2016•2017
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: energie

Masterproef

Automatische optimaleparameterbepaling van random bin picking visiealgoritmen

Promotor :
Prof. dr. ir. Eric DEMEESTER

Copromotor :
De heer Maarten VERHEYEN

Martijn Palmans

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

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2016•2017

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterproef

Automatische optimaleparameterbepaling van random bin picking visiealgoritmen

Promotor :
Prof. dr. ir. Eric DEMEESTER

Copromotor :
De heer Maarten VERHEYEN

Martijn Palmans

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

universiteit hasselt | KU LEUVEN

Voorwoord

Bij het beëindigen van mijn eindwerk wil ik enkele mensen bedanken die het mogelijk gemaakt hebben om deze oprecht mooie jaren van deze studie af te ronden.

Op de eerste plaats wil ik mijn ouders bedanken die mij de kans gegeven hebben om deze studie te volgen door zowel hun financiële als morele steun.

De uitwerking van dit eindwerk is tot stand gekomen dankzij de hulp van heel wat mensen. Daarvoor wil ik ook een gepast dankwoord tot hen richten.

In het bijzonder wil ik mijn promotor dr. Ir. Eric Demeester en co-promotor ing. Maarten Verheyen bedanken voor hun bereidheid om mijn vragen steeds te willen te beantwoorden en mij nauwgezet te begeleiden met deze opdracht.

Ook de heer Johan De Vidts, verkoopleider bij Batenburg Mechatronica B.V., wil ik bedanken voor het bezorgen van de maandelijkse licentie voor het visieprogramma Halcon van het bedrijf MVTec Software Gmbh.

Verder zou ik graag ook nog een dankwoordje richten tot dr. Jeroen Lievens voor zijn taalkundige ondersteuning gedurende het hele academiejaar.

Inhoudsopgave

Voorwoord 1
Lijst van tabellen 5
Lijst van figuren 7
Verklarende woordenlijst 11
Abstract 13
Abstract in English 15
1 Inleiding 17
1.1 Situering 17
1.2 Probleemstelling 17
1.3 Doelstellingen 20
1.4 Materiaal en methode 21
2 Literatuurstudie 23
2.1 Inleiding 23
2.2 Halcon 23
2.3 Verschillende methodes/technieken om automatisch parameters te bepalen 26
2.4 Conclusie 30
3 3D-visietechnologie 31
3.1 Inleiding 31
3.2 Stereovisie 31
3.2.1 Algemeen 31
3.2.2 Werkingsprincipe 31
3.2.3 Soorten stereovisiecamera's 34
3.3 Sheet-of-light 36
3.3.1 Algemeen 36
3.3.2 Werkingsprincipe 36
3.3.3 Photonfocus-camera 38
3.4 Bin picking-opstelling 40
4 Automatische parameterbepaling 43
4.1 Inleiding 43

4.2 Parameters en hun bereik 43
4.2.1 Parameters van operator “create_surface_model” 43
4.2.2 Parameters van de operator “find_surface_model” 44
4.3 Parameterbepaling aan de hand van geneste for-lussen 46
4.4 Matching problemen 48
4.4.1 Positieprobleem 48
4.4.2 Meerdere matches 49
4.4.3 Scoreprobleem 50
4.4.4 Oplossing 51
4.5 Experimenten/methodes 53
4.5.1 Geneste for-lussen 53
4.5.2 Outputs Score, Pose en tijd 55
4.5.3 Trainingsscenes en testscenes 56
4.6 Algoritme voor automatische optimaleparameterbepaling 58
4.6.1 Stap 1: bepalen basisparameters 59
4.6.2 Stap 2: parameterbepaling met behulp van trainingsscenes 67
4.6.3 Stap 3: controle parameters aan de hand van testscenes 73
4.7 Besluit 81
5 Besluit 83
Bibliografie 85
Bijlagen 87
Bijlage A: parametercombinaties 87

Lijst van tabellen

Tabel 3-1: Specificaties Ensenso N35-602-16-IR 34

Tabel 3-2: Specificaties Kinect V2 35

Tabel 3-3: Specificaties Photonfocus-camera 39

Tabel 3-4: Specificaties lens Cosmicar Pentax C21211KP 41

Tabel 3-5: Specificaties bandpassfilter 42

Tabel 3-6: Specificaties robot Epson C3-A601S 42

Tabel 4-1: Parameters met hun waarde en stapgrootte 61

Tabel 4-2: Beste basisparameters 63

Tabel 4-3: Parameterset 1 + generische parameters voor trainingsscenes 67

Tabel 4-4: Parameters + waardes 74

Tabel 4-5: Parameters met nieuwe minimale en maximale waarde + stapgrootte voor testscenes 77

Tabel 4-6: Parameters met nieuwe minimale en maximale waarde + stapgrootte met verlaagde score voor testscenes 79

Tabel 4-7: Overzicht resultaten per methode 81

Lijst van figuren

Figuur 0-1: Voorbeeld van een operator en de parameters die meegegeven moeten worden 11
Figuur 0-2: Voorbeeld parameterset 12
Figuur 0-3: Voorbeeld van een scene 12
Figuur 1-1: 3D-geprinte brilmonturen die als model kunnen dienen 18
Figuur 1-2: 3D-geprinte zolen die als model kunnen dienen 18
Figuur 1-3: Opstelling met lineaire geleiding (1), Photonfocus-camera (1A), laser (1B) en pikrobot (2) 19
Figuur 1-4: Opstelling met Ensenso-sensor 20
Figuur 2-1: Lay-out Halcon met (1) grafisch venster, (2) operatorvenster, (3A) iconische variabelen, (3B) controlevariabelen en (4) programmeervenster 24
Figuur 2-2: Matchingmethodes in functie van de toepassingen 26
Figuur 2-3: Flowchart voor het bepalen van de optimale thresholdwaarde 29
Figuur 3-1: Werkingsprincipe stereovisie 32
Figuur 3-2: Kalibratierooster stereovisieopstelling 33
Figuur 3-3: Ensenso N35-602-16-IR, stereocamera 34
Figuur 3-4: Opbouw Kinect V2 35
Figuur 3-5: Opstelling sheet-of-light 36
Figuur 3-6: Occlusies en schaduweffecten naargelang de opstelling van de camera en laser 37
Figuur 3-7: Verandering van de laserlijn wijst op een hoogteverschil 37
Figuur 3-8: Laserlijn en vervormingen waargenomen door Photonfocus-camera 38
Figuur 3-9: Laserlijn als Gausscurve 38
Figuur 3-10: Photonfocus-camera 39
Figuur 3-11: Bin picking-opstelling met pikrobot en lineaire geleiding met camera en laser . 40
Figuur 3-12: Cameraconfiguratie met (1) Photonfocus-camera, (2) lens en (3) bandpassfilter 41
Figuur 4-1: Effecten van verschillende waardes voor RelSamplingDisctance. Respectievelijk 0,3; 0,5 en 0,7 43
Figuur 4-2: Flowchart automatische parameterbepaling aan de hand van geneste for-lussen 47
Figuur 4-3: Goede match vergeleken met slechte match 48

Figuur 4-4: Meerdere matches gevonden op eenzelfde plaats 49
Figuur 4-5: Gevonden matches bij scores hoger als 1. Respectievelijk scorewaarde 4, 7, 10 en 14 51
Figuur 4-6: Gevonden match (wit) in een scene van meerdere brillen (groen) 52
Figuur 4-7: Flowchart extra controlefactor 53
Figuur 4-8: Voorbeeld van een geneste for-lus 54
Figuur 4-9: Geneste for-lussen met opsplitsing van stapgrootte 54
Figuur 4-10: Algoritme dat hoogste score en bijhorende parameters bepaalt 55
Figuur 4-11: Voorbeeld voor tupleberekeningen en bijhorende resultaten 56
Figuur 4-12: Flowchart gehele code, opgesplitst in 3 stappen 58
Figuur 4-13: STL-model 59
Figuur 4-14: Puntenwolk van het ingeladen STL-model 59
Figuur 4-15: Foutmeldingen in verband met geheugentekort van enerzijds Windows en anderzijds Halcon 60
Figuur 4-16: Scene enkel object voor en na uitvoeren threshold 61
Figuur 4-17: Goede (links) en foute (rechts) match. Rechterbril is 180° gedraaid. 62
Figuur 4-18: Nodige tijd in functie van elke parametercombinatie 63
Figuur 4-19: Behaalde score in functie van elke parametercombinatie 64
Figuur 4-20: Positieverschil in mm tussen de gevonden matches en de op voorhand bepaalde positie van het object 65
Figuur 4-21: Afwijkingen van de posities tussen gevonden match en object 66
Figuur 4-22: Gevonden match met de laagste score 66
Figuur 4-23: Gevonden match met de slechtste positie 67
Figuur 4-24: 5 Trainingsscenes 68
Figuur 4-25: Gevonden matches na eerste positiecontrole 69
Figuur 4-26: Trainingsscenes met bijhorende matches na strengere positiecontrole 69
Figuur 4-27: Beste sample distance voor elke trainingsscene 71
Figuur 4-28: Beste key point fraction voor elke trainingsscene 71
Figuur 4-29: Beste max overlap distance voor elke trainingsscene 72
Figuur 4-30: Aantal herhalingen die gebruikt worden voor de positiebepaling 73
Figuur 4-31: Testscenes en bijhorende matches bij gemiddelde parameters 75

Figuur 4-32: Testscenes en bijhorende matches bij gemiddelde parameters en verlaagde score 76

Figuur 4-33: Gevonden matches door het variëren van de parameters rond de gemiddelde waardes 78

Figuur 4-34: Gevonden matches door het variëren van de parameters rond de gemiddelde waardes en verlaagde minimale score 80

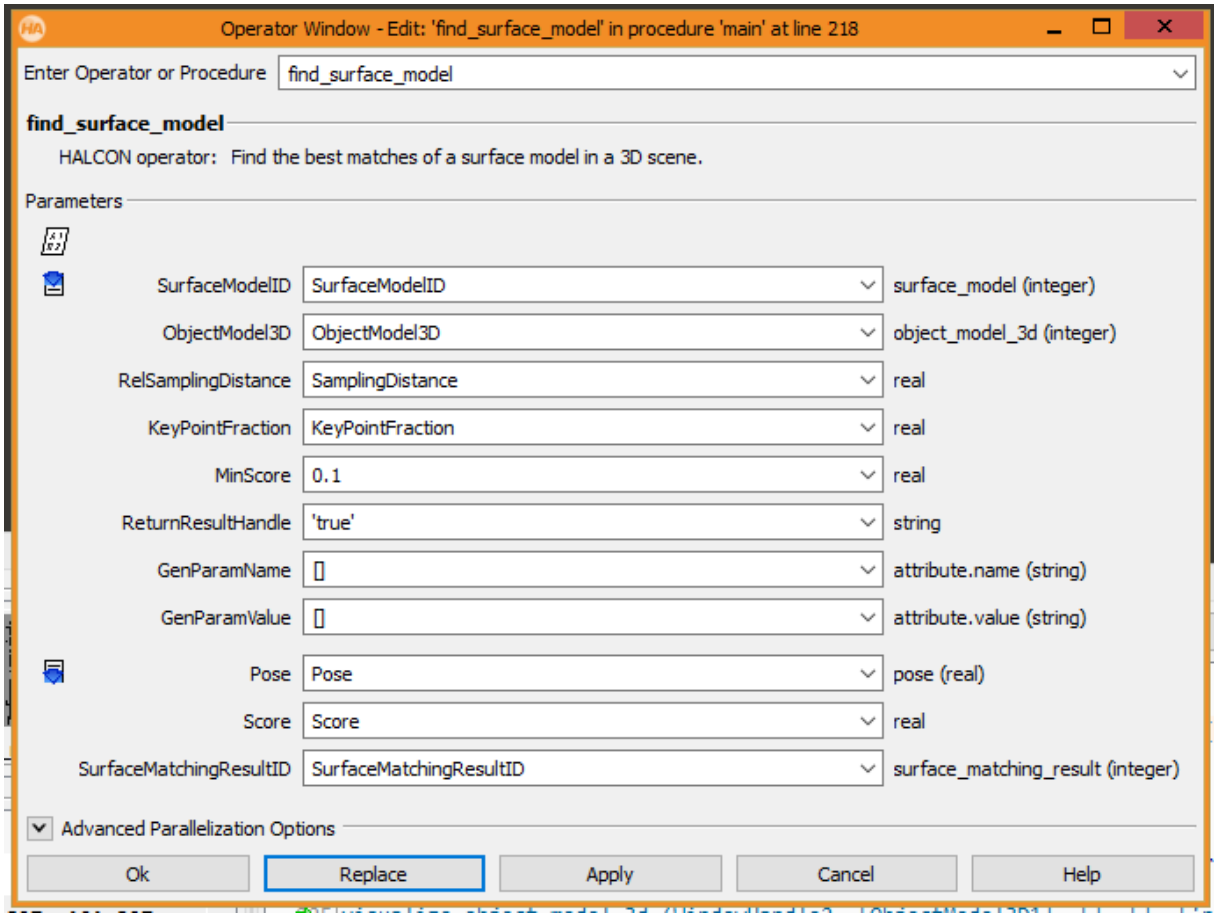
Verklarende woordenlijst

KPF: **K**ey **P**oint **F**raction, een basisparameter van de functie find_surface_model; zie paragraaf 4.2.2 pagina 44.

Match: een term die gebruikt wordt wanneer het visieprogramma Halcon een overeenkomst vindt tussen een object in de puntenwolk van een scene en de puntenwolk van een model.

Model: een 3D-model van het object dat in een (puntenwolk van de) scene gezocht gaat worden. Een model kan gemaakt worden uit een 3D-scan van een object of uit een CAD-tekening. In deze thesis worden de modellen waarvoor een match in de puntenwolk gevonden wordt, altijd in het wit afgebeeld in de figuren.

Operator: een bestaande functie van het programma Halcon. Hieraan moeten parameters meegegeven worden. Als resultaat krijgt men een output. Figuur 0-1 geeft een voorbeeld weer van een operator en de parameters die meegegeven moeten worden.



Figuur 0-1: Voorbeeld van een operator en de parameters die meegegeven moeten worden

Parameterset: (set van parameters, setting van parameters, parametercombinatie) verzameling van verschillende soorten parameters. Deze worden opgeslagen in tuples. In

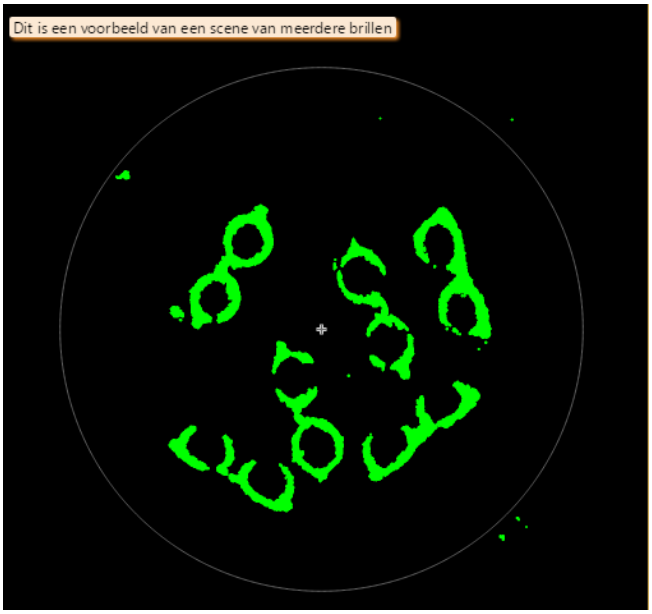
Figuur 0-2 is een eenvoudig voorbeeld weergegeven van een parameterset bestaande uit verschillende parameters.

```
1 Parameter1:=[0,1,2,3,4,5,6,7,8,9,10]
2 Parameter2:=[10,9,8,7,6,5,4,3,2,1,0]
3 Parameter3:=[7,7,1,2,3,5,3,4,2,7,7]
4
5 Parameterset0:=[Parameter1[0],Parameter2[0],Parameter3[0]]
6
```

Figuur 0-2: Voorbeeld parameterset

De tuple Parameterset0 zou er als volgt uitzien: [0, 10, 7]

Scene: een puntenwolk van een verzameling van objecten waarin matches gezocht worden. In deze thesis worden de scenes afgebeeld in de figuren, altijd in het groen weergegeven. Figuur 0-3 geeft een voorbeeld van een scene weer.



Figuur 0-3: Voorbeeld van een scene

SD: **S**ample **D**istance, een basisparameter van de functie find_surface_model; zie paragraaf 4.2.2 pagina 44.

Tuple: een geordende lijst van objecten. Het programma Halcon gebruikt een tuple om bepaalde variabelen in op te slaan. De eerste variabele die in een tuple zit heeft de positie 0. Tuples worden in deze thesis aangeduid met vierkante haken: […]. Onderstaand voorbeeld geeft een tuple weer van een reeks getallen. Het getal dat als eerste in deze tuple is weergegeven is het getal dat op de 0^e plaats staat in deze tuple. In dit geval is dat het getal 7.

Voorbeeld_tuple:=[7; 1; 2; 3; 5; 3; 4; 2]

Abstract

De industrie automatiseert almaar meer menselijke taken door automatische handelingen. Machinevisie speelt hierin een steeds belangrijkere rol. Bijna alle machinevisietoepassingen gebruiken parameters die typisch door de gebruiker ingesteld moeten worden. Dit vereist kennis en ervaring, en het is bovendien een tijdsintensieve procedure met vaak suboptimale resultaten.

Deze masterproef onderzoekt methodes voor automatische optimaleparameterbepaling van visiealgoritmes voor random bin picking. Een visiealgoritme zoekt objecten in de bin picking scene. Dit algoritme verwacht parameters waarvan de waarden afhankelijk zijn van de scene en het te zoeken object. Het automatisch bepalen van de parameterwaarden is een uitdaging in vele ingenieurstoepassingen, maar het lijkt zelden toegepast te zijn in machinevisie.

Deze masterproef voorziet een algoritme zodat deze parameters automatisch bepaald worden. Dit algoritme werkt aan de hand van trainingsscenes en testscenes. Het algoritme evalueert exhaustief alle parameterwaarden op de trainingsscenes aan de hand van geneste for-lussen. Indien er een correcte overeenkomst gevonden wordt tussen het model en een object in de scene, wordt deze parameterset opgeslagen. Deze setting wordt vervolgens gebruikt op testscenes waarin geen informatie van de objecten gekend is.

Door het automatisch bepalen van de parameters resulteert deze masterproef enerzijds in het verhogen van de nauwkeurigheid in het zoeken naar correcte matches van minimaal 80% en anderzijds in tijdsbesparing.

Abstract in English

In industry, tasks by human operators are more and more automated. Machine vision plays an increasingly important role in this transition. Almost all machine vision applications use parameters that are typically set by the user. This requires knowledge and experience, and it is a time-consuming procedure too, with often suboptimal results.

This master's thesis investigates methods for automatic optimal parameter determination of vision algorithms for random bin picking. A robot uses a 3D vision system to locate objects that are arbitrarily ("random") placed in a box ("bin") and eventually grabs the objects ("picking"). A vision algorithm detects objects in the bin picking scene. The algorithm requires some parameters, the values of whom depend on the scene and the objects to pick. Automatically determining parameter values is a challenge in many engineering applications, but it seems to be rarely applied in machine vision, or at least that there is only little literature available on this topic.

This master's thesis provides an algorithm that automatically determines the parameters, where this was performed manually in the past. This algorithm works on the basis of training scenes and test scenes. The algorithm exhaustively evaluates all parameter values for the training scenes using a series of nested for-loops. The parameter setting is saved if the position of the found object, which is determined in advance, is correct. This parameter setting is then used on the test scenes, for which no information regarding the objects' position is known, to verify if the found parameters work well on unseen data.

By automatically determining the parameters, this master's thesis results on the one hand in the increase of the accuracy in the search for correct matches to a minimal of 80% and on the other hand on the time saving aspect.

1 Inleiding

1.1 Situering

Deze masterproef werd voltooid in het onderzoekscentrum ACRO te Diepenbeek. ACRO staat voor Automatiseringscentrum Research en Opleiding. ACRO is een onderzoeksgroep van de KU Leuven [1].

ACRO doet onderzoek rond visie en robotica en doet verder aan dienstverlening en training rond Profibus & Profinet en geïntegreerde automatisering.

Deze masterproef situeert zich in het domein Vision & Robotics. Machine Vision wordt veelvuldig ingezet in industriële toepassingen ter vervanging van de menselijke visuele waarneming. Een visiesysteem bepaalt snel, objectief en nauwkeurig objecteigenschappen. ACRO's bekendste voorbeeld hiervan is de automatische fruitplukmachine, waarbij een robot appels plukt op basis van camerabeelden [2].

Een typisch voorbeeld van het gecombineerd gebruik van computervisie en robotica is random bin picking. ACRO heeft een bin picking-opstelling die gebruik maakt van een Epson C3-A601S-robot en een visiesysteem bestaande uit een laser en een Photonfocus-camera die samen op een lineaire geleiding bevestigd zijn.

Door middel van de computervisiesoftware Halcon worden producten herkend die willekeurig (random) in een bak (bin) liggen. Verder wordt hun 3D-positie en oriëntatie bepaald, waarna de robot wordt aangestuurd om het gelokaliseerde product te grijpen (picking). ACRO heeft in het verleden de haalbaarheid van bin picking aangetoond, en heeft een werkende bin picking-setup die ter beschikking gesteld werd voor de masterproef.

1.2 Probleemstelling

Bij random bin picking kunnen allerlei soorten producten in de bak liggen. Deze masterproef focust op 3D-geprinte onderdelen zoals brilmonturen en schoenzolen. De brilmonturen kunnen verschillen wat betreft design en elk design kan daarenboven ook verschillende afmetingen hebben, omdat de monturen op maat gemaakt worden. Ook de schoenzolen worden op maat gemaakt en variëren in lengte, breedte en hoogte.

Figuur 1-1: 3D-geprinte brilmonturen die als model kunnen dienen

Figuur 1-2: 3D-geprinte zolen die als model kunnen dienen

Het doel van de bin picking-opstelling is om een vooraf gedefinieerd object op te pikken uit een verzameling van meerdere objecten, gebruik makend van een visiesysteem in combinatie met een robot en grijpsysteem. De componenten van het visiesysteem kunnen variëren. Zo is er bij ACRO een opstelling die gebruik maakt van een Photonfocus-camera en een laser om een beeld te verkrijgen. Deze opstelling maakt gebruik van het sheet-of-lightprincipe. Een andere opstelling die bij ACRO gebruikt wordt is de Ensenso-camera. Deze opstelling wordt in deze paragraaf niet verder besproken, maar de werking ervan wordt uitgelegd in paragraaf 3.2.

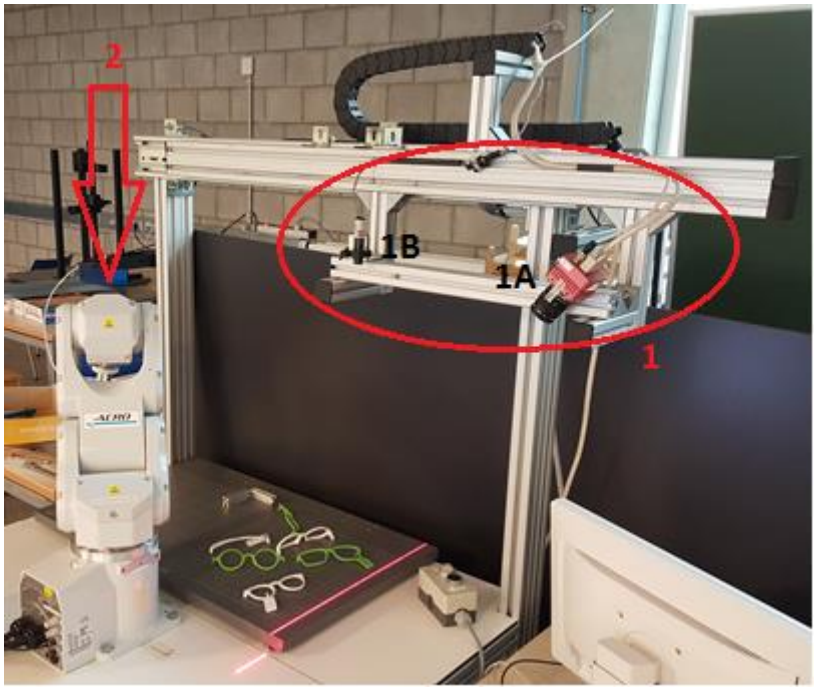
Vooraleer het mogelijk is om de robot een bepaald object op te laten pikken, is er een model nodig van het te zoeken object. Dit model kan op twee manieren verkregen worden. Beide mogelijkheden zijn in het proces naar het uiteindelijke algoritme gebruikt en worden daarom hieronder uitgelegd.

1. Een eerste manier bestaat erin het object “in te scannen” en daar een model van te maken. Hierbij wordt het te vinden object op het werkblad geplaatst. Van dit fysiek object wordt er dan een 3D-beeld/puntenwolk gegenereerd door middel van het sheet-of-lightprincipe en het beeldverwerkingsprogramma. Dit komt tot stand door een laserlijn samen met een camera lineair boven het werkblad te verplaatsen. De Photonfocus-camera neemt beelden van het werkblad met de objecten. De gebruikte software analyseert de laserlijn en de onderbrekingen, die door de hoogteverschillen

van het object ontstaan in de laserlijn. Deze onderbrekingen maken het mogelijk om samen met de nodige bewerkingen in Halcon een 3D-beeld te creëren. Met het softwareprogramma Halcon kunnen er dan verschillende bewerkingen worden uitgevoerd, die het uiteindelijke 3D-model maken. Een 3D-model is een puntenwolk van het object.

2. Een andere mogelijkheid voor het maken van het hierboven beschreven 3D-model, is het inladen van een CAD-model om vervolgens de juiste onderdelen te vinden in de reeks van objecten.

Uiteindelijk is het de bedoeling dat de pikrobot de juiste brilmonturen of schoenzolen zal opnemen uit een reeks van verschillende brilmonturen/schoenzolen. Hierbij moet er ook rekening gehouden worden met de grijpbaarheid van elk onderdeel (zonder botsingen tussen robotgrijper en andere objecten of de bak zelf). Zo is het bijvoorbeeld veel gemakkelijker om het juiste onderdeel te nemen dat het hoogste is gelegen.



