Hoe kan de robotarm tafeltennissen?

Onderzoeksrapport – AP Project – Iets Technisch 1

15-1-2016

HAN

Remco van Alen, Bas van Summeren, Michiel Buevink, Paul Verhoeven, Thomas Fransen

Versie 1.0

# Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie nummer | Omschrijving | Datum | Opgeleverd aan |
| 0.1 | Eerste opzet | 22-10-2015 |  |
| 0.2 | Conclusies toegevoegd | 02-11-2015 |  |
| 0.3 | Concept versie (Theoretisch deel) | 10-11-2015 | Leon Bronckers |
| 0.4 | Praktisch onderzoek beeldherkenning bijgewerkt | 15-12-2015 |  |
| 0.5 | Resultaten van methoden toegevoegd aan praktisch onderzoek beeldherkenning | 11-1-2015 |  |
| 0.6 | Praktisch onderzoek over traject en aansturing toegevoegd en een conclusie en samenvatting geschreven |  |  |
| 1.0 | Definitieve versie | 14-01-2015 | Leon Bronckers |

# Inhoudsopgave

[Versiebeheer 1](#_Toc440634255)

[Inhoudsopgave 2](#_Toc440634256)

[1 Managementsamenvatting 4](#_Toc440634257)

[1.1 Tafeltennis 4](#_Toc440634258)

[1.2 De robotarm 4](#_Toc440634259)

[1.3 De montage van het batje 4](#_Toc440634260)

[1.4 Veiligheid 4](#_Toc440634261)

[1.5 Beeldherkenning 5](#_Toc440634262)

[1.6 De programmeertaal 5](#_Toc440634263)

[1.7 Communicatie 5](#_Toc440634264)

[1.8 Conclusie 6](#_Toc440634265)

[1.9 Aanbeveling 6](#_Toc440634266)

[2 Inleiding 7](#_Toc440634267)

[3 Tafeltennis 9](#_Toc440634268)

[3.1 Bereik 9](#_Toc440634269)

[3.2 Reactietijd 10](#_Toc440634270)

[4 De robotarm 11](#_Toc440634271)

[4.1 De snelheid 11](#_Toc440634272)

[4.2 De kracht 12](#_Toc440634273)

[4.3 Draaihoeken 13](#_Toc440634274)

[4.4 De montage van het batje 15](#_Toc440634275)

[5 Veiligheid 16](#_Toc440634276)

[5.1 Wat is veiligheid? 16](#_Toc440634277)

[5.2 Hoe wordt de veiligheid gehandhaafd? 16](#_Toc440634278)

[6 Beeldherkenning 18](#_Toc440634279)

[6.1 Nodige eigenschappen voor de beeldherkenning 19](#_Toc440634280)

[6.2 Beschikbare beeldherkenning technieken/middelen 19](#_Toc440634281)

[7 De programmeertaal 28](#_Toc440634282)

[7.1 Real-time 28](#_Toc440634283)

[7.2 Communicatie 28](#_Toc440634284)

[7.3 Objecttracking 28](#_Toc440634285)

[7.4 User base/support 28](#_Toc440634286)

[7.5 De programmeertaal 29](#_Toc440634287)

[8 Communicatie 31](#_Toc440634288)

[8.1 Communicatiemethoden 31](#_Toc440634289)

[8.2 Vergelijking 35](#_Toc440634290)

[8.3 Coördinaten of hoeken? 35](#_Toc440634291)

[9 Tussentijdse conclusies 36](#_Toc440634292)

[9.1 Programmeertaal 36](#_Toc440634293)

[9.2 Besturingssysteem 36](#_Toc440634294)

[9.3 Beeldherkenning 37](#_Toc440634295)

[9.4 Eigenschappen 37](#_Toc440634296)

[9.5 Veiligheid 38](#_Toc440634297)

[9.6 Communicatie 38](#_Toc440634298)

[10 Praktijk onderzoek 39](#_Toc440634299)

[10.1 Beeldherkenning 39](#_Toc440634300)

[10.2 Traject bepalen 50](#_Toc440634301)

[10.3 Reactiesnelheid 53](#_Toc440634302)

[10.4 Aansturing 54](#_Toc440634303)

[11 Conclusies 58](#_Toc440634304)

[11.1 Beeldherkenning 58](#_Toc440634305)

[11.2 Aansturing 58](#_Toc440634306)

[11.3 Traject berekening 59](#_Toc440634307)

[11.4 Hoeken berekening 59](#_Toc440634308)

[11.5 Algemene conclusie 60](#_Toc440634309)

[11.6 Aanbevelingen 60](#_Toc440634310)

[Literatuurlijst 61](#_Toc440634311)

[Bijlagen 63](#_Toc440634312)

[1. Daily inspection items 63](#_Toc440634313)

[2. Periodic inspection 64](#_Toc440634314)

[3. Demo video 64](#_Toc440634315)

# Managementsamenvatting

Dit project is uitgevoerd voor de minor ‘Advanced Programming’. Voor dit project is er onderzoek verricht naar de robotarm ‘RV-2AJ’. De ICA wil deze robot graag toepassen in het onderwijs, maar heeft daarvoor meer informatie over limieten en mogelijkheden van de robot nodig. Ook is het een wens om de robot vanuit C of C++ aan te sturen.

Om een uitdagend project op te zetten waarin al deze aspecten onderzocht worden is er bedacht om de robot te laten tafeltennissen. Daarbij is de volgende hoofdvraag opgesteld: “Hoe kan de robotarm, Melfa RV-2AJ, tafeltennissen?”

## Tafeltennis

Door een groep van vijf studenten is er een onderzoek uitgevoerd. Aan de hand van de tussentijdse conclusies is er een praktijkonderzoek verricht. Voor het theoretische onderzoek is eerst het spel ‘tafeltennis’ geanalyseerd. Daaruit is onder andere voortgekomen wat de gemiddelde snelheden zijn van het spel. Bij een professioneel spel kunnen de snelheden van het balletje oplopen tot 35 m/s. Om van een laag speelniveau uit te gaan is er een video gemaakt van twee tafeltennissende personen. Hierbij was het niveau zeer laag en werden de snelheden gemeten rond de 5 m/s.

## De robotarm

Om vast te stellen of de robotarm kan tafeltennissen zijn er diverse eigenschappen van de robot onderzocht. De onderzochte eigenschappen zijn: de snelheid, de kracht, de houding en het bereik. Deze zijn geselecteerd, omdat ze het gedrag van de robot het meest beïnvloeden voor het spelen van tafeltennis. De snelheden van de robot zijn vastgelegd per scharnier. De onderste scharnier heeft een maximale snelheid van 180 graden per seconde. Hieruit kon al geconcludeerd worden dat de robot minimaal één seconde de tijd nodig heeft om het batje van links naar rechts te houden. Wanneer het balletje met 5 m/s wordt gespeeld heeft de robot ongeveer 1 seconden tussen de slagen. In deze tijd moet ook het balletje worden getraceerd en het traject hiervan worden berekend.

Ook moet er rekening worden gehouden met de grenzen van alle scharnieren. Deze grenzen zijn per scharnier verschillend. Het onderste scharnier kan bijvoorbeeld van -150 tot +150 graden en daardoor kunnen de achterste 60 graden niet worden bereikt. Deze grenzen zijn ook zeer belangrijk voor het vaststellen van de te bereiken posities. Het is namelijk mogelijk dat de robotarm zichzelf raakt. Wanneer een batje aan de arm wordt gemonteerd zullen deze grenzen opnieuw vastgesteld moeten worden.

## De montage van het batje

Het is belangrijk om het batje zo efficiënt mogelijk te monteren. Er zijn twee mogelijke mogelijkheden onderzocht. Het batje kan in het verlengde en haaks worden geplaatst. De haakse positie heeft als voordeel dat er een slag beweging kan worden gemaakt. Daarentegen heeft het batje in het verlengde plaatsen als voordeel dat er met hetzelfde batje een groter bereikt is ten opzichte van de haakse positie. Uiteindelijk is er gekozen voor de haakse positie, omdat de lengte van het batje altijd kan worden verlengd, en dan heeft deze positie ook nog een extra voordeel van de slagpositie.

## Veiligheid

In technische systemen staat veiligheid voorop en daarom wordt in dit onderzoek uitgebreid gekeken naar de handhaving van de veiligheid. Wanneer het batje op de robot wordt gemonteerd is er een groot vlak om de robot dat bereikt kan worden. Daarom wordt er een veiligheidszone om het werkgebied van de robot geplaatst. Deze veiligheidszone zal worden aangegeven met een lint, omdat dit duidelijk en mobiel is.

## Beeldherkenning

Om te kunnen tafeltennissen, moet het systeem weten waar het balletje is. Er is gekozen om de beelden op te vangen door middel van twee webcams, omdat webcams gemakkelijk in gebruik zijn en toch goede informatie hebben. Eén camera zal boven de tafel bevestigd worden voor het bovenaanzicht en de andere zal aan de zijkant geplaatst worden voor het zijaanzicht. Dit vormt samen een driedimensionale locatie.

Een aantal eigenschappen van de camera’s zijn belangrijk. Als de camera niet genoeg beelden per seconde krijgt zullen de verschillen in de positie van het balletje te groot zijn. Ook moet de kijkhoek groot zijn zodat de camera’s binnen een acceptabele afstand geplaatst kunnen worden.

Daarbij zijn diverse beeldherkenningsmethoden onderzocht. Hieruit zijn twee methoden gekozen, ‘Hough Circle Detection’ en ‘Particle Filter Color Tracking’.

De Hough Circle Detection is gekozen omdat deze methode de vorm van het balletje goed zou kunnen herkennen in ieder beeld. Echter bleek het balletje in de praktijk niet gezien te worden door de methode. Door de beweging is het balletje niet volledig rond meer, maar wordt het uitgerekt(waas). Dit komt door een te hoge sluitertijd van de camera. Hierdoor is deze methode niet zeer geschikt voor dit project.

Ook is er voor Particle Filter Color Tracking methode gekozen omdat deze snel en doelgericht een object van een specifieke kleur zal kunnen herkennen. Om dit proces te verbeteren is dan ook voor oranje pingpong balletjes gekozen. Deze methode heeft minder problemen met vervormingen door een te lage sluitertijd. Echter bij zeer hoge snelheid zullen de kleuren vervagen door de lage sluitertijd.

Uit een snelheidsmeting blijkt ook dat de Particle Filter dubbel zoveel beeldjes per seconde kan verwerken als de Hough Circle Detection. Hieruit is geconcludeerd dat de Particle Filter de betere beeldherkenningsmethode is voor dit project. Echter zouden de resultaten beter zijn als de camera’s een lage sluitertijd hebben.

## De programmeertaal

Voor de ontwikkeling van het systeem is het belangrijk dat er een real-time programmeertaal wordt gebruikt. Er zijn diverse talen met elkaar vergeleken op basis van real-time gedrag, op welke besturingssystemen die beschikbaar is, of er communicatie naar buiten mogelijk is en of er libraries voor objecttracking beschikbaar zijn. Daaruit is voortgekomen dat C++ de meest geschikte taal is voor dit project. Achteraf is gebleken dat deze taal ook de voorkeur had van de opdrachtgever.

## Communicatie

De aansturing van de robot is onderverdeeld in twee stukken, de aansturing op de controller en de aansturing in de software. In het programma wordt er een seriële poort geopend waarna er een bericht wordt gestuurd. In dit bericht zullen de hoeken worden geplaatst die zijn berekend.

## Conclusie

Er kan worden geconcludeerd dat de Robotarm alleen snel genoeg kan reageren als de bal meer dan een halve seconde nodig heeft om de tafel over te steken. De robotarm reageert niet snel genoeg op de input en beweegt niet snel genoeg om het balletje terug te kunnen slaan. Daarbij heeft de arm niet genoeg bereik om diagonaal gespeelde ballen(die aan de zijkant van de tafel af zullen gaan) terug te kunnen slaan. Vervolgens kan de beeldherkenning geen snelle ballen detecteren doordat de webcams een te hoge sluitertijd hebben. Met onscherpe beelden kan het balletje niet gedetecteerd worden.

Met deze resultaten kan de hoofdvraag worden beantwoord:

“Hoe kan de robotarm, Melfa RV-2AJ, tafeltennissen?”

Met de huidige opstelling kan geconcludeerd worden dat de Melfa RV-2AJ niet kan tafeltennissen.

## Aanbeveling

Er is dus geconcludeerd dat de robot Melfa RV-2AJ niet in staat is om te tafeltennissen. De robotarm en de camera’s zijn daarbij het grootste knelpunt. Daarom wordt er aanbevolen om de robotarm te vervangen met een snellere robotarm, met een hogere reactiesnelheid. Hierdoor zal de arm sneller op de gewenste locatie kunnen komen waardoor hij meer ballen kan tegenhouden. Daarnaast wordt er aanbevolen om de webcams te vervangen met andere camera’s die een kortere sluitertijd hebben.

# Inleiding

De afdeling ICA heeft een robotarm, van het type Melfa RV-2AJ, overgenomen van de afdeling Elektrotechniek. Deze arm heeft op dit moment geen functie en daarom wil de opdrachtgever graag meer weten over de mogelijkheden van de robotarm. In de loop van het onderzoek is gebleken dat de opdrachtgever ook het doel heeft om deze robotarm in te kunnen zetten in het verdiepende semester van het Technische Informatica profiel. Hiervoor moet documentatie komen hoe de robot ingesteld moet worden en hoe deze aangestuurd kan worden vanuit C en/of C++.

Aan de hand van de hoofdvraag “Hoe kan de robotarm, Melfa RV-2AJ, tafeltennissen?” zijn diverse deelvragen opgesteld. Deze zijn opgesteld in het onderzoeksplan.

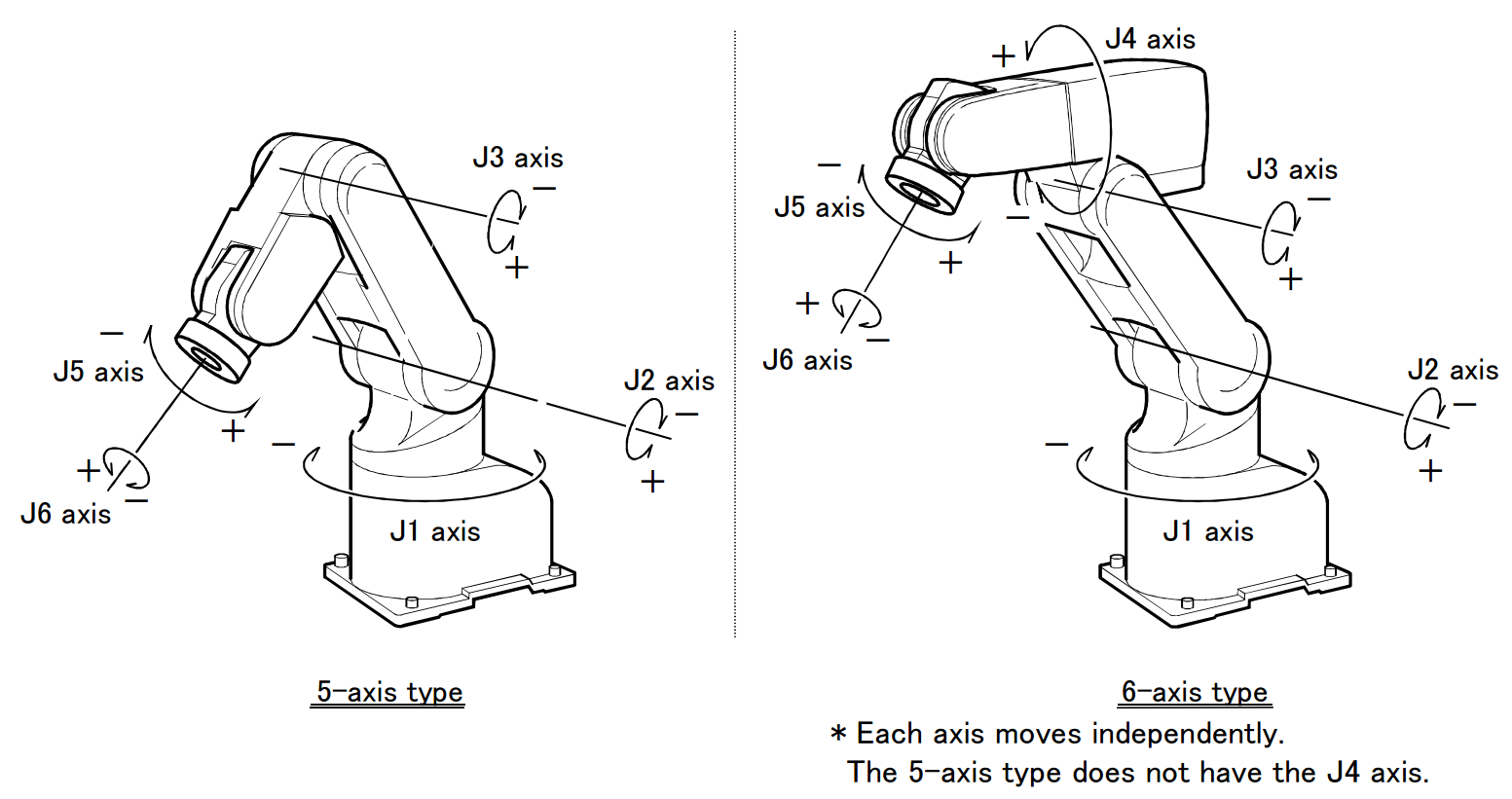
De deelvragen zijn onderverdeeld in een aantal hoofdstukken waarin ze worden behandeld. De onderverdeling van de deelvragen staat in Tabel 1.

Tabel Onderverdeling deelvragen

|  |  |
| --- | --- |
| Hoofdstuk | Deelvraag |
| 2  Tafeltennis | 1. Welke eigenschappen zijn van belang voor de robotarm om deze robotarm te laten tafeltennissen? |
| 4 De robotarm | 2. Wat zijn de technische limieten van de robotarm? |
|  | 3. Welke aanpassingen moeten verricht worden aan de robotarm om de robotarm te laten tafeltennissen? |
|  | 6. Hoe kan de robotarm communiceren tussen de software en hardware? |
| 5 Veiligheid | 4. Hoe kan de veiligheid worden gegarandeerd? |
| 6 Beeldherkenning | 7. Hoe wordt het tafeltennisballetje gedetecteerd? |
| 7 De programmeertaal | 5. Welke programmeertaal is het meest geschikt om de robotarm te programmeren? |
| 8 Communicatie | 5. Welke programmeertaal is het meest geschikt om de robotarm te programmeren? |

Dit onderzoek is uitgevoerd in twee delen, een theoretisch deel en een praktisch deel. In dit verslag zijn eerst de deelvragen theoretisch onderzocht en beantwoord in de tussentijdse conclusies. Daarop volgt een hoofdstuk met praktische bevindingen en algemene conclusies.

De robotarm is een industriële robotarm en wordt ook binnen bedrijven gebruikt. Het is een ‘articulated robotarm’, want de arm heeft 5 draaipunten en lijkt op een menselijke arm. De specifieke serie van de robotarm, Melfa RV-2AJ, heeft twee verschillende uitvoeringen. De ‘RV-2AJ’ en de ‘RV-2AJC-SB’. Tijdens dit project wordt de ‘RV-2AJ’ gebruikt en dit is het 5-axis type. Dit is de linker robot in Figuur 1.



Figuur De verschillende types (Mitsubishi Electric Corporation, 2002)

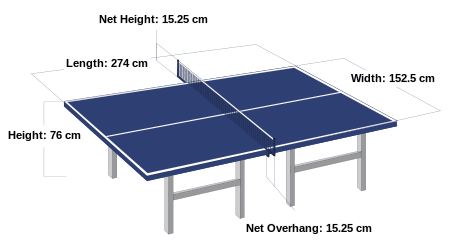
# Tafeltennis

|  |
| --- |
| Behandelde deelvraag:   1. Welke eigenschappen zijn van belang voor de robotarm om deze robotarm te laten tafeltennissen? |

In dit hoofdstuk zullen de eigenschappen worden besproken die de robot nodig heeft om te kunnen tafeltennissen. Het zal hier gaan over het bereik en de reactietijd. Deze eigenschappen zijn relevant omdat de robotarm alleen zal kunnen worden gebruikt wanneer deze de bal kan raken. De reactietijd is belangrijk omdat de robotarm snel moet kunnen reageren.

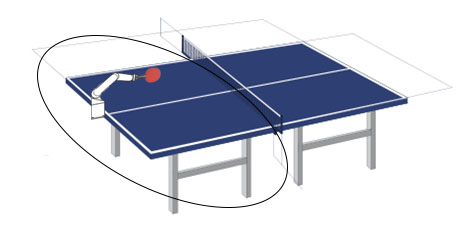
## **Bereik**

Een pingpongtafel is 274cm lang en 152,5cm breed. Tijdens een wedstrijd is er geen limiet aan de hoogte van de slag, behalve die van de hoogte van het plafond. Tijdens een officiële wedstrijd moet deze minimaal vier meter hoog zijn (NTTB). De robotarm zal worden gemonteerd op het midden van de rand. Hier is voor gekozen omdat de robotarm hier het meeste bereik heeft. De rand van de tafeltennistafel is 152,5cm breed. De robotarm zal dus minimaal een bereik van 76,25cm naar beide kanten moeten hebben. Omdat in de praktijk de bal regelmatig schuin wordt gespeeld is er een groter bereik nodig.



