

Katholieke Universiteit Leuven

Faculteit Toegepaste Wetenschappen

Departement Werktuigkunde

Afdeling PMA

Celestijnenlaan 300B, B-3001 LEUVEN (Heverlee), België

Tel: 016 / 32 24 80 Fax: 016 / 32 29 87

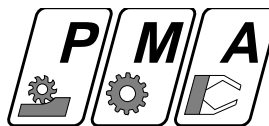


Promotor :

Assessoren :

Eindwerk, voorgedragen tot
bekomen van de graad van
Burgerlijk Ingenieur
door

*Production Engineering
Machine Design
Automation*



*Productietechnieken
Machinebouw
Automatisering*

© Copyright by K.U.Leuven

Zonder voorafgaande schriftelijke toestemming van de promotoren en de auteurs is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie in verband met het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wendt u zich tot de K.U.Leuven, Dept. Werktuigkunde, Afdeling PMA, Celestijnenlaan 300B, B-3001 Heverlee (België), tel. 016/322480.

Voorafgaande schriftelijke toestemming van de promotor is vereist voor het aanwenden van de in dit afstudeerwerk beschreven (originele) methoden, producten, toestellen, schakelingen en programma's voor industrieel nut en voor inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Onze welgemeende dank gaat uit naar alle personen die ons van dichtbij en veraf bijgestaan en geholpen hebben gedurende het verloop van dit eindwerk. In het bijzonder vermelden we enkele mensen wiens hulp onmisbaar was. Eerst en vooral willen we onze begeleidende assistent, R. Waarsing, bedanken voor zijn geduldige en altijd enthousiaste hulp bij diverse aspecten van dit eindwerk. Zijn superieure kennis van het programmeren in het algemeen en van C++ in het bijzonder was van grote waarde bij de implementatie van de sturing.

Verder danken wij professor H. Van Brussel en M. Nuttin voor hun begeleiding, hun voortdurende opvolging van het onderzoek en hun kritische bemerkingen. Deze hebben een grote bijdrage geleverd tot het bekomen resultaat.

Ook willen we graag professor H. Bruyninckx, K. Gadyne en P. Soetens bedanken voor de aangename sfeer op het departement en de geboden hulp. Af en toe sproten uit hun opmerkingen enkele vruchtbare ideeën voort.

Tenslotte mogen we onze ouders zeker niet vergeten, die ons de mogelijkheid gegeven hebben om deze opleiding te volgen. Ook gedurende het verloop van dit eindwerk waren ze een steun, waarop altijd beroep gedaan kon worden.

Inhoudsopgave

1	Inleiding	1
1.1	Situering	1
1.2	Concrete opdracht	2
1.3	Gevolgde werkwijze	3
1.4	Resultaten	3
2	Literatuurstudie	5
2.1	Indeling van robotsturingen	5
2.1.1	Modelgebaseerd vs. niet-modelgebaseerd	5
2.1.2	Deliberatief vs. reactief	7
2.2	Gedragsgebaseerde sturing	9
2.2.1	Het concept van de gedragsgebaseerde sturing	9
2.2.2	Het combineren van gedragingen	11
2.3	De mobiele manipulator LiAS	13
2.3.1	Opbouw	13
2.3.2	Sensoren	14
3	De programma-architectuur	16
3.1	Keuze van de programma-architectuur	16
3.2	Het concept 'agent'	17
3.3	Design van de programma-architectuur	17
3.4	Componenten van de programma-architectuur	18
3.4.1	Multi Agent Controller	18
3.4.2	Sensoren en inputs	21
3.4.3	Outputs en actuatoren	23
3.5	Implementatie van de programma-architectuur	24
3.5.1	ContollerAgent	25
3.5.2	Sensoren en inputs	26
3.5.3	Outputs en actuatoren	27
3.6	Besluit	28

4	Gedragsgebaseerde mobiele manipulatie	30
4.1	De mobiele manipulator opsplitsen	30
4.2	Het complexe gedrag opsplitsen in basisgedragingen	32
4.2.1	Navigeren naar de deur	33
4.2.2	Openen van de deur	33
4.2.3	Navigeren door de deuropening	40
4.2.4	Opeenvolging van fases	42
4.2.5	Berekenen van gewrichtssnelheden	43
4.2.6	Gehele taak	44
4.2.7	Overgang tussen verschillende fases	45
4.3	Werking van de agents	45
4.3.1	FollowForce	45
4.3.2	MoveArm en MoveBase	49
4.3.3	TurnHandle	50
4.3.4	PullDoor	50
4.3.5	LimitArmlength	51
4.3.6	LimitWristAngle	52
4.3.7	GotoOptimalPos	52
4.3.8	KinInverse	54
4.3.9	KinTranspose	54
4.4	Werking van de coördinatie-objecten	55
4.4.1	WeightedSum	55
4.4.2	Competitief	56
4.4.3	Parallel	56
4.4.4	Sequence	57
5	Resultaten	58
5.1	Overzicht van de beweging	58
5.2	Benadering vanuit de deelgedragingen	58
5.2.1	Het draaien van de deurklink	59
5.2.2	Het trekken aan de deur	61
5.2.3	Het beperken van de manipulatorlengte	63
5.2.4	Het beperken van de polshoek	65
5.3	Benadering vanuit regeltechnisch standpunt	66
6	Conclusie en aanbevelingen	69
6.1	Conclusie over de gedragsgebaseerde theorie	69
6.2	Aanbevelingen voor verder onderzoek	70
6.2.1	Uitbreidingen voor de programma-architectuur	70
6.2.2	Uitbreidingen van dit eindwerk	70
A	Implementatie van de klassen in C++	72
	Bibliografie	73

Lijst van figuren

2.1	Een onderverdeling van robotsturingen	6
2.2	Situering van de gedragsgebaseerde sturing	9
2.3	Arbitratie op basis van de activatieniveaus	11
2.4	Hiërarchische arbitratie. De 'S' betekent 'Suppress'.	12
2.5	Combinatie op basis van de gewogen som	12
2.6	Combinatie op basis van stemmen	12
2.7	De mobiele manipulator LiAS	13
2.8	Het bereik van de manipulator	14
2.9	De grijper op het uiteinde van de manipulator	14
2.10	De laserscanner	15
3.1	Overzicht van de programma architectuur	18
3.2	Schematische weergave van een agent	19
3.3	Schematische weergave van een agency	20
3.4	De werking van sensoren en inputs	22
3.5	De werking van outputs en actuatoren	23
3.6	De structuur van de programma-architectuur	25
3.7	De hiërarchische structuur van sensoren en inputs	27
3.8	De hiërarchische structuur van outputs en actuatoren	28
4.1	De gewrichten van de manipulator	32
4.2	Het navigeren naar een deur	34
4.3	Verschillende methodes voor het draaien van een deurklink	35
4.4	De beweging van de grijper staat steeds loodrecht op de deur	36
4.5	De lengte van de manipulator moet beperkt blijven	37
4.6	De hoek van het polsgewricht moet beperkt blijven	37
4.7	Beweging volgens de vrijheidsgraad van de deur	38
4.8	De combinatie van de gedragingen van de manipulator	39
4.9	De combinatie van de gedragingen van het mobiele platform	39
4.10	Het openen van een deur	41
4.11	De deuropening localiseren	42
4.12	Navigeren door de deuropening	43
4.13	De opeenvolging van fases	43

4.14	De berekening van de gewrichtssnelheden	44
4.15	De gehele mobiele manipulatietaak	45
4.16	Voorstelling van de gedefiniëerde assenstelsels	46
4.17	De eindeffector van LiAS	47
4.18	Het activatie-niveau van TurnHandle	50
4.19	Het activatie-niveau van PullDoor	50
4.20	Het activatie-niveau van LimitArmLength	51
4.21	Het activatie-niveau van LimitWristAngle	52
4.22	Het activatie-niveau van GotoOptimalPos	53
5.1	Het traject dat LiAS aflegt	59
5.2	De beweging van LiAS bij het openen van een deur	60
5.3	De activatieniveau's doorheen het openen van de deur	61
5.4	Het draaien van de deurklink	62
5.5	Het moment rond de X-as van de grijper.	62
5.6	Het trekken aan de deur	63
5.7	Het activatieniveau van LimitArmLength	64
5.8	Het activatieniveau van LimitWristAngle	65
5.9	Regeltechnische benadering van de gedragingen	66
5.10	Impedantiekarakteristieken	67
A.1	Inhoud van de cd-rom	72

Hoofdstuk 1

Inleiding

1.1 Situering

Binnen de robotica, en doorheen de geschiedenis ervan, zijn verschillende domeinen te onderscheiden. Een eerste domein is dat van de industriële robot. Deze robots staan meestal statisch opgesteld en worden gekenmerkt door een hoge precisie en snelheid. Het is een domein waarin veel onderzoek en ontwikkeling gedaan werd en wordt. Vrij haaks erop, staat het domein van de mobiele robot. Deze robots opereren vaak in een dynamische omgeving. Veiligheid en flexibiliteit zijn in dit domein belangrijker dan snelheid en precisie.

Bij het begin van de robotica, rond de jaren '70, was onderzoek en ontwikkeling voornamelijk toegespitst op de industriërobot en haar toepassingen [16]. Deze industriële toepassingen, zoals montage of verpakking van stukgoederen, vormen in veel bedrijven nog steeds het enige toepassingsgebied van robots. Ze bestaan meestal uit een cyclische herhaling van eenzelfde reeks voorgeprogrammeerde handelingen. Hierbij neemt de robot de omgeving dus niet waar. Verder is de werkomgeving van de robot statisch bepaald en voor de robot geoptimaliseerd.

Naar mate de technologie vorderde, werden uitgebreidere robotsturingen beschikbaar. De integratie van sensoren laat toe een hogere nauwkeurigheid of performantie te bekomen [7]. Voorbeelden hiervan zijn de uitrusting van een lasrobot met inductieve sensoren om een naad te volgen of het gebruik van krachtsensoren bij robots die instaan voor montagetoeepassingen. Robots met dergelijke sturingen bezitten reeds een beperkte vorm van waarneming van en reactie op de omgeving.

Hoewel nog steeds veel onderzoek gedaan wordt op het gebied van de industriële robot, hebben recentere ontwikkelingen steeds vaker betrekking op

mobiele robots [11]. Een toepassingsgebied hiervan is dat van de service-robots. Deze service-robots zijn mobiele robots die de mens bijstaan bij dagdagelijkse handelingen. Voorbeelden van dergelijke taken zijn het rond-dragen van post in een bedrijf en het schoonmaken van ruimtes.

Een ander voorbeeld van een service-toepassing, waarbij meer manipulatie te pas komt, is de assistentie van mindervaliden. Zo kan een op een rolstoel gemonteerde robotarm gebruikt worden om deuren of lades te openen, om voorwerpen op te nemen en te verplaatsen, etc. Verschillende gradaties van autonomie zijn hierbij mogelijk: de mindervalide kan de robotarm volledig zelf bedienen of de robotsturing kan bepaalde onderdelen tot zelfs de volledige afhandeling van de taak op zich nemen.

Typisch aan deze service-taken is dat ze uitgevoerd worden in een aan de mens aangepaste omgeving. Een dergelijke omgeving is niet gestandaardiseerd en in hoge mate indeterministisch. Zo blijven bijvoorbeeld de afmetingen van een huiskamer over het algemeen constant, maar tafels verschuiven, mensen lopen rond,... De omgeving waarin service-robots hun taken uitvoeren is dus intrinsiek suboptimaal als werkomgeving voor robots. De banaalste taken, vanuit menselijk standpunt, vormen daardoor vaak een vrij groot probleem voor robots. De implementatie van dergelijke handelingen op een robot is dus een grote uitdaging. Een dergelijke uitdaging, namelijk het openen van een deur met een mobiele manipulator, werd dan ook gekozen als onderwerp voor dit eindwerk.

1.2 Concrete opdracht

De concrete opdracht van dit eindwerk is het uitvoeren van een manipulatietaak met een mobiele robot, op een gedragsgebaseerde manier. Bij het inzetten van robots in de service-sector, is manipulatie erg belangrijk. Naast navigatie, vormt manipulatie een essentieel onderdeel om met een robot dagdagelijkse taken uit te voeren [15].

In dit eindwerk is de uit te voeren manipulatietaak het openen van een deur met behulp van de mobiele manipulator LiAS (Leuven intelligent Autonomous System).

De gedragsgebaseerde sturing van robots gaat uit van een aantal elementaire gedragingen, die gecombineerd worden om zo tot het uiteindelijke, gewenste gedrag van de robot te komen. Wat dit precies inhoudt en wat de voordelen en nadelen zijn van een dergelijke aanpak, wordt uitvoerig behandeld in paragraaf 2.2.

1.3 Gevolgde werkwijze

Een literatuurstudie vormde het begin van het eindwerk. Vervolgens werd de gebruikte programma-architectuur bestudeerd en uitgewerkt. Daarna werden specifieke gedragingen opgesteld en binnen de architectuur geïmplementeerd.

De literatuurstudie (hoofdstuk 2) bestaat uit drie paragrafen. De eerste paragraaf geeft een overzicht van enkele verschillende strekkingen binnen de robotica. De volgende paragraaf bespreekt uitvoerig de gedragsgebaseerde sturing en situeert deze binnen de besproken strekkingen. Tenslotte wordt de mobiele manipulator die ter beschikking stond, LiAS, besproken.

Een robuuste en flexibele programma-architectuur vormt de basis van elk uitgebreid programma. Vandaar ook dat een belangrijke stap bij de afhandeling van het eindwerk bestond uit het bestuderen en uitwerken van die architectuur. De globale structuur van deze architectuur –de ruggegraat ervan, als het ware– was reeds beschikbaar. De specifieke invulling van alle componenten diende echter nog te gebeuren. Hoofdstuk 3 behandelt de gedetailleerde beschrijving van de programma-architectuur, de componenten ervan en hun implementatie.

Eens de programma-architectuur volledig uitgewerkt was, werden gedragingen opgesteld om een deur te openen. Deze werden dan binnen de architectuur geïmplementeerd en uitgebreid getest op LiAS. Hoofdstuk 4 gaat uitvoerig in op deze gedragingen en hun implementatie.

Hoofdstuk 5 tenslotte, bespreekt de behaalde resultaten en geeft een visie op een aantal mogelijke domeinen voor verder onderzoek.

1.4 Resultaten

De resultaten van dit eindwerk zijn tweevoudig.

De eerste verwezenlijking binnen dit eindwerk was de uitwerking van de programma-architectuur. Bij het begin van dit eindwerk was de architectuur volledig nieuw, slechts vrij summier getest en bevatte deze naast een aantal bugs nog conceptuele fouten. Tevens bestond de architectuur toen slechts uit een lege omkadering. Binnen deze omkadering werden alle ontbrekende onderdelen om LiAS aan te sturen ingevuld en uitvoerig getest. Hierbij kwamen dan ook de meeste fouten aan het licht en konden deze opgelost worden.

Vervolgens werden de gedragingen opgesteld die samen leiden tot het openen

van een deur. Dit was de ultieme test van de architectuur, en het eigenlijk onderwerp van dit eindwerk. Deze gedragingen werden geïmplementeerd en de verschillende parameters werden empirisch bepaald. Het slagen van deze opzet toont enerzijds aan dat de architectuur uitgerijpt is en ook werkt voor ingewikkelde problemen. Anderzijds bewijst het dat complexe manipulatie-taken mogelijk zijn op een gedragsgebaseerde manier.

Hoofdstuk 2

Literatuurstudie

Dit hoofdstuk bespreekt de literatuurstudie. Een eerste paragraaf behandelt twee mogelijke criteria om robotsturingen onder te verdelen. Deze paragraaf schept hiermee het nodige kader om de gedragsgebaseerde sturing te situeren. De gedragsgebaseerde sturing wordt besproken in de tweede paragraaf. De derde paragraaf tenslotte, gaat dieper in op de mobiele manipulator die ter beschikking stond: LiAS.

2.1 Indeling van robotsturingen

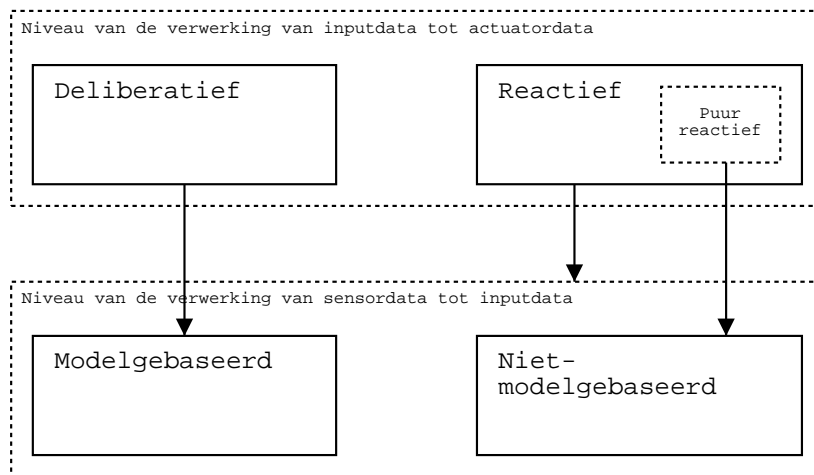
Elke vorm van robotsturing komt neer op het periodiek bepalen van geschikte waardes om de actuatoren van de robot in te stellen. Dergelijke waardes zijn bijvoorbeeld de hoeksnelheden voor de actuatoren van een manipulator of een rotatie- en een translatiesnelheid voor een mobiel platform. Bij de berekening ervan, wordt vaak uitgegaan van sensordata, al dan niet geïnterpreteerd in een model. Een robotsturing berekent dus, eventueel op basis van sensorwaardes, de stuurdata om de actuatoren in te stellen.

Deze paragraaf bespreekt twee mogelijke criteria om robotsturingen onder te verdelen. Het eerste criterium verdeelt de robotsturingen in de modelgebaseerde sturingen en de niet-modelgebaseerde sturingen. Het tweede criterium verdeelt de sturingen in deliberatieve en reactieve sturingen¹. Figuur 2.1 geeft een schematisch overzicht.

2.1.1 Modelgebaseerd vs. niet-modelgebaseerd

Bij het puur *modelgebaseerd sturen* worden de sensorwaardes opgenomen in een model van de omgeving [2]. De vorm van het model ligt vooraf vast.

¹De literatuur is niet erg specifiek in de definitie van deze begrippen. In veel werken worden de begrippen uitgelegd en gebruikt, maar een formele, algemeen aanvaarde definitie is er niet. Deze paragraaf bespreekt dan ook hoe de verschillende begrippen binnen dit eindwerk opgevat zijn.



Figuur 2.1: Een onderverdeling van robotsturingen

De invulling ervan, dus de data die in het model opgenomen wordt, kan reeds vooraf grotendeels vastliggen op basis van voorkennis over de omgeving. In dit geval dienen de sensorwaardes enkel om de data in het model accuraat te houden. Indien gewerkt wordt zonder voorkennis over de omgeving, wordt de data in het model volledig opgebouwd uit sensorwaardes. In beide gevallen worden echter alle sensorwaardes geïnterpreteerd en in het model opgenomen. Het model vormt de enige representatie van de omgeving binnen het systeem en vormt de basis om actuatorwaardes te berekenen.

Aan het gebruik van een model zijn een aantal voordelen verbonden. De opbouw van een model transformeert de ruwe sensordata naar een nieuwe vorm. Deze kan geschikter zijn om op een intuïtieve manier beslissingen te nemen. Zo bieden de ruwe outputwaardes van twee camera's vrij weinig rechtstreeks bruikbare informatie. Als op basis van deze data echter een 3D-model van de omgeving opgebouwd wordt, staat deze nieuwe representatie toe zinnige acties te ondernemen binnen de omgeving.

Een ander voordeel is het geschiedenis-aspect van een model. Elke sensorwaarde wordt opgenomen in het model en zorgt dus voor een zekere aanpassing van het model. Niet enkel de momentane sensorwaardes, maar dus ook de vorige sensorwaardes, kunnen een rol spelen bij de berekening van nieuwe aanstuurwaardes voor de actuatoren.

Het gebruik van een model heeft ook twee grote nadelen. Vooreerst heeft elk model een beperkte nauwkeurigheid. De ontwerper van het model bepaalt hoe gedetailleerd het model in het beste geval met de werkelijkheid overeen kan komen. Bij een stijgende mate van detail, verhoogt de complexiteit van het model gevoelig. Bij de omzetting van ruwe sensordata naar een model

gaat dus steeds informatie verloren.

Een ander nadeel, verbonden aan een model, is dat het slechts een beperkte tijd accuraat blijft. In dynamische en ongestructureerde omgevingen, zoals de omgevingen waarin mensen rondlopen, levert het accuraat houden van een model grote problemen op.

Bij het andere extreem, de *niet-modelgebaseerde aanpak*, worden de sensorwaardes niet opgenomen in een model, maar rechtstreeks aangewend [6]. Hierbij gaat dus geen informatie verloren. Vermits de sensorwaardes niet eerst geïnterpreteerd worden, staat de niet-modelgebaseerde sturing toe hogere reactiesnelheden te behalen op veranderingen in de omgeving. De keerzijde hiervan is, dat ook kortstondige veranderingen in de sensorwaardes, zoals bijvoorbeeld meetfouten, sterk gevolgd worden. Bij de niet-modelgebaseerde sturing is er ook geen geschiedenis: er kan enkel gereageerd worden op wat momentaan waargenomen wordt.

2.1.2 Deliberatief vs. reactief

In een *deliberatieve sturing* worden de handelingen van de robot gepland [12]. Hiervoor is een model van de omgeving nodig. Het model dient als basis om te plannen welke acties ondernomen worden. De ondernomen actie is dus niet direct gerelateerd aan de waarneming, maar volgt uit het redeneren over het opgebouwde model. Op basis van het model worden mogelijke acties en de resultaten van die acties bepaald. Afhankelijk van een bepaald criterium (bijvoorbeeld minimale tijd om de taak te volbrengen), wordt dan beslist tot welk van de mogelijke acties overgegaan wordt. Plannen leidt dus tot optimale performantie ten opzichte van het gebruikte model en criterium.

Een voorbeeld is de navigatie van een robot binnen een gebouw. Op basis van een representatie van het gebouw (dit kan een kaart zijn, met de locaties van kamers, deuren, gangen, ...) bepaalt de deliberatieve sturing een pad om van één locatie naar een andere te gaan. Indien er meerdere mogelijkheden zijn, kan bijvoorbeeld het snelste pad gekozen worden of het pad waarbij het minst energie verbruikt wordt.

Zonder abstracte representatie van de omgeving en zonder redenering op basis van die representatie, is het heel moeilijk om tot een goede afhandeling te komen voor taken zoals de navigatietask uit het voorbeeld. Bij dergelijke taken is deliberatieve sturing dus de meest voor de hand liggende oplossing. Een ander voordeel van de deliberatieve sturing is, dat deze expliciet toelaat de verschillende mogelijke acties af te wegen op basis van een criterium.

Vermits deliberatieve sturing steunt op het redeneren in een model, be-

zit deze sturing ook alle nadelen, verbonden aan het gebruik van een model. De veronderstelling dat het model een betrouwbare representatie van de omgeving vormt staat centraal bij de deliberatieve sturing. Indien hieraan niet voldaan is, kunnen de genomen beslissingen volledig verkeerd zijn. In ongekende omgevingen dient het model volledig opgebouwd te worden uit sensorwaardes. Dit opbouwen van een kaart is absoluut geen triviaal probleem. Het gebruik van deliberatieve sturingen binnen een ongekende omgeving gaat dus gepaard met de nodige moeilijkheden.

