmodules (behaviors) die onafhankelijk van elkaar ontworpen kunnen worden.

Met behulp van het besproken fuzzy inferentiesysteem worden de diverse concurrerende behaviors gelijktijdig maar in verschillende mate geactiveerd. De uitvoer van de verschillende gedragsmodules worden samengenomen (command fusion). Dit kan op een aantal manieren gebeuren. In [13] wordt een methode aangedragen die gebruikt maakt van een activatieschema. De gedragsmodules hebben een dynamisch activatieniveau. Deze niveaus worden bepaald door diverse metaregels die daarmee dus bepalen welke gedragsmodules dominant zijn bij het aansturen van de actoren (bijvoorbeeld acceleratie en stuursnelheid).

2.3.1 Systemen gebaseerd op gedragsmodules

In de robotica wordt het probleem van autonome navigatie uitvoerig bestudeerd. Een populaire en succesvolle benadering is de zogenaamde '*behavior-based*' benadering [5][16][17]. Hierbij wordt complex gedrag verkregen met behulp van meerdere elementaire gedragsmodules (behaviors), die elk één specifieke deeltaak voor hun rekening nemen. Op een niveau daarboven worden het gedrag gecombineerd.

Zo kan als eenvoudig voorbeeld (zie figuur 2.10) het gedrag van een mobiele robot, die zich van A naar B moet bewegen tussen obstakels door, opgedeeld worden in de twee gedragsmodules 'vermijd obstakels' en 'ga naar doel'. De modules afzonderlijk richten zich uitsluitend op hun eigen taak: als de robot enkel bestuurd werd door 'vermijd obstakels' dan zou hij doelloos maar botsvrij rondrijden; terwijl als de 'ga naar doel' module het alleen 'voor het zeggen had' dan zou de robot proberen in een rechte lijn naar zijn doel gaan, zonder rekening te houden met enig obstakel.



Figuur 2.10 – Autonome navigatie door middel van een behavior-based systeem
In dit voorbeeld wordt het proces van autonome navigatie opgebouwd uit twee behaviors, namelijk 'vermijd obstakels' en 'ga naar doel'.

2.3.2 Coördinatie van de gedragsmodules: arbitrage en combinatie

Een belangrijk punt bij de behavior-based benadering is behavior co-ordination. Het idee is om steeds uit de verschillende navigatievoorstellen, afkomstig van de primitieve gedragsmodules, de beste actie te distilleren. De wijze waarop dit gebeurt is cruciaal voor het functioneren van het systeem.

In het coördinatieproces kan men twee fasen onderscheiden: arbitrage en combinatie. Met arbitrage wordt bedoeld het kiezen van één of meerdere primitieve gedragsmodules, die gaan bepalen wat de volgende actie is. Voor het geval dat er meer dan één gedragsmodule gekozen is, moet er vastgesteld worden hoe de verschillende navigatievoorstellen gecombineerd worden tot één actie. Dit laatste proces wordt ook wel '*command fusion*' genoemd [13].

In de literatuur worden veel verschillende arbitrage- en combinatiemethoden beschreven, waarvan de belangrijkste hieronder worden besproken.

Een veelvoorkomende benadering maakt gebruik van een prioriteringsschema waarin de navigatievoorstellen van slechts één primitieve gedragsmodule uit meerdere concurrerende modules wordt gekozen; aanbevelingen van de overige modules worden genegeerd. Een andere benadering neemt de aanbevelingen mee van alle modules die van toepassing zijn op de huidige situatie.

Hoe het ook zij, er moet altijd per cyclus bepaald worden welke primitieve gedragsmodules in welke mate de keuze voor de actie van dat moment zullen beïnvloeden. Dit kan aan de hand van een statisch, vooraf bepaald arbitrageschema. Beter is echter gebruik te maken van dynamische arbitrage, waarbij de keuze en het gewicht van de primitieve gedragsmodules afhangt van de huidige (sub-)doelen en omgevingssituatie [12][13]. Het is natuurlijk ook mogelijk om de prioriteiten en gewichten deels vooraf en deels on-the-fly te bepalen.

Bij dynamische arbitrage worden, in de fuzzy benadering, voor samengesteld gedrag fuzzy regels geformuleerd met wegende consequenties die de mate van toepasselijkheid bepalen van primitieve gedragsmodules op een lager niveau. Een voorbeeld:

> ALS *afstand_tot_obstakel* is *klein* DAN *vermijd_obstakels_gewicht* is *groot*

Voor deze regels wordt meestal de algemene term 'metaregels' gebruikt, maar ze worden ook wel 'applicability rules' [17] of 'context-rules' [13] genoemd. Meta-regels kunnen overigens ook gebruikt worden om verschillend gedrag in een bepaalde volgorde af te werken. In de premissen staat dan de voortgang van een bepaald doel:

> ALS *doel_1* is *niet_bereikt* DAN *gedrag_1* is *actief*
> ALS *doel_1* is *bereikt* DAN *gedrag_2* is *actief*
> ...

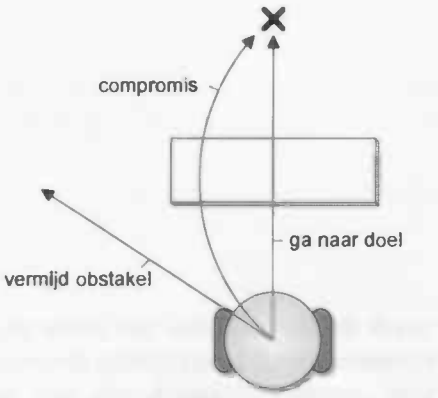
Als er in de arbitragefase meerdere behaviors zijn gekozen die samen de volgende actie zullen bepalen, dan moet hun output gecombineerd worden. Hierbij is het mogelijk gebruik te maken van simpele vectorsommatie na de defuzzificatie van alle afzonderlijke output. Er wordt dus eerst gedefuzzificeerd en dan gecombineerd. Een andere mogelijkheid is om standaard fuzzy inferentie-technieken te gebruiken. In de klassieke fuzzy controller worden immers ook de output van de fuzzy regels gewogen (afhankelijk van hun waarheidsgehalte in de huidige context) en daarna samengenomen tot één fuzzy output set die vervolgens gedefuzzificeerd kan worden. Bij deze variant wordt dus eerst gecombineerd en dan gedefuzzificeerd.

Voor mijn implementatie heb ik gekozen voor dynamische fuzzy arbitrage en voor fuzzy combinatie. De details hiervan worden besproken in hoofdstuk 3.

2.3.3 Conflict resolutie

Het combinatieproces kent een heel belangrijke complicatie: er zal meer dan eens een conflict ontstaan tussen de diverse navigatievoorstellen van de primitieve gedragsmodules. Om

dit toe te lichten haal ik nog een keer het simpele voorbeeldmodel van figuur 2.10 aan. Hierin worden in een mobiele robot twee primitieve gedragsmodules gecombineerd: één leidt de robot in een rechte lijn naar het doel ('ga naar doel') en de andere zorgt dat obstakels vermeden worden ('vermijd obstakels'). Stel nu dat de robot zich in de situatie bevindt waarbij hij recht voor een obstakel staat terwijl zijn reisdoel recht achter dit obstakel ligt (zie figuur 2.11). De 'ga naar doel' module zal voorstellen om rechtdoor te rijden, terwijl de 'vermijd obstakels' module zal aandringen om vooral niet rechtdoor te rijden maar links of rechts te sturen.
Worden deze output nu samengenomen, dan kan het compromis zijn dat de robot toch op het obstakel afrijdt.
Een goede en simpele oplossing voor deze conflicten is om van te voren vast te stellen dat zeer belangrijke primitieve gedragsmodules, zoals 'vermijd-obstakels' altijd prioriteit hebben boven andere gedragsmodules. Voorts is de keuze van de defuzzificatiemethode ook van grote invloed op het succes van de conflict resolutie. De populaire Center of Area (COA) methode levert altijd een compromisoplossing, wat vaak in het geval van een conflict een van-alles-net-niks oplossing betekent. De Centroid of Largest Area (CLA) defuzzificatie-methode zal meer neigen naar een oplossing conform het voorstel van de gedragsmodule met het zwaarste gewicht. Bij de Maximum of Means (MoM) is dit effect nog sterker:



Figuur 2.11 – Conflicterend gedrag in een eenvoudige robot
Om naar het doel te gaan dient de robot rechtdoor te rijden. Echter om het obstakel te vermijden moet flink uitgeweken worden, bijvoorbeeld naar links. Deze twee acties komen niet overeen. Een gewoon compromis tussen de acties leidt in dit geval niet tot zinvol gedrag.

MOM - winner takes all
COA - centrum van hele fuzzy set
CLA - centrum van grootste fuzzy subset

Een laatste mogelijkheid is om bij command fusion de vereniging van 'gewenste' en 'niet-verboden' richtingen te nemen (met behulp van de fuzzy MIN operator). Modules als 'ga naar doel' genereren typisch de 'gewenste' richtingen en modules als vermijd-obstakels de 'niet-verboden' richtingen. Voorwaarde is dus wel dat er overlap is tussen beide, anders is de vereniging leeg en is er sprake van een dead-lock.

2.4 Waarderen en vergelijken van modellen

Aangezien het veelal de bedoeling is van kunstmatige verkeersdeelnemers om een realistische verkeersomgeving te creëren, is een eerste vraag hoe de autonomie en het responsieve gedrag van het gesimuleerde verkeer aanvoelt voor de toeschouwer in vergelijking met echt verkeer. Doet dit realistisch aan? Hoewel het zeer moeilijk is om deze informatie te kwantificeren, is het erg belangrijk om op zijn minst een kwalitatieve indruk van de proefpersonen vast te leggen, aangezien die kunnen aanzetten tot het denken over missende of onnatuurlijke elementen in het model.

Naast deze kwalitatieve proefpersoongeörienteerde evaluatie is er nog een tweede mogelijkheid. Sommige verkeersmodellen bevatten kwantitatieve parameters die aangepast kunnen worden en in alle gevallen produceren ze een kwantitatieve uitvoer die vergeleken kan worden met echt bestuurdersgedrag. Voor de vergelijking dient dan wel eerst kwantitatieve data verkregen te worden van echt verkeer of bestuurdersgedrag [2].

De kwantitatieve vergelijking valt buiten het bereik van dit onderzoek.

3 Onderzoeksmethode

3.1 Inleiding

3.1.1 Motivatie voor de implementatie

Een deel van de bestuurderstaak zal geïmplementeerd worden in de vorm van een hiërarchisch fuzzy gedragsysteem dat een auto in een simulatie aanstuurt. Deze keuze wordt gemotiveerd door elementen uit het literatuuronderzoek. Enerzijds wordt daarin de potentie van een dergelijk systeem voor autonome navigatie in het algemeen duidelijk gemaakt. Anderzijds blijkt dat, in vergelijking met een aantal andere modelleermethoden, de toepassing van een hiërarchisch fuzzy logic model voor de specifieke taak van autobesturing onvoldoende onderzocht is. Aan de hand van deze implementatie kan de geschiktheid van deze methode voor het modelleren van de bestuurderstaak verder bepaald worden. De verwachte voor- en nadelen kunnen worden bekrachtigd of ontkend en mogelijk wordt de lijst uitgebreid met nieuwe punten. De vergelijking met andere methoden wordt zo beter mogelijk gemaakt.

3.1.2 Resumptie van de literatuurstudie

In het vorige hoofdstuk is ten eerste een inventarisatie gemaakt van de typen modellen in de verkeerskunde, en is de categorie geduid waarin in een hiërarchisch fuzzy gedragsysteem valt. Ten tweede is de rijtaak in zijn algemeenheid beschreven en is het idee dat de rijtaak een hiërarchische aard heeft verdedigd. Als derde zijn als vergelijkingsmateriaal de bevindingen van anderen bij het modelleren van de bestuurderstaak met bekende en beproefde architecturen beschreven. Ten vierde is uitgelegd hoe 'fuzzy controllers' en hiërarchische fuzzy gedragsystemen – zoals gebruikt in de robotica – werken. Als vijfde en laatste zijn in paragraaf 2.4 enkele methoden uit de literatuur voor het waarderen en vergelijken van modellen voor bestuurdersgedrag aangedragen.

3.1.3 Afbakening van de te implementeren taak

Het onderdeel van de bestuurderstaak dat geïmplementeerd zal worden is het rijden op de snelweg. De keuze is hierop gevallen omdat het bewegen op de snelweg een geïsoleerd en slechts beperkt deel van de subtaken behelst: de snelweg is een sterk gestructureerde taakomgeving. Het is door mensen aangelegd, met de uiteindelijk er op uit te voeren taak indachtig. Voor het manoeuvreren op de snelweg heeft de bestuurder slechts een subset van alle deeltaken uit de bestuurderstaak nodig. Daar een snelweg ongelijkvloerse kruisingen heeft, zijn alle deeltaken die betrekking hebben op het omgaan met kruisend verkeer niet aan de orde. Verkeerslichten komen op een snelweg niet voor. Het type verkeer dat zich op de snelweg bevindt is uniform: het bestaat enkel zwaar gemotoriseerd verkeer, waardoor de extra regels voor de interactie tussen verschillende verkeerstypen, zoals bijvoorbeeld auto's en fietsers, wegvallen.
Door de gereduceerde complexiteit van de bestuurderstaak op de snelweg wordt de implementatie van een model vergemakkelijkt. Bij succes geeft dit goede hoop dat de aanpak in de toekomst zal kunnen worden uitegebreid naar complexere verkeerssituaties.

3.2 Gebruik van het programma fuzzyTECH

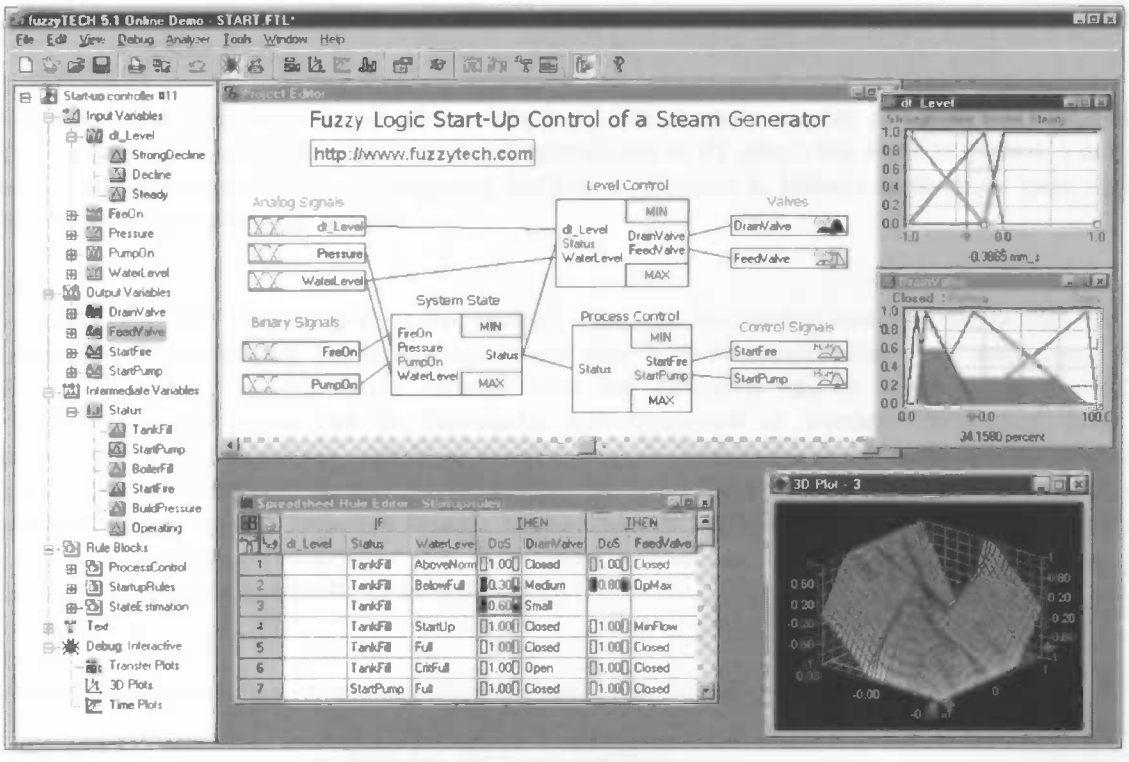
3.2.1 Inleiding

De Rijksuniversiteit Groningen heeft voor dit onderzoek tijdelijk een versie van het programma fuzzyTECH beschikbaar gesteld. Het eigenlijke gedragssysteem dat de auto in de simulatie aanstuurt wordt hierin vervaardigd.
Het programma biedt voldoende mogelijkheden om zo'n systeem te vervaardigen en te testen: Middels een GUI kan een compleet fuzzy logic systeem gemaakt worden met in- en uitvoer-variabelen, fuzzy tussenvariabelen, membershipfuncties en productieregels (zie figuur 3.1). De regels kunnen gerangschikt worden in regelblokken. Per regelblok kan de inferentie- en compositiemethode worden gekozen. Verder kan voor iedere uitvoervariabele de defuzzificatiemethode worden bepaald. Tot slot bevat het programma diverse on- en offline debugmogelijkheden.

3.2.2 FuzzyTECH als server

De simulatie kan communiceren met fuzzyTECH. Vanuit de simulatie wordt het juiste gedragsysteem in fuzzyTECH geladen en telkens doorgerekend. De debugmogelijkheden zijn op deze manier legio. Het gedragssysteem kan zelfs tijdens de simulatie gewijzigd worden, zodat de effecten direct zichtbaar zijn. De hoeveelheid tijd die benodigd is om op deze wijze het gedragsysteem door te rekenen is echter bij grotere systemen al snel te groot voor een real-time simulatie. Hiervoor is de simulatie aangepast zodat ze ook kan werken met een veel snellere binaire versie van het fuzzy gedragsysteem.

Figuur 3.1 – Screenshot van de fuzzyTECH gebruikersinterface
FuzzyTECH kent vele mogelijkheden voor het ontwerpen en testen van fuzzy logic systemen. In het project venster (midden boven) wordt de structuur van het systeem in een diagram weergegeven. De membershipfuncties van een variabele kunnen grafisch bekeken en gewijzigd worden (rechts boven). De productieregels kunnen worden bewerkt in een spreadsheet (midden onder) of als matrix. De screenshot toont fuzzyTECH in 'online debug mode', waarin bij alle fuzzy sets en regels in real-time aangegeven wordt in welke mate zij vuren.



(bron: http://www.fuzzytech.de)

3.2.3 Gegenereerde code

De simulatie bevat tevens de functionaliteit om te communiceren met een op zichzelf staande binaire versie van het fuzzy gedragsysteem, gegenereerd door fuzzyTECH. De zeer lage doorrekentijd van deze variant van het systeem (volgens de specificaties circa 1/10 milliseconde op een i486 PC voor een middelgroot regelsysteem) maakt de toepassing in een real-time simulatie mogelijk. Het gedragsysteem kan nu ook zonder de tussenkomst van fuzzyTECH doorgerekend worden. De debug-mogelijkheden zijn echter beperkt: er wordt alleen een getalsmatige monitor op de in- en output geboden: bij het starten van het simulatieprogramma opent zich een venster met daarin alle in- en uitvoervariabelen. Van elk van deze variabelen kan de minimum-, maximum en huidige waarde bekeken worden. Als fuzzyTECH echter wel op het systeem aanwezig is, kan er vanuit fuzzyTECH via het DDE communicatiekanaal verbinding worden gemaakt met de binaire versie van het gedragsysteem en zijn de uitgebreide debug-mogelijkheden alsnog beschikbaar.

3.3 Te implementeren onderdelen van de rijtaak

3.3.1 Inleiding

In de implementatie dient een model van het proces van het maken van beslissingen over verkeershandelingen geïmplementeerd te worden. De rijtaak is echter, zoals uit het literatuuronderzoek immers gebleken is, breder dan dit beslissingsproces. Ik verwijs in dit verband nog eens naar de deeltaken van Aasman [1], genoemd in paragraaf 2.1. Gelukkig is het voor dit onderzoek niet noodzakelijk elke deeltaak volledig en natuurgetrouw te implementeren. De hiervoor benodigde tijd zou de termijn van dit onderzoek vele malen overschrijden. Ik ga de deeltaken nog een keer bij langs, voeg er daarenboven een toe die mijns inziens ontbreekt, en bespreek in hoeverre en hoe natuurgetrouw de desbetreffende deeltaak geïmplementeerd dient te worden.

3.3.2 Visuele oriëntatie

Omschrijving
'Een automobilist heeft de continue taak van op het juiste moment naar de juiste plek kijken, om de voor de verkeerstaak relevante objecten en gebeurtenissen in de omgeving waar te nemen. Voor een inspectie van alles wat zich in de omgeving bevindt en afspeelt is immers in de regel geen tijd; er moeten beslissingen genomen worden.'

Toelichting
Voor het nemen van beslissingen in het verkeer dient de bestuurder zich bewust te zijn van de verkeersituatie om zich heen. Dit bewust-zijn noemt men in de literatuur '*situational awareness*' [15]. Voor het verkrijgen van een beeld relevant voor het uitvoeren van de verkeerstaak dienen de zintuigelijke waarnemingen van de bestuurder geïnterpreteerd te worden. Bovendien dient de aandacht voortdurend dusdanig gericht te worden, dat de bestuurder deze benodigde zintuigelijke waarnemingen kan vergaren binnen het tijdsbestek dat het maken van real-time beslissingen in een dynamische omgeving met zich mee brengt. Tot slot kan de bestuurder door te redeneren informatie over de verkeersituatie verkrijgen die niet direct waarneembaar is, zoals de intenties van medeweggebruikers en de situatie in de dode hoek [15].

Implementatie
Voor het uitvoeren van de verkeerstaak dient het beeld van de situatie minimaal te bevatten:

- positie, oriëntatie en snelheid van het eigen voertuig;

- positie, oriëntatie en snelheid van obstakels, met name andere voertuigen.

De term *positie* slaat op de laterale en longitudinale positie ten opzichte van de weg, maar ook de rijstrook waaraan een voertuig of obstakel is toegekend, een relevant gegeven bij het maken van beslissingen tijdens de rijtaak [15]. Er bestaan diverse geavanceerde technieken waarmee men in een 'real-life' situatie geautomatiseerd voertuigen of obstakels aan rijstroken kan toekennen aan de hand van sensorische informatie, eventueel ondersteund met elektronische wegenkaarten en GPS [15]. In het simulatieprogramma wordt de rijstrook waarop een voertuig zich bevindt eenvoudigweg direct bepaald uit de – reeds bekende – laterale positie.

3.3.3 Snelheidshandhaving

Omschrijving
'De bestuurder moet ook steeds in kritische situaties de snelheid aanpassen aan de taakomgeving, bijvoorbeeld bij het naderen van een kruispunt of bij het volgen van iemand op de snelweg. Deze taak (snelheidshandhaving) is sterk gerelateerd aan de visuele oriëntatie- en de motorische controletaak.'

Toelichting
Om de gewenste maximumsnelheid te bepalen kijkt men naar de geometrie van de weg, het zicht, het weer, de prestaties van de auto en de aangegeven maximumsnelheid. In het simulatieprogramma is de gewenste maximumsnelheid voor een bestuurder constant.

Implementatie
Of deze maximumsnelheid daadwerkelijk gereden kan worden, hangt af van de verkeersituatie. Als de bestuurder achter een auto rijdt met een lagere snelheid, zal hij zijn snelheid daar aan aan moeten passen, teneinde een botsing te voorkomen. De kunstmatige bestuurder zal uitgerust moeten worden met een dergelijke 'auto-volg' module.

3.3.4 Recht rijden en bochten nemen

Omschrijving
'Ook moet de bestuurder steeds zorgen dat hij de wagen binnen de rijstrook houdt. Bij bochten moet bepaald worden wanneer de bocht wordt ingezet en welke stuurhoek vereist is. Deze taak is eveneens sterk gerelateerd aan de visuele oriëntatie- en de motorische controletaak.'

Implementatie
Voor de eenvoud en het gemak is de weg in de simulatie helemaal recht. Daardoor is de laterale positie ten opzichte van het midden van de rijstrook direct te bepalen. De uitbreiding naar het bepalen van deze laterale positie op een weg met bochten is eenvoudig. Het principe voor het aanhouden van de rijstrook blijft gelijk: middels een regelkring wordt de laterale afwijkingen van het midden van de rijstrook telkens gecorrigeerd door te sturen.

3.3.5 Rijstrook kiezen

Eén punt ontbreekt in Aasman's opsomming van de deeltaken: er zal ook van tijd tot tijd bepaald moeten worden *welke* rijstrook er gevolgd moet worden. Dit speelt een rol bij het inhalen van langzamer verkeer, het ruimte geven aan inhalend (sneller) verkeer en het nemen van een afslag.

3.3.6 Motorische controle bij het bedienen van instrumenten

Omschrijving
'Om te sturen of van snelheid te veranderen, moeten de instrumenten worden bediend. Gas geven, remmen, sturen, schakelen en dergelijke zijn voor de geoefende rijder geheel geautomatiseerde taken.'

Implementatie
Deze deeltaak is zeer eenvoudig voor de kunstmatige bestuurder. De commando's voor de voertuig-controle worden als een vector van getallen doorgegeven aan het natuurkundig voertuigmodel.

3.3.7 Navigatie

Omschrijving
'Vervolgens is er het probleem van bepalen waar men heen moet bij afslag of kruispunt. Dit vergt een stuk planning vooraf, maar ook aanpassen aan de omgeving in het geval van afzettingen, omleidingen, files, etcetera.'

Implementatie
De navigatietaak ligt – volgens de indeling van de rijtaak in drie niveaus van Michon zoals beschreven in paragraaf 2.1 – in het bovenste (strategische) taakniveau. Deze laag ligt boven het door mij geïmplementeerde deel van de bestuurderstaak.

3.3.8 Coördinatie en multitasking

Omschrijving
'Als laatste moet de juiste actie op het juiste moment, en soms meerdere tegelijk, worden uitgevoerd.'

Implementatie
Dit is de essentie van de uit te proberen hiërarchische 'behavior-based' benadering. De uitvoer van de primitieve gedragsmodules wordt op het niveau daarboven gecoördineerd.

3.4 Richtlijnen voor het gedragsysteem

3.4.1 Inleiding

Alvorens aan het creëren van het gedragsysteem te beginnen dient er een inventarisatie gemaakt te worden van de benodigde primitieve gedragsmodules en van de meta-regels die deze modules coördineren. Aan de hand van deze regels wordt bepaald welke invoer – in de zin van informatie over de auto en zijn omgeving – er benodigd is. De uiteindelijke uitvoer staat reeds vast. Dit zijn namelijk een acceleratie en stuursnelheid.

3.4.2 Gedragsmodules

Primitieve gedragsmodules
De volgende drie primitieve gedragsmodules worden in het gedragssysteem geïmplementeerd, als bouwstenen voor het uiteindelijke complexe gedrag:

1. houd een bepaalde rijstrook aan

Figuur 3.2 – Twee voorbeelden van 'vaste' gedragsreeksen uit de bestuurderstaak
Bij het inhalen en het nemen van een afslag lijkt een vaste set deelhandelingen in een vaste volgorde te worden uitgevoerd.

2. volg een auto (in termen van snelheid)
3. houd gewenste maximumsnelheid aan

Modules voor gedragscoördinatie
De coördinatiemodules bevinden zich in de hiërarchie boven de primitieve modules. Ze bevatten meta-regels die de primitieve modules coördineren. Eén of meer van deze coördinatiemodules zullen worden geïmplementeerd, zodat de bestuurder zich op de snelweg kan handhaven. In hoofdstuk 2 zijn verschillende methoden voor het coördineren besproken. Deze worden uitgeprobeerd teneinde de beste variant te bepalen.