Figuur 1-3: Opstelling met lineaire geleiding (1), Photonfocus-camera (1A), laser (1B) en pikrobot (2)

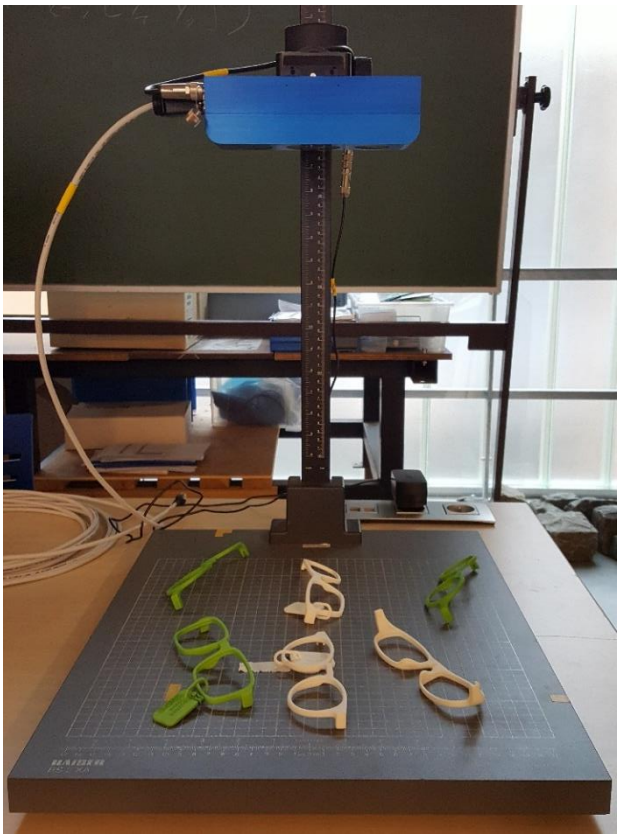
Bij het programma dat gebruikt werd bij de start van deze thesis en dat geschreven is in Halcon, worden er voor elk soort object bepaalde parameters meegeven. Deze parameters zorgen ervoor dat het juiste onderdeel gevonden wordt in de puntenwolk, die door Halcon wordt samengesteld. Zolang het object niet van type verandert, is er geen probleem. Maar van zodra er een volledig nieuw object gevonden moet worden, moeten bepaalde parameters ook veranderen, opdat het juiste object zo nauwkeurig mogelijk gevonden zou worden. Dit komt omdat sommige objecten veel details hebben, andere weinig, sommige objecten groot zijn en andere klein. Deze parameterbepaling gebeurt nu nog handmatig wat veel tijd en kennis vergt. Het bepalen van de optimale parameters voor de computervisiealgoritmes gebeurt idealiter op een volautomatische manier.

1.3 Doelstellingen

De hoofddoelstelling van deze masterproef is het automatisch genereren van de parameters voor het terugvinden van de juiste 3D-geprinte onderdelen.

Zoals eerder vermeld zijn er twee opties om een model te maken van het te zoeken onderdeel. Enerzijds is het mogelijk om van het werkelijke onderdeel een scan te maken en aan de hand van Halcon een 3D-model van te maken. Anderzijds is het mogelijk om een bestaand model, zoals CAD-files, in Halcon in te laden van het desbetreffende onderdeel.

Een ander belangrijk onderzoeksaspect is de invloed van de camera op de performantie van de bin picking en op de computervisieparameters. ACRO heeft namelijk meerdere sensoren die compatibel zijn met het programma Halcon en die voor deze opdracht gebruikt kunnen worden. Zoals eerder vermeld maakt de huidige opstelling gebruik van een camera van het merk Ensenso. Figuur 1-4 geeft deze opstelling weer. De andere sensoren die er ter beschikking zijn, zijn de Photonfocus en de Kinect V2. Deze drie sensoren kunnen vervolgens vergeleken worden met elkaar. Zo kan er een onderscheid gemaakt worden op basis van snelheid en nauwkeurigheid tussen deze 3 sensoren. Deze vergelijking is in deze thesis niet gemaakt kunnen worden, omdat de opstelling gebaseerd op de Photonfocus-camera defect was en er nog geen mogelijkheid was om de Kinect V2 met het visieprogramma Halcon te verbinden.



Figuur 1-4: Opstelling met Ensenso-sensor

1.4 Materiaal en methode

Om deze thesis tot een goed einde te brengen is het zeer belangrijk om de gebruikte software te begrijpen. In dit geval is dat MVTec Halcon. Halcon is een uitgebreide software voor machinevisie met een geïntegreerde ontwikkelomgeving. Dankzij de flexibele architectuur van coderen, zijn er snelle ontwikkelingen mogelijk. De manier van coderen werkt aan de hand van operators. Dit zijn voorgeprogrammeerde functies, waar als input en output parameters mee worden gegeven. Deze kunnen iconische parameters zijn (afbeeldingen, grafische beelden) of controle parameters, in de vorm van “string”, “integers”, …

Daarnaast is het belangrijk om de hardware, waarmee de software Halcon moet kunnen werken, te begrijpen. De hardware bestaat in dit geval uit een sensor en een lichtopstelling die zo beelden kan inlezen. Aan de hand van de uitgewerkte voorbeelden die in Halcon geïntegreerd zijn, is het mogelijk om per gebruiksgebied van machinevisie een beeld te krijgen van het gebruik van de operators.

Zoals reeds eerder vermeld bestaat de opstelling, waarmee er een 3D-model gemaakt wordt van een fysiek onderdeel, uit een lineaire geleiding waarop een laser en een Photonfocus-camera staat geïnstalleerd. Met behulp van het sheet-of-lightprincipe (zie paragraaf 3.3.2 pagina 36) wordt hier een scan gemaakt van het oppervlak dat zich hieronder bevindt en wordt er een 3D-opname gemaakt van de onderdelen die hieronder geplaatst worden.

Vervolgens is het belangrijk om de bestaande code te begrijpen, zodat er verder kan gegaan worden met het automatisch instellen van de parameters. Aangezien de parameters allemaal verschillen per type 3D-geprint object, is het noodzakelijk te weten welk effect elke parameter heeft op het vinden van het juiste onderdeel.

Hierna moet er naar een methode gezocht worden, die de parameters zo goed mogelijk en automatisch kan laten aanpassen. Automatische parameterbepaling is op zichzelf niet zo nieuw [3], [4], [5], [6]. Het vernieuwende aan deze opdracht is dat parameterbepaling nog niet veel besproken/onderzocht is op het gebied van machinevisie. Aan de hand van onderzoeken naar het automatisch bepalen van parameters, is het mogelijk om dit te implementeren in visiecontrole. Hieruit zal dan blijken welke methode het beste past bij het behalen van de doelstellingen van deze masterproef.

2 Literatuurstudie

2.1 Inleiding

Om een geschikte techniek/methode te vinden voor het algoritme dat gemaakt moet worden, moest er een literatuurstudie uitgevoerd worden om een beeld te krijgen van de reeds bestaande methodes. Aan de hand van deze methodes is het dan mogelijk om voor het probleem dat deze thesis behandelt de beste methode te kiezen. Allereerst zijn de gebruikshandleidingen van het programma Halcon bestudeerd om een beeld te krijgen wat de mogelijkheden zijn met dit programma. Daarna wordt er beroep gedaan op studies die automatisch parameters bepalen om kennis op te doen over de mogelijkheden.

2.2 Halcon

Deze sectie geeft een algemeen beeld over het programma en gaat daarna dieper in op de functies/methodes die in deze thesis gebruikt worden.

Halcon HDevelop is een softwareprogramma voor het ontwikkelen van machinevisietoepassingen. Het maakt gebruik van een interactieve programmeeromgeving voor het ontwikkelen en testen van deze toepassingen. Dankzij de ingebouwde bibliotheek is Halcon een visiesysteem dat geschikt is voor productontwikkeling, onderzoek en onderwijs. Om beeldanalysetoepassingen te ontwikkelen met behulp van Halcon zijn er vier fundamentele manieren die gebruikt kunnen worden [7]:

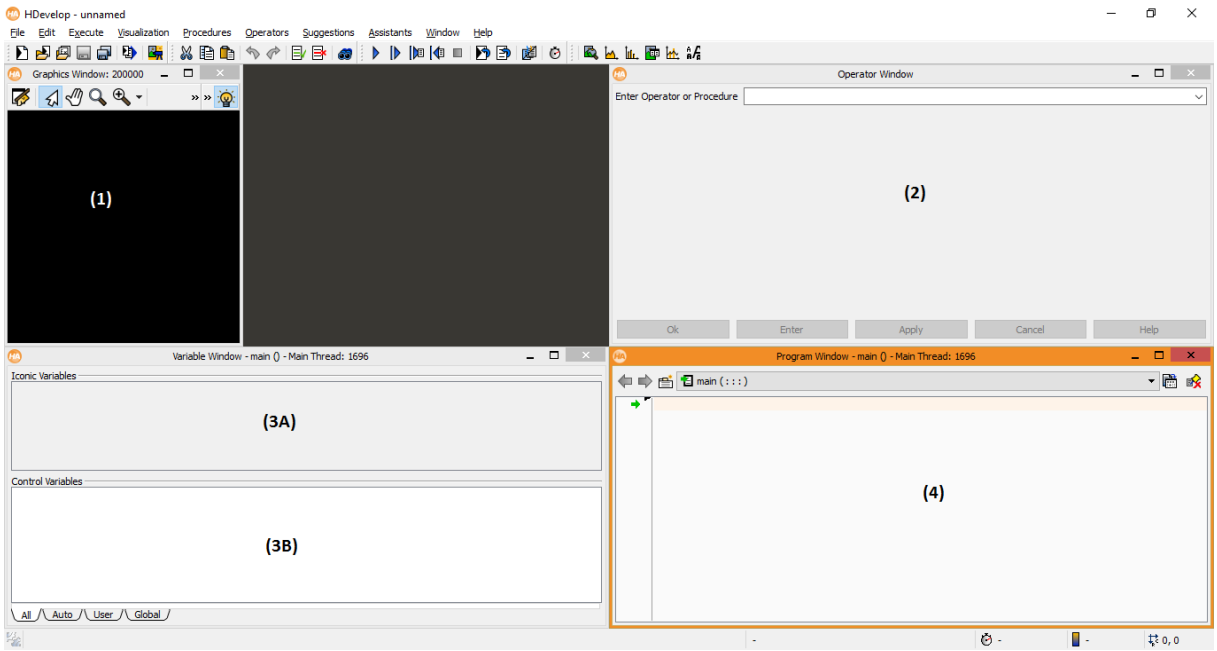
- Rapid prototyping in de interactieve omgeving van Halcon HDevelop: Halcon kan gebruikt worden om de optimale operators en parameters te vinden om beeldanalyses uit te voeren. Hierna kan er dan een applicatie gemaakt worden aan de hand van verschillende programmeertalen, bijvoorbeeld C, C++, C#, Visual Basic, .NET en Delphi.
- Ontwikkeling van een applicatie die in het programma Halcon zelf werkt: in Halcon is het mogelijk om een volledige beeldanalyseapplicatie te maken. De voorbeeldprogramma's die in Halcon zijn inbegrepen kunnen gebruikt worden als basisblokken voor een eigen toepassing.
- Uitvoeren van programma's in HDevelop: naast applicaties ontwikkelen vertrekkende vanuit Halcon, is het ook mogelijk om reeds bestaande applicaties in bovenstaande programmeertalen te importeren en te gebruiken in Halcon.
- Exporteren van applicaties in verschillende programmeertalen.

Deze thesis is gebaseerd op een combinatie van zowel het eerste als het tweede punt hierboven beschreven. De code is geschreven in Halcon zelf en runt ook in Halcon. Er was geen nood om deze toepassing naar een andere programmeertaal om te zetten. Naast de hierboven beschreven manieren om te werk te gaan met Halcon, biedt Halcon ook nog tal van voordelen die het gebruiksgemak vergroten bij het ontwikkelen van een applicatie. Hieronder zijn de meest toepasselijke opties voor deze thesis opgesomd.

- Dankzij de gebruikersinterface kunnen operators en visuele objecten onmiddellijk worden geselecteerd, geanalyseerd en gewijzigd binnen dezelfde werkomgeving.
- Halcon adviseert op basis van de gebruikte operators, welke ervoor of erna gebruikt kunnen worden om de applicatie te optimaliseren. Bovendien worden alle operators per onderwerp gesorteerd in een overzichtelijke lijst om zo sneller de juiste operator te vinden.
- De resultaten van de uitgevoerde operators worden onmiddellijk weergegeven. Aanpassingen van parameters of operators kunnen zo direct met elkaar vergeleken worden, zonder dat het programma zelf moet wijzigen.
- Variabelen worden opgesplitst in twee soorten. Enerzijds zijn er de iconische variabelen, die afbeeldingen en modellen bijhoudt. Anderzijds zijn er de controlevariabelen die getallen en andere variabelen bijhouden.

Om een beter beeld te verkrijgen van bovenstaande punten, geeft Figuur 2-1 een screenshot weer van de lay-out van het programma met:

(1) Het grafisch venster waarin afbeeldingen of andere visuele objecten in worden weergegeven.
(2) Operatorvenster waarin alle operators opgezocht kunnen worden en bijhorende parameters ingevoerd kunnen worden.
(3) (3A) Iconische variabelen die afbeeldingen en visuele objecten bijhouden. (3B) controlevariabelen waarin alle andere variabelen zoals integers, strings, … opgeslagen worden.
(4) Het programmeervenster waarin de eigenlijke code wordt geschreven.



Figuur 2-1: Lay-out Halcon met (1) grafisch venster, (2) operatorvenster, (3A) iconische variabelen, (3B) controlevariabelen en (4) programmeervenster

Machinevisieprogramma's kunnen in diverse toepassingsgebieden gebruikt worden om handelingen te automatiseren of controleren. De toepassingsgebieden die het meest gebruik maken van machinevisieprogramma's zijn hieronder opgesomd:

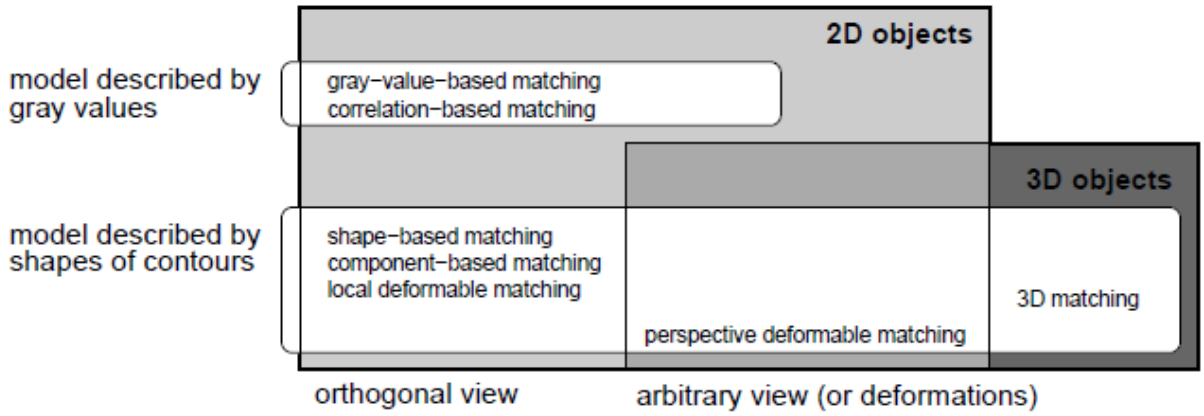
- Positiebepaling (2D & 3D)
- Objectherkenning (2D & 3D)
- Oppervlakte-inspectie
- Robotvisie
- Meten en vergelijken

Deze thesis situeert zich in het vakgebied van objectherkenning (2D & 3D). Bovenstaande tekst is een algemene beschrijving van het programma Halcon. Hieronder wordt een meer gedetailleerd beeld gegeven van de functies en toepassingsgebieden die bij deze thesis horen.

Allereerst wordt er een antwoord op de vraag gegeven wat matching nu eigenlijk juist is. Matching is een methode in het programma Halcon om de positie van objecten in scenes te bepalen. Aan de hand van de eisen van de toepassing waarvoor het matchen gebruikt wordt, zijn er meerdere manieren mogelijk om de positie te bepalen. De verschillende manieren bestaan uit een reeks van operators en bijhorende parameters. Het basisidee van matching berust op een object waarvan een model gemaakt wordt om dit model vervolgens te gebruiken om naar overeenkomsten te zoeken in scenes. Bij het zoeken wordt er naar overeenkomsten gezocht tussen het model en de objecten in de scene. De verschillende mogelijkheden voor matching verschillen in de opbouw van de scene waarvan het model gemaakt wordt. Sommige matchingmethodes gebruiken het verschil in grijswaarden tussen het object en de omgeving om een model te maken. Andere methodes gebruiken de vorm van het object om het model te maken. Het resultaat van matching is altijd informatie over de positie en oriëntatie van het gevonden object in de doorzochte scene. De verschillende soorten matching zijn hieronder per methode opgesomd [8]:

- Model gemaakt aan de hand van grijswaarden:
    - Correlation-based matching
- Model gemaakt aan de hand van vormen en contouren:
    - Shape-based matching
    - Component-based matching
    - Local deformable matching
    - Perspective deformable matching

Aan de hand van de toepassing wordt er een keuze gemaakt van de matchingmethode. Figuur 2-2 [8] geeft een overzicht weer dat aangeeft in welke situatie voor welke matchingmethode gekozen moet worden. Deze thesis situeert zich in het gebied van de 3D-objecten en orthogonaal zicht. De matchingmethode shape-based matching is hiervoor de beste manier gebleken.



Figuur 2-2: Matchingmethodes in functie van de toepassingen

Ondanks de verschillende soorten matching blijft de hoofdprocedure hetzelfde en deze bestaat uit volgende stappen:

(1) Creëer een model van het object
(2) Vind het model in afbeeldingen
(3) Verwijder het model uit het geheugen

2.3 Verschillende methodes/technieken om automatisch parameters te bepalen

In deze paragraaf wordt de verworven kennis beschreven van artikels en papers die handelden in het automatisch bepalen van parameters. Hoewel er geen concrete literatuur beschikbaar was over het automatisch bepalen van parameters in machinevisie, wordt in dit hoofdstuk een beknopt overzicht meegegeven. De gebruikte papers en artikels behandelen hetzelfde probleem als deze thesis, met het verschil dat de toepassingsgebieden verschillend zijn.

De voornaamste reden voor het automatiseren van parametersinstellingen is in alle gelezen literatuur grotendeels identiek. Om parameters manueel in te stellen in software, heeft de gebruiker kennis nodig van deze software en van het proces waarvoor dit gebruikt wordt. Deze verplichte kennis wil men nu vervangen door algoritmes die parameters zelf kunnen bepalen. Hiervoor kunnen verschillende methodes/technieken gebruikt worden:

- Geneste for-lussen: een reeks van for-lussen, waarvan eerst de laatste for-lus doorlopen wordt totdat deze zijn maximale waarde heeft bereikt. Daarna wordt de waarde van de bovenliggende for-lus met één stapgrootte vergroot en doorloopt de laatste for-lus opnieuw het gehele bereik. Dit proces wordt herhaald totdat de bovenste for-lus zijn gehele bereik heeft doorlopen. Op deze manier worden voor x-aantal parameters alle mogelijke combinaties overlopen.
- Bayesian analysis [4]: deze methode is gebaseerd op de waarschijnlijkheid dat iets gaat voorkomen. De huidige kennis van de modelparameters wordt uitgedrukt door een kansverdeling van de parameters. Wanneer er nieuwe informatie beschikbaar

komt over de modelparameters, wordt er gekeken naar de evenredigheid van de nieuwe data vergeleken met de huidige modelparameters. Op deze manier kan men een voorspelling maken voor de parameters.

- Evolutionary algorithms [4]: algoritme dat gebruik maakt van artificiële of kunstmatige intelligentie om oplossingen te vinden voor optimalisatie- en zoekproblemen.
- Evaluatiefunctie en foutschatting [3]: hierbij wordt er een evaluatie gemaakt van de verkregen resultaten op basis van bepalende factoren en wordt er een schatting gemaakt van de kans dat er een fout resultaat gaat voorkomen. Deze factoren kunnen verschillen naargelang de toepassing.

Voor men een algoritme kan maken voor parameterbepaling is het noodzakelijk om te weten welke parameters er ingesteld moeten worden en wat het bereik is van deze parameters.

Een van de belangrijkste gegevens die nodig zijn om parameters in machinevisie automatisch te generen is een goede scene van de objecten. Dit kan enkel als de cameraopstelling goed gekalibreerd is. De belangrijkste parameter die hiervoor verantwoordelijk is bij stereocamera's, is de disparity search range [5]. Deze bepaald in welk bereik op de z-as de objecten zijn gelokaliseerd. De z-as is de as die verantwoordelijk is voor de hoogte in een scene/afbeelding. Voor het laagstgelegen en hoogstgelegen object wordt er dan een parameterwaarde (d_{min} en d_{max}) toegewezen waartussen alle objecten gelegen zijn. Deze parameterwaarden worden respectievelijk N-keer met factor α vergroot en verkleind totdat er punten buiten de selectie vallen. Wanneer dit het geval is worden de parameterwaardes $d_{min} + (N-1) * \alpha$ en $d_{max} - (N-1) * \alpha$ gebruikt als de instelwaardes voor de disparity range.

Zodra de methode om de parameters in te stellen bekend is, is het belangrijk voor elk algoritme te bepalen wat de invloed is van de parameters die men automatisch wilt bepalen. Deze informatie valt niet onder dit hoofdstuk aangezien dit voor elk probleem anders is en deze kennis niet verworven is uit een artikel of paper. Deze informatie wordt beschreven in paragraaf 4.2.1 pagina 43 en 4.2.2 pagina 44.

Als geweten is welke parameters er gebruikt moeten worden en wat de functie hiervan is, is het nodig een evaluatiemethode te creëren die het mogelijk maakt om parametersets te beoordelen [3]. Afhankelijk van de outputs van een functie zijn er meerdere mogelijkheden om een evaluatiemethode op te stellen. Hieronder volgen 3 manieren die voor deze thesis toepasselijk zijn:

(1) Score: dit is een waarde die berekend wordt door het visieprogramma. Afhankelijk van het gekozen scoretype wordt deze waarde bepaald. Het standaard scoretype geeft een waarde tussen 0 en 1, dat overeenkomt met de hoeveelheid overeenkomstige punten uit de puntenwolk van het model en de puntenwolk in de

scene. Stel dat het model uit 100 punten bestaat en er wordt een object gevonden in de scene waarvan 50 punten overeenkomen, zal de score 0,5 bedragen.

(2) Tijd: uitgedrukt in seconden. De rekentijd die het programma nodig heeft om voor elke parameterset naar matches te zoeken in de scene.

(3) Positie: dit zijn de X-, Y- en Z-coördinaten van de gevonden match. Op basis van deze coördinaten is het mogelijk om te controleren of de match al dan niet op de juiste positie is gelegen door deze te vergelijken met de op voorhand bepaalde positie van het object in de scene.

Voor elk van deze outputs is het mogelijk om een evaluatie te maken. Een combinatie van meerdere is ook mogelijk. Daarnaast is het noodzakelijk dat er rekening gehouden moet worden dat niet elk positief resultaat voor deze parameters, ook effectief een goede parameterset is. Hiervoor is het belangrijk dat men een schatting maakt van betrouwbaarheid van de evaluatiemethode. Dit kan op volgende manier:

$$A = \frac{Aantal\ gevonden\ goede\ matches}{aantal\ gevonden\ matches} \quad (2.1)$$

en

$$B = \frac{Aantal\ foute\ gevonden\ matches}{Aantal\ objecten\ in\ de\ scene} \quad (2.2)$$

Met de eigenlijke schattingsfactor:

$$E = \lambda * A - (1 - \lambda) * B \quad (2.3)$$

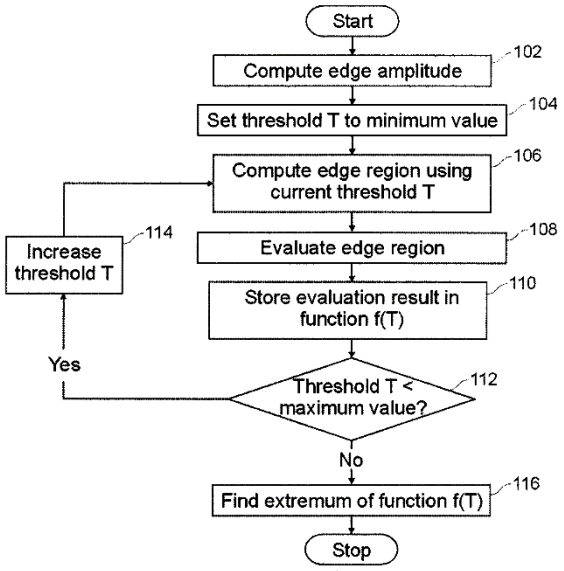
Voor deze thesis zal er geen automatische foutschatting gemaakt kunnen worden, aangezien het enkel visueel mogelijk is om te beoordelen of een match al dan niet goed is.

In de volgende sectie wordt een methode gebruikt om de parameters automatisch te bepalen aan de hand van één enkel beeld. De gebruikte literatuur is de beschrijving van een patent van het visieprogramma Halcon [9]. Daar waar vorige methodes aan de hand van een terugwerkende kracht werkten op basis van de verkregen resultaten, wordt er in deze methode gebruik gemaakt van één afbeelding om op basis hiervan de parameters te bepalen die het beste bij deze afbeelding horen. Hieronder volgt eerst een opsomming van de parameters die gebruikt worden en vervolgens volgt bij elke parameter de uitleg hoe er hiermee te werk wordt gegaan.

- Contrast van een object in een scene
- Minimale grootte van objecten in de scene
- Reduceren van het aantal punten van het model
- Minimale contrast van de afbeelding
- Discretisatie van de stapgrootte

Contrast van een object in een scene:

Aan de hand van het contrast van een object worden de optimale thresholdwaarden bepaald. Een threshold zorgt ervoor dat enkel de belangrijke objecten worden opgenomen in de puntenwolk. Dit gebeurt door de belangrijke randen van het object op te nemen in de beschrijving van het model en de randen die ontstaan door ruis niet op te nemen in de beschrijving. Eerst worden alle mogelijke thresholdwaarden overlopen, die gecontroleerd worden door een beoordelingscriterium. De threshold die overeenkomt met de uiterste waarde waarbij de belangrijkste punten in puntenwolk zitten en de ruis erbuiten valt, wordt als optimale threshold gebruikt. Aan de hand van deze waarde wordt nu de onderste en bovenste waarde van de thresholdparameter bepaald. Doordat dit voor elke oriëntatie (X, Y, Z) van een afbeelding apart kan gebeuren, wordt er van elk 3D-probleem, 3 1D-problemen gemaakt. Figuur 2-3 [9] geeft de flowchart weer voor het bepalen van de optimale thresholdwaarde.



