2015•2016
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: energie

Masterproef

Implementatie van robotcel voor het grijpen van 3D-delen van een transportband

Promotor :
dr. ir. Johan BAETEN

Promotor :
dr. ir. WIM PERSOONS

Bert Van Den Hende

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2015•2016
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: energie

Masterproef

Implementatie van robotcel voor het grijpen van 3D-delen van een transportband

Promotor :
dr. ir. Johan BAETEN

Promotor :
dr. ir. WIM PERSOONS

Bert Van Den Hende
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

universiteit hasselt | KU LEUVEN

Woord vooraf

Deze thesis is het laatste deel van mijn opleiding master in de industriële wetenschappen met afstudeerrichting automatisering die ik volg aan de universiteit UHasselt/KU Leuven.

Als eerste wil ik mijn externe promotor Wim Persoons en alle andere medewerkers van KUKA Automatisering + Robots N.V. bedanken voor het mogelijk maken van deze masterproef en voor hun technische ondersteuning.

Vervolgens wil ik mijn interne promotor Johan Baeten bedanken voor zijn technische en taalkundige ondersteuning bij deze masterproef.

Tot slot wil ik Niels Vandekerkhof, met wie ik het eerste semester van dit schooljaar heb samengewerkt, bedanken.

Bert Van Den Hende
Mei 2016

Inhoudsopgave

Lijst van figuren

Verklarende woordenlijst

GUI	Graphical User Interface
KRL	KUKA Robot Language, de programmeertaal die robotsturing gebruikt
RDC	Resolver Digital Converter
Resolver	Elektrische transformator die gebruikt wordt om de hoekpositie van een roterende as te meten.
TCP	Tool Center Point, het uiteinde van het robotgereedschap
XML	Extensible Markup Language, een opmaaktaal om gegevens gestructureerd te weergeven als platte tekst

Abstract

KUKA Automatisering + Robots NV wil onderzoeken of een laserlijnscanner boven een transportband kan functioneren als 3D-camera in een industriële robotcel. Eerst moet het visiesysteem van een voorgaande masterproef, waarbij een snelheidsterugkoppeling van de transportband nog ontbreekt, afgewerkt worden. Dit visiesysteem kan reeds bij constante bandsnelheden een 3D-beeld vormen en objecten van elkaar onderscheiden. De tweede taak is de implementatie van een robot die in staat moet zijn de ingescande objecten te volgen over de bewegende transportband. Het doel van deze masterproef is om te achterhalen hoe nauwkeurig, betrouwbaar en snel zulk een systeem is.

De visiesoftware van de scanner is aangepast om via de robotsturing de snelheidsmetingen van de transportband in te lezen en deze te gebruiken bij het vormen en herkennen van de 3D-beelden. Vervolgens is de robot zodanig geprogrammeerd dat hij het gescande object over de transportband volgt. Tot slot is de snelheid en nauwkeurigheid van de installatie experimenteel bepaald.

Uit de proeven blijkt dat bij bandsnelheden lager dan 1 m/s het visiesysteem in staat is objecten 180 keer op 180 te herkennen. De betrouwbaarheid van de objectherkenning neemt af bij hogere snelheden. De robot is in staat om de gescande objecten te volgen met 5 mm nauwkeurigheid. Het visiesysteem heeft minder dan een seconde nodig om een object in te scannen, een 3D-beeld te vormen, objectherkenning uit te voeren en om het resultaat naar de robot te sturen.

Abstract in English

KUKA Automatisering + Robots NV wants to investigate if a laserline scanner above a conveyor could function as a 3D-camera in an industrial robot cell. In a first step the vision-system from a previous master's thesis, where the speed-feedback of the conveyor hasn't been fully implemented, needs to be finished. This vision-system can already form a 3D-image and differentiate objects from each other at constant conveyor speeds. The second task is the implementation of a robot that is to follow objects over the moving conveyor. The goal of this master's thesis is to determine how accurate, reliable and fast such a system is.

The vision-software of the scanner has been modified to read the speed of the conveyor through the robot's controller and to use these measurements with the creation and recognition of the 3D-images. Then the robot has been programmed to follow the scanned object over the conveyor. Finally the speed and accuracy of the installation has been experimentally determined.

With tests at conveyor speeds lower than 1m/s the vision-system is able to recognize objects 180 out of 180. The reliability of the object-recognition decreases at lower speeds. The robot is capable of following the scanned objects with an accuracy of 5 mm. The vision-system needs less than a second to scan an object, form a 3D-image, recognize an object and to send the result to the robot.

1 Inleiding

Situering

Deze masterproef wordt uitgevoerd in samenwerking met KUKA Automatisering + Robots NV. Dit is een dochterbedrijf van het Duitse KUKA Systems GmbH en KUKA Roboter GmbH, één van de bekendste fabrikanten van industriële robots ter wereld. De Belgische vestiging in Houthalen is in 1985 opgestart met als doel de verkoop en service van industriële robots en opleidingen rond robotica. Tegenwoordig bouwt KUKA automatisering + robots N.V. ook zelfstandig complete robotcellen en installaties.

Nieuwe robottoepassingen maken steeds vaker gebruik van visie om de flexibiliteit te verhogen. Het herkennen van 3D-objecten is echter nog geen evidentie. Bovendien zijn nauwkeurige 3D-visiesystemen vaak behoorlijk duur en niet in eigen beheer volledig toe te passen. Met dit project wil KUKA Automatisering + Robots NV meer kennis verwerven op vlak van visiesystemen.

De hoofddoelstelling van deze masterproef is het realiseren van een systeem waarbij een industriële robot willekeurige 3D-delen op een bewegende transportband herkent en grijpt. Hiervoor moet de nodige beeldverwerkings- en controlesoftware geschreven worden.

De masterproef is in twee delen opgesplitst. Het eerste deel van de opdracht is het herkennen van objecten met behulp van een laserlijnscanner die als 3D-camera functioneert. Dit was ook meteen het onderwerp van de eerder dit jaar gerealiseerde masterproef van Niels Vandekerkhof [1]. In deze thesis is er aangetoond dat een laserlijnscanner kan functioneren als 3D-camera en is er een programma geschreven dat gescande objecten kan herkennen.

Probleemstelling/onderzoeksvraag

Omdat er nog geen snelheidsterugkoppeling beschikbaar was werkt het systeem in de thesis Niels Vandekerkhof enkel bij constante bandsnelheden [1]. Dit heeft tot gevolg dat er bij iedere aanpassing van de snelheid een manuele kalibratie nodig is. Deze manuele kalibratie heeft een grote onnauwkeurigheid in de meting tot gevolg.

Bovendien wil KUKA ook weten hoe zo'n visiesysteem presteert in een robotcel op vlak van snelheid, nauwkeurigheid en betrouwbaarheid.

Doelstellingen

Zoals eerder vermeld, is het volledig project opgesplitst in twee delen: het visiegedeelte en de robotimplementatie.

Voortbouwend op het reeds verwezenlijkte visiesysteem rest de realisatie van de snelheidsterugkoppeling zodat de beeldverwerking rekening kan houden met de actuele bandsnelheid. Hierbij moet achterhaald worden:

- hoe de snelheidsterugkoppeling gerealiseerd wordt;
- in welke mate de objectherkenning zal verbeteren door gebruik te maken van de correcte bandsnelheid bij het samenstellen van het 3D-beeld;
- hoe de betrouwbaarheid van de objectherkenning in relatie tot de bandsnelheid evolueert;
- hoeveel tijd de beeldverwerking in beslag neemt.

De volgende doelstelling is het integreren van het visiesysteem met een industriële robot. De robot moet in staat zijn de herkende objecten op een bewegende band te grijpen. Hiervoor moet er uitgezocht worden:

- hoe we een synchronisatie tussen de transportband, robot en de beeldverwerking bereiken;
- hoe groot de dode tijd in de communicatie tussen de verschillende componenten is;
- hoe groot de snelheid en nauwkeurigheid van het systeem is.

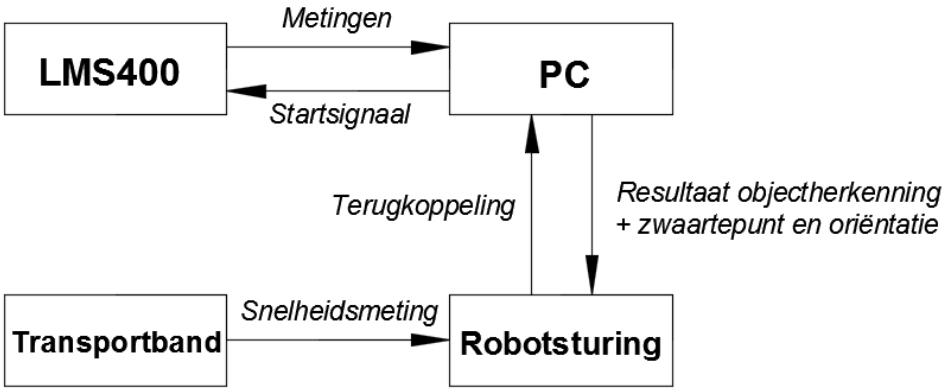
Deze doelstellingen moeten verwezenlijkt zijn binnen het tweede semester van het academiejaar 2015-2016.

Materiaal en methode

De robotcel bestaat uit drie hoofdcomponenten: een transportband, een laserafstandsscanner en een industriële robot. Op de transportband is een resolver gemonteerd die rechtstreeks met de robotsturing verbonden zal worden om met het softwarepakket ConveyorTech de robot met de transportband te synchroniseren.

De laserscanner is de LMS400 van SICK. Dit is een nauwkeurige en robuuste scanner. De visiesoftware kan de ingescande 3D-objecten herkennen [1]. Dit programma werkt op een pc en is geschreven in de programmeertaal C#, de standaard in het bedrijf.