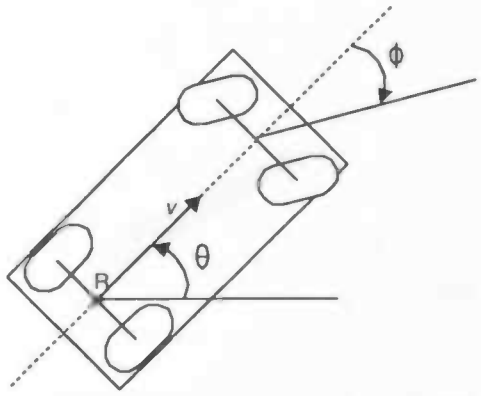
Vóór het samenstellen van de gedragsmodules worden het beoogde gedrag en de oorspronkelijke heuristieken uit de expertkennis in natuurlijke taal uitgeschreven. Aan de hand daarvan worden de in- en uitvoervariabelen, de benodigde fuzzy sets en de productieregels bepaald. Een beschrijving van het uiteindelijke systeem, inclusief de productieregels en de bijbehorende oorspronkelijke heuristieken, zal als bijlage worden opgenomen in het verslag.

3.4.3 Sequentieel gedrag

Sommige verkeershandelingen bestaan uit een reeks handelingen die doorgaans in een vaste volgorde worden uitgevoerd. Twee voorbeelden, namelijk het inhalen en het nemen van een afslag, zijn uitgewerkt in figuur 3.2. In hoofdstuk 2 (paragraaf 2.3.2) is reeds een methode aan de hand gedaan om met behulp van meta-regels de mogelijkheid tot het uitvoeren van dergelijke gedragsreeksen te implementeren.

Hierbij worden alvast twee kanttekening geplaatst. 1) De genoemde gedragsreeksen worden zoals gezegd *doorgaans* in vaste volgorde uitgevoerd. Mogelijk echter wordt een gedragsreeks onderbroken, omdat de gehele uitvoering niet (tijdig) haalbaar lijkt. 2) Gedragreeksen zijn mogelijk '*emerging properties*' van navolgen van verschillende doelen tegelijkertijd.

Tijdens de implementatie van het gedragsysteem wordt de noodzaak en de praktische waarde van het expliciet implementeren van modules voor het uitvoeren van gedragsreeksen proefondervindelijk vastgesteld.



Figuur 3.3 – Schematische weergave van de auto met toestandsparameters (naar Fraichard [5])
Voor het uitdrukken van de positie, de snelheid u en de rotatie θ van het voertuig in het platte vlak wordt het punt R (midden tussen de achterwielen) als referentiepunt genomen. Doordat de achteras star is en men met de voorwielen stuurt, is de positie van dit punt in het locale assenstelsel namelijk bij benadering constant. De rotatie van de voorwielen wordt uitgedrukt met de variabele φ.

3.5 Simulatie van voertuig en taakomgeving

3.5.1 Inleiding

Om het model in de praktijk te toetsen is een eenvoudig simulatieprogramma vervaardigd. Het programma is geschreven in C++ en maakt gebruik van de 3D visualisatie- en simulatiekernel 4Space, ontwikkeld door Vesc BV. Tijdens de simulatie worden één of meerdere door het gedragsysteem bestuurde auto's, alsmede de weg en de overige auto's uit het scenario gevisualiseerd. In de komende paragrafen volgen de details. Een beknopte handleiding bij het simulatieprogramma is opgenomen als bijlage aan het eind van dit verslag.

3.5.2 Natuurkundig automodel

Om in een simulator op elk moment de positie en oriëntatie van een voertuig te kunnen bepalen is een natuurkundig automodel benodigd. In dit model worden de toestand van de auto beschreven, alsmede de wijze waarop deze toestand verandert als functie van de tijd, de invoer van de bestuurder en eventuele factoren van buitenaf.

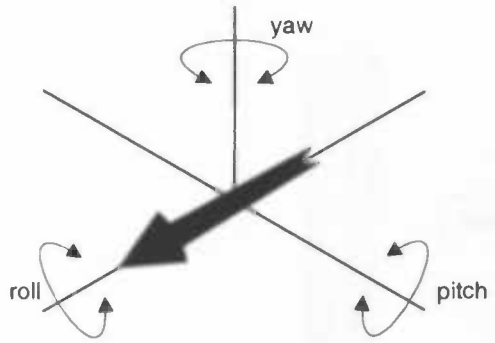
Toestandsbeschrijving
Aangenomen wordt dat de auto zich beweegt over een plat vlak (R^2). Hij heeft twee starre achterwielen en twee stuurbare voorwielen. De huidige toestand wordt beschreven met de volgende vijf parameters:

- x, y: de coördinaten van het middelpunt R van de achteras;
- θ: de oriëntatie van de auto, dit is de hoek tussen de lengterichting van de weg en de hoofdas van het voortuig;
- φ: de oriëntatie van de voorwielen ten opzichte van de auto;
- v: de snelheid van punt R gemeten langs de hoofdas van de auto.

Ter verduidelijking zijn deze parameters ingetekend in een schematische weergave van een auto in het platte vlak (figuur 3.3).

Invoer
Om een auto te besturen moeten het stuurwiel, het gaspedaal en de rem bediend worden. De invoerparameters voor de auto zijn aan de hand hiervan de volgende:

- de longitudinale versnelling $v' \in [-v'_{min}, v'_{max}]$;



Figuur 3.5 – 'Roll, Pitch and Yaw'
De drie-dimensionale oriëntatie van het chassis van een auto wordt beschreven door middel van drie rotatiewaarden, namelijk: 1) 'roll', de rotatie om de lengte-as, als gevolg van de centrifugale kracht; 2) 'pitch', de rotatie waarbij de neus van het object omhoog of omlaag gaat als gevolg van het optrekken of afremmen en 3) 'yaw', de rotatie waarbij de neus van het object naar links danwel rechts wordt bewogen als gevolg van bijvoorbeeld het sturen.

- de stuursnelheid $\varphi' \in [-\varphi'_{min}, \varphi'_{max}]$.

De kunstmatige bestuurder geeft kortom commando's in de vorm (υ', φ') aan de auto.

Doorrekenen van het model
Het gebruikte natuurkundig automodel is een extreme versimpeling van de werkelijkheid. De auto heeft geen interactie met zijn omgeving en de stuurhoek en acceleratie worden vrijwel direct vertaald naar translatie en rotatie van de auto als object in de gesimuleerde omgeving (zie figuur 3.4).

'Hollywood-effecten'
Echte auto's zijn geveerd en dat is ook zichtbaar in het gedrag van de auto. De effecten van de veren komen tot uitdrukking in de oriëntatie van het chassis. Deze oriëntatie wordt vaak uitgedrukt in de rotatie om de lengte- en de breedte-as, die respectievelijk 'roll' en 'pitch' worden genoemd (zie figuur 3.5). Bij het optrekken helt het chassis naar achteren, en bij het afremmen naar voren. Als de auto een bocht maakt, helt hij over naar de buitenkant van de bocht. Om dit effect natuurgetrouw na te bootsen, dient er een systeem van veren en dempers geïmplementeerd te worden dat krachten uitoefent op het chassis, die vervolgens de rotatie van het chassis bepalen. FuzzyDriver bepaalt de 'roll' en de 'pitch' rechtstreeks uit respectievelijk de laterale en longitudinale versnelling van de auto. Deze methode levert geen natuurgetrouwe simulatie van de werkelijkheid, maar maakt wel dat het gedrag van de auto natuurlijker oogt. Dit lijkt een detail, maar het is wel een detail dat het oordeel van de experts over de 'natuurlijkheid' van het gedrag anders negatief zou kunnen beïnvloeden.

3.5.3 Toestand & Perceptie

Toestand van de bestuurder
Van de bestuurder is bekend welke maximumsnelheid (constant) en welke laterale positie (variabel) hij graag wil aanhouden.

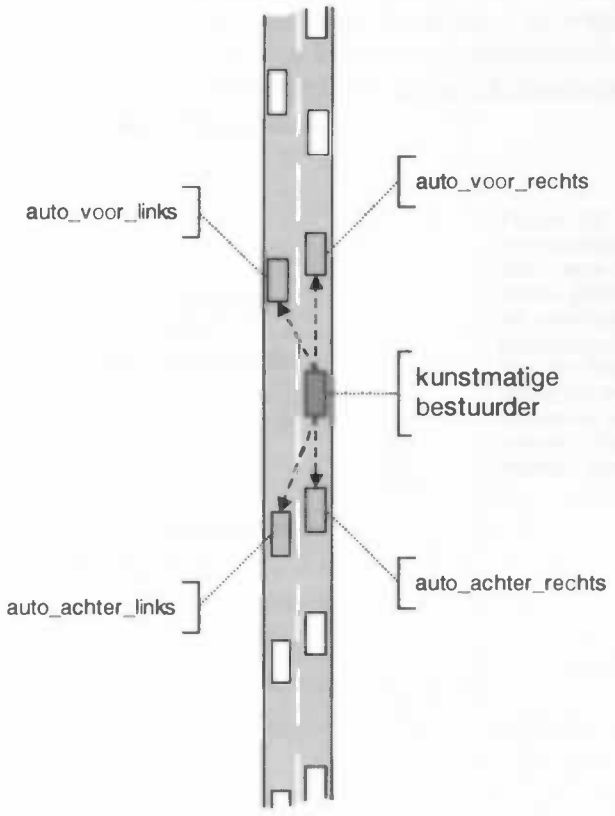
Toestand van het eigen voertuig
Van het eigen voertuig is bekend de afwijking van de gewenste laterale positie en de afwijking van de door de bestuurder gewenste snelheid. Ook is de afstand en daarmee de tijd tot een eventuele te nemen afslag bekend.

stuurhoek += stuursnelheid * Δtijd
rotatie += $\tan^{-1}$ (tan (stuurhoek * (Δx / c)))

snelheid += acceleratie * Δtijd
Δx = snelheid * Δtijd

$translatie_{longitudinaal}$ += Δx * sin (rotatie)
$translatie_{lateraal}$ += Δx * cos (rotatie)

Figuur 3.4 – Gebruikte formules voor het bijwerken van de translatie en rotatie van de auto als functie van acceleratie, stuursnelheid en frametijd
De snelheid wordt binnen een minimum- en maximumwaarde gehouden. De maximaal mogelijke stuurhoek wordt kleiner bij hogere snelheid.



Figuur 3.6 – Perceptie van andere auto's
De kunstmatige bestuurder neemt maximaal vier auto's waar. Dit zijn de auto's direct voor en achter hem, op de linker- en rechter rijstrook.

Perceptie van obstakels (andere voertuigen)
Stilstaande obstakels komen in de simulatie niet voor. Wel wordt de noodzakelijke informatie over voertuigen in de directe omgeving doorgegeven. Telkens als de perceptie-module wordt aangeroepen wordt er in de directe omgeving (momenteel binnen de arbitraire grens van 100 meter) gezocht naar een viertal voertuigen, namelijk steeds de meest dichtbijzijnde voertuigen voor en achter en op de linker- en rechter rijstrook (zie figuur 3.6). De afstand tot en snelheid van deze voertuigen wordt geregistreerd. Dit is de minimale informatie nodig om te kunnen manoeuvreren door het verkeer.

Toekennen van voertuigen aan rijstroken
Het toekennen van auto's aan rijstroken gebeurt in de implementatie op eenvoudige wijze: als het middelpunt van de auto zich links van de scheidslijn tussen de twee rijstroken bevindt, dan wordt de auto aan de linker rijstrook toegekend en anders aan de rechter. De vraag is of dit niet eigenlijk met vage logica geïmplementeerd zou moeten worden. Dit zou wel leiden tot soepelere overgangen, maar het toekennen van obstakels aan rijstroken is een perceptuele taak en de focus van dit onderzoek ligt op besluitvorming en gedrag en niet op perceptie. De genoemde simpele implementatie voldoet voor dit onderzoek. Eventuele haperingen die als gevolg hiervan in het gedrag waargenomen worden door de experts, bijvoorbeeld als de testauto wisselt van rijstrook, kunnen als zodanig afgedaan worden.

3.5.4 Scenario's

De gesimuleerde omgeving bestaat uit een lange rechte weg met twee rijstroken die dezelfde rijrichting hebben, zoals op een snelweg. Dit is een constant gegeven. Variabel is het aantal en het type verkeersdeelnemers, alsmede de eigenschappen die hun gedrag bepalen. Deze variabele gegevens vormen het scenario waarin het gedrag van FuzzyDriver getoetst wordt. De simulatie kan op elk gewenst moment gestopt en opnieuw gestart worden met een ander scenario. Een scenario wordt gedefinieerd in een tekstbestand, waarin de relevante eigenschappen van 'meespelende' auto's staan

vermeld. Het type besturing (fuzzy of scripted) wordt hier aangegeven, alsmede de initiële positie, toestand en – in het geval van fuzzy besturing – de waardes voor de strategische gewichten. Figuur 3.7 geeft een voorbeeld van een scenario-definitiebestand.
Doordat de scenario's vastgelegd zijn in de tekstbestanden en het systeem deterministisch is, is het gedrag volledig reproduceerbaar.

```
...
// car 0
{
        m_control_type = fuzzy
        m_speed = 33.3
        m_desired_speed = 33.3
        m_x = -.25
        m_y = 0.0
}
// car 1
{
        m_control_type = scripted
        m_speed = 27.8
        m_x = -.25
        m_y = 100.0
}
...
```

Figuur 3.7 – Fragment uit een tekstbestand waarin een scenario wordt gedefinieerd
Alle relevante eigenschappen (het type besturing, de initiële positie en snelheid, de eventuele doelsnelheid en eventuele afwijkende waardes voor strategische gewichten en initiële acceleratie, rotatie, stuurhoek) van de auto's die een rol spelen in het scenario worden in een tekstbestand beschreven. Bij het op- of herstarten van de simulatie wordt het desbetreffende bestand ingelezen en de beginsituatie voor het scenario gevormd.

3.6 Evaluatie

3.6.1 Inleiding

Er zal een systeem ontwikkeld worden die het model van de kunstmatige bestuurder implementeert. Dit systeem zal: 'FuzzyDriver' worden gedoopt. De simulatie en het programma FuzzyTECH samen bieden voldoende mogelijkheden om tot een evaluatie van FuzzyDriver te komen. In de nu volgende subparagrafen wordt beschreven hoe één en ander in zijn werk gaat. De evaluatie zal op de volgende drie punten plaatsvinden:

1. Ontwerpproces
2. Implementatie
3. Prestaties

Ten aanzien van het ontwerpproces wordt gekeken naar de ontwerpkosten en de mogelijkheden voor een incrementeel ontwerpproces. Bij 'implementatie' wordt gekeken naar de algemene opzet (architectuur) van FuzzyDriver, de gebruikte fuzzy variabelen en membership functies, de primitieve gedragsmodules, de coördinatie van deze modules en tot slot de strategische gewichten die de rijstijl van FuzzyDriver bepalen. De prestaties, tot slot, worden in kaart gebracht en geëvalueerd aan de hand van open en closed loop metingen, observatie van het gedrag door experts en door het bepalen van de computationele eisen van het systeem.

3.6.2 Ontwerpproces

Ontwerpkosten
Lage ontwerpkosten hebben een positieve invloed op het oordeel over de geschiktheid van een modelleermethode. Die kosten worden gevormd door de hoeveelheid tijd en de hoeveelheid en het type middelen ('resources') die benodigd zijn om een model of systeem te vervaardigen. Om een idee

te krijgen van deze ontwerpkosten dient een inventarisatie gemaakt te worden van de stappen waaruit het ontwerptraject is opgebouwd. Van elk van deze stappen wordt beschreven welke investering ze behoeven in termen van tijd en middelen. Verder doet het er toe of de benodigde middelen schaars en/of kostbaar zijn.
Er dient een schifting gemaakt te worden tussen ontwerpstappen die eenmalig zijn voor dit onderzoek en die welke voor de ontwikkeling van ieder toekomstig model weer noodzakelijk zullen zijn. Bij de eenmalige onderdelen moet men denken aan het zoeken naar de juiste algemene architectuur. De onderdelen die gemoeid zijn met het aanwenden van deze architectuur voor het modelleren van een specifieke taak zullen juist telkens terugkeren.

Incrementeel ontwerpproces
Tot slot dient vastgesteld te worden of de hiërarchische fuzzy logic methode zich leent voor een zogenaamd *incrementeel ontwerpproces*. De mogelijkheid tot een incrementeel ontwerpproces is mede bepalend voor de ontwerpkosten.
Het pad van een papieren ontwerp naar een volledig geïmplementeerd systeem kan, afhankelijk van de gekozen methode, op verschillende manieren bewandeld worden. Idealiter kan er initieel heel snel een volwaardig systeem vervaardigd worden, dat echter een beperkt deel van de gedragsfunctionaliteit bevat. Zo kan er in een vroege fase bepaald worden of het idee achter het ontwerp werkt. Voorwaarde is wel dat er daarna steeds (gedrags)functionaliteit toegevoegd kan worden, zonder dat het bestaande systeem 'omvalt' en feitelijk weer van de grond af aan moet worden opgebouwd. Een dergelijke manier van ontwerpen noemt men een incrementeel ontwerpproces. Als het hiërarchische fuzzy gedragsysteem zich hiervoor leent, is dat een grote pré.
Om deze 'uitbreidbaarheid' te testen, implementeer ik eerst een volledig functioneel systeem, zoals dat eerder in dit verslag beschreven is. Zodra dit getest is en volledig werkt, kan getracht worden het systeem uit te breiden. Voor deze test wordt het gedrag toegevoegd waarmee de kunstmatige bestuurder een afslag kan nemen.

3.6.3 Implementatiedetails

In de resultaten zullen de implementatiedetails van het uiteindelijke gedragsysteem uitvoerig besproken worden. Tijdens de implementatie zal een aantal keuzes gemaakt moeten worden. Deze keuzen dienen onderbouwd en achteraf besproken te worden.

Algemene architectuur
Het systeem zal worden opgebouwd uit twee á drie lagen. Eerder werd al genoemd dat er in hogere taakniveaus steeds de voorwaarden worden bepaald voor de beslissingen op de lagere niveaus. In FuzzyDriver wordt deze communicatie tussen de verschillende lagen vormgegeven door variabele gewichten. De productieregels in de hogere lagen hebben als uitvoer het gewicht waarmee de regels op een lagere laag gewogen worden. Deze gewichten kunnen eventueel nog per variabele gemengd worden met statische gewichten.

Variabelen en membershipfuncties
Als een gedrag- of regelsysteem eenmaal globaal opgezet is, breekt het moment aan om te experimenteren met de waarden van de parameters, teneinde het beoogde gedrag te bewerkstelligen. Een groot aantal instelbare parameters geeft een generiek systeem met de mogelijkheid om het op maat te maken voor verschillende omstandigheden. Mensen die ervaring hebben met het kalibreren van regelsystemen weten echter dat het optimaliseren van zo'n stelsel van meestal onderling afhankelijke parameters zeer complex en tijdrovend kan zijn. Uit ervaring blijkt dat bij een groot aantal parameters het beoogde optimum door middel van handmatig afstellen zelfs vaak niet binnen redelijke tijd wordt gevonden. Geautomatiseerd optimaliseren met behulp van bijvoorbeeld neurale netwerken, genetische algoritmen of simulated annealing kan dan een oplossing bieden. Als dit soort

algoritmen niet beschikbaar of te kostbaar zijn, dan kan men idealiter met een kleine set instelbare parameters een breed spectrum van gedrag creëren.
De mate waarin deze ideale situatie bereikt wordt is een criterium voor het meten van de geschiktheid van de te onderzochte modelleermethode. De potentieel instelbare parameters van het hiërarchische fuzzy gedragsysteem van FuzzyDriver zijn achtereenvolgend:

- de statische gewichten;
- de degree of support per productieregel;
- de membershipfuncties van de fuzzy sets;
- de parameters voor het (lineair) transformeren van de gedefuzzificeerde in- en uitvoerwaarden.

Bij de implementatie zal blijken welke parameters ingesteld moeten worden voor het verkrijgen van normatief gedrag en hoeveel paramaters er nodig zijn om de persoonlijke rijstijl te kunnen variëren.

Primitieve gedragsmodules
De onderste laag van het systeem wordt gevormd door de primitieve gedragsmodules. Dit zijn de elementaire bouwstenen van al het gedrag dat het systeem kan genereren. De verzameling van primitieve gedragsmodules dient daartoe uitputtend te zijn, dus *al* het beoogde gedrag moet te definiëren zijn als resultaat van het (gecombineerd) aanwenden van één of meerder van deze modules. Tegelijkertijd moet er naar gestreefd worden het aantal modules te minimaliseren. Redundantie in deze laag is ongewenst. Het leidt tot ongewenste gevolgen, namelijk onnodig rekenwerk voor de processor, het compliceren van de coördinatie van de modules en bovendien is de kans groot dat de wijze waarop complex gedrag samengesteld kan worden niet meer eenduidig is.

Gedragscoördinatie
Het coördineren van de gedragsmodules geschiedt door het bepalen van de randvoorwaarden voor de modules en de gewichten waarmee de uitvoer per module meetelt voor de systeemuitvoer. Bepaalde gewichten kunnen desgewenst a priori ingegeven worden, de andere gewichten en de randvoorwaarden worden dynamisch bepaald.
De gewogen samenname is een resultaat van het defuzzificatieproces. In hoofdstuk 2 is een drietal populaire methoden voor defuzzificatie besproken, alsmede hun betekenis voor de arbitrage en/ of combinatie van de uitvoer der primitieve gedragsmodules. Voor het gemak volgt hier een kort resumé:

- Center of Area (CoA) defuzzificatie – nauwkeurig berekend compromis;
- Center of Maximum (CoM) defuzzificatie – globaal compromis;
- Mean of Maximum (MoM) defuzzificatie – absolute keuze voor één uitvoer.

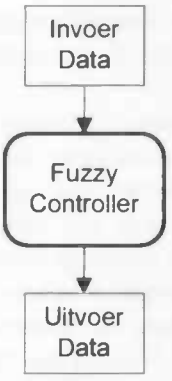
Voor de precieze berekeningswijzen van de methoden wordt verwezen naar hoofdstuk 2. Het doel is om (beredeneerd) experimenteel vast te stellen welke defuzzificatiemethoden tot de beste resultaten leiden. Het is hierbij zaak om een passende balans tussen democratie en besluitvaardigheid te vinden.
Tot slot dient het gedragsysteem functionaliteit te bevatten om vaste reeksen van gedrag, zoals bijvoorbeeld het inhalen, te kunnen uitvoeren. Bij de evaluatie wordt met het oog op de implementatie van deze functionaliteit gelet op de volgende aspecten:

- *compatibiliteit* met de gebruikte hiërarchische structuur;
- correctheid van *volgorde en timing* van de onderdelen van de gedragsreeks;
- *flexibiliteit* indien de gedragsreeks bijvoorbeeld halverwege afgebroken moet worden.

Strategische gewichten en rijstijl
Een klein aantal gewichten, bepaald op het strategisch niveau, bepaalt de rijstijl van FuzzyDriver. De uit te voeren (rij)taak blijft gelijk, maar de wijze waarop deze uitgevoerd wordt kan met deze gewichten gevarieerd worden. Het komt de beoordeling ten goede als met een klein aantal gewichten

er een hoge mate van controle is over de rijstijl en dat bovendien het hierdoor beschikbare spectrum aan stijlen breed is. Voor dit onderzoek wordt het strategisch niveau op zichzelf niet verder uitgewerkt: de strategische gewichten worden a priori ingegeven.

3.6.4 Prestaties



Figuur 3.8 – Open loop systeem
Schematische weergave van een fuzzy controller in een open loop systeem. Zoals te zien is er geen mogelijkheid tot manipulatie van de omgeving of het verkrijgen van feedback daaruit.

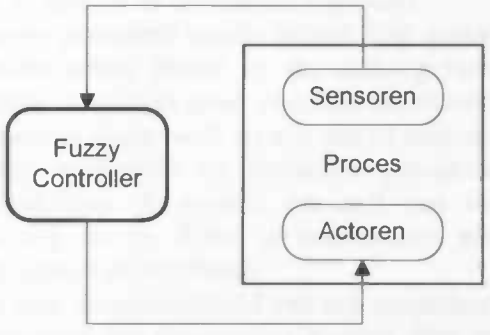
Wellicht het belangrijkst voor de evaluatie zijn de uiteindelijke prestaties van het systeem. We kunnen die prestaties bekijken in termen van de systeemrespons, de taak-uitvoering en de rekenlast. Hiertoe wordt een viertal methoden aangewend, namelijk:

1. het doen van metingen aan Fuzzy-Driver als open loop systeem (de systeemrespons);
2. het doen van metingen aan FuzzyDriver als closed loop systeem, met behulp van de simulatie (taakuitvoering);
3. het observeren van het gedrag van FuzzyDriver in de simulatie, door experts (taakuitvoering);
4. het opmeten van de benodigde tijd voor het doorrekenen van het systeem en het bepalen van de minimale frequentie waarmee het systeem doorgerekend moet worden (rekenlast).

Deze vier methoden worden nu één voor één verder toegelicht.

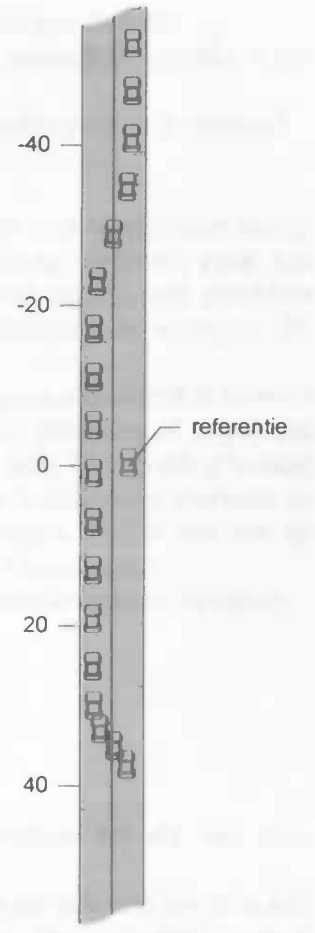
Beschrijving van de systeemrespons (open loop)
Het is mogelijk om FuzzyDriver lost van de simulatie te testen als zogenaamd 'open loop' systeem. Hiermee wordt de systeemrespons bepaald, door rechtstreeks invoerdata aan de fuzzy controller aan te bieden en vervolgens de daaruit volgende uitvoer uit te lezen (zie figuur 3.8). Door in de open loop situatie systematisch bepaalde invoervariabelen constant te houden en andere te variëren kunnen de elementaire gedragingen in beeld worden gebracht. De relevante uitvoer wordt hiertoe in een grafiek uitgezet tegen de gevarieerde invoervariabele(n).
FuzzyTECH heeft namelijk de mogelijkheid om grote hoeveelheden invoerdata in te lezen uit een tekstbestand. Het systeem wordt dan achtereenvolgend doorgerekend met alle daarin opgenomen sets van invoerwaarden. De bijbehorende uitvoerwaarden worden dan aan die sets toegevoegd en als geheel weggeschreven naar een uitvoer-tekstbestand, aan de hand waarvan de genoemde grafieken gemaakt kunnen worden.