Figuur 2-3: Flowchart voor het bepalen van de optimale thresholdwaarde

Minimale grootte van objecten in de scene:

Op deze manier wordt vermeden dat te kleine onderdelen niet opgenomen worden in de puntenwolk voor het model. Kleine onderdelen zorgen namelijk sneller voor ruis bij het zoeken naar matches. De minimale grootte bepalen ze aan de hand van een nieuwe threshold. Deze threshold moet ervoor zorgen dat een zeker percentage (bijvoorbeeld 5%) van de randpixels uit het model worden verwijderd. Het kleinste onderdeel dat na deze threshold nog aanwezig is, wordt gelijkgesteld aan de minimale grootte van de objecten.

Reduceren van het aantal punten van het model:

Hoe meer punten een puntenwolk bevat hoe langer het duurt om hier bewerkingen op uit te voeren. Door het aantal punten te reduceren, wordt er naar een hogere snelheid gestreefd. Er zou geen afname zijn van de robuustheid, noch van de nauwkeurigheid van het model.

Het gaat vooral over grote objecten waar een model van gemaakt wordt. Het optimaal aantal punten dat het model moet bevatten, wordt bepaald aan de hand van verschillende voorbeeldobjecten van verschillende grootte en vorm. De grootte van het object wordt dan herkend en op basis hiervan wordt het aantal punten ingesteld.

Minimale contrast van de afbeelding:

Men kan de ruis onderdrukken door de contrastwaarde van de afbeelding te verkleinen. Dit gebeurt aan de hand van volgende methode: eerst wordt aan de hand van bestaande methodes bepaald wat de standaarddeviatie is voor de ruis in de afbeelding. Eens deze bepaald is worden alle randpixels die een randwaarde hebben die lager ligt als de standaarddeviatie verwijderd. Men kan deze standaarddeviatie ook vermenigvuldigen met een factor om de kans te verlagen dat randen veroorzaakt door ruis foutief worden geaccepteerd.

Discretisatie van de stapgrootte:

Methode beschreven voor het automatisch bepalen van de optimale discretisatie stapgrootte voor rotatie, het schalen en andere transformaties van het model. Dit wordt bepaald aan de hand van de hoekstapgrootte. Deze bepaald in hoeveel verschillende posities het object weergegeven kan worden. Stel dat de hoekstapgrootte ingesteld is op 10, dan zal het object enkel in posities weergegeven kunnen worden waarbij de hoek 36° (=360°/10) verschilt. Als deze te groot is ingesteld, zorgt dit voor een verhoging van de tijd en het nodige geheugen. Zodra de stapgrootte te laag wordt ingesteld, zorgt dit voor een vermindering van de robuustheid voor herkenning.

2.4 Conclusie

Van al de mogelijke methodes die hierboven zijn beschreven, is ervoor gekozen om de methode met de geneste for-lussen te gebruiken, in combinatie met de evaluatiefunctie. De methode aan de hand van de geneste for-lussen is rechttoe rechtaan om de parameters te variëren. Deze methode in combinatie met de evaluatiefunctie zorgt ervoor dat alle parameters overlopen worden en geëvalueerd, waarna de beste set van parameters bepaald kan worden. De andere methodes vergen meer kennis van machinevisie. Het gebruiksgemak van de for-lussen en evaluatiemethode, is tevens een reden waarom er in deze thesis mee wordt gewerkt.

3 3D-visietechnologie

3.1 Inleiding

In dit hoofdstuk wordt een algemeen beeld geschetst van visietechnologie. De verschillende soorten technieken en bijhorende camera's worden beschreven. Voor deze thesis worden enkel de technieken en camera's die beschikbaar zijn bij ACRO beschreven. Dit zijn camera's die werken aan de hand van stereovisie en sheet-of-light. Voor elk van deze technieken wordt een algemene, theoretische achtergrond, het werkingsprincipe en de soorten camera's gegeven.

3.2 Stereovisie

3.2.1 Algemeen

De gebruikte 3D-visietechnologie maakt deel uit van machinevisie. Dit is gebaseerd op de informatie van digitale beelden afkomstig van camera's of sensoren. Afhankelijk van de toepassing wordt een cameraprincipe gekozen. Visiesystemen worden voornamelijk gebruikt bij het ontwerpen, inspecteren en opsporen van fouten van objecten. De mens beschikt over het meest geavanceerde en krachtigste visiesysteem voor het winnen van informatie uit de omgeving. Het visiesysteem van de mens is beter bekend als het menselijk zicht dat tot stand komt door de ogen. Soortgelijke systemen worden ontworpen voor robottoepassingen in machinevisie en wordt stereovisie genoemd.

Een stereovisiesysteem is ontworpen om 3D-informatie uit digitale beelden te halen. Enerzijds wordt de informatie die men verkrijgt uit deze digitale beelden gebruikt om objecten te onderzoeken en hun positie te achterhalen in een scene. Anderzijds wordt de informatie gebruikt om objecten te herkennen in verschillende omgevingen.

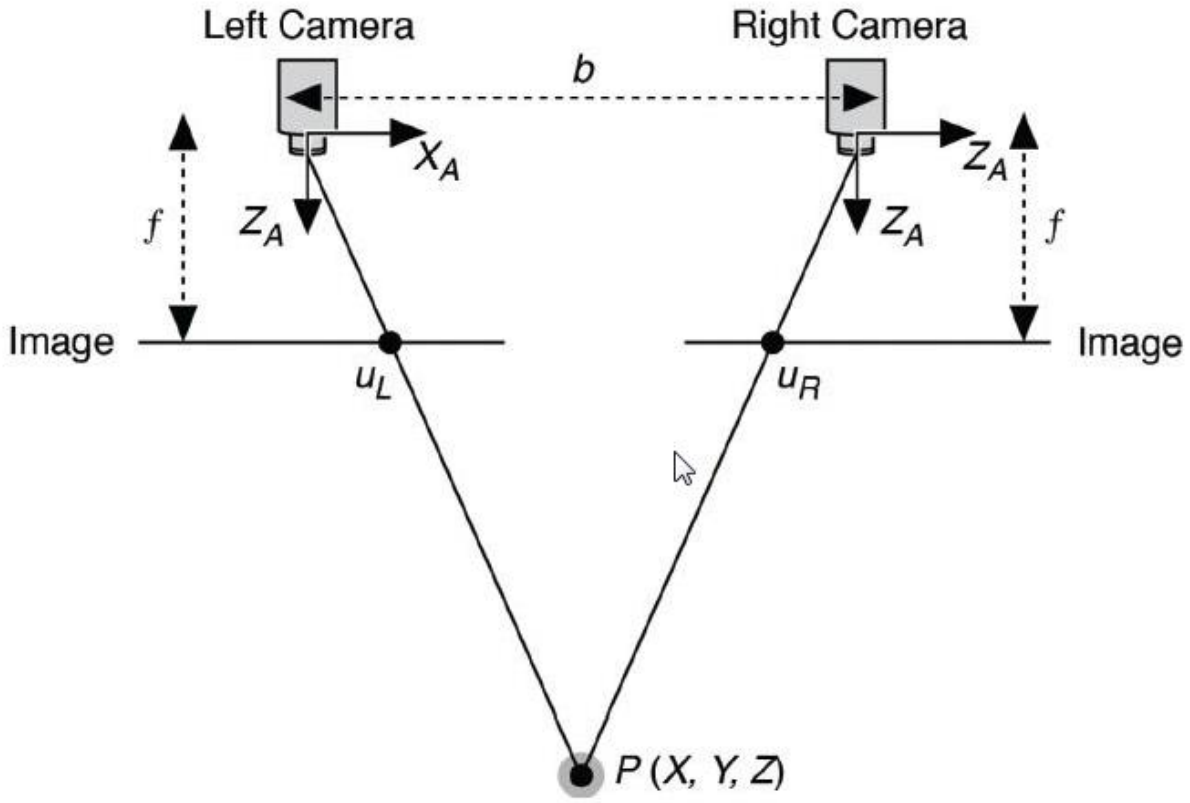
Daar waar de mens twee ogen heeft, heeft een stereovisiesysteem twee camera's of sensoren die op een gekende afstand van elkaar zijn gepositioneerd en op hetzelfde ogenblik foto's van dezelfde omgeving nemen. Met behulp van de geometrie van de gebruikte camera's kunnen er algoritmes worden toegepast en kan de geometrie van de omgeving worden bepaald.

Betrouwbaarheid en doeltreffendheid zijn de grootste voordelen van stereovisiesystemen. Doordat stereovisie gebruik maakt van passieve sensoren worden deze niet beïnvloed door de omgeving en is het een ideale technologie voor gebruik in verschillende visiedoeleinden zoals het detecteren van objecten.

3.2.2 Werkingsprincipe

In Figuur 3-1 [10] is de schematische werking uitgelegd van een stereovisiesysteem. Deze figuur stelt een vereenvoudigde stereovisieopstelling voor waarbij beide camera's perfect

zijn uitgelijnd ten opzichte van elkaar. Ze staan evenwijdig aan elkaar en hebben exact dezelfde brandpuntsafstand.



Figuur 3-1: Werkingsprincipe stereovisie

De variabelen van Figuur 3-1 hebben volgende betekenis:

- b = de basislijn of de afstand tussen de 2 camera's
- f = de brandpuntsafstand van de camera
- X_A = de X-as van de camera
- Z_A = de optische as van de camera
- P = Het punt in de echte wereld gedefinieerd door de coördinaten X, Y en Z
- U_L = de projectie van punt P in een beeld verkregen door de linker camera
- U_R = de projectie van punt P in een beeld verkregen door de rechter camera

Aangezien de twee camera's op afstand "b" van elkaar staan, zien beide camera's hetzelfde punt P maar vanuit een ander standpunt.

De X-coördinaten van de punten U_L en U_R worden berekend aan de hand van formule 3.1 en 3.2:

$$U_L = f \cdot \frac{X}{Z} \quad (3.1)$$

en

$$U_R = f \cdot \frac{(X - b)}{Z} \quad (3.2)$$

De afstand tussen deze twee camera's wordt gezien als ongelijkheid. De grootte van deze ongelijkheid kan men gebruiken om de diepte, de afstand tussen punt P in de echte wereld en de camera's, te berekenen.

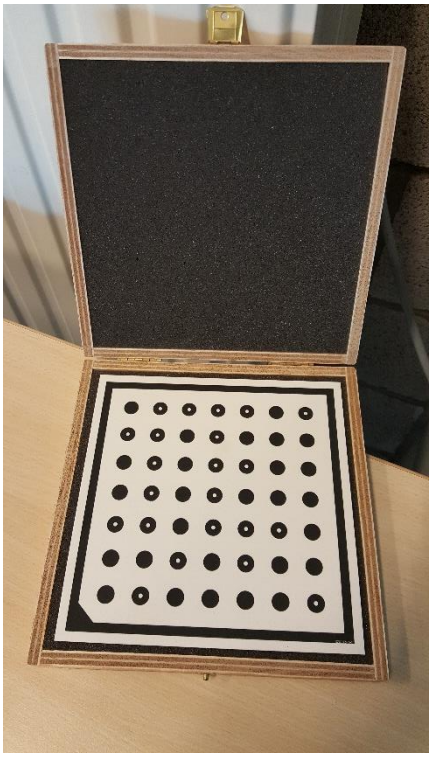
$$Ongelijkheid = U_L - U_R = f \cdot \frac{b}{Z} \quad (3.3)$$

en

$$Diepte = f \cdot \frac{b}{Ongelijkheid} \quad (3.4)$$

In werkelijkheid is een stereovisiesysteem veel complexer als het hierboven aangehaalde voorbeeld. Maar alle basisprincipes zijn wel steeds hetzelfde.

De aannames van hierboven zijn enkel in ideale omstandigheden voor de vereenvoudigde stereovisie systeem en kunnen niet worden gebruikt voor de reële stereovisiesystemen. Zelfs de beste camera's en lenzen hebben een bepaalde vervorming van hun beeld. Om dit te compenseren is het nodig dat elk stereovisiesysteem na wijziging van opstelling, gekalibreerd wordt. Het kalibratieproces maakt gebruik van een kalibratierooster. Deze zorgt ervoor dat men in verschillende standpunten de beeldvervorming van de camera's kan berekenen om zo de exacte relatie tussen de camera's te bepalen. Figuur 3-2 toont het gebruikte kalibratierooster voor de Ensenso die bij ACRO staat opgesteld.



Figuur 3-2: Kalibratierooster stereovisieopstelling

Een stereovisie systeem is nuttig in de industriële robotica & automatisering voor het uitvoeren van taken zoals "bin picking" of "krat handling". Autonome voertuigen gebruiken

de diepte informatie om de grootte en de afstand van obstakels te bepalen en zorgt voor een nauwkeurige routeplanning om obstakels te vermijden. Tevens kunnen stereovisiesystemen zeer nuttige 3D-informatie voorzien voor applicaties zoals een navigatiesysteem. Dit komt omdat stereovisie niet gevoelig is voor wisselende lichtomstandigheden en daardoor goed kunnen blijven presteren.

3.2.3 Soorten stereovisiecamera's

3.2.3.1 Ensenso

De Ensenso Stereo 3D-Camera werkt volgens het principe *projected texture stereo vision.* Elk model heeft twee geïntegreerde CMOS-sensoren en een projector die een willekeurig puntenpatroon projecteert op een object. Dit puntenpatroon wordt door de camera opgenomen en vastgelegd. Op deze manier kan de textuur en het oppervlak van het object waargenomen worden. De compacte en robuuste aluminium behuizing van de Ensenso-camera's duiden op de geschiktheid van de camera voor industrieel gebruik. Daarnaast beschikt deze camera over een Halcon interface en een object-georiënteerde API. API is de afkorting van **A**pplication **P**rogramming **I**nterface en is een verzameling van definities waarop een computerprogramma kan communiceren met externe apparatuur [11]. Figuur 3-3 [12] geeft de Ensenso 3D-Camera weer die onderdeel uitmaakt van de opstelling bij ACRO. Deze camera kan tot 30 frames per seconde nemen met een volledige resolutie.

Figuur 3-3: Ensenso N35-602-16-IR, stereocamera

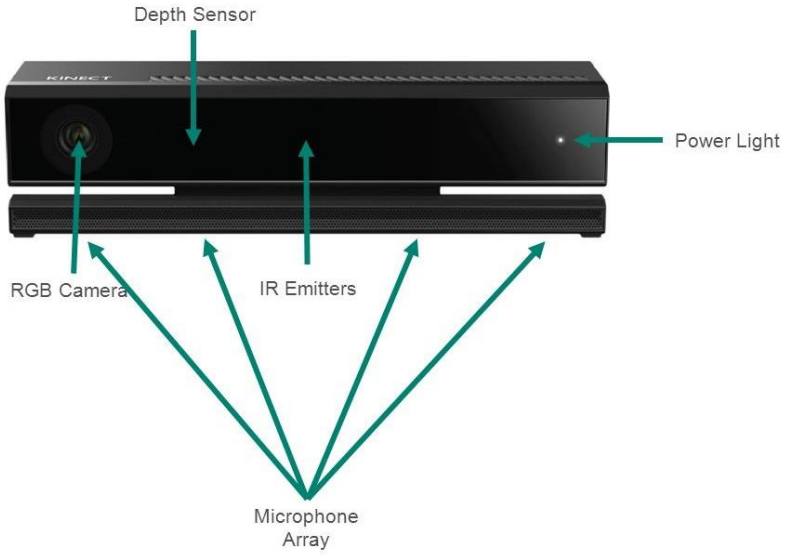
De belangrijkste specificaties van de Ensenso die gebruikt wordt tijdens deze masterproef worden in Tabel 3-1 weergegeven.

Tabel 3-1: Specificaties Ensenso N35-602-16-IR

Specificaties Ensenso N35-602-16-IR	
Interface	Gigabit Ethernet
Resolution	1280 x 1024 (1,3 MP)
Frames per second	10 (1,3 MP); 30 (2 x binning)
Focus distance	950 mm
Operating distance	460 - 3000 mm

3.2.3.2 Kinect V2

De Kinect V2 is een sensor ontworpen door de computergigant Microsoft. Oorspronkelijk is het toestel gemaakt voor interactie met de Xbox ONE. Dit toestel kan ook als camera gebruikt worden op een Windows-computer. Omdat de bin picking-opstelling in de universiteit van Leuven gebruik maakt van een Kinect V1, was het oorspronkelijk ook de bedoeling om in Diepenbeek een opstelling te maken met de Kinect V2. Dit zou de communicatie tussen beide opstellingen vereenvoudigen. Het was helaas niet mogelijk om de Kinect V2 met het visieprogramma Halcon te koppelen, doordat de nodige drivers (nog) niet waren vrijgegeven. Deze camera werkt ook volgens het principe van stereovisie. Figuur 3-4 [13] geeft de Kinect V2 en bijhorende onderdelen weer.



Figuur 3-4: Opbouw Kinect V2

De belangrijkste specificaties van de Kinect V2 worden in Tabel 3-2 weergegeven [14].

Tabel 3-2: Specificaties Kinect V2

Specificaties Kinect V2	
Viewing angle	60° vertical by 70° horizontal field of view
Depth range	0,4 m - 0,45 m
Frame rate	1920 x 1080 @30 fps
Audio format	48-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC)
Interface	USB 3.0

3.3 Sheet-of-light

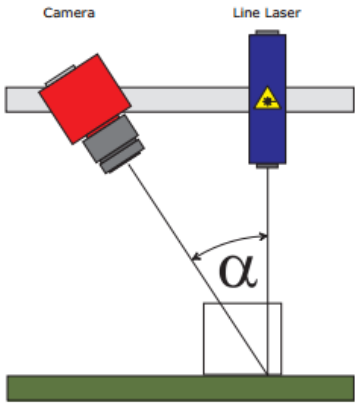
3.3.1 Algemeen

Het Sheet-of-lightprincipe wordt ook wel lasertriangulatie genoemd. In deze 3D-meetmethode wordt een laserbundel gericht op het oppervlak van de te inspecteren scene. Met behulp van een Photonfocus-camera wordt er aan de hand van triangulatie onder een hoek α een beeld van de laserlijn genomen. Door de opstelling van de laser en de camera, wordt elke wijziging aan de oppervlakte van het werkstuk waargenomen.

De camera en laser verplaatsen zich via een lineaire geleiding over het te inspecteren oppervlak. De taak van de Photonfocus-camera is om met een maximale snelheid en precisie de positie van de laserlijn te bepalen. Zo kan het hoogteprofiel van het object worden bepaald.

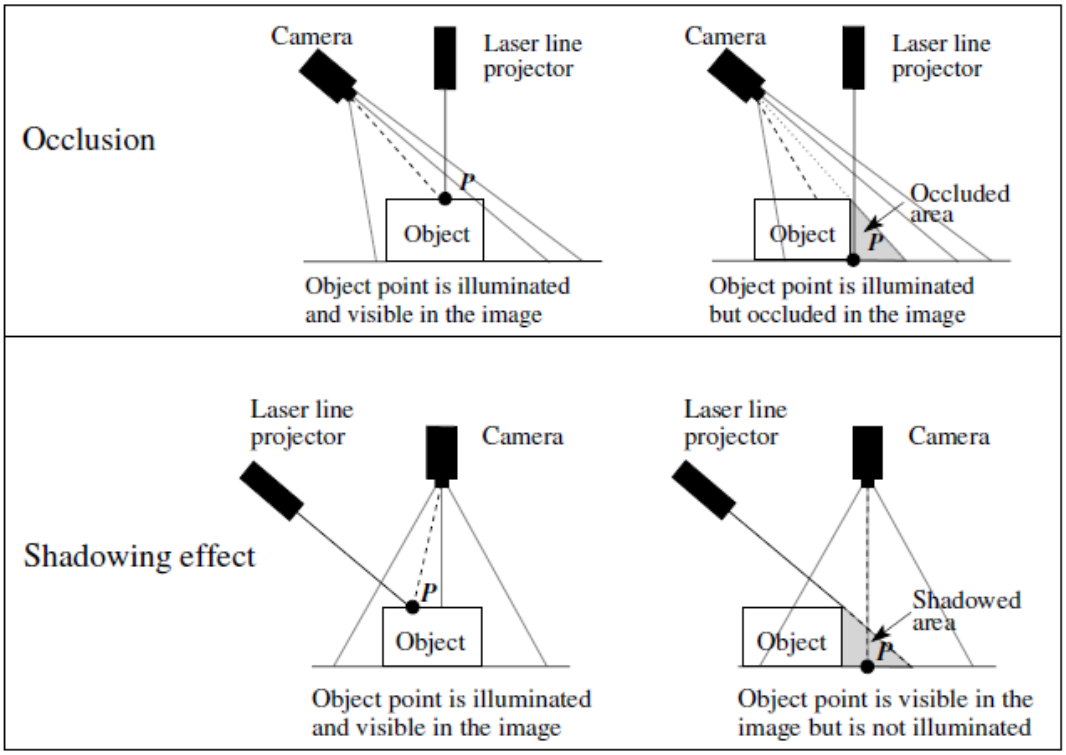
3.3.2 Werkingsprincipe

In Figuur 3-5 [15] wordt het principe en de opstelling van het Sheet-of-lightprincipe afgebeeld. Deze manier van opstelling gebruikt men bij ACRO ook. Namelijk een schuin gepositioneerde camera en een loodrecht gepositioneerde laser op het grondvlak. De hoek tussen de camera en de laserlijn wordt aangegeven als hoek α. Deze hoek wordt ook wel de triangulatiehoek genoemd.



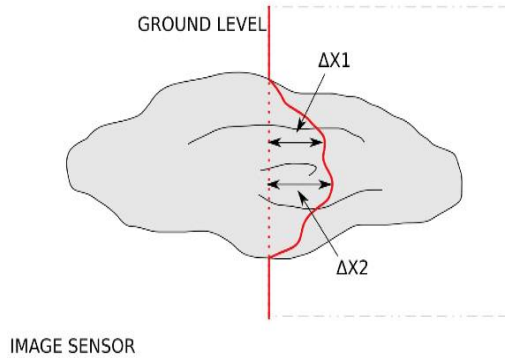
Figuur 3-5: Opstelling sheet-of-light

Andere mogelijkheden zijn het schuin positioneren van de laser en een loodrecht gepositioneerde camera ten opzichte van het grondvlak. Een combinatie van beide is ook mogelijk, waar zowel de camera als de laser onder een bepaalde hoek van het grondvlak staan. De manier van opstelling is afhankelijk van de geometrie van de objecten die gescand worden, om zo min mogelijk schaduwvorming en occlusies te bekomen. Figuur 3-6 geeft per verschillende opstelling de mogelijk occlusies en schaduwvormingen weer. Wat de beste opstelling is voor welke situatie wordt in deze thesis niet besproken omdat dit buiten het doel van deze thesis valt.



Figuur 3-6: Occlusies en schaduweffecten naargelang de opstelling van de camera en laser

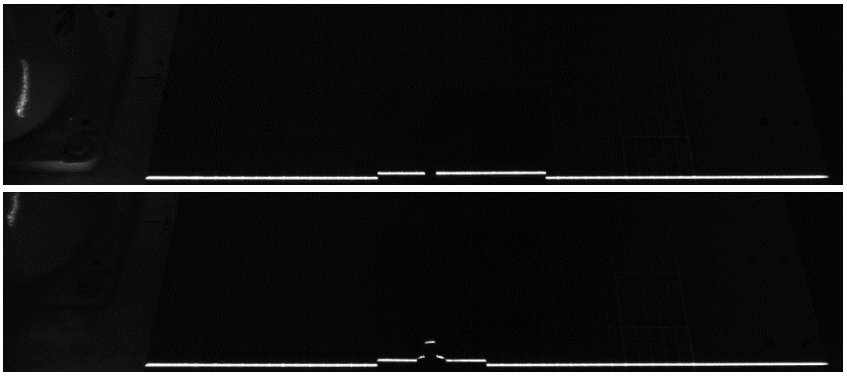
In werkelijkheid wordt de diepte-informatie (ΔZ) verkregen door het waarnemen van de verandering in X- of Y- richting van de laserlijn. De camera detecteert de laserlijn en met behulp van een sheet-of-lightalgoritme kan het diepteverschil en de contouren van het object worden bepaald. Een grotere ΔX komt overeen met een grotere variatie in diepte (ΔZ). Figuur 3-7 geeft dit principe weer.



Figuur 3-7: Verandering van de laserlijn wijst op een hoogteverschil

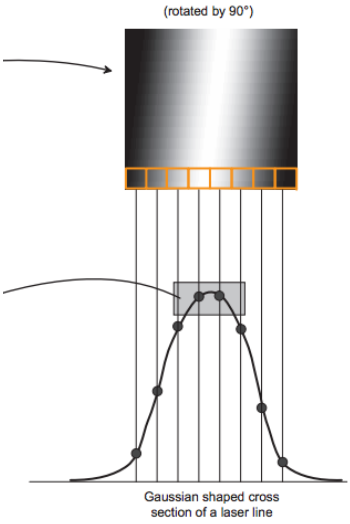
De camera bestaat uit een niet-lineaire laserlijndetectie die het mogelijk maakt de positie van de laserpiek nauwkeurig op sub-pixelniveau te detecteren. Deze interpolatiemethode biedt betere resultaten dan het gebruikelijke algoritme dat wordt gebruikt in het softwareprogramma Halcon.

De Photonfocus-camera zal de laserlijn en de veranderingen ervan waarnemen zoals afgebeeld in Figuur 3-8.



Figuur 3-8: Laserlijn en vervormingen waargenomen door Photonfocus-camera

Sheet-of-lightopstellingen zijn voornamelijk gebaseerd op een correcte bepaling van de piekpositie van de laserlijn waarbij de pixelwaardes een Gausscurve hebben. Figuur 3-9 geeft dergelijke laserlijn weer in de vorm van een Gausscurve. Deze vorm wordt verkregen aan de hand van non-lineaire interpolatietechnieken. Op deze manier zijn subpixelresultaten mogelijk. Deze techniek is superieur aan andere technieken zoals het “center” of zwaartepuntsalgoritme dat in de meeste triangulatiesystemen gebruikt wordt. Ook Halcon maakt gebruik van dit zwaartepuntsalgoritme.



