Hoe kan een robotarm tafeltennissen?

Onderzoeksrapport – AP Project – Iets Technisch 1

5-11-2015

HAN

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

# 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) | 05-11-2015 |  |
| 1.0 |  |  |  |

# Inhoudsopgave

[Versiebeheer 1](#_Toc434926351)

[Inhoudsopgave 2](#_Toc434926352)

[1 Inleiding 4](#_Toc434926353)

[2 Tafeltennis 5](#_Toc434926354)

[2.1 Bereik 5](#_Toc434926355)

[2.2 Reactietijd 6](#_Toc434926356)

[3 De robotarm 7](#_Toc434926357)

[3.1 De snelheid 7](#_Toc434926358)

[3.2 De kracht 8](#_Toc434926359)

[3.3 Draaihoeken 9](#_Toc434926360)

[3.4 De montage van het batje 11](#_Toc434926361)

[4 Veiligheid 12](#_Toc434926362)

[4.1 Wat is veiligheid? 12](#_Toc434926363)

[4.2 Hoe wordt de veiligheid gehandhaafd? 12](#_Toc434926364)

[5 Beeldherkenning 14](#_Toc434926365)

[5.1 Nodige eigenschappen voor de beeldherkenning 15](#_Toc434926366)

[5.1.1 Resistentie tegen verschillen 15](#_Toc434926367)

[5.1.2 Uitvoer tijd 15](#_Toc434926368)

[5.1.3 Accuraatheid 15](#_Toc434926369)

[5.1.4 Herstelmogelijkheid wanneer het balletje buiten beeld valt 15](#_Toc434926370)

[5.2 Beschikbare beeldherkenning technieken/middelen 15](#_Toc434926371)

[5.2.1 Opvangen beeld materiaal / Invoer beeldmateriaal 15](#_Toc434926372)

[5.2.2 Detecteren van de mogelijke object posities (Object Detectie) 17](#_Toc434926373)

[5.2.3 Herkenning van het object (Object Classificatie) en vaststellen positie 19](#_Toc434926374)

[6 De programmeertaal 24](#_Toc434926375)

[6.1 Real-time 24](#_Toc434926376)

[6.1.1 Besturingssysteem 24](#_Toc434926377)

[6.1.2 Programmeertaal 24](#_Toc434926378)

[6.2 Communicatie 24](#_Toc434926379)

[6.3 Objecttracking 24](#_Toc434926380)

[6.4 User base/support 24](#_Toc434926381)

[6.5 De programmeertaal 25](#_Toc434926382)

[6.5.1 Toelichting bij tabel: 25](#_Toc434926383)

[7 Communicatie 27](#_Toc434926384)

[7.1 Communicatiemethoden 27](#_Toc434926385)

[7.1.1 Activeren van de RS-232C poort en configuratie van de software 28](#_Toc434926386)

[7.1.2 Commando’s sturen over RS-232C 28](#_Toc434926387)

[7.1.3 Software op de robot 29](#_Toc434926388)

[7.1.4 Vergelijking 30](#_Toc434926389)

[8 Conclusies 31](#_Toc434926390)

[8.1 Programmeertaal 31](#_Toc434926391)

[8.2 Operating system 31](#_Toc434926392)

[8.3 Beeldherkenning 32](#_Toc434926393)

[8.4 Eigenschappen 33](#_Toc434926394)

[8.4.1 Het bereik 33](#_Toc434926395)

[8.4.2 De montage van het batje 33](#_Toc434926396)

[8.4.3 De kracht 33](#_Toc434926397)

[8.4.4 De snelheid 33](#_Toc434926398)

[8.5 Veiligheid 34](#_Toc434926399)

[8.6 Communicatie 34](#_Toc434926400)

[Bijlagen 37](#_Toc434926401)

[Daily inspection items 37](#_Toc434926402)

[Periodic inspection 38](#_Toc434926403)

# Inleiding

De afdeling ICA heeft een robotarm, van het type Melfa RV-2AJ-S12, 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.

Aan de hand van de hoofdvraag “Hoe kan de robotarm, Melfa RV-2AJ-S12, tafeltennissen?” zijn diverse deelvragen opgesteld. Deze zijn onderverdeeld in een aantal hoofdstukken waarin ze worden behandeld.