Een ander nadeel van de deliberatieve sturing is een tekort aan responsiviteit in ongestructureerde en dynamische omgevingen. Alle ondernomen acties worden bepaald door te plannen op basis van een model. Zowel de responsiviteit van het plannen zelf, als de snelheid waarmee een model zich aanpast aan nieuwe sensorwaardes, is relatief laag. Binnen dynamische omgevingen vertoont deliberatieve sturing dus tekortkomingen.

Bij *reactieve sturing*, daarentegen, is de output heel sterk gekoppeld aan de input. De kern van de reactieve sturing bestaat erin, dat er voor elke inputtoestand (dus elke mogelijke set inputwaardes) exact één overeenkomstige set outputwaardes voor de actuatoren bestaat. Deze relatie tussen input en output is bij het ontwerp van de sturing vastgelegd. Bij het bepalen van de outputwaardes wordt dus niet gepland.

De literatuur is niet erg specifiek over wat nu precies input mag zijn voor een reactief systeem [4, 13, 14]. Grofweg kunnen de meningen onderverdeeld worden in twee strekkingen. De eerste strekking gaat ervan uit dat reactieve systemen enkel gebruik mogen maken van rechtstreekse sensordata. Volgens deze strekking impliceert reactieve sturing dus dat de sturing niet-modelgebaseerd is.

De andere strekking stelt dat de input zowel kan bestaan uit rechtstreekse sensordata, als uit data, opgevraagd uit een model. Reactieve sturing kan volgens deze strekking dus zowel modelgebaseerd als niet-modelgebaseerd zijn, of bestaan uit een mengvorm van deze beiden.

In dit eindwerk wordt de tweede strekking beschouwd als de meest algemeen inzetbare en meest gangbare strekking. Naar deze vorm van robotsturing wordt verwezen als *reactieve sturing*. Naar de sturing volgens de eerste strekking, die een deelverzameling vormt van de reactieve sturing, wordt verwezen als *puur reactieve sturing*.

Een voordeel van reactieve sturing is de hoge reactiesnelheid. De sterke koppeling tussen input en output staat een heel snelle reactie toe op veranderingen in de omgeving. Deze hoge reactiesnelheid is vanzelfsprekend het meest uitgesproken bij puur reactieve sturing, vermits daarbij geen model gebruikt wordt.

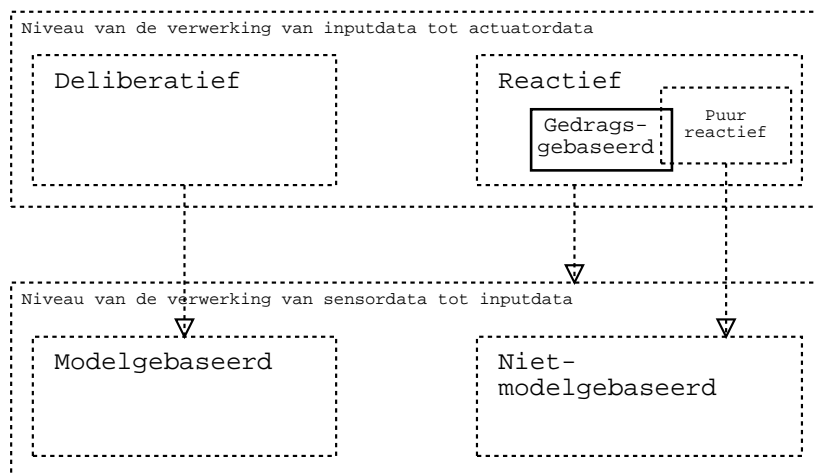
Tevens levert de reactieve sturing eenvoudige oplossingen voor toepassingen zoals bijvoorbeeld het ontwijken van hindernissen.

Een nadeel aan de reactieve sturing is, dat een aantal controleproblemen veel moeilijker, misschien zelfs onmogelijk te realiseren zijn op een reactieve manier. Bij reactieve sturing gebeurt alle planning door de ontwerper, vermits deze de relatie tussen input en output vastlegt [14]. Om taken, zoals bijvoorbeeld complexe navigatietaken binnen een gebouw², reactief uit te voeren, moeten voor elke mogelijke start- en eindlocatie de relaties tussen input en output vastgelegd worden. Het ontwerp van een reactieve sturing voor dergelijke taken is dus erg uitgebreid.

2.2 Gedragsgebaseerde sturing

2.2.1 Het concept van de gedragsgebaseerde sturing

De gedragsgebaseerde sturing is een reactieve sturing (zie figuur 2.2). Het gedrag van de robot is dus sterk gekoppeld aan wat de robot waarneemt. Het gedrag van de robot wordt niet bepaald door te plannen op basis van een model. Het gebruik van een model is wel toegestaan, maar het model levert enkel (een deel van) de set inputwaardes, die de inputtoestand bepalen. Met elke inputtoestand komt dan een set outputwaardes voor de actuatoren overeen.



Figuur 2.2: Situering van de gedragsgebaseerde sturing

De gedragsgebaseerde theorie gaat ervan uit dat elk complex gedrag, dus elke complexe taak, onder te verdelen valt in deelgedragingen, die elk op

²Hier worden uitgebreide navigatietaken bedoeld, zoals 'Ga naar het secretariaat en haal een fax op.' en niet meer elementaire navigatietaken, zoals 'Rij vooruit en ontwijk hindernissen'. Voor deze laatste navigatietaken is een reactieve sturing meer aangewezen, wegens het dynamische karakter van de taak.

zich een onafhankelijk, minder complex stuurprobleem vormen [4]. Het verder opsplitsen van de deelgedragingen leidt dan uiteindelijk tot elementaire gedragingen. Deze elementaire gedragingen zijn relatief eenvoudige relaties tussen waarneming en ondernomen actie. Elk van deze elementaire gedragingen bepaalt dus op een reactieve manier aanstuurwaardes voor de actuatoren.

Om tot het volledige, complexe gedrag te komen, worden alle berekende aanstuurwaardes gecombineerd. De manier waarop deze combinatie gebeurt is specifiek aan de uit te voeren taak en wordt besproken in paragraaf 2.2.2.

Een eenvoudig voorbeeld is het navigeren van een mobiel platform naar een bepaalde positie, in een ruimte met hindernissen. Deze complexe gedraging kan onderverdeeld worden in twee deelgedragingen:

- Een gedraging die beweegt in de richting van de eindpositie. Deze gedraging heeft bijvoorbeeld als output een translatiesnelheid voor het platform. De snelheid is gericht van het mobiel platform naar de eindpositie en bijvoorbeeld evenredig met de afstand tot die positie. Verder is de snelheid begrensd op een maximale waarde.
- Een gedraging die hindernissen ontwijkt. Als bijvoorbeeld links voor het platform een object waargenomen wordt, dan is de output van deze gedraging een snelheid naar rechts.

Beide gedragingen leveren eigen aanstuurwaardes voor de actuatoren. Aangestuurd met de waardes van één van de gedragingen, wordt enkel de overeenkomstige deelgedraging gerealiseerd (zoals in de richting van de eindpositie rijden of hindernissen ontwijken). Het combineren van de aanstuurwaardes, bijvoorbeeld door de gewogen som te nemen, leidt tot de uiteindelijke aanstuurwaardes. Aangestuurd met deze waardes, voert de robot het complexe gedrag uit.

De meest fundamentele eigenschap van de gedragsgebaseerde sturing is dus de gedecentraliseerde structuur. Een gedragsgebaseerde sturing bestaat uit een aantal parallel functionerende gedragingen. Er is geen centraal beslissingsorgaan dat redeneert om tot actie over te gaan.

Dat de gedecentraliseerde structuur niet leidt tot een onoverzichtelijke complexiteit, is te danken aan de minimale afhankelijkheid en communicatie tussen de gedragingen. Bij de gedragsgebaseerde sturing, vormt elke gedraging een aparte entiteit. De communicatie tussen de gedragingen verloopt voornamelijk via de omgeving. De uitwisseling van kennis binnen het systeem gebeurt overwegend impliciet, door de omgeving waar te nemen [14].

2.2.2 Het combineren van gedragingen

Elke deelgedraging bepaalt aanstuurwaarden voor de actuatoren. Aangestuurd met deze waarden voert de robot de deelgedraging uit. Om tot het volledige, complexe gedrag te komen, dienen de gedragingen (of dus eigenlijk eerder hun output) gecombineerd te worden.

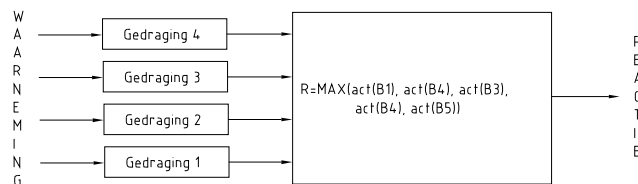
Er bestaan talrijke mogelijkheden om gedragingen te combineren [17]. Twee groepen combinatiemethodes zijn de competitieve methodes en de coöperatieve methodes.

Competitieve methodes

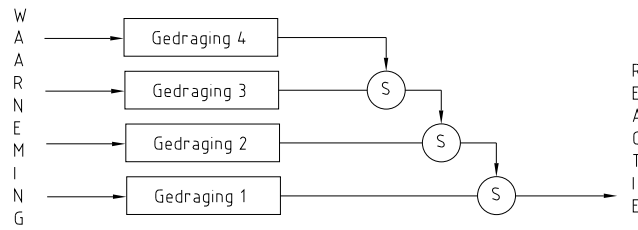
Competitieve methodes combineren meerdere gedragingen, die niet samen actief mogen zijn. Slechts één van de gedragingen krijgt de toelating om actief te zijn. Competitieve combineringsmethodes zijn dus arbitraire methodes. Deze arbitratie kan op verschillende manieren gebeuren. Twee vaak voorkomende manieren zijn de arbitratie op basis van de activatieniveau's van de gedragingen en de arbitratie op basis van een strikte hiërarchie tussen de gedragingen.

In de eerste manier (zie figuur 2.3) bepalen de gedragingen zelf hoe actief ze willen zijn. Zo kan een gedraging die hindernissen ontwijkt zichzelf inactief maken als er geen hindernissen gedetecteerd worden binnen een bepaalde afstand. De gedraging met het hoogste activatie-niveau krijgt de controle over de uiteindelijke output.

In de tweede manier (zie figuur 2.4) bepalen de gedragingen enkel of ze actief willen zijn of niet. De gedragingen staan in een hiërarchische volgorde. Als een gedraging actief wil zijn, dan onderdrukt deze de uitvoer van alle gedragingen die lager staan in de hiërarchie. De uiteindelijke output is dus die van de hiërarchisch hoogste gedraging die actief wil zijn.



Figuur 2.3: Arbitratie op basis van de activatieniveau's

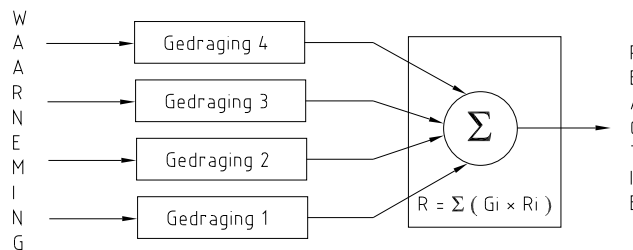


Figuur 2.4: Hiërarchische arbitratie. De 'S' betekent 'Suppress'.

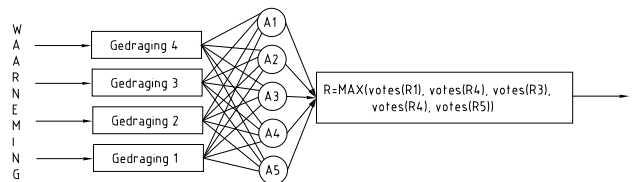
Coöperatieve methodes

Coöperatieve methodes combineren meerdere gedragingen, die wel samen actief mogen zijn. De uiteindelijke output is een combinatie van de output van elk van de gedragingen. Een vaak voorkomende manier is het nemen van de gewogen som van alle aanstuurwaardes, met de activatieniveau's van de gedragingen als wegingsfactor (zie figuur 2.5).

Indien er slechts een discreet aantal aanstuurwaardes mogelijk zijn, is het mogelijk een keuze te maken op basis van een mechanisme van stemmen (zie figuur 2.6). Elke gedraging krijgt een aantal stemmen toegekend, die het mag verdelen over de mogelijke aanstuurwaardes. De set aanstuurwaardes met het meeste stemmen wordt uiteindelijk gekozen.



Figuur 2.5: Combinatie op basis van de gewogen som



Figuur 2.6: Combinatie op basis van stemmen

2.3 De mobiele manipulator LiAS

De mobiele manipulator die ter beschikking stond in het kader van dit eindwerk, is LiAS (Leuven Intelligent Autonomous System). Figuur 2.7 toont een foto van LiAS. Deze paragraaf behandelt eerst de opbouw van LiAS. Vervolgens komen de sensoren van LiAS aan bod.



Figuur 2.7: De mobiele manipulator LiAS

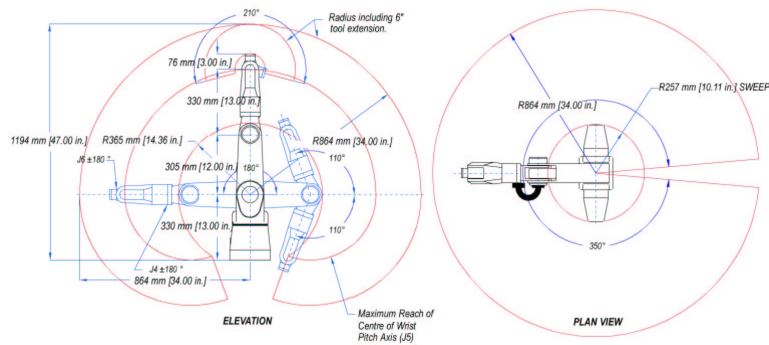
2.3.1 Opbouw

LiAS bestaat uit een mobiel platform, met daarop een manipulator. Het mobiel platform heeft vier wielen. De twee wielen vooraan zijn vrij draaiende zwenkwielen. De twee wielen achteraan worden onafhankelijk van elkaar aangedreven. Met behulp van deze twee onafhankelijk aanstuurbare vrijheidsgraden kan het mobiel platform elke gewenste positie en oriëntatie in een vlak aannemen ³.

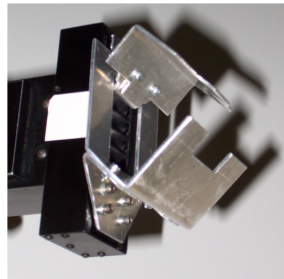
Bovenop het mobiele platform is een manipulator gemonteerd (zie figuur 2.7). Deze manipulator is een zes-assige industriële robotarm. Elk van de zes assen is onafhankelijk aanstuurbaar. De manipulator kan dus elke gewenste positie en oriëntatie binnen zijn werkingsruimte aannemen. De nominale last, die de arm kan dragen, bedraagt 3 kg.

Aan het uiteinde van de manipulator is een grijper bevestigd. Figuur 2.9 toont hiervan een detailfoto. De grijper werd in het kader van dit eindwerk ontwikkeld. Ze is dan ook geschikt voor het vastklemmen van een deurklink.

³Vanzelfsprekend kunnen de drie vrijheidsgraden in het vlak niet onafhankelijk ingesteld worden, vermits het mobiel platform slechts twee actuatoren bezit.



Figuur 2.8: Het bereik van de manipulator



Figuur 2.9: De grijper op het uiteinde van de manipulator

2.3.2 Sensoren

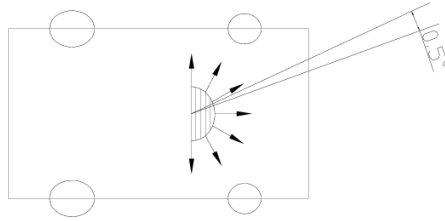
LiAS is uitgerust met verschillende sensoren: een laserscanner, 16 ultrasone sensoren, 2 encoders om de positie van de mobiele basis te bepalen, 6 encoders op de positie van de manipulator te bepalen, een krachtsensor, twee camera's en een gyroscoop.

Laserscanner

De laserscanner is aangebracht aan de voorkant van het mobiele platform en scant de omgeving af over een bereik van 180° . Op elke 0.5° binnen dit bereik meet de scanner de afstand tot het dichtbijzijnde voorwerp, in discrete stappen van 5 cm , met een maximaal bereik van 50 m . De meetfrequentie bedraagt 1.4 Hz (Figuur 2.10). Zelfs voorwerpen met een zeer lage reflectiviteit en wanden onder een grote invalshoek (tot 80° t.o.v. de normale) worden waargenomen.

Ultrasone sensoren

In het mobiel platform zijn 16 ultrasone sensoren aangebracht. De volgorde waarin deze afgevuurd worden is instelbaar. Door deze volgorde zó te kie-



Figuur 2.10: De laserscanner

zen, dat nooit twee nabije sensoren na elkaar afgevuurd worden, wordt de kans op cross-talk sterk verminderd. Vermits de sensoren zowel zenden als ontvangen, hebben de sensoren een dode zone.

Krachtsensor

Tussen het uiteinde van de arm en de grijper bevindt zich een krachtsensor. Deze meet drie krachten en drie momenten in onderling loodrechte richting. In de z -richting kunnen krachten tot $130N$ gemeten worden, met een resolutie van $0.10N$. In de x - en y -richting kunnen krachten tot $65N$ gemeten worden, met een resolutie van $0.05N$. Momenten worden gemeten met een resolutie van $0.003Nmm$. De maximale meetfrequentie bedraagt $\frac{1000}{6} Hz$.

Hoofdstuk 3

De programma-architectuur

Het uiteindelijke doel van dit eindwerk is de gedragsgebaseerde sturing van een mobiele manipulator. Zoals bij elk uitgebreid programma, is er ook bij deze sturing nood aan een robuuste en flexibele programma-architectuur. In dit eindwerk werd geopteerd voor een programma-architectuur op basis van agents [1].

De eerste paragraaf geeft een overzicht van de eisen waaraan de programma-architectuur moet voldoen. De volgende paragraaf bespreekt het concept van de programma-architectuur, legt uit wat agents zijn en verklaart waarom deze architectuur geschikt is voor een gedragsgebaseerde sturing van robots. De derde paragraaf geeft een overzicht van de opbouw van de architectuur. Deze bestaat uit vijf grote delen: sensoren, inputs, MAC, outputs en actuatoren. De werking van deze delen, en hun implementatie, wordt in de daarop volgende paragrafen in detail besproken.

3.1 Keuze van de programma-architectuur

In dit eindwerk wordt de mobiele manipulator LiAS op een gedragsgebaseerde manier aangestuurd. LiAS is uitgerust met meerdere sensoren (laserscanner, ultrasone sensoren, encoders van mobiele basis en manipulator) en actuatoren (mobiele platform, manipulator). Al deze sensoren en actuatoren hebben verschillende eigenschappen op het vlak van frequentie, snelheid, enz. Zo werken bijvoorbeeld de encoders op een hogere frequentie dan de laserscanner en worden actuatoren van de manipulator aan een hogere frequentie aangestuurd dan de actuatoren van het mobiele platform. De programma-architectuur moet dus meerdere sensoren en actuatoren, elk op hun eigen frequentie, kunnen aanspreken. [18]

De aansturing van de mobiele manipulator gebeurt volgens de gedragsgebaseerde theorie. De programma-architectuur moet dus een eenvoudige, flexibele manier aanbieden voor het implementeren en combineren van verschillende gedragingen.

In dit eindwerk werd geopteerd voor een programma-architectuur op basis van agents [1]. Deze architectuur voldoet aan alle eerder vermelde criteria. Het concept van deze architectuur is afkomstig van de Van Breemen architectuur [5]. Dit idee werd echter nog uitgebreid, met bijvoorbeeld de mogelijkheid om onderdelen van het systeem op verschillende frequenties te laten werken. De architectuur is geïmplementeerd in de object georiënteerde programmeertaal C++.

3.2 Het concept 'agent'

Een agent is een object dat instaat voor een welbepaalde taak. Elke agent heeft zijn eigen taak en kan beschouwd worden als de specialist in deze taak. Elke agent kan zijn taak uitvoeren, onafhankelijk van de andere agents. De gebruikte programma-architectuur is gebaseerd op agents en vormt een omkadering waarbinnen verschillende agents actief kunnen zijn.

Verschillende agents kunnen gegroepeerd worden in een agency. Een agency bundelt de krachten van alle agents die ertoe behoren. De taak die een agency uitvoert is een combinatie van alle taken die zijn agents uitvoeren.

Bij de realisatie van een robotsturing volgens de gedragsgebaseerde theorie, wordt het gewenste gedrag opgesplitst in verschillende eenvoudige basisgedragingen. Elk van deze basisgedragingen kan eenvoudig uitgevoerd worden door een agent. Door het combineren van agents tot een agency, worden verschillende basisgedragingen gecombineerd tot een meer complex gedrag. Een agency vormt dus op zich terug een agent, die de complexere gedraging uitvoert. Door verschillende agency's steeds verder te combineren binnen nieuwe agency's, ontstaat uiteindelijk een agency die het gewenste gedrag realiseert. De gekozen architectuur leent zich dus uitermate goed tot de realisatie van een robotsturing volgens de gedragsgebaseerde theorie.

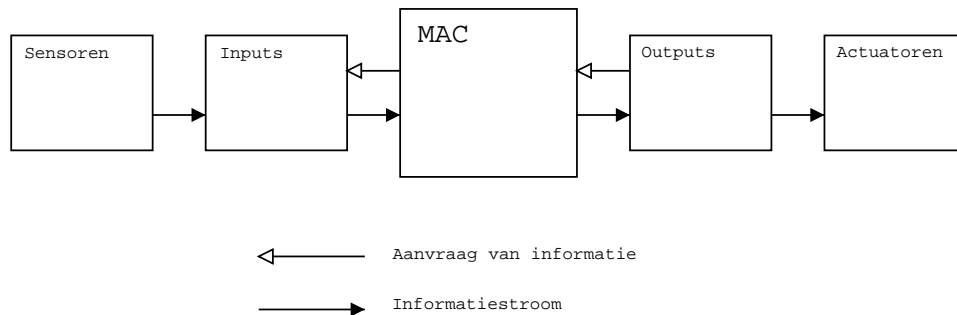
3.3 Design van de programma-architectuur

Figuur 3.1 geeft een schematisch overzicht van de programma-architectuur. Hierin zijn vijf delen te onderscheiden: sensoren, inputs, MAC, outputs en actuatoren. De Multi Agent Controller of MAC vormt het hart van de structuur. Binnen de MAC bevinden zich alle agents. Het is hier dat het gewenste gedrag wordt gerealiseerd. De inputs vormen buffers tussen de sensoren en de MAC. De outputs vormen buffers tussen de MAC en de actuatoren.

De informatiestroom binnen de architectuur vertrekt van de sensoren en loopt doorheen de structuur tot bij de actuatoren. De meetwaarden van de sensoren worden via buffers (de inputs) doorgegeven aan de MAC. Binnen

de MAC verwerken de agents de sensorinformatie tot aanstuurdata voor de actuatoren. De MAC zendt deze aanstuurdata over buffers (de outputs) naar de actuatoren.