Figuur 3-9: Laserlijn als Gausscurve

3.3.3 Photonfocus-camera

De Sheet-of-lightopstelling maakt naast gebruik van een laser ook gebruik van een Photonfocus-camera om de laserlijn te detecteren. Het Zwitserse Photonfocus is een van de marktleiders in de ontwikkeling van CMOS-beeldsensoren die gebruikt worden voor machinevisie. De camera die in opgesteld is bij ACRO, is de Photonfocus MV1- D1312IE-160-CL. De beschreven camera is uitermate geschikt omwille van de efficiënte integratie met de software van Halcon. Verder is gebleken dat de mogelijkheid tot laserlijndetectie zeer geschikt is en tot slot heeft de camera de optie om in een 3D-mode te werken. Figuur 3-10

geeft de Photonfocus-camera weer die gebruikt wordt in de bin picking-opstelling. Tabel 3-3 [16].

Figuur 3-10: Photonfocus-camera

De belangrijkste specificaties van de Photonfocus-camera worden in Tabel 3-3 weergegeven [16].

Tabel 3-3: Specificaties Photonfocus-camera

Cameraspecificaties Photonfocus MV1-D1312IE-160-CL	
Interface	Camera Link
Frame rate	108fps
Pixel clock	80MHz
Exposure time range	10µs - 419ms
Trigger Modes	Free running (non-triggered), external Trigger, SWTrigger
Features	Configurable region of interest (ROI), Up to 512 regions of interest (MROI), Decimation in y-direction, Image correction, 2 look-up tables (12-to-8Bit) on user-defined image region (Region-LUT), Constant frame rate independent of exposure time, Crosshairs overlay on the image, 3x3 convolver for image preprocessing, Temperature monitoring of sensor and camera, Camera information readable over SDK, Ultra low trigger delay and low trigger jitter, Extended trigger input and strobe output functionality, Status line in picture

3.4 Bin picking-opstelling

In deze paragraaf wordt er in detail de bin picking-opstelling besproken. Deze bestaat uit een lineaire geleiding waarop een laser en een cameraconfiguratie op bevestigd is en een robot. Figuur 3-11 geeft de gehele opstelling weer met robot, camera en laser.

Figuur 3-11: Bin picking-opstelling met pikrobot en lineaire geleiding met camera en laser

De cameraconfiguratie bestaat echter niet enkel uit de Photonfocus-camera. De gehele configuratie bestaat uit een camera (1), een lens (2) en een bandpassfilter (3), zoals weergegeven in Figuur 3-12.

Figuur 3-12: Cameraconfiguratie met (1) Photonfocus-camera, (2) lens en (3) bandpassfilter

De desbetreffende Photonfocus-camera is hierboven in paragraaf 3.3.3 besproken met bijhorende specificaties in Tabel 3-3. De gebruikte lens is van het merk Cosmicar Pentax. Deze lens is speciaal gemaakt voor het gebruik op CMOS-beeldsensoren en de focusringen die men vast kan schroeven zijn optimaal voor het gebruik op bewegende en/of trillende opstellingen. Bijhorende specificaties zijn in Tabel 3-4 weergegeven [17].

Tabel 3-4: Specificaties lens Cosmicar Pentax C21211KP

Specificaties Cosmicar Pentax C21211KP	
Brandpuntafstand	12,5 mm
Maximale apertuur	1:1,4
Minimale objectafstand	0,3 m
Horizontale beeldhoek	16,21° - 38,65°
Gewicht	135 g

Het laatste onderdeel van de cameraconfiguratie is de bandpassfilter. Deze bandpassfilter laat enkel lichtgolven door met een golflengte van 630 nm tot 690 nm en maakt het zo mogelijk dat enkel het licht dat de laserstraal produceert opgenomen wordt door de camera.

De gebruikte bandpassfilter is de Scheider Kreuznach IFG BP 660-60 RoHS. Tabel 3-5 geeft de specificaties van deze bandpassfilter weer [18].

Tabel 3-5: Specificaties bandpassfilter

Specificaties Scheider Kreuznach IFG BP 660-60 RoHS	
Golflengte doorlaatbaarheid	630 nm - 690 nm
Piek doorlaatbaarheid	> 80%
Dikte	2,0 mm
Toepassingen	Laserapplicaties, inspectie semiconductor, inspectie eten en drank, wetenschappelijk onderzoek, biomedica

Tot slot bestaat de bin picking-opstelling nog uit een robot om de objecten op te pikken. Tijdens deze thesis is hier niet mee gewerkt, maar Tabel 3-6 geeft de belangrijkste specificaties van de robot weer.

Tabel 3-6: Specificaties robot Epson C3-A601S

Specificaties Epson C3-A601S	
Bouwwijze	Robot met 6 assen
Laadvermogen	Maximaal 3 kg
Horizontaal bereik	600 mm
Werkbereik	As 1: +/- 170 °, As 2: + 65 ° / - 160 °, As 3: + 225 ° / - 51 °, As 4: +/- 200 °, As 5: +/- 135 °, As 6: +/- 360 °
Gewicht	27 kg

4 Automatische parameterbepaling

4.1 Inleiding

In dit hoofdstuk wordt beschreven hoe het hoofddoel van deze thesis is bereikt, namelijk het automatisch bepalen van de paramaters. Eerst wordt er uitgelegd welke parameters er moeten variëren en wat de mogelijke parameterwaarden hiervoor zijn. Daarna wordt een algemeen beeld geschetst van de methode die gebruikt is om het algoritme te creëren. Hierbij horen ook de problemen die in dit proces zijn voorgekomen. Tot slot worden de experimenten en bijhorende resultaten beschreven, die doorlopen zijn om tot het finale algoritme te komen.

4.2 Parameters en hun bereik

Vooraleer het mogelijk is om de code te schrijven, is het nodig om de belangrijkste operators met bijhorende parameters te bestuderen. Deze operators zijn reeds bestaande functies in het visieprogramma waarbij er inputparameters meegegeven moeten worden die een output als resultaat geven. De belangrijkste operators voor het matching-proces zijn “create_surface_model” en “find_surface_model”. Hieronder staat een beschrijving van alle inputparameters en outputs die in deze operators voorkomen.

4.2.1 Parameters van operator “create_surface_model”

Inputparameters: ObjectModel3D: 3D-model waarvan er een surface-model van gemaakt moet worden.
RelSamplingDistance: sample-afstand ten opzichte van de diameter van het object. Deze waarde heeft volgend bereik:
0 < RelSamplingDistance < 1. Figuur 4-1 geeft voor 3 verschillende waardes van deze parameter een voorbeeld.

Figuur 4-1: Effecten van verschillende waardes voor RelSamplingDisctance. Respectievelijk 0,3; 0,5 en 0,7

Hoe groter de keuze voor RelSamplingDistance hoe minder punten er gebruikt worden om het model te maken.

GenParamName: Generische parameters die worden meegegeven om het model te genereren.

- Model_invert_normals: door deze functie op true of false te zetten worden de normalen van het model omgedraaid.

Afhankelijk van hoe het model gecreëerd wordt, moet de richting van de normalen aangepast worden. Wanneer er bijvoorbeeld een CAD-model gebruikt wordt, moeten de normalen vaak omgedraaid worden.

- Pose_ref_rel_sampling_distance: parameter die de sample-afstand bepaald voor de positieverfijning. Ook hier wordt een getal mee gegeven dat tussen 0 en 1 is gelegen. Afhankelijk van het soort object zal deze parameter ingesteld worden, maar vaak zal de default-waarde van 0,01 volstaan.
- Feat_step_size_rel: het wordt afgeraden deze waarde aan te passen door de makers van het visieprogramma. De default waarde van deze parameter staat op de waarde van RelSamplingDistance en heeft dus ook hetzelfde bereik als de parameter RelSamplingDistance.
- Feat_angle_resolution: ook niet aanpassen. Geeft geen voordelen bij het creëren van het model in deze thesis. Deze parameter wordt enkel aangepast als het model gemaakt wordt van een afbeelding die veel ruis bevat.

Output: SurfaceModelID: het gemaakte model wordt hierin opgeslagen.

4.2.2 Parameters van de operator "find_surface_model"

Inputparameters: SurfaceModelID: hierin wordt het SurfaceModel meegegeven dat gemaakt is aan de hand van de functie "create_surface_model".

ObjectModel3D: dit is de 3D-scene waarin het object (surfaceModelID) gevonden moet worden.

RelSamplingDistance: sample-afstand ten opzichte van de diameter van het object, die het aantal punten bepaald waaruit de puntenwolk bestaat. Deze waarde heeft volgend bereik: 0 < RelSamplingDistance < 1. Hoe groter de keuze voor RelSamplingDistance hoe minder punten er gebruikt worden om het model te maken. Wel is het aan te raden dat deze sample-afstand niet kleiner genomen wordt dan de sample-afstand die gebruikt wordt in de operator "create_surface_model".

KeyPointFraction: de waarde van deze parameter geeft het percentage weer van het aantal keypoints dat geselecteerd wordt uit de bemonsterde scene. Als de waarde van de parameter 0,1 is, wordt er 10% van de gesamplede scene gebruikt als keypoints. Hoe groter de waarde van deze parameter hoe groter de kans is dat de matches die

gevonden worden ook effectief juist zijn (stabiel), maar hoe trager het programma werkt. Aangezien er in deze thesis gezocht wordt naar de optimale parameters, wordt er geen rekening gehouden met de tijd.

MinScore: “Score” (zie uitleg Score bij output) is de waarde die het programma berekent op basis van de overeenkomsten met de matches. Hoe lager deze score hoe minder overeenkomsten er zullen zijn tussen het meegegeven model en de match. MinScore is een waarde tussen 0 en 1, die meegeven wordt zodat er geen (foutieve) matches gevonden worden, die lager liggen dan de MinScore.

ReturnResultHandle: Deze waarde zal in alle onze gevallen altijd op “true”, staan. Dit zorgt ervoor dat er ook effectief een surface matching resultaat wordt meegegeven in de output SurfaceMatchingResultID.

GenParamName: hierin kunnen nog extra generische parameters worden meegegeven. Deze gaan zeker belangrijk zijn bij het gebruik in scenes met meerdere objecten. De parameters die hierin meegegeven kunnen worden en variëren zijn:

- Num_matches: bepaalt het maximaal aantal matches dat gevonden kan worden. De waarde van deze parameter is groter of gelijk aan 1.
- Max_overlap_dist_rel: bepaalt in hoeverre er overlapping mag zijn met een gevonden match die een hogere score heeft. Het bereik van deze parameter ligt tussen 0 en 1. Waarbij er geen overlapping mag zijn als de waarde is ingesteld op 1.
- Score_type: bepaalt de manier waarop er een score wordt toegekend aan een match.
- Scene_normal_computation: bepaalt op welke manier de normalen berekend worden. De 2 mogelijk manieren zijn “fast” of “mls”, waarbij “fast” de snelle manier is en “mls” voor moving least squares staat.
- Sparse_pose_refinement: parameter die bepaalt of er gebruik gemaakt wordt van deze positieverfijning. Parameterwaarde: “true” of “false”.
- Dense_pose_refinement: parameter die bepaalt of er gebruik gemaakt wordt van de positieverfijning op basis van dichtheid van de punten in de puntenwolk. Parameterwaarde: “true” of “false”.
- Pose_ref_num_steps: aantal herhalingen die er gebruikt worden voor de operator dense_pose_refinement en

sparse_pose_refinement. Deze parameterwaarde moet groter zijn als 0.

- Pose_ref_use_scene_normals: bepaalt of er al dan geen gebruik wordt gemaakt van normalen bij de pose_refinement. Parameterwaarde: “true” of “false”.

GenParamValue: hierin wordt de waarde meegegeven van de generische parameters die hierboven opgesomd zijn.

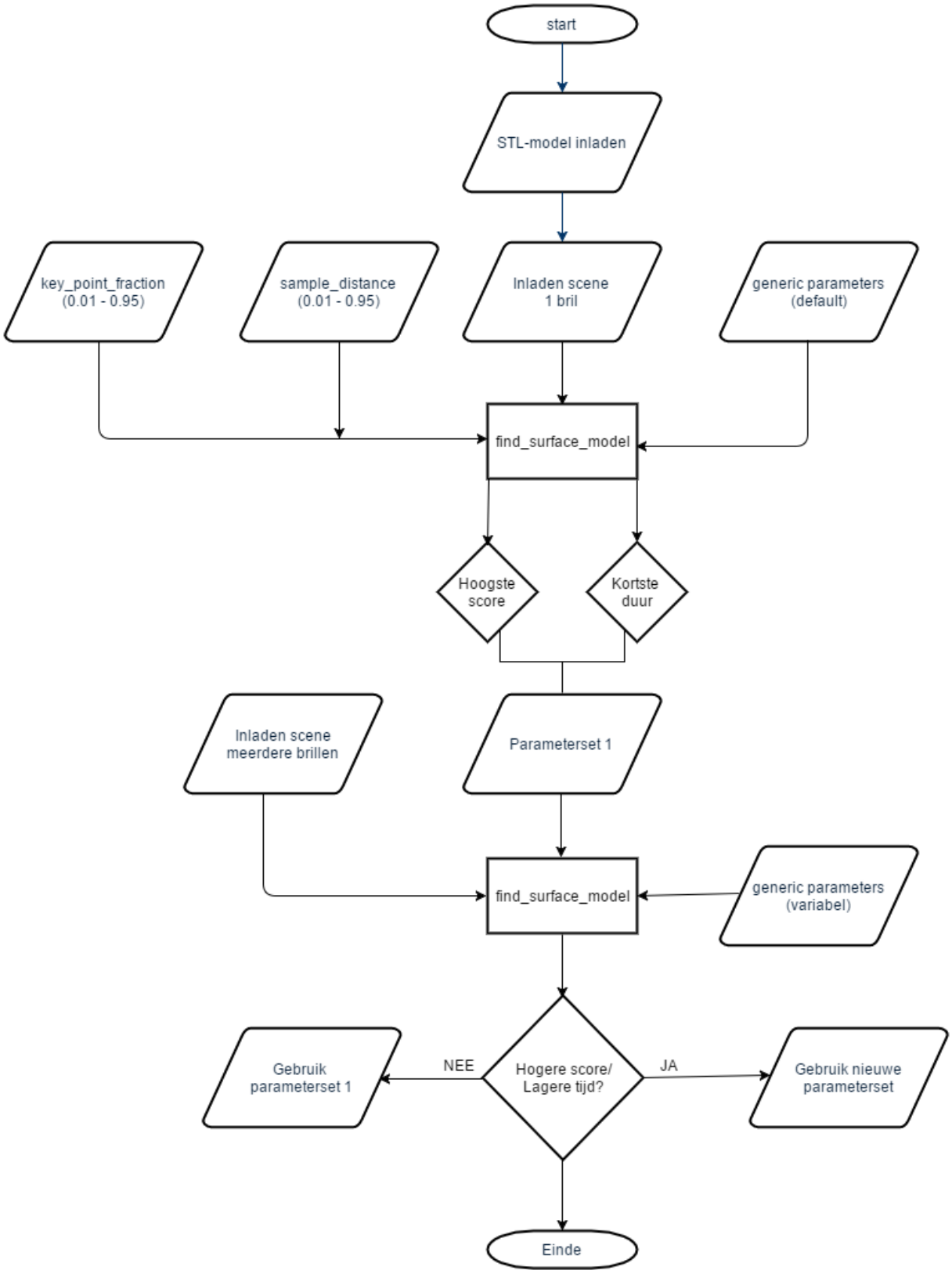
Output: Pose: de X-, Y- en Z-positie uitgedrukt in millimeter waar de gevonden matches liggen in de scene.

Score: de waarde die het programma berekent op basis van de overeenkomsten met de matches. Hoe lager deze score hoe minder overeenkomsten er zullen zijn tussen het meegegeven model en de match.

SurfaceMatchingResultID: de matches die uiteindelijk gevonden worden, en hierin worden verzameld om erna op te vragen of te visualiseren.

4.3 Parameterbepaling aan de hand van geneste for-lussen

De algemene aanpak voor het automatisch bepalen van de optimale parameterset is hieronder in Figuur 4-2 aan de hand van een flowchart weergegeven.



Figuur 4-2: Flowchart automatische parameterbepaling aan de hand van geneste for-lussen

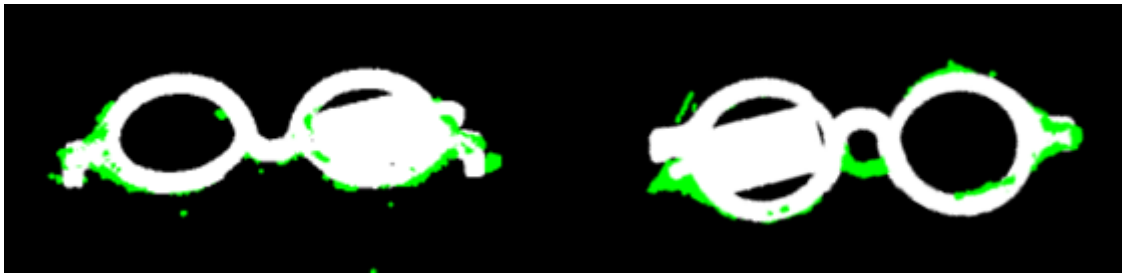
Aan de hand van deze flowchart bepaalt het programma welke parameterset de beste match behaalt. Deze “beste match” wordt bepaald op basis van het aantal overeenkomstige punten van het model in de scene en de tijd die het programma hiervoor nodig heeft.

4.4 Matching problemen

Tijdens het proces van het automatisch bepalen van de parameters zijn er enkele problemen opgedoken. Deze problemen worden hieronder beschreven en uiteindelijk van een gezamenlijke oplossing voorzien.

4.4.1 Positieprobleem

Met het huidig algoritme gebeurt het dat er parametersets gevonden worden die de hoogste score opleveren, maar geen goede match als resultaat hebben. Het meest voorkomende is dat de positie van de match niet overeenkomt met de positie/oriëntatie van het object. Onderstaande Figuur 4-3 geeft twee matches weer. De match links afgebeeld is een goede match, want de witte bril ligt volledig over de groene scene. De match rechts is geen goede match, want de witte bril ligt niet volledig over de groene scene.



Figuur 4-3: Goede match vergeleken met slechte match

Om zo eenvoudig mogelijk te beginnen wordt er eerst een scene ingeladen van 1 object/bril. Hierop variëren de basisparameters “sample_distance” en “key_point_fraction” (van de operator find_surface_model) aan de hand van geneste for-lussen tussen respectievelijk 0,01 – 0,95 (stapgrootte = 0,01) en 0,01 – 0,95 (stapgrootte = 0,01). Hieruit wordt dan op basis van de hoogste score, de beste parameterset bepaald.

Voor dit voorbeeld zijn onderstaande sets degene die de hoogste score verkrijgen:

beste_sample_distance = [0,07; 0,07; 0,04; 0,07; 0,07; 0,03; 0,03; 0,04]

beste_key_point_fraction = [0,17; 0,18; 0,19; 0,19; 0,2; 0,21; 0,23; 0,25]

Naast de verplichte parameters sample_distance en key_point_fraction, moeten ook de generische parameters automatisch bepaald worden. Dit is voor volgende generische parameters gebeurd: “max_overlap_dist_rel”, “sparse_pose_refinement”, “dense_pose_refinement”, “pose_ref_num_steps” en “pose_ref_use_scene_normals”. Geen enkele van deze parameters had een invloed op de score, omdat dit enkel zal gebeuren in scenes met meerdere objecten en er nu slechts één object per scene is.

Hier is vervolgens op ingespeeld door de enkel de parameters sample_distance en key_point_fraction te laten variëren op de scene van één object. Vervolgens worden de parameters beste_sample_distance en beste_key_point_fraction gebruikt op een scene met

meerdere objecten. Hierbij worden de generische parameters nu wel gebruikt om te kijken of deze variatie invloed heeft. Uiteraard worden er nu wel andere resultaten behaald door gebruik te maken van parametersets met verschillende generische parameters.

Maar hier duikt ook het probleem op. Er zijn 3 factoren waarop een parameterset beoordeeld kan worden: 1. Score 2. Pose 3. Tijd. Op basis van score en tijd, kan men bepalen of de match al dan niet veel overeenkomsten heeft en of dit proces snel is verlopen. De pose bepaalt of de match goed gelegen is in de scene. Dit kan men echter niet bepalen zonder dat er eerst visueel gekeken wordt naar de pose(s) die gevonden moeten worden. Hierdoor zal het niet volledig mogelijk zijn om alles automatisch te bepalen, tenzij de posities op voorhand zijn bepaald van de te zoeken objecten. Zo kunnen de poses gebruikt worden als feedback.

4.4.2 Meerdere matches

Doordat “score” en “tijd” de enige outputs zijn waarop een match automatisch gecontroleerd kan worden, is er een veel voorkomend probleem met matches die over elkaar geplaatst worden op hetzelfde object. Figuur 4-4 geeft dit probleem weer.

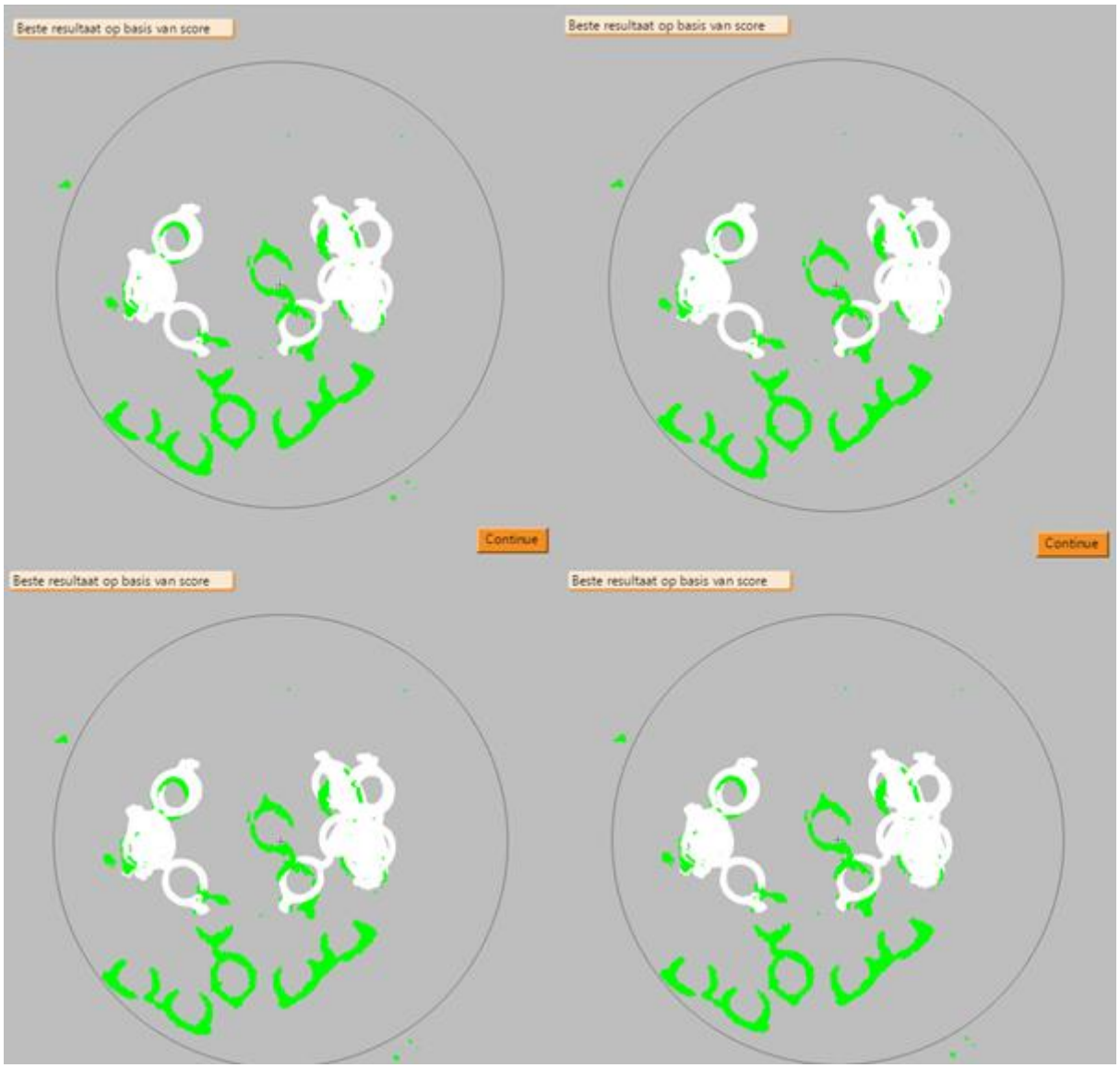
Figuur 4-4: Meerdere matches gevonden op eenzelfde plaats

In bovenstaande afbeelding worden namelijk 7 matches gevonden, die op dezelfde plaats boven elkaar liggen. Het is moeilijk om deze 7 matches onderling te onderscheiden op Figuur 4-4. Aan de hand van de brug van de bril, kan men wel vaststellen dat er minstens 2 matches

gevonden worden voor hetzelfde object in de scene. Volgens het geschreven algoritme is dit het resultaat van de beste parameterset. Dit wil zeggen dat met deze parameters de hoogste score wordt bereikt. Dit resultaat kan voorkomen worden door de generische parameter “max_overlap_distance” aan te passen aan de hand van de positie die de objecten in de scene hebben.

4.4.3 Scoreprobleem

Bij het automatisch variëren van de generische parameters “max_overlap_dist_rel”, “sparse_pose_refinement”, “dense_pose_refinement”, “pose_ref_num_steps” en “pose_ref_use_scene_normals” was de score bij bepaalde parametersettings hoger dan 1. De hoogste score was namelijk 14. Dit is geen voorspelbaar resultaat, aangezien de ingestelde scorewaarde normaal gelegen is tussen 0 en 1. Echter blijkt dit enkel zo te zijn als de generische parameters “dense_pose_refinement” en “sparse_pose_refinement” de waarde “true” hebben. Indien minstens een van deze parameters de waarde “false” heeft, is de waarde van de output “Score” groter of gelijk aan 0. Na controle blijkt dat de matches die gevonden worden bij een score hoger dan 1, niet correct zijn. In Figuur 4-5 vindt men de voorbeelden van de matches voor de scorewaardes van respectievelijk 4, 7, 10 en 14.