Het programma wordt verder uitgebreid om in twee richtingen te communiceren met de robotsturing. De robotsturing stuurt de snelheidsmetingen van de transportband door naar de pc en geeft het startsignaal voor een nieuwe meting met de LMS400. De pc informeert de robotsturing wanneer een object herkend is, op welke positie het zich bevindt en hoe het georiënteerd is (zie figuur 1).



Figuur 1: Schema opstelling robotcel

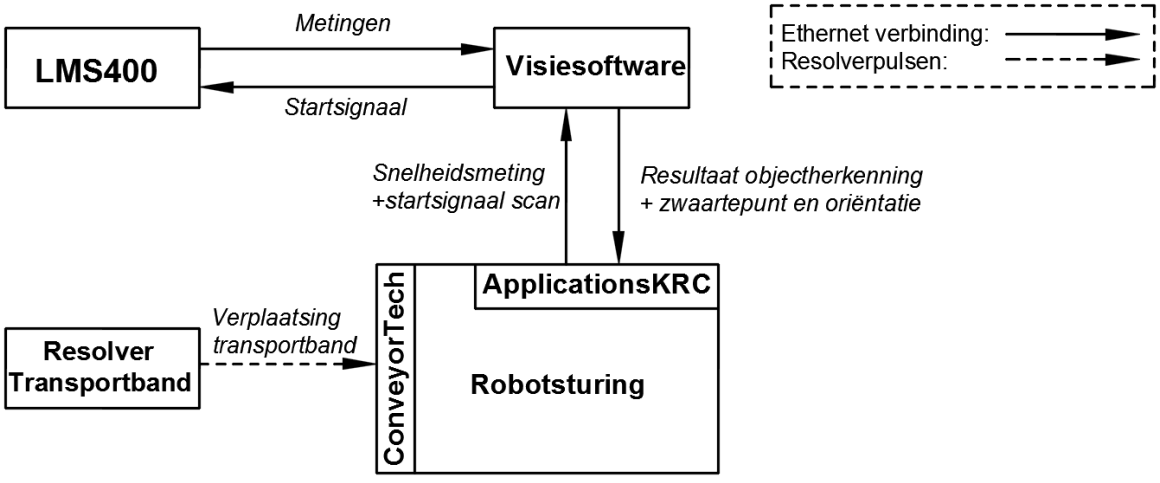
Vervolgens is er een robotprogramma geschreven dat de robot in staat stelt het gescande object te volgen over de bewegende transportband. Daarna is de snelheid en nauwkeurigheid van de installatie experimenteel bepaald.

2 Opstelling robotcel

De robotcel is opgebouwd uit drie onderdelen die met elkaar communiceren: de transportband, de LMS400 laserlijnscanner en een industriële robot (zie figuur 2). Dit hoofdstuk bespreekt kort deze drie onderdelen.

Figuur 2: Robotcel

Figuur 3 toont een schematisch overzicht van de robotcel. Dit schema geeft aan op welke manier de componenten met elkaar communiceren.



Figuur 3: Schema robotcel

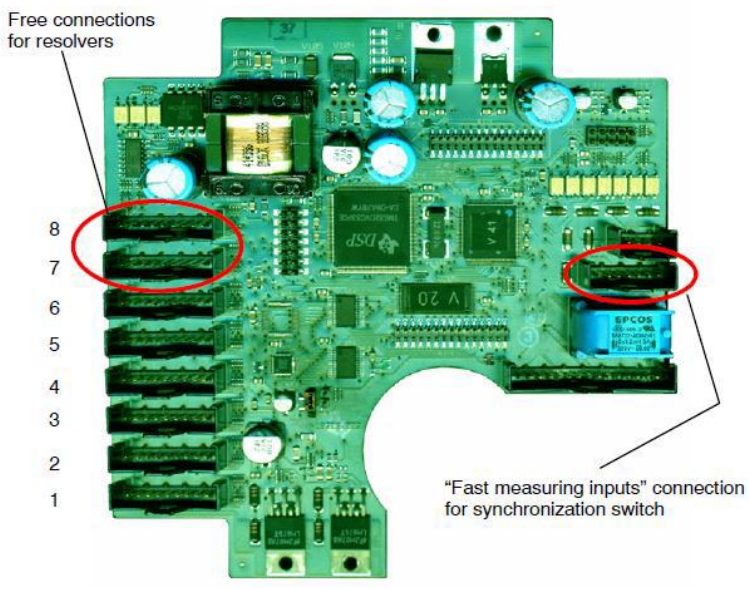
2.1 Transportband

De aandrijving van de transportband gebeurt met een asynchrone motor die door een frequentieregelaar is aangedreven. Het aanpassen van de snelheid gebeurt met een potentiometer op de stuurkast die de gebruiker handmatig kan verdraaien.

Aan het uiteinde van de transportband is een resolver gemonteerd (zie figuur 4). Deze dient om de positie van de transportband te meten zodat de robot synchroon met de band kan lopen.

Figuur 4: Resolver op de transportband

De resolver is verbonden met de RDC-kaart van de robot (zie figuur 5). Dit dient om de analoge signalen van de resolver om te zetten naar een digitaal signaal zodat de robot de afgelegde bandafstand kan inlezen.



Figuur 5: RCD-kaart robot [2]

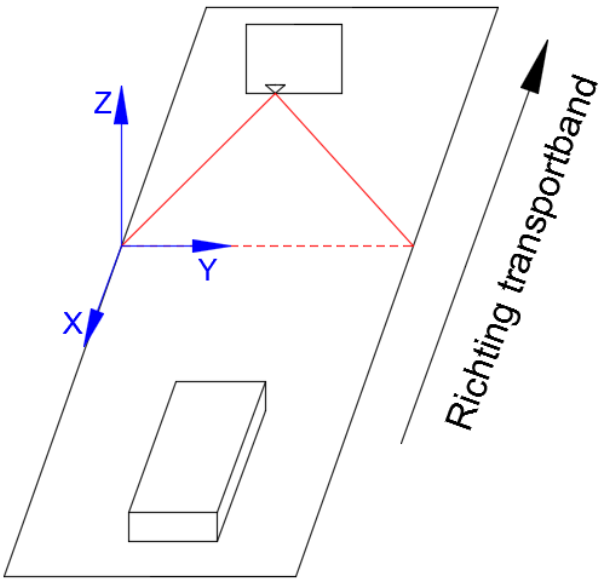
Op de RDC-kaart bevindt zich ook een snelle meetingang. Deze is in dit project gebruikt om de synchronisatieschakelaar in te lezen. Dit is nodig om de robotsturing een signaal te geven wanneer de bandsynchronisatie moet starten. Als schakelaar is de OPTEX WL160 "Photoelectric retro-reflective sensor" gekozen van het merk SICK (zie figuur 6) die vlak voor

de laserlijnscanner gemonteerd is. De schakelaar is met een harting stekker verbonden met de RDC-kaart.

Figuur 6: OPTEX WL160

2.2 Laserlijnscanner

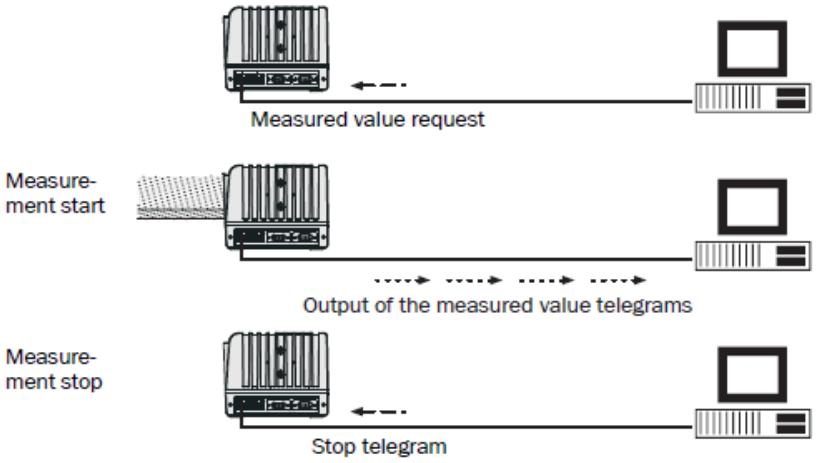
Als laserlijnscanner is de LMS400 van SICK gekozen. Dit is in feite een afstandsmeter die over de breedte van de transportband op 102 punten de hoogte meet. De visiesoftware kan een 3D-beeld vormen doordat objecten over de transportband bewegen. Figuur 7 toont hoe de LMS400 boven de transportband gemonteerd is.



Figuur 7: Opstelling van LMS400 (niet op schaal)

Met behulp van het programma SOPAS is het mogelijk om de parameters van de LMS400 op een eenvoudige manier aan te passen. De ingestelde parameters voor dit project zijn terug te vinden in bijlage A.

Het inlezen van de metingen gebeurt over een ethernet-interface. Zodra de LMS400 een starttelegram ontvangt zal het de metingen continu via telegrammen terugsturen met een frequentie van 178,9 Hz. Het is ook mogelijk om een meting te starten met een extern triggersignaal, dit is in dit project niet gebeurd. Zodra de LMS400 een stoptelegram ontvangt zal het stoppen met het continu doorsturen van data.



Figuur 8: Ethernetcommunicatie tussen LMS400 en pc [3]

2.3 Industriële robot

In figuur 9 staat de robot die in dit project gebruikt is. De aansturing van de robot gebeurt met een KR C2 stuurkast.

