Universiteit Hasselt

Masterproef

Scene acquisition using structured light and registration

Auteur:
Nick Michiels

Promotor
Prof. dr. Philippe Bekaert
Begeleiders
dr. Bert De Decker
Dhr. Tom Haber

Eindverhandeling voorgedragen tot het behalen van de graad van
Master in de Informatica afstudeervariant Multimedia

School voor Informatie Technologie
Transnationale Universiteit Limburg

Academiejaar 2010-2011

Abstract

Het opbouwen van 3D modellen op basis van bestaande objecten in de wereld is een sterk groeiend domein en wordt binnen verschillende doeleinden zoals entertainment of archeologie gebruikt. Daarnaast zijn computers die deze uitgebreide 3D informatie kunnen weergeven en gebruiken veel betaalbaarder geworden.

Deze thesis beschrijft een actieve correspondentietechniek, namelijk structured light, dat met camera en projector 3D informatie van een scène bekomt. Er worden een aantal structured light patronen voorgesteld die al dan niet dit probleem kunnen oplossen. Zo zijn er de time-multiplexed patronen die op een robuuste manier een groot aantal correspondenties tussen twee beelden bekomen, maar hebben het nadeel dat ze slecht overweg kunnen met bewegende objecten. Een betere aanpak, waarbij de camera vrij doorheen de scène kan rondlopen, is gebruik makend van spatial neighborhood patronen. Hierbij wordt bij het coderen van een pixel ook de aanliggende omgeving beschouwd. De meest populaire formele aanpak is het coderen met De Bruijn sequenties. Daarnaast worden er ook een aantal niet-formele spatial neighborhood technieken gegeven. Er zal getoond worden dat de keuze van een goed patroon tesamen met een global optimization techniek zoals dynamic programming het mogelijk maakt per frame een dieptemap te bekomen en deze om te zetten in een 3D puntenwolk.

De camera loopt doorheen de scène en per tijdsinstantie wordt een dieptemap van de wereld berekend. Deze dieptemappen hebben elk een andere kijk op de wereld. Om het volledige model te verkrijgen, worden alle dieptemappen in elkaar geregistreerd. De bekendste techniek hiervoor is Iterative Closest Point (ICP), dewelke op een iteratieve manier de beste transformatie zoekt om een puntenwolk in een andere puntenwolk te schuiven. ICP zal uitgebreid aan bod komen in deze thesis. Daarnaast worden er ook optimalisaties, zoals het uitbuiten van geometrische parameters (normaal, curvature), van ICP besproken. Als laatste worden er nog een aantal alternatieve algoritmes gegeven die specifiek werken op een bepaalde representatie van de 3D informatie. Uit de resultaten zal blijken dat ICP robuust werkt voor sterk overlappende puntenwolken, maar dat het problemen ondervindt met minder overlappende puntenwolken die weinig diepte bevatten.

Woord vooraf

Veel dank aan iedereen die mij gesteund heeft bij het schrijven van deze thesis. Hierbij wil ik uitdrukkelijk mijn promotor Prof. dr. Philippe Bekaert en begeleiders dr. Bert De Decker en Dhr. Tom Haber bedanken voor het nalezen, hun achtergrondkennis, hun technische hulp bij de uitwerking en hun vele goede tips.

Ook mijn familie wil ik graag bedanken om mij de mogelijkheid te geven om deze thesis te kunnen schrijven en mij al die jaren de kans te geven om verder te studeren. Hierbij wil ik ook mijn vriendin bedanken voor de steun en het nalezen van mijn thesis. Ook mijn medestudenten, bij wie ik altijd terecht kon voor vragen, mag ik niet vergeten.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

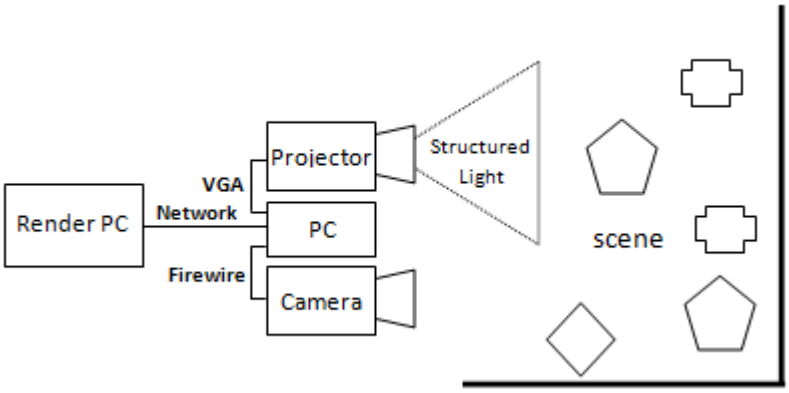
Hoofdstuk 1

Inleiding

De 3D visualisatie van scènes is het laatste decennia sterk gepopulariseerd en gecommercialiseerd. Media zoals Google Streetview hebben veel aan populariteit gewonnen. Het is ondertussen zelfs mogelijk om, vanuit de zetel in uw huiskamer, 3D scènes te bekijken op uw tv. Dit is één van de vele vruchten dat decennia van onderzoek binnen het domein van stereo vision heeft afgeworpen. Het doel van stereo vision is vaak de vorm van een scène te weten te komen met behulp van optische technieken (camera's, projectoren). Deze thesis zal ook met behulp van een stereo vision techniek de 3D informatie van een scène trachten te bekomen.

1.1 Doelstelling

Het doel van deze thesis is m.b.v. een camera en projector een kamer in te scannen. Hierbij wordt er geen beperking gelegd op de bewegingsruimte van de camera. De camera moet vrij door de kamer kunnen rondlopen en incrementeel de 3D scène aanvullen. Om de 3D informatie van de scène te verkrijgen, zal er gebruik worden gemaakt van een stereo vision techniek. Het gebruik van een camera en projector valt onder de noemer van actieve correspondentie, waarbij de projector informatie toevoegt aan de scène. Een mogelijke opstelling is bijvoorbeeld Figuur 1.1, waar één camera en projector wordt gebruikt. Het belangrijk dat in elke frame een dieptemap kan worden berekend omdat er anders geen beweging van de camera mogelijk is. Doordat voor elk opeenvolgend beeld, opgenomen door de camera, er een dieptemap wordt berekend, zal dit resulteren in een grote verzameling van dieptemappen, elk met hun eigen kijk op de scène. Elke afzonderlijke dieptemap is een puntenwolk ten opzichte van een eigen coördinatensysteem. Dit coördinatensysteem wordt bepaald door de positie van de camera in de scène. Er moet dus een techniek worden gebruikt om al deze verschillende puntenwolken met elk een verschillend coördinatensystemen in één globaal coördinatensysteem om te zetten en zo één grote puntenwolk over te houden die overeenkomt met de totale scène. Algemeen noemt met dit het registreren van 3D data in een 3D model. Er zal nu eerst dieper worden ingegaan op stereo vision en meer bepaald de verschillen tussen actieve en passieve correspondentie en de mogelijke werkwijzes voor de registratie van de dieptemappen.



Figuur 1.1: Voorbeeld opstelling van een 3D opname systeem.

1.2 Shape acquisition met stereo vision

Stereo vision is een populaire techniek binnen de computer vision om vanuit twee beelden een dieptemap te berekenen. Algemeen tracht een stereo vision techniek het correspondentie probleem op te lossen [74]. Binnen het domein van stereo vision is al zeer veel onderzoek gedaan en met de dag komen er nieuwe algoritmes bij. Naast een aantal oudere overzichten van bestaande technieken [4, 12, 20], is deze van *Scharstein en Szeliski* [74] de meest recente. De paper geeft een samenvatting van de state of the art technieken binnen computer vision. Elk besproken algoritme krijgt een score toegekend op basis van een aantal parameters. Het correspondentie probleem oplossen kan op verschillende manieren gebeuren. De algoritmes worden opgedeeld tussen passieve en actieve correspondentie [2, 60, 31].

Passieve correspondentie

Bij een passieve techniek wordt er gebruik gemaakt van meerdere input beelden. Er wordt geen extra informatie aan de scène toegevoegd, maar de correspondenties worden puur gezocht op basis van beschikbare opgenomen camerabeelden van de scène. Een bekende passieve technieken is structure from motion [88, 43]. In sommige gevallen zullen passieve technieken geen maximale resolutie verzekeren. Het aantal correspondenties dat wordt gevonden, hangt af van het type object of scène. In sommige delen van de scène kan het dus goed zijn dat er hierdoor geen of weinig correspondenties worden gevonden [60]. Dit gebeurt voornamelijk in regio's met weinig textuur. Een actieve techniek zorgt voor een verhoogde textuur en vindt dus meer correspondenties. Toch zijn de passieve state of the art technieken in kracht toegenomen. Feature detection algoritmes zijn enorm krachtig geworden. Denk maar aan de *Image Piramides* [1] die, onafhankelijk van de schaal, patronen kunnen terugvinden en veel gebruikt worden voor face recognision of het zoeken van een afbeelding in een andere. Een een andere bekende techniek voor feature detection is het gebruik maken van *SIFT* (*Scale Invariant Feature Transform*) [50]. De hardware die gebruikt moet worden hangt sterk af van het gebruikte algoritme. Algoritmes die

feature tracking doen m.b.v. één camera vereisen maar een beperkte opstelling, maar er zijn ook technieken die meerdere camera's nodig hebben.

Avtieve correspondentie

Bij passieve technieken is het mogelijk dat er maar weinig textuur aanwezig is en zo ook maar weinig correspondenties worden gevonden. Een actieve techniek lost dit op door extra textuur toe te voegen aan de scène, bijvoorbeeld het toevoegen van een gecontroleerde belichting. Deze aanpak is origineel voorgesteld door *Kang et al.* [33]. Laser scanning is een veel gebruikte maar dure techniek [48]. Eén van de eerste goedkope technieken is structured light. Structured light maakt enkel gebruik van een projector en één of meerdere camera's. De projector projecteert één of meerdere patronen en met behulp van de camera's is het mogelijk elk punt in het beeld te reconstrueren door het te matchen met zijn overeenkomstig punt in het patroon (of matchen van overeenkomstige punten tussen de camera's onderling). Net zoals bij passieve technieken zijn er algoritmes die meer hardware nodig hebben dan andere. In tegenstelling tot een passieve techniek zal er altijd een projector nodig zijn als extra hardware. Een nadeel van een actieve techniek is dat het aantal nodige afbeeldingen drastisch kan vermeerderen naarmate de resolutie groter wordt [60]. Er is heel wat onderzoek gedaan om dit nadeel tegen te gaan. In deze thesis zal gebruik maken van de actieve structured light techniek om dieptemappen van een scène te bekomen.

Registratie

Veel actieve of passieve technieken geven als resultaat een 3D puntenwolk. Bij passieve technieken zijn dit de dieptes die bekomen worden uit de overeenkomst van features tussen twee of meerdere afbeeldingen en bij actieve technieken zijn dit de correspondenties die gevonden worden in het geprojecteerde patroon. Typisch worden er puntenwolken verkregen voor verschillende camera standpunten en wil men deze samenbrengen in een globale puntenwolk. Dit proces noemt men de registratie van beelden [90]. Bij passieve technieken valt dit onder de noemer van feature tracking. Een bekende feature tracking is deze met behulp van een Kalman filter [84]. Deze berekent met behulp van predictie en feature detection (zoals bijvoorbeeld SIFT) de verplaatsing van een feature tussen twee beelden. Op deze manier kan de verplaatsing van de camera worden berekend en beide beelden gecombineerd worden. Dit is een toepassing die veel gebruikt wordt in de filmwereld, waarbij computer gegenereerde objecten geplaatst worden in een opgenomen scène. Deze objecten moeten exact overeenstemmen met de camerabeweging. Ook bij actieve technieken kunnen features worden gezocht, maar dikwijls beperken ze zich tot enkel puntgebaseerde voorstelligen. Er moeten dus ook voor puntenwolken algoritmes worden gegeven die mooi de puntenwolken in elkaar laten schuiven.

1.3 Overzicht

In de laatste decennia zijn er heel veel patronen ontwikkeld binnen de structured light methode met elk hun voor- en nadelen. In het eerste hoofdstuk van deze thesis wordt er een overzicht gegeven van de verschillende beschikbare patronen. Er wordt vooral bij stil gestaan hoe deze methodes hulp bieden binnen de gemaakte doelstellingen.

De algoritmes in het eerste hoofdstuk resulteren in verschillende puntenwolken, met elk een specifieke invalshoek op de scène. Om de scène in zijn totaliteit te tonen, worden de verschillende puntenwolken met elkaar gealigneerd in een enkele voorstelling. Dit is de zogenaamde registratie en wordt toegelicht in het derde hoofdstuk. Er zal blijken dat er verschillende duurdere en minder dure aanpakken bestaan om de registratie te bekomen. Liefst gebeurt dit volledig automatisch. Het vierde hoofdstuk zal zich focussen op de voornaamste automatische, puntwolk-gebaseerde, registratie methodes

In het vijfde hoofdstuk staat een overzicht van technieken die hulp bieden bij de drie voorgaande hoofdstukken. Zo zijn er offline preprocessing technieken zoals de calibratie tussen projector en camera, alsook de calibratie van gebruikte kleuren. Daarnaast worden er ook een aantal algemene technieken toegelicht zoals image rectification, dynamic programming en RANSAC omdat deze regelmatig terugkomen bij structured light methodes en registratietechnieken.

Het hoofdstuk dat daarop volgt bespreekt de implementatie details van een actieve correspondentie techniek en registratie. Er zal beschreven worden hoe met behulp van een De Bruijn patroon en dynamic programming er een 3D reconsructie van een opgenome scène kan gebeuren. Daarnaast wordt er uitgelegd hoe Iterative Closest Point (ICP) kan gebruikt worden om een registratie van puntenwolken uit te voeren. Hierbij kunnen al dan niet extra geometrische parameters van de scène worden uitegebuit.

De resultaten die uit het voorgaande implementatie hoofdstuk komen, worden uitvoerig besproken in het zevende hoofdstuk. Er wordt vooral dieper ingegaan om de kwaliteit en snelheid van de algoritmes, alsook de eventuele tekortkomingen ervan. De thesis sluit af met een korte conclusie van de bereikte resultaten binnen de opgestelde doelstelling.

Hoofdstuk 2

Structured light

Het doel van stereo vision is het vinden van correspondenties tussen twee beelden die gebruikt worden voor de opbouw van een dieptemap. Er moet m.a.w. een één-op-één relatie worden gevonden tussen twee waargenomen beelden. Het vinden van deze correspondenties hangt sterk af van de opbouw van de scène. Vooral als er weinig textuur aanwezig is, zullen er weinig overeenkomsten worden gevonden. Daarom wordt bij een structured light methode een camera vervangen door een projector. Hierdoor wordt er extra textuur (in de vorm van patronen) geprojecteerd op de scène, die helpen bij het vinden van correspondenties. Het algoritme weet nu exact hoe het patroon er uit ziet en er kan dus ook makkelijker gezocht worden naar dezelfde patronen in het waargenomen beeld. In elk opgenomen beeld wordt er gezocht naar kenmerkende punten van het patroon en doordat elk kenmerkend punt van een patroon geassocieerd wordt met een positie in het patroon, is het correspondentieprobleem opgelost. Er zijn verschillende mogelijkheden voor de opbouw van een patroon. De wijze waarop ze verschillen wordt in de volgende sectie aangehaald. Vervolgens wordt in sectie 2.2 de verschillende criteria aangehaald die belangrijk zijn bij het kiezen van een juist structured light algoritme. In de daarop volgende secties 2.4, 2.5, 2.6 en 2.7 worden de verschillende structured light strategiën en de verschillende algoritmes die voor elke strategie bestaan besproken. Ten slotte wordt er nog een vergelijking gemaakt van bestaande algoritmes die inzetbaar zijn binnen de gemaakte doelstelling van deze thesis.

2.1 Classificatie en taxonomie

Het ingenieuze van een structured light techniek is de manier waarop het beeld wordt gecodeerd. Deze codering bepaalt in grote mate het eindresultaat en het heeft invloed op de resolutie, de snelheid en de kwaliteit. De bedoeling is elke pixel of groep van pixels een bepaald codewoord te geven om zo een kleine oppervlak uniek te bepalen. Dit wordt gerealiseerd door de projectie van één of meerdere patronen op de waargenome scène. Als resultaat heeft elke pixel zijn eigen codewoord en bijgevolg ook elke pixel in de scène een mapping op zijn overeenkomstige coördinaat van het patroon zelf. Het zoeken van deze mapping is gekend als het correspondentie probleem. Doordat de camera en projector

ten opzichte van elkaar zijn gecalibreerd, waardoor m.a.w. de positie ten opzichte van elkaar is gekend, wordt met behulp van triangularisatie de diepte verkregen. De keuze van het patroon heeft dus een sterke invloed op het aantal correspondenties dat wordt gevonden. Maar de codering is niet de enige stap die veel invloed heeft op het resultaat. Ook de decodering mag niet worden vergeten. Hoe solider de decodering wordt uitgevoerd, hoe meer correspondenties er worden gevonden en hoe beter de 3D reconstructie wordt uitgevoerd. Bij het decoderen wordt er getracht het indexing probleem op te lossen. Met moet voor elk waargenomen en gecodeerd oppervlak de juiste bron zoeken in het oorspronkelijk patroon [38]. Het is vaak een niet triviale stap omdat de volgorde van het geprojeceerde patroon niet altijd overeenkomt het het waargenomen patroon.

Doordat de manier van coderen het meest bepalende is voor een structured light techniek, zal dit de basis vormen voor de classificatie van patronen. Over het algemeen zijn er drie verschillende categorieën. De eerste is time-multiplexing. Bij deze techniek worden de codewoorden opgesteld doorheen de tijd. Een projectie van een aantal patronen na elkaar geeft voor elke pixel of oppervlak een bepaald codewoord. Deze categorie wordt besproken in sectie 2.4. Een tweede categorie tracht al de code-informatie te stoppen in een enkel patroon. Hierdoor zal er niet meer gecodeerd worden over de tijd maar wel over de oppervlak van het patroon. De buren van elke pixel van het patroon worden ook beschouwd om de pixel te kunnen coderen. Deze categorie noemt spatial neighborhood en wordt uitgebreid besproken in sectie 2.5. De derde en laatste categorie is die van direct coding. Hiermee tracht men direct voor elke pixel een code te geven (meestal met behulp van grijswaardes of andere gradiënte patronen). Hoe dit in zijn werk gaat kunt u lezen in sectie 2.6. De drie verschillende aanpakken worden door *Salvi, Pagès* en *Battle* grondig samengevat [59]. Als laatste wordt er in sectie 2.7 nog een extra algoritme gegeven dat net zoals direct coding temporeel en spatiaal onafhankelijk is en door middel van de correcte plaatsing van verschillende camera's een binair patroon voor de posities opbouwt.

2.2 Criteria

Een structured light patroon kan op verschillende wijzen worden opgebouwd. De keuze van het patroon hangt dikwijls af van de context van de scène en de doelen die men vooraf opsteld. In deze sectie worden enkele veel voorkomende doelen die effect hebben op de keuze van het patroon en het resultaat ervan besproken.

2.2.1 Context van de scène

Een veel voorkomende techniek is het coderen in het temporele domein. In deze strategie gaat de opeenvolging van geprojecteerde patronen het codewoord van een pixel bepalen (een bekend voorbeeld is gray codes [39]). Dit zorgt voor temporele vereisten: de scène mag namelijk niet te snel bewegen of opeenvolgende codes zullen niet op dezelfde elementen binnen de scène vallen [86]. Indien de context weinig beweging vereist, kan

codering in de tijd wel een goede keuze zijn omdat deze patronen meestal een grote resolutie aankunnen met een beperkte window van codes. Voor binaire codes zijn $\log N$ codes voldoende om N pixels te coderen. Als daarentegen de context van de scène grote bewegingen vereist, kan men beter kiezen voor one-shot coding waarbij alles gecodeerd wordt in de projectie van één enkel patroon. Zo kan er voor elke momentopname in de tijd correspondenties worden gezocht. Het nadeel van deze techniek is dat hij spatiale beperkingen heeft. Een codewoord van een pixel hangt m.a.w. af van de plaats waarop hij zich bevindt in het patroon [86]. Deze beperking bemoeilijkt aanzienlijk de decoderingsstap omdat het complexer is om het correcte codewoord te vinden en er zo eventueel geen of verkeerde correspondenties worden gevonden. Daarenboven wordt de resolutie ook verkleind, omdat nu alle coderingsdata in één patroon wordt verwerkt [61]. Er zijn hybride methodes voorgesteld die profiteren van de voordelen van zowel temporele als spatiale coderingsstrategiën [85].

Een tweede invloed van de scène op het resultaat is het al dan niet aanwezig zijn van discontinuiteïten. Plotse overgangen in de scène zorgen ook voor plotse aanpassingen van het waargenomen patroon. Een slechte keuze van patroon en decoderingstechniek kan nefaste gevolgen hebben op het eindresultaat. Het is dus belangrijk dat ofwel het patroon overweg kan met discontinuiteïten, ofwel de decoderingsstap fouten kan detecteren en eventueel zelfs kan oplossen. Veel patronen kunnen dit probleem nog niet aan, maar er zijn er ook een aantal die proberen een oplossing te bieden voor dit probleem. Zo voegen sommige algoritmes redundante informatie toe aan het patroon [71]. Verder kunnen er zich nog een aantal praktische problemen voordoen: occlusions; niet gekende codes door shaduwen en reflecties; een perfecte match wordt zelden gevonden; meerdere matchen kunnen worden gevonden wegens beperkte resolutie; correspondenties van verschillende illuminaties kunnen inconsistenties geven [85, 75].

Veel van de keuzes worden gemaakt in functie van het doeldomein. Applicaties voor industriële doeleinden concentreren zich meer op accuraatheid en zijn meer tolerant t.o.v. artefacten in de resultaten. Dit is geen probleem, zolang deze maar geen invloed hebben op de productie. Daarentegen zijn applicaties binnen het graphics domein meer veeleisend ten opzichte van de visuele correctheid. Ze zijn met andere woorden wel tolerant ten opzichte van fouten, zolang ze niet zichtbaar zijn in het eindresultaat [70].

Het is dus duidelijk dat bij het zoeken naar een geschikt patroon voor een eigen implementatie de context zeer belangrijk is. Het is dus handig om te weten welk algoritme goed gebruikt kan worden binnen welke context. Deze thesis probeert tegemoet te komen aan deze noden door duidelijk per algoritme te stellen welke context er is gebruikt en welke doelen er zijn voldaan.

2.2.2 Gebruikte hardware

Een basis structured light techniek kan werken met één camera en één projector om de dieptemap van een scène te verkrijgen. Toch zijn er algoritmes ontwikkeld die gebruik

maken van meerdere camera's en/of projectoren. Dit om de kwaliteit van het resultaat te verhogen of om het mogelijk te maken een bewegende scène op te nemen. *Waschbüsch et al.* [83] maken gebruik van zogenaamde video bricks die elk bestaan uit een projector, pc en drie camera's. Elke brick gaat een eigen structured light uitvoeren en een diepte map van dat viewpoint berekenen. Een combinatie van meerdere bricks (op voorwaarde dat de bricks ten opzichte van elkaar zijn gecalibreerd) maakt het mogelijk de totale geometrie van een object bekomen op één exacte tijdsinstantie. Het nadeel is dat ten eerste er veel camera's en projectoren nodig zijn en ten tweede dat alle bricks vast ten opzichte van elkaar moeten worden gecalibreerd. Een andere bekende techniek is viewpoint-coded structured ligth [86]. Deze maakt gebruik van meerdere camera's en epipolaire geometrie om de geometrie van een scène te bekomen. Het is niet noodzakelijk meerdere camera's te gebruiken, maar het steunt wel op het gebruik van verschillende camerastandpunten van dezelfde scène tesamen met een geprojecteerd binair patroon.
Over het algemeen kan er worden gesteld dat meerdere camera's of projectoren de kwaliteit van het resultaat verhoogt, maar dit ten koste van andere beperkingen zoals het vrij rondbewegen doorheen de scène. Dikwijls wil men ook bewust een goedkoop algoritme dat maar één camera en één projector bevat. Zoals in de volgende secties duidelijk zal worden, zijn er veel technieken voorhanden om goede resultaten met een beperkte opstelling te bekomen.

In de doelstelling van deze paper wordt er bewust gekozen voor één camera en één projector. De algoritmes die hier worden aangehaald zullen zich dan ook vooral toespitsen op deze doelstelling.

2.2.3 Gebruik van kleuren

Het voordeel van het gebruik van kleuren in patronen is dat de resolutie kan worden verhoogd. Een gekleurde pixel bevat meer gecodeerde data dan een grijswaarde pixel [89]. Dit heeft als gevolg dat voor eenzelfde grootheid van data een kleiner oppervlak wordt gecodeerd en zo de resolutie wordt verhoogd. Zoals in de volgende sectie zal blijken, zijn er veel methodes die deze voordelen willen uitbuiten en gebruik maken van kleur in patronen. Bij het gebruik van kleuren is het belangrijk dat de kleuren veel van elkaar verschillen [38].

Toch is het over het algemeen niet zo'n goed idee om kleuren te gebruiken. Het probleem ligt vooral in de decoderingsstap. Het is moeilijk om een onderscheid te maken tussen kleuren van objecten en kleuren van het patroon [89]. De oppervlakte van de scène kan best over het hele lichtspectrum reflecteren om de juiste kleurwaardes te kunnen interpreteren. Deze limitatie kan de kwaliteit van het resultaat sterk beïnvloeden. Ook hier zijn kleurenpatronen ontwikkeld die overweg kunnen met dit probleem [73]. Als laatste is een goede calibratie van kleuren ook belangrijk.

Er kan best een afweging worden gemaakt tussen twee parameters: resolutie en accuraatheid. Grijswaarden patronen (zeker time-multiplexing patronen) hebben de eigen-

schap een goede accuraatheid te hebben, maar kunnen maar een relatief beperkte resolutie aan. Daarentegen gaan kleurwaarden patronen meer bits gebruiken voor te coderen en kunnen dus een grotere resolutie aan. Dit gaat ten koste van de nauwkeurigheid en kwaliteit van de resultaten. Sommige papers proberen beide voordelen uit te buiten en maken gebruik van een hybride patroon [60].

2.2.4 Resolutie

De manier waarop een patroon wordt gecodeerd, zal ook de resolutie beïnvloeden. De resolutie wil men altijd zo hoog houden. Dus hoe meer pixels er kunnen gecodeerd worden in een beeld hoe gedetailleerder de resultaten. Met de resolutie wordt voornamelijk het aantal gevonden corresponenties bedoeld en hoe meer gevonden correspondenties, hoe gedetailleerder de dieptemappen zullen zijn. In de vorige secties zijn er al een aantal voorbeelden gegeven die invloed hebben op de resolutie. Bij binaire patronen of gray codes kan elke pixel worden gecodeerd, waardoor de optimale resolutie wordt bereikt. Daarnaast kan het gebruik van kleuren ook de resolutie verhogen omdat er meerdere bits worden gebruikt. Een time-multiplexing patroon codeert elke pixel uniek zodat het de maximale optimale resolutie bereikt. Het gebruik van spatial neighborhood, waar de codes worden opgebouwd met een enkel patroon en de omliggende pixels worden beschouw, zal daarentegen een kleinere resolutie aankunnen omdat alle code informatie in eenzelfde patroon moet worden verwerkt.

2.2.5 Real-time

Vaak is het gewenst het incannen van een scène real-time uit te voeren. Dit zorgt ervoor dat tijdens het scannen ook direct de resultaten kunnen gerendered worden. Dit heeft als voordeel dat tijdens het scannen direct geïnspecteerd kan worden welke delen van de scène nog verder ingescand moeten worden en welke delen niet zo goed gelukt zijn. Er kunnen allerhande optimalisatietechnieken worden toegepast om de beschikbare algoritmes te versnellen. Zo zal dynamic programming (zie sectie 5.5) veel sneller werken als eerst rectification (zie sectie 5.3) op de inkomende beelden wordt toegepast.

2.2.6 Minimaliseren en detecteren van fouten

Belangrijk bij de keuze van een algoritme is dat het aantal fouten zo klein mogelijk blijft. Typisch voegt een structured light technieken veel ruis toe aan de scène of worden er een aantal verkeerde correspondenties gevonden. Het is daarom noodzakelijk dat de algoritmes overweg kunnen met deze fouten. Het algoritme kan ofwel krachtig zijn en weinig fouten introduceren ofwel een compensatie uitvoeren voor gemaakte fouten. Waschbüsch et al. [83] maken gebruik van discontinuity optimization, welk een two-phase post-processing algoritme is op de dieptemappen. Ten eerste gaan ze foute correspondenties wegfilteren en ten tweede gaan ze nieuwe correspondenties interpoleren vanuit hun omgeving. Het algoritme maakt gebruik van gelijkenissen tussen textuurkleur. Bij structured light patronen worden soms redundante bits toegevoegd die gebruikt kunnen

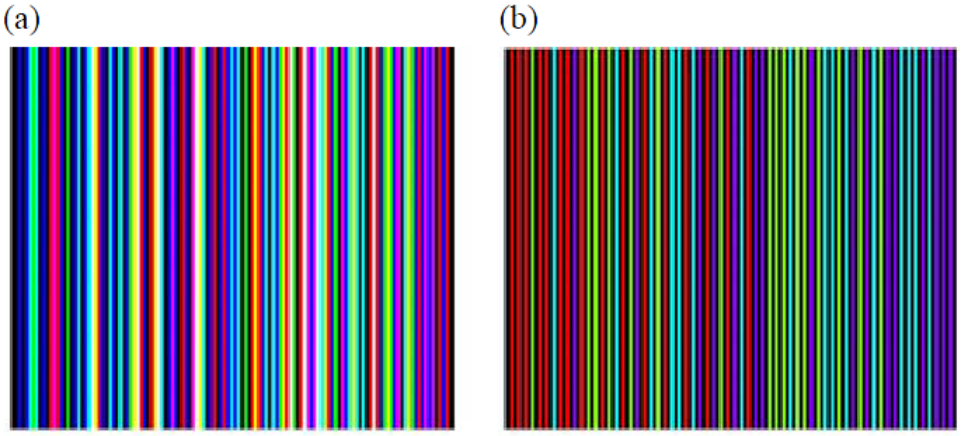
worden voor de detectie en correctie van fouten. Zo voegt Salvi et al. [71] redundante bits toe op kruisingen van grid patroonen waardoor ze extra tests kunnen uitvoeren bij het decoderen.