In tegenstelling tot de informatiestroom, loopt de aanvraag van informatie in de omgekeerde richting. De outputs vragen nieuwe aanstuurdata voor de actuatoren aan bij de MAC. Deze stuurt dan op zijn beurt een aanvraag voor nieuwe sensordata aan de inputs. Door elke aanvraag van informatie komt er dus een nieuwe informatiestroom op gang.



Figuur 3.1: Overzicht van de programma architectuur

3.4 Componenten van de programma-architectuur

De programma-architectuur bestaat uit vijf grote delen: sensoren, inputs, MAC, outputs en actuatoren. De volgende paragraaf bespreekt in detail de werking van elk van deze delen.

3.4.1 Multi Agent Controller

De MAC vormt het hart van de hele structuur. Hij kan beschouwd worden als de organisator van het hele agent-gebeuren. De MAC bevat één agency, waarbinnen alle agents werkzaam zijn. Deze agency realiseert het gewenste gedrag van de robot.

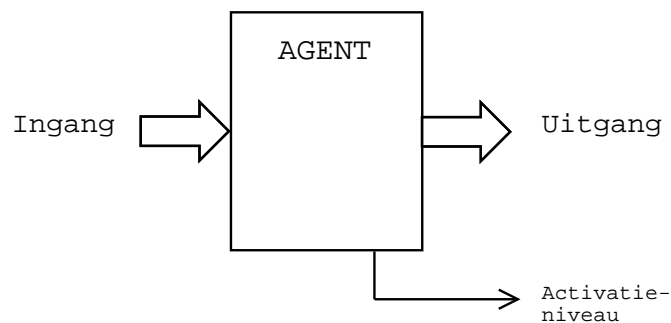
Er zijn drie belangrijke taken voor de MAC te onderscheiden:

- Op bepaalde tijdstippen geeft de MAC aan de agency te kennen dat de sensorinformatie moet verwerkt worden tot aanstuurdata voor de actuatoren. De MAC start dus een hele berekeningscyclus op, waarin alle agents hun 'gedrag' uitvoeren. Verder in deze paragraaf wordt duidelijk dat de MAC enkel een berekeningscyclus opstart, als de outputs hierom vragen.
- De MAC zorgt ervoor dat de inputs, aan het begin van elke cyclus, de meest recente sensorinformatie ter beschikking stellen van alle agents.

Het is belangrijk dat alle agents hun berekeningen uitvoeren met dezelfde sensordata. Gedurende het tijdsinterval dat de agents bezig zijn met hun berekeningen wordt de ter beschikking gestelde sensorinformatie niet gewijzigd, ook al zou een sensor nieuwe informatie aanbieden aan een input.

- Aan het einde van een berekeningscyclus geeft de MAC het resultaat van alle berekeningen, de sturingsdata voor de actuatoren, door aan de outputs. De outputs zorgen er dan voor dat de actuatoren aangestuurd worden.

Agent



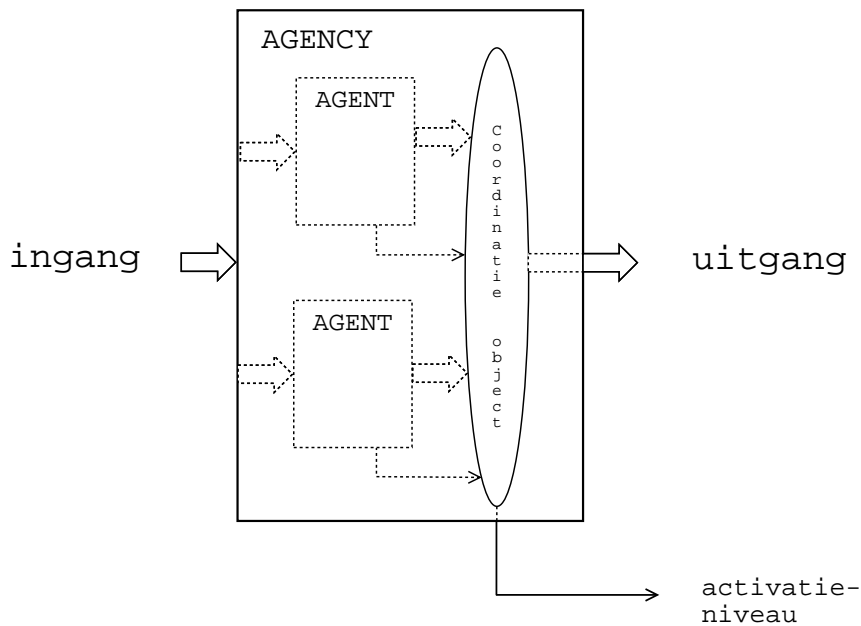
Figuur 3.2: Schematische weergave van een agent

Een agent heeft een activatie-niveau, een ingang en een uitgang. Figuur 3.2 geeft dit schematische weer. Het uitgangssignaal volgt uit berekeningen op het ingangssignaal. Het activatie-niveau van de agent maakt kenbaar in welke mate hij actief wil zijn. Dit niveau ligt ergens tussen volledig actief en volledig inactief. Dit activatie-niveau is echter slechts een wens die de agent uit. Of de agent ook daadwerkelijk actief mag worden, wordt van buitenaf bepaald. Een agent is dus een semi-autonoom object: hij voert zijn berekening volledig autonoom uit, maar beslist niet zelf wanneer hij actief mag zijn.

Omdat de robotsturing binnen dit eindwerk op een gedragsgebaseerde manier verloopt, bestaat de taak die elke agent opgelegd krijgt uit het uitvoeren van één basisgedraging. Dit is een eenvoudige berekening waarbij uit sensorinformatie van een of meerdere sensoren (ingang van de agent), sturingsdata voor een of meerdere actuatoren (uitgang van de agent) berekend wordt. Het tijdrovende opbouwen en raadplegen van een model, dat in een dynamische omgeving snel achterhaald is, wordt hierbij achterwege gelaten.

Agency

Een agency groepeert een aantal agents tot een nieuw geheel. Binnen de agency voert elke agent onveranderd zijn taak uit. De agent weet zelfs niet van het bestaan van de agency af. De agency gaat enkel alle deeltaken die de agents uitvoeren, combineren, tot één nieuwe, complexe taak. Een agency is dus vergelijkbaar met een agent die een complexe taak uitvoert. Een agency heeft, net als een agent, een activatie-niveau, een ingang en een



Figuur 3.3: Schematische weergave van een agency

uitgang. Figuur 3.3 geeft dit schematische weer. Het uitgangssignaal volgt uit de combinatie van de uitgangssignalen van alle agents binnen de agency. Het activatie-niveau van een agency maakt kenbaar in welke mate de agency actief wil zijn. Dit activatie-niveau is, net als bij de agents, slechts een wens die de agency uit. Of de agency ook daadwerkelijk actief mag worden, wordt van buitenaf bepaald.

Naar buiten toe gedraagt een agency zich dus exact zoals een agent. Het is dan ook duidelijk dat ook verschillende agency's gecombineerd kunnen worden binnen een nieuwe agency. Door dit combineren van agents en agency's binnen een nieuwe agency steeds opnieuw toe te passen, ontstaat er uiteindelijk één agency. Deze agency zorgt voor het gewenste gedrag van de robot.

De organisatie binnen een agency gebeurt door een coördinatie-object (Figuur 3.3). Er zijn drie belangrijke taken voor het coördinatie-object te onderscheiden:

- Het coördinatie-object bepaalt hoe de uitgangen van de verschillende agents gecombineerd worden tot de uitgang van de agency.
- Aan de hand van de activatie-niveaus van de agents, bepaalt het coördinatie-object het activatie-niveau van de agency .
- Het coördinatie-object bepaalt welke agents actief mogen zijn. Enkel de agents die nodig zijn voor het berekenen van de output van de agency, krijgen toelating actief te zijn.

Hoe dit combineren van uitgangen en berekenen van activatie-niveaus gebeurt, is verschillend voor elk coördinatie-object. Hier is een voorbeeld voor een competitief en een coöperatief coördinatie-object uitgewerkt:

- Een competitief coördinatie-object zal de agent met het hoogste activatie-niveau actief maken. Zowel de uitgang als het activatie-niveau van de agency zijn dezelfde als deze van de ene actieve agent.
- Een coöperatief coördinatie-object activeert elke agent die zelf actief wenst te zijn. De uitgang van de agency is de gewogen som van de uitgangen van de agents. Als wegingsfactor wordt het activatie-niveau van de overeenkomstige agent gebruikt. Het activatie-niveau van de agency is het activatie-niveau van de meest actieve agent.

3.4.2 Sensoren en inputs

Figuur 3.4 geeft een gedetailleerd beeld van de sensoren en inputs. Bij elke sensor hoort een overeenkomstige input.

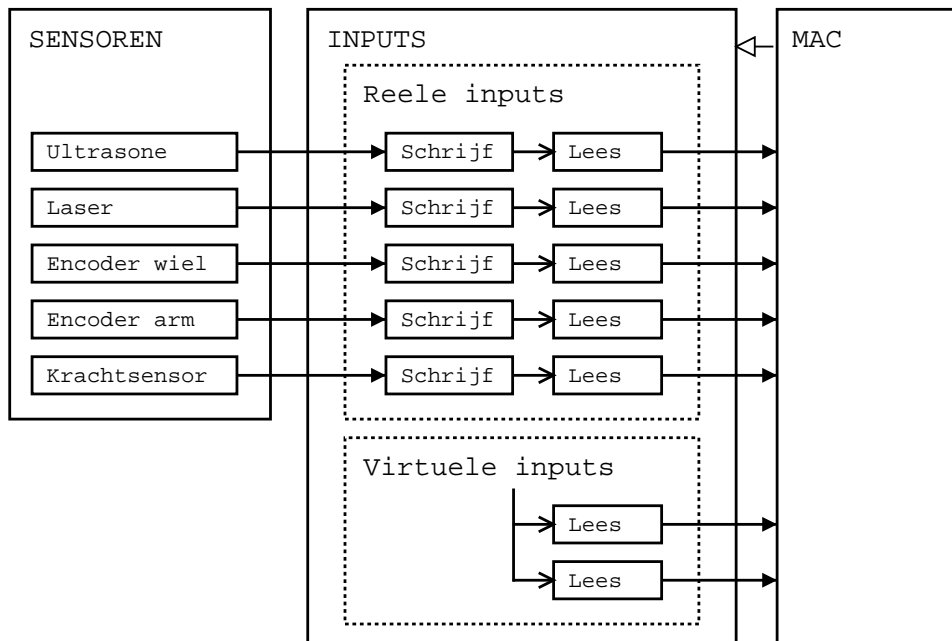
Sensoren

Binnen 'sensoren' bevinden zich alle fysisch aanwezige sensoren: de ultrasone sensoren, de laserscanner, de encoders van de mobiele basis en de manipulator, en de krachtsensor. Elk van deze sensoren werkt op een eigen frequentie, onafhankelijk van de andere sensoren. Telkens als een sensor nieuwe informatie ter beschikking heeft, zendt hij deze door naar zijn input. Elke sensor kan dus beschouwd worden als een autonoom, onafhankelijk proces dat telkens de meest recente sensorinformatie doorgeeft aan zijn input.

Inputs

Binnen 'inputs' zijn twee categoriën te onderscheiden: de reële inputs en de virtuele inputs.

- Een reële input bestaat uit één schrijf- en één leesbuffer per fysische sensor. De informatie die een sensor aanbiedt, wordt opgeslagen in



Figuur 3.4: De werking van sensoren en inputs

de overeenkomstige schrijfbuffer. Een schrijfbuffer bevat dus altijd de meest recente sensorinformatie. Het is echter niet voorspelbaar wanneer de informatie in deze schrijfbuffer zal vernieuwd worden, aangezien de sensoren, die dit doen, autonoom werken.

Een leesbuffer is toegankelijk voor alle agents. Telkens als er een nieuwe berekeningscyclus start, wordt de inhoud van de schrijfbuffer (dit is de meest recente sensorinformatie) naar de leesbuffer gekopieerd. Gedurende het tijdsinterval van de berekeningscyclus, blijft de leesbuffer onveranderd. Op deze manier is gegarandeerd dat alle agents hun berekening doen met dezelfde sensorinformatie, namelijk de sensorinformatie die zich gedurende de cyclus in de leesbuffer bevindt.

- De virtuele inputs bevatten enkel een leesbuffer. Deze buffer komt niet overeen met een fysisch aanwezige sensor. De virtuele inputs bevatten geen informatie die rechtstreeks afkomstig is van een sensor, maar informatie die volgt uit een aantal berekeningen. Voor deze berekeningen maken ze gebruik van informatie afkomstig van één of meerdere reële inputs.

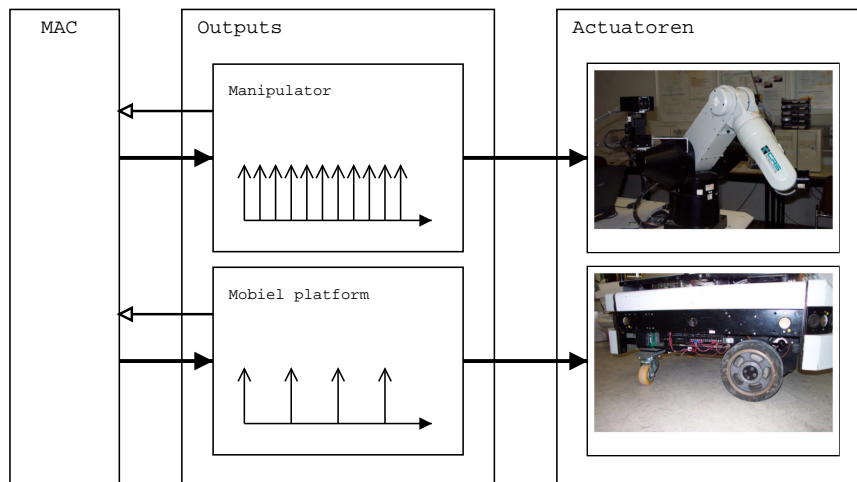
Voorbeeld: een kracht, uitgeoefend op de manipulator, doet een moment ontstaan in de assen van de manipulator. Een virtuele input zou dan aan de hand van informatie, afkomstig van de krachtsensor en de encoders van de manipulator, deze momenten kunnen berekenen. Deze virtuele input krijgt bij wijze van spreken informatie van virtuele

koppelsensoren in de assen van de manipulator. Een ander voorbeeld van een virtuele input is een 'kinematica input'. Deze berekent uit de encoderwaarden van de manipulator informatie betreffende de kinematica van de manipulator: Jacobiaan, rotatiematrices,...

Bij het begin van een nieuwe berekeningscyclus kopiëren de reële inputs de inhoud van hun schrijfbuffer in hun leesbuffer. De virtuele inputs berekenen met informatie uit deze leesbuffers hun eigen virtuele sensorinformatie. Het resultaat van deze berekening stellen ze ter beschikking in hun eigen leesbuffer. Naar buiten toe is er dan ook geen verschil tussen een reële en een virtuele input: beide stellen informatie ter beschikking in hun leesbuffer.

3.4.3 Outputs en actuatoren

Figuur 3.5 geeft een gedetailleerd beeld van de outputs en actuatoren. LiAS heeft twee groepen van actuatoren: de actuatoren van het mobiele platform en de actuatoren van de manipulator. Met elk van deze twee groepen actuatoren komt één output overeen.



Figuur 3.5: De werking van outputs en actuatoren

Actuatoren

Alle actuatoren zijn snelheidsgestuurd: ze blijven met een constante snelheid bewegen, tot hun overeenkomstige output een nieuwe bewegingssnelheid oplegt. De actuatoren van het mobiele platform worden ingesteld met een translatie- en een rotatiesnelheid. De actuatoren van de manipulator worden ingesteld met zes rotatiesnelheden, één voor elke as.

Outputs

De outputs zijn de drijvende kracht achter de architectuur. Elke output moet op bepaalde tijdstippen zijn actuatoren instellen en heeft dan nieuwe stuurdata nodig. Hij stuurt hiervoor een aanvraag naar de MAC. De MAC stuurt deze aanvraag door naar de inputs, die hierop hun meest recente sensorinformatie beschikbaar stellen van alle agents. Daarna start de MAC een berekeningscyclus van de agents op. In de berekeningscyclus worden enkel de agents die de actuatoren van de overeenkomstige output aansturen, actief gemaakt. Als bijvoorbeeld de output van de manipulator een aanvraag stuurt aan de MAC, worden enkel de agents die de manipulator aansturen, actief gemaakt. Het resultaat van de berekeningscyclus, de aanstuurdata voor een groep actuatoren, stuurt de MAC naar de output die de aanvraag stuurde.

Een output bepaalt zelf op welke frequentie hij aanvragen naar de MAC stuurt. De frequentie waarop dit gebeurt, kan voor elke output verschillend zijn. Zo werkt de output van het mobiele platform op $2Hz$ en de output van de manipulator op $6Hz$. Een output is dus een autonoom object, dat bepaalt op welke frequentie de berekeningscyclus wordt uitgevoerd.

3.5 Implementatie van de programma-architectuur

De programma-architectuur is ontwikkeld in de objectgeoriënteerde programmeertaal C++, binnen de MoRE (Mobile Robot Environment) omgeving [19]. MoRE is een software-omgeving die algemene interfaces naar de robot-hardware voorziet. Deze interfaces moeten voor elke individuele robot geïmplementeerd worden. Voor LiAS waren deze implementaties reeds voorhanden.

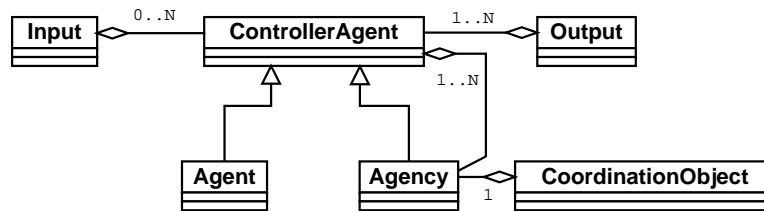
Als ondersteuning voor de architectuur is gebruik gemaakt van de volgende bibliotheken:

- ACE (Adaptive Communication Environment). Met behulp van deze bibliotheek is de architectuur onafhankelijk van het gebruikte besturingssysteem. Meer informatie hierover is beschikbaar op [3].
- NEWMATH [8] biedt de mogelijkheid om complexere wiskundige bewerkingen (matrixbewerkingen,...) uit te voeren.
- ROBOOP (Robotics Object Oriented Package) [10] biedt functies aan voor de berekening van de kinematica van de manipulator (Jacobi-aan, rotatiematrixes,...). Hiervoor maakt de ROBOOP bibliotheek gebruik van de mogelijkheden van de NEWMATH bibliotheek.

De volgende paragraaf bespreekt de implementatie en de hiërarchische structuur van de klassen binnen de programma-architectuur. Ook komen de belangrijkste methodes van elke klasse aan bod.

3.5.1 ContollerAgent

Naar buiten toe is er geen verschil tussen een agent en een agency. Beiden verwerken een ingang tot een uitgang en beiden geven een activatie-niveau weer. Deze gemeenschappelijke eigenschappen zitten in *ControllerAgent* vervat. Zowel *Agent* als *Agency* is een *ControllerAgent*, met elk nog extra specifieke functies. Figuur 3.6 toont de hiërarchische opbouw van de *Agent* en *Agency* klassen.



Figuur 3.6: De structuur van de programma-architectuur

Agent

De belangrijkste methodes van *Agent* zijn:

- *activation()* geeft het activatie-niveau van de agent weer. De agent bepaalt zelf hoe hoog dit activatie-niveau ligt.
- *acknowledge(bool)* legt aan de agent op of hij actief mag worden of niet. Dit wordt beslist door het coördinatie-object, dat hiervoor rekening houdt met het activatie-niveau van de agent.
- *execute()* zal, indien de agent actief mag zijn, de methode *calculate()* oproepen.
- *calculate()* geeft opdracht aan de agent zijn specifieke opdracht of gedraging uit te voeren.

CoordinationObject

De belangrijkste methodes van *CoordinationObject* zijn:

- *resolute()* berekent aan de hand van de activatie-niveaus van de agents, het activatie-niveau van de agency.

- *decide(bool)* legt aan de agents binnen de agency op of ze actief mogen zijn of niet.
- *combine()* combineert de uitgangen van de agents tot de uitgang van de agency.

Agency

De belangrijkste methodes van *Agency* zijn:

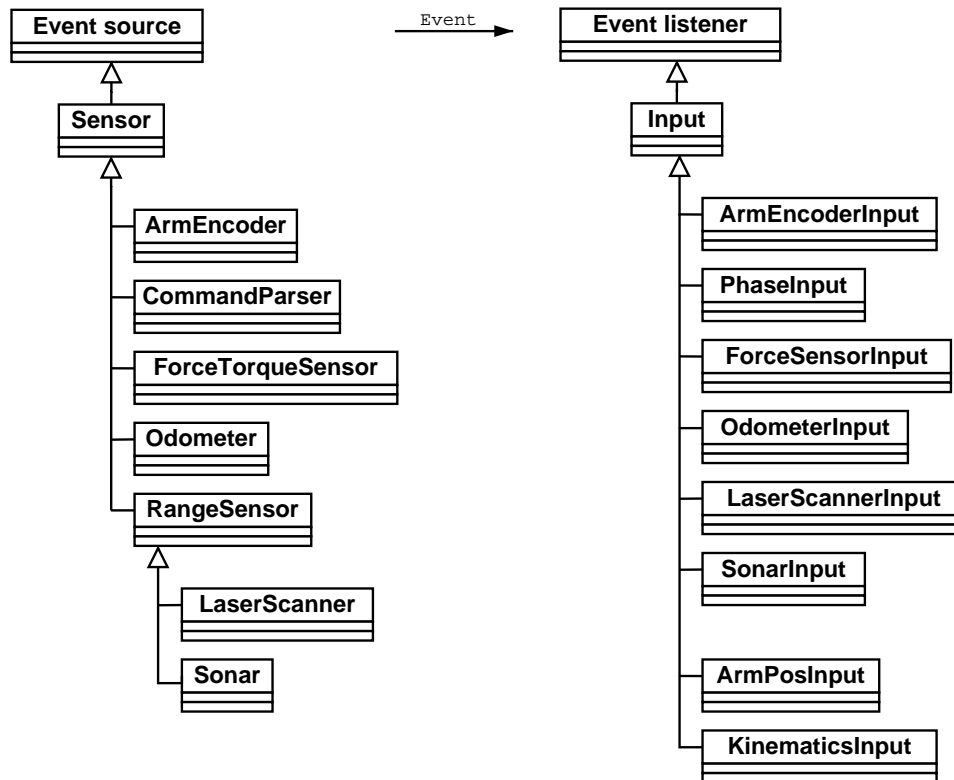
- *activation()* geeft het activatie-niveau van de agency weer. De berekening hiervan wordt uitgevoerd door de *resolute()* methode van het coördinatie-object.
- *acknowledge(bool)* roept de *decide(bool)* methode van het coördinatie-object op.
- *execute()* overloopt alle agents van de agency, en roept telkens *execute()* op. Vervolgens wordt de *combine()* methode van het coördinatie-object opgeroepen.