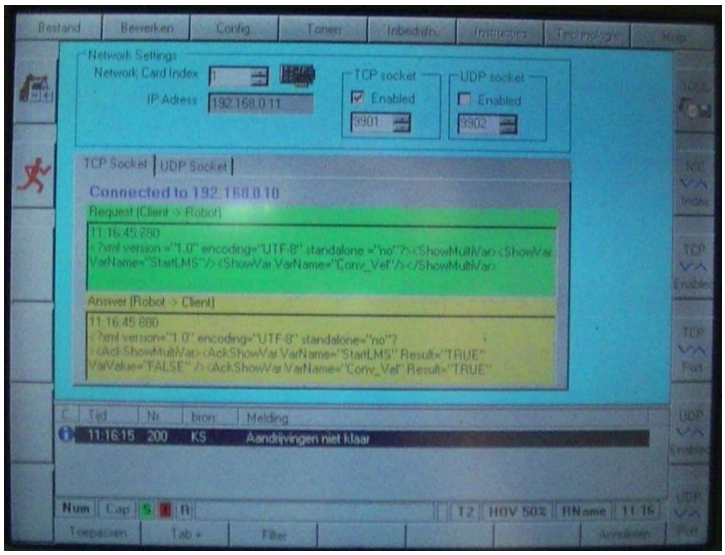
Figuur 9: Industrële robot

Voor dit project zijn er specifieke technologiepakketten nodig die geïnstalleerd moeten zijn. Om over een ethernetinterface te communiceren met de pc is het pakket “KUKA ApplicationsKRC” nodig. Om de robot te synchroniseren met de lopende band is het pakket “ConveyorTech” nodig. Dit laatste pakket maakt gebruik van de GUI van “UserTech”, dus dit moet ook geïnstalleerd zijn.

3 Ethernet communicatie met robot

3.1 Instellingen robot

Om een correct 3D-beeld te vormen moet de bandsnelheid gekend zijn. De robot leest de pulsen van de resolver in en berekent hiermee de snelheid van de transportband. De snelheidswaarde moet vervolgens ingelezen worden in de visiesoftware. In dit project is dit gedaan via ethernettelegrammen. Hiervoor is er gebruik gemaakt van sockets. Dit zijn eindpunten van netwerkverbindingen en worden gedefinieerd door een IP-adres en een IP-poort nummer. In de robotsturing kan je dit instellen via de “Jobmanager socket configuration” (zie figuur 10). Om gebruik te maken van socketcommunicatie moet het technologiepakket “applications” geïnstalleerd zijn op de robotsturing.



Figuur 10: Jobmanager socket configuration

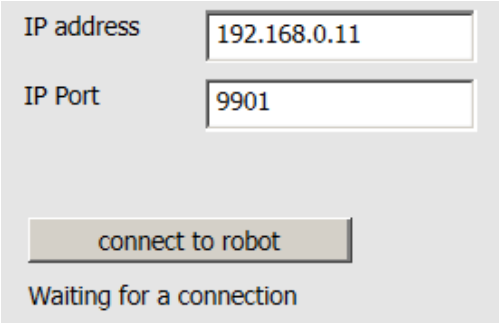
In de “Jobmanager socket configuration” kan de gebruiker instellen dat de robot een TCP-socket zal gebruiken. Het poortnummer is ingesteld op 9901. Het IP-adres is ingesteld op 192.168.0.11, dit kan de gebruiker enkel veranderen in de netwerkinstelling van de pc waarop de robotsturing draait. Vervolgens moet de gebruiker op de “filter”-knop drukken om in te geven welke KRL-variabelen via een extern programma gewijzigd en ingelezen mogen worden. De in te lezen variabelen zijn:
-Conv_Vel (de snelheid van de transportband),
-StartLMS (een boolean die de aangeeft dat de LMS400 moet starten met zijn scan).

De variabelen die naar de robot geschreven worden zijn:
-Xz (aangrijpingspunt in de X-richting),
-Yz (aangrijpingspunt in de Y-richting),
-Zz (aangrijpingspunt in de Z-richting),
-Angle (de hoekverdraaiing rond de Z-as),
-K (identificatienummer gescand object).

Al deze variabelen zijn gedeclareerd als globale variabelen in het “$config.dat”-bestand van de robot (zie bijlage B).

3.2 Instelling pc

Het IP-adres van de pc is ingesteld als 192.168.0.10, het is belangrijk om op te letten dat er geen twee dezelfde IP-adressen zijn in hetzelfde netwerk. De gebruiker moet het IP-adres en het poortnummer van de robot ingeven om een socketverbinding te kunnen starten. Figuur 11 toont het gebruikersinterface van de visiesoftware.



Figuur 11: Gebruikersinterface visiesoftware

Wanneer de gebruiker in het visieprogramma drukt op de knop “connect to robot” zal het de code in figuur 12 uitvoeren. Dit opent dan een nieuwe socketverbinding tussen de pc en de robot.

```
private void connectButtonRobot_Click(object sender, RoutedEventArgs e)
{
    //verbinding maken met de robot
    if (connectButtonRobot.Content.Equals("connect to robot"))
    {
        clientRobot = new TcpClient();
        clientRobot.BeginConnect(ipAddressRobot.Text, Int32.Parse(ipPortRobot.Text), connectedRobot, null);
    }
    else if (connectButtonRobot.Content.Equals("disconnect robot"))
    {
        ipAddressRobot.IsEnabled = true;
        ipPortRobot.IsEnabled = true;
        clientRobot.Close(); //verbinding verbreken
        connectButtonRobot.Content = "connect to robot";
        connectStatusRobot.Text = "disconnected";
    }
}
```

Figuur 12: Verbinding met robot maken

3.3 Structuur van robottelegrammen

De communicatie zal gebeuren met behulp van ethernettelegrammen die een XML-structuur hebben. De robotsturing herkent vier soorten telegrammen die betrekking hebben tot het lezen en schrijven van variabelen:

- ShowVar (om de waarde van één variabele op te vragen),
- ShowMultiVar (om de waardes van meerdere variabelen op te vragen),
- SetVar (om een nieuwe waarde naar één variabele te schrijven),
- SetMultiVar (om nieuwe waardes naar meerdere variabelen te schrijven).

In deze thesis zijn enkel de ShowmultiVar en de SetMultivar telegrammen gebruikt.

3.3.1 ShowMultiVar-telegram

In dit project is het ShowMultiVar gebruikt om de bandsnelheid en de startbit van de robot op te vragen. De code die in figuur 13 staat zal het programma iedere 100ms uitvoeren. De functie “InlezenVariabelenRobot()” zal de robotvariabelen effectief inlezen. Als de startbit van de robot actief is zal een nieuwe meting van de LMS400 starten.

```
//Methode die afloopt bij ieder timer-interval
private void TimerEventProcessor(Object myObject, EventArgs myEventArgs)
{
    myTimer.Stop();
    if(clientRobot.Connected == true && cyclischKRLVariabelenOpvragen.IsChecked == true)
    {
        InlezenVariabelenRobot();
    }
    if(scanActief == false && startBitRobot == true)
    {
        StartMetingLMS400();
    }
    myTimer.Enabled = true;
}
```

Figuur 13: Code die het programma cyclisch uitvoert

De eerste taak van de functie “InlezenVariabelenRobot()” is het definiëren van het XML-telegram. Figuur 14 toont de structuur van dit telegram. Het programma slaat dit telegram op als string. Vervolgens zal het programma deze string als ethernettelegram verzenden naar de robot.

```
<?xml version="1.0" encoding="UTF-8"?>
<ShowMultiVar>
    <ShowVar VarName="StartLMS"/>
    <ShowVar VarName="Conv_Vel"/>
</ShowMultiVar>
```

Figuur 14: XML-structuur ShowMultiVar-telegram

Nadat het telegram verzonden is zal de robot een telegram terugsturen met de waardes van de opgevraagde variabelen. De structuur van dit telegram is weergeven in figuur 15. Uit deze string kan vervolgens de toestand van de startbit en de snelheid van de transportband gefilterd worden.

```
<?xml version="1.0" encoding="UTF-8"?>
<AckShowMultiVar>
    <AckShowVar VarValue="FALSE"
        Result="TRUE"
        VarName="StartLMS"/>
    <AckShowVar VarValue="0.087"
        Result="TRUE"
        VarName="Conv_Vel"/>
</AckShowMultiVar>
```

Figuur 15: XML-structuur antwoordtelegram

3.3.2 SetMultiVar-telegram

Het SetMultiVar-telegram is in dit project gebruikt om de coördinaten van het berekende aangrijpingspunt van het gescande object, de hoek rond de Z-as en het resultaat van de objectherkenning door te sturen naar de robot. Het programma zal dit telegram verzenden zodra het klaar is met het matchingalgoritme. Figuur 16 toont de structuur van het telegram.