2.3 Peak-based en edge-based reconstructie

Voordat er wordt begonnen met de bespreking van de verschillende patronen is er nog een belangrijke opdeling dat vaak terugkomt. Veel patronen maken gebruik van zogenaamde slits. Slits zijn snedes doorheen het patroon met een bepaalde intensiteitswaarde. Deze slits kunnen smal of breed zijn, naargelang de opbouw van het patroon. Er zijn twee klassieke aanpakken die gebruik maken van slits, de multi-slit patronen en de stripe patronen. Beide aanpakken verschillen op basis van het terugvinden van stripes in een beeld.
Een multi-slit patroon intoduceert zwarte hiaten tussen gekleurde banden. Hierdoor kunnen opeenvolgende slits dezelfde kleur hebben. De zwarte gaten zorgen voor intensiteitspieken die kunnen gedetecteerd worden in het waargenomen beeld. Elke intensiteitspiek beschrijft de centrale positie van een slit. Dus bij het decoderen moet het centrale punt van de geprojecteerde slit getriangulariseerd worden met het punt van de intensiteitspiek. Dit wordt ook wel *peak-based reconstruction* genoemd. Een voorbeeld is weergegeven in Figuur 2.1b.
Een tweede aanpak is het gebruik van stripe patronen. Bij stripe patronen worden er geen zwarte banden toegevoegd tussen de gekleurde stripes (slits). Opeenvolgende stripes mogen dus niet dezelfde kleur hebben. In zulke patronen worden de randen van de sripes gezocht. Elke rand van het waargenomen beeld kan getriangulariseerd worden met de rand van de overeenkomstige geprojecteerde stripe. Dit noemt met *edge-based reconstruction*. Salvi et al. [60] geven een mooi overzicht van deze twee technieken. Een voorbeeld is weergegeven in Figuur 2.1a. Samengevat zoekt een multi-slit patroon naar intensiteitsverschillen en zoekt een stripe patroon naar randen tussen opeenvolgende stripes. Het voordeel van de stripe patronen is dat de hele resolutie kan worden gebruikt en er geen zwarte banden nodig zijn. Daarentegen zijn de slits van multi-slit patronen makkelijker terug te vinden. Het is niet mogelijk om met een multi-slit patroon edge-based reconstruction toe te passen. Dit omdat de intensiteit wordt geïntegreerd over de zwarte regios. Dit is ook de reden waarom bij binaire patronen ook best de negatieve versie van het patroon wordt geprojecteerd. Het zijn ook Salvi et al. [60] die probeerden om met een combinatie van beide technieken de voordelen van elk uit te buitten.

2.4 Time-multiplexing

Een veel gebruikte techniek van coderen is het projecteren van patronen in het temporeel domein. Bij temporele codering gaat doorheen de tijd een sequentie of codewoord voor een bepaalde pixel/oppervlak worden bepaald. Zo worden er bijvoorbeeld een aantal patronen geprojecteerd in een scène. Het codewoord voor elke pixel wordt bepaald door een opeenvolging van de verschillende belichtingswaarden van die pixel. De bits van een

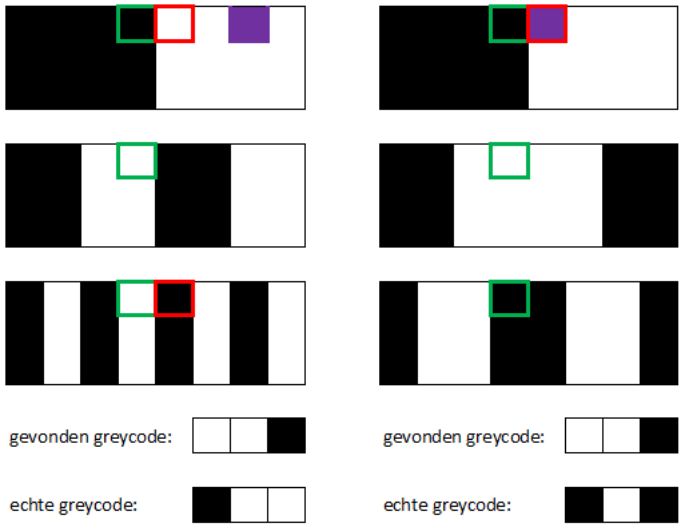


Figuur 2.1: Stripe patronen en multi-slit patronen [60]. (a) Stripe-patroon door Zhang et al. [89]. (b) Multi-slit patroon gelijksoortig aan degene voorgesteld door Monks et al. [55].

codewoord worden verdeeld (gemultiplexed) doorheen de tijd. Het voordeel van deze techniek is dat de resolutie groot is omdat men tot op pixelniveau een uniek codewoord kan aanmaken. Een bijkomend voordeel is dat de codes dikwijls vrij kort zijn omdat er meestal maar een beperkt aantal patronen nodig zijn. Het nadeel van time-multiplexed coding is dat de scène of camera vrij statisch moet zijn. Bij snel bewegende scenes is het niet meer mogelijk om alle geprojecteerde pixels van de verschillende patronen op hetzelfde achtereenvolgend punt te laten vallen. Doordat in de gemaakte doelstelling er geen beperking wordt gelegd op de beweging van de camera, zijn de time-multiplexing technieken niet bruikbaar. Toch zullen de belangrijkste kort worden aangehaald.

2.4.1 Binaire Patronen

Het bekendste temporele patroon is het traditionele binaire patroon, origineel voorgesteld door *Posdamer en Altschuler* [64]. In deze patronen worden enkel de binaire codes 0 en 1 gebruikt. Deze zijn gecodeerd in zwart en wit. Vervolgens wordt het codewoord voor een pixel samengesteld door de waargenomen pixels in de sequentie van geprojecteerde patronen. Het voordeel van het binaire patroon is dat er maar $\log N$ patronen nodig zijn om een resolutie van N pixels te coderen. Het nadeel van algemene binaire patronen is dat een aantal randen voortdurend blijven samenvallen. Hierdoor kan het zijn dat op de rand een verkeerde waarde wordt gekozen. Het kiezen van een verkeerde waarde bij een binair patroon kan drastische gevolgen hebben in het resultaat omdat totaal andere posities als resultaat worden teruggegeven. Het linkse deel van Figuur 2.2 laat zien hoe het codewoord kan afwijken bij een foute keuze. Om deze problemen tegemoet te komen werden gray codes ontwikkeld.



Figuur 2.2: Vergelijking van binaire patronen en gray codes. Deze figuur [54] toont aan waarom het beter is graycode patronen te gebruiken in plaats van de traditionele binaire patronen. De linkse figuur maakt gebruik van binaire patronen. De groene pixel is de pixel waarvoor de code wordt gezocht, de rode is de pixel die wordt gekozen. Doordat de randen tussen zwart en wit voortdurend blijven samenvallen, kunnen pixels meermaals fout worden gekozen. In dit geval wordt de pixel tweemal fout gekozen, waardoor de gevonden pixel (paarse) drie plaatsen van zijn correcte plaats afwijkt. In de rechtse figuur—dewelke gebruik maakt van graycodes—worden gebieden van dezelfde kleur twee aan twee samengenomen. Hierdoor zal de groene pixel maximaal twee maal op een rand vallen waardoor hij ook maar twee maal fout kan worden gekozen. Dit geeft een maximale afwijking van 2 pixels. In dit voorbeeld is er maar één pixel afwijking.

2.4.2 Gray codes

De binaire codes werden door *Inokuchi, Sato et al.* [39] verbeterd en werden vervangen door Gray codes. Gray codes zijn gecodeerd zoals in Figuur 2.3. Het voordeel van deze codes is dat opeenvolgende codes maar één bit verschillen en dat bij een foute waarneming er geen grote afwijking op zal zitten. Dit kunt u zien aan de rechterkant van Figuur 2.2.

2.5 Spatial neighborhood

Bij spatial neighborhood zal men trachten al de coding informatie te stoppen in een enkel patroon. Hierdoor is het voldoende om ook maar één patroon te projecteren waarbij het waargenomen beeld direct kan worden gedecodeerd. In de literatuur wordt deze techniek ook wel een one-shot techniek genoemd. Het codewoord dat een bepaald punt bestempelt, is samengesteld uit de informatie van het punt zelf en de punten in de nabijheid.

Figuur 2.3: Graycodes ontwikkeld door *Inokuchi et al.* [39]. Deze Figuur [54] geeft graycodes weer over een bereik van 32 pixels. De graycode voor pixel 25 is bijvoorbeeld 01010

Meestal worden de intensiteiten of kleurwaarden van de pixels gebruikt om de codewoorden samen te stellen. Het valt direct op dat deze techniek de decoderingsstap aanzienlijk complexer maakt, men moet nu namelijk ook alle aangrenzende pixels beschouwen. De kans dat alle pixels correct worden gevonden is kleiner dan bij andere technieken die maar over één enkel punt decoderen. Het meest bepalende onderdeel om zo'n techniek real-time te krijgen, is de belasting van het decoderen op het process. Meestal staat de decoderingsstap in verband met de kwaliteit van het eindresultaat. Er moet dus een afweging worden gemaakt tussen beide om binnen een aanzienlijke tijd voldoende correcte resultaten te verkrijgen [59].
Binnen spatial neighborhood patronen zijn er drie grote classificaties: niet-formele neighborhood, zonder te voldoen aan een formele methode; gebruik makend van De Bruijn pseudo random sequenties; en gebruik makend van M-arrays [59].

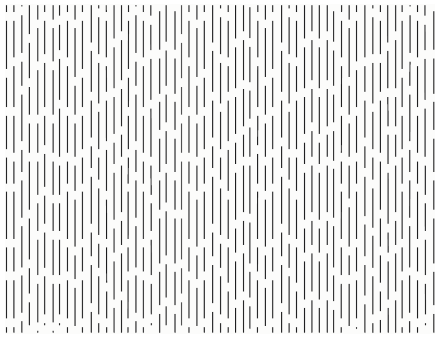
2.5.1 Niet-formele patronen

Bij niet-formele patronen wordt er geen gebruik gemaakt van formele wiskundige methodes om een bepaalde regio te coderen. De methodes zorgen voor een eigen indeling van regios waarvoor unieke codes worden opgesteld. Er wordt onderscheid gemaakt tussen patronen die bestaan uit slits en patronen die bestaan uit een grid structuur.

Horizontal/vertical slits

Een eerste methode is het gebruik maken van een binair patroon, opgebouwd met verticale of horizontale slits. *Maruyama en Abe* [52] gebruiken zo bijvoorbeeld verticale slits die willekeurig worden onderbroken. Elke slit wordt zo opgedeeld in een aantal smalle lijnsegmenten. Een voorbeeld van zo'n patroon vindt u in Figuur 2.4. Deze segmenten worden tussen het patroon en de afbeelding gematched om zo 3D data te bekomen. Om dubbele overeenkomsten te voorkomen, zullen aangrenzende relaties tussen segmenten worden gebruikt om de foutieve correspondenties te minimaliseren. De segmenten die volledig overeenkomen (dus ook de aangrenzende segmenten), worden beschouwd als correcte correspondenties. Bij de overige segmenten zullen de relaties tussen aangrenzende segmenten nog verder worden uitgebuit om zo toch tot correcte overeenkomsten te komen. Het zoeken in de omgeving hoeft niet enkel verticaal, maar kan ook horizontaal.

Volgens *Maruyama en Abe* [52] is het zelfs robuuster in de horizontale richting te zoeken omdat er dan volgens de epipolaire lijn wordt gezocht. Het zoeken naar de segmenten is optimaal opgebouwd door de segmenten voor te stellen in een MD-tree. Het algoritme is ontwikkeld voor continue oppevlakken. Bij plotse overgangen worden de segmenten verkeerd geïnterpreteerd. Een ander groot nadeel van deze techniek is dat de lengte van de segmenten afhangt van de positie van de camera en projector ten opzichte van de scène [59].



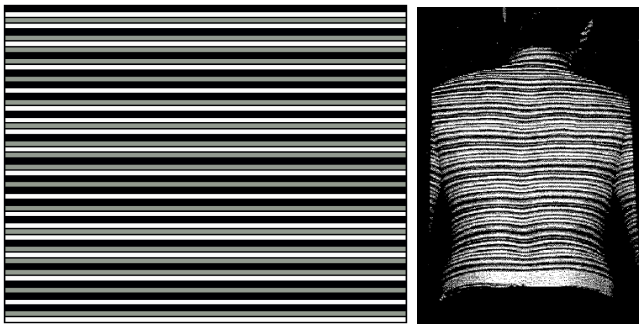
Figuur 2.4: Slits willekeurig afgesneden, zoals voorgesteld door *Maruyama en Abe* [52].

Een ander voorstel is het patroon periodiek opdelen in horizontale slits gecodeerd met drie verschillende intensiteit levels (255 voor helder (w), 0 voor zwart (b) en 127 voor tussenliggende grijswaardes (g)), voorgesteld door *Durdle et al.* [22] in 1998. Er zijn verschillende mogelijkheden om de sequentie van intensiteitswaarden op te stellen. In de paper van *Durdle et al.* [22] wordt er gebruik gemaakt van de volgende sequentie:

BWG,WBG,WGB,GWB,GBW,BGW

met B = bbbb, G = gggg, W = wwww, b = 1 pixel met intensiteit 0, g = 1 pixel met intensiteit 127 en w = 1 pixel met intensiteit 255. Deze sequentie wordt over het hele patroon periodiek herhaald tot de verticale resolutie is bereikt. *Durdle et.* [22] kozen deze sequentie omdat in de decoderingsfase het patroon makkelijk kan worden teruggevonden zonder het introduceren van foute matches. Het patroon ziet er uit zoals in Figuur 2.5a. Geprojecteerd op een oppervlak zal het waargenomen beeld er uit zien zoals in Figuur 2.5b. Het patroon kan deels overweg met discontinuïteiten en zijn toegelaten zolang ze geen periode van het patroon overschrijden.

Het decoderen gebeurt in twee stappen. Als eerste zoekt men per kolom de beginpunten vr de verschillende perdiodes. Hiervoor wordt een template matching algoritme gebruikt. Er wordt een template gekozen ter grootte van 54 pixels, wat ongeveer overeenkomt met één periode van het geprojecteerde patroon. Het template matching algoritme gaat correlatiepieken zoeken in het waargenomen beeld. Deze correlatiepieken zijn plaatsen waar de template sterk overeenkomt met het waargenomen beeld. Zo kunnen de beginpunten van de periodes worden gevonden. Als eenmaal de beginpunten



(a) Patroon (b) Geprojecteerd patroon

Figuur 2.5: Periodiek patroon van horizontale slits, gecodeerd in drie grijswaardes, zoals voorgesteld door *Durdle et al.* [22]: (a) het patroon; (b) het patroon geprojecteerd op een oppervlak.

gekend zijn zoekt men ten tweede naar de subsequenties BWG, WBG en GBW door gebruik te maken van een tweede template van 18 pixels groot. Deze twee correlaties worden herhaald over elke kolom van het waargenomen beeld om zo alle correspondenties te vinden. Doordat het beeld is rechtgetrokken met image rectification (zie sectie 5.3), moeten enkel nog de locaties van het patroon en het beeld op elkaar gematcht worden om de dieptemap te bekomen.

Boyer en Kak [11] hebben een stripe patroon geïntroduceerd die gebruik maakt van verticale slits, gecodeerd met de 3 basiskleuren (rood, groen en blauw), die van elkaar gescheiden worden met zwarte banden. De sequenties van de gekleurde slits zijn zo gekozen zodat er, bij het opdelen van het patroon in subpatronen, geen enkele herhaling kan plaatsvinden. *Boyer en Kak* zagen dat de morfologie van de oppervlakken in een scène het patroon verstoort. Het opgenomen beeld kunnen stoornissen of zelfs verwijderingen bevatten van delen van het patroon. Ze zijn één van de eerste die een algoritme probeerden te ontwikkelen om hiermee overweg te kunnen, namelijk het stripe indexing process:

I Correlation: Voor elke epipolaire lijn wordt elk uniek subpatroon van het originele patroon gecorreleerd met het ontvangen beeld om voor alle posities een perfecte match te vinden.

II Crystal growing: Er wordt region growing toegepast op de gematchte subpatronen. Zo worden zoveel mogelijk overeenkomstige slits gevonden.

III Fitting process: De slechte overeenkomsten worden verwijderd. Dit gebeurt door enkel de grootste overeenkomstige matches te behouden en de kleinere weg te laten.

IV Ordinal Assignment: Er wordt een update gedaan van de gevonden overeenkomsten en elke overeenkomst krijgt een index.

Het nadeel van dit algoritme is de vele processing bij het decoderen. Het is niet triviaal om het stripe indexing process uit te voeren. Het vergt dus veel optimalisaties om dit algoritme real time te krijgen. Het algoritme kan wel goed overweg met bewegende objecten.

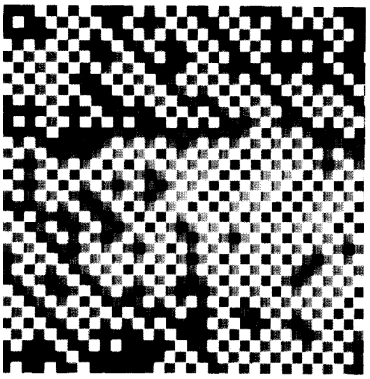
Ook *Chen et al.* [14] maken gebruik van een sequentie van verticaal gekleurde slits, gescheiden van zwarte banden. Ze hebben niet veel onderzoek gedaan naar de opbouw van de sequentie. De kleuren worden gekozen met een trial-and-error algoritme dat een sequentie zoekt met een lage autocorrelatie tussen kleurwaardes. In hun paper maken ze gebruik van twee camera's, maar het is perfect mogelijk hetzelfde uit te voeren met één camera. *Chen et al.* legden vooral de nadruk op de decoderingsstap. Deze bestaat uit twee delen: *intra-scanline search* en *inter-scanline consistency*. Intra-scanline search gaat zoeken naar correspondenties van randen tussen twee epipolaire lijnen (er van uit gaand dat image rectification (sectie 5.3) is toegepast, komen de epipolaire lijnen overeen met de horizontale scanlines van de afbeelding). Elke scanline lijn bestaat uit een sequentie van gekleurde slits en zwarte banden. Doordat de twee camera's een verschillend kijkpunt hebben op de scène, gaan de sequenties niet altijd mooi overeenkomen. Om toch de sequenties juist te kunnen matchen met elkaar wordt dynamic programming (zie sectie 5.5) toegepast. De sterkte van dynamic programming is dat hij het aantal optimale veranderingen die nodig zijn op een sequentie—om de sequentie overeen te laten komen met een tweede sequentie–berekent. Toch kan dynamic programming niet volledig overweg met verwisselingen van slits ten gevolge van bijvoorbeeld occlusions. Daarom introduceerden *Chen et al.* ook de inter-scanline consistency. Deze tracht toch nog randen te machen die in de intra-scanline search niet zijn gematched, door gebruik te maken van informatie uit aanliggende scanlines. Het nadeel van deze techniek is dat de opbouw van het patroon niet robuust is. Hiervoor zouden eventuele andere spatial neighborhood patronen gebruikt kunnen worden.

Grid patronen

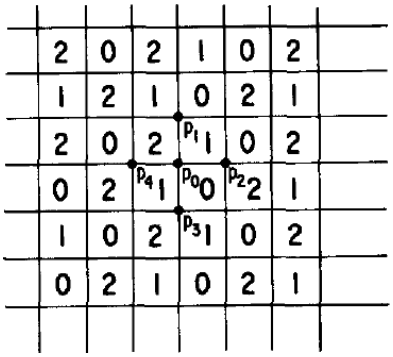
Een tweede aanpak is het gebruik maken van grid patronen. Het spatiale oppervlak wordt onderverdeeld in cellen. Elke cel zal een aantal gegevens bevatten om zo een plaats uniek te coderen.

In 1994 werd er een schaakbordpatroon ontwikkeld door *Ito en Ishii* [42]. Elke cel wordt opgevuld met één van de drie mogelijke intensiteitswaardes. De waarde van een cel is zo gekozen zodat alle vier aangrenzende cellen niet dezelfde intensiteitswaarde hebben. De punten waar vier cellen samenvallen worden gedefinieerd als nodes. In Figuur 2.6a ziet u een voorbeeld van het schaakbord patroon. In Figuur 2.6b worden de nodes weergegeven. U ziet dat P_0 is omgeven door vier cellen die in wijzerszin de volgende intensiteiten hebben: 2, 1, 0 en 1. Elke node wordt gerepresenteerd door deze

vier intensiteitswaardes. Dit is de *main code*. Dus voor P_0 is de main code (2101). Er worden maar drie verschillende intensiteitswaardes gebruikt. Hierdoor zijn er maar 18 verschillende main codes mogelijk (3x2x2x2). Een *subcode* wordt gedefinieerd als de combinatie van de vier main codes van de aangrenzende nodes. De subcode voor node P_0 is bijgevolg $(1012, 1020, 1012, 0212)$. Dit geeft een totaal van $18\text{x}3^4 = 1458$ verschillende combinaties. Om de positie van een subcode in het patroon te bepalen is het voldoende om een spatiale omgeving van 12 cellen van het grid te onderzoeken. Het algoritme is pas robuust als elke subcode maar éémaal in het patroon voorkomt. In de paper van Ito en Ishii is er hierop geen beperking gelegd. Ze laten herhalingen van subcodes toe in hun patroon. Om een onderscheid te maken tussen nodes met dezelfde subcode wordt er gebruik gemaakt van epipolaire relaties tussen camera en projector.
Het is een vrij snelle techniek en kan mits optimalisaties real-time worden gemaakt. Het patroon is ontwikkeld voor de industrie, waar maar weinig foute overeenkomsten mogen worden gevonden. Het hoofdoel van dit patroon is dan ook betrouwbaarheid. Daarnaast heeft het ook een vrij grote spatiale resolutie.



(a) Schaakbord patroon door Ito en Ishii



(b) Schaakbord patroon gecodeerd met nodes door Ito en Ishii

Figuur 2.6: Voorbeeld van het schaakbord patroon van *Ito en Ishii* [42]: (a) het patroon wordt opgedeeld met drie verschillende intensiteitswaardes, hierbij mogen de vier direct aangrenzende cellen niet dezelfde grijswaarde hebben; (b) De nodes van het schaakbord patroon. Een node is gedefinieerd als een snijding van vier cellen. De main code wordt bepaald door vier intensiteit levels van de aangrenzende cellen van een node. De codes voor node p_0, p_1, p_2, p_3 en p_4 zijn $(2101), (1012), (1020), (1012)$ en (0212). De subcode van p_0 is $(1012, 1020, 1012, 0212)$.

2.5.2 De Bruijn patronen

In de vorige sectie werden over het algemeen informele technieken gebruikt om patronen te coderen. Er bestaan ook technieken die meer formeel zijn gedefinieerd. Eén van de meest bekende formele technieken voor het coderen van patronen zijn degene die gebaseerd zijn op *De Bruijn sequenties*. In de eerste sectie zal kort worden toegelicht

wat een De Bruijn seqentie inhoudt en vervolgens zullen er een aantal technieken worden besproken die gebruik maken van zulke sequenties. Ook hier is weer een opdeling gemaakt van patronen gebruik makend van slits of cellen.

De Bruijn sequenties

Het doel van een De Bruijn sequentie is er voor te zorgen dat een aantal opeenvolgende elementen (bv. kleuren) uniek voorkomen in het de sequentie. Een De Bruijn sequentie definieert een zogenaamde window grootte en verzekert dat een subsequentie van deze grootte maar exact één keer voorkomt in de sequentie. Dit zal veel hulp bieden bij het coderen van de coördinaten in een patroon. Eerst zal er kort de definitie worden gegeven van De Bruijn sequenties en kort toegelicht worden hoe zo'n De Bruijn sequentie kan worden verkregen. De notatie is gebaseerd op de overzichtspaper van *Salvi en Pagès* [59].

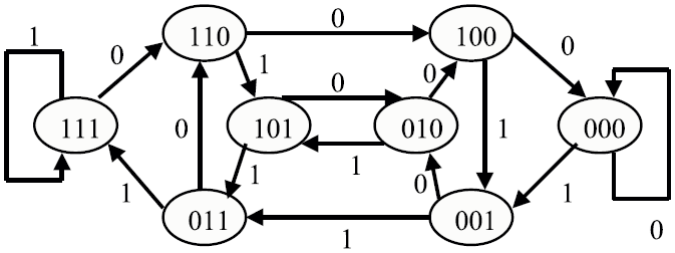
De Bruijn 1 *Een De Bruijn sequentie van order m over een alfabet van n symbolen is een circulaire string van lengte n^m waarbij het elk m-woord exact één keer bevat.*

Het verkrijgen van zo'n sequentie gebeurt op basis van een *De Bruijn graaf*. De definitie [76] van een De Bruijn graaf is de volgende:

De Bruijn 2 *Voor elke positieve gehele getallen $n \geq 2$ en $m \geq 1$ is de De Bruijn graaf $G(n, m)$ een gerichte graaf met een verzameling knopen $\{1, 2, \cdots, n-1\}^m$ dat een edge bevat van knoop $(a_1, a_2, \cdots, a_m)$ naar knoop $(b_1, b_2, \cdots, b_m)$ als en slechts als $b_i = a_{i+1}$ voor $1 \leq i \leq m-1$.*

Het aantal inkomende en uitgaande knopen van zo'n De Bruijn graaf is voor elke knoop m. Het aantal knopen is $N = n^m$. Beschouw nu een n-array shift register met lengte m (n-array shift registers worden beschreven door *Fredricksen* [27]). Er bestaat een één-op-één correspondentie tussen alle verschillende toestanden van het shift register en de knopen van een De Bruijn graaf $G(n, m)$. Er bestaat een edge tussen knoop x_i en x_j als de knoop (de toestand van het register is voorgesteld als knoop) x_j kan worden bereikt van de toestand van knoop x_i met één shift (en een nieuw input getal, dat meestal als label van de edge wordt aangegeven). Een De Bruijn graaf is m.a.w. een state transition diagram van het shift register. Een voorbeeld een de Bruijn graaf vindt u in Figuur 2.7. Een knoop (toestand) van de De Bruijn graaf kan voorgesteld worden als een sequentie van m getallen.

Het opstellen van de De Bruijn sequentie kan nu op twee manieren: het zoeken van *Eulerian circuits* of van *Hamiltonian circuits* [27]. Een *Eulerian circuit* in een graaf G is een pad dat doorheen alle edges gaat van de graaf loopt en terug uitkomt bij de startende knoop. Door de sequentie van edge labels te beschouwen, bekomt men een De Bruijn sequentie van order m. Stel nu Figuur 2.7 waar alle woorden van lengte $m-1$ (met $m = 4$) in de nodes worden geschreven. Een Eulerian cycle is dan bijvoorbeeld 1000010111101001, dit is ook een De Bruijn sequentie van order m. Een *Hamiltonian circuit* wordt in dezelfde graaf gezocht. In dit geval moet het pad niet exact één keer



Figuur 2.7: Een voorbeeld van een de Bruijn graaf. Elke knoop is een shift register en een edge stelt een shift voor op de toestand met een extra getal als input. Een De Bruijn graaf wordt gebruikt om een De Bruijn sequentie op te stellen. [59].

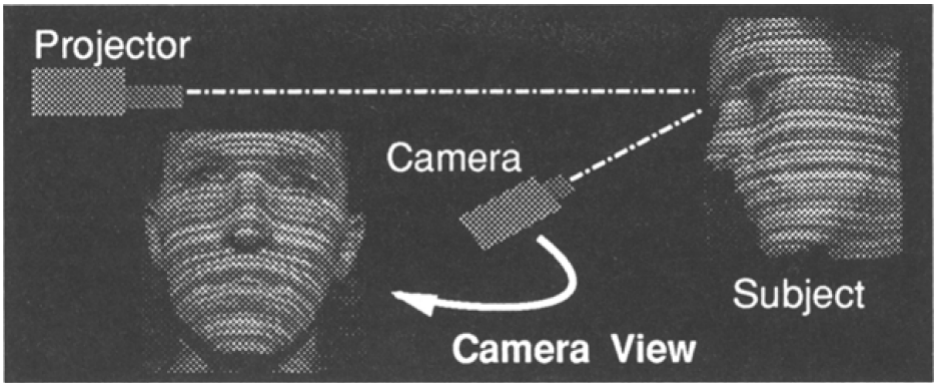
doorheen elke edge gaan, maar wel één keer doorheen elke knoop. Ook hier moet hij starten en stoppen bij dezelfde knoop. Dit geeft dan bijvoorbeeld de sequentie 00101110, dewelke een De Bruijn sequentie is van order $m - 1$. Een De Bruijn sequentie heeft als bijkomende eigenschap dat het een vlakke autocorrelation functie is. Dit wil zeggen dat hij een unieke piek geeft op moment nul. Dit is bijzonder interessant voor het gebruik in template matching, zoals eerder gezien in sectie 2.5.1. Er zullen nu enkele patronen worden gegeven die gebruik maken van De Bruijn sequenties.

Horizontal/vertical slits

Het patroon van *Boyer en Kak* [11] uit sectie 2.5.1 werd door *Hügli en Maïtre* [38] in 1988 verbeterd. Ze gebruiken ook een patroon van horizontale slits met kleuren, maar nu wordt de sequentie van slits samengesteld met behulp van een De Bruijn sequentie. Het voordeel van deze sequentie is dat de globale positie beter kan worden bepaald. In het patroon van *Boyer en Kak* is de positie meer gebonden aan een blok van gekleurde slits om de pixels te indexeren. Dit in tegenstelling tot een patroon dat werd opgebouwd met een pseudo random sequentie, waar de codering onafhankelijk is van de positie. Bij de beschrijving van de doelen binnen structured light werd gezegd dat kleuren nefaste gevolgen kunnen hebben op de resultaten. Over het algemeen kunnen kleuren worden herkend, zolang ze maar genoeg verschillen. Als elke slit een andere kleur moet hebben en er maar een klein aantal verschillende kleuren beschikbaar is, zullen er ook maar een klein aantal slits kunnen worden gebruikt waardoor de resolutie enorm verkleint. Bij het gebruik van een De Bruijn sequentie zullen de slits lokaal kunnen gecodeerd worden met sterk contrasterende kleuren. Doordat spatial neighborhood (color) coding wordt toegepast zullen ook de slits gecodeerd worden met deze lokale kleuren (window grootte w). Het voordeel is dat het aantal indices nu veel groter is als het aantal kleuren (volgens de definitie van De Bruijn sequentis uit sectie 1 geeft dit $kleuren^w$). Dit impliceert nu een grote resolutie bij het coderen van de indices, maar zal een iets kleinere resolutie geven in het verkrijgen van de dieptemap. Om een oppervlak correct te kunnen indexeren moeten er meerdere slits op vallen. Het oppervlak moet dus groter dan als de lokale structuur van het patroon. In het patroon van *Hügli en Maïtre* [38] wordt er gebruik gemaakt van Q kleuren en een window grootte van N. Ze vermijden twee opeen-

volgende slits van dezelfde kleur. Dit geeft een totale sequentie grootte van $Q(Q-1)^{N-1}$.

De pseudo random sequentie die in het patroon van *Hügli en Maïtre* [38] werd gebruikt, wordt ook toegepast in het patroon van *Monks et al.* [55]. Hun opstelling bevat ook één camera en één projector (zie Figuur 2.8). Het patroon gebruikt zes kleuren om horizontale slits in te kleuren, waarbij slits worden onderscheiden door zwarte banden (een De Bruijn patroon van order drie met zes symbolen). Deze sequentie zorgt er voor dat elke subsequentie van drie kleuren maar één keer voorkomt. De zes kleuren zijn samengesteld in de HSV kleurenruimte en verschillen van hun hue component (rood, groen, blauw, geel, magenta en cyan). Bij het decoderen trachten ze een oplossing te vinden voor problemen die kunnen voortkomen uit discontinuïteiten in de scène. Hierdoor kan het zijn dat sommige slits niet zichtbaar zijn en om een positie correct te kunnen decoderen moet de kleur van de slit zelf en de kleur van de slits erboven en eronder correct worden waargenomen.