3.5.2 Sensoren en inputs

Elke sensor werkt op een eigen frequentie, onafhankelijk van de andere sensoren. Telkens als een sensor nieuwe informatie ter beschikking heeft, moet deze naar de schrijfbuffer van de input gekopieerd worden. Tussen de sensor en zijn input is er echter geen directe vorm van communicatie. De communicatie verloopt via een event. De sensor genereert, op het moment dat hij nieuwe informatie heeft, een event. Een sensor is dus een event-generator. Aan elke event-generator kunnen één of meerdere event-ontvangers gekoppeld worden. In het geval van de sensor, is de event-ontvanger de overeenkomstige input van deze sensor. Een event, gegenereerd door een sensor, wordt dus ontvangen door zijn input. Als een input een event ontvangt, kopieert hij de beschikbare sensor-informatie naar zijn schrijfbuffer.

Figuur 3.7 toont de hiërarchische opbouw van de *Sensor*- en *Input*klassen. De klassen *EventSource* en *EventListener* bieden de mogelijkheid aan om met events om te gaan. De belangrijkste methodes van *Input* zijn:

- *operator []* geeft de mogelijkheid de inhoud van de leesbuffer uit te lezen. Deze methode wordt voornamelijk door de agents gebruikt, om de meest recente sensor-informatie op te vragen.
- *getNewValues()* geeft aan de input de opdracht de inhoud van zijn schrijfbuffer te kopiëren in zijn leesbuffer.

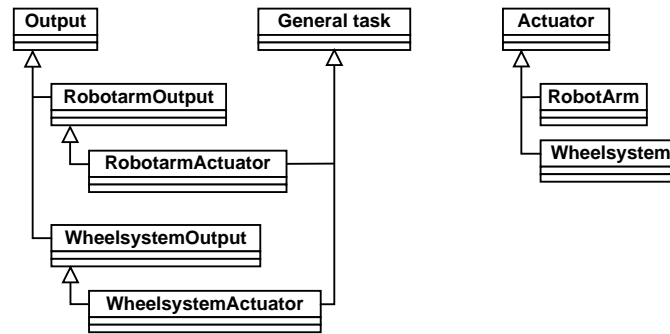


Figuur 3.7: De hiërarchische structuur van sensoren en inputs

- *dataNr()* geeft een tellerstand, die verhoogd wordt telkens als er nieuwe data in de leesbuffer komt. Dit biedt de gebruiker van de input de mogelijkheid om na te gaan of hij reeds de meest recente informatie uit de leesbuffer heeft uitgelezen: telkens de gebruiker van de input informatie uitleest, zal hij ook de tellerstand bijhouden. Zolang deze tellerstand niet verhoogd is, is het ook niet nuttig om de leesbuffer nogmaals uit te lezen.
- *onEvent()* wordt uitgevoerd telkens als de sensor een event genereert. Dit is elke keer dat de sensor nieuwe sensor-informatie ter beschikking heeft. De input kopieert hierop de sensor-informatie in zijn schrijfbuffer.

3.5.3 Outputs en actuatoren

Figuur 3.8 toont de hiërarchische opbouw van de *Output*- en *Actuator*klassen. De klasse *GeneralTask* biedt de mogelijkheid aan om actieve objecten te maken. Dit wordt gebruikt om *RobotarmActuator* en *WheelsystemActuator* zelf te laten bepalen wanneer ze nieuwe stuurdata voor de actuatoren wensen. *Output* is een buffer, waar elke agent de resultaten van zijn berekening tij-



Figuur 3.8: De hiërarchische structuur van outputs en actuatoren

delijk in kan bewaren. *RobotarmOutput* bevat data voor de actuatoren van de manipulator en *WheelsystemOutput* bevat data voor de actuatoren van het mobiele platform.

De belangrijkste methodes van *Output* zijn:

- *operator []* geeft de mogelijkheid data in de outputbuffer weg te schrijven en uit te lezen.
- *isTime()* geeft aan of de actuator van deze output de berekeningscyclus startte. Aangezien er verschillende outputs een berekeningscyclus kunnen starten, is het voor de agents belangrijk te weten of hun output (de output waarvoor zij berekeningen uitvoeren) of een andere output de cyclus startte. Een agent zal enkel zijn berekening uitvoeren als zijn output de cyclus startte.

De actieve outputs, die een berekeningscyclus van de MAC op gang zetten, zijn *RobotarmActuator* en *WheelsystemActuator*. Ze hebben dezelfde functionaliteit als een *Output*, uitgebreid met:

- *sendData()* stelt de overeenkomstige actuator in, met de meest recente aanstuurdata voor deze actuator.

Deze actieve outputs ontvangen hun stuurdata niet van een agent (zoals bij een gewone output), maar rechtstreeks van de MAC.

3.6 Besluit

Voor de gedragsgebaseerde sturing van een mobiele manipulator is er nood aan een programma-architectuur, die kan omgaan met meerdere sensoren en actuatoren, elk met eigen specificaties en werkingsfrequenties. Ook moet de programma-architectuur een eenvoudige, flexibele manier aanbieden voor het implementeren en combineren van verschillende gedragingen. In dit eindwerk werd geopteerd voor een programma-architectuur op basis van agents

[1]. Het concept agents werd in een tweede paragraaf besproken.

De sterke kanten van de gekozen programma-architectuur worden hier nog eens opgesomd:

- Gedragingen kunnen eenvoudig geïmplementeerd worden in agents.
- Het combineren van basisgedragingen tot complexe gedragingen gebeurt door agents en agency's te combineren tot nieuwe agency's.
- Het is heel eenvoudig een boomstructuur van agents en agency's op te stellen of uit te breiden.
- De sensoren kunnen onafhankelijk van elkaar op verschillende frequenties werken.
- Tijdens een berekeningscyclus beschikken alle agents over dezelfde sensor-informatie.
- De mobiele basis en de manipulator kunnen op een verschillende frequentie werken.

Hoofdstuk 4

Gedragsgebaseerde mobiele manipulatie

De concrete opdracht voor dit eindwerk is het uitvoeren van een typische service-taak: het openen van een willekeurige deur, met LiAS, gebruik makende van de gedragsgebaseerde theorie. Eerst leidt de mens LiAS naar de deurklink, met behulp van krachtvolgving. Vervolgens begint LiAS met het openen van de deur. Tenslotte, als de deur volledig geopend is, navigeert LiAS door de deuropening.

De eerste paragraaf beschrijft de opsplitsing van de mobiele manipulator in twee delen: de manipulator en het mobiele platform. Elk van deze delen zal afzonderlijk aangestuurd worden. De tweede paragraaf beschrijft voor elk van de drie fases (navigeren naar de deur, openen van de deur, navigeren door de deuropening) de opsplitsing van de taak in basistaken of basisgedragingen. De functie van alle basisgedragingen wordt op een kwalitatieve manier besproken, zonder dieper in te gaan op de exacte werking ervan. Ook het combineren van de basisgedragingen tot één complexe gedraging wordt besproken. De derde en vierde paragraaf tenslotte, bespreken de exacte werking en implementatie van de basisgedragingen en combinatie-methodes.

4.1 De mobiele manipulator opsplitsen

De mobiele manipulator LiAS heeft acht aanstuurbare vrijheidsgraden. LiAS is dus een redundante robot met een aantal extra vrijheidsgraden. Er zijn verschillende methodes om met deze extra vrijheidsgraden om te gaan. Eén van deze methodes, is het opdelen van de robot in aparte niet-redundante entiteiten. De indeling in meerdere entiteiten kan vooraf vastgelegd zijn of door het controle-programma tijdens de uitvoering aangepast worden. Binnen dit eindwerk is gekozen voor een vaste opdeling in twee functionele entiteiten: het mobiele platform en de manipulator. Deze methode werd bij

vroeger onderzoek op LiAS [9] reeds met succes toegepast.

Uitgaande van de opsplitsing van LiAS in de twee bovengenoemde niet-redundante entiteiten, kunnen de gedragingen voor de mobiele manipulatie-taak in twee groepen onderverdeeld worden. De eerste groep gedragingen stuurt de actuatoren van de manipulator aan, de tweede groep die van het mobiele platform. Tussen de verschillende gedragingen is er geen expliciete communicatie. Het afwezig zijn van communicatie past volledig binnen de gedragsgebaseerde theorie. Enkel via de omgeving is er interactie tussen de manipulator en het mobiele platform. Aan de hand van veranderingen in de omgeving kan een gedraging waarnemen wat het effect is van wat de andere gedragingen doen.

Bijvoorbeeld: de manipulator is vast met de omgeving verbonden. Bij een beweging van het mobiele platform, ontstaat een kracht tussen de manipulator en de omgeving. Door deze kracht waar te nemen (met behulp van een krachtsensor) krijgt de manipulator informatie over de bewegingen van het mobiele platform.

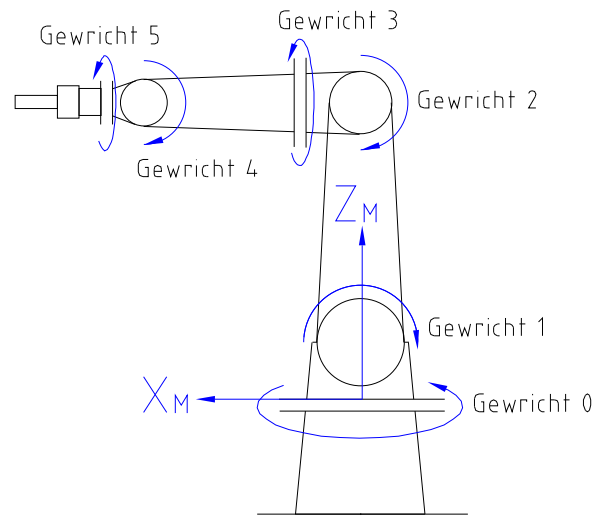
De interactie tussen verschillende gedragingen gebeurt dus indirect, door veranderingen in de omgeving waar te nemen met sensoren. Om een robotsturing op basis van de gedragsgebaseerde theorie te realiseren, zijn er dus steeds voldoende sensoren nodig die veranderingen in de omgeving waarnemen¹.

Opmerking i.v.m. de manipulator: De manipulator is een niet-redundante entiteit, met zes gewrichten en zes vrijheidsgraden (zie figuur 4.1). De eerste drie gewrichten voeren de grote bewegingen uit. Naar dit gedeelte wordt later verwezen als het positionerings-gedeelte. De laatste drie gewrichten voeren de fijnere bewegingen uit. Naar dit gedeelte wordt later verwezen als het oriëntatie-gedeelte.

De eindeffector kan elke mogelijke positie en oriëntatie in de ruimte aannemen, vanzelfsprekend beperkt door het bereik van de manipulator.

Opmerking i.v.m. het mobiele platform: Bij het aansturen van het mobiele platform wordt steeds een snelheid opgelegd in de oorsprong van het manipulator-assenstelsel. Uit deze snelheid worden vervolgens snelheden voor de afzonderlijke wielen berekend. Met een snelheid van het mobiele platform wordt dus steeds de snelheid van de voorkant van het platform bedoeld.

¹In dit eindwerk wordt LiAS door de mens naar de deurklink geleid. Mits het gebruik van extra sensoren (bijvoorbeeld camera's) kan ook deze taak volledig autonoom worden uitgevoerd.



Figuur 4.1: De gewrichten van de manipulator

4.2 Het complexe gedrag opsplitsen in basisgedragingen

Centraal in deze paragraaf staat het opsplitsen van het complexe gedrag in verschillende eenvoudige basisgedragingen, die dan gecombineerd worden. Veel van deze basisgedragingen zijn onafhankelijk van de specifieke taak en kunnen ook gebruikt worden bij het oplossen van andere problemen. De complexe gedraging echter, is bepaald door zowel de basisgedragingen als de manier waarop ze gecombineerd worden. De complexe gedraging voert de specifieke taak uit.

Een eerste opsplitsing van de manipulatie-taak wordt op een hoog niveau gemaakt. Zoals eerder vermeld wordt er een onderscheid gemaakt tussen enerzijds gedragingen die de manipulator aansturen en anderzijds gedragingen die het mobiele platform aansturen. Deze opsplitsing volgt uit het redundantie-probleem: de taak wordt opgesplitst in deeltaken voor niet-redundante entiteiten. De eerste entiteit, de manipulator, is in staat snelle, fijne bewegingen te maken. De manipulator zal dan ook instaan voor het manipuleren van de deurklink en het openen van de deur. De tweede entiteit, het mobiele platform, beweegt trager en minder nauwkeurig. Het mobiele platform zal dan ook instaan voor grotere bewegingen. Meer concreet, zal het mobiele platform zo bewegen, dat de gewrichten van de manipulator niet in een uiterste stand geraken.

Deze paragraaf bespreekt voor elk van de drie fases hoe de complexe taak

wordt opgesplitst in basisgedragingen. Ook het combineren van deze basisgedragingen tot een complexe gedraging, die de manipulatie-taak realiseert, komt aan bod. De bespreking gebeurt op een kwalitatieve manier.

4.2.1 Navigeren naar de deur

In de eerste fase van de opdracht, leidt de mens de grijper naar de deurklink. Dit gebeurt met behulp van krachtvolging. De robot moet zich op een intuïtieve manier laten leiden.

Als de mens een kracht op de grijper uitoefent, beweegt deze in de richting van de kracht. Oefent de mens een moment uit op de grijper, roteert deze volgens het moment. De beweging kan gerealiseerd worden door het mobiele platform of door de manipulator. Om een zo vlot mogelijke beweging toe te laten, worden de zes vrijheidsgraden verdeeld tussen het mobiele platform en de manipulator. Het mobiele platform, dat enkel in een horizontaal vlak kan bewegen, staat in voor bewegingen in dit vlak. De manipulator staat in voor verticale bewegingen en oriëntatie-veranderingen.

Bewegen van de manipulator: *MoveArm*

Oefent de mens een verticale kracht uit op de grijper, dan beweegt de manipulator in de richting van deze kracht, met een snelheid evenredig met de grootte van de kracht. Krachten die in een horizontaal vlak liggen, worden niet door de manipulator gevolgd. Oefent de mens een moment uit op de grijper, dan beweegt het oriëntatie-gedeelte in de richting van dit moment, met een rotatiesnelheid evenredig met de grootte van het moment.

Bewegen van het mobiele platform: *MoveBase*

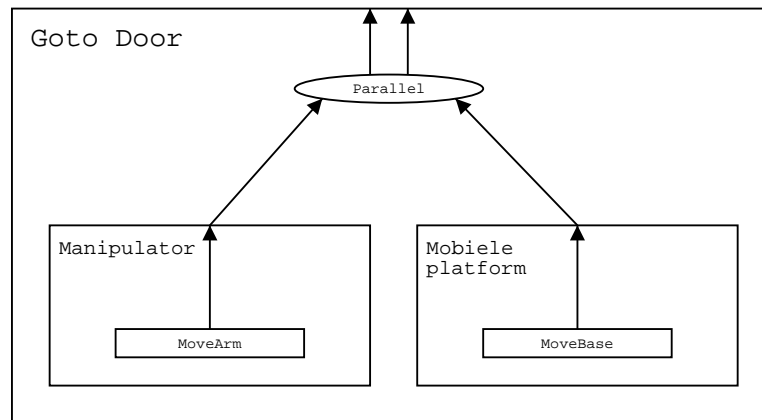
Oefent de mens een horizontale kracht uit op de grijper, dan beweegt het mobiele platform in de richting van deze kracht, met een snelheid evenredig met de grootte van de kracht.

Combineren van de gedragingen

De twee besproken gedragingen sturen elk een ander deel van de mobiele manipulator aan. Ze worden samengevoegd door een parallelle combinatie. Dit betekent dat de twee gedragingen naast elkaar actief zijn, zonder echt gecombineerd te worden. Figuur 4.2 geeft hiervan een overzicht.

4.2.2 Openen van de deur

In de tweede fase van de opdracht opent LiAS volledig autonoom de deur. De manipulator staat in voor het manipuleren van de deurklink en het bewegen



Figuur 4.2: Het navigeren naar een deur

van de deur. In principe zou de manipulator de hele taak op zich kunnen nemen. Het bereik van de manipulator is echter zeer beperkt. Afhankelijk van de grootte van de deur, kan de manipulator in een uiterste stand terecht komen. Om dit te vermijden en dus ook grotere deuren te kunnen openen, wordt het mobiele platform ingezet. De beweging van het mobiele platform moet vermijden dat de manipulator in een uiterste stand geraakt.

Krachtvolgving: *FollowForce*

Tijdens het openen van de deur is de manipulator verbonden met de omgeving: de grijper omklemt de deurklink. Door deze verbinding, is de eind-effector verplicht de bewegingen van de deurklink en de deur te volgen. Het volgen van de bewegingen van de deurklink wordt gerealiseerd door de krachtvolgings-gedraging.

Draaien van de deurklink: *TurnHandle*

Een tweede gedraging heeft als taak de deurklink te draaien, in samenwerking met de krachtvolgings-gedraging. De gedraging heeft hiervoor geen enkele informatie over de deurklink ter beschikking. Het is dus onbekend waar de grijper de klink omvat, of de klink links- of rechtsdraaiend is, of de klink stroef of soepel beweegt,... De gedraging moet toch steeds in staat zijn de klink te draaien.

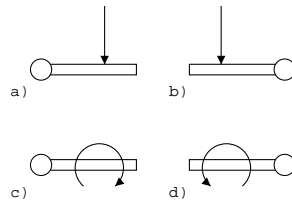
De meest voor de hand liggende methode hiervoor is de grijper een neerwaartse beweging te laten maken. Tijdens de neerwaartse beweging van de deurklink, begint deze te draaien. Dit veroorzaakt een moment op de grijper. Dit moment wordt door de krachtvolgings-gedraging gecompenseerd, door de grijper de draaibeweging van de deurklink te laten volgen. Met

deze methode kunnen zowel links- als rechtsdraaiende deurklinken gedraaid worden.

Experimenteel bleek echter dat om de neerwaardse kracht te realiseren, in de gewrichten van het oriëntatie-gedeelte koppels tot $7Nm$ noodzakelijk zijn. De actuatoren van de manipulator zijn hiervoor onvoldoende krachtig. Deze werkwijze is hierdoor praktisch niet realiseerbaar met LiAS.

Als oplossing werd geopteerd een draaibeweging uit te voeren voor het draaien van de deurklink, in plaats van een neerwaartse beweging. Door deze draaibeweging begint de deurklink neerwaarts te bewegen. Dit veroorzaakt een kracht op de grijper. Deze kracht wordt door de krachtvolgingsgedraging gecompenseerd, door de grijper de neerwaartse beweging van de deurklink te laten volgen. Bij deze methode blijven de momenten in de gewrichten steeds beperkt. Het nadeel van deze aanpak is, dat er voor links- en voor rechtsdraaiende deurklinken een draaibeweging in de andere richting nodig is. Deze methode is dus minder algemeen toepasbaar².

Figuur 4.3 geeft een overzicht van de verschillende methodes. a) en b) tonen de methode met een neerwaartse kracht. c) en d) tonen de methode met een aangelegd moment.



Figuur 4.3: Verschillende methodes voor het draaien van een deurklink

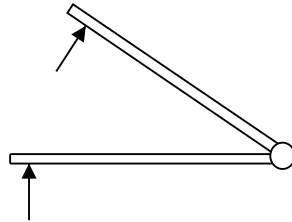
Draaien van de deur: *PullDoor*

Een derde gedraging staat in voor het draaien van de deur, in samenwerking met de krachtvolgingsgedraging. De gedraging heeft hiervoor geen enkele informatie over de deur ter beschikking. Het is dus onbekend hoe breed de deur is en of het een zware of lichte deur is. De gedraging moet toch steeds in staat zijn deur te openen.

Een eenvoudige methode om dit te realiseren, is de grijper een horizontale beweging te laten uitvoeren, loodrecht op de deur (zie figuur 4.4). Door het trekken aan de deur, begint deze te draaien. Dit veroorzaakt een moment

²Merk op dat een mens een rechtsdraaiende deurklink steeds met zijn rechterhand zal draaien en een linksdraaiende deurklink steeds met zijn linkerhand

op de grijper. Dit moment wordt door de krachtvolgings-gedraging gecompenseerd, door de grijper de draaibeweging van de deur te laten volgen.



Figuur 4.4: De beweging van de grijper staat steeds loodrecht op de deur

Beperken van de manipulator-lengte: *LimitArmLength*

Het positionerings-gedeelte van de manipulator, dit zijn de eerste drie gewrichten (zie figuur 4.1), mag niet in een uiterste stand geraken.

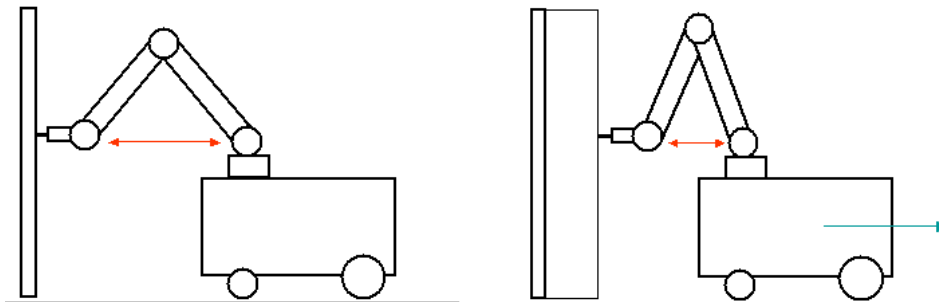
Dit probleem kan aangepakt worden door, voor elk van de drie gewrichten afzonderlijk na te gaan of ze hun uiterste stand naderen. Vervolgens kan het mobiele platform zó bewegen, dat het probleemgewricht zich van zijn uiterste stand verwijdt. Het bepalen van de juiste beweging voor het mobiele platform, afhankelijk van de stand van de manipulator, is echter zeer complex.

Een interessantere en eenvoudiger methode maakt handig gebruik van de beperking van de manipulator: deze is vast verbonden met de deur, zodat slechts een beperkt aantal configuraties mogelijk zijn. Zoals figuur 4.5 illustreert, wordt het vermijden van een uiterste stand herleid tot het beperken van de 'manipulator-lengte'. De 'manipulator-lengte' is de afstand tussen de basis van de manipulator en de eeffector. Als deze afstand te klein wordt, beweegt het mobiele platform achteruit, in het verlengde van de manipulator. Wordt deze afstand te groot, dan beweegt het mobiele platform vooruit, in het verlengde van de manipulator. Het verband tussen de positie van de manipulator en de beweging van het mobiele platform, is dus zeer eenvoudig.

