2019•2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Visiegebaseerd kwaliteitscontrolesysteem voor integratie in een geautomatiseerd plooiproces: evaluatie met RGB-beelden en 3D-puntenwolken

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

PROMOTOR :

ing. Kim VAESEN

BEGELEIDER :

ing. Maarten VERHEYEN

Jan Dewez, Brecht Nijssen

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

Gezamenlijke opleiding UHasselt en KU Leuven

UHASSELT

KU LEUVEN

UHASSELT

KU LEUVEN

2019 • 2020

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Visiegebaseerd kwaliteitscontrolesysteem voor integratie in een geautomatiseerd plooiproces: evaluatie met RGB-beelden en 3D-puntenwolken

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

PROMOTOR :

ing. Kim VAESEN

BEGELEIDER :

ing. Maarten VERHEYEN

Jan Dewez, Brecht Nijssen

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

UHASSELT KU LEUVEN

Deze masterproef werd geschreven tijdens de COVID-19 crisis in 2020. Deze wereldwijde gezondheidscrisis heeft mogelijk een impact gehad op de opdracht, de onderzoekshandelingen en de onderzoeksresultaten.

Woord vooraf

Voor u ligt onze masterthesis ‘Visiegebaseerd kwaliteitscontrolesysteem voor integratie in een geautomatiseerd plooiproces: evaluatie met RGB-beelden en 3D-puntenwolken’. Deze thesis met bijhorend onderzoek is opgezet door twee studenten industrieel ingenieur van de gezamenlijke opleiding aan de Universiteit Hasselt en Katholieke Universiteit Leuven, genaamd Dewez Jan en Nijssen Brecht. In opdracht van Metes nv hebben wij het afgelopen jaar twee projecten opgestart en tot het gewenste eindresultaat proberen te leiden. Wij hopen met dit eindwerk op een geslaagde manier de richting elektromechanica met afstudeerrichting energie focus automatisering af te ronden.

Het onderwerp “visie” wordt binnen de opleiding licht aangehaald maar heeft altijd al onze interesse gewekt. Bij het lezen van dit masterproefonderwerp was onze keuze dan ook zeer snel gemaakt. Deze masterthesis biedt ons de mogelijkheid om onze theoretische achtergrond verder uit te breiden en tevens een toepassing in de praktijk te implementeren.

Deze masterthesis is enkel tot dit resultaat gekomen met de hulp van enkele specifieke individuen. Dit gezegd zijnde willen wij graag onze interne promotoren prof. dr. ir. Demeester Eric en interne begeleider ing. Verheyen Maarten bedanken voor hun opvolging, hulp en constructieve samenwerking tijdens het academiejaar. In het bijzonder willen wij onze externe promotor ing. Vaesen Kim, quality engineer bij Metes nv bedanken voor de fijne samenwerking en de leerrijke ervaring.

En ten slotte nog enkele externen aan wie wij onze dank willen betuigen, namelijk ing. Peter Aerts en de heer De Vidts Johan. Om wille van opgelegde maatregelen door het coronavirus waren we verhinderd om onze testen te ACRO verder aan te vatten. Peter heeft ons uit de nood geholpen door zijn Python-programma ter beschikking te stellen aan ons om toch simulaties uit te kunnen voeren. Johan heeft ons maandelijks de Halcon-licenties bezorgd en heeft ons daarbovenop begeleidt in het kiezen van de juiste onderdelen om een opstelling voor te stellen aan Metes.

Zonder al deze mensen was deze masterproef niet wat hij nu is en daarom zijn wij bovengenoemden enorm dankbaar.

Inhoud

Lijst van tabellen

Lijst van figuren

Abstract

Metes nv., een producent van totaalconcepten in plaatbewerking, kampt met verscheidene uitdagingen in de plooiafdeling. Ten eerste zijn bij het plooien van U-vormige beugels de beentjes vaak niet haaks geplooid wat de assemblage bemoeilijkt. De beentjes moeten een loodrechtheid (geometrische tolerantie) van 0,5 bezitten. Tot op heden voert een operator een visuele controle uit met het blote oog. Een tweede probleem treedt op bij een complexer plooiproduct. Een robotarm positioneert dit onderdeel in een plooibank, waarna het een reeks plooibewerkingen ondergaat. Metes stelt vast dat er op meerdere plekken fouten optreden. Voor beide problemen wenst Metes de integratie van een geautomatiseerde visiecontrole in de huidige productieprocessen.

Deze masterproef zoekt een optimale oplossing, zowel hardware- als softwarematig, voor beide uitdagingen en dit met testopstellingen te ACRO en de beeldverwerkingssoftware Halcon en Merlic. Beide projecten kennen een analoge aanpak, met name het ontwerpen van een opstelling en het zoeken naar methodes om de correcte producten te onderscheiden van de foutieve. Het daarnaast beoogde resultaat bij het project van de beugels is het opstellen van een handleiding voor Metes. Het tweede project vertrekt van een theoretische verbreding in de vorm van een literatuurstudie met als doel een methode te ontwikkelen om een optimale en betrouwbare 3D-scan te maken.

Uit de testen voor het beugelproject blijkt dat beide visieprogramma's een geschikte oplossing bieden. De gemiddelde afwijkingen van de hoekmetingen met Halcon en Merlic zijn respectievelijk 0,45° en 0,40°. De optimale opstelling bestaat uit een camera met twee LED-bars met diffusiefilter voor gepaste belichting. Uit de testen van het tweede project blijkt dat een SOL-opstelling (sheet-of-light) met laserlijngenerator en 2D camera op een lineaire geleiding geschikt is voor het verkrijgen van de gewenste resultaten. Twee mogelijk toe te passen foutdetectiemethodes leiden tot de gewenste resultaten, namelijk het uitvoeren van alle metingen in een 2D-omgeving en een tweede methode die de afstand meet tussen een scan en een referentie. De eerste methode verzekert de detectie van foutieve producten. De tweede methode vereist verder onderzoek om de vooropgestelde doelstellingen te behalen. De optimale oriëntatie van de SOL-opstelling t.o.v het werkstuk komt voort uit simulaties in Python.

Abstract in English

This master's thesis was commissioned by Metes nv. In their sheet metal bending department, Metes is confronted with various challenges. First, there are issues when bending U-shaped braces. The legs of the braces are often not perpendicular, which makes the assembly difficult. The legs must have a perpendicularity (geometric tolerance) of 0,5. To date, a worker manually checks these braces with the human eye, which is imprecise. A second issue occurs when bending another, more complex, sheet metal product. An industrial robot is used to present the product to a bending bench to perform a series of bending operations. Metes noticed that bending errors occur at several locations on the product. For both issues, Metes wishes to have an automated solution that can be implemented in the current production processes.

This master thesis seeks an optimal solution for both challenges. For this, test set-ups located at ACRO were adopted, as well as two image processing programs, Halcon and Merlic. Both projects follow analogous steps, including the hardware and software design of a vision solution that can discern correct from out-of-tolerance products. A further targeted result is a manual for further implementation by Metes. For the second project, a literature study is carried regarding optimal and reliable 3D scanning.

To summarize the results of the tests with the two vision programs, both vision programs are equally suitable for the first challenge with an average angular deviation of 0,45° and 0,40° for respectively Halcon and Merlic. The optimal test set-up contains a camera with two LED bars equipped with diffuse filter for appropriate exposure. The tests for the second project show that a sheet-of-light (SOL) set-up with a laser line generator and a 2D camera mounted on a linear guidance is suitable for acquiring the desired results. Two potential error detection methods yield the desired results, namely conducting all measurements in a 2D environment and a second method that works by calculating the distance between the scan and a reference. The first method successfully identifies out-of-tolerance products, while the second method needs more research to meet the aforementioned objectives. The optimal orientation of the SOL set-up relative to the product results from simulations in Python.

1 Inleiding

1.1 Situering

Metes NV, gelegen te Dilsen-Stokkem, is een producent die zich toelegt op het leveren van totaalconcepten in plaatbewerking, lichte metaalconstructies en elektro-assemblage. Sinds 2007 heeft Metes een tweede vestiging gelegen in Smederevo, Servië. Tot op heden werken er ruim 480 werknemers in de twee vestigingen samen. Waar de vestiging in Servië grotendeels instaat voor de productie fungeert de vestiging in Dilsen-Stokkem als front office waarbij de nadruk vooral gelegd wordt op nieuwe projecten, prototyping en geautomatiseerde projecten. Bovendien is Metes de centrale hub naar diverse klanten.

Met haar vele internationale klanten in uiteenlopende marktsegmenten (compressoren, airconditioning, medische sector, vliegtuigindustrie, etc.) is Metes aanwezig in een brede waaier van sectoren. De *core activities* van Metes zijn de volgende [1]:

- ponsen,
- laseren,
- plooien,
- lassen,
- rivetten,
- poedercoaten,
- assembleren,
- prototyping.

Om de machines maximaal te benutten, gebeurt de programmatie grotendeels off-line. De fine-tuning van de programma's gebeurt aan de machine zelf. Het bewaken van de kwaliteitsprocessen gebeurt vooral door procescontroles tijdens de operatie. Door deze kwaliteitsprocessen, daar waar mogelijk te automatiseren, tracht Metes de kwaliteit automatisch te borgen om extra controles achteraf te vermijden.

1.2 Probleemstelling

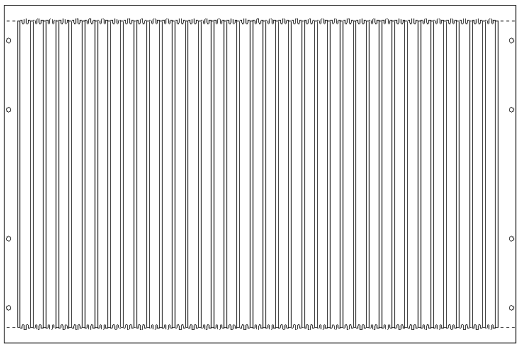
Metes heeft twee uitdagingen vooropgesteld voor deze masterproef.
De eerste uitdaging is de controle van U-vormige beugels, waarvan een voorbeeld terug te zien is in figuur 1.

Figuur 1: U-vormige beugel (gele component, reeds geassembleerd in geraamte)

Meer specifiek bevindt het voornaamste probleem zich bij de hoek van de twee verticale beentjes. Deze beugels behoren na montage tot het geraamte van een uit kunststof vervaardigde kabelgoot. Deze is te zien in figuur 2. De beugels zelf zijn uit staal vervaardigd. Door aan het lipje van de beugel te trekken, zal deze uit de kabelgoot schuiven. Doordat de tippen van de beugel zijn omgeplooid, zal deze stoppen aan het einde van zijn geleiding en zal zo dit deel van de kabelgoot als een lade uit het geraamte schuiven. Hierbij is het essentieel dat de beentjes een hoek van 90° maken om zo de correcte assemblage en beoogde werking te verzekeren. Een afwijking van een hoek bemoeilijkt het in- en uitschuiven in de kabelgoot.

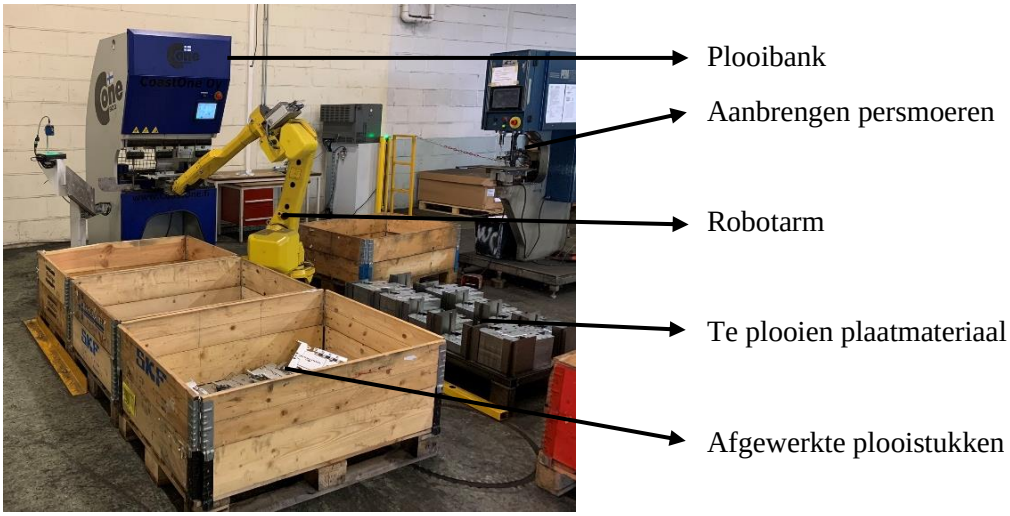
Metes lasert de beugels uit een stalen platen. Hierna zijn de beugels nog niet volledig los, maar hangen ze met de uiteindes nog vast aan elkaar. Dit resulteert in 37 aan-elkaar-hangende beugels. Deze plaat ondergaat eerst een plooiproces. De plaat voor het plooien, is te zien in figuur 3. Het plooien van de beugels is nodig om de typische U-vorm te verkrijgen. Na het plooien, breekt een arbeider beugel per beugel af. Deze arbeider controleert de beugels, die nauwe toleranties bezitten, momenteel met het blote oog. Dit maakt het controleproces onnauwkeurig met als gevolg dat de arbeider foutieve beugels soms goedkeurt.

Figuur 2: Geraamte kabelgoot



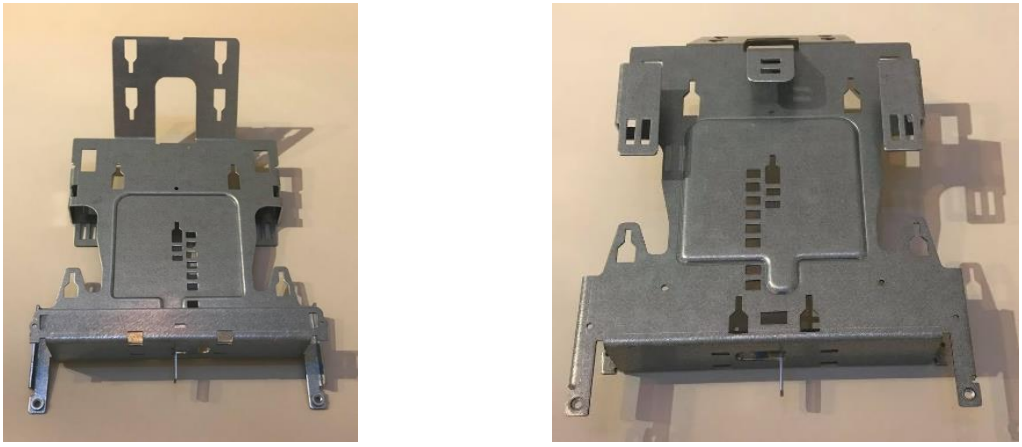
Figuur 3: Serie beugels

Ook voor de tweede uitdaging bij Metes is een automatische visuele controle op plooifouten noodzakelijk. De huidige opstelling bestaat uit een plooibank die het te plooien product aangevoerd krijgt door een robotarm met elektromagneet. De opstelling te Metes is te zien in figuur 4.



Figuur 4: Plooirobot te Metes

De robotarm oriënteert het product, dat te zien is in figuur 5 (a) en (b), zodat de plooibank de correcte plooibewerking uitvoert.

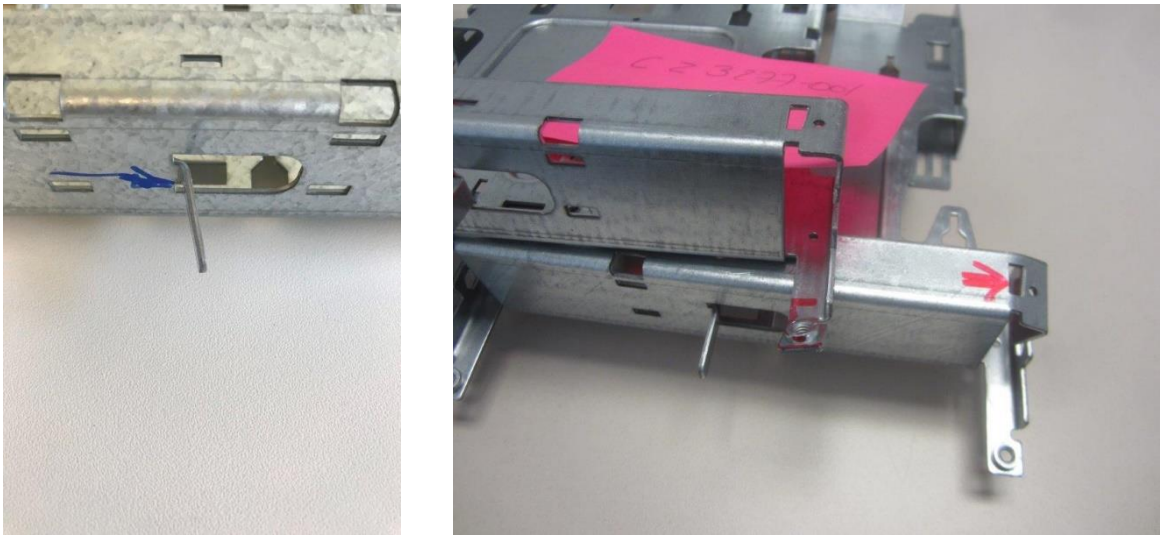


(a): Bovenaanzicht *(b): Onderaanzicht*

Figuur 5: Afgewerkte plooiproduct

De plooistukken zijn gemaakt uit plaatmateriaal. De robotarm grijpt de plooistukken, die nu nog slechts een stalen plaat zijn, via de elektromagneet vast. De eerste stap van het proces is het persen van de persmoeren. Deze persmoeren spelen een belangrijke rol in de assemblage en er is momenteel een 2D camera voorzien die het product controleert op de aanwezigheid en positie van deze persmoeren. Hierna begint de robotarm met het plooiproces. Tijdens het plooien moet de grijper het product regelmatig anders vastgrijpen. Hiervoor is een aanslagtafel voorzien waar de robotarm het onderdeel inlegt terwijl die het product volgens de gewenste oriëntatie vastgrijpt. Verder is er een magnetische aanslag aanwezig die de grijper de kans biedt om van positie te wijzigen. Zo kan de grijper bijvoorbeeld zijn positie veranderen van de bovenkant naar de onderkant van het onderdeel.

De fouten die hier optreden, zijn heel uiteenlopend, maar hebben altijd gevolgen voor de assemblage. In figuur 6 (a) en (b) zijn enkele plooifouten weergegeven.



(a): Lipje geen 90° *(b): Kantje scheef geplooid*

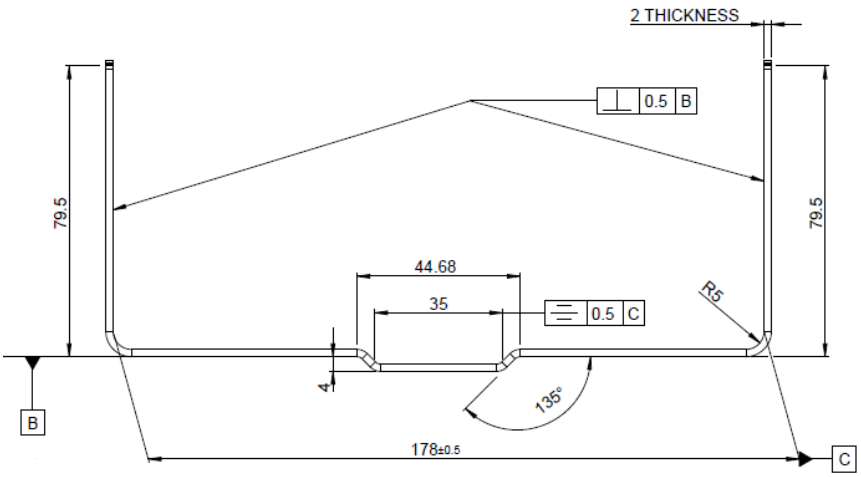
Figuur 6: Voorbeeld optredende fouten

De meeste fouten treden op door het foutief positioneren van de plooiplaat. Het foutief heropnemen van de plaat door de elektromagneet van de aanslagtafel is een mogelijke oorzaak. Een tweede en meer belangrijke oorzaak is het verschuiven van de plooiplaat op de elektromagneet door te snelle bewegingen van de robotarm.

1.3 Doelstellingen

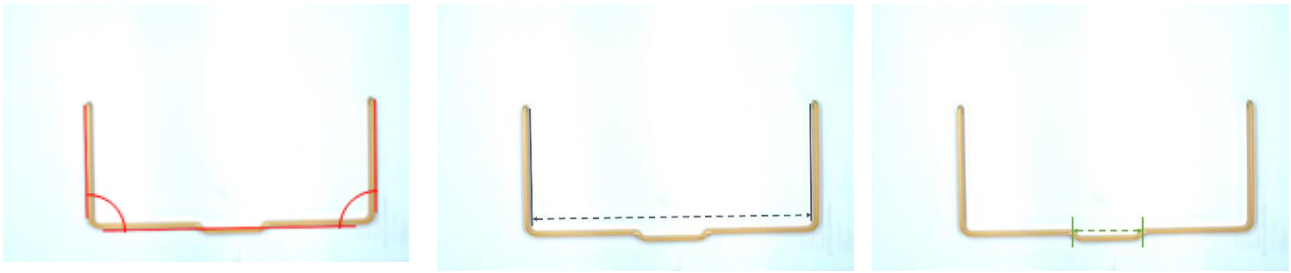
1.3.1 Doelstellingen beugelcontrole

Ten eerste verwacht Metes een geautomatiseerde visiecontrole die de afkeur van foutieve beugels verzekert en daarenboven afscheidt van de goede delen, ervan uitgaande dat deze installatie geen nood heeft aan een extra arbeider maar gewoon op zichzelf kan functioneren. Er zal echter wel een arbeider instaan voor het losbreken (en dus aanvoer) van de beugels. De belangrijkste doelstelling is het controleren van de juiste vorm van de beugels. Zoals figuur 7 toont, geldt er een geometrische tolerantie van loodrechtheid van 0,5 op de twee verticale beentjes waarbij het referentievlak B overeenkomt met de rug van de beugel.



Figuur 7: Detailtekening beugel

Aangezien de beugels moeten passen in een ander onderdeel zijn twee metingen van groot belang. Een eerste belangrijke meting is de hoekmeting, te zien in figuur 8 (a). De hoeken van de beentjes moeten een hoek van 90° vormen met de rug van de beugel. Een andere belangrijke meting is de afstand tussen de beentjes, te zien in figuur 8 (b). De afstand moet 178 mm bedragen met een tolerantie van 0,5 mm. Een minder kritische maat is deze van het lipje, zichtbaar in figuur 8 (c). Dit lipje dient voor het uittrekken van de beugel uit de module. Deze heeft enkel een praktisch nut en de mate van nauwkeurigheid van deze controle is ondergeschikt aan die van de hoek- en afstandsmeting van de beentjes.



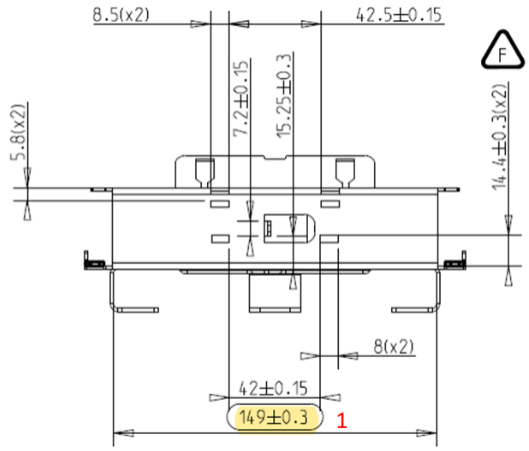
(a) Hoekmeting *(b) Afstand beentjes* *(c) Lipje*

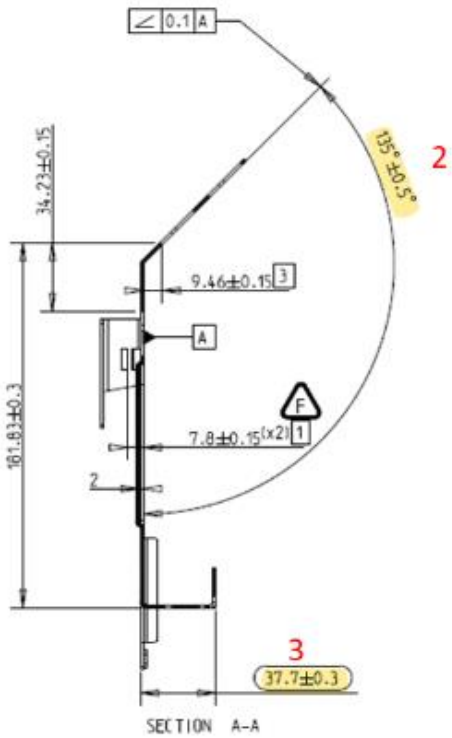
Figuur 8: Te meten eigenschappen beugel

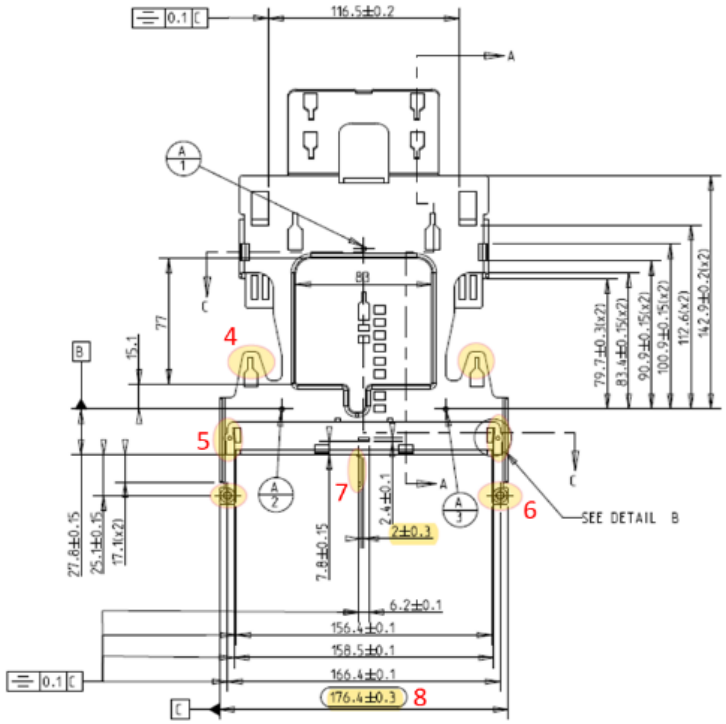
1.3.2 Doelstellingen plooirobot

Metes verwacht voor het tweede project een betrouwbare foutdetectie die verzekert dat er geen foutieve producten meer naar de klant vertrekken. Om een (te) grote verhoging van de doorlooptijd van het product tegen te gaan, moet de controle plaatsvinden op het einde van het plooiproces en niet tussen de plooistappen zelf. Omdat sommige plooifouten frequenter voorkomen dan anderen wenst Metes een programma dat eenvoudig aan te passen is.

De voornaamste optredende fouten zijn aangeduid met een gele kleur in Figuur 9 en opgelijst volgens de nummering aangebracht in de tekeningen.







Figuur 9: 2D-tekeningen met te controleren afmetingen

1) Binnenafstand tussen steunvoetjes
2) Kophoek
3) Hoogte opstaande kant
4) Rechtheid van flesjes
5) Rechtheid van lipjes opstaande deel
6) Aanwezigheid persmoeren
   Positie persmoeren
   Rechtheid van persmoerbeentjes
7) Centerafstand staartje
   Rechtheid staartje
8) Afstand omgeplooide wand persmoeren

2 Literatuurstudie

2.1 Inleiding

Deze literatuurstudie bekijkt bij 2.2 eerst 2D- en 3D-visiesystemen in het algemeen om vervolgens op beiden apart te focussen (2.3 en 2.4). Binnen de 2D-wereld focust dit vooronderzoek zich vooral op de beperkingen van 2D-visiesystemen, gevolgd door uitleg over 2D-camerakalibratie (2.3.5). Vervolgens werpt deze literatuurstudie zijn licht op mogelijke 3D-visiesystemen om 3D-puntenwolken te genereren. Het laatste gedeelte (2.4.3.2) bespreekt verschillende matchingprincipes binnen Halcon om gegenereerde puntenwolken te kunnen verwerken.

2.2 Algemeen 2D- en 3D-visie

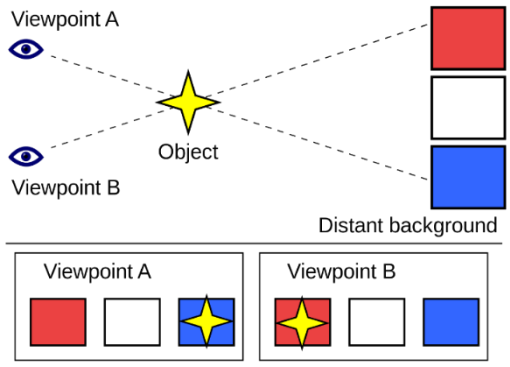
Visietoepassingen zijn afhankelijk van vijf kritische elementen. Een eerste element is belichting. De belichting bepaalt wat maar vooral hoe een camera een object waarneemt. Daarom is belichting noodzakelijk om objecten zichtbaar te maken en om sommige onderdelen eventueel extra op te lichten of te onderdrukken. Zonder belichting is niks zichtbaar maar het tegendeel, overbelichting, is ook een belangrijk aandachtspunt. Het foutief belichten van objecten kan leiden tot informatieverlies. Ten tweede vormt de optische lens een kritische component. De gebruikte lens bepaalt mee de kwaliteit van de afbeeldingen. Verder bezit deze een aantal parameters die eigen zijn aan de specifieke optische lens. Een derde kritische component is de *image capturing sensor*. Deze sensor, ook wel camera genoemd, zorgt voor het vastleggen van beelden. Verder staat de camera in voor de resolutie en de kwaliteit van de afbeelding. Dit kan gebeuren a.d.h.v. CMOS- of CCD-techniek. Een vierde kritische component is de verwerkingssoftware. Deze dient om de beelden te verwerken tot bruikbare informatie. Een laatste kritische component is communicatie. Niet alleen communicatie met machineonderdelen, maar eveneens communicatie voor het opslaan van data om later analyses mee door te voeren [1]–[4].

2.3 2D-visiesysteem

Een 2D-visiesysteem bestaat dus globaal uit een digitale camera, een lens en belichting . Dit systeem heeft echter een aantal begrenzingen zoals parallax, focusdiepte, omgevingslicht, variaties in contrast, etc. [5]. De secties hieronder leggen deze begrenzingen dieper uit.

2.3.1 Parallax

Bij het bekijken van een object vanop verschillende plaatsen kan de achtergrond er anders uitzien terwijl de afstand tussen de twee kijkafstanden niet zo veel verschilt. Figuur 10 geeft dit weer. Waar het object voor *viewpoint* A voor een blauwe achtergrond lijkt te staan, staat het object vanuit viewpoint B voor een rode achtergrond [6].

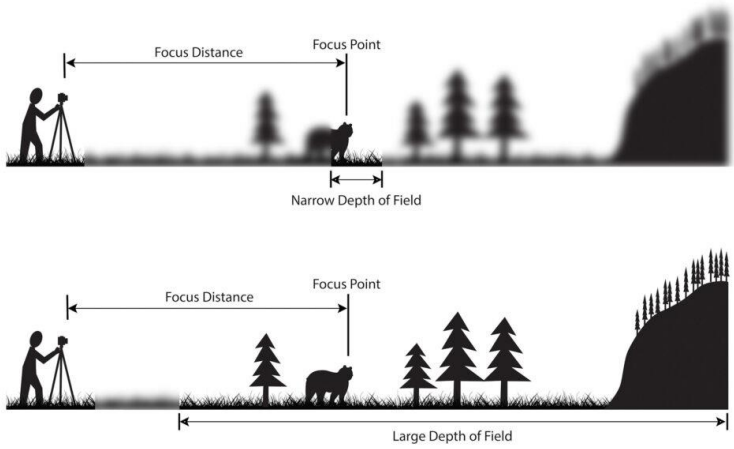


Figuur 10: Parallax [6].

Dit fenomeen zorgt typisch voor problemen bij de segmentatie van het object. De drempelwaardes zullen verschillen naargelang de achtergrond en dus verschillen afhankelijk van het viewpoint.

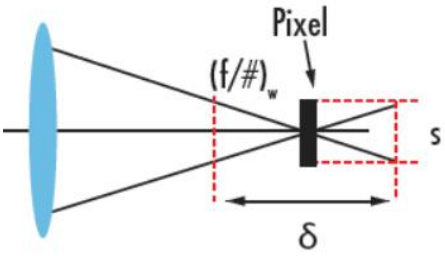
2.3.2 Focusdiepte

Depth of focus (focusdiepte) en *depth of field (DOF)* (scherptediepte) zijn twee niet te verwarren doch complementaire begrippen. DOF beschrijft het gebied waarbinnen een object als scherp verschijnt voor de camera uitgedrukt in macroscopische eenheden. Figuur 11 geeft hiervan een duidelijk voorbeeld. In de eerste situatie is de DOF klein. Het gebied waarvan de camera een scherp beeld krijgt, is klein. In de onderste situatie is de DOF groot, wat maakt dat er een groot gedeelte van de achtergrond als scherp vast te leggen is. Het vastleggen van een DOF is enkel nuttig bij het meedefiniëren van de resolutie en het contrast [7], [8].



Figuur 11: Depth of field [8]

De focusdiepte is het complement van DOF en bevindt zich aan de sensorzijde van de lens. De focusdiepte is gerelateerd aan hoe de kwaliteit van de focus verandert bij het verplaatsen van de camera terwijl het object blijft stilstaan. Het is m.a.w. de afstand waarover het beeldvlak kan verschuiven zonder dat de afbeelding zijn scherpte verliest [9]. Het uitdrukken van de focusdiepte gebeurt veelal in microscopische eenheden. De focusdiepte is sterk afhankelijk van het aantal pixels en aangezien camera's steeds meer pixels bezitten, versterkt dat dit probleem. Vooral bij lijnscantoepassingen is extra aandacht nodig bij de uitlijning tussen object, lens en sensor. In figuur 12 is de depth of focus voorgesteld door δ en de blauwe cilinder is de lens [7].



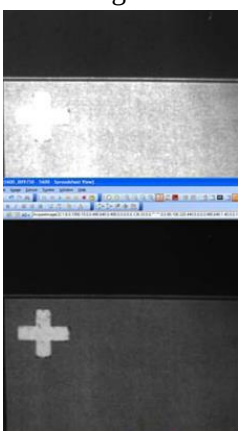
Figuur 12: Focusdiepte [7]

2.3.3 Omgevingslicht

Omgevingslicht kan nadelige effecten hebben bij 2D-visie omdat het niet op elk moment van de dag dezelfde eigenschappen heeft. Zo zullen de eigenschappen van ochtendlicht anders zijn dan het licht 's middags of 's avonds. Dit is wederom nadelig voor de segmentatie waarbij de drempelwaardes op voorhand vastliggen. Bij het blootstellen van het object aan licht met variërende eigenschappen is een aanpassing van de drempelwaardes noodzakelijk om fouten te vermijden. Daarbovenop heeft elk object een andere vorm, kleur en oppervlaktestructuur wat ervoor zorgt dat het invallende licht altijd anders reageert [10]. Verder kan artificieel licht ook voor problemen zorgen. Te veel of te weinig licht of schaduw kunnen het beeld verstoren of zelfs contouren doen vervagen [2]. Door een gepaste afscherming en lichtbron, waarvan de eigenschappen gekend zijn, te voorzien, is de invloed van omgevingslicht te elimineren.