Figuur Afmetingen tafeltennistafel (Panguluri, 2014)

Hieronder staat een schematische weergave van het bereik van de robotarm welke deze nodig heeft om alle ballen terug te kunnen slaan. De robot staat links weergegeven.



Figuur Schematische weergave tafeltennis

## Reactietijd

Een professionele tafeltennisspeler kan de bal slaan met een snelheid van 35m/s. (Tang, Mizoguchi, & Toyoshima). De tafel is 2.74 meter lang. Dit zorgt ervoor dat wanneer een topspeler de bal slaat deze er 2.74/35 = 0,078 seconden over doet om aan de andere kant van de tafel te komen. De robot die tijdens dit onderzoek gebruikt wordt is niet tot deze snelheden in staat. Tijdens dit project wordt er van uitgegaan dat de robot een halve seconde de tijd krijgt heeft om een balletje terug te slaan. Dit is gebleken uit een praktijktest waarbij er op amateuristisch niveau is getafeltennist. De bal mag dus met een maximale snelheid van 2,74/0,5 = 5,48 m/s worden geslagen. In de berekende afstanden komt geen kromming voor. Tijdens een wedstrijd zal een bal bijna nooit recht worden geslagen maar zal er altijd een kromming in zitten. Dit zorgt ervoor dat de robot iets meer tijd heeft om te reageren.

# De robotarm

|  |
| --- |
| Behandelde deelvragen:   1. Wat zijn de technische limieten van de robotarm?    1. Wat is de maximale snelheid?    2. Wat is de maximale kracht?    3. Wat zijn de draaihoeken van alle scharnieren? 2. Welke aanpassingen moeten verricht worden aan de robotarm om de robotarm te laten tafeltennissen? 3. Hoe kan de robotarm communiceren tussen de software en hardware?    1. Welke protocollen zijn van belang?    2. Hoe wordt de robot aangesloten?       1. Welke software is nodig op de computer?       2. Welke hardware is nodig om de robot aan te sluiten op een computer? |

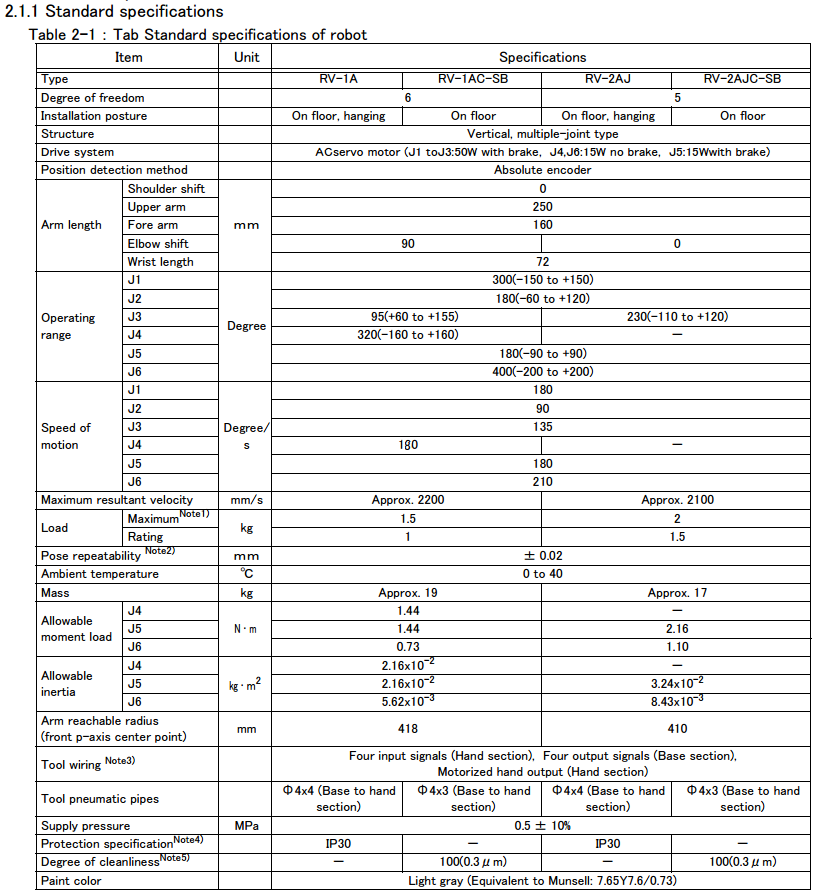
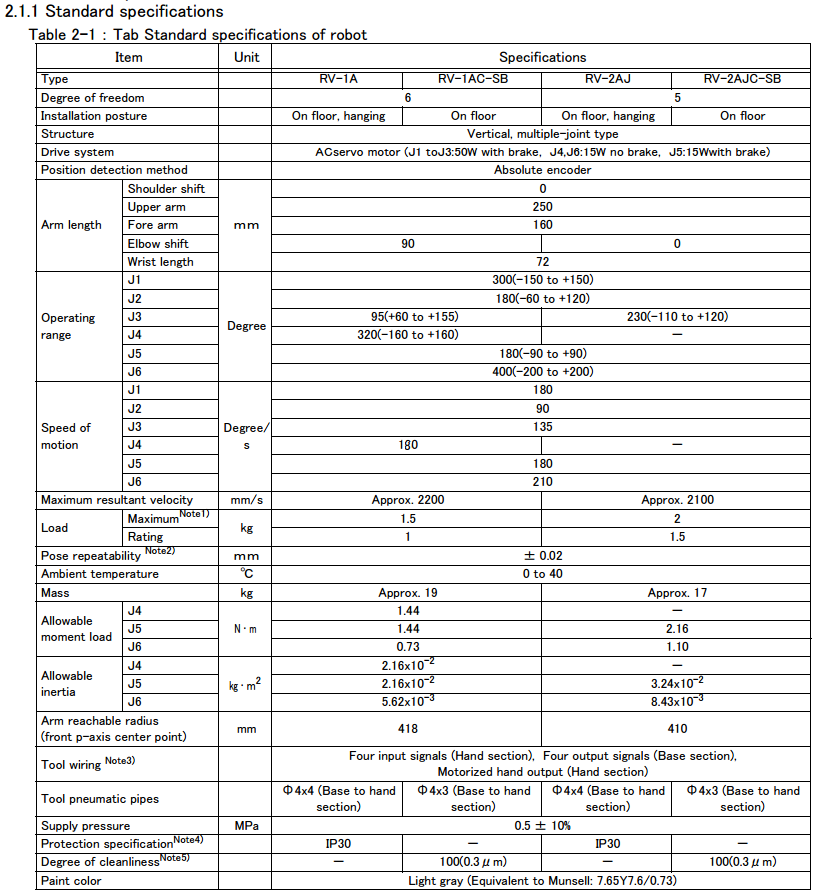
Om vast te stellen of de robotarm kan tafeltennissen zijn er diverse eigenschappen van de robot onderzocht. De onderzochte eigenschappen zijn: de snelheid, de kracht, de houding en het bereik. Deze zijn geselecteerd, omdat ze het gedrag van de robot het meest beïnvloeden voor het spelen van tafeltennis.

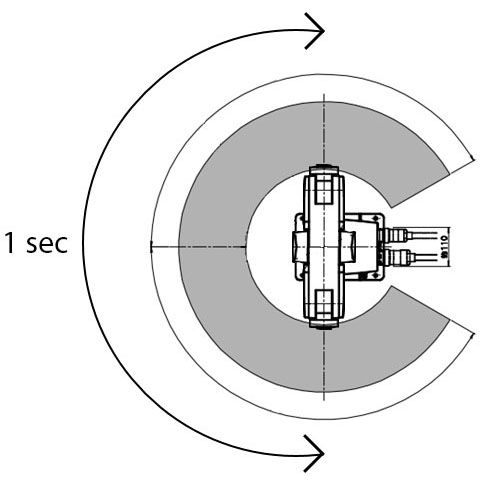
## De snelheid

De snelheid van de robot is zeer belangrijk om vast te kunnen stellen of de robot kan tafeltennissen. Wanneer het positioneren van de robotarm te lang duurt, is het balletje al van de tafel.

De robotarm heeft niet één maximale snelheid, maar een maximale snelheid per scharnier. Elk scharnier bevat een andere motor en heeft een andere maximale draaisnelheid.

Tabel 2 Speed of motion (Mitsubishi Electric Corporation, 2002)





Figuur Draaisnelheid van J1 (Mitsubishi Electric Corporation, 2002)

In de tabel, Tabel 2, staan in de kolom ‘Item’ de namen van de scharnieren en in de kolom ‘Specifications’ de maximale draaisnelheid in graden per seconde.

Uit deze gegevens kan geconcludeerd worden dat het scharnier ‘J1’ minimaal één seconde nodig heeft om van de linkerkant naar de rechterkant te draaien. Dit scharnier bevindt zich op het onderste draaipunt van de robot en bepaalt de kijkrichting.

Door middel van een berekening kan de maximale snelheid van het tafeltennisbatje berekend worden. Hiervoor zijn de maximale rotatiesnelheid van J6 nodig en de lengte van het batje ten opzichte van het middelpunt van J6. De rotatiesnelheid is 210 deg/s en de lengte van het batje tot j6 in 0,47 meter. Deze gegevens komen voort uit het onderzoek ‘De montage van het batje’.

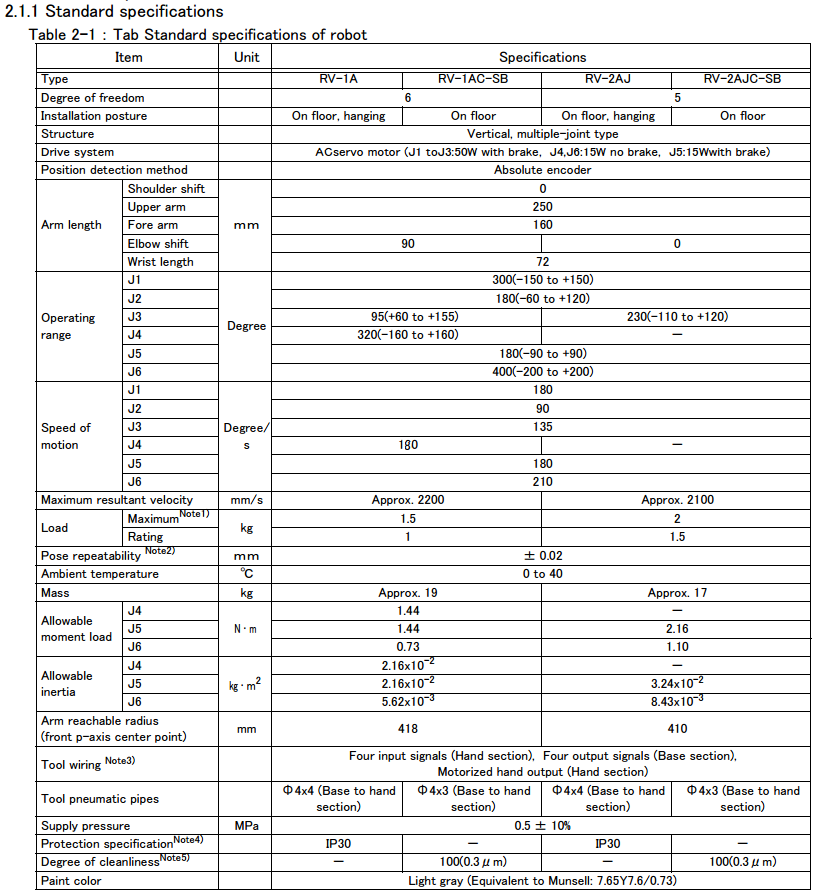
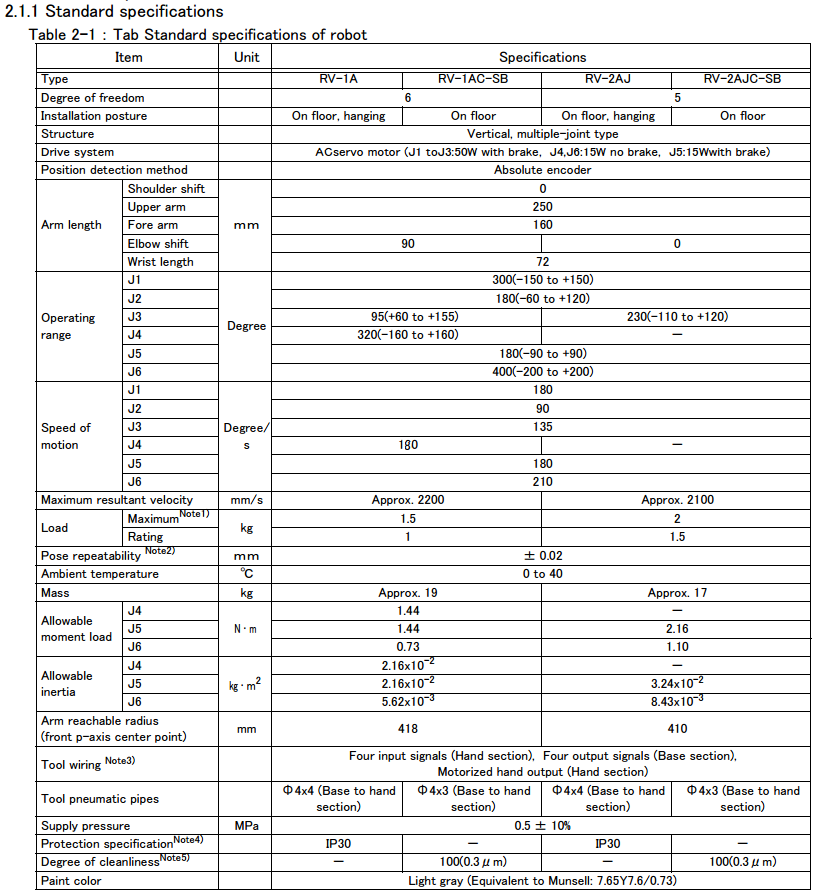
De formule die hiervoor wordt gebruikt is:

Het batje krijgt dus een snelheid van 1,72m/s

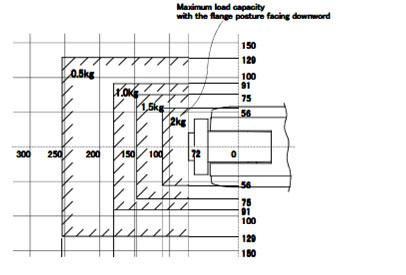
## De kracht

Om de robotarm een balletje te laten slaan heeft de robot een bepaalde kracht nodig. Aan het uiteinde (J6) zal een batje worden bevestigd en op dat punt moet de kracht voldoende zijn om een balletje te kunnen slaan. De krachten van de verschillende motoren in de scharnieren staan in Tabel 3.

Tabel 3 Allowable moment load (Mitsubishi Electric Corporation, 2002)



Het scharnier waaraan het batje bevestigd zal worden is J6 en deze heeft een kracht van 1,10 newtonmeter.

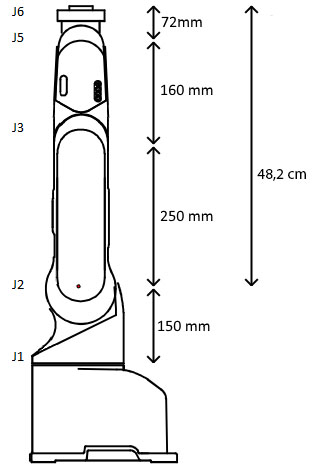


Figuur Maximum draagkracht in kg per afstand in mm (Mitsubishi Electric Corporation, 2002)

Het gewicht dat de robot kan tillen is maximaal 2 kilo. Dit is afhankelijk van de afstand tussen het object en de robot. De maximale kracht per afstand is weergegeven in Figuur .

## Draaihoeken

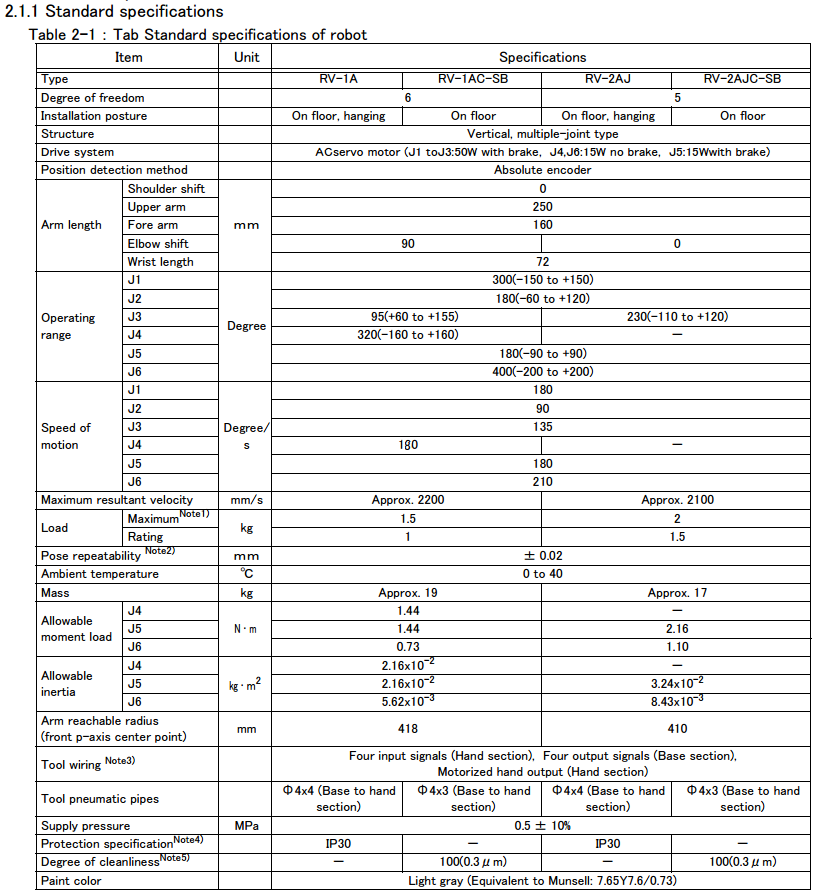
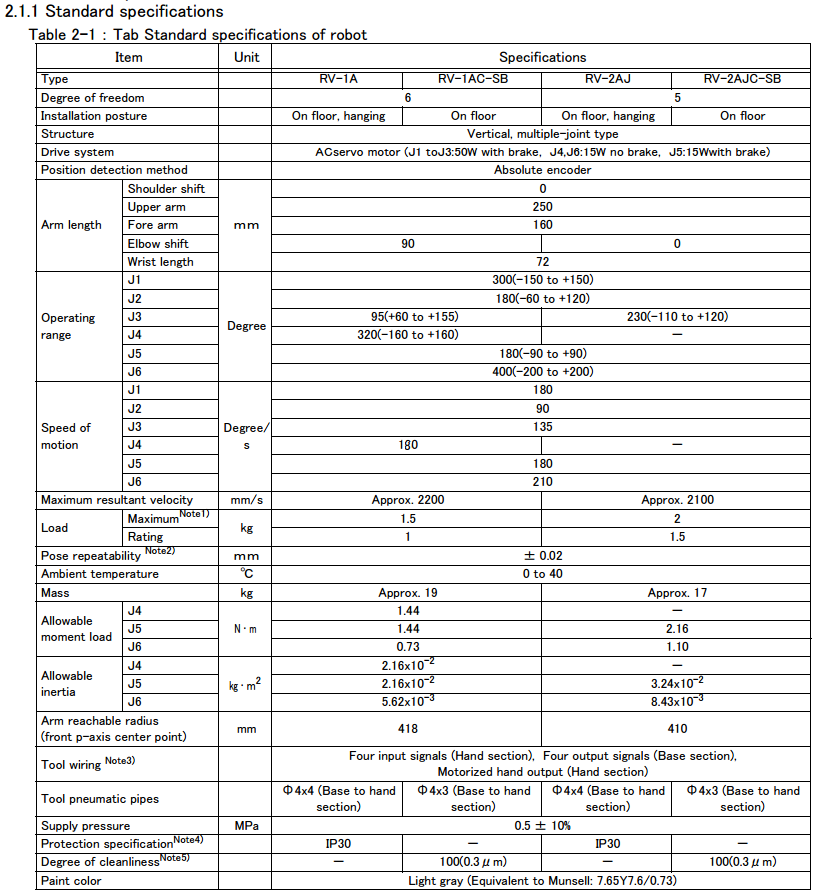
De robotarm heeft 5 assen. In Figuur 6 staat de robotarm op zijn nulpunt. De hoeken van alle motoren staan dan op nul graden. Het is belangrijk om deze waarden goed te zetten, zodat de robot niet over zijn hardware-grenzen heen zal gaan. De grenzen zijn softwarematig vastgelegd.