Figuur 2.8: Het patroon voorgesteld door Monks et al. [55] geprojecteerd op een scène.

Om een oplossing te bieden stoppen ze heel de topologie van de strepen in een acyclische gerichte graaf (DAG of directed acyclic graph). Hierbij is elke node van de graaf een edge segment—dus de overgang tussen twee strepen—en elke edges tussen twee nodes een verticaal zichtbare streep. M.a.w. beschrijft deze graaf alle mogelijkheden om een kleurensequentie te lezen op basis van de edge segementen van de afbeelding. Een pad doorheen de graaf beschrijft een overgang van edge segement naar edge segment. Elke kolom is dus een pad doorheen de graaf.
Eenmaal de graaf volledig is opgesteld, wordt er met behulp van minimum cost matching een overeenkomst gezocht tussen de graaf en de originele kleurensequentie. Over het algemeen kan cost matching voor vele doeleinden worden gebruikt (o.a. speech recognision, matching van dna sequenties, ...). Het is het een bijzonder interessante techniek die ook bij andere patronen kan worden ingezet. Het matching algoritme gaat aan elke operatie, zoals het toevoegen/verwijderen/kleurverandering van nodes, een kost toekennen en vervolgens de minimale kost berekenen om van de waargenomen sequentie terug de referentie sequentie te maken. Het voordeel (en ook tegelijk een nadeel) is dat er gespeeld

kan worden met de kostfuncties. In een omgeving met veel ruis kan bijvoorbeeld de kost van het verwijderen van een node sterk omlaag worden gehaald.

Cost matching algoritme

Het cost matching algoritme voor lineaire sequenties is een veel gebruikte techniek. Daarom zal er nu kort worden ingegaan op de definitie ervan. *Monks et al.* [55] geven een duidelijk beeld hoe het algoritme is opgebouwd.
Stel $T_i, 1 \leq i \leq n$ is de uitgezonden sequentie en $R_j, 1 \leq j \leq m$ de ontvangen sequentie (in dit geval opgenomen met de camera). T^i en R^j zijn alle subsequenties die gebruik maken van alle symbolen tot respectievelijk i en j in de overeenkomstige sequenties. Al de symbolen T_i en R_j zitten in het alfabet S, dus $T_i, R_j \in S$. De kosten van de operaties op de ontvangen sequentie worden gedefinieerd als volgt, waarbij $x, y \in S$:

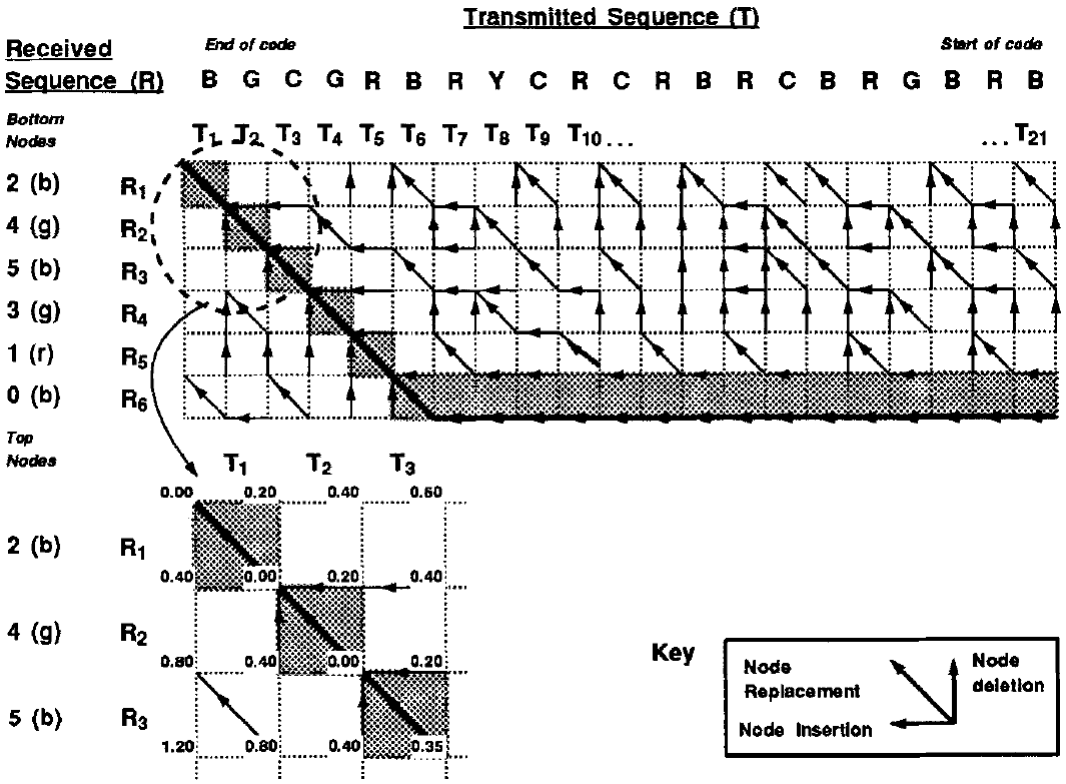
Substitution $x \rightarrow y$	$w(x, y)$
Deletion x	$w_{del}(x)$
Insertion y	$w_{ins}(y)$

Stel nu $d(R, T)$ de minimale kost voor de transformatie van R naar T, gebruik makend van de bovenstaande gewichten (ofnog de Levenshtein difference), en $d(R^j, T^i)$ de minimale kost voor de transformatie van subsequentie R^j naar T^i, cost matching gebeurt dan in twee stappen: cost generation en backtracking. Cost generation gaat recursief doorheen de sequentie om $d(R^j, T^i)$ te vinden en telkens i en j verhogen totdat $d(R^m, T^n) = d(R, T)$ is bereikt. De minimale kost per sequentie $d(R^j, T^i)$ wordt bijgehouden in een $n \times m$ array A. Hierbij bestaat cel $A_{(i,j)}$ uit de kost $d(R^j, T^i)$. De kosten worden berekend door te starten vanaf $d(R^0, T^0)$ en dan vervolgens recursief $d(R^j, T^i)$ te berekenen, gebaseerd op zijn drie voorgangers $d(R^j, T^{i-1})$, $d(R^{j-1}, T^i)$ en $d(R^{j-1}, T^{i-1})$ met behulp van de volgende vergelijking:

$$d(R^j, T^i) = min \begin{cases} d(R^j, T^{i-1}) + w_{ins}(T_i) (\text{insertion of } T_i) \\ d(R^{j-1}, T^i) + w_{del}(R_j) (\text{deletion of } R_j) \\ d(R^{j-1}, T^{i-1}) + w(R_j, T_i) (\text{substitution } R_j \rightarrow T_i) \end{cases}$$

Elke keuze wordt opgeslaan in een andere array D met als celpositie $D(R^j, T^i)$ totdat $d(R, T) = d(R^m, T^n)$ is bereikt. Een volgende stap is backtracking. Backtracking is een populaire manier om op een redelijk efficiënte manier van een zoekprobleem naar een oplossing over te gaan. Er wordt een oplossing gekozen uit alle plausibele oplossingen. Elke oplossing bevat een aantal stappen. Als de stappen niet leiden tot een oplossing van het probleem, moet men terugkeren naar een eerder keuzemoment en een nieuwe keuze maken. In dit geval gaat de backtracking als volgt: er wordt gestart op positie $A_{(m,n)}$ en voor elke cel past men insertion, deletion of substitution toe op R. Dit wordt herhaald volgens het pad dat opgeslagen is in D tot $A_{(0,0)}$ wordt bereikt. M.a.w. encodeert D de minimale kost sequentie van aanpassingen om van R naar T over te gaan.

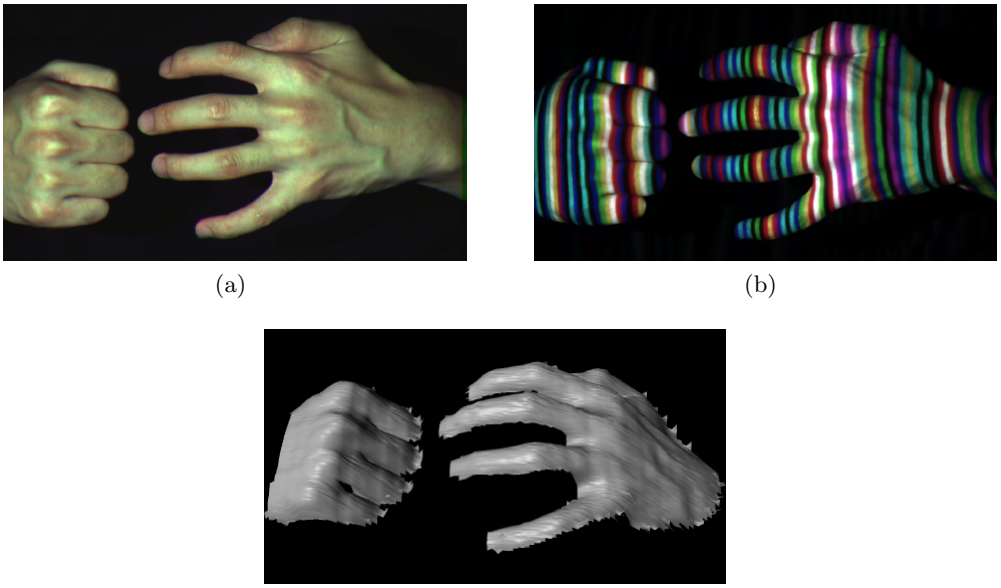
Een voorbeeld wordt gegeven door *Monks et al.* [55], waarbij ze $R = \{brgbgb\}$ willen matchen met $T = \{bgcgrbrycrcrbrcbrgbrb\}$ en een alfabet $s = \{r, g, b, y, m, c\}$. R representeert een verticale slit doorheen het opgenomen patroon en T representeert de code van het patroon dat is geprojecteerd waarbij elke subsequentie van 3 symbolen uniek is en geen twee dezelfde kleuren naast elkaar liggen. In Figuur 2.9 wordt de inhoud van



Figuur 2.9: Een voorbeeld van minimal cost matching door *Monks et al.* [55].

array D uitgetekend, dewelke de minimale kost bevat om R in T te veranderen. De bovenkant van de tabel A zijn de bladeren van de graaf. De kostberekeningen starten daarom van aan onderkant en de backtracking begint aan de bovenkant. Door de pijlen te volgen, startend van elke $d(R^j, T^i)$, krijgt men de backtracking informatie die nodig is om elke T_i te matchen met R_j en in de Figuur 2.9 is het minimale kost pad van $D(R, T)$ grijs ingekleurd.
De moeilijkste en tevens ook belangrijkste stap is het toekennen van de correcte gewichten aan de verschillende operaties om goede resulaten te bekomen, het komt vooral neer op trial-and-error. De paper reikt ook nog een optimaal backtracking algoritme aan om het process te versnellen. Over het algemeen is het een zeer robuuste decodering die goed overweg kan met discontinuïteiten. De auteur zegt ook dat mits een goede optimalisatie en een implementatie van het optimale backtracking algoritme het algoritme aan video snelheid kan werken.

Zhang et al. [89] gebruiken een stripe patroon gebaseerd op een De Bruijn sequentie. Hierbij matchen ze kleurovergangen (randen) van het geprojecteerde patroon met geobserveerde randen in de afbeelding. Ze gebruiken een eigen multi-pass dynamic programming algoritme om zowel high-speed scans van bewegende objecten met een enkel patroon als high-resolution scans van een statische scène. Een voorbeeld van deze aanpak vindt u in Figuur 2.10.



(a) (b)

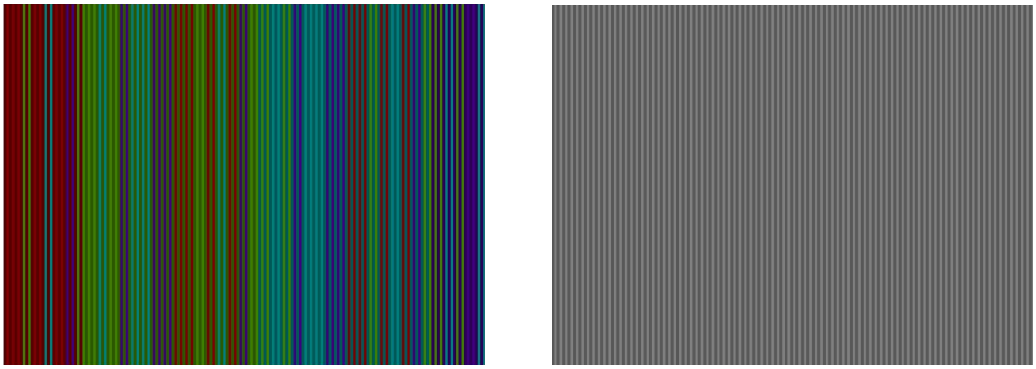
(c)

Figuur 2.10: One-shot method voorgesteld door *Zhang et al.* [89]. (a) Een scène dat ingescand moet worden. (b) het stripe patroon van *Zhang et al.* geprojecteerd op de scène. (c) Na de multi-pass dynamic programming en geometric renconstruction kan de scène vanuit een ander viewpoint worden gerendered.

Doordat er gebruik wordt gemaakt van kleuren zullen er meerdere bits worden gecodeerd in de randen van het patroon. Dit gaat ten koste van een aantal beperkingen betreffende de reflectie van de scène. Bij het zoeken naar correspondenties kunnen meerdere mappings worden gevonden. Dit komt door ruis of occlusies in het opgenomen beeld. Dit wordt opgelost door, in plaats van gewone dynamic programming, een multi-pass dynamic programming algoritme te gebruiken. Dynamic programming (zie sectie 5.5) is een veel gebruikte techniek binnen stereo vision. Toch vereist het een aantal beperkingen op de scène, zoals het afwezig zijn van discontinuïteiten. Een tweede doorgang van het algoritme zal de oppervlakken met discontinuïteiten herstellen. Indien men meer accurate resultaten wil verkrijgen, is het perfect mogelijk meerdere beelden tegelijk te gebruiken. Het algoritme gaat er van uit dat de positie van camera en projector ten opzichte van elkaar gekend zijn (zie sectie 5.1) en dat het inkomende beeld

is recht getrokken (zie sectie 5.3). De stripes van het patroon worden voorgesteld als $P = (p_0, p_1, \cdots, p_N)$. Belangrijk is dat *Zhang et al.* [89] niet geïnteresseerd waren in de kleuren van de stripes zelf, maar wel in de overgang van kleuren. Een sequentie ziet er dan uit zoals $Q = (p_0, p_1, \cdots, p_{N-1})$, met $q_j = (q_j^r, q_j^g, q_j^b)$, waarbij elk kleurkanaal $-1, 0$ of 1 kan aannemen. Deze stellen respectievelijk het vallen, constant blijven of stijgen van het kleurkanaal voor. Belangrijk is om de keuze van de sequentie zo goed mogelijk te kiezen. Daarom opteerden *Zhang et al.* [89] ook voor een de Bruijn sequentie. In totaal zijn er 27 unieke overgangen mogelijk, maar omdat $(0, 0, 0)$ niet echt een overgang is zijn dit er 26. Als de verschillende RGB waardes enkel een 1 of 0 kunnen aannemen en de opeenvolging van kleuren wordt gedefinieerd als een XOR operatie op de RGB van de vorige en een bepaalde term $d_l \in \{001, \cdots, 111\}$ met daarbij een initiële kleur $p_0 \in \{000, \cdots, 111\}$, zijn er in totaal 7 transities mogelijk. *Zhang et al.* hadden maar 125 stripes nodig en werkten dus met een sequentie van orde 5 en window grootte 3 ($n^m = 5^3 = 128$). Hierbij zijn 110 en 111 weggelaten omdat simultaan groen en rood overgangen het meeste last hebben van errors. De opgenomen scanline wordt voorgesteld als $E = (e_0, e_1, \cdots, e_{M-1})$ met $e = (e^r, e^g, e^b)$. Deze wordt bij het decoderen voor elke stripe gezocht door alle gradiënten van de scanline uit te middelen en zo de juiste kleurwaarde te bekomen. Er is geweten dat er door ruis of andere problemen verkeerde labels kunnen worden toegekend, waardoor er meerdere labels per stripe ontstaan. Dit wordt opgelost door het toestaan van *multiple hypotheses*. Bij dit algoritme wordt niet een unieke label toegekend aan elke stripe in de afbeelding, maar wel een een label met daarbij een kans op overeenkomst. Uiteindelijk kan op basis van deze kansen de juiste label worden uitgekozen, waarbij rekening wordt gehouden met occlusions, shaduwen en discontinuïteiten. Zoals u kunt zien in sectie 5.5, maakt dynamic programming gebruik van het optimale pad. Het nadeel ervan is dat het optimale pad niet meer kan worden gevonden als het oppervlak niet monotoon is (kan voorkomen bij occlusies of discontinuïteiten).

In sectie 2.3 werd uitgelegd dat in de klassieke papers er vaak een onderscheid wordt gemaakt tussen multi-slit en stripe patronen. De eerste categorie is bruikbaar voor intensiteitspieken te vinden in de afbeelding en de tweede heeft als doel het zoeken van randen. *Salvi et al.* [60] ontwikkelden daarom een nieuw soort algoritme op basis van gekleurde patronen dat zowel intensiteitspieken als randen kan terugvinden, zonder verlies van nauwkeurigheid en waarbij het aantal kleurwaardes zo klein mogelijk wordt gehouden. Het hybride patroon zal in de RGB ruimte gebruikt worden als een stripepatroon en in het intensiteitsprofiel als een peak-based patroon. De de Bruijn sequentie heeft nu maar $n = 4$ verschillende hue waardes nodig om een patroon te bekomen van 128 stripes met een window grootte van $m = 3$. Volgens de definitie zou dit normaal een sequentie geven van $n^m = 64$, maar omdat er nu een afwisseling wordt gemaakt van hoge intensiteit en middelmatige intensiteit kunnen er dubbel zoveel stripes worden gecodeerd. *Salvi et al.* [60] geven een uitgebreide uitleg over hoe het patroon moet worden opgesteld en hoe de labeling gebeurt.
Om te kunnen beginnen met decoderen moeten eerst de stripes worden gesegmenteerd. Hiervoor wordt het intensiteitsprofiel gebruikt, omdat deze afwisselend per stripe een



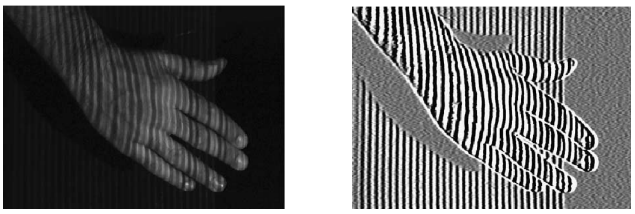
(a) (b)

Figuur 2.11: Het hybride one-shot patroon van *Salvi et al.* (a) Hybride patroon in de RGB ruimte. (b) Intensiteitsprofiel van het hybride patroon.

hoge of medium intensiteit heeft. Gegeven $M(i) = max(R(i), G(i), B(i))$ met $i = 1 \cdots \text{width(scan-line)}$, dan kan met de tweede afgeleide de verschillende stripes worden gevonden met behulp van

$$g(i) = \sum_{c=1}^{o/2} (M(i+c) - M(i-c))$$

waarbij g de linearie afgeleide is dewelke twee keer moet worden toegepast om de tweede afgeleide te bekomen. De order o moet worden gekozen in functie van de breedte van de afzonderlijke stripes. Belangrijk is dat de breedte van de stripes altijd groter zijn dan de order o. Een voorbeeld van de tweede afgeleide is gegeven in Figuur 2.12. Voor



(a) (b)

Figuur 2.12: Segmentatie van de stripes m.b.v. de tweede afgeleide. (a) M channel van het geprojecteerd hybryde patroon van *Salvi et al.* [60]. (b) Gesegmenteerde stripes gebruik makend van de tweede afgeleide.

het decoderen moet nu enkel nog voor elke stripe de hue waarde worden bepaald. Als laatste wordt er dynamic programming (zie sectie 5.5) toegepast om de juiste labels te verkrijgen. De auteurs houden niet veel rekening met problemen zoals het wegvallen van stripes of foute labeling van stripes en concluderen enkel dat hier nog een oplossing

voor moet gezocht worden. Eventueel zou ook het cost-matching algoritme van *Monks et al.* [55] gebruikt kunnen worden om de juiste labels te verkrijgen.

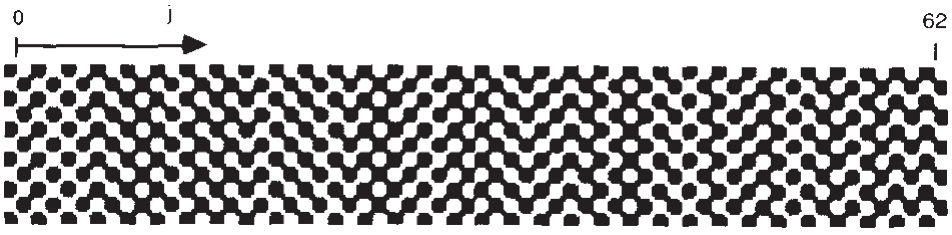
Grid patronen

Vuylsteke en Oosterlinck [81] hebben op basis van De Bruijn sequenties een binair grid patroon ontwikkeld. Er worden in totaal 63 kolommen ge-encodeerd in een enkel patroon. Het algoritme kan in theorie dus ook goed overweg met een bewegende scène. Het patroon is opgebouwd als een schaakbord waarbij de kolom van elke gridpoint wordt gecodeerd. De volgende De Bruijn sequentie is gebruikt (order 6 en lengte 63), weergegeven in vergelijking 2.1.

$$\{c_k\} = 111111000001000011000101001111010001110010010110111011001101010 \quad (2.1)$$
$$\{b_k\} = \{c_{k-17}\}$$

Hierbij is $\{b_k\}$ dezelfde sequentie als c_k geshift met $\varphi = 17$ posities. Beide sequenties worden gecombineerd in een binair patroon. Dit geeft een gecombineerde window van 2×3. Een voorbeeld van het patroon vindt u in Figuur 2.14. Enkel de kolommen worden gecodeerd. Hierdoor hebben blokken die ondereen liggen hetzelfde codewoord. Dit is geïllustreerd in Figuur 2.15b. Dit effect wordt bekomen door de codewoorden zo te genereren zodat eerst de elementen van c_k komen en dan pas de elementen van b_k. Het codewoord voor Figuur 2.15b is bijgevolg $((c_3, c_4, c_5), (b_3, b_4, b_5))$. Nu moeten de elementen van c_k nog kunnen onderscheiden worden van de elementen van b_k. Dit wordt gedaan door verschillende representaties binnen het patroon te gebruiken. De gridpoints geven de states 0 en 1 aan. Voor state 0 wordt het gridpoint met een donkere vlek voorgesteld, voor state 1 met een witte vlek. Verder worden de aanliggende cellen van het gridpoint ingekleurd naargelang het overeenkomstige sequentie element tot c_k of b_k hoort. In totaal zijn er vier mogelijke primitieven die worden weergegeven in Figuur 2.15a. De binaire states voor c_k zijn gelabeled als $0+$ en $1+$ en die van b_k met $0-$ en $1-$.

Bij het decoden moet voor elke window van 2×3 het codewoord worden gezocht. Dit



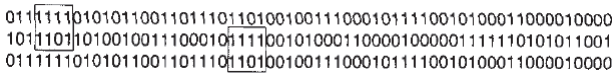
Figuur 2.13: Het patroon voorgesteld door *Vuylsteke en Oosterlinck* [81].

resulteert in de posities van de kolommen. Om een codewoord te zoeken moeten eerst de gridpunten gekend zijn. Het zoeken van deze punten is gebaseerd op correlatie met

het volgende referentie patroon:

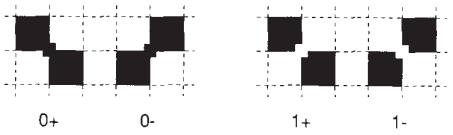
$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 \end{bmatrix}.$$

Vuylsteke en Oosterlinck [81] beweren dat het gebruik van rechthoekige windows het voordeel geniet van dat ze minder last hebben van grote verschillen in oppervlak. Een kleinere omgeving beschouwen heeft een kleinere kans om verstoord te worden dan een grotere. Het is eventueel mogelijk om nog meerdere rijen te beschouwen. Het gebruik van meerdere rijen zal ook extra problemen introduceren. Stel er worden t rijen gebruikt. Het is mogelijk dat voor elk window per positie er t dezelfde windows bestaan, geshift over de verschillende rijen. Figuur 2.15 toont dit aan met $t = 2$. Dit probleem kan enkel

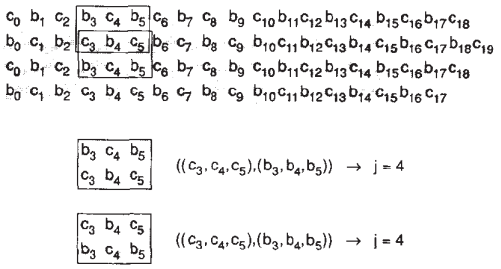


Figuur 2.14: Er kunnen ambiguïteiten ontstaan als het patroon gebruik maakt van meerdere sequenties. De aangeduide windows behoren tot een verschillende kolom en hebben toch hetzelfde codewoord [81].

worden opgelost als de rij indexen gekend zijn. Het is een robuust algoritme dat overweg kan met een uitgebreid aantal scènes. Volgens hun resulaten heeft het algoritme nog een speedup factor van 300 nodig om real-time te kunnen werken. Dit is voorlopig nog een beperking van het systeem. Daarnaast ondervindt het ook nog wat problemen van texture interferentie. Hiervoor zijn er een aantal oplossingen voor handen, met name het gebruik van kleuren in het patroon.
Ook *Pagès en Salvi* [61] ontwikkelden een grid patroon om bewegende objecten te kunnen scannen. Ze integreerden code-informatie in de cross-points van het grid. Daarbij probeerden ze vooral de focus te leggen op het corrigeren van decoding errors. Dit doen ze door beide assen van het patroon te coderen. Het voordeel van deze aanpak is dat het de nauwkeurigheid verhoogt, het horizontale en verticale slits kan reconstrueren en dat het een error detectie en eventueel correctie kan toepassen. Hierdoor wordt één van de grote problemen bij het coderen in een enkel patroon opgelost, namelijk de onmogelijkheid om discontinue oppervlakken te reconstrueren. De coderingsaanpak wordt in een andere paper van *Salvi et al.* [71] beschreven. Het is een gridpatroon dat wordt samengesteld uit gekleurde verticale en horizontale slits met elk een bepaalde dikte over een zwarte achtergrond. De volgorde van kleuren voor de horizontale en verticale slits wordt bepaald door een De Bruijn sequentie. De horizontale en verticale slits gebruiken dezelfde De Bruijn sequentie, enkel de verzameling van kleuren verschillen. Al de pixels



(a) De vier primitieven gebruikt in het patroon van Vuylsteke en Oosterlinck



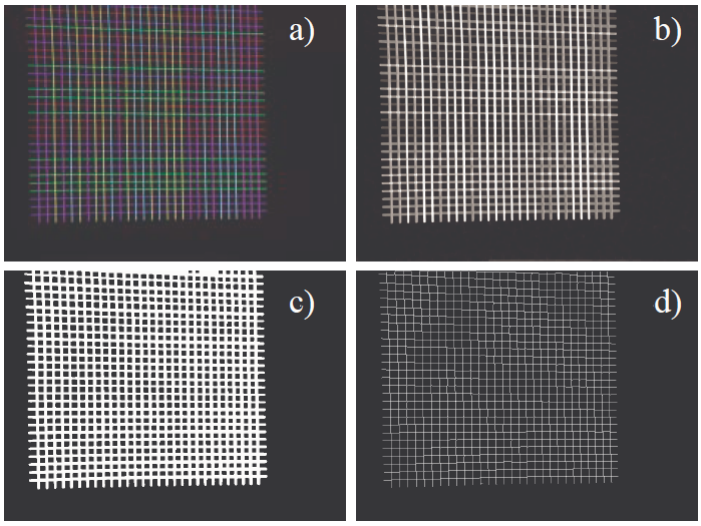
(b) Overeenkomsten van primiteiven bij het patroon van Vuylsteke en Oosterlinck

Figuur 2.15: Het patroon van *Vuylsteke en Oosterlinck* [81] is opgebouwd uit vier primitieven. (a) De vier primitieven [81]. (b) Er worden twee binaire sequenties bits-gewijs toegekend aan het grid, zodat elke blok van 2×3 bits een unieke code heeft voor elke positie. U kan zien dat om de twee rijen een blok wordt herhaald [81].

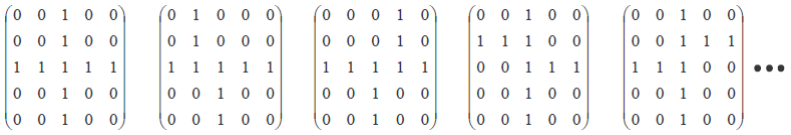
die behoren tot een horizontale of verticale slit krijgen een bepaald codewoord toebedeeld. Elke cross point bevat twee codewoorden waardoor ze ook meer accuraat kunnen worden gereconstrueerd. Er wordt met andere woorden redundante data toegevoegd aan het patroon (twee codewoorden per cross point). Deze redundante informatie zal uitermate belangrijk zijn voor het detecteren en zelfs het oplossen van fouten. De horizontale slits bestaan uit de kleuren rood, groen en blauw en de verticale slits bestaan uit de kleuren geel, cyan en magenta (het belangrijkste is dat de hue waardes voldoende verschillen). De De Bruijn sequentie heeft een window grootte van drie en een orde van drie kleuren. Het patroon is weergegeven in Figuur 2.16a. Op de waargenomen beelden wordt (mits offline processing: color calibration, radial correction,...) segmentation toegepast. Hiervoor wordt een Sobel filter [30] gebruikt. Er worden vervolgens nog drie *Close* morphological operations toegepast. Deze *merge* operatie zorgt voor het dichten van hiaten in opeenvolgende lijnen. Om het skelet van het grid te bekomen moet enkel nog een *thinning* operatie worden uitgevoerd. Meer details over deze operaties vindt u in het boek *Digital Image Processing* van *Gonzalez en Woods* [30]. Eens het skelet is gekend, kunnen er cross points worden gezocht. Dit gebeurt ook met binary masks (zie Figuur 2.17), dewelke geconvuleerd worden met de afbeelding.

Elke pixel van de afbeelding, waar één van de masks groter is dan een bepaalde threshold, zal worden beschouwd als een cross point van het grid. Nu alle (of zo goed als alle) cross points gekend zijn, moet er enkel nog een mapping op het oorspronkelijke patroon worden berekend. Het komt er op neer de twee codewoorden van het crosspoint te zoeken, deze bepalen namelijk uniek de plaats binnen het patroon. Het zoeken van deze codewoorden gebeurt met het volgende algoritme:

1. Opstellen van een *cross point adjacency graaf*. Dit gebeurt door vanaf elke cross point te starten en de edges van het skelet te volgen in vier richtingen om zo zijn naaste buren te bekomen.