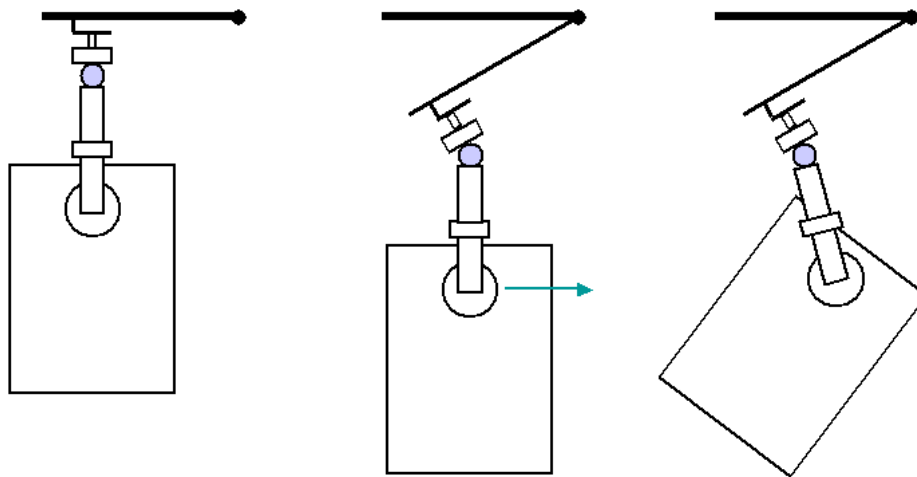
Beperken van de polshoek: *LimitWristAngle*

Ook het oriëntatie-gedeelte van de manipulator, dit zijn de laatste drie gewrichten (zie figuur 4.1), mag niet in een uiterste stand geraken.

Ook hier is het interessant en eenvoudiger om gebruik te maken van de beperking van de manipulator: het oriëntatie-gedeelte staat steeds in een



Figuur 4.5: De lengte van de manipulator moet beperkt blijven



Figuur 4.6: De hoek van het polsgewricht moet beperkt blijven

ongeveer horizontale positie, loodrecht op de deur. Zoals figuur 4.6 illustreert, wordt het vermijden van een uiterste stand herleid tot het beperken van de 'polshoek'. De 'polshoek' is de hoek, in een horizontaal vlak, tussen de eeffector en de manipulator. Als deze hoek te groot wordt, en het oriënterings-gedeelte dus een uiterste stand nadert, beweegt het mobiele platform opzij, loodrecht op het verlengde van de manipulator.

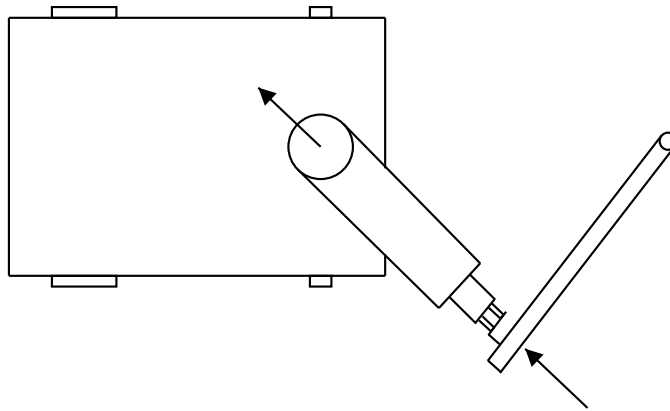
Het vermijden van een uiterste stand van het positionerings-gedeelte gebeurt dus door bewegingen van het mobiele platform, in het verlengde van de manipulator. Een uiterste positie van het oriëntatie-gedeelte wordt vermeden door een beweging van het mobiele platform, loodrecht op het verlengde van de manipulator.

Voorkomen van een uiterste stand door de manipulator: *GotoOptimalPos*

Het mobiele platform zorgt ervoor dat de manipulator niet in een uiterste stand komt. Om dit te realiseren is er echter van uit gegaan dat de gripper vast verbonden is met de omgeving en dus geen enkele vrijheidsgraad meer heeft. Dit is echter niet het geval omdat de gripper verbonden is met de deur, die nog één vrijheidsgraad heeft, namelijk de rotatie van de deur. De gripper kan dus vrij bewegen in de richting loodrecht op de deur. De verschillende gedragingen van het mobiele platform, die moeten voorkomen dat de manipulator in een uiterste stand geraakt, zijn hierdoor in bepaalde gevallen niet effectief.

Bijvoorbeeld: Als de manipulator loodrecht op de deur staat (zie figuur 4.7), kan het mobiele platform de lengte van de manipulator niet meer beperken: bij een beweging het mobiele platform in de richting van de manipulator, ondervindt de manipulator geen krachten omdat de deur deze beweging toelaat.

Als de manipulator evenwijdig met de deur staat, is om dezelfde reden de gedraging die de polshoek van de manipulator beperkt niet meer effectief.



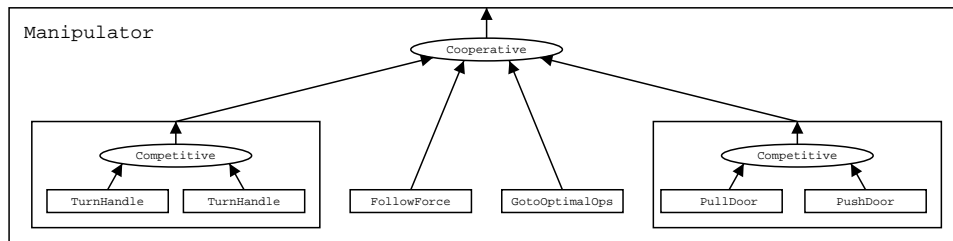
Figuur 4.7: Beweging volgens de vrijheidsgraad van de deur

In deze situaties moet de manipulator er zelf zorgen dat hij niet in een uiterste stand geraakt. Hiervoor wordt er aan de eeffector een snelheid opgelegd, in de richting van de ene vrijheidsgraad. Een snelheid in een andere richting zou via de krachtvolgingsgedraging gecompenseerd worden.

Combineren van de gedragingen

Eerst wordt de combinatie van de gedragingen voor de manipulator besproken, vervolgens die voor het mobiele platform.

Door de combinatie van de basisgedragingen voor de *manipulator*, ontstaat een complex gedrag, dat in principe in staat is een deur te openen. Figuur 4.8 geeft een overzicht van de gedragingen van de manipulator, gecombineerd tot één enkele complexe gedraging.

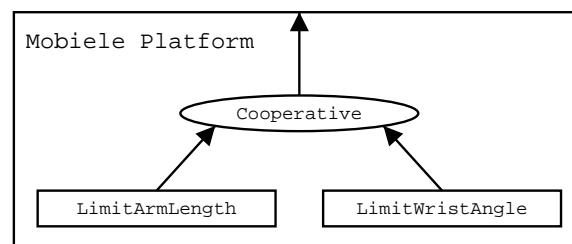


Figuur 4.8: De combinatie van de gedragingen van de manipulator

Twee basisgedragingen realiseren samen het draaien van de deurklink. Eén voor een linksdraaiende deurklink en één voor een rechtsdraaiende deurklink. Door deze twee gedragingen op een competitieve manier te combineren, is het mogelijk zowel link- als rechtsdraaiende deurklinken te draaien. Eerst zal bijvoorbeeld de linksdraaiende agent actief worden. Merkt deze agent dat, zelfs met grote krachten, de deurklink niet draait, wordt deze agent inactief. Hierdoor wordt automatisch de rechtsdraaiende agent actief.³

Ook het openen van de deur wordt gerealiseerd door twee basisgedragingen. Eén om de deur open te duwen en één om de deur open te trekken. Door deze twee gedragingen op een competitieve manier te combineren, is het mogelijk zowel een trek- als een duwdeur te openen.

Het complexe gedrag van de manipulator wordt gerealiseerd door een coöperatieve combinatie van de krachtvolging, het draaien van de deurklink, het openen van de deur en het blokkeren van de laatste vrijheidsgraad. Elk van deze deelgedragingen draagt dus bij tot het complexe gedrag.



Figuur 4.9: De combinatie van de gedragingen van het mobiele platform

Het *mobiële platform* voorkomt dat de gewrichten van de manipulator in

³In dit eindwerk wordt de aard van de deurklink als reeds gekend verondersteld.

een uiterste stand geraken. Het complexe gedrag van het mobiele platform wordt gerealiseerd door een coöperatieve combinatie van de gedraging die de lengte van de manipulator beperkt en de gedraging die de polshoek van de manipulator beperkt (zie figuur 4.9). Elk van de deelgedragingen draagt dus bij tot het complexe gedrag.

De parallelle combinatie van de manipulator-gedragingen en de platform-gedragingen, leidt tot een complex gedrag dat in staat is een willekeurige deur te openen. Figuur 4.10 geeft dit schematisch weer.

4.2.3 Navigeren door de deuropening

In de derde fase van de opdracht navigeert LiAS door de deur deuropening. Eerst wordt met behulp van de laserscanner de deuropening gedetecteerd. Vervolgens begint de navigatie door de deuropening. Dit deel behoort niet tot de eigenlijke opdracht van dit eindwerk, aangezien dit navigatie is en geen mobiele manipulatie, en is dus ook minder volledig uitgewerkt.

De positie waarin LiAS zich bevindt na het openen van een deur is sterk afhankelijk van de beginpositie. De richting waarin LiAS moet rijden om de deur te detecteren verschilt dus sterk van situatie tot situatie. Voor dit eindwerk is het detecteren van de deuropening uitgewerkt voor de meest voorkomende situatie. In deze situatie eindigt het mobiele platform in de richting van de beweging van de deur.

Zoeken van de deuropening: *DetectDoor*

Deze gedraging zoekt een opening die begrenst wordt aan beide zijden. Indien een voldoende grote opening gedetecteerd is, draait het mobiele platform in de richting van deze opening.

Om zeker de juiste deuropening te detecteren (en niet deze van een andere, nabijgelegen deur) kijkt deze gedraging enkel in de richting van geopende deur.

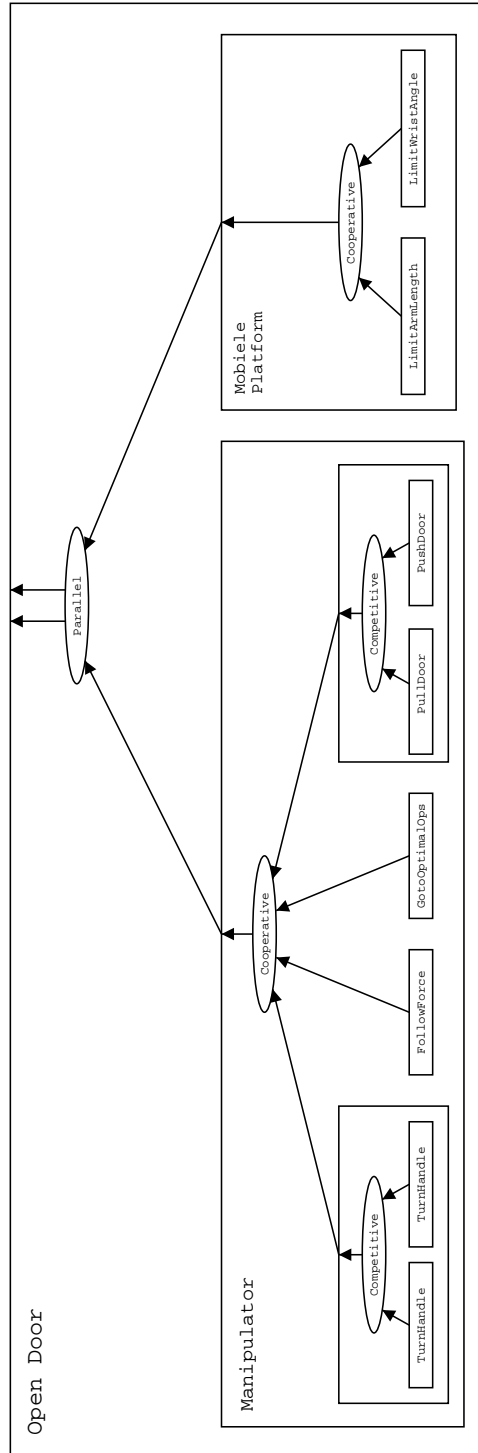
Vermijden van een botsing: *DontTurn*

Deze gedraging voorkomt dat het mobiele platform bij het draaien tegen iets aanbotst. Indien nodig blokkeert deze gedraging een rotatie van het platform.

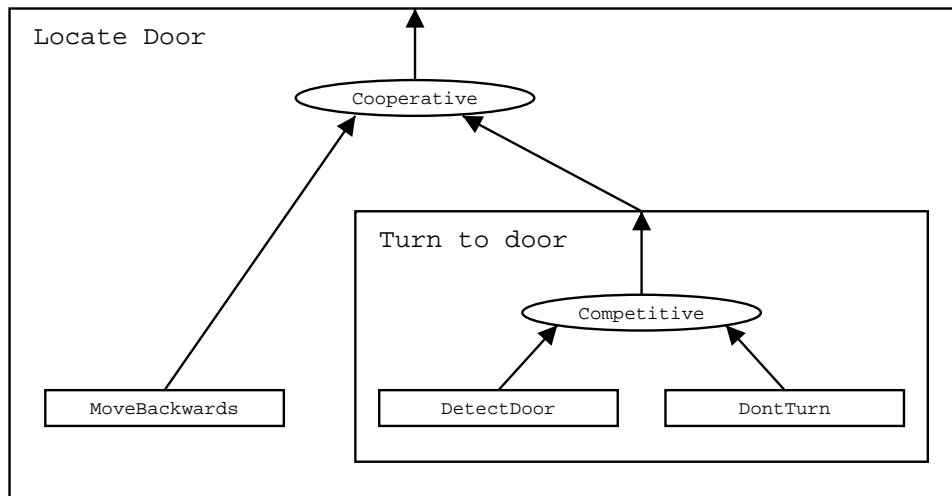
Combineren van de gedragingen

Door de competitieve combinatie van de gedraging die de deuropening zoekt en de gedraging die een botsing vermijdt, kan het platform zich naar de deuropening richten. Figuur 4.11 geeft dit schematisch weer.

HOOFDSTUK 4. GEDRAGSGEBASEERDE MOBIELE MANIPULATIE41



Figuur 4.10: Het openen van een deur



Figuur 4.11: De deuropening localiseren

Ontwijken van obstakels: *AvoidObstacle*

Deze gedraging kijkt naar één zijde van de robot en probeert elk obstakel dat hij ziet te ontwijken. Een object aan de rechterkant van het platform zal steeds langs links ontweken worden en omgekeerd.

Rijden naar de deuropening: *MoveStraight*

Deze gedraging beweegt het mobiele platform in de richting van de deuropening.

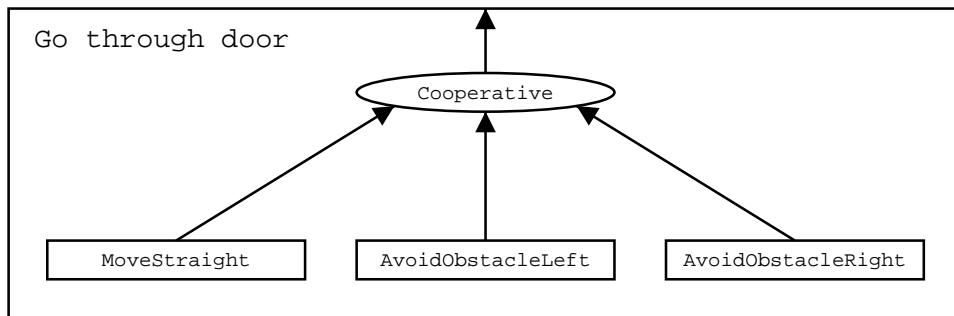
Combineren van de gedragingen

Door de coöperatieve combinatie van een gedraging die obstakels langs links ontwijkt, een gedraging die obstakels langs rechts ontwijkt en de gedraging die naar de deuropening rijdt, navigeert LiAS door de deuropening. Figuur 4.12 geeft dit schematisch weer.

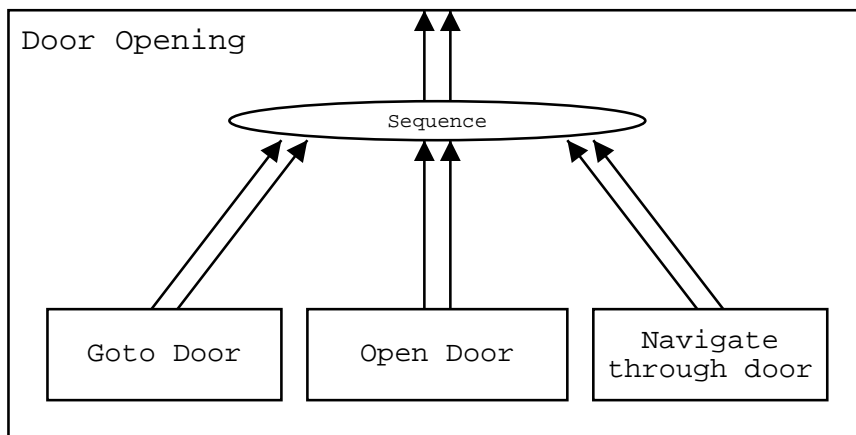
4.2.4 Opeenvolging van fases

De complexe taak bestaat uit een opeenvolging van de navigatie naar de deur, het openen van de deur en het navigeren door de deuropening. Dit is schematisch weergegeven in figuur 4.13.

De combinatie van de verschillende deeltaken gebeurt sequentieel. Dit betekent dat de deeltaken één voor één actief gemaakt worden. De overgang tussen de verschillende fases van de sequentie gebeurt automatisch. Enkel



Figuur 4.12: Navigeren door de deuropening



Figuur 4.13: De opeenvolging van fases

de overgang tussen het navigeren naar de deur en het openen van de deur gebeurt op commando van de mens. Met behulp van bijvoorbeeld de camera's kan ook deze overgang geautomatiseerd worden.

4.2.5 Berekenen van gewrichtssnelheden

Elk van de gedragingen voor de manipulator, heeft als output een snelheid waarmee de eeffector moet bewegen. Deze snelheid is gedefiniëerd in het eeffector-assenstelsel. De manipulator wordt echter niet met een eeffector-snelheid aangestuurd, maar met zes gewrichtssnelheden. Het berekenen van deze gewrichtssnelheden, uit de gewenste eeffector-snelheid, kan op verschillende manieren gebeuren:

Een eerste methode, $KinInverse$, maakt gebruik van de inverse jacobiaan. De gewrichtssnelheden worden rechtstreeks uit de gewenste eeffector-snelheid berekend. Deze methode levert de exacte gewrichtssnelheden om de eeffector met de gewenste snelheid te bewegen.

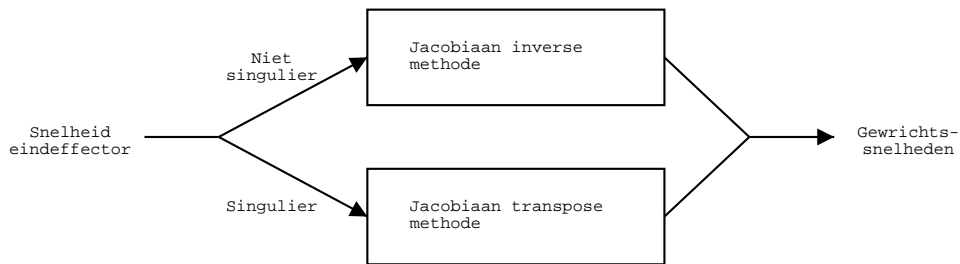
In de buurt van singuliere posities van de manipulator kent deze methode echter problemen. De jacobiaan is in dergelijke posities immers singulier. De inverse van de jacobiaan bevat dus bijna-oneindige waarden. Ook is de invertering rekentechnisch niet stabiel. Daarom zullen de rotatiesnelheden van de gewrichten bij het naderen van een singulariteit naar oneindig gaan.

Een tweede methode, *KinTranspose*, maakt gebruik van de getransponeerde jacobiaan. Deze methode levert geen exacte gewrichtssnelheden om de eindeffector met de gewenste snelheid te bewegen. De eindeffector-snelheid is slechts een benadering van de gewenste snelheid.

Het grote voordeel van deze methode is, dat de berekende gewrichtssnelheden steeds eindig zijn, ook al bevindt de manipulator zich in een singuliere positie.

Beide methodes hebben hun eigen voor- en nadelen. In dit eindwerk worden de positieve elementen van beide methodes gecombineerd: in normale situaties berekent de jacobiaan-inverse methode de gewrichtssnelheden. In de buurt van singuliere posities wordt echter overgeschakeld op de jacobiaan-transpose methode.

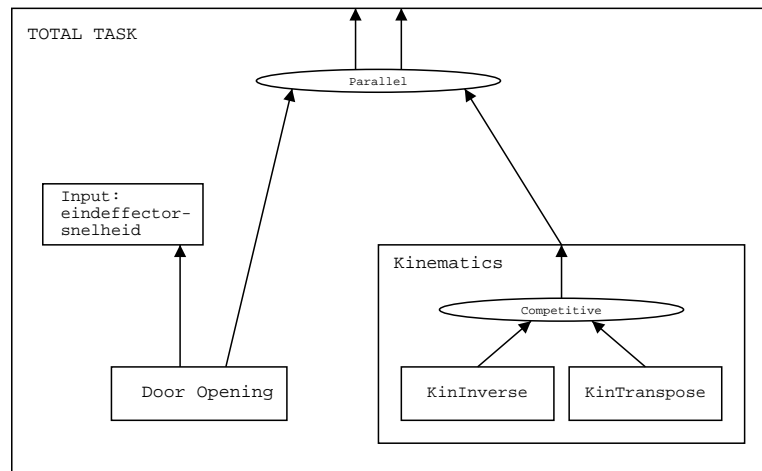
De uitvoering van deze berekeningen gebeurt door twee gedragingen. Een eerste gedraging berekent de gewrichtssnelheden, gebruik makende van de jacobiaan-inverse methode. Een tweede gedraging gebruikt hiervoor de jacobiaan-transpose methode. Figuur 4.14 geeft dit schematisch weer. Deze tweede gedraging wordt actief bij het naderen van een singuliere positie.



Figuur 4.14: De berekening van de gewrichtssnelheden

4.2.6 Gehele taak

Om binnen de gebruikte programma-architectuur eindeffectorsnelheden om te zetten naar gewrichtssnelheden, wordt de gewenste eindeffector-snelheid omgeleid via een input. De twee gedragingen die instaan voor de omrekening, maken van deze input gebruik. Dit is schematisch weergegeven in figuur 4.15.



Figuur 4.15: De gehele mobiele manipulatietaak

4.2.7 Overgang tussen verschillende fases

De eigenlijke opdracht bestaat uit drie fases: het leiden naar de deur, het openen van de deur en het navigeren door de deur. Telkens een fase is afgehandeld, moet er worden overgegaan naar de volgende fase. De overgang naar de tweede fase gebeurt op commando van de mens. De overgang naar de derde fase gebeurt automatisch als de deur voldoende ver geopend is (90°)

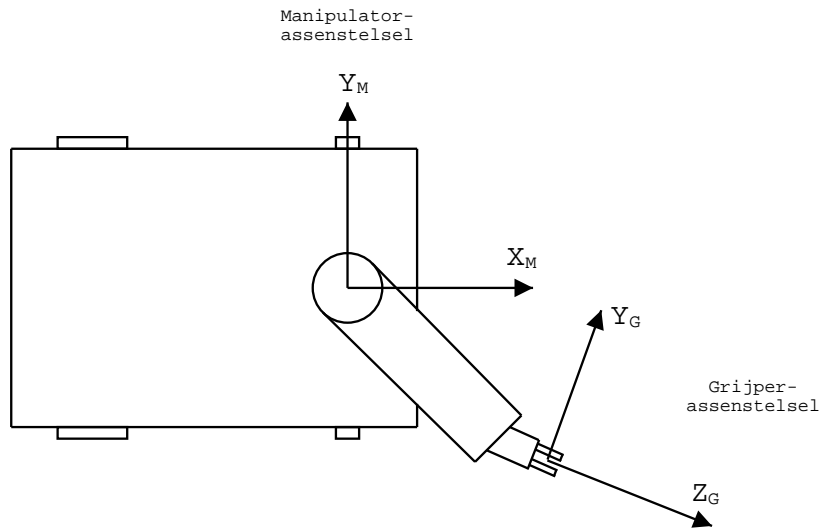
4.3 Werking van de agents

Figuur 4.16 geeft de assenstelsels weer die in deze paragraaf gebruikt worden. Het manipulator-assenstelsel is vast verbonden met het mobiele platform. Het grijper-assenstelsel is vast verbonden met de eideffector. Zoals reeds eerder besproken, wordt elke gedraging door een agent gerealiseerd. Elke agent voert juist één gedraging uit. Elke agent wordt gekenmerkt door een activatie-niveau en een berekening waarin sensorwaardes omgezet worden tot aanstuurdata voor de actuatoren. Deze paragraaf geeft een gedetailleerde beschrijving van de werking en de implementatie van deze agents.