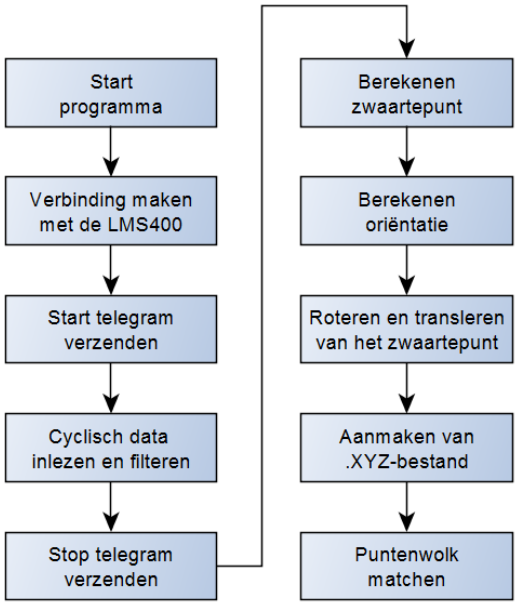
```
<?xml version="1.0" encoding="UTF-8"?>
<SetMultiVar>
    <SetVar VarValue="67.26"
        VarName="Xz"/>
    <SetVar VarValue="105.32"
        VarName="Yz"/>
    <SetVar VarValue="57" VarName="Zz"/>
    <SetVar
        VarValue="-60.35"
        VarName="Angle"/>
    <SetVar VarValue="1" VarName="K"/>
</SetMultiVar>
```

Figuur 16: XML-structuur SetMultiVar-telegram

4 Visiesoftware

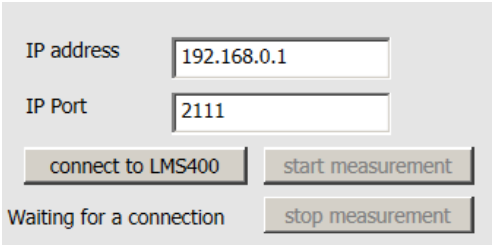
4.1 Bestaande software

Zoals eerder vermeld bouwt deze thesis verder op een eerder verwezenlijkte programma waarbij een snelheidsterugkoppeling ontbrak, figuur 17 toont een schematisch overzicht hiervan. Dit hoofdstuk bevat een korte samenvatting van dit programma. Een meer diepgaande uitleg is terug te vinden in de thesis van Niels Vandekerkhof [1].



Figuur 17: Visiesoftware Niels Vandekerkhof

Als eerste moet een socketverbinding gemaakt worden tussen de LMS400 en de pc. Hiervoor moet het IP-adres en de IP-poort van de LMS400 gekend zijn. Wanneer er verbinding gemaakt is kan de gebruiker op de “start measurement”-knop drukken om een starttelegram naar de LMS400 te sturen. Vervolgens zal de LMS400 cyclisch metingen doorsturen naar de PC met een frequentie van 179,8 Hz. De pc zal de data inlezen en filteren. De LMS400 zal cyclisch data blijven versturen tot het een stoptelegram ontvangt. Dit kan ofwel manueel gebeuren door op de stopknop te drukken of automatisch wanneer de scanner een bepaalde afstand geen object meer waarneemt. Figuur 18 toont de bediening van de communicatie met de LMS400.



Figuur 18: Bediening communicatie LMS400

Wanneer alle data is ingelezen en opgeslagen zal het programma de zwaartepuntscoördinaten en de oriëntatie van het ingescande voorwerp berekenen. Hiervoor is er gebruik gemaakt van de volgende formules:

$$Xz = \frac{\sum(m_i \,.\, x_i)}{\sum m_i}$$

$$Yz = \frac{\sum(m_i \,.\, y_i)}{\sum m_i}$$

$$\alpha = \frac{1}{2} \,.\, tan^{-1}\left(\frac{2.I_{xy}}{I_{yy} - I_{xx}}\right)$$

Waarbij: Xz = zwaartepuntscoördinaat in de X-richting
Yz = zwaartepuntscoördinaat in de Y-richting
m_i = massa van meetpunt i (ieder meetpunt heeft dezelfde massa die gelijk is aan 1)
x_i = X-coördinaat van meetpunt i
y_i = Y-coördinaat van meetpunt i
α = oriëntatie van het object
I_{yy} = traagheidsmoment t.o.v. de Y-as
I_{xx} = traagheidsmoment t.o.v. de X-as
I_{xy} = traagheidsproduct

De volgende stap is het vormen van het 3D-beeld. Het programma zal de meetdata gebruiken om een XYZ-bestand aan te maken en op te slaan. Dit bestand bevat de puntenwolk van het ingescande object. Figuur 19 toont een voorbeeld van een hamer die is ingescand.

Figuur 19: Puntenwolk van een hamer

Nadat de puntenwolk is opgeslagen kan het programma deze via een algoritme vergelijken met een eerder opgeslagen puntenwolk om objecten te kunnen herkennen. Als dit algoritme klaar is zal het een “fitness”-score geven. Een score van 0 wil zeggen dat de twee puntenwolken een perfecte match zijn. Hoe groter de score, hoe meer de puntenwolken van elkaar afwijken. Een score van 10 is bij deze toepassing een aanvaardbare score voor een match.

4.2 Implementatie snelheidsterugkoppeling

In het programma dat geschreven is door Niels Vandekerkhof is er nog geen snelheidsterugkoppeling aanwezig. De bandsnelheid is steeds handmatig gemeten en ingevoerd in het programma. De bedoeling van deze thesis is dat robot de actuele bandsnelheid berekent zodat de visiesoftware deze vervolgens kan gebruiken bij het vormen van de puntenwolk. In hoofdstuk 3.3.1 is eerder al uitgelegd hoe het programma cyclisch de snelheid van de transportband opvraagt.

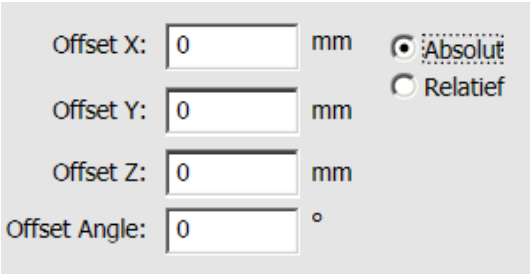
Zoals eerder vermeld zal de LMS400 tijdens het scannen om de 5,2 ms een nieuwe meting doorsturen. Bij iedere scan berekent het programma aan de hand van de laatst ingelezen snelheid de afstand die de transportband sinds de vorige scan heeft afgelegd. Deze afstanden worden opgeslagen in een lijst. Bij het vormen van de puntenwolk gebruikt het programma deze lijst.

4.3 Berekening van de door te sturen variabelen

Zoals eerder vermeld zullen er vijf variabelen doorgestuurd worden naar de robot:

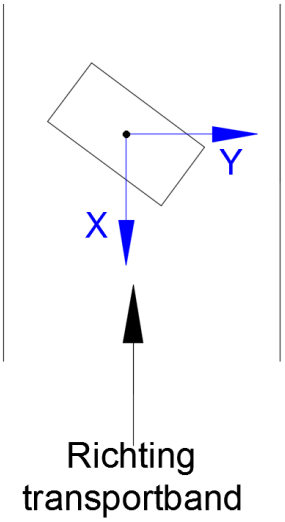
- het X-coördinaat van het aangrijpingspunt,
- het Y-coördinaat van het aangrijpingspunt,
- het Z-coördinaat van het aangrijpingspunt,
- de hoek rond de Z-as,
- Het identificatienummer K.

Het programma berekent de coördinaten van het aangrijpingspunt en de oriëntatie van het object. De gebruiker kan echter ook nog een offset meegeven. Dit kan op 2 manieren: met absolute of relatieve coördinaten. Figuur 20 toont hoe de gebruiker dit in de visiesoftware kan instellen.



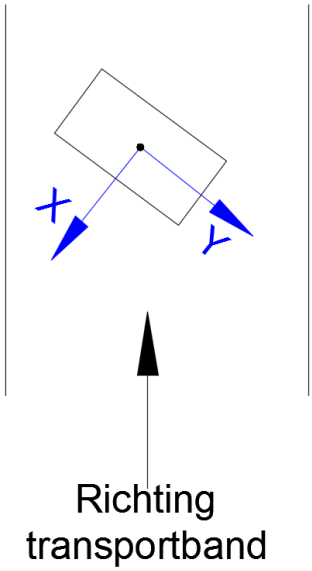
Figuur 20: Gebruikersinterface offset

De absolute offset wil zeggen dat het X-en Y-coördinaat van het aangrijpingspunt verschoven wordt ten opzichte van een niet-geroteerd assenstelsel dat meebeweegt met de transportband (zie figuur 21). Het aangrijpingspunt zal dus verschuiven t.o.v. het zwaartepunt, onafhankelijk van de oriëntatie.



Figuur 21: Offset t.o.v. een absoluut assenstelsel

Bij de relatieve offset zal het assenkruis verdraaien met de berekende hoek. Dit zorgt ervoor dat het aangrijpingspunt altijd op een specifiek punt van het object kan aangrijpen. Figuur 22 toont dit assenstelsel.



Figuur 22: Offset t.o.v. een relatief assenstelsel

De Z-coördinaat die de visiesoftware doorstuurt is die van het hoogst gedetecteerd punt van de puntenwolk om zeker te zijn dat het gereedschap van de robot het object niet raakt. De doorgestuurde hoek is de hoek die eerder berekend is. De gebruiker kan ook deze twee variabelen een offset meegeven.

De laatste variabele die de visiesoftware zal doorsturen naar de robot is het identificatienummer K. Dit nummer geeft het resultaat van het matching algoritme. Dit is een integer waarvan de waarde de volgende betekenis heeft:

K	Betekenis
-1	Geen object herkend
0	Scan nog niet klaar
1	Object 1 herkend
2	Object 2 herkend
…	…

De gebruiker heeft de mogelijkheid om verschillende objecten te herkennen en deze een uniek identificatienummer te geven zodanig dat het robotprogramma verschillende handelingen kan uitvoeren voor verschillende onderdelen. In dit project is er steeds maar van één object gebruik gemaakt.

Het volledig programma is terug te vinden in bijlage C.