Figuur 4-5: Gevonden matches bij scores hoger als 1. Respectievelijk scorewaarde 4, 7, 10 en 14

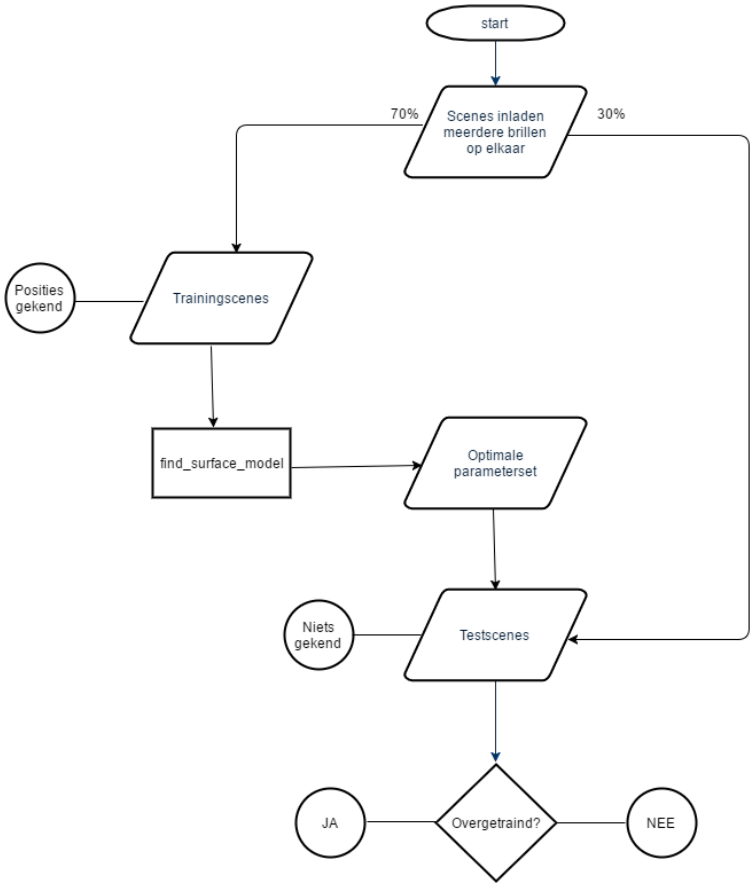
Eens de score groter is dan 1, blijken de matches hetzelfde te blijven. Op Figuur 4-5 is er geen verschil zichtbaar tussen de gevonden matches met andere scorewaardes bij een score hoger dan 1. Voor het automatisch bepalen van de parameterset zullen alle waarden die hoger liggen dan 1 geëlimineerd worden.

4.4.4 Oplossing

In de huidige code wordt de beste parameterset bepaald op basis van de hoogste score. Deze score wordt berekend op basis van de overeenkomstige punten van het model in een scene. Een voorbeeld van dergelijke gevonden match in een scene is weergegeven in Figuur 4-6.

Figuur 4-6: Gevonden match (wit) in een scene van meerdere brillen (groen)

Momenteel blijkt dat enkel op basis van de score niet altijd een betrouwbaar resultaat wordt bekomen. Hiervoor zal er een extra controlefactor geïmplementeerd worden op basis van de positie van de objecten in de scene. De algemene aanpak van deze controlefactor is afgebeeld in Figuur 4-7.



Figuur 4-7: Flowchart extra controlefactor

Dit principe is gebaseerd op een aantal scenes die dienen als trainingsscenes en testscenes. De trainingsscenes worden gebruikt om de optimale parameters te bepalen en de testscenes worden gebruikt om na te gaan of deze optimale parameters ook werken op willekeurige scenes. Van de trainingsscenes zijn de posities van de objecten op voorhand bepaald en worden de gevonden matches vergeleken met deze posities. Indien de positie overeenkomt, wordt de score en bijhorende parameters opgeslagen om vervolgens op de testscenes te gebruiken waarvan geen posities bekend zijn.

4.5 Experimenten/methodes

Het schrijven van een algoritme is een procedure van trial-and-error. Daarom wordt er in deze paragraaf besproken welke stappen ondernomen zijn, die geleid hebben tot het huidige algoritme.

4.5.1 Geneste for-lussen

Om de beste parameterset te bepalen voor het vinden van matches, moeten alle mogelijke parametercombinaties getest worden. Dit gebeurt aan de hand van geneste for-lussen. Dit is een for-lus waarin nog een of meerdere for-lussen gebruikt worden. In Figuur 4-8 is een

voorbeeld weergegeven van een simpele, geneste for-lus. Op deze manier wordt elke mogelijke combinatie van de parameter sampleDistance (=SD) en keyPointFraction (=KPF) overlopen (regel 6 & 7 in Figuur 4-8) en respectievelijk bijgehouden in de tuples SD_tuple en KPF_tuple (regel 11&12 in Figuur 4-8). Een tuple is een eindige rij van objecten, waarin verschillende soorten data in verzameld kunnen worden.

```
1  SurfaceModelID:='model van het object dat gezocht wordt in een scene'
2  ObjectModel3D1:='scene waarin er naar het model wordt gezocht'
3
4  teller:=0
5
6  for SD:=0 to 1 by 0.01
7      for KPF:=0 to 1 by 0.01
8
9          find_surface_model (SurfaceModelID, ObjectModel3D1, SD, KPF, 0.1, 'true', [], [], Pose, Score, SurfaceMatchingResultID)
10
11         SD_tuple[teller]:=SD
12         KPF_tuple[teller]:=KPF
13
14         teller:=teller+1
15
16     endfor
17 endfor
```

Figuur 4-8: Voorbeeld van een geneste for-lus

In het algoritme worden geneste for-lussen gebruikt om de parameters van de operator “find_surface_model” te laten variëren. Aan de hand van de outputs “Pose” en “Score”, krijgt men respectievelijk de positie van de match en een waarde, die de mate van overeenkomst tussen het model en de gevonden match in de scene weergeeft.

Een andere mogelijkheid voor het opstellen van de geneste for-lus was aan de hand van opsplitsing. Waarbij er in Figuur 4-8 het bereik wordt afgelopen in stappen van 0,01, is het ook mogelijk om deze stapgrootte eerst te vergroten. Wanneer dan duidelijk is dat een bepaalde waarde een goed resultaat geeft, kan men de parameterwaarde rond deze waarde laten variëren en in kleinere stappen overlopen. Deze methode is weergegeven in Figuur 4-9, regel 6 & 7 en 22 & 23.

```
1  SurfaceModelID:='model van het object dat gezocht wordt in een scene'
2  ObjectModel3D1:='scene waarin er naar het model wordt gezocht'
3
4  teller:=0
5
6  for SD:=0 to 1 by 0.1
7      for KPF:=0 to 1 by 0.1
8
9          find_surface_model (SurfaceModelID, ObjectModel3D1, SD, KPF, 0.1, 'true', [], [], Pose, Score, SurfaceMatchingResultID)
10
11         SD_tuple[teller]:=SD
12         KPF_tuple[teller]:=KPF
13
14         teller:=teller+1
15
16     endfor
17 endfor
18
19 SD_goed_resultaat:='sample disctance waarbij in de vorige geneste for-lus een goed resultaat werd behaald'
20 KPF_goed_resultaat:='key point fraction waarbij in de vorige geneste for-lus een goed resultaat werd behaald'
21
22 for SD:=SD_goed_resultaat-0.2 to SD_goed_resultaat+0.2 by 0.01
23     for KPF:=KPF_goed_resultaat-0.2 to KPF_goed_resultaat+0.2 by 0.01
24
25         find_surface_model (SurfaceModelID, ObjectModel3D1, SD, KPF, 0.1, 'true', [], [], Pose2, Score2, SurfaceMatchingResultID)
26
27     endfor
28 endfor
```

Figuur 4-9: Geneste for-lussen met opsplitsing van stapgrootte

Hierbij worden beide parameters eerst in stapgroottes van 0,1 doorlopen en vervolgens worden in een bereik van +/- 0,2 de parameterwaardes die een goed resultaat behaalden gevarieerd in stapgroottes van 0,01.

Dit bleek echter niet altijd te werken zoals verwacht. Het eerste probleem is dat de waardes van de eerste for-lus niet klein genoeg zijn om correcte matches te vinden. Het gevolg hiervan is dat de behaalde scores allemaal gelijk zijn 0, waardoor de tweede geneste for-lus alle parameters van de voorgaande for-lus ging overlopen in kleinere stappen. Wat dus hetzelfde resultaat geeft als de methode zonder opsplitsing, uitgelegd in Figuur 4-8, als resultaat heeft.

Een tweede probleem is dat de tijd die het programma nodig had voor de methode met opsplitsing groter was dan de tijd die nodig was zonder opsplitsing.

Omwille van deze redenen is er gekozen om gebruik te maken van de methode zonder opsplitsing, aangezien het tijdbesparende effect niet altijd vervuld kan worden.

4.5.2 Outputs Score, Pose en tijd

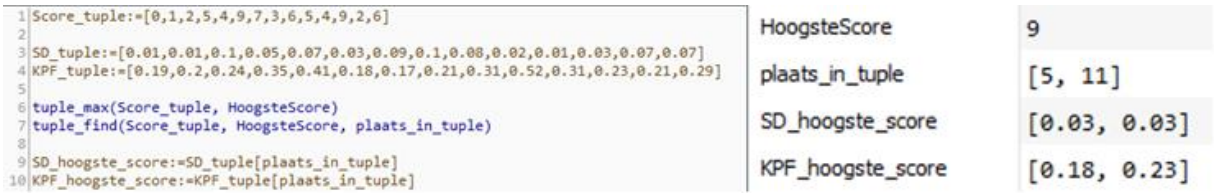
Om een bepaalde parameterset en bijhorende match(es) te kunnen beoordelen, zal er gebruik gemaakt worden van de outputs van de operator “find_surface_model”. Zoals reeds vermeld in paragraaf 4.5.1 zijn dit “Score” en “Pose”.

Om de hoogste score te bepalen, zal er eerst voor elke parameterset, de score van de eventuele match bijgehouden worden in een tuple (regel 14 in Figuur 4-10). Eens de geneste for-lussen zijn doorlopen, is het mogelijk om uit de tuple de maximale waarde te bepalen (regel 21 in Figuur 4-10). De parameterset met de hoogste score zou namelijk de beste match moeten vinden. In Figuur 4-10 is een vereenvoudigd algoritme weergegeven hoe de beste score bepaald wordt.

```
1  SurfaceModelID:='model van het object dat gezocht wordt in een scene'
2  ObjectModel3D1:='scene waarin er naar het model wordt gezocht'
3
4  teller:=0
5
6  for SD:=0 to 1 by 0.01
7      for KPF:=0 to 1 by 0.01
8
9          find_surface_model (SurfaceModelID, ObjectModel3D1, SD, KPF, 0.1, 'true', [], [], Pose, Score, SurfaceMatchingResultID)
10
11         SD_tuple[teller]:=SD
12         KPF_tuple[teller]:=KPF
13
14         Score_tuple[teller]:=Score[0]
15
16         teller:=teller+1
17
18     endfor
19 endfor
20
21 tuple_max(Score_tuple, HoogsteScore)
22 tuple_find(Score_tuple, HoogsteScore, plaats_in_tuple)
23
24 SD_hoogste_score:=SD_tuple[plaats_in_tuple]
25 KPF_hoogste_score:=KPF_tuple[plaats_in_tuple]
```

Figuur 4-10: Algoritme dat hoogste score en bijhorende parameters bepaalt

In regel 21 van bovenstaande code wordt de hoogste waarde gezocht uit de tuple Score_tuple. Deze waarde wordt bewaard in de variabele “HoogsteScore”. Vervolgens wordt er in regel 22 gezocht naar de plaats waar deze waarde zich in de tuple bevindt. Aan de hand van deze plaats is het nu mogelijk om de parameters op te vragen die verantwoordelijk zijn voor de match met de hoogste score. Dit gebeurt in regel 24 en 25. Onderstaande Figuur 4-11 geeft een praktisch voorbeeld van de hierboven beschreven code en de resultaten.



Figuur 4-11: Voorbeeld voor tupleberekeningen en bijhorende resultaten

Op deze manier is het dus mogelijk om aan de hand van de hoogste score de beste parameterset te bepalen of op te vragen. Dit bleek echter niet altijd voldoende om de beste parameterset te bepalen, zie paragraaf 4.4 Matching problemen.

Om dit probleem te voorkomen is er een positiecontrole toegevoegd aan het algoritme. Indien er een match gevonden wordt, controleert dit stuk code of de positie van de match overeenkomt met de op voorhand bepaalde positie van het object. Indien dit het geval is, wordt de score opgeslagen en anders wordt de score gelijkgesteld aan 0. Zo wordt voorkomen dat een foute match met bijhorende parameters en hoge score, gezien wordt als een goede match.

4.5.3 Trainingsscenes en testscenes

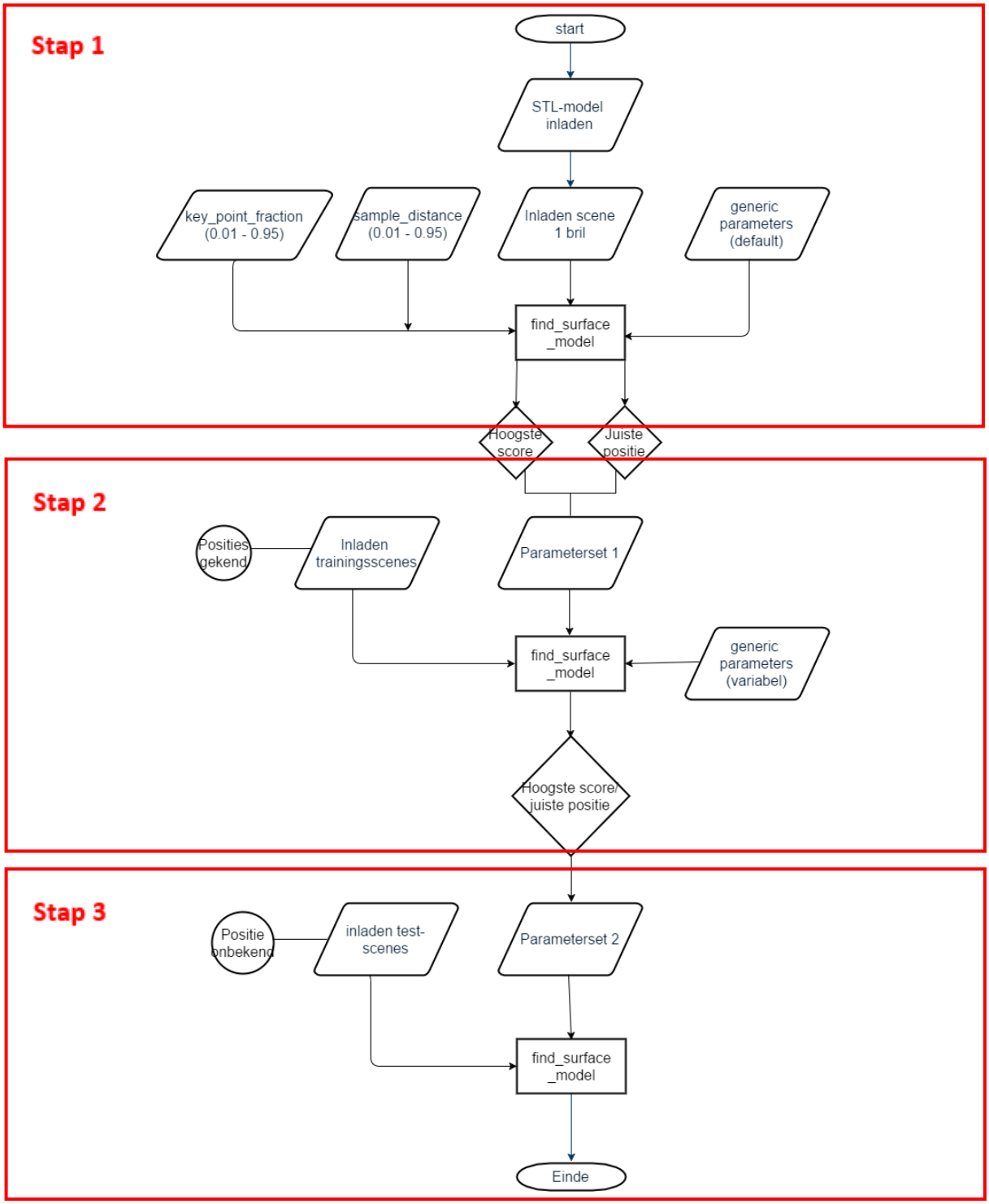
Aangezien het niet mogelijk is om een correct functionerend algoritme te maken dat volledig automatisch werkt, moest er hiervoor een alternatief gezocht worden waarbij het grootste deel toch automatisch verloopt. Dit is er gekomen door te werken met trainings- en testscenes. De benamingen van de scenes verwijzen naar de functie die ze bekleden in het algoritme. De trainingsscenes zorgen ervoor dat de parameters getraind worden, zodat de beste settings overblijven. De testscenes controleren dan of deze parameters ook voor andere scenes werken en of deze dus niet overtraind zijn. Een overtrainde parameterset betekent dat deze enkel goed werkt op de scene waarop deze getraind is en op andere scenes niet of minder goed.

In deze masterproef wordt er gebruik gemaakt van 5 trainingsscenes. Deze 5 scenes zijn een verzameling van enkele objecten, meerdere objecten en meerdere soorten type brillen. Hierop worden één voor één alle parameters uitgetest en gezocht naar de beste setting voor die scene. Aan de hand van de hierboven besproken positiecontrole en behaalde score, wordt bepaald welke setting goed of slecht is. Zodra dit voor alle vijf scenes is gebeurd, is er dus een verzameling van parametersets. Van al deze parameterwaardes voor elke soort

parameter wordt het gemiddelde genomen en gebruiken we deze gemiddelde parameterwaardes op de testscenes. De uitleg over de methode die gebruikt wordt bij de testscenes kan men terugvinden in de paragraaf 4.6.3.

4.6 *Algoritme voor automatische optimaleparameterbepaling*

Deze paragraaf legt uit hoe het algoritme in elkaar zit en werkt. De belangrijkste keuzes worden ook toegelicht. Hieronder worden de 3 grootste stappen van het algoritme één voor één uitgelegd met desbetreffende keuzes die gemaakt zijn. Om een algemeen beeld te verkrijgen is hieronder, in Figuur 4-12, de flowchart weergegeven van het gehele algoritme opgesplitst in de drie belangrijkste stappen.



Figuur 4-12: Flowchart gehele code, opgesplitst in 3 stappen

4.6.1 Stap 1: bepalen basisparameters

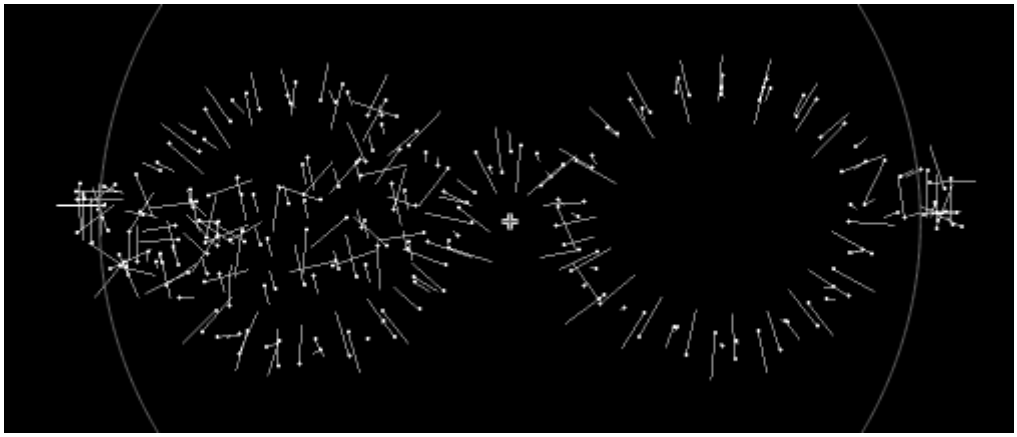
In stap 1 wordt er een STL-model van het te detecteren object ingeladen in het visieprogramma. Figuur 4-13 geeft een STL-model weer. STL staan voor **S**tandard **T**essellation **L**anguage en is een computerbestand dat speciaal ontworpen is voor de stereolithografie en dat veel wordt toegepast voor rapid prototyping [19].

Figuur 4-13: STL-model

Bij het inlezen van dit model moeten er parameters meegeven worden die niet automatisch bepaald kunnen worden. Deze parameters zijn afhankelijk van de dimensies waarin het model ontworpen is en kunnen zonder specialistische kennis ingesteld worden door de gebruiker. Enkele hiervan zijn:

- De schaal die gebruikt is voor het ontwerpen (meter, millimeter,…).
- Het al dan niet beschikken van normalen.
- De keuze of de normalen geïnverteerd moeten worden.

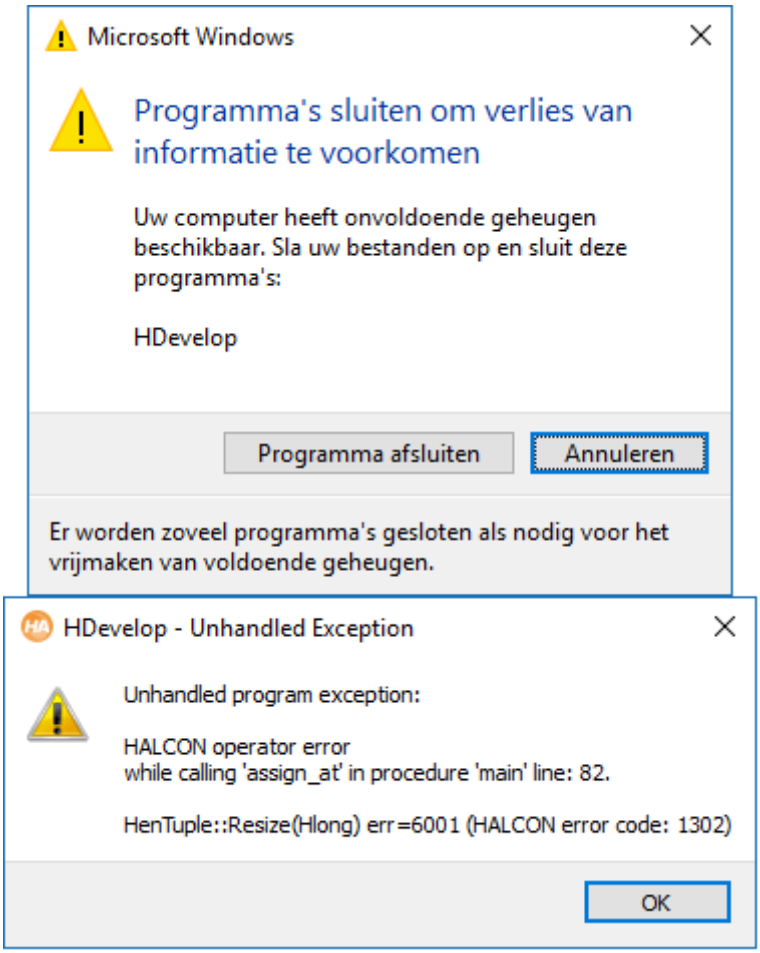
Van dit ingeladen model, wordt er een puntenwolk gemaakt op basis van de parameters die meegegeven worden. De puntenwolk van het model is weergegeven in Figuur 4-14.



Figuur 4-14: Puntenwolk van het ingeladen STL-model

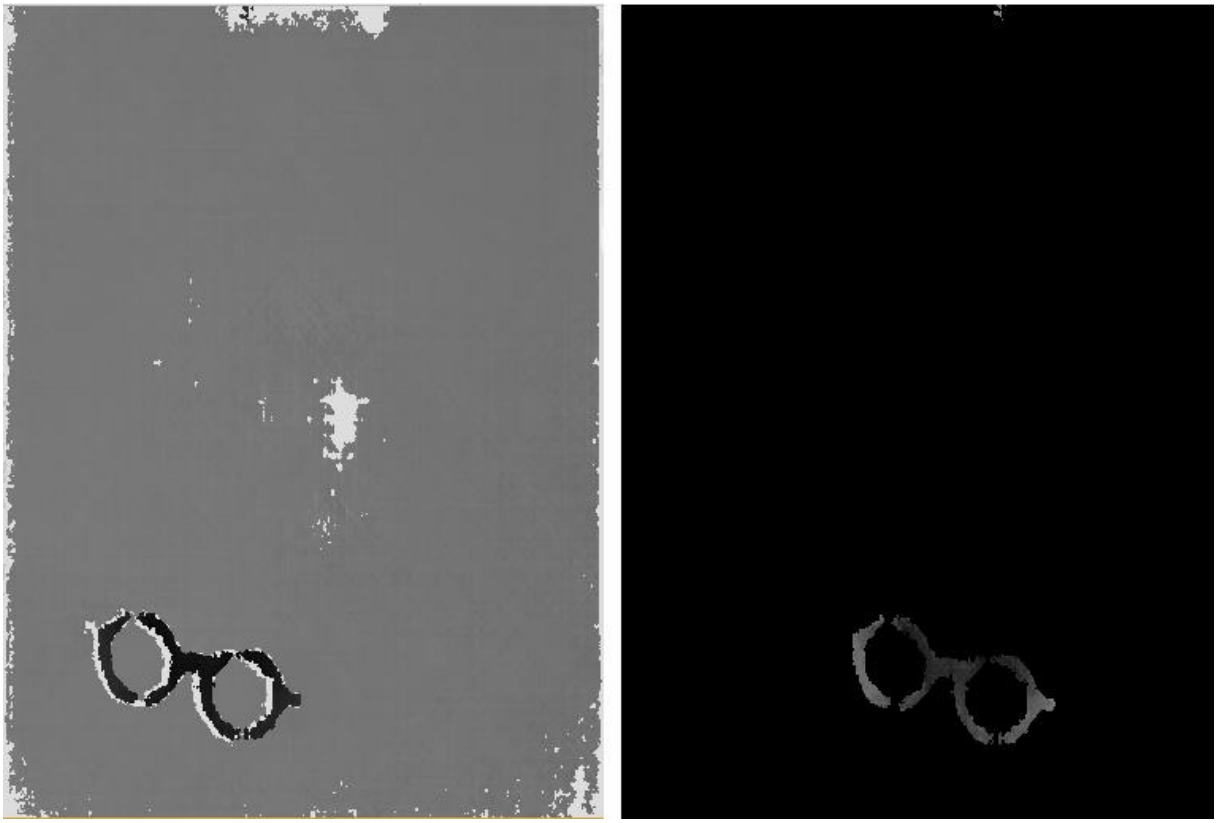
Nadat de puntenwolk is gecreëerd, wordt de eerste scene ingeladen. Voor deze scene is gekozen voor een beeld van één object dat overeenkomt met het model. Er wordt gebruik gemaakt van één object omdat hierdoor de basisparameters best (sample_distance en key_point_fraction) het bepaald kunnen worden. Het is namelijk ook niet mogelijk om zowel de basisparameters als de generische parameters op één scene te laten variëren. Er moeten

dan 3.534.400 parametercombinaties overlopen worden per scene. Dit geeft voor de gebruikte laptop geheugenproblemen op. Figuur 4-15 geeft de foutmeldingen van dit probleem weer.