2.3.4 Variaties in contrast

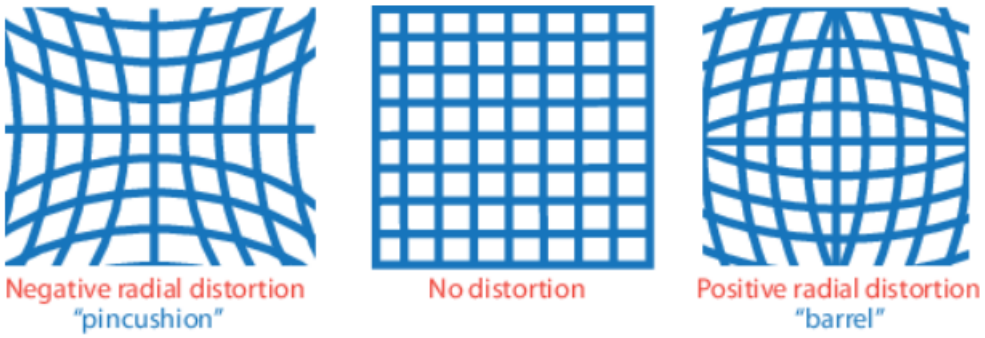
2D-visie berust op heldere contrasten om goede beelden te maken. Dit zal moeilijker zijn bij zeer matte of net heel glimmende objecten door het gebrek aan contrast. Figuur 13 geeft twee afbeeldingen weer van hetzelfde object. Op beide afbeeldingen is een kruisje aanwezig in de linkerbovenhoek. Waar het kruisje in de bovenste afbeelding, degene met een lager contrast, moeilijker te onderscheiden is van de rest, is het kruisje in de onderste afbeelding, waar het contrast veel groter is, eenvoudiger te onderscheiden [11].



Figuur 13: Variaties in contrast [11]

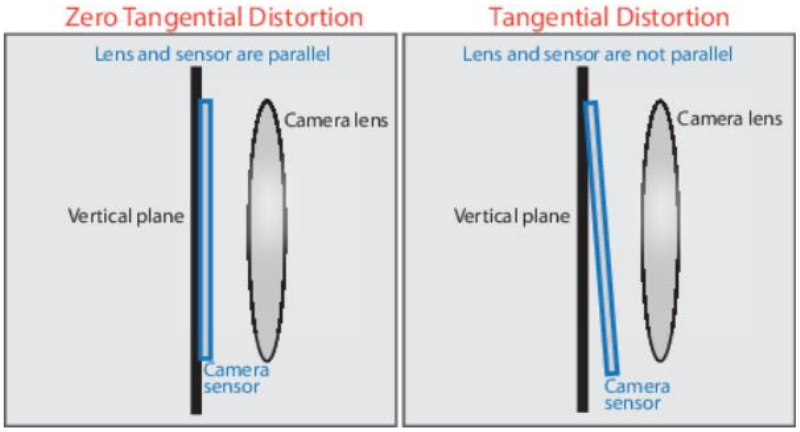
2.3.5 2D-kalibratie

Om de dimensies van op te meten objecten om te zetten in wereldcoördinaten is een kalibratie van de camera vereist. Een bijkomend voordeel van een kalibratie is het verwijderen van distorsies [12]. Een cameraopstelling bestaat uit intrinsieke en extrinsieke parameters. Waar de intrinsieke parameters de eigenschappen van de camera en lens zijn, stellen de extrinsieke parameters de pose van de camera voor. Deze pose vertaalt zich in zes parameters, namelijk drie parameters voor de translatie en drie parameters voor de rotatie [13]. Bij radiale distorsie buigen de lichtstralen aan de rand van de lens meer af dan in het optisch centrum [14]. Dit is zichtbaar in figuur 14.



Figuur 14: Radiale distorsie [14]

Tangentiële distorsie daarentegen treedt op als de lens en het beeldvlak niet parallel zijn met elkaar [14]. In figuur 15 is een voorbeeld te zien dat dit verduidelijkt.



Figuur 15: Tangentiële distorsie [14]

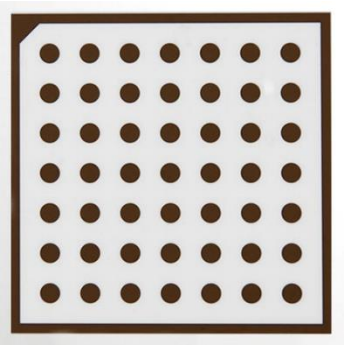
Figuur 16 (a) toont de afbeelding voor kalibratie en (b) toont dezelfde afbeelding na kalibratie. Hier is duidelijk uit op te maken dat de gebogen lijnen van (a) worden omgevormd naar rechte lijnen in (b). Dit is ook het doel van deze kalibratie. Door het verdwijnen van lensdistorsies is het uitvoeren van metingen betrouwbaarder [12].

(a): Voor kalibratie *(b): Na kalibratie*

Figuur 16: Verwijdering distorsie d.m.v. kalibratie [15, p. 14]

Om een transformatie naar wereldcoördinaten door te voeren binnen Halcon zijn er twee opties. Een eerste optie is het uitvoeren van de metingen op het niet-getransformeerde beeld om daarna de transformatie uit te voeren op de resultaten van de metingen. Anderzijds is het mogelijk om eerst het volledige beeld te transformeren en vervolgens op dit getransformeerde beeld de metingen uit te voeren. Optie één is frequenter toegepast o.w.v. de hogere snelheid aangezien de transformatie op een geringe hoeveelheid data gebeurt. Optie twee transformeert het hele beeld en bevat dus veel meer te transformeren data. De aangewezen werkwijze is afhankelijk van de soort van de meting. Om een meting uit te voeren m.b.v. *contours* kan best eerst de meting en daarna pas de transformatie plaatsvinden. Bij gebruik van *regions* is het aangeraden eerst de transformatie uit te voeren alvorens te meten. Door het selecteren van bepaalde grijswaardes deelt de afbeelding op in één of verschillende regio's. Deze regio's bezitten alle pixels die binnen een region horen. Een contour daarentegen stelt enkel de rand voor van het gesegmenteerde object waar de interne pixels komen te vervallen. De transformatie bij contours gebeurt met *contour_to_world_plane_xld.* Voor regions is er echter geen directe operator voorhanden om de gehele verkregen regio te transformeren. Een region omzetten naar een contour (*gen_contour_region_xld*) gevolgd door *contour_to_world_plane_xld* is een indirecte manier om regions toch om te zetten in wereldcoördinaten [12], [16].

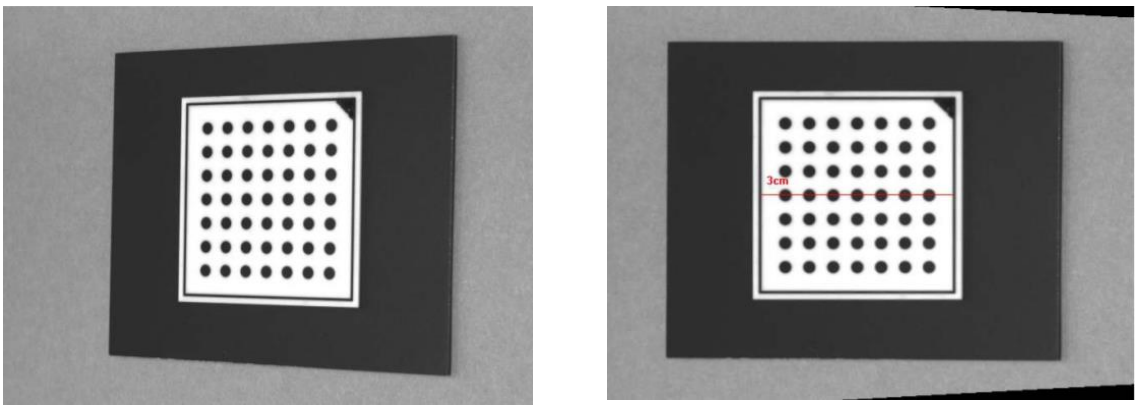
Het meten in wereldcoördinaten is opgedeeld in drie stappen, namelijk de kalibratie van de camera, de transformatie van de afbeelding en het uitvoeren van de metingen. De eerste stap is de kalibratie van de camera. Voor een correcte kalibratie van de camera zijn meerdere afbeeldingen van de kalibratieplaat nodig. Deze kalibratieplaat is zichtbaar in figuur 17. Alle genomen afbeeldingen moeten de kalibratieplaat bevatten en deze plaat moet ook in elke afbeelding een andere pose hebben. Bij minstens één afbeelding moet de kalibratieplaat op het ondervlak liggen [12].



Figuur 17: Kalibratieplaat [17]

Om de camera te kalibreren, moeten de intrinsieke cameraparameters gekend zijn [12]. Het uitvoeren van de kalibratie begint bij het zoeken naar de punten op een kalibratieplaat met gekende geometrie. Na het vinden van de 2D-positie volgt het bepalen van de pose van de kalibratieplaat. Deze pose ligt aan de basis van het bepalen van de intrinsieke en excentrieke cameraparameters [12].

Na de camerakalibratie moet de afbeelding een transformatie ondergaan. Dit gebeurt o.b.v. de extrinsieke cameraparameters. Dit is zichtbaar in onderstaande afbeeldingen. Figuur 18 (a) geeft de genomen afbeelding weer die nog distorsies bezit. Figuur 18 (b) geeft de genomen afbeelding weer die getransformeerd is a.d.h.v. de gegevens van de kalibratie. Na het transformeren van de afbeelding volgen de metingen o.b.v. de opgemaakte regions [12].



(a) voor transformatie *(b) na transformatie*

Figuur 18: Afbeelding kalibratieplaat [12, p. 89]

2.4 3D-visiesysteem

2.3 haalt reeds enkele belangrijke beperkingen van 2D-visie aan. Sommige applicaties of productielijnen botsen op de fysieke grenzen van 2D-visie en is er nood aan 3D-visie [2]. 2D-visiesystemen geven geen informatie over de hoogte (Z). Het verkregen beeld is een plat vlak (X en Y). Men ziet de contouren van een 3D-object en afhankelijk van het viewpoint gaan die verschillen. Hier stuit men al meteen op een ongewenste situatie, zeker bij vorm-kritische toepassingen [2].

3D-visiesystemen maken een puntenwolk op in de plaats van een 2D-beeld. Het is soms echter mogelijk om een 2D-beeld terug te krijgen waarop de z-afstand in grijswaarden is aangegeven. De puntenwolk bevat de 3D-coördinaten van alle punten, m.a.w. x-, y- en z-coördinaten. Enkele zaken zoals contrast en belichting zullen hier doorgaans minder parten spelen. Er zijn echter enkele bedenkingen bij 3D-visietechnologie. Zo zal een 3D-visiesysteem meer tijd vragen en processor- en software-intensiever werken [2].

De toepassingen van een 3D-camera zijn [2]:

- Dikte-, hoogte- en volume-meting,
- dimensionering en ruimtebeheer,
- opmeten van vormen, gaten, hoeken en curves,
- detectie van oppervlaktes of assemblagefouten,
- kwaliteitscontrole en verificatie met CAD-modellen,
- robotbegeleiding en *surface tracking*,
- *bin picking* voor plaatsing, verpakken en assemblage,
- objectdigitalisatie.

Een 3D-scantechniek is een niet-contact techniek. Er is dus geen contact tussen scanner en object. Deze niet-contact technieken komen vooral voor bij *reverse engineering*, *virtual assembly*, *surface inspection*, *rapid prototyping*, etc. Voor het bepalen van de punten op het oppervlak van een object zijn er verschillende mogelijkheden. In tabel 1 zijn enkele hoofdklasses weergegeven [16]. Twee categorieën van 3D-visie onderscheiden zich, namelijk passieve en actieve technieken [18].

Tabel 1: Algemene 3D-scantechnieken [16, p. 12], [19]

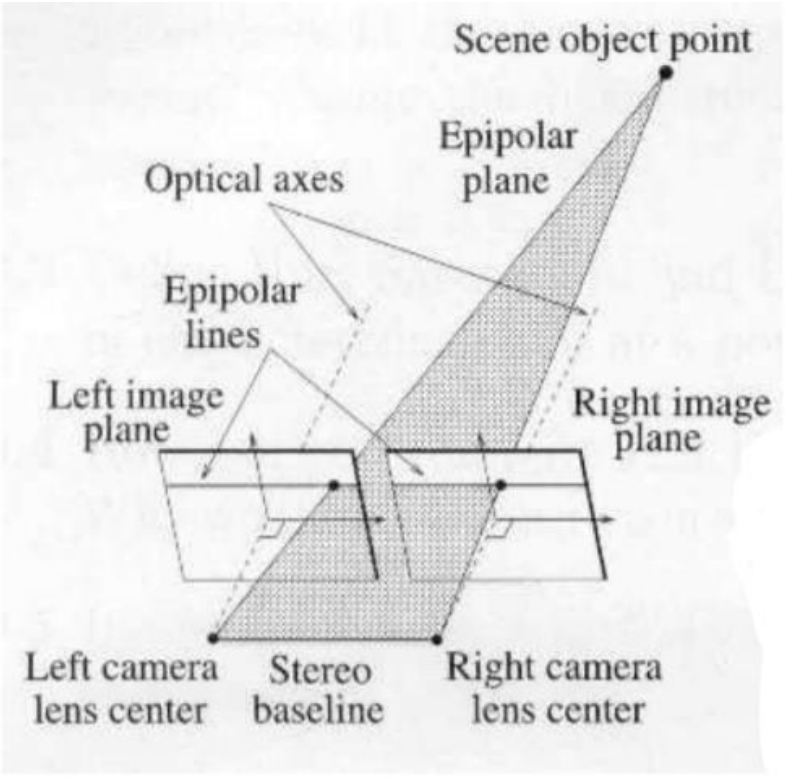
3D Reconstruction Approach	**Hardware Requirements**	**Object Size**	**Possible Results**
Stereovision	Multiple cameras	> 10 cm	X,Y,Z coordinates , disparity image , 3D object model
Structured Light	Camera(s), structured light projector	1 cm – 3 m	3D object model
Sheet-Of-Light	Camera, laser line projector, unit to move the object	Object must fit onto the moving object	X,Y,Z images , disparity image , 3D object model
3D Sensors	Special camera like calibrated TOF	30 cm – 5 m	X,Y,Z images

2.4.1 Passieve technieken

Bij passieve 3D-scantechnieken is de bron van belichting afkomstig van omgevingslicht. Omgevingslicht is meestal direct beschikbaar, maar infrarood is bijvoorbeeld ook mogelijk. Het zijn vrij goedkope technieken waarbij geen specifieke hardware nodig is maar waar simpele digitale camera's volstaan. Een typisch voorbeeld van een passieve techniek is stereovisie. Het is echter mogelijk om stereovisie actief te gebruiken door een patroon uit te zenden op het te inspecteren object.

2.4.1.1 Stereovisie

Kenmerkend aan stereovisie is de flexibiliteit maar gelimiteerde focus [20]. Bij stereovisie genereert de combinatie van twee 2D-beelden een 3D-beeld. Dit is te zien in figuur 19.



Figuur 19: Principe van stereovisie [21, p. 2]

Deze techniek zoekt naar overeenkomende punten in beide afbeeldingen. Deze overeenkomstige punten liggen aan de basis van een *disparity map*. Deze disparity map ligt op zijn beurt aan de basis van het opstellen van een 3D-object. Dit kan alleen indien de geometrische posities van de camera's t.o.v. elkaar gekend zijn.

2.4.2 Actieve technieken

In tegenstelling tot de passieve technieken is er bij de actieve 3D-visietechnieken wel externe belichting aanwezig. Dit gebeurt door straling of licht uit te zenden en de reflectie te detecteren. Het gaat hier om emissie van infrarood, X-ray, UV-licht, etc.

2.4.2.1 Time of flight (TOF)

Een TOF-camera kan zowel lengte, breedte als diepte waarnemen. Het principe van (directe) TOF is het kort laten flitsen van een licht op een object. Het vinden van de afstand tot het object is afhankelijk van de tijd tot de detectie van het gereflecteerde licht op de beeldsensor. Met de lichtsnelheid c gekend en de tijd t opgemeten, geeft onderstaande formule de afstand l tot een object [18].

$$l = \frac{c * t}{2} \qquad (1)$$

De precisie is afhankelijk van de meetnauwkeurigheid van *t*. Een goed voorbeeld is LiDaR of *Light Detection and Ranging*. Deze methode werkt doorgaans met een laser. Door punt per punt te meten en tegelijk de richting van de laser aan te passen, genereert een computer een beeld van de omgeving. Bij deze methode is snelle elektronica nodig aangezien het aantal genomen punten per seconde tussen de 10 000 en 100 000 punten ligt [18], [22]. In tabel 2 zijn de voor- en nadelen van TOF opgesomd.

Tabel 2: Voor- en nadelen van TOF [18], [23]

Voordelen	**Nadelen**
Grote afstanden	Nauwkeurigheid
Grote afmetingen (bvb. gebouwen)	Distorsie t.g.v. beweging
Tot 100 frames-per-second	Random ruis

2.4.2.2 Phase shift

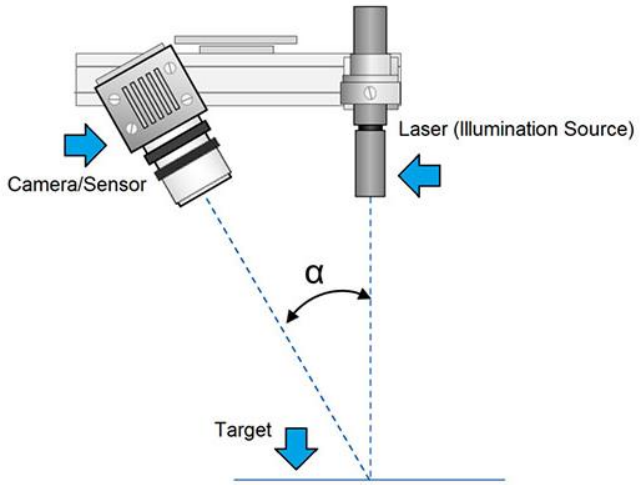
De *phase shift*-methode (ook wel indirecte TOF-methode genoemd) maakt tevens gebruik van een laser. Deze methode vergelijkt de faseverschuiving van het gereflecteerde licht met het uitgezonden licht. Er is dus een referentie signaal (verzonden signaal) en een gemeten signaal (gereflecteerd signaal) aanwezig. De afstand volgt uit de faseverschuiving van deze twee signalen. Meten met hogere frequenties, behaalt typisch nauwkeurigere resultaten maar het meetbereik is wel kleiner. Het bereik is namelijk beperkt tot de helft van de golflengte. Deze methode is sneller in het nemen van beelden dan directe TOF maar bevat wel meer ruis. In tegenstelling tot TOF waar het volgende signaal pas vertrekt als het vorige 'terug' is, worden er bij de phase shift continu pulsen uitgestuurd [24], [25]. Hierbij vormt het geen probleem om faseverschillen op te meten die binnen een periode van 2π vallen. Toch kan een fout op het faseverschil optreden. Dit komt dan door de elektrische ruis van de elektronische signaalverwerkingssystemen [25].

Na het afleggen van een afstand *d* reflecteert het licht op het oppervlak waardoor een faseverschil f optreedt. In het geval van de phase shift-methode is de fase tussen het referentie en het weerkaatste signaal proportioneel afhankelijk met de afgelegde afstand van het licht. Als het uitgestuurde licht een frequentie f bezit, bepaalt onderstaande formule de maximum afstand [25].

$$d = \frac{c}{2f} * \frac{\varphi}{2\pi} \qquad (2)$$

2.4.2.3 Triangulation

De naam *triangulation* oftewel triangulatie verwijst naar de opstelling. Meer specifiek verwijst de naam naar de driehoekvormige opstelling die de lichtbron, de camera en het te scannen object maken. Typisch bij deze methode is het richten van een lichtbron op een object om vervolgens m.b.v. een camera het gereflecteerde licht op te nemen. Een frequent gebruikte lichtbron is een laser. Afhankelijk van de hoogte, oriëntatie en oppervlakteruwheid van het object waarop de laser invalt, zal deze laser weerkaatsen tot in de camera. Door de verschillende eigenschappen van het object zal de laser weerkaatsen en invallen op verschillende plaatsen in de *field of view* van de camera. Figuur 20 geeft de principiële opbouw van een sheet-of-lightopstelling volgens het triangulatieprincipe. Triangulatie is tevens toe te passen bij stereovisie en structured-light. Bij de sheet-of-lightopstelling is de hoek α de hoek tussen het uitgestuurde en het gereflecteerde laserlicht. Afhankelijk van de objecteigenschappen kan deze hoek veranderen en op andere plaatsen op de camera invallen [26]. Typisch bedraagt deze hoek tussen de 30° en 60° [27].

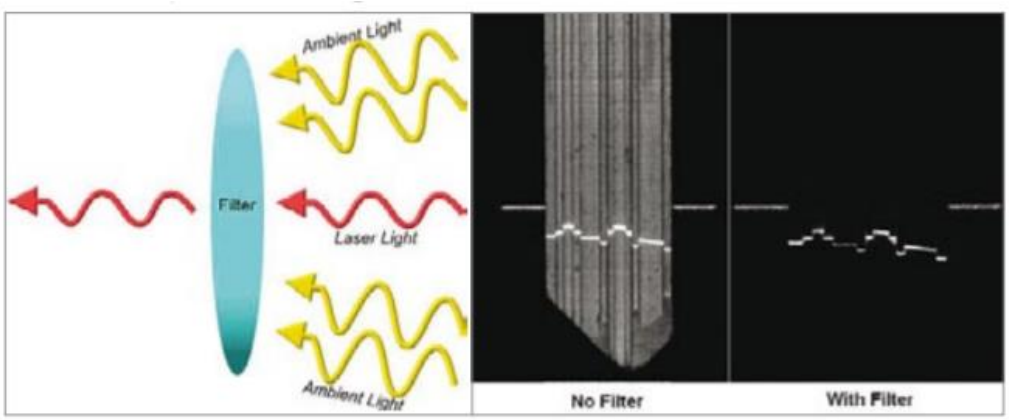


Figuur 20: Triangulatie: sheet-of-lightopstelling [28]

De afstand van de camera tot de laser is gekend. Verder is de hoek waaronder de laser staat ook gekend. Het bepalen van de hoek van de camera volgt uit het bepalen van de positie van de laser in de field of view van de camera. Zo is de grootte en vorm van de driehoek en de locatie van de *laser dot hoek* van de driehoek gekend [18]. Een *laser stripe* of *laser line* zijn tevens opties i.p.v. een laser dot. Het grootste voordeel van deze methode is de nauwkeurigheid die tot tientallen micrometers kan gaan. Deze methode is echter wel temperatuurafhankelijk en het bereik is gelimiteerd door de grootte van de laserline [20].

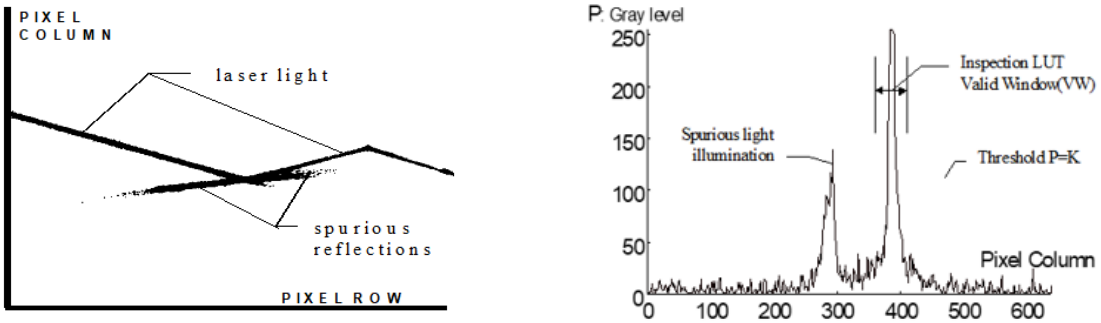
De meeste laserlijnscanners zijn uitgerust met filters om het omgevingslicht te blokkeren en enkel het laserlicht door te laten. Het onderdrukken van potentiële ruis, geïnduceerd door andere lichtbronnen, bevordert de prestaties van het systeem. Dit is te zien in figuur 21. Hierbij is de figuur links een principiële voorstelling die aantoont dat de filter enkel het laserlicht doorlaat. De figuur rechts toont op zijn beurt een effectieve scan. Aan de linkerkant van deze scan is het object nog zichtbaar, dit komt doordat er nog geen filter is toegepast. Aan de rechterkant van de scan is dit object niet meer zichtbaar, maar is enkel de laser zichtbaar [26]. Data wordt *slice* per slice gecollecteerd en elke slice bevat honderden punten op zich. De hoeveelheid punten is afhankelijk van de grootte van de camerasensor en natuurlijk hoeveel van het object binnen de field of view ligt. De snelheid van slices nemen is afhankelijk van de *frame capture frequency* van de camera. Voor een camera met een frame capture frequency van 30 Hz betekent het dat deze camera 30 slices per seconde kan maken [26]. In de praktijk blijkt dat het moeilijk is deze maximale frame capture frequency te halen is omdat deze afhankelijk is van de

ingestelde sluitertijd van de camera. De sluitertijd van een camera is de tijd gedurende dewelke licht invalt op de beeldsensor van de camera [29].



Figuur 21: Lasertriangulatie met camerafilter [26, p. 1]

Een typisch probleem dat optreedt bij deze methode is ongewenste reflectie. Dit effect is groter bij glimmende objecten. De camera zal niet enkel het gereflecteerde licht van de laser opvangen maar ook ander, ongewenst, omgevingslicht. Het is mogelijk dat er *spurious reflections* optreden. Deze vorm van reflectie treedt op bij het belichten van een glimmend deel en dit deel andere delen van het object belicht. Hierdoor kunnen foutieve metingen optreden [30]. Figuur 22 (a) en (b) tonen respectievelijk de gevolgen van ongewenste reflectie voor het verkregen camerabeeld en de grijswaardeverdeling. De spurious reflections die de camera waarneemt, vertalen zich in een lokaal maximum in de intensiteitsverdeling [31].



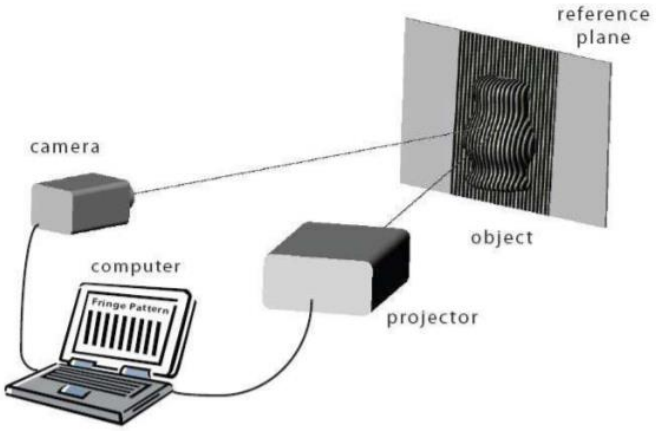
(a) Camera afbeelding *(b) Grijswaardeverdeling*

Figuur 22: Sheet of light reflectiedefecten [31, p. 365]

T.g.v. ongewenste reflectie kan de camera onderdelen van het object die van belang zijn slechter of helemaal niet waarnemen. Dit bemoeilijkt bijvoorbeeld het detecteren van randen. Een ander gevolg van ongewenste reflectie is het ontstaan van foute punten in de 3D-puntenwolk van de scan. Het is echter mogelijk om o.b.v. de 3D-datawolkdichtheid ongewenste reflecties te elimineren [31].

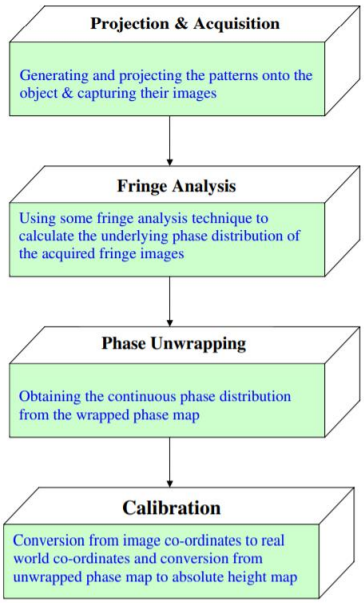
2.4.2.4 Structured-light projection

Het principe van *structured-light projection* berust op het projecteren van een gestructureerd patroon op een object. In figuur 23 is de klassieke structured-lightopstelling weergegeven. T.g.v. reliëf of vorm van het object zal dit patroon vervormingen vertonen. De camera legt dit vervormd patroon vast. Buiten een projector en camera is ook een referentievlak nodig om de vervorming van het patroon mee te vergelijken [32].



Figuur 23: Structured-lightprincipe [32, p. 133]

De volgende stap bestaat uit het bepalen van de fasemodulatie d.m.v. een structured-light analysetechniek. De mogelijke analysetechnieken zijn Fouriertransformatie, ruimtelijke fasedetectie en fasestapeling. Om een continue distributie te verkrijgen, is een fase-uitpakalgoritme nodig. De laatste stap is het kalibreren van het systeem om de reële 3D-wereldcoördinaten te verkrijgen. Figuur 24 geeft deze stappen chronologisch weer [32].



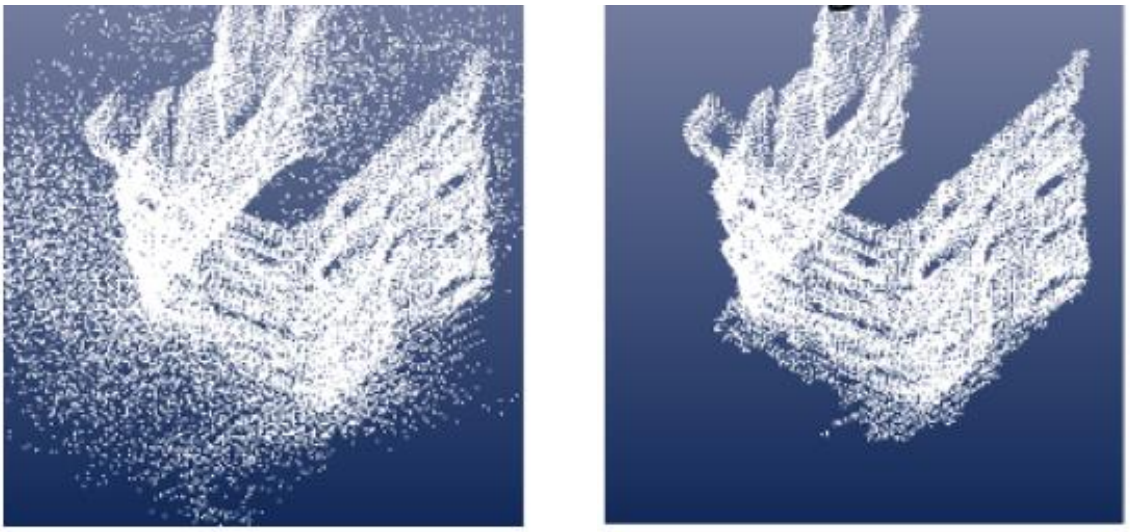
Figuur 24: Stappenplan structured-light [32, p. 134]

Structured-light projection wordt op heden voor een hele hoop toepassingen ingezet. Om zo bijvoorbeeld de vlakheid van platen (2,5m x 0,45m) op te meten. Ook heeft deze projectie zijn weg gevonden tot biomedische , industriële en kinematische toepassingen. [32].

De sterkte van deze techniek is dat deze in een hoge resolutie een volledige 3D-reconstructie van het object genereert zonder contact te maken met het oppervlak. Verder is het scannen van zowel kleine als grote objecten mogelijk. Tegenover deze voordelen staan ook nadelen. Zo kunnen reflecterende objecten en interreflectie voor problemen zorgen. Dit nadeel valt weer weg te werken door een *multi frequency fringes* toe te passen [25], [33].

2.4.3 Puntenwolk

Uit een 3D-camera komt, zoals eerder vermeld, een 3D-puntenwolk. Idealiter bevat deze puntenwolk uitsluitend nuttige punten. In de praktijk is dit echter niet het geval en bevatten puntenwolken initieel veel ruis. Het verwijderen van deze ruis biedt de mogelijkheid om nauwkeurige voorstellingen, metingen, etc. te verkrijgen. figuur 25 geeft een voorbeeld van een 3D-puntenwolk weer met ruis (a) en na het verwijderen van ruis (b).



(a):Voor ruisvermindering *(b) Na ruisvermindering*

Figuur 25: Voorbeeld puntenwolk [34]

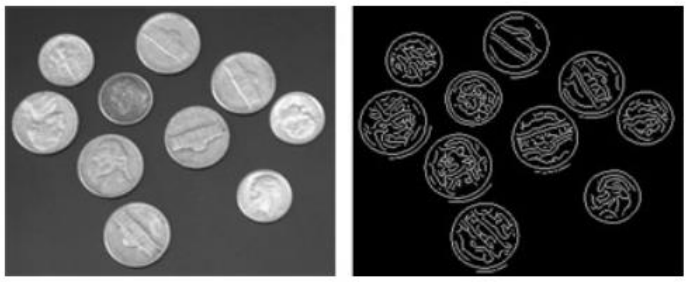
2.4.3.1 Ruisvermindering van puntenwolk

Door het stappenplan, dat hieronder is weergegeven, te volgen, is het mogelijk de puntenwolk te verwerken en een beslissing te nemen.

1. Puntenwolk opnemen.
2. Object segmenteren en detecteren.
3. Metingen uitvoeren op scan.

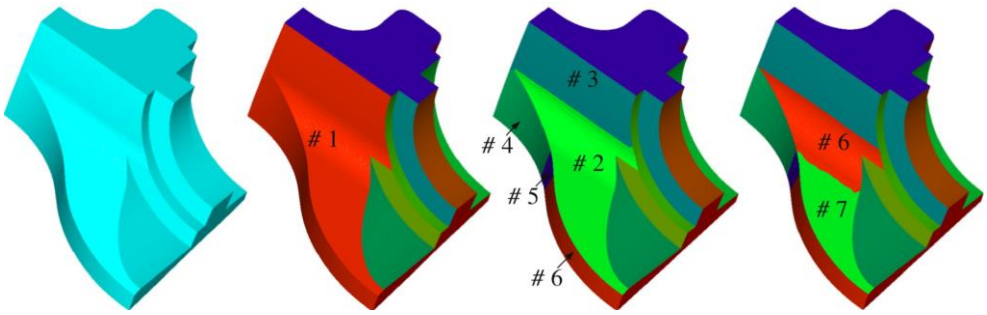
Om een betere segmentatie uit te voeren is het nuttig om de puntenwolk af te vlakken. Dat wil zeggen de ruis op een gepaste manier te onderdrukken. Men onderscheidt vier mogelijke segmentatietechnieken, namelijk *edge-based, surface-based, scan-line based* en *model-based segmentation* [35].

Een eerste techniek is edge-based segmentatie, deze techniek bestaat globaal uit twee onderdelen, zijnde randdetectie en het groeperen van de punten in een rand. Randen worden vastgelegd door verandering in lokale oppervlak-eigenschappen t.o.v. een vastgelegde drempelwaarde. Deze oppervlak-eigenschappen zijn in de meeste gevallen oppervlak-normalen, hellingshoeken of hogere orde afgeleide. Er zijn verschillende methodes om aan edge-based segmentatie te doen, zo zijn er bijvoorbeeld de *Sobel-*, *Canny-* en *Fuzzy-logic*-methode. Een voorbeeld van de Canny-methode is in figuur 26 te zien [35]–[37].



Figuur 26: Randdetectie volgens de Canny-methode [36]

Vervolgens is er surface-based segmentatie. Deze techniek gebruikt ook lokale oppervlak-eigenschappen en groepeert punten die ruimtelijk dicht bij elkaar liggen en gelijkaardige oppervlak-eigenschappen hebben. Dit maakt surface-based segmentatie minder ruisgevoelig. Bijgevolg benaderen deze resultaten globaal gezien de werkelijkheid beter dan edge-based segmentatie. Bij surface-based segmentatie zijn er twee mogelijke manieren namelijk *bottom-up* en *top-down*. Bottom-up werkt met een startpixel die als een zaadje uitgroeit tot segmenten met dezelfde criteria. Het nauwkeurig kiezen van deze startpixel is belangrijk omdat deze de segmenten vastlegt. De werking van top-down is geheel omgekeerd. Hier worden de pixels eerst toegewezen aan één segment. Daarna wordt dit segment steeds opgedeeld in subsegmenten zolang dat de drempelwaarde niet bereikt wordt. Algemeen wordt bottom-up regelmatiger toegepast dan top-down in 3D-visietoepassingen. Figuur 27 toont hoe surface-based segmentatie kan verlopen. De eerste stap groepeert oppervlakken met gelijkaardige normalen. Stap twee brengt de kromming nog in rekening om uiteindelijk na stap drie een verfijnde surface segmentatie te bekomen [35], [38] Het onderstaande voorbeeld komt tot stand volgens de top-down methode. Initieel bestaat de voorstelling uit één segment waarna het opsplitst in subsegmenten.