Figuur 2.16: Het grid patroon voorgesteld door *Salvi en Pagès* [61]. (a) het geprojecteerd patroon. (b) edge detection met Sobel, drie close operaties. (c) binair maken. (d) thinning operatie tot het skelet is bereikt.



Figuur 2.17: De binary masks die gebruikt worden voor het detecteren van cross points van een grid [61].

2. Voor elke cross point, de kleur van de horizontale en verticale slit berekenen. Deze wordt tijdens het tracken bekomen door middel van uitmiddeling langs de scan-line.

3. Decodeer alle cross points waarvan de beide codewoorden zijn gevonden met behulp van de adjacency graaf, meer bepaald het zoeken van de correspondentie in het referentie patroon en met behulp van triangularisatie de diepte berekenen. Dit decoderen gebeurt enkel als de buren ook gekend zijn.

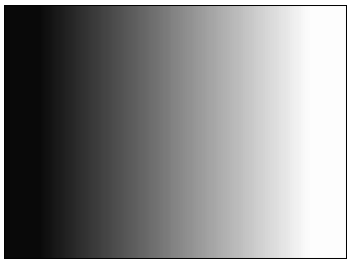
4. Overdragen van codewoorden naar buren die nog niet gedecodeerd zijn.

5. Correctie van inconsistente codewoorden. Dit gebeurt door de huidige codewoorden te vergelijken met zijn buren en kijken of hij wel correct is.

Zonder het overdragen en het corrigeren van inconsistenties zou het algoritme geen kleine oppervlakken of grote discontinuïteiten aankunnen. Dankzij het algoritme worden deze beperkingen verwijderd en vindt men op een robuuste manier de codewoorden. Daarnaast is een real-time implementatie mogelijk.

2.6 Directe codering

Bij directe codering tracht men een patroon te ontwikkelen zodat elke pixel uniek wordt gecodeerd [59]. Dus het hele codewoord voor een punt wordt opgenomen in een enkele pixel. Dit kan op twee manieren worden gedaan: ofwel een heel groot bereik van kleuren of grijwaares, ofwel periodiciteit invoeren. Het nadeel van deze technieken is dat ze zeer gevoelig zijn aan ruis [59]. De minste afwijking kan een verkeerd codewoord geven. Daarnaast is de waargenomen kleur ook sterk afhankelijk van de kleur van het oppervlak van de scène. In praktijk zal deze techniek enkel goed werken als het object een monotone kleur heeft. Kleurcalibratie is bij deze techniek uitermate belangrijk. Er zijn twee categorieën van direct coding. De eerste is gebaseerd op grey levels en zal de pixels coderen in verschillende tinten van grijs. De tweede techniek gebruikt het brede spectrum van kleuren. *Carrihill en Hummel* [13] hebben een patroon ontwikkeld op basis van grey levels. Het is een gradiënt patroon dat van zwart overgaat in wit. Er wordt voor elke pixel een ratio gemeten tussen de pixel waargenomen met het geprojecterd patroon en de pixel waargenomen met een constante belichting. Op basis van deze ratio kan men de kolom bepalen binnenin het patroon. Het patroon vindt u in Figuur 2.18. *Carrihill en Hummel* ondervonden zeer veel ruis in hun resultaten. Ook moeten er telkens twee afbeeldingen worden geprojecteerd waardoor het algoritme niet overweg kan met bewegende beelden.

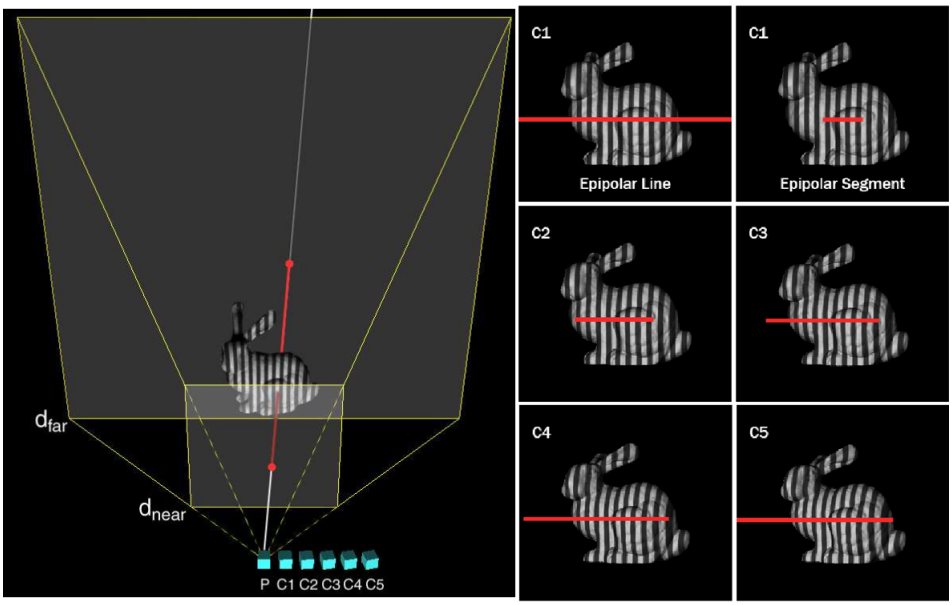


Figuur 2.18: Het grey level patroon van *Carrihill en Hummel*. Afbeelding van *Pagès et al.* [59].

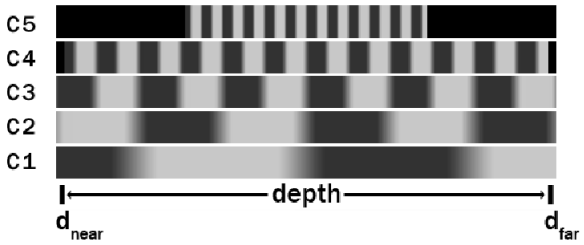
2.7 Viewpoint codering

Young et al. [86] introduceerden in 2007 een nieuwe techniek waar tijdsgecodeerde patronen (zoals in sectie 2.4) worden vervangen door viewpoint codes. Ze tonen aan dat elke toegevoegde camera een enkel patroon van de temporele binaire code kan vervangen. Er wordt dus geen gebruik gemaakt van time-multiplexing of spatial neighborhood coding. Er worden een aantal camera's op vooraf geselecteerde posities opgesteld en de projector toont een stripe patroon met hoge frequentie. Viewpoint coded structured light gaat gebruik maken van epipolaire segmenten. Een epipolair segment is een een deel van een epipolaire lijn waar de correspondentie kan worden gevonden. Door deze beperking van oplossingen zal het matching algoritme ook veel efficiënter werken i.p.v.

het gebruik van de hele ruimte. De opstelling is geïllustreerd in Figuur 2.19. Door de epipolaire segmenten te projecteren op elk van de geplaatste camera's, krijgt elke camera een verschillend patroon. De combinatie van deze verschillende patronen zorgt voor een unieke code. Het voordeel van deze aanpak is dat het resultaat niet meer afhangt van temporele of spatiale coherentie. Belangrijk is dat alle camera's op een correcte positie zijn geplaatst. Indien dit niet het geval is, zullen de codes niet uniek genoeg zijn om zonder errors de juiste diepte te berekenen. Om hieraan tegemoet te komen, geven ze een algoritme voor het berekenen van de correcte cameraposities. Een voorbeeld van goed gekozen posities—die correcte viewcodes geven—ziet u in Figuur 2.20. De sterkte van deze techniek is dat er een directe codering gebeurt die ook goed overweg kan met beweging. Daarnaast is er ook geen zware decoderingsstap noodzakelijk omdat er niet in de spatial neighborhood moet worden gezocht. Het nadeel is dat het een grotere opstelling vereist omdat er meerdere camera's nodig zijn. Om het aantal camera's in te perken is het mogelijk viewpoint coding te combineren met een gekleurd stripe patroon met bijvoorbeeld c kleuren. Op deze manier is het mogelijk N unieke dieptes te bepalen met $\log_c N$ camera posities.



Figuur 2.19: Plaatsing van de viewpoint coderings camera's [86]. Het epipolaire segment is een enkele geprojecteerde pixel (aangegeven in rood) en is de snijding tussen een straal vanuit de projector P en het object. Rechts bovenaan links ziet u de standaard epipolaire lijn waarin wordt gezocht naar de juiste correspondentie en aan de rechterkant ziet u enkel het epipolaire segement. Dit segment geeft dus een veel kleiner bereik waarin moet worden gezocht naar correspondenties. De rest van de rechterkant van de figuur toont ook de epipolaire segmenten voor de camera's C2, C3, C4 en C5.



Figuur 2.20: Door de correcte posities van de camera's, vormen de geprojecteerde epipolaire segmenten van de verschillende viewpoints een unieke binaire code [86].

2.8 Discussie

Patronen die gecodeerd worden in het temporele domein zijn over het algemeen zeer krachtig. Ze werken zeer robuust en worden veel gebruikt om betrouwbare en nauwkeurige dieptemappen op te stellen. Het grote nadeel is dat ze enkel overweg kunnen met statische scènes, waardoor ze voor veel opstellingen niet bruikbaar zijn. In de doelstelling van deze thesis is het net de bedoeling om vrij met een camera en projector rond te bewegen en zo incrementeel de scène op te bouwen. Temporeel coding voldoet dus niet aan één van de hoofdeisen.

Meer bruikbaar zijn de one-shot technieken. Deze zijn gemaakt om ogenblikkelijk een dieptemap op te stellen. Zolang er op elk ogenblik een dieptemap kan worden opgebouwd, kan er met de camera doorheen de scène worden rondgewandeld en nieuwe 3D informatie aan de totale 3D-wolk worden toegoegd. Het tweede belangrijkste criteria binnen onze doelstelling dat moet worden voldaan is het overweg kunnen met discontinuïteiten. Bij spatial neighborhood technieken is er het nadeel dat de omgevende punten ook in rekening moeten worden gebracht om tot unieke posities te komen. Dit impliceert dat het oppervlak groter moet zijn dan de omgeving die gebruikt wordt om de posities te coderen. Daarnaast moet het oppervlak ook monotoon zijn. Gelukkig zijn er enkele technieken ontwikkeld die proberen hieraan tegemoet te komen. Zo gebruiken *Zhang et al.* [89] een multi-pass dynamic programming algoritme om het discontinuïteitsprobleem op te lossen. Een andere goede oplossing voor dit probleem is het gebruik van een cost matching algoritme, voorgesteld door *Monks et al.* [55]. Het voordeel van deze twee technieken is dat ze zich niet enkel beperken tot de algoritmes van *Zhang en Monks*, maar ook inzetbaar zijn voor andere patronen. De twee algoritmes zijn goed inzetbaar om binnen onze doelstelling tot een goede oplossing te komen. Daarnaast is het bij een one-shot techniek ook aan te raden een formele wiskundige techniek te gebruiken voor de opbouw van het patroon. Een De Bruijn patroon zal al direct een aantal algemene problemen zoals periodiciteit oplossen.

Als laatste is er ook nog de categorie van direct coding. Deze zijn maar weinig bruikbaar omdat ze veel ruis bevatten in de resultaten. Daarnaast vergen ze ook te veel

eisen betreffende de scène zelf. Viewpoint coded structured light is wel een goede directe coderingstechniek. Het nadeel ervan is dat er meerdere camera's moeten worden gebruikt om bewegende scènes te kunnen scannen. Doordat er bewust gekozen is voor het gebruik van één camera is deze dus niet toepasbaar.

Hoofdstuk 3

Registratie methodes

Het is niet mogelijk om een volledig object in één keer in te scannen wegens geometrische en topologische beperkingen. Het is namelijk onmogelijk rechtstreeks al de 3D informatie te bekijken vanuit één enkele invalshoek. Vaak wordt een object vanuit een aantal camerastandpunten waargenomen, hieruit wordt per camerastandpunt een dieptemap verkregen en zo krijgt men na het inscannen een verzameling van dieptemappen. Nu is het wenselijk deze verschillende dieptemappen te kunnen samensmelten in één algemene driedimensionale voorstelling van de scène. Het samensmelten van de verschillende 3D-data wordt *point cloud alignment* of meer algemeen *registratie* genoemd.

3.1 Doel

Het doel van point cloud alignment is dus de verschillende puntenwolken die bekomen worden van de scanner te aligneren met een bepaald assenstelsel. Dit aligneren met een bepaald assenstelsel wordt in de literatuur ook *de registratie van 3D informatie in een model* genoemd. Van zodra deze registratie is uitgevoerd, kan de volledige scène samen worden gerendered [7].

Er zijn een aantal dure en minder dure technieken ontwikkeld om deze registratie te verwezenlijken. In dit hoofdstuk wordt er kort besproken welke verschillende aanpakken er mogelijk zijn. Er zijn dure technieken die dikwijls specifieke hardware of menselijke actoren nodig hebben om de dieptemappen met elkaar te aligneren. Dit zijn meestal niet de gewenste technieken om te gebruiken in een eigen implementatie van een 3D scanner. Deze zijn specifiek ontwikkeld voor de industrie, waar men een hoge nauwkeurigheid verwacht. Daarnaast zijn er ook een aantal algoritmes die automatisch de dieptemappen aligneren binnen een beperkte tijd en zonder inbreng van een menselijke actor. De bekendste automatische registratie techniek is het Iterative Closest Point algoritme [9, 15]. Bij het onderzoek naar bestaande algoritmes beperkt deze thesis zich op het gebruik van puntenwolken of oppervlakken. Feature detection, markers of andere geometrische detecties zullen buiten beschouwing worden gelaten. Een theoretische visie op registratie wordt gegeven door Brown [12]. Hij bespreekt in detail de vervorming die er aanwezig kan

zijn tussen twee views en deelt de registratie technieken op in point mapping methodes en Fourier methodes.

3.2 Accurate Tracking

In een eerste aanpak wordt gebruik gemaakt van specifieke sensoren. Aan het scantoestel wordt een sensor gehangen die de positie en oriëntatie van het toestel met een grote nauwkeurigheid vastlegt [7]. In sommige gevallen wordt er gebruik gemaakt van een robotarm. Deze kunnen met een grote nauwkeurigheid verschillende standpunten innemen rond een in te scannen scène. Zo weet men exact waar de robotarm zich bevindt. Een robotarm gebruiken, zal de bewegingsruimte sterk beperken. Het is enkel mogelijk een aantal vaste posities rond het object in te nemen. Hierdoor is het onmogelijk om vrij doorheen een scène te lopen en incrementeel het model aan te vullen. Daarnaast is het gebruik van hardware zoals sensoren of robots een dure aangelegenheid. Zulke technieken zullen daarom voornamelijk voor industriële doeleinden worden gebruikt. Andere algoritmes maken gebruik van een draaischijf waarop het in te scannen object wordt geplaatst. Ook hier is er een beperkte bewegingsruimte en mag het object niet groter zijn dan de draaischijf zelf. Het inscannen van grote kamers is dus niet mogelijk.

3.3 Optical Tracking

Een camerastandpunt kan ook vastgelegd worden op basis van optical tracking [7]. Er worden features of markers (bijvoorbeeld barcodes) aangebracht in de scène. Op basis van deze markers kan de positie in de scène worden berekend. Ook dit zijn vaak dure technieken en zijn moeilijk toepasbaar in grote scènes. Tegenwoordig zijn er goedkope algoritmes beschikbaar om markers te herkennen, maar omdat deze thesis enkel werkt op basis van puntenwolken en dus geen markers toevoegt aan de scène, zal er niet dieper op optical tracking worden ingegaan.

3.4 Interactieve Alignment

Bij interactieve alignment gaat een menselijke actor twee overlappende dieptemappen bekijken en indentificieert hierop drie of meerdere overeenkomstige features. Op basis van deze overeenkomsten wordt de transformatie bepaald [7]. Doordat er voortdurend een persoon moet tussenkomen, zal het proces veel tijd in beslag nemen. Daarom is er nood aan een automatische methode die alles volledig zelfstandig uitvoert, onafhankelijk van de grootte, kijkrichting of vorm van de scène.

3.5 Automatic alignment

In de vorige secties werd aangehaald dat het gebruik van een menselijke tussenkomst vaak een bottleneck vorm en dat alternatieve technieken duur zijn in opzet. Het is

daarom wenselijk methodes te gebruiken die het uitlijnen volledig automatiseren. Eén van de meest gebruikte automatische registratie techniek is *Iterative Closest Point* (ICP). De verschillende automatische registratie technieken worden in het volgende hoofdstuk uitgebreid besproken.

Hoofdstuk 4

Automatische Registratie

Zoals aangehaald in vorig hoofdstuk, gaat een automatisch registratie algoritme de alignatie van de coördinatensystemen van verschillende puntenwolken volledig zelf doen, zonder tussenkomst van een menselijke actor. Het populairste algoritme hiervoor is *Iterative Closest Point*. Deze wordt uitgebreid in de volgende sectie besproken, tesamen met al zijn variaties om het algoritme te optimaliseren. Het is ook mogelijk in de parameterruimte te zoeken naar een mogelijke kandidaten voor de transformatie met behulp van een particle filter. Deze particle filter moet ook een methode bevatten om de kandidaten te evalueren. Vervolgens bespreekt het hoofdstuk het Levenberg-Marquardt algoritme dat het minimaliseren van de evaluatiefunctie zou verbeteren ten opzichte van ICP. De laatste sectie bespreekt een variatie dat op basis van Extenden Gaussian Images de registratie van de rotatie afzondelijk uitvoert. Het hoofdstuk wordt afgesloten met een discussie van de verschillende technieken en hoe ze toepasbaarbaar zijn binnen de gemaakte doelstellingen van deze thesis.

4.1 Iterative Closest Point

Iterative Closest Point (ICP) is een algoritme dat werd voorgesteld door *Besl en McKay* [9] in 1992 en *Chen et al.* [15] in 1991. Als er een lijst van range afbeeldingen beschikbaar is, bijvoorbeeld verkregen met structured light, wil men deze vaak registreren in één globaal model met één enkel coördinatensysteem. Dit wordt verwezenlijkt door de transformatie, die nodig is om een dieptemap in het model te laten passen, voor elke dieptemap te berekenen. Beter is de transformatie tussen opeenvolgende dieptemappen te berekenen, omdat deze maar weinig van elkaar afwijken. Het passen van een dieptemap in een model wordt de registratie in een 3D model genoemd. Iterative Closest Point is een iteratief algoritme om de best mogelijke transformatie te vinden. Hierbij maakt het gebruik van een zogenaamde error metric functie die de transformatie evalueert. Het minimaliseren van deze evaluatiefunctie geeft de best mogelijke transformatie. Dit minimaliseren is beter dan te zoeken doorheen de transformatieruimte naar de best mogelijke transformatie, zoals wordt gedaan bij particle filters (zie sectie 4.2). Iterative Closest Point verwacht een initiële transformatie dewelke al een goede benadering is van de gewenste

transformatie. Hierdoor zal het algoritme sneller convergeren naar het juiste resultaat.

Om de bomen door het bos te zien, zal eerst het originele algoritme worden gegeven, ontwikkeld door *Chen et al.* [15]. Dit zal de werking van Iterative Closest Point verduidelijken. De methode van *Chen et al.* beperkt zich tot oppervlakken en afstanden tot oppervlakken van twee dieptemappen en gebruikt normaal-metrieken om de iteraties te verfijnen. Er is al veel onderzoek gedaan naar de verschillende stappen binnen ICP. Zo geven *Best en McKay* [9] een uitgebreide wiskundige uitleg over alle mogelijke representaties die de geometrische data kan aannemen (puntenwolken, lijnsegmenten, curves, geparametriseerde curves, triangles, oppervlakken, geparametriseerde oppervlakken, $\cdots$) en welke verschillende metrieken kunnen gebruikt worden om een evaluatiefunctie hiervoor op te stellen. Na dit algoritme zullen de verschillende stappen genuanceerd worden en wordt er dieper ingegaan op de alternatieve metrieken die beschikbaar zijn om transformaties te zoeken en te evalueren.

4.1.1 ICP volgens Chen et al.

Twee oppervlakken worden beschouwd als geregistreerd als ze samenvallen. Meer algemeen zullen twee oppervlakken geregistreerd zijn als alle punten (p_i, q_i), van de twee puntenwolken die hetzelfde oppervlak voorstellen, worden samengebracht door een vaste transformatie T, zodat

$$\forall p_i \in P, \exists q_j \in Q | \, \|Tp_i - q_j\| = 0 \tag{4.1}$$

of nog

$$D(P,Q) = \int \int_{\Omega} \|Tp(u,v) - q(f(u,v), g(u,v))\|^2 \, dudv = 0 \tag{4.2}$$

waar: $p(u,v) \in P; q(u,v) \in Q$; P en Q zijn twee afbeeldingen van hetzelfde oppervlak; $(u,v) \in \mathbb{R} \times \mathbb{R}$ een punt op de afbeelding P en Q; f en g zijn beide de mappings van de correspondenties (dus $p(u,v)$ en $q(f(u,v), g(u,v))$ stellen hetzelfde oppervlaktepunt voor); Tp_i past de transformatie T toe op P_i; Ω is de regio van overlappen van P en Q. In homogene coördinaten wordt T voorgesteld als

$$\begin{aligned} T &= T(\alpha, \beta, \lambda, t_x, t_y, t_z) \qquad (4.3) \\ &= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & t_x \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & t_y \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Deze transformatie heeft 6 vrijheden. De bedoeling is de optimale waarde voor deze 6 parameters te vinden zodat $D(P,Q)$ uit vergelijking 4.2 minimaal is. Een lokaal minimum wordt gevonden in de punten waar de afgeleide 0 is. Dus $dD(P,Q) = 0$. De kandidaat minimim extremum is bijgevolg de oplossing van volgende partiële afgeleiden:

$$\frac{\partial D}{\partial \alpha} = 0, \frac{\partial D}{\partial \beta} = 0, \frac{\partial D}{\partial \gamma} = 0, \frac{\partial D}{\partial t_x} = 0, \frac{\partial D}{\partial t_y} = 0, \frac{\partial D}{\partial t_z} = 0 \tag{4.4}$$

Het probleem is dat dit een niet-lineaire operatie is. Er moet dus een iteratieve manier worden gebruikt om dit op te lossen. Daarnaast is het ook niet altijd zeker dat een globaal minimum wordt bereikt. Een lokaal minimum voldoet altijd aan de vergelijkingen van 4.4, maar een kandidaat voor het oplossen van de vergelijkingen is niet altijd een minimum. Als laatste is de functie f en g meestal niet gekend. Belangrijk binnen het ICP algoritme is dat er al een groffe registratie is gebeurd. Dit is het gevolg van twee doelen. Ten eerste omdat ICP vooral tracht een fijne registratie van dieptemappen te vinden, er bestaan namelijk andere technieken om high-level matching toe te passen. Ten tweede omdat het een goede initiële transformatie vereist om snel te kunnen convergeren en er zeker van te zijn dat een minimum wordt gevonden. Om tot een goede transformatie te komen, moet er een functie zijn die de transformatie kan evalueren voor de oppervlak registratie. Als er N aantal correspondentiepunten gekend zijn tussen twee views van hetzelfde oppervlak, is de controlefunctie niet meer dan het minimaliseren van de totale afstand:

$$e = \sum_{i=1}^{N} \|Tp_i - q_i\|^2 \tag{4.5}$$

op voorwaarde dat N groter is dan drie. Het probleem is dat er dikwijls niet zoveel correspondentiepunten worden gevonden, zeker niet als er enkel met puntenwolken wordt gewerkt i.p.v. oppervlakken. Beter is het minimaliseren van de afstanden van alle punten van een oppervlak tot één enkel oppervlak van het model. Dus het minimaliseren van:

$$e = \sum_{i=1}^{N} \|Tp_i - q_j\|^2 \text{, met } q_j = q | \min_{q \in Q} \|Tp_i - q\| \tag{4.6}$$

Maar ook dit is moeilijk te implementeren, omdat het vinden van q_j niet triviaal is. Er kan wel een iteratieve manier worden ontwikkeld als er een initiële transformatie T^0 wordt gegeven, die P al in dichte registratie bij Q legt. In elke iteratie k wordt de vorige transformatie T^{k-1} gebruikt om een nieuwe q_j^k te berekenen:

$$e = \sum_{i=1}^{N} \left\|T^k p_i - q_j^k\right\|^2 \text{, met } q_j^k = q | \min_{q \in Q} \left\|T^{k-1} p_i - q\right\| \tag{4.7}$$

Door deze iteratieve methode aan te houden, is het probleem eenvoudiger geworden. Een ander probleem bij het gebruik van correspondentiepunten is dat deze fouten kunnen bevatten. Deze fouten zullen voor een veel tragere convergentie van het algoritme zorgen of er zelfs voor zorgen dat hij helemaal geen oplossing teruggeeft. *Chen et al.* lossen dit op door in vergelijking 4.6 een benadering voor het oppervlak Q te nemen. Het raakvlak S_j op punt q_j voor Q wordt gekozen als benadering. De vergelijking wordt nu herschreven als volgt:

$$e = \sum_{i=1}^{N} \left\|Tp_i - q_j'\right\|^2 \text{, met } q_j' = q | \min_{q \in S_j} \|Tp_i - q\| \tag{4.8}$$

Ook hier moet q_j gekozen worden volgens een initiële transformatie T^0. Samengevat wordt de functie herschreven als een lineare functie:

$$e^k = \sum_{i=1}^{N} d_s^2(T \circ T^{k-1} p_i, S_i^k) \tag{4.9}$$

waarbij

$T \circ T^{k-1} = T^k$,
$S_j^k = \{s | n_{qj}^k \cdot (q_j^k - s) = 0\}$ is het raakvlak voor Q in punt q_j^k,
n_{qj}^k is de normaal van oppervlak Q in punt q_j^k,
$q_j^k = (T^{k-1} l_i) \cap S$ is het intersectiepunt van Q met de lijn $T^{k-1} l_i$,
$l_i = \{a | (p_i - a) \times n_{pi} = 0\}$ is de lijnnormaal voor P in punt p_i,
$p_i \in P$ is een punt op P,
d_s is de negatieve of positieve afstand van een punt tot een vlak

Bij het zoeken van correspondentiepunten voor P kunnen er random een aantal punten uit P worden gekozen. Indien alle punten worden beschouwd, gaat het algoritme veel meer berekeningstijd nodig hebben. Best kunnen punten gekozen worden die dicht bij elkaar liggen om zo de berekeningen voor Q makkelijker te maken. Samengevat ziet het algoritme van ICP, waarbij e^k wordt geminimaliseerd, er uit als volgt:

1. Selecteer een aantal correspondentiepunten $p_i \in P (i = 1 \cdots N)$ en bereken de normalen n_{pi} voor elk gekozen punt.

2. Voor elke iteratie k, herhaal het volgende totdat het algoritme convergeert:

   [a] Voor elk correspondentiepunt p_i,

   Transformeer het punt p_i en de normaal n_{pi} volgens T^{k-1} om zo p'_i en n'_{pi} te bekomen;

   Zoek de intersectie q_j^k van oppervlak Q met de normaallijn gedefinieerd door p'_i en n'_{pi};

   Bereken het raakvlak S_i^k in het punt q'_i aan Q

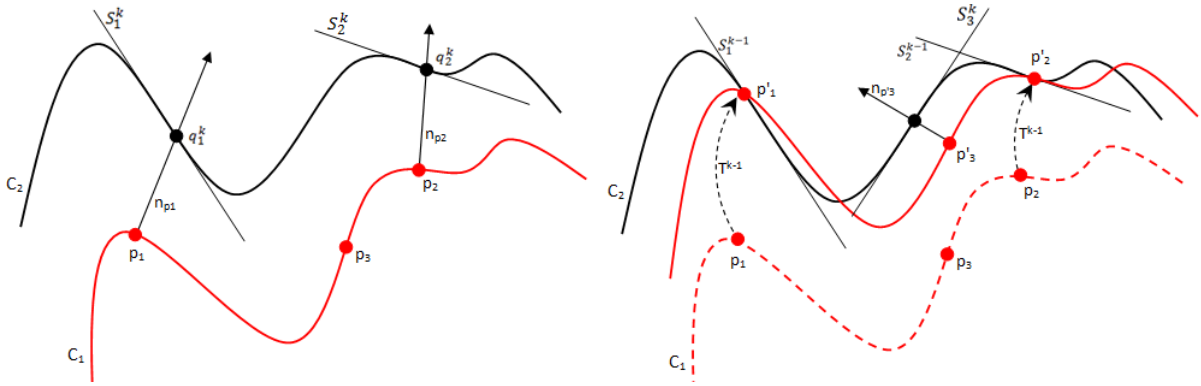
   [b] Zoek de transformatie T die e^k uit vergelijking 4.9 minimaliseert door gebruik te maken van de kleinste kwadratische afstand, hierbij is $T^k = T \circ T^{k-1}$.

De convergentie kan getest worden door opeenvolgende iteraties te vergelijken met de volgende vergelijking:

$$\delta = \frac{\left\| e^k - e^{k-1} \right\|}{N'} \leq \epsilon_e, (\epsilon_e > 0) \tag{4.10}$$

waar: ϵ_e een threshold is die manueel wordt ingesteld; N' is het aantal punten gekozen uit P zonder die van Q.
Een illustratie van het ICP algoritme volgens *Chen et al.* wordt gegeven in Figuur 4.1. In

Figuur 4.1a ziet u twee curves. De onderste (rode) curve C_1 moet geregistreerd worden in de bovenste (zwarte) curve C_2. In eerste instantie worden er een aantal controlepunten gekozen. Dit zijn de drie punten p_1, p_2 en p_2 op de onderste curve. In de eerste iteratie worden enkel de controlepunten p_1 en p_2 gekozen. De initiële transformatie T^{k-1} is al uitgevoerd voor de onderste curve en de normalen n_{p1} en n_{p2} zijn berekend in de punten p_1 en p_2 voor het oppervlak van C_1. Vervolgens wordt voor beide punten de intersectie berekend in C_2. Dit zijn respectievelijk de zwarte punten q_1^k en q_2^k. In beide punten wordt het raakvlak S_1^k en S_2^k aan de curve C_2 opgesteld. Nu is het algoritme klaar om de best mogelijke transformatie te zoeken die vergelijking 4.9 minimaliseert. M.a.w. de som van de kwadratische afstand van elk controle punt tot het raakvlak waar zijn normaal de andere curve snijdt moet zo klein mogelijk zijn. Voor de Figuur is dit de transformatie T^{k-1} die in Figuur 4.1b werd toegepast op C_1. In dit geval vallen de controlepunten p_1' en p_2' al vrij goed samen met C_2. Er wordt ook al een deel van de volgende iteratie geïllustreerd. In de tweede iteratie wordt enkel p_3' beschouwd als controlepunt om de Figuur overzichtelijk te houden. Ook hier wordt de normaal berekend en het snijpunt ervan met C_1, enz.



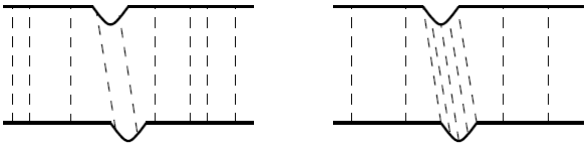
Figuur 4.1: Werking van het ICP algoritme. (a) Iteratie $k-1$. Er worden een aantal controlepunten p_1 en p_2 gekozen waarvoor de normaal op C_1 wordt bepaald. Op het snijpunt van de normaal met C_2 wordt het raakvlak berekend. Dit raakvlak wordt gebruikt in de evaluatiefunctie. De beste transformatie, die de afstand van controlepunten tot hun overeenkomstig raakvlak op het model minimaliseert, wordt gebruikt in een volgende iteratie. (b) Iteratie k. De gevonden transformatie uit de vorige iteratie wordt toegepast op C_1 en het algoritme wordt herhaald. Deze keer wordt er voor de duidelijkheid enkel p_3' gekozen als controlepunt.