4.3.1 FollowForce

Berekening

De FollowForce agent berekent uit informatie van de krachtsensor en de kinematica van de manipulator, de gewenste snelheid van de eideffector, om zo de kracht en het moment in de grijper nul te maken. De eideffector moet hierbij in de richting van de waargenomen kracht en het waargenomen moment bewegen. Figuur 4.17 toont de eideffector van LiAS. Deze bestaat



Figuur 4.16: Voorstelling van de gedefiniëerde assenstelsels

uit de krachtsensor en de grijper.

De kracht en het moment, waargenomen door de krachtsensor, zijn niet onmiddellijk bruikbaar voor de krachtvolging. Deze bevatten immers nog componenten veroorzaakt door de massa van de grijper. Ook zijn niet de krachten en momenten in de krachtsensor nodig, maar wel de krachten en momenten in het Tool Center Point (TCP). Het TCP is de plaats waar de grijper de deurklink omvat.

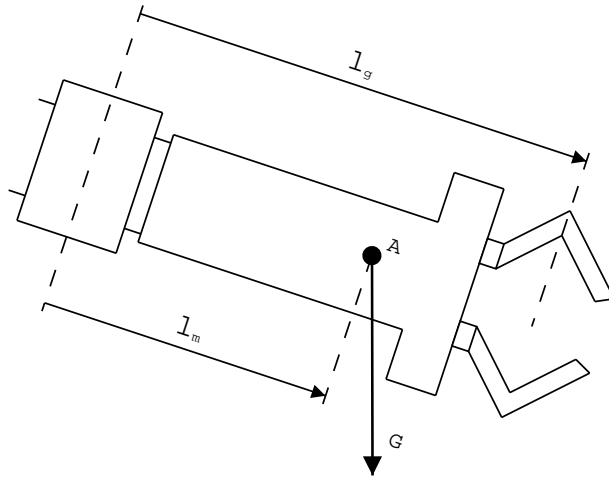
Het neutraliseren van het effect van de massa van de grijper en het transformeren van de krachten en momenten naar het TCP gebeurt door een virtuele input, *ForceInput*.

De massa van de grijper neutraliseren:

De kracht en het moment, waargenomen door de krachtsensor, bevatten componenten, veroorzaakt door de massa van de grijper. Deze componenten zijn afhankelijk van de oriëntatie van de eideffector. De gravitatiekracht is immers steeds naar beneden gericht, terwijl de eideffector elke willekeurige oriëntatie kan aannemen. De kracht, gemeten in het manipulator-assenstelsel en veroorzaakt door de massa van de grijper, is:

$$\mathbf{F}_{\text{massa}}^{\text{M}} = \mathbf{e}_z \cdot m \cdot g \quad (4.1)$$

- $\mathbf{F}_{\text{massa}}^{\text{M}}$ is de gravitatiekracht, gezien in het manipulator-assenstelsel,
- \mathbf{e}_z is de eenheidsvector in de z-richting,
- g is de gravitatieconstante,



Figuur 4.17: De eindeffector van LiAS

- m is de massa van de grijper.

Deze gravitatiekracht, getransformeerd naar het grijper-assenstelsel:

$$\mathbf{F}_{\text{massa}}^{\text{G}} = \mathbf{R}_0^6 \cdot \mathbf{F}_{\text{massa}}^{\text{M}} \quad (4.2)$$

- $\mathbf{F}_{\text{massa}}^{\text{G}}$ is de gravitatiekracht, gezien in het grijper-assenstelsel,
- \mathbf{R}_0^6 is de rotatie-matrix tussen manipulator-assenstelsel en grijper-assenstelsel.

Ter hoogte van de krachtsensor veroorzaakt deze gravitatiekracht een kracht en een moment. Deze worden gegeven door:

$$\begin{pmatrix} \mathbf{F}_{\text{fsenor}}^{\text{G}} \\ \mathbf{M}_{\text{fsenor}}^{\text{G}} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{\text{massa}}^{\text{G}} \\ \mathbf{l}_m \times \mathbf{F}_{\text{massa}}^{\text{G}} \end{pmatrix} \quad (4.3)$$

- $\mathbf{F}_{\text{fsenor}}^{\text{G}}$ is de kracht, gemeten in de krachtsensor, veroorzaakt door de massa van de grijper,
- $\mathbf{M}_{\text{fsenor}}^{\text{G}}$ is het moment, gemeten in de krachtsensor, veroorzaakt door de massa van de grijper,
- \mathbf{l}_m is de afstandsvector tussen de krachtsensor en het massacentrum van de grijper (zie figuur 4.17).

Deze kracht en dit moment worden van de meetwaarde van de krachtsensor afgetrokken om het effect van de massa van de grijper te neutraliseren. De grijper wordt dus niet meer 'gevoeld'.

Berekenen van de kracht en het moment in het TCP:

De kracht en het moment die de krachtsensor meet, zijn verschillend van deze in het TCP. Om de kracht en het moment, uitgeoefend in het TCP te kennen, wordt de gementen waarde van de krachtsensor, herkend naar het TCP. Het resultaat hiervan is:

$$\begin{pmatrix} \mathbf{F}_{\text{tcp}}^{\mathbf{G}} \\ \mathbf{M}_{\text{tcp}}^{\mathbf{G}} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{\text{fsensor}}^{\mathbf{G}} \\ \mathbf{M}_{\text{fsensor}}^{\mathbf{G}} + \mathbf{l}_{\mathbf{g}} \times \mathbf{F}_{\text{fsensor}}^{\mathbf{G}} \end{pmatrix} \quad (4.4)$$

- $\mathbf{F}_{\text{tcp}}^{\mathbf{G}}$ is de kracht, uitgeoefend in het TCP,
- $\mathbf{M}_{\text{tcp}}^{\mathbf{G}}$ is het moment, uitgeoefend in het TCP,
- $\mathbf{l}_{\mathbf{g}}$ is de afstandsvector tussen de krachtsensor en het TCP (zie figuur 4.17).

De kracht en het moment in het TCP, worden met een evenredigheidsfactor omgezet naar een translatie- en een rotatiesnelheid:

$$\begin{aligned} \dot{\mathbf{x}}^{\mathbf{G}} &= \mathbf{K} \cdot \begin{pmatrix} \mathbf{F}_{\text{tcp}}^{\mathbf{G}} \\ \mathbf{M}_{\text{tcp}}^{\mathbf{G}} \end{pmatrix} \\ &= \begin{pmatrix} 130 & & & & & \\ & 130 & & & & \\ & & 130 & & & \\ & & & 0.008 & & \\ & & & & 0.008 & \\ & & & & & 0.003 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{F}_{\text{tcp}}^{\mathbf{G}} \\ \mathbf{M}_{\text{tcp}}^{\mathbf{G}} \end{pmatrix} \end{aligned} \quad (4.5)$$

- $\dot{\mathbf{x}}^{\mathbf{G}}$ is de translatie- en rotatiesnelheid van het TCP,
- \mathbf{K} is een matrix met experimenteel bepaalde evenredigheidsfactoren.

$\dot{\mathbf{x}}^{\mathbf{G}}$ vormt het uiteindelijke resultaat van de FollowForce agent.

Activatie-niveau

De FollowForce agent wordt actief als er een voldoende grote externe kracht op de grijper aangrijpt. Het effect van de massa van de grijper wordt hierbij geneutraliseerd.

De FollowForce agent wordt actief als:

$$a_1 \cdot \|\mathbf{F}_{\text{tcp}}^{\mathbf{G}}\| + a_2 \cdot \|\mathbf{M}_{\text{tcp}}^{\mathbf{G}}\| > 1 \quad (4.6)$$

- a_1 is de scaleringsfactor voor de kracht, 0.3
- a_2 is de scaleringsfactor voor het moment, 0.01

4.3.2 MoveArm en MoveBase

Berekening

De MoveArm en MoveBase agents zijn een variatie op de FollowForce agent. De berekening van gemeten kracht naar een snelheidsvector gebeurt op exact dezelfde wijze.

Analoog aan vergelijking 4.5:

$$\dot{\mathbf{x}}^G = \mathbf{K} \cdot \begin{pmatrix} \mathbf{F}_{\text{tcp}}^G \\ \mathbf{M}_{\text{tcp}}^G \end{pmatrix}. \quad (4.7)$$

Deze snelheden worden opgesplitst in stuursnelheden voor het mobiele platform en stuursnelheden voor de manipulator:

$$\begin{aligned} \dot{\mathbf{x}}_{\text{pl}}^G &= \begin{pmatrix} \dot{\mathbf{x}}^G(1) \\ \dot{\mathbf{x}}^G(2) \end{pmatrix} \\ \dot{\mathbf{x}}_{\text{ee}}^G &= \dot{\mathbf{x}}^G \cdot \begin{pmatrix} 0 & & & & \\ & 0 & & 0 & \\ & & 1 & & \\ & & & 1 & \\ & 0 & & & 1 & \\ & & & & & 1 \end{pmatrix} \end{aligned} \quad (4.8)$$

- $\dot{\mathbf{x}}^G(i)$ is de i^e component van $\dot{\mathbf{x}}^G$,
- $\dot{\mathbf{x}}_{\text{pl}}^G$ is de gewenste snelheid van het mobiele platform,
- $\dot{\mathbf{x}}_{\text{ee}}^G$ is de gewenste snelheid van de eeffector.

Activatie-niveau

De MoveArm en MoveBase agents worden actief als er een voldoende grote externe kracht op de grijper aangrijpt. Het effect van de massa van de grijper wordt hierbij geneutraliseerd.

De MoveArm en MoveBase agents worden actief als:

$$a_1 \cdot \|\mathbf{F}_{\text{tcp}}^G\| + a_2 \cdot \|\mathbf{M}_{\text{tcp}}^G\| > 1 \quad (4.9)$$

- a_1 is de scaleringsfactor voor de kracht, 0.3
- a_2 is de scaleringsfactor voor het moment, 0.01

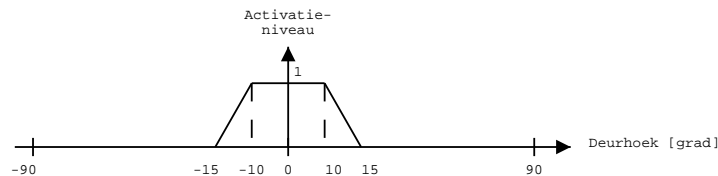
4.3.3 TurnHandle

Berekening

De PullDoor agent stuurt de eindeffector met een constante rotatie rond de z-as van het grijper-assenstelsel.

Activatie-niveau

De TurnHandle agent blijft actief, tot de deur over een kleine hoek gedraaid is. Op dat moment is het slot met zekerheid geopend, en mag de deurklink teruggedraaid worden. TurnHandle is volledig inactief als de deur over een hoek van 15° gedraaid is. Dit is voorgesteld in figuur 4.19.



Figuur 4.18: Het activatie-niveau van TurnHandle

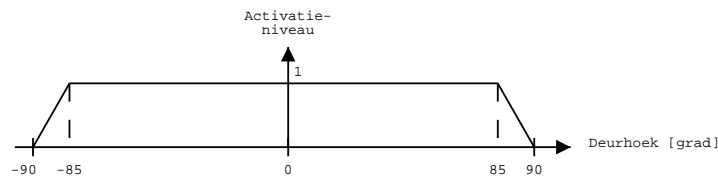
4.3.4 PullDoor

Berekening

De PullDoor agent stuurt de eindeffector met een constante translatie-snelheid in de z-richting van het grijper-assenstelsel.

Activatie-niveau

De PullDoor agent blijft actief, tot de deur volledig geopend is. Als de deur bijna over 90° gedraaid is, neemt het activatie-niveau langzaam af. PullDoor is volledig inactief als de deur over een hoek van 90° gedraaid is. Dit is voorgesteld in figuur 4.19.



Figuur 4.19: Het activatie-niveau van PullDoor

4.3.5 LimitArmlength

Berekening

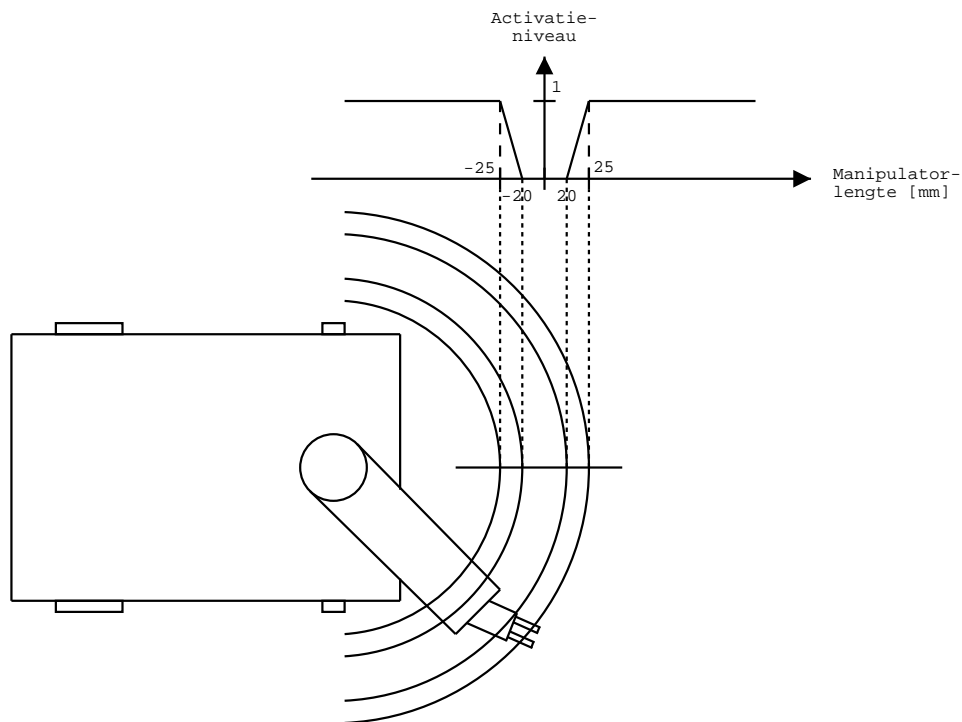
De LimitArmlength agent legt een snelheid op aan het mobiele platform, in het verlengde van de manipulator, in de oorsprong van het manipulator-assenstelsel.

$$\mathbf{v}^{\mathbf{M}} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} v_{lim} \\ 0 \end{pmatrix} \quad (4.10)$$

- $\mathbf{v}^{\mathbf{M}}$ is de gewenste snelheid in het manipulator-assenstelsel,
- θ is de hoek tussen het mobiele platform en de manipulator,
- v_{lim} is de snelheid in het verlengde van de manipulator, $35[mm/s]$

Activatie-niveau

Zoals weergegeven in figuur 4.20, is het bereik van de manipulator onderverdeeld in een aantal gebieden. Het centrale gebied, is een neutrale zone. Hier is de LimitArmlength agent volledig inactief. Daarrond liggen twee overgangsgebieden, waarin het activatie-niveau lineair oploopt. De LimitArmlength agent is volledig actief in de twee uiterste gebieden.



Figuur 4.20: Het activatie-niveau van LimitArmlength

4.3.6 LimitWristAngle

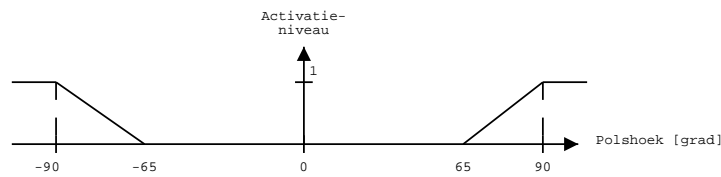
Berekening

De LimitWristAngle agent legt een snelheid op aan het mobiele platform, loodrecht op het verlengde van de manipulator, in de oorsprong van het manipulator-assenstelsel.

$$\mathbf{v}^{\mathbf{M}} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \cdot \begin{pmatrix} 0 \\ v_{wri} \end{pmatrix} \quad (4.11)$$

- $\mathbf{v}^{\mathbf{M}}$ is de gewenste snelheid in het manipulator-assenstelsel,
- θ is de hoek tussen het mobiele platform en de manipulator.
- v_{wri} is de snelheid loodrecht op het verlengde van de manipulator, $30[mm/s]$

Activatie-niveau



Figuur 4.21: Het activatie-niveau van LimitWristAngle

De LimitWristAngle agent wordt lineair actief vanaf het moment dat de polshoek van de manipulator groter is dan 65° . LimitWristAngle is volledig actief als de polshoek over een hoek van 75° gedraaid is. Dit is voorgesteld in figuur 4.20

4.3.7 GotoOptimalPos

Berekening

De GotoOptimalPos agent legt aan de eindeffector een snelheid op, loodrecht op de deur gericht. Deze snelheid moet voorkomen dat de manipulator in een uiterste stand geraakt:

$$\mathbf{v}^{\mathbf{G}} = \begin{pmatrix} 0 \\ -\mathbf{v}_{arm} \cdot \sin \theta \\ \mathbf{v}_{arm} \cdot \cos \theta \\ 0 \\ 0 \\ 0 \end{pmatrix} \cdot \cos \alpha \cdot Sgn + \begin{pmatrix} 0 \\ -\mathbf{v}_{wrist} \cdot \sin \theta \\ \mathbf{v}_{wrist} \cdot \cos \theta \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.12)$$

HOOFDSTUK 4. GEDRAGSGEBASEERDE MOBIELE MANIPULATIE53

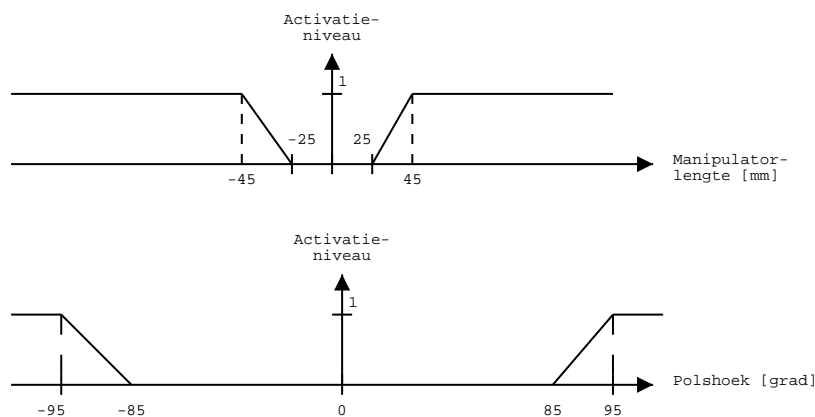
- v_{arm} is de gewenste snelheid van de eeffector in het grijper-assenstelsel, om de lengte van de manipulator beperkt te houden,
- v_{wrist} is de gewenste snelheid van de eeffector in het grijper-assenstelsel, om de hoek van het polsgewricht beperkt te houden,
- θ is de hoek tussen de eeffector en een loodrechte op de deur. Deze hoek is verschillend van nul omdat de grijper de klink met een zekere soepelheid omvat
- α is de hoek van het polsgewricht,
- Sgn is +1 of -1, respectievelijk als de lengte van de manipulator te klein of te groot is.

Activatie-niveau

Analoog aan de LimitArmLength agent wordt de GotoOptimalPos agent actief als de lengte van de manipulator te veel afwijkt van de neutrale lengte. GotoOptimalPos wordt echter pas actief als LimitArmLength reeds volledig actief is. Dit is nodig om te vermijden dat er een situatie kan ontstaan waar de hele mobiele manipulator stilstaat.

Analoog aan de LimitWristAngle agent wordt de GotoOptimalPos agent actief als de polshoek van de manipulator te groot wordt. Ook hier wordt GotoOptimalPos pas actief als LimitWristAngle reeds volledig actief is.

De volledige activatie is het maximum van de twee deel-activaties. Deze zijn voorgesteld in figuur 4.22



Figuur 4.22: Het activatie-niveau van GotoOptimalPos

4.3.8 KinInverse

Berekening

De KinInverse agent berekent de gewrichtssnelheden van de manipulator uit de gewenste eindeffector-snelheid. Hiervoor maakt deze agent gebruik van de inverse jacobiaan:

$$\dot{\mathbf{q}} = \mathbf{J}^{-1} \cdot \dot{\mathbf{x}} \quad (4.13)$$

- $\dot{\mathbf{q}}$ stelt de zes gewrichtssnelheden van de manipulator voor,
- \mathbf{J} is de jacobiaan,
- $\dot{\mathbf{x}}$ is de snelheid van de eindeffector.

Deze methode levert de exacte gewrichtssnelheden om de eindeffector met de gewenste snelheid te bewegen. In de buurt van singuliere posities van de manipulator ontstaan er echter problemen.

Activatie-niveau

De KinInverse agent is steeds volledig actief.

4.3.9 KinTranspose

Berekening

De KinTranspose agent berekent uit een gewenste eindeffector-snelheid de gewrichtssnelheden van de manipulator. De berekening hiervan gebeurt via een virtuele kracht op de eindeffector:

$$\mathbf{F} = \mathbf{K}^{-1} \cdot \dot{\mathbf{x}} = \begin{pmatrix} \frac{1}{130} & & & & & \\ & \frac{1}{130} & & & & \\ & & \frac{1}{130} & & & \\ & & & \frac{1}{0.008} & & \\ & 0 & & & \frac{1}{0.008} & \\ & & & & & \frac{1}{0.003} \end{pmatrix} \cdot \dot{\mathbf{x}} \quad (4.14)$$

- \mathbf{F} is de virtuele kracht op de grijper,
- \mathbf{K} is een matrix met experimenteel bepaalde evenredigheidsfactoren, die het verband geven tussen de eindeffector-snelheid en de virtuele kracht op de eindeffector.⁴

⁴Merk op dat het hier over de zelfde matrix \mathbf{K} gaat als in vergelijking 4.5

Deze virtuele kracht op de eeffector veroorzaakt virtuele momenten in elk van de gewrichten van de manipulator. Deze momenten worden gegeven door:

$$M = \mathbf{J}^T \cdot \mathbf{F} \quad (4.15)$$

- \mathbf{M} stelt de virtuele momenten in de gewrichten voor,

Door aan de gewrichten een snelheid op te leggen, evenredig met de virtuele momenten, beweegt de eeffector ongeveer in de gewenste richting:

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{C} \cdot \mathbf{M} \\ &= \begin{pmatrix} 0.0012 & & & & & \\ & 0.0009 & & & & \\ & & 0.0012 & & & \\ & & & 0.0012 & & \\ & & & & 0.06 & \\ & & & & & 0.03 \end{pmatrix} \cdot \mathbf{M} \end{aligned} \quad (4.16)$$

- \mathbf{C} is een matrix met experimenteel bepaalde evenredigheidsfactoren, die het verband geven tussen de het virtuele moment in en de rotatiesnelheid van de gewrichten.