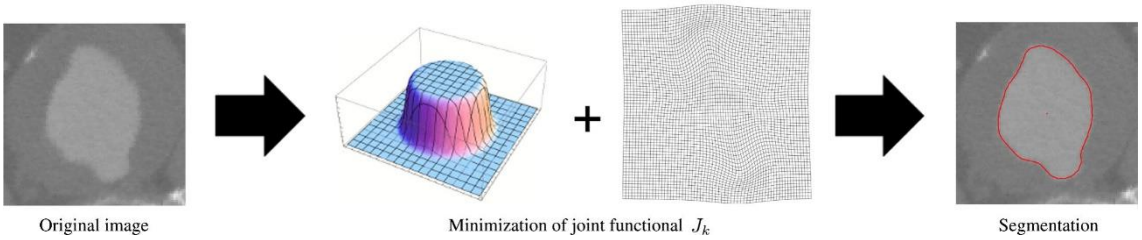
Figuur 27: Voorbeeld stapsgewijze surface-based segmentatie [38, p. 665]

Ten derde is er de scanline-based segmentatie. Hierbij is elke rij van een afbeelding voor te stellen als een scanlijn. Deze techniek is gebaseerd op het feit dat een scanlijn op een 3D-vlak een 3D-lijn maakt. Door aangrenzende lijnen met dezelfde eigenschappen te groeperen, ontstaan vlakke segmenten. Dit is typisch een heel snelle methode maar haar nut komt enkel tot haar recht bij het scannen van

puntenwolken waar de dichtheid niet te veel varieert [37]. Er zijn nog vergelijkbare methodes ontwikkeld, zoals bijvoorbeeld de *Besl and Jain*-methode (BJ-methode). Deze methode doorloopt twee fases. De eerste fase bestaat uit een grove segmentatie gebaseerd op het schatten van het gemiddelde en de Gaussiaanse kromming van elk punt. De tweede stap, *region growing*, verfijnt deze ruwe segmentatie. Dit is gebaseerd op passende bivariate polynomiale oppervlakken. Deze methode is echter zeer tijdrovend [35].

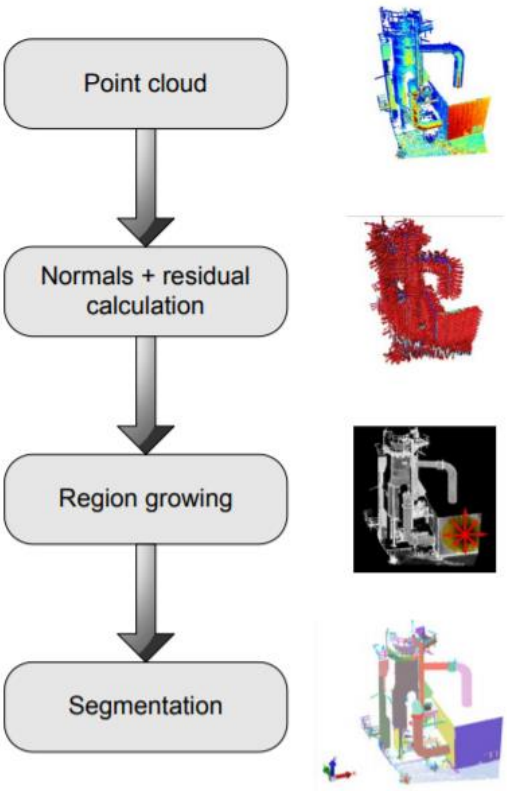
Deze twee stappen aanpassen, resulteert in een reductie van tijd én kwaliteit. Daarom zal de eerste stap bestaan uit het opstellen van een set curves die worden opgesteld m.b.v. de *splitting*-methode. Door in de tweede stap deze curves te groeperen, ontstaan oppervlakken. Deze techniek heet de JBM-methode. Deze kan op industriële schaal toegepast worden maar heeft ook een beperking. Deze methode bestaat namelijk uit het opstellen en groeperen van scanlijnen maar bij een lijnscan bestaan nog niet alle punten vooraf [35].

Tot slot is er model-based of *knowledge-based* segmentatie. Deze techniek wordt veelal toegepast in de medische sector waar beelden dikwijls heel hard van elkaar kunnen verschillen en waar de afbeeldingen eerder een slechte kwaliteit (veel ruis) bezitten. Dokters herkennen objecten ongeacht de vorm. De kennis over deze objecten zit in het hoofd van de dokter. Deze segmentatiemethode berust op het gebruik van slimme algoritmes om deze kennis digitaal te gebruiken. In onderstaand voorbeeld is m.b.v. een *intensity model* (aangevuld met een *elastic model*) de segmentatie mogelijk van een pathologische aorta [39].



Figuur 28: Voorbeeld model-based segmentatie in de medische sector [40, p. 1190]

Figuur 29 en Figuur 30 tonen respectievelijk een stappenplan en bijhorend algoritme voor het doorvoeren van een segmentatie. Bij het opstellen van dit algoritme werden een aantal aandachtspunten in het achterhoofd gehouden. Een eerste aandachtspunt is het starten met ruwe data. Vervolgens worden enkel oppervlaktenormalen gebruikt, aangezien deze betrouwbaar geschat kunnen worden bij aanwezigheid van ruis. Ten derde kan de gebruiker zelf beslissen welke graad van over- of ondersegmentatie toegepast wordt. Dit kan door het aanpassen van een aantal parameters. Als laatste aandachtspunt is er het feit dat het segmentatie-programma zo weinig mogelijk tijd in beslag moet nemen en dat het aantal aan te passen parameters eerder beperkt blijft [35].



Figuur 29: Stappenplan segmentatie [34, p. 249]

Algorithm 1 Segment a given point cloud using smoothness constraint

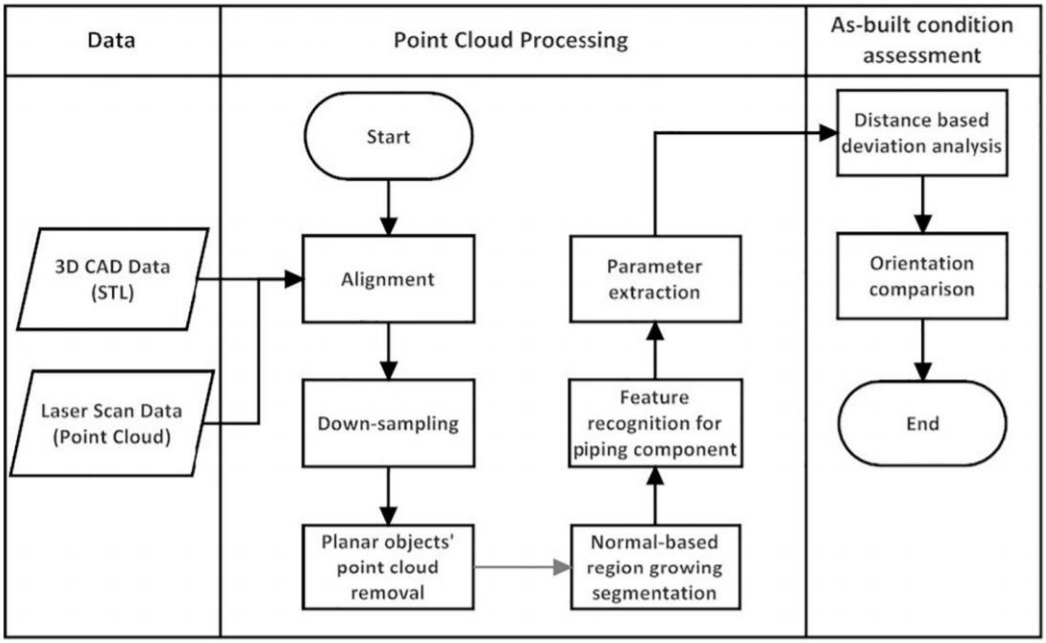
Inputs: Point cloud = $\{\mathbf{P}\}$, point normals $\{\mathbf{N}\}$, residuals $\{\mathbf{r}\}$, neighbor finding function $\mathbf{\Omega}(.)$, residual threshold r_{th}, angle threshold θ_{th}
Initialize Region List $\{\mathbf{R}\} \leftarrow \Phi$, Available points list $\{\mathbf{A}\} \leftarrow \{1 \cdots P_{count}\}$
while $\{\mathbf{A}\}$ is not empty **do**
  Current region $\{\mathbf{R}_c\} \leftarrow \emptyset$, Current seeds$\{\mathbf{S}_c\} \leftarrow \Phi$
  Point with minimum residual in $\{\mathbf{A}\} \rightarrow P_{min}$
  $P_{min} \overset{\text{insert}}{\rightarrow} \{\mathbf{S}_c\} \& \{\mathbf{R}_c\}$
  $P_{min} \overset{\text{remove}}{\rightarrow} \{\mathbf{A}\}$
  for $i = 0$ to **size**$(\{\mathbf{S}_c\})$ **do**
    Find nearest neighbors of current seed point $\{\mathbf{B}_c\} \leftarrow \mathbf{\Omega}(\mathbf{S_c}\{\mathbf{i}\})$
    for $j = 0$ to **size**$(\{\mathbf{B}_c\})$ **do**
      Current neighbor point $P_j \leftarrow \mathbf{B}_c\{j\}$
      if $\{\mathbf{A}\}$ contains P_j and $\cos^{-1}(|\langle \mathbf{N}\{\mathbf{S}_c\{i\}\}, \mathbf{N}\{P_j\}\rangle|) < \theta_{th}$ **then**
        $P_j \overset{insert}{\rightarrow} \{\mathbf{R}_c\}$
        $P_j \overset{remove}{\rightarrow} \{\mathbf{A}\}$
        if $\mathbf{r}\{P_j\} < r_{th}$ **then**
          $P_j \overset{insert}{\rightarrow} \{\mathbf{S}_c\}$
        end if
      end if
    end for
  end for
  Add current region to global segment list $\{\mathbf{R}_c\} \overset{insert}{\rightarrow} \{\mathbf{R}\}$
end while
Sort $\{\mathbf{R}_c\}$ according to the size of the region.
Return $\{\mathbf{R}_c\}$

Figuur 30: Algoritme segmentatie [34, p. 250]

De eerste stap, het normaalschatten, schat de normaal door een vlak te bepalen door naburige punten. Om te bepalen of deze al dan niet naburig zijn, bestaan twee algoritmes. Een eerste algoritme is *K-nearest-neighbours* (KNN). Deze methode bepaalt de k-punten met een minimale afstand voor een gegeven punt van de puntenwolk. Als het aantal k-punten vast ligt, wordt de *area of interest* (AOI) aangepast zodat de gewenste hoeveelheid punten hierbinnen valt. Bij het gebruik van een grotere AOI in gebieden met lage puntendensiteit is de normaal beter te schatten. Een ander mogelijk algoritme is *fixed-distance-neighbours* (FDN). Hier ligt de AOI vast en is het aantal punten afhankelijk van de puntdichtheid. Deze methode heeft echter niet het aanpassingsvermogen dat de KNN-methode wel bezit aangezien hier het aantal punten proportioneel is met de puntendensiteit. Als de puntdichtheid nagenoeg niet verandert, is het beter om de FDN-methode toe te passen. De volgende stap is *plane fitting*. Om een vlak te bepalen, wordt de kleinste kwadratenmethode toegepast [35]. Na de normaalschatting volgt region growing. Deze stap maakt gebruik van de normalen met hun restanten. Hierbij is het belangrijk dat men niet aan over- of ondersegmentatie doet [35].

2.4.3.2 Vergelijking 3D-puntenwolk met CAD-model

Figuur 31 geeft een stappenplan weer om een puntenwolk te vergelijken met een CAD-model. Meer specifiek is dit opgesteld voor de controle van pijpleidingen in opbouw [41].



Figuur 31: Stappenplan vergelijking CAD-model en puntenwolk [41, p. 45]

De verschillende operatoren aangehaald in onderstaande uitleg zijn beschikbaar in Halcon. Een eerste methode voor matching is *surface-based matching*. Deze methode haalt een *surface model* uit een 3D-object. Dit gebeurt in twee stappen. De eerste stap is het creëren van *surface normals*. Als dit gedaan is, volgt het creëren van een surface model met de operator *create_surface_model*. Hierbij is het van belang dat alle normalen in dezelfde richting liggen. Het visualiseren van de normalen van zowel het CAD-model als van de scan maakt duidelijk of de normalen correct liggen of niet. Indien ze niet correct liggen (inwaarts of uitwaarts), biedt de parameter *model_invert_normals* een oplossing door de normalen te inverteren. Het object in het surface model vinden, kan met *find_surface_model*. Deze operator geeft een pose en score terug van de matching die uitgevoerd is [16]. De interpretatie van de score is afhankelijk van de activatie van *poserefinement*. Is deze geactiveerd, dan geeft de score een cijfer tussen 0 en 1 terug afhankelijk van de geschatte fractie van het object dat zichtbaar is in de scene. Is *poserefinement* niet geactiveerd, dan geeft de score het geschatte aantal punten terug van de scene die zich in het voorwerp bevinden [42].

Een tweede methode voor matching is *register*. Binnen de registerfunctie zijn twee verschillende functies, namelijk *register_object_model_3d_pair* en *register_object_model_3d_global*. *Register_object_model_3d_pair* zoekt naar de transformatie om een 3D-object optimaal te aligneren met een ander 3D-object. Hierbij moet de parameter *Method* op '*matching*' staan. Dit heet ook wel *pair-wise registration*. Deze operator geeft een pose en score terug. Met deze pose is het mogelijk om de twee 3D-objecten te laten samenvallen. De score stelt de ratio overlappende delen op niet-overlappende delen voor [43]. Verder staat deze functie in voor het verbeteren van de positie. Om deze positieverbetering uit te voeren, moet Method op *'icp'* staan en moet het object en het CAD-model zich in hetzelfde assenstelsel bevinden. R*egister_object_model_3d_global* daarentegen zoekt naar de relatieve positie tussen verschillende 3D-objecten. Dit wordt eerder toegepast als een referentie 3D-model bestaat uit meerdere 3D-objecten. Deze functie geeft een matrix terug die de verschillende posities bezit. Na het uitvoeren van deze functies moet de positie van één van de 3D-objecten aangepast

worden. Dit kan gebeuren door *affine_trans_object_model_3d* [16]. Bij *register_object_model_3d_global* worden het aantal gevonden neighbours die een overlap bezitten weergegeven in de score, dit gebeurt voor alle gevonden objecten [44].

2.5 Conclusie

Een eerste conclusie omtrent deze literatuurstudie is dat het van belang is voldoende aandacht te besteden aan het selecteren van de onderdelen van het visiesysteem bij het opbouwen van een testopstelling. Meer precies aan de belichting, lenzen en keuze van de camera. De camera bepaalt samen met de lens de kwaliteit van de beelden en onrechtstreeks ook de kwaliteit van een meting. Met gepaste belichting is het mogelijk belangrijke delen van een object te accentueren of net te onderdrukken. Ten tweede is het interessant voor de nauwkeurigheid van metingen om een 2D-kalibratie door te voeren en de afbeelding om te zetten naar wereldcoördinaten. Voor de toepassing van het project omtrent de plooirobot is een 3D-visiecontrole nodig. De derde conclusie omvat drie technieken die voor deze toepassing interessant zijn, namelijk stereovisie, structured-light en lasertriangulatie (sheet-of-light). Naast de techniek om 3D-data te verkrijgen, is ook de techniek om ruis te verwijderen een belangrijk aspect. De naar voor gekomen methodes om ruis te verminderen bij een 3D-puntenwolk zijn edge-based, surface-based en scanline-based segmentatie. Ten slotte haalt deze literatuurstudie twee matchingsprincipes aan die tevens binnen Halcon toe te passen zijn, namelijk surface-based matching en register-based matching. Aangezien matching een belangrijke opstap is naar het uitvoeren van metingen van een 3D-scan t.o.v. een CAD-model is het van belang hier genoeg aandacht aan te besteden bij het ontwerpen van een controleprogramma.

3 Opdracht 1: geplooide beugels

3.1 Methode

Het doel van dit project is het opstellen van een kosten- en haalbaarheidsanalyse. De onderzoeksgroep ACRO in het technologiecentrum te Diepenbeek beschikt over de middelen en plaats om een kleinschalige testopstelling te bouwen. Na het uitdenken van een theoretisch concept kan o.b.v. experimentele vaststellingen een finaal concept worden voorgelegd. Zoals aangehaald bij de doelstellingen moet de theoretische uitwerking minstens twee zaken bevatten: een visiecontrole-unit en een transport-en sorteersysteem.

De testopstelling dient hoofdzakelijk om de mogelijkheden van het visieprogramma in kaart te brengen. 3.2 gaat hier dieper op inn.

Voor het gemak van Metes wordt er in twee verschillende softwarepakketten, zijnde Merlic en Halcon, gewerkt. In 3.3.1 en 3.3.2 volgt meer uitleg over beiden. Merlic is een programma met een meer gebruiksvriendelijke userinterface, terwijl Halcon uitgebreide standaardsoftware is voor machinevisie. Beide softwarepakketten maken gebruik van RGB-beelden om de metingen op uit te voeren.

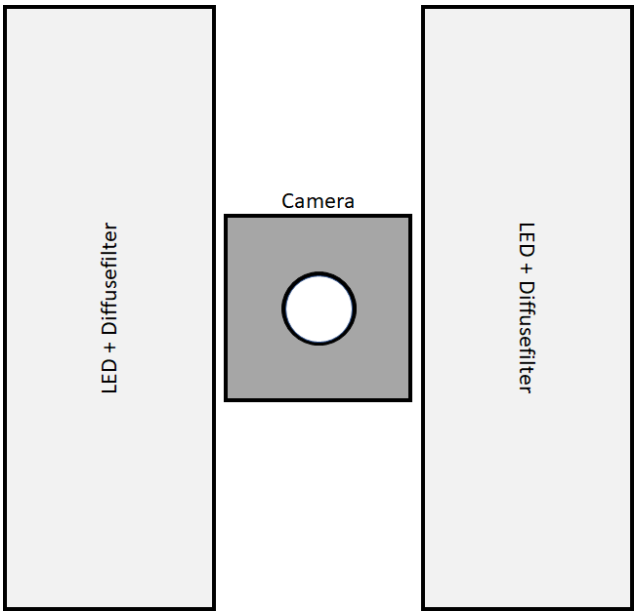
De kostprijsbepaling gebeurt o.b.v. de mechanische en elektrische onderdelen samen met de prijs voor de licentie van één van beide softwarepakketten. Via online prijsoffertes verstrekken verschillende leveranciers informatie i.v.m. kostprijs van de verschillende onderdelen. Voor de onderdelen die specifiek gepaard gaan met visie wordt er een prijsofferte aangevraagd bij Phaer. Een uitgebreide kostprijsbepaling is onder 3.5 terug te vinden.

De haalbaarheidsanalyse gebeurt o.b.v. het opmeten van een aantal foutieve beugels. Beide programma’s meten de hoeken op. Door deze opgemeten hoeken te vergelijken met de werkelijke hoekgrootte kunnen er o.b.v. de afwijkingen van de hoeken conclusies getrokken worden o.b.v. haalbaarheid voor elk programma.

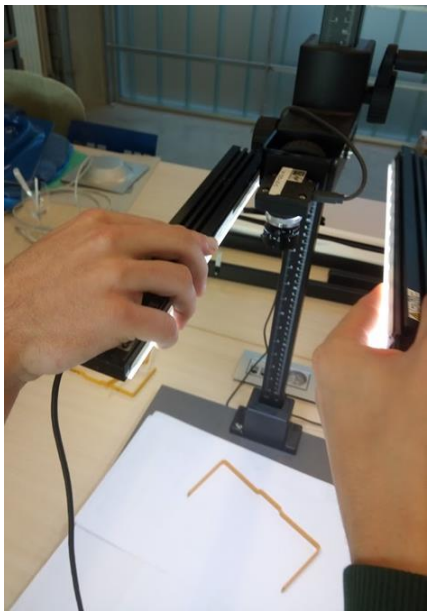
3.2 Testopstelling

Zoals in 3.1 besproken is, is de testopstelling nodig om controleprogramma's te testen en zo de mogelijkheden van elk visieprogramma te bekijken.

De principiële opbouw van de testopstelling is zichtbaar in figuur 32, terwijl figuur 33 weergeeft hoe er praktisch te werk is gegaan om de foto's te verzamelen voor de foutieve en de correcte beugels.



Figuur 32: Principiële opbouw testopstelling



Figuur 33: Praktische opbouw testopstelling

Uit de testen blijkt dat er nood is aan een camera met twee LED-bars met bijhorende diffusiefilters. De gebruikte camera voor het testen was UI 1225le c. De specificaties van deze camera zijn terug te vinden in bijlage A. De camera is uitgerust met een lens die een manueel aanpasbaar diafragma en scherptestelling heeft. De afmetingen van deze lens zijn 1:1.4 8mm Ø25.5. Hier beschrijft 1.4 de diafragmaopening en 8mm de brandpuntsafstand van de lens.

Om een bruikbaar beeld te verkrijgen, is de belichting een cruciale factor. Na een aantal testen met verschillende belichtingsmogelijkheden (LED-ring, omgevingslicht, LED-bars, etc.), blijkt dat twee LED-bars met een diffusiefilter het beste resultaat geven.

3.3 Software

3.3.1 Merlic

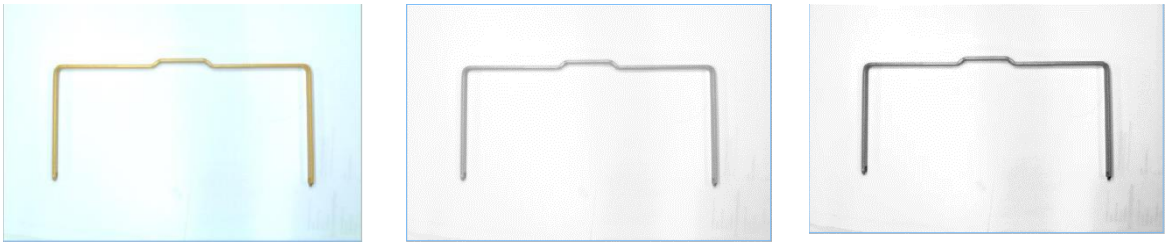
3.3.1.1 Globaal overzicht

Figuur 34 toont schematisch de functies die het programma doorloopt om de beugels te controleren. Zoals op onderstaande figuur te zien is, is het programmaverloop opgedeeld in vijf delen.

Figuur 34: Opbouw Merlicprogramma

3.3.1.2 Preprocessing

Binnen kader één van figuur 34 zal eerst een beeld van de camera ingelezen worden waarna het met enkele bewerkingen omgezet wordt tot een meer bruikbaar, duidelijker beeld. De foto's in figuur 35 (a), (b) en (c) tonen de evolutie van het beeld wanneer die het programma in kader 1 doorloopt. Het nemen van afbeeldingen is in de testfase nog in freerun-mode gebeurd maar in de finale opstelling zal een sensor de camera triggeren wanneer de beugel op een juiste positie onder de camera ligt.



(a) Initiële afbeelding

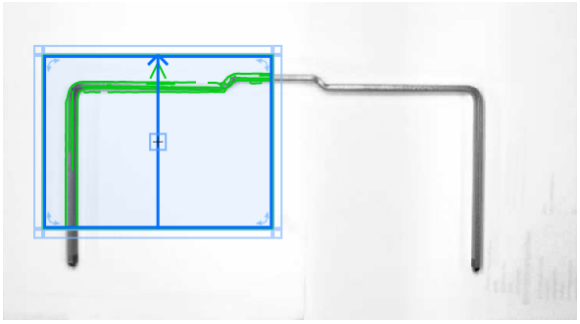
(b) Initiële afbeelding omgezet naar grijswaardes

(c) Pixels converteren

Figuur 35: Preprocessingstappen binnen Merlic

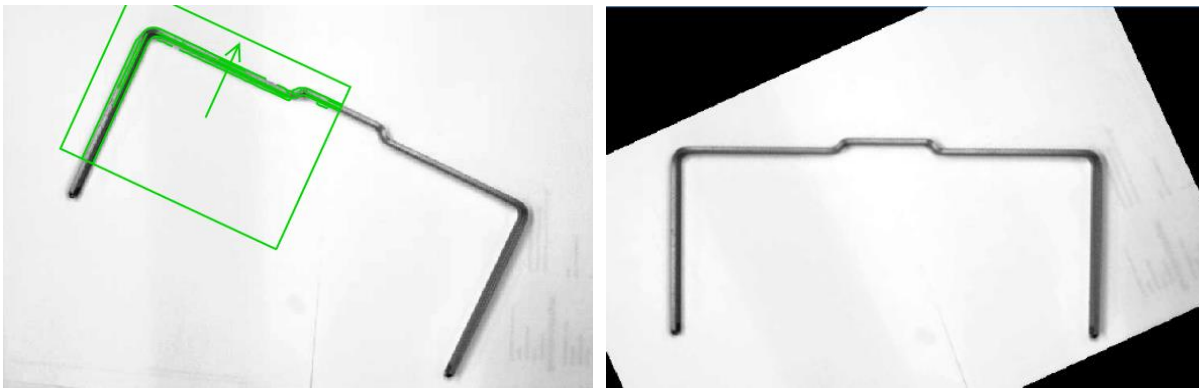
3.3.1.3 Aligneren

Omdat de beugels verschillende oriëntaties hebben wanneer ze op de band liggen, is er nood aan een bewerking die elk beeld zo draait dat de beugels dezelfde oriëntatie hebben. Het programmagedeelte dat hier verantwoordelijk voor is, bevindt zich in kader twee van figuur 34. Dit is te realiseren door het genomen camerabeeld te vergelijking met een referentiebeeld dat op voorhand wordt vastgelegd. Door een bepaalde kenmerkende zone of vorm vast te leggen in dit referentiebeeld zal het programma bij elk voorgeschoteld beeld op zoek gaan naar die kenmerkende zone met een bepaalde oriëntatie. Figuur 36 toont een voorbeeld van een potentieel kenmerkende zone.



Figuur 36: Referentiebeeld met geselecteerde kenmerkende zone

Indien het programma een zone kan vinden die voldoende overeenkomt met die van het referentiebeeld, zal het beeld volgens de oriëntatie van de kenmerkende zone gealigneerd worden. Figuur 37 (a) toont het camerabeeld en figuur 37 (b) het gealigneerde camerabeeld. Deze methode is vrij waterdicht. Moest de oriëntatie van een beugel toch lichtjes afwijken, zal dat geen probleem vormen. Volgende topics gaan hier verder op in.

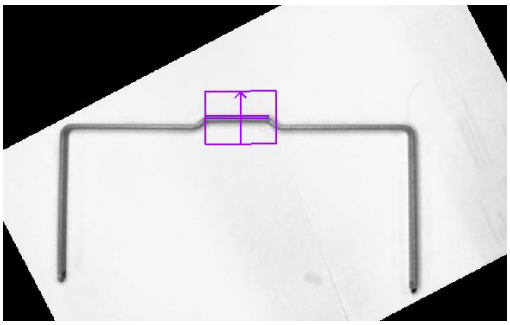


(a) voor alignatie *(b) na alignatie*

Figuur 37: Camerabeeld beugel

3.3.1.4 Meting lipje

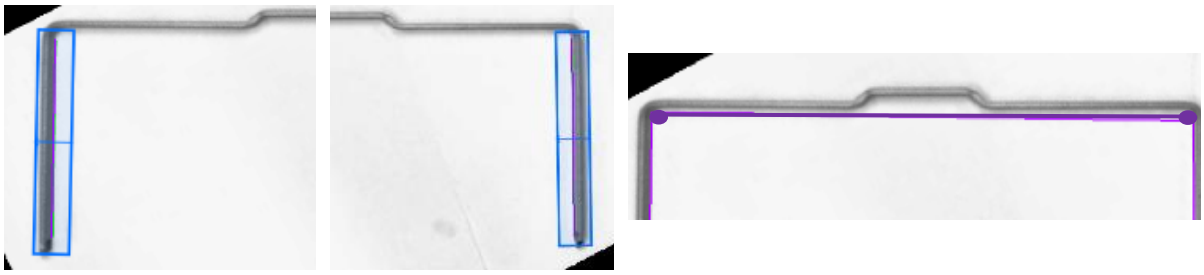
Nu het beeld volledig geoptimaliseerd is, kunnen de effectieve controles beginnen. Het derde kader van figuur 34 staat in voor het controleren van de lengte van het lipje. Deze meting is zichtbaar in figuur 38. De paarse kader is een vooraf gekozen zone. Binnen deze zone gaat Merlic op zoek naar een segment. Door het kader groot genoeg (niet te groot) te kiezen, is het mogelijk de eerdergenoemde potentiële oriëntatieafwijking te compenseren. Zelfs wanneer het lipje schuin in de paarse kader zou liggen, zou Merlic nog altijd het juiste segment vinden.



Figuur 38: Meting lipje

3.3.1.5 Meting breedte

Het vierde kader van figuur 34 bevat de functieblokken die de afstand tussen de twee beentjes controleert. Het is belangrijk op te merken om de afstand aan de rug te bekijken en niet aan de uiteinden van de beentjes want dit zou foute meetresultaten kunnen geven als de beentjes schuin geplooid zijn. figuur 39 (a) en (b) tonen hoe Merlic binnen een vooraf vastgelegde rechthoek (blauw) een segment (paarse lijn) van respectievelijk het linker- en rechterbeen zoekt. De afstand tussen deze segmenten zal, na het omrekenen van pixel-coördinaten naar mm, een waarde geven voor de breedte van de beugel. Deze breedtemeting is te zien in figuur 39 (c).

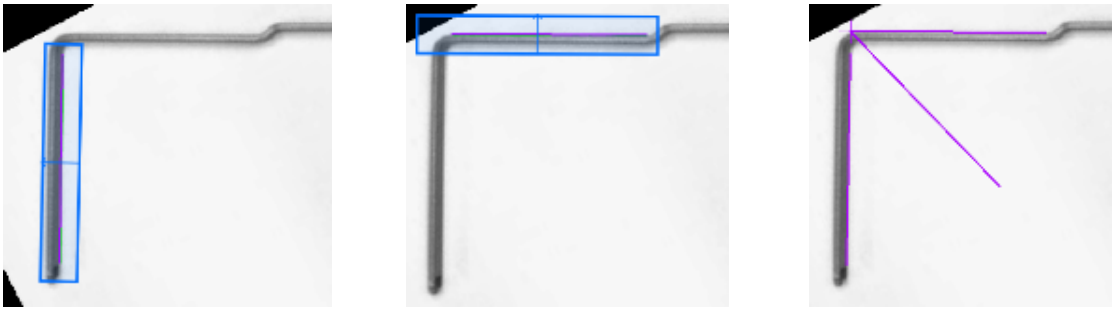


(a): Segment linkerbeen *(b): Segment rechterbeen* *(c): Breedtemeting*

Figuur 39: Breedtemeting binnen Merlic

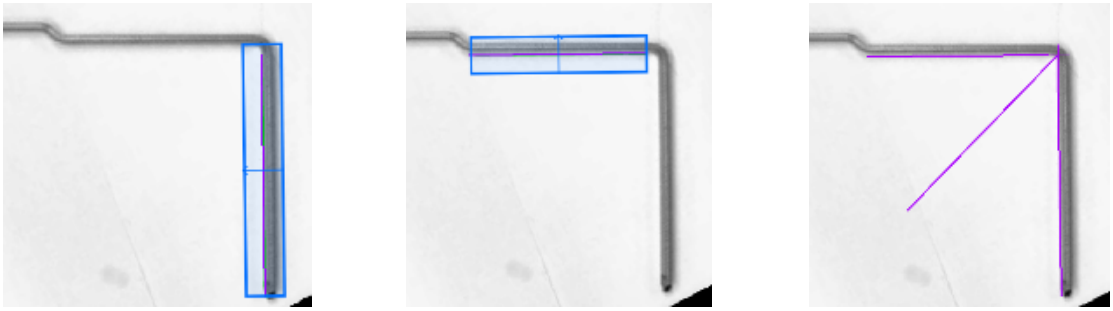
3.3.1.6 Meting hoeken

Het programmagedeelte binnen het vijfde kader van figuur 34 zal de twee hoeken van de beugel controleren op loodrechtheid. Analoog aan de methode bij 3.3.1.5 zoekt Merlic twee segmenten. Dit keer om er de hoek tussen te bepalen. Figuur 40 en figuur 41 verduidelijken dit principe.



(a): Segment linkerbeen *(b): Segment bovenbeen* *(c): Hoekmeting links*

Figuur 40: Hoekmeting links

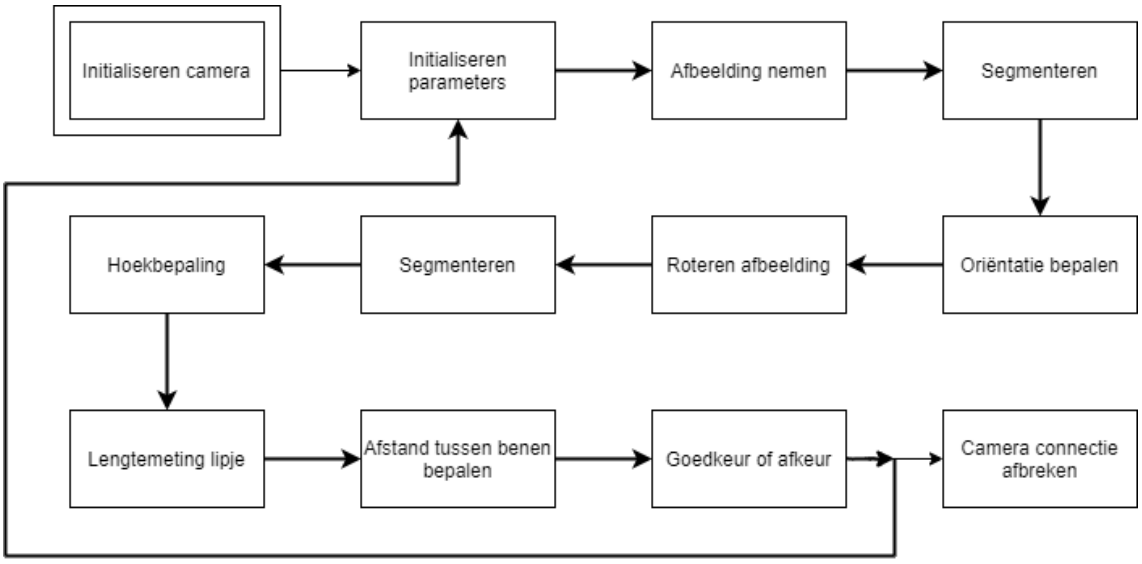


(a): Segment rechterbeen *(b): Segment bovenbeen* *(c): Hoekmeting rechts*

Figuur 41: Hoekmeting rechts

3.3.2 Halcon

De principiële afloop van de code staat beschreven in figuur 42. Het programma begint met het initialiseren van de camera en parameters. Een deel van deze parameters moet elke cyclus opnieuw geïnitialiseerd worden. Na de initialisatie volgt het nemen van beelden en het bepalen van de oriëntatie. O.b.v. de gevonden oriëntatie ondergaan de beelden een rotatie zodat ze recht liggen. De volgende stap is het segmenteren van de afbeelding met als doel om metingen uit te kunnen voeren. Er vinden drie metingen plaats, zijnde de hoekmetingen, de breedtemeting en de lengtemeting van het lipje. De gegevens van deze metingen liggen aan de basis van het oordeel of een beugel goed of slecht is. Deze cyclus herhaalt zich voor de volgende beugel, waarbij het niet meer nodig is om de stap 'Initialiseren camera' te doorlopen. Bij het beëindigen van het programma breekt het programma de connectie met de camera af.