De onderverdeling van de deelvragen:

**2. Tafeltennis**

1. Welke eigenschappen zijn van belang voor de robotarm om deze robotarm te laten tafeltennissen?

**3. De robotarm**

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?

**4. Veiligheid**

1. Hoe kan de veiligheid worden gegarandeerd?
   1. Wat wordt er onder veilig verstaan?

**5. Beeldherkenning**

1. Hoe wordt het tafeltennis balletje gedetecteerd?

**6. De programmeertaal en 7 Onderzoek communicatie**

1. Welke programmeertaal is het meest geschikt om de robotarm te programmeren?

# Tafeltennis

|  |
| --- |
| Behandelde deelvragen:   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, de slagkracht en de reactietijd.

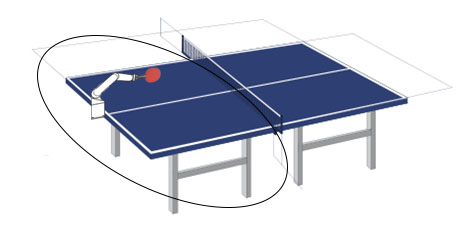
## **Bereik**

Een pingpongtafel is 2,74m lang en 1,525m 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. Deze rand 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

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 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. Er vanuit gaande dat de robot een halve seconde nodig heeft om het balletje terug te slaan mag de bal 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 robot arm, RV-2AJ, kan tafeltennissen zijn er diverse eigenschappen van deze robot onderzocht. Deze eigenschappen hebben invloed op de snelheid, de kracht, de houding en het bereik.

## 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. In de onderstaande tabel, Tabel 1 Speed of motion, staat de snelheid in graden per seconden per scharnier.

Tabel Speed of motion





Figuur Draaisnelheid van J1

Uit de bovenstaande gegevens kan geconcludeerd worden dat het scharnier ‘J1’ 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 bepaald de kijkrichting.

## 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 2 Allowable moment load.

Tabel Allowable moment load



Het scharnier waaraan het batje bevestigd zal worden is J6 en deze heeft een kracht van 1,10 newtonmeter. 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 4 Maximum load capacity.



Figuur Maximum load capacity

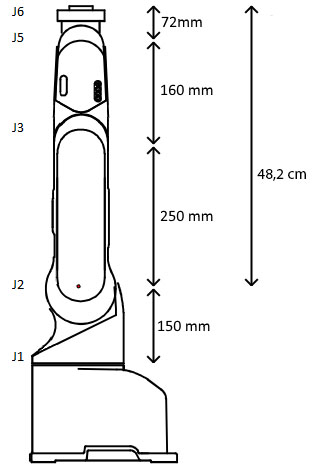
## Draaihoeken

Er zijn twee varianten van de robotarm, de ‘RV-2AJ’ en de ‘RV-2AJC-SB’. Tijdens dit project wordt de ‘RV-2AJ’ gebruikt en dit is de 5-axes type. Dit is de linker robot in Figuur 5 De verschillende types.



Figuur De verschillende types

In Figuur 6 Het nulpunt staat de robot arm op zijn nulpunt. Alle motoren staan dan op de coördinaten (0,0). 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 3, Operating range staan per as de maximale bereikbare hoeken. Figuur 7 Draaihoeken daaronder geeft dit schematisch weer.

Tabel 3, Operating range





Figuur Draaihoeken

De robotarm heeft een ingebouwde beveiliging dat hij niet voorbij de grenzen van de scharnieren kan gaan, maar hij kan wel zichzelf raken binnen de grenzen. Wanneer er een batje op gemonteerd wordt zullen deze grenzen veranderen. De rode lijnen geven het gebied van J5 aan. Dit is het uiterste draaipunt en is daarom het belangrijkste scharnier om in de gaten te houden.

## 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. Het batje kan in het verlengde en haaks worden geplaatst, dit is weergegeven in Figuur 8 Mogelijke positie van het batje.



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 4 Voor- en nadelen van mogelijke posities biedt een overzicht van de voor- en nadelen per positie.

Tabel Voor- en nadelen van mogelijke posities

|  |  |  |
| --- | --- | --- |
| Positie | Voordelen | Nadelen |
| In het verlengde van de robot | Heeft 23 cm extra bereik  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  De breedte van een batje is 14 cm  De breedte van het handvat is 3 cm.  ((14 - 3) / 2) + 3 = 8,50 cm |

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.