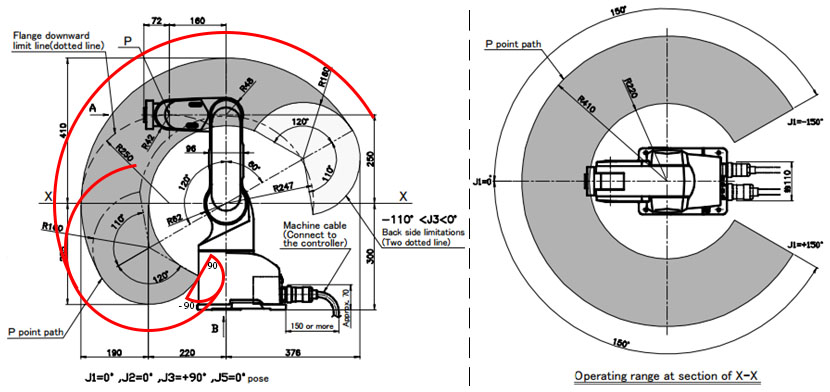


Figuur Het nulpunt

Elke as heeft een andere maximale hoek. In Tabel 4 Operating range staan per as de maximale bereikbare hoeken. Figuur 7 daaronder geeft dit schematisch weer.

Tabel 4 Operating range (Mitsubishi Electric Corporation, 2002)





Figuur 7 Draaihoeken (Mitsubishi Electric Corporation, 2002)

Uit praktisch onderzoek is gebleken dat de robotarm een ingebouwde beveiliging heeft zodat de arm niet voorbij de grenzen van de scharnieren kan gaan. Dit werkt alleen als het nulpunt in ingesteld. Dit onderzoek is uitgevoerd door twee scenario’s te testen, een met ingesteld nulpunt en een zonder ingesteld nulpunt.

De eerste test is uitgevoerd door het nulpunt niet in te stellen. Vervolgens is de robotarm naar een uiterste hoek bewogen. Dit zorgde ervoor dat de controller zichzelf in een ‘error-state’ zette.

De tweede test is uitgevoerd door het nulpunt wel in te stellen. Vervolgens is de robot wederom naar een uiterste hoek bewogen. Dit zorgde ervoor dat de robotarm stopte op de maximale positie, zonder dat de controller zichzelf in een ‘error-state’ zette.

De arm kan zichzelf wel raken binnen de grenzen, maar uit een ander praktisch onderzoek is gebleken dat de robotarm bij een bepaalde tegendruk in een ‘error-state’ terecht komt en dan gaan alle motoren uit. Wanneer er een batje op gemonteerd wordt zullen deze grenzen veranderen. De rode lijnen in figuur 7 geven het gebied van J5 aan. Dit is het uiterste draaipunt en is daarom het belangrijkste scharnier om in de gaten te houden. Dit onderzoek is uitgevoerd door een weerstand voor de robot te plaatsen. Hieruit bleek dat de robotarm uit zichzelf stopte. Hierbij zette de controller zichzelf in een ‘error-state’.

## De montage van het batje

De positie van het batje op de robotarm is erg belangrijk. Dit heeft invloed de houding/positie van de arm en de manier van slaan. Er is ervoor gekozen om twee mogelijkheden te onderzoeken. Het batje kan in het verlengde en haaks worden geplaatst, dit is weergegeven in Figuur 8.



Figuur Mogelijke positie van het batje (Redactie, 2014)

Wanneer het batje in het verlengde wordt geplaatst heeft de arm een natuurlijkere houding, maar zoals weergegeven in de bovenstaande afbeeldingen kan deze niet slaan. Wel kan het batje om zijn eigen as draaien, maar dan kan het balletje niet recht naar voren worden gekaatst. De tweede mogelijkheid is de haakse positie. Dit verkort de lengte van de arm, maar geeft de mogelijkheid om het balletje recht te kaatsen. Tabel 5 biedt een overzicht van de voor- en nadelen per positie. Het batje in de verlengde positie heeft een extra lengte van: 23 – 8,50 = 14,50 cm ten opzichte van het batje in een haakse positie.

Tabel Voor- en nadelen van mogelijke posities

|  |  |  |
| --- | --- | --- |
| Positie | Voordelen | Nadelen |
| In het verlengde van de robot | Heeft 23 cm extra bereik ten opzichte van de arm  De lengte van een batje is 23 cm | Kan alleen om de as draaien |
| Haaks op de robot | Kan een slagbeweging maken |  |
|  | Heeft 8,50 cm extra bereik ten opzichte van de arm  De breedte van een batje is 14 cm  De breedte van het handvat is 3 cm.  ((14 - 3) / 2) + 3 = 8,50 cm |  |

# Veiligheid

|  |
| --- |
| Behandelde deelvragen:   1. Hoe kan de veiligheid worden gegarandeerd?    1. Wat wordt er onder veilig verstaan? |

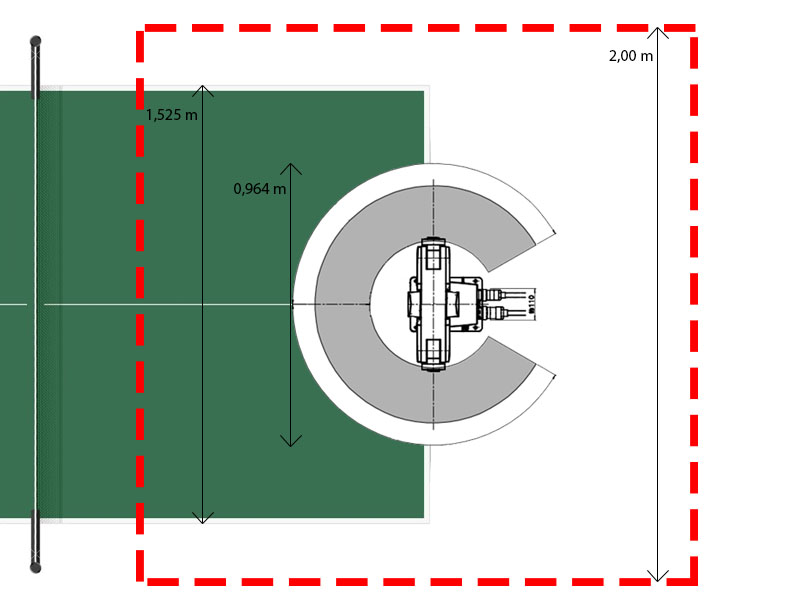
In technische systemen staat veiligheid voorop en daarom wordt in dit onderzoek uitgebreid gekeken naar de handhaving van de veiligheid.

## Wat is veiligheid?

Wanneer er wordt gesproken over veiligheid in dit project wordt er bedoeld dat er geen mensen en objecten worden beschadigd.

## Hoe wordt de veiligheid gehandhaafd?

Om de veiligheid te handhaven zal er een veiligheidszone worden ingericht, zodat personen niet een contact kunnen komen met de robot. Dit omvat minimaal het maximale bereik van de robotarm inclusief de lengte van het batje. Deze veiligheidszone zal worden aangegeven met een lint, omdat dit duidelijk en mobiel is. Deze grenzen mogen niet worden overschreden door onbevoegden en dat wordt aangegeven met een bord. In Figuur 9 is deze zone schematisch weergegeven.



Figuur Veiligheidszone

Het is op dit moment voor de robot mogelijk om bewegingen te maken waarbij de arm zichzelf kan raken. Om de veiligheid voor de robot te garanderen zullen er grenzen worden vastgesteld voor de draaihoeken van de scharnieren.

Om de veiligheid en de werking van de hardware te garanderen zal er elke dag een ‘Daily inspection’ worden gehouden en elke maand een ‘Periodic inspection’. Voor deze twee inspecties zijn twee lijsten met verschillende onderdelen die worden gecontroleerd. Deze lijsten staan in bijlage Daily inspection items en bijlage Periodic inspection.

# Beeldherkenning

|  |
| --- |
| Behandelde deelvraag:   1. Hoe wordt het tafeltennisballetje gedetecteerd? |

Om het balletje terug te kunnen slaan moet het systeem weten waar het balletje zich bevindt. Hiervoor moet het systeem uit een informatiebron (Camera, Infrarood, sonar, etc.) de nodige informatie kunnen halen en verwerken tot informatie die door het systeem toegepast kan worden, dit heet Object Tracking. Dit proces kan samengevat worden met het volgende diagram dat gebaseerd is op A Survey on Object Detection and Tracking Methods (Himani, Darshak, & Udesang, 2014). Deze stappen zijn de standaard verwerking van beeldmateriaal. Om een object te herkennen moet eerst een beeld worden gemaakt. Vervolgens moet er gekeken worden of er iets gevonden kan worden. Hierna moet het systeem de exacte locatie van het object vinden. Ten slotte mag deze informatie verwerkt worden. Deze onderdelen worden in de paragraaf ‘Beschikbare beeldherkenning technieken/middelen’ verder toegelicht.

Figuur Cyclus beeldherkenning

## Nodige eigenschappen voor de beeldherkenning

Om het systeem te correct te laten tafeltennissen zijn een aantal aspecten van de beeldherkenning van toepassing:

### Resistentie tegen verschillen

Tussen de beelden in zullen een aantal verschillen ontstaan. Deze kunnen veroorzaakt worden door beweging op de achtergrond, verschil in lichtsterkte of het vallen van schaduw. Hierdoor zal het mogelijk zijn dat het balletje niet correct wordt gevonden of dat het systeem het balletje op een andere locatie waargenomen. Dit wordt een false positive genoemd.

### Uitvoertijd

Doordat het systeem maar een beperkte tijd heeft om te reageren zal het balletje snel getraceerd moeten worden. Wanneer de beelden niet snel genoeg verwerkt worden zal de robotarm niet op tijd kunnen reageren.

### Accuraatheid

Het systeem moet weten waar het balletje zich bevindt en zal hiermee moeten uitrekenen waar het balletje zich zal bevinden. Echter hoeft dit niet volledig accuraat uitgevoerd te worden. Dit komt doordat de arm zal reageren met een batje waardoor een verschil van enkele centimeters weinig verschil zal maken.

### Herstelmogelijkheid wanneer het balletje buiten beeld valt

Wanneer een speler of de arm een punt scoort zal het balletje van de tafel vallen en waarschijnlijk buiten het beeld terecht komen. Ook kan een van de spelers zich zo positioneren dat zijn batje (of arm) het beeld van de camera blokkeert. In deze situaties moet het systeem het balletje snel terug kunnen vinden om te zorgen dat door gespeeld kan worden.

## Beschikbare beeldherkenning technieken/middelen

Er zijn een aantal hulpmiddelen en algoritmen beschikbaar om te helpen met het vaststellen van de positie van de bal. Hieronder worden deze per categorie weergegeven:

### Opvangen beeldmateriaal / Invoer beeldmateriaal

De eerste stap in het implementeren van Object Tracking is een invoerbron waaruit informatie word toegediend. Dit zijn ruwe beelden die meerdere malen per seconde worden opgenomen. Hiervoor bestaan de volgende technieken:

#### Sonar/Echo

Een manier om het balletje te detecteren is met behulp van Sonar of Echo. Dit houdt in dat een sensor een geluidsignaal verzendt en de tijd meet totdat de sensor dit signaal terug ontvangt. Hiermee kan een afstand worden uitgerekend tot de sensor. Echter is deze apparatuur niet beschikbaar.

#### Infrarood

Een infrarood sensor kijkt naar de warmte die een object uitstraalt. Echter zijn de meeste sensoren gericht en moeten dus al bekend zijn met de locatie van het balletje. Infrarood camera’s registreren echter de warmte van het balletje en de objecten in de omgeving, die beide op kamertemperatuur zullen zijn.

#### Camera met kleur of grijswaarden

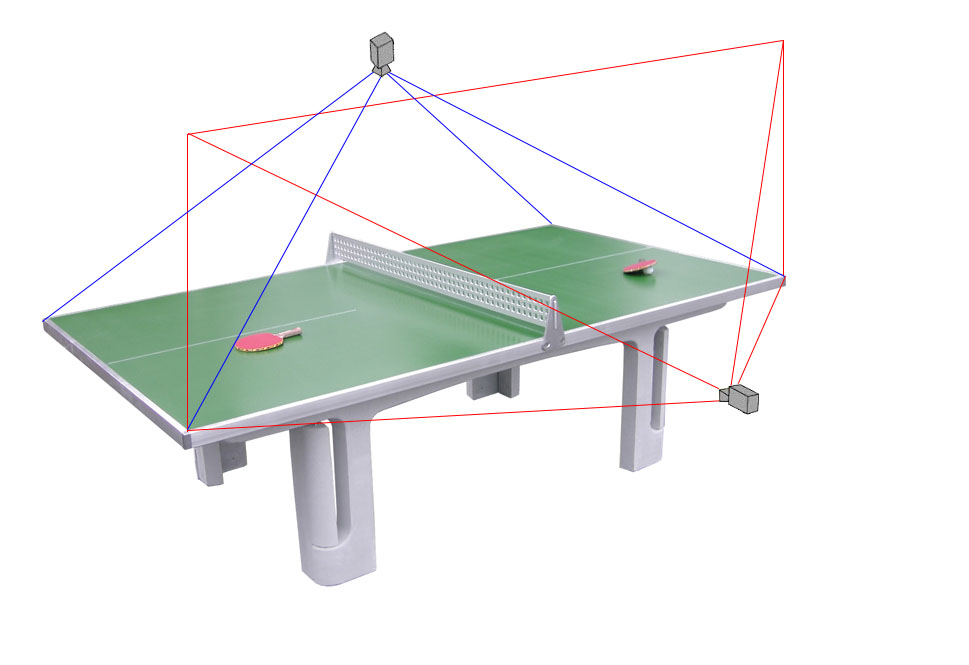
Ten slotte kan worden gewerkt met beelden uit een camera. Iedere camera zal de nodige beelden kunnen produceren, echter zullen camera’s met meer beelden per seconden een accuratere herkenning opleveren doordat het verschil tussen de beelden minder groot is.

#### Vergelijking

Voor het opvangen van het beeldmateriaal moet gelet worden op het feit dat het resultaat in 3D zal moeten zijn. Geen van de beschreven methode zal dit met een enkel meetpunt een 3D locatie kunnen berekenen.

Om dit correct te laten werken met een 3D omgeving zullen 2 meet punten nodig zijn om de locatie vast te stellen.

Een manier om dit te realiseren is een opstelling met een camera recht boven de tafel, waar een breedte en lengte locatie mee gedetecteerd kan worden. Vervolgens is er een zijcamera nodig om de hoogte te registreren. De beelden van de camera’s kunnen op een andere manier verwerkt worden.



Figuur Camera posities

### Detecteren van de mogelijke object posities (Object Detectie)

Vervolgens moet een techniek worden toegepast om een mogelijke positie van het balletje te verkrijgen. Het is mogelijk dat de technieken meerdere mogelijkheden retourneren, dit zal in een volgende stap worden opgelost.

#### Frame Differencing

Met Frame Differencing (Singla, 2014) wordt het huidige beeld vergeleken met een voorgaand beeld. Ieder verschil dat ontstaat is een mogelijke positie van een bewegend object. Doordat deze methode zeer eenvoudig is, is dit ook een vrij snelle methode om mogelijke locaties vast te stellen. Echter kunnen verschillen ontstaan door veranderingen in het licht en mogelijke bewegingen op een achtergrond.

#### Optical Flow

Optical flow processing (The University of Iowa Engineering, 1997) is een algoritme dat ogenschijnlijke beweging van een object waarneemt door iedere wijziging tussen beelden te registreren en te verwerken met een serie formules. Deze methode is in staat beweging zeer gedetailleerd op te vangen, echter is deze methode zeer intensief en niet goed in staat om in variërende omstandigheden te werken.

#### Background substraction

Background substraction (Cheung & Kamath, 2004) verwerkt beelden aan de hand van een vastgestelde achtergrond (template). Vervolgens wordt in iedere frame deze achtergrond uit het beeld gehaald. Hierdoor blijven alleen de bewegende objecten over in het beeld. Echter is deze methode niet resistent tegen verschillen in de omgeving zoals licht en beweging op de achtergrond. Hierdoor zal deze methode een stabiele achtergrond nodig hebben met een vastgestelde lichtinbreng. Ook moet het template gekalibreerd worden aan de positie van de camera.

#### Kleur herkenning

Ten slotte kan de locatie worden geschat op basis van een kleur. Dit houdt in dat ieder deel van het beeld, dat niet aan de kleurvereisten voldoet, genegeerd zal worden. De resterende informatie zal een mogelijke positie zijn van een object. Deze methode vereist ook dat er weinig objecten met een soortgelijke kleur aanwezig zijn in de omgeving. Ook kan een verschil in het lichtniveau een probleem opleveren voor het systeem.

#### Canny Edge Detection

Objecten als een geheel herkennen is een ingewikkeld proces voor een computer. Een manier om het eenvoudiger te maken is het toepassen van Canny Edge Detection (Canny, 1986). Dit algoritme gebruikt een Gaussian Filter (Blur) om eventuele verstoringen en minieme veranderingen in het beeld te verwijderen. Vervolgens wordt de intensiteit van iedere pixel berekend(grijswaarde) en deze worden vergeleken met nabije pixels. Waar de intensiteit sterk verschilt van de aangrenzende pixels kan gesproken worden over een rand. Deze gegevens worden in het zwart-wit getekend op een afbeelding van dezelfde grootte als het origineel. Hierdoor raken de kleuren uit het beeld verloren. Echter is deze methode aanzienlijk minder gevoelig voor ruis en verlichting als de bovenstaande methoden.

#### Vergelijking

De detectie methode is van invloed op de mogelijke stappen die genomen kunnen worden om deze te verwerken. De methoden zijn hieronder in een tabel geplaats. Hierbij is gekeken naar de accuraatheid van de methode, de gevoeligheid voor verstoringen en de complexiteit van de methode.

Tabel Detectie Methoden

|  |  |  |  |
| --- | --- | --- | --- |
| Methode | Accuraatheid | Verstoringsgevoelig | Complexiteit |
| Frame Differencing | Laag (Alle verschillen worden opgevangen, zoals belichting) | Zeer gevoelig (elke verandering is een nieuwe locatie) | Zeer laag (aantal operaties gelijk aan de hoeveelheid pixels) |
| Optical Flow | Zeer Hoog (Iedere detail, waaronder rotatie, wordt waargenomen) | Extreem gevoelig (Iedere verandering is een beweging die meegerekend wordt) | Zeer hoog (Meerdere wiskundige formules per pixel) |
| Background Subtraction | Hoog (Alleen de locatie van het object blijft over) | Zeer gevoelig(elke verandering is een nieuwe locatie) | Laag (Voor elke pixel een controle of het achtergrond is en een mogelijke reset) |
| Kleur Herkenning | Gemiddeld (Objecten met een soort gelijke kleur worden opgevangen) | Gevoelig (Licht kan de kleur van het object buiten bereik duwen) | Zeer laag (aantal operaties gelijk aan de hoeveelheid pixels) |
| Canny Edge Detection | Hoog (Alleen vormen blijven over) | Laag (Door het toepassen van een Gaussian zijn de meeste verstoringen verwerkt) | Hoog (5 operaties zijn benodigd) |

In de methoden die weergegeven zijn is te zien dat het oplossen van de verstoringsgevoeligheid leidt tot een hogere complexiteit. Echter biedt deze uitkomst nog geen garanties. Dit komt doordat een aantal van de hogere tracking algoritmes afhankelijk zijn van deze stap.

### Herkenning van het object (Object Classificatie) en vaststellen positie

Rafael Nieto heeft een overzicht gemaakt van mogelijke Object classificatie technieken die worden gebruikt in zijn Master Thesis (Nieto, 2013) heeft omschreven. Hieronder staat een beknopt overzicht van deze methoden.

#### Template Matching (TM)

Bij Template Matching wordt op een beeld het juiste object opgespoord door deze te vergelijken met een vooraf gesteld beeld (template). Dit wordt gedaan door een convolutie (Berekening van de overlap van twee signalen, of beelden) uit te rekenen en de locatie met de hoogste convolutie waarde is het object dat gevonden dient te worden.

Doordat het algoritme uit weinig stappen bestaat is het eenvoudig toe te passen. Deze methode verwerkt zijn gegevens snel genoeg om in een real-time applicatie te kunnen draaien.

Echter kan het algoritme niet goed tegen transformaties van het doelobject (Vervormingen, rotaties en verandering in formaat). Deze problemen zullen niet veel voorkomen bij de pingpongbal doordat deze te allen tijde rond zullen zijn.

Verder kunnen problemen ontstaan wanneer de kleur van het object afwijkt van de template. Dit kan komen door bijvoorbeeld de belichting van het object. Dit probleem kan echter opgelost worden in de Object Detectie stap.