Figuur 4-15: Foutmeldingen in verband met geheugentekort van enerzijds Windows en anderzijds Halcon

Vooraleer het model gebruikt kan worden om in deze scene matches te zoeken, moet er eerst een “threshold” in de hoogte uitgevoerd worden op deze scene. Dit zorgt ervoor dat de achtergrond van de scene weggewerkt wordt en dat enkel de punten overblijven die nodig zijn voor de matching. Figuur 4-16 geeft de scene voor en na de threshold weer.



Figuur 4-16: Scene enkel object voor en na uitvoeren threshold

Op deze scene worden de verplichte/belangrijkste parameters van de operator "find_surface_model" gevarieerd tussen het gehele bereik van deze parameters. Dit zijn de parameters "Sample_distance" en "Key_point_fraction". Deze variëren afwisselend tussen 0 en 0,95, zodat uiteindelijk alle mogelijke parametercombinaties worden overlopen en de beste parameterset bepaald kan worden. In Bijlage A: parametercombinaties worden de eerste 120 parametercombinaties weergegeven om een beeld te geven hoe de parameters SD en KPF variëren ten opzichte van elkaar. Tabel 4-1 geeft de waardes weer van de parameters en hun stapgrootte.

Tabel 4-1: Parameters met hun waarde en stapgrootte

Parameter	Minimale waarde	Maximale waarde	Stapgrootte
sample_distance	0,01	0,95	0,01
key_point_fraction	0,01	0,95	0,01

Het algoritme gaat nu voor elke unieke set van parameters, matches proberen vinden aan de hand van het model in een scene, respectievelijk afgebeeld in Figuur 4-14 en Figuur 4-16. De operator "find_surface_model" geeft na elke parameterset een score en een positie terug van de eventueel gevonden match(es). De score is een getal tussen 0 en 1, die de procentuele overeenkomst weergeeft van de puntenwolk van het model en de puntenwolk van de scene. Deze score wordt bijgehouden in een tuple en zodra alle parametersets zijn

overlopen, wordt de hoogste score uit deze tuple opgevraagd. De overeenkomende parameters waarbij deze score is behaald, worden eveneens opgeslagen.

Het bleek niet mogelijk om de beste parameterset te bepalen, enkel op basis van de hoogste score. De score houdt enkel rekening met de procentuele hoeveelheid overeenkomsten tussen beide puntenwolken. Hierdoor worden matches die niet goed gepositioneerd zijn soms toch gezien als goede matches. Figuur 4-17 toont bijvoorbeeld twee matches met dezelfde score, maar met andere parameterwaarden. Hier is duidelijk te zien dat de linkse match een goede match is en de rechtse match een foute (180° gedraaid).

Figuur 4-17: Goede (links) en foute (rechts) match. Rechterbril is 180° gedraaid.

Om te voorkomen dat het programma dit toch als een goede match ziet (want de behaalde score is even hoog als de score van de goede match) wordt er een positiecontrole toegevoegd aan het algoritme. De positie van het object in de scene wordt op voorhand bepaald en elke gevonden match wordt vergeleken met de juiste positie. De positie wordt als volgt opgeslagen in het visieprogramma:

Algemeen: [TransX, TransY, TransZ, RotX, RotY, RotZ, Type]

Voorbeeld: [167,407; 64,693; -77,1585; 277,879; 358,914; 180,597; 0]

Als de match niet binnen de gestelde grenzen van de juiste positie ligt, wordt de score van de match gelijkgesteld aan 0. De grenzen zijn +/-10 voor de waardes van TransX, TransY, TransZ.

Minimale waarde positie: [157.407, 54.693, -87.1585, 277.879, 358.914, 180.597, 0]

Maximale waarde positie: [177.407, 74.693, -67.1585, 277.879, 358.914, 180.597, 0]

Door deze controle toe te voegen worden enkel de parameters die een match met de hoogste score én de juiste positie opleveren, opgeslagen.

Deze parameterset(s) worden in de flowchart op Figuur 4-12 afgebeeld als “Parameterset 1”. Voor het voorbeeld met de bril(len) als object zijn de beste parametersets in Tabel 4-2 weergegeven.

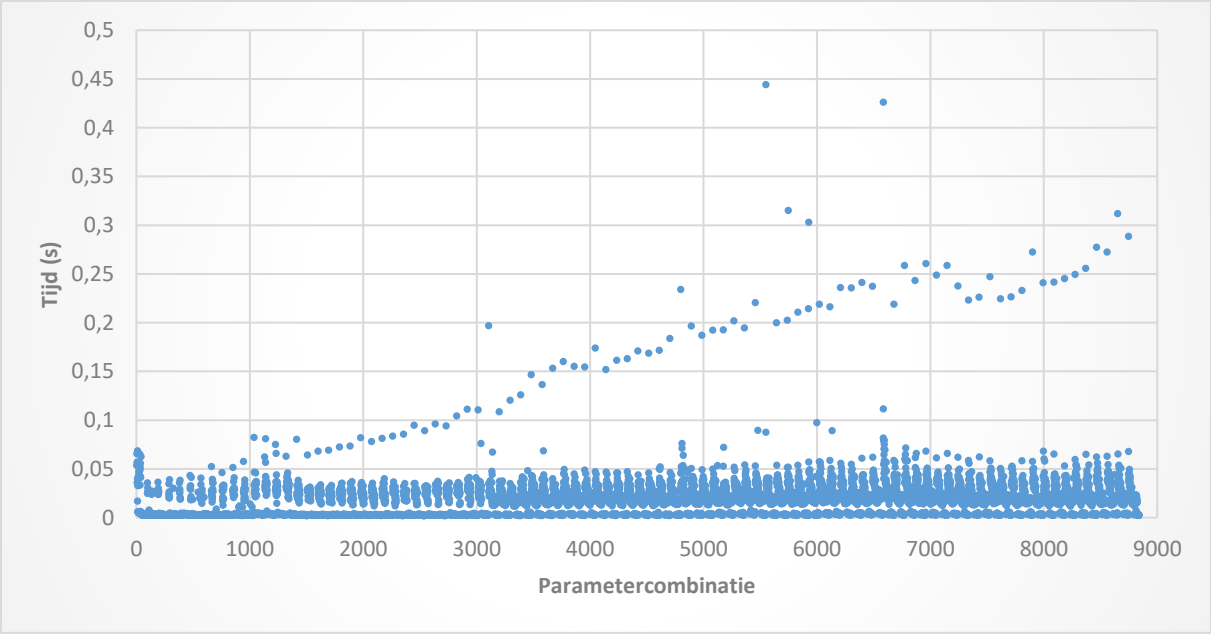
Tabel 4-2: Beste basisparameters

Parameterset	SD	KPF
1	0,07	0,17
2	0,07	0,18
3	0,04	0,19
4	0,07	0,19
5	0,07	0,2
6	0,03	0,21
7	0,03	0,23
8	0,04	0,25
9	0,07	0,39
10	0,03	0,46
11	0,07	0,54

4.6.1.1 Resultaten stap 1: bepalen basisparameters

In deze paragraaf worden de belangrijkste resultaten van de eerste stap van het algoritme weergegeven. In deze stap heeft het algoritme 8.836 (94 waardes voor sample distance * 94 waardes voor key point fraction) verschillende parametersets overlopen. De tijd die het programma hiervoor nodig had was 132 seconden.

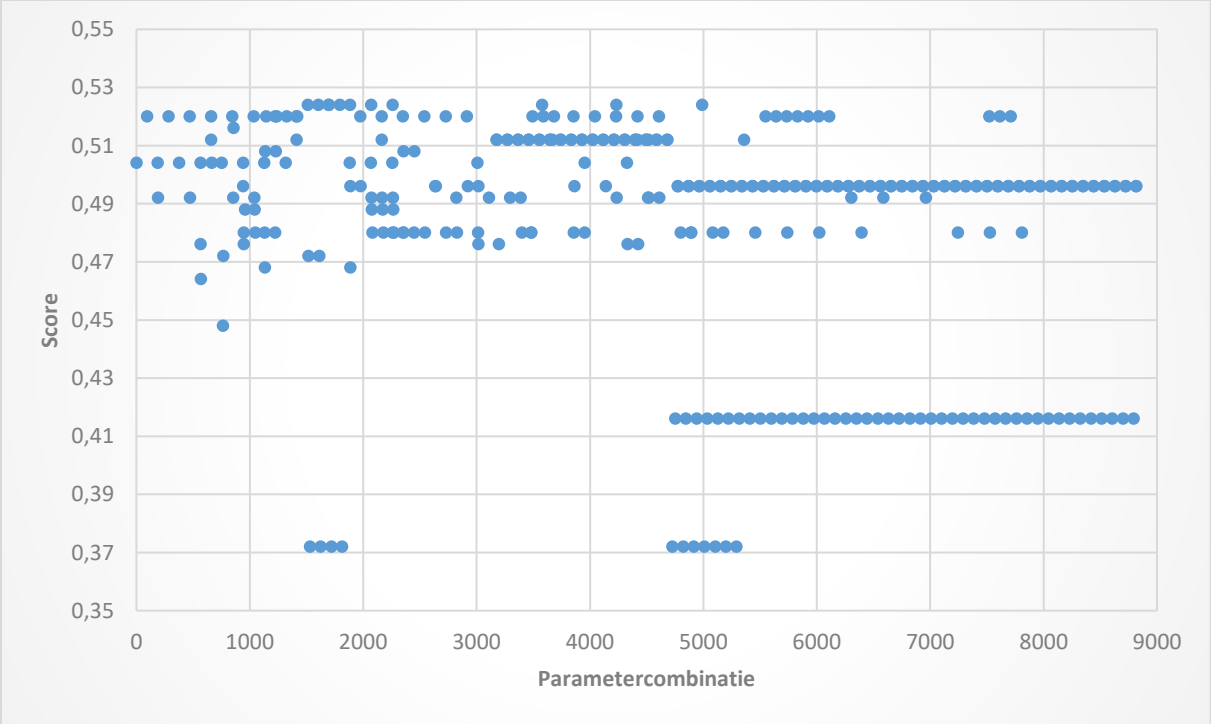
De tijd die het programma nodig had om voor elke parameterset te zoeken naar matches wordt hieronder in Figuur 4-18 weergegeven. Op de x-as worden de 8.836 verschillende parametercombinaties uitgezet en op de y-as wordt de tijd uitgezet. Dit is de tijd die het visieprogramma nodig heeft om voor elke paramatercombinatie matches te zoeken in de scene.



Figuur 4-18: Nodige tijd in functie van elke parametercombinatie

Het meest links gelegen punt op de x-as is de parameterset [sample_distance; key_point_fraction]=[0,01; 0,01]. Het meest recht gelegen punt op de x-as is de parameterset [sample_distance; key_point_fraction]=[0,95; 0,95]. In Bijlage A: parametercombinaties zijn de eerste 120 parametercombinaties met bijhorende sample_disctance en key_point_fraction opgesomd. Zo is het mogelijk om aan de hand van de parametercombinatie de bijhorende parameters te achterhalen. De punten die hiertussen gelegen zijn, liggen tussen deze waardes. Op de grafiek is te zien dat hoe lager de waardes voor beide parameters, hoe minder tijd er nodig is om naar eventuele matches te zoeken. Dit geldt wel enkel voor de uitschieters, algemeen ligt de rekentijd rond 0,05 seconden of lager.

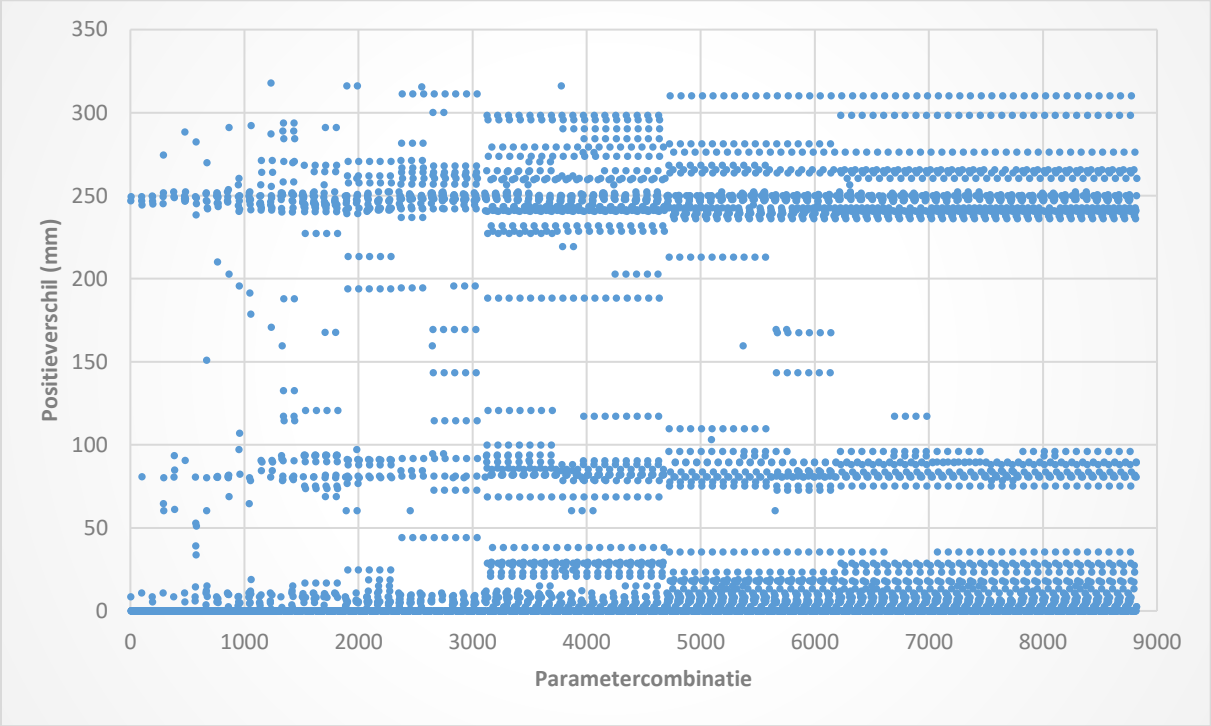
Daarnaast is het interessant om te weten welke score behaald wordt aan de hand van elke paramatercombinatie. Onderstaande Figuur 4-19 geeft een beeld weer van de score die elke parameterset behaald heeft. Wel moet er rekening gehouden worden dat dit enkel de scores zijn van de parametersets met een goede match als resultaat. Dit wil zeggen dat de positie van de gevonden match overeenkomt met de op voorhand bepaalde positie van het object. Indien er geen match of een foute match gevonden wordt, is de score gelijk aan 0. Om het overzicht te bewaren worden de scores met waarde 0 niet opgenomen in de grafiek.



Figuur 4-19: Behaalde score in functie van elke parametercombinatie

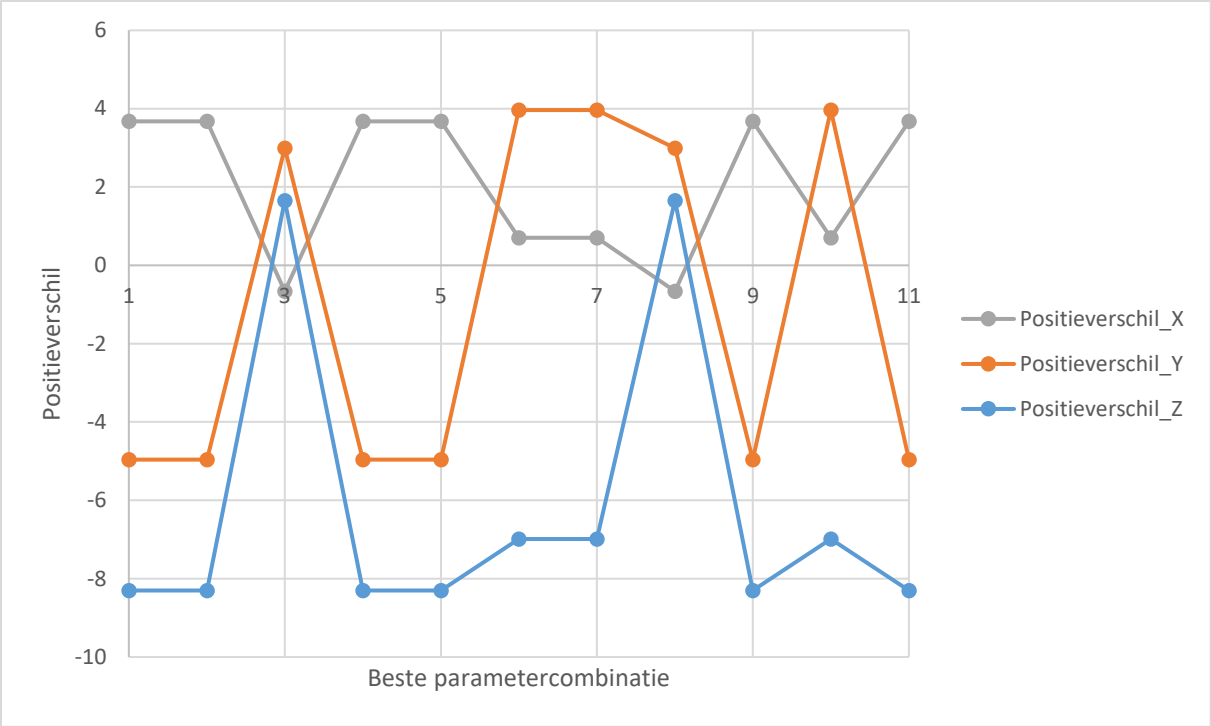
Om tot slot een beeld te krijgen van de posities van alle gevonden matches, toont Figuur 4-20 het verschil tussen de positie van de gevonden match en de op voorhand bepaalde positie van het object. Het positieverschil wordt bepaald aan de hand van formule 4.1 bepaald. Hoe verder het positieverschil verwijderd is van 0, hoe minder nauwkeurig de match gelegen is.

$$positieverschil = \sqrt{(positieverschil\ x)^2 + (positieverschil\ y)^2 + (positieverschil)^2} \quad (4.1)$$



Figuur 4-20: Positieverschil in mm tussen de gevonden matches en de op voorhand bepaalde positie van het object

De X-, Y- en Z-afwijking voor de beste parameterssets zijn in Figuur 4-21 weergegeven. Dit zijn de parameters die zowel de hoogste score hebben én de juiste positie. Zie Tabel 4-2 pagina 63. Terwijl in de vorige grafieken de resultaten van alle matches zijn weergegeven, wordt er in de grafiek die volgt enkel de beste paramatersets opgenomen.



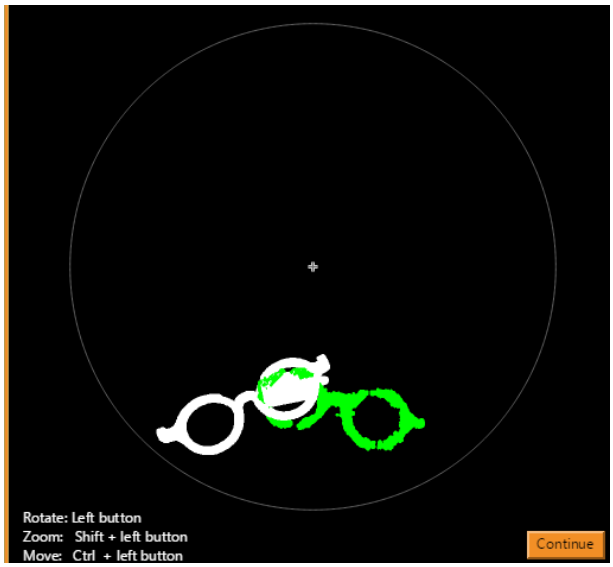
Figuur 4-21: Afwijkingen van de posities tussen gevonden match en object

Dit is een voorspelbaar resultaat aangezien de marges van de positiecontrole waren ingesteld op +/- 10 voor de X-, Y- en Z- coördinaat. Men kan concluderen dat de positiecontrole daadwerkelijk werkt zoals verwacht.

Om een beeld te krijgen van de match met de laagste score en de match waarvan het positieverschil het grootst was vergeleken met de op voorhand bepaalde juiste positie, zijn deze matches hieronder weergegeven. Figuur 4-22 geeft de match weer van de laagst behaalde score. Deze figuren zijn enkel toegevoegd uit interesse om te kijken wat daadwerkelijk de laagste score en hoogste positieverschil oplevert.

Figuur 4-22: Gevonden match met de laagste score

Deze match zou niet gevonden worden indien er met de positiecontrole zou gewerkt worden. Deze figuur is enkel een illustratie van de laagst behaalde score, zonder enkele controle. De respectievelijke score is 0,104. Figuur 4-23 geeft de match weer die het slechtste resultaat behaalde op basis van de positie.



Figuur 4-23: Gevonden match met de slechtste positie

Het spreekt voor zich dat ook hier geen positiecontrole op toegepast is. De score van deze match bedraagt 0,144.

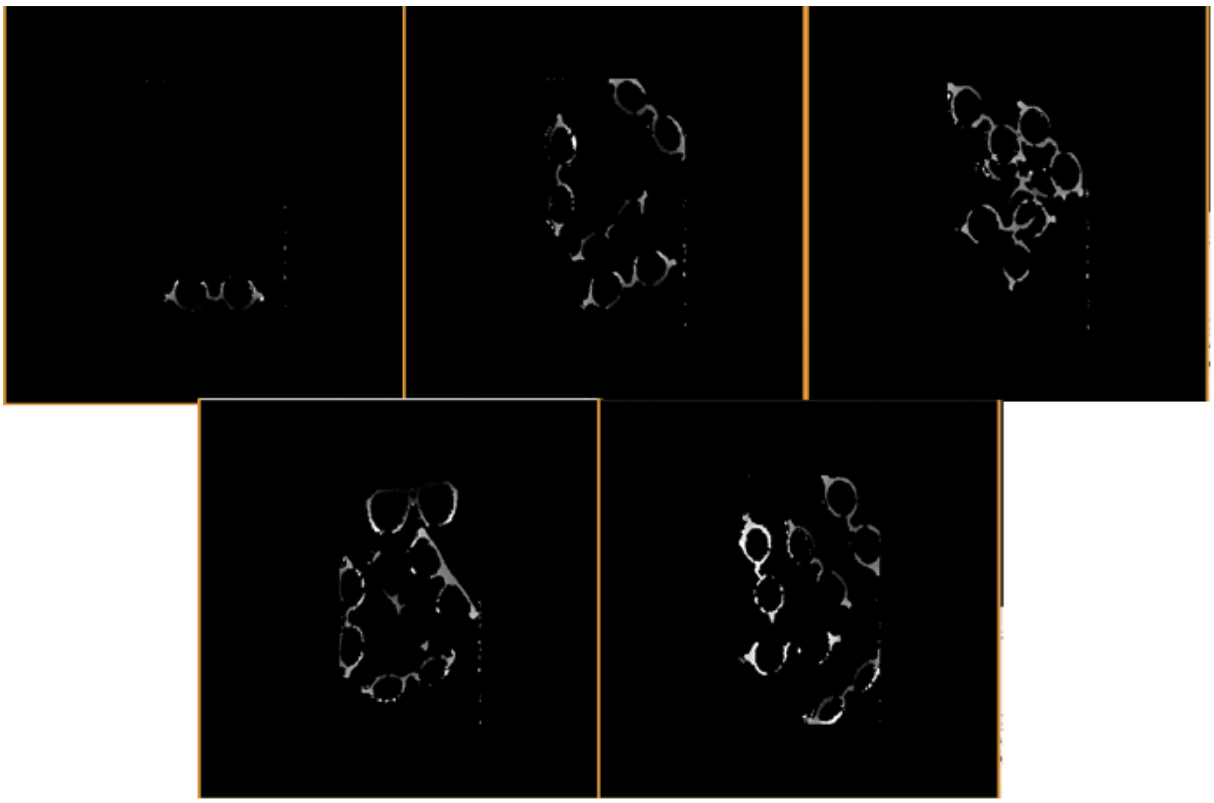
4.6.2 Stap 2: parameterbepaling met behulp van trainingsscenes

In deze stap worden er 5 scenes gebruikt waarop zowel parameterset 1 als de generische parameters zullen variëren. Deze 5 scenes zijn een verzameling van enkele objecten, meerdere objecten en meerdere soorten type brillen. In onderstaande Tabel 4-3, staan de parameters en generische parameters met hun waardes.

Tabel 4-3: Parameterset 1 + generische parameters voor trainingsscenes

Parameter	**Waarde**		
sample_distance	[0,07; 0,07; 0,04; 0,07; 0,07; 0,03; 0,03; 0,04; 0,07; 0,03; 0,07]		
key_point_fraction	[0,17; 0,18; 0,19; 0,19; 0,2; 0,21; 0,23; 0,25; 0,39; 0,46; 0,54]		
Generische parameter	**Minimale waarde**	**Maximale waarde**	**Stapgrootte**
max_overlap_distance_rel	0,1	1	0,1
pose_ref_num_steps	1	20	1
pose_ref_use_scene_normals	'true'	'false'	/

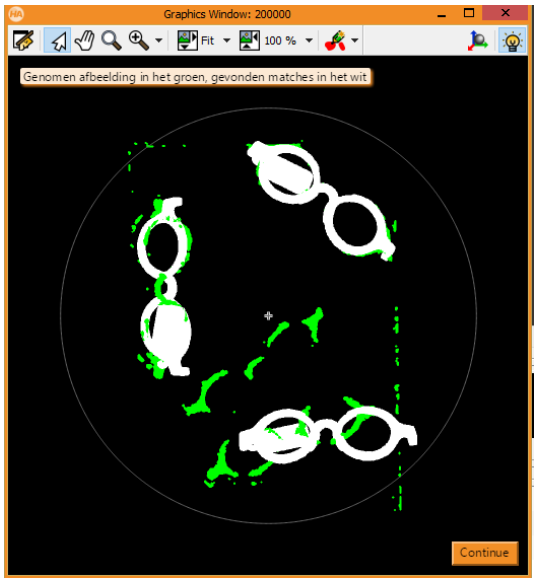
Deze 5 scenes worden de trainingsscenes genoemd, omdat op deze scenes de parameters getraind worden om uiteindelijk het beste resultaat te bekomen. Onderstaande Figuur 4-24 geeft de gebruikte trainingsscenes weer.