Figuur 42: Principiële afloop Halconcode project 1

3.3.2.1 Initialisatie

De initialisatiestap is de eerste stap van het Halcon controleprogramma en is te zien in figuur 43.

```
*
*
* Image Acquisition 01: Code generated by Image Acquisition 01
close_all_framegrabbers ()
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, 'false', 'default', '1', 0, -1, AcqHandle)
grab_image_start (AcqHandle, -1)
*while (true)
grab_image_async (Image, AcqHandle, -1)
get_image_size (Image, Width, Height)
init_visualization (Image, 3, 'white', 'margin', Width, Height, WindowID)

*Schaalfactor voor het schalen van afstand op foto naar afstand op
DistanceScale := 38.61
HoekOndergrens := 89.82
HoekBovengrens := 90.18
Schalingsfactor :=2
MaxDistanceBeentjes := 180
MinDistanceBeentjes :=177.5
```

Figuur 43: Initialisatie camera

Eerst moet de camera zich ontkoppelen van eventueel andere applicaties buiten Halcon. Er kan namelijk maar één verbinding zijn tussen de camera en software (*close_all_framegrabbers*). Hierna volgt het daadwerkelijk openen en configureren van de camera (*open_framegrabber*). Het nemen van een beelden gebeurt door andere operatoren (*grab_image_start* & *grab_image_async*) [45]. Deze operatoren genereert Halcon automatisch bij het initialiseren van een camera. Deze manier van beelden verkrijgen, is ideaal voor ontwikkeling van de code. Bij implementatie te Metes zal het programma werken met synchrone beeldname, getriggered door een naderingssensor wanneer de beugel op een juiste plaats

onder de camera ligt. Dit komt neer op het verwerken van een afbeelding terwijl de camera tegelijkertijd nieuwe beelden neemt [46].

De volgende stap is het bepalen van de grootte van de afbeelding (*get_image_size*). Deze grootte is nodig voor het initialiseren van een scherm om het genomen beeld in te tonen (*init_visualization*). Door de afmetingen van de afbeelding niet te wijzigen, blijven de verhoudingen correct behouden tijdens de visualisatie. Deze operatoren samen met het aanmaken van variabelen zitten in een *while-loop* om cyclisch te kunnen werken. Het begin van deze while-loop is in figuur 44 zichtbaar. De variabele '*ThresholdValue*' legt de *threshold*-waarde vast die later nog nodig is om de afbeelding te segmenteren en aangezien de condities van de opstelling altijd dezelfde blijven, is dit een constante waarde.

```
while (true)
      * Image Acquisition 01: Do something

    *Afbeelding grabben + threshold → openingcircle en dilliationcircle = betere contour verkrijgen*
    **
    ThresholdValue := 200
    DilationCircleValue:=1

    * Var voor While verderop
    EdgeBereikt1 := 0
    EdgeCor1 :=[0,0]
    EdgeBereikt2 := 0
    EdgeCor2 :=[0,0]

    *Schaalfactor voor het schalen van afstand op foto naar afstand op
    grab_image_async (Image, AcqHandle, -1)
    get_image_size (Image, Width, Height)
```

Figuur 44: Aanmaken lege variabelen en nemen nieuwe afbeelding

3.3.2.2 Schaling

Om bepaalde lengtes uitgedrukt in pixels om te zetten naar lengtes in mm is een schaling nodig. Deze code is te zien in figuur 45.

```
    dev_display (Image)
    dev_disp_text ('Click here for calibration', 'window', Width/2, Width/4, 'black', [], [])
    dev_disp_text ('Click here for measuring', 'window', Width/2, Width*3/4, 'black', [], [])

    get_mbutton (WindowID, RowClick, ColumnClick, Button)
    if(ColumnClick<Width/2)
        ScalingFactor := true
    endif
    if(ColumnClick>Width/2)
        ScalingFactor := false
    endif
```

Figuur 45: Keuze voor schaling

Binnen dit stuk code kan een ingenieur kiezen om enerzijds de schalingsfactor in te stellen (m.b.v. een referentiebeugel) of anderzijds een schaling te doen op verkregen resultaten. Voor deze keuze zijn twee knoppen voorzien. Met *get_mbutton* weet Halcon welke knop ingeduwd is en ook waar er geklikt is. De mogelijkheid bestaat om deze klikmodule na de integratie van het systeem te vervangen door een drukknop waar de ingenieur/operator kan kiezen of een object een referentie-object is of een te controleren beugel.

3.3.2.3 Preprocessing

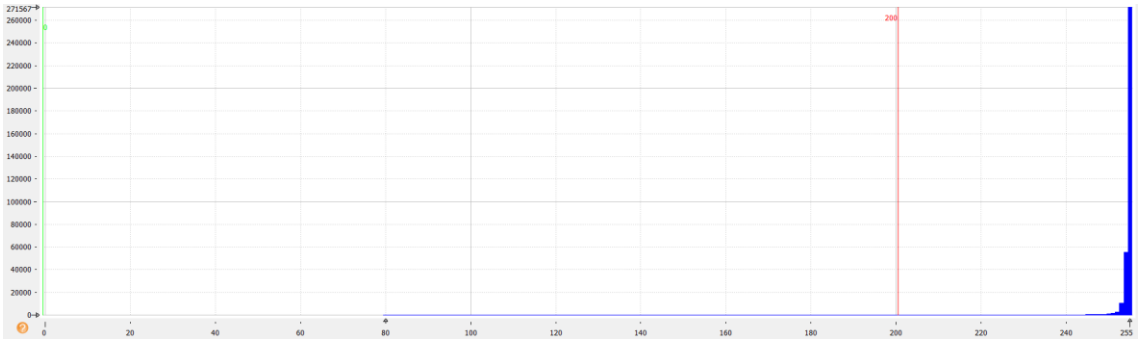
In onderstaande figuur is de preprocessingstap te zien. Allereerst segmenteert het programma de nuttige informatie uit de afbeelding om vervolgens de contouren van de beugel te bepalen o.b.v. de verkregen regions.

```
55    *filter
56    gauss_image (Image, ImageGauss, 5)
57    *threshold op bepaalde channel/grijswaarden
58    access_channel (ImageGauss, ImageChannel3, 3)
59    threshold (ImageChannel3, Regions, 0, ThresholdValue)
60    *kleine Regio's verwijderen
61    opening_circle (Regions, RegionOpening, 2)
62    *kleine gaten in contour samennemen
63    dilation_circle (RegionOpening, RegionDilation1, DilationCircleValue)
64    remove_noise_region (RegionDilation1, OutputRegion, 'n_48')
65    ***
66    *Contour verkrijgen
67    ***
68
69    gen_contour_region_xld (OutputRegion, Contours, 'center')
```

Figuur 46: Segmentatie afbeelding

De eerste operator (*gauss_image*) zorgt voor vermindering van de ruis door een Gaussfilter toe te passen. Deze filter zal de overgang van een achtergrond naar het voorwerp beperken en zal de scherpte behouden, wat echter niet het geval is bij de filter *mean_image* [10].

De tweede stap is *access_channel*, wat gepaard gaat met *threshold*. De threshold dient om een afbeelding te segmenteren. Dit gebeurt op basis van grijswaarden. *Access_channel* extraheert een bepaald kanaal van het kleurenbeeld. Een kleurenbeeld bestaat uit drie kanalen: het rode, het groene en het blauwe kanaal. In deze code moet *access_channel* channel 3, oftewel de grijswaarden van het blauwe kanaal, extraheren. Hoe meer blauw er zich in de afbeelding bevindt, des te hoger de grijswaarde is, tot maximaal 255. De blauwe schijn van de achtergrond waarop de beugel ligt, rechtvaardigt de keuze voor channel 3. Hierna zet de threshold, o.b.v. het gekozen channel, de afbeelding om naar een regio. De waarde van de threshold is proefondervindelijk vastgelegd a.d.h.v. een histogram [47] en in de variabele *ThresholdValue* gestoken, die eerder al uitgelegd is in 3.3.2.1. Het is niet altijd nodig om een channel te kiezen, maar voor deze specifieke situatie was dit de beste oplossing. De histogram voor het vastleggen van de thresholdwaarde is te zien in figuur 47. Hier is duidelijk zichtbaar dat er een grote piek is bij 245 tot 255.



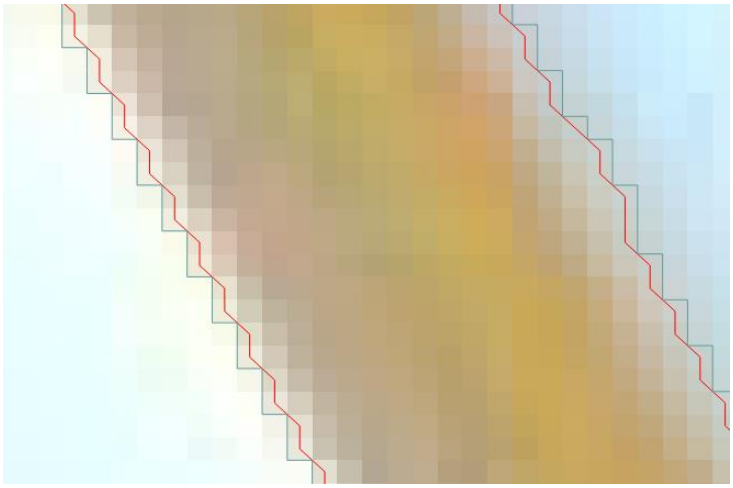
Figuur 47: Histogram van channel 3

Kleine structuren die eveneens tijdens de threshold worden aangemaakt, worden ook verwijderd (*opening_circle*). Zo kunnen onzuiverheden die eventueel op het ondervlak aanwezig zijn, verwijderd worden [48]. Hier wordt de straal van deze cirkels ook proefondervindelijk vastgesteld.

Daarna worden kleine gaten in de contour weggewerkt afhankelijk van de vooraf gestelde straal (*dilation_circle*). Hierbij worden er cirkelvormige regio's opgesteld met een vastgelegde straal. Indien de aanwezige gaten in de contour kleiner zijn dan deze cirkels worden deze tot de contour zelf gerekend [49]. Deze straal is ook hier weer proefondervindelijk vastgesteld.

Wegfilteren van ruis op de regio na het uitvoeren van de *dilation_circle* gebeurt via *remove_noise_region*. De derde parameter die moet worden meegegeven is de mode van ruisverwijdering. In deze toepassing is er gekozen voor n_48 aangezien deze alle pixels rond de geselecteerde pixel bekijken [50].

Er kan een contour worden gegenereerd uit de bestaande regio's (*gen_contour_region_xld*). De laatste parameter is gekozen als center. Deze neemt de centrums van de pixelrand en gebruikt deze dan als contourpunten [51]. Hierbij wordt er een betere afronding van de gehoekte regio bekomen. Dit wordt weergegeven in figuur 48, hierbij is de blauwe lijn, de contour die wordt bekomen door de rand van de regio te nemen en is de rode lijn deze van de centers. Hier is duidelijk zichtbaar dat de rode beter de beugel benaderd. Dit komt doordat de rode lijn zich op subpixelniveau bevindt.

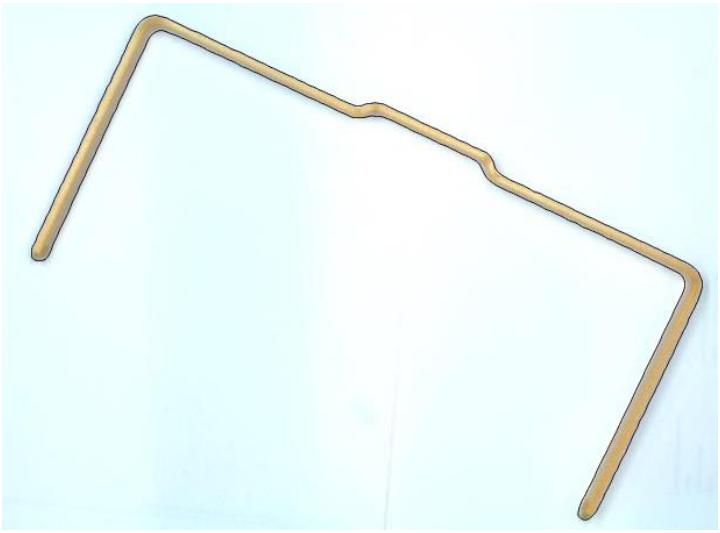


Figuur 48: Verschil tussen center en border van pixels

De laatste operator die gebruikt wordt om de nodige vorm uit de foto te segmenteren is *smooth_contours_xld*. Deze operator zorgt voor de afvlakking van de contour [52]. Dit is zichtbaar in figuur 49, hierbij stelt de rode lijn de gegenereerde contour voor en de zwarte lijn is de uitkomst na het uitvoeren van *smooth_contours_xld.*

Figuur 49: Effect van smooth_contours_xld

Het resultaat na de segmentatie is zichtbaar in figuur 50. Hierbij vormt de zwarte rand rond de beugel de contour die het resultaat is van de voorgaande stappen.



Figuur 50: Eindresultaat segmentatie

De volgende stap na het segmenteren van de afbeelding is het vinden van het middelpunt en de oriëntatie van de beugel. Dit is te zien in figuur 51.

```
73    ***
74    *Middelpunt van fig vinden + orientatie
75    ***
76    area_center (OutputRegion, AreaRegion, RowCenterRegion, ColumnCenterRegion)
77    orientation_region (OutputRegion, Rotatie)
78    dev_set_color ('black')
79    disp_cross (WindowID, RowCenterRegion, ColumnCenterRegion, 10, 0)
80    stop()
```

Figuur 51: Bepalen centrum en rotatie

Het eerste wat er in figuur 51 gebeurt, is het middelpunt bepalen o.b.v. de verkregen regio. De output is het rij- en kolomcoördinaat van het middelpunt. De oriëntatie wordt verkregen met *orientation_region*. De verkregen oriëntatie is de verdraaiing van de afbeelding, uitgedrukt in radialen.

3.3.2.4 Rotatie en segmentatie

Het draaien van de afbeelding is nodig voor het uitvoeren van de stappen die volgen. De werking van het draaien staat weergegeven in figuur 52.

```
84    ***
85    *afbeelding draaien
86    ***
87    dev_clear_window()
88    dev_open_window (0, 0, Width, Height, 'black', WindowHandle)
89    *Rotatiehoek omrekenen naar graden
90    tuple_deg (Rotatie, RotatieGrad)
91    *Afbeelding draaien
92    rotate_image (Image, ImageRotate, -RotatieGrad, 'constant')
```

Figuur 52: Draaien van afbeelding

De oriëntatie bepaald in figuur 51 staat in radialen maar alvorens de afbeelding te draaien moet deze hoek in graden staan [53]. Dit kan met *tupl_deg*. Nu kan het roteren van de afbeelding gebeuren (*rotate_image*). Hierbij zijn twee mogelijke posities, beiden te zien zijn in figuur 53 en figuur 54.

Figuur 53: Rotatiemogelijkheid 1

Figuur 54: Rotatiemogelijkheid 2

De stappen van de segmentatie zijn identiek als in 3.3.2.3. Bij het segmenteren na rotatie is er nog een extra laatste stap toegevoegd. De code voor het uitvoeren van deze segmentatie is te zien in figuur 55.

```
95    ***
96    * Segmenteren
97    ***
98    *threshold op bepaalde channel/grijswaarden
99    access_channel (ImageRotate, ImageChannel3_2, 3)
100   threshold (ImageChannel3_2, Regions2, 0, ThresholdValue)
101   *te kleine Regio's weg gooien
102   opening_circle (Regions2, RegionOpening2, 0.5)
103   *kleine gaten in contour samennemen
104   dilation_circle (RegionOpening2, RegionDilation2, DilationCircleValue)
105   remove_noise_region (RegionDilation2, OutputRegion1, 'n_48')
106   *contour verkrijgen
107   gen_contour_region_xld (OutputRegion1, Contours2, 'center')
108   smooth_contours_xld (Contours2, SmoothedContours2, 5)
109   *Selecteren van de contour
110   select_contours_xld (SmoothedContours2, SelectedContours, 'contour_length', 30, 10000, -0.5, 0.5)
```

Figuur 55: Segmentatie na rotatie

In de laatste regel van figuur 55 staat *select_contours_xld.* Deze operator selecteert alle contouren die aan de voorwaarden voldoen. De parameter *'contour_length'* geeft alle contouren weer die een lengte hebben tussen de 30 en 10 000 [54]. Deze operator is gebruikt om de kleine contouren, gevormd door ruis, niet mee te nemen in de verdere verwerking van de afbeelding. De maximale grens zou oneindig mogen zijn doch is dit niet mogelijk en is de waarde 10 000 tevens groot genoeg.

De code getoond in figuur 56 geeft een controle stap weer. Hier gebeurt het controleren van de oriëntatie van de beugel.

```
112   *centrum van regio bepalen
113   area_center (OutputRegion1, AreaRegion, RowCenterRegion2, ColumnCenterRegion2)
114   *orientatie vastleggen
115   orientation_region (OutputRegion1, Rotatie2)
116   dev_set_color ('blue')
117   *display van een kruis op de centrum positie
118   disp_cross (WindowHandle, RowCenterRegion2, ColumnCenterRegion2, 10, 0)
```

Figuur 56: Bepalen van centrum en rotatie

3.3.2.5 Bepalen kleinste rechthoek

Smallest_rectangle2 bepaalt de kleinste rechthoek die een regio omringt. Voor deze rechthoek wordt het middelpunt bepaald. De parameters '*Row_Region2*' en '*Column_Region2*' bevatten de coördinaten van dit middelpunt. '*Length1_Region2*' stelt de helft van de lengte van de rechthoek voor en

'*Length2_Region2*' stelt de helft van de breedte van de rechthoek voor. In figuur 57 is de code te zien om deze te bepalen.

```
120    *Kleinste rechthoek door regio
121    smallest_rectangle2 (OutputRegion1, Row_Region2, Column_Region2, Phi_Region2, Length1_Region2, Length2_Region2)
122    *Tekenen van rechthoek
123    gen_rectangle2 (Rectangle2, Row_Region2, Column_Region2, Phi_Region2, Length1_Region2, Length2_Region2)
124    stop()
```

Figuur 57: Bepalen kleinste rechthoek

De volgende stap genereert een rechthoek met de verkregen gegevens (*gen_rectangle2*). Deze dient enkel als visuele ondersteuning.

3.3.2.6 Opstellen horizontale en verticale snijlijnen

Voor de hoekbepaling worden er eerst horizontale en verticale snijlijnen opgesteld. Deze lijnen worden afgeleid uit de kleinste rechthoek die in 3.3.2.5 is besproken. Deze snijlijnen zijn nodig om cameracoördinaten te bepalen. Aan de hand van deze coördinaten kunnen er lijnen worden opgesteld die op de benen van de beugel liggen, waaruit de hoek gemeten kan worden. Het opstellen van de lijnen is te zien in figuur 58.

```
127    *Lijn1 → Horizontaal
128    **
129    *opstellen van coordinaten → rij (1 cordi=horiz) en col (2 cordi=verti)
130    RijCor0 := Row_Region2+Length1_Region2/4
131    ColBeg0 := Column_Region2 + 3*Length2_Region2
132    ColEin0 := Column_Region2 - 3*Length2_Region2
133
134    dev_set_color('orange')
135    disp_line (WindowHandle, RijCor0,ColBeg0, RijCor0, ColEin0)
```

Figuur 58: Horizontale lijn boven centerpunt

Voor het tekenen van een horizontale lijn, is er enkel nood aan één rijcoördinaat en twee kolomcoördinaten. De coördinaat voor de rij wordt bepaald door het center van de rechthoek op te tellen bij een kwart van de lengte van deze rechthoek. De ligging van de kolomcoördinaten is niet heel kritisch, zolang deze coördinaten verder liggen dan de contour is, is dit in orde. Daarom wordt hier gerekend met drie maal de helft van de breedte. Met *disp_line* kan deze lijn op de figuur worden getoond. Deze werkwijze is hetzelfde voor de andere horizontale lijn, die boven het centerpunt van de rechthoek komt te liggen. De werkwijze is zichtbaar in figuur 59.

```
137    *Lijn2 → Horizontaal
138    **
139    *opstellen van coordinaten → rij (1 cordi=horiz) en col (2 cordi=verti)
140    RijCor1 := Row_Region2-Length1_Region2/3.5
141    ColBeg1 := Column_Region2 + 3*Length2_Region2
142    ColEin1 := Column_Region2 - 3*Length2_Region2
143    disp_line (WindowHandle, RijCor1,ColBeg1, RijCor1, ColEin1)
```

Figuur 59: Horizontale lijn onder centerpunt

Hierbij wordt de rijcoördinaat bepaald door de halve lengte van de rechthoek te delen door een waarde, die proefondervindelijk is vastgelegd en vervolgens af te trekken van de centerpuntcoördinaat. De coördinaten voor de kolomcoördinaten blijven dezelfde.

In figuur 60 zijn de twee horizontale lijnen te zien. Hierbij is de oranje lijn de eerste horizontale lijn (figuur 58) en is de zwarte lijn (figuur 59) de tweede horizontale lijn.

Figuur 60: Horizontale lijnen

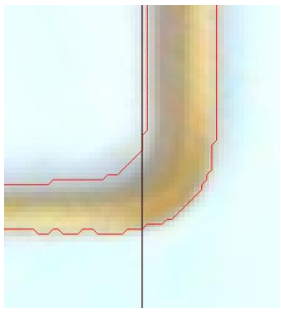
Verder worden de verticale lijnen bepaald. Hierbij is er nood aan één kolomcoördinaat en twee rijcoördinaten. De bepaling van de verticale lijnen is zichtbaar in figuur 61. De werkwijze is hier grotendeels hetzelfde enkel worden de proefondervindelijke parameters anders gekozen. Hierbij is het belangrijk dat de verticale lijnen de beentjes niet snijden of dat deze ver genoeg van de afronding liggen.

```
147     *Lijn3 → Verticaal
148     **
149     *opstellen van coordinaten → rij (2 cordi=horiz) en col (1 cordi=verti)
150     RijBeg0 := Row_Region2+Length1_Region2
151     RijEin0 := Row_Region2-Length1_Region2
152     ColCor0 := Column_Region2 + 1.7*Length2_Region2
153
154     disp_line (WindowHandle, RijBeg0, ColCor0, RijEin0, ColCor0)
155
156     *Lijn4 → Verticaal
157     **
158     *opstellen van coordinaten → rij (2 cordi=horiz) en col (1 cordi=verti)
159     RijBeg1 := Row_Region2+Length1_Region2
160     RijEin1 := Row_Region2-Length1_Region2
161     ColCor1 := Column_Region2 - 1.7*Length2_Region2
162
163     disp_line (WindowHandle, RijBeg1, ColCor1, RijEin1, ColCor1)
```

Figuur 61: Verticale lijnen

In figuur 62 zijn de verticale lijnen te zien op de afbeelding. De blauwe lijn is de eerste lijn (lijn3) van figuur 61 en de rode lijn is de tweede lijn (lijn4). Hierbij is het zeer belangrijk dat deze verticale ver genoeg van de beentjes van de beugel af liggen. Als deze te dicht bij de beentjes liggen, kunnen deze snijden. Dit snijden kan voorkomen aangezien de figuur niet perfect is gedraaid en hierdoor de beugel niet onder een hoek van nul graden ligt. Ook is het mogelijk dat de verticale lijn te dicht bij het beentje van de beugel ligt. Hierdoor zal er een minder nauwkeurige meting uitgevoerd worden. Dit is zichtbaar in figuur 63.

Figuur 62: Verticale lijnen



Figuur 63: Snijding verticale lijn te dicht bij afronding

3.3.2.7 Snijpunt tussen lijn en contour bepalen

Het bepalen van de snijpunten van de lijnen en de contour is te zien in figuur 64.

```
165    intersection_line_contour_xld(SelectedContours,RijCor0, ColBeg0, RijCor0, ColEin0, Row0, Col0, IsOverlapping)
166    intersection_line_contour_xld(SelectedContours,RijCor1, ColBeg1, RijCor1, ColEin1, Row1, Col1, IsOverlapping)
167    intersection_line_contour_xld(SelectedContours,RijBeg0, ColCor0, RijEin0, ColCor0, Row2, Col2, IsOverlapping)
168    intersection_line_contour_xld(SelectedContours,RijBeg1, ColCor1, RijEin1, ColCor1, Row3, Col3, IsOverlapping)
```

Figuur 64: Snijpunt bepalen

Voor de bepaling van de snijpunten wordt *intersection_line_contour_xld* toegepast. Deze bepaalt alle snijpunten van een contour en een lijn. Hierbij wordt de geselecteerde contour gebruikt en worden de coördinaten van de snijlijnen gebruikt die in de voorgaande stappen zijn vastgelegd. Hieruit komen de parameters ‘Row0’ en ‘Col0’, specifiek voor het eerste geval. In dit eerste geval worden de snijpunten van de onderste horizontale en de contour bekeken. Deze lijn snijdt de contour op 4 plaatsen, namelijk aan de binnen en buiten kant van de twee beentjes. Daarom zijn de outputparameters arrays van vier variabele. Verder op wordt er beschreven welke parameters belangrijk zijn.

Voor de derde en vierde component wordt er gekeken naar een verticale lijn. Hier zijn er ook outputcoördinaten die worden weergegeven in ‘Row2’ en ‘Col2’. Deze parameters bezitten geen vier variabelen maar twee. Deze snijden namelijk niet de beentjes maar enkel de rug zoals te zien is in figuur 62. Daarom zijn deze parameters arrays van twee variabelen. Voor het bepalen van de hoek is het nodig om de ligging van de beentjes en de rug te bepalen. Dit wordt in deze stap opgesteld en is te zien in figuur 65.

```
170    *Linkse lijn
171    Hoek_lijn_links:=[Row0[0], Col0[0], Row1[0],Col1[0]]
172    disp_cross(WindowHandle, Row0[0], Col0[0], 20, 0)
173    disp_cross(WindowHandle, Row1[0],Col1[0], 20, 0)
174
175    *Rechtse lijn
176    Hoek_lijn_rechts:=[Row0[3], Col0[3], Row1[3],Col1[3]]
177    disp_cross(WindowHandle, Row0[3], Col0[3], 20, 0)
178    disp_cross(WindowHandle, Row1[3],Col1[3], 20, 0)
179
180    *Horizontale  lijn
181    if(Row2[0] > 250)
182        Hoek_lijn_horizontaal:=[Row2[1], Col2[1], Row3[1],Col3[1]]
183        disp_cross(WindowHandle, Row2[1], Col2[1], 20, 0)
184        disp_cross(WindowHandle, Row3[1],Col3[1], 20, 0)
185    else
186        Hoek_lijn_horizontaal:=[Row2[0], Col2[0], Row3[0],Col3[0]]
187        disp_cross(WindowHandle, Row2[0], Col2[0], 20, 0)
188        disp_cross(WindowHandle, Row3[0],Col3[0], 20, 0)
189    endif
```

Figuur 65: Bepaling ligging beentjes en rug

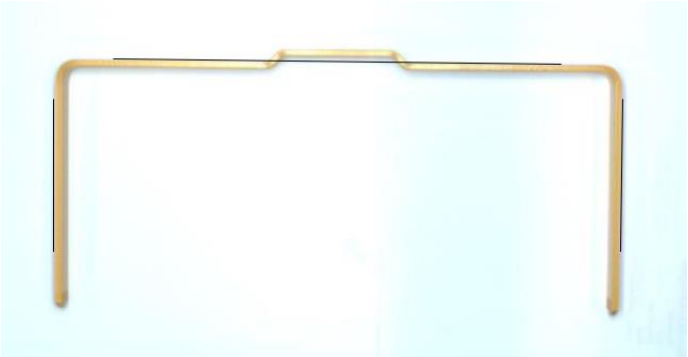
De eerste coördinaat van ‘Row0’ is het meest linkse snijdingspunt. Het laatste element van ‘Row0’ is dan weer het meest rechtse snijdingspunt. Dit zijn de twee posities die nodig zijn om een linkse en rechtse lijn te bepalen. Ter visualisatie worden hier de begin- en eindcoördinaten van de lijn voorgesteld met een kruisje (*disp_cross*). Dit kruisje wordt enkel gebruikt voor de ontwikkeling en controle van de software.

Bij de horizontale lijn, is het belangrijk dat er altijd de buitenste lijn wordt genomen aangezien deze meestal nauwkeuriger is dan de binnenste en aangezien voor de beentjes ook de buitenste contour is genomen. Om te controleren hoe de beugel is georiënteerd, wordt gebruik gemaakt van een if-lus. Deze kijkt dan of de hoogte van de rijcoördinaat hoger of lager is dan 250 (de helft van de hoogte van de afbeelding). Als deze waarde groter is dan 250 ligt de beugel zoals in figuur 54 en als deze lager is dan 250 dan ligt de beugel zoals in figuur 53. Het visualiseren van de lijnen is ook weer enkel nodig voor de visuele controle van de programmeur. Deze code is zichtbaar in figuur 66.

```
192    *lijnen visualiseren op afbeelding
193    dev_clear_window()
194    dev_display(ImageRotate)
195    disp_line (WindowHandle, Hoek_lijn_links[0], Hoek_lijn_links[1], Hoek_lijn_links[2], Hoek_lijn_links[3])
196    disp_line (WindowHandle, Hoek_lijn_rechts[0], Hoek_lijn_rechts[1], Hoek_lijn_rechts[2], Hoek_lijn_rechts[3])
197    disp_line (WindowHandle, Hoek_lijn_horizontaal[0], Hoek_lijn_horizontaal[1], Hoek_lijn_horizontaal[2], Hoek_lijn_horizontaal[3])
```

Figuur 66: Visualisatie lijnen

Figuur 67 is het resultaat van figuur 66. De lijnen op de beentjes en rug van de beugel zijn duidelijk zichtbaar gemaakt.



Figuur 67: Visualisatie buitenste lijnen beugel

3.3.2.8 Hoekbepaling

De code om de hoeken te bepalen is zichtbaar in figuur 68.

```
205    *Afhankelijk van figuur orientatie beginpunt van hoekmeting vastleggen
206    *Hoek beginpunten liggen 'samen'
207    *links
208    if(Hoek_lijn_horizontaal[0]>250)
209      disp_cross(WindowHandle, Hoek_lijn_links[0], Hoek_lijn_links[1],20, 0)
210      disp_cross(WindowHandle, Hoek_lijn_horizontaal[2], Hoek_lijn_horizontaal[3],20, 0)
211      angle_ll ( Hoek_lijn_links[0], Hoek_lijn_links[1],Hoek_lijn_links[2], Hoek_lijn_links[3], Hoek_lijn_horizontaal[2], Hoek_lijn_horizontaal[3],\
                 Hoek_lijn_horizontaal[0], Hoek_lijn_horizontaal[1], Loodrechtheidshoek1)
212    else
213      disp_cross(WindowHandle, Hoek_lijn_links[2], Hoek_lijn_links[3],20, 0)
214      disp_cross(WindowHandle, Hoek_lijn_horizontaal[2], Hoek_lijn_horizontaal[3],20, 0)
215      angle_ll ( Hoek_lijn_links[2], Hoek_lijn_links[3],Hoek_lijn_links[0], Hoek_lijn_links[1], Hoek_lijn_horizontaal[2], Hoek_lijn_horizontaal[3],\
                 Hoek_lijn_horizontaal[0], Hoek_lijn_horizontaal[1], Loodrechtheidshoek1)
216    endif
217
218    tuple_abs (Loodrechtheidshoek1, Loodrechtheidshoek1)
219    tuple_deg (Loodrechtheidshoek1,Loodrechtheidshoek1)
```

Figuur 68: Bepalen van hoek tussen linkse beentje en loodrechte deel

Angle_ll bepaalt de hoek tussen twee lijnen. Halcon beschouwt deze lijnen als vectoren. ‘Hoek_lijn_links[0]’, ‘Hoek_lijn_links[1]’ en ‘Hoek_lijn_horizontaal[2]’, ‘Hoek_lijn_horizontaal[3]’ vormen de startcoördinaten van de vectoren. De eindcoördinaten zijn dan respectievelijk ‘Hoek_lijn_links[2]’, ‘Hoek_lijn_links[3]’ en ‘Hoek_lijn_horizontaal[0]’, ‘Hoek_lijn_horizontaal[1]’. De juiste volgorde van de punten moet echter verzekerd zijn om de juiste hoek te bepalen. De output van deze operator staat in radialen. Afhankelijk van de beugelpositie is deze hoek positief of negatief. *Tuple_abs* en *tuple_deg* vormen deze output om naar een absolute waarde in graden.

Afhankelijk van de beugelpositie moet het beginpunt van de linkerlijn anders liggen. Bezit de beugel de positie zoals in figuur 69 dan zal het startpunt van de linkerlijn ook aan de bovenkant moeten liggen, zoals eveneens te zien is in deze afbeelding. Het is dus cruciaal dat de beginpunten bij elkaar liggen. Het andere geval is te zien in figuur 70 waar de startpunten links onder bij elkaar liggen.

Figuur 69: Startpunten links boven

Figuur 70: Startpunten links onder

De methode om de hoek te bepalen aan de rechterkant kent vanzelfsprekend een analoge methode als aan de linkerkant.

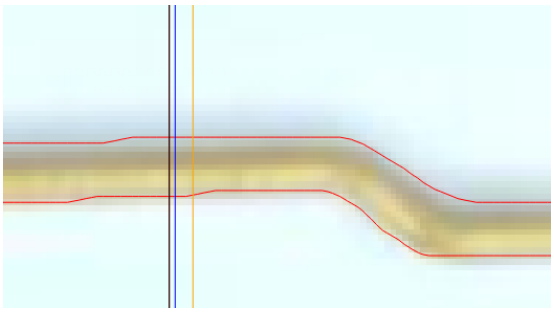
3.3.2.9 Lengtemeting lipje

De volgende stap in het controleproces is de lengtemeting van het lipje. Deze mag niet te kort zijn, aangezien het lipje instaat voor de goede hanteerbaarheid van de beugel. De aanpak is hier anders dan bij het meten van de hoek. Het meten van deze lengte gebeurt o.b.v. het veranderen van de helling van de rug van de beugel. Door te detecteren wanneer het lipje overgaat naar de rug van de beugel is het mogelijk om te bepalen hoelang het lipje precies is. De code is te zien in figuur 71.