#### Hough Circle Detection (HCD)

Een methode die van toepassing is voor het tafeltennis programma, die niet omschreven is door Rafael Nieto is de Hough Circle Detection (Rhody, 2005). Het gebruikt een aantal punten om een cirkel vorm in te schatten.

Doordat het algoritme op veel onderdelen in een afbeelding kan reageren. Hierdoor kan het algoritme zeer traag worden. Om dit op te lossen wordt veelal gebruik gemaakt van een Intensiteitsmap (De afbeelding in grijswaarden) of een Edge Detector (Canny Edge Detector). Ook maakt het voor het algoritme een verschil of de straal van de cirkel bekend is. Doordat een pingpongbal in iedere rotatie op een 2D afbeelding een cirkel vorm is, is het mogelijk deze methode te gebruiken.

#### Mean-Shift (MS)

Mean Shift beschrijft een proces waarbij de nieuwe locatie wordt uitgerekend aan de hand van een eerdere positie in combinatie met een herkenningspunt. Dit herkenningspunt kan een template, een kleur combinatie of een ander herkenningspunt zijn.

Het algoritme rekent locaties uit die overeenkomen met het herkenningspunt. Echter is deze herkenning niet zo strikt als bij template matches, waarbij iedere de volledige overeenkomst telt, maar in plaats hiervan zullen alle overeenkomsten gemarkeerd worden. Vervolgens wordt de Epanechnikov Kernel methode (Struijker) toegepast om de overeenkomsten dicht bij de eerdere locatie prioriteit te geven over de anderen overeenkomsten. Vervolgens wordt het algoritme iteratief uitgevoerd over het resultaat totdat de locaties van de objecten samenvallen.

Deze methode kan ook doorberekend worden wanneer het object buiten beeld valt (Obstructie) door met de eerder uitgerekende snelheid en locatie de verwachte locatie van het object aan te passen. Hierdoor kan ieder frame een voorspelling gemaakt worden totdat een nieuwe cluster overeenkomsten in de buurt van de verwachte locatie komt waardoor het object teruggevonden kan worden.

Het algoritme presteert goed wanneer een specifiek kenmerk het object omschrijft. Hierdoor kan het object eenvoudig herkend worden.

Dit algoritme presteert echter minder goed wanneer het object buiten het beeldbereik valt. Dit komt doordat het algoritme zal aannemen dat het object door beweegt met als gevolg dat de voorspelde locatie ver buiten het beeld zal vallen.

#### Particle Filter-based Color Tracking (PFC)

Dit algoritme werkt op een verglijkbare manier als de bovenstaande Mean-Shift methode, echter werkt deze alleen met kleur. Vervolgens wordt hier ook de bovenstaande Epanechnikov Kernel methode (Struijker) toegepast om de overeenkomsten dicht bij de vorige positie een hogere waarde te geven.

Echter gebruikt deze methode het gemiddelde van alle potentiele locaties om de daadwerkelijke locatie te bepalen. Deze vergelijking van locaties wordt herhaaldelijk toegepast totdat de locaties samenvallen op een locatie. Deze uiteindelijke locatie is het resultaat van het algoritme.

Dit algoritme presteert beter als anderen in complexe situaties doordat iedere mogelijke locatie meegenomen wordt in het eind resultaat. Hierdoor is dit een van de meest gebruikte algoritmes voor beeldherkenning.

#### Lucas-Kanade Tracking (LK)

Lucas-Kanade Tracking is een vorm van Optical Flow herkenning (Rojas). Het werkt door de verschillen in grijswaarden (Intensiteit) te meten. Hierbij worden beelden vergeleken en probeert het algoritme in te schatten welke richting het object op beweegt zodat de verschillen in intensiteit verklaard kunnen worden.

Echter gaat het algoritme er van uit dat het verschil in tijd en afgelegde afstand tussen de beelden niet groot is. Dit wil zeggen dat het algoritme gebouwd is voor langzaam bewegende objecten. Hierdoor is deze niet geschikt voor het bijhouden van snelle objecten.

#### Incremental Learning for Robust Visual Tracking (IVT)

Incremental Learning gebruikt een aantal templates om een inschatting te maken van de mogelijke transformaties van een object. Vervolgens gaat het algoritme van ieder nieuw beeld het object registreren in een nieuwe template. Door het beeld te vergelijken met alle templates zal het algoritme eventuele veranderingen kunnen opvangen doordat het algoritme het nieuwe object zal herkennen. Echter kan het algoritme niet goed omgaan met het verlies van het object en zal het algoritme steeds intensiever worden doordat de hoeveelheid templates groter wordt.

#### Tracking Learning Detection (TLD)

De Tracking Learning Detection (Nieto, 2013) is een tracker die zich aanpast aan het beeld dat deze ontvangt. Dit wordt gerealiseerd door informatie van de frames bij te houden en te gebruiken. Deze tracker gaat er hierdoor vanuit dat de beweging tussen de frames klein is (lage snelheid) en dat het object in beeld is. Wanneer het object buiten beeld valt zal de tracker het object niet meer terug kunnen vinden.

#### Corrected Background-Weighted Histogram Tracker (CBWH)

Het doel van de Background weighted histogram tracker is om de Mean-Shift methode te verbeteren door de invloeden vanuit de achtergrond te verminderen. (Yang, et al., 2013) Echter is het algoritme door deze methode minder efficiënt wanneer er kleurverschillen optreden. Om dit probleem op te lossen kan een Kalman Filter toegepast worden.

Deze methode kan de hoeveelheid iteraties van de Mean-Shift methode verkleinen. Hierdoor zal deze variant het object sneller kunnen vinden. Ook zal het een hogere precisie kunnen behalen doordat minder objecten meetellen in de berekening.

Het algoritme presteert goed wanneer er een duidelijk verschil is tussen het doel en de achtergrond. Echter kan het algoritme problemen krijgen wanneer er meerdere soortgelijke objecten bij elkaar aanwezig zijn.

#### Scale and Orientation Adaptive Mean-Shift Tracking (SOAMST)

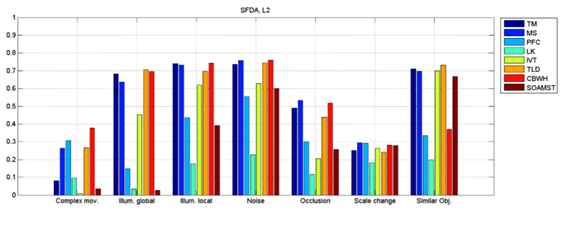
Dit algoritme is wederom een aanpassing op het Mean-Shift systeem (Vojir, Noskova, & Matas, 2013). Hierbij probeert het systeem een schaal van het object in te schatten. Hiermee kan het systeem het object terugvinden wanneer de schaal veranderd. Deze aanpassing presteert beter op beelden waar de schaal van het object veranderd.

Echter creëert het systeem nieuwe problemen bij beelden waarin de schaal niet veranderd. Om dit probleem op te lossen wordt gewerkt met een “backward consistency check” (Vojir, Noskova, & Matas, 2013) die de beelden in de omgekeerde volgorde verwerkt om een meetpunt te verkrijgen voor het nieuwe beeld. Hierdoor kost dit algoritme meer operaties als de originele Mean-Shift methode.

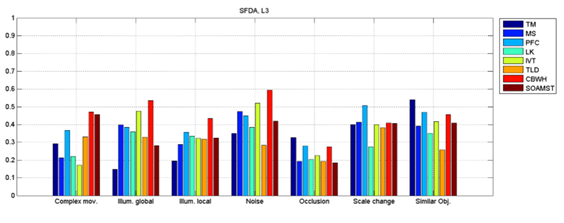
#### Vergelijking

De methode van Herkenning heeft twee belangrijke factoren. Ten eerste dient het object op een juiste locatie herkent te worden. Vervolgens dient dit zeer snel te gebeuren, doordat het systeem nog tijd nodig heeft om een tegenaanval in te plannen.

De accuraatheid van deze tracking mechanismen is uitgerekend door Rafael Martin Nieto (Nieto, 2013). Hieronder zijn twee grafieken (Nieto, 2013)) zichtbaar voor de accuraatheid van de tracking mechanismen in twee situaties. De eerste grafiek laat de resultaten zien voor een simpele simulatie (Laboratorium set) van een beweging. De tweede grafiek geeft de resultaten weer voor een serie van complexere sequenties (Simpele Realiteit).



Figuur Tracking nauwkeurigheid voor het Laboratorium Set (Nieto, 2013)



Figuur Tracking nauwkeurigheid voor Simpele Realiteit (Nieto, 2013)

In de grafieken 12 en 13 is de accuraatheid van de tracking algoritme berekend volgens hoofdstuk 3.3 van Rafael Martin Nieto. Hierin staat vermeld dat een hogere score inhoud dat het algoritme slechter presteert. Echter heeft Rafael Martin Nieto hierin niet gekeken naar de tijdsduur van de uitvoeringen. Ook zijn deze methoden zonder verdere bewerking op het beeldmateriaal uitgevoerd.

Deze vergelijking heeft een aantal video sequenties vergleken:

1. Complex Movements: Het doel object wisselt snel van richting en snelheid.
2. Global Illumination: De lichtomstandigheden in de omgeving veranderen (zoals bewolking).
3. Local Illumination: De lokale lichtomstandigheden veranderen, zoals wanneer het object door een schaduw beweegt.
4. Noise: Willekeurige variaties in het beeld.
5. Occlusion: Het object raakt buiten beeld voor een periode.
6. Scale change: het weergegeven object wordt groter of kleiner.
7. Similiar Object: Een object met een soortgelijke kleur of vorm verschijnt in de buurt van het object.

Vervolgens moet gekeken worden naar de snelheid van de trackers. Echter zijn er weinig externe onderzoeken verricht die deze tracking methoden vergelijken. Deze zullen later gemeten moeten worden.

Echter zijn de bovenstaande resultaten te koppelen aan de andere benodigde eigenschappen. De meeste van deze methoden gebruik maken van Frame Differencing (Behalve TM). Echter is het mogelijk een aantal van dezen te combineren met Background Substraction (BS) en Canny Edge Detection(CED). Door te combineren met Canny Edge Detection kan de verstoring van licht en andere verstoringen verbeterd worden, echter zal hiermee alle kleur verloren gaan. Door te combineren met Background Substraction, zal alleen het object dat afwijkt van de achtergrond overblijven, echter is deze methode gevoelig voor verstoring. De combinatie van deze gegevens is opgenomen in de onderstaande tabel.

Tabel Beeldherkenning methoden

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algoritme | Licht-gevoeligheid | Storings-gevoeligheid | Herstelbaarheid | BS | CED | Overige |
| TM | Hoog | Hoog | Zeer herstelbaar | ✔ | ✔ | Is alleen een controle |
| HCD | Hoog | Hoog | Zeer herstelbaar | ✔ | ✔ | Afhankelijk van voorstap en radius |
| MS | Hoog | Hoog | Herstelbaar | ✔ | X | Meer mogelijkheden is meer iteraties |
| PFC | Laag | Gemiddeld | Herstelbaar | ✔ | X | Meer mogelijkheden is meer iteraties |
| LK | Laag | Gemiddeld | Zeer herstelbaar | ✔ | X | Heeft een klein verschil in tijd en afstand nodig |
| IVT | Gemiddeld | Laag | Niet herstelbaar | ✔ | X | Gedurende tijd wordt het algoritme zwaarder |
| TLD | Gemiddeld | Laag | Niet herstelbaar | ✔ | X | Heeft een klein verschil in tijd en afstand nodig |
| CBWH(MS) | Hoog | Zeer Hoog | Herstelbaar | ✔ | X | Minder iteraties als MS |
| SOAMST | Gemiddeld | Gemiddeld | Herstelbaar | ✔ | X | MS voor schaal veranderingen |

# De programmeertaal

|  |
| --- |
| Behandelde deelvraag:   1. Welke programmeertaal is het meest geschikt om de robotarm te programmeren? |

Het kiezen van de juiste programmeertaal voor het project is een belangrijke stap. De taal op zich is het fundament van de applicatie. Wanneer de verkeerde taal gebruikt wordt, kan achteraf blijken dat het systeem niet of slecht zal werken. Ook kan het gebruik van de juiste programmeertaal veel tijd besparen.

## Real-time

Bij Technische Informatica is real-time een belangrijk begrip. Het principe real-time binnen de scope van het project betekent dat de robot gegarandeerd reageert binnen een gestelde tijd. Dit is belangrijk bij het halen van deadlines in tijd kritische applicaties. Bijvoorbeeld bij het terugslaan van een balletje, dit moet binnen de gestelde tijd gebeuren.

### Besturingssysteem

Wanneer er gebruik wordt gemaakt van een normaal besturingssysteem kan het zijn dat het programma door de scheduler of achtergrondprocessen vertraagd of onderbroken wordt. In het artikel “Windows Internals” wordt beschreven dat in een niet real-time besturingssysteem apparaat-interrupts niet op een controleerbare manier geprioriteerd worden (Russinovich, Solomon, & Jonescu, 2009). Hierdoor kunnen user-level applicaties tijdelijk vertraagd worden.

Een real-time besturingssysteem (verder RTOS genoemd) kan zorgen voor een gegarandeerde uitvoertijd van een programma. Hierbij wordt een eigen real-time kernel in een aangepaste hardware abstractie laag gezet. Doordat het Windows gedeelte enkel als userinterface draait en dus een lagere prioriteit heeft dan het gedeelte dat het apparaat aanstuurt.

### Programmeertaal

In de technische achtergrond van een programmeertaal kunnen ook elementen zitten die ervoor kunnen zorgen dat een programmeertaal wel of niet real-time is. De Java garbage collector kan er bijvoorbeeld voor zorgen dat het programma vertraging oploopt. In bijvoorbeeld de aansturing van een airbag kan dit levensbedreigend zijn.

## Communicatie

In hoofdstuk 8.1 staat beschreven dat de communicatie via de COM poort verloopt. Het is dus belangrijk dat de programmeertaal communicatie via een COM poort ondersteunt.

## Objecttracking

Voor het project is het nodig om objecten te kunnen traceren door middel van video-interpretatie. Het is belangrijk dat de taal hiervoor een mogelijkheid ondersteunt. Ook is het wenselijk dat het traceren van een object ‘snel’ gaat.

## User base/support

Het is erg wenselijk dat de programmeertaal een grote gebruikersbasis heeft en mogelijkheid tot support of help van een community.

## De programmeertaal

Met oog op de verschillende eisen kunnen de programmeertalen vergeleken worden. Hiermee kan worden bepaald welke programmeertaal geschikt is voor onze opdracht. Dit is uitgewerkt in de tabel hieronder.

Tabel Eigenschappen programmeertalen (Cass, 2015) (Tiobe Software) (tecosystems, 2015)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taal | Real-time (hard) | Besturingssystemen | Communicatie | Object tracking | Userbase (0 tot 5) |
| C | ✔ | Indien compiler beschikbaar | ✔ | ✔ | 4 |
| C++ | ✔ | Indien compiler beschikbaar | ✔ | ✔ | 3 |
| Java | X | Draait enkel op JVM | ✔ | ✔ | 5 |
| Real Time Java | ✔ | Een aantal geselecteerde | ✔ | X | 1 |
| C# | X | Draait op .NET | ✔ | ✔ | 4 |
| Python | X | Een aantal geselecteerde | ✔ | ✔ | 3 |

### Vergelijking programmeertalen

In Tabel 8 staan diverse programmeertalen en deze worden in de onderstaande hoofdstukken toegelicht.

#### Real-time

Bij real-time wordt onderscheid gemaakt tussen “hard” en “soft” real-time. Bij hard real-time is het halen van een deadline cruciaal. Mocht de deadline niet gehaald worden dan zal dit leiden tot schade. Bij soft real-time is het wenselijk dat het systeem bijvoorbeeld 99% van de tijd op tijd is. Daarbij is het 1x missen van een deadline niet gevaarlijk maar vaak niet wenselijk. Bij soft real-time zou een Windows besturingssysteem voldoende moeten zijn.

C/C++ is hard real-time (wanneer uitgevoerd op een RTOS). Dit wil zeggen dat de timing deterministisch/berekenbaar is. De code wordt gecompileerd naar machinetaal die direct uitgevoerd kan worden. Er is geen garbage collector of virtual machine die het programma pauzeert.

Java/C#(.NET)/Python is per definitie niet real-time (Reynolds, 2010). De code draait op een virtual machine, deze zorgt voor run-time optimalisatie en garbage collection. Door het run-time optimaliseren is er niets of weinig te zeggen over hoe de code wordt uitgevoerd en daarnaast kan de garbage collection het programma vertragen (Anderson, 2014).

#### Besturingssysteem

Voor C/C++ geldt dat elke architectuur waarvoor een compiler beschikbaar is in principe gebruikt kan worden. Bij Java/C# kan elk besturingssysteem gebruikt worden mits daar de virtual machine op kan draaien. Bij een RTOS wordt C/C++ meestal ondersteund aangezien deze ook real-time gedrag kan implementeren. Ook is het in sommige gevallen mogelijk om Java/C#/Python te gebruiken en zal dit de tijdsafwijking verminderen tegenover een niet real-time besturingssysteem.

In het artikel “Linux configureren voor real-time operaties” is beschreven hoe Linux kan worden afgestemd voor (soft) real-time gedrag. Ook wordt hierin beschreven dat de x86 structuur al interne service management interrupts (SMI) heeft. Deze kunnen niet gezien of bewerkt worden door het besturingssysteem.

Linux configureren voor real-time operaties (Kernel.org, 2014):

* Verminder SMI interrupts (PS/2 muis/keyboard (geen USB, uitzetten in BIOS))
* Compileer een ACPI-enabled Kernel
* Zet TCO timer generation van SMI’s uit
* Verminder DMA bus apparaten (SATA/PATA/SCSI, netwerk adapters, HDD en GPU)
* Zet energiebeheer op maximale prestaties
* Zet hyper-threading uit
* Zet CPU schaling uit
* Zet CPU sleep uit
* Zet VGA-console uit
* Verhoog de proces-prioriteit en affiniteit

In het bovenstaand genoemd artikel wordt dus beschreven dat SMI (noodzakelijke interrupts van de processor) niet uitgeschakeld kunnen worden. Ook wordt beschreven dat de uitvoertijden hiervoor ver onder de 1ms liggen en dus vaak niet eens voor hard real-time iets uitmaakt. Er zijn dus real-time besturingssystemen., waar bijvoorbeeld de linux kernel voor gebruikt kan worden.

#### C++ in combinatie met C of assembly

C++ compilers kunnen ook C compileren. Er zijn echter kleine, vaak syntactische verschillen (Allain). Ook kan in C++ inline assembly gebruikt worden (jain.pk, 2006). Echter kan inline assembly ervoor zorgen dat de code niet werkt op een ander platform aangezien assembly geen high-level programmeertaal is. Toch kan men bij het gebruik van een C++ compiler dus effectief ook programmeren in C of assembly.

# Communicatie

|  |
| --- |
| Behandelde deelvraag:   1. Welke programmeertaal is het meest geschikt om de robotarm te programmeren? |

De robot bestaat uit twee onderdelen: de controller en de arm. De controller verzorgt de aansturing van de arm. Voor het aansturen van de controller bestaat de mogelijkheid om deze aan te sturen met de RS-232C aansluiting aan de voorzijde (Mitsubishi Electric, 2005). Het onderzoek naar communicatie is uitgevoerd door informatie uit de handleiding in de praktijk uit te proberen en de bevindingen op te schrijven.



Figuur Schematische weergave aansluiting robot

## Communicatiemethoden

Volgens de uitgebreid specificaties van de controller (Mitsubishi Electric, 2005) kan er op twee manieren via de seriële RS-232C verbinding gecommuniceerd worden:

* COSIMIR commando’s sturen
* Communiceren met de zelf ontwikkelde software op de robotarm

Het communiceren met zelf ontwikkelde software functioneert ook via COSIMIR commando’s

De directe commando’s ondersteunen meerdere robots op één communicatie kanaal. Daarom moet bij het geven van een commando het robotnummer aangegeven worden. Voor multitasking bevat de robot verschillende sloten waar taken worden uitgevoerd. Deze sloten zullen dan parallel worden uitgevoerd. Hierdoor moet ook worden aangegeven met welk slot gecommuniceerd moet worden.

Het volledige commando komt er afhankelijk van of er één of meerdere parameters nodig zijn als volgt uit te zien:

* “Robotnr;slotnr;commando=(parameters)”
* “Robotnr;slotnr;commando parameter”

### Activeren van de RS-232C poort en configuratie van de software

Voor de communicatie via de RS-232C poort dient de poort te worden geconfigureerd. Dit kan door middel van de Teaching Pendant of via de COSIMIR software. Voor de COSIMIR software is het een vereiste dat de poort al is geactiveerd. Welke parameters dit zijn, waarop ze ingesteld moeten zijn en wat het betekend is te vinden in de Tabel 9 Overzicht van de in te stellen parameters (Mitsubishi Electric, 2005). Om met deze communicatie methode de robot te laten bewegen moet de sleutelschakelaar op de controller op auto(ext) staan.