4.1.2 De verschillende stappen van ICP

Rusinkiewicz en Levoy [68, 69] ontleden het ICP algoritme in zes stappen. Hierdoor kunnen varianten beschreven worden in functie van één van deze zes stappen. De meeste

technieken in deze sectie worden in de paper van *Rusinkiewicz en Levoy* [69] met elkaar vergeleken op verschil in convergentiesnelheid.

De eerste stap is *Selection*. Hierin worden punten gekozen uit één of beide datasets. *Besl en McKay* [9] maken gebruik van alle beschikbare punten, maar ICP kan sterk worden versneld door het gebruik van sub-sampling. Naast de snelheidswinst kan het kiezen van goede sub-samples de kwaliteit van de transformatie verbeteren. De makkelijkste manier van samplen is random sub-sampling of uniform sub-sampling. Uniform subsampling van de beschikbare punten wordt toegepast in de implementatie van *Turk en Levoy* [80]. Een verbetering van uniform subsampling is normal-space sampling. Het verschil wordt geïllustreerd in Figuur 4.2. Bij random of uniform sampling (random sampling ziet u in Figuur 4.2a) worden er minder samples in de groeves gekozen dan bij normal-space sampling (Figuur 4.2b). De gedachte hierachter is dat er drie (één rotatie en twee translaties) van de zes parameters van de transformatie worden bepaald door het vlakke stuk in de figuur en de andere drie (twee translaties en één rotatie) door de groeve. Het is dus belangrijk dat de groeve ook redelijk wat samples krijgt toegewezen. Er zijn ook algoritmes die gebruik maken van textuur of kleur van de scène om te helpen bij het aligneren [68]. Door het kiezen van de juiste samples—degene die de belangrijkste features beschrijven—zal het algoritme sneller convergeren en betere resultaten opleveren.



Figuur 4.2: Subsampling in de selection stap van ICP [68]. (a) random sampling (b) normal-space sampling.

De tweede stap van ICP is *Matching*. Deze stap bepaalt het grootste deel van de berekeningstijd van het algoritme en er zijn veel variaties voor te vinden. De overeenkomstige correspondentiepunten worden gezocht in de tweede dieptemap. Zoals eerder aangehaald, wordt er door *Chen et al.* [15] gebruik gemaakt van *normal shooting*, waarbij ze het snijpunt zoeken tussen een ray, die begint in het bronpunt en de normaal als richting heeft, en het oppervlak waarop hij invalt in de tweede set. *Besl en McKay* [9] zoeken daarentegen het dichtstbijzijnde punt van de andere mesh door gebruik te maken van een kd-boom. Andere technieken projecteren punten op de andere mesh op basis van de kijkhoek van de camera [10, 56] of variaties hiervan [6, 21]. Naast het zoeken van overeenkomstige punten op basis van een bepaalde metriek, kan er ook gebruik worden gemaakt van kleuren [66].

Nu moet elk paar van gecorrespondeerde punten een gewicht krijgen. Dit is de zogenaamde *Weighting* stap. Deze gewichten zullen ook in grote mate de kwaliteit van het resultaat bepalen. Daarom is het vaak trial-and-error totdat het gewenste effect wordt bereikt. De makkelijkste manier is aan elk paar punten een constant gewicht toe te kennen, maar dikwijls wordt er extra informatie gebruikt om de gewichten te berekenen. Het voordeel ervan is de punten met een grote kans om goede correspondentieparen te vormen ook meer invloed zullen hebben in het algoritme dan correspondentieparen die niet zo goed zijn gekozen, dewelke dus een klein gewicht krijgen en op de achtergrond worden geschoven. Dit impliceert ook direct een nadeel: Wat als de Weighting stap slecht is afgesteld? Dan zullen slechte correspondentieparen een groot gewicht toegekend krijgen en dat kan een zware invloed hebben op de snelheid van convergentie. Een eerste voorbeeld van een Weighting techniek is het gebruik maken van gewichten in functie van de punt-tot-punt afstand tussen de twee correspondentiepunten. In een tweede techniek maakt men gebruik van gewichten op basis van de overeenkomst tussen de normaal van beide oppervlakken. De eerste techniek wordt gebruikt door *Godin et al.* [29] en geeft het volgende gewicht:

$$\text{weight} = 1 - \frac{\text{distance}(p_1, p_2)}{\text{distance}_{max}}$$

Hierbij gaan paren die al dicht bij elkaar liggen een groot gewicht toegekend krijgen. De tweede techniek gebaseerd op normalen van oppervlakken geeft een gewicht van:

$$\text{weight} = n_1 \cdot n_2$$

Hierbij gaan correspondentiepunten waarvan de normalen sterk overeenkomen beschouwd worden als goede correspondentiepunten en dus ook een groot gewicht toegekend krijgen. Eventueel is het mogelijk beide technieken te combineren. *Godin* [29] heeft ook onderzoek gedaan naar het toekennen van gewichten op basis van kleuren.

De vierde stap is *Rejecting*. Deze stap komt sterk overeen met de Weighting stap. Alleen gaan nu correspondentieparen helemaal worden uitgesloten. Dit is noodzakelijk omdat deze in sommige gevallen verkeerd worden gekozen en nefaste gevolgen kunnen hebben op de convergentiesnelheid van het algoritme en het verkrijgen van de juiste transformatie. Het is daarom belangrijk om elk paar te controleren op correctheid en als er outliers worden gedetecteerd deze te verwijderen. Er zijn hiervoor enkele strategiën voorgesteld. Een eerste mogelijkheid is het verwijderen van correspondentiepunten die verder dan een bepaalde threshold van elkaar verwijderd liggen. Deze krijgen standaard al een groot gewicht toegekend. Het is ook mogelijk een procentueel aantal slechtste correspondenties te verwijderen. Zo verwijderen *Pulli et al.* [65] de 10% correspondentieparen waarvan de punten het verst van elkaar liggen. *Masuda et al.* [53] gaan dan weer de correspondentiepunten verwijderen waarvan de afstand groter is dan 2.5 maal de standaardafwijking van de afstanden. *Dorai* [21] gaat de consistentie bij continue plaatsen in het oppervlak beschouwen op basis van twee correspondentieparen (p_1, q_1) en (p_2, q_2). Bij continue oppervlakken zijn deze enkel consistent indien

$$|\text{distance}(p_1, p_2) - \text{distance}(q_1, q_2)|$$

kleiner is dan een bepaalde threshold. Deze threshold wordt bepaald door

$$0.1 \cdot max(\text{distance}(p_1, p_2), \text{distance}(q_1, q_2)).$$

Doordat het algoritme een complexiteit van $O(n^2)$ heeft, zal het de performantie sterk beïnvloeden. Daarom wordt er in vele gevallen maar een beperkt aantal correspondenties met elkaar vergeleken. Een goede keuze van na te kijken correspondentieparen geeft een grote snelheidswinst in het zoeken naar de juiste transformatie, ook al worden er extra berekeningen uitgevoerd. Naast de snelheidswinst geeft ICP een beter resultaat terug.

Het toekennen van een errormetriek gebeurt in de *Error Metric stap*. De bedoeling is de beste transformatiematrix te zoeken die een bepaalde error metriek minimaliseert. Voor deze metriek wordt vaak de som van de kwadratische afstanden tussen correspondentiepunten of tussen punten en oppervlakken gekozen. Bij *Chen et al.* [15] werd er gebruik gemaakt van de minimale kwadratische afstand van een punt, loodrecht op het vlak waarin het correspondentiepunt ligt. Andere algoritmes maken gebruik van point-to-point metrieken met verschillen tussen punten of verschillen tussen kleuren [44].

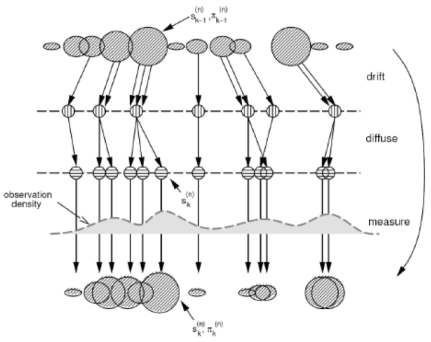
Voor het oplossen van point-to-point metrieken wordt over het algemeen gebruik gemaakt van numerieke methodes zoals singular value decomposition (SVD) [3], quaternions [36], orthonormale matrices [37], dual quaternions [82], $\cdots$. Bij het zoeken van de transformatie wordt er per iteratie gebruik gemaakt van de transformatie uit de voorgaande iteratie. Deze nieuwe transformatie zal de error metriek minimaliseren, zoals gezien in het algoritme van Chen in de vorige sectie. Dit is ook tevens de laatste stap van ICP: *Minimizing*.

4.1.3 Varianten van Iterative Closest Point

Een uitbreiding van het Iterative Closest Point algoritme wordt gegeven door *Chetverikov et al.* [17], de zogenaamde *Trimmed Iterative Closest Point* (TrICP). Het algoritme maakt gebruik van de Least Trimmed Squares (LTS) en kan overweg met een onderlinge rotatie van 30°. Er is al aangehaald dat *Besl en McKay* [9] gebruik maken van de kortste afstand van elk punt tot het model en dat op basis hiervan de transformatie met de de kleinste kwadratische afstand wordt gezocht. De Trimmed ICP werkt op dezelfde wijze, maar gebruikt de Least Trimmed Squares om overweg te kunnen met outliers, fouten in puntenwolken en gedeeltelijke overlapping. Least Trimmed Squares wordt ook gebruikt om de optimale transformatie te vinden en om de globale evaluatiefuntie, die moet worden geminimaliseerd, op te stellen. Het verschil tussen Least Trimmed Squares [19] en Least Squares is dat LTS een statistische methode is die enkel gebruik maakt van een subset van punten die invloed hebben op het registratieproces en de andere buiten beschouwing laat. Hierdoor zullen er geen foute transformaties worden gevonden ten gevolge van ruis of andere problemen. Het nadeel van deze techniek is dat, voor een beperkte dataset, het aligneren met TrICP twee seconden in beslag neemt. Daarnaast is het uitrekenen van de LTS een niet triviale stap. Toch is het volgens de auteurs een sneller en robuuster algoritme dan de standaard ICP.

4.2 Particle Filters

In de inleiding werd al kort het gebruik van Kalman filters [84] aangehaald die vooral bedoeld zijn voor feature tracking in passieve technieken. Er zijn ook nog andere methodes die geometrische objecten zoeken en/of volgen in een scène, zoals bijvoorbeeld de particle filters [49]. Een particle filter gaat algemeen een aantal particles genereren. Elke particle gaat een bepaalde toestand van een object beschrijven. Deze toestand kan bijvoorbeeld een transformatie zijn ten opzichte van de vorige plaats van het object. Stel dat men bijvoorbeeld een blad papier in een beeld heeft herkend en men wil dit papier opnieuw zoeken in het volgende beeld. Er zullen dan een aantal particles worden opgesteld die een transformatie van het blad papier voorstellen. Elke particle krijgt een bepaald gewicht toegekend op basis van de waarschijnlijkheid dat die toestand klopt. Elke particle wordt nagekeken op correctheid. Zolang de correctheid niet binnen een bepaalde verwachting valt, zullen de particles opnieuw worden gesampled en worden er nieuwe gewichten toegekend volgens de voorgaande evaluatie [54]. Het kiezen van de juiste gewichten valt onder de noemer van importance sampling [41]. Als een particle filter specifiek een object moet volgen, wordt meestal gebruik gemaakt van het aangepaste condensation particle filter algoritme, voorgesteld door *Isard en Blake* [40]. Hierin is er, net zoals in het voorbeeld met het papier, een initiële plaats voor het papier waarna volgens een bepaalde verdeling (meestal een Gaussiaanse verdeling) de verplaatsing wordt voorspeld. Omdat dit algoritme voornamelijk gebruikt wordt voor het tracken van features of geometrische objecten zal er in deze thesis niet verder op worden ingegaan. Toch kan een particle filter ook gebruikt worden voor het registreren van puntenwolken. Hoe dit in functie van puntenwolken werkt wordt nu kort toegelicht.



Figuur 4.3: Een stap in het condensation algoritme [40]. In eerste instantie (drift) worden nieuwe particles aangemaakt op basis van de gewichten van de vorige stap. Grotere gewichten zullen dus typisch resulteren in meerdere particles. In tweede instantie (diffuse) zal voor elk nieuw aangemaakte particle een voorspelling worden gedaan. Deze voorspelling is meestal zo gekozen, zodat de directe omgeving meer kans heeft om voorspeld te worden. In een laatste stap (measure) wordt voor elke voorspelde particle nagegaan hoe goed hij overeenkomt met de observatie. Deze mate van overeenkomst zal een indicatie zijn voor het aanmaken van de nieuwe gewichten per particle.

Een particle filter bestaat uit 3 hoofdstappen. Deze zijn gevisualiseerd in Figuur 4.3. De eerste stap is *Select* (in de Figuur noemt deze drift). Hierin worden er een aantal particles gegenereerd. Deze worden zo gegenereerd dat de particles met een groot gewicht van een vorige iteratie meer particles krijgen in deze iteratie. Het gewicht van de vorige iteratie wordt uniform verdeeld over de gewichten van de nieuwe particles. Het doel hiervan is dat particles met een groot gewicht (dus particles die dicht bij een uitkomst zitten), meerdere keren worden gekozen en dus meer worden uitgebuit. In de tweede stap *Predict* (in de Figuur diffuse), worden de nieuwe toestanden voorspeld. Voor elke transformatie zal er zo—volgens een bepaalde verdeling—een waarde worden opgeteld om tot een nieuwe toestand te komen. In deze stap wordt dus een transformatie van de puntenwolk berekend. Vervolgens wordt in een derde en laatste stap, *Measure*, de nieuwe toestand van de particles getoetst aan de werkelijkheid. De grootte van het gewicht wordt bepaald naargelang deze evaluatie. Indien een particle beter overeenkomt dan een bepaalde threshold zal het algoritme stoppen en wordt deze particle als juist resultaat beschouwd.

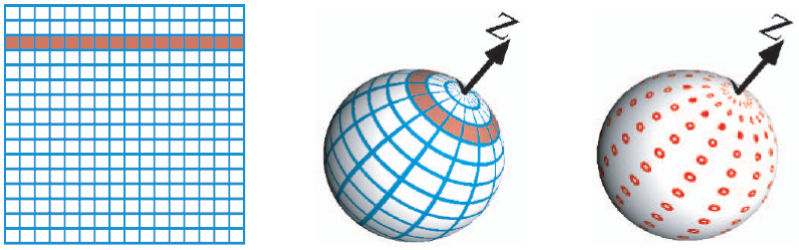
Het nadeel van een particle filter is dat deze doorheen de transformatieruimte zoekt naar de juiste transformatie voor registratie. Het zou goed kunnen dat er geen oplossing wordt gevonden of dat het veel te lang duurt voordat de juiste oplossing wordt gevonden. In veel gevallen is dit niet wenselijk. Indien de evaluatiefunctie voor het vergelijken van puntenwolken goed wordt gekozen, er genoeg overlapping is en er niet veel onderlinge transformatie is, zal het algoritme wel bruikbaar zijn. Onze doelstelling gaat ervan uit dat er veel opeenvolgende puntenwolken worden bekomen en deze reeds vrij goed met elkaar overeenkomen. Er zou kunnen geopteerd worden om eerst een gelimiteerde tijd te voorspellen met een particle filter en indien er binnen deze beperkte tijd geen degelijk resultaat wordt gevonden, overstappen op een meer robuuste algoritme zoals Iterative Closest Point. *Sandhu et al.* [72] creëerden een algoritme dat gebruik maakt van particle filters als optimalisatie van Iterative Closest Point. Ze gebruiken een particle filter om een schatting te maken van een mogelijke transformatie. Door in het ICP deze voorspellingen te gebruiken, gaat de rekentijd verminderen en ICP sneller convergeren naar een resultaat.

4.3 The Levenberg-Marquardt algorithm

Fitzgibbon [26] introduceerde een alternatief algoritme voor het registreren van puntenwolken. Volgens hem is het beter om het general-purpose niet-lineaire Levenberg-Marquardt algoritme te gebruiken. Deze minimaliseert direct de errormetriek die ook gebruikt wordt binnen ICP. Doordat de evaluatiefunctie direct minimaliseert is het ook inzetbaar voor robuuste registratie en gaat het de convergentie sterk versnellen. Volgens Fitzgibbon zou de implementatie van dit algoritme makkelijker en sneller zijn dan ICP. Daarenboven werkt het ook goed voor robuuste registratie, zonder het verlies van snelheid.

4.4 Extended Gaussian Images

De meeste registratietechnieken werken enkel voor fijne registratie. *Makadia et al.* [51] probeerde daarom een algoritme te ontwikkelen dat—volledig automatisch—twee weinig overlappende puntenwolken met elkaar uitlijnt. Voor een ruwe registratie maakt het algoritme gebruik van correlaties tussen twee Extended Gaussian Images (EGI's) in het Fourier domein. Het idee achter een Extended Gaussian Image is dat een scène wordt voorgesteld op een boloppervlak. Het oppervlak van de bol kan opgedeeld worden in bins. Deze bins worden voorgesteld in een grid, dit is het zogenaamde oriëntatiehistogram (zie Figuur 4.4). Elke bin bevat een klein stuk van het oppervlak en is verantwoordelijk voor alle normalen die dat klein oppervlak op de bol bevat. Het is de bedoeling dat alle aanwezige normalen in de scène toegevoegd worden aan de bin waar de normaal mee overeenkomt. Stel dat bin b_1 verantwoordelijk is voor de normaal n_1 en stel dat in de scène er nu ook eenzelfde normaal n_1 voorkomt, dan zal de bin b_1 met één worden verhoogd. Op deze manier bekomt men een ruwe voorstelling van de oriëntatie van een scène. Het voordeel van deze voorstelling is dat, onafhankelijk van de translatie, de rotatie direct kan worden bepaald. Hierdoor kan het algoritme opgedeeld worden in het uitlijnen van rotatie en het uitlijnen van translatie.



(a) (b) (c)

Figuur 4.4: Voorstelling van een oriëntatiehistorgram [51]. (a) Een oriëntatiehistorgram met 256 cellen; (b) De 256 bins uit (a) voorgesteld in bolcoördinaten; (c) De cellcentra weergegeven in bolcoördinaten.

Om de onderlinge rotatie te bekomen wordt er een correlatie toegepast op de twee EGI's. Een eerste triviale manier is de correlatie te berekenen tussen de twee oriëntatiehistogrammen. Dit is een complexe berekening omdat er veel onderlinge vergelijkingen moeten worden gemaakt. Beter is het gebruik maken van de *Spherical Fourier Transform*. Deze kan op een snelle en correcte manier correlaties vinden tussen rotaties van twee EGI's. Nu de rotatie is gevonden, is de volgende stap het zoeken van de translatie. Hiervoor wordt de paarsgewijze aanwezigheid van punten in 3D gemaximaliseerd. Dus de positie waar de grootste overlap is tussen alle punten van de twee dieptemappen. De dieptemappen worden voorgesteld als 3D coördinaten waar punten aanwezig zijn met de volgende

functie:

$$F(x) = \begin{cases} 1 \text{ als er een punt aanwezig is met coördinaat } x \in \mathbb{R}^3 \\ 0 \text{ anders} \end{cases}$$

Er wordt op basis van deze functie gezocht naar de translatie τ die de volgende correlatiefunctie maximaliseert:

$$G(\tau) = \int_{x \in \mathbb{R}^3} F_1(x) F_2(x - \tau) dx$$

Dit is een convolutie integraal is, waardoor hij in het Fourier domein overeenkomt met $\hat{G}(k) = \hat{F}_1(k)\hat{F}_2(k)$. Om te evalueren of de transformatie wel correct is, wordt er nog een verificatie stap uitgevoerd. Deze verificatie gaat een aantal globale parameters checken op consistentie, zoals het overeenkomen van normalen voor overlappende regio's. Daarnaast wordt er ook een zichtbaarheidstest uitgevoerd. Stel er is een puntenwolk ingescand vanuit een bepaald camerastandpunt. Na de uitlijning kan een andere puntenwolk deze puntenwolk zijn zicht niet belemmeren vanuit zijn standpunt, omdat er vanuit wordt gegaan dat dieptemappen alle zichtbare informatie bevat.

Het moeilijkste aan deze techniek is ten eerste het schatten van oppervlakken op basis van puntenwolken en daarna het opstellen van de oriëntatiehistogrammen. Als deze correct worden opgesteld, zal het algoritme zeer robuust werken en kan het dieptemappen aan die maar voor 45% overlappen. Doordat binnen onze doelstelling er gebruik wordt gemaakt van een bewegende camera, gaat er normaal geen ruwe registratie nodig. Daarnaast impliceert dit ook dat twee opeenvolgende beelden al veel overlapping hebben, waardoor de techniek minder bruikbaar is. Desalniettemin is het een goede techniek die in bepaalde contexten een aantal voordelen geniet.

4.5 Discussie

Interactive Closest Point is een zéér krachtig algoritme dat veel gebruikt wordt voor het registreren van puntenwolken. Het grootste nadeel is dat er al een goede initiële transformatie moet gekend zijn alvorens het algoritme snel en robuust werkt. Gelukkig wordt er binnen de probleemstelling gebruik gemaakt van een camera die vrij rondbeweegt. Voor elk binnenkomend beeld van de camera wordt continue een dieptemap berekend. Opeenvolgende dieptemappen zullen dus nooit veel van elkaar afwijken. ICP is dus een goede kandidaat om goed te werken. Daarnaast is het niet nuttig om de Extended Gaussian Image voorstelling te gebruiken. Deze zal veel extra overhead geven en is vooral bedoeld voor een registratie met weinig overlapping tussen twee dieptemappen.

De sterkte van het algoritme hangt ook sterk af van de keuzes voor de verschillende stappen binnen ICP. Er moet gekozen worden voor technieken die goed werken en weinig overhead geven. Als bijvoorbeeld de afstand van een punt tot een raakvlak wordt beschouwd, is het belangrijk dat dit raakvlak snel kan worden opgesteld. Ook het minimaliseren van

de error metriek moet zo performant mogelijk werken. Er zijn veel goede metrieken toegereikt die tot goede resultaten leiden.

Particle filters werken goed voor het volgen van objecten. Binnen bepaalde omstandigheden kunnen ze snel werken, mar voor grote scènes kunnen ze evengoed traag werken en eventueel helemaal geen oplossing bieden. Zoals gezegd is het eventueel mogelijk een hybride methode te ontwikkelen. Als er bijvoorbeeld met ICP na een bepaalde tijd nog altijd niet geconvergeerd is, wil dit misschien zeggen dat er maar weinig overlapping is. Er kan in dit geval bijvoorbeeld gebruik worden gemaakt van een particle filter of een ruwe registratie met Extended Gaussian Images om eerst de ruwe registratie uit te voeren en vanaf dan over te stappen op ICP voor de fijne registratie.

Hoofdstuk 5

Algemene algoritmes

Om via structured light dieptemappen te bekomen of puntenwolken te registreren zijn er nog een aantal technieken die nuttig of noodzakelijk zijn om dit te kunnen verwezenlijken. Zo moet in eerste instantie externe calibratie worden toegepast om de onderlinge positie van camera en projector te bepalen en daarnaast moeten de interne parameters van camera en projetor worden gecalibreerd voor tegemoet te komen aan vervormingen zoals bijvoorbeeld radiale distortie of brandpuntsafstand. Ook worden er twee algortimes—image rectification en RANSAC—besproken die de berekeningen sterk kunnen versnellen. Als laatste wordt dynamic programming, de techniek die veel terugkomt bij het decoderen van patronen, kort besproken.

5.1 Calibratie

De calibratietaak wordt opgedeeld in intrinsieke en extrinsieke calibratie. Een intrinsieke calibratiematrix beschrijft de inwendige kenmerken van een camera en welke vervorming er op het beeld aanwezig is. Een extrinsieke calibratiematrix beschrijft de translatie en rotatie van de camera ten opzichte van de wereldcoördinaten.

In de intrinsieke calibratiestap wordt er gezocht naar parameters (zoals brandpuntsafstand, optisch centrum, radiale distortie, enz.) van de camera zelf. Omdat deze stap maar één keer moet worden uitgevoerd, wordt hij ook wel statische calibatie genoemd [87]. Beide calibraties kunnen als voorbereidende stap gebeuren. De intrinsieke parameters van een camera worden beschreven in de volgende matrix:

$$A = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

waarbij f_u en f_v de brandpuntafstand van de camera voorstellen rond de u- en v-as, $(u_0, v_0)^T$ is het kardinaalpunt en s is een radiale vervormingsfactor. Deze parameters hangen af van het specifieke cameramodel. Eenzelfde intrinsieke matrix kan opgesteld worden voor de projector. Het voordeel van een projector is dat deze vervorming bijna

verwaarloosbaar is.

Extrinsieke calibratie is het zoeken van de positie van de camera ten opzichte van zijn omgeving. Deze bevat zes parameters: drie voor de rotatie van de camera en drie voor de translatie van de camera. Samengevat probeert de extrinsieke calibratie een relatie te zoeken tussen 2D afbeeldingen en de 3D wereld.

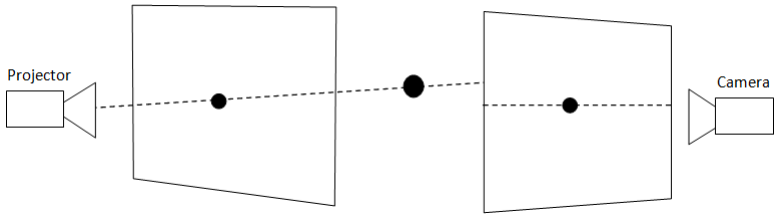
De totale calibratie stap—die gebruik maakt van een pinhole camera—ziet er nu als volgt uit [24]:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = A \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} G \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.2}$$

waarbij $[su, sv, s]$ de afbeeldingscoördinaten zijn, s een schaleringsfactor verschillend van 0, $[X, Y, Z, 1]$ wereldcoördinaten, A de 3×3 een transformatiematrix die de intrinsieke parameters van de camera beschouwt (zoals vergelijking 5.1) en G de verplaatsingsmatrix die de extrinsieke parameters zoals camera positie en oriëntatie beschouwt. De matrix M is dus de perspectieve transformatie die 3D wereldcoördinaten relateert met 2D afbeelingdscoördinaten.

De calibratiematrices kunnen op verschillende manieren worden verkregen. Een eerste manier is een bekend patroon in verschillende houdingen voor de camera te houden en de vervormingen hierop te berekenen. Zo gebruikt de GML Toolbox [46] een schaakbordpatroon en kan het de intrinsieke en extrinsieke calibratiematrix voor de camera als resultaat teruggeven. Er kan ook gebruik worden gemaakt van de *Camera Calibration Toolbox* [79]. Deze laat de gebruiker met een laser in de camera's schijnen. Op basis van de laserposities kunnen de goede calibratiematrices berekend worden. *Zhang et al.* [87] geeft een automatische manier die zich baseert op patronen voor het verkrijgen van de intrinsieke en extrinsieke parameters.

Bij structured light calibratie zal het coördinatensysteem van de camera moeten gerelateerd worden aan dat van de projector [5] en niet aan wereldcoördinaten. De bedoeling is dat als de projector een lichtstraal uitzendt, er exact geweten is met welke lijn in het camerabeeld deze lichtstraal overeenkomt (zie Figuur 5.1). Het voordeel van het gebruik van een projector is dat deze weinig of geen intrinsieke calibratie vereist, omdat er verwaarloosbare vervorming op aanwezig is. Ook de parameters voor deze calibratie kunnen verkregen worden met de *Camera Calibration Toolbox* [79]. De laserstralen moeten nu gesimuleerd worden met de projector



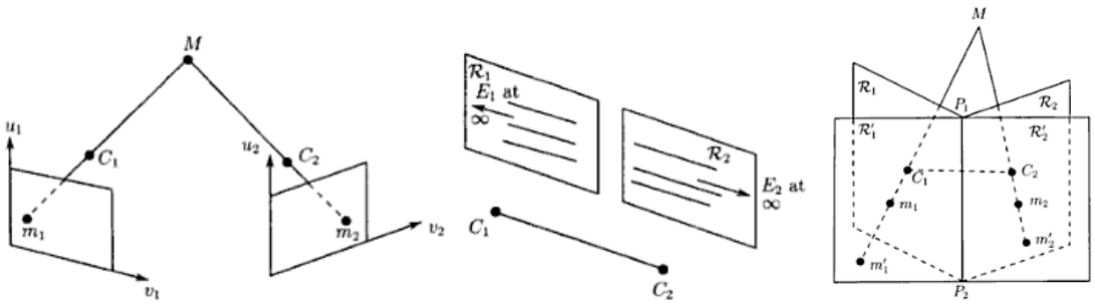
Figuur 5.1: Calibratie tussen projector en camera. Het doel van calibratie is de relatie tussen een lichtstraal—uitgezonden door de projector—en de lijn in het camerabeeld dat hiermee overeenkomt te bekomen.

5.2 Illumination calibratie

Bij structured light technieken wordt de scène belicht met patronen. Hierbij weet men hoe sterk de bestraling van de projector is [70]. Voor elke geprojecteerde pixel moet de hoeveel irradiantie gekend zijn. Voordat een projector wordt gebruikt, moet de calibratie stap de lichtintensiteit berekenen van de geprojecteerde pixels. Dit kan gedaan worden door de geprojecteerde pixels te calibreren op een sterkt diffuus oppervlak. Dit is ook zeer belangrijk voor het gebruik van kleuren in patronen. Op deze manier is er direct een verband tussen een gebruikte kleur en de hoe de kleur zich gedraagt in de scène.

5.3 Image rectification

Bij image rectification [23, 78, 16] tracht men de de epipolen te plaatsen op oneindig zodat de epipolaire lijnen evenwijdig zijn met de horizontale rijen van de afbeelding (zie Figuur 5.2). Herinner dat de epipolaire lijn de snijding is van de lijn, gevormd tussen de twee projectiecentra, en beide afbeeldingen. De epipolen zijn de punten waar de lijn, gevormd tussen de twee projectiecentra, beide afbeeldingen snijdt. Door het kiezen van de epipolen op oneindig, hoeft bij het scannen enkel langs één horizontale lijn naar correspondenties worden gezocht (zie Hoofdstuk 2) en is het zoeken m.a.w. beperkt tot één dimensie. Dit zal het proces van stereomatching aanzienlijk versnellen. Image rectification gaat beide afbeeldingen projecteren op een nieuw vlak dat evenwijdig loopt met de lijn tussen de twee projectiecentra. Het kiezen van dit vlak moet in functie van een minimale distortie op het beeld. Daarnaast worden intensiteiten van tussenliggende punten, waarvoor geen mapping werd gevonden, geïnterpoleerd met hun buren. Het is mogelijk een camera en projector zo te plaatsen, zodat de epipolen altijd op oneindig staan. Het nadeel is dat het niet triviaal is om de projector exact af te stellen op de camera.