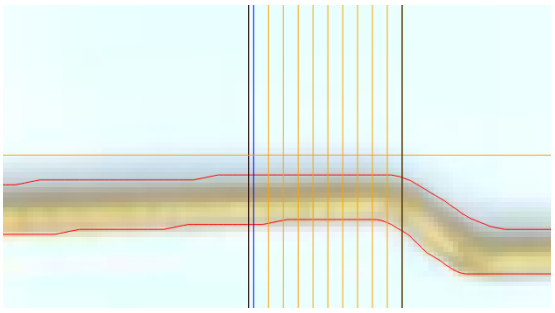
```
    ***
    *Lijn1 → verticaal => zoeken helling
    ***
    *opstellen van een verticale lijn die op een kleine afstand ligt van het uitstekende deel
    RijBeg_edge0 := Row_Region2+Length1_Region2
    RijEin_edge0 := Row_Region2-Length1_Region2
    ColCor_edge0 := Column_Region2 - 0.66*Length2_Region2
    *0.54 is ideaal
    disp_line (WindowHandle, RijBeg_edge0, ColCor_edge0, RijEin_edge0, ColCor_edge0)
    intersection_line_contour_xld(SelectedContours,RijBeg_edge0,ColCor_edge0,RijEin_edge0,ColCor_edge0,RowEdge0, ColEdge0, IsOverlapping1)
    disp_line (WindowHandle,RowEdge0[0], 0, RowEdge0[0], 400)

    while(EdgeBereikt1 == 0)
        dev_set_color('black')
        disp_line (WindowHandle, RijBeg_edge0, ColCor_edge0, RijEin_edge0, ColCor_edge0)
        intersection_line_contour_xld(SelectedContours,RijBeg_edge0,ColCor_edge0,RijEin_edge0,ColCor_edge0,RowEdge0, ColEdge0, IsOverlapping1)
        *disp_line (WindowHandle,RowEdge0[0], 0, RowEdge0[0], 400)

        if(RowEdge0[0]>250)
            if(EdgeCor1[0] == 0)
                EdgeCor1 := [RowEdge0[1],ColEdge0[1]]
                ColCor_edge0 := ColCor_edge0 +1
            elseif(RowEdge0 [1] > EdgeCor1[0]+0.8)
                EdgeBereikt1 :=1
            else
                EdgeCor1 := [RowEdge0[1],ColEdge0[1]]
                ColCor_edge0 := ColCor_edge0 +3
            endif
        endif

        if(RowEdge0[0]<250)
            if(EdgeCor1[0] == 0)
                EdgeCor1 := [RowEdge0[0],ColEdge0[0]]
                ColCor_edge0 := ColCor_edge0 +1
            elseif(RowEdge0[0]+0.8 < EdgeCor1[0] )
                EdgeBereikt1 :=1
            else
                EdgeCor1 := [RowEdge0[0],ColEdge0[0]]
                ColCor_edge0 := ColCor_edge0 +3
            endif
        endif

    endwhile
```

Figuur 71: Bepaling startcoördinaat schuine helling van lipje

Zoals in figuur 71 te zien is, wordt ook hier een verticale lijn aangemaakt. Deze heeft echter andere coördinaten dan de verticale lijn bij 3.3.2.6. De coördinaten moeten namelijk dichter bij het lipje liggen. Het vinden van deze coördinaten gebeurt proefondervindelijk. De volgende stap is het zoeken naar het snijpunt met de contour (*intersection_line_contour_xld*). Na het vinden van dit snijpunt volgt het bepalen van de helling. Als het programma de helling van de overgang detecteert, eindigt de while-lus. Daarna wordt er gekeken naar de oriëntatie van de beugel. Ligt deze met de rug naar boven, dan is 'RowEdge[0]' groter dan 250. Er wordt dan gekeken of dit de eerste meting is of niet. Als dit de eerste meting is, worden de waarden van 'RowEdge[0]' en 'ColEdge[0]' in 'EdgeCor1' geplaatst. Als dit niet de eerste meting is, controleert het programma de helling. Is die steil genoeg, dan stopt de while-lus en wordt de coördinaat behouden. Indien de helling nog niet steil genoeg is, slaat de software de snijcoördinaat op en wordt een nieuw kolomcoördinaat bepaald dat dichter bij de helling ligt. De bovenstaande stappen zijn te zien in figuur 72. Hierbij is de zwarte lijn de lijn die wordt vastgelegd bij het bepalen van de eerste coördinaten. Vervolgens is de blauwe lijn deze die verkregen wordt doordat de code nog niet is uitgevoerd. Tot slot is de oranje lijn deze waar de hoogte nog niet bereikt is en dus na deze lijn komt er een nieuwe iteratie. Deze iteraties gaan door tot het gewenste hoogte verschil bereikt is, zoals te zien is in figuur 73.



Figuur 72: Start zoektocht helling



Figuur 73: Einde zoektocht helling

De werking zoals hierboven uitgelegd is identiek dezelfde als deze voor 'RowEdge[0]' kleiner dan 250. Ook is deze werking identiek aan deze van de rechterkant. De code hiervoor is te zien in figuur 74.

```
    ***
    *Lijn2 → verticaal => zoeken helling
    ***
    *opstellen van een verticale lijn die op een kleine afstand ligt van het uitstekende deel
    RijBeg_edge1 := Row_Region2+Length1_Region2
    RijEin_edge1 := Row_Region2-Length1_Region2
    ColCor_edge1 := Column_Region2 + 0.66*Length2_Region2
    disp_line (WindowHandle, RijBeg_edge1, ColCor_edge1, RijEin_edge1, ColCor_edge1)
    intersection_line_contour_xld(SelectedContours,RijBeg_edge1,ColCor_edge1,RijEin_edge1,ColCor_edge1,RowEdge1, ColEdge1, IsOverlapping1)
    disp_line (WindowHandle,RowEdge1[0], 0, RowEdge1[0], 400)

    while(EdgeBereikt2 == 0)
        disp_line (WindowHandle, RijBeg_edge1, ColCor_edge1, RijEin_edge1, ColCor_edge1)
        intersection_line_contour_xld(SelectedContours,RijBeg_edge1,ColCor_edge1,RijEin_edge1,ColCor_edge1,RowEdge1, ColEdge1, IsOverlapping1)
        disp_line (WindowHandle,RowEdge1[0], 0, RowEdge1[0], 400)

        if(RowEdge1[0]>250)
            if(EdgeCor2[0] == 0)
                EdgeCor2 := [RowEdge1[1],ColEdge1[1]]
                ColCor_edge1 := ColCor_edge1 -1
            elseif(RowEdge1 [1] > EdgeCor2[0]+0.8)
                EdgeBereikt2 :=1
            else
                EdgeCor2 := [RowEdge1[1],ColEdge1[1]]
                ColCor_edge1 := ColCor_edge1 -3
            endif
        endif

        if(RowEdge1[0]<250)
            if(EdgeCor2[0] == 0)
                EdgeCor2 := [RowEdge1[0],ColEdge1[0]]
                ColCor_edge1 := ColCor_edge1 -1
            elseif(RowEdge1[0]+0.8 < EdgeCor2[0] )
                EdgeBereikt2 :=1
            else
                EdgeCor2 := [RowEdge1[0],ColEdge1[0]]
                ColCor_edge1 := ColCor_edge1 -3
            endif
        endif

    endwhile
```

Figuur 74: Bepaling eindcoördinaat schuine helling van lipje

Als beide coördinaten gevonden zijn, kan de afstand bepaald worden tussen deze twee. Deze afstandsbepaling wordt gedaan in figuur 75.

```
    dev_clear_window()
    dev_display (ImageChannel3_2)
    disp_line (WindowHandle,EdgeCor1[0],EdgeCor1[1],EdgeCor2[0],EdgeCor2[1])

    *berekend absolute waarde van uiterste cor van uitstekend deel
    tuple_abs(EdgeCor1[1]-EdgeCor2[1],LengteUitsteeksel)

    *uitvoeren als er geschaald moet worden
    if(ScalingFactor == true)
        DistanceScale := 44.68/LengteUitsteeksel
    endif

    *totale afstand van uitstekend deel
    TotDistance := DistanceScale * LengteUitsteeksel
```

Figuur 75: Afstandsbepaling tussen twee coördinaten

Hier wordt eerst weer ter visualisatie een lijn getekend die de twee coördinaten met elkaar verbindt. Aan de hand van *tuple_abs* wordt de absolute waarde van deze afstand bepaald en wordt deze afstand weggeschreven in de parameter 'LengeUitsteeksel'. Als nu de schalingsfactor op true staat, zoals in 3.3.2.2 vermeld is, zal de waarde geschaald worden naar de referentiewaarde van de beugel. Als dit niet zo is, wordt de schalingsfactor overgenomen van de vorige afbeelding.

3.3.2.10 Breedtemeting

De snijpunten van de evenwijdige van de horizontale raaklijn van de beugel en de contour bepaalt de breedte van de beugel. Het construeren van deze evenwijdige gebeurt volgens het algoritme in figuur 76. Het is hierbij belangrijk dat de richtingscoëfficiënt van de raaklijn gekend is, aangezien de rotatie van de beugel niet perfect is.

```
    ***
    *Afstand tussen 2 benen
    ***
    *eerst positie van afbeelding afvragen
    if(Hoek_lijn_horizontaal[0]>250)
        *Evenwijdige lijn met raaklijn aan horizontale deel trekken.
        rico_raaklijn_horizontaal := (Hoek_lijn_horizontaal[2] - Hoek_lijn_horizontaal[0]) / (Hoek_lijn_horizontaal[3] - Hoek_lijn_horizontaal[1])
        Row1_verschoven_lijn := rico_raaklijn_horizontaal * (0 - Hoek_lijn_horizontaal[1])+Hoek_lijn_horizontaal[0]-25
        Row2_verschoven_lijn := rico_raaklijn_horizontaal * (Width - Hoek_lijn_horizontaal[1])+Hoek_lijn_horizontaal[0]-25

    endif


    if(Hoek_lijn_horizontaal[0]<250)
        rico_raaklijn_horizontaal := (Hoek_lijn_horizontaal[2] - Hoek_lijn_horizontaal[0]) / (Hoek_lijn_horizontaal[3] - Hoek_lijn_horizontaal[1])
        Row1_verschoven_lijn := rico_raaklijn_horizontaal * (0 - Hoek_lijn_horizontaal[1])+Hoek_lijn_horizontaal[0]+25
        Row2_verschoven_lijn := rico_raaklijn_horizontaal * (Width - Hoek_lijn_horizontaal[1])+Hoek_lijn_horizontaal[0]+25

    endif
```

Figuur 76: Bepalen van evenwijdige met horizontale raaklijn

Het zoeken naar de snijpunten met de contour is te zien in figuur 77. De operator *intersection_line_contour_xld* helpt opnieuw met het vinden van de snijpunten tussen de contour en de lijn. Er zijn vier snijpunten met de contour, maar enkel die van de binnenkant van de beugel zijn gewenst. M.b.v. de wet van Pythagoras kan hieruit de tussenliggende afstand bepaald worden.

```
        disp_line (WindowHandle, Row1_verschoven_lijn, 0, Row2_verschoven_lijn, Width)
        intersection_line_contour_xld (SelectedContours, Row1_verschoven_lijn, 0, Row2_verschoven_lijn, Width, y_hoogtesnijden, x_breedtesnijden, IsOve
        dev_set_color ('red')
        tuple_pow ((y_hoogtesnijden[2]-y_hoogtesnijden[1]),2, y_verschil_kwadraat)
        tuple_pow ((x_breedtesnijden[2]-x_breedtesnijden[1]),2,x_verschil_kwadraat)

        tuple_sqrt (y_verschil_kwadraat+x_verschil_kwadraat,afstand_beentjes)

        disp_cross (WindowHandle, y_hoogtesnijden[2], x_breedtesnijden[2], 6, Phi_Region2)
        disp_cross (WindowHandle, y_hoogtesnijden[1], x_breedtesnijden[1], 6, Phi_Region2)

        if(ScalingFactor == true)
            Schalingsfactor := 178/afstand_beentjes
        endif

        Inside_distance_beentjes := Schalingsfactor * afstand_beentjes
```

Figuur 77: Breedtemeting tussen de beentjes

De optie om te schalen blijft ook hier behouden. Het delen van 178 en de gemeten waarde levert een schalingsfactor op. Is er geen schaling nodig, dan zal de vorige schalingsfactor hergebruikt worden. Om nu de effectieve afstand te bepalen, vermenigvuldigt deze factor met de gemeten afstand in pixels.

3.3.2.11 Beslissingsalgoritme

De keuze voor goed- of afkeur van de beugel gebeurt o.b.v. enkele voorwaarden waaraan de beugel moet voldoen.

```
402     if(42 < TotDistance and TotDistance < 60  and HoekOndergrens < Loodrechtheidshoek1 and Loodrechtheidshoek1 < HoekBovengrens and HoekOndergrens < Loodrechtheidshoek2
            and Loodrechtheidshoek2 < HoekBovengrens and Inside_distance_beentjes < MaxDistanceBeentjes and Inside_distance_beentjes>MinDistanceBeentjes)
403         dev_disp_text('De beugel voldoet', 'window', Height/2, Width/2, 'black', 'box_color', 'forest green')
404     else
405         dev_disp_text('De beugel voldoet niet', 'window', Height/2, Width/2, 'black', 'box_color', 'red')
406         if(42 > TotDistance)
407             dev_disp_text('Lipje is te klein', 'window',Height/2 + 50, Width/2 + 50,'white','box_color', 'black' )
408         endif
409         if(TotDistance > 60)
410             dev_disp_text('Lipje is te groot', 'window',Height/2 + 50, Width/2 + 50,'white','box_color', 'black' )
411         endif
412         if(HoekOndergrens > Loodrechtheidshoek1 or Loodrechtheidshoek1 > HoekBovengrens)
413             dev_disp_text('Hoek1 heeft een te grote afwijking '+Loodrechtheidshoek1, 'window',Height/2 + 100, Width/6 - 100,'white','box_color', 'black' )
414         endif
415
416         if(HoekOndergrens > Loodrechtheidshoek2 or Loodrechtheidshoek2 > HoekBovengrens)
417             dev_disp_text('Hoek2 is heeft een te grote afwijking '+Loodrechtheidshoek2, 'window',Height/2 + 100, Width/2 + 100,'white','box_color', 'black' )
418         endif
419
420         if(Inside_distance_beentjes>MaxDistanceBeentjes)
421             dev_disp_text('Afstand tussen beentjes is te groot', 'window', Height/4, Width/4, 'white','box_color','black')
422         elseif(Inside_distance_beentjes< MinDistanceBeentjes)
423             dev_disp_text('Afstand tussen beentjes is te klein', 'window', Height/4, Width/4, 'white','box_color','black')
424         endif
```

Figuur 78: Beslissingsalgoritme

Een eerste voorwaarde is dat de afstand tussen de beentjes voldoet aan volgende tolerantie: 178 mm ± 0,5. Het programma geeft het resultaat van de meting terug in pixels. Met de schalingsfactor (zie 3.3.2.10) gebeurt de omzetting naar mm. De grenzen zijn vastgelegd op 176 mm en 180 mm. Deze grenzen overschrijden de tolerantiegrenzen maar zijn bij te stellen indien er vóór de beeldname een 2D-kalibratie op de camera wordt doorgevoerd. De volgende voorwaarde heeft te maken met de loodrechtheid van de hoeken. De opgelegde tolerantie voor de hoeken is een loodrechtheid van 0,5 wat hier neerkomt op een maximale hoekafwijking van 0,18°. De onder- en bovengrens zijn dus respectievelijk 89,82° en 90,18°. Een laatste beoordelingscriterium is de lengte van het lipje dat 44,68 mm moet zijn. Analoog aan de eerste voorwaarde zal het resultaat van de meting na het in rekening brengen van de schalingsfactor een waarde in mm zijn. Ook hier zijn de grenzen vrij ruim gekozen, namelijk 42 tot 60mm maar in tegenstelling tot de eerste voorwaarde is deze lengte minder kritisch aangezien het lipje enkel instaat voor het grijpcomfort.

Indien de beugel niet voldoet aan één of meerdere voorwaarden toont het programma welke voorwaarden specifiek niet vervuld zijn. Zo zijn optredende fouten gemakkelijk en snel op te sporen.

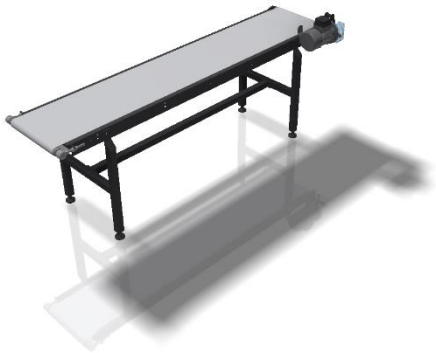
3.3.2.12 Doorlooptijd

De doorlooptijd van het programma is de benodigde tijd om het programma uit te voeren. De doorlooptijd begint te lopen bij de initialisatie en eindigt nadat het beslissingsalgoritme is uitgevoerd. De doorlooptijd bedraagt Beslissingsalgoritme 0,2 s.

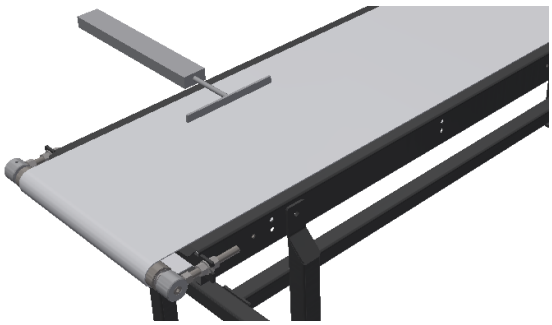
3.4 Uitwerking opstelling

Een andere doelstelling, naast de controle van de beugels, is dat de verwerking volledig automatisch gebeurt. Zo is er geen nood aan een extra operator. De uitwerking van de opstelling die hieronder volgt, is puur conceptueel en enkel om een idee te krijgen hoe het proces zou kunnen verlopen.

Een eerste onderdeel is een transportband. In figuur 79 is de conceptuele uitwerking voor de transportband te zien. Om de correcte onderdelen te onderscheiden van de foutieve is er een afwerpsysteem geïntegreerd. Deze is te zien in figuur 80. Dit afwerpsysteem bestaat uit een cilinder en duwplaat en krijgt de naam *pusher*.



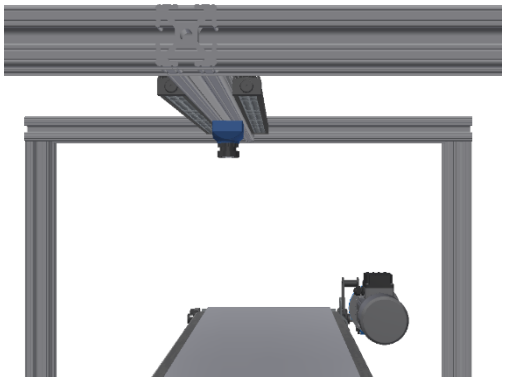
Figuur 79: Transportband



Figuur 80: Pusher

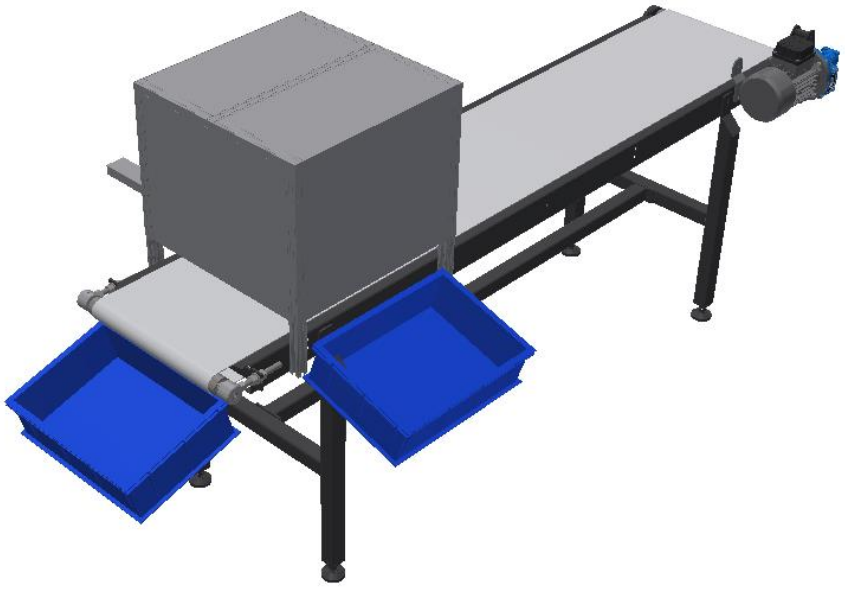
De camera hangt boven de transportband aan een aluminium frame. Aan dit frame hangen tevens de LED-bars. Het frame en de montage van de camera met de LED-bars zijn zichtbaar in respectievelijk figuur 81 en figuur 82.

Figuur 81: Transportband met frame



Figuur 82: Montage camera en LED-bars

Om invloeden van de omgeving te elimineren, is een omkasting geplaatst rond het frame. Zo valt er minder omgevingslicht op de beugels tijdens de beeldname. De hele conceptuele uitwerking van de transportband met visiesysteem is te zien in figuur 83. Er zijn tevens bakken voorzien om de beugels op te vangen.



Figuur 83: Volledige opstelling

Tenslotte is er een naderingssensor die detecteert of de beugels de camera passeren. Dit is nodig omdat een arbeider de beugels manueel op de transportband legt en er zo synchronisatieproblemen kunnen ontstaan. Met de aanwezigheid van deze naderingssensor is er geen synchronisatie nodig tussen de transportband en de camera.

3.5 Kostprijsbepaling

De camera-eenheid bestaat uit een camera met bijhorende lens, twee LED-bars, elk met een diffusiefilter en een voeding voor de lichtbron. Na navraag bij Phaer is een prijsofferte opgesteld om een correcte prijs te bepalen, welke terug te vinden is in bijlage B. De gebruikte camera tijdens de testfase was een UI-1225LE-C. Aangezien deze camera niet meer verkrijgbaar is, zal een gelijkaardige camera, een Sony IMX287, de basis vormen van de kostprijsbepaling. De technische gegevens van dit alternatief zijn terug te vinden in bijlage C. De totale prijs voor al deze onderdelen bedraagt €3779,85 incl. BTW.

Verder is nog een transportband [55] nodig. Een motor, voorzien van frequentieregelaar om de snelheid te regelen, drijft deze transportband aan [47]. Het regelen van de frequentieregelaar gebeurt met een PLC [56] met extra ingangskaart [57]. Een profinetswitch verzorgt de communicatie tussen de PLC en de PC voor de camera [58]. Om de beugel af te werpen, is er nood aan een pusher [59]. Tot slot is er naast de sensoren [60], om een onderdeel te detecteren, nog nood aan drukknoppen [61] en een noodstop [62]. De totale kostprijs van al deze onderdelen bedraagt €1319,70 incl. BTW.

Afhankelijk van het gekozen programma zal de prijs variëren. De aankoop van de software van Merlic werkt via een eenmalige aankoop. Hierbij is er keuze tussen *standard*, *advanced* en *professional*. Voor het werken met Merlic zal de keuze voor de standard volstaan. De licentie van Merlic standard heeft een kostprijs van €1883. Indien Metes Halcon prefereert, zijn er twee opties. Ten eerste is er de Halcon *runtime*-licentie. Deze biedt Metes de mogelijkheid om een Halcon programma te runnen en kost €691.

Ten tweede is er de licentie voor Halcon *development*. Deze heeft een kostprijs van €6200 maar geeft Metes wel de optie om nog aanpassingen te doen aan de Halconprogramma's waar dit bij de runtime-licentie niet mogelijk is. Beide systemen zijn uitbreidbaar met een USB-doggle. Deze kost €155 en maakt het eenvoudiger om de licentie te gebruiken op meerdere PC's.

3.6 Aflooptijd

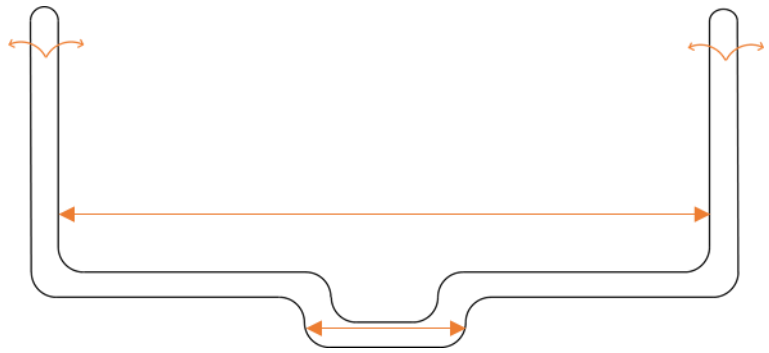
Op vraag van Metes is een schatting gemaakt van de tijd die de installatie erover zou doen om één beugel aan te voeren, het programma te doorlopen en indien nodig uit te werpen.

In Halcon bestaat hiervoor de operator *count_seconds*. Deze operator bepaalt het aantal seconden dat het programma al loopt. Dus door het aantal seconden te bepalen aan het begin en einde van het programma is de aflooptijd gekend. De tijd om een beeld te nemen bedraagt 0,85 s. Dit was echter met een eerder oude versie van de camera, nieuwere modellen zullen sneller beelden nemen. Zoals eerder vermeld, is de doorlooptijd van de Halcon-code 0,2 s. Voor Merlic-code bedraagt de doorlooptijd 0,32 s. Dit is de aflooptijd op een HP EliteBook 850 G3 computer. Deze tijden kunnen verschillen op een andere PC. De sorteerunit heeft een maximale slagsnelheid van 500 $\frac{mm}{s}$ bezitten. Aangezien de slaglengte 200 mm bedraagt, duurt de heen-en-weergaande beweging 0,8 s.

3.7 Haalbaarheidsanalyse

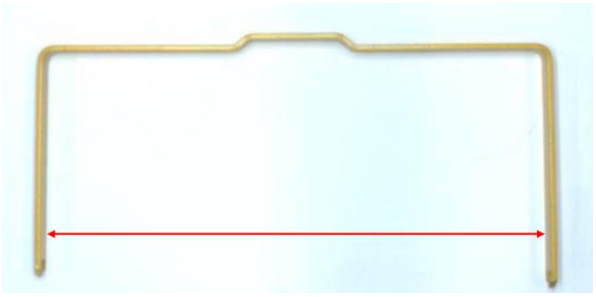
Om te kijken welk programma de meest nauwkeurige resultaten geeft, wordt een vergelijking tussen beiden gemaakt. Door enkele beugels foutief te plooien, is het mogelijk de programma's met elkaar te vergelijken. Door deze beugels, met gekende afwijking, aan beide programma's te presenteren, kunnen de programma's hun controle uitvoeren op de beugels. Van elke foutieve beugel zijn drie foto's genomen waarbij de beugel steeds anders georiënteerd is. Beide programma's verwerken vervolgens elke foto.

Figuur 84 toont op welke manier er voornamelijk fouten optreden. Het is zichtbaar dat de benen naar binnen en naar buiten afwijken, wat respectievelijk overeen komt met een hoek kleiner en groter dan 90°.



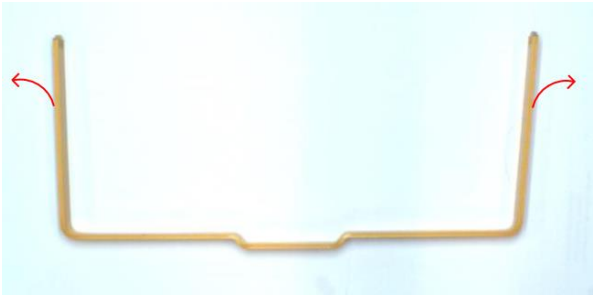
Figuur 84: Plooirichting beugel

Door de beugels handmatig te plooien en deze afwijkingen vervolgens zelf op te meten met een hoekmeter die Metes ter beschikking heeft, ontstaan verschillende testbeugels met gekende afwijkingen. De digitale hoekmeter (Schut) meet tot op 0,08° nauwkeurig. Met behulp van deze testbeugels zijn de verschillende programma's te evalueren. De eerste testbeugel is zichtbaar in figuur 85. De beentjes zijn niet fout geplooid maar de afstand tussen de beentjes is te kort.



Figuur 85: Eerste foutieve beugel

Bij de tweede testbeugel zijn beide beentjes naar buiten geplooid. Één been is met een minimale afwijking geplooid terwijl de andere onder een grotere hoek omgeplooid is. Dit is zichtbaar gemaakt in figuur 86.



Figuur 86: Tweede foutieve beugel

De resultaten van Merlic en Halcon voor elke hoek per foto zijn in tabel 3 terug te vinden

Tabel 3: Gemeten hoeken tweede foutieve beugel

	Hoek links [°]		**Hoek rechts [°]**	
	Merlic	Halcon	Merlic	Halcon
Gemeten	92,081		95,135	
Foto1	92,28	91,76	95,5	95,14
Foto2	92,06	91,41	95,48	95,32
Foto3	91,65	91,39	95,52	95,5

Analoog aan de tweede testbeugel zijn nog drie andere testbeugels gemaakt. Telkens met andere afwijkingen. Zo is de derde testbeugel met beide beentjes naar binnen geplooid. Dit is aangeduid m.b.v. de rode pijlen in figuur 87.

Figuur 87: Derde foutieve beugel

Tabel 4 geeft een overzicht van de gemiddelde afwijkingen van alle testbeugels voor zowel de hoek links als rechts.

Tabel 4: Afwijkingen testbeugels

	Hoek links [°]		**Hoek rechts [°]**	
	Merlic	Halcon	Merlic	Halcon
Afwijking testbeugel 1	0,61	0,12	0,19	0,29
Afwijking testbeugel 2	0,22	0,56	0,50	0,32
Afwijking testbeugel 3	0,47	0,41	0,28	0,35
Afwijking testbeugel 4	0,34	0,24	0,77	0,81
Afwijking testbeugel 5	0,14	0,59	0,44	0,88
Gemiddelde afwijking	0,36	0,38	0,44	0,53

3.8 Conclusie

Uit tabel 4 blijkt dat de gemiddelde afwijkingen van Merlic en Halcon quasi hetzelfde zijn met een afwijking van respectievelijk 0,40° en 0,45°. Deze resultaten, samen met de lengtemetingen van het lipje en de breedtemetingen, zijn te verbeteren door vooraf een kalibratie door te voeren[1]. De kalibratie is nodig om lensafwijkingen teniet te doen om zo de metingen in wereldcoördinaten uit te voeren. Het vergelijken van de resultaten na 2D-kalibratie vormt een interessant toekomstig onderzoek. De totale aflooptijd (nemen afbeelding + verwerken door code) bedraagt voor Merlic 1,17 s en voor Halcon 1,05 s. De mogelijkheden binnen Halcon zijn een stuk uitgebreider dan die van Merlic. Daar tegenover staat dat Halcon buiten minder gebruiksvriendelijk ook een stuk duurder is dan Merlic. Om toch voordeliger uit te komen voor Halcon is er de mogelijkheid om voor de runtime-licentie te kiezen. Deze licentie staat echter wel niet toe aanpassingen aan te brengen in een Halconprogramma, wat binnen Merlic wel mogelijk is. Een licentie om aanpassingen te kunnen doen binnen Halcon kost €6200 en een runtime-licentie bedraagt €691 terwijl die van Merlic €1883 kost. Zoals eerder vermeld, was de oorzaak van de kromme beugels te vinden bij het afbreekproces van de serie beugels. Een andere optie die Metes heeft, is het investeren in een toestel of tool dat het afbreken van de beugels op een foutloze manier uitvoert. Zo is er minder afkeur en zal Metes op lange termijn profiteren van dit voordeel.

[1] T.g.v. de maatregelen omtrent COVID-19 was het opnieuw uitvoeren van de metingen mét een 2D-kalibratie onmogelijk.

4 Opdracht 2: plooirobot

4.1 Methode

Uit 2.5 blijkt dat er drie methodes geschikt zijn om een 3D-puntenwolk te verkrijgen, namelijk stereovisie, sheet-of-light (SOL) en structured-light. Omdat er voor een structured-lightopstelling geen beschikbare hardware aanwezig was te ACRO en de prijs voor zo'n systeem vrij hoog is, is er besloten om deze methode achterwege te laten en verder de focus te leggen op stereovisie en SOL. Voor beiden volgt een korte theoretische uitleg gevolgd door de toelichting hoe ze werken en hoe de proefopstelling in elkaar zit. Op basis van de testresultaten volgt de keuze welke methode het best toe te passen is voor dit project. De eerste methode en proefopstelling is voor een camera met stereovisie te testen. Op basis van de beelden die uit deze testen komen, volgt een analyse en ten slotte een conclusie. De tweede methode en proefopstelling is van een meer complexe aard, namelijk een sheet-of-light opstelling. Analoog aan stereovisie volgt na de testen een analyse en conclusie.

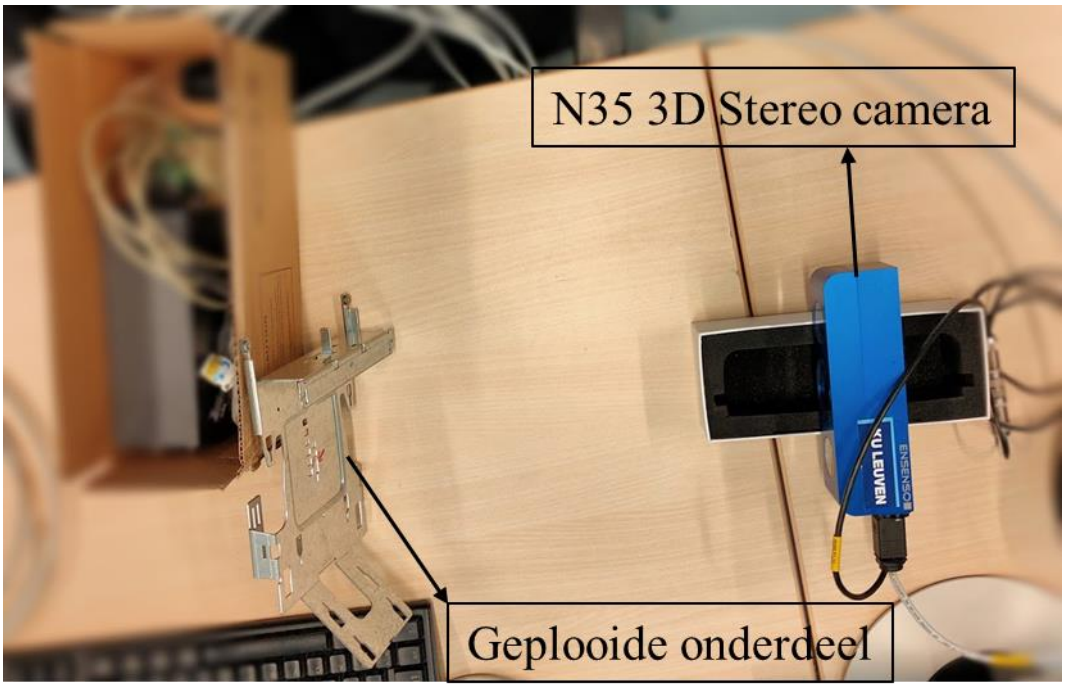
4.2 Projected texture stereovisie (PTS)

4.2.1 Theorie

Projected texture stereovisie berust op het principe waarbij d.m.v. het zoeken naar overeenkomstige punten in de beelden van meerdere camera's een 3D-puntenwolk op te stellen is, waarbij een projector in de sensor een infraroodpatroon uitzendt op het object om deze overeenkomstige punten eenvoudiger te vinden. Een meer diepgaande uitleg omtrent stereovisie is terug te vinden in de literatuurstudie onder 2.4.1.1.

4.2.2 Opstelling

De beschikbare camera van ACRO is een N35 3D stereo camera van Ensenso. De opstelling is getoond in figuur 88.

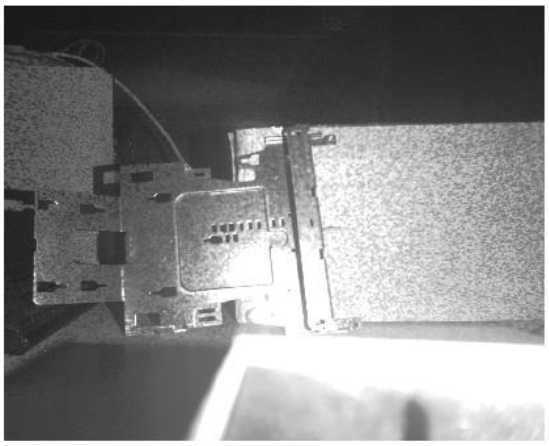


Figuur 88: Testopstelling PTS

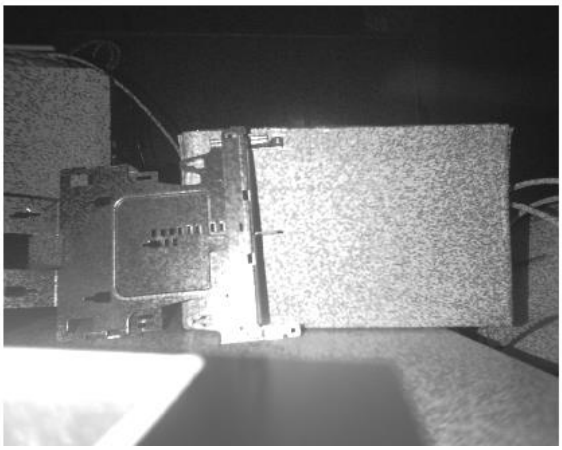
In deze testopstelling zijn twee onderdelen zichtbaar. Rechts bevindt zich de stereovisie camera in het blauw en links het geplooide onderdeel. Om het bovenaanzicht in beeld te brengen, ondersteunt een doos het onderdeel. Verder dient deze doos als referentie voor de camera. Deze referentie is nodig om de cameraparameters af te stellen aangezien het rechte oppervlak van de doos eenvoudig te herkennen is.