Figuur 4-24: 5 Trainingsscenes

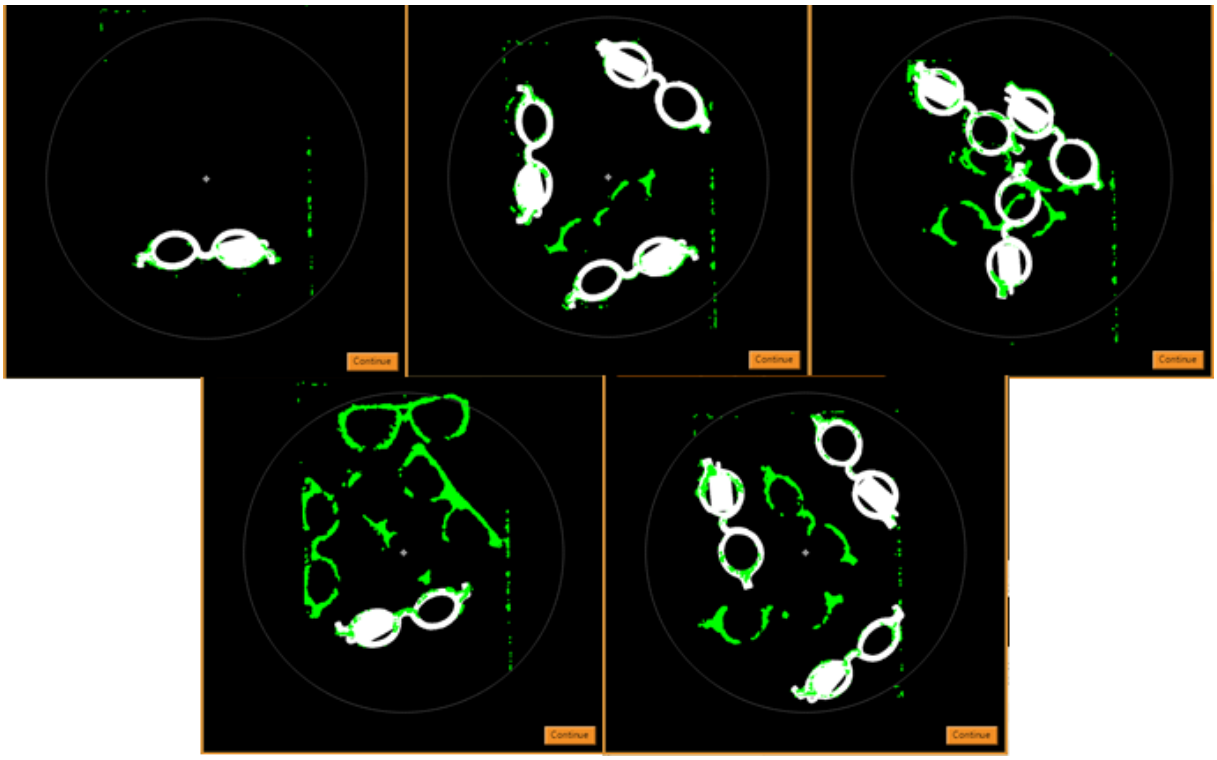
Ook voor deze scenes wordt er een threshold uitgevoerd voor het verwijderen van de achtergrond. Vervolgens worden enerzijds de parameters gebruikt die in de vorige stap het beste resultaat behaalden. Anderzijds worden nu de generische parameters, die in de vorige stap niet werden aangepast, gebruikt. Deze variëren opnieuw tussen het gehele bereik.

Ook nu wordt er gebruik gemaakt van de positiecontrole die controleert of minstens één gevonden match met één positie van een object uit de scene overeenkomt. Dit bleek echter niet voldoende te zijn. Het gebeurde al te vaak dat er één match overeenkwam met één van de op voorhand bepaalde posities, maar dat de rest van de matches niet overeenkwamen. Een voorbeeld hiervan is weergegeven in Figuur 4-25. Op deze figuur is duidelijk te zien dat de match die rechtsboven gelegen is, correct is en de 2 andere niet.



Figuur 4-25: Gevonden matches na eerste positiecontrole

Hierdoor is er een strengere positiecontrole toegevoegd die nagaat of elke gevonden match met één van de op voorhand bepaalde posities overeenkomt. Indien dit het geval is, wordt de score opgeslagen. Zodra alle parameters overlopen zijn, wordt de hoogste score met bijhorende parameters opgevraagd. Deze worden voor elke scene apart opgeslagen. De resultaten van de gevonden matches op de trainingsscenes zijn weergegeven in Figuur 4-26.



Figuur 4-26: Trainingsscenes met bijhorende matches na strengere positiecontrole

Als de scenes uit Figuur 4-26 vergeleken worden met de oorspronkelijke scenes uit Figuur 4-24, kan men vaststellen dat er enkel correcte matches gevonden zijn. Een andere

vaststelling die men kan maken, is dat niet alle mogelijke objecten gevonden zijn. De scene rechtsonder in Figuur 4-26 toont 2 objecten die niet gevonden zijn. Dit is de opportuniteitskost van de strengere positiecontrole. Enkel parametersets waarvan elke match een juiste match is, worden opgeslagen. Deze setting van parameter(s) wordt op de flowchart voorgesteld als “Parameterset 2”.

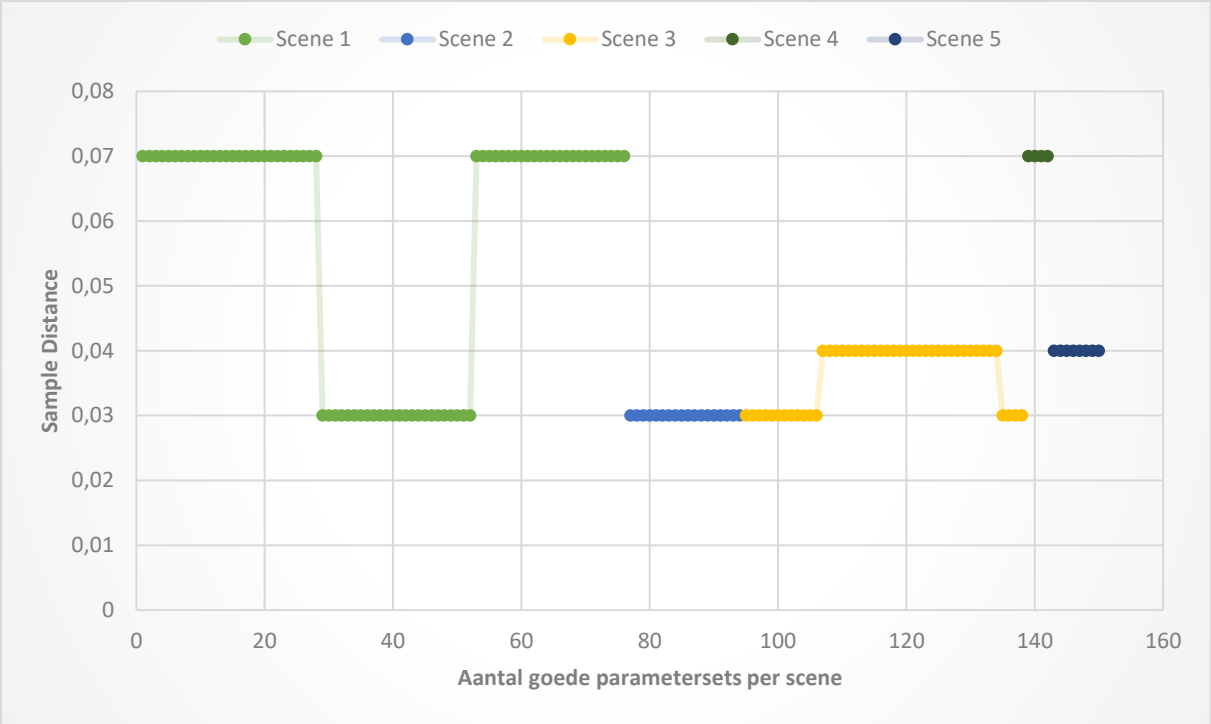
4.6.2.1 Resultaten stap 2: trainingsscenes

In deze paragraaf worden de belangrijkste resultaten van de tweede stap van het algoritme weergegeven. In deze stap heeft het algoritme 22.000 (11 verschillende basisparameters * 400 generische parametercombinaties * 5 verschillende scenes) verschillende parametersets overlopen. De tijd die het programma hiervoor nodig heeft is 2m225 seconden. Dit komt overeen met 37 minuten.

Voor elke grafiek is de naam van de parameter op de y-as weergegeven en het aantal goede parametersets per scene op de x-as. Deze zijn respectievelijk:

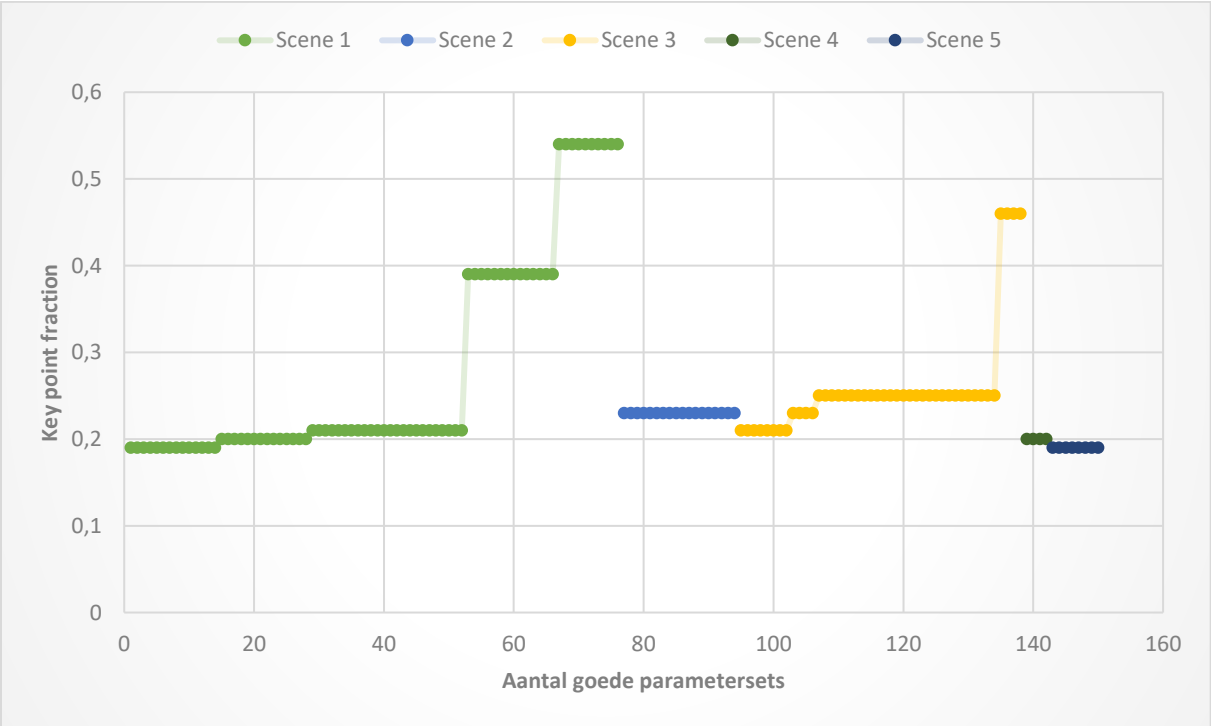
- 76 parametersets voor scene 1
- 18 parametersets voor scene 2
- 44 parametersets voor scene 3
- 4 parametersets voor scene 4
- 8 parametersets voor scene 5

Figuur 4-27 geeft de beste sample distance per scene weer. Dit betekent dat voor deze sample distance de hoogste score behaald werd in combinatie met de juiste positie(s) van het/de object(en).



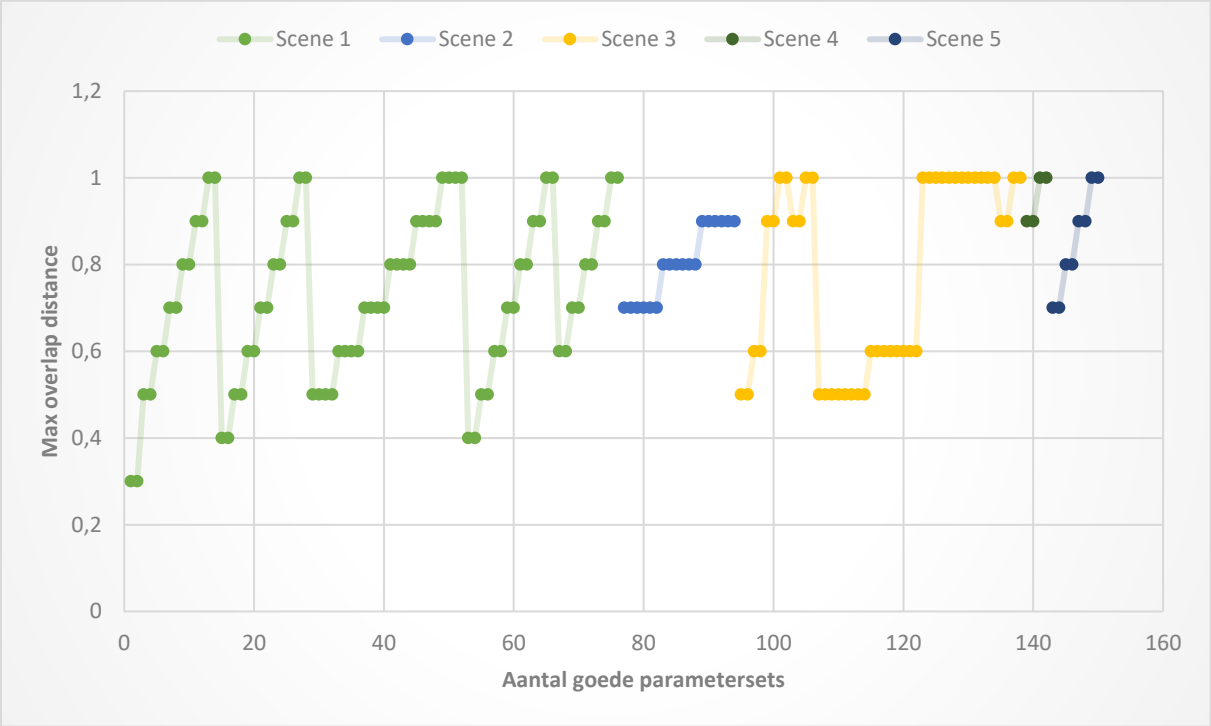
Figuur 4-27: Beste sample distance voor elke trainingsscene

Het mag niet verbazen dat de beste SD's altijd 0,03; 0,04 of 0,07 zijn, aangezien deze in de vorige stap bepaald waren en zijn weergegeven in Tabel 4-2 pagina 63 en Tabel 4-3 pagina 67. Ook de KPF zijn in de vorige stap bepaald en zullen in deze stap een element zijn van de waardes uit Tabel 4-2 en Tabel 4-3. Figuur 4-28 geeft de beste key point fraction per scene weer.



Figuur 4-28: Beste key point fraction voor elke trainingsscene

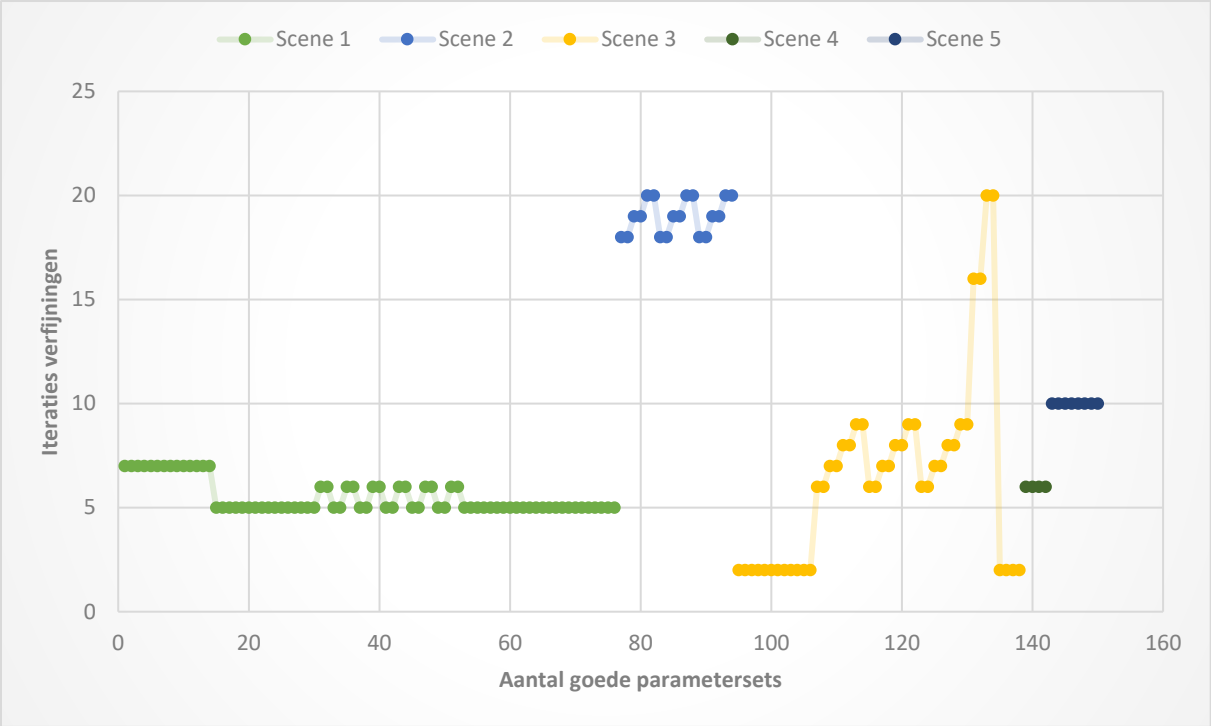
Figuur 4-29 geeft de beste generische parameter max_overlap_distance weer per scene. Deze wordt in stap 2 wel gevarieerd, dus kan alle waarden van het bereik aannemen. Deze parameter bepaalt of er al dan niet overlapping mag zijn tussen verschillende matches en hoe groot de eventuele overlapping mag zijn.



Figuur 4-29: Beste max overlap distance voor elke trainingsscene

Wanneer men de resultaten van Figuur 4-29 vergelijkt met de scenes uit Figuur 4-24 pagina 68, ziet men dat scene 1, 2, 4 en 5 geen overlappende objecten bevat. In tegenstelling tot scene 3 die wel overlapping bevat. Daar waar voor scene 1, 2, 4 en 5 de beste parameterwaardes voor max_overlap trapsgewijs verlopen is dit voor scene 3 niet het geval. Voor scenes waar geen overlapping is, zijn de waardes van max_overlap niet zo belangrijk, terwijl dit voor de scene met overlapping wel het geval is.

Tot slot is in Figuur 4-30 de beste generische parameter pose_ref_nem_steps weergegeven per scene. Ook deze wordt in stap 2 gevarieerd en kan dus alle waardes van het bereik aannemen. Deze generische parameter bepaalt hoeveel keer de positieverfijning herhaald wordt om de positie optimaal te bepalen.



Figuur 4-30: Aantal herhalingen die gebruikt worden voor de positiebepaling

Ook hier vertoont de curve die bij scene 3 hoort een ander karakter door de overlapping als de overige scenes zonder overlapping.

Voor de generische parameter pose_ref_use_scene_normals is er geen grafiek opgesteld omdat de waarde hiervan even vaak true als false was. Deze had dus geen invloed op de gebruikte trainingsscenes. Ook de generische parameters dense_pose_refinement en sparse_pose_refinement zijn niet in een grafiek verwerkt omdat deze altijd de waarde true hebben voor deze testscenes.

4.6.3 Stap 3: controle parameters aan de hand van testscenes

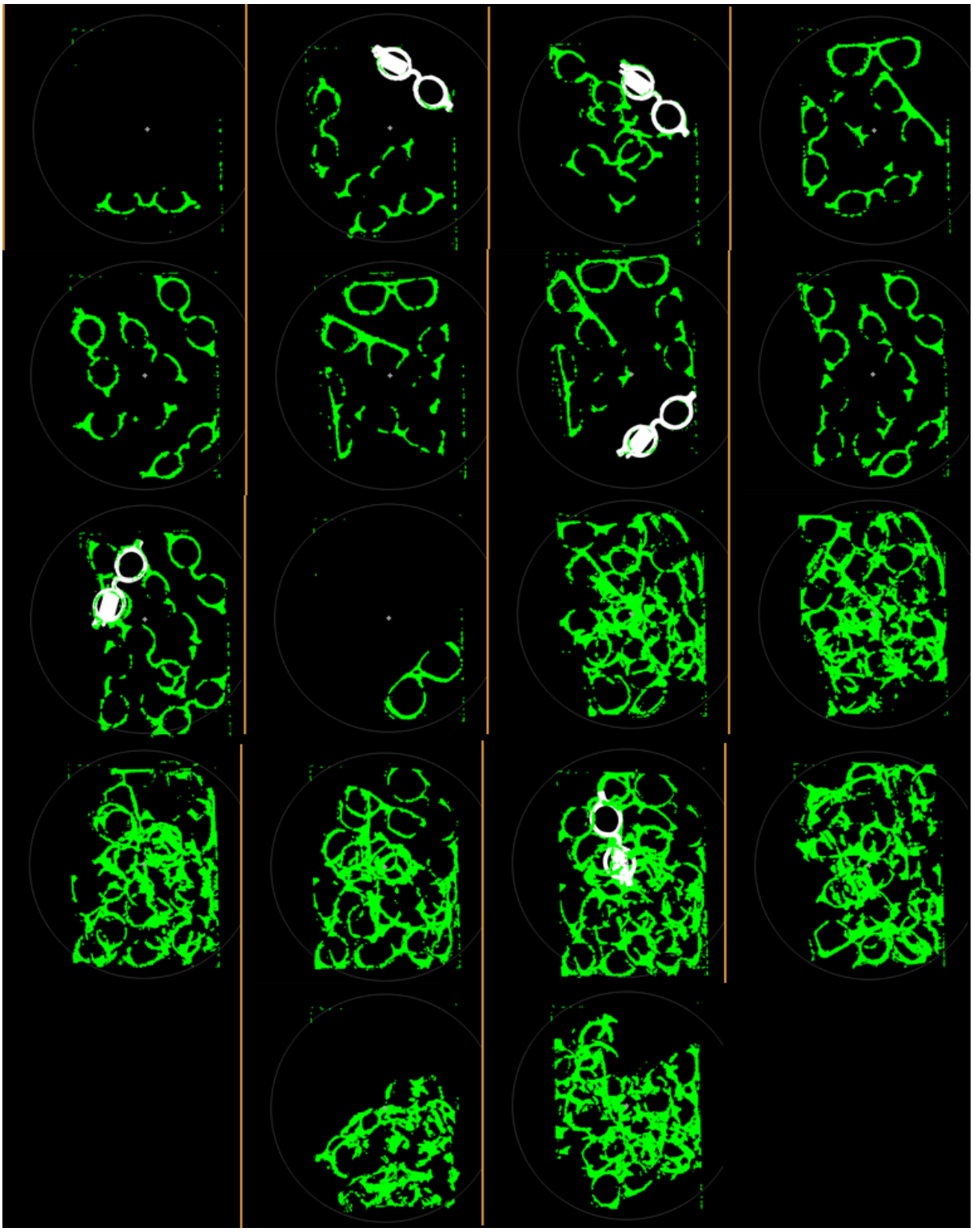
In de laatste stap van het algoritme worden testscenes gebruikt om te controleren of de getrainde parameterset(s) ook voor goede resultaten zorgen op ongeziene scenes waarbij er geen enkele vorm van controle wordt uitgevoerd. Van alle parameters die opgeslagen zijn uit voorgaande stap, wordt voor elke soort parameter apart het gewogen gemiddelde genomen. Het gewogen gemiddelde zorgt ervoor dat de parametersets die meer goede matches vinden, zwaarder doorwegen. Deze waardes worden nu in de operator “find_surface_model” gebruikt om in de testscenes naar matches te zoeken. De minimale score van de te zoeken matches wordt gelijkgesteld aan het gemiddelde van alle scores die behaald zijn bij de trainingsscenes.

Om een beter beeld te creëren van de waarde van de parameters van de operator, is hieronder een samenvatting weergegeven in Tabel 4-4.