Tabel Overzicht van de in te stellen parameters (Mitsubishi Electric, 2005)

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Standaard waarde | Nieuwe waarde | Betekenis |
| COMDEV | RS232, , , , , , , | RS232, OPT11, , , , , , | Zet het communicatie kanaal open voor invoer van buiten. De lege ruimte is om extra invoer van de controller te activeren. OPT11 is om op slot 1 de COM poort uit te lezen. |
| CBAU232 | 9600 | 115200 | Baudrate van de COM poort |
| CPRTY232 | 2 | 2 | Pariteit van de verbinding (even) |
| CSTOP232 | 2 | 2 | Aantal stop bits |
| CTERM232 | 0(CR) | 0(CR) | Terminator voor commando’s |
| CPRC232 | 0 | 2 | Communicatie op datalink zetten zodat er in de software gebruik gemaakt kan worden van de COM poort |

### Commando’s sturen over RS-232C

Het sturen van commando’s is geen officieel ondersteunde methode om te communiceren met de controller. Melfa beschrijft dan ook een beperkt aantal commando’s in hoofdstuk 6.3 van de uit gebreide specificaties van de controller (Mitsubishi Electric, 2005). Hier staan geen andere commando’s beschreven dan jog commando’s wat neerkomt op iedere keer een kleine bewegingen in plaats van naar een bepaalde hoek of coördinaat. In de COSIMIR software is dit wel mogelijk. Door te luisteren met seriële poort monitor software op de communicatie tussen de robot en de COSIMIR zijn extra commando’s ontdekt. De gevonden commando’s zijn in de uitgebreide handleiding (Mitsubishi Electric, 2002) te vinden. Om deze commando’s uit te kunnen voeren dient het COSIMIR programma op één van de sloten van de controller geactiveerd zijn. Wanneer dit programma niet aanwezig is op de robot kan dit met de COSIMIR software worden toegevoegd.

Tabel Beschrijving van niet gedocumenteerde commando's

|  |  |
| --- | --- |
| Commando | Effect |
| OPEN=usertool | Verbinding openen |
| CNTLON | Control starten |
| SRVON | Servo aan |
| EXECJOVRD 100.0 | Snelheid instellen |
| EXECJCOSIROP=(-90.00,-60.00,-30.00,-30.00,-30.00,-30.00) | Joint Coördinaten instellen |
| EXECMOV JCOSIROP | Arm bewegen coördinaat |
| JPOSF | Joint posities |
| STATE | Huidige toestand van de arm |
| PPOSF | Huidige coördinaten |
| EXECPCOSIROP=(70.00,0.00,782.00,0.00,0.00,0.00)(6,0) | XYZ Coördinaten instellen |
| EXECMOV PCOSIROP | Arm bewegen naar coördinaat |
| CNTLOFF | Control uit |
| SRVOFF | Servo uit |
| RSTALRM | Alarm resetten |
| EXECSPD 200.0 | Uitvoersnelheid instellen |
| EXECMVS PCOSIROP | Jog arm beweging |
| PARRLNG | Onbekend |

### Software op de robot

Naast het sturen van commando’s via RS-232C is het ook mogelijk om software op de controller aan te sturen zoals beschreven staat in de uitgebreide handleiding (Mitsubishi Electric, 2005). Voor het programmeren van de software op de controller zijn twee programmeertalen mogelijk:

Melfa basic IV of Movemaster-command.

Waar deze talen zich in onderscheiden is geen onderzoek voor te vinden met zoekwoorden “melfa basic vs movemaster command”, “movemaster command” en “Melfa programming” op Google en Bing. In de documentatie van de controller wordt wel Movemaster-command toegelicht maar wordt niet gesproken over voordelen of nadelen van specifieke talen. Hierdoor wordt aangenomen dat de talen voornamelijk verschillen in syntax. In de documentatie worden de voorbeelden veelal in Melfa Basic gegeven.

Om de keuze voor de programmeertaal in te stellen is er de parameter RLING nodig. Wanneer deze parameter op 1 staat kan Melfa-basic 4 gebruikt worden en wanneer deze op 0 staat kan Movemaster-command gebruikt worden. Wanneer er gecommuniceerd gaat worden met behulp van software op de robot moet er in de software een COM poort worden geopend, te zien op regel 10 in het voorbeeld hieronder. Daarna wordt er data ontvangen via RS-232C en weer terug gestuurd.

**Command Comment**

10 OPEN "COM2:" AS #1 'Opens RS-232C communication.

20 INPUT #1,C1$ 'Inputs data.

30 PRINT #1,C1$ 'Outputs data.

40 GOTO 20 ‘Back to data input

Om via de seriële verbinden de data bij de software te krijgen is het volgende commando nodig:

“PRN data”. Dit commando kan op twee manieren worden aangeroepen en dit staat in Tabel 11.

Tabel Commandos

|  |  |  |
| --- | --- | --- |
| Commando | Return | Effect |
| “PRN 1,(0,0,0,0,0,0,0)\r” | QoK (0,0,0,0,0,0) | Beweegt de robot naar de opgegeven positie. De doelpositie wordt meegegeven bij de return. |
| “PRN 2,(0,0,0,0,0,0,0)\r” | QoK (0,0,0,0,0,0) | Beweegt de robot relatief naar de opgegeven positie. De doel positie wordt meegegeven bij de return |

## Vergelijking

Om het verschil tussen de twee communicatie methoden duidelijk te maken is in de tabel hieronder een vergelijking te zien tussen de twee methoden op twee manieren.

Tabel Communicatie methoden

|  |  |  |
| --- | --- | --- |
|  | COSIMIR commando’s | Communicatie met zelf ontwikkelde software |
| Leercurve | Behalve een aantal gegeven commando’s die uitgevoerd moeten worden hoeft er niets extra’s geleerd worden | Er moet een extra programmeertaal, Movemaster-command of Melfa Basic geleerd worden. |
| Reactiesnelheid | Het is niet bekend hoe de robot het commando in het COSIMIR programma uitvoert en daardoor kan er ook geen uitspraak gedaan worden over de reactiesnelheid. | Omdat de commando’s op de robot met zelf geschreven software wordt uitgevoerd kan er meer gezegd worden over de volgorde waarop opdrachten worden uitgevoerd. Echter zijn er nog steeds veel onbekenden, maar minder als bij gebruik van de COSIMIR commando’s |
| Parallel uitvoeren van taken | Alle taken worden hoogst waarschijnlijk Sequentieel uitgevoerd omdat het programma maar in één slot draait maar de bron is niet bekend. | Er is een mogelijkheid om taken parallel uit te voeren in meerdere sloten. Echter wordt er maar één regel code tegelijk uitgevoerd waardoor de uitvoertijd niet sneller zal zijn (Mitsubishi Electric, 2005) . |
| Communicatie | Altijd posities verzenden dus voor een slag beweging zullen meerdere positie commando’s gestuurd moeten worden. Daarnaast zijn moet er voor het veranderen van de positie twee commando’s gestuurd worden. | Er kunnen specifieke eigen commando’s verstuurd hierdoor hoeven niet alle locaties worden verstuurd en kan een slag beweging mogelijk met een enkel commando verstuurd worden. |
| Debugging/Error afhandeling | Alle fouten afhandelen op de pc en er kan geen data gepushed worden voor debugging alleen via polling is dit mogelijk. | In de software kunnen fouten worden afgehandeld en er kan ook data gepushed worden naar de server. |

## Coördinaten of hoeken?

Er zijn twee mogelijkheden om de robot aan te sturen. De robot kan aan de hand van een coördinaat zelf naar dat coördinaat bewegen. Voor dit project is er een batje aan de robotarm gemonteerd en daar heeft de robot geen kennis van. Ook is een groot nadeel dat de robot precies op een coördinaat wil komen. Wanneer een coördinaat niet precies op de millimeter bereikt kan worden zal hij hier niet naartoe gaan.

Daarom is ervoor gekozen om handmatig de hoeken te bereken en die hoeken naar de robot te sturen. In deze berekening wordt wél rekening gehouden met het batje. Daarnaast wordt er binnen een bereik van 5 cm gezocht naar dit punt, omdat dit het aantal coördinaten dat bereikt kan zeer gehoogd. Het batje heeft een breedte van 14 cm en dit is daarom geen probleem voor het raken van het balletje. De berekening staat in het vervolg “Traject bepalen”.

# Tussentijdse conclusies

In dit hoofdstuk is te lezen welke conclusies er getrokken zijn naar aanleiding van het theoretisch onderzoek. Dit wordt vervolgd met bevindingen uit de praktijk die zijn onderzocht door middel van het praktische deel van het onderzoek. Bij niet alle onderdelen kan een enkele beste keuze gemaakt worden, daarom staat niet overal één beste keuze maar zijn er dan meerdere keuzes.

## Programmeertaal

Er kan geconcludeerd worden dat C++ de beste keuze voor het programmeren van een aansturingsprogramma voor de robot is. Dit doordat het, afhankelijk van het besturingssysteem, real-time is. Daarmee kan C++ op veel real-time besturingssystemen werken. Ook worden alle benodigdheden ondersteund en heeft C++ een uitgebreide user base.

C++ kan worden gecompileerd naar machine instructies, in principe kan bij C/C++ bepaald worden wat de maximale executietijd van een blok code is (aantal instructies\*tijd per instructie). Daarnaast is het atomisch uitvoeren van code, afhankelijk van besturingssysteem, mogelijk. Men weet dan zeker dat twee instructies elkaar volgen. Het nadeel is dat het besturingssysteem zijn taken moet afwisselen. Hierdoor zal er dus in de praktijk een vertraging ontstaan, dit wordt ook wel “jitter” genoemd. Real-time Java was in ons onderzoek een andere kanshebber. RT-Java is wel een interessante mogelijkheid, maar het is onbekend of de videoherkenning libraries ermee compatibel zijn. Daarnaast is de ondersteuning van RT-java door real-time besturingssystemen kleiner dan die van C++.

Keuze voor C++:

* Compatibiliteit met C en assembly
* Real-time
* Draait niet op een Virtual Machine
* Benodigde randvoorwaarden behaald COM-poort ondersteuning en videoherkenning libraries
* Goede user base

## Besturingssysteem

Tijdens het vooronderzoek is geconcludeerd dat een besturingssysteem, bijvoorbeeld Windows, jitter kan introduceren. Het probleem hierbij is dat Windows op de achtergrond taken uitvoert en de performance van het systeem kunnen benadelen.

De mogelijkheid om Linux als besturingssysteem te gebruiken is ook onderzocht. Daarnaast kan er voor gekozen worden om een RTOS te gebruiken. Deze zouden een hard real-time implementatie mogelijk moeten maken ondanks dat hardware interrupts en BIOS instructies niet aangepast kunnen worden door het besturingssysteem (Kernel.org, 2014).

## Beeldherkenning

Uit de resultaten van het theoretische onderzoek naar de beschikbare methoden zijn een tweetal methoden gekomen die voldoen.

De eerste methode is Particle Filter-based Color Tracking. Deze is geschikt doordat de methode een hoge resistentie heeft tegen veranderingen in het beeld. Deze methode maakt gebruik van Frame Differencing om het verschil te zien tussen locaties op basis van een Mean Shift methode.

De tweede methode is Hough Circle Detection. Doordat de bal vanuit alle oogpunten rond zal zijn kan de bal gevonden worden door de methode. Ook kan de methode filtreren op basis van de doorsnee van het object waarmee valse positieven uitgesloten kunnen worden. Deze methode heeft geen Detectie methode nodig, maar presteert beter met behulp van een Edge Detection Methode.

Beide methoden zouden uitgebreid kunnen worden met Background subtraction. Echter zal dit zeer afhankelijk zijn van de plaatsing en verlichting van de omgeving.

## Eigenschappen

### Het bereik

Er is onderzocht dat de robot een minimaal bereik van 1,525 m in de breedte moet hebben. Dit komt overeen met de breedte van de tafeltennistafel. In de praktijk blijkt dat de bal niet altijd recht over de tafel wordt gespeeld, maar dat deze schuin wordt gespeeld en dus naast de tafel kan belanden. De robot heeft dus een groter bereik nodig om alle ballen terug te kunnen slaan.

De robot heeft een bereik van 48,2 cm vanaf het middelpunt en dus een totaal bereik van 96,4 cm. Hier komt nog de lengte van het batje bij, maar dat is afhankelijk van de montage van het batje.

Uit deze gegevens blijkt dat het bereik van de robot niet groot genoeg is. Tijdens dit onderzoek zal dat probleem vermeden worden door het balletje recht over de tafel te slaan. Dit wordt dus ook een randvoorwaarde.

### De montage van het batje

Er zijn twee manieren onderzocht voor het monteren van het batje op de robotarm, namelijk in het verlengde of haaks op de robot. Deze verschillen in het bereik en in de slagbeweging. Uit het onderzoek is gebleken dat het voordeliger is om het batje haaks op de robot te monteren. Dit heeft een kleiner bereik, maar kan makkelijker een slagbeweging kunnen maken. Om het bereik te vergroten is een mogelijkheid om het batje te verlengen. Dit zal in de praktijk worden onderzocht.

### De snelheid

In een professioneel tafeltennisspel wordt het balletje met een snelheid van 35 m/s geslagen. Dat betekent dat het balletje in 0,078 seconden de lengte van de tafel aflegt. De onderste as van de robot kan draaien met 180 graden/seconde. Er kan dus geconcludeerd worden dat het balletje maximaal in 1 seconden heen en weer mag en dan mag het balletje dus maximaal met een snelheid van 5,48 m/s worden geslagen. Echter zal de robot waarschijnlijk te langzaam zijn.

## Veiligheid

De veiligheid is erg belangrijk en daarom is er onderzocht hoe dit het beste gehandhaafd kan worden. Daar is uitgekomen dat er een veiligheidszone moet worden ingericht die niet mag worden overschreden door onbevoegden.

Ook is er gebleken dat de robotarm zichzelf kan raken. Wanneer er een batje aan vast zit is dit nog sneller het geval. Er moet dus is het programma een veilige zone worden vastgesteld waarin de robot zich kan bewegen, zonder dat de arm zichzelf kan raken.

Om de veiligheid van de hardware te garanderen zal er elke dag een ‘Daily inspection’ worden gehouden en elke maand een ‘Periodic inspection’.

## Communicatie

In de vergelijking, uit paragraaf 8.2, is te zien dat behalve op het punt leercurve het realiseren van een eigen programma voor de controller de betere keuze is. Doordat er minder commando’s moeten worden gestuurd kan er al performance winst geboekt worden ten opzichte van het gebruik van COSIMIR commando’s. Voor de realisatie van de tafeltennis functionaliteit is het van belang dat de reactiesnelheid optimaal is. Doordat het zelf geschreven programma op de robot data kan verzenden zonder dat hier specifiek om gevraagd wordt kan er voor zorgen dat er sneller gereageerd kan worden op fouten in de robot door de desktop software.

De keuze voor een programmeer taal om software voor de controller te programmeren gaat tussen Melfa Basic IV en Movemaster-command. De verschillen tussen deze twee talen worden nergens toegelicht en lijken zich dus enkel te onderscheiden in syntax. Echter wordt in de documentatie bij voorbeeld code overal Melfa Basic gebruikt, om die reden is de keuze voor Melfa Basic IV het meest logisch. Door de diverse voorbeelden is het ook makkelijker om de taal te leren.

# Praktijk onderzoek

Om verder te controleren of onze bevindingen kloppen en een duidelijker beeld te krijgen is een applicatie gerealiseerd voor het tafeltennissen met de robotarm. Deze applicatie is uitgevoerd in C++. Hierin zijn de delen uitgewerkt, getest en gemeten om tot de volgende resultaten te komen.

## Beeldherkenning

Om het balletje te lokaliseren zijn methoden opgesteld om deze taken uit te voeren. Echter zijn deze afhankelijk van hardware (camera’s) die gebruikt zijn voor de herkenning. In dit hoofdstuk wordt dieper ingegaan op het gebruik van de camera’s en welke punten van belang zijn voor de camera’s. Vervolgens worden de geïmplementeerde beeldherkenningsmethoden besproken.

### Camera’s

Tijdens het project zijn er drie camera’s vergeleken. In de hoofdstukken hieronder zijn de conclusies hiervan verwerkt. De onderzochte camera’s zijn diverse webcams, omdat deze via USB aangesloten en live uitgelezen kunnen worden.

Er zijn tests ondernomen om een GoPro Hero 4 te gebruiken, maar directe videodoorgave via USB was niet mogelijk. Het is wel mogelijk doormiddel van WiFi, maar deze methode levert een grote vertraging op. Daarnaast zou een capture-card gebruikt kunnen worden. Echter was hier niet direct beschikking over en dit zou mogelijk tot kwaliteit/snelheidsverlies leiden.

Ook is er gekeken naar de Genius WideCam F100, aangezien deze een grotere kijkhoek heeft. Echter waren verschillende reviews op het internet erg negatief over de beeldkwaliteit. Het hebben van een grote kijkhoek zal namelijk geen voordeel zijn (naast compactheid van opstelling) als de beeldkwaliteit slecht is. Aangezien de totale informatie per pixel (keurkwaliteit/scherpte) dan niet verbeterd en het balletje dus nauwelijks zichtbaar is.

De camera’s die van tevoren ter beschikking waren zijn de Trust 17676 en Logitech C615 webcams. Echter bleken de kijkhoek en beeldkwaliteit van deze camera’s slecht te zijn. De Trust 17676 heeft maar 2 megapixels, waardoor een full-hd frame/afbeelding (1920x1080) onscherp is. Daarnaast is het contrast van de camera’s slecht, het verschil in huidskleur en fel oranje is onduidelijk op de beelden. Wat tot problemen leidt met beeldherkenning. De kijkhoek zorgt ervoor dat de camera’s op een grote afstand geplaatst moeten worden om de hele tafeltennistafel er op te kunnen krijgen. Daarbij zijn de camera’s niet gemaakt voor grote afstand. Het vervangen van de camera’s met nieuwe Logitech C920 web- / conferencecams leide tot een sterke verbetering in deze onderdelen.

### Beeld/camera kwaliteit

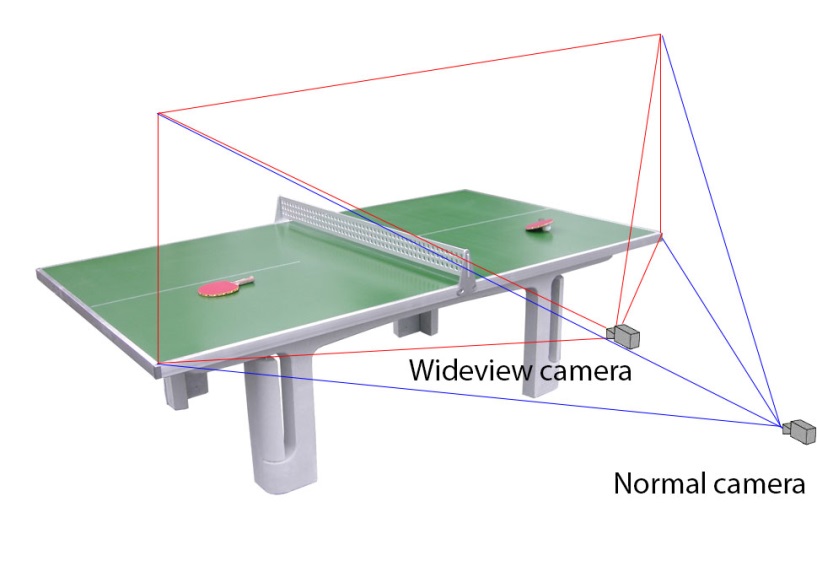
De beeldkwaliteit van de camera’s is erg belangrijk. Aangezien de tafel 2,74m breed is, kan het zijn dat de camera ver weg moet staan om de tafel in het geheel in beeld te krijgen (zie 10.1.4). Daarnaast is in vergelijking tot de tafel het tafeltennisballetje een klein object. Hierdoor is het belangrijk dat de camera een scherp beeld heeft (lagere sluitertijd). In het geval van objecttracking doormiddel van kleurherkenning, is het daarnaast erg belangrijk dat het contrast van de camera erg goed is.

### Handmatige instellingen

Het is belangrijk dat de instellingen van de webcam handmatig gedaan kunnen worden. Een camera zonder mogelijkheden tot het instellen van contrast etc. kan niet afgesteld worden op het omgevingslicht. Het in software aanpassen hiervan zou niet nodig moeten zijn.

### Kijkhoek

Uit ons onderzoek blijkt dat een normale webcam niet bruikbaar is voor een compacte opstelling. Een gemiddelde webcam voor thuisgebruik is ontwikkeld voor een object op korte afstand. Daarbij is de kijkhoek van de webcam minimaal. De geteste camera’s “Logitech C615” en “Trust 17676-02” bleken een kijkhoek van ongeveer 40° te hebben. Volgens de berekeningen moeten de camera’s met een kijkhoek van 40°, 3,8m van de tafel staan om deze in het geheel te kunnen overzien (bij een tafel-breedte van 2,74m). Een camera met brede kijkhoek is dus wenselijk. Zo kan deze dichterbij staan maar toch de gehele tafel overzien. Een camera met een kijkhoek van 108° hoeft maar 1m van een tafel af te staan om deze volledig te kunnen overzien.