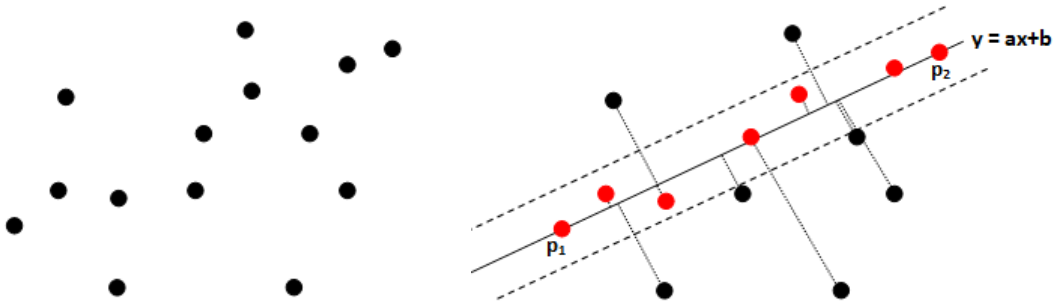


(a) (b) (c)

Figuur 5.2: Image rectification [23]. (a) Een scène wordt waargenomen vanuit twee standpunten. (b) Dit is de situatie dat men graag wilt hebben, namelijk de epipolen op oneindig zodat de epipolaire lijnen evenwijdig zijn met de horizontale assen. (c) Image rectification gaat de twee afbeeldingen projecteren op een nieuw vlak dat evenwijdig is met de epipolaire lijn.

5.4 RANSAC

RANSAC of RANdom SAmple Consensus is een heuristische methode, voorgesteld door *Fischer en Bolles* [25]. Het algoritme schat iteratief de parameters van een wiskundig model uit een aantal beschikbare punten. Uit deze beschikbare punten neemt RANSAC een aantal punten waardoor hij het wiskundig model kan fitten. Alle andere punten die binnen een bepaalde threshold voldoen aan deze punten classificeert hij als inliers en de ander als outliers. Een voorbeeld wordt geveven in Figuur 5.3. Standaard worden alle punten outliers genoemd. Deze outliers zijn weergegeven in Figuur 5.3a. RANSAC gaat vervolgens zo veel mogelijk samples van punten nemen waardoor hij het model fit. Voor een lijn in 2D zijn dit twee aan twee punten. In Figuur 5.3b wordt er bijvoorbeeld punt p_1 en p_2 gekozen. Er wordt een lijn opgesteld door deze twee punten. Voor alle andere punten wordt er nagekeken of ze binnen een bepaalde afstand van de lijn liggen. Zo ja, wordt het punt beschouwd als een inlier. De twee punten waarvoor de meeste inliers zijn gevonden, is het best overeenkomende resulaat. Hoe meer paren van punten—waardoor het wiskundig model kan worden gefit— er worden gecontroleerd, hoe meer kans dat de beste oplossing wordt gevonden. Het is een zeer robuuste techniek en komt terug in veel algoritmes om outliers te verwijderen. Ook bij het fitten van puntenwolken in andere puntenwolken kan het gebruikt worden voor outliers te detecteren. Er zijn bijvoorbeeld drie punten nodig om een puntenwolk in 3D te fitten in een andere puntenwolk. Als RANSAC lang genoeg wordt uitgevoerd, zal hij het beste resultaat geven waarvoor de puntenwolk fit in de andere. In sommige registratie algoritmes wordt RANSAC gebruikt voor de ruwe registratie [67]. Er worden een aantal basisvormen van de scène gezocht en deze worden met RANSAC in elkaar gefit. Een fijne registratie kan vervolgens uitgevoerd worden met een andere techniek zoals ICP. Structured light technieken voegen standaard redelijk wat ruis toe aan een dieptemap, dus het is aan te raden om RANSAC te gebruiken



(a) (b)

Figuur 5.3: RANSAC voor het zoeken van lijnen. (a) De beschikbare punten (outliers). (b) RANSAC neemt een paar punten waar hij een lijn door fit. In dit geval punt p_1 en p_2. Alle punten die binnen een threshold liggen (aangeduid met een stippellijn) van de lijn, worden beschouwd als inliers (rode punten).

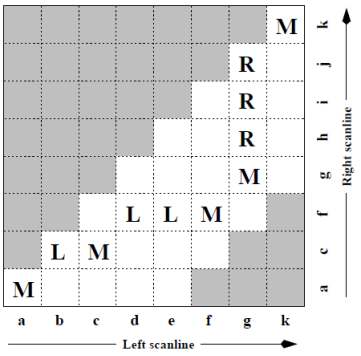
zodat de grootste outliers er worden uitgehaald en zo ICP sneller convergeert.

5.5 Dynamic Programming

Dynamic programming is een algoritme dat een optimaal pad zoekt binnen een bepaald probleem en zo ook een optimale oplossing kan teruggeven [62]. Het wordt veel gebruikt voor het zoeken naar een optimaal pad binnen een netwerk [62]. Het idee van dynamic programming is het achterwaards werken, te beginnen met de laatste beslissing en zo terug te gaan naar de eerste beslissing. Het is m.a.w. een backtracking algoritme [62]. Stel dat er een sequentie van N beslissingen $D_1, D_2, \cdots, D_N$ moeten worden genomen, dan is de sequentie optimaal als de laatste k beslissingen, $D_{N-k+1}, D_{N-k+2}, \cdots, D_N$ optimaal zijn. Meestal wordt bij dynamic programming het probleem opgedeeld in kleinere stadia waarin beslissingen plaatsnemen en zoekt het de optimale weg om terug te gaan naar een vorige stap in de sequentie [62].

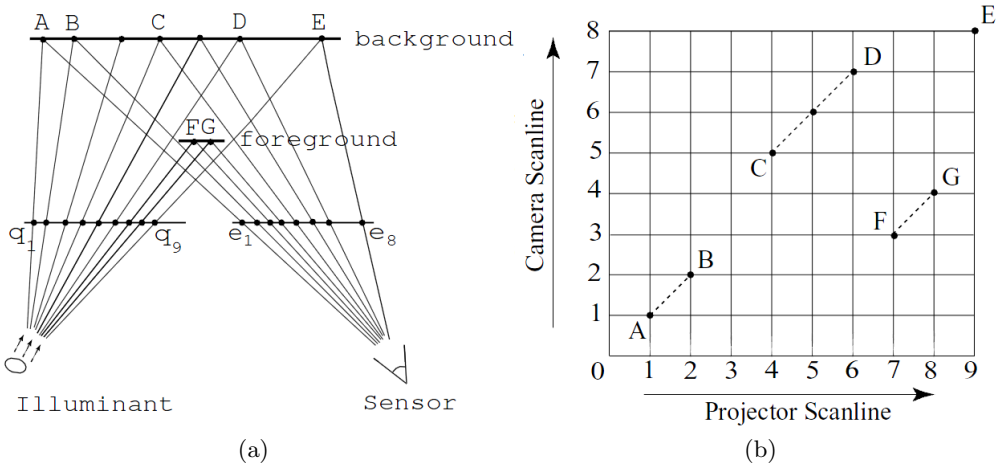
In het geval van stereo vision technieken wordt dynamic programming gebruikt voor het vinden van overeenkomsten tussen twee beelden [18, 74]. Meer precies kan dynamic programming worden gebruikt voor het matchen van twee scanlines. Het hoort thuis in de categorie van global optimization waar men tracht een bepaalde kost voor het matchen te minimaliseren. Stel bijvoorbeeld dat er een patroon wordt geprojecteerd op een scène en het geprojecteerd patroon moet worden gematched met een waargenomen scène. Deze kunnen vergeleken worden door het minimaliseren van een kostfunctie tussen twee scanlines. Deze scanlines lopen volgens epipolaire lijnen tussen camera en projector. Om deze scanlines horizontaal in de afbeelding te nemen, is het voldoende beide epipolen op oneindig te leggen, zoals gezien in sectie 5.3 bij Image Rectification. Samengevat komt het er dus op neer om paargewijs de horizontale scanlines te matchen door met behulp

van dynamic programming de minimale kost te zoeken. Elke operatie in het pad zal een bepaald gewicht hebben op de kostfunctie. Het is belangrijk de juiste gewichten te bepalen voor matchen die gevonden zijn in beide scanlines en deels zichtbare matches in de linker of rechter scanline.



Figuur 5.4: Stereo matching van twee scanlines met dynamic programming [74]. Het patroon is opgebouwd uit een sequentie van a tot k. Deze sequentie wordt waargenomen in twee scanlines, vanuit twee verschillende camerastandpunten. De hoofdletters in de figuur geven het gekozen pad doorheen de matrix aan: Matches in beide scanlines worden aangegeven met M; Deels zichtbare punten zijn aangeduid met L of R die aangegeven of ze enkel zichtbaar zijn in respectievelijk het linker of rechter scanline. Elke operatie heeft een bepaalde kost en dynamic programming heeft als doel dit pad met laagste kost te vinden. Horizontale lijnen in het pad geven aan dat het linkerbeeld is occluded, bij verticale lijnen het rechterbeeld.

Een voorbeeld van een gematched pad tussen twee scanlines ziet u in Figuur 5.4. Een M komt overeen met een gevonden pad waarvoor de twee scanlines matchen. Een horizontale lijn geeft aan dat de er occlusions zijn in de linker scanline, een verticale geeft aan dat er occlusions zijn in de rechter scanline. Het vinden van dit optimale pad gebeurt met dynamic programming, waar hij als backtracking algoritme gaat starten bij de oplossing en vervolgens alle mogelijke voorgaande stappen afgaat tot hij bij het begin aankomt (in de Figuur rechtsboven starten en linksonder eindigen). Het nadeel van dynamic programming is dat hij niet overweg kan met monotoniciteits- en orderingsbeperkingen die worden opgelegd. De relatieve ordering van pixels tussen twee scanlines moet worden verzekerd [74]. Dit probleem komt voor als er objecten dichtbij de camera liggen en wordt verduidelijkt in Figuur 5.5. U ziet dat de volgorde van de waargenomen randen door de camera niet overeenkomt met de volgorde van de geprojecteerde randen. Dit is dus een schending van de monotoniciteit. Er bestaan hier oplossingen voor zoals het multi-pass dynamic programming algoritme van *Zhang et al.* [89]. Bij dit algoritme zou ABCDE in de eerste pass worden gevonden en FG in de tweede pass. Naast monotoniciteit moet er ook voldaan zijn aan uniekheid. Dit wil zeggen dat het pad niet terug mag gaan naar voorgaande delen van het patroon om dubbele matches te vermijden.



(a) (b)

Figuur 5.5: Monotoniciteits- en orderingsprobleem bij dynamic programming [89] (a) Er worden negen kleurovergangen geprojecteerd op een scène waarbij er een klein voorwerp op de voorgrond staat ten opzichte van een vlakke achtergrond en daarbij worden er acht randen gedetecteerd door de camera. De derde geprojecteerde rand is niet zichtbaar voor de camera, omdat het zich achter het object bevindt. (b) In de matching matrix kunt u zien dat de volgorde van de gevonden cameraranden niet overeenkomt met de volgorde van de geprojecteerde randen.

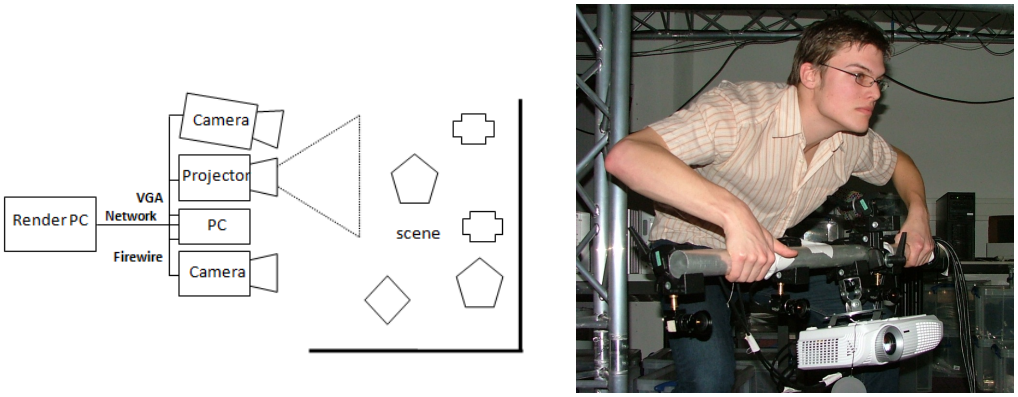
Hoofdstuk 6

Implementatie

Er onderscheiden zich duidelijk een aantal belangrijke stappen die nodig zijn om van twee inputbeelden een volwaardige 3D scène te bekomen. In dit hoofdstuk worden de implementatie details van deze verschillende stappen besproken.

6.1 Overzicht

De opstelling die wordt gebruikt voor de implementatie is uitgetekend in Figuur 6.1. In Figuur 6.1a staat een abstracte voorstelling van hoe het systeem er zou kunnen uitzien. Figuur 6.1b toont de uiteindelijke uitwerking ervan. Er wordt gebruik gemaakt van twee *Point Grey Flea* [63] camera's. De projector is ingesteld op een resulutie van 800x600, alsook de twee camera's. De algoritme's worden verwerkt op een *Intel Core 2 Duo CPU* (aan 2.10 GHz) met 4.00 GB RAM. Het systeem gebruikt een *ATI Mobility Radeon HD 2600* grafische kaart. De meeste matrices berekeningen gebeuren m.b.v. *OpenCV* 2.1 [58].

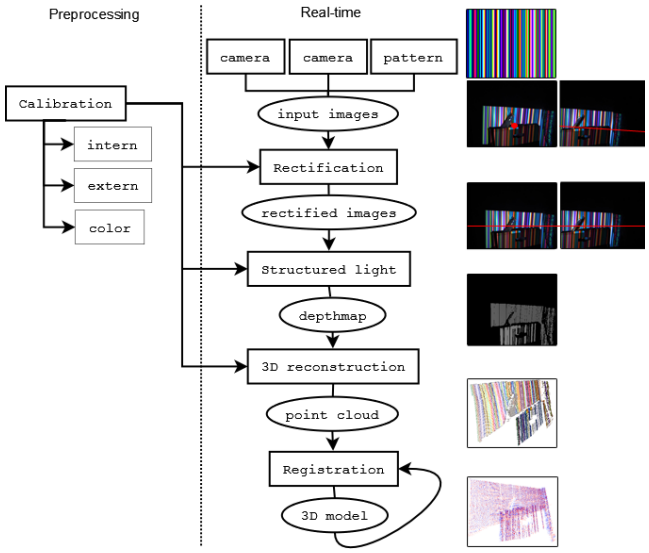


(a) Abstracte voorstelling 3D opname systeem (b) Uitwerking van het 3D opname systeem

Figuur 6.1: Implementatie van het 3D opname systeem.

De aandachtige lezer ziet dat er in de huidige opstelling twee camera's worden gebruikt, waar in de voorgaande hoofdstukken er telkens maar één camera noodzakelijk was. In theorie is het mogelijk om vanuit één camera en één projector de diepte te bepalen op voorwaarde dat beide correct zijn gecalibreerd. Toch geeft dit een aantal moeilijkheden. Ten eerste is er een kleurenvervorming aanwezig bij het projecteren van een kleur. Scènes die zelf kleuren bevatten zullen een geprojecteerde kleur sterk vervormen, waardoor het moeilijk is om punten in het waargenomen beeld te matchen met punten in het geprojecteerde beeld. De ideale scène zou een witte scène zijn, waardoor kleuren minimaal vervormd worden. Daarnaast moet er een kleurencalibratie worden toegepast (ook voor een witte scène). Deze kleurencalibratie maakt een mapping tussen de geprojecteerde kleur en de hoe de kleur zich gedraagt in een scène. Deze calibratie is voor elke scène uniek, waardoor het moeilijker wordt niet-statische scènes te scannen.
Beide problemen worden opgelost door een extra camera te gebruiken. Er worden nu geen correspondenties meer gezocht tussen de projector en de camera, maar wel tussen de twee camera's onderling. Doordat zij dezelfde kleurvervorming waarnemen zullen ze hiervan geen problemen ondervinden. Let op, in Figuur 6.1b zijn er zelfs drie camera's aanwezig. De meest linkse camera heeft in het algemene pipeline geen enkel nut, maar is gebruikt als hulpmiddel bij calibratie.

De literatuurstudie besprak al de belangrijkste stappen en algoritmes die nodig zijn binnen de pipeline. Deze zijn respectievelijk: calibratie van de camera's, rectification van de inputbeelden, het correspondentie probleem oplossen door extra textuur toe te voegen onder de vorm van patronen, reconstructie van de 3D scène en registratie van puntenwolken. Figuur 6.2 illustreert de pipeline. Deze verschillende stappen zullen in de secties van dit hoofdstuk verder genuanceerd worden met als laatste een discussie van de resultaten.



Figuur 6.2: Pipeline van het systeem.

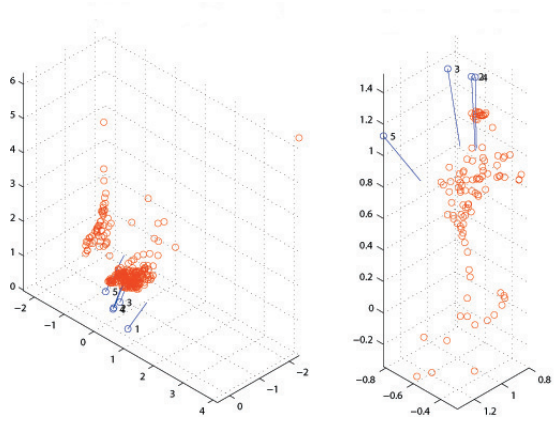
6.2 Calibratie

Alvorens er juiste interpretaties van de camerabeelden kunnen plaatsvinden, moeten deze eerst worden gecalibreerd. Concreet wil dit zeggen dat de interne en externe geometrie ervan gekend moeten zijn.
In deze thesis wordt hiervoor de calibratietoolbox *Multi-Camera Self-Calibration* van *Svoboda et al.* [79] gebruikt. Deze toolbox bevat allerhande functionaliteit om drie of meer camera's te calibreren. De gebruiker hoeft enkel met een laser rond te lopen doorheen de scène terwijl alle camera's, gesynchroniseerd, beelden opnemen. Voor het synchroniseren van de camera's wordt de *Point Grey Toolbox* van de camera's [63] zelf gebruikt. Op basis van deze beelden kan de Matlab toolbox vervolgens de geometrie van de camera's schatten.
Hoe kan deze techniek nu worden toegepast binnen de opstelling van deze thesis? In eerste instantie was het plan enkel gebruik te maken van één camera en één projector. Omdat de tweede camera nu wordt vervangen door een projector en ook hiervoor een interne geometrie moet gekend zijn, kan er geen laser meer worden gebruikt. De laser kan nu wel gesimuleerd worden met de projector. De projector simuleert random een aantal laserpunten door ze te projecteren op de scène. Ook zijn eigen projectiebeeld wordt als input van de toolbox gebruikt. Daarnaast moet er enkel nog in de andere camera's naar het laserpunt worden gezocht. Doordat de toolbox op zijn minst drie camera's nodig heeft en doordat de kwaliteit van de calibratie stijgt naarmate het aantal camera's toeneemt, zijn er een aantal dummy camera's geïntroduceerd die helpen bij de calibratie. Deze dummy camera's worden later in de pipeline niet meer gebruikt. In Figuur 6.3a ziet u een reconstructie van de gecalibreerde camera's. Deze reconstructie komt echter niet overeen met de echte posities van camera's in de scène. De laser vervangen door een projectie lijkt niet goed te werken. In theorie werkt deze aanpak wel. De projectie die hier werd gebruikt zal niet goed afgesteld geweest zijn, alsook kan er teveel ruis aanwezig geweest wegens lichtvervuiling. Als daarentegen een laser wordt gebruikt en de projector niet mee wordt gecalibreerd, is de reconstructie zoals in Figuur 6.3b. Deze reconstructie is juister. Waarschijnlijk komt dit omdat de toolbox geoptimaliseerd is voor de herkenning van laserpunten en onze projectie deze laserpunten niet goed heeft gesimuleerd. Daarnaast is er voor de zekerheid gebruik gemaakt van een veel groter aantal inputbeelden.

Als ook deze laatste aanpak niet de gewenste resultaten teruggeeft, is het ook mogelijk om binaire patronen te projecteren zoals besproken in sectie 2.4 over time-multiplexing patronen. Zo kan er op pixelniveau van de projector naar correspondenties worden gezocht. Deze zullen er voor zorgen dat elke pixel in het beeld uniek gemapped kan worden, waardoor er veel meer correspondentiepunten worden gevonden, waardoor ook de calibration toolbox nauwkeuriger zijn resultaat kan berekenen.

6.2.1 Output van calibratie

De *Multi-Camera Self-Calibration toolbox* van *Svoboda et al.* [79] geeft voor elke camera een projectiematrix terug. Een projectiematrix projecteert een 3D homogeen punt op



(a) Calibratie met slechte reconstructie (b) Calibratie met goede reconstructie

Figuur 6.3: 1 is de projector en 2 tot en met 5 de camera's

een 2D image coördinaat [32]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \tag{6.1}$$

De toolbox biedt ook de mogelijkheid om deze projectiematrix te ontbinden in een intrinsieke 3×3 calibratie marix K, een rotatiematrix R, een translatiematrix T en een positie van het camera centrum C in het wereldcoördinatensysteem. Daarnaast geeft het ook de parameters van de radiale vervorming terug.

6.3 Rectification

Herinner uit de literatuurstudie dat de correspondentie van een punt kan worden gevonden op de epipolaire lijn in het andere beeld. Deze epipolaire lijn wordt berekend m.b.v. de fundamentele matrix. Deze fundamentele matrix kan worden benaderd door de correspondentiepunten, berekend bij calibratie, te gebruiken in een *8-point algorithm* [32] of m.b.v. *RANSAC*. Hiervoor kan onderstaande OpenCV functie [57] worden gebruikt.

```
int cvFindFundamentalMat(const CvMat* points1, const CvMat* points2,
                         CvMat* fundamentalMatrix, int method=CV_FM_RANSAC,
                         double param1=1.0, double param2=0.99, CvMat* status=NULL)
```

In Figuur 6.4 gaat er bij wijze van voorbeeld een random punt worden gekozen in het ene beeld (Figuur 6.4a en wordt de overeenkomstige epipolaire lijn uitgetekend in het

(a) Een gekozen punt in camera 1 (b) Epipolaire lijn van gekozen punt in camera 2

Figuur 6.4: Gebruik van de fundamentele matrix. Bij het kiezen van een punt in het linkse beeld (groen punt), zal het overeenkomstige punt gezocht moeten worden in de epipolaire lijn van het rechtse beeld (groene lijn).

andere beeld (Figuur 6.4b). Het is duidelijk dat het overeenkomstige punt zich, mits een beetje afwijking, bevindt op de epipolaire lijn.

Het nadeel van deze aanpak is dat er extra processing moet gebeuren. Voor elk punt moet er nu namelijk de overeenkomstige epopilaire lijn worden berekend via de fundamentele matrix. Daarnaast varieert de zoekruimte van de epipolaire lijn in 2D. Beter is zoals aangehaald in sectie 5.3 de beelden te rectificeren. Dit zal er voor zorgen dat de epipolaire lijnen parallel met de beeldcoördinaten komen te liggen . Het voordeel is dat de fundamentele matrix niet meer moet worden berekend. Daarnaast is de zoekruimte nu nog maar ééndimensionaal.
OpenCV [57] heeft een functie dat op basis van de camera geometrie een beeld kan rectificeren.

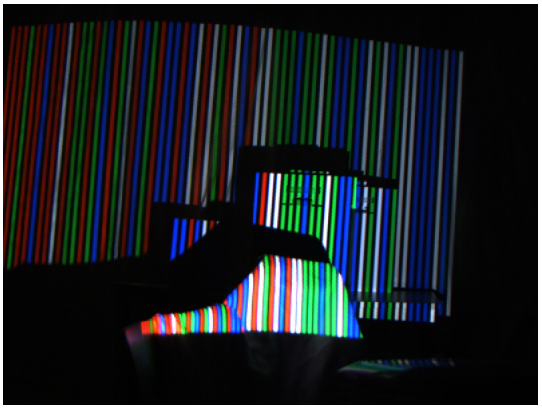
```
void cvInitUndistortRectifyMap(const CvMat* cameraMatrix, const CvMat* distCoeffs,
                               const CvMat* R, const CvMat* newCameraMatrix,
                               CvArr* map1, CvArr* map2)
```

Deze functie berekent, op basis van de geometrie, twee mapping functies. Deze mapping functies kunnen worden gebruikt om het beeld te warpen naar een gerectificeerd beeld met behulp van de volgende funcie.

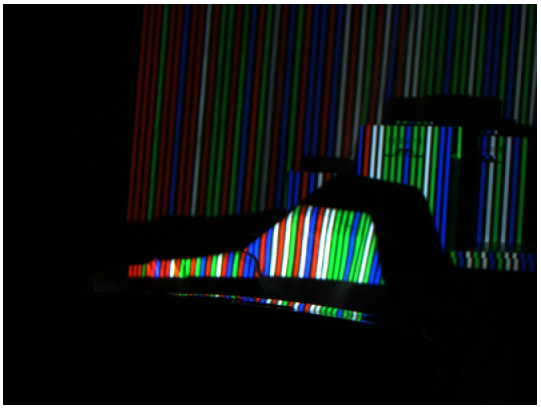
```
void cvRemap(const CvArr* src, CvArr* dst, const CvArr* mapx, const CvArr* mapy,
             int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,
             CvScalar fillval=cvScalarAll(0))
```

Het gebruik van deze rectification methode geeft echter in praktijk niet de gewenste resultaten. Waarschijnlijk is dit het gevolg van de verkeerde omzetting van de camera geometry, bekomen uit de matlab toolbox, naar de voorstelling die in OpenCV wordt

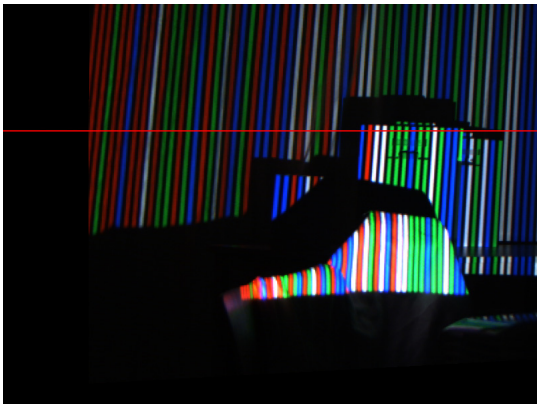
gebruikt. Daarom is in deze thesis overgestapt naar de *Matlab Rectification Kit* van *Andrea Fusiello et al.* [28]. De toolbox berekent op basis van de projectiematrices een projectieve transformatie voor de inputbeelden, dewelke gebruikt wordt om het beeld te warpen. Het resultaat hiervan ziet u in Figuur 6.5. Op deze rechtgetrokken beelden kan er nu makkelijk naar correspondenties worden gezocht omdat alle epipolaire lijnen horizontaal liggen.



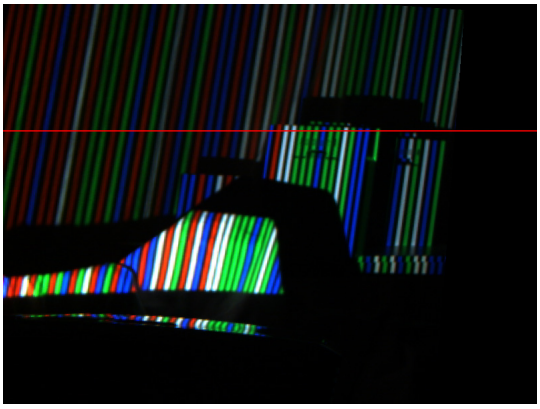
(a) Input rechts



(b) Input links



(c) Gerectificeerd rechts



(d) Gerectificeerd links

Figuur 6.5: Rectificeren van camerabeelden. Bovenaan ziet u de beelden die waargenomen worden door de camera (a) en (b); Onderaan de rechtgetrokken camerabeelden (c) en (d) volgens de projectieve transformatie bekomen uit de *Matlab Rectification Kit*. De rode lijn stelt de epipolaire lijn voor. Punten op de linkse rode lijn zullen overeenkomsten hebben met punten in de rechtse rode lijn en omgekeerd.

6.4 Textuur toevoegen onder de vorm van patronen

Alle prepocessing stappen zijn nu uitgevoerd. Per frame is het de bedoeling om overeenkomstige punten te vinden in de twee camerabeelden. De projectie van een patroon maakt dit zo optimaal mogelijk omdat er zo meer textuur aan de scène wordt toevoegd. Het voordeel hiervan is dat correspondenties makkelijker terug te vinden zijn. In deze sectie zal worden uitgelegd welke patronen er hiervoor zijn aangemaakt. De volgende sectie zal vervolgens uitleggen hoe er vanuit de scanlines correspondenties worden bekomen.

In de literatuurstudie zijn er een aantal mogelijke patronen aangehaald. Er werd uitgelegd dat De Bruijn patronen over het algemeen beter zijn omdat ze in ideale omstandigheden geen dubbele overeenkomsten hebben. Voor deze thesis zijn er een aantal patronen ontworpen om na te gaan welke bruikbaar zijn en welke niet.