Beschrijving van het bestuurdersgedrag (closed loop)
In hoofdstuk 2 werd de architectuur van een fuzzy controller beschreven. Figuur 2.4 gaf aan hoe de fuzzy controller via sensoren en actoren interacteert met een proces. Voor het gemak is deze figuur hier herhaald in een vereenvoudigde versie (zie figuur 3.9). Hierin is te zien dat het geheel een gesloten lus vormt: men spreekt dan van een 'closed loop' systeem. In het geval van FuzzyDriver wordt het proces waarmee de interactie plaatsvindt gevormd door de simulatie van het voertuig in een dynamische verkeerssituatie. De actoren zijn rem, gas en stuur. De sensoren nemen de toestand van het eigen voertuig en de verkeerssituatie waar.
Het complexere gedrag van FuzzyDriver kan geobserveerd en gemeten worden in deze closed loop situatie, door FuzzyDriver middels de simulatie in diverse aangemaakte verkeersscenario's te plaatsen. In deze scenario's komen typische verkeershandelingen zoals het afstand houden, het inhalen en het nemen van een afslag, eventueel in combinatie aan bod. Daarnaast zal FuzzyDriver in ongewone situaties worden gezet



Figuur 3.9 – Closed loop systeem
Vereenvoudigde weergave van figuur 2.4. De fuzzy controller beinvloedt het proces via actoren en verzameld feedback over de procesbesturing via sensoren.

en zullen er ook scenario's worden gebruikt met een startsituatie die van FuzzyDriver direct actie vereist. Door af te wijken van de ideale situatie zal blijken of het systeem robuust is.
Het gedrag dat FuzzyDriver in de diverse scenario's vertoont kunnen we meten en in kaart brengen. Met behulp van grafieken kan het verloop van relevante variabelen over de tijd worden weergegeven. Ook het afgelegde traject kan middels een speciale figuur in beeld worden gebracht: hiertoe heb ik gebruik gemaakt van een stroboscopische weergave. Deze weergave wordt verkregen door tijdens de simulatie telkens na een vast tijdsinterval de positie en oriëntatie van het voertuig weg te schrijven naar een bestand. Al deze posities worden geschaald en op deze geschaalde posities wordt een autootje geplot in de uiteindelijke figuur. Eén en ander resulteert dan in een plaatje zoals figuur 3.10. Vaak is het handig met de relatieve positie van het voertuig ten opzichte van een ander voertuig te werken. Het andere voertuig vormt dan een referentiepunt dat een vaste plek heeft in de figuur. Zo wordt het traject van het eigen voertuig (en eventueel overige voertuigen) weergegeven *ten opzichte van* het referentievoertuig. In de resultaten zal het nut hiervan verder blijken.



Figuur 3.10 – Voorbeeld van een stroboscopische weergave van het voertuigtraject
De stroboscopische weergave wordt verkregen door telkens na een vast tijdsinterval de oriëntatie en de (relatieve) positie op te slaan.

Observatie van het bestuurdersgedrag door experts
Het gedrag van de kunstmatige autobestuurder kan uiteraard ook bestudeerd worden door de auto's in het gesimuleerde scenario te (laten) observeren. Eén of meerdere experts – ofwel autobestuurders – kunnen zo hun oordeel geven over het gedrag van de kunstmatige bestuurder in de verkeerssituatie op de gesimuleerde snelweg.
Voor het observeren en beoordelen van het gedrag van FuzzyDriver is de hulp van enkele externe experts ingeroepen, aangezien deze een meer frisse en objectieve kijk zullen hebben op het gedrag dan de ontwerper zelf.
Het simulatieprogramma is uitgerust met allerlei mogelijkheden om het gedrag van FuzzyDriver nauwkeurig te kunnen observeren. De camera volgt altijd een auto. Welke auto dit is kan tijdens de simulatie veranderd worden. Het camerastandpunt ten opzichte van de auto en de kijkrichting van de camera kan de gebruiker eveneens wijzigen met behulp van muis en toetsenbord. In een informatiescherm worden de gegevens over de toestand van de auto en de bestuurder afgebeeld.
Voor de beoordeling door de experts is nog een extra scenario gecreëerd waarin in een flink aantal, zeg twintig FuzzyDrivers met verschillende (doel)snelheden achter elkaar op de snelweg rijdt. Hiermee wordt de doorsnee situatie op een snelweg in de echte wereld zo goed mogelijk nagebootst. De simulatie wordt met een bepaald scenario zonodig meermalen uitgevoerd, terwijl één of meerder experts toekijken en kennis geven van hun observaties aangaande de hieronder genoemde beoordelingscriteria. Bovendien bestaat er de mogelijkheid voor de experts om zelf met het toetsenbord één van de auto's in de simulatie te besturen en zo direct te interacteren met FuzzyDriver(s). De observaties worden verzameld en opgenomen in de resultaten.
In hoofdstuk 1, figuur 1.1 zijn de criteria genoemd waarop men de geschiktheid van een modelleermethode kan beoordelen. Een aantal van deze criteria aangaande het eindproduct kunnen door de experts worden beoordeeld. Met name zijn dit:

1. Normatief gedrag – Van een autobestuurder op een snelweg wordt een bepaald gedrag verwacht, vertoont de kunstmatige bestuurder dit gedrag?

2. Veiligheid – Kan het gedrag van de kunstmatige bestuurder als optimaal veilig beschouwd worden als hij daarvoor geconfigureerd is?
3. Efficiëntie – is alle gedrag gericht op en toereikend voor het uitvoeren van de taak?
4. Natuurlijk gedrag – worden de verkeershandelingen op een natuurlijk ogende wijze uitgevoerd?
5. Differentiatie van rijstijlen – is het mogelijk om zichtbaar verschillende rijstijlen te creëren?

Differentiatie van rijstijlen
Voor de beoordeling op het laatstgenoemde criterium is een wat uitgebreidere onderzoeksopzet nodig. Er is nu niet langer meer sprake van één FuzzyDriver die het modale gedrag vertoond, maar van meerdere versies, elk geconfigureerd om een andere rijstijl te hanteren. Deze configuratie geschiedt door de waarden van de vier strategische gewichten in onderlinge verhouding te wijzigen. De verschillende versies worden hierna 'profielen' genoemd.
De experts noemen we nu voor het gemak proefpersonen. Dit kunnen overigens niet dezelfde mensen zijn die de vorige observaties hebben gemaakt, aangezien zij het modale profielen al uitgebreid hebben leren kennen. De proefpersonen krijgen alle bestuurdersprofielen te zien. Hen wordt gevraagd het gedrag van elk van de profielen te observeren. Ze krijgen van te voren te horen welke profielen ze kunnen verwachten, maar niet in welke volgorde ze die gepresenteerd krijgen. Het is dan aan de proefpersonen om de profielen aan de hand van het geobserveerde gedrag te identificeren.
In totaal zal een viertal bestuurdersprofielen worden aangemaakt met de volgende typische rijstijlen:

1. Profiel A – Modaal/ Normatief
2. Profiel B – Overdreven voorzichtig
3. Profiel C – Asociaal
4. Profiel D – Snel en roekeloos

In figuur 3.11 is weergegeven welke configuratie van strategische gewichten bij elk van deze profielen hoort.
Profiel A vertegenwoordigt de standaardconfiguratie van FuzzyDriver, zoals die gebruikt wordt in alle overige experimenten en metingen. De rijstijl is doorsnee, 'zoals het hoort', dus veiligheid voorop; daarna de verkeersregels; dan het volgen van de route en pas als laatste zo vlot mogelijk rijden. Binnen deze prioritering wordt elk doel zo goed mogelijk behartigd.
Voor profiel B is de onderlinge prioritering van de doelen hetzelfde, alleen is het gewicht voor 'ga snel' met 0,2 verlaagd en het gewicht voor 'volg regels' met 0,2 verhoogd. Hiermee wordt een overdreven 'brave' en voorzichtige rijstijl beoogd.
Profiel C is weer identiek aan A, maar nu is het 'ga snel' gewicht sterk verhoogd, naar 0,8. Hierdoor zal 'ga snel' vaker 'volg regels' overschaduwen en bovendien sterker concurreren met 'bots niet', waardoor een meer asociale rijstijl ontstaat.
Profiel D is gelijk aan profiel C, maar dan zijn ook nog eens de gewichten voor 'volg regels' en 'bots niet' verlaagd, namelijk naar 0,8 en 0,2, respectievelijk. Dit heeft al met al een buitengewoon onstuimige en gevaarlijke rijstijl tot gevolg.

	Strategische gewichten			
Bestuurdersprofiel	**Bots niet**	**Volg regels**	**Volg route**	**Ga snel**
Profiel A – Modaal/ Normatief	1,0	0,6	0,5	0,4
Profiel B – Overdreven voorzichtig	1,0	0,8	0,5	0,2
Profiel C – Asociaal	1,0	0,6	0,5	0,8
Profiel D – Snel en roekeloos	0,8	0,2	0,5	0,8

Figuur 3.11 – Bestuurdersprofielen
In deze tabel staan de vier bestuurdersprofielen gepresenteerd. Voor elk van de profielen is aangegeven welke configuratie van de strategische gewichten zij vertegenwoordigen. Zoals te zien in de tabel is het 'volg route' gewicht niet gevarieerd. De reden hiervoor is dat er zich in de simulatie maar één afslag bevindt, waar dit gewicht tot uitdrukking kan komen. Op basis daarvan is er te weinig mogelijkheid tot differentiatie tussen profielen met verschillende waarden voor dit gewicht.

Rekenlast

In hoofdstuk 2 is 'computationele efficiëntie' opgevoerd als maatstaaf om modellen voor bestuurdersgedrag te waarderen en vergelijken. Daarom zullen in de resultaten de doorrekentijd en de specificaties van het testsysteem vermeld worden (in de literatuur heb ik overigens helaas geen getalsmatig vergelijkingsmateriaal gevonden).

Voor het bepalen van de rekenlast is tevens van belang *hoe vaak* het systeem doorgerekend moet worden. Het geïmplementeerde gedragsysteem is namelijk een discrete benadering van de bestuurderstaak, die in ieder geval gedeeltelijk continue is [8]. De frequentie f waarmee het systeem aangeroepen wordt, bepaald de frametijd $T = t/f$. Hoe kleiner deze frametijd is, hoe nauwkeuriger het continue proces benaderd wordt.

T_{max} is de maximale frametijd (laagste frequentie) waarbij het proces nog voldoende goed simuleren. Deze waarde zal voor FuzzyDriver bepaald worden door de frametijd stapsgewijs op te voeren en het gedrag te observeren. Idealiter ligt deze T_{max} zo hoog mogelijk. Als het systeem minder vaak doorgerekend hoeft te worden, is er meer rekentijd over voor eventuele overige processen. Een ideaal systeem is bovendien in zijn functioneren onafhankelijk van de frametijd. Dat wil zeggen dat voor iedere frequentie met een bijbehorende frametijd tussen 0 en T_{max} het systeem hetzelfde gedrag produceert.

4 Resultaten

4.1 Inleiding

In dit hoofdstuk beschrijf ik mijn bevindingen bij het vervaardigen en testen van FuzzyDriver. Een gedetailleerde beschrijving van het uiteindelijke hiërarchisch fuzzy gedragssysteem is opgenomen als bijlage achterin dit verslag. In de volgende paragrafen worden achtereenvolgens het ontwerpproces, de implementatiedetails en de prestaties besproken.

4.2 Ontwerpproces

4.2.1 Inleiding

De onderstaande resultaten – aangaande het ontwerpproces – beschrijven ten eerst de bevindingen wat betreft de kosten van het vervaardigen van het hiërarchisch fuzzy gedragsysteem. Hieruit zal blijken dat de meeste tijd zit in het opzetten van de architectuur.
Aan het eind wordt geïllustreerd dat de methode zich leent voor een incrementeel ontwerpproces, aan de hand van experimenten met het telkens uitbreiden van het systeem met meer functionaliteit.

4.2.2 Ontwerpkosten

Opzetten van de architectuur
Zoals gezegd zit de meeste tijd zit in bedenken van de algemene controle-structuur van het hiërarchisch systeem. Dit is overigens een eenmalige tijdsinvestering.
De globale structuur wordt reeds gedeeltelijk bepaald door de toepassing van het drie taakniveaus-paradigma en het hiërarchische karakter van de behavior-based benadering.
Er is sprake van een wisselwerking tussen de kennisacquisitie en het opzetten van de architectuur. Enerzijds geeft de vergaarde expertkennis verder inzicht in de structuur van de te implementeren rijtaak, anderzijds kan aan de hand van deze globale structuur richting gegeven worden aan de kenniselicitatie. Het is de kunst om uit deze kennis zo weinig mogelijk verschillende elementaire behaviors te extraheren. Hiernaar dient actief te worden gezocht. Het gaat om zoeken naar gemene delers. Verder is door kennis van de typische tijdspanne van cognitieve processen en handelingen van de diverse taakniveaus (zie hoofdstuk 2) goed te bepalen of bepaalde beschreven taken nog opgedeeld moeten worden in meer elementaire onderdelen. Het hiërarchische denken maakt het makkelijker te zien waar specifieke taken omgezet kunnen worden naar generieke, of juist opgebouwd kunnen worden uit reeds beschreven processen.

Kennisacquisitie
Omdat het gedragsysteem werkt met heuristieken die rechtstreeks afgeleid zijn van de expertkennis, is het noodzakelijk om eerst de expertkennis van een automobilist te eliciteren.
De expertkennis is in dit geval noch schaars, noch kostbaar. In de westerse maatschappij is de autobestuurderstaak een zeer alledaagse taak. Ter illustratie: In 2002 waren er in Nederland op 16 miljoen inwoners circa 9,8 miljoen mensen in het bezit van het rijbewijs B en waren er zo'n 6,8 miljoen auto's (bron: http://statline.cbs.nl). Bovendien is in het geval van dit onderzoek de ontwerper van het systeem tevens expert; een situatie die uiteraard ook in de verdere praktijk allesbehalve ondenkbaar is.
Er is enige inspanning nodig om de verkeershandelingen, zoals bijvoorbeeld inhalen, terug te brengen tot de elementaire handelingen, zoals 'ga een baan naar links', 'accelereer' en 'ga naar rechts'. Je ziet

overigens al snel dezelfde deelhandelingen terugkomen in meerdere complexe acties (denk aan inhalen en afslag nemen).

Van expertkennis naar gedragsysteem
Fuzzy logic werkt met linguïstische variabelen. Daardoor is de stap van expertkennis naar fuzzy logic productieregels klein en eenvoudig te maken. Het samenstellen van het totale bestand aan productieregels was een relatief eenvoudige taak. In bijlage A zijn al deze regels opgenomen, zowel in natuurlijke taal (expertkennis) als in de vorm van productieregels zoals die gebruikt zijn in het fuzzy systeem. Wie dat overzicht bestudeert ziet al snel de sterke analogie tussen beide representaties.

Afstelling van membershipfuncties en parameters
In hoofdstuk 3 is beschreven hoe men tijdens de simulatie de activatieniveau's van de fuzzy sets en regels kan bekijken en de membershipfuncties van de fuzzy sets kan bijstellen. Dit maakt het afstellen van het systeem tot een zeer eenvoudige taak, waarvoor de benodigde tijd vrijwel verwaarloosbaar is in relatie tot de tijd die nodig is voor het initiële ontwerp.
Wel kwam het voor dat het systeem specifiek werd afgesteld voor een bepaalde deeltaak, waarmee de afstelling voor gedrag op ander gebied ontregeld werd.

4.2.3 Incrementeel ontwerpproces

Het is mogelijk gebleken om met één primitieve gedragsmodule te beginnen en daarna steeds één of meerdere toe te voegen, alsmede eventuele extra benodigde coördinerende meta-regels, terwijl er in iedere fase sprake is van een werkend prototype. Gedurende de ontwikkeling van de kunstmatige autobestuurder waren de volgende fasen te onderscheiden:

1. De huidige rijstrook en de gewenste snelheid werden aangehouden.
2. Toegevoegd: afstand houden tot voorligger.
3. Toegevoegd: kiezen van een rijstrook voor het inhalen
4. Toegevoegd: mogelijkheid tot afstand houden tot voorligger per rijstrook
5. Toegevoegd: meta-regels voor het volgen van de verkeersregels bij het inhalen
6. Toegevoegd: meta-regels voor het kiezen van een rijstrook voor het nemen van een afslag. De implementatie is compleet.

4.3 Implementatie

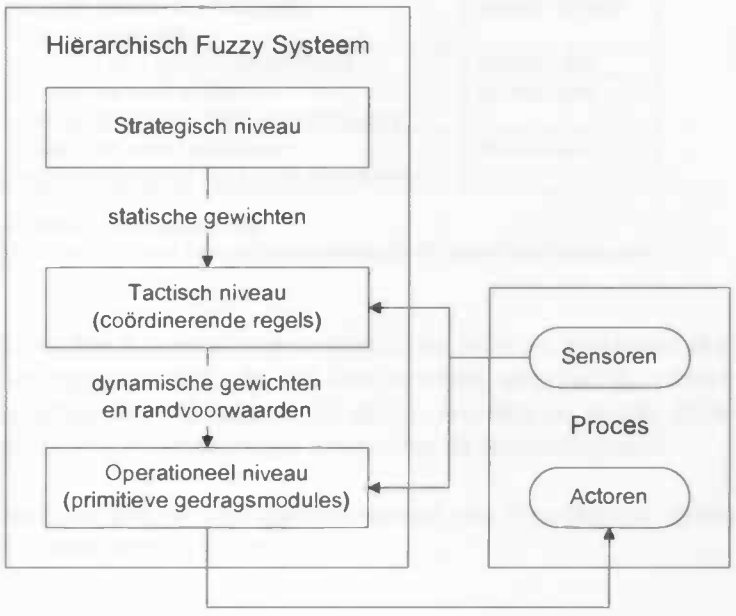
4.3.1 Algemene architectuur

In deze paragraaf wordt de algemene opzet van de implementatie beschreven. De lezer wordt voor een gedetailleerd overzicht allereerst verwezen naar bijlage A. Hierin is een schematische weergave van het fuzzy systeem opgenomen, alsmede alle productieregels en alle in- en uitvoervariabelen met bijbehorende membership functies. De algemene opzet van het hiërarchische fuzzy systeem is schematisch weergegeven in figuur 4.1.

Lagen van het systeem
Het systeem bestaat uit drie lagen. De eerste en onderste laag bevat de zogenaamde primitieve gedragsmodules. De tweede en middelste laag bevat regels die de primitieve gedragsmodules coördineren. De derde en bovenste laag bevat doelen, waarvan de onderlinge prioritering de coördinerende regels in de tweede laag beïnvloedt. Deze drie lagen komen overeen met respectievelijk het operationele, tactische en strategische niveau uit het paradigma van Michon (zie hoofdstuk 2).

Feedback en feedforward tussen de lagen

Het systeem werkt in directe zin enkel met feedforward. De informatiestroom gaat dus alleen omlaag in de hiërarchische structuur. De strategische laag levert wegingsfactoren op voor de regels in de tactische laag. De tactische laag levert wegingsfactoren op voor de uitvoer van de primitieve gedragsmodules op het operationele niveaus, alsmede eventuele randvoorwaarden voor deze modules. Van directe feed*back* is geen sprake, hoewel er op het tactische en operationele niveau wel informatie uit de omgeving (situatie) wordt gebruikt, die weer onder directe invloed staat van het operationele niveau. In die zin wordt er gebruik gemaakt van *indirecte* feedback. In figuur 4.2 is in formulevorm samengevat welke feedback en -forward er plaatsvindt.



Figuur 4.1 – Algemene architectuur van het hiërarchische fuzzy systeem
De gebruikte architectuur is hier in algemene zin schematische weergegeven. Aan de pijlen is te zien dat er binnen het systeem alleen feedforward plaatsvindt. Ook geven de pijlen aan dat er zowel op het tactische als het operationele niveau gebruik wordt gemaakt van informatie uit de omgeving. Aangezien deze omgeving weer onder invloed staat van het fuzzy systeem, kan dit gezien worden als indirecte feedback.

4.3.2 Variabelen en membership functies

De in- en uitvoervariabelen zijn gedefinieerd aan de hand van membershipfuncties. In bijlage A is te zien dat elk van de 21 fuzzy variabelen gedefinieerd is aan de hand van 2 á 3 membershipfuncties (fuzzy set definities). In de eerste instanties zijn 5 membershipfuncties per variabele gebruikt in verband met de precisie, maar dat bleken er overbodig veel te zijn. Door de interpolerende werking van het defuzzificeren voldoet het kleinere aantal. Voor de membershipfuncties zijn uitsluitend lineaire functies gebruikt. Dit zijn namelijk de eenvoudigste type functies en het blijkt dat zij volstaan.

De definities voor de membershipfuncties zijn in de eerste instantie aan de hand van schattingen gemaakt. Om niet voor elke bijstelling van een variabele 2 á 3 functies te moeten aanpassen, is er gebruikt gemaakt van lineaire transformaties via de C++ omgeving. Men zou er ook voor kunnen kiezen om voor de overzichtelijkheid de membershipfuncties volledig te normaliseren en de vertaling naar de praktijkwaarden volledig via de lineaire transformaties te laten plaatsvinden. De membershipfuncties dienen dan al wel in onderlinge relatie correct te zijn gedefinieerd.

strategisch_gewicht X situatie → dynamisch gewicht
strategisch_gewicht X situatie → randvoorwaarde

randvoorwaarde X situatie → voorlopige_uitvoer

voorlopige_uitvoeren X dynamisch_gewichten → einduitvoer

einduitvoer → situatie

Figuur 4.2 – Feedforward en feedback in het hiërarchische fuzzy systeem
Elke feedback- en feedforwardstap die plaatsvindt is hier in formulevorm weergegeven. De feedforward vindt plaats via dynamische gewichten en randvoorwaarden. De feedback vindt plaats via de omgevingssituatie

4.3.3 Primitieve gedragsmodules

In de schematische weergave van het fuzzy systeem van bijlage A zijn in totaal 4 primitieve gedragsmodules te zien. Voor het gemak zijn ze in figuur 4.3 nog een keer opgesomd. Het zijn uiterst eenvoudige fuzzy

behavior	invoer	Uitvoer
volg huidige rijstrook	laterale positie tov middellijn laterale snelheid	stuursnelheid
houd gewenste snelheid aan	afwijking van gewenste snelheid	acceleratie
volg auto linksvoor	afstand tot auto linksvoor snelheidsverschil met auto linksvoor	acceleratie
volg auto rechtsvoor	afstand tot auto rechtsvoor snelheidsverschil met auto rechtsvoor	acceleratie

Figuur 4.3 – De vier primitieve gedragsmodules van FuzzyDriver
Deze vier eenvoudige regelsystemen bepalen aan de hand van de verkeerssituatie de gewenste acceleratie en stuursnelheid.

regelsystemen, met elk één uitvoervariabele, één a twee invoervariabelen en twee tot maximaal drie productieregels. Dit zijn overigens wel de enige modules in het hele systeem waarvan de uitvoer direct betrekking heeft op de voertuigbesturing, dat wil zeggen gas geven, remmen en sturen. Deze vier simpele gedragsmodules vertegenwoordigen het operationele niveau van de bestuurderstaak.

De bijdrage van elke primitieve gedragsmodule aan de uiteindelijke uitvoer van FuzzyDriver wordt dynamisch bepaald door de bovenliggende coördinerende laag.

4.3.4 Gedragscoördinatie

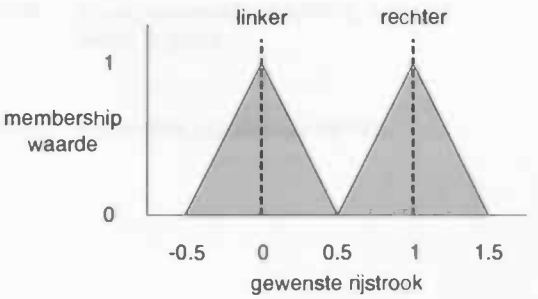
Productieregels
Een verzameling van 13 productieregels coördineert de primitieve gedragsmodules. Deze regels bevinden zich op het tactische niveau van de bestuurderstaak. De regels zijn ondergebracht in twee blokken, één voor het coördineren van de acceleratie-uitvoer (4 regels) en één voor het coördineren van de stuur-uitvoer (9 regels). Aan de invoerkant vinden we de strategische gewichten en omgevingsvariabelen. Aan de uitvoerzijde vinden we de gewichten waarmee de uitvoerwaardes van de primitieve gedragsmodules gewogen worden, alsmede eventuele randvoorwaarden voor die modules. Met behulp van de gewichten wordt de uitvoer van de primitieve gedragsmodules gecombineerd tot één waarde voor acceleratie en één voor stuursnelheid[3]. FuzzyDriver schept op tactisch niveau momenteel slechts op één plek een randvoorwaarde voor het operationele niveau, namelijk de gewenste rijstrook waar de module 'volg huidige rijstrook' van uit gaat.

Samenname en defuzzificatie van de uitvoer
Alle behaviors doen een gedragsvoorstel, ofwel ze geven een uitvoerwaarde voor acceleratie of stuursnelheid. Al deze uitvoeren worden gewogen samengenomen. Als ze niet tegenstrijdig zijn, dan zullen deze gedragingen elkaar alleen maar versterken. Als ze wel strijdig zijn, dan bepalen de dynamische gewichten welke gedragsvoorstellen het meest het uiteindelijke gedrag vormen. Dit is een gevolg van het gebruik van de 'center-of-area' defuzzificatiemethode. Deze methode is toegelicht in hoofdstuk 2. CoA-defuzzificatie faciliteert de specifieke en vaak gewenste eigenschap van fuzzy regelsystemen dat de uitvoerwaarde over het gehele spectrum van de uitvoervariabele geïnterpoleerd wordt. Dit zorgt voor de typische 'smooth' gedragsovergangen van fuzzy regelsystemen. Voor de uitvoervariabelen 'acceleratie' en 'stuursnelheid' leidt dit tot de gewenste resultaten.

Kiezen van de gewenste rijstrook
De keuze voor de gewenste rijstrook dient eenduidig te zijn. Er is (juist) geen behoefte aan interpolatie of 'smooth' overgangen: je kiest of de linker of de rechter rijstrook. Door gebruik te maken van 'mean of maximum' (MoM) defuzzificatie wordt precies dit effect verkregen. Deze

[3] Als gevolg van de afbakening van de bestuurderstaak voor dit onderzoek bevat FuzzyDriver slechts één primitieve module voor het bepalen van de stuursnelheid. Wegen en combineren is daarom voor deze uitvoer niet noodzakelijk. Voor een uitgebreider systeem zou hier echter ook hetzelfde principe kunnen worden toegepast.

methode geeft als uitvoer de gemiddelde waarde van de fuzzy set die het sterkst vuurt. Met behulp van de membershipfuncties voor de variabele 'gewenste rijstrook' zoals in figuur 4.4 wordt er zo altijd hard een keuze gemaakt tussen de linker (0) of rechter rijstrook (1).



Figuur 4.4 – Definitie van de fuzzy sets voor de variabele 'gewenste rijstrook'
In deze variabele zijn twee fuzzy sets gedefinieerd, namelijk 'linker' en 'rechter'. Bij de definitie is er op gelet dat de gemiddelde waarden van de sets 0 respectievelijk 1 zijn. De variabele levert nu na MoM-defuzzificatie altijd de waarde 0 of 1 op. De waarde 0 staat hierbij voor de linker en 1 voor de rechter rijstrook.