Figuur Camera mogelijke kijkhoeken

### Focusafstand

De punten uit de voorgaande tellen ook voor de focusafstand van de camera. Webcams worden vaak op een monitor geplaatst, de focusafstand is daarbij vaak ook standaard vast ±40cm. Het beeld op een afstand van de berekende 3,8m zou dus totaal vervaagd zijn. Des te meer reden om een wide-view- / conferencecam te nemen. Met een aanpasbare focusafstand of een die voldoet aan de eisen qua focusafstand.

### Beeldherkenningsmethoden

In het theoretisch onderzoek is gekozen om de Particle Filter en de Hough Circle Detection te implementeren.

Om deze te implementeren moesten de nodige functies aanwezig zijn. Om de functionaliteiten voor de beeldherkenning te implementeren is gekozen om gebruik te maken van de OpenCV (<http://opencv.org/>) library. Deze library heeft de functionaliteiten om beelden van de camera op te vangen, te bewerken en weer te geven. Daarbij is deze geoptimaliseerd om de snelheid te halen en werkt deze met de gekozen programmeertaal.

Echter zijn er niet twee maar vier beeldherkenningsmethode gemaakt. Deze zijn in te delen in de kleur en de vorm herkenning.

De kleur herkenningsmethoden zijn:

* Color Filter
* Particle Filter

De vorm herkenningsmethoden zijn:

* Hough Circle detection op Canny Edge Detection
* Hough Circle detection op Kleur filter

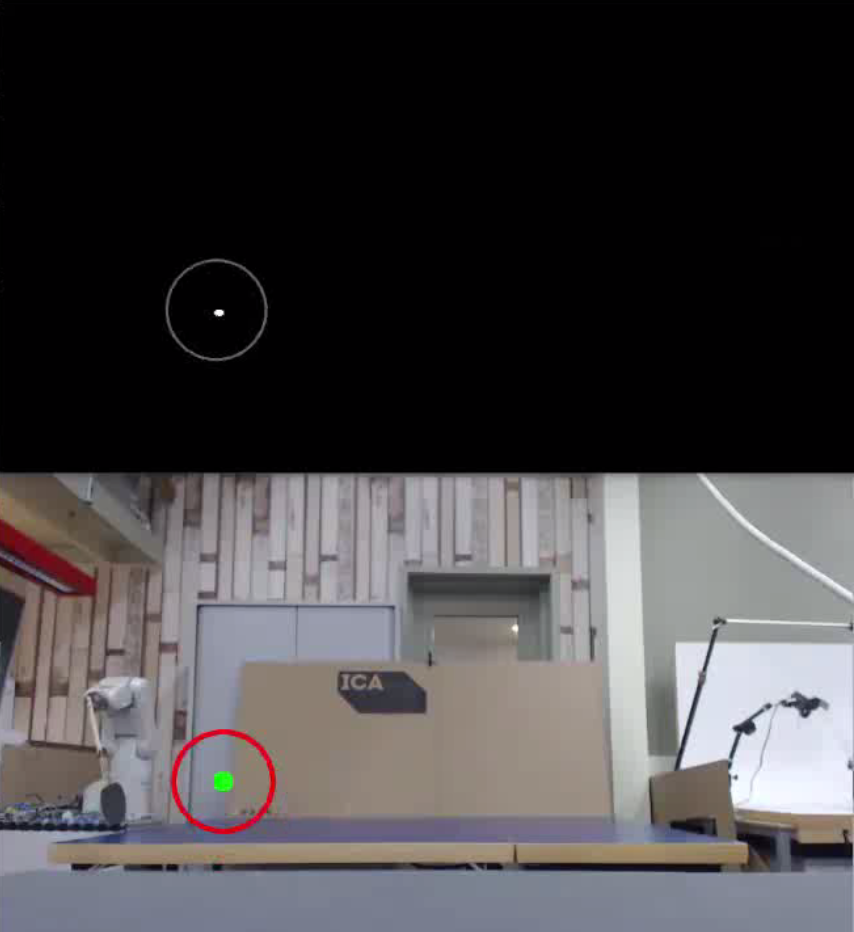
Deze zullen hieronder besproken worden, waarna een vergelijking volgt.

Voor de balletjes is gekozen voor de Oranje pingpong balletjes. Deze zijn gekozen doordat de kleur beter opvalt voor de kleuren filters en het een van de toegestane kleuren is voor een tafeltenniswedstrijd.

#### Color Filter

De Color Filter is ontstaan als de meest eenvoudige methode om een balletje te vinden op een kleur. De Methode is geïmplementeerd in de “PFCVision.hpp” en “PFCVision.cpp” als de klasse “ColorFilter”. Deze methode werkt zeer snel doordat het een kleurenfilter toepast op het beeld om vervolgens de eerste pixel die voldoet aan deze kleur als locatie te selecteren.

Hierdoor is dit de meest eenvoudige methode die geïmplementeerd is, echter kan deze niet omgaan met afleiding.



Figuur ColorFilter

In Figuur 16 is te zien dat de kleur goed gelokaliseerd wordt. Echter gaat deze methode duidelijk uit van de meest links boven pixel. Dit is een risico voor de afleiding.

#### Particle Filter

De Particle Filter methode is een complexere methode die een gemiddelde berekend met gewicht. Ook deze methode is geïmplementeerd in de “PFCVision.hpp” en “PFCVision.cpp” als de klasse “ColorFilter”. Net zoals de Color Filter gebruikt deze methode een kleuren filter om de pixels van toepassing te vinden. Echter is er geen gebruik gemaakt van de Epanechnikov Kernel zoals was gedefinieerd in het eerdere hoofdstuk, maar is deze vervangen door een eenvoudigere methode.

Per gevonden pixel wordt een gewicht berekend aan de hand van de Manhattan distance (Improved Outcomes Software, 2016). Deze methode voorkomt dat een wortel uitgerekend moet worden voor een Euclidean distance. Voor deze afstand wordt de voorgaande gevonden locatie vergleken met de locatie van de afstand.

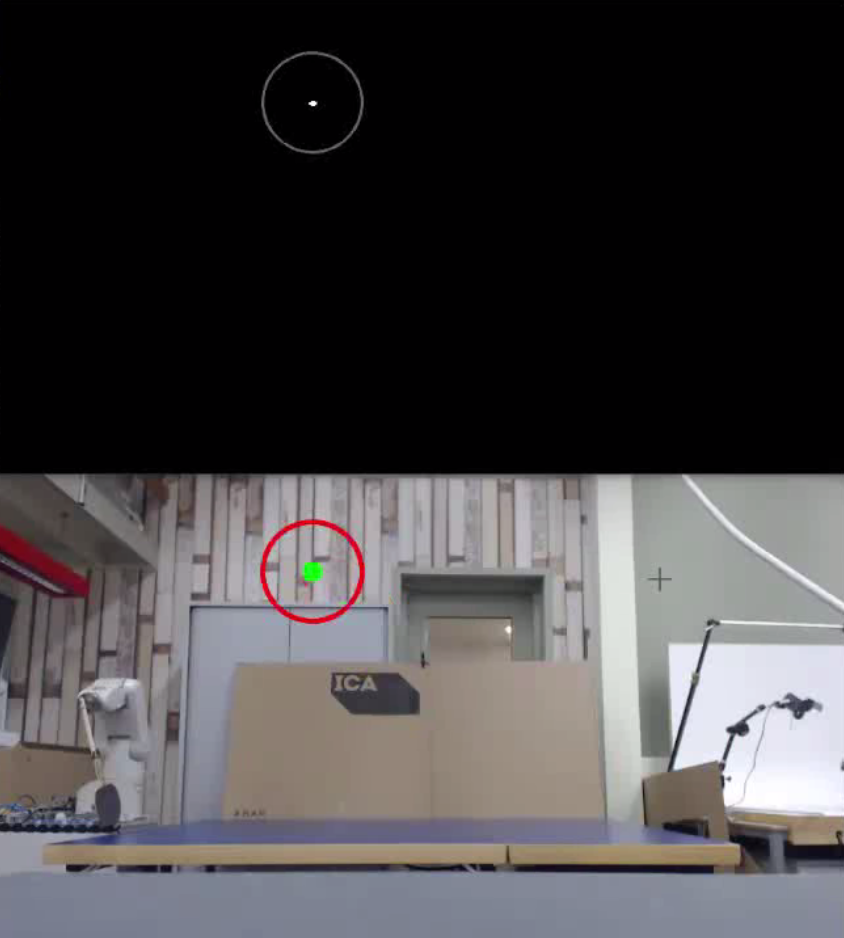
Vervolgens wordt de volgende formule uitgevoerd om het gewicht te berekenen:

G = Wf / (D \* Wd) + Ws

Hierin is Wf het gewicht van het middenpunt, Wd het afvallen van het gewicht over de afstand, Ws het standaard gewicht en D de Manhattan distance tot de pixel.

De uitkomst (G) is het gewicht van de pixel. Een hoger gewicht betekend een hogere invloed op het resultaat.

Hierdoor zullen resultaten dichterbij de vorige positie voorrang krijgen. Voor een beweging is dit ideaal doordat het object vanuit de vorige locatie verder zal beweging. Hierdoor zal het verschijnen van een ander object met een soortgelijke kleur minder invloed hebben op het resultaat.



Figuur Particle Filter

In Figuur 17 is te zien dat het middelpunt van de gevonden pixels wordt genomen.

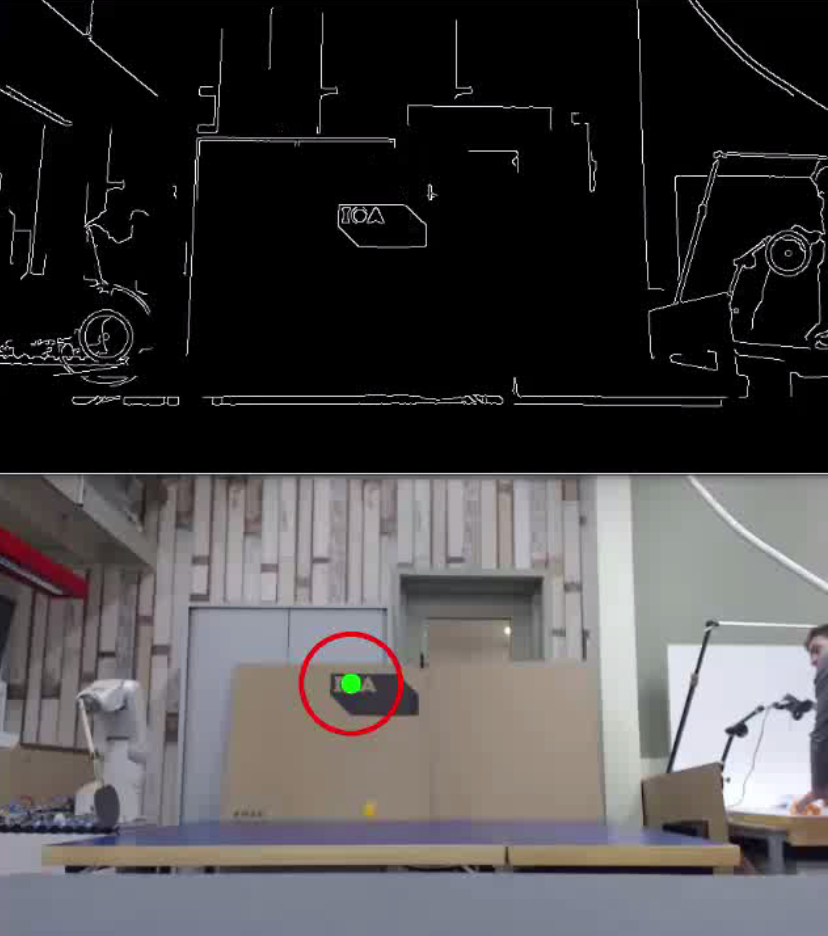
#### Hough Circle detection met Canny Edge Detection

De Hough Circle detection is geïmplementeerd in de “HCDVision.hpp” en “HCDVision.cpp” als de klasse “CannyHCDVision”.

De Hough Circle detectie is de enige methode in deze lijst die geen gebruik maakt van een kleuren filter. De implementatie gebruikt een Gaussian blur om kleine verschillen te verminderen. Vervolgens wordt de afbeelding omgezet naar grijswaarden om het verschil in intensiteit zichtbaar te maken. Hierop wordt de Canny Edge Detection toegepast om de randen te zien. Ten slotte wordt met de Hough Circle Detection toegepast.

Hierin kan afgesteld worden op de radius van de bal om valse waarden buiten te sluiten. Ook kan een accumulator waarden worden afgesteld. Deze accumulator bepaald de minimale score die een cirkel moet halen om geselecteerd te worden. Bij een lagere accumulator zullen meer cirkels gevonden worden. Echter zal bij een zeer hoge waarden geen cirkel meer gevonden worden.

Deze methode is minder afhankelijk van licht en kleur, echter bleek deze methode meer moeite te hebben met het vinden van bewegende objecten. Ook is in sommige omstandigheden het contrast niet voldoende waardoor de randen niet goed gezien worden.



Figuur Canny Edge Detection

In Figuur 18 is te zien dat het beeld niet correct wordt verwerkt. Door de omstandigheden in de omgeving worden meerdere cirkels gevonden. Echter wordt het balletje niet gevonden doordat het contrast verschil met de achtergrond niet genoeg is. Hierdoor is de methode zeer gevoelig voor fouten en omstandigheden.

#### Hought Circle detection op Kleur Filter

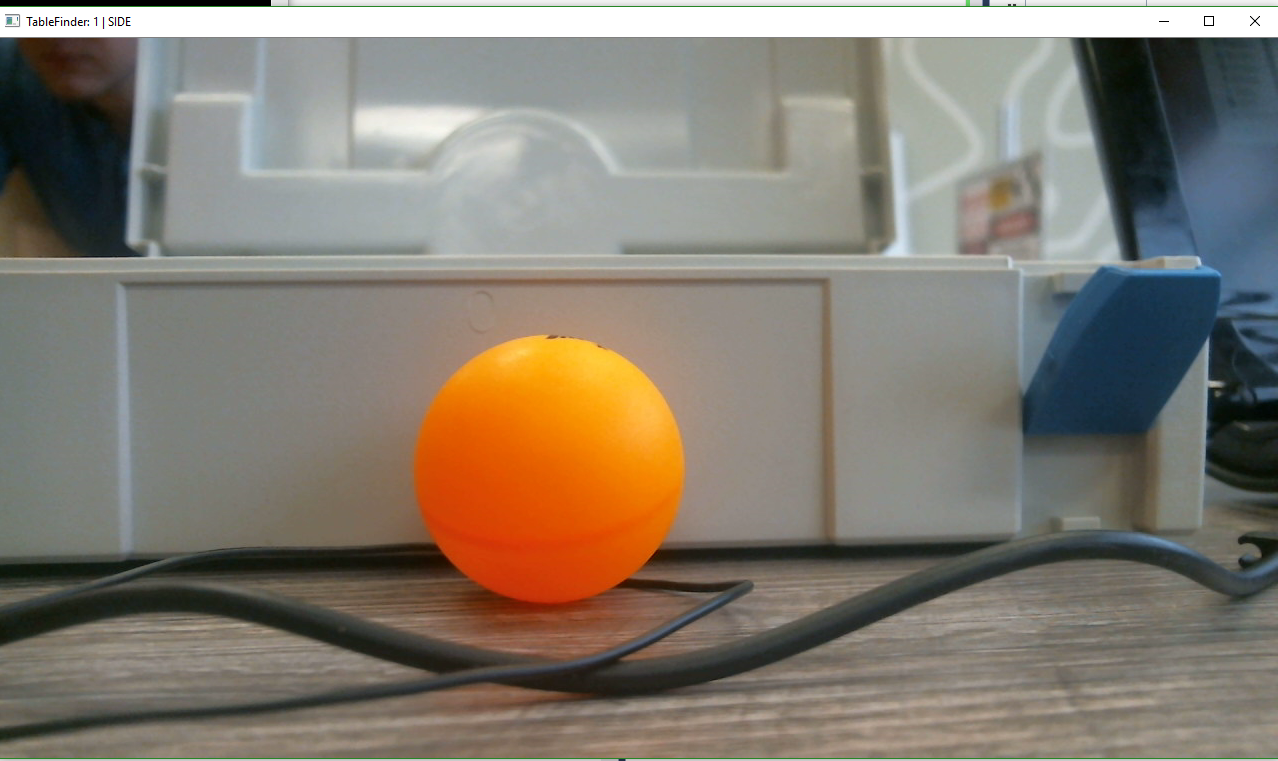
De Hough Circle detection met kleur filter is geïmplementeerd in de “HCDVision.hpp” en “HCDVision.cpp” als de klasse “ColorHCDVision”.

Om het contrast probleem van de normale Circle detection op te vangen is een vorm van Canny Edge Detection gemaakt die een kleuren filter toepast. Hierdoor zullen de kleuren en contrasten verloren gaan, maar de randen van de gevonden kleuren zullen als grenzen gezien worden. Hierdoor zal het minder goed om kunnen gaan wanneer de bal overlapt met een object van dezelfde kleur. Daarentegen past het de combinatie van kleur en vorm toe waardoor er een beter criterium is voor de bal.

Wanneer de methode correct is afgesteld zal deze vrij goed werken. Echter zal de cirkel verloren raken wanneer er een oranje object achter het balletje zichtbaar is.

#### Vergelijking van de snelheid

Om de snelheid van de algoritmen vast te stellen zijn een aantal tests uitgevoerd om de snelheid van de algoritmen vast te stellen. In deze tests is een tijdmeting gemaakt van de totale uitvoer tijd van een vastgesteld aantal pogingen. Aan de hand van deze resultaten kan uitgerekend worden hoeveel tijd er nodig is om het resultaat van een beeld te ontvangen. De resultaten van deze meting zullen afhankelijk zijn van de omstandigheden. De tests zijn uitgevoerd op een laptop met een 2.4Ghz processor (de test gebruikt alleen de hoofdthread waardoor de hoeveelheid cores weinig invloed zal hebben). Ook kan de afstelling van de gebruikte klassen en libraries invloed hebben op de uitvoer. De testopstelling van deze test staat in Figuur 19.



Figuur Testopstelling

##### 500 Wisselende Beelden

In deze meting is voor iedere uitvoering een nieuw beeld opgevangen met de camera (Inbegrepen in de tijd). Er is 500 keer een beeld opgehaald en verwerkt door de gemeten methode. De gebruikte camera heeft een Framerate (Beelden per seconden) van 30 FPS. Het duurt 0.03333333 seconden voordat de camera een nieuw beeld heeft. Voor 500 beelden is dit ongeveer 16,66666666666667 seconden voordat de camera alle beelden ontvangen heeft.

Daarbij is gekeken hoe frequent de methode een resultaat teruggaf.

Tenslotte heeft het object tijdens de proef licht bewogen door stimulatie met een koordje op de onderzijde van de bal. Hierdoor is de bal in het beeld gebleven. In Tabel 13 staan de resultaten.

Tabel 500 wisselende beelden

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Tijdsduur | Tijd per Beeld | Gevonden | Percentage |
| CannyHCD | 42,49303 | 0,084986064 | 497 | 99,4 |
| ColorHCD | 41,33407 | 0,082668134 | 500 | 100 |
| PFCVision | 16,73527 | 0,03347054 | 500 | 100 |
| ColorFilter | 16,8129 | 0,033625806 | 500 | 100 |

De getoonde namen zijn de namen van de geïmplementeerde klassen. De weergegeven tijden zijn in seconden. Hierin is te zien dat de Hough Circle Detection methode aanzienlijk meer tijd nodig heeft dan de kleuren filters. Ook lijken de kleuren detectie methode nauwelijks meer tijd nodig te hebben dan het kost om een nieuw beeld te ontvangen.

##### 1000 Gelijke Beelden

Om het effect van de camera te verminderen is een test uitgevoegd door de methoden meerdere malen uit te voeren op hetzelfde beeld. Hierdoor zal de invloed van de Camera verminderd worden.

Voor deze proef zijn de methoden 1000 maal uitgevoerd. Hierbij is niet meer gekeken naar het resultaat, maar naar de snelheid alleen. De resultaten staan in Tabel 14.

Tabel 1000 wisselende beelden

|  |  |  |
| --- | --- | --- |
|  | Tijdsduur | Tijd per Beeld |
| CannyHCD | 82,923496 | 0,082923496 |
| ColorHCD | 81,698315 | 0,081698315 |
| PFCVision | 34,568851 | 0,034568851 |
| ColorFilter | 25,411043 | 0,025411043 |

De getoonde namen zijn de namen van de geïmplementeerde klassen. De weergegeven tijden zijn in seconden. Ook in deze tests presteren de kleuren filters aanzienlijk beter dan de Hough Circle Detection methoden.