6.4.1 Multi-Slit patronen

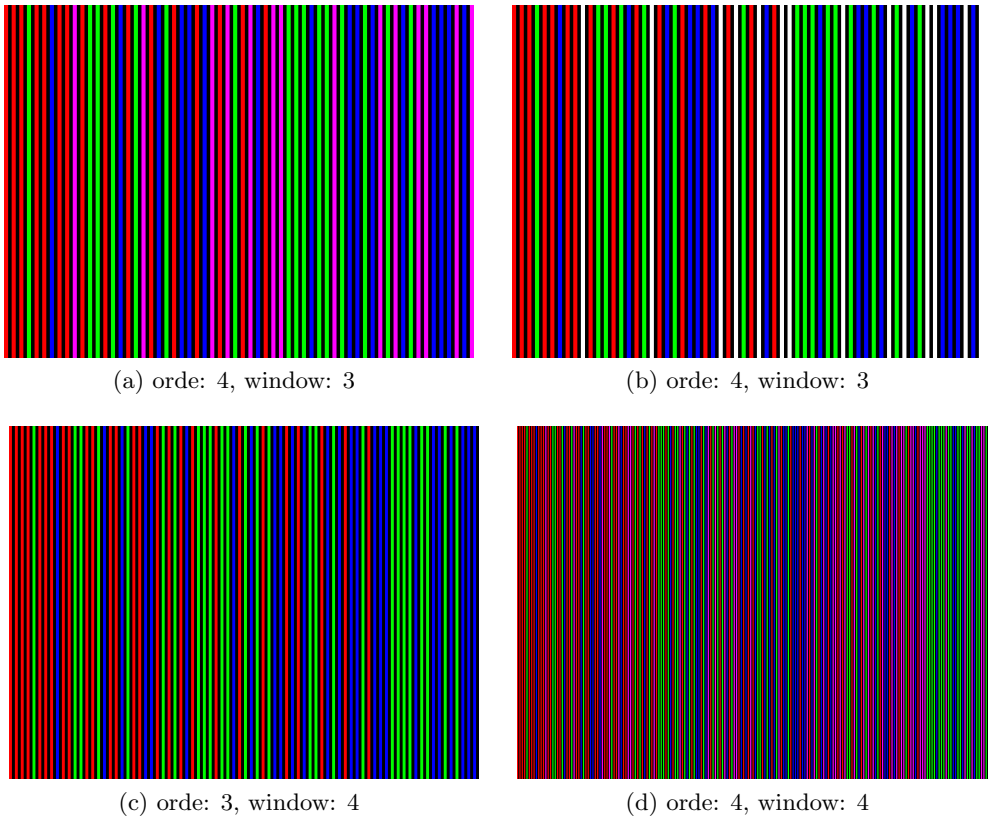
Een eerste groep patronen zijn gemaakt op basis van gekleurde verticale banden, gescheiden met zwarte banden. De zogenaamde multi-slit patronen (zie sectie 2.3). De gekleurde banden zijn opgesteld volgens een De Bruijn sequentie. De sequenties zijn bekomen met behulp van een online applet van *Hakan* [45]. In de applet kunnen verschillende waardes voor k en n worden ingesteld. Een sequentie met een alfabet (n) van drie en windowgrootte (k) van drie ziet er als volgt uit:

`0 0 0 1 0 0 2 0 1 1 0 1 2 0 2 1 0 2 2 1 1 1 2 1 2 2 2`

. Figuur 6.6 toont een aantal multi-slit patronen die opgebouwd zijn op basis van De Bruijn sequenties met verschillende parameters. Figuur 6.6a en 6.6b gebruiken beide dezelfde *De Bruijn sequentie*, maar met andere kleuren. Het is ook belangrijk te kijken welke kleuren het beste waarneembaar zijn en het minst vervormen in de scène.

6.4.2 Stripe patronen

Een tweede groep zijn de stripe patronen (zie sectie 2.3). Dit zijn patronen met verticaal gekleurde banden, maar nu niet meer gescheiden met zwarte banden. Het moeilijke aan het opstellen van dit soort patronen is dat het niet rechtstreeks kan gebeuren op basis van de opgestelde De Bruijn sequentie. De gekleurde banden worden nu namelijk tegen elkaar geplaatst, waardoor twee buren niet dezelfde kleur mogen hebben. In deze thesis worden de stripe patronen opgebouwd volgens het algoritme van *Zhang et al.* [89]. Het opzet is niet zozeer rechtstreeks de De Bruijn sequentie te gebruiken voor de kleuren, maar wel als overgangen van de kleuren. In de paper wordt aangeraden om een De Bruijn sequentie van orde vijf en window grootte drie te gebruiken. De overgang 110 en 111 worden weggelaten omdat een gelijke overgang van rood en groen het meeste last heeft van errors. Er zijn nu in totaal vijf overgangen gedefinieerd. Bij het begin van het patroon wordt een initiële kleur gekozen. In ons geval is dit 001. De volgende kleur wordt vervolgens bekomen door een *XOR* operatie toe te passen op de voorgaande kleur

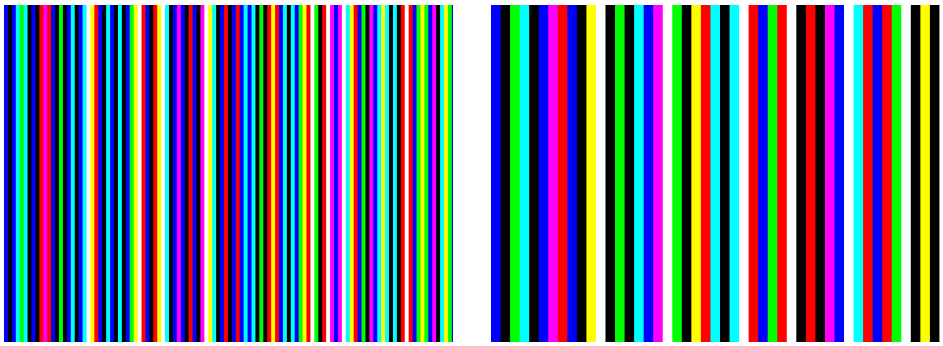


(a) orde: 4, window: 3 (b) orde: 4, window: 3

(c) orde: 3, window: 4 (d) orde: 4, window: 4

Figuur 6.6: De Bruijn stripe patronen met verschillende orde en window grootte.

en de overgang (aangegeven door de De Bruijn sequentie). De opgestelde patronen kan u zien in Figuur 6.7.

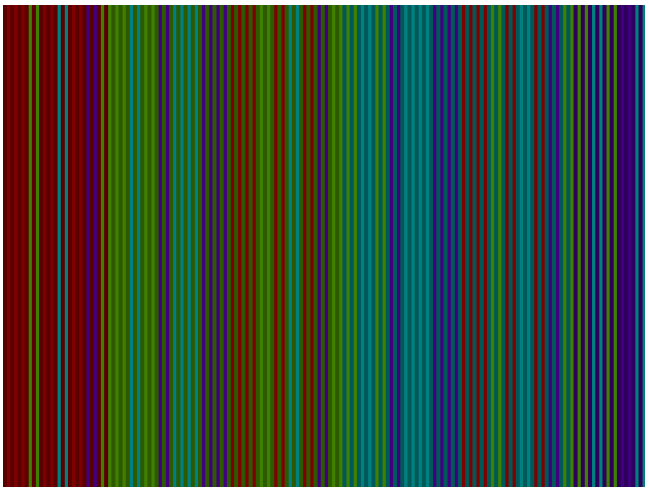


(a) orde: 5, window: 3 (b) orde: 7, window: 2

Figuur 6.7: De Bruijn stripe patronen volgens *Zhang et al.* [89] met verschillende orde en window grootte.

6.4.3 Hybride patronen

Als laatste is er nog het hybride patroon van *Salvi et al.* [60] (zie sectie 2.5.2). Het maakt gebruik van hoge intensiteit en lage intensiteit in combinatie met een stripe patroon van kleuren. De matlab implementatie van dit hybride patroon vindt u terug op de volgende url[1]. Het resultaat ziet u in Figuur 6.8.



Figuur 6.8: Hybride patroon van *Salvi et al.* [60].

6.4.4 Discussie

Er zijn verschillende soorten patronen die gebruikt kunnen worden. Welke is nu het beste? Veel hangt af van de implementatie van het correspondentie algoritme.

Ten eerste is er de zichtbaarheid. De strepen mogen niet te smal zijn, zodat ze nog goed zichtbaar zijn in het camerabeeld. Zo kan er niet veel smaller worden gegaan als Figuur 6.7a voor objecten die verder liggen. Om betere resultaten te bekomen bij het zoeken naar correspondenties wordt het zwart in de inputbeelden weggefilterd. Dit heeft als nadeel dat de multi-slit patronen minder bruikbaar worden, omdat er daar veel zwarte banden aanwezig zijn.

Daarnaast moet er ook de vraag worden gesteld of het nuttig is om de Bruijn sequenties te gebruiken. Ook dit hangt weer sterk af van hoe correspondenties worden gezocht. Als er puur op kleur en opeenvolging van kleur wordt gezocht is het aan te raden een De Bruijn sequentie te gebruiken omdat ze een unieke code geeft aan elke positie in het beeld. Maar als er een global optimization techniek wordt toegepast is een De Bruijn sequentie niet rechtstreeks een meerwaarde voor het algoritme. Let op, het is wel nuttig om extra textuur te hebben voor een global optimization techniek. Eventueel kan in de

[1] http://www.feed-back.be/nick/masterthesis/files/CreatePattern.m

De Bruijn sequentie toch nog worden uitgebuit om extra logische testen toe te voegen en zo het een robuuster algoritme te maken.

Het is duidelijk dat de keuze van het patroon en of dit noodzakelijk een De Bruijn patroon moet zijn, sterk afhangt van het algoritme om correspondenties te zoeken. De techniek die in deze thesis is toegepast, maakt gebruik van een global optimization techniek, namelijk dynamic programming. Deze zal nu worden besproken.

6.5 Correspondenties zoeken

Nu er extra textuur aan de scène is toegevoegd, is het tijd om correspondenties te zoeken in de twee beelden. De scanlines zijn makkelijk te vinden omdat het beeld is gerectificeerd. Dit zijn twee overeenkomstige horizontale lijnen. Een eerste naïve aanpak is het handmatig zoeken van opeenvolgende kleuren. Een tweede, betere aanpak, is een global optimization techniek, zoals dynamic programming. Herinner dat, initieel, het ene beeld het geprojecteerde patroon is en het andere het opgenomen beeld van de camera. Er is al aangehaald dat er wordt afgestapt van deze aanpak, maar wordt overgaan naar het gebruik van twee camera's.

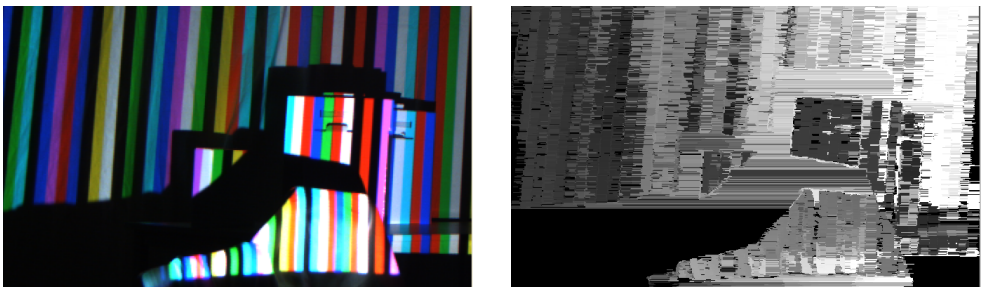
6.5.1 Handmatig de kleuren zoeken

Het is mogelijk per scanline het aantal opeenvolgende kleuren te zoeken ter grootte van de windowgrootte. Indien deze codes zijn gevonden, moet er enkel nog naar de overeenkomstige code in de andere scanline worden gezocht. Er is echter een sterke vervorming van de kleuren aanwezig op de scène, waardoor het bijna onmogelijk is om de correspondenties juist te vinden. Een oplossing is om eerst een kleurencalibratie toe te passen, dit zowel tussen projector en camera alsook voor de scène zelf. De projector vervormt alle kleuren waardoor de camera niet exact dezelfde kleuren gaat waarnemen. Maar ook de scène zelf zal instaan voor een sterke vervorming. Het zorgt ervoor dat de kleuren donkerder of lichter worden of zelfs helemaal vervormen bij een gekleurde scène. Dit zou goed werken in een statische scène, maar omdat uiteindelijk wordt verwacht dat de camera vrij rondbeweegt, is er geen vaste scène gegeven en is m.a.w. een kleurencalibratie niet nuttig. Dit is ook de hoofdreden waarom er een extra camera wordt bijgehaald. Op deze manier worden er geen correspondenties meer gezocht tussen projector en één camera, maar wel tussen twee camera's onderling. Doordat de camera's ongeveer hetzelfde zijn afgesteld, zullen beide camera's dezelfde vervorming waarnemen en is het ook mogelijk dezelfde vervormde kleuren met elkaar te matchen

In Figuur 6.9 ziet u in een voorbeeld hoe de correspondenties worden gezocht. In dit geval worden er drie opeenvolgende kleuren gezocht in de bovenste scanline, meer bepaald *rood-blauw-rood*. Vervolgens wordt diezelfde code in de tweede scanline gezocht. Het verschil in horizontale afstand tussen de twee wordt de dispariteit genoemd. Deze dispariteit is omgekeerd evenredig met de afstand van het punt tot de camera.

Figuur 6.9: Het zoeken van de dispariteit op basis van kleuren.

Als zo elke scanline van het beeld wordt afgegaan en zoveel mogelijk correspondenties worden gezocht, bekomt met een dispariteitsmap. Deze ziet er uit als Figuur 6.10 voor de bovenstaande methode. Zoals u kunt zien zijn dit niet de gewenste resultaten en zit er nog veel ruis in. Daarnaast houdt het ook geen rekening met occlusies of schaduwen. Daarom kan er beter gebruik worden gemaakt van een global optimization techniek, zoals bijvoorbeeld dynamic programming



(a) Camerabeeld (b) Dispariteitsmap

Figuur 6.10: Kleur gebaseerde dispariteitsmap

6.5.2 Dynamic programming

Dynamic programming valt onder de categorie van global optimization. Concreet wil dit zeggen dat de techniek tracht om globaal twee scanlines met een zo laag mogelijke kost met elkaar te matchen. De één-aan-één kosten om punten tussen de twee scanlines te matchen wordt voorgesteld in een scoring matrix. Op basis van deze scoring kan er een cost matrix worden opgesteld. Dit is tevens de belangrijkste stap van het algoritme. Doorheen de cost matrix kan het optimale pad in de buurt van de diagonaal worden gezocht. Hoe dichter bij de diagonaal hoe kleiner de dispariteit en omgekeerd. In Figuur 6.11 wordt dit optimale pad voorgesteld in de scoring matrix met een rode lijn. De diagonaal wordt aangegeven met een groene lijn. In de scoring matrix is het duidelijk welke punten goed overeenkomen (zwart) en waar de objecten zich bevinden, namelijk de zwarte strepen.

Scoring Matrix

De scoring matrix bevat de mate van overeenkomst tussen twee pixels in twee scanlines. De verticale coördinaten staan voor de pixels van de scanline van het linker beeld, de horizontale voor de pixels van de scanline van het rechter beeld. Elke cel van de matrix bevat een score die beschrijft in welke mate twee pixels tussen de twee beelden overeenkomen. Deze wordt bepaald met behulp van een *score functie*. De invulling van deze functie is de wortel van de som van de kwadratische afstand van de RGB waardes, ofnog

$$score(q,e) = \sqrt{(q.red - e.red)^2 + (q.green - e.green)^2 + (q.blue - e.blue)^2}. \quad (6.2)$$

Cost Matrix

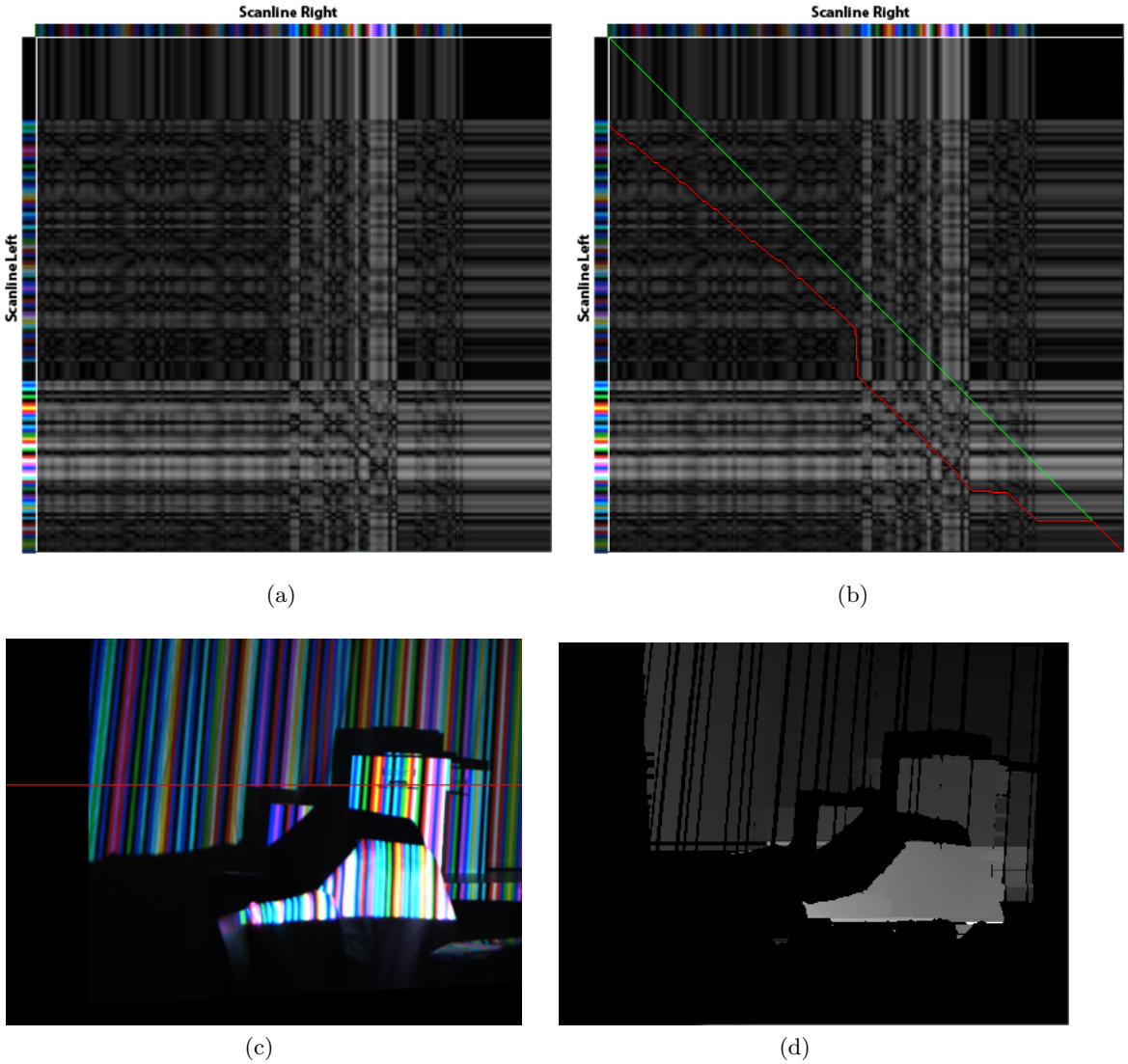
Er wordt gebruik gemaakt van de notatie van *Cox et al.* [18]. Stel $\phi^*_{(j,i)}$ is het optimale pad naar positie (j,i). De kost hiervan wordt bepaald door $\sigma(\phi^*_{(j,i)})$. Deze cost matrix kan recursief worden berekend als volgt:

$$\sigma(\phi^*_{(j,i)}) = \left\{ \begin{array}{cc} 0, & \text{if } j=0 \text{ or } i=0; \\ max \left\{ \begin{array}{c} \sigma(\phi^*_{(j-1,i-1)}) + score(q_j, e_i) \\ \sigma(\phi^*_{(j-1,i)}) \\ \sigma(\phi^*_{(j,i-1)}) \end{array} \right\}, & \text{otherwise} \end{array} \right\}. \quad (6.3)$$

In pseudocode ziet dit er uit als volgt:

```
for (i=1;i<M;i++) costMatrix(i,0) = Occlusion;
for (i=1;i<N;i++) costMatrix(0,i) = Occlusion;
for(j=1;j<M;j++){
  for(i=1;i<N;i++){
    min1 = costMatrix(j-1,i-1)+score(q_j,e_i);
    min2 = costMatrix(j-1,i)+Occlusion;
    min3 = costMatrix(j,i-1)+Occlusion;
    costMatrix(j,i) = cmin = min(min1,min2,min3);
if(min1==cmin) M(j,i) = DP_MATCH;
if(min2==cmin) M(j,i) = DP_VERTICAL;
if(min3==cmin) M(j,i) = DP_HORIZONTAL;
  }
}
```

waarbij *Occlusion* een bijkomende kost is indien er een verticale of horizontale occlusie aanwezig is. Deze extra kost zal trachten occlusies te vermijden. In deze thesis wordt de enkelvoudige kost van de huidige cel gebruikt als occlusion kost. Daarnaast wordt er een extra matrix M bijgehouden die aangeeft of er voor die positie een match, een horizontale occlusie of een verticale occlusie is.



(a) (b)

(c) (d)

Figuur 6.11: Scoring matrix voor een voor een scanline. (a) de scoring matrix opgebouw met vergelijking 6.2; (b) afstand van het optimale pad (rood) tot de diagonaal (groen) geeft de dispariteit; (c) de rode lijn geeft de scanline aan waarvoor de voorgaande scoring matrix werd opgesteld; (d) dieptemap: het uiteindelijke resultaat van dynamic programming voor alle scanlines.

Zoeken van optimale pad

In de cost matrix kan nu worden gezocht naar het optimale pad, gegeven door $\sigma(\phi^*_{(N,M)})$, dat van rechtsonder naar linksboven loopt. Dit pad is optimaal als de kost zo laag mogelijk is. Hiervoor wordt er gebruik gemaakt van de matrix M die samen met de cost

matrix werd opgesteld. Vertrekkende van rechtsonder geeft hij aan in welke richting het pad moet gevolgd worden. De pseudocode voor dit algoritme ziet er uit als volgt:

```
p=M;
q=N;
while(p!=0 && q!=0){
  switch(M(p,q)){
    case DP_MATCH:
      p--;q--;
      break;
    case DP_VERTICAL:
      p--;
      break;
    case DP_HORIZONTAL:
      q--;
      break;
  }
}
```

In Figuur 6.11b wordt dit optimale pad aangegeven met een rode lijn. U kunt zien dat dit overeenkomt met de plaatsen in de scoring matrix waar er een lage cost is voor het matchen van de twee scanlines (zie Figuur 6.11a). Figuur 6.11c geeft aan welke scanline van het linkerbeeld in de scène werd gebruikt voor het opstellen van de scoring en cost matrix. De doos op de voorgrond en het doek op de achtergrond zijn duidelijk aanwezig in de scoring matrix. De doos ligt iets op de voorgrond, waardoor het dus verder van de diagonaal (aangegeven met een groene lijn in Figuur 6.11b) ligt. De afstand van het rode optimale pad tot de groene diagonaal geeft de dispariteit aan van het punt. Deze dispariteit kan later gebruikt worden om de diepte te bekomen (diepte is omgekeerd evenredig met de dispariteit). Zo ziet u dat het doek op de achtergrond een beetje schuin hangt en rechts verder van de camera is verwijderd dan links.

Performantie

Het opstellen van de cost matrix heeft een complexiteit van $O(MN)$ voor elke scanline. Daarnaast heeft het verkrijgen van het optimale pad een complexiteit van $O(M+N)$ [89]. Doordat deze techniek scanline per scanline werkt, leent het zich uitstekend uit aan een paralleliseerbare oplossing, bijvoorbeeld op de GPU. Het is ook mogelijk een aantal scanlines uit te voeren in aparte threads om het te versnellen. Voor elke scanline een aparte thread is daarentegen minder performant omdat er ook extra overhead moet worden bijgerekend voor het opstarten van de thread.
Er zijn enkele optimalisaties mogelijk. Een eerste is het beperken van de maximale depth range [89]. Deze kan bijvoorbeeld 10% zijn van de maximale depth range. Hierdoor moet maar een beperkt deel van de cost matrix worden opgesteld.
Een tweede optimalisatie is downsampling. Het algoritme geeft ook goede resultaten als

de resolutie van de beelden wordt gehalveerd. Dit omdat de stripes van de patronen breed genoeg zijn en het voldoende is om sparse dieptemappen te bekomen. Dit zorgt ervoor dat de complexiteit van de costmatrix van $O(\frac{MN}{4})$ en de complexiteit van het zoeken van het pad van $O(M+N)$ naar $O(\frac{M+N}{2})$.
In hoofdstuk 7 wordt er een overzicht gegeven van de impact van een aantal factoren op de snelheid van het algoritme.

Discussie

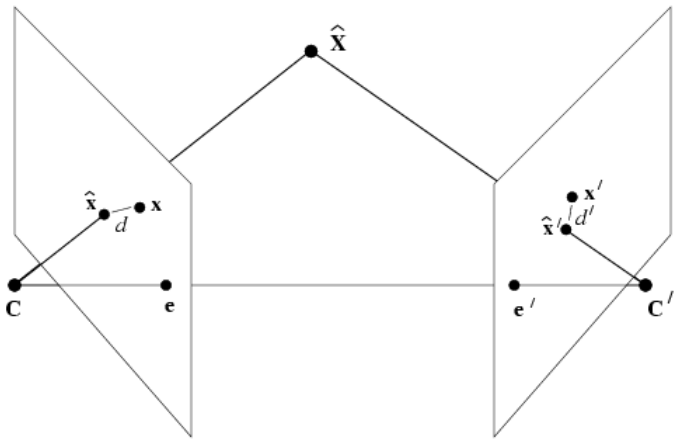
Dynamic programming kan niet goed overweg met discontinuïteiten in de scène. Ook in bovenstaande implementatie is er geen oplossing voor gegeven. Op plaatsen waar een abrupte overgang is van voorgrond naar achtergrond geeft het soms slechte resultaten waardoor de rand soms gaat samensmelten met de achtergrond. Het is mogelijk om het dynamic programming algoritme uit te breiden naar een multi-pass algoritme, zoals voorgesteld door *Zhang et al.* [89]. Deze gaat eerst optimale subpaden zoeken en deze vervolgens correct aan elkaar plakken.

Om het algoritme zo robuust mogelijk te maken is het aan te raden een soort van background substraction toe te passen door het zwart weg te filteren en deze punten niet te gebruiken. Hierdoor gaan enkel maar de pixels waarop een patroon geprojecteerd is worden gematcht. Het is dan ook aan te raden patronen te ontwikkelen zonder zwarte banden. Dit is in deze thesis niet het geval, waardoor er in de dieptemap voor bepaalde banden geen match is gevonden omdat deze punten weggefiterd zijn.

6.6 3D Reconstructie

In de 3D reconstructie stap wordt er getracht om vanuit de dispariteiten uit de vorige stap, 3D posities in wereldcoördinaten te berekenen. Dit is de zogenaamde triangularisatie. Vanuit de twee projectiematrices en de twee corresponderende 2D punten wordt een 3D punt geschat (zie Figuur 6.12). Doordat de backprojections van de twee punten niet ideaal snijden, wordt er gebruik gemaakt van SVD. SVD is een algoritme dat gebruikt kan worden om de meest ideale oplossing te verkrijgen.

Stel twee correspondentiepunten $(x_1, y_1, 1)$ en $(x_2, y_2, 1)$ die hetzelfde punt (X, Y, Z, W) in 3D beschrijven en hebben respectievelijk een projectiematrix P^1 en P^2. Backprojectie



Figuur 6.12: Triangularisatie is het zoeken van een 3D punt in wereldcoördinaten op basis van twee corresponderende 2D image punten (x en y in de Figuur). De backprojectie gaat nooit ideaal een snijpunt teruggeven omdat x en y niet accuraat genoeg kunnen worden berekend. Met behulp van SVD wordt de meest ideale 3D coördinaat berekend. Dit is $\hat{X}$. De triangularisatie error is gedefinieerd als het verschil in afstand van de projectie van $\hat{X}$ op de afbeeldingen. In de Figuur is deze aangegeven met d en d'.

gebeurt met de volgende vergelijkingen:

$$\begin{aligned}
P^1_{00}X + P^1_{01}Y + P^1_{02}Z + P^1_{03}W &= x_1; && (6.4)\\
P^1_{10}X + P^1_{11}Y + P^1_{12}Z + P^1_{13}W &= y_1; && (6.5)\\
P^1_{20}X + P^1_{21}Y + P^1_{22}Z + P^1_{23}W &= 1; && (6.6)\\
P^2_{00}X + P^2_{01}Y + P^2_{02}Z + P^2_{03}W &= x_2; && (6.7)\\
P^2_{10}X + P^2_{11}Y + P^2_{12}Z + P^2_{13}W &= y_2; && (6.8)\\
P^2_{20}X + P^2_{21}Y + P^2_{22}Z + P^2_{23}W &= 1. && (6.9)\\
& && (6.10)
\end{aligned}$$

Deze worden herschreven in de volgende vorm:

$$A \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0. \qquad (6.11)$$

Deze uitdrukking wordt herschreven in de volgende vier vergelijkingen:

$$\begin{aligned}
(6.4) &= [(6.4) - x_1(6.6)] && (6.12)\\
(6.5) &= [(6.5) - y_1(6.6)] && (6.13)\\
(6.7) &= [(6.7) - x_2(6.9)] && (6.14)\\
(6.8) &= [(6.8) - y_2(6.9)]\,, && (6.15)\\
& && (6.16)
\end{aligned}$$

ofnog

$$\begin{bmatrix} x_1P^1_{20} - P^1_{00} & x_1P^1_{21} - P^1_{01} & x_1P^1_{22} - P^1_{02} & x_1P^1_{23} - P^1_{03} \\ y_1P^1_{20} - P^1_{10} & y_1P^1_{21} - P^1_{11} & y_1P^1_{22} - P^1_{12} & y_1P^1_{23} - P^1_{13} \\ x_2P^1_{20} - P^2_{00} & x_2P^2_{21} - P^2_{01} & x_2P^2_{22} - P^2_{02} & x_2P^2_{23} - P^2_{03} \\ y_2P^1_{20} - P^2_{10} & y_2P^2_{21} - P^2_{11} & y_2P^2_{22} - P^2_{12} & y_2P^2_{23} - P^2_{13} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0. \qquad (6.17)$$

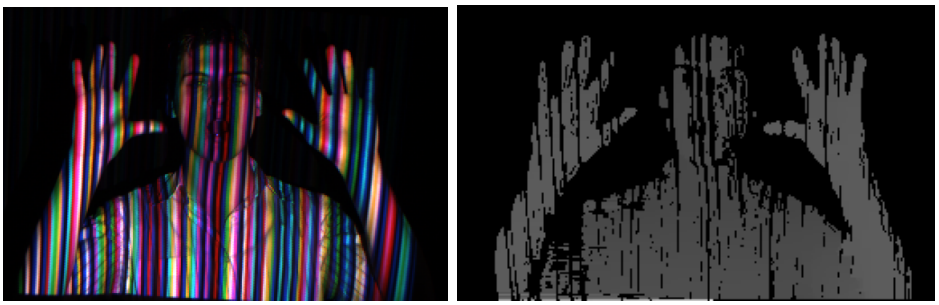
De matrix A gaat met behulp van Singular Value Decomposition (SVD) worden opgedeeld in UDV^t, waarbij U en V orthogonaal zijn; de kolommen van U de eigenvectoren zijn van AA^t; de kolommen van V de eigenvectoren zijn van A^tA en D een diagonaalmatrix met de eigenwaardes van A. De eigenvector met de kleinste eigenwaarde voor A^tA is de beste uitkomst voor de opgestelde vergelijkingen. Deze is de de laatste kolom van V. In het geval van triangularisatie, zal de eigenvector met kleinste eigenwaarde niet het exacte snijpunt zijn van de twee backprojecties, maar wel het midden zijn van het lijnstuk loodrecht op de twee backprojecties.