5 Robotimplementatie

5.1 Inbedrijfname robot

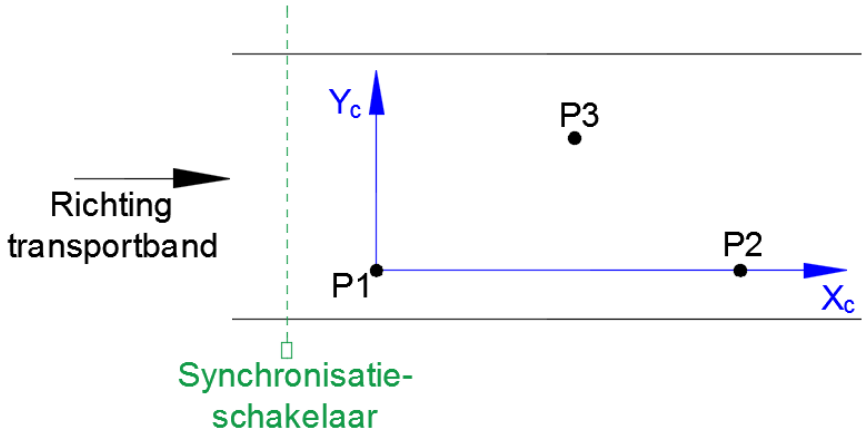
In dit project is het technologiepakket “ConveyorTech” gebruikt om de robot met de transportband te synchroniseren. Om een bandsynchronisatie uit te kunnen voeren moet eerst de transportband en de partbase aangeleerd worden. Hiervoor is een KRL-programma voorzien genaamd “conv_msr”. Om dit te kunnen gebruiken moet de robot in stand T1 staan en moet er een tool gedefinieerd zijn. De tool in dit project is een draadstang met een scherpe punt. Deze is gedefinieerd met de vier-punt methode.

Bij het starten van “conv_msr” zal de gebruiker moeten aanduiden of hij een transportband of een partbase wil opmeten. Beide procedures zijn in de volgende hoofdstukken uitgelegd.

5.1.1 Opmeten van transportband

Het programma “conv_msr” zal eerst vragen welke transportband de gebruiker wil opmeten. De robotsturing is in staat om drie verschillende transportbanden te volgen. In dit project is er maar één transportband dus is er gekozen voor transportband 1. Vervolgens vraagt het programma of dit een lineaire of een cirkelvormige transportband is, in dit geval is het een lineaire transportband.

Bij de volgende stap zegt het programma dat het aan het wachten is op het synchronisatiesignaal. Dus moet de gebruiker een object op de transportband leggen en deze vervolgens voorbij de synchronisatieschakelaar laten gaan. Zodra de transportband terug gestopt is zal het programma vragen dat de gebruiker het TCP naar het eerste referentiepunt verplaatst. Dit mag een willekeurig punt op de band zijn. Zodra de gebruiker dit punt heeft aangeleerd moet het tweede referentiepunt P2 aangeleerd worden. Dit punt moet de positieve X-as van de transportband aangeven. Tot slot moet de gebruiker P3 aanleren. Deze zal de oriëntatie van het XY-vlak bepalen. Figuur 23 toont de drie punten.



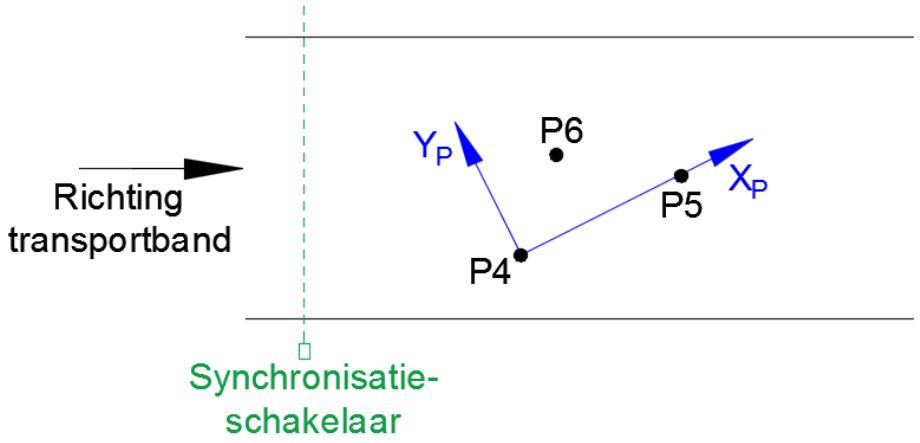
Figuur 23: Assenstelsel transportband

5.1.2 Opmeten van partbase

Het opmeten van de partbase is nodig om een assenstelsel te definiëren dat meebeweegt met de transportband. Dit assenkruis dient als basis voor de coördinaten van het berekend zwaartepunt van het ingescande object.

Om de partbase op te meten moet de gebruiker opnieuw het programma “conv_msr” openen en dit keer kiezen voor het opmeten van de partbase. Nu moet de gebruiker opnieuw kiezen voor band 1. Vervolgens zal het programma wachten op de synchronisatiepuls. De gebruiker moet nu een object over de transportband laten lopen. Wanneer de band terug stopt zal het programma de berekende afstand tonen en vragen of dit overeenkomt met de werkelijk afgelegde afstand. Als dit niet het geval is moet de gebruiker de correcte afstand ingeven. Dan herhaalt dit proces zich weer totdat de gebruiker aangeeft dat de berekende afstand correct is. De robot heeft nu de correcte verhouding tussen resolverpulsen en de bandafstand opgeslagen. In dit project is deze verhouding 0,005891.

Vervolgens vraagt het programma om het TCP naar het referentiepunt P4 te verplaatsen. Dit is de oorsprong van de partbase. Vervolgens moet de gebruiker P5 aanleren, dit geeft de positieve X-richting aan. Ten slotte zal de gebruiker met P6 het XY-vlak definiëren. Figuur 24 toont deze drie punten op de transportband.



Figuur 24: Partbase assenstelsel

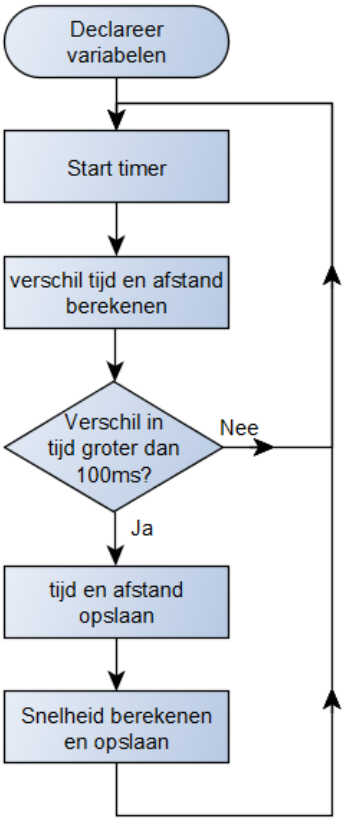
Nu de transportband en de partbase zijn opgemeten is het mogelijk om een programma te schrijven waarmee de robot een object over de bewegende transportband kan volgen.

5.2 Robotprogramma

Het robotprogramma bestaat uit twee delen. Het eerste deel dient om cyclisch de snelheid van de transportband te berekenen. Het tweede deel dient om de gesynchroniseerde robotbewegingen uit te voeren.

5.2.1 Berekening snelheid

De resolver van de transportband meet enkel de verplaatsing van de transportband. Daarom moet de robotsturing de snelheid nog berekenen. Deze berekening is uitgevoerd in het “sps.sub”-bestand (ook gekend als de “submit interpreter”) dat terug te vinden is in de systeemmap van de robotsturing. De robot voert dit programma cyclisch uit op de achtergrond met een periode van ±12 ms. Het schema van het gebruikersprogramma is weergeven in figuur 25.



Figuur 25: Schema voor het berekenen van de snelheid

Als eerste moet de gebruiker de nodige variabelen declareren. In dit project zijn de volgende variabelen gedeclareerd:

- REAL distDiff (verschil in afstand),
- REAL distMem (opgeslagen afstand),
- INT timeDiff (verschil in tijd),
- INT timeMem (opgeslagen tijd).

Vervolgens is de timer gestart, in dit project is timer 10 gekozen. Dan berekent het programma het verschil tussen de huidige en de voorheen opgeslagen timerwaarde. Hetzelfde gebeurt voor de bandafstand. Dan controleert het programma of het verschil in timerwaarde groter is dan 100 ms, als dit het geval is berekent het programma de snelheid en slaat het de huidige tijd en bandafstand op. Het programma schrijft de snelheid naar een globale variabele genaamd “Conv_Vel”. Als het verschil in tijd niet groter is dan 100 ms zal het programma wachten totdat dit wel het geval is. Dit heeft tot gevolg dat het programma de

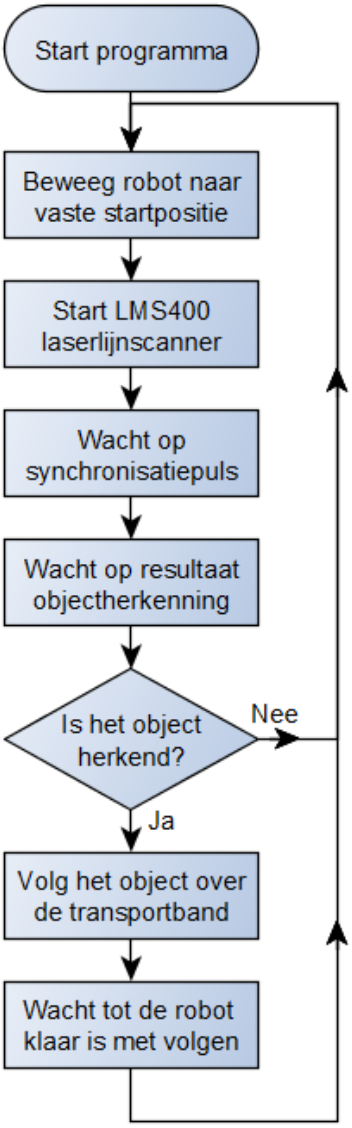
snelheid berekent in intervallen van meer dan 100 ms, dit dient om de snelheid nauwkeurig te kunnen berekenen. Het volledig programma is terug te vinden in bijlage D.