##### Vergelijking van resultaten

Om de invloed van de camera in perspectief te zien zijn de resultaten van de tests samengevoegd, dit staat in Figuur 20.

Figuur Tijdsmeting

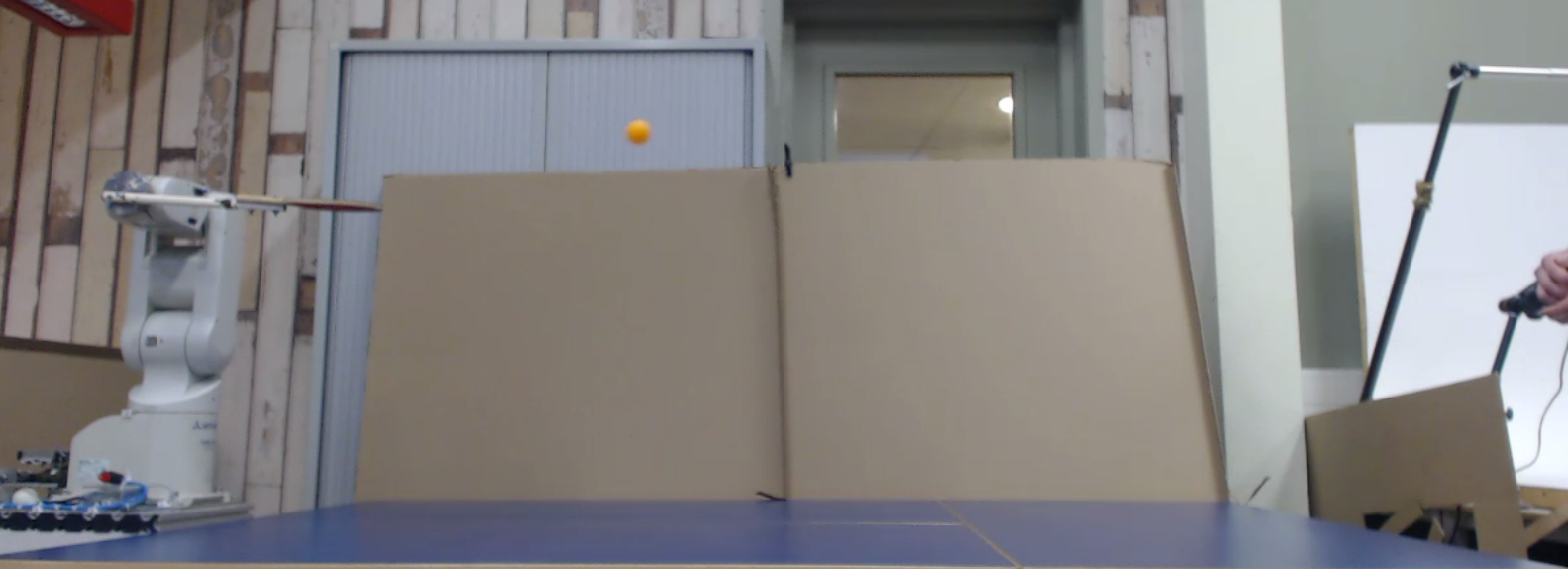
Hierbij is de tijd in seconden weergegeven in de hoogte. In deze grafiek is te zien dat de kleur gebaseerde detectie methoden aanzienlijk sneller presteren dan de Hough Circle Detection methoden. Ook is te zien dat invloed van de camera op de Hough Circle Detection laag is. De ColorFilter kan sneller presteren dan de Camera waardoor deze meer beelden kan verwerken dan de camera kan geven.

De resultaten van de Particle Filter wijken af van de verwachting. Dit is waarschijnlijk het gevolg van een verschil in omstandigheden. Echter lijkt deze methode ongeveer even snel te zijn als de camera waarmee getest is.

Om de optimale snelheid te behalen is een van de kleuren filter methoden de betere keuze.

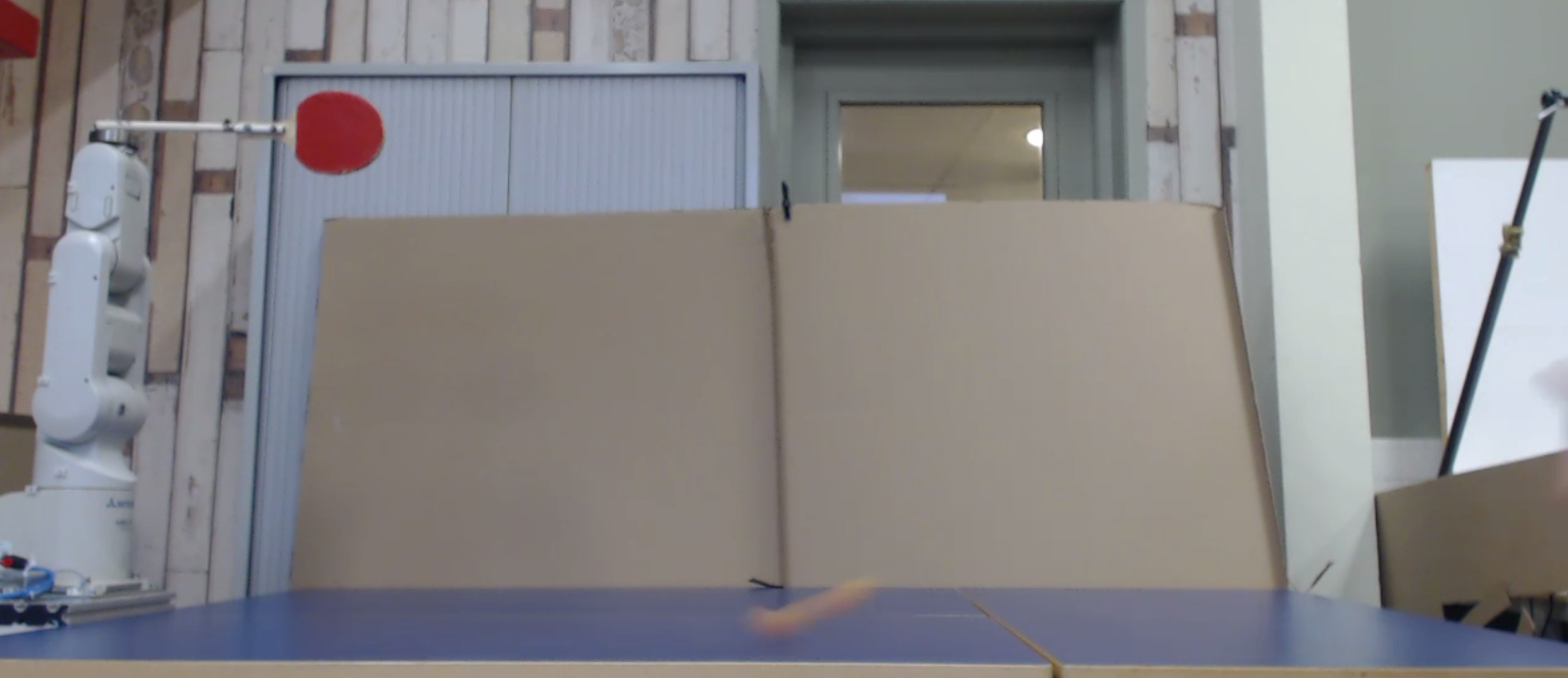
##### Betrouwbaarheid

De geïmplementeerde methoden hebben allemaal een criteria om correct te werken. Ook hebben ze faal criteria. Voor de Hough Circle Detection moeten de binnenkomende beelden niet uitgetrokken zijn. Dit komt veelal door een te lange sluitertijd (Camera kan niet genoeg beelden per seconden maken). De Kleur filters zijn afhankelijk van de kleur. Een goed beeld voor beiden staat in Figuur 21. Hierin is het balletje duidelijk rond met een kleurwaarde in HSV tussen de (24, 136, 157) en de (24, 235, 87).



Figuur Duidelijk beeld

Echter met de gebruikte camera’s en een hoge snelheid ontstaat een vaag beeld, dit staat in Figuur 22. Hierin is de bal te zien tijdens de stuiter beweging. De vorm is verloren doordat de gehele beweging zichtbaar is. De kleurwaarde is nu tussen de (15, 38, 123) en de (234, 33, 108). Bij deze snelheid is het object niet meer duidelijk te herkennen met de geïmplementeerde methoden en deze camera.



Figuur Snel beeld

In dit beeld koste het de bal 0.33 seconden om de tafel van 2.1 meter over te steken. Hiermee heeft de bal een snelheid van 6.26 m/s. Dit is hoger als de richtlijn van de opdracht, echter geeft het weer waar problemen kunnen ontstaan in de beeldherkenning. Dit probleem wordt veroorzaakt door een te lange belichting van een beeld voordat deze verwerkt kan worden. Hiervoor moet gekeken worden naar camera’s met een kortere sluitertijd (Minder tijd belicht per beeld) om scherpere beelden te ontvangen.

## Traject bepalen

Om te bepalen waar de bal heen gaat is informatie nodig over waar de bal heen gaat. Daarvoor zijn minimaal twee frames van het boven en zijaanzicht nodig. Deze gegevens worden vervolgens geanalyseerd in verschillende functies geanalyseerd. Het is namelijk niet nodig om de functies te combineren, omdat de uitkomsten gecombineerd zullen worden tot één locatie met een tijdstip.

### Bovenaanzicht

De enige informatie die uit het bovenaanzicht wordt gehaald is waar de bal land als die het einde van de tafel zal zijn in het horizontale bereik van de robot, in Figuur 16 is een schets te zien van de situatie. Voor de eindpositie is dan de volgende formule te gebruiken:

In deze formule is Punt 1 aangegeven als p1 en Punt 2 als p2. Verder hebben beide punten een positie op de X-as en Y-as.

Figuur Bovenaanzicht van de tafel

Y –as 🡪

X –as 🡪

Punt 1

Punt 2

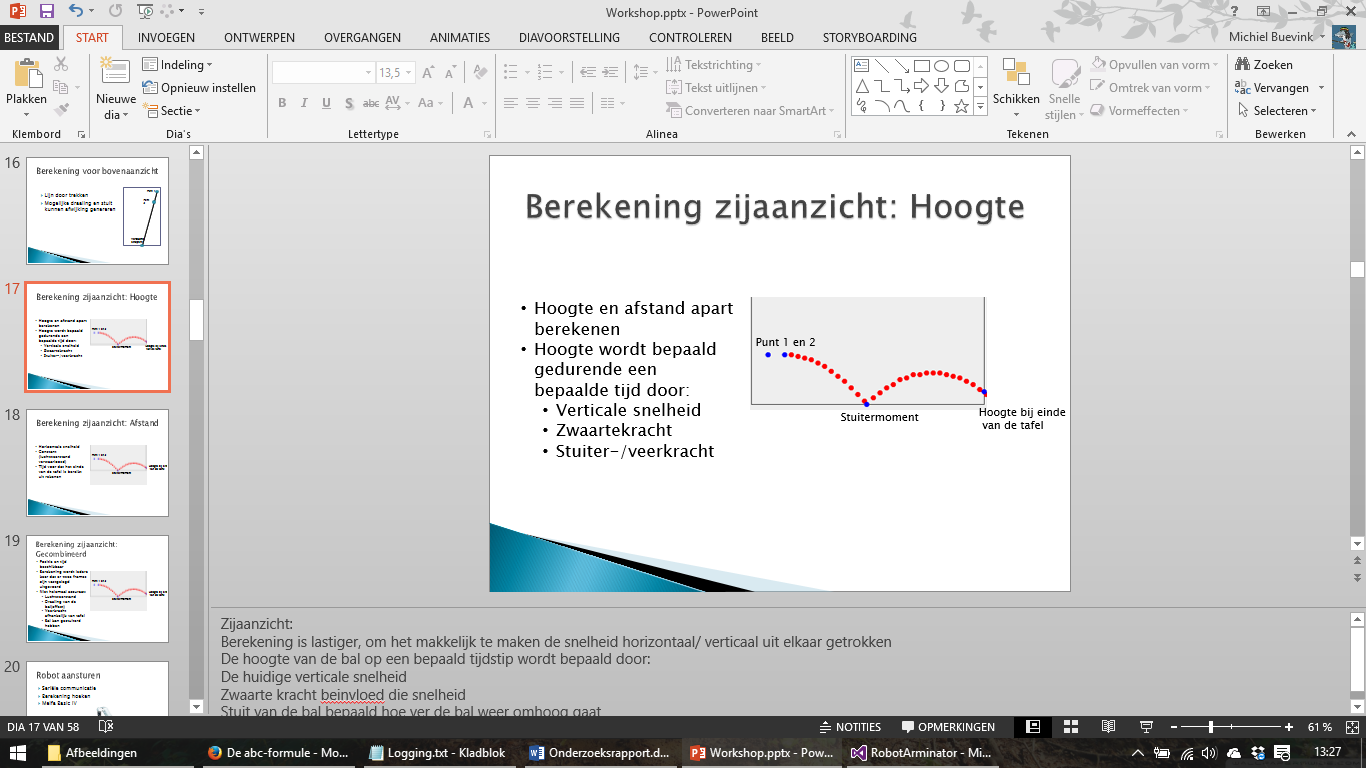
Verwachte

eindpunt

### Zijaanzicht

Voor het zijaanzicht moet de baan van de bal bepaald worden. Deze berekeningen zijn opgesplitst in een berekening voor de horizontale verplaatsing en de verticale verplaatsing.

Figuur Situatieschets voor het zijaanzicht



#### Horizontaal

Bij horizontale verplaatsing wordt er rekening mee gehouden dat de horizontale snelheid constant is en dat bij een stuit de snelheid met een vaste factor verminderd. Om die manier kan de afgelegde afstand tot het stuiter moment worden berekend. Vervolgens kan vanaf dat punt met de nieuwe snelheid het tijdstip waarop het einde van de tafel wordt bereikt worden berekend.

#### Verticaal

Voor het bepalen van de huidige verticale snelheid kan die gemiddelde snelheid over het afgelegde stuk tussen de punten worden berekend. Omdat de gemiddelde snelheid niet de snelheid op het tweede punt. Zal het snelheid verschil tussen de twee punten worden bepaald. Het snelheidsverschil ontstaat door de valversnelling dus als die wordt meegenomen in de berekening ontstaat de onderstaande formule.

Om de hoogte te berekenen na een bepaalde tijd is een combinatie van de afgelegde afstand bij een vaste versnelling en start hoogte nodig. Als deze samengevoegd worden komt neer op de volgende formule:

Echter is het eerste wat uitgerekend moet worden wanneer de bal op de tafel stuitert. Dus is de formule met behulp van de ABC-formule omgekeerd resulterend in de volgende functie:

De opwaartse snelheid na een stuit wordt bepaald aan de hand van de snelheid die de bal heeft gekregen door de valversnelling op het moment van de stuit in omgekeerde richting met een vaste vertragingsfactor.

### Gecombineerd

Wanneer de horizontale en verticale formules gecombineerd worden kan uitgerekend worden waar de bal op de tafel komt. Dit wordt gedaan door met de derde formule van de verticale berekeningen te nemen, daarmee kan uitgerekend worden hoelang het duurt voor de bal de grond raakt. Vervolgens wordt met de horizontale snelheid bepaald hoeveel afstand horizontaal is afgelegd. Dit wordt herhaald tot het berekende punt waar de op de tafel zal landen achter de tafel ligt.

Op dat moment kan met de horizontale berekening bepaald worden hoelang het duurt voor het einde van de tafel bereikt is. En met de tweede verticale formule bepaald worden hoe hoog te bal op dat moment is. Dus dan is uit het zijaanzicht berekend op welke hoogte en na welke tijd het einde van de tafel bereikt is. Van het bovenaanzicht wordt dan berekend waar de bal komt in het verticale bereik van de tafel.

### Afwijkingen

De resultaten van de traject bepaling zijn niet volledig accuraat. Dit komt omdat niet met alle factoren rekening gehouden is of rekening mee gehouden kan worden. Mede omdat het de complexiteit van de berekeningen verhoogt. Een aantal van deze factoren zijn:

* Luchtweerstand
* Draaiing of effect van de bal
* Stuitkracht van de tafel en bal
* De bal kan gestuiterd hebben tussen de twee gemeten punten

Echter is in de demo video gezien dat in de meeste situaties de uiteindelijke positie van de bal wordt benaderd op 20cm nauwkeurig, hier zit de 5cm marge van de aansturing van de arm bij in. Deze video staat in de bijlage ‘Demo video’.

## Reactiesnelheid

In het proces vanaf het gooien van een balletje tot het slaan zijn zeer veel tijdsfactoren aanwezig. Dit begint bij de beeldherkenning. Uit het vorige hoofdstuk is gebleken dat dit met de toegepaste methode “PFC” ongeveer 35 milliseconden duurt.

Voor de tijdsduur van de traject bepaling zijn meerdere berekeningen achter elkaar gemeten en daarvan is een gemiddelde genomen. Hetzelfde geld voor de berekening van de hoeken. Om de tijden vanaf het zenden tot de stilstand van de robot te meten is een video gemaakt. Deze video is geanalyseerd en daarin zijn de tijden opgemeten.

In Figuur 25 staan de gemeten tijden in een cirkeldiagram. Hieruit kan geconcludeerd worden dat de bewegingstijd veel te lang duurt. Helaas is dit niet een enige vertragende factor en zal voor een optimale oplossing ook de verzend- en verwerkingstijden moeten worden verlaagd.

Figuur Reactietijd

## Aansturing

Het bericht dat naar de robot wordt verstuurd bevat de verschillende hoeken waarin

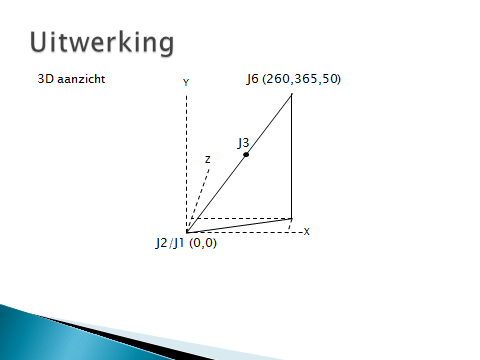
de robotarm moet worden gezet. De trajectbepaling heeft de positie berekend waar

de robotarm naartoe moet bewegen.

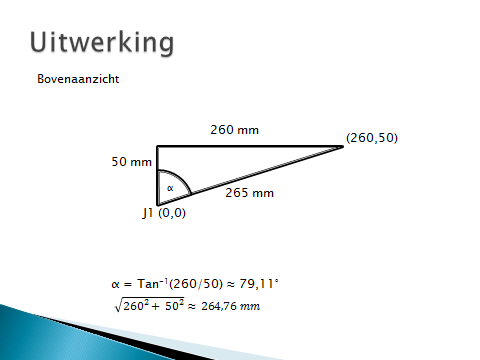
### Het berekenen van de hoeken

Om de robotarm op de berekende coördinaten te zetten moeten de hoeken worden berekend. Er zijn twee mogelijke manieren voor gevonden, ‘Inverse Kinematics’ en ‘Brute Force’. Inverse kinematics is een wiskundige methode voor het berekenen van de hoeken. Dit kan worden toegepast tot 3 assen. De onderstaande berekening is toegepast op het coördinaat (260,365,50).

### Beginsituatie

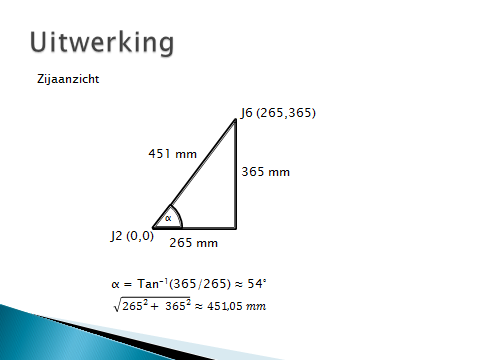


#### Bovenaanzicht



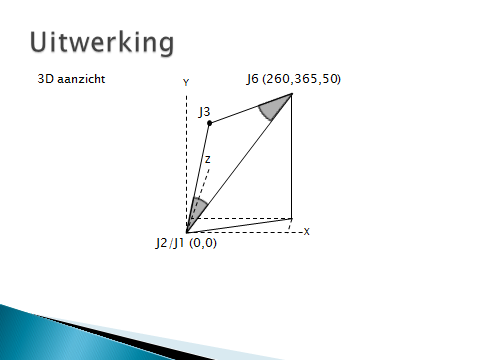
Dus de hoek van J1 = 79,11˚

#### Zijaanzicht

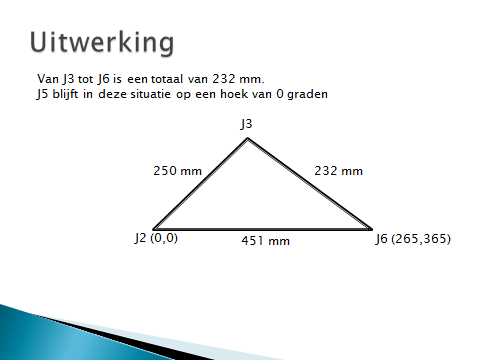


Wanneer er gerekend wordt met 2 hoeken zal dit de berekening zijn van J2. Dan is J2 dus 54˚. Bij een berekening met 3 hoeken zal de hoek van J2 veranderen om coördinaten te bereiken die dichter bij de robot liggen dan de totale robotarmlengte. Dan wordt J3 ook berekend.