Een technisch detail: variabelen die in fuzzyTECH doorgegeven worden tussen regelblokken – zogenaamde 'intermediate variables' – worden niet geduzzificeerd. Om de 'volg rijstrook' module toch met een eenduidig gekozen rijstrook te laten werken heb ik met succes de volgende 'hack' toegepast: In het simulatieprogramma wordt uit de uitvoer 'gewenste rijstrook' en de 'laterale positie' van de auto op de weg de 'laterale afstand ten opzichte van de middellijn' bepaald. Deze laatste waarde wordt als invoer voor de primitieve gedragsmodule 'volg rijstrook' aan het fuzzy systeem aangeboden. Deze lus is in de schematische weergave in bijlage A tevens aangegeven.

De menselijke autobestuurder rijdt door de bank genomen zo constant mogelijk. Als hij bijvoorbeeld voor een bepaalde rijstrook heeft gekozen, zal hij geneigd zijn daar op te blijven. Dit heeft te maken met een algemeen menselijk fenomeen dat een brede bekendheid heeft in de psychologie: de mens heeft de neiging om aan zijn gemaakte beslissingen vast te houden, zelfs al zijn er inmiddels betere alternatieven voorhanden [6]. Om ook FuzzyDriver spaarzaam te laten zijn met het aantal rijstrookwisselingen, is een tweetal regels toegevoegd aan de 'kies rijstrook' module die ervoor zorgen dat de huidige rijstrook aangehouden wordt:

ALS *huidige_rijstrook* is *linker* DAN *gewenste_rijstrook* is *linker*
ALS *huidige_rijstrook* is *rechter* DAN *gewenste_rijstrook* is *rechter*

Deze regels creëren als het ware een drempel voor het overgaan tot wisselen van rijstrook. Deze drempel mag echter niet te hoog zijn, anders worden noodzakelijke rijstrookwisselingen niet of niet tijdig uitgevoerd. De 'degree of support' van deze regels staat daarom op tien procent van de overige regels. Het toevoegen van deze regels heeft geleid tot een natuurlijker ogend rijgedrag: de rijstrookwisselingen worden doorgaand volledig doorgezet, ze zijn laag-frequent en met duidelijke reden.

Sequentieel gedrag
Deels tijdens de kennisacquisitie en deels tijdens de implementatie is gebleken dat de gedrag-sequenties – die besproken zijn in hoofdstuk 3 – helemaal niet als zodanig expliciet voorgeschreven hoeven worden. Het inhalen bijvoorbeeld is een 'emerging property' van het gelijktijdig nastreven van de doelen 'voorkom botsingen', 'rij met de gewenste maximumsnelheid' en 'blijf standaard rechts rijden'. Als het nastreven van deze doelen geïmplementeerd is, volgt het inhalen hieruit vanzelf. Betreft het genoemde voorbeeld: de volgende regels op het tactische niveau zijn achtereenvolgens verantwoordelijk voor een standaard inhaalmanoeuvre:

1) Haal langzame auto's in:
ALS ga_snel_gewicht is groot EN
afstand_tot_auto_rechtsvoor is dichtbij EN
snelheid_auto_rechtsvoor is te_langzaam
DAN gewenste_rijstrook is linker

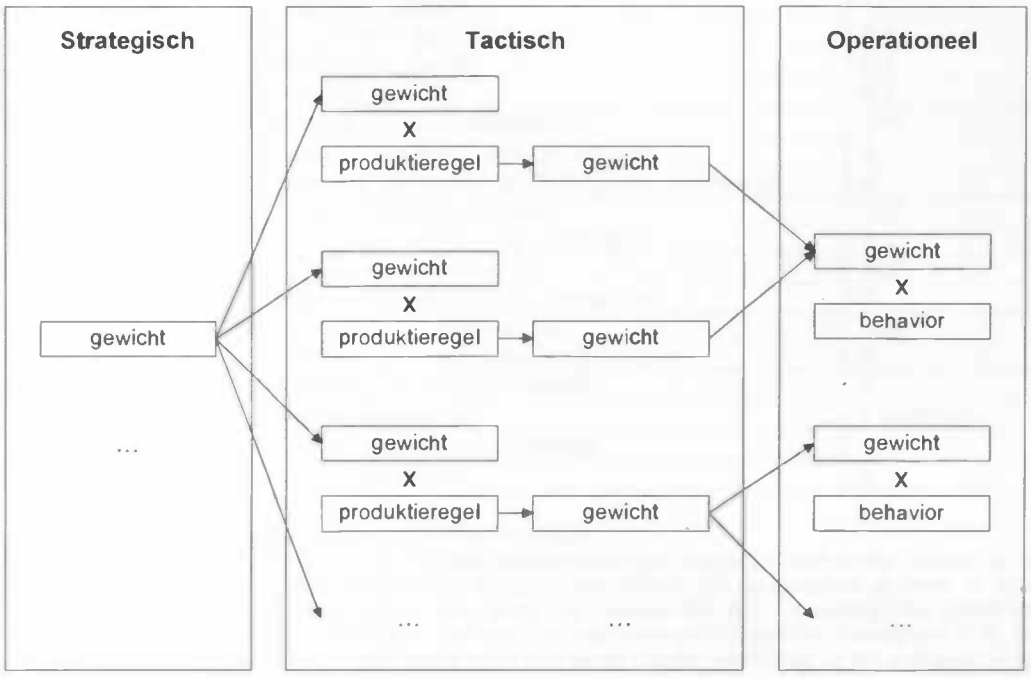
2) Rijd met de snelheid die je wenst:

ALS ga_snel_gewicht is groot **DAN** houd_gewenste_snelheid_aan_gewicht is groot

3) Blijf niet onnodig links rijden:
ALS volg_regels_gewicht is groot EN afstand_tot_auto_rechtsvoor is ver EN snelheid_auto_rechtsvoor is ok **DAN** gewenste_rijstrook is rechter

4.3.5 Strategische gewichten en rijstijl

Elke verkeershandeling dient zeer helder één van de vier strategische doelen. Dit vergemakkelijkt het implementeren en maakt de mate van controle over het gedrag via de strategische gewichten zeer groot. Aan elke coördinerende regel is dan ook één strategisch gewicht gekoppeld. Elke coördinerende regel bepaalt slechts het gewicht van één behavior. Het gewicht van een bepaalde behavior kan door meerdere coördinerende regels bepaald worden en dus indirect door meerdere strategische gewichten. Eén en ander is schematisch uitgebeeld in figuur 4.5. Aangezien er wordt gewerkt met wegingen is de *onderlinge verhouding* van de strategische gewichten uiteindelijk bepalend voor welk bijbehorend gedrag de overhand heeft.



Figuur 4.5 – Invloedstroom van strategische gewichten op het gedrag
Elke productieregel op het tactische niveau staat onder invloed van één strategisch gewicht en bepaald slechts het gewicht voor één operationele gedragsmodule. Eén strategisch gewicht kan meerdere tactische productieregels beïnvloeden en het gewicht van één operationele gedragsmodule kan door meerdere tactische productieregels bepaald worden.

De verdere bevindingen betreft de invloed van de strategische gewichten op de rijstijl worden in de loop van paragraaf 4.4 verder beschreven.

4.4 Prestaties

4.4.1 Inleiding

De prestaties van FuzzyDriver worden hieronder in kaart gebracht aan de hand van diverse waarnemingen. Achtereenvolgend worden de open loop metingen, de closed loop metingen, de observaties van de experts en ten slotte de rekenlast beschreven.

4.4.2 Beschrijving van de systeemrespons (open loop)

In deze paragraaf wordt de systeemrespons in kaart gebracht zoals die bepaald is door middel van open loop metingen (zie paragraaf 3.6). Telkens worden slechts één of twee invoervariabelen gevarieerd. De overige invoervariabelen hebben steeds een vaste standaardwaarde (zie de tabel in figuur 4.6). De waarden van de variërende in- en uitvoervariabelen worden steeds tegen elkaar uitgezet in een grafiek. Elke meting zal mede aan de hand van de verwachtingen besproken worden.

Strategische gewichten	bots_niet_gewicht	1
	volg_regels_gewicht	0.6
	volg_route_gewicht	0.5
	ga_snel_gewicht	0.4
Auto rechtsvoor	afstand_tot_auto_rechtsvoor	100
	relatieve_snelheid_auto_rechtsvoor	0
	afwijking_gewenste_snelheid_auto_rechtsvoor	0
Auto linksvoor	afstand_tot_auto_linksvoor	100
	relatieve_snelheid_auto_linksvoor	0
Auto linksachter	afstand_auto_linksachter	100
	relatieve_snelheid_auto_linksachter	0
Auto rechtsachter	afstand_auto_rechtsachter	100
	relatieve_snelheid_auto_rechtsachter	0
Eigen voertuig	afwijking_gewenste_snelheid	0
	huidige_rijstrook	1 (rechter)
	laterale_positie_tov_middellijn	0
	laterale_snelheid	0

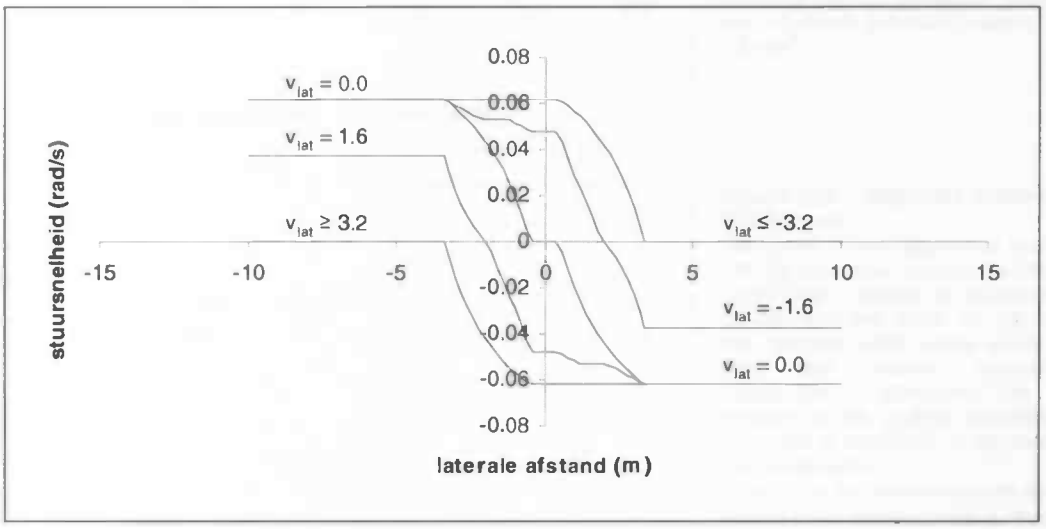
Figuur 4.6 – Standaard invoerwaarden voor de open loop metingen
De standaardwaarden zijn zo gekozen dat ze een situatie vertegenwoordigen waarin de auto netjes midden op de rechter rijstrook rijdt, zonder laterale snelheid en zonder verkeer om zich heen, rijdend met de gewenste snelheid. In deze situatie geeft FuzzyDriver geen stuur- of acceleratie-acties, daar deze niet noodzakelijk zijn. De strategische gewichten zijn zo ingesteld dat er sprake is van een modale bestuurder: het vermijden van botsingen is veruit het belangrijkst (1.0), daarna het volgen van de verkeersregels (0.6), dan het volgen van de route (0.5) en de laagste prioriteit ligt bij het snel rijden (0.4).

De metingen zijn gedaan aan de hand van de volgende vier elementaire gedragingen:

1. Rijstrook volgen
2. Gewenste snelheid aanhouden
3. Volg auto rechtsvoor
4. Volg auto linksvoor
5. Moment van inhalen

1. Rijstrook volgen
Wellicht de meest basale elementaire subtaak van FuzzyDriver is het volgen van de huidige rijstrook. Deze module wordt ook gebruikt om van rijstrook te wisselen. Voor deze taak worden de laterale positie (ten opzichte van het midden van de te volgen rijstrook) alsmede de laterale snelheid gemeten. Aan de hand van deze waarden wordt de stuursnelheid bepaald.

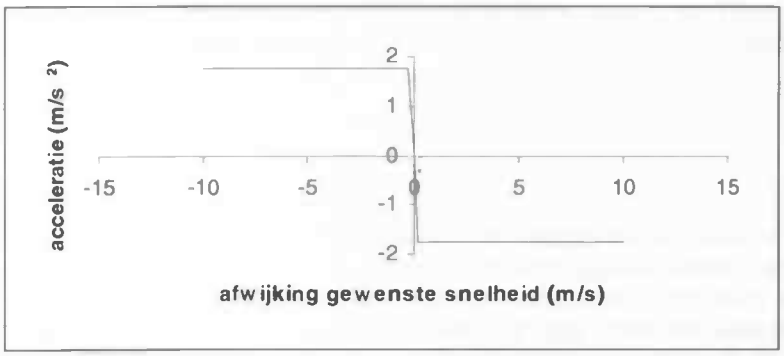
De stuursnelheid is in figuur 4.7 tegen de laterale positie uitgezet voor diverse laterale snelheden. Het resultaat spreekt voor zich: Naarmate FuzzyDriver zich meer links van het midden bevindt, wordt de stuursnelheid positief (naar rechts) en vice versa. Als het voertuig reeds een laterale snelheid in de goede richting heeft, dan is de stuurcorrectie minder hevig. Bevindt de auto zich ten slotte nagenoeg op de middellijn, maar heeft ze nog wel een laterale snelheid, dan is een tegengestelde stuurbeweging zichtbaar.



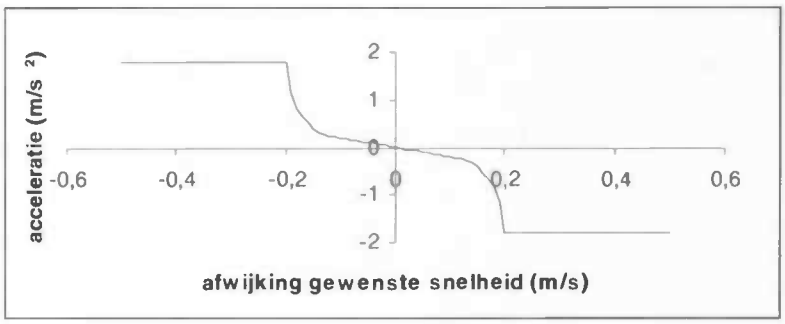
Figuur 4.7 –Stuursnelheid versus laterale positie tov middellijn
In deze grafiek is de stuursnelheid uitgezet tegen de laterale positie en opzichte van de middellijn, voor verschillende waarden van de laterale snelheid. De module 'volg huidige rijstrook' is verantwoordelijk voor deze respons.

2. Gewenste snelheid aanhouden

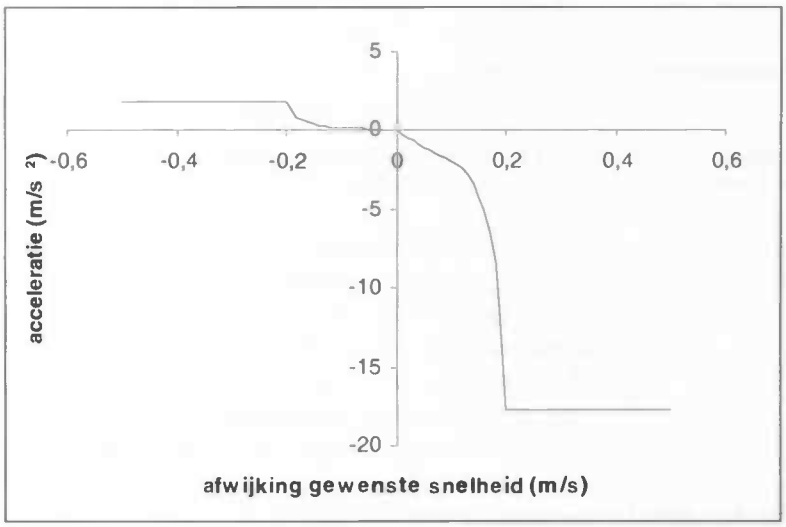
Als tweede bevat FuzzyDriver een meet- en regelsysteem voor het aanhouden van de gewenste snelheid. De gewenste snelheid is voor FuzzyDriver momenteel nog een statisch gegeven. Het systeem zorgt ervoor dat er afgeremd wordt als de huidige snelheid boven de gewenste snelheid ligt. Is de huidige snelheid daarentegen lager dan de gewenste, dan wordt er geaccelereerd. Als de snelheid ten slotte correct is gebeurt er niks. Dit gedrag wordt beschreven in de grafiek van figuur 4.8. In deze figuur lijkt het bijna een stapfunctie te betreffen, die bij het nulpunt van plus naar min 1.8 m/s^2 gaat. Dit is echter niet helemaal juist. Om dit te laten zien is de grafiek in figuur 4.9 sterk ingezoomd langs de x-as. Dan zien we dat de acceleratie nagenoeg nul is op en rond het nulpunt en dan verder exponentieel toe- dan wel afneemt bij een negatieve respectievelijk positieve snelheidsafwijking. Eén en ander resulteert in een soort s-vormige grafiek. Met name de sterke afzwakking rond het nulpunt voorkomt in theorie en ook in praktijk het optreden van oscillaties. Het bleek noodzakelijk om de negatieve acceleratie-uitvoer te schalen door vermenigvuldiging met een factor tien, omdat er vaak harder geremd moet worden dan opgetrokken. De uiteindelijke uitvoer aan het natuurkundig voertuigmodel is daarom zoals weergegeven in figuur 4.10.



Figuur 4.8 – Acceleratie versus afwijking gewenste snelheid
Als de huidige snelheid van het voertuig onder de gewenste snelheid ligt, dan geeft FuzzyDriver een positieve acceleratie en vice versa.

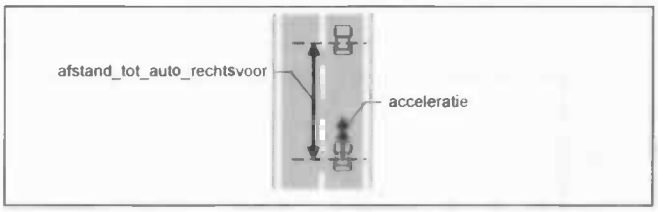


Figuur 4.9 – Acceleratie versus afwijking gewenste snelheid (x-as sterk ingezoomd)
Als we sterk inzoomen zien we wat er rond het nulpunt precies gebeurd. De acceleratie neemt exponentieel toe met de afwijking van de gewenste snelheid, tot een maximum (absolute) waarde van zo'n 1.8 m/s^2.



Figuur 4.10 – Eigenlijke acceleratie van FuzzyDriver
Het bleek in het algemeen noodzakelijk om de negatieve acceleratie-uitvoer van FuzzyDriver, oftewel het afremmen, op te schalen met een factor 10. Dit is op zich niet vreemd, want echte auto's kunnen ook veel sterker remmen dan accelereren[4]. Bovendien is sterker remmen in de praktijk noodzakelijk om botsingen te vermijden, in tegenstelling tot het accelereren.
Al met al is de systeemrespons dus feitelijk zoals weergegeven in deze figuur.

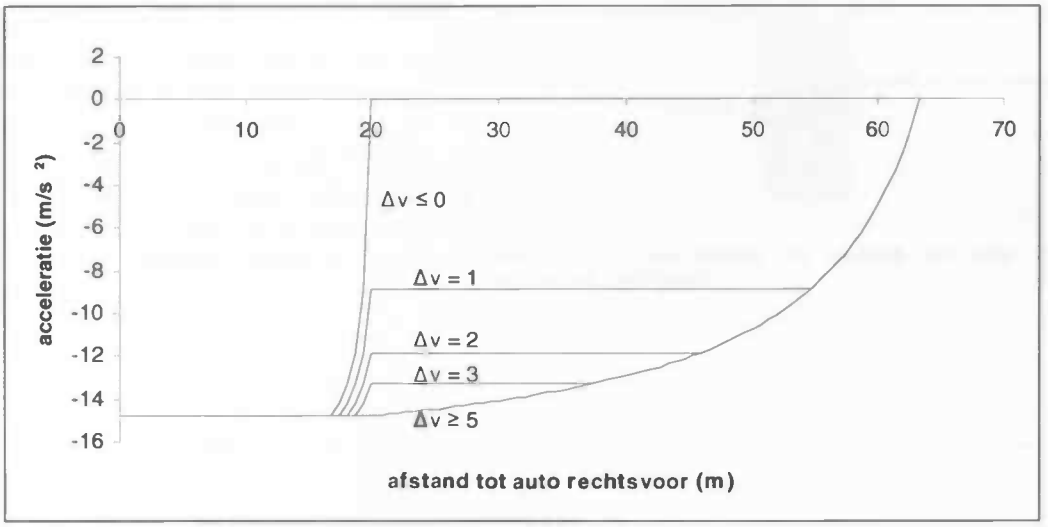
3. Volg auto rechtsvoor



Figuur 4.11 – Acceleratie en afstand tot auto rechtsvoor (FuzzyDriver rijdt rechts)

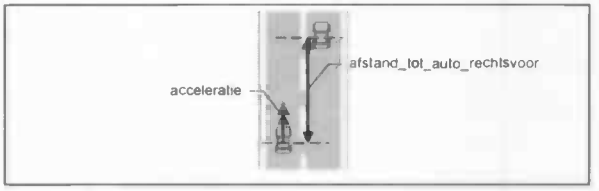
De modules voor het volgen van de auto linksvoor en rechtsvoor zijn in termen van membershipfuncties en productieregels volledig identiek. Afhankelijk van de afstand tot het te volgen voertuig en het snelheidsverschil daarmee leveren ze een bepaalde acceleratie-uitvoer op (nul of negatief). Door toedoen van de coördinerende laag is de einduitvoer van het gehele systeem in verschillende situaties telkens anders. In de eerste situatie wordt de auto rechtsvoor gevolgd, terwijl FuzzyDriver daarachter tevens op de rechter baan rijdt. Deze situatie is geschetst in figuur 4.11. De systeemuitvoer voor acceleratie is dan zoals beschreven in figuur 4.12. Daar zien we dat de auto bij een te kleine afstand altijd flink remt. Naarmate de afstand groter wordt neemt het remmen af. Deze afname verloopt sneller bij lagere snelheidsverschillen. Wel is er bij deze afname sprake van een tussentijds plateau.

[4] Het opschalen van de deceleratie resulteert in waarden tot maximaal zo'n 18 m/s^2. Het dient vermeld te worden dat dergelijke waarden niet realistisch zijn. In werkelijkheid kunnen auto's maximaal met ongeveer 1g oftewel zo'n 10 m/s^2 remmen. Door middel van fine-tuning van FuzzyDriver en/of de simulatie is het waarschijnlijk mogelijk tot realistischer waarden te komen. Dit vergt echter een aardige tijdsinvestering, terwijl het onderzoek er niet in grote mate mee gedient wordt. Een dergelijke aanpassing is daarom achterwege gelaten

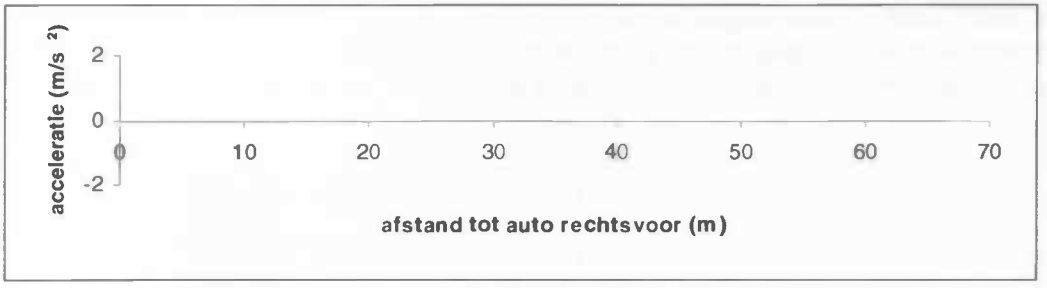


Figuur 4.12 – Acceleratie versus afstand tot auto rechtsvoor (rechter baan)
Naarmate het snelheidsverschil groter is, wordt er eerder – dus bij een grotere afstand – geremd.

De tweede mogelijke situatie voor het volgen van de auto rechtsvoor is die waarbij FuzzyDriver op de linkerbaan rijdt. Deze situatie is weergegeven in figuur 4.13. Aangezien er nu echter geen botsingsgevaar is en het bovendien toegestaan is een auto links in te halen, wordt de uitvoer van de module volledig onderdrukt door de coördinerende systeemlaag. De acceleratie-uitvoer van het systeem als geheel is dus onafhankelijk van afstand en snelheidsverschil altijd nul. Voor de volledigheid is ook deze uitvoer weergegeven in figuur 4.14.



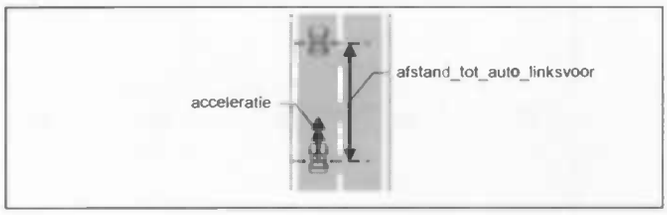
Figuur 4.13 – Acceleratie en afstand tot auto rechtsvoor (FuzzyDriver rijdt links)



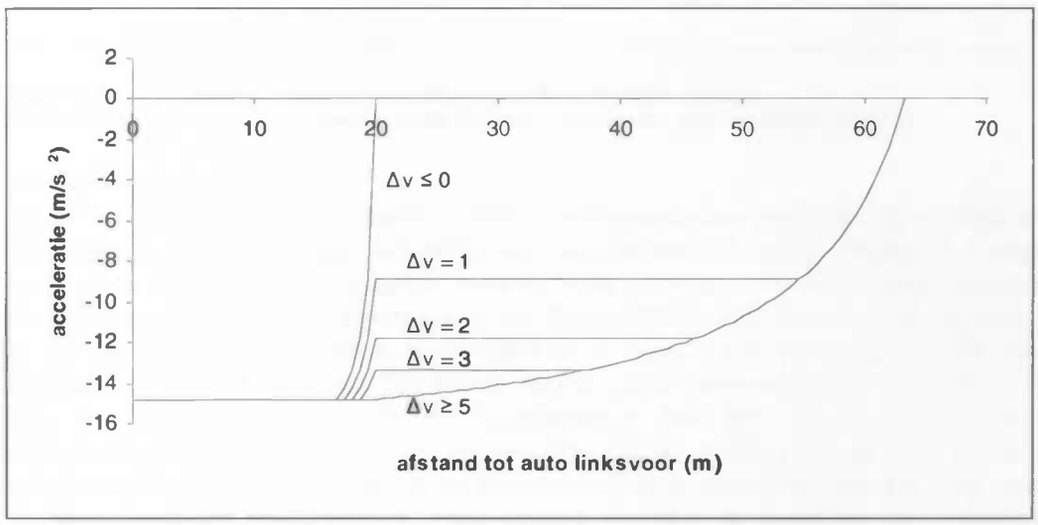
Figuur 4.14 – Acceleratie versus afstand tot auto rechtsvoor (linkerbaan)
De acceleratie is ongeacht de afstand altijd nul.

4. Volg auto linksvoor

De situatie waarin FuzzyDriver op de linker baan rijdt en de auto linksvoor volgt (zie figuur 4.15) is vergelijkbaar met de bovengenoemde situatie 'rechter auto vanaf de rechter baan volgen'. De resulterende uitvoer is te zien in de grafiek in figuur 4.16 en is inderdaad identiek aan die in figuur 4.12.