4.2.3 Gegenereerde beelden

De Ensenso N35 camera projecteert een infraroodpatroon (850nm) [63]. Dit patroon valt in op het object en twee camera's leggen dit patroon vast. Voor het menselijk oog valt hier geen patroon van op te maken maar voor de camera vormt dit geen probleem aangezien die gevoelig is voor het infraroodspectrum. In figuur 89 en figuur 90 zijn de 2D-afbeeldingen zichtbaar met het infraroodpatroon op.

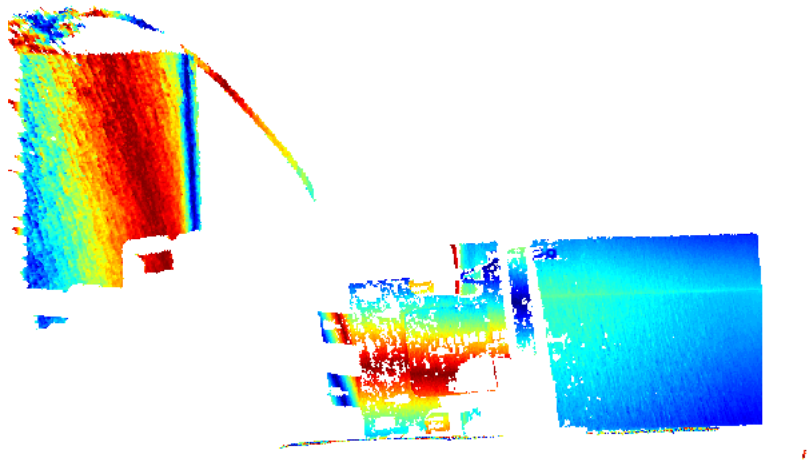


Figuur 89: Afbeelding camera links Ensenso N35



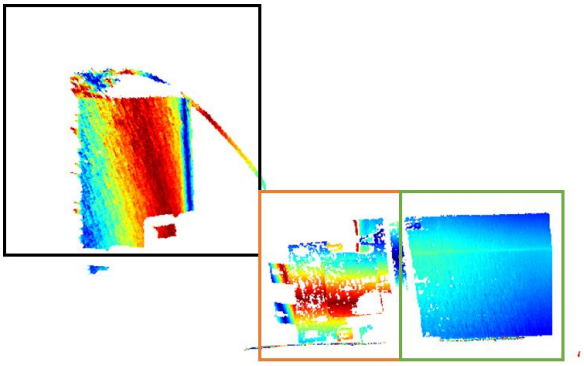
Figuur 90: Afbeelding camera rechts Ensenso N35

Verder is in deze twee afbeeldingen ook zichtbaar dat er een grote hoeveelheid reflectie optreedt op het onderdeel zelf. De reflectie is duidelijk zichtbaar aan de heldere vlekken op beide figuren en is een typisch gevolg van overbelichting. De overbelichting is afkomstig van een interne LED die via de camerasoftware niet uit te schakelen is. De camera software zet deze twee afbeeldingen om in een *depth image.* De bijhorende depth image is zichtbaar in figuur 91.

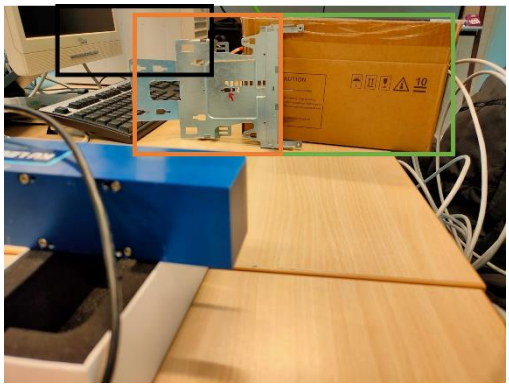


Figuur 91: Depth image

Ter verduidelijking zijn in figuur 92 en figuur 93 de verschillende onderdelen van de opstelling aangeduid. Hier stelt het groene kader de kartonnen doos van de achtergrond voor, het zwarte kader is een deel van de computer die zich op de achtergrond bevindt. Het oranje kader stelt het plooistuk voor.

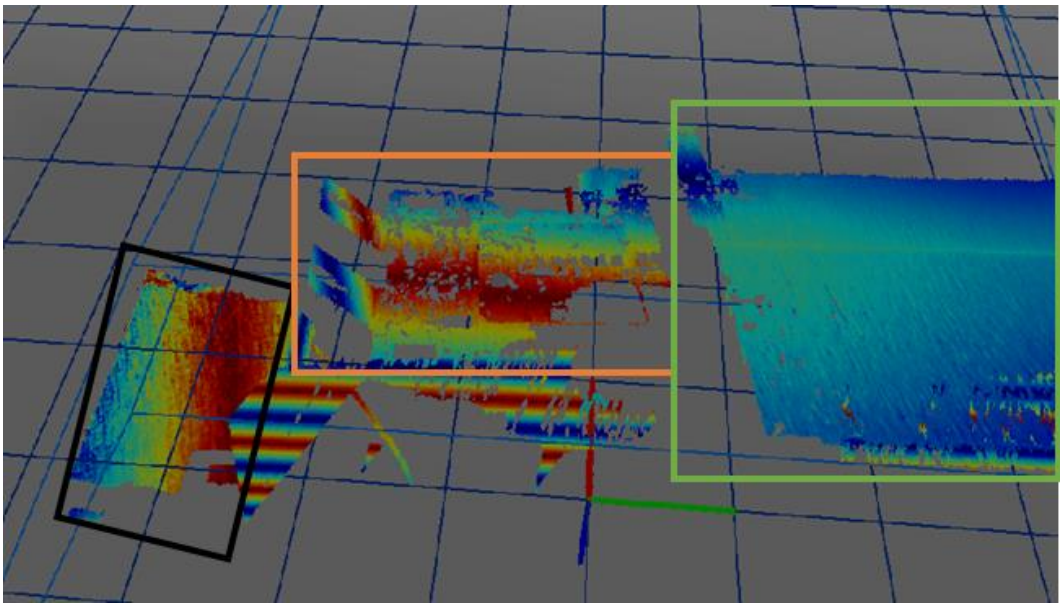


Figuur 92: Depth image met aanduiding



Figuur 93: Opstelling met aanduiding onderdelen

De kleuren van de depth image komen overeen met de afstand van de objecten (punten) tot de camera. Op basis van deze kleuren stelt NXView (standaard software van de camera) de puntenwolk op. Deze is te zien in figuur 94.



Figuur 94: 3D-puntenwolk met PTS

4.2.4 Conclusie stereovisie

De conclusie van deze test is dat de gegenereerde 3D-puntenwolk, en bijgevolg deze methode, niet geschikt is om de gewenste controles uit te voeren. Er zijn te weinig details zichtbaar van het geplooide onderdeel waardoor het onmogelijk zal zijn om elke vereiste meting uit te voeren. De puntenwolk uit figuur 94 is te geperforeerd om correcte metingen op uit te voeren. In figuur 89 en figuur 90 is zichtbaar dat er veel reflectie optreedt. Deze reflectie is te wijten aan het feit dat er een externe lichtbron aanwezig is in de camera die niet uit te schakelen is via de ENSENSO-software. Binnen Halcon is het echter wel mogelijk deze *front lighting* uit te schakelen. Buiten het feit dat de puntenwolk heel geperforeerd is, kan er worden opgemaakt dat de nauwkeurigheid sowieso onvoldoende was om meting uit te voeren die voldoen aan de vereiste toleranties.

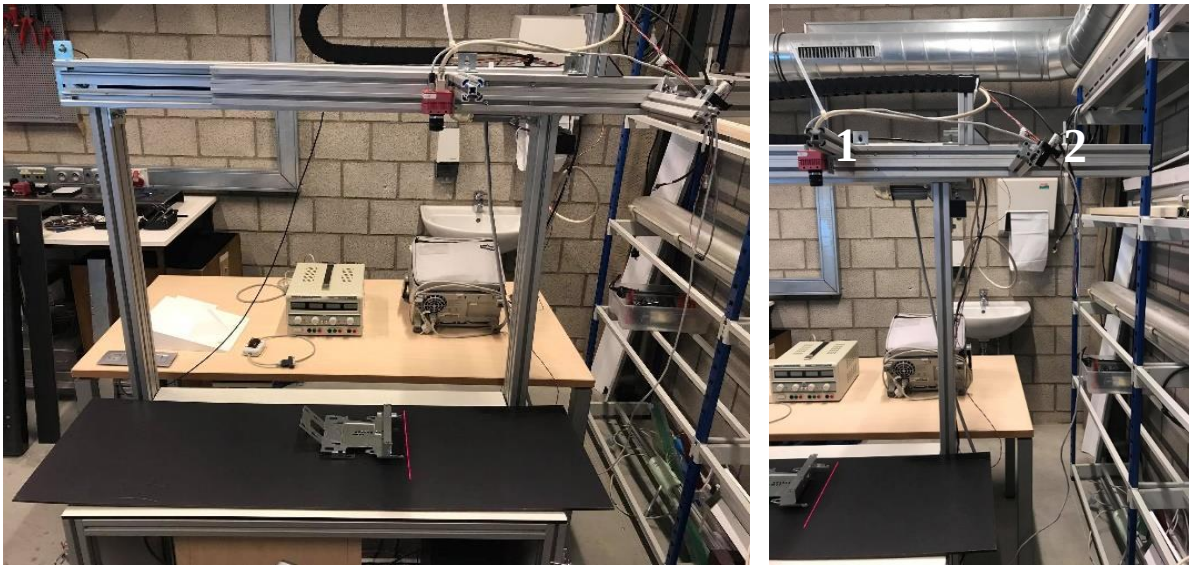
4.3 Sheet-of-light scanning

4.3.1 Theorie

ACRO heeft een sheet-of-light (SOL) opstelling ter beschikking. Deze berust op het principe van triangulatie. Hierbij staan de camera en laser onder een bepaalde hoek t.o.v. elkaar zijnde de triangulatiehoek. Door de laserlijn over het object te bewegen, kan de camera het gereflecteerde laserlicht opvangen en een beeld (puntenwolk) vormen. Hier is tevens een uitgebreidere uitleg van terug te vinden in de literatuurstudie onder 2.4.2.3.

4.3.2 Opstelling

De opstelling is te zien in figuur 95. De voornaamste componenten van deze opstelling zijn de camera (1) en laser (2). De camera staat loodrecht op het vlak, terwijl de laser onder een bepaalde hoek staat. Deze twee zijn samen gemonteerd op een profiel dat over de lengte van het scanoppervlak kan bewegen m.b.v. een elektrische motor. Normaal bezit de opstelling een wit oppervlak maar aangezien een donker oppervlak voor betere resultaten zorgt, is een zwarte plaat als ondergrond gebruikt.

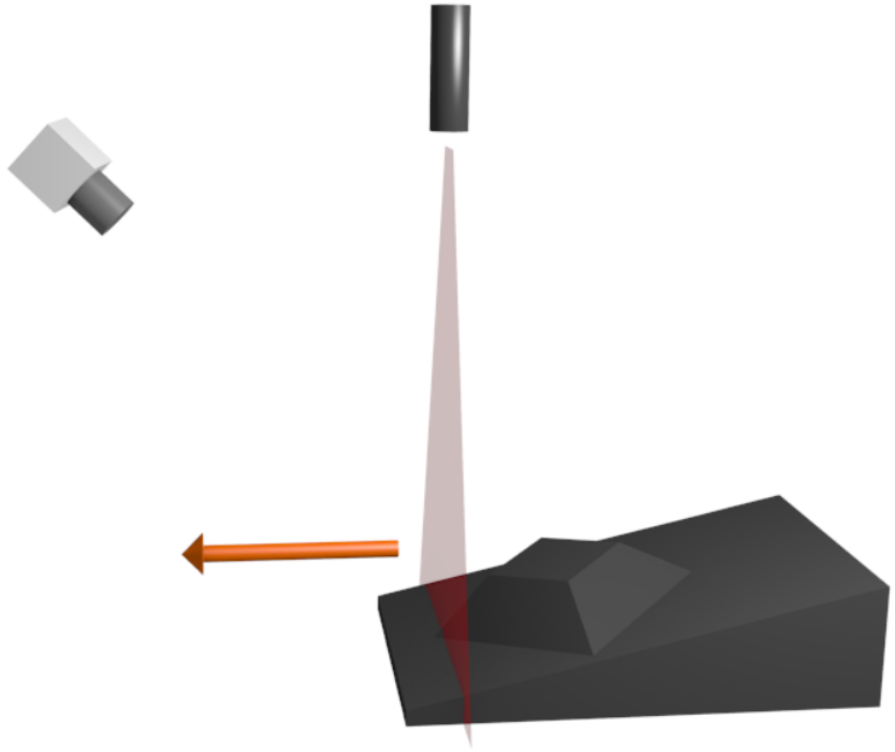


(a) Geheel *(b) Focus camera en laser*

Figuur 95: Sheet-of-lightopstelling ACRO

4.3.3 Kalibratie van sheet-of-lightopstelling

Halcon heeft ingebouwde functies om een ruim gamma aan visietoepassingen te kalibreren. Zo is er een functie om een sheet-of-lightopstelling te kalibreren, namelijk *calibrate_sheet_of_light*. Deze kalibratie gebeurt op basis van een *disparity image* van een 3D-object [64]. De kalibratie verloopt door het kalibratie-object te scannen zoals in figuur 96.



Figuur 96: Kalibratiescan sheet of light [64]

Dit object heet in Halcon *"calibration plate"(CP)*. De CP bezit een schuine helling met daarop een afgeplatte piramide. Verder bezit de CP een dunne en dikke kant respectievelijk de voor- en achterkant. Ten slotte is er een gat aanwezig met een diameter van 0,075 keer de breedte van de CP [65].

Alvorens het uitvoeren van deze kalibratie zijn er een aantal voorafgaande stappen. Deze stappen zijn hieronder opgesomd:

- SOL model creëren.
- Vastleggen van initiële camera parameters.
- CP aanmaken.

Uit de kalibratie komen een aantal gegevens voort, namelijk camerapose, lichtvlakpose en bewegingspose.

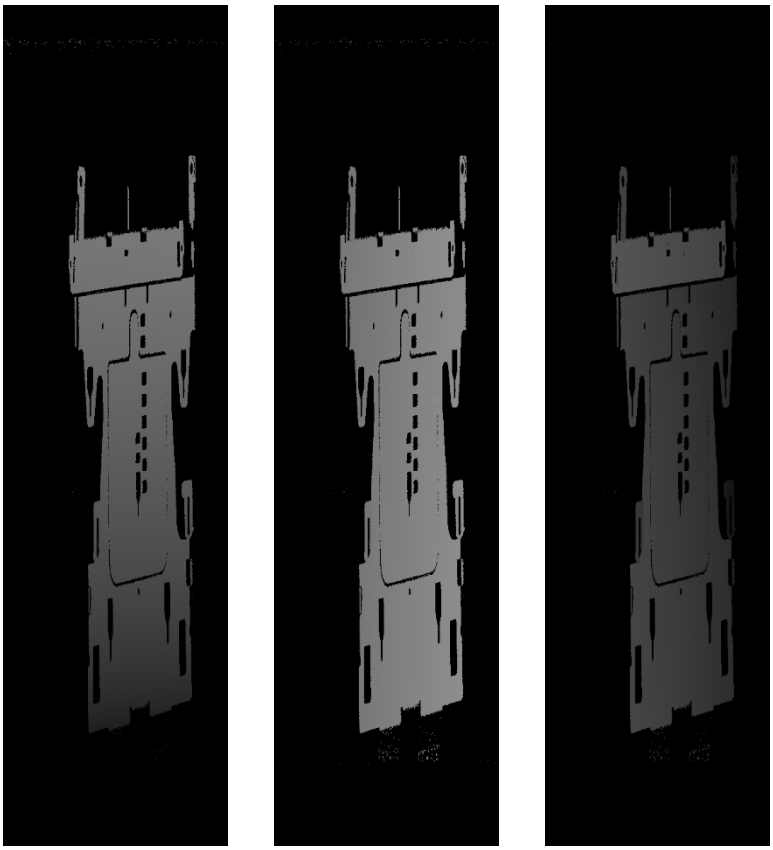
4.3.4 Gegenereerde beelden

Als de SOL-opstelling over het object beweegt, maakt deze verschillende beelden. Op zo'n beeld is de geprojecteerde laserlijn op het object zichtbaar.

Figuur 97: SOL-beeld

Bovenstaande afbeelding vormt samen met de andere genomen beelden een sheet-of-lightmodel. Uit dit model komen X-, Y-, Z-beelden voort, zie Figuur 98.

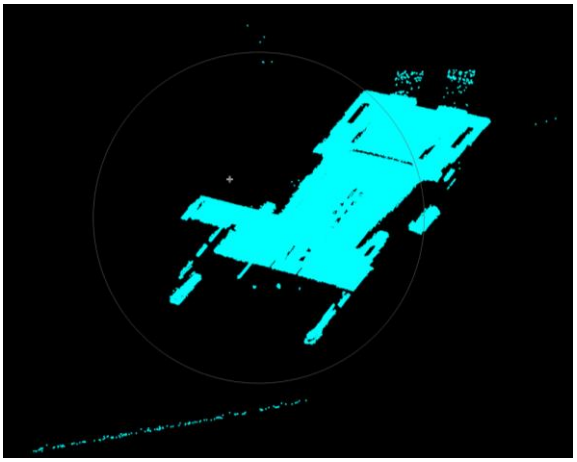
De X-beelden stellen de gekalibreerde X-coördinaten van het gereconstrueerde oppervlak voor. Dit geldt eveneens voor de Y- en Z-beelden. Met het blote oog valt er niet veel op te maken uit deze beelden, maar Halcon kan deze afbeeldingen gebruiken om een 3D-model mee op te stellen [16].



(a) X-beeld *(b) Y-beeld* *(c) Z-beeld*

Figuur 98: X,Y,Z-beelden

Het samenvoegen van deze drie afbeeldingen met *xyz_to_object_model_3d* creëert een puntenwolk. Deze is te zien in figuur 99.



Figuur 99: Puntenwolk SOL

4.3.5 Conclusie sheet-of-light scanning

De SOL-opstelling genereert bruikbare puntenwolken om metingen op uit te voeren. Echter bezitten deze enige ruis en vallen er delen van de objecten weg, in de vorm van 3D-data gaten. Hierdoor zal er verder onderzoek nodig zijn naar de optimale positie van het object t.o.v. de SOL-opstelling, alsook naar de optimale instelling van de parameters van de opstelling.

4.4 Software

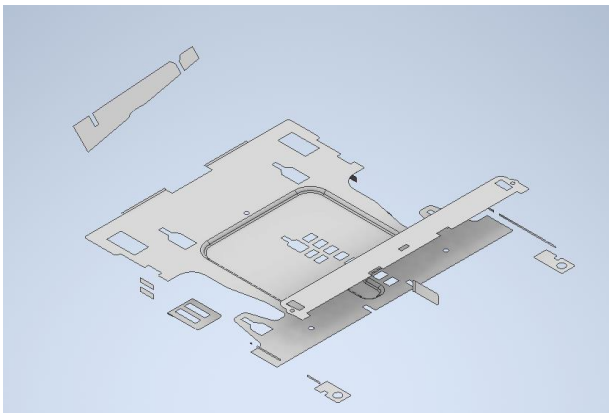
4.4.1 Halcon

4.4.1.1 Matchingsprincipes

a) Uitgevoerde controles

Halcon bevat twee matchingprincipes, namelijk register- en surface-based matching, respectievelijk afgekort RB-matching en SB-matching. De literatuurstudie licht beiden toe in 2.4.3.2. Oorspronkelijk vindt RB-matching zijn toepassing in het zoeken naar de overlap tussen 3D-puntenwolken met als doel deze puntenwolken samen te hangen. Het onderliggende algoritme is echter zeer gelijkend op dat van SB-matching en kan een potentieel toegevoegde waarde betekenen voor de prestaties van het controleprogramma [66].

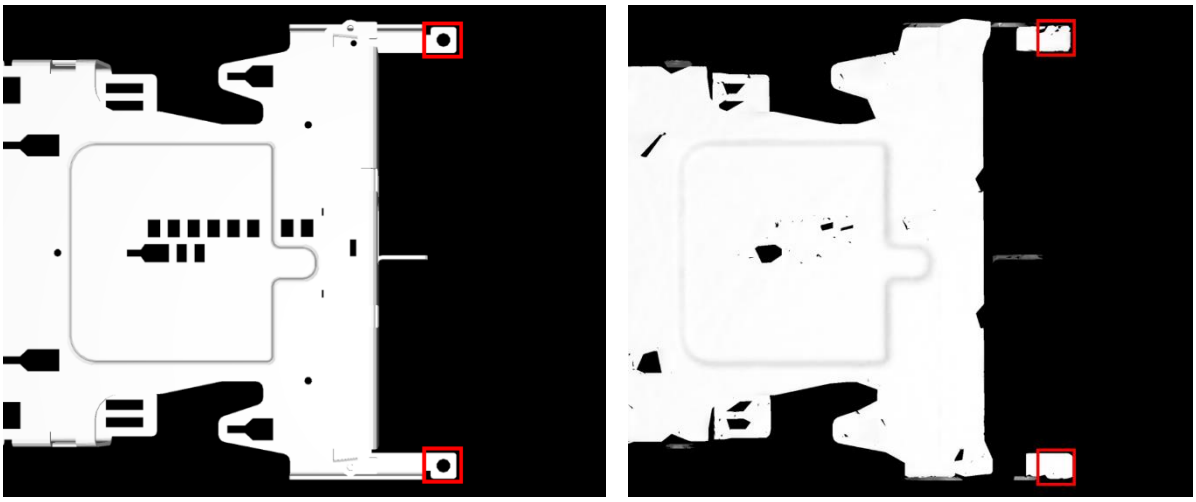
De vergelijking tussen beide principes gebeurt a.d.h.v. CAD-files en dus niet a.d.h.v. een gegenereerde 3D-puntenwolk. Eén CAD-file dient als referentie en blijft onaangeroerd. De andere CAD-file ondergaat enkele aanpassingen die het model meer op de scans doet lijken. Bij het nemen van de scans blijken er namelijk verschillende zones te zijn waarvan maar enkele of zelfs geen punten gescand zijn. Door de CAD-files van deze 'missende datazones' te voorzien, lijken ze meer op de effectieve scans. Figuur 100 toont dit aangepast CAD-model.



Figuur 100: Aangepast CAD-model

De typische output van een matching-instructie is een score. Afhankelijk van het gebruikte matchingprincipe zal de betekenis van die score verschillend zijn. Diepgaandere uitleg is tevens terug te vinden in de literatuurstudie (2.4.3.2). Omdat de score alleen niet volstaat, is een extra controle noodzakelijk. Aangezien de twee CAD-modellen na het matchen dezelfde pose bezitten, laat Halcon toe om een verdere controle uit te voeren, namelijk de controle van de centerpunten aan de uiteindes van de beentjes.

In figuur 101 (a) en (b) gaat Halcon binnen de rode vierkantjes op zoek naar de centerpunten. Is het object niet 100% correct gematcht, dan vertaalt dat zich in een foutieve centerpuntafstand.



(a) referentie CAD *(b) aangepaste CAD*

Figuur 101: Controle centerpunten beentjes

b) Opbouw software

De matchingprincipes zijn zowel apart als gecombineerd getest. De matchingprocedures hebben buiten een input ook enkele parameters. Nog voor de definitieve keuze voor een bepaald matchingprincipe is eerst de optimale set van parameters nodig. Er zijn verschillende parameters die het eindresultaat beïnvloeden maar de twee belangrijkste parameters zijn *RelSamplingDistance (RSD)* en *KeyPointFraction (KPF)*. Het vinden van de optimale set van parameters volgt na de integratie van geneste for-lussen [67]. Een deel van deze for-lussen is te zien in figuur 102. Onder de operator *find_surface_model* komt de meting van de centerpunten om te controleren welke parameters de kleinste centerafstand geven.

```
for SD := 0.01 to 1 by 0.01
    for KPF := 0.01 to 1 by 0.01
        create_surface_model (ObjectModel3DNormals, SD, [], [], SurfaceModelID)
        find_surface_model (SurfaceModelID, CADModel, SD, KPF, 0.1, 'true', [], [], Pose, Score, SurfaceMatchingResultID)
```

Figuur 102: Geneste for-lussen

Uit deze testen blijkt dat 0,03 en 0,25 de optimale parameterset is voor respectievelijk RSD en KPF.

Na het bepalen van de optimale parameterset volgt nu het bepalen van het optimale matchingprincipe. In figuur 103 (a) en (b) zijn de meetresultaten te zien van respectievelijk RB- en SB-matching.

(a): Register *(b): Surface-based*

Figuur 103: Metingen losse matchingsprincipes

Opmerking bij deze resultaten is dat indien de verplaatsing tussen de gesimuleerde scan en referentiemodel te groot is, SB-matching zal falen in het matchen. Hier heeft RB-matching veel minder problemen mee. Verdere testen waarbij beide matchingprincipes aan te pas kwamen, bleken ook zinvolle resultaten terug te geven. In figuur 104 (a) en (b) is respectievelijk RB-matching gevolgd door SB-matching en SB-matching gevolgd door RB-matching weergegeven. Ten eerste valt op dat de toepassing van gecombineerde matchingprincipes betere resultaten geeft dan bij één enkel matchingprincipe.

Verder is er uit figuur 104 af te leiden dat de meest nauwkeurige matchingsmethode het toepassen van RB-matching is, gevolgd door SB-matching. Een kritische afweging hier is of de kleine stijging in nauwkeurigheid de extra bewerkingstijd waard is.

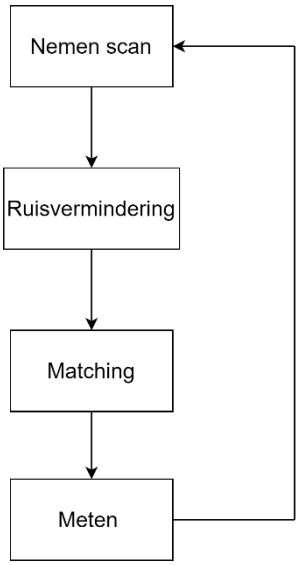
(a) *(b)*

Figuur 104: Gecombineerde matchingprincipes

4.4.1.2 Code

Er zijn twee manieren om het probleem van de 3D-metingen aan te pakken. Een eerste manier is om elke eigenschap van het object apart te bekijken. Hiervoor zijn op zich weer verschillende mogelijkheden. Om metingen uit te kunnen voeren, al dan niet t.o.v. de CAD-files, is het nodig om doorsnedes en aanzichten om te zetten in contouren. Een tweede manier om het gescande object te beoordelen, is door gebruik te maken van een afstandsmeting tussen een referentie en de gemaakte scan.

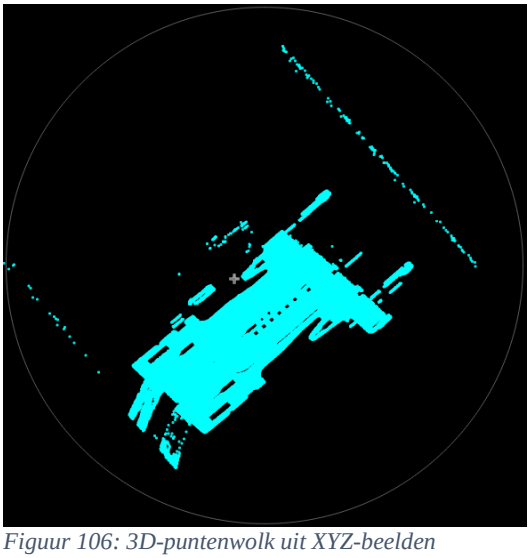
In figuur 105 is het functieschema te zien. Dit is van toepassing op beide methodes. De eerste drie stappen zijn identiek. Het grote verschil zit in de laatste stap: het meten.



Figuur 105: Functieschema code project 2

Het functieschema beschrijft de volgende stappen in het verwerkingsproces van de scans, met name het nemen van een scan, ruisvermindering toepassen op de scan, het matchen met het CAD-model en ten slotte het uitvoeren van de metingen.

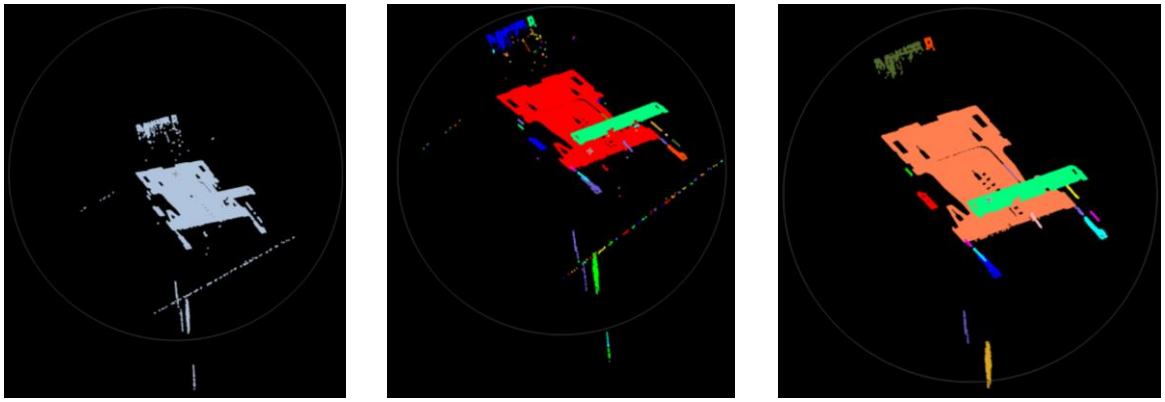
Voor het nemen van een scan komt eerst nog het initialiseren van de scanner en enkele parameters. Tijdens het testen voor dit project gebeurde het nemen van de scans op voorhand en las het programma de XYZ-beelden in. Deze XYZ-beelden kan Halcon simpelweg omzetten naar een 3D-puntenwolk, die te zien is in figuur 106.



Figuur 106: 3D-puntenwolk uit XYZ-beelden

Vervolgens leest Halcon de referentie CAD-file in, in STL-formaat. Een belangrijke opmerking hier is dat alvorens het CAD-model in te lezen een schaling nodig is omdat de lengte-, hoogte-, en breedte-verhoudingen niet meer correct zijn na het nemen van een scan. Dit zorgt voor problemen bij het matchen waar deze verhoudingen tussen CAD en scan essentieel zijn om een goed resultaat te bekomen. Het schalen gebeurt met software van Cura, een sliceprogramma gebruikt bij 3D-printers.

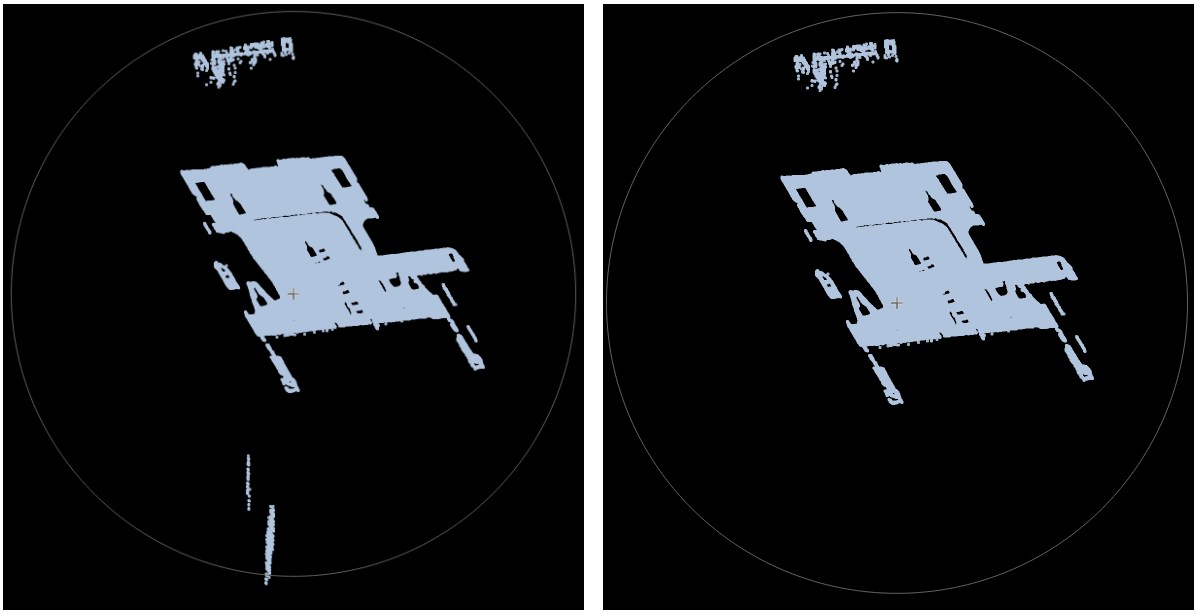
Voor het matchen van de scans moet de ruis verdwijnen. Zonder deze ruisvermindering gaan de scan en CAD-model incorrect matchen. De ruisvermindering gebeurt in twee stappen. Er is 'kleine' en 'grote' ruis respectievelijk ruis met lage en hoge densiteit. De eerste stap is alle kleine ruis verwijderen en dan pas de grote ruis. De verwijdering van de grote ruis gebeurt tijdens het matchen. De ruisvermindering voor kleine ruis gebeurt met *connection_object_model_3d* nog voor het matchen. Deze operator deelt de scan op in verschillende delen. Door de *Feature*-parameter op '*distance_3d'* te zetten, verdeelt de operator de scan o.b.v. de afstand tussen de punten. Door in de *Value*-parameter een waarde mee te geven, beschouwt Halcon alle punten met onderlinge afstand kleiner dan die meegegeven waarde als geconnecteerd. Het resultaat hiervan is te zien in figuur 107 (b). Hierna gebeurt een selectie van welke punten tot de scan behoren en welke punten ruis zijn. *Select_object_model_3d* selecteert de punten o.b.v. het aantal punten dat het bevat. De delen met een voldoende puntendensiteit die overblijven zijn zichtbaar in figuur 107 (c) De belangrijkste ruis verdwijnt tussen de 50 en oneindig aantal punten. Ten slotte vormen alle losse elementen met *union_object_model_3d* terug één geheel.



(a): Initiële puntenwolk *(b): Connection_object_model_3d* *(c): Select_object_model_3d*

Figuur 107: Ruisvermindering van kleine ruis

De volgende stap in de software is de uitvoering van de matching tussen de scan en de CAD-file. Zoals uit de literatuurstudie en resultaten van 4.4.1.1 blijkt, zijn er twee verschillende methodes. Uit 4.4.1.1 blijkt tevens dat het beste resultaat volgt uit een combinatie van RB-matching en SB-matching. Tussen deze twee matchingmethodes in verdwijnt de grote ruis. Dit gebeurt direct na het uitvoeren van RB-matching. Deze geeft namelijk de pose terug van het object t.o.v. de CAD-file en o.b.v. deze pose verwijderen een combinatie van *distance_object_model_3d* en *select_points_object_model_3d* de grote ruis. figuur 108 toont deze ruisvermindering.