De resulterende eeffector-snelheid is niet volledig gelijk aan de gewenste snelheid. Toch is deze methode nuttig in de buurt van singuliere posities van de manipulator. De resulterende gewrichtssnelheden zijn immers steeds beperkt.

Activatie-niveau

De KinTranspose agent wordt actief bij het naderen van een singuliere positie. De determinant van de jacobiaan geeft een goede indicatie over singuliere posities. Deze wordt namelijk kleiner bij het naderen van een singuliere positie, om nul te worden in de singuliere positie zelf.

4.4 Werking van de coördinatie-objecten

4.4.1 WeightedSum

Combineren van agents

Het WeightedSum coördinatie-object heeft als output de gewogen som van de outputs van alle agents. Als wegingsfactor wordt het activatie-niveau van de overeenkomstige agent gebruikt.

$$output_j = \frac{\sum_i [output_j(agent_i) \cdot acti(agent_i)]}{\sum_i [acti(agent_i)]} \quad (4.17)$$

- $j = 1 \dots$ aantal outputs,
- $i = 1 \dots$ aantal agents.

Activatie-niveau van de agency

Het WeightedSum coördinatie-object overloopt alle agents en gaat daarbij op zoek naar de agent met het maximale activatie-niveau. Elke agent, met een activatie-niveau groter dan nul, wordt actief gemaakt.

$$acti = \max[acti(agent_1), acti(agent_2), \dots, acti(agent_N)] \quad (4.18)$$

- $N =$ het aantal agents in de agency.

4.4.2 Competitief

Combineren van agents

Het Competitieve coördinatie-object heeft als output de output van zijn ene actieve agent, vermenigvuldigd met zijn activatie-niveau.

$$output_j = output_j(agent_i) \cdot acti(agent_i) \quad (4.19)$$

- $i =$ de ene actieve agent,
- $j = 1 \dots$ aantal outputs van $agent_i$

Activatie-niveau van de agency

Het Competitieve coördinatie-object overloopt alle agents en gaat daarbij op zoek naar de agent met het maximale activatie-niveau. Deze agent wordt als enige actief gemaakt.

$$acti = \max[acti(agent_1), acti(agent_2), \dots, acti(agent_N)] \quad (4.20)$$

- $N =$ het aantal agents in de agency.

4.4.3 Parallel

Combineren van agents

Het Parallel coördinatie-object neemt de output van elke agent over, vermenigvuldigd met zijn activatie-niveau. Het aantal outputs van dit coördinatie-object is dus:

$$aantaloutputs = \sum_i [aantaloutputs(agent_i)] \quad (4.21)$$

$$output_k = output_j(agent_i) \cdot acti(agent_i)$$

- $i = 1 \dots$ aantal agents
- $j = 1 \dots$ aantal outputs($agent_i$)
- $k = 1 \dots$ aantal outputs

Activatie-niveau van de agency

Het Parallel coördinatie-object overloopt alle agents en gaat daarbij op zoek naar de agent met het maximale activatie-niveau. Elke agent, met een activatie-niveau groter dan nul, wordt actief gemaakt.

$$acti = \max[acti(agent_1), acti(agent_2), \dots, acti(agent_N)] \quad (4.22)$$

- $N =$ het aantal agents in de agency.

4.4.4 Sequence

Combineren van agents

Het Sequence coördinatie-object heeft als output de output van zijn ene actieve agent, vermenigvuldigd met zijn activatie-niveau.

$$output_j = output_j(agent_i) \cdot acti(agent_i) \quad (4.23)$$

- $i =$ de ene actieve agent,
- $j = 1 \dots$ aantal outputs van $agent_i$

Activatie-niveau van de agency

Het Sequence coördinatie-object maakt één voor één zijn agents actief. Het activatie-niveau is dat van zijn actieve agent.

$$acti = acti(agent_i) \quad (4.24)$$

- $i =$ de ene actieve agent.

Hoofdstuk 5

Resultaten

Dit hoofdstuk bespreekt de interacties tussen de gedragingen tijdens het openen van een deur en hoe deze interacties zich manifesteren in de activatieniveaus van de gedragingen.

De eerste paragraaf geeft een overzicht van de beweging die de robot uitvoert bij het openen van de deur. De tweede paragraaf behandelt de interacties op het niveau van de afzonderlijke deelgedragingen. De derde paragraaf tenslotte, behandelt het geheel vanuit een regeltechnisch standpunt.

5.1 Overzicht van de beweging

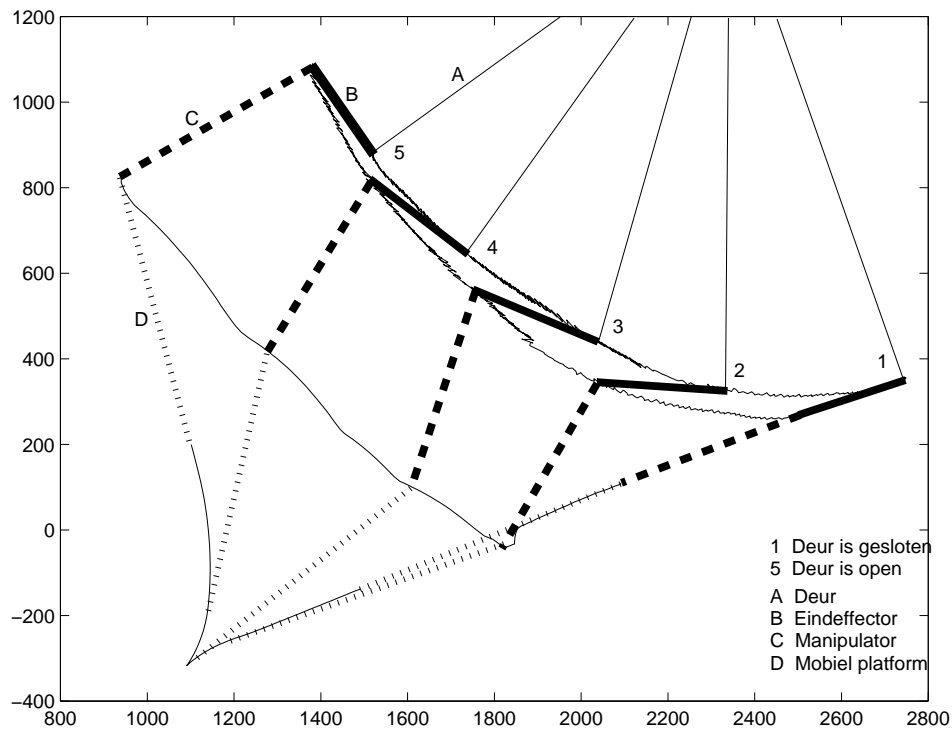
Figuur 5.1 toont het traject dat LiAS volgt tijdens het openen van een deur. Dit traject is opgemeten tijdens de uitvoering van een praktische test. In vijf standen is op een schematische wijze de positie en oriëntatie van LiAS aangegeven.

Deze figuur geeft heel duidelijk weer hoe de polshoek evolueert. Tussen stand 1 en 3 vergroot de polshoek. Vanaf stand 3 stabiliseert de polshoek op 90° .

Figuur 5.2 geeft een gedetailleerdere voorstelling van LiAS in deze standen.

5.2 Benadering vanuit de deelgedragingen

Het openen van de deur resulteert uit het combineren van zes gedragingen: PullDoor, TurnHandle, GotoOptimalPosition, Followforce, LimitArmLength en LimitWristAngle. De eerste vier van deze gedragingen sturen de manipulator aan, de laatste twee het mobiele platform. Samen staan ze in voor het draaien van de deurklink, het trekken aan de deur, het beperken van de manipulatorlengte en het beperken van de polshoek. Figuur 5.3 toont het verloop van de activatieniveau's van deze zes gedragingen tijdens het openen van een deur.



Figuur 5.1: Het traject dat LiAS aflegt

5.2.1 Het draaien van de deurklink

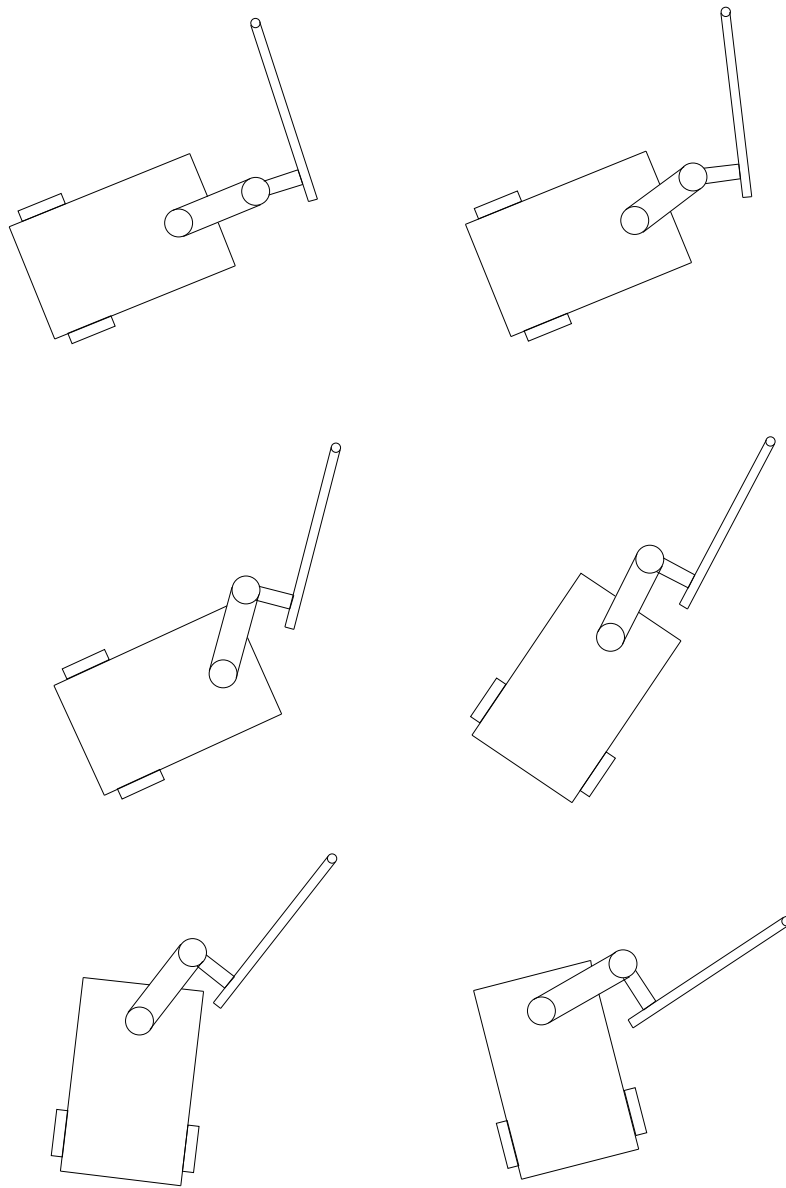
De output van TurnHandle is een constante rotatiesnelheid rond de Z-as van het grijper-assenstelsel (dus de rotatierichting van de deurklink). Hierdoor begint de deurklink te draaien (zie figuur 5.4, vanaf de 28^e seconde¹).

Zolang de deurklink zijn uiterste positie niet bereikt heeft, geeft deze een tegenkoppel (gemiddeld $2Nm$). Dit koppel wordt gemeten in de kracht- en momentensensor. FollowForce stuurt de manipulator aan met een tegengestelde rotatiesnelheid, evenredig met het gemeten koppel.

TurnHandle en FollowForce worden coöperatief gecombineerd. De gecombineerde output is een gewogen som van de afzonderlijke outputs. De uiteindelijke output is dus een rotatiesnelheid rond de as van de deurklink, die wat kleiner is dan de output van TurnHandle.

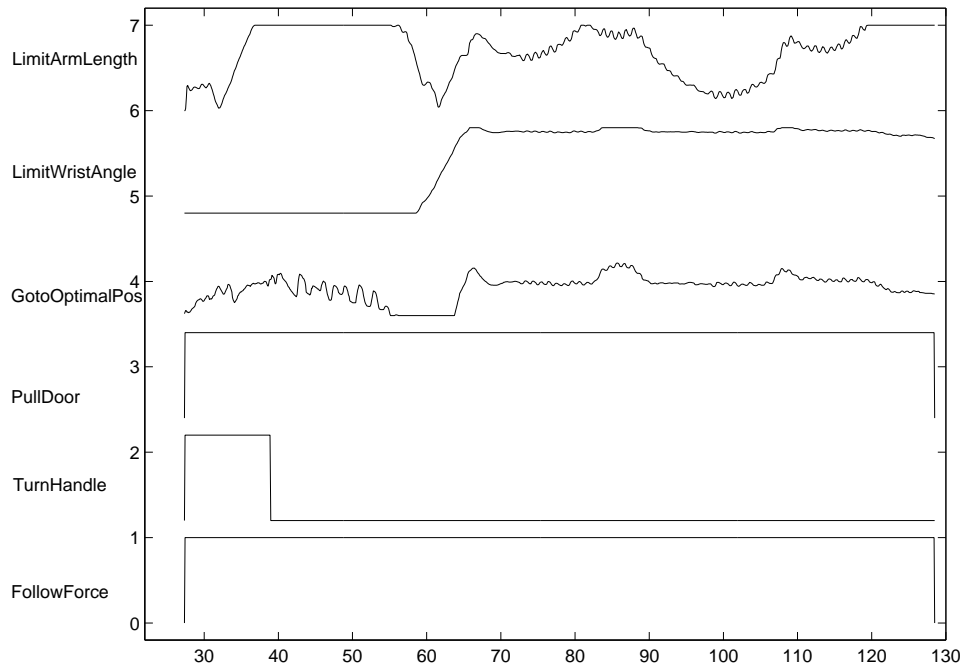
Op het moment dat de deurklink zijn uiterste stand bereikt (op de 33^e seconde), neemt het tegenkoppel ogenblikkelijk toe. FollowForce stuurt de manipulator hierop aan met een grotere rotatiesnelheid. Uiteindelijk stelt

¹Deze figuur toont het moment op de klink en de rotatie van de 6^e as van de robot. Deze rotatie komt dus niet exact overeen met die van de klink, maar geeft een goede indicatie.



Figuur 5.2: De beweging van LiAS bij het openen van een deur

zich een evenwicht in: het tegenkoppel in de grijper is net zo groot (gemiddeld $3Nm$), dat de rotatiesnelheid die FollowForce uitstuurt even groot is en tegengesteld aan de rotatiesnelheid van TurnHandle. De gewogen som van beide gedragingen is dan een snelheid nul, zodat de grijper stopt met draaien en de deurklink in zijn uiterste stand gedraaid blijft. In deze stand oefent de grijper een constant koppel uit op de deurklink.



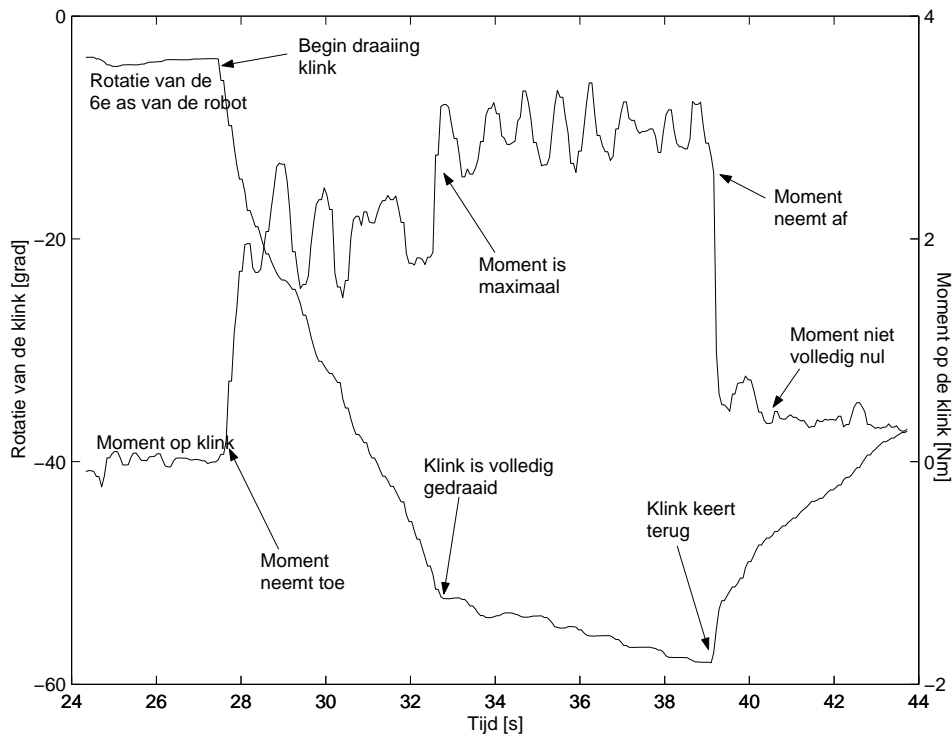
Figuur 5.3: De activatieniveau's doorheen het openen van de deur

Eens de deur voldoende ver geopend is, wordt TurnHandle inactief. De deurklink geeft nog steeds een constant tegenkoppel, zodat FollowForce nog steeds een tegengestelde rotatiesnelheid uitstuurt, waardoor de deurklink en de grijper terug recht draaien. Het tegenkoppel is nu veel lager dan voordien. Dit komt doordat de manipulator in de beginfase, als de deur nog gesloten is, reeds aan de deur trekt. Het slot komt hierdoor onder spanning te staan, zodat het draaien van de deurklink stroever verloopt zolang de deur dicht is.

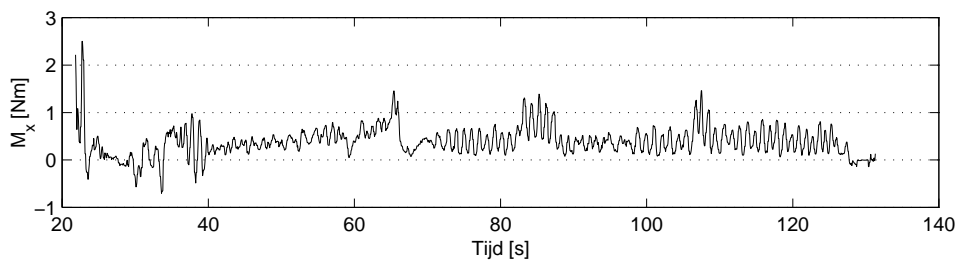
Als de deurklink terug rechtgedraaid is, valt het moment rond de Z-as van het grijper-assenstelsel (zie figuur 4.16) terug op nul. Het moment rond de Y-as schommelt ook rond nul. Enkel het moment rond de X-as van de grijper is gemiddeld $0,4N$ (zie figuur 5.5). Dit komt doordat de robotsturing niets weet over het draaien van de deur en deze draaiing enkel volgt op basis van de momenten die waargenomen worden.

5.2.2 Het trekken aan de deur

Pulldoor stuurt de manipulator zo aan, dat deze trekt aan de deur. Zolang de deurklink nog niet voldoende gedraaid is, beweegt de deur echter niet. Hierdoor ontstaat een trekkracht in de grijper. FollowForce probeert deze te



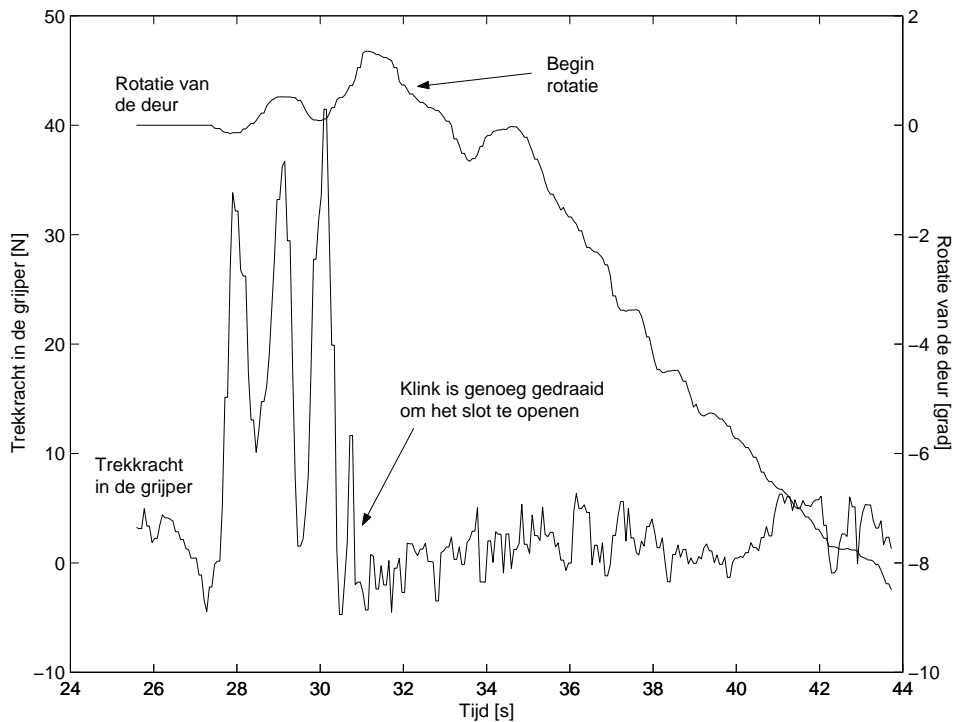
Figuur 5.4: Het draaien van de deurklink



Figuur 5.5: Het moment rond de X-as van de grijper.

compenseren door de manipulator in de tegengestelde richting aan te sturen. De combinatie van PullDoor en FollowForce zorgt ervoor dat een trekkracht ontstaat in de grijper en dat deze stilstaat.