# 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 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. Dit omvat het maximale bereik van de robotarm inclusief de lengte van het batje en een kleine buffer. Deze veiligheidszone zal worden aangegeven met een lint. Deze grenzen mogen niet worden overschreden door onbevoegden. In Figuur 9 Veiligheidszone is deze zone schematisch weergegeven.



Figuur Veiligheidszone

Het is op dit moment voor de robot mogelijk om bewegingen te maken waarbij hij 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 de bijlagen, Daily inspection items en Periodic inspection.

# Beeldherkenning

|  |
| --- |
| Behandelde deelvragen:   1. Hoe wordt het tafeltennis balletje 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).

Figuur Cyclus beeldherkenning

Deze onderdelen worden in het paragraaf “Beschikbare beeldherkenning technieken/middelen” verder toegelicht.

## 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.

### Uitvoer tijd

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 beeld materiaal / 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 hij dit signaal terug ontvangt. Hiermee kan een afstand worden uitgerekend tot de sensor. Echter kan dit signaal worden verstoord door andere geluidsbronnen in de omgeving, zoals het contact van het balletje op een batje, het stuiteren van het balletje of het spreken van mensen in de omgeving.

#### 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 object, dat gelijk zal zijn aan zijn omgeving.

#### 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 beeld materiaal 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 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 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

Dit model 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 hij het nieuwe object zal kennen. 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 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 hij 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)

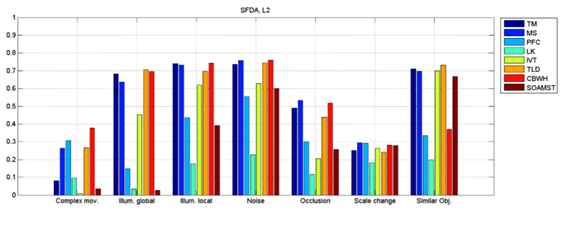
Dit algoritme is wederom een aanpassing op het Mean-Shift systeem (Vojir, Noskova, & Matas, 2013). Hierbij probeert het systeem een schaal van het object in te schatten. Hiermee kan het systeem het object terugvinden wanneer de schaal veranderd. Deze aanpassing presteert beter op beelden waar de schaal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

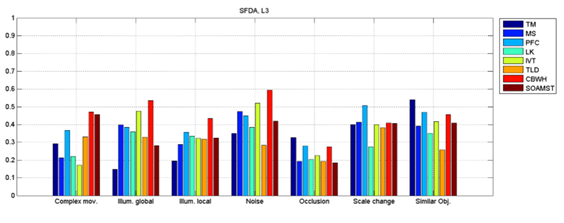
#### Vergelijking

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 mechanisme is uitgerekend door Rafael Martin Nieto (Nieto, 2013). Hieronder zijn twee grafieken (Nieto, 2013)) zichtbaar voor de accuraatheid van de tracking mechanisme 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 12 Tracking nauwkeurigheid voor het Laoratorium Set (Nieto, 2013)



Figuur 13 Tracking nauwkeurigheid voor het Laoratorium Set (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 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 deelvragen:   1. Welke programmeertaal is het meest geschikt om de robotarm te programmeren? |

Het kiezen van de juiste programmeertaal voor het project is een belangrijke (onderzoeks) stap. De taal op zich is het fundament van je 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 (in de scope van het project ) betekent dat de robot gegarandeerd reageert binnen een gestelde tijd.

### Besturingssysteem

Wanneer er gebruik wordt gemaakt van een (normaal) besturingssysteem kan het zijn dat je programma door de scheduler of achtergrondprocessen vertraagd of onderbroken wordt.

Een real-time operating system (verder RTOS genoemd) kan zorgen voor een gegarandeerde uitvoertijd van een programma.

### 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.

## Communicatie

Communiceren met de robot gaat via een USB -> RS232 connector. Wanneer deze wordt aangesloten (en de juiste drivers worden geïnstalleerd) zal de PC dit identificeren als een COM poort.

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 verscheidenen eisen kunnen programmeertalen met elkaar vergeleken. Hiermee kan worden bepaald welke programmeertaal geschikt is voor onze opdracht.