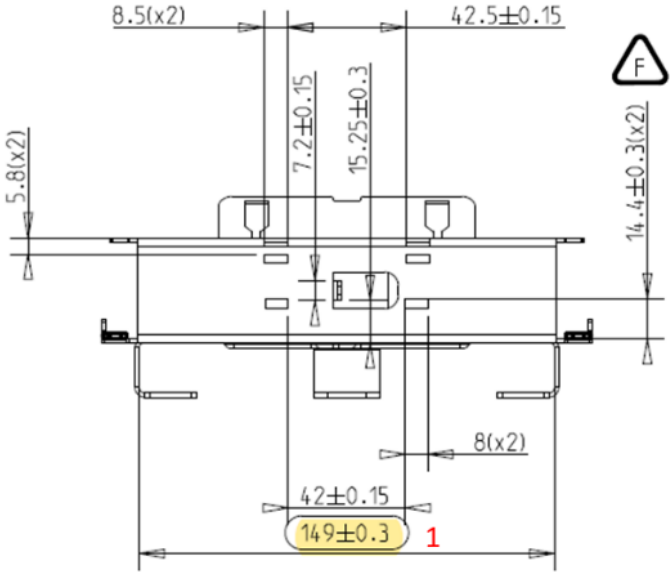


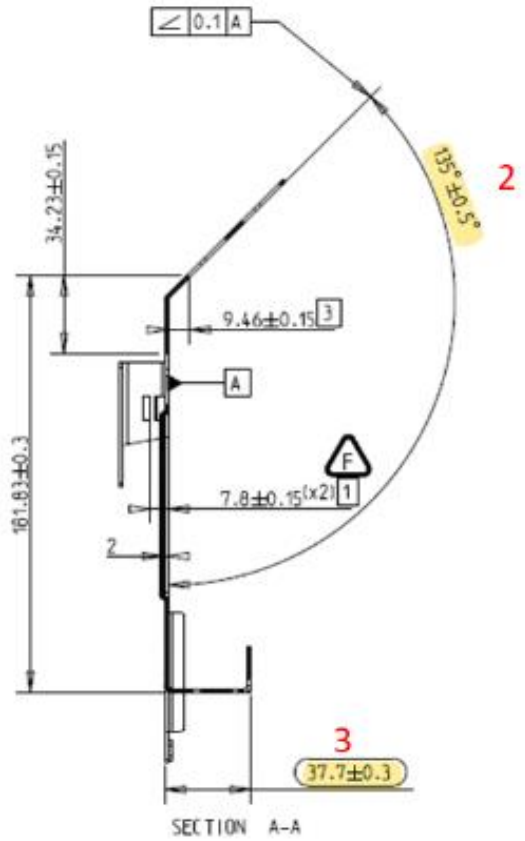
(a): Resultaat na verwijdering kleine ruis *(b): Resultaat na verwijdering grote ruis*

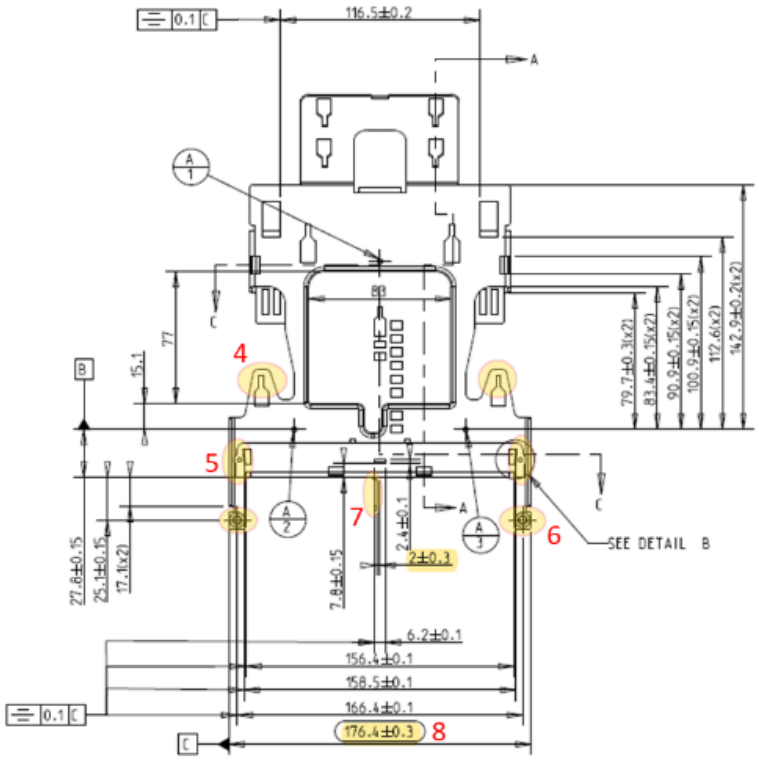
Figuur 108: Ruisvermindering van grote ruis

a) Metingen o.b.v. doorsnedes en aanzichten

Een deel van de metingen zijn gedaan o.b.v. het CAD-model. Voor elke meting is een aparte procedure gemaakt. Binnen Halcon vormen deze procedures functies die uit te voeren zijn door ze in de code op te roepen. In Figuur 109 zijn alle te controleren maten aangeduid en genummerd.







Figuur 109: Nummering na te meten eigenschappen

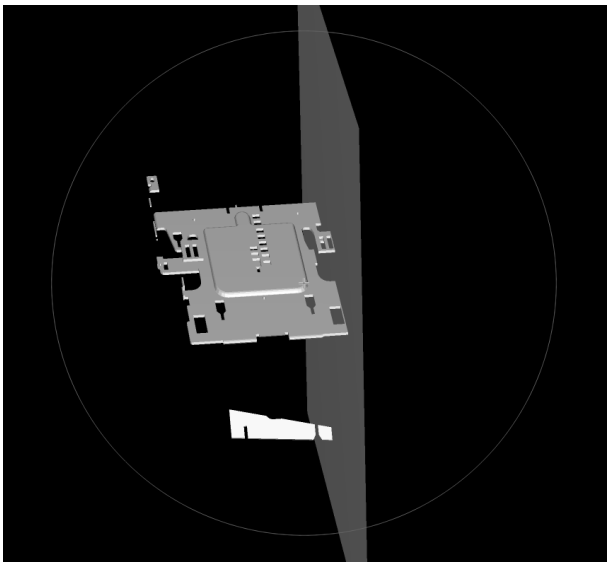
Hieronder staan al de aanwezige procedures gelinkt aan welke eigenschap van het object ze opmeten:

1. controle_steunvoetjes
2. controle_kophoek
3. controle_hoogte_37_7
4. controle_flesjes
5. controle_uitstekenddeel_uiteindes
6. controle_persmoeren
7. controle_staartje
8. controle_persmoeren

De volledige procedures zijn terug te vinden in de Halcon-code. Hieronder volgt een korte uitleg van enkele belangrijke operatoren gevolgd door een voorbeeld.

Om te werken met doorsnedes moet er eerst een vlak aanwezig zijn. Dit vlak moet een bepaalde pose en grootte bezitten. Door een pose vast te leggen met *create_pose* en vervolgens de operator *gen_plane_object_model_3d* uit te voeren, legt Halcon het vlak vast als een 3D-object. Het maken van een doorsnede met het aanwezige vlak gebeurt via *intersect_plane_object_model_3d*. De output van deze operator is een 3D-puntenwolk. Het omzetten van deze puntenwolk naar een 2D-omgeving gebeurt met *project_object_model_3d* wat de contour van de puntenwolk teruggeeft. Deze contour ligt aan de basis van de 2D-metingen.

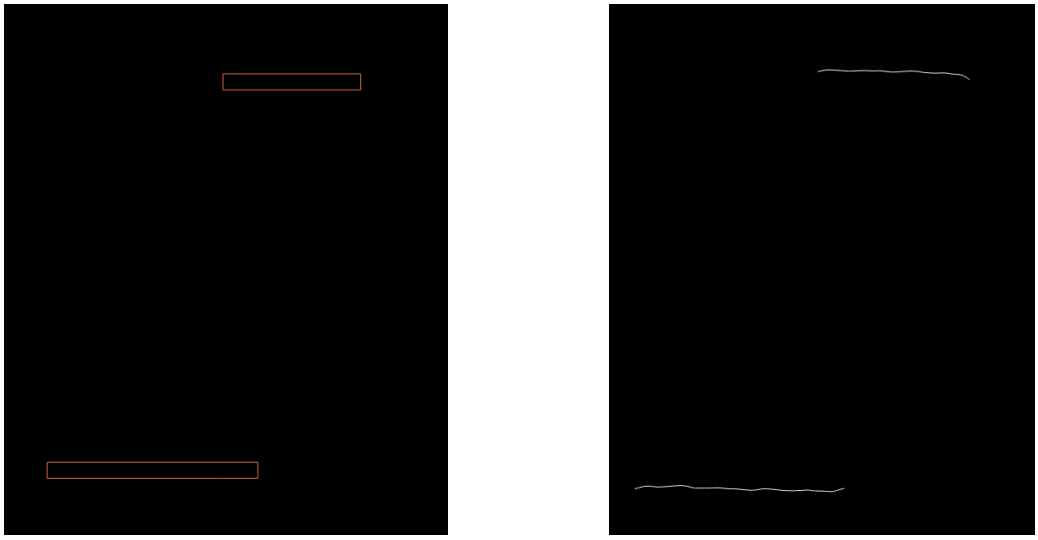
Zoals eerder beschreven, gebeuren de metingen o.b.v. van doorsnedes en aanzichten. Hieronder volgt een visueel voorbeeld over de stappen in de procedure ‘controle_hoogte_37_7’. Figuur 110 geeft het CAD-model weer en een snijvlak. Op basis van dit snijvlak ontstaat een doorsnede van het object, getoond in figuur 111. Door hier een ROI oftewel *region of intrest* te definiëren, is het mogelijk nuttige informatie te extraheren uit het beeld. De ROI is aangeduid met een rood kader in figuur 111. Ditzelfde gebeurt ook met scan.



Figuur 110: CAD-model met sectievlak

Figuur 111: Doorsnede CAD-model

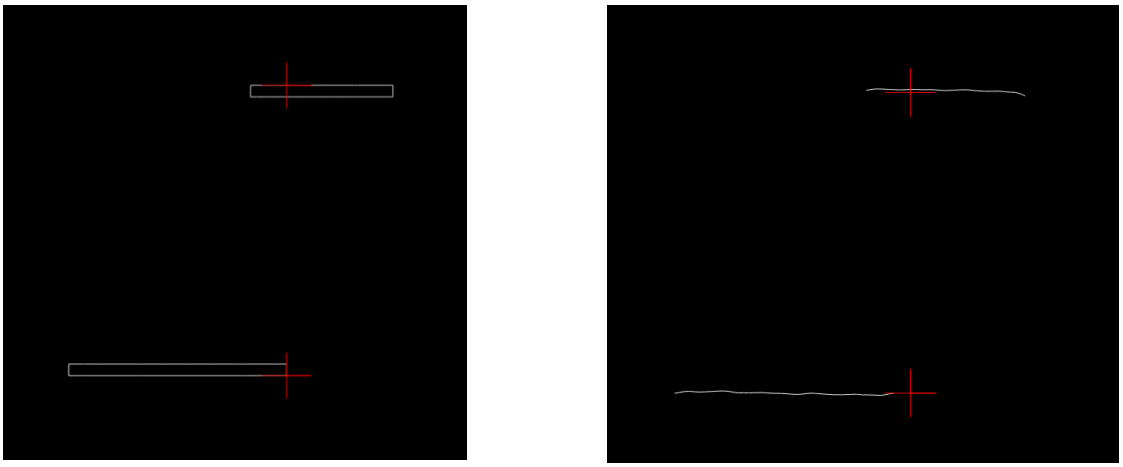
Het gedeelte van de doorsnede dat binnen de ROI valt is zichtbaar in Figuur 112 (a). Figuur 112 (b) toont het resultaat van een analoge aanpak maar dan voor de werkelijke 3D-scan. Op deze beelden volgt nu een 2D-meting van de hoogte.



(a): CAD *(b): Scan*

Figuur 112: Doorsnedes binnen ROI

Bij het CAD-model gebeurt de hoogtebepaling o.b.v. de uiterste punten van de verkregen contouren. Deze punten zijn in figuur 113 (a) aangeduid met een rood kruis. Met een analoge aanpak kan de onderlinge afstand bij de echte scan bepaald worden. Deze blijkt 37,5 te zijn. De waardes moeten binnen vooraf opgelegde grenzen liggen om niet buiten de toleranties te vallen. Deze grenzen zijn 37,4 en 38,0.



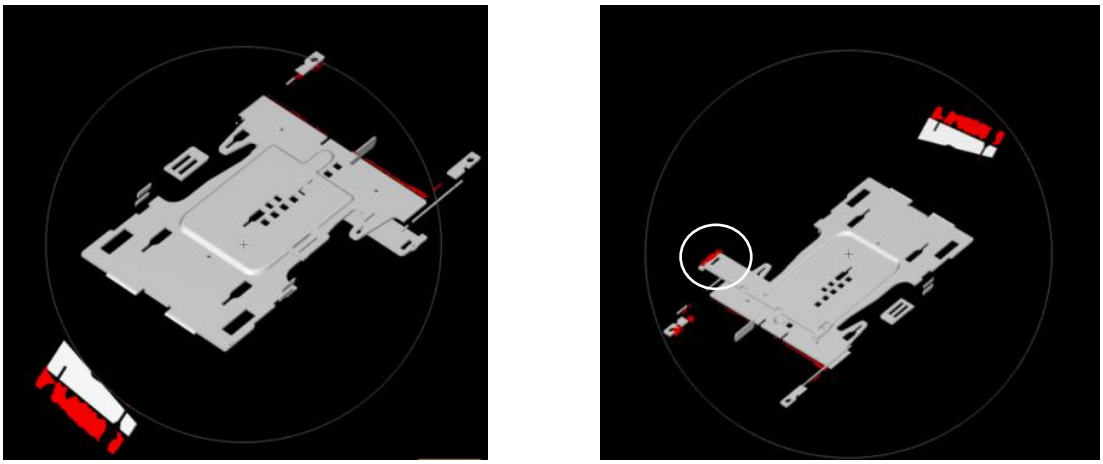
(a): CAD *(b) Scan*

Figuur 113: Aanduiding centers op ROI

b) Metingen o.b.v. afstand tot referentiepuntenwolk

Bij de tweede methode meet Halcon o.b.v. de afstand tot een referentiepuntenwolk. Door bij de *Attrib*-parameter van *select_points_object_model_3d* '*&distance*' in te vullen werkt deze operator samen met *distance_object_model_3d* om de afstand tussen een scan en een referentie te bepalen. Twee opties als mogelijke referentie zijn geschikt, namelijk een (aangepast) CAD-model en een combinatie van verschillende scans van een correct object.

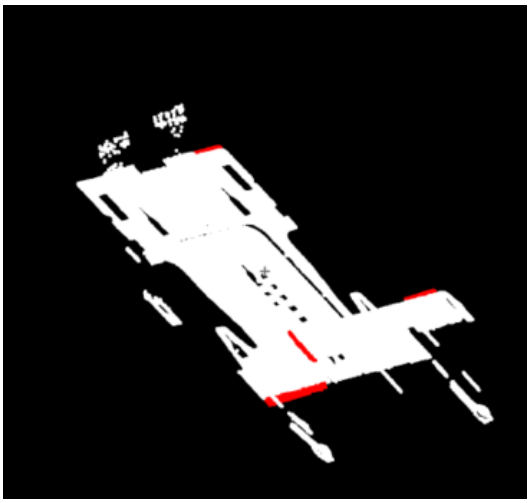
Het resultaat van de metingen t.o.v. het CAD-model is te zien in figuur 114. De fouten in het model zijn in het rood aangeduid. De enige fout die aanwezig is, is het omgeplooide lipje aan de zijkant (aangeduid met witte cirkel). De andere aangeduide zones (in het rood) zijn 'valse fouten'. Deze vorm van fouten detecteren t.o.v. een CAD-model is dus niet optimaal.



(a) *(b)*

Figuur 114: Fouten van scan t.o.v. CAD

Figuur 115 toont het resultaat van het vervangen van het CAD-model met een echte scan. Er zijn nog steeds valse fouten aanwezig maar door meerdere scans van een correct object te combineren, ontstaat een juiste referentie[2]. Binnen Halcon is er een operator aanwezig die deze verschillende scans kan combineren tot één puntenwolk, namelijk *union_object_model_3d*. Het is hierbij van belang de verkregen puntenwolken eerst te matchen met elkaar, alvorens ze te combineren [68], [69].



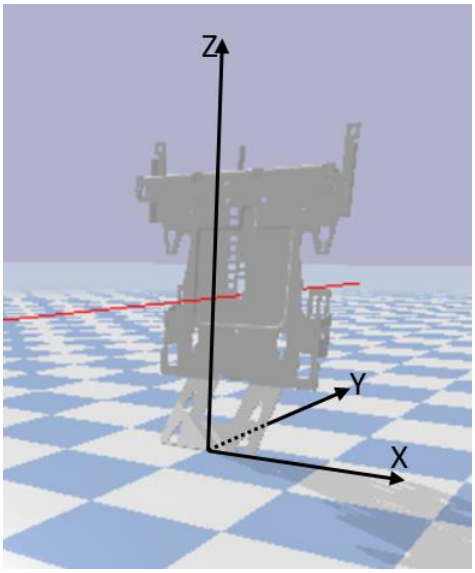
Figuur 115: Fouten van scan t.o.v. referentiescan

[2] T.g.v. de maatregelen omtrent COVID-19 was het nemen van bijkomende scans onmogelijk en zal de referentiescan maar uit één enkele scan bestaan.

4.4.2 Optimale scanhoeksimulatie in Python

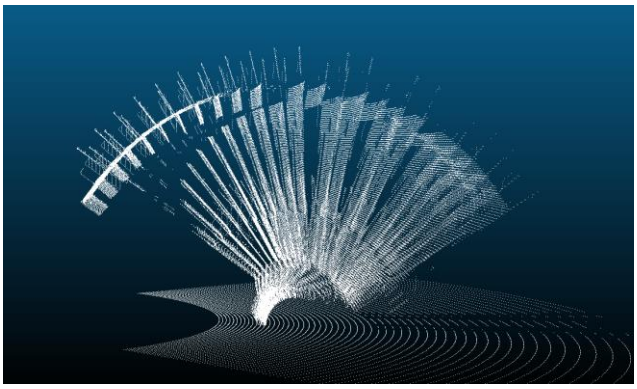
De optimale oriëntatie van het object t.o.v. de scanner opdat de scans van optimale kwaliteit zijn, is bepaald door de scans te simuleren in Python. Dit programma, geschreven door ing. Peter Aerts, voert eigenlijk een puntscan uit en is dus niet 100% compatibel met de situatie van dit project. Echter zal de uitkomst van deze scan een goede indicatie zijn voor de werkelijke optimale oriëntatie. Wederom is er gekozen om te werken met het CAD-model. Eerst leest het programma de STL-file van het onderdeel in. Vervolgens toont het programma verschillende beelden van het onderdeel afhankelijk van het standpunt van de camera t.o.v. het onderdeel.

Om de optimale pose van het object t.o.v. de camera te weten, is als volgt tewerk gegaan: de camera krijgt een vaste positie die zo gepositioneerd is dat deze start ter hoogte van het midden van het object. Daarna start de scanner links onder met een laserlijn uit te sturen, deze schuift steeds verder op naar rechts tot de maximum opgegeven waarde. Als de uiterste waarde rechts bereikt is, schuift de scanlijn één rij naar boven en gaat weer van links naar rechts. Dit proces herhaalt zich tot de grenswaarde bereikt is. De assen van het object in Python zijn te zien in figuur 116. Eerst roteert het object rond de x-as. Hierna volgt een check-up om te kijken bij welke rotatie(s) de meeste kenmerken duidelijk zichtbaar zijn. Vervolgens roteert het object rond de z-as. Wederom gevolgd door een check-up op zoek naar de optimale rotatie. Een combinatie van de optimale rotatiehoeken resulteert in de optimale scanhoek.

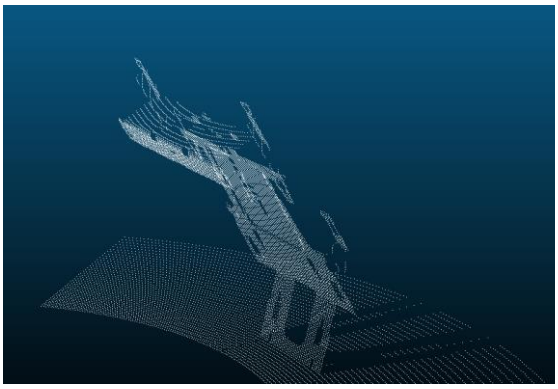


Figuur 116: Toewijzing assen pythonomgeving

In figuur 117 zijn de rotaties van het object rond de x-as te zien. Deze rotatie gaat van -55° tot 55° en tussen elke rotatie zit 5°. Door elke scan afzonderlijk visueel te beoordelen op vlak van aanwezige, te controleren, eigenschappen komt voort dat een rotatie van 30° rond de x-as een goede scan geeft. Deze scan is getoond in figuur 118.

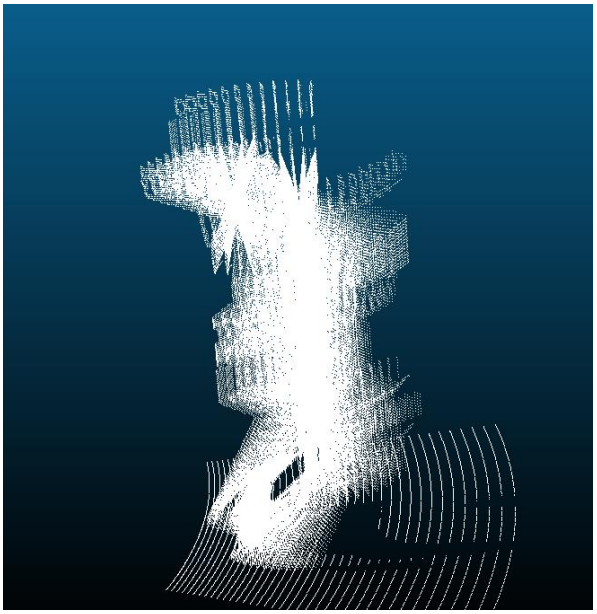


Figuur 117: Rotatie rond x-as met intervallen van 5°

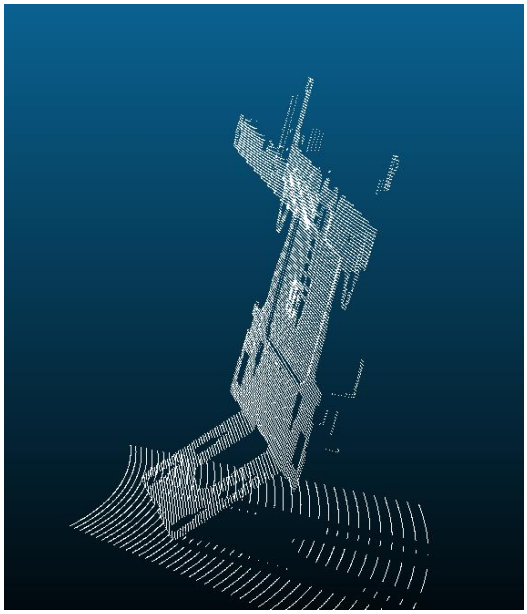


Figuur 118: Rotatie rond x-as bij 30°

Door het optreden van symmetrie roteert het object rond de z-as maar langs één zijde, namelijk van 0° tot 55° en is te zien in figuur 119. Analoog aan het beslissingsproces van optimale rotatie rond de x-as, blijft dat de optimale rotatie rond de z-as 25° bedraagt. Deze scan is te zien in figuur 120.

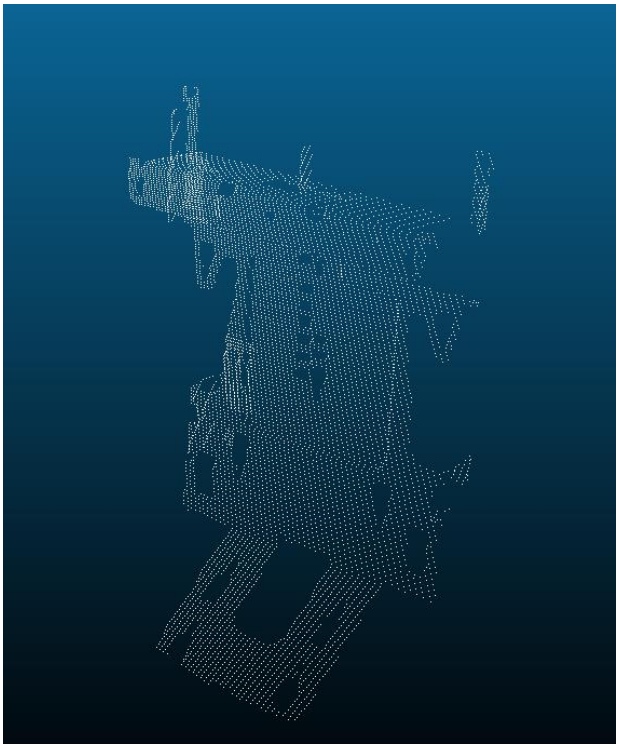


Figuur 119: Rotatie rond z-as met intervallen van 5°



Figuur 120: Rotatie rond z-as bij 25°

De optimale combinatie van rotaties rond de x- en z-as bedragen respectievelijk 30° en 25°. De scan met deze rotaties is te zien in figuur 121. Alle componenten van het object zijn aanwezig om alle metingen uit te kunnen voeren. Echter is hier wel op te merken dat de methode om deze puntenwolken te verkrijgen een puntscan is en geen enkele invloed van reflecties in rekening brengt.



Figuur 121: Optimale combinatie object t.o.v. scanner

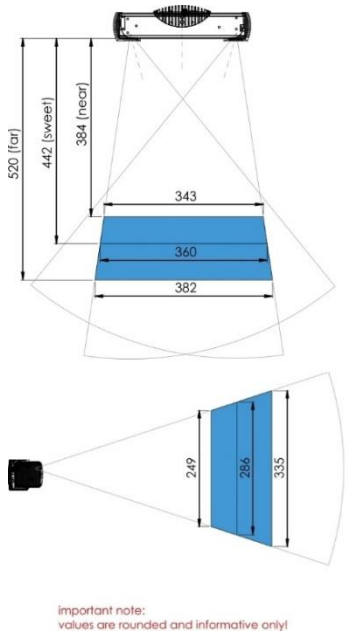
4.5 Kostprijsbepaling

Op basis van de kostprijs van de verschillende losse onderdelen van de proefopstelling is het mogelijk een schatting van de kostprijs te maken voor een opstelling bij Metes. Omdat niet verzekerd is dat deze opstelling altijd perfect dienst zal doen, is in samenwerking met Stemmer Imaging een meer professioneel voorstel bedacht, zie 4.6. De proefopstelling zoals te zien in figuur 95 kent drie hoofdonderdelen, namelijk de camera, de laser en de lineaire geleiding. Aangezien de robotarm van Metes zelf dienst kan doen als lineaire geleiding, valt deze laatste weg. Het is echter belangrijk te beseffen dat een robotarm trillingen en een relatief onnauwkeurige beweging met zich meebrengt. Indien dit optreedt en significante invloed heeft op de metingen is de implementatie van een lineaire geleiding toch vereist.

De prijsofferte voor een camera met bijgeleverde kabels en voeding is in Bijlage E terug te vinden. De prijs voor deze drie componenten is respectievelijk €4068,00, €150,00 en €189,00. De StreamLine Laser – SL komt op €555,00 en een bijhorende montage beugel kost €45,00. Deze laser komt met een 24V DC-voeding die €35,00 kost. De datasheet van de laser is terug te vinden in Bijlage F. De prijs voor de opstelling met losse componenten komt zo op een totaal van €5 042. Met dit bedrag heeft Metes alle benodigde componenten ter beschikking. Niet te vergeten dat er aanzienlijk wat tijd steekt in het juist opbouwen, afstellen, parametreren, kalibreren, etc. van de hele opstelling.

4.6 Voorstel Stemmer Imaging

Stemmer Imaging biedt een kant-en-klare oplossing, zijnde de '*Photoneo Phoxi S'*. De Photoneo Phoxi S berust op het structured-light principe en scant binnen een range van 384 mm tot 520 mm en heeft een optimale focusdiepte van 442 mm waar een object tot 360 mm breed binnen de *scanning range* valt. In dit geval is het te scannen object 176,4 mm breed. De scanner heeft een kalibratienauwkeurigheid van 0,050 mm en een *point-to-point distance* van 0,174 mm [70]. Deze scanner beantwoordt dus aan de gevraagde nauwkeurigheid. De kostprijs bedraagt €11 602 wat bijna dubbel zo duur is als het voorstel met de losse componenten. De Phoxi S heeft echter veel bijkomende voordelen, met name een hogere betrouwbaarheid, robuustheid en lagere gevoeligheid voor omgevingslicht. Deze kant-en-klare oplossing verzekert nauwkeurige scans waar er bij een opstelling met losse onderdelen veel werk zal steken om die nauwkeurigheid (misschien) te halen. Bovendien neemt deze scanner sneller scans dan de SOL-opstelling. Aangezien de verwerkingstijd zo beperkt mogelijk moet zijn, is de keuze voor deze scanner des te interessant. De datasheet van de scanner is beschikbaar in Bijlage G.



Figuur 122: Scanning range Photoneo Phoxi S [70]

4.7 Aflooptijd

De totale aflooptijd van het eerste Halconprogramma (aparte feature-controle) bedraagt 12,79s. De aflooptijden voor de procedures ‘controle_flesjes’, ‘controle_staartje’, ‘controle_hoogte_37_7’, ‘controle_kophoek’, ‘controle_persmoeren’ en ‘controle_steunvoetjes’ bedragen respectievelijk 0,91s, 0,23s, 0,009s, 0,013s, 1,12s en 0,46s.

De aflooptijd voor het tweede Halconprogramma (*distance_object_model_3d*) bedraagt 9,37s. Dit is de aflooptijd op een HP EliteBook 850 G3 computer. Deze tijden kunnen verschillen op een andere PC.

4.8 Conclusie

De genomen conclusie in 4.3.5 blijkt terecht te zijn. De scans met de SOL-opstelling zijn bruikbaar om te onderwerpen aan betrouwbare controles. Metes heeft de keuze uit twee methodes om controlemetingen uit te voeren. Een eerste methode voert metingen uit a.d.h.v. aparte feature-controles (4.4.1.2 a)). Deze methode zal veel tijd vereisen bij het implementeren aangezien elke procedure finetuning vereist. Bij het meten volgens de tweede methode, die metingen uitvoert o.b.v. de afstand tussen referentie en scan (4.4.1.2 b)), zal het nemen van meerdere scans (om een juiste referentie te verkrijgen) het grootste deel van de tijd in beslag nemen. De eerste methode verzekert een correcte controle van de objecten maar mits wat aanvullend onderzoek kan methode twee een meer dynamische en eenvoudiger te implementeren oplossing bieden. Verder liggen de objecten bij het maken van nieuwe scans best onder een hoek van 30° en 25° met respectievelijk de x- en z-as. Het programma van de scanhoeksimulatie voert echter een puntscan uit en geen lijnscan zoals bij de SOL-opstelling. Daarenboven is geen enkele vorm van reflectie in rekening gebracht. De optimale positie van het object t.o.v. de SOL-opstelling kan dus verschillen van de waarde bekomen in de huidige Pythonsimulatie. Dit is wel een bruikbare oplossing voor het gebruik van de Photoneo scanner.

5 Besluit

Uit de testen met de proefopstelling van het eerste project blijkt dat beide softwarepakketten bruikbaar zijn om de nodige metingen uit te voeren. Het gebruiksvriendelijke Merlic, met een kostprijs van €1883, is goedkoper dan Halcon, met een kostprijs van €6200 en €691 voor respectievelijk de development- en run-timelicentie. De mogelijkheden binnen Halcon zijn echter veel uitgebreider dan binnen Merlic. Toch volstaat het controleprogramma van Merlic om betrouwbare metingen uit te voeren voor dit project. De gemiddelde hoekafwijkingen bedragen 0,40° (Merlic) en 0,45° (Halcon). Het verschil in aflooptijd tussen beiden is niet significant en zal geen invloed hebben op de beoordeling welk programma het best presteert. Met zowel het technische als het economische aspect in acht genomen, schuift deze thesis Merlic naar voor als beste optie voor deze uitdaging.

Aangezien het manueel uitbreken van de beugels met een tang de voornaamste bron van fouten is bij de beugels, is het doorvoeren van verder onderzoek met betrekking tot het foutloos uitbreken van de beugels interessant.

Voor het tweede project blijkt dat, na het vergelijken van twee scanprincipes, een SOL-opstelling scans maakt die bruikbaar zijn om metingen op door te voeren. Dit is experimenteel bewezen. Het geschreven controleprogramma in Halcon toont aan dat alle vereiste metingen mogelijk zijn en tevens in staat is om out-of-tolerance objecten te identificeren. Dit programma stelt o.b.v. aanzichten en doorsnedes 2D-contouren op. Deze contouren liggen aan de basis van de verschillende 2D-metingen. Deze thesis bespreekt nog een tweede methode die minder tijdrovend en minder complex is. Deze methode zoekt de afstand van een scan tot een referentie (bestaande uit een combinatie van meerdere scans van een referentie-object). Wegens omstandigheden was het onmogelijk deze methode diepgaand te testen, maar verder onderzoek hieromtrent kan interessante resultaten opleveren aangaande de nauwkeurigheid en flexibiliteit. Om voor beide methodes de kwaliteit van de scans te optimaliseren, is een scanhoeksimulatie uitgevoerd om de optimale scanhoek t.o.v. het object te weten. Door het object 25° en 30° te roteren rond respectievelijk de x- en z-as ligt het object perfect georiënteerd.

Aanvullend onderzoek naar structured-light scanners is, blijkend uit de literatuurstudie en de expertisekennis van Stemmer Imaging, interessant en zeker de moeite waard. Omwille van gebrek aan middelen en tijd heeft deze thesis het pad met diepgaander onderzoek omtrent structured-light scanning onbewandeld gelaten.

Bibliografie

[1] Cognex, “Components of Machine Vision ”. [Online]. Beschikbaar op: https://www.cognex.com/what-is/machine-vision/components. [Geraadpleegd: 07-apr-2020].

[2] H. Schumann-Olsen, “Why 3D machine vision? What’s wrong with 2D machine vision?”, *Zivid*, 25-jun-2018. [Online]. Beschikbaar op: https://blog.zivid.com/why-3d-machine-vision-whats-wrong-with-2d-machine-vision. [Geraadpleegd: 25-mrt-2020].

[3] Cognex, “Machine Vision Lighting”. [Online]. Beschikbaar op: https://www.cognex.com/what-is/machine-vision/components/lighting. [Geraadpleegd: 07-apr-2020].

[4] Cognex, “Machine Vision Lenses”. [Online]. Beschikbaar op: https://www.cognex.com/what-is/machine-vision/components/lenses. [Geraadpleegd: 07-apr-2020].

[5] Hermary, “3D or 2D Machine Vision? -”. [Online]. Beschikbaar op: https://www.hermary.com/learning/when-to-use-2d-over-3d-scanners/. [Geraadpleegd: 22-mrt-2020].

[6] J. Wick, “Parallax Example”, 2006. [Online]. Beschikbaar op: https://commons.wikimedia.org/wiki/File:Parallax_Example.svg. [Geraadpleegd: 25-mrt-2020].

[7] Edmund Optics, “Depth of Field and Depth of Focus”. [Online]. Beschikbaar op: https://www.edmundoptics.com/knowledge-center/application-notes/imaging/depth-of-field-and-depth-of-focus/. [Geraadpleegd: 29-mei-2020].

[8] E. Gray, “Understanding Depth of Field - A Beginner’s Guide”, 2016. [Online]. Beschikbaar op: https://photographylife.com/what-is-depth-of-field. [Geraadpleegd: 29-mei-2020].

[9] L. Lewis, *Introduction to Photographic Principles*, 2de ed. New York: Dover Publications, 1965.

[10] E. Claesen, *Machinevisie: praktische aspecten [cursus]*. Diepenbeek:Gezamenlijke opleiding Industriële Ingenieurswetenschappen UHasselt & KU Leuven, 2009.

[11] Stamos John, “Good images make it easier for vision tools to work”, 13-aug-2013. [Online]. Beschikbaar op: https://www.cognex.com/blogs/machine-vision/good-images-make-it-easier-for-vision-tools-to-work. [Geraadpleegd: 29-mei-2020].

[12] Halcon, “Solution Guide III-B: 2D Measuring”.

[13] E. Demeester, *Visie: beeldvorming [presentatie]*. Diepenbeek: Gezamenlijke opleiding Industriële Ingenieurswetenschappen UHasselt & KU Leuven, 2019.

[14] Matlab, “What Is Camera Calibration? ”. [Online]. Beschikbaar op: https://de.mathworks.com/help/vision/ug/camera-calibration.html. [Geraadpleegd: 06-apr-2020].