Tabel 4-4: Parameters + waardes

Parameter	Waarde
Gemiddelde_sample_distance	0,0473
Gemiddelde_key_point_fraction	0,2617
Gemiddelde_max_overlap_distance	0,764
Gemiddelde_pose_ref_num_steps	7,6667
Gemiddelde_pose_ref_use_normals	true
Minimale_score	0,3768

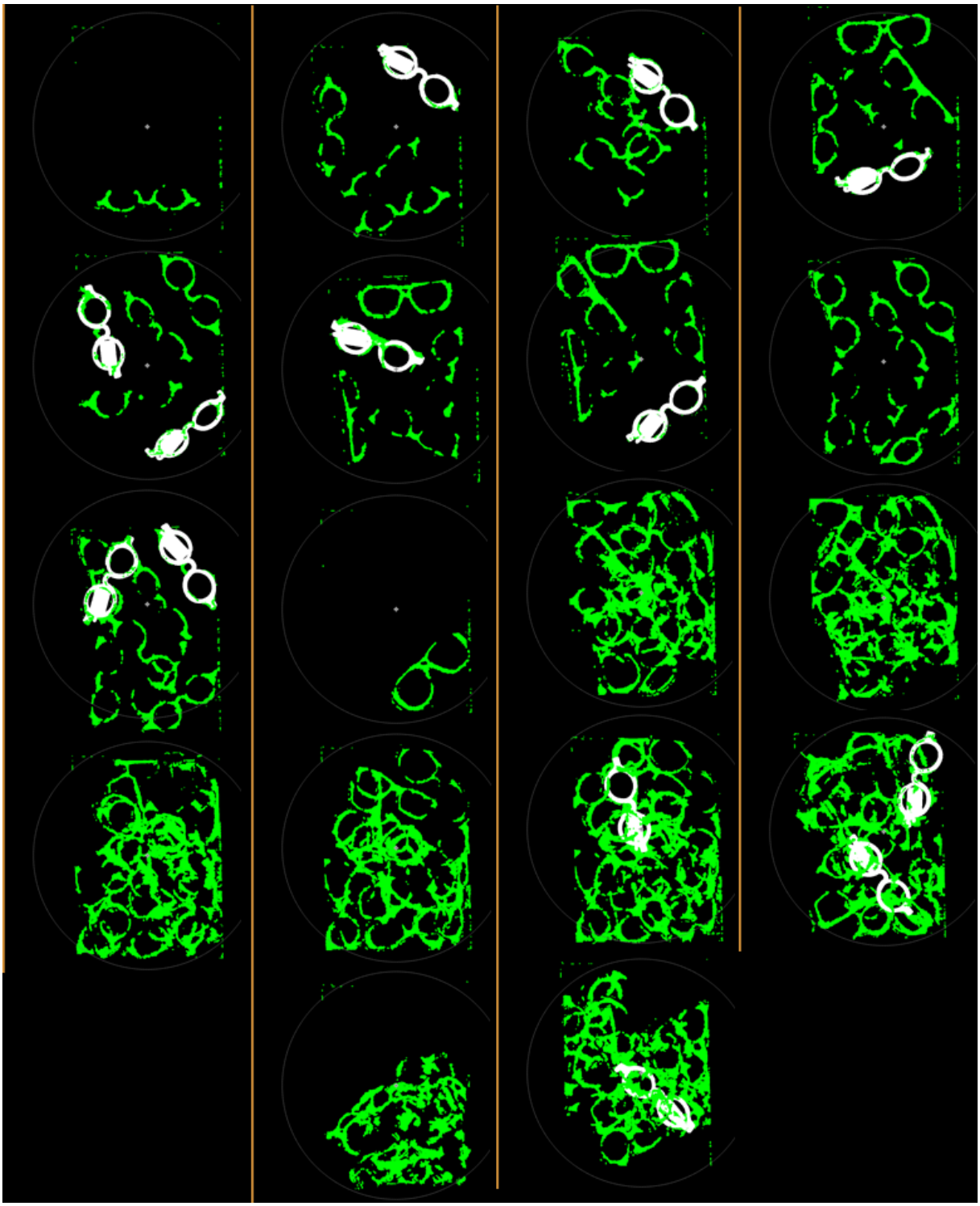
Deze waardes worden gebruikt voor de testscenes. Hieronder worden de resultaten van de testscenes weergegeven in Figuur 4-31.



Figuur 4-31: Testscenes en bijhorende matches bij gemiddelde parameters

Op bovenstaande afbeelding, Figuur 4-31, is te zien dat er 5 matches gevonden zijn, waarvan er 4 juiste en 1 foute. Omdat er relatief weinig matches gevonden worden op deze manier, werd er naar een manier gezocht waarop er meer matches gevonden zouden worden en het aantal foute matches toch relatief laag bleef. De eerste optie was de minimale score te verlagen. Hierdoor zullen er meer matches gevonden worden omdat de overeenkomsten

tussen het model en het gevonden object in de scene minder hoog moeten zijn. Alle parameters die weergegeven zijn in Tabel 4-4 worden behouden, behalve de minimale score. Deze wordt verlaagd tot 0,2768. In Figuur 4-32, zijn de resultaten weergegeven met deze parameters.



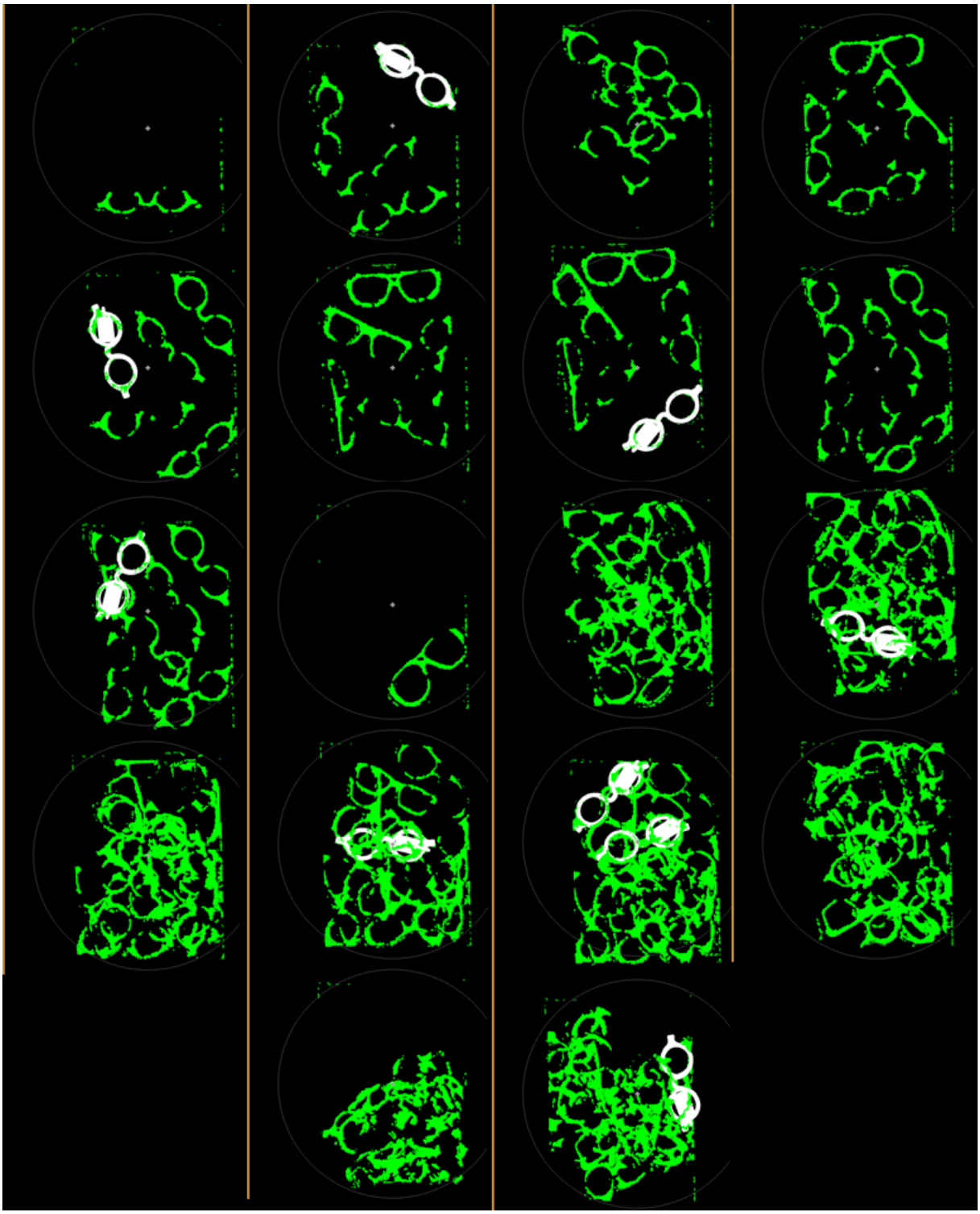
Figuur 4-32: Testscenes en bijhorende matches bij gemiddelde parameters en verlaagde score

Op deze manier worden er zoals verwacht meer matches gevonden. In totaal worden er 13 matches gevonden, waarvan 7 correcte matches en 6 foute. Dit is geen gunstig resultaat aangezien het aantal foute matches meer stijgt dan het aantal goede matches. Een oplossing voor dit probleem was om de parameters van de operator te laten variëren rond de gemiddelde waardes. Doordat op die manier het bereik van de parameters groter werd, werd ook de kans groter dat er matches gevonden zouden worden, zonder een grote toename in foute matches. In onderstaande Tabel 4-5 zijn de nieuwe waarden van de parameters weergegeven. De minimale score wordt ook terug gelijkgesteld aan de gemiddelde waarde.

Tabel 4-5: Parameters met nieuwe minimale en maximale waarde + stapgrootte voor testscenes

Parameter	Minimale waarde	Maximale waarde	Stapgrootte
Gemiddelde_sample_distance	0,0373	0,0573	0,01
Gemiddelde_key_point_fraction	0,2117	0,3117	0,01
Gemiddelde_max_overlap_distance	0,1640	0,7640	0,1
Gemiddelde_pose_ref_num_steps	2,6667	8,6667	1
Gemiddelde_pose_ref_use_normals	true		/
Minimale_score	0,3768		/

Door deze methode toe te passen werden er meer matches gevonden. De resultaten van deze methode zijn weergegeven in Figuur 4-33. In totaal zijn er 7 goede matches gevonden en 2 foute matches. Aan de hand van deze methode zijn er dus 3 goede matches en slechts 1 foute match meer gevonden vergeleken met de resultaten in Figuur 4-31.



Figuur 4-33: Gevonden matches door het variëren van de parameters rond de gemiddelde waardes

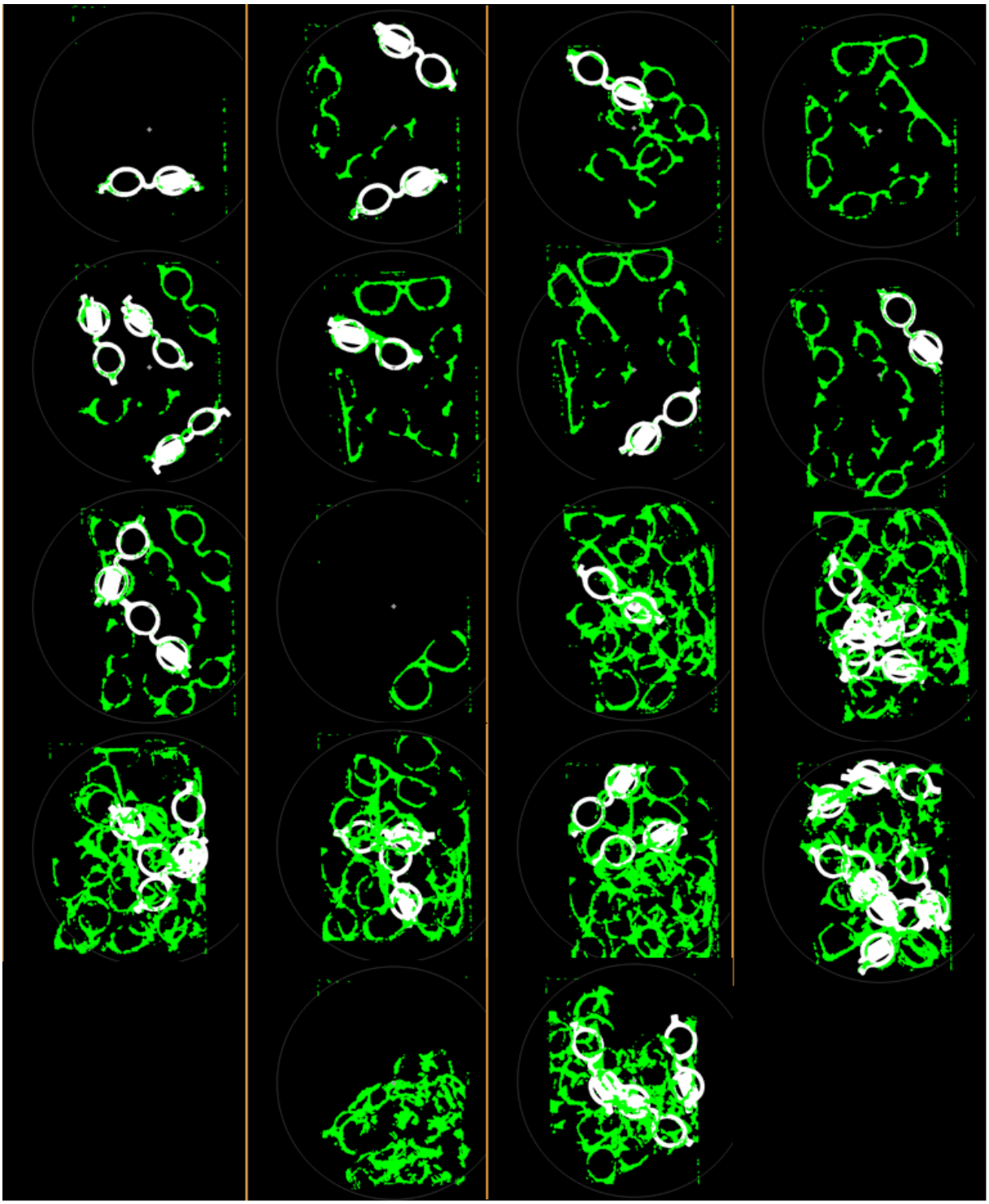
Een andere methode die werd getest om meer matches te vinden, was door de minimale score voor de matches te verlagen. Doordat de score verlaagd wordt, zijn er meer matches die gevonden worden. Een gevolg hiervan is dat er ook meer foute matches gevonden gaan worden, want de minimale overeenkomst daalt door het verlagen van de score. Dit is niet

gunstig. In onderstaande Tabel 4-6 is een overzicht gegeven van de parameters en score die voor deze methode gebruikt zijn.

Tabel 4-6: Parameters met nieuwe minimale en maximale waarde + stapgrootte met verlaagde score voor testscenes

Parameter	Minimale waarde	Maximale waarde	Stapgrootte
Gemiddelde_sample_distance	0,0373	0,0573	0,01
Gemiddelde_key_point_fraction	0,2117	0,3117	0,01
Gemiddelde_max_overlap_distance	0,1640	0,7640	0,1
Gemiddelde_pose_ref_num_steps	2,6667	8,6667	1
Gemiddelde_pose_ref_use_normals	true		/
Minimale_score	0,2768		/

De parameters zijn hetzelfde gebleven als in Tabel 4-5, enkel de minimale_score is verlaagd met 0,1. De resultaten van deze methode zijn in Figuur 4-34 weergeven.



Figuur 4-34: Gevonden matches door het variëren van de parameters rond de gemiddelde waardes en verlaagde minimale score

Zoals verwacht worden er meer matches gevonden. Echter stijgt het aantal foute matches meer dan de goede matches. Er worden namelijk 13 goede matches gevonden en 24 foute matches. Dit is absoluut niet gunstig voor praktijk gerelateerde cases. Het is beter om minder matches te vinden, maar het merendeel goede matches dan andersom.

Om een overzicht van elke methode te geven met de behaalde resultaten, is in Tabel 4-7 een overzicht gegeven voor elke methode. Methode 1 en 3 behalen ongeveer hetzelfde percentage (# goede matches/# matches), met het verschil dat methode 3 meer matches behaald. Methode 2 en 4 behalen vooral veel foute matches. Hiervoor is er gekozen om methode 3 te gebruiken in het algoritme.

Tabel 4-7: Overzicht resultaten per methode

	Methode 1	Methode 2	Methode 3	Methode 4
Aantal goede matches	4	7	7	13
Aantal foute matches	1	6	2	24
Totaal aantal matches	5	13	9	37
Percentage	80	53,85	77,78	35,14

4.7 Besluit

Ondanks er goede resultaten bekomen worden met het huidig algoritme, is er altijd ruimte voor verbetering. In deze paragraaf worden enkele punten aangehaald waarbij de mogelijkheid bestaat dat er betere resultaten behaald zullen worden.

Het eerste verbeterpunt gaat omtrent het model. Zoals in afbeeldingen Figuur 4-13 en Figuur 4-14 is weergegeven bevat het model een label. Dit label wordt samen met de bril 3D-geprint. Op dit label kunnen eigenschappen van de bril staan, zoals kleur, maat,… Door dit label worden er echter punten opgenomen in de puntenwolk die nooit overeen zullen komen met de scenes. De objecten in de scenes hebben namelijk geen label en het was niet mogelijk om het label uit de puntenwolk te verwijderen. Door een model te gebruiken waarbij het label niet aanwezig is, zouden er enerzijds matches gevonden worden met een hogere score en anderzijds minder foute matches.

Een tweede mogelijkheid tot het verbeteren van de resultaten is het gebruiken van scenes gemaakt aan de hand van een sheet-of-lightopstelling. Daar waar er nu gebruik gemaakt wordt van een opstelling op basis van stereovisie. Oorspronkelijk was het de bedoeling om beide opstellingen te vergelijken met elkaar. Door een defect aan de sheet-of-lightopstelling, is dit echter niet kunnen gebeuren. De kwaliteit van een scene gemaakt door een SOL-opstelling is beter en het onderscheid tussen verschillende objecten is duidelijker. Hierdoor zullen er minder foute matches gevonden worden, waar dit nu wel gebeurt bij overlappende objecten.

Tot slot zou de tijd die het algoritme nodig heeft om de optimale parameters te bepalen, korter mogen. Doordat het programma voor elke mogelijk waarde van een bepaalde parameter, moet controleren of dit een goede waarde is, duurt het een uur vooraleer het algoritme doorlopen is. Een kanttekening hierbij is wel dat indien een persoon dit zou moeten doen, minstens even lang, zo niet langer bezig zou zijn voor het instellen van de

parameters. Het programma doet dit volledig automatisch, waardoor er niemand nodig is en hierbij nog altijd tijd uitgespaard wordt.

5 Besluit

Door de doelstellingen die in de inleiding beschreven zijn te toetsen aan de realisaties, wordt het mogelijk een conclusie te maken van deze masterproef. Dit wordt in de paragraaf hieronder besproken.

De hoofddoelstelling was het creëren van een algoritme dat automatisch de optimale parameterset zou genereren. De tijd heeft echter geleerd dat de combinatie van beide niet geheel mogelijk was. Indien er een volledig automatisch algoritme gemaakt zou worden, zou er geen garantie zijn dat de gegenereerde parametersets effectief de beste waren. Om dit probleem te bestrijden, moest er een toegeving gemaakt worden op gebied van automatisatie. Die is er gekomen door te werken met trainingsscenes en testscenes. De objecten in de trainingsscenes werden op voorhand bepaald en de gevonden matches werden met deze posities gecontroleerd. Aangezien de posities op voorhand bepaald moeten worden, kan er niet gesproken worden over een volledig automatisch algoritme. Dit is echter een kleine factor in het gehele algoritme.

Er kan dus besloten worden dat de vooropgestelde doelstellingen behaald zijn en dat er een automatisch algoritme is gecreëerd dat de optimale parameters bepalen in een visiesysteem. Om dit extra kracht bij te zetten, was het ideaal geweest als dit algoritme ook op andere objecten had kunnen werken in plaats van enkel te werken met de brilmonturen. Hier was echter niet genoeg tijd voor en de nodige CAD/STL-bestanden van andere objecten waren niet voorhanden. Daar het algoritme niet objectafhankelijk is en dus alle mogelijke parametercombinaties overloopt, kunnen we met enige zekerheid stellen dat dit ook zal werken voor andere objecten.

In de toekomst is het mogelijk om de kwaliteit van de resultaten van het algoritme te verbeteren door gebruik te maken van een betere cameraopstelling. Verder onderzoek moet uitwijzen of de scenes die gemaakt worden met een sheet-of-lightopstelling de kwaliteit van het algoritme effectief verbeteren.

Bibliografie

[1] ACRO, „Onderzoeksgroep ACRO,” 5 Oktober 2016. [Online]. Available: http://iiw.kuleuven.be/onderzoek/acro/about.

[2] J. Baeten, K. Donné, S. Boedrij, W. Beckers en E. Claesen, „Autonomous Fruit Picking Machine: A Robotic Apple Harvester,” *HAL - Inria,* p. 10, 2007.

[3] G. Appenzeller en J. Crowley, „Automatic parameter control for experimental evaluation of vision systems,” p. 8.

[4] E. Yeguas, M. Luzón, R. Pavón, R. Laza, G. Arroyo en F. Díaz, „Automatic parameter tuning for evolutionary algorithms using a Bayesian case-based reasoning system,” *Elsevier,* p. 10, 2014.

[5] C. E. Cheung, J. Wong, J. Chan en J. Pan, „Optimization-based automatic parameter tunig for stereo vision,” *IEEE,* p. 7, 2015.

[6] J. M. Casado-Díaz, L. Martínez-Bernabéu en F. Floréz-Revuelta, „Automatic parameter tuning for functional regionalisation methods,” *ResearchGate,* p. 26, 2016.

[7] M. S. Gmbh, „Documentation for Halcon - HDevelop User's Guide,” [Online]. Available: http://www.mvtec.com/fileadmin/Redaktion/mvtec.com/documentation/halcon/halcon-13.0-hdevelop-users-guide.pdf. [Geopend 20 September 2016].

[8] M. H. Gmbh, „Documentation for Halcon - Solution Guide II-B - Matching,” [Online]. Available: http://www.mvtec.com/fileadmin/Redaktion/mvtec.com/documentation/halcon/halcon-13.0-solution-guide-ii-b-matching.pdf. [Geopend 20 september 2016].

[9] M. Ulrich en C. Steger, „System and methods for automatic parameter determination in machine vision,” Mvtec Software Gmbh, 2011.

[10] D. Nair, „A Guide to Stereovision and 3D Imaging,” 1 oktober 2012. [Online]. Available: http://www.techbriefs.com/component/content/article/14925?start=1. [Geopend 6 december 2016].

[11] „Application programming interface,” 13 november 2016. [Online]. Available: https://nl.wikipedia.org/wiki/Application_programming_interface. [Geopend 10 januari 2017].

[12] Ensenso, „Specifications N35 Series,” [Online]. Available: http://www.ensenso.com/support/modellisting/?id=N35-602-16-IR. [Geopend 6 december 2016].

[13] C. Genesis, „Featuring: Speech Bubbles for Kinect WakeUpAndCode.com.,” 15 februari 2015. [Online]. Available: http://slideplayer.com/slide/3357546/. [Geopend 10 januari 2017].

[14] C. Rogue, „Everything Kinect in one place,” [Online]. Available: http://123kinect.com/everything-kinect-2-one-place/43136/. [Geopend 6 december 2016].

[15] „The Principle of Laser-Triangulation or Sheet of Light,” [Online]. Available: http://www.automationtechnology.de/cms/en/cx-technologie/. [Geopend 6 december 2016].

[16] Photonfocus, „Datasheet MV1-D1312-160-CL,” [Online]. Available: http://www.photonfocus.com/products/camerafinder/camera/?no_cache=1&prid=1. [Geopend 8 december 2016].

[17] A. V. S. Inc., „Pentax Cosmicar : C21211KP (RICOH FL-BC1214D-VG),” [Online]. Available: http://www.avsupply.com/ITM/1540/c21211kp-b1214d-2.html. [Geopend 11 januari 2017].

[18] S. Kreuznach, „Bandpass Filters,” april 2012. [Online]. [Geopend 11 januari 2017].

[19] „STL-bestand,” 8 December 2014. [Online]. Available: https://nl.wikipedia.org/wiki/STL-bestand. [Geopend 10 januari 2017].

Bijlagen

Bijlage A: parametercombinaties

Combinatie	SD	KPF						
1	0,01	0,01	**41**	0,41	0,01	**81**	0,81	0,01
2	0,02	0,01	**42**	0,42	0,01	**82**	0,82	0,01
3	0,03	0,01	**43**	0,43	0,01	**83**	0,83	0,01
4	0,04	0,01	**44**	0,44	0,01	**84**	0,84	0,01
5	0,05	0,01	**45**	0,45	0,01	**85**	0,85	0,01
6	0,06	0,01	**46**	0,46	0,01	**86**	0,86	0,01
7	0,07	0,01	**47**	0,47	0,01	**87**	0,87	0,01
8	0,08	0,01	**48**	0,48	0,01	**88**	0,88	0,01
9	0,09	0,01	**49**	0,49	0,01	**89**	0,89	0,01
10	0,1	0,01	**50**	0,5	0,01	**90**	0,9	0,01
11	0,11	0,01	**51**	0,51	0,01	**91**	0,91	0,01
12	0,12	0,01	**52**	0,52	0,01	**92**	0,92	0,01
13	0,13	0,01	**53**	0,53	0,01	**93**	0,93	0,01
14	0,14	0,01	**54**	0,54	0,01	**94**	0,94	0,01
15	0,15	0,01	**55**	0,55	0,01	**95**	0,01	0,02
16	0,16	0,01	**56**	0,56	0,01	**96**	0,02	0,02
17	0,17	0,01	**57**	0,57	0,01	**97**	0,03	0,02
18	0,18	0,01	**58**	0,58	0,01	**98**	0,04	0,02
19	0,19	0,01	**59**	0,59	0,01	**99**	0,05	0,02
20	0,2	0,01	**60**	0,6	0,01	**100**	0,06	0,02
21	0,21	0,01	**61**	0,61	0,01	**101**	0,07	0,02
22	0,22	0,01	**62**	0,62	0,01	**102**	0,08	0,02
23	0,23	0,01	**63**	0,63	0,01	**103**	0,09	0,02
24	0,24	0,01	**64**	0,64	0,01	**104**	0,1	0,02
25	0,25	0,01	**65**	0,65	0,01	**105**	0,11	0,02
26	0,26	0,01	**66**	0,66	0,01	**106**	0,12	0,02
27	0,27	0,01	**67**	0,67	0,01	**107**	0,13	0,02
28	0,28	0,01	**68**	0,68	0,01	**108**	0,14	0,02
29	0,29	0,01	**69**	0,69	0,01	**109**	0,15	0,02
30	0,3	0,01	**70**	0,7	0,01	**110**	0,16	0,02
31	0,31	0,01	**71**	0,71	0,01	**111**	0,17	0,02
32	0,32	0,01	**72**	0,72	0,01	**112**	0,18	0,02
33	0,33	0,01	**73**	0,73	0,01	**113**	0,19	0,02
34	0,34	0,01	**74**	0,74	0,01	**114**	0,2	0,02
35	0,35	0,01	**75**	0,75	0,01	**115**	0,21	0,02
36	0,36	0,01	**76**	0,76	0,01	**116**	0,22	0,02
37	0,37	0,01	**77**	0,77	0,01	**117**	0,23	0,02
38	0,38	0,01	**78**	0,78	0,01	**118**	0,24	0,02
39	0,39	0,01	**79**	0,79	0,01	**119**	0,25	0,02
40	0,4	0,01	**80**	0,8	0,01	**120**	0,26	0,02

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
**Automatische optimaleparameterbepaling van random bin picking visiealgoritmen
**

Richting: **master in de industriële wetenschappen: energie-automatisering**
Jaar: **2017**

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

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

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

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

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

Voor akkoord,

Palmans, Martijn

Datum: **18/01/2017**