5.2.2 Hoofdprogramma

In het hoofdprogramma zal de robot zijn gesynchroniseerde bewegingen uitvoeren. Hiervoor zijn de commando's van het technologiepakket ConveyorTech nodig. Om deze commando's te kunnen gebruiken moet de gebruiker bij het aanmaken van het programma kiezen voor het sjabloon "conveyor". De commando's zijn:

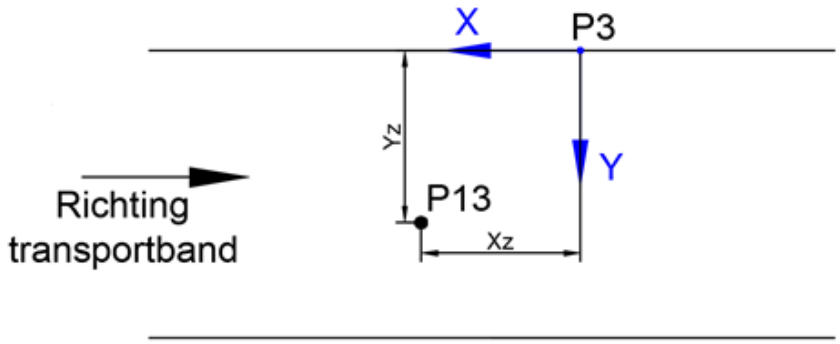
- CONV_INI_OFF() : initialiseert de transportband,
- CONV_ON(): zorgt dat het programma wacht op de synchronisatiepuls,
- CONV_FOLLOW(): laat de robot aangeleerde punten volgen op de transportband,
- CONV_QUIT(): subprogramma dat de robot uitvoert bij abnormale situaties.

De structuur van het hoofdprogramma is te zien in figuur 26.



Figuur 26: Schema hoofdprogramma

Als eerste zal het programma de nodige variabelen initialiseren om zeker te zijn dat “StartLMS” gereset is. Vervolgens opent het programma een lus die als eerste de variabele “K” reset. Dan zal het de coördinaten van punt 3 schrijven naar punt 13. Punt 3 is de vaste oorsprong van een denkbeeldig assenkruis dat meebeweegt met de band (zie figuur 27). Punt 13 is een variabel punt dat de robot zal volgen over de transportband.



Figuur 27: Punt 3 en punt 13 op de transportband

Zodra de initialisatie voltooid is zal de robot bewegen naar punt 2, wat een vast punt boven de transportband, vlak bij de scanner, is. Dit punt dient als vertrekpunt. Zodra dit punt bereikt is maakt het programma de merker “StartLMS” hoog. De pc zal dit inlezen en zal vervolgens een meting van de LMS400 starten.

Vervolgens initialiseert het programma de transportband met het “CONV_INI_OFF()”-commando. Dit reset de verplaatste bandafstand. Meteen daarna voert het programma het “CONV_ON()”-commando uit. Hierdoor zal het programma wachten totdat een voorwerp de synchronisatieschakelaar passeert. Eens dit gebeurd is zal het programma “CONV_FOLLOW()” uitvoeren, dit zal een subprogramma oproepen waarin de robot gesynchroniseerde bewegingen met de band kan uitvoeren.

Het eerste dat het subprogramma doet is de merker “StartLMS” terug laag maken om te voorkomen dat een nieuwe meting start voordat de robot klaar is met zijn beweging. Vervolgens wacht het programma tot K niet langer gelijk is aan 0, dit geeft aan dat de scan en de objectherkenning voltooid is en dat de pc de coördinaten van het berekende aangrijpingspunt doorgestuurd heeft. Als K gelijk is aan -1 wil dat zeggen dat het object niet herkend is en dat de lus opnieuw begint.
Wanneer K gelijk is aan 1 is het object wel herkend en zullen de berekende coördinaten van het object opgeteld worden bij de coördinaten van punt 13. Vervolgens zal de robot punt 13 volgen over de transportband. Wanneer de transportband een totale afstand van 1,5 m heeft afgelegd is het subprogramma klaar en zal de loop opnieuw beginnen.

In bijlage E staat het volledige hoofdprogramma.

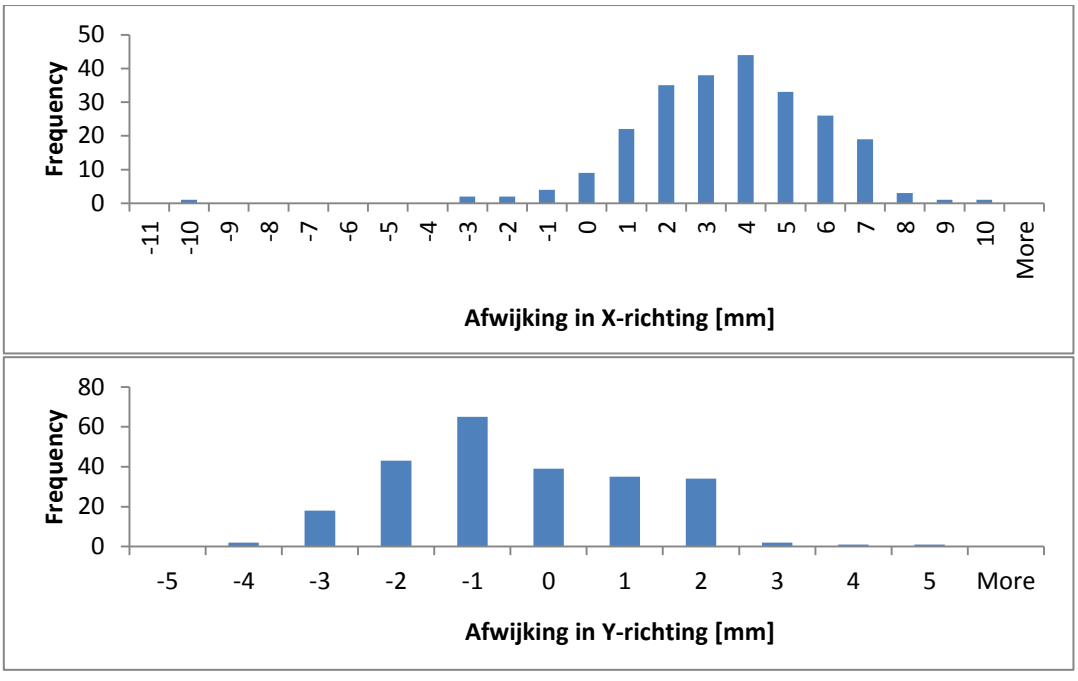
6 Testresultaten

Om te achterhalen hoe nauwkeurig, snel en betrouwbaar het systeem is zijn er een reeks proeven uitgevoerd bij een bandsnelheid die toeneemt in stappen van 0,1 m/s. Bij iedere snelheidsstap is twintig keer en houten blokje met gekend zwaartepunt willekeurig op de transportband geplaatst (zie figuur 28). Nadat het blokje de scanner passeert zal het TCP van de robot naar het zwaartepunt van het blokje bewegen. Dan is de transportband stilgelegd en is de afwijking in de X- en Y-richting gemeten met een schuifmaat. In het visieprogramma is de fitness-score en de tijdsduur van de scan te zien.

Figuur 28: Houten blok

6.1 Nauwkeurigheid

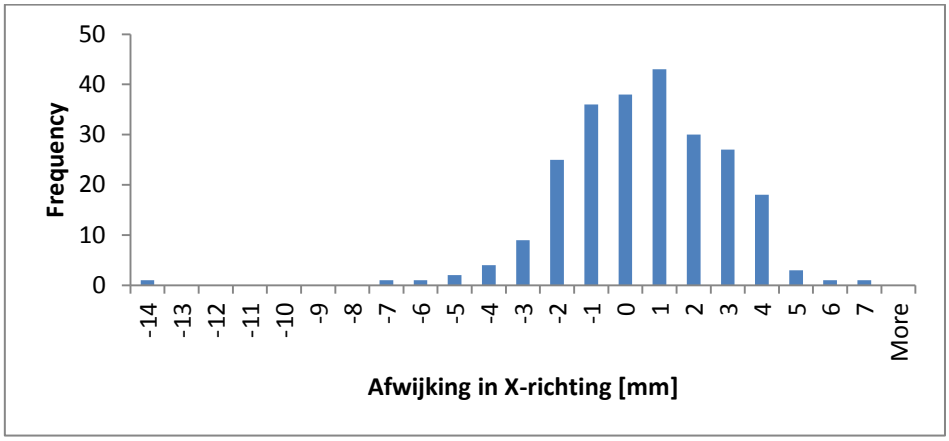
Figuur 29 toont een histogram van iedere gemeten afwijking. In de X-richting bedraagt de afwijking gemiddeld 3,1 mm in de positieve zin. In de Y-richting is dit -0,9 mm. Uit testen blijkt dat deze afwijkingen onafhankelijk zijn van de bandsnelheid of de hoogte van het object.



Figuur 29: Afwijking in de X- en Y-richting

De standaarddeviatie voor de afwijking in de X-richting is 2,4 mm. In de Y-richting is dit 1,6 mm.