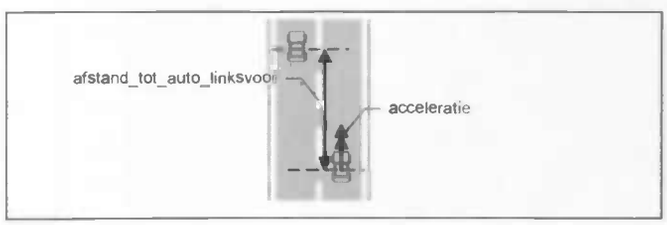


Figuur 4.15 – Acceleratie en afstand tot auto linksvoor (FuzzyDriver rijdt links)

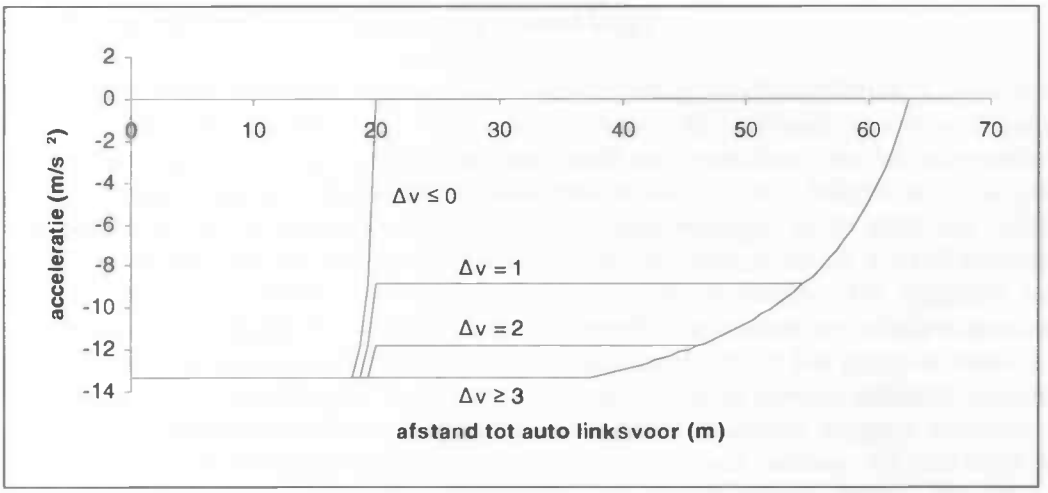


Figuur 4.16 – Acceleratie versus afstand tot auto linksvoor (linkerbaan)
Naarmate het snelheidsverschil groter is, wordt er eerder – dus bij een grotere afstand – geremd.

In de laatste mogelijke situatie rijdt FuzzyDriver rechts en het te volgen voertuig links, zoals in figuur 4.17. In deze situatie is er geen botsingsgevaar, maar het is wel strijdig met de verkeersregels om rechts in te halen. Deze situatie levert weer een afwijkende uitvoer op, zoals weergegeven in figuur 4.18.



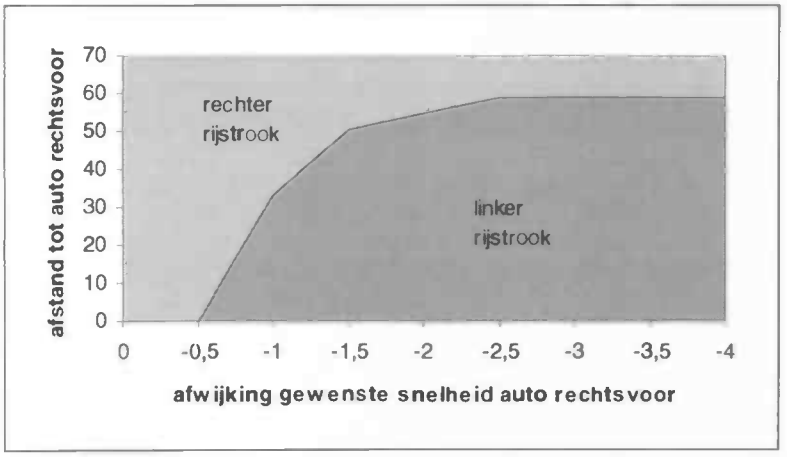
Figuur 4.17 – Acceleratie en afstand tot auto linksvoor (FuzzyDriver rijdt rechts)



Figuur 4.18 – Acceleratie versus afstand_tot_auto_linksvoor (rechterbaan)
Naarmate het snelheidsverschil groter is, wordt er eerder – dus bij een grotere afstand – geremd.

5. Moment van inhalen

Zoals gezegd kunnen er op het tactische niveau randvoorwaarden gedefinieerd worden voor het operationele niveau. Een goed (en in het geval van FuzzyDriver het enige) voorbeeld daarvan is de gewenste rijstrook die FuzzyDriver dient te volgen. In de paragraaf 4.3 werd reeds beschreven dat deze informatie in verband met de beperkingen van FuzzyTECH enkel via een omweg, namelijk bij de gratie van een closed loop systeem, uiteindelijk in de juiste vorm bij de betreffende primitieve gedragsmodule kan worden gebracht. Het kiezen van de juiste rijstrook speelt, zoals men zich kan voorstellen, een zeer belangrijke rol bij het manoeuvreren door het verkeer op de snelweg. Het merendeel van de coördinerende regels op het tactische niveau is dan ook aan deze keuze gewijd. Deze regels komen vaak in wat complexere verkeersconstellaties tot uitdrukking, die in de open loop-situatie moeilijk te reconstrueren zijn. Een uitzondering hierop is de baankeuze in verband met het inhalen. Het moment waarop besloten wordt om naar links te gaan wordt primair bepaald door de afstand tot het in te halen voertuig en diens afwijking van de gewenste snelheid van FuzzyDriver. Het bepalen van dit moment kan eenvoudig gebeuren door beide genoemde variabelen te variëren. Het resultaat hiervan is te zien in figuur 4.19.



Figuur 4.19 – Moment van inhalen
Als FuzzyDriver op de rechterbaan achter een auto rijdt en deze rijdt langzamer dan FuzzyDriver wil rijden, zal hij die auto willen inhalen. Afhankelijk van de afwijking van de gewenste snelheid van dat voertuig zal hij bij een andere afstand willen beginnen met inhalen. De gewenste rijstrook wordt dan 'linker'. Deze grafiek geeft aan wanneer dit gebeurt.

4.4.3 Beschrijving van het bestuurdersgedrag (closed loop)

In deze paragraaf worden metingen gepresenteerd van het gedrag van FuzzyDriver in een closed loop systeem, met behulp van de simulatie. Deze metingen worden getoond in een stroboscopische weergave van het afgelegde traject, eventueel aangevuld met grafieken van het relevantie in- en uitvoerwaarden uitgezet tegen de tijd. In de stroboscopische weergaven is telkens om de seconde een meting weergegeven. Om de nuances en de soms snelle veranderingen in de grafieken zichtbaar te kunnen maken, is daarbij met een hogere temporele resolutie gewerkt, namelijk T = 0,5 seconden.
In de stroboscopische weergave wordt gewerkt met relatieve posities ten opzichte van een referentievoertuig, bijvoorbeeld een auto die ingehaald wordt. Het werken met relatieve posities geeft een enigszins vertekend beeld van het werkelijk afgelegde traject. Maar het grote voordeel is dat de verkeershandeling nu ten opzichte van het relevante voertuig in kaart worden gebracht. Bovendien is de positie van het referentievoertuig 'ten opzichte van zichzelf' constant, hetgeen voorkomt dat de trajectweergaven door elkaar gaan lopen en er een brij van voertuigen ontstaat. Tot slot blijft door het werken met relatieve posities ook de ruimte die zo'n figuur inneemt enigszins binnen de perken.
Er zijn closed loop metingen verricht tijdens de volgende handelingen:

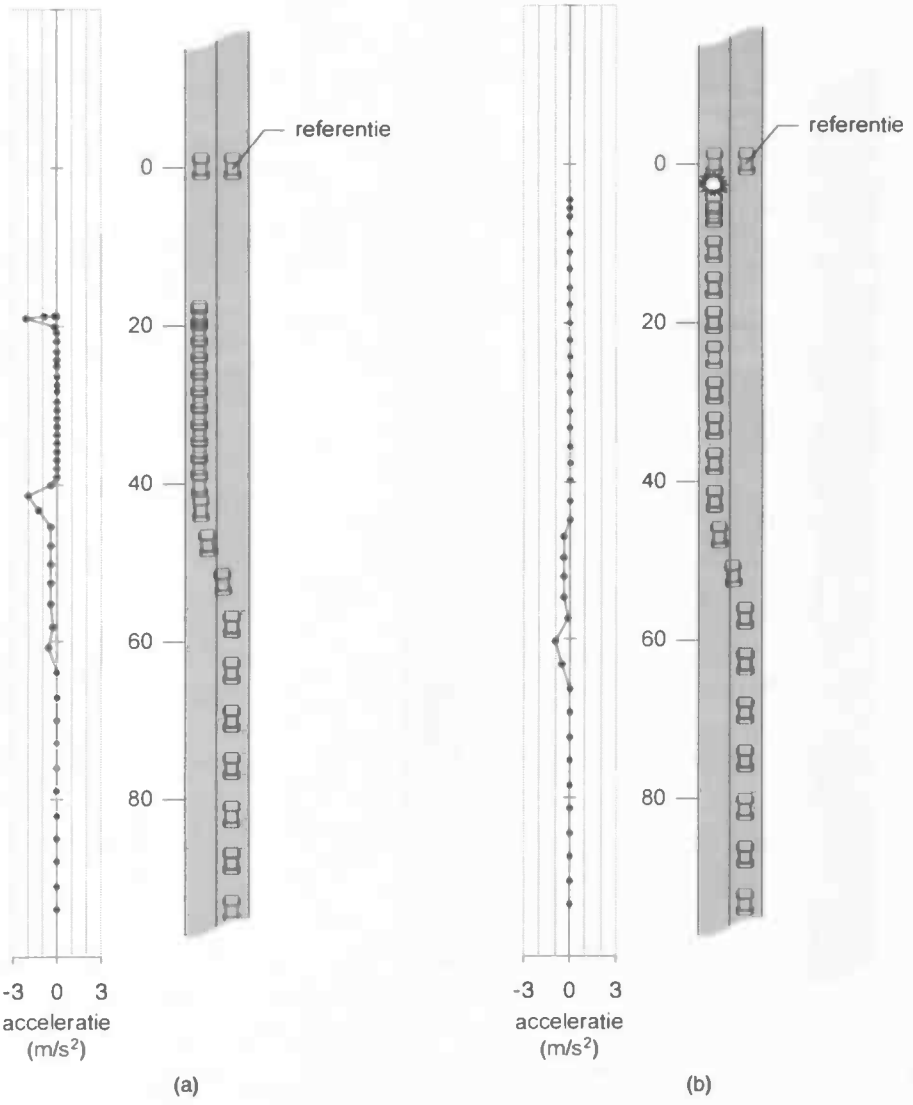
1. Afstand houden tot twee voertuigen naast elkaar
2. Afstand houden tot één voertuig op de linkerbaan
3. Inhalen van één voertuig
4. Inhalen van twee voertuigen achter elkaar
5. Combinatie inhalen en afstand houden
6. Inhalen met achteropkomend verkeer
7. Inhalen bij een te nemen afslag

1. Afstand houden tot twee voertuigen naast elkaar

Om het gedrag met betrekking tot afstand houden geïsoleerd te kunnen bekijken, is FuzzyDriver achter twee auto's geplaatst die met constante snelheid naast elkaar rijden. FuzzyDriver zal dan eerst geneigd zijn naar de linkerbaan te gaan om de auto die rechts rijdt in te halen, maar zal vervolgens genoodzaakt om achter de auto's te blijven rijden en afstand te bewaren, aangezien beide rijstroken geblokkeerd worden.

In figuur 4.20a is te zien dat dit gedrag werkelijk plaatsvindt. Het remmen geschiedt tijdig en geleidelijk. Uiteindelijk blijft FuzzyDriver op een kleine 19 meter achter de andere auto's rijden, ongeacht het initiële snelheidsverschil. Als we afwijken van de standaardwaarden en het 'bots niet gewicht' van 1,0 terugbrengen dan zien we dat het snel willen rijden de overhand neemt en FuzzyDriver op zijn voorligger inrijdt en botst (zie figuur 4.20b).

Het is wellicht veiliger om de volgafstand wat groter te maken. Dit kan bereikt worden door de membershipfuncties van de variabele 'afstand tot auto linksvoor' bij te stellen.



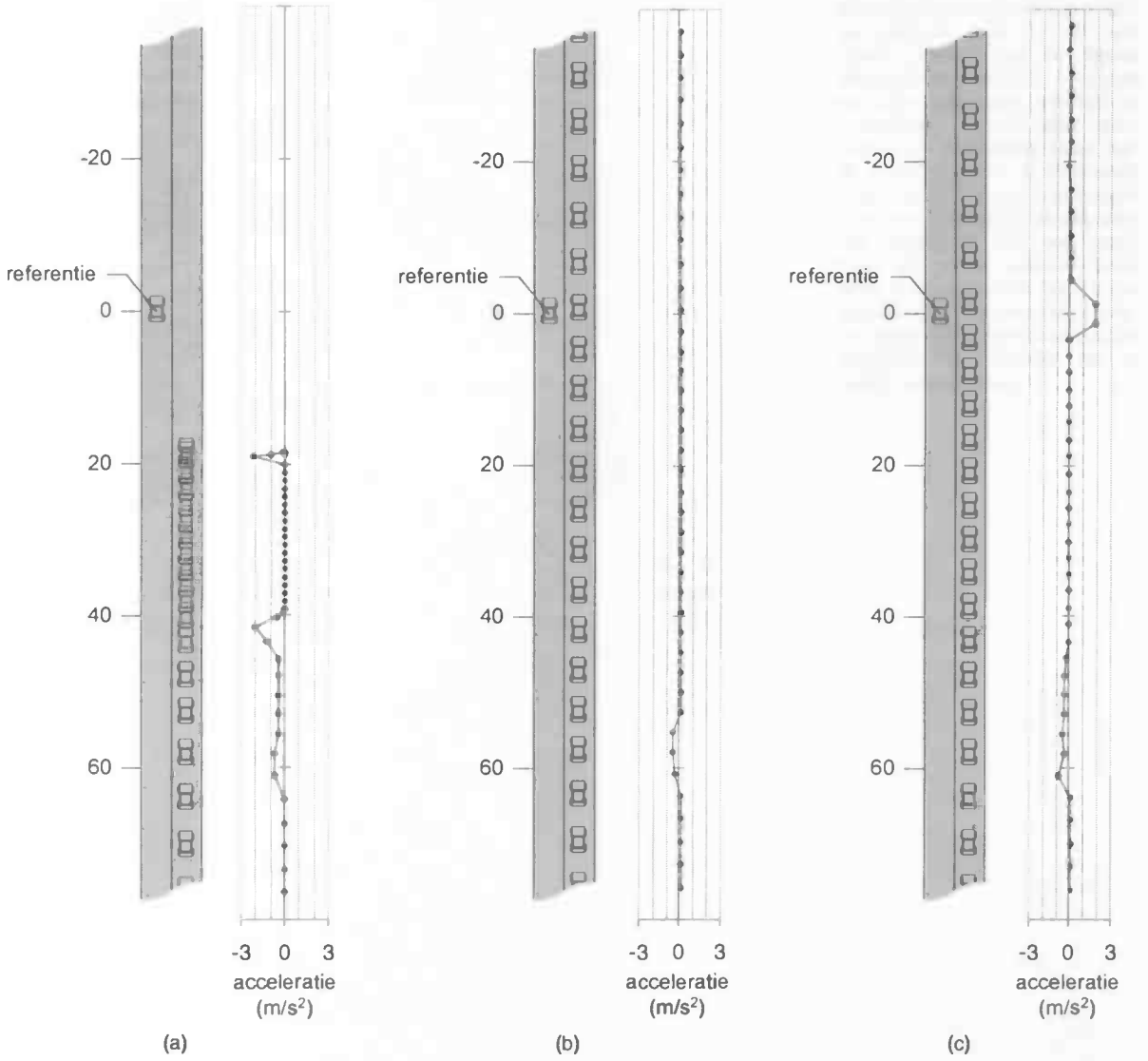
Figuur 4.20 – Afstand houden tot twee voertuigen naast elkaar
Twee auto's rijden naast elkaar met een snelheid van 100 km/u. FuzzyDriver start 200 meter daarachter, met zijn gewenste snelheid van 120 km/u. In (a) zijn voor de strategische gewichten de standwaarden gebruikt en in (b) is het 'bots niet' gewicht teruggebracht van 1,0 naar 0,4. Als gevolg daarvan zien we dat in (a) er keurig afstand wordt gehouden door af te remmen, terwijl er in (b) niet meer voldoende afgeremd wordt, hetgeen uiteindelijk een botsing tot gevolg heeft.

2. Afstand houden tot één voertuig op de linker baan

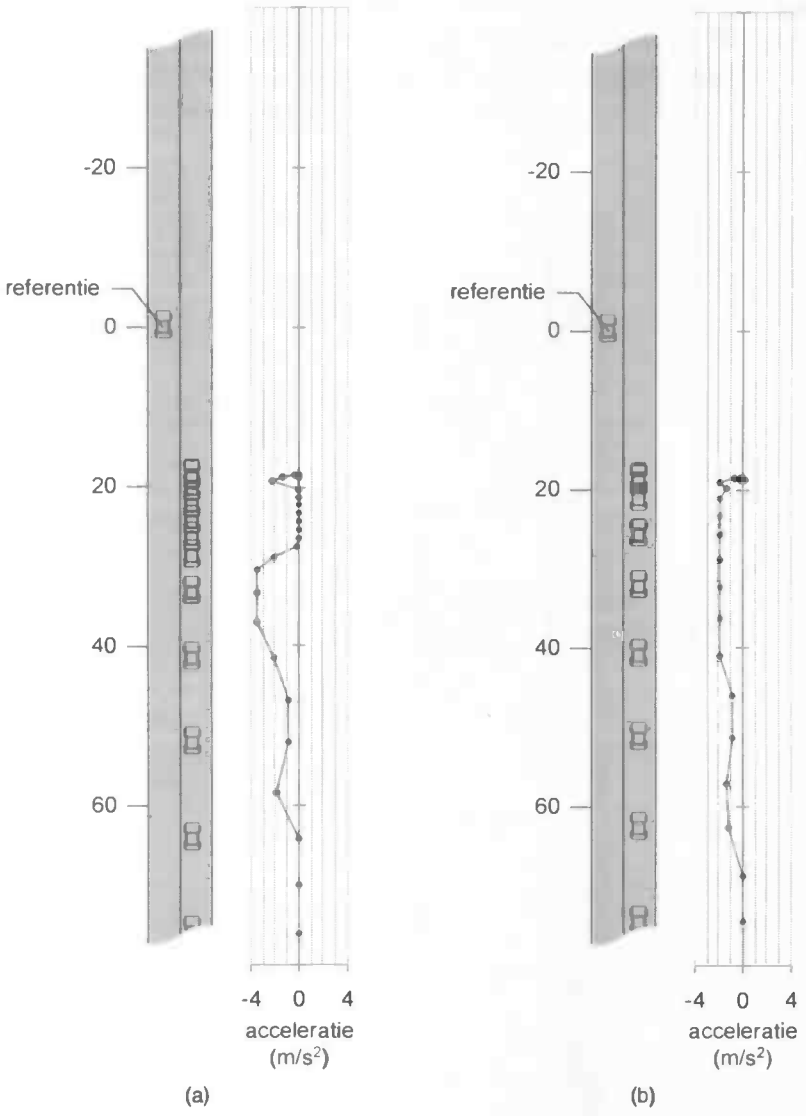
Wanneer FuzzyDriver rechts rijdt achter een voertuig op de linkerbaan, dient hij daar in principe achter te blijven en afstand te houden, aangezien er niet rechts ingehaald mag worden. Dat is ook precies wat er gebeurt (zie figuur 4.21). En wederom wordt er een afstand bewaard van een kleine 19 meter. Als we het 'volg regels gewicht' reduceren raakt FuzzyDriver steeds meer geneigd om toch rechts in te gaan halen. Bij een waarde van 0.4 mindert hij eerst nog enigszins vaart, maar bij 0.2 rijdt hij al haast onverstoord door.

Doen we hetzelfde experiment met een snelheidsverschil van 40 km/u, dan zien we precies hetzelfde patroon. FuzzyDriver rijdt nu 120 en de andere auto rijdt 80 km/u. Gegeven het grote snelheidsverschil remt FuzzyDriver wel erg laat, zoals te zien is in figuur 4.22. Bij een waarde voor 'volg regels gewicht' van 0,5 lijkt er zelfs sprake van een regelrechte noodstop.

Beter zou zijn als FuzzyDriver bij grotere snelheidsverschillen eerder zou beginnen met remmen. Dit is wederom een kwestie van het aanpassen van membershipfuncties, in dit geval van 'afstand tot auto linksvoor' en eventueel 'snelheidsverschil auto linksvoor'.



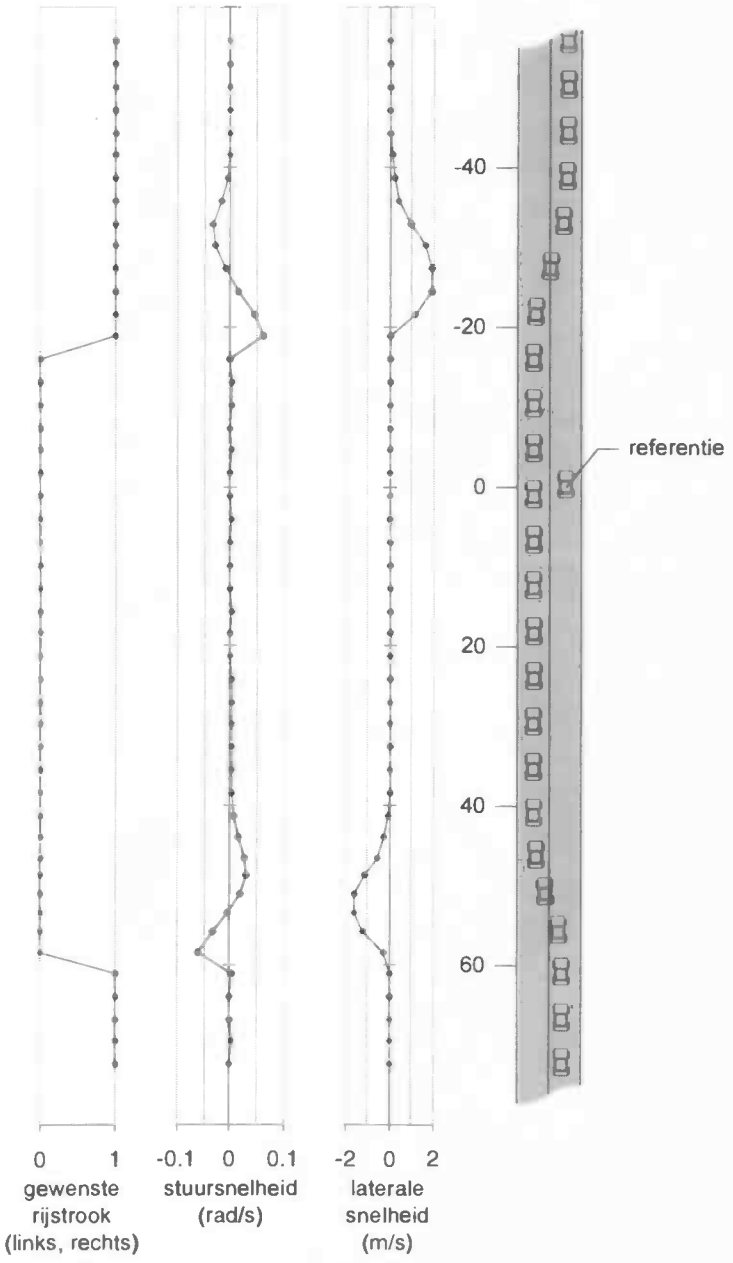
Figuur 4.21 – Afstand houden tot één voertuig op de linker baan (1)
FuzzyDriver rijdt op de rechterbaan, met een snelheid van 120 km/u. Dit is tevens zijn gewenste snelheid. Daarvoor rijdt een andere auto op de linkerbaan met een constante snelheid van 10 km/u. Het 'volg regels' gewicht wordt als volgt gevarieerd: In (a) is heeft het 'volg regels' gewicht de standaardwaarde, namelijk 0,6. In (b) is dit nog 0,4 en in (c) nog slechts 0,2. Naarmate dit gewicht daalt, neemt de neiging om conform de verkeersregels af te remmen en afstand te houden af.



Figuur 4.22 – Afstand houden tot één voertuig op de linker baan (2)
Wederom rijdt FuzzyDriver op de rechterbaan achter een langzamer voertuig op de linkerbaan. Het snelheidsverschil is nu echter groter. FuzzyDriver rijdt weer met een snelheid van 120 km/u, maar de andere auto heeft nu een snelheid van slechts 80 km/u.
In principe ontstaat hier weer hetzelfde patroon als in figuur 4.21. Bijzondere aandacht vraagt echter het geval weergeven in (b). Daar waar in (a) de standaardwaarden voor de gewichten zijn gebruikt, is in (b) het 'volg regels' gewicht met één tiende teruggebracht naar 0,5. Hier zien we een bijzonder grensgeval: FuzzyDriver maakt later aanstalten om te remmen, maar doet dit uiteindelijk toch in voldoende mate om achter zijn voorganger te blijven. Dit wordt bereikt door het toepassen van een korte maar zeer forse remactie. Deze actie is in de grafiek helaas net tussen de meetmomenten gevallen, maar uit de stroboscopische weergave is duidelijk op te maken wat er gebeurt.

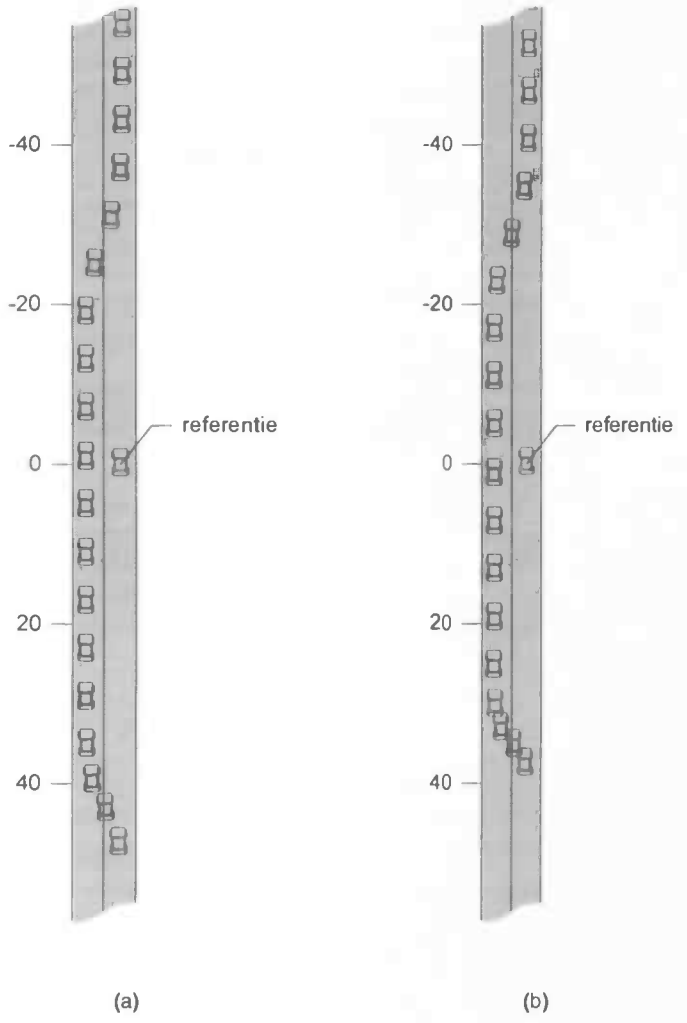
3. Inhalen van één voertuig

Normaal gesproken vindt het inhalen natuurlijk langs de linkerkant plaats. We testen FuzzyDriver op deze handeling door hem achter een langzamer voertuig op de rechterbaan te plaatsen en de simulatie te laten draaien. Bij respectievelijke snelheden van 120 en 100 km/u haalt FuzzyDriver in zoals in figuur 4.23. Deze beweging ziet er zeer goed uit, conform de 'norm' en de werkelijkheid. Over het juiste moment van inhalen en van het weer teruggaan naar de rechterbaan kunnen de meningen uiteraard verschillen.