[15] G. Vass en T. Perlaki, “Applying and removing lens distortion in post production”, *Proc. 2nd Hungarian Conf. Comput. Graph. Geom.*, pp. 9–16, 2009.

[16] Halcon, *Solution Guide III-C*. 2011.

[17] Dot-vision, “Halcon Compatible Series-chrome on ceramic calibration target”. [Online]. Beschikbaar op: https://www.dot-vision.com/Product/Halcon-Compatible-Series-chrome-on-ceramic.html. [Geraadpleegd: 07-apr-2020].

[18] E. Mostafa Abdel-Bary, “3D Laser Scanners’ Techniques Overview”, *Int. J. Sci. Res.*, vol. 4, nr. 10, pp. 323–331, okt. 2015.

[19] Emmrich Jean-Francois, “What Is Structured Light Scanning?” [Online]. Beschikbaar op: https://blog.medit.com/medit/what-is-structured-light-scanning. [Geraadpleegd: 06-jun-2020].

[20] A. K. Bedaka, A. M. Mahmoud, S. C. Lee, en C. Y. Lin, “Autonomous robot-guided inspection

system based on offline programming and RGB-D model", *Sensors (Switzerland)*, vol. 18, nr. 11, nov. 2018.

[21] M. Nicolescu, *Stereo Vision [presentatie]*. Reno: University of Nevada.

[22] J. P. Kruth en L. De Jonge, *Dimensionele meettechniek deel2: 3D digitalecöordinaten meettechnieken [cursus]*. Acco, 2014.

[23] Adept Turnkey, "The benefits and advantages of Time-of-Flight industrial cameras | AT". [Online]. Beschikbaar op: http://www.adept.net.au/news/newsletter/201111-nov/article_tof_Mesa.shtml. [Geraadpleegd: 25-mrt-2020].

[24] SPAR 3D, "Time-of-Flight vs. Phase-Based Laser Scanners: Right Tool for the Job ", 22-jun-2004. [Online]. Beschikbaar op: https://www.spar3d.com/news/related-new-technologies/time-of-flight-vs-phase-based-laser-scanners-right-tool-for-the-job/. [Geraadpleegd: 08-jun-2020].

[25] H. Yoon, H. Song, en K. Park, "A phase-shift laser scanner based on a time-counting method for high linearity performance", *Rev. Sci. Instrum.*, vol. 82, nr. 7, jul. 2011.

[26] Faro UK, "Laser Line Scanning", apr-2009. [Online]. Beschikbaar op: https://www.faro.com/wp-content/uploads/download-center/download-centre/_0_faro_whp_2009_04ref707-027_-_laser_line_scanning_en_0.pdf?sfvrsn=10. [Geraadpleegd: 26-mrt-2020].

[27] Halcon, "create_sheet_of_light_model [HALCON Operator Reference]". [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/12/en/create_sheet_of_light_model.html. [Geraadpleegd: 26-mrt-2020].

[28] 3D natives, "Laserscanner oder 3D-Scan mit Streifenprojektion - Welcher ist die bessere Wahl? - 3Dnatives", 08-aug-2019. [Online]. Beschikbaar op: https://www.3dnatives.com/de/laserscanner-oder-3d-scan-mit-streifenprojektion-welcher-ist-die-bessere-wahl-080820191/. [Geraadpleegd: 26-mrt-2020].

[29] Vink Laura, "Wat is sluitertijd?" [Online]. Beschikbaar op: https://vinkacademy.nl/fotografietips/basiskennis-fotografie-uitleg-over-sluitertijd/. [Geraadpleegd: 31-mei-2020].

[30] Y. M. Amir en B. Thörnberg, "High Precision Laser Scanning of Metallic Surfaces", *Int. J. Opt.*, vol. 2017, pp. 1–13, 2017.

[31] Jihong Chen, Daoshan Yang, Huicheng Zhou, en S. Buckley, "Avoiding spurious reflections from shiny surfaces on a 3D real-time machine vision inspection system", in *IMTC/98 Conference Proceedings. IEEE Instrumentation and Measurement Technology Conference. Where Instrumentation is Going (Cat. No.98CH36222)*, 1998, vol. 1, pp. 364–368.

[32] C. Yang, N. D. Aranoff, P. Green, en N. Tavassolian, "Fringe Projection Techniques: Whither we are?", in *Optics and Lasers in Engiering*, vol. 48, nr. 2, 2010, pp. 133–140.

[33] H. Zhao, H. Jiang, H. P. Seidel, H. P. A Lensch, Y. Xu, en D. G. Aliaga, "3D shape measurement in the presence of strong interreflections by epipolar imaging and regional fringe projection", *Opt. Express*, vol. 84, nr. 32, p. 104581, 2013.

[34] PhotoModeler, "PhotoModeler's New Point Cloud Meshing Capabilities", 01-mei-2013. [Online]. Beschikbaar op: https://www.photomodeler.com/photomodelers-new-point-cloud-meshing-capabilities/. [Geraadpleegd: 03-jun-2020].

[35] T. Rabbani, F. A. van den Heuvel, en G. Vosselman, "Segmentation of point clouds using smoothness constraint", Delft, jan. 2006.

[36] MathWorks, "Edge detection methods for finding object boundaries in images". [Online]. Beschikbaar op: https://nl.mathworks.com/discovery/edge-detection.html. [Geraadpleegd: 02-jun-2020].

[37] K. Klasing, “Surface-based Segmentation of 3D Range Data”, *Control Eng.*, pp. 1–18, 2009.

[38] J. Wang en Z. Yu, “Surface feature based mesh segmentation”, *Comput. Graph.*, vol. 35, nr. 3, pp. 661–667, 2011.

[39] T. Heimann en H. Delingette, *Biological and medical physics, biomedical engineering*. Berlijn: Springer, 2010.

[40] A. Biesdorf *e.a.*, “Segmentation and quantification of the aortic arch using joint 3D model-based segmentation and elastic image registration”, *Med. Image Anal.*, vol. 16, nr. 6, pp. 1187–1201, 2012.

[41] C. H. P. Nguyen en Y. Choi, “Comparison of point cloud data and 3D CAD data for on-site dimensional inspection of industrial plant piping systems”, *Autom. Constr.*, vol. 91, nr. September 2017, pp. 44–52, 2018.

[42] Halcon, “find_surface_model [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/12/en/find_surface_model.html. [Geraadpleegd: 01-mei-2020].

[43] Halcon, “register_object_model_3d_pair [HALCON Operator]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/register_object_model_3d_pair.html. [Geraadpleegd: 01-mei-2020].

[44] Halcon, “register_object_model_3d_global [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/register_object_model_3d_global.html. [Geraadpleegd: 01-mei-2020].

[45] Halcon, “open_framegrabber [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/11/en/open_framegrabber.html. [Geraadpleegd: 12-dec-2019].

[46] Halcon, “HALCON Application Note: The Art of Image Acquisition”, *Edition 1*, 2002. .

[47] Halcon, “threshold [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/threshold.html. [Geraadpleegd: 08-dec-2019].

[48] Halcon, “opening_circle [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/11/en/opening_circle.html. [Geraadpleegd: 08-dec-2019].

[49] Halcon, “dilation_circle [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/dilation_circle.html. [Geraadpleegd: 08-dec-2019].

[50] Halcon, “remove_noise_region [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/12/en/remove_noise_region.html. [Geraadpleegd: 08-dec-2019].

[51] Halcon, “gen_contour_region_xld [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/11/en/gen_contour_region_xld.html. [Geraadpleegd: 08-dec-2019].

[52] Halcon, “smooth_contours_xld [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/11/en/smooth_contours_xld.html. [Geraadpleegd: 08-dec-2019].

[53] Halcon, “rotate_image [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/11/en/rotate_image.html. [Geraadpleegd: 08-dec-2019].

[54] Halcon, “select_contours_xld [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/12/en/select_contours_xld.html. [Geraadpleegd: 08-dec-2019].

[55] Manutan, “Motortransportband MIB - Met band met breedte 250 mm”. [Online]. Beschikbaar

op: https://www.manutan.be/nl/mab/aangedreven-transportband-mib-somefi. [Geraadpleegd: 07-mrt-2020].

[56] R. Components, “Siemens 6ES7211-1BE40-0XB0”. [Online]. Beschikbaar op: https://benl.rs-online.com/web/p/plc-cpus/8624455/. [Geraadpleegd: 07-mrt-2020].

[57] R. Elektronik, “SM 1223 16 24V”. [Online]. Beschikbaar op: https://www.reichelt.com/be/nl/s7-1200-digitale-uitvoer-signaalmodule-sm-1223-16-24v-p201778.html?&trstct=pos_0&nbc=1. [Geraadpleegd: 07-mrt-2020].

[58] reichelt elektronik, “WAGO 852-111”. [Online]. Beschikbaar op: https://www.reichelt.com/be/nl/5-port-100base-tx-industrial-eco-switch-wago-852-111-p114436.html?&trstct=pos_0&nbc=1. [Geraadpleegd: 07-mrt-2020].

[59] Tameson, “Dubbelwerkende Mini Cilinder 16-200mm ISO-6432”. [Online]. Beschikbaar op: https://magneetventielshop.nl/pneumatiek/cilinder/rond-iso-6432-8-25mm/mcmi-11-16-200-16-200mm-dubbelwerkende-mini-cilinder-iso-6432.html. [Geraadpleegd: 07-mrt-2020].

[60] RS Components, “Sick WL11-2P2430”. [Online]. Beschikbaar op: https://benl.rs-online.com/web/p/photoelectric-sensors/1212719/. [Geraadpleegd: 07-mrt-2020].

[61] RS Components, “Schneider Electric XB4BA21”. [Online]. Beschikbaar op: https://benl.rs-online.com/web/p/push-button-complete-units/3308616/. [Geraadpleegd: 07-mrt-2020].

[62] RS Components, “Schneider Electric XALK178”. [Online]. Beschikbaar op: https://benl.rs-online.com/web/p/emergency-stop-push-buttons/7951295/. [Geraadpleegd: 07-mrt-2020].

[63] Ensenso GmbH, “Ensenso N30 / N35”. [Online]. Beschikbaar op: https://www.ensenso.com/portfolio-item/n3x/. [Geraadpleegd: 14-mrt-2020].

[64] Halcon, “calibrate_sheet_of_light [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/calibrate_sheet_of_light.html. [Geraadpleegd: 26-mrt-2020].

[65] Halcon, “create_sheet_of_light_calib_object [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/12/en/create_sheet_of_light_calib_object.html. [Geraadpleegd: 26-mrt-2020].

[66] C. van Kessel, “Re: Matchingsprincipes”. Persoonlijke email, (Juni 2, 2020).

[67] M. Palmans, “Automatische optimaleparameterbepaling van random bin picking visiealgoritmen”, Diepenbeek: Gezamenlijke opleiding Industriële Ingenieurswetenschappen UHasselt & KU Leuven, 2017.

[68] Halcon, “union_object_model_3d [HALCON Operator Reference]”. [Online]. Beschikbaar op: https://www.mvtec.com/doc/halcon/13/en/union_object_model_3d.html. [Geraadpleegd: 07-jun-2020].

[69] Halcon, “reconstruct_3d_object_model_for_matching”. MVTec.

[70] Photoneo, “PhoXi 3D Scanner S”. [Online]. Beschikbaar op: https://www.photoneo.com/products/phoxi-scan-s/. [Geraadpleegd: 21-mei-2020].

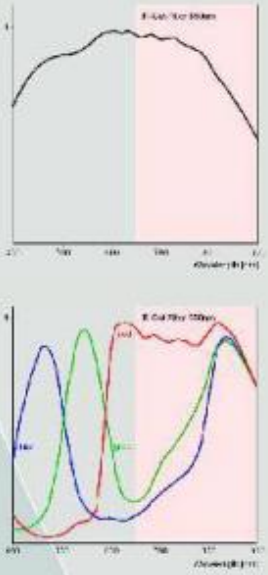
Bijlagen

Bijlage A

- 752 x 480 pixels (WVGA)
- High quality CMOS sensors with square pixels, Global Shutter
- Up to 87 frames/s in full frame mode, over 100 frames/s using Area of Interest (AOI)
- Digital output
- uEye SDK (compatible to the FALCON SDK) and demo programs in scope of supply
- C-Mount
- Ultra small size and less weight: 34 x 32 x 27,4 mm (HxWxD), 62g
- Camera control via USB 2.0
- Mini-B USB AND screwable Micro D-SUB connector for industrial use
- Driver and Interfaces: Windows and Linux SDK, Twain, Direct Show (WDM), ActiveX, ActivVisionTools, Common Vision Blox, HALCON and NeuroCheck
- OEM modules available

uEye®

USB2.0 camera



CAMERA SPECIFICATIONS	
	UI-1220-M/C
Resolution (pixels)	752x480
M - Model (b/w)	Yes
C - Model (colour)	Bayer RGB Pattern
Chip type / Chip size	1/3 Inch/5,4 mm
Cellsize (HxV) in µm	6 x 6
Shutter	Global
Scanning System	Progressive Scan
Sensor dynamics	60 dB
Frames per second	87
Binning	hor. & vert.
Subsampling	--
AOI	hor. & vert.
Mirror /Flip	hor. & vert.
Pixel Clock Frequency (Min/Max)	5/40 MHz
Shutter speed	60µs - 1600ms
Hardware Gain	0-6dB
Hardware Trigger	async
Lens mount	C-Mount
IR filter	yes, removable
Interface	USB 2.0
Power supply	< 1 Watts
Operating temperature	0° - 50° Celsius
Regulations	CE, FCC Class B
Size (HxWxD) in mm	34 x 32 x 27,4
Weight	62 g

The uEye CMOS cameras are ultra compact cameras with USB 2.0 interface for industrial use. They are equipped with high quality sensors, industrial Design, C-Mount, a standard USB and additionally a screwable USB interfaces which offers also the trigger and digital output. Ready for professional use.
With it's USB 2.0 architecture, the cameras are ideally suited for use with PC, Notebook and Embedded Systems in image processing and factory automation.

The powerful -free of charge- uEye SDK and the yet available drivers and Interfaces for popular machine vision software allow e asy integration in applications. It's so easy!

AVAILABLE DRIVER AND INTERFACES

Windows 2000 und XP: uEye SDK, TWAIN, ActiveX, Direct Show (WDM), ActivVisionTools, Common Vision Blox, HALCON, NeuroCheck. Linux: uEye SDK

ACCESSORIES:

- USB cable
- Lenses
- USB2.0 Interfaces
- Software

The uEye cameras on the internet: www.net-usa-inc.com

Bijlage B

PHAER
COMPUTER VISION EXPERTS

Prijsofferte

Prijsofferte nr.	19000376
Klantnummer	855
Documentdatum	19/02/2020
Einddatum	19/03/2020
Ondernemingsnr.	BTW BE 0208.359.859
Uw referentie	Prijsofferte visiesysteem
Incoterms	DAP - Diepenbeek

Universiteit Hasselt
Brecht Nijssen
Wetenschapspark 2
B -3590 Diepenbeek

Gentbrugge, 19/2/2020

Geachte heer, mevrouw,

Naar aanleiding van uw prijsaanvraag, is het ons een genoegen om u de volgende aanbieding te bezorgen:

Artikel	*Omschrijving*	*Aantal*	*E.P.*	*Korting*	*Prijs*
	Casenummer: 2875 Omschrijving: Prijsofferte visiesysteem Behandelaar: De Witte Diego				
9136	Baumer VCXU-04C *USB3, Color, Sony IMX287, 1/2.9? CMOS, 728 × 544 pixel, 431 fps, C-mount*	1,00	426,92	7%	397,04
6706	Kowa LM8JCM-V *8mm/F1.4; C mount - Ruggedized Version - all glued*	1,00	192,92	7%	179,42
7912	CCS LDL2-158X16SW2-WD *Led Bar Light*	2,00	770,00	7%	1.432,20
0117	CCS DF-LDL2-158X16 *Diffuser*	2,00	33,00	7%	61,38
9725	CCS CN-4024-2-EIPT	1,00	880,00	7%	818,40

Phaer BVBA | Kerkstraat 108 | B -9050 Gentbrugge, België
T. +32 9 261 61 60 | F. +32 9 261 61 69 | E. phaer@phaer.eu | www.phaer.eu
BTW BE-0889.536.619 - RPR Gent | Belfius BE27 0682 4755 8373 - BIC: GKCCBEBB

PHAER
COMPUTER VISION EXPERTS

Prijsofferte

Prijsofferte nr.	19000376
Klantnummer	855
Documentdatum	19/02/2020
Einddatum	19/03/2020
Ondernemingsnr.	BTW BE 0208.359.859
Uw referentie	Prijsofferte visiesysteem
Incoterms	DAP - Diepenbeek

Universiteit Hasselt
Brecht Nijssen
Wetenschapspark 2
B -3590 Diepenbeek

Gentbrugge, 19/2/2020

Artikel	*Omschrijving*	*Aantal*	*E.P.*	*Korting*	*Prijs*
	Power Supply - Light intensity and other settings can be controlled with explicit messages via EtherNet/IPTM communications and with I/O commands via TCP/IP communications.				
	Transport- en handling kost	1,00	18,00		18,00
	De levertermijn is 15 tot 20 werkdagen. De 7% universiteitskorting wordt eenmalig per jaar op deze aangeboden artikelen toegekend.				

De genoemde prijzen zijn ex BTW, en de incoterms zijn zoals vermeld in de offerte. Onder beding van uw goedkeuren, maken wij uw goederen klaar voor transport en geven wij UPS opdracht om te leveren met 'standard service'. Verzekering voor verlies en schade bij transport is optioneel en beperkt tot condities van www.ups.com.

Handelsvoorwaarden Phaer zijn van toepassing. Die vindt u in bijlage en op www.phaer.be/nl/phaer.

BTW	21	21 %	2.906,44	610,35	**Subtotaal**	**2.906,44**
					BTW	610,35
Betalingsvoorwaarden: 30 dagen factuurdatum					Totaal EUR	3.516,79

Wij danken u voor de interesse in onze producten en diensten. Indien u verdere vragen heeft, dan is onze customer care manager graag bereikbaar via phaer@phaer.eu. Wij kijken ernaar uit om uw opdracht te mogen ontvangen via orders@phaer.eu, die we met zorg en spoed zullen behandelen.

Hoogachtend,

Bijlage C

Allied Vision

GigE Vision

Manta

G-040

- Sony IMX287 CMOS sensor
- Power over Ethernet option
- Angled-head and board level variants
- Video-iris lens control

GigE Vision camera featuring the Sony IMX287 CMOS sensor

Manta G-040 is a 0.40 megapixel machine vision camera with a GigE compliant Gigabit Ethernet port and Hirose I/O port. Manta G-040 incorporates the high quality Type 1/2.9 (6.3 mm diagonal) Sony IMX287 CMOS sensor with Pregius global shutter technology. The Sony IMX287 CMOS sensor provides high frame rates, high sensitivity, and excellent picture quality. The Manta G-040 is offered in monochrome and color models. At full resolution, this camera achieves 286.3 frames per second. With a smaller region of interest, higher frame rates are possible. You can also achieve a higher frame by adjusting the packet size. The Manta G-040 is an ideal replacement for legacy CCD models.
Manta is one of Allied Vision's versatile GigE Vision cameras with a wide range of features. Particular highlights are the three look-up tables, sophisticated color correction capabilities, a robust metal housing, and many modular options. By default monochrome models ship with B 270 ASG protection glass and color models ship with a Type Hoya C-5000 IR cut filter.

Benefits and features

- Monochrome (G-040B) and color (G-040C) models
- GigE Vision interface with Power over Ethernet option
- Screw mount RJ45 Ethernet connector for secure operation in industrial environments
- Supports cable lengths up to 100 meters (CAT-6 recommended)
- Trigger over Ethernet Action Commands allow for a single cable solution to reduce system costs
- Easy camera mounting via standard M3 threads on top and bottom of housing or optional tripod adapter
- Comprehensive I/O functionality for simplified system integration
- Popular C-Mount lens mount
- Easy software integration with Allied Vision's Vimba Suite and compatibility to the most popular third party image-processing libraries.

Bijlage D

N30 / N35

N30 and N35 Stereo 3D Cameras

The N30 and N35 are our newest models of the compact N-series. They feature a Gigabit Ethernet connector with PoE support and a high-power LED pattern projector. The N35 projector is addionally equipped with our new FlexView technique. The camera housings are IP65 and IP67 rated and thus water and dust proof.

Specifications

- 1280 x 1024 px 1/1.8" global shutter CMOS sensors
- FlexView pattern projection module (N35 only)
- Interface: Gigabit Ethernet with Power over Ethernet
- Water and dust proof according to IP65, IP67
- Optional 12-24V external power supply
- Available focal lengths: 6 – 16 mm
- Light sensitive optics with F=1.6 apertures
- Available with blue (465nm) and infrared (850nm) wavelengths
- 12-24V GPIO, trigger input and flash output
- All connectors lockable
- Dimensions: 175 x 50 x 52 mm
- Factory calibrated

All camera models are available at IDS

Bijlage E

photon focus

Quote **VO2000708**

UHasselt
Jan Dewez
Agoralaan building D
Campus Diepenbeek
3590 Diepenbeek
Belgium

Document Date:	22.04.2020	VAT No.:	CHE-114.626.353
Customer No.:	5217	O/Contact:	Sergio Manuel/dic
Quote Date:	16.04.2020	O/E-Mail:	manuel@photonfocus.com
External Doc. No.:	E-MAIL	O/Phone No.:	+41 55 451 00 00
Shipping Date:	3-4 weeks ARO		

Pos.	Description	Shipping	Quantity		Price EUR	Amount EUR	
10	605020.002 MV1-D1024E-3D02-160-G2-8 The product is in conformity with the provisions of the following European Directives: - 2014/35/EU (LVD) - 2014/30/EU (EMC) - 2011/65/EU (RoHS) For details consult the EC Declaration of conformity EU_DOC_No_PFD000005 Country of Origin: CH Customs Tariff No.: 8525.8000		1	pcs.	4'068.00	4'068.00	Z
20	705020.004 Universal power supply 12pin 12V Power plug: US/JP, EU or GB Country of Origin: EU / Customs Tariff No.: 8504.4000		1	pcs.	189.00	189.00	Z
30	704090.011 P/T/S cable, Hirose, 2m (power/trigger/strobe) Country of Origin: CH Customs Tariff No.: 8544.4221		1	pcs.	150.00	150.00	Z

VAT ID	VAT %	VAT Base Amount	VAT Amount			
Z	0	4'407.00	0.00			
				Total excl. VAT	EUR	4'407.00
				Total VAT	EUR	0.00
				Total incl. VAT	**EUR**	**4'407.00**

This quote is valid: :	30 days
Payment Terms:	prepayment
Shipping Terms:	EXW

The delivery will be charged to UHasselt.

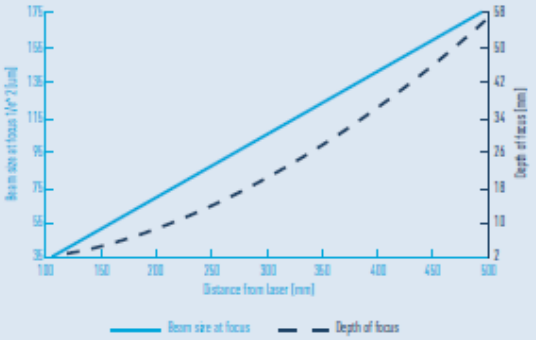
Bijlage F

LASER DIODE MODELS AND FOCUSING OPTIONS

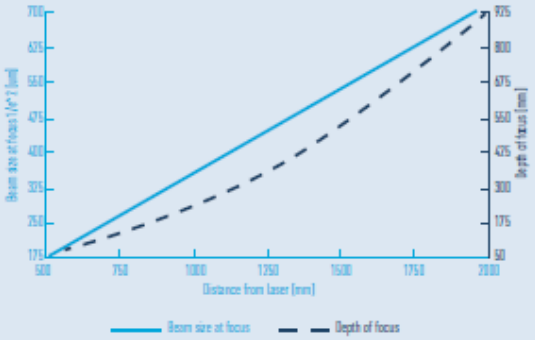
At Osela we provide many different focusing options giving you the flexibility to choose the one that best suits your application. The Streamline laser is free focusable externally without removing any optics. From the graphs below, note the beam size and Depth of Focus (DOF) values and then multiply by the K constants for the laser diode model and focus option of choice (A, B, C, D or E).

Example: From the graphs at 400 mm working distance, Focus = 140µm, DOF = 36 mm. Than for Laser Model 660 nm 130 mW the line thickness at focus for OPTION A will be 212um (i.e.140 um x 1.52). Its depth of focus will be 88.92mm (i.e. 36mm x 2.47).

SHORT RANGE



LONG RANGE



DIODE MODEL				FOCUSING & DOF OPTIONS AND CONSTANT									
WAVELENGTH (nm)	DIODE POWER (mW)	WAVELEGNTH TOLERENCE (nm)	OPERATING CURRENT[2] (mA)	A		B		C		D		E	
				K_{FOCUS}	K_{DOF}	K_{FOCUS}	K_{DOF}	K_{FOCUS}	K_{DOF}	K_{FOCUS}	K_{DOF}	K_{FOCUS}	K_{DOF}
405[1]	35	+5/-5	50	0.68	0.80	1.65	4.74	0.28	0.13	0.98	1.69	2.39	10.00
	100	+10/-5	70	0.64	0.72	1.46	3.74	0.26	0.12	0.93	1.52	2.12	7.88
450[1]	100	+10/-10	100	0.66	0.69	1.95	6.00	—	—	0.96	1.45	2.83	12.65
520[1]	50	+10/-5	145	0.74	0.75	2.80	10.77	0.41	0.24	1.07	1.58	4.06	22.69
635	5	+5/-5	50	0.79	0.70	2.58	7.44	0.45	0.22	1.15	1.47	3.75	15.67
	10	+8/-4	60	0.96	1.02	2.58	7.44	0.54	0.32	1.39	2.15	3.75	15.67
640	25	+3/-10	90	0.96	1.02	2.29	5.86	0.54	0.32	1.39	2.15	3.33	12.35
	45, 80	+5/-5	120, 185	0.96	1.02	2.06	4.73	0.54	0.32	1.39	2.15	2.99	9.98
	150	+3/-8	185	0.65	0.47	1.88	3.93	0.37	0.15	0.94	0.99	2.74	8.28
650	5	+10/-5	48	0.73	0.56	2.38	6.09	0.41	0.18	1.05	1.19	3.46	12.84
	10	+10/-5	55	0.73	0.56	2.26	5.46	0.41	0.18	1.05	1.19	3.27	11.51
660	35	+5/-10	100	0.95	0.96	2.52	6.84	0.53	0.30	1.37	2.02	3.66	14.41
	50, 100	+5/-5	125,175	1.52	2.47	2.14	4.92	0.85	0.41	2.20	5.22	3.11	10.37
	130	+5/-5	200	1.52	2.47	2.14	4.92	0.85	0.41	2.20	5.22	3.11	10.37
690	35	+5/-10	95	1.15	1.36	2.62	7.10	0.47	0.23	1.66	2.87	3.80	14.96
	50	+10/-10	150	1.09	1.22	2.11	4.63	0.44	0.20	1.58	2.57	3.07	9.75
785	75, 120	+10/-10	150, 200	1.57	2.23	2.83	7.25	0.64	0.37	2.28	4.71	4.11	15.27
810	150	-0.6	230	1.52	2.03	3.29	9.46	0.62	0.34	2.21	4.27	4.77	19.94
830	50, 100	+10/-10	115, 135	1.19	1.21	3.00	7.66	0.67	0.38	1.73	16.14	4.35	16.14
	200	+10/-10	240	1.03	0.91	3.37	9.72	0.58	0.29	1.50	1.93	4.90	20.48

[1] 9 to 30V operation

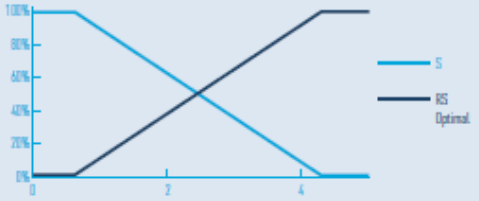
[2] Measured at 25°C at operating voltage of 5V for ≥635nm and 12V for 405, 450 and 520nm models)

MODULATION

The Streamline laser can be modulated by an external 0 to 5V external signal through the white wire. The **S type** modulation is included by default with the Streamline Module.

FUNCTION	CODE	ON	OFF
TTL	T	0 to 2V	3 to 5V
Reverse TTL	RT	3 to 5V	0 to 2V

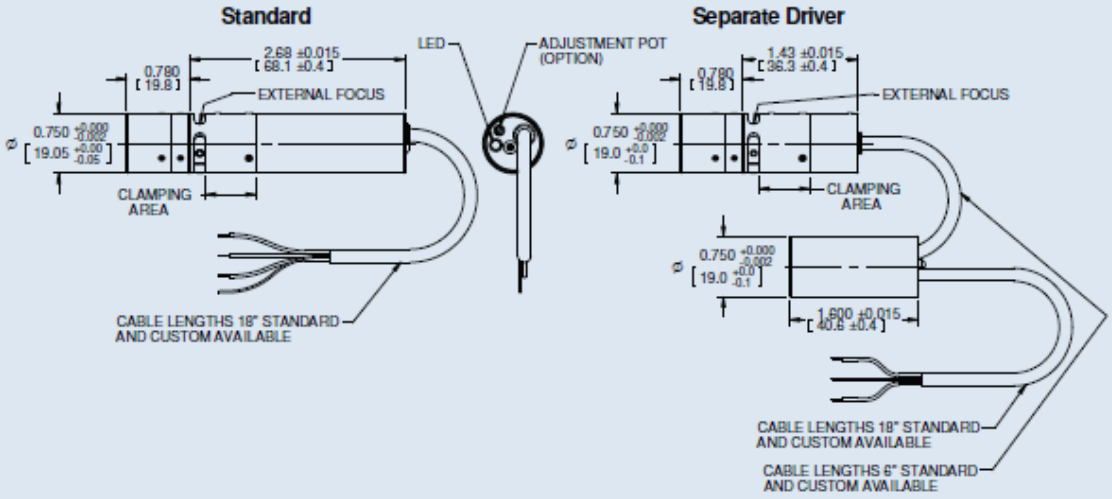
Note: One modulation input needs to be selected, S (default), RS, T or RT



SPECIFICATIONS

Bore sight (mrad)	< 3 mrad
Wavelength Drift	= 0.25 nm/ degC
Pointing Stability	< 6 µrad/°C
Modulation Rise/Fall time	< 5µ sec, 100% modulation depth (10 Kohm input impedance)
Protections (Built in)	ESD, Over voltage (up to 30 VDC), Over-temp Shutoff (> 50 deg C)
Long term Power stability (8 hours)	< 3 %, 2 minute warm up time
Operating Voltage	5 ± 0.5VDC, 4.5 to 30V Optional (9-30V for < 635 nm)
Working Temp Range	-10 to to +50 °C (housing)
Weight	< 50 g
Power Supply Cable	18 inches 3 conductors Belden 9533, with flying leads
ESD Protection	Level 4
Shock Tolerance	30g, 6ms, functional

MECHANICAL SPECIFICATIONS



STREAMLINE SINGLE LINE GENERATOR

FIG 1 - INTENSITY DISTRIBUTION ALONG THE LINE

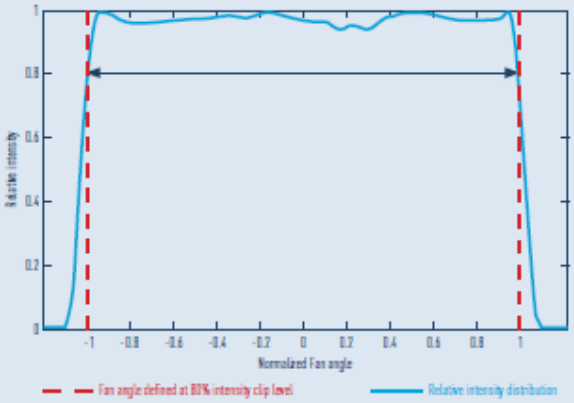
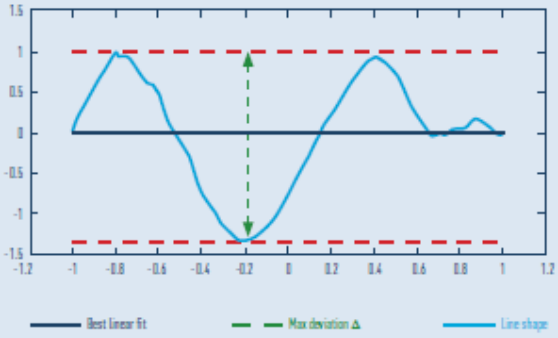


FIG 2 - LINE STRAIGHTNESS



SPECIFICATIONS

SPECIFICATIONS		VALUES
Uniformity (line intensity distribution along the line)[2]	$\frac{Imax - Imin}{Imax + Imin}$	20% (typical) ≤7.5%[1]
Relative intensity clip that define the fan angle		80%
Contained energy In the fan angle	$\frac{\text{Energy in fan angle}}{\text{total energy}}$	≥95%
Line Straightness (deviation from the best linear fit)[2]	$\frac{\Delta}{L\ (\text{line lenght})}$	≤0.1% ≤0.05%[1]
Fan angle		1 to 90° [3]
Fan angle tolerance (line diverging angle from the tip of the laser)		+1.0/-0.5° (FA <30 °) +1.5/-0.5° (FA ≥30°)

[1] For SL *Plus* (see Streamline *PLUS* datasheet).
[2] Uniformity and straightness are measured at 80% of the fan angle (100% for SL *Plus*).
[3] Available Fan Angle (°) 1, 5, 10, 15, 20, 30, 38, 45, 60, 75, 90 custom upon demand.

ORDERING CODE

SL	-	XXX Wavelength	-	XXX Diode Power	-	X Electronic	X Focusing Option	-	XX Fan Angle	-	XXX-XX Multi beams	-	XXXXX Option
		see table		see table		S	A		1, 5, 10		(Optional)		SD
						RS	B		15, 20		Refer to the		24V
						T	C		30, 38		Multi-dots and		
						RT	D		45, 60		Multi-Lines		
							E		75, 90		page		

OSELA inc. 1869 32nd Avenue, Lachine, QC, H8T 3J1 Canada • info@osela.com • 514 631.2227 osela.com

Bijlage G

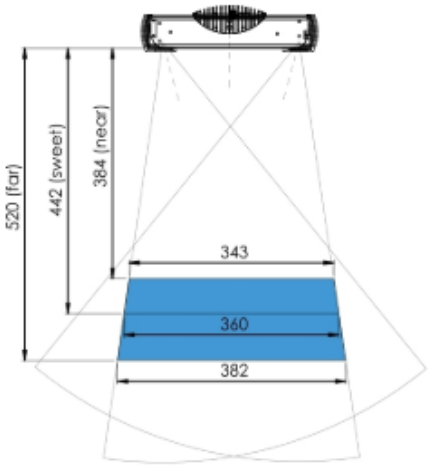
PhoXi 3D Scanner S

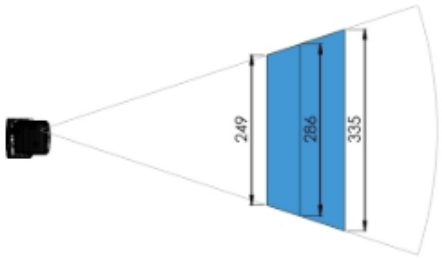
Average scanning volume:
360 x 290 x 70 mm.

Datasheet

Parameter	Value
Resolution	Up to 3.2 Million 3D points
Scanning range	384 – 520 mm
Optimal scanning distance (focus)	442 mm
Scanning area (at focus distance)	360 x 286 mm
Point to point distance (at focus distance)	0.174 mm
Calibration accuracy	0.050 mm
Temporal noise	0.050 mm
Scanning time	250 – 2250 ms
Dimensions	77 x 68 x 296 mm
Baseline	230 mm
Weight	900 g
3D points throughput	16 Million points per second
GPU	NVIDIA Maxwell™ 1 TFLOPS with 256 NVIDIA® CUDA® Cores

Scanning range





important note:
values are rounded and informative only!