Tabel Eigenschappen programmeertalen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Taal | Real-time (hard) | Operating systems | 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 |

### Toelichting bij tabel:

**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 wel jammer. Bij soft real-time zou een Windows OS voldoende moeten zijn.

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

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

**Operating system:**

Voor C/C++ geldt dat elke architectuur waarvoor een compiler beschikbaar is in principe gebruikt kan worden. Bij Java/C# kan elk operating system 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 OS.

Op een internet-artikel is beschreven hoe je Linux afstemt voor (soft) real-time gedrag (HOWTO: Build an RT-application). Ook wordt in deze bron beschreven dat de x86 structuur al interne service management interrupts (SMI) heeft. Deze kunnen niet gezien of bewerkt worden door het OS.

Linux configureren voor real-time operaties:

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

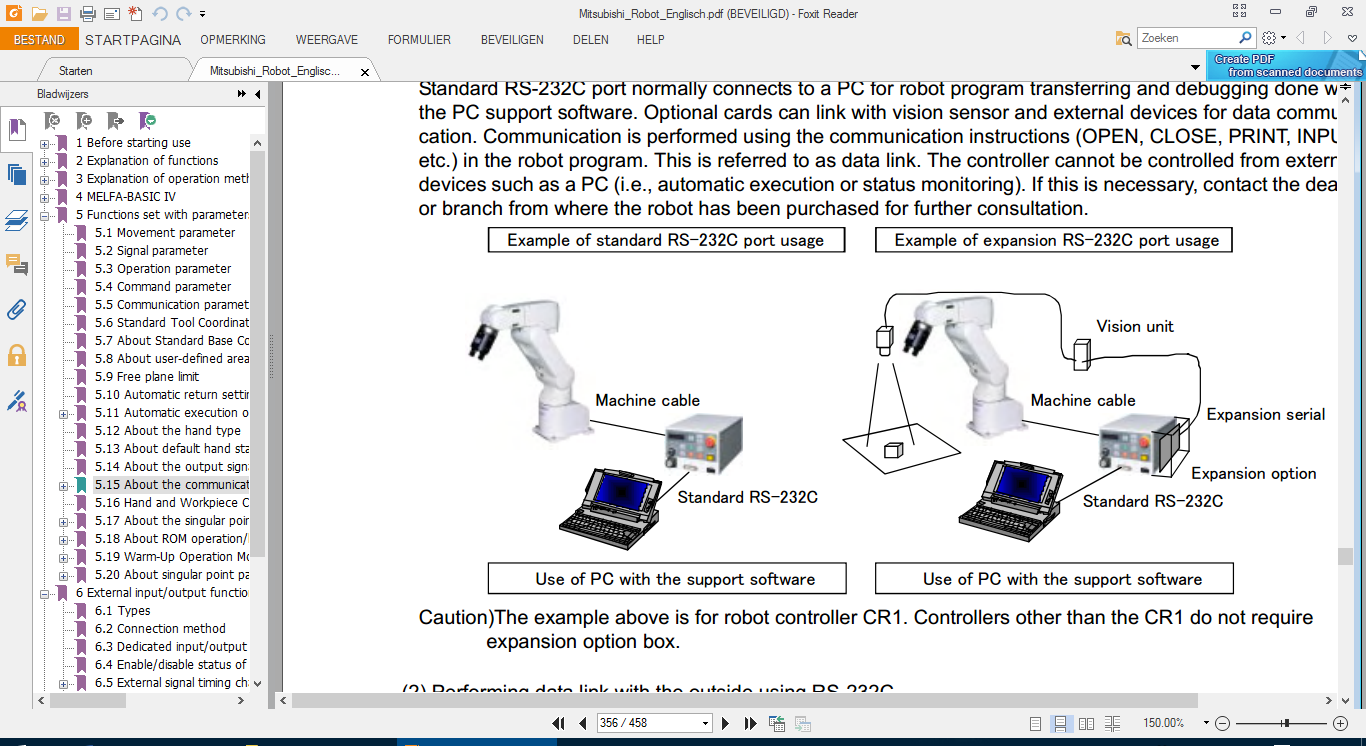
In het verhaal van Linux (voor real-time) wordt dus beschreven dat SMI (noodzakelijke interrupts van de processor) niet uitgeschakeld kunnen worden. Ook wordt beschreven dat de uitvoertijden hiervoor ver onder de 1ms liggen en dus vaak niet eens voor hard real-time iets uitmaakt.