Vanaf het moment dat de deurklink voldoende gedraaid is, kan de deur vrij rond zijn as draaien. De richting loodrecht op de deur wordt dan een vrijheidsgraad. Deze richting is tevens de richting waarin PullDoor de manipulator aanstuurt. PullDoor veroorzaakt hierdoor slechts heel kleine krach-



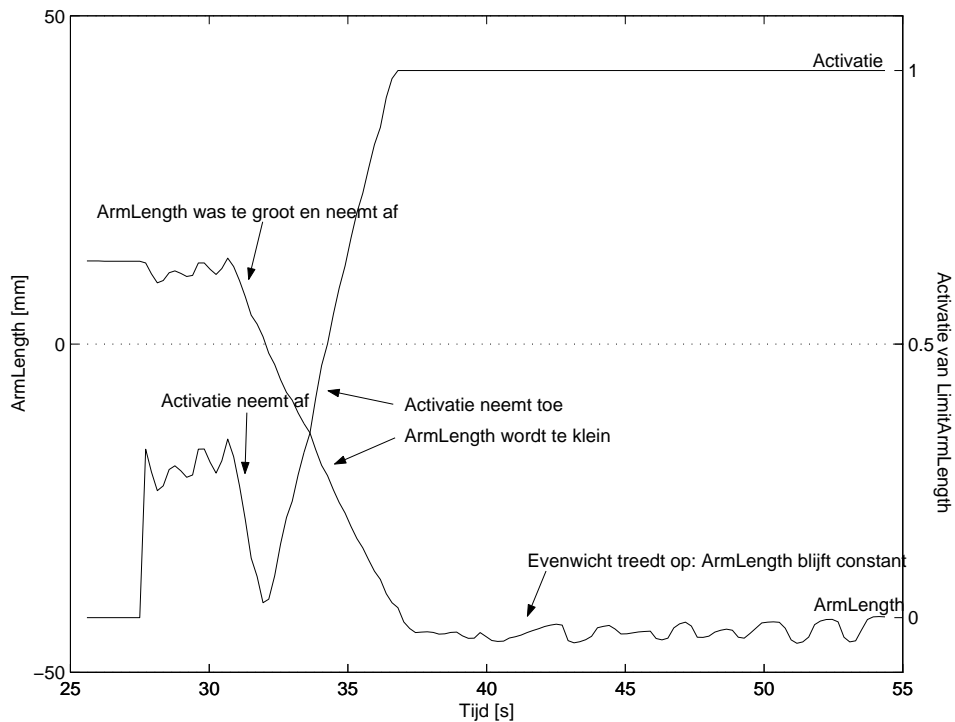
Figuur 5.6: Het trekken aan de deur

ten in de grijper². De deur begint nu open te draaien en de manipulator kort in. Figuur 5.6 toont de trekkracht in de grijper. Zo lang de deur toe is (tot de 31^e seconde), schommelt deze met een gemiddelde waarde van 20N. Vanaf het moment dat de deur opengaat, valt deze gemiddelde waarde terug op 1,5N.

5.2.3 Het beperken van de manipulatorlengte

Vermits de manipulator inkort bij het optrekken van de deur, wordt `LimitArmLength` na verloop van tijd actief (zie figuur 5.7). Deze gedraging legt een snelheid aan het mobiele platform aan, in de richting van de manipulator. De manipulator staat op dat moment echter ongeveer loodrecht op de deur, vermits de polshoek vrij klein is. Het achteruitrijden verloopt dus volgens de vrijheidsgraad van de deur. De deur wordt verder opgetrokken, zonder dat er een noemenswaardige kracht ontstaat, die echter nodig is om de manipulator terug langer te maken. De manipulator voelt (via `FollowForce`) geen verandering van de omgeving en blijft inkorten. In deze stand van de robot volstaat de gezamenlijke werking van `FollowForce`

²Er ontstaan wel momenten, doordat de deur roteert. De `FollowForce`-gedraging zorgt op basis van deze momenten dat de manipulator de draaibeweging van de deur volgt.



Figuur 5.7: Het activatieniveau van LimitArmLength

en LimitArmLength dus niet, om de manipulatorlengte te beperken.

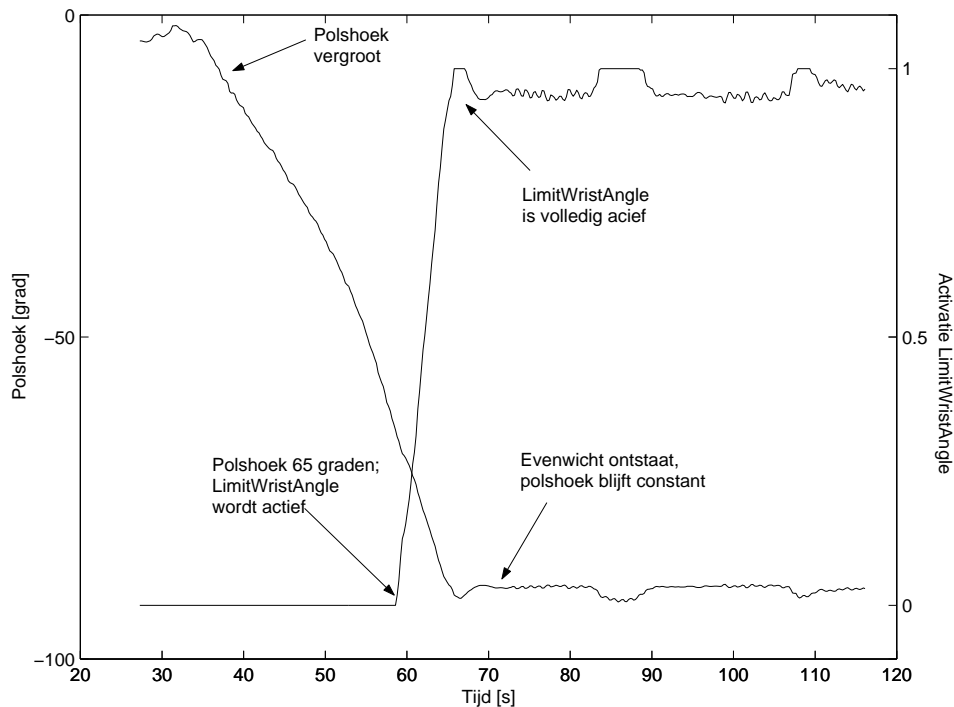
Als de manipulatorlengte te kort wordt, wordt GotoOptimalPosition actief. GotoOptimalPosition stuurt de manipulator zo aan, dat deze uit zijn uiterste stand beweegt. De enige richting die hiervoor geschikt is, is de richting, loodrecht op de deur (dus volgens de vrijheidsgraad van de deur). Als GotoOptimalPosition de manipulator in een andere richting aan zou sturen, dan zou dit resulteren in extra (en ongewenste) krachten en momenten in de grijper³.

De coöperatieve combinatie van PullDoor en GotoOptimalPosition resulteert in een evenwicht, waarbij de manipulatorlengte constant blijft. GotoOptimalPosition wordt net voldoende actief om de werking van PullDoor te gecompenseren. Het openen van de deur wordt nu dus gedaan door het platform. Het is niet mogelijk dat de robot in een dode positie belandt bij het innemen van het evenwicht, vermits LimitArmLength eerder actief wordt dan GotoOptimalPosition. Op het moment dat de manipulator stopt met inkorten, is het mobiele platform dus al aan het rijden.

³Deze krachten en momenten zouden dan net zo groot zijn, dat FollowForce de bewegingen tegenwerkt die GotoOptimalPosition wil aanleggen, vermits de grijper niet kan bewegen in andere richtingen dan de vrijheidsgraad van de deur.

Naar mate de deur verder opent, wordt de polshoek groter. De manipulator en dus ook de richting waarin `LimitArmLength` het mobiele platform aanstuurt, staat dus niet meer loodrecht op de deur. Als de polshoek stijgt, stuurt `LimitArmLength` het mobiele platform dus minder en minder aan volgens de vrijheidsgraad van de deur. De bewegingen van het mobiele platform veroorzaken daardoor krachten in de grijper, die gevolgd worden door `FollowForce`. Als de polshoek groter wordt, wordt de koppeling via de omgeving tussen `LimitArmLength` en `FollowForce` groter, zodat de interactie tussen deze gedragingen volstaat en `GotoOptimalPosition` niet meer nodig is om de manipulatorlengte te beperken.

5.2.4 Het beperken van de polshoek

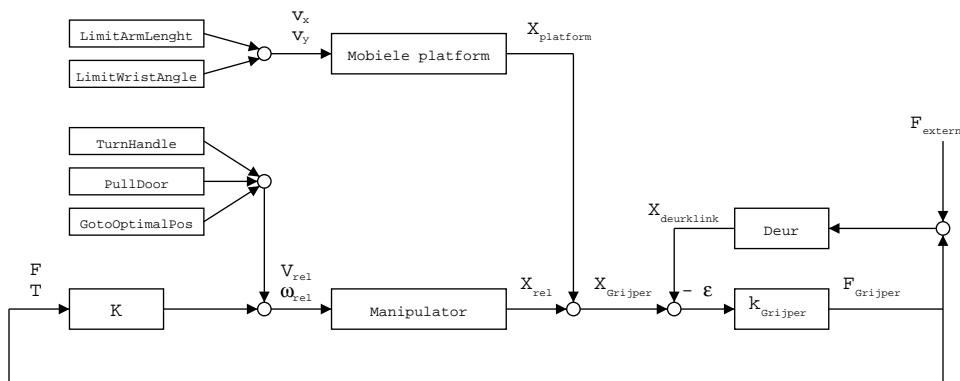


Figuur 5.8: Het activatieniveau van `LimitWristAngle`

Het omgekeerde effect treedt op bij `LimitWristAngle`. Deze gedraging wordt actief bij een polshoek van 65° (zie figuur 5.8) en beweegt het mobiele platform in een richting, loodrecht op de manipulator. Naar mate de polshoek 90° nadert, wordt deze richting meer en meer de richting van de vrijheidsgraad van de deur. `LimitWristangle` wordt dus minder effectief naarmate de polshoek groter wordt.

Als de polshoek te groot wordt, wordt GotoOptimalPosition actief. Deze stuurt de manipulator aan volgens de richting van de vrijheidsgraad van de deur, zodat dat de manipulator uit zijn extreme positie gaat. De coöperatieve combinatie van PullDoor en GotoOptimalPosition resulteert in een evenwicht, waarbij de polshoek constant blijft. GotoOptimalPosition wordt net zo actief, dat de werking van PullDoor en de invloed op de polshoek van het draaien van de deur, gecompenseerd worden. Het is niet mogelijk dat de robot in een dode positie belandt bij het innemen van dit evenwicht, vermits LimitWristAngle eerder actief wordt dan GotoOptimalPosition. Op het moment dat de manipulator stopt met bewegen is het mobiele platform al aan het rijden.

5.3 Benadering vanuit regeltechnisch standpunt



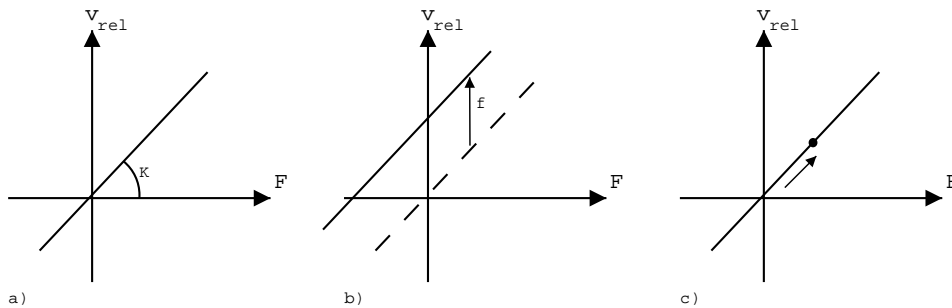
Figuur 5.9: Regeltechnische benadering van de gedragingen

In deze regeltechnische benadering (zie figuur 5.9) wordt de robotsturing beschouwd vanuit het standpunt van de manipulator. FollowForce staat in voor impedantiecontrole. Deze agent legt het verband vast tussen de kracht⁴ in en de relatieve snelheid van de gripper⁵. Dit verband is proportioneel (zie figuur 5.10 a) en wordt voor elk van de zes richtingen afzonderlijk vastgelegd.

De gripper is met een bepaalde stijfheid verbonden met de omgeving. Een verschil tussen de positie van de gripper en die van de deurklink, uit zich hierdoor als een kracht in de gripper. Deze kracht wordt opgemeten en teruggekoppeld. De richting loodrecht op de deur is een vrijheidsgraad van de deur. In deze richting zijn er dus enkel traagheidskrachten en eventuele wrijvingskrachten van de deur.

⁴Met 'kracht' wordt hier zowel 'kracht' als 'moment' bedoeld. De term 'positie' duidt hier zowel op positie als op oriëntatie.

⁵relatief ten opzichte van het mobiele platform.



Figuur 5.10: Impedantiekarakteristieken

PullDoor, TurnHandle en GotoOptimalPosition geven stuurwaardes voor de manipulator (dus relatieve manipulatorsnelheden) en komen in het regelschema terecht als externe storingen aan de ingang van de manipulator. Deze gedragingen geven aanleiding tot het verschuiven van de impedantie-karakteristiek (zie figuur 5.10 b). De karakteristiek gaat dan niet meer door de oorsprong.

In een richting die geen vrijheidsgraad van de deur is, moet de snelheid van de grijper nul zijn. Hierdoor resulteert het verschuiven van de karakteristiek in het ontstaan van een kracht in de grijper.

In de richting loodrecht op de deur, die wel een vrijheidsgraad is, is de optredende kracht ongeveer nul en resulteert de verschuiving van de karakteristiek in een snelheid.

Daar PullDoor en GotoOptimalPosition de manipulator aansturen volgens de vrijheidsgraad van de deur, resulteren deze storingen dus enkel in een beweging van de deur en niet in het ontstaan van krachten.

De situatie is analoog voor TurnHandle. Zolang de deurklink nog niet volledig gedraaid is, is er een rotatiesnelheid. Vermits de klink een constant tegenkoppel geeft, ligt het werkingpunt niet op het snijpunt van de impedantie-karakteristiek met de X -as, maar ergens tussen het snijpunt met de X -as en dat met de Y -as. Op het moment dat de deurklink in de uiterste positie aanbeldt, stopt de beweging echter⁶. Het werkingpunt is dan het snijpunt van de karakteristiek met de Y -as, zodat er een blijvend moment is in de grijper (zie figuur 5.4).

De beweging van de wielbasis kan beschouwd worden als een externe storing aan de uitgang van de manipulator. Als de beweging niet volgens de vrijheidsgraad van de deur ligt, geeft deze aanleiding tot een tijdelijke verschuiving van het werkingpunt in de impedantie-karakteristiek (zie 5.10 c). Het verschil in positie tussen de grijper en de deurklink veroorzaakt een be-

⁶In de veronderstelling dat het mobiele platform stilstaat.

paalde kracht. Met die kracht komt dan volgens de impedantie karakteristiek een manipulatorsnelheid overeen (dus een relatieve snelheid; absoluut is de snelheid nul).

Hoofdstuk 6

Conclusie en aanbevelingen

De eerste paragraaf bespreekt enkele conclusies over het uitgevoerde onderzoek. De tweede paragraaf geeft een aantal suggesties voor verder onderzoek.

6.1 Conclusie over de gedragsgebaseerde theorie

Dit eindwerk toont aan dat de gedragsgebaseerde theorie niet enkel voor navigatietaken, maar ook voor manipulatie taken toepasbaar is. Deze paragraaf bespreekt zowel het potentiële als de problemen van gedragsgebaseerde sturing.

Het potentiële ligt in het feit dat complexe taken kunnen opgesplitst worden in eenvoudige deeltaken. Het oplossen van het totale probleem bestaat dan uit het oplossen van de afzonderlijke deelproblemen. Elk van deze deelproblemen kan apart uitgewerkt en getest worden. De deeloplossingen worden nadien samengevoegd. Het volledige probleem moet dus niet in één keer aangepakt worden, maar kan in verschillende stappen opgelost worden. Dit maakt het probleem gemakkelijk beheersbaar. Indien later de taak nog wordt uitgebreid, kan dit door het extra deelprobleem op te lossen en dit toe te voegen aan het geheel.

Een probleem van de gedragsgebaseerde sturing is de moeilijke voorspelbaarheid van het uiteindelijke gedrag. Het effect van elke gedraging afzonderlijk is direct gekend. Het is echter moeilijker om het effect van alle gedragingen samen te voorspellen. Deze worden immers op verschillende manieren met elkaar gecombineerd, niet elke gedraging is steeds actief, gedragingen interageren met elkaar via de omgeving,...

Een ander probleem is de parameterafhankelijkheid van de verschillende gedragingen. Het experimenteel bepalen van alle parameters is zeer tijdrovend. Een mogelijke oplossing hiervoor is het gebruik van leergedragingen, die zelf hun parameters bepalen.

6.2 Aanbevelingen voor verder onderzoek

6.2.1 Uitbreidingen voor de programma-architectuur

De gebruikte programma-architectuur biedt een flexibele structuur aan waarbinnen verschillende agents actief kunnen zijn. Ook biedt de architectuur de mogelijkheid aan om met meerdere, zeer verschillende sensoren en actuatoren om te gaan. In dit eindwerk werd deze architectuur voor de eerste keer in de praktijk getest.

In deze paragraaf komen een aantal ideeën aan bod om de architectuur nog te verbeteren.

Elk van de delen van de mobiele manipulator (het mobiele platform en de manipulator) hebben verschillende vrijheidsgraden. Een gedraging die één van deze delen aanstuurt, moet voor elk van de vrijheidsgraden dat deel een aanstuurwaarde bepalen. Het komt echter vaak voor dat een gedraging slechts een nuttige aanstuurwaarde heeft voor één van de vrijheidsgraden.

Bijvoorbeeld: De PullDoor agent legt enkel een snelheid op, loodrecht op de deur gericht. In alle andere richtingen kan PullDoor geen nuttige stuurdata geven.

Een eerste uitbreiding van de architectuur realiseert dit door de agents de mogelijkheid te geven een vrijheidsgraad met 'don't care' aan te sturen.

In de huidige architectuur berekent een agent steeds uit een aantal inputs de aanstuurwaardes voor een aantal actuatoren. Een agent kan echter de output van een andere agent niet rechtstreeks als input gebruiken. Het gebruik van een model, waarbij een agent een model opstelt en een andere agent op basis van dat model zijn aanstuurwaardes berekent, is dus moeilijk te implementeren.

Een tweede uitbreiding van de architectuur realiseert dit door mogelijkheid de output van een agent rechtstreeks te gebruiken als input voor een andere agent.

6.2.2 Uitbreidingen van dit eindwerk

De realisatie van dit eindwerk, namelijk een aantal gedragingen die samen in staat zijn een willekeurige deur te openen, is nog verder uitbreidbaar.

Nu leidt de mens LiAS met behulp van krachtvolging naar de deurklink. Deze fase kan met behulp van extra sensoren (bijvoorbeeld camera's) volledig geautomatiseerd worden. LiAS kan autonoom de positie van de deurklink zoeken en vervolgens in de richting ervan bewegen.

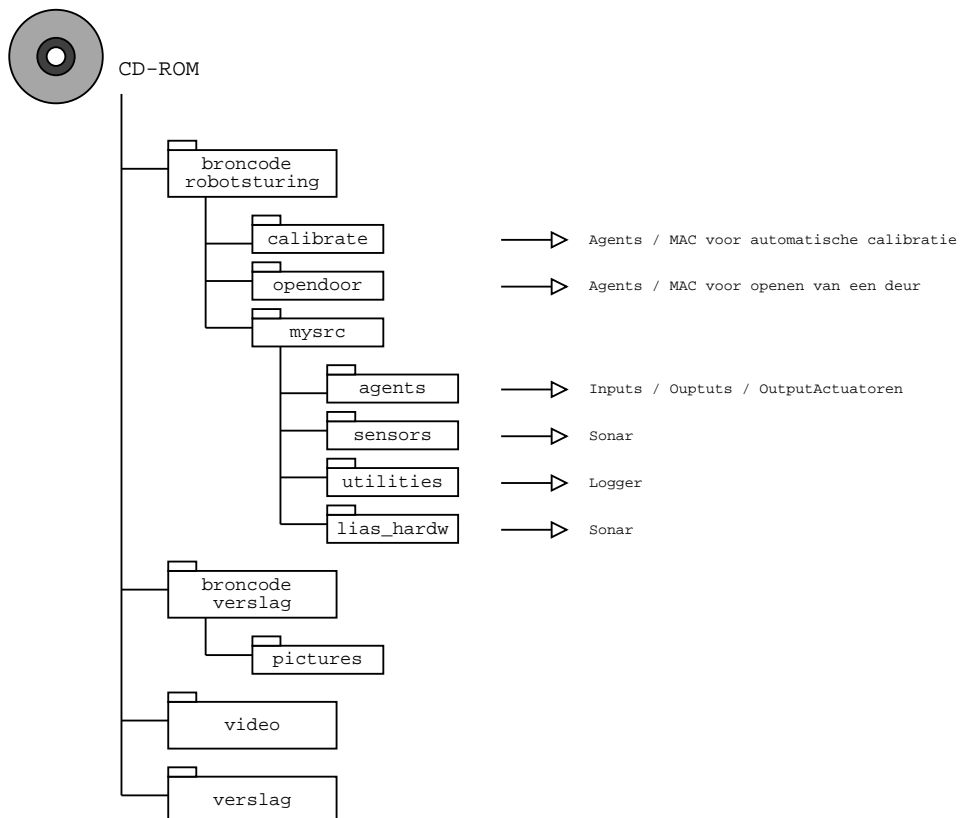
Zoals reeds eerder vermeld is het experimenteel bepalen van de parameters

van de verschillende gedragingen zeer tijdrovend. Met behulp van leergerdringen kan het bepalen van deze parameters door het programma overgenomen worden.

Bijlage A

Implementatie van de klassen in C++

De omvang van de in dit eindwerk geschreven C++ code is te groot om deze volledig af te drukken. Daarom is de volledige implementatie beschikbaar op de bijgevoegde cd-rom. Figuur A.1 geeft de inhoud van de cd-rom weer.



Figuur A.1: Inhoud van de cd-rom

Bibliografie

- [1] A. Astigarraga Pagoaga. Agent-based architecture for a mobile robot. Master's thesis, Departement Mechanica, Katholieke Universiteit Leuven, 2001.
- [2] M. D. Adams. *Sensor Modelling, Design and Data Processing for Autonomous Navigation*. World Scientific, 1999.
- [3] Adaptive Communication Environment. <http://www.cs.wustl.edu/~schmidt/ACE.html>.
- [4] R. C. Arkin. *Behavior-based Robotics*. The MIT Press, 1998.
- [5] A. V. Breemen. *Agent-based multi-controller systems*. PhD thesis, Enschede, 2001.
- [6] R. Brooks. *Cambrian Intelligence*. The MIT Press, 1999.
- [7] A. Casals. *Sensor Devices and Systems for Robotics*. Springer - Verlag, 1987.
- [8] R. B. Davies. Newmath, a matrix library in c++, 1997.
- [9] G. Pinte, R. Kuppens. Gedragsgebaseerde sturing van een mobiele manipulator. Master's thesis, Departement Mechanica, Katholieke Universiteit Leuven, 2001.
- [10] R. Gourdeau. *A Robotics Object Oriented Package in C++*. PhD thesis, École Polytechnique de Montréal, 2001.
- [11] S. S. Iyengar. *Autonomous Mobile Robots*. IEEE Computer Society Press, 1991.
- [12] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [13] M. Kasper, G. Fricke, K. Steuernagel, E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, 34:153–164, 2001.

- [14] M. J. Mataric. Behavior-based control: Main properties and implications. *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pages 46–54, May 2000.
- [15] R. M. Murray. *A mathematical introduction to robotic manipulation*. CRC Press, 1994.
- [16] T. Owen. *Assembly with Robots*. The Anchor Press, 1985.
- [17] P. Pirjanian. Behavior coordination mechanisms – state-of-the-art. Technical report, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California, 1999.
- [18] R. Waarsing, H. Van Brussel. A software framework for control of a multi-sensor, multi-actuator system. *submitted for Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.
- [19] R. Waarsing. More - mobile robot environment. Technical report, Departement Mechanica, Katholieke Universiteit Leuven, 2002.