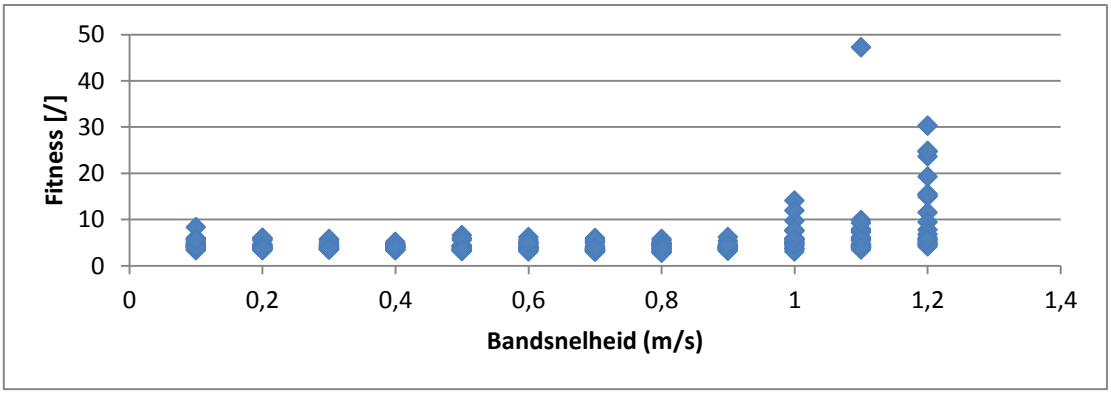
Door een constante offset te programmeren is het mogelijk de histogram te verschuiven zodanig dat de variabele afwijking beter rond het nulpunt komt te liggen (zie figuur 30).



Figuur 30: Histogram met offset in X-richting

6.2 Betrouwbaarheid

In figuur 31 is de verhouding tussen de fitness-score en de bandsnelheid te zien. Bij snelheden lager dan 1 m/s blijft deze score steeds kleiner dan tien. Bij hogere snelheden zal het resultaat steeds onbetrouwbaarder worden. De gemiddelde score bij snelheden onder de 1 m/s bedraagt 4,23.

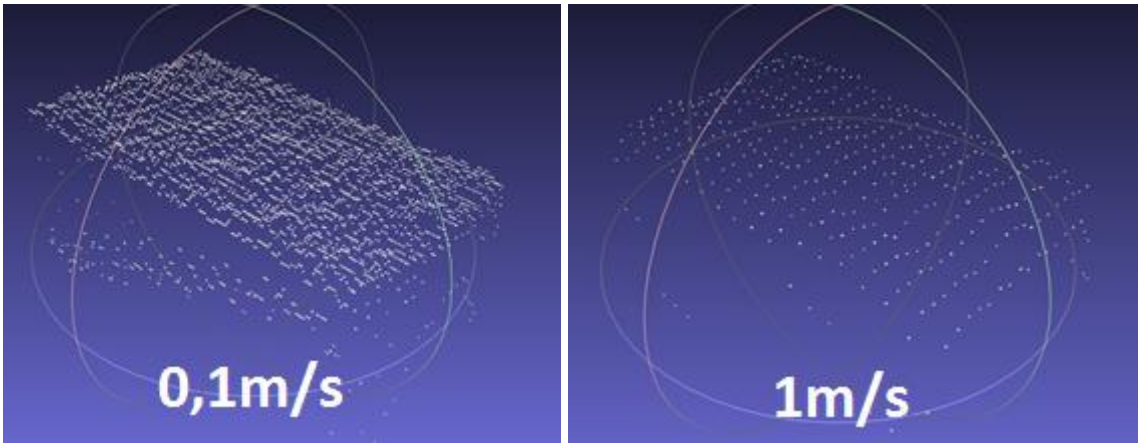


Figuur 31: Fitness-score i.f.v. de bandsnelheid

Dit betekent echter niet dat de objectherkenning onbruikbaar is bij hogere snelheden. Bij objecten met compleet verschillende afmetingen is de fitness-score, zelfs bij hogere snelheden, ruim boven de 300. Met andere woorden, zolang de vorm en afmetingen van de ingescande objecten voldoende van elkaar verschillen kan het systeem ze op zeer betrouwbare wijze van elkaar onderscheiden.

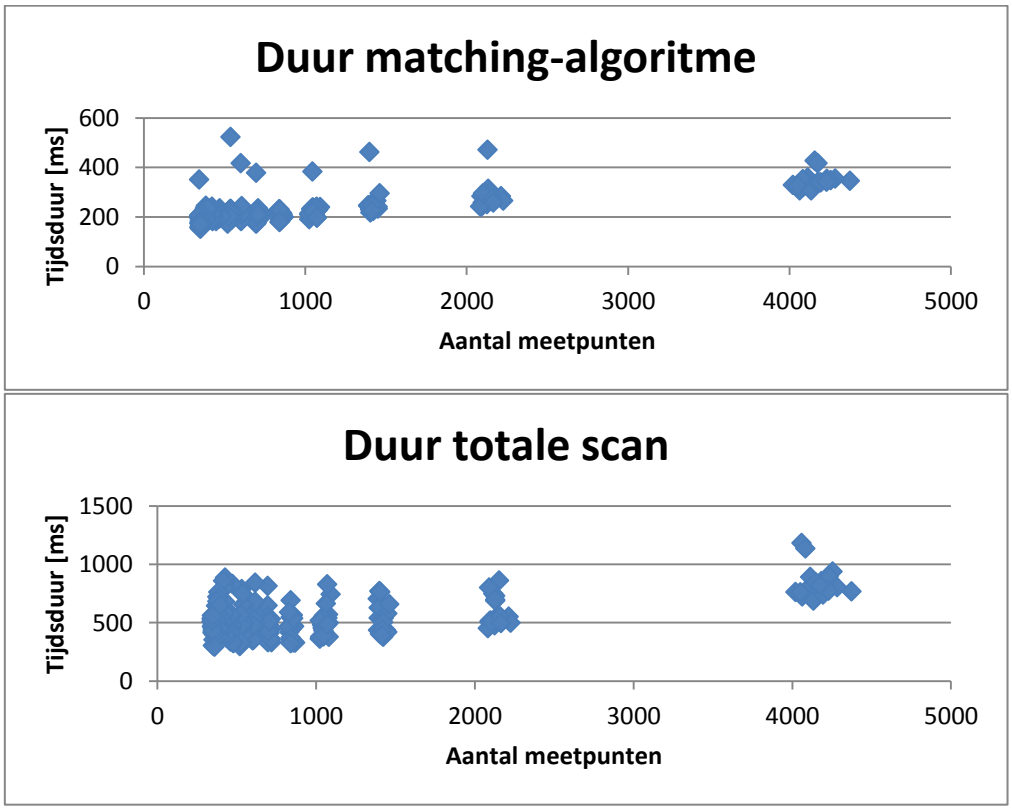
6.3 Snelheid visiesoftware

Hoe groter de puntenwolk is, hoe meer tijd het programma nodig heeft om de puntenwolk te vormen en te vergelijken. Aangezien de scanfrequentie van de LMS400 constant is, zal de grootte van de puntenwolk enkel afhankelijk zijn van de snelheid van de transportband (zie figuur 32).



Figuur 32: Puntenwolken bij verschillende bandsnelheden

Bij de proeven is de tijdsduur van het matchingalgoritme en de totale tijd die de scanner nodig heeft gemeten (d.w.z. vanaf het moment dat het object de scanner raakt totdat de visiesoftware zijn berekende waardes doorstuurt naar de robot). Figuur 33 toont deze tijdsduren in functie van het aantal meetpunten.

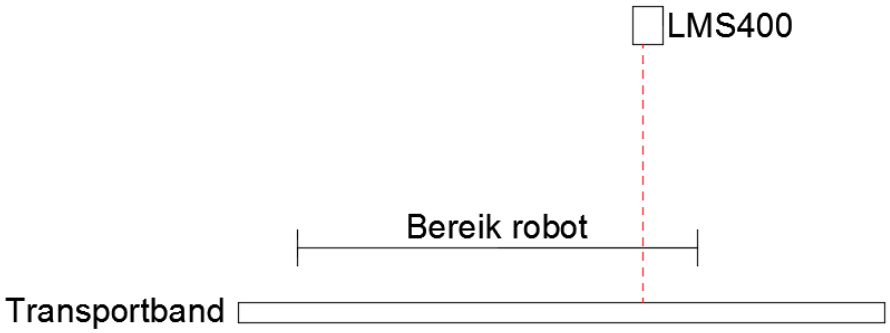


Figuur 33: Tijdsduur beeldverwerking

De totale duur van de scan blijft in de meeste gevallen kleiner dan 1 s. Enkel bij zeer lage bandsnelheden kan het langer dan 1 s duren maar als dit het geval is heeft de robot meer dan genoeg tijd om het object te bereiken.

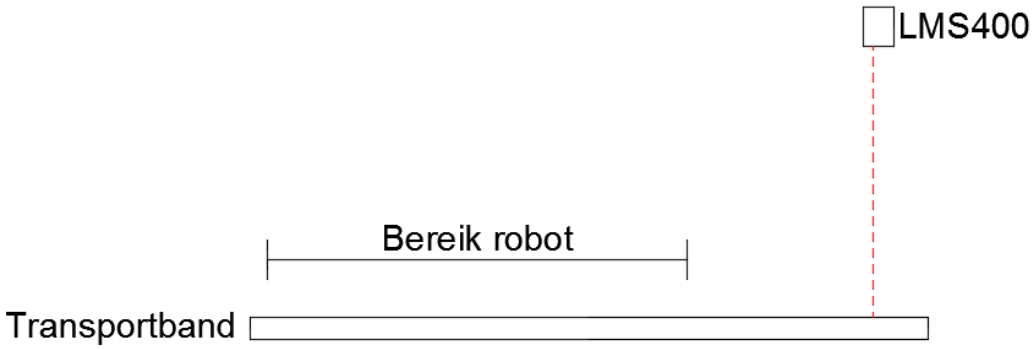
In dit project vergelijkt de visiesoftware de scan met slechts één opgeslagen puntenwolk. Het is mogelijk om de scan met meerdere verschillende puntenwolken te vergelijken, het programma zal in dat geval kijken welk van de opgeslagen puntenwolken het beste overeenkomt met het ingescande object. Wanneer het programma meerdere objecten met elkaar vergelijkt zal de beeldverwerking uiteraard meer tijd in beslag nemen.