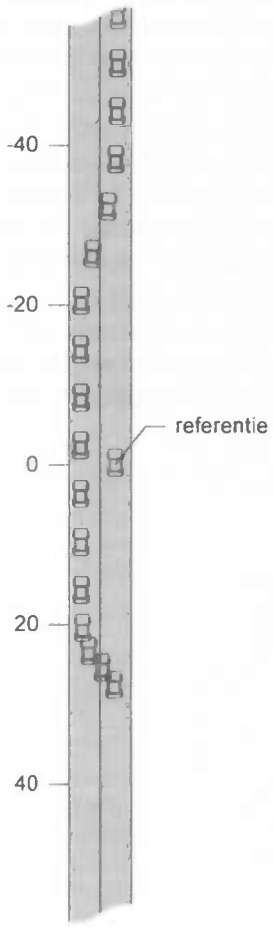


Figuur 4.23 – Eén auto inhalen
FuzzyDriver haalt met 120 km/u en auto in die 100 rijdt. Let op, aangezien er in de afbeelding met relatieve posities is gewerkt lijkt het verwisselen van rijstrook over een kortere afstand te gebeuren. De grafieken laten zien hoe achtereenvolgend de gewenste rijtrook verandert, aan de hand waarvan een stuuractie uitgevoerd wordt, die weer een (verandering in de) laterale snelheid teweeg brengt.

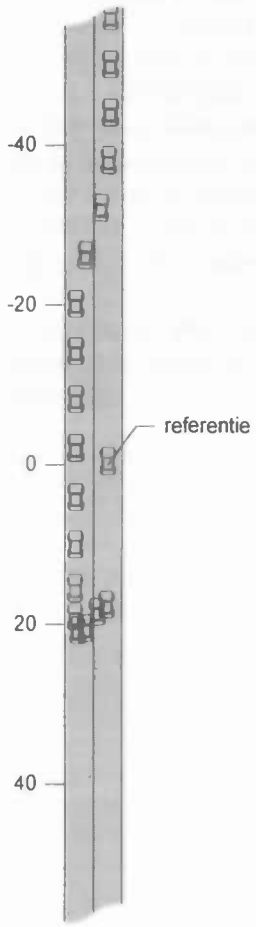
Om de robuustheid van FuzzyDriver te testen heb ik hem steeds dichter achter het andere voertuig laten beginnen. Er wordt zo een start gemaakt in een situatie waarin FuzzyDriver normaal niet terecht zou komen en die bovendien direct actie vereist. Het resultaat van dit experiment is weergegeven in figuur 4.24. FuzzyDriver past zichtbaar de hevigheid van het remmen en het sturen aan aan de omstandigheden. Dit gebeurt op adequate wijze, behalve dan dat het remmen en sturen gelijktijdig plaatsvindt, hetgeen de auto kan doen laten uitbreken of omslaan. Dat het beter is in zulke situaties éérst te remmen en dán te sturen of vice versa, maar niet tegelijkertijd, zou expliciet in de regels van FuzzyDriver verwerkt moeten worden.



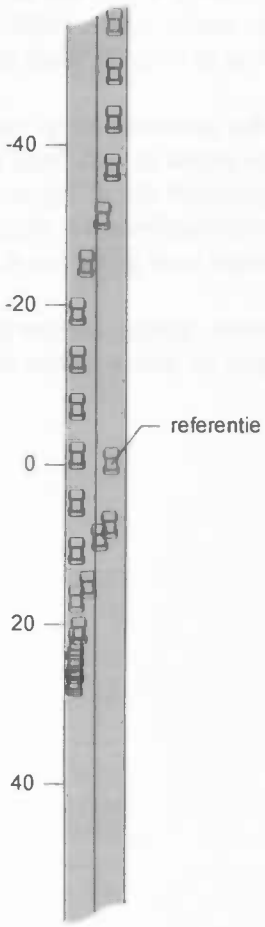
Figuur 4.24 – Reactie van FuzzyDriver bij een start vlak achter een trager voertuig
Om FuzzyDriver te testen op robuustheid is het inhaalscenario nog vijfmaal herhaald, maar nu startend op een afstand van achtereenvolgens (a) 50, (b) 40, (c) 30, (d) 20 en (e) 10 meter achter het voertuig. FuzzyDriver heeft een beginsnelheid van 120 km/u, gelijk aan zijn gewenste snelheid. Het tragere voertuig ervoor rijdt met een snelheid constante snelheid van 100 km/u.
In de afbeeldingen is te zien dat de eerste reactie van FuzzyDriver afremmen is, om direct daarna naar de linker rijstrook te gaan voor het inhalen. Met name bij een initiële afstand van 20 en 10 meter is de remreactie zeer hevig. De stuurreactie volgt zoals gezegd direct of wordt zelfs gecombineerd met het remmen, hetgeen gevaarlijk is in verband met een extreme gewichtsverplaatsing. Hierdoor kan in een realistische situatie de auto aan de achterkant gaan uitbreken of zelfs omslaan.


-40
-20
referentie
0
20
40

(c)


-40
-20
referentie
0
20
40

(d)


-40
-20
referentie
0
20
40

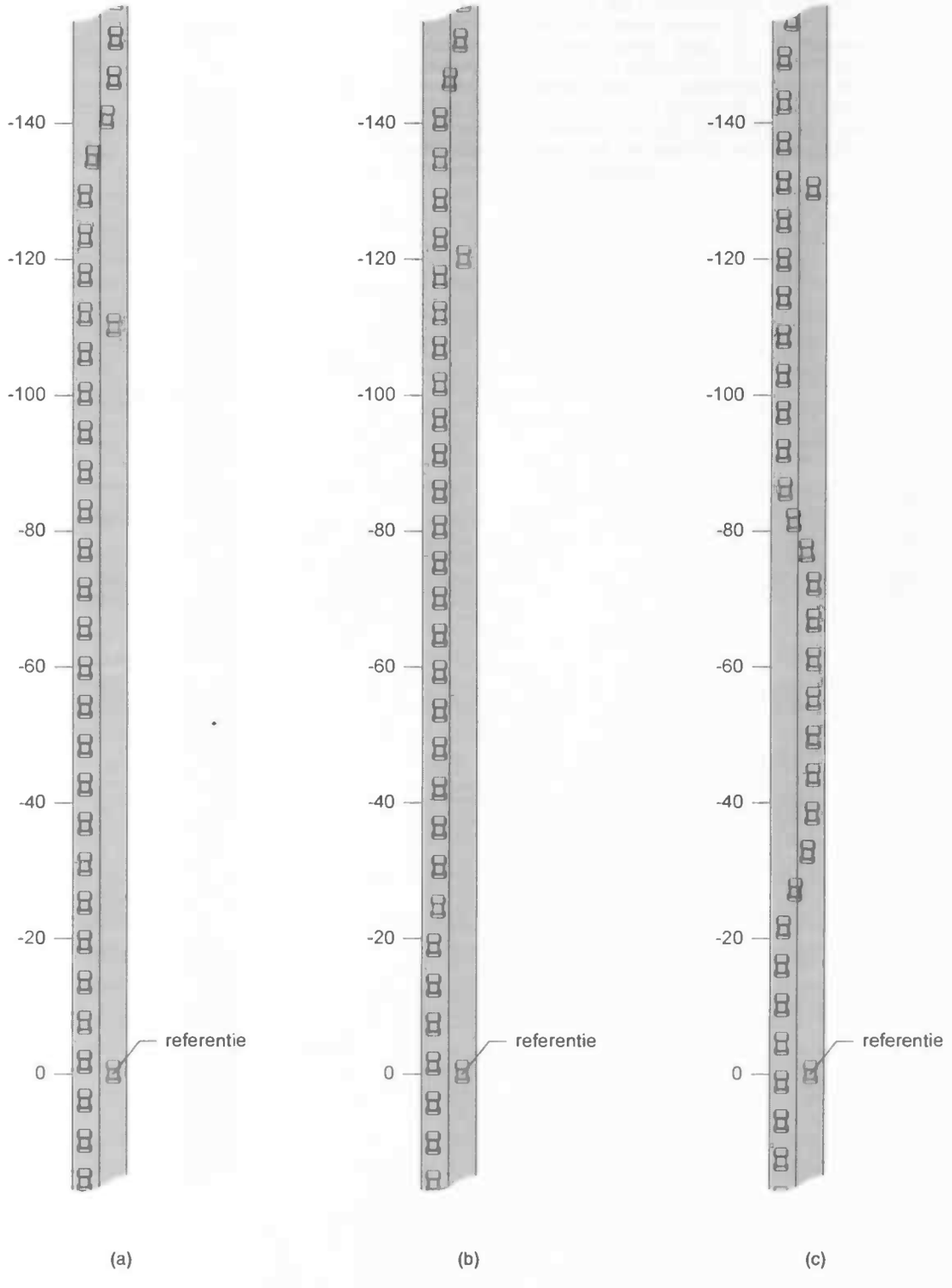
(e)

4. Inhalen van twee voertuigen achter elkaar

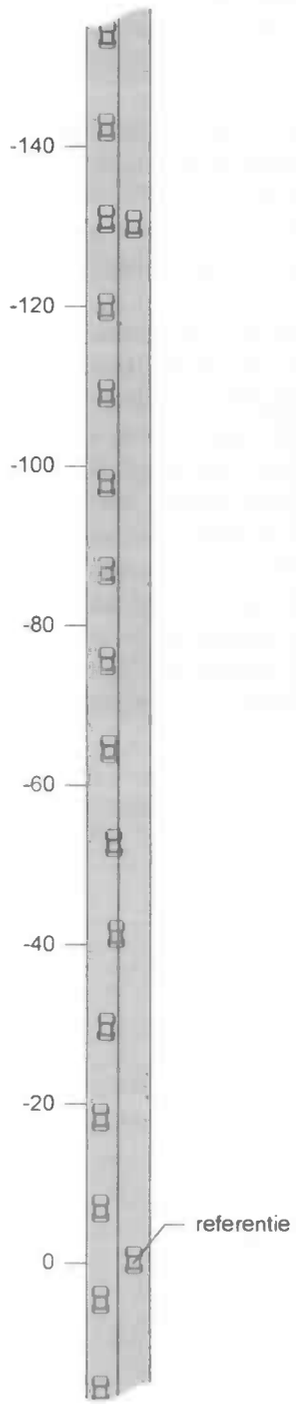
Op de snelweg is het zeer gangbaar dat een bestuurder meerdere voertuigen achter elkaar inhaalt. Het is vaak zelfs niet wenselijk om tussen het inhalen van twee voertuigen die dicht achter elkaar rijden terug te gaan naar de rechter rijstrook. Anderzijds dient er ook niet onnodig links gereden te worden als het eerstvolgende in te halen voertuig nog ver vooruit rijdt.

We zetten FuzzyDriver een aantal keren achter twee langzamere voertuigen op rechterbaan, telkens met een andere onderlinge afstand tussen de in te halen auto's. FuzzyDriver heeft dan de keuze om óf tussen de twee voertuigen in tijdelijk weer naar rechts te gaan, óf beide voertuigen in één beweging in te halen. Dit zal afhankelijk zijn van de afstand tussen de in te halen voertuigen, het snelheidsverschil en de waarde van 'volg regels gewicht', aangezien het conform de regels is om terug naar rechts te gaan als je niet meer aan het inhalen bent.

Uit de figuren 4.25 en 4.26 blijkt dat FuzzyDriver het verwachte en gewenste gedrag vertoont: naarmate de onderlinge afstand tussen de beide in te halen voertuigen groter wordt, neemt de neiging om tussen het inhalen door naar rechts te gaan toe.



Figuur 4.25 – Twee auto's achter elkaar inhalen (1)
FuzzyDriver rijdt met zijn gewenste snelheid van 120 km/u achter twee auto's met een constante snelheid van 100 km/u. De onderlinge afstand tussen de twee in te halen auto's is in **(a)** 110 meter, in **(b)** 120 meter en in **(c)** 130 meter. Pas in (c) gaat FuzzyDriver tussen het inhalen van de beide auto's door naar rechts.



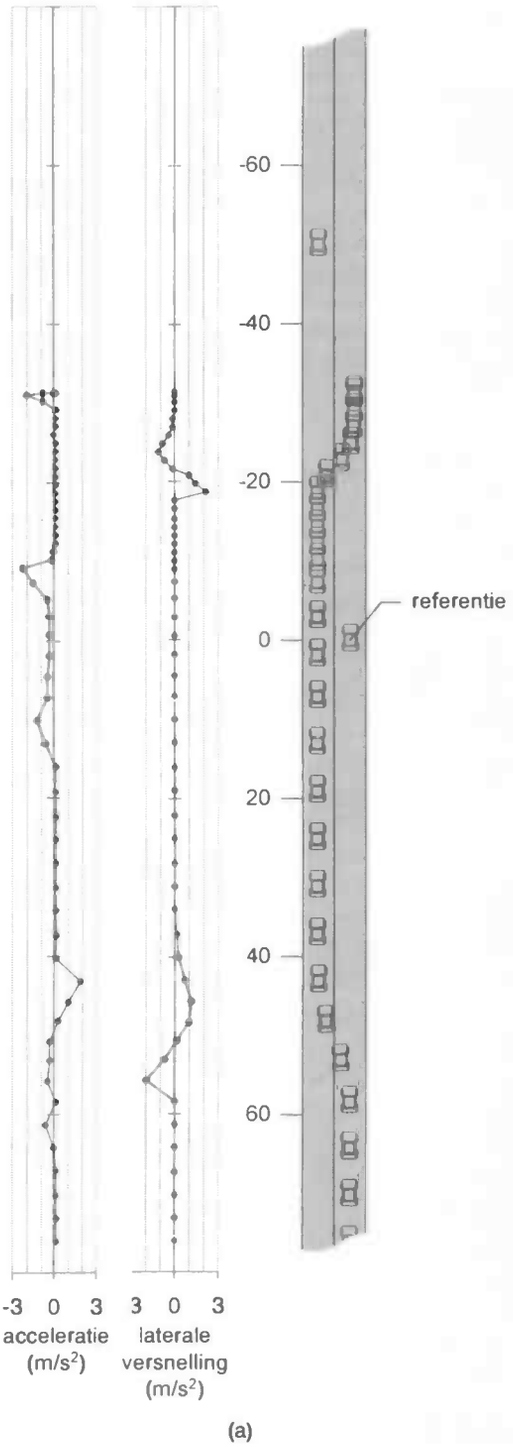
Figuur 4.26 – Twee auto's achter elkaar inhalen (2)
In figuur 4.25 zagen we dat FuzzyDriver bij een onderlinge afstand van 130 meter tussen de in te halen voertuigen door naar rechts gaat. In dit tweede voorbeeld behouden we de afstand van 130 meter, maar verhogen het snelheidsverschil. FuzzyDriver rijdt nu 120 km/u en de andere beide auto's slechts 80. Nu is FuzzyDriver toch weer minder genelgd tussendoor naar de rechterbaan te gaan, iets wat men van een menselijke bestuurder ook zou verwachten.

5. Combinatie inhalen en afstand houden

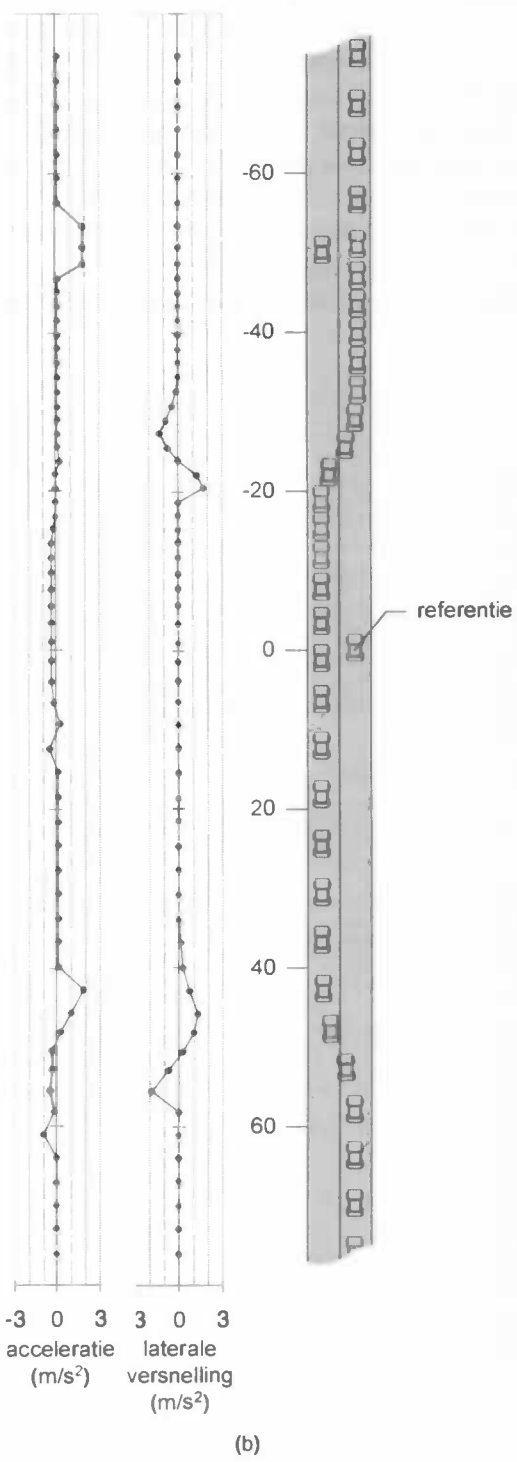
In het volgende scenario worden inhalen en afstand houden in combinatie gebruikt. FuzzyDriver wordt in de beginsituatie met 120 km/u achter een voertuig op de rechterbaan geplaatst, die met 100 km/u rijdt. Nog eens 50 meter daarvoor rijdt een tweede voertuig, ook met een snelheid van 100 km/u, op de linkerbaan.

De verwachting is dat FuzzyDriver in de standaardconfiguratie het eerste voertuig zonder snelheid te minderen zal inhalen, om dan nog tijdens het inhalen reeds de snelheid aan te passen aan het voertuig op de linkerbaan. Uiteindelijk zal FuzzyDriver dan naar de rechterbaan gaan, daarbij afstand houdend tot dat tweede voertuig. Om het alternatief te bewerkstelligen waarbij FuzzyDriver de tweede auto langs de rechterkant inhaalt, kan of het 'wees snel' groter gemaakt worden, of het 'volg regels' gewicht kleiner.

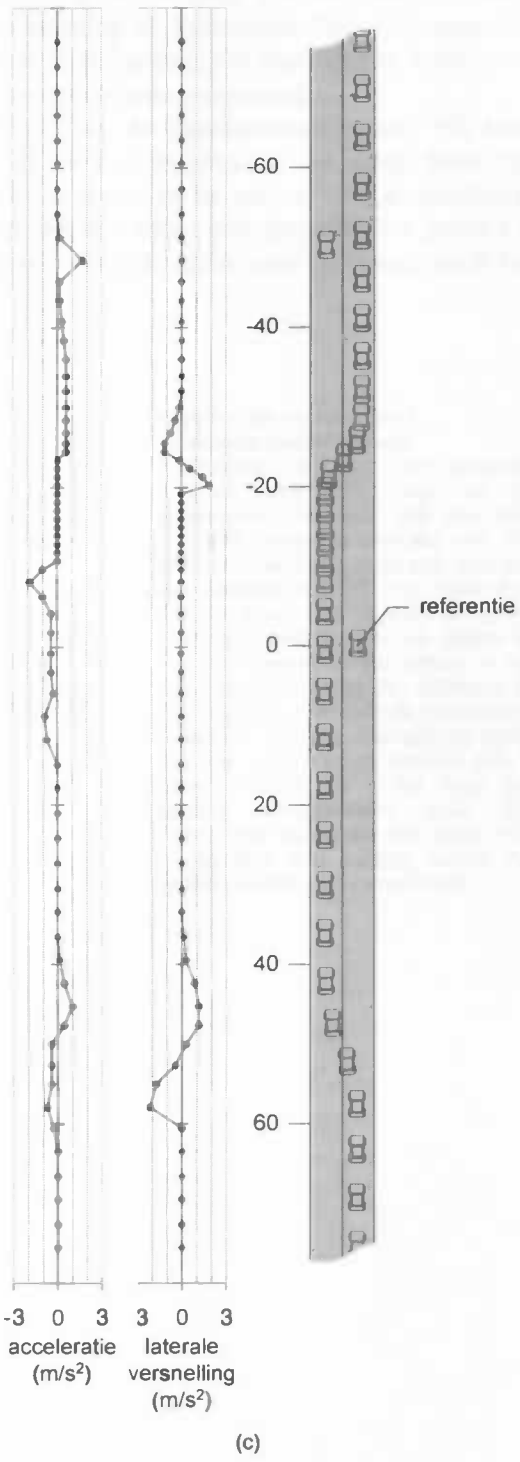
Het draaien van de simulatie met dit scenario leverde de plaatjes op zoals die te zien zijn in figuur 4.27. Het traject dat FuzzyDriver in de standaardconfiguratie aflegt is precies zoals verwacht. De overige twee plaatjes verdienen enige verdere toelichting: In beide gevallen zien we dat er rechts ingehaald wordt. In alle gevallen gaat FuzzyDriver – conform de verkeersregels – na het inhalen terug naar de rechter baan. Normaal gesproken verbieden de verkeersregels dat het tweede voertuig rechts ingehaald wordt, maar dit gedrag wordt onderdrukt door het volgen van de verkeersregels minder belangrijk maken of juist te overschaduwen met het belang van het snel rijden. Bij het reduceren van het 'volg regels' gewicht zien we wel dat FuzzyDriver eerst nog flink afremt in verband met het tweede voertuig, waar dat bij het vergroten van het 'wees snel' gewicht veel minder het geval is. Deze laatste variant is potentieel gevaarlijk: het snel rijden en het niet botsen zijn twee doelen die vaak tegengestelde acties tot gevolg hebben. FuzzyDriver botst dan ook enkel niet op zijn voorliggers bij de gratie van het verwisselen van rijstrook.



Figuur 4.27 – Combinatie inhalen en afstand houden
FuzzyDriver rijdt met 120 km/u op de rechterbaan achter een auto die 100 rijdt op de rechterbaan. Daar weer 50 meter voor rijdt nog een auto, ook met een snelheid van 100 km/u, maar dan op de linkerbaan.
In (a) zijn de standaardwaarden voor de gewichten gebruikt. FuzzyDriver haalt de eerste auto in, remt voor de tweede en gaat terug naar de rechterbaan om daar netjes afstand te houden tot de auto op de linkerbaan. In (b) is de waarde van het 'wees snel' gewicht verhoogd naar 0,8. FuzzyDriver remt nu nauwelijks af voor de auto op de linkerbaan en zigzagt als het ware tussen beide voortuigen door. In de grafiek van de acceleratie is te zijn dat FuzzyDriver wel wacht met weer accelereren tot ook het tweede voertuig (rechts) ingehaald is. In (c) staan de strategische gewichten weer op hun normale waarde, behalve het 'volg regels' gewicht dat nu terug is gebracht naar 0,2. Ook dit is reden voor FuzzyDriver om de tweede auto rechts in te halen. Wel wordt er in de eerst instantie meer afgeremd.


-60
-40
-20
0
20
40
60
referentie
-3 0 3
acceleratie (m/s²)
3 0 3
laterale versnelling (m/s²)

(b)

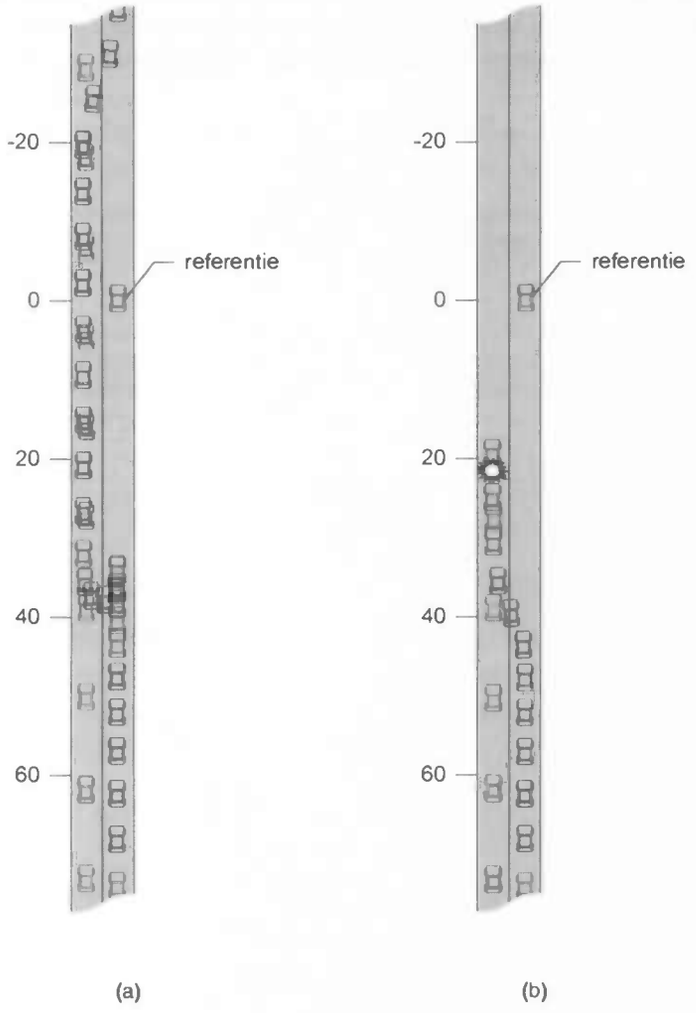

-60
-40
-20
0
20
40
60
referentie
-3 0 3
acceleratie (m/s²)
3 0 3
laterale versnelling (m/s²)

(c)

6. Inhalen met achteropkomend verkeer

Een andere situatie waarin tegengestelde belangen gelden, is die waarbij een langzamer voertuig ingehaald kan worden, terwijl er sneller verkeer achterop komt op de linkerbaan. Enerzijds noopt het doel om snel te rijden tot inhalen, anderzijds is het meer conform het doel om botsingen te voorkomen om op de rechter baan te blijven rijden tot het achteropkomend verkeer gepasseerd is.