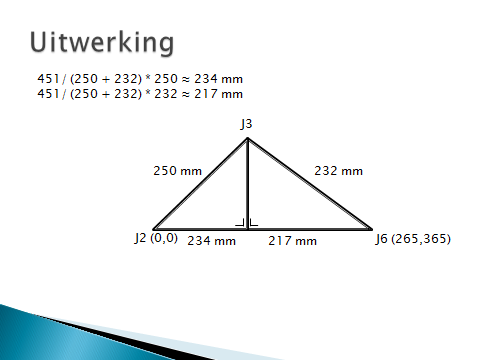
### Nieuwe beginsituatie



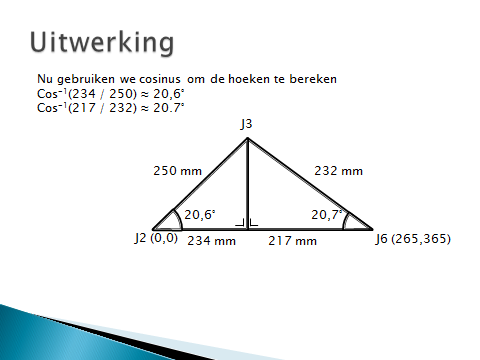
#### Zijaanzicht



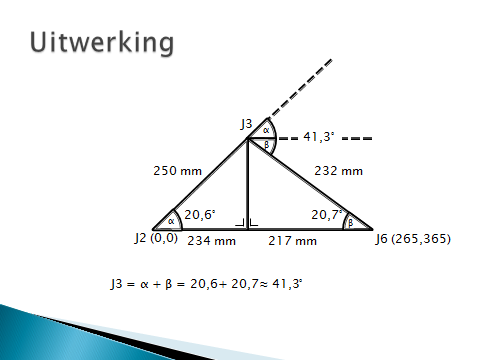
Doormiddel van de armlengte te schalen op de te bereken lengte kunnen er twee nieuwe rechte hoeken worden gecreëerd.

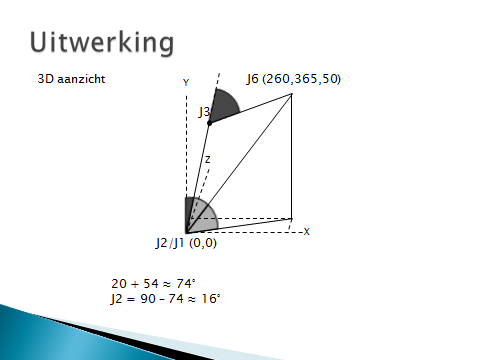


Aan de hand van deze gegevens kunnen de volgende hoeken worden berekend.

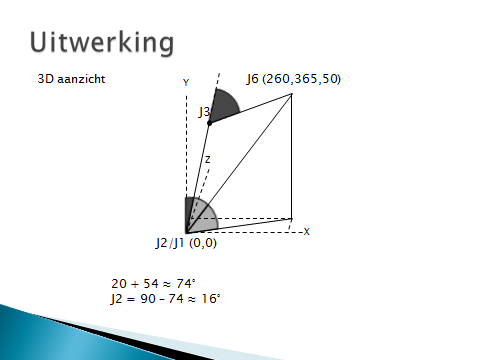


Aan de hand van de twee berekende hoeken kan de totale hoek van J3 ten opzichte van J2 worden berekend.





Wanneer J2 op een hoek van 0 graden staat, staat de arm recht omhoog. Daarom moet de volgende berekeningen nog worden toegepast.



Dit is relatief een eenvoudige berekening met erg veel herhalende onderdelen. Wanneer er 4 assen berekend moeten worden wordt berekening helaas zeer complex en worden er vaak aannames gemaakt. Bijvoorbeeld dat de eerste of de laatste as parallel of loodrecht op de ondergrond staat. Dit kan in dit project niet worden gegarandeerd en daarom is er voor een andere berekening gekozen, namelijk een bruteforce algoritme.

### Bruteforce

Omdat de berekeningen door middel van inverse kinematics zeer complex worden bij meer dan 3 assen is ervoor gekozen om de positie om te rekenen naar hoeken door middel van een bruteforce methode. Voor elke mogelijke combinatie van hoeken wordt gekeken of dit overeenkomt met de gewenste positie. Dit heeft een marge van vijf cm. Wanneer een positie niet mogelijk is zullen alle hoeken op 0 worden gezet. Omdat dit zeer lang kan duren is het geoptimaliseerd door de hoeken in stappen van een centimeter te bekijken. Dit zorgt ervoor dat het een accurate oplossing is.

De robot kan bepaalde hoeken alleen bereiken wanneer sommige assen worden omgedraaid. Omdat de positie niet 100% accuraat hoeft te zijn omdat deze zich in het midden van het batje bevindt. Een afwijking van enkele centimeters maakt hierbij niet uit. Daarom is er een functie ingebouwd die de hoeken van de robot berekent en daarbij de hoeken pakt waarbij de minste beweging nodig is.

Hieronder staat het bruteforce algoritme kort uitgelegd. De x en y die hier worden berekend worden vergeleken met de gewenste positie. Wanneer dit binnen 5 centimeter overeenkomt zullen deze hoeken worden doorgestuurd naar de robot.

# Conclusies

In dit hoofdstuk worden de conclusies uit het Theoretische deel van het onderzoek(Hoofdstuk 8) aangevuld met de resultaten die behaalt zijn tijdens het praktijk onderzoek.

## Beeldherkenning

Voor de beeldherkenning zijn de betrouwbaarheid en de snelheid belangrijk.

In het eerste deel van het onderzoek is gekozen om gebruik te maken van camera’s met een herkenningsmethode. Deze camera’s worden boven de tafel geplaatst voor een bovenaanzicht, en langs de tafel voor het zijaanzicht. Hierdoor kan het balletje gevonden worden.

Vervolgens zijn 9 van deze herkenningsmethode onderzocht. Voor de snelheid en betrouwbaarheid is gekozen om Hough Circle Detection en Particle Filter Color Tracking te realiseren.

Deze methoden zijn gerealiseerd en getest. In de onderstaande grafiek wordt weergegeven hoelang het de methoden kost om een beeld te herkennen.

Hieruit is geconcludeerd dat de Kleuren gebaseerde methoden sneller zijn. Op het gebied van betrouwbaarheid zijn de methoden gelimiteerd door de gebruikte camera’s. Doordat de sluitertijd van de camera’s te laag is ontstaat een waas waardoor vorm en kleur verloren gaan. Echter blijkt dat de kleur minder snel vervaagt dan de vorm, waardoor de kleur filters wederom beter presteren. Doordat de Particle Filter Implementatie beter om kan gaan met de mogelijke afleidingen in het beeld is deze methode betrouwbaarder dan de kleuren filter.

Hierdoor wordt geconcludeerd dat in de omstandigheden van het project, uit de onderzochte methoden de Particle Filter de beste keuze is voor dit project.

## Aansturing

De aansturing van de robot is onderverdeeld in twee stukken. Dit is de aansturing op de controller en de aansturing in de software.

In de software wordt er een seriële poort geopend. Hierna wordt er een bericht gestuurd. In dit bericht zullen de hoeken worden geplaatst die zijn berekend.

De robot heeft twee threads. De 1e thread is verantwoordelijk voor de beweging van de robot. De 2e thread is verantwoordelijk voor de flow van de Melfa Basic software. Deze krijgt opdrachten binnen en stuurt deze door naar de 1e thread. Ook stuurt deze een bevestiging van ontvangst of een errorbericht terug naar de software.

Indien er een error optreedt, zal de 2e thread proberen om de error te herstellen.

Dit zorgt ervoor dat er bij kleine fouten doorgewerkt kan worden en dat de robot niet handmatig moet worden herstart.

## Traject berekening

Het traject wordt in twee stappen berekend, eerst het bovenaanzicht en daarna het zijaanzicht. Het bovenaanzicht wordt berekend door een lineaire lijn te berekenen door twee coördinaten. Voor het zijaanzicht wordt apart gekeken naar de horizontale en verticale snelheid. Echter is een afwijking van ongeveer 15 centimeter in lab omstandig heden niet heel nauwkeurig. Maar voor het formaat van het batje zal dit voldoende zijn. Eventuele verbetering zouden doorgevoerd kunnen worden als er hinder ondervonden wordt van de gebrekkige nauwkeurigheid.

## Hoeken berekening

Er zijn dus twee manieren om de hoeken te berekenen, ‘Inverse Kinematics’ en ‘Brute Force’. Tot drie assen is Inverse Kinematics zeer handig, maar dit is voor vijf assen zeer complex en het valt buiten de scope van dit project. Daarom is brute force toegepast, omdat dit in verband met de projecttijd en de functionaliteit de meeste voordelen heeft. Dit algoritme kijkt voor elke mogelijke combinatie van hoeken of dit overeenkomt met de gewenste positie. Dit heeft een marge van vijf cm. Wanneer een positie niet mogelijk is zullen alle hoeken op 0 worden gezet.

## Algemene conclusie

Zoals beschreven in de tussentijdse conclusies over de snelheid(Hoofdstuk 9.4.3) is geconcludeerd dat de Robotarm alleen snel genoeg kan reageren als de bal meer dan een halve seconde nodig heeft om de tafel over te steken. Deze hypothese is in het praktijk onderzoek afgekeurd, doordat de totale tijd die het systeem nodig heeft om te reageren groter is dan 1 seconde, zoals geïllustreerd in Figuur 18 . De robotarm reageert niet snel genoeg op de input en beweegt niet snel genoeg om het balletje terug te kunnen slaan.

Daarbij heeft de arm niet genoeg bereik om diagonaal gespeelde ballen(die aan de zijkant van de tafel af zullen gaan) terug te kunnen slaan.

Vervolgens kan de beeldherkenning geen snelle ballen detecteren doordat de webcams een te hoge sluitertijd hebben, wat leidt tot onscherpe beelden die niet verwerkt kunnen worden.

Met deze resultaten kan de hoofdvraag worden beantwoord:

“Hoe kan de robotarm, Melfa RV-2AJ, tafeltennissen?”

Met de huidige opstelling kan geconcludeerd worden dat de Melfa RV-2AJ niet kan tafeltennissen.

## Aanbevelingen

Er is dus geconcludeerd dat de robot Melfa RV-2AJ niet in staat is om te tafeltennissen. De robotarm inclusief de controller en de camera’s zijn daarbij het grootste knelpunt. Daarom wordt er aanbevolen om de robotarm te vervangen met een snellere robotarm, met een hogere reactiesnelheid. Hierdoor zal de arm sneller op de gewenste locatie kunnen komen waardoor hij meer ballen kan tegenhouden.

Ook is het aanbevolen om de webcams te vervangen met andere camera’s. De gewenste eigenschappen voor de camera’s gesorteerd op prioriteit zijn:

* Grote kijkhoek (De gebruikte kijkhoek is 78 graden, dit is minimaal vereist)
* Korte sluitertijd
* Hoge framerate (Beeldjes per seconden)

Een hogesnelheidscamera met een hoge kijkhoek zal hiervoor geschikt zijn. Dit zal leiden tot betere beelden om te verwerken, waardoor het systeem de snelle ballen zal kunnen detecteren.

# Literatuurlijst

Allain, A. (sd). *Cprogramming*. Opgeroepen op 09 03, 2015, van Where C and C++ Differ: http://www.cprogramming.com/tutorial/c-vs-c++.html

Anderson, T. (2014, 05 01). *Microsoft’s new open source direction for C# and .NET (and native compilation too): Anders Hejlsberg explains*. Opgeroepen op 09 03, 2015, van Tim Anderson's ITWriting: http://www.itwriting.com/blog/8361-microsofts-new-open-source-direction-for-c-and-net-and-native-compilation-too-anders-hejslberg-explains.html

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 679-698.

Cass, S. (2015, 07 20). *The 2015 Top Ten Programming Languages.* Opgeroepen op 09 03, 2015, van Spectrum: http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages

Cheung, S.-C., & Kamath, C. (2004). *Robust techniques for background subtraction in urban.* Opgehaald van Computation: https://computation.llnl.gov/casc/sapphire/pubs/UCRL-CONF-200706.pdf

Himani, P., Darshak, T., & Udesang, J. (2014, Februari). *A Survey on Object Detection and Tracking Methods.* Opgeroepen op Oktober 1, 2015, van ijircce: http://www.ijircce.com/upload/2014/february/7J\_A%20Survey.pdf

Improved Outcomes Software. (2016, 1 4). *Improved Outcomes Software*. Opgehaald van Manhattan Distance: http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering\_Parameters/Manhattan\_Distance\_Metric.htm

jain.pk. (2006, 10 14). *Using Inline Assembly is C/C++*. Opgeroepen op 09 03, 2015, van Code Project: http://www.codeproject.com/Articles/15971/Using-Inline-Assembly-in-C-C

Kernel.org. (2014, January 8). *Build an RT-application*. Opgehaald van Kernel.org: https://rt.wiki.kernel.org/index.php/HOWTO:\_Build\_an\_RT-application

Mitsubishi Electric. (2002, Juli 1). Melfa Industrail Robots Specifications Manual RV-1a/RV2AJ Series.

Mitsubishi Electric. (2005, Juli 14). Melfa Industrial Robots Instruction Manual (Detailed explanations of functions and operations) CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller.

Mitsubishi Electric Corporation. (2002). *Instruction Manual Robot arm setup & maintenance.* Tokyo.

Mitsubishi Electric Corporation. (2002). *Standard Specifications Manual.* Tokyo.

Mitsubishi Electric Corporation. (2004). *Safety Manual.* Tokyo.

Nieto, R. M. (2013, September). *On the fusion of single-target video objects tracking algorithms.* Opgeroepen op 11 29, 2015, van Escuela Politecnica: http://www.eps.uam.es/nueva\_web/intranet/ga/tfdm/trabajos/Rafael\_Martin\_Nieto.pdf

NTTB. (sd). *NTTB richtlijnen C-accommodatie.* Opgehaald van NTTB: http://www.nttb.nl/userfiles/Clubadvies/NTTB\_richtlijnen\_c-accommodatie.pdf

Panguluri, V. (2014). Opgehaald van My blog about Table Tennis: http://myblogabouttabletennis.blogspot.nl/

Redactie. (2014, 05 26). *Transformeer dit bereau in een pingpongtafel*. Opgeroepen op 10 19, 2015, van Madpac: http://www.madpac.nl/gear/transformeer-dit-bureau-een-pingpongtafel/

Reynolds, V. (2010, 08 24). *Is a garbage collector (.net/java) a problem for real-time systems?* Opgeroepen op 09 3, 2015, van Stackoverflow: http://stackoverflow.com/questions/3559878/is-a-garbage-collector-net-java-a-problem-for-real-time-systems

Rhody, H. (2005, 10 11). *Lecture 10: Hough Circle Transform.* Opgeroepen op 10 19, 2015, van Rochester Institute of Technology: https://www.cis.rit.edu/class/simg782/lectures/lecture\_10/lec782\_05\_10.pdf

Rojas, R. (sd). *Lucas-Kanade in a Nutshell.* Opgeroepen op 10 6, 2015, van Freie Universität Berlin: http://www.inf.fu-berlin.de/inst/ag-ki/rojas\_home/documents/tutorials/Lucas-Kanade2.pdf

Russinovich, M., Solomon, D., & Jonescu, A. (2009). *Windows Internals.* Redmond, Washington: Microsoft Press.

Singla, N. (2014). *Motion Detection Based on Frame Difference Method.* Opgehaald van Research India Publications: http://www.ripublication.com/irph/ijict\_spl/ijictv4n15spl\_10.pdf

Struijker, I. (sd). *Kernel Density.* Opgeroepen op 10 12, 2015, van Lancaster University: http://www.lancs.ac.uk/~struijke/density/kernel.html

Tang, H.-p., Mizoguchi, M., & Toyoshima, S. (sd). Opgehaald van ITTF.com: http://www.ittf.com/ittf\_science/SSCenter/docs/200200027%20-%20Tang%20-%20Speed.pdf

tecosystems. (2015, 07 01). *The RedMonk Programming Language Rankings: June 2015*. Opgeroepen op 09 03, 2015, van RedMonk: http://redmonk.com/sogrady/2015/07/01/language-rankings-6-15/

The University of Iowa Engineering. (1997). *Chapter 14, Motion Analysis: Optical flow.* Opgehaald van The University of Iowa Engineering: http://user.engineering.uiowa.edu/~dip/lecture/Motion2.html

Tiobe Software. (sd). *The Coding Standards Company.* Opgeroepen op 09 03, 2015, van Tiobe Software: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Vojir, T., Noskova, J., & Matas, J. (2013). *Robust Scale-adaptive Mean-Shift for Tracking.* Opgeroepen op 10 15, 2015, van Center for Machine Perception: http://cmp.felk.cvut.cz/~vojirtom/publications/scia2013.pdf

Yang, Y., Jia, Y., Rong, C., Zhu, Y., Wang, Y., & Yue, Z. G. (2013, April). *Object Tracking Based on Corrected Background-Weighted Histogram Mean Shift and Kalman Filter.* Opgeroepen op 10 15, 2015, van Research Gate: https://www.researchgate.net/publication/266649404\_Object\_Tracking\_Based\_on\_Corrected\_Background-Weighted\_Histogram\_Mean\_Shift\_and\_Kalman\_Filter

# Bijlagen

## Daily inspection items

Tabel 15 Daily inspection items (Mitsubishi Electric Corporation, 2004)

|  |  |  |  |
| --- | --- | --- | --- |
| Procedure | Inspection item (details) | | Remedies |
| Before turning power ON (Check the following items before turning the power ON.) | | | |
| 1 | Are any of the robot installation bolts loose? | (Visual) | Securely tighten the bolts. |
| 2 | Are any of the cover tightening screws loose? | (Visual) | Securely tighten the screws. |
| 3 | Are any of the hand installation bolts loose? | (Visual) | Securely tighten the bolts |
| 4 | Is the power supply cable securely connected? | (Visual) | Securely connect. |
| 5 | Is the machine cable between the robot and controller securely connected?  (Visual) | | Securely connect. |
| 6 | Are there any cracks, foreign contamination or obstacles on the robot and controller cover? | | Replace with a new part, or take remedial measures. |
| 7 | Is any grease leaking from the robotarm? | (Visual) | After cleaning, replenish the grease. |
| 8 | Is there any abnormality in the pneumatic system? Are there any air leaks, drain clogging or hose damage? Is the air source normal?  (Visual) | | Drain the drainage, and remedy the air leaks (replace the part). |
| After turning the power ON (Turn the power ON while monitoring the robot.) | | | |
| 1 | Is there any abnormal motion or abnormal noise when the power is turned ON? | | Follow the troubleshooting section. |
| During operation (try running with an original program) | | | |
| 1 | Check whether the movement points are deviated? Check the following points if there is any deviation.  1. Are any installation bolts loose?   1. Are any hand installation section bolts loose? 2. Are the positions of the jigs other than the robot deviated? 3. If the positional deviation cannot be corrected, refer to "Troubleshooting", check and remedy. | | Follow the troubleshooting section. |
| 2 | Is there any abnormal motion or abnormal noise? | (Visual) | Follow the troubleshooting section. |

## Periodic inspection

Tabel 16 Periodic inspection items (Mitsubishi Electric Corporation, 2004)

|  |  |  |
| --- | --- | --- |
| Procedure | Inspection item (details) | Remedies |
| Monthly inspection items | | |
| 1 | Are any of the bolts or screws on the robotarm loose? | Securely tighten the bolts. |
| 2 | Are any of the connector fixing screws or terminal block terminal screws loose? | Securely tighten the screws. |
| 3 | Remove the cover at each section, and check the cables for wear damage and adherence of foreign matter. | Check and eliminate the cause.  If the cables are severely damaged, contact the Mitsubishi Service Department. |
| 3-month inspection items | | |
| 1 | Is the timing belt tension abnormal? | If the timing belt is loose or too tense, adjust it. |
| 6-month inspection items | | |
| 1 | Is the friction at the timing belt teeth severe? | If the teeth are missing or severe friction is found, replace the timing belt. |
| Yearly inspection items | | |
| 1 | Replace the backup battery in the robotarm. | Exchange it referring to "5.3.5 Replacing the backup battery" on page 54. |
| 2-year inspection items | | |
| 1 | Lubricate the grease at the harmonic reduction gears for each axis. | Lublicate it referring to "5.3.4 Lubrication" on page  52. |

## Demo video

Zie bestand “Bijlagen\demovideo.avi”.