In de huidige opstelling is de snelheid van de visiesoftware een zeer belangrijke factor aangezien de scanner zich in het werkingsgebied van de robot bevindt (zie figuur 34). Dit wil zeggen dat de robot minder tijd heeft om een stuk te grijpen omdat de robot nog moet wachten tot de visiesoftware klaar is.



Figuur 34: Scanner binnen het werkgebied van de robot

Als de scanner ver genoeg verwijderd is van de robot dan kan de verwerkingstijd van de visiesoftware irrelevant zijn (zie figuur 35). Dit heeft als voordeel dat het volledig werkingsgebied van de robot bruikbaar is.



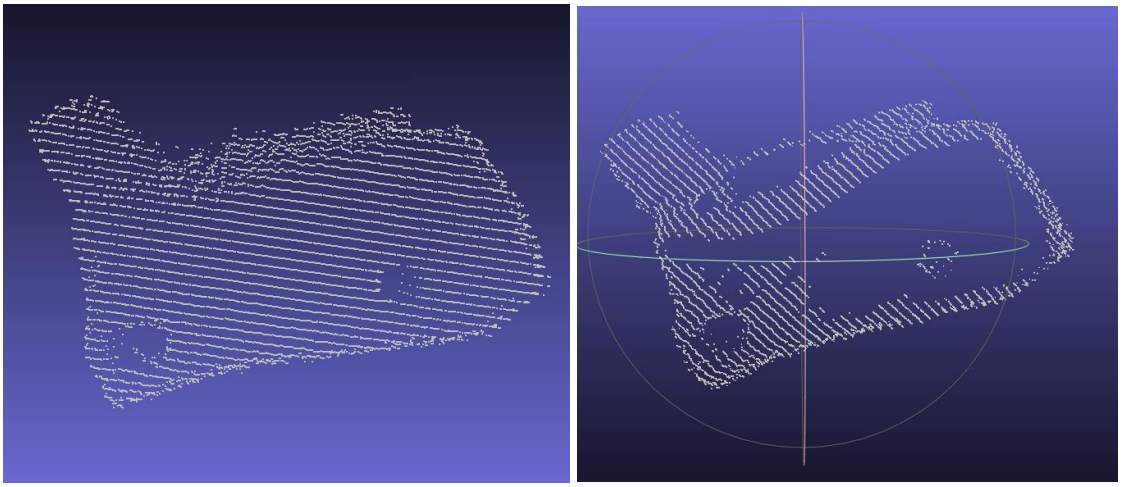
Figuur 35: Scanner buiten het werkgebied van de robot

7 Besluit

Uit deze masterproef blijkt dat een LMS400 laserlijnscanner kan functioneren als 3D-camera. In hoofdstuk 6.1 is reeds aangetoond dat de standaarddeviatie in de X- en Y-richting respectievelijk 2,4 en 1,6 mm bedraagt. Dit wil zeggen dat het systeem niet geschikt is voor toepassingen die een hoge nauwkeurigheid vereisen. Het systeem zou echter zeer geschikt zijn voor pick-and-place toepassingen waarbij objectherkenning nodig is aangezien de snelheid, oriëntatie en positie op de band weinig tot geen invloed hebben op de nauwkeurigheid of de objectherkenning. Dit systeem is niet geschikt voor zeer kleine objecten zoals bijvoorbeeld bouten en moeren. In dit geval zijn er te weinig meetpunten om een goede objectherkenning uit te voeren. Bij grote voorwerpen moet de gebruiker erop letten dat het object volledig binnen het bereik van de LMS400 zit. Als dit niet het geval is kunnen er meetpunten verloren gaan wat een grote afwijking tot gevolg kan hebben.

Het programma is in eigen beheer gemaakt, dit wil zeggen dat er altijd aanpassingen mogelijk zijn. Door de visiesoftware uit te breiden is het mogelijk om meerdere verschillende objecten van elkaar te onderscheiden en deze een uniek identificatienummer toe te kennen zodat de robotsturing zich eventueel hieraan kan aanpassen. Bovendien is het mogelijk om een offset te programmeren zodanig dat de robot het object op een specifiek punt kan grijpen.

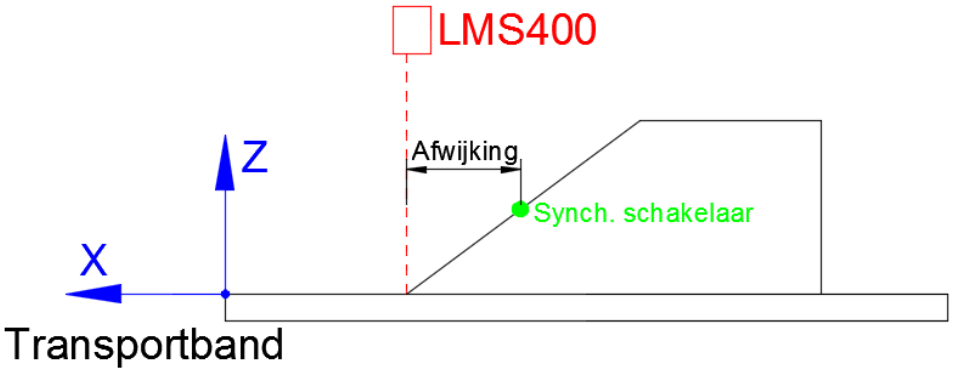
Het systeem heeft echter nog een aantal beperkingen. Als de gebruiker meerdere objecten gelijktijdig op de transportband legt zal de visiesoftware dit zien als één object dus kan het systeem voorlopig maar één object tegelijk herkennen. Bovendien kan de LMS400 problemen krijgen bij het inscannen van reflecterende oppervlaktes. Dit zorgt ervoor dat er meetpunten verloren gaan wat een grote fout bij het berekenen van het zwaartepunt en oriëntatie tot gevolg heeft (zie figuur 36).



Figuur 36: Voorbeeld verlies van meetpunten bij reflecterende oppervlaktes

Bovendien is er ook nog een probleem wanneer de randen van het object niet loodrecht op de transportband staan. Figuur 37 toont aan dat het mogelijk is dat de synchronisatieschakelaar en de laserlijnscanner op een verschillend punt het object

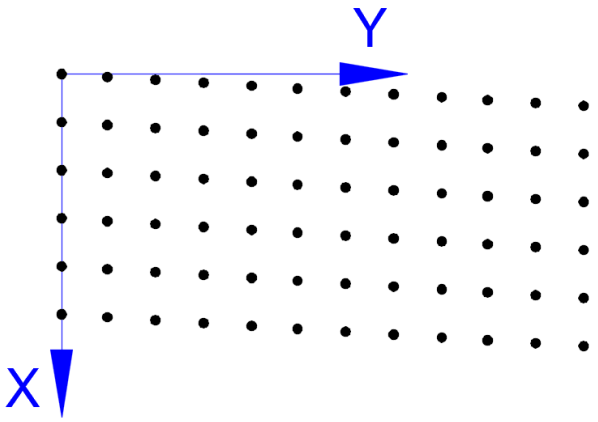
aangrijpen. Dit zorgt ervoor dat er een verschuiving in de X-richting ontstaat tussen de partbase van de robot en het assenstelsel van de puntenwolk.



Figuur 37: Afwijking door verschuiving 3D-beeld en partbase robot

Als de grootte van deze verschuiving voor iedere mogelijke oriëntatie gekend is kan de gebruiker dit oplossen met een eenvoudige offset. Een andere mogelijke oplossing is het gebruiken van een lichtgordijn met fijne vermazing als synchronisatie schakelaar.

De LMS400 meet de hoogte op 102 punten in de breedte van de transportband. De scanner meet deze punten niet gelijktijdig op, het meet ieder punt één voor één. In deze masterproef is er steeds vanuit gegaan dat bij iedere scan van 102 meetpunten alle meetpunten dezelfde X-waarde hebben. In de realiteit is dit echter niet het geval, door de beweging van de band zal ieder punt van de scan een beetje verschuiven ten opzichte van zijn voorgaand punt. Figuur 38 toont dit principe, hierbij start iedere scan aan de linkerkant. Bij grotere snelheden zal de verschuiving vergroten.



Figuur 38: Verschuiving meetpunten

Als het tijdsinterval tussen ieder punt gekend is, kan de visiesoftware aan de hand van de bandsnelheid de verschoven afstand van elk punt berekenen om de verschuiving te compenseren. Wegens een gebrek aan tijd is dit niet meer geïmplementeerd.

Literatuurlijst

[1] N. Vandekerkhof, *Laserscanmeting voor de herkenning van 3D-delen op een lopende band,* UHasselt-Faculteit Industriële ingenieurswetenschappen, 2015.

[2] *KUKA.ConveyorTech 3.2,* interne publicatie van KUKA Automatisering + Robots N.V., 2006.

[3] SICK, *LMS400 Laser measurement sensor Operating instructions,* 2013.

Bijlagen

Bijlage A: Ingestelde parameters LMS400

Bijlage B: Config-bestand van robotsturing

Bijlage C: Code visiesoftware

Bijlage D: sps.sub-bestand van de robot

Bijlage E: Hoofdprogramma van de robot

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Implementatie van robotcel voor het grijpen van 3D-delen van een transportband

Richting: **master in de industriële wetenschappen: energie-automatisering**
Jaar: **2016**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Van Den Hende, Bert

Datum: **6/06/2016**