Wederom is dit scenario eerst weer getest met FuzzyDriver met de standaardconfiguratie. Hij heeft een snelheid van 120 km/u, de auto daarvoor rijdt met 100 km/u en de auto die van achter komt rijdt met 140 km/u. In de standaardconfiguratie wacht FuzzyDriver tot de snelle auto voorbij is, om daarna het langzame voertuig in te halen (zie figuur 4.28a). Als we het 'botst niet gewicht' ver genoeg af laten nemen, dan haalt FuzzyDriver meteen in, wat een botsing met de snelle auto tot gevolg heeft (zie figuur 4.28b). Dit gedrag is conform de verwachtingen.



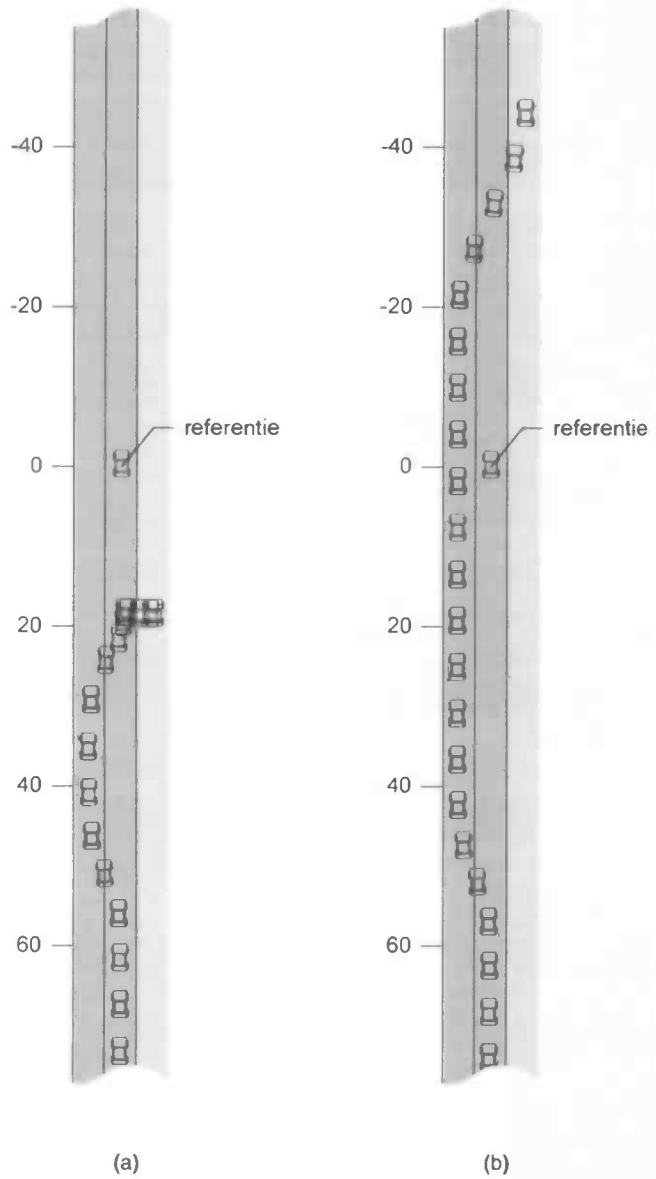
Figuur 4.28 – Inhalen met achteropkomend verkeer
FuzzyDriver rijdt met zijn gewenste snelheid van 120 km/u op de rechterbaan. Daarvoor rijdt een auto met een constante snelheid van 100 km/u. Van achteren komt een auto met een snelheid van 140 km/u, rijdend op de linkerbaan. Om FuzzyDriver en de achteropkomende auto uit elkaar te kunnen houden, is die laatste in het grijs afgedrukt. In (a) zijn wederom de standaardwaarden voor de strategische gewichten gebruikt. FuzzyDriver wacht dan met inhalen tot de snellere auto is gepasseerd. In (b) is het 'bots niet' gewicht teruggebracht naar 0,4. FuzzyDriver wacht dan niet, haalt in en er ontstaat een botsing tussen het snellere voertuig en FuzzyDriver.

7. Inhalen bij een te nemen afslag

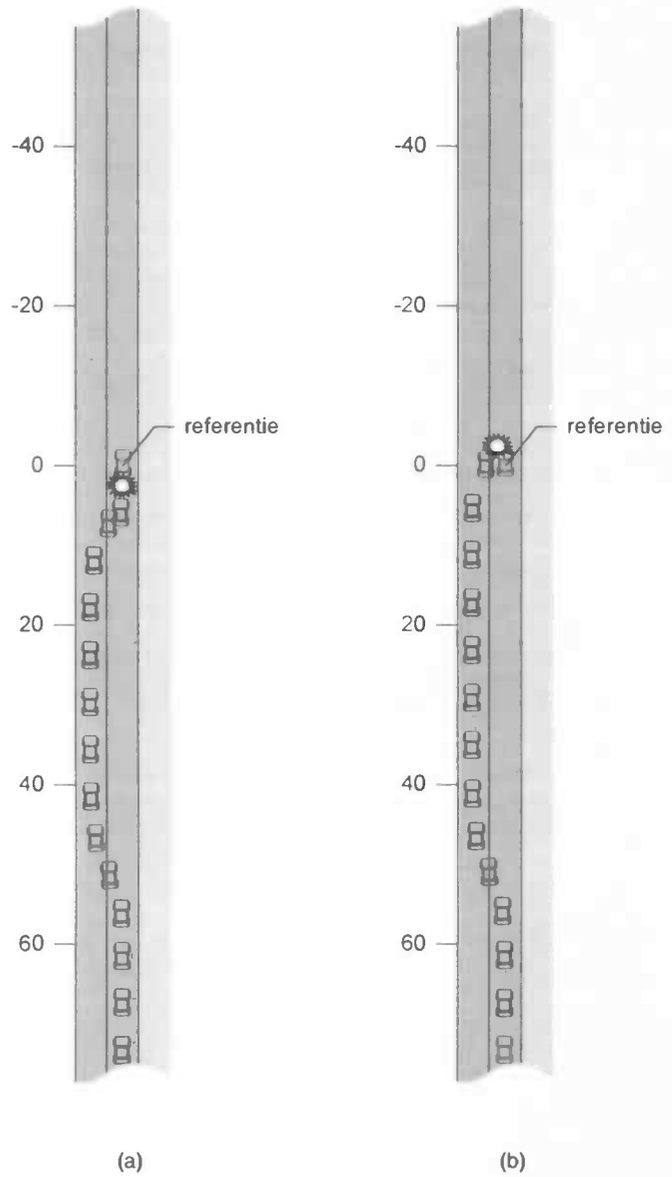
In het laatste scenario wordt FuzzyDriver wederom achter een langzamer voertuig geplaatst, maar ditmaal in de buurt van een afslag die genomen moet worden. Afhankelijk van diverse factoren zal FuzzyDriver zijn voorligger inhalen of er juist achter blijven hangen, teneinde de afslag te kunnen nemen. Om dit gedrag in kaart te brengen is de initiële afstand tot de afslag gevarieerd, alsmede het 'volg route gewicht'. De resultaten zijn te zien in de figuren 4.29, 4.30 en 4.31.

Tijdens deze tests bleek al snel dat de standaardconfiguratie niet het gewenste gedrag oplevert. FuzzyDriver is, conform de wensen en de verwachtingen, wel geneigd om bij grotere initiële afstanden tot de afslag eerst nog in te halen en bij veel kleinere afstanden juist niet (zie figuur 4.29). Maar op het grensgebied tussen beide treden er problemen op: FuzzyDriver begint dan eerst met inhalen, maar besluit dan halverwege toch richting rechter baan te gaan voor het nemen van de afslag. Hierbij wordt het in te halen voertuig rakelings gepasseerd of zelfs aangereden (zie figuur 4.30).

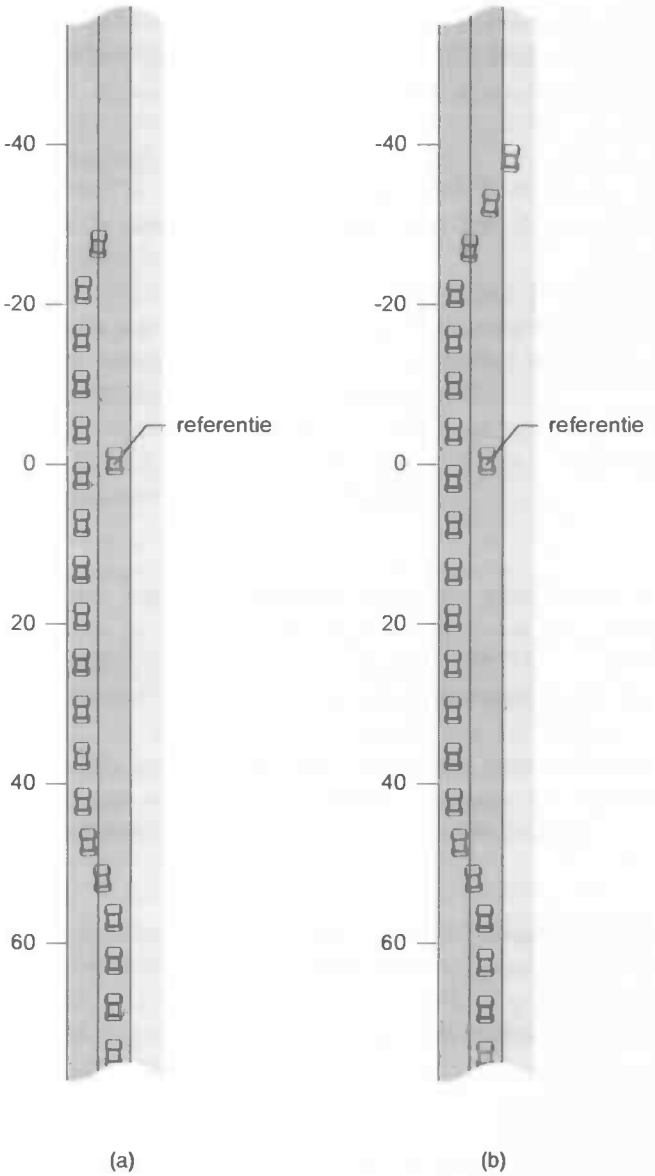
Naar aanleiding van deze bevindingen zijn verdere tests uitgevoerd met een 'volg route gewicht' van 0,4 in plaats van 0,5 (zie figuur 4.31). Deze wijziging lost het probleem van de botsingen volledig op. In een enkel geval werd de afslag nu wel gemist. Bij een initiële afstand van 600 meter besloot FuzzyDriver wel (op een nette manier) in te halen en miste daardoor de afslag. De vraag of er een nog meer ideale configuratie van het systeem bestaat is in verband met de beschikbare tijd niet beantwoord.



Figuur 4.29 – Inhalen bij een afslag (1)
FuzzyDriver rijdt met een snelheid van 120 km/u en nadert een te nemen afslag. Hij heeft de keuze om eerst nog een andere auto in te halen, die 100 meter voor hem rijdt met een snelheid van 100 km/u. In zowel (a) als (b) zijn de standaardwaarden voor de strategische gewichten gebruikt. In (a) begint de beschreven situatie 500 meter voor de afslag. FuzzyDriver begint met een inhaalmanoeuvre, maar besluit die uiteindelijk niet af te maken. Hij blijft achter het langzamere voertuig op de rechterbaan rijden en neemt de afslag. In (b) begint dezelfde situatie 700 meter voor de afslag. Nu is er voldoende ruimte om in te halen en de daarna de afslag te nemen, wat FuzzyDriver dan ook doet.



Figuur 4.30 – Inhalen bij een afslag (2)
In bepaalde gevallen levert de standaardconfiguratie van de strategische gewichten problemen op. We laten FuzzyDriver weer in de zelfde situatie beginnen als in figuur 4.29, maar nu beginnend in (a) 600 meter en in (b) 650 meter voor de afslag. FuzzyDriver zet in beide gevallen weer de inhaalmanoeuvre in, maar besluit halverwege toch de afslag te willen nemen, daarbij het ingehaalde voertuig rammend.



Figuur 4.31 – Inhalen bij een afslag (3)
We herhalen nu de experimenten uit 4.30, maar nu met een 'volg route' gewicht van 0,4. In (a) wordt weer 600 meter voor de afslag begonnen. FuzzyDriver maakt de inhaalbeweging af zonder botsingen te veroorzaken, maar mist daardoor de afslag. In (b) start de situatie 650 meter voor de afslag. De inhaalmanoeuvre wordt afgemaakt en de afslag wordt gehaald.

4.4.4 Observaties van de experts

De experts hebben het gedrag van FuzzyDriver van diverse kanten bekeken volgens de methode genoemd in hoofdstuk 3. De observaties zijn hieronder weergegeven, ingedeeld op de criteria die ze aangaan. De resultaten zijn bondig, maar geven een goede samenvatting van de observaties.

Normatief gedrag
Allereerst was de vraag aan de experts of FuzzyDriver aan de minimale verwachting voldeed, in termen van de taakuitvoering in enge zin: van A naar B komen zonder kleerscheuren, de verkeersregels in acht nemend.
Op dit punt zijn er geen aanmerkingen op het gedrag. Het produceren van normatief gedrag was überhaupt een mijlpaal die in het implementatieproces eerst bereikt diende te worden alvorens verder te gaan naar de testfase. Met het modale profiel worden de strategische doelen die het gedrag sturen met de juiste onderlinge prioriteit nagestreefd.
FuzzyDriver focust zichtbaar op het nastreven van de gewenste snelheid en het volgen van de route, tenzij de hieruit voortvloeiende acties in een bepaalde verkeersituatie strijdig zijn met de verkeersregels en/of de verkeersveiligheid.

Veiligheid
Het nastreven van de veiligheid komt tot uitdrukking in het houden van afstand tot eventuele voorliggers en in het wachten op 'vrij baan' bij het wisselen van rijstrook. Bij het modale profiel wordt dit gedrag steeds succesvol nagestreefd. Het profiel behoefde nog wel een kleine aanpassing: zoals beschreven werd in de vorige subparagraaf, leidde het oorspronkelijke profiel in sommige gevallen bij het nemen van een afslag nog tot snijden of zelfs botsen. De aanpassing heeft dit probleem zoals gezegd verholpen. Een ander punt van onveiligheid is dat bij het afremmen geen acht wordt geslagen op achteropkomend verkeer. In FuzzyDriver is momenteel ook geen gedrag geïmplementeerd om daar wel rekening mee te houden.

Efficiëntie
De experts maakten geen melding van overbodige of inefficiënte handelingen. De rijstrookwisselingen die FuzzyDriver maakt worden telkens, gegeven de situatie, als logische keuzes beschouwd. En zolang de situatie het toelaat, wordt er met de gewenste snelheid gereden. Een enkele keer vertoont FuzzyDriver een soort schijnbeweging naar rechts tussen het inhalen van twee auto's, maar dat wordt niet zozeer als inefficiënt aangemerkt.

Natuurlijk gedrag
Verwachte voordelen van het gebruik van fuzzy logic zijn soepele – in tegenstelling tot abrupte – handelingen en het uitblijven van oscillaties in gedrag en gedragskeuze. Hoog frequente oscillaties van enige betekenis worden in de praktijk dan ook niet waargenomen.
Wel werden er in enkel gevallen door meerdere experts laag-frequente oscillaties waargenomen. FuzzyDriver neigde dan zichtbaar na het inhalen kort terug te gaan naar de rechterrijstrook alvorens vrij direct weer naar links te gaan om een volgende auto in te halen (zie ook het kopje 'efficiëntie'). Dit gedrag is enkele malen gereproduceerd en daarbij bleek dat het aantal oscillatieperiodes telkens beperkt bleef tot één, hooguit twee.
Het feit dat het gedrag vloeiend en bovendien altijd consequent is, geldt onder dit criterium ook als negatief punt. De experts herkenden daarin niet altijd de menselijke autobestuurder, die van keer tot keer verschillend kan reageren op dezelfde situatie, die wel eens slingert binnen zijn rijstrook en bovendien dikwijls zichtbaar onder invloed staat van emoties, die mede middels het rijgedrag gecommuniceerd worden naar de andere weggebruikers.

4.4.5 Differentiatie van rijstijlen

Om te bepalen of er succesvol verschillende rijstijlen geproduceerd kunnen worden is aan zeven proefpersonen gevraagd om vier verschillende bestuurdersprofielen te identificeren aan de hand van de rijstijl. Naast het modale profiel dat in de observaties uit de vorige subparagraaf is gebruikt, is er hiertoe nog een drietal profielen aangemaakt. De profielen werden reeds in hoofdstuk 3 genoemd en ze worden hier voor het leesgemak hier in figuur 4.32 herhaald.

	Strategische gewichten			
Bestuurdersprofiel	**Bots niet**	**Volg regels**	**Volg route**	**Ga snel**
Profiel A – Modaal/ Normatief	1,0	0,6	0,5	0,4
Profiel B – Overdreven voorzichtig	1,0	0,8	0,5	0,2
Profiel C – Asociaal	1,0	0,6	0,5	0,8
Profiel D – Snel en roekeloos	0,8	0,2	0,5	0,8

Figuur 4.32 – Bestuurdersprofielen
Herhaling van figuur 3.11. In deze tabel staan de vier bestuurdersprofielen gepresenteerd. Voor elk van de profielen is aangegeven welke configuratie van de strategische gewichten zij vertegenwoordigen. Zoals te zien in de tabel is het 'volg route' gewicht niet gevarieerd. De reden hiervoor is dat er zich in de simulatie maar één afslag bevindt, waar dit gewicht tot uitdrukking kan komen. Op basis daarvan is er te weinig mogelijkheid tot differentiatie tussen profielen met verschillende waarden voor dit gewicht.

In 93 procent van de gevallen werden de profielen juist herkend. De resultaten zijn weergegeven in figuur 4.33. Profiel A viel op door niet op te vallen: alle verkeershandelingen verliepen normaal. Profiel B werd vooral herkend door het feit dat de bestuurders reeds volledig ingezette inhaalpogingen afbraken en weer naar de rechterbaan gingen op het moment er verkeer van achter op de linkerbaan naderde. Profiel A en B werden eenmaal met elkaar verward. Profiel C werd vooral herkend doordat de bestuurder snel geneigd was rechts in te halen. Wel werd er nog enige afstand gehouden als beide rijstroken geblokkeerd werden door langzamere voertuigen. Bij profiel D werd snel herkend, doordat de bestuurders tegen de bumper van hun voorganger aanreden als deze te langzaam reed op de linkerbaan. Bovendien werd er nauwelijks acht geslagen op het verkeer op de linkerbaan bij het naar links gaan, wat ook botsingen tot gevolg had.

	Identificatie door de zeven proefpersonen:							
Werkelijk profiel:	#1	#2	#3	#4	#5	#6	#7	**Totaal**
Profiel A	B	A	A	A	A	A	A	86%
Profiel B	A	B	B	B	B	B	B	86%
Profiel C	C	C	C	C	C	C	C	100%
Profiel D	D	D	D	D	D	D	D	100%
Totaal	50%	100%	100%	100%	100%	100%	100%	**93%**

Figuur 4.33 – Identificatie van de profielen
Elke rij in de tabel stelt een profiel voor. In de kolommen is per proefpersoon aangegeven hoe hij of zij elk van de vier profielen geïdentificeerd heeft. De identificatie was bijna altijd correct: één keer werden de profielen A en B met elkaar verward.

4.4.6 Rekenlast

Maximale frametijd
Om de minimale frequentie te bepalen waarmee het fuzzy systeem doorgerekend moet worden, ben ik begonnen met een frequentie van 100 Hertz, om deze vervolgens steeds met een factor tien af te laten nemen. De minimale frequentie ligt bij tien Hertz, het proces oogt dan nog volledig continu. Als het systeem één keer per seconde of nog minder vaak wordt doorgerekend, begint het te haperen en wordt onstabiel.

Benodigde rekenkracht

Om een idee te krijgen van de benodigde rekenkracht voor het doorrekenen van een hiërarchisch fuzzy gedragsysteem, is het doorrekenen van FuzzyDriver getimed op een AMD Athlon 900MHz PC, onder Microsoft Windows 2000. Het één miljoen keer doorrekenen van het systeem neemt ongeveer 20 seconden in beslag. Per seconde kan het systeem dus zo'n 50.000 keer doorgerekend worden, wat neerkomt op circa 0.02 microseconde per keer. In het theoretische geval dat de computer geheel toegewijd zou zijn aan het doorrekenen van FuzzyDrivers, zouden er dus 5000 stuks tegelijk kunnen rondrijden op de minimale doorrekenfrequentie van tien Hertz.

5 Conclusies en aanbevelingen

5.1 Conclusies

5.1.1 Samenvatting

Het autobestuurdersgedrag kan eenvoudig gemodelleerd worden met een hiërarchisch fuzzy logic systeem. De heuristieken van de taak (de expertkennis) kunnen in bijna natuurlijke taal worden ingevoerd. Dit houdt de applicatie bovendien doorzichtig en eenvoudig aan te passen. Het nauwkeurig afstellen van het systeem is pas na het invoeren van de heuristieken met behulp van een simulatie uit te voeren, via een trial-and-error proces. De hiërarchische ordening van de gedragsmodules komt het overzicht ten goede en maakt het hergebruik van lagere-orde modules mogelijk.

Het systeem is geschikt voor een incrementeel ontwikkelproces. Door het modulaire karakter van het systeem kan het gedragsspectrum uitgebreid worden, zonder dat de balans verstoord wordt en het geheel herschreven moet worden.

Het gebruik van fuzzy logic draagt bij aan natuurlijk ogend gedrag. Het voorkómen van oscillaties en plotselinge, bruuske bewegingen is inherent aan het gebruik van deze vage logica.

5.1.2 Ontwerpproces

De ontwikkelkosten van een systeem als FuzzyDriver zijn laag, ook in vergelijk met de andere onderzochte modelleermethoden. De oorzaken hiervan zijn de volgende:

- Het model maakt gebruik van expertkennis. Daar het de zeer wijd verspreide kennis van de autobestuurderstaak betreft, valt er goedkoop aan die kennis te komen. Bruikbare experts zijn alom aanwezig, ze hoeven enkel een rijopleiding te hebben gehad.
- De analogie tussen de representatie van de expertkennis in natuurlijke taal en die als productieregels in het systeem is groot. De vertaalstap tussen die twee is derhalve zeer gering en brengt dus weinig kosten met zich mee.
- De implementator hoeft geen programmeur te zijn. Hij of zij hoeft enkel te werken met een programma zoals fuzzyTECH. Wel dient hij of zij bekende te zijn met de architectuur en de implementatievoorschriften.
- De methode leent zich voor een incrementeel ontwerpproces. De voordelen hiervan zijn divers. Het belangrijkste punt aangaande de ontwerpkosten is dat uitbreiding plaats kan vinden met behoud van het reeds ontwikkelde model.

5.1.3 Implementatie

Het hiërarchische fuzzy systeem heeft een heldere en doorzichtige structuur:

- De kern van het systeem bestaat uit als zodanig herkenbare expertkennis.
- De schematische weergave van het systeem in de interface van fuzzyTECH komt de overzichtelijkheid ten goede.
- De modulaire opzet maakt het lokaliseren van al dan niet problematische gedragsaspecten tot een eenvoudige taak.

- Doordat het systeem enkel gebruik maakt van feedforward is de informatiestroom en de causaliteit zeer helder.
- Er zijn genoeg defuzzificatiemethodes beschikbaar om coördinatie correct te laten verlopen. Maar er zijn wel verschillende methodes nodig en de keuze daarvoor kost tijd.

5.1.4 Prestaties

Systeemrespons
De systeemrespons voldoet aan de verwachtingen. De vijf groepen metingen die verricht zijn weerspiegelen elk het geïsoleerde effect van één bepaalde operationele gedragsmodule. Deze resultaten helpen dus niet direct bij het beantwoorden van de onderzoeksvraag; deze is immers toegespitst op het tactisch niveau.

Gedrag - metingen
De simulatie en de closed loop metingen die daarmee gedaan zijn spelen een centrale rol in het beantwoorden van de onderzoeksvraag. Ze kunnen over een langere periode het interactieve bestuurdersgedrag laten zien. Ook de 'grotere' beslissingen – zoals het wel of niet inhalen en het kiezen van de rijstrook – worden zichtbaar. Dit zijn de beslissingen die op het tactisch niveau worden gemaakt.
De belangrijkste observatie is dat FuzzyDriver (met het modale profiel) in alle geteste situaties de juiste beslissing op het juiste moment maakt, zowel verkeerskundig als wat betreft veiligheid en efficiëntie.
Daarnaast kan geconcludeerd worden dat FuzzyDriver voorspelbaar reageert op veranderingen in de prioriteiten. Het veroorzaken van botsingen is bijvoorbeeld geen gewenst gedrag, maar het is wel conform de verwachtingen als je het 'bots niet' gewicht verlaagd.

Gedrag - observaties
FuzzyDriver is in staat veilig en efficiënt te rijden.
Het gedrag van FuzzyDriver oogt natuurlijk in termen van soepelheid, geleidelijkheid en het uitblijven van oscillaties. De extreme regelmaat in het gedrag wordt wel als onnatuurlijk ervaren.

Verschillende rijstijlen
Het is mogelijk om met FuzzyDriver verschillende rijstijlen te creëren, slechts door het aanpassen van een viertal parameters. Die rijstijlen zijn vervolgens voor mensen (automobilisten) herkenbaar.

Rekenlast
In tegenstelling tot de opmerkingen uit [2] is FuzzyDriver computationeel efficiënt. Efficiënt genoeg voor real-time toepassingen en het simuleren van duizenden FuzzyDrivers tegelijkertijd met behulp van een doorsnee PC.

5.2 Aanbevelingen

5.2.1 Doel van de simulatie

Partijen die gebruik willen maken van een hiërarchisch fuzzy systeem voor het simuleren van autobestuurdersgedrag dienen zich eerst af te vragen wat het doel is van de simulatie. Dit doel zal bepaalde eisen stellen aan de simulatie, waaraan een systeem als FuzzyDriver in het ene geval zeer goed zal voldoen en in het andere geval juist niet.
De onderzochte modelleermethode voldoet aan de volgende eisen:

1. Het uiterlijke gedrag van de kunstmatige bestuurders doet realistisch aan.
2. FuzzyDriver combineert planmatig en reactief rijgedrag.
3. Het systeem kan eenvoudig geconfigureerd worden voor het simuleren van verschillende rijstijlen
4. Er kunnen vele (duizenden) bestuurders tegelijk gesimuleerd worden op een modaal systeem.
5. Het systeem is deterministisch

Voorts wordt er aan de volgende eisen *niet* voldaan:

1. Het systeem is niet bedoeld om het exacte gedrag van de bestuurder van moment tot moment van te voren te kunnen bepalen.
2. FuzzyDriver is inwendig niet per se een realistisch cognitief of psychologisch model van een menselijke autobestuurder.

5.2.2 Beschikbare middelen

Mensen
Voor het ontwikkelen van een systeem als FuzzyDrivers zijn de volgende mensen benodigd:

- één of meerdere experts,
- een kennisacquisiteur en
- een ontwikkelaar.

De experts hoeven enkel verstand te hebben van het besturen van een auto. Het in het bezit zijn van een rijbewijs B is hier voldoende. De kennisacquisiteur dient interviewtechnieken te beheersen en kennis te hebben van het drie-taakniveau's paradigma en de wijze waarop complexere verkeershandelingen uit elementaire handelingen kunnen zijn opgebouwd. De ontwikkelaar, tot slot, heeft kennis van fuzzyTECH of een vergelijkbaar pakket, is bekend met de architectuur van een systeem als FuzzyDriver en hij of zij kan communiceren met de programmeurs die de eventuele simulatie verzorgen.
Van de kennisacquisiteur en de ontwikkelaar wordt een HBO/WO opleiding verwacht. Het is overigens niet ondenkbaar dat de twee functies door één persoon worden vervuld.