Er zijn dus real-time operating systems.

# Communicatie

|  |
| --- |
| Behandelde deelvragen:   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. Het onderzoek naar communicatie is uitgevoerd door informatie uit de handleiding in de praktijk uit te proberen en de bevindingen op te schrijven.



Figuur 13 Schematische weergave aansluiting robot (Mitsubishi Electric, 2005)

## 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 robot arm

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 8 Overzicht van de in te stellen parameters (Mitsubishi Electric, 2005). Om de communicatie te kunnen gebruiken om de robot te laten bewegen moet de sleutelschakelaar op de controller op auto(ext) staan.

Tabel 8 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) | 1(CRLF) | 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 (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 9 Beschrijving van niet gedocumenteerde commando's

|  |  |
| --- | --- |
| Commando | Effect |
| OPEN=usertool | Verbinding openen |
| CTNLON | 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. Voor het programmeren van de software op de controller zijn twee programmeertalen mogelijk: Melfa-basic 4 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”.

### 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 10 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. |

# Conclusies

TODO

## Programmeertaal

Uiteindelijk is C++ de beste keuze voor het programmeren van een aansturingsprogramma voor de robot. Door C++ te gebruiken kunnen we ook in C schrijven. C++ compilers kunnen in zekere zin ook C compileren. Er zijn kleine (vaak syntactische) verschillen (Allain, sd). Ook kan in C++ inline assembly gebruikt worden (jain.pk, 2006).

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 (in theorie) mogelijk. Men weet dan zeker dat twee instructies elkaar volgen. Het vervelende is wel dat een besturingssysteem de processor kan kapen waarbij tussendoor andere code wordt uitgevoerd. Hierdoor zal er dus in de praktijk “jitter” ontstaan (meer hierover bij RTOS). 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.

Ook voldoet C++ aan de eisen om COM-poort ondersteuning en videoherkenning te hebben. En ten slot is C++ een bekende taal. Hierdoor zijn er meer mensen te vinden die antwoord kunnen geven op vragen. Als met een onbekende taal gewerkt wordt, nemen we het risico dat niemand weet waarom iets niet werkt.

Keuze voor C++:

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

## Operating system

Tijdens het onderzoek kwamen we tot de conclusie dat een operating system (vooral 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 OS te gebruiken is ook onderzocht.

Ook kunnen we ervoor kiezen om een RTOS te gebruiken. Deze zouden een hard real-time implementatie mogelijk moeten maken. Wat in principe raar is aangezien volgens de bron de hardware interrupts/BIOS instructies niet aangepast kunnen worden door het OS.

## 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 kracht

Tijdens het onderzoek is berekend dat een balletje op de hoogte van 50 cm van de tafel met een snelheid van 5 m/s een kracht nodig heeft van 0,0006N om naar de andere kant van het net te worden geslagen. Uit het onderzoek naar de robotarm is gebleken dat de motor waaraan het batje wordt bevestigd een kracht heeft van 1,10 Newton\*meter. Bij een batje van 25 cm is dit een kracht van 4,4 Newton. Er kan dus geconcludeerd worden dat de kracht van de robot hoog genoeg is om het balletje terug te kunnen slaan.

### 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.

## 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 waar geen 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 hij 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 7.1.4, is te zien dat behalve op het punt leercurve het realiseren van een eigen programma voor de controller de betere keuze. 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 zo snel mogelijk is. Doordat het zelf geschreven programma op de robot kan data kan verzenden zonder dat hier specifiek op 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.

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

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

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

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.

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

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

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

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

T, A. (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

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/

Ugal, A. S. (2009, 10 01). *Hard Real Time Linux\* using Xenomai\* on Intel® Multi-Core Processors.* Opgeroepen op 09 03, 2015, van http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/multicore-real-time-linux-xenomai-paper.pdf

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 11 Daily inspection items (details)

|  |  |  |  |
| --- | --- | --- | --- |
| 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 robot arm? | (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 12 Periodic inspection items (details)

|  |  |  |
| --- | --- | --- |
| Procedure | Inspection item (details) | Remedies |
| Monthly inspection items | | |
| 1 | Are any of the bolts or screws on the robot arm 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 robot arm. | 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. |