Software en hardware
In de meeste gevallen zal een modale PC met een pakket als FuzzyTECH en een eventuele programmeeromgeving voor de simulatie voldoen.

Tijd
De benodigde hoeveelheid tijd is afhankelijk van de omvang van de taak die men wil simuleren. De architectuur van het systeem staat reeds vast, dus dat wiel hoeft niet opnieuw uitgevonden te worden. Wel is het verstandig ruim tijd in te plannen voor de testfase(n), aan de hand waarvan het systeem mogelijk een aantal keren bijgesteld of herzien dient te worden.

5.2.3 Ontwikkeltraject

In het ontwikkeltraject worden de volgende fasen onderscheiden:

1. Kennisacquisitie
2. Bepalen van de benodigde in- en uitvoervariabelen
3. Bepalen van de bijbehorende fuzzy sets danwel membership functies
4. Omzetten van expertkennis naar productieregels

5. Testen en bijstellen of herzien van het systeem

6 Suggesties voor verder onderzoek

6.1 Inleiding

De implementatie van FuzzyDriver heeft duidelijk gemaakt dat de hiërarchische fuzzy logic benadering uitermate geschikt is voor het modelleren van – in ieder geval een deel van – het autobestuurdersgedrag. Dit onderzoek is vanzelfsprekend, even als elk onderzoek, beperkt door de hoeveelheid beschikbare tijd. In eventueel verder onderzoek is het zinvol om te experimenteren met het uitbreiden van het systeem, het toepassen van alternatieven in de implementatiedetails en het doen van een gedegen empirische vergelijkingsonderzoek met de overige modelleermethoden, die genoemd werden in het literatuuronderzoek.

6.2 Uitbreiding van het systeem

6.2.1 Inleiding

De implementatie van FuzzyDriver is beperkt gebleven tot een specifiek deel van de taak. Een meer volledige implementatie zou veel meer verkeershandelingen aan moeten kunnen, om zich bijvoorbeeld in het veel complexere verkeer van een stadse omgeving te handhaven. Bovendien zouden er meer componenten, zoals bijvoorbeeld perceptie, realistisch geïmplementeerd kunnen worden.

6.2.2 Uitbreiding van het mogelijk gedrag

Mijn experimenten aangaande de uitbreidbaarheid van het systeem met meer gedragsmodules hadden een positieve uitslag. Toch is dit onderzoek onvoldoende om dit type systeem als volkomen schaalbaar te beschouwen. Ik ben zeer benieuwd wat een drastische uitbreiding van de functionaliteit – zoals hieronder beschreven bijvoorbeeld – doet met dingen zoals de overzichtelijkheid van het systeem, de mate van controle over de rijstijl via de strategische gewichten en de onderlinge onafhankelijkheid van de primitieve gedragsmodules.

FuzzyDriver is gemaakt om zich te kunnen handhaven op een recht stuk snelweg. Dit is zoals gezegd een bewuste taakafbakening geweest. De omvang en complexiteit van de complete rijtaak waartoe de menselijke bestuurder in staat is, is vele malen groter. Een logische volgende stap zou zijn om FuzzyDriver de functionaliteit te geven om zich te handhaven buiten de snelweg. Deze uitbreiding van het takenpakket bevat zaken als het omgaan met gelijkvloerse kruisingen, voorrangsregels, verkeerslichten, ongemotoriseerd verkeer, voetgangers, maar ook het – al dan niet dynamisch – plannen van de route en zelfs, boven alles uitstijgend, de keuze van een vervoermiddel.

Mijn voorspelling is dat de mate van controle over de rijstijl via de strategische gewichten ook na een dergelijke forse uitbreiding even sterk zal blijven. De reden om dit te denken is dat deze mate van controle inherent is aan de hiërarchische opzet. Gebleken is dat elke coördinerende regel, in zijn eenvoud als regel op zich, volkomen helder en eenduidig één van de vier geformuleerde strategische doelen dient en als zodanig ook enkel gekoppeld is aan het daarbij horende gewicht.

Zowel de overzichtelijkheid van het systeem als de onafhankelijkheid van de primitieve gedragsmodules zijn niet per se inherent aan de architectuur. Hierbij ligt een grote verantwoordelijkheid bij de implementator. De overzichtelijkheid wordt behouden door het slim groeperen van de coördinerende regels. Vooruitlopend op het onafhankelijk houden van de primitieve gedragsmodules

het volgende: Reeds bij het implementeren van het beperkte deel van de rijtaak uit dit onderzoek heb ik gemerkt dat invoervariabelen, bijvoorbeeld afstand en snelheid ten opzichte van andere voertuigen, al snel door meerdere modules gebruikt worden. Bij het kalibreren van de membership functies voor deze variabelen bleek al dat het uitvoeren van correcties ten bate van het ene gedragsaspect ongewenst invloed kunnen hebben op andere gedragingen die afhankelijk zijn van dezelfde variabelen. Waarschijnlijk is het noodzakelijk om onderscheid te maken tussen bijvoorbeeld de variabele 'afstand tot auto linksvoor' gebruikt voor het afstand houden en diezelfde variabele gebruikt voor bijvoorbeeld het bepalen van het moment van inhalen. De afstand in meters die voor het ene geval 'te dichtbij' betekent kan voor het andere geval zeer goed anders zijn. Mits deze distinctie correct wordt gemaakt en toegepast, zal de onafhankelijkheid van de primitieve gedragsmodules – en daarmee de eenvoudige uitbreidbaarheid van het systeem – gehandhaafd kunnen blijven, ook na een forse uitbreiding.

6.2.3 Integratie van perceptie in het hiërarchische fuzzy logic systeem

Eveneens vraag ik me af hoe andere componenten uit de informatiekringloop die zich binnen de menselijke bestuurder afspelen, zoals perceptie, gemodelleerd en geïntegreerd kunnen worden in de hiërarchische fuzzy logic opzet. Mijn ideeën voor de praktische uitvoering hiervan zijn beperkt. Een uitstekende voorzet zou wellicht het voorstel voor de fuzzy perceptie van Automan [6] kunnen zijn, die bij mijn weten nog nooit geïmplementeerd is.

6.3 Aanpassingen aan het systeem

6.3.1 Inleiding

De keuzes die gemaakt zijn in de details van de implementatie zijn veelal onderbouwd (zie hoofdstukken 3 en 4). Evengoed zijn er mogelijkheden voor het experimenteren met wijzigingen binnen de huidige implementatie. De uitslag van zulke experimenten kunnen de gemaakte keuzes bevestigen of juist ontkrachten. In de nu volgende subparagraaf staat een aantal suggesties.

6.3.2 Voorstellen voor aanpassingen

Definitie van de rijstroken
De implementatie van FuzzyDriver is momenteel zeer specifiek ingericht naar het manoeuvreren over een snelweg met precies twee rijstroken. Alles is gedefinieerd in termen van 'links', 'rechts'. Beter zou zijn om een meer generieke opzet te hanteren, waarmee FuzzyDriver elk willekeurig aantal rijstroken aankan. Een dergelijke opzet is waarschijnlijk wel af te leiden uit de huidige implementatie, door de termen 'links' en 'rechts' relatief ten opzichte van de huidige rijstrook te zien in plaats van ten opzichte van de gehele weg. Wel zullen er dan vaak minstens drie rijstroken in ogenschouw genomen moeten worden, namelijk de huidige rijstrook, en de beide direct links en rechts daarvan. Een nog meer verfijnde implementatie zal ook informatie over het verkeer op de eventuele rijstroken die nog verder weg liggen meenemen bij het maken van bepaalde beslissingen.

Wisselen van rijstrook
De huidige versie van FuzzyDriver wisselt van rijstrook door het verleggen van de aan te houden laterale positie naar het midden van die andere rijstrook. Indirect zorgt dan de gedragsmodule 'volg huidige rijstrook' voor de wisseling. Wellicht is het zinvol deze gedragingen van elkaar los te koppelen.

Frametijden
Deze resultaten aangaande de maximale frametijden komen overeen met de opmerking in het literatuuronderzoek dat bij de menselijke bestuurder de processen op het laagste taakniveau, namelijk het operationele niveau, zo'n 10 tot 500 milliseconden duren. In het literatuuronderzoek staat ook vermeld dat de processen op de hogere taakniveaus langer duren en lagere frequenties hebben. Het is zeer goed mogelijk dat de processen op dit niveau bij de kunstmatige bestuurder ook minder vaak doorgerekend hoeven te worden. Het experimenteren met dit gegeven viel buiten het tijdsbestek van dit onderzoek en kunnen een mooi uitgangspunt vormen voor verder onderzoek.

6.4 Vergelijking met andere modelleermethoden

Om de geschiktheid van een hiërarchische fuzzy logic systeem te bepalen als modelleermethode voor autobestuurdersgedrag heb ik mijn bevindingen met het implementeren ervan naast die van andere methoden uit de literatuur gelegd.
De ultieme vergelijking zou, wat mij betreft, gemaakt kunnen worden door de verschillende modelleermethoden echt te benchmarken. Deze vergelijking zou vorm gegeven kunnen worden als een wedstrijd, waarin verschillende gelijkwaardige teams elk aan de hand van één van de verschillende methoden een kunstmatige autobestuurder implementeren. Elke bestuurder wordt gemaakt om (hetzelfde onderdeel van) de rijtaak uit te kunnen voeren. Vervolgens zou men onder andere met de methoden uit dit onderzoek, waaronder het gebruiken van een soort jury van experts, de resultaten onderling kunnen vergelijken.

Literatuurlijst

[1] Aasman, J. (1995) *Modelling driver behavior in Soar*. Dissertatie, Rijksuniversiteit Groningen. Leidschendam: KPN Research.

[2] Bayarri, S. & Férnandez, M. (1996) 'Design & validation of traffic models in driving simulation experiments'. Symposium on the design and validation of driving simulators, Valencia, Spanje.

[3] Coa, M. & Hall, E., (1998) 'Fuzzy logic control for an automated guided vehicle'. In: *Proceedings of the intelligent robots and computer vision XVII, SPIE conference, volume 3522*. Boston, MA.

[4] Cnossen, F. (2000) *Adaptive strategies and goal management in car drivers*. Dissertatie, Rijksuniversiteit Groningen.

[5] Fraichard, T. & Garnier, P. (2000) 'Fuzzy control to drive car-like vehicles'. In: *Robotics and autonomous systems, volume 34, nummer 1*, 1-22.

[6] Heemskerk, M., Quispel, L. & Warris, S. (1999) 'Automan'. Projectverslag, Rijksuniversiteit Groningen.

[7] McKnight, A.J. & Adams, B.B. (1970) *Driver education task analysis, volume 1: Task descriptions*. Washington, DC: National Highway Traffic Safety Administration.

[8] Michon, J.A. (1985) 'A critical view of driver behavior models: what do we know, what should we do?'. In: Evans, L. & Schwing, R.C. *Human behavior and traffic safety*, 485-520. New York: Plenum Press.

[9] Michon, J.A. (1989) 'Modellen van bestuurdersgedrag'. In: Knippenberg, C.W.F. van, Rothengatter, J.A. & Michon, J.A. (redacteurs), *Handboek sociale verkeerskunde*, 207-231. Assen: Van Gorcum.

[10] Pirjanian, P. & Mataric, M. (1999) 'A decision-theoretic approach to fuzzy behavior coordination'. In: *Proceedings of the 1999 IEEE International Symposium on Computational Intelligence in Robotics & Automation (CIRA'99)*, Monteray, CA.

[11] Ross, T. J. (1995) *Fuzzy logic with engineering applications*. New York: McGraw-Hill.

[12] Saffiotti, A., Ruspini, E.H. & Konolige, K. (1993) 'Blending reactivity and goal-directedness in a fuzzy controller'. In: *Proceedings of the 2nd IEEE conference on fuzzy systems*, 134-139. San Fransisco, CA.

[13] Saffiotti, A. (1997) 'Fuzzy logic in autonomous robotics: behavior coordination'. In: *Proceedings of the 6th IEEE international conference on fuzzy systems*, 573-578. Barcelona, Spanje.

[14] Salvucci, D. D., Boer, E.R. & Liu, A. (2001). 'Toward an integrated model of driver behavior in a cognitive architecture'. In: *Transportation research record, nummer 1779*.

[15] Sukthankar, R. (1994) 'Situational awareness for driving in traffic'. Dissertatie, Robotics Institute, School of Computer Science, Carnegie Mellon University.

[16] Tunstel, E. (1995) 'Coordination of distributed fuzzy behavior in mobile robot control'. In: *IEEE International Conference on Systems, Man and Cybernetics*, 4009-4014. Vancouver, BC, Canada.

[17] Tunstel, E., Danny, H., Lippincott, T. & Jamshidi, M. (1997) 'Adaptive fuzzy-behavior hierarchy for autonomous navigation'. In: *Proceedings of the IEEE International Conference on Robotics and Automation, volume 1*, 829-834. Albuquerque, NM.

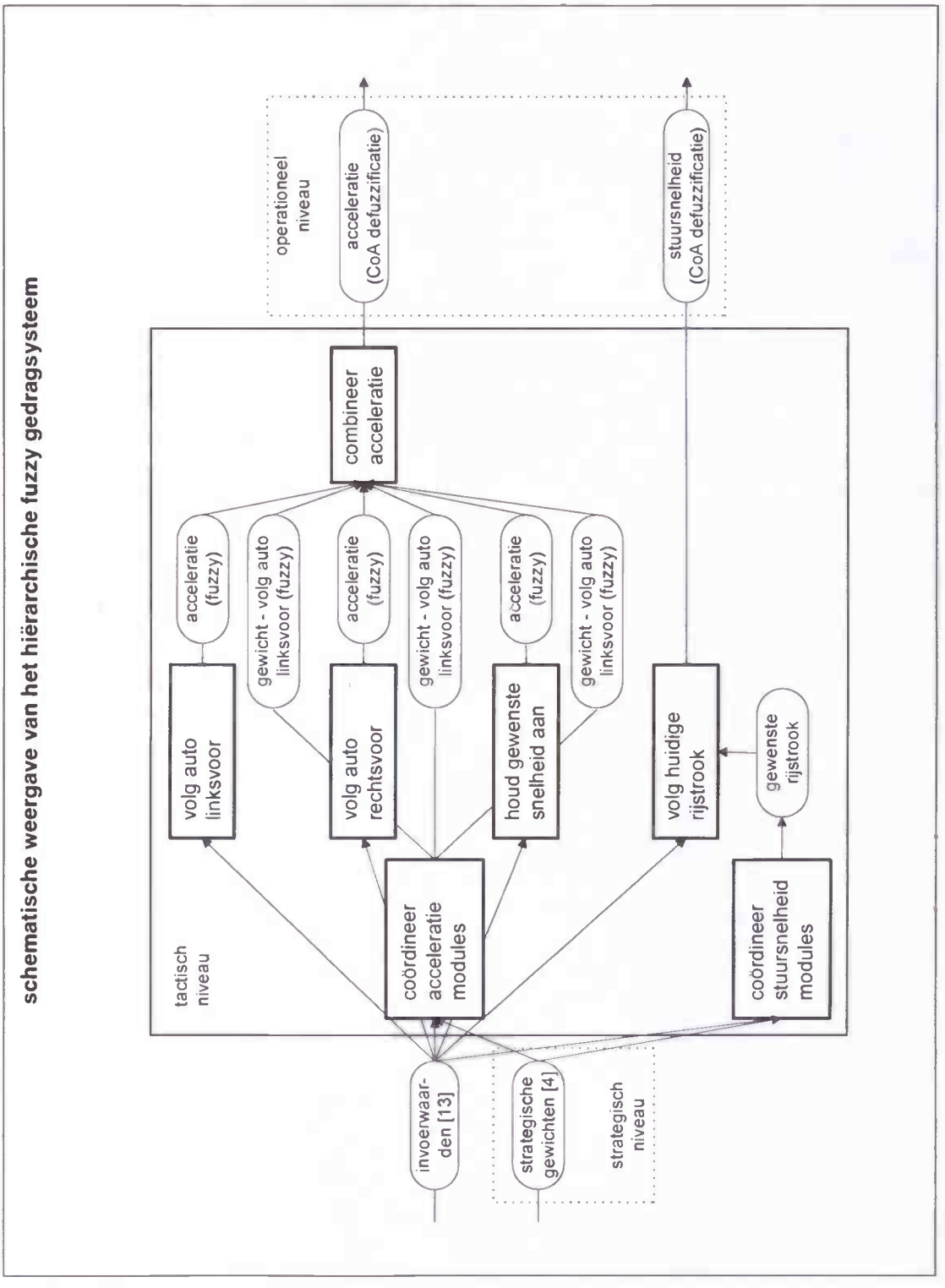
[18] Voudouris, C., Chernett, P., Wang, C.J. & Callaghan, V.L. (1994) 'Fuzzy hierarchical control for autonomous vehicles'. University of Essex, Dept. of Computer Science. Colchester, UK. Technical Report CSM-216.

[19] Wierda, M. (1993) *PACT, een geautomatiseerde rij-instructie in een virtuele omgeving.* Verkeerskundig Studiecentrum, Haren.

[20] Zadeh, L.A. (1965) 'Fuzzy Sets'. In: *Information and control, volume 8*, 338-353.

Bijlage A – Beschrijving van het fuzzy systeem

In deze bijlage vind u een beschrijving van het fuzzy systeem, zoals gebruikt in de uiteindelijke implementatie van FuzzyDriver. Het systeem wordt hieronder schematisch weergegeven en op de volgende pagina's zijn de gebruikte productieregels per module uitgeschreven. Daarbij heb ik mij beperkt tot de productieregels van de coördinerende modules, aangezien die van de primitieve gedragsmodules geen centrale rol in het onderzoek hebben.



Regels voor coördinatie van de acceleratiemodules:

Als je achter iemand rijdt, moet je je snelheid aan die van hem aanpassen:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_linksvoor is dichtbij EN huidige_rijstrook is linker	**DAN**	volg_auto_links_voor_gewicht is groot

Je mag niet rechts inhalen:

ALS	volg_regels_gewicht is groot EN afstand_tot_auto_linksvoor is dichtbij EN huidige_rijstrook is rechter	**DAN**	volg_auto_links_voor_gewicht is groot

Als je achter iemand rijdt, moet je je snelheid aan die van hem aanpassen:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_rechtsvoor is dichtbij EN huidige_rijstrook is rechter	**DAN**	volg_auto_rechts_voor_gewicht is groot

Rijd met de snelheid die je wenst:

ALS	ga_snel_gewicht is groot	**DAN**	houd_gewenste_snelheid_aan_gewicht is groot

Regels voor coördinatie van de stuurmodule (bepalen van de te volgen rijstrook):

Als je afslag dichtbij is, kun je beter rechts gaan rijden:

ALS	volg_route_gewicht is groot EN afstand_tot_afslag is dichtbij	**DAN**	gewenste_rijstrook is rechter

Haal langzame auto's in:

ALS	ga_snel_gewicht is groot EN afstand_tot_auto_rechtsvoor is dichtbij EN snelheid_auto_rechtsvoor is te_langzaam	**DAN**	gewenste_rijstrook is linker

Blijf niet onnodig links rijden:

ALS	volg_regels_gewicht is groot EN afstand_tot_auto_rechtsvoor is ver EN snelheid_auto_rechtsvoor is ok	**DAN**	gewenste_rijstrook is rechter

Haal niet in als je zelf ingehaald wordt:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_linksachter is dichtbij EN relatieve_snelheid_auto_linksachter is sneller EN huidige_rijstrook is rechter	**DAN**	gewenste_rijstrook is rechter

Haal niet in als er een auto vlak linksachter je zit:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_linksachter is te_dichtbij EN huidige_rijstrook is rechter	**DAN**	gewenste_rijstrook is rechter

Haal niet in als er een auto vlak linksvoor je zit:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_linksvoor is te_dichtbij EN huidige_rijstrook is rechter	**DAN**	gewenste_rijstrook is rechter

Ga niet naar rechts als een auto nog niet ver genoeg ingehaald hebt:

ALS	bots_niet_gewicht is groot EN afstand_tot_auto_rechtsachter is te_dichtbij EN huidige_rijstrook is linker	**DAN**	gewenste_rijstrook is linker

Blijf op je huidige_rijstrook (deze regels hebben 10% van het gewicht van de overige regels):

ALS	huidige_rijstrook is linker	**DAN**	gewenste_rijstrook is linker
ALS	huidige_rijstrook is rechter	**DAN**	gewenste_rijstrook is rechter

Regels voor het aanhouden van de gewenste snelheid:

Als je huidige snelheid lager is dan je gewenste snelheid, accelereer dan:
ALS afwijking_gewenste_snelheid is te_langzaam **DAN** acceleratie is accelereren

Als je huidige snelheid gelijk is aan je gewenste snelheid, accelereer noch rem dan:
ALS afwijking_gewenste_snelheid is correct **DAN** acceleratie is nul

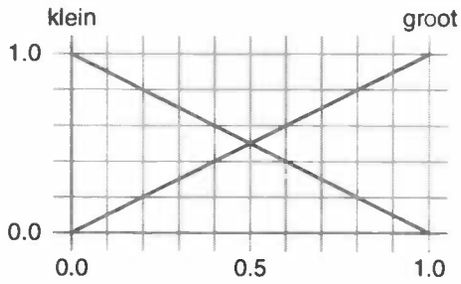
Als je huidige snelheid hoger is dan je gewenste snelheid, rem dan af:
ALS afwijking_gewenste_snelheid is te_snel **DAN** acceleratie is remmen

Regels voor het volgen van je voorganger

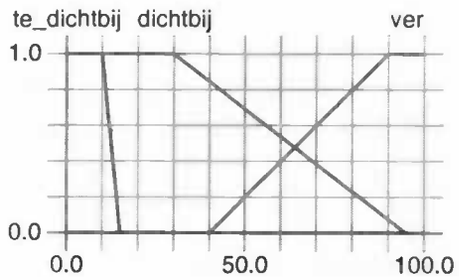
Als je te dicht op je voorganger rijdt, rem dan af:
ALS afstand_tot_auto_**xxx**voor is te_dichtbij **DAN** acceleratie is remmen

Als je dicht op je voorganger rijdt en je snelheid is hoger dan die van hem, rem dan af:
ALS afstand_tot_auto_**xxx**voor is dichtbij EN **DAN** acceleratie is remmen
snelheidsverschil_auto_**xxx**voor is positief

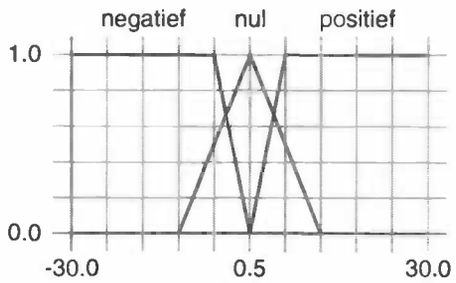
Membershipfuncties van de fuzzy invoervariabelen



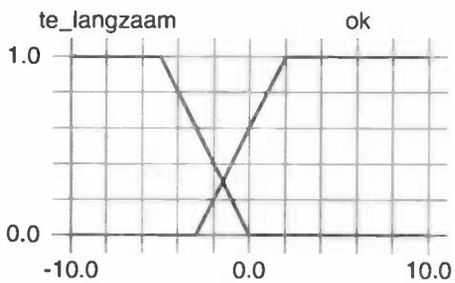
xxx_gewicht



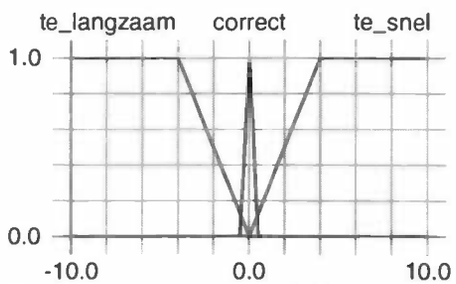
afstand_tot_auto_xxx



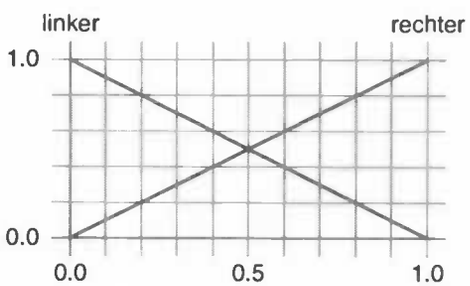
snelheidsverschil_auto_xxx



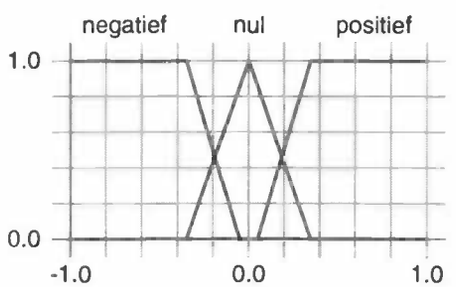
afwijking_gewenste_snelheid_auto_xxx



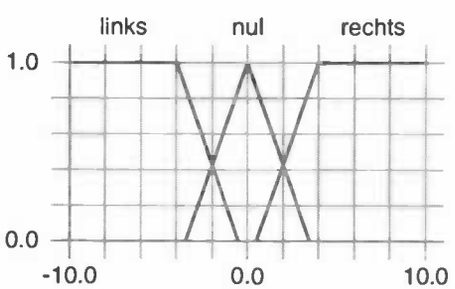
afwijking_gewenste_snelheid



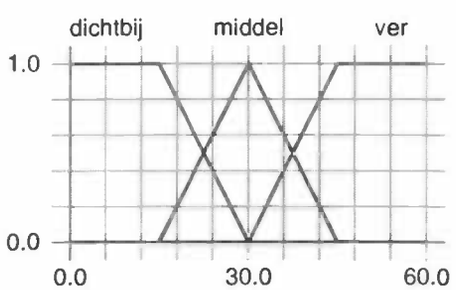
huidige_rijstrook



laterale_snelheid

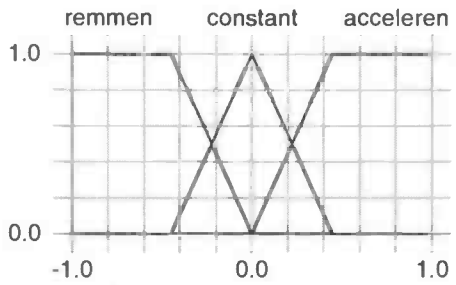


laterale_positie_tov_middellijn

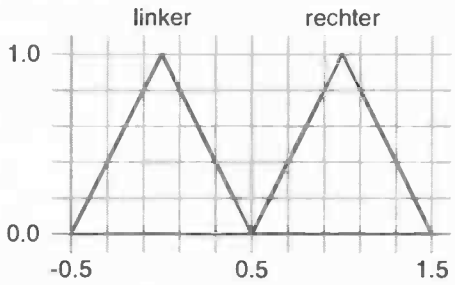


tijd_tot_afslag

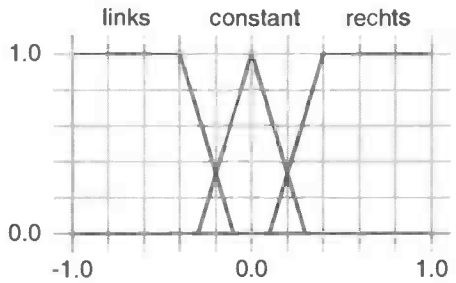
Membershipfuncties van de fuzzy uitvoervariabelen



acceleratie



gewenste_rijstrook



stuursnelheid