Voorbeeld

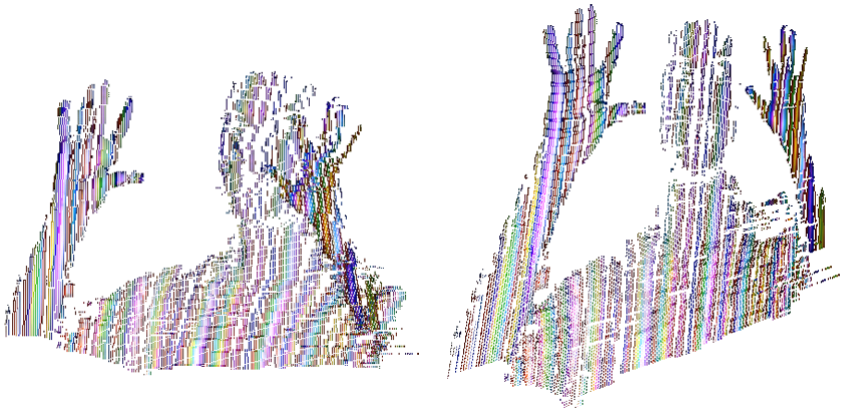
Gegeven een dieptemap zoals Figuur 6.13c. Elk punt in de dieptemap staat voor een dispariteit en is een verschil in coördinaat van twee punten op de twee beelden. Bij een gerectificeerd beeld gaat de x-coördinaat van beide punten overeenkomen en wordt de dispariteit weerspiegeld op de verandering in y-coördinaat. Deze twee punten, tesamen met de projectiematrices van beide camera's vormen de input voor het opstellen van de matrix A. Elk bekomen 3D punt wordt vervolgens gerenderd met OpenGL en krijgt de kleur mee van de pixel uit het camerabeeld. De output van dit algoritme resulteert in een 3D puntenwolk zoals weergegeven in Figuur 6.13. In Figuur 6.14 vindt u een uitgebreider voorbeeld van een scène waar een hoofd wordt bewogen. Indien op de puntenwolken een surface reconstruction algoritme wordt toegepast, bijvoorbeeld het *Ball-Pivoting Algorithm* van *Bernardini et al.* [8], worden deze puntenwolken omgezet in een mesh (zie Figuur 6.15). De rendering is uitgevoerd met *MeshLab* [47].

(a) (b) (c)

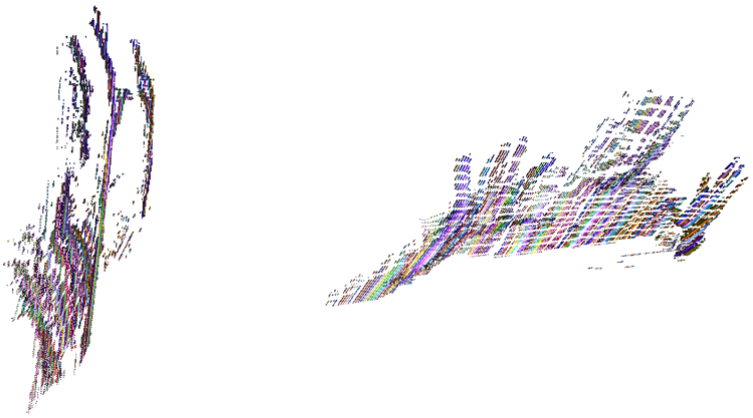
Figuur 6.13: Links: inputbeeld; Midden: dieptemap voor reconstructie; Rechts: 3D gereconstruuerde puntenwolk na triangularisatie.



(a) Inputbeeld (b) Dieptemap

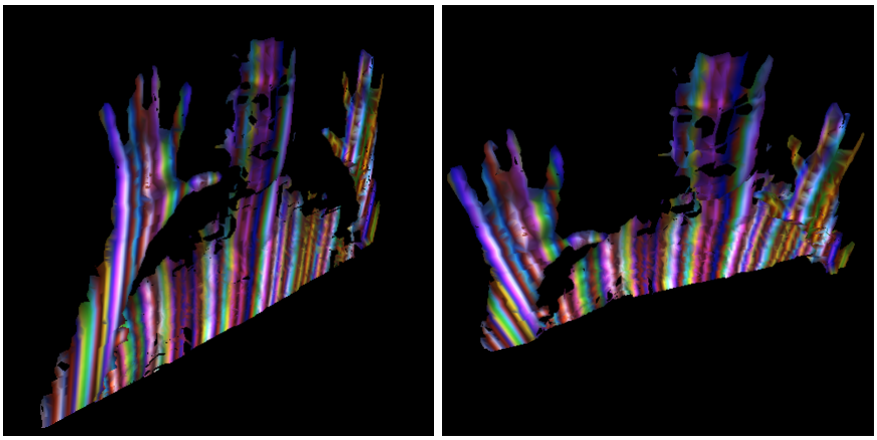


(c) Puntenwolk (d) Puntenwolk



(e) Puntenwolk (f) Puntenwolk

Figuur 6.14: 3D reconstructie van een hoofd. (a) het inputbeeld; (b) dieptemap; (c) tot (f) 3D puntenwolk bekeken vanuit verschillend standpunten.



(a) (b)

Figuur 6.15: Surface reconstruction met het *Ball-Pivoting Algorithm* van *Bernardini et al.* [8].

6.7 Registratie

Nu puntenwolken worden gegenereerd, is het ook belangrijk om de verschillende puntenwolken, afkomstig uit de verschillende frames, te registreren in een globaal model. Een veel gebruikte techniek om data te registreren in een model is *Iterative Closest Point* (ICP). In de eerste sectie beschrijft deze thesis een standaard implementatie van ICP die punten aan elkaar koppelt op basis van de kortste afstand. Daarnaast kan het algoritme worden verfijnd door ook extra geometrie van de scène te beschouwen. Dit zal ten koste gaan van extra berekeningen en processing tijd.

6.7.1 ICP op basis van dichtste punt in het model

Als testmodel wordt er gebruik gemaakt van een puntenwolk die de geometrie bevat van een kubus. Daarnaast is er een tweede puntenwolk die geregistreerd moet worden in het model. Om het algoritme te testen is er voor gekozen een ideale puntenwolk te kiezen. Deze wordt verkregen door op de puntenwolk van het model een perspectieve transformatie toe te passen. In Figuur 6.16 is het model aangegeven in het rood en de puntenwolk die moet worden geregistreerd in het blauw.

De eerste stap van het algoritme bestaat er uit een aantal correspondentieparen te kiezen. De data puntenwolk wordt gesampled (het kiezen van een deelverzameling van de punten) en vervolgens worden er correspondentieparen gezocht in het model. In de meest eenvoudige implementatie wordt er gekozen voor het punt waarvan de kwadratische afstand het kleinste is. Dit punt wordt op een optimale manier bekomen door het volledige model

in een KD-tree te plaatsen (bijvoorbeeld in een implementatie van *Martin F. Kraft*[2]) en vervolgens het punt van de data puntenwolk als target te kiezen. Indien er N samples worden gemaakt, worden de correspondentieparen als volgt voorgesteld:

$$\left(p_i, p'_i\right), \text{ met } 0 \leq i < N.$$

Eens de correspondentieparen zijn gevonden, moet de optimale transformatie worden berekend. Dit kan aan de hand van de *covariantie matrix* [77]. De bedoeling is de euclidische afstand tussen de correspondentieparen te minimaliseren, ofnog het minimaliseren van de volgende error:

$$E = \sum_{i=0}^{N-1} \left\| p'_i - (Rp_i + T) \right\|^2 \tag{6.18}$$

. Het is aan te raden om de rotatie te ontkoppelen van de translatie. Dit kan door de centroid (p, p') van beide puntenwolken te berekenen en deze van elk punt af te trekken ($q_i = p_i - p$ en $q'_i = p'_i - p'$). Zo hebben beide puntenwolken hetzelfde centrum en zijn ze genormaliseerd. Nadat de optimale rotatie voor deze genormaliseerde puntenwolken is berekend, volgt hieruit ook de optimale translatie.

Rotatie

In de technische beschrijving van *Stamos et al.* [77] toont men aan dat de kostfunctie kan herschreven worden naar het maximaliseren van de volgende waarde:

$$\begin{aligned} E' &= \sum_{i=0}^{N-1} q_i'^t R q_i & (6.19)\\ &= Trace(R \sum_{i=0}^{N-1} q_i q_i'^t) & (6.20)\\ &= Trace(RH) & (6.21) \end{aligned}$$

, waarbij $H = \sum_{i=0}^{N-1} q_i q_i'^t$ gekend is als de covariantie matrix van twee genormaliseerde puntenwolken. Een eigenschap van de covariantie matrix is dat het een eigenvector heeft die de rotatie as beschrijft, dewelke de twee puntenwolken op elkaar mapt. Om tot de rotatie matrix te komen moet H eerst ontleed worden met behulp van SVD in $H = UDV^t$. Er kan vervolgens worden bewezen [77] dat $Trace(RH)$ maximaal is voor de volgende uitdrukking:

$$R = VU^t. \tag{6.22}$$

Opgelet, de determinant van R mag niet 1 zijn. Indien dit wel het geval is, kan het algoritme foute resultaten teruggeven. Op deze site[3] worden er meer technische details

[2] http://libkdtree.alioth.debian.org/

[3] http://www.morethantechnical.com/2010/06/06/iterative-closest-point-icp-with-opencv-w-code/

over de implementatie gegeven. In het geval dat de determinant 1 is, gebruiken ze de volgende rotatie:

$$R = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(V^t U) \end{bmatrix} V^t. \tag{6.23}$$

Translatie

Uit deze gevonden rotatie matrix kan nu ook de optimale translatie worden berekend. Dit gebeurt door gebruik te maken van de twee centroids p' en p:

$$T = p' - Rp \tag{6.24}$$

. Het probleem bij het gebruik van deze translatie is dat de twee middelpunten altijd mooi op elkaar worden geschoven. De doelstelling is juist additief een scène te vergroten. Het is dus niet de bedoeling om telkens de puntenwolken mooi op elkaar te leggen, maar wel mooi tegen elkaar te schuiven. Daarom kan ook een tweede methode voor translatie worden gebruikt. Namelijk het berekenen van de gemiddelde translatievector voor elk correspondentiepunt. In plaats van enkel op basis van de twee centroids de verplaatsing te berekenen, wordt nu elke (p_i, p'_i) gebruikt om tot een totale gemiddelde translatie te komen, dus

$$T = \frac{\sum_{i=0}^{N-1} p'_i - Rp_i}{N}. \tag{6.25}$$

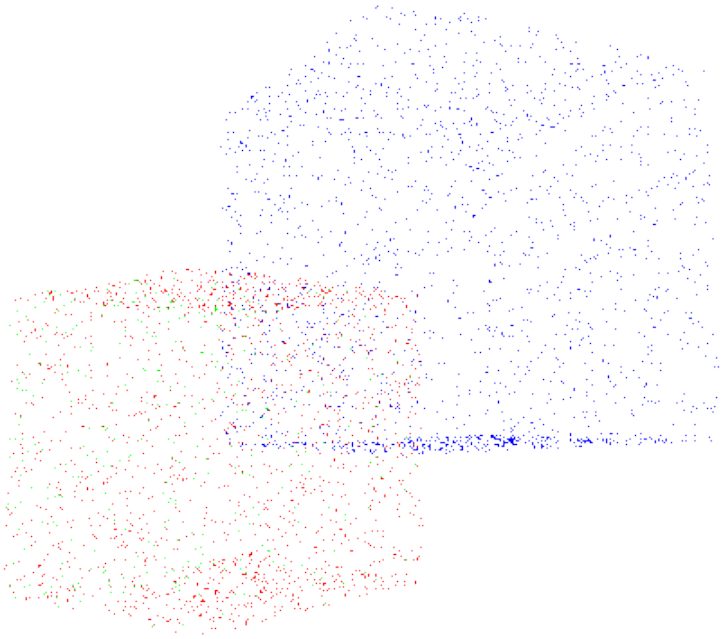
Vervolg ICP

Deze voorgaande berekeningen vormen een enkele iteratie in het ICP algoritme. Er gaan meerdere iteraties moeten worden uitgevoerd totdat er een bepaalde stopconditie wordt bereikt. In deze thesis zal het algoritme stoppen als de root-mean-square distance gelijk is aan die van de vorige iteratie, meer bepaald, als de registratie convergeert. In Figuur 6.16 ziet u het resultaat van het ICP algoritme. Zoals eerder aangehaald, wordt de blauwe puntenwolk in de rode geregistreerd. De groene puntenwolk is het resultaat van deze registratie. Omdat de groene en rode punten zo goed als samenvallen, zijn niet alle rode en groene punten even goed zichtbaar.

Problemen van ICP

Het probleem bij deze implementatie is dat er veel locale minima aanwezig zijn. Het kan goed zijn dat het algoritme te vroeg of naar het verkeerde minimim convergeert. Stel een scène met een doos op de voorgrond en een doek op de achtergrond. Indien beide objecten vrij vlak zijn is er geen unieke plaats waar de twee puntenwolken overeen kunnen schuiven. Zeker niet als het maar half overlappende puntenwolken zijn. Het is dus noodzakelijk dat er *genoeg diepte in de scène* zit om het algoritme goed te laten werken. Om de kwaliteit van de registratie te verhogen zal dus ook extra informatie over

scène in rekening worden gebracht bij het zoeken van juiste correspondentieparen. In de volgende sectie gaat er worden uitgelegd hoe er meer informatie over de geometrie van de scène kan worden gebruikt.



Figuur 6.16: Iterative Closest Point op een ideale puntenwolk, gebruik makend van het kortste punt in het model. Blauwe puntenwolk wordt in de rode geregistreerd. De groene punten zijn het resultaat van de registratie.

6.7.2 ICP gebruik makend van extra geometrie

Bae en Lichti [34] beschrijven in hun paper hoe de geometrie van de scène kan worden uitgebuit om het ICP algoritme te optimaliseren. Dit doen ze door twee extra factoren in rekening te nemen bij het kiezen van correspondenties. Ten eerste gebruiken ze het hoekverschil tussen de *normalen*. Daarnaast gaan ze ook nog de *verandering in curvature* gebruiken. Deze beschrijft of het oppervlak vrij vlak is of dat er veel vervorming op zit. In de eerste twee secties wordt uitgelegd hoe deze twee parameters worden bekomen voor een puntenwolk. De notatie uit de vorige sectie blijft behouden waar (p, p') de centroids van de puntenwolken zijn en (q_i, q_i') de genormaliseerde punten van de puntenwolk.

Normaal vector

De normaal vector wordt in elk punt van de puntenwolk geschat, gebruik makend van k neighborhood punten. Deze k kan vrij worden gekozen, maar hoe groter hij is, hoe beter de schatting. Deze neighborhood punten kunnen makkelijk worden gevonden met behulp

van de KD-boom die ook in het standaard algortime werd gebruikt. Deze moet nu k keer worden uitgelezen. De normaal vector van het oppervlak gevormd door p_i en zijn k neighourhood punten wordt benaderd door de eigenvector van de covariantie matrix met de kleinste eigenwaarde [35]. Deze covariantiematrix wordt als volgt opgesteld:

$$\begin{aligned} COV &= \sum_{n \in k(q_i)} (n - p_i) \otimes (n - p_i) \\ &= \sum_{n \in k(q_i)} \begin{bmatrix} (n.x - p_i.x)(n.x - p_i.x) & (n.x - p_i.x)(n.y - p_i.y) & (n.x - p_i.x)(n.z - p_i.z) \\ (n.y - p_i.y)(n.x - p_i.x) & (n.y - p_i.y)(n.y - p_i.y) & (n.y - p_i.y)(n.z - p_i.z) \\ (n.z - p_i.z)(n.x - p_i.x) & (n.z - p_i.z)(n.y - p_i.y) & (n.z - p_i.z)(n.z - p_i.z) \end{bmatrix} \end{aligned} \tag{6.26}$$

, waarbij $\otimes$ het outer product op een vector en $k(q_i)$ de verzameling van neighborhood punten van punt q_i.

Als deze covariantie m.b.v. SVD wordt ontleed in UDV^t zal de eigenvector met kleinste eigenwaarde, ofnog de laatste kolom van V, de beste schatting zijn van de normaal op het vlak gevormd door q_i en zijn neighborhood.

Change of curvature

De tweede parameter die wordt gebruikt is de verandering in curvature. Er zijn twee mogelijkheden om deze te berekenen. Ofwel op basis van de eigenwaardes van de covariantie matrix. Als λ_i de eigenwaardes zijn met $\lambda_1 \leq \lambda_2 \leq \lambda_3$. De verandering in oppervlak komt overeen met de afwijking van het opppervlak met het tangentieel oppervlak. De ratio van de minimale eigenwaarde en de som van de eigenwaardes benadert de verandering van het geometrische oppervlak.

$$M_{cc}(q_i) = \frac{\lambda_1}{\sum_{i=1}^{3} \lambda_i} \tag{6.27}$$

Beter is de geometrische verandering van oppervlak te schatten op basis van de normaal en de normalen van de buren.

$$M_{curve}(q_i) = \frac{1}{k} \sum_{n \in k(q_i)} \|normal_{q_i} - normal_n\| \tag{6.28}$$

De verandering in curvature kan nu worden uitgedrukt als volgt:

$$M_{cc}(q_i) = \frac{1}{k} \sum_{n \in k(q_i)} |M_{curve}(p_i) - M_{curve}(n)| \tag{6.29}$$

. De curvature bepaalt de afwijking van de naburige normalen, waar de change of curvature ook de curvature van de buren in rekening neemt en dus ook de afwijking van de normalen van de buren van de neighbordhood. De berekeningen van de parameters hoeft maar één keer te gebeuren als preprocessing stap. De verandering in curvature zal

niet veranderen per iteratie en per iteratie kunnen de normalen per punt mee getransformeerd worden. Soms kan het wel nuttig zijn om de geometrische parameters voor het model opnieuw te berekenen als er al een aantal registraties hebben plaatsgevonden. Dit omdat het model nieuwe punten bevat en dus eventueel ook nieuwe informatie om de geometrische parameters correcter te bekomen. Het berekenen van de geometrische parameters is een zware processing stap.

Hoe gebruiken binnen ICP?

Nu voor elk punt in de puntenwolk extra informatie is verkregen betreffende de geometrie, kan deze worden uitgebuit in het zoeken naar correspondenties in het model. Zo zal er voor een punt q_i niet enkel meer worden gezocht naar een correspondentiepunt q'_i op basis van de nabijheid, maar ook op basis van de hoek tussen de normalen en de verandering in curvature.
Een goede metriek om het verschil tussen twee normalen aan te geven is de volgende hoek θ:

$$\theta(q_i, q'_i) = cos^{-1}(\eta_{q_i} \cdot \eta_{q'_i}), \tag{6.30}$$

waar η_{q_i} en $\eta_{q'_i}$ respectievelijk de geschatte normalen zijn van q_i en q'_i.
Het verschil in geometrische curvature tussen twee punten kan beschreven worden als

$$\beta(q_i, q'_i) = \left| M_{cc}(q_i) - M_{cc}(q'_i) \right|. \tag{6.31}$$

Doordat $M_{cc}(p_i)$ onafhankelijk is van de oriëntatie of positie van de puntenwolk is het nuttig om in de eerste iteraties samples te kiezen waar de M_{cc} hoog is. Dit zorgt voor een ruwe registratie. Per iteratie kan deze threshold worden verlaagd. Eens deze ruwe registratie heeft plaatsgevonden, is het belangrijk de normalen méér in rekening te nemen. Dit kan in combinatie met het traditionele dichtste punt eerst. De registratie zal robuuster werken omdat er nu ook meer informatie over de geometrie wordt gebruikt. De voorwaarde is wel dat beide parameters zo nauwkeurig mogelijk worden berekend. Indien er veel ruis aanwezig is, zal dit al een groter probleem vormen en zal de neighborhood k moeten worden verhoogd.

6.7.3 Samengevat

Voor een model puntenwolk W_1 en een puntenwolk W_2 werkt ICP als volgt:

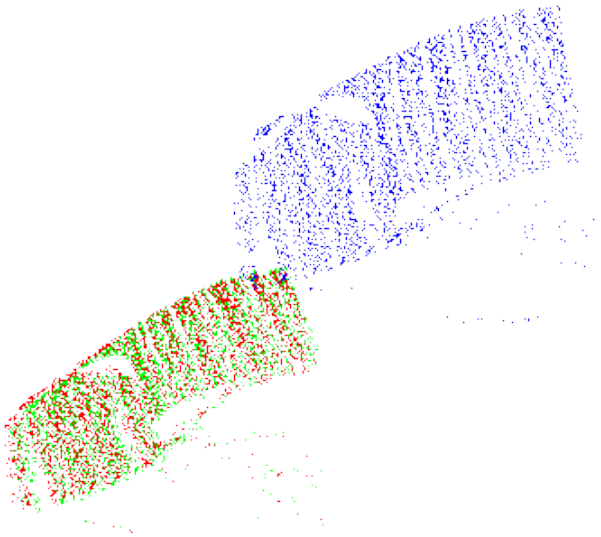
1. Vind de k buren van elk punt in W_1 en W_2 en bereken de geometrische parameters.
2. Neem een aantal samples q_i uit W_2 waarvan de M_{cc} groter is als een bepaalde threshold.
3. Zoek de correspondentiepunten q'_i uit W_1 die de volgende parameters minimaliseren: $\theta(q_i, q'_i)$, $\beta(q_i, q'_i)$ en $distance(q_i, q'_i)$).
4. Bereken de optimale transformatie R en T die de punten q_i mapt op de punten q'_i.

5. Pas de rotatie R en translatie T toe op W_2, aslook op de normalen van W_2.

6. Update eventueel de thresholds die worden gebruikt voor sampling en het zoeken van correspondentiepunten.

7. Bereken de registratie error $\epsilon^{iter=i}$, dewelke de root-mean-square distance is van de punten p_i tot q'_i. Indien $\epsilon^{iter=i} \geq \epsilon^{iter=i-1}$ stopt het algoritme, anders ga naar stap 2.

Een voorbeeld van een realistische scène met meer geometrie en ruis ziet u in Figuur 6.17.



Figuur 6.17: Iterative Closest Point op basis van kortste punt en geometrische parameters.

Hoofdstuk 7

Resultaten

Voor deze thesis zijn er veel technieken besproken en geïmplementeerd. Er werd aangetoond dat met behulp van een aantal toolboxen de calibratie relatief makkelijk kan worden uitgevoerd, alsook werd de lezer aangereikt hoe de rectification kan gebeuren m.b.v. OpenCV en Matlab. Om textuur toe te voegen aan de scène zijn er drie soorten patronen ontwikkeld in een Matlab implementatie die elk opgebouwd zijn met een De Bruijn sequentie. Het toevoegen van textuur aan de scène heeft voor een correcte implementatie van structured light gezorgd. Er is ten eerste een naïf zoekalgoritme op basis van het zoeken naar kleuren uitgewerkt in een Matlab implementatie om te concluderen dat het niet de meest ideale oplossing is om hoog kwaliteit dieptemappen te bekomen. Er is daarom overgestapt naar dynamic programming, die als global optimization techniek wel de gewenste dieptemap teruggeeft. Om een 3D puntenwolk te verkrijgen uit een dispariteitsmap worden er backprojecties van de 2D punten naar 3D punten berekend. SVD is een grote hulp bij het oplossen van dergelijke wiskundige vergelijkingen. Als laatste zijn er verschillende variaties van het registratie algoritme ICP uitgetest. Zo is er het algemene algoritme dat puntenwolken mooi volgens de centroids in elkaar laat schuiven (zie voorbeeld kubus in Figuur 6.16), alsook een aantal verbeteringen die realistische puntenwolken in elkaar laten te schuiven op basis van extra geometrie. Er is ook gespeeld met een soort van convolutie, maar deze zou met een Fast Fourier Transform moeten worden geïmplementeerd om voldoende snel te werken. Dit hoofdstuk toont meerdere resultaten van bovenstaande implementaties en geeft een discussie van de verkregen resultaten. In het eerste deel van dit hoofdstuk wordt nader gekeken naar de 3D reconstructie resultaten, waar in het tweede deel de registratie wordt besproken.

7.1 3D reconstructie

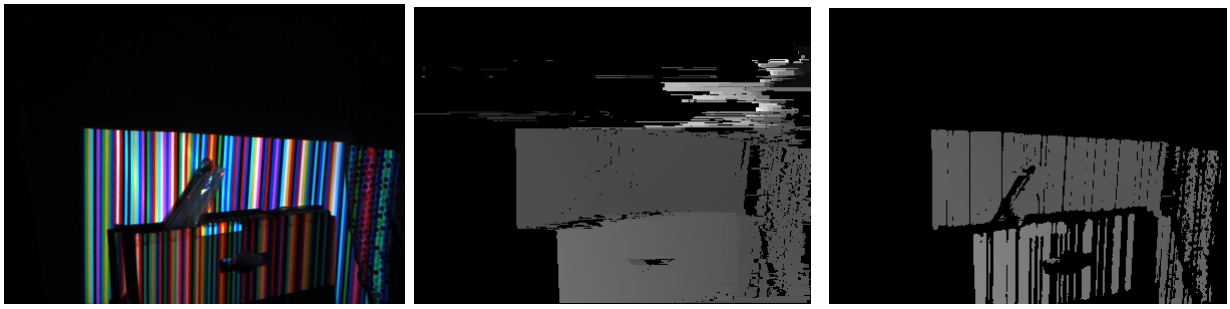
In Figuur 6.14 en Figuur 6.15 van het vorige hoofdstuk werden al een aantal voorbeelden van 3D reconstructie gegeven. In Bijlage A worden er extra resultaten van andere datasets getoond. Daarnaast staan er op de website van deze thesis nog 3D

reconstructies van bewegende scènes[1].

Kwaliteit

De kwaliteit van de dieptemap en 3D reconstructie hangt af van een aantal factoren. Ten eerste moeten de strepen van het geprojecteerd paroon klein genoeg zijn om voldoende detail te kunnen herkennen. Aan de andere kant mogen ze ook weer niet te smal zijn zodat ze slecht worden herkend. Het is aan te raden de dikte van de strepen te kiezen naargelang het detail van de scène en de afstand vanwaar deze wordt opgenomen.

Een tweede factor is het wegfilteren van de achtergrond. Het is aan te raden zwarte pixels te negeren en zo een minimale background substraction uit te voeren. Hierdoor gaan enkel punten waar extra textuur van het patroon aanwezig is, worden gebruikt in het 3D reconstructie algoritme. Dit zorgt dat er geen verkeerde informatie van de zwarte pixels wordt gebruikt en de dieptemappen zo minder fouten zullen bevatten. In Figuur 7.1 wordt dit geïllustreerd. Helemaal links is het waargenomen camerabeeld. De middelste afbeelding in de Figuur is de dieptemap zonder wegfilteren van zwart, de rechtste met het wegfilteren. Het is duidelijk dat de rechtse dieptemap een betere kwaliteit heeft dat de linkse. Het nadeel is dat nu ook het zwart uit de patronen is weggefilterd. Dit zorgt voor de zwarte verticale strepen in de dieptemap. Het is dus beter patronen te ontwikkelen die geen zwart bevatten. Hier is het geen groot probleem omdat alle 3D informatie niet ineens zichtbaar hoeft te zijn in één enkele frame. Door opeenvolgende frames samen te smelten zouden deze leegtes moeten worden opgevuld.



(a) Input image (b) Zonder filtering (c) Met filtering

Figuur 7.1: Zwart negeren in de reconstructie stap zorgt voor betere resultaten.

Performantie

De performantie van het 3D reconstructie algoritme vindt u terug in Tabel 7.1. U kunt zien dat de 3D reconstructie met de volledige resolutie van het inputbeeld lange tijd in beslag neemt. Een oplossing voor het versnellen, is het downsamplen van de input images.

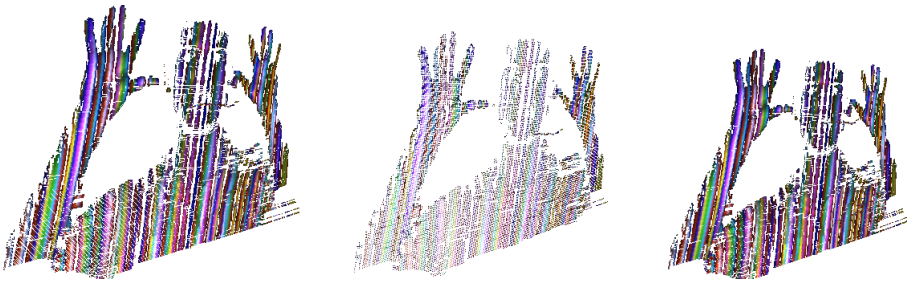
[1] http://feed-back.be/nick/masterthesis/resultaten.htm

Uit de Tabel kunt u afleiden dat er nu nog maar 12.45% van de totale tijd nodig is voor de berekeningen. Downsampling heeft niet direct een invloed op de triangulation (dit omdat er ongeveer evenveel punten wordt getraingulariseerd), maar wel op de dynamic programming stap zelf. In deze dynamic programming step is er 99% van de processing nodig voor het opstellen van de cost matrix. Het uittekenen van debug images in de user interface zorgt ervoor dat er een enorme hap extra processing nodig is.

Tabel 7.1: Vergelijkingen snelheden 3D reconstructie met en zonder downsampling

	Resolution	Dynamic progr.	Triangulation	Total scanline	Total depthmap
Original	800x600	766 ms	3.5 ms	770 ms	396.542 s
Downsampled	400x300	185 ms	3.3 ms	189 ms	49.350 s
Debug images	400x300	/	/	278 ms	74.178 s

Men kan zich nu afvragen: “Heeft het downsamplen een effect op de kwaliteit van de 3D reconstructie?”. In Figuur 7.3 worden twee gereconstrueerde puntenwolken naast elkaar gelegd. Figuur 7.2a is opgebouwd zonder downsampling en Figuur 7.2b met. U kunt zien dat de puntenwolk dense genoeg is met downsampling zodat er maar minimaal verlies is van 3D info. Dit verlies kan nog verder worden beperkt door ook upsampling toe te passen alvorens de punten te triangulariseren (zie Figuur 7.2c).



(a) 3D reconstructie zonder downsampling

(b) 3D reconstructie met downsampling

(c) 3D reconstructie met downsampling en upsampling

Figuur 7.2: 3D reconstructie. (a) met downsampling; (b) zonder downsampling; (c) met downsampling, maar alvorens triangularisatie ook nog upsampling.

Discussie

Over het algemeen geeft de 3D reconstructie de gewenste resultaten zoals vooropgesteld in deze thesis. Voor niet al te speculaire scènes met niet al te veel detail en scherpe overgangen lijkt het algoritme robuust te werken. Indien er veel discontinuïteiten aanwezig zijn, zal er een aangepaste dynamic programming moeten worden gemaakt. De second-pass implementatie van dynamic programming van *Zhang et al.* [89] is een uitstekende

kandidaat om dit probleem op te lossen. Dit gaat wel ten koste van extra processing omdat er nu meerdere malen doorheen het dynamic programming algoritme moet worden doorlopen.

Extra verbeteringen kunnen nog worden aangebracht door ook meer de De Bruijn sequentie van het patroon uit te buiten. Foute correspondenties kunnen worden verwijderd door logische tests toe te passen in verband met de opeenvolging van kleuren die gespecifieerd zijn in de sequentie.

Er kunnen zich ook problemen voordoen als er objecten op verschillende scherptelagen worden waargenomen. De focus van de projector ligt namelijk in een bepaald bereik waardoor de textuur op verdere of dichtere objecten onscherp kan worden. Dynamic programming kan hier vrij robuust met overweg omdat het voornamelijk naar de kleuren kijkt. Toch geeft het de beste resultaten als de volledige opname vanop een min of meer vaste afstand wordt uitgevoerd.

7.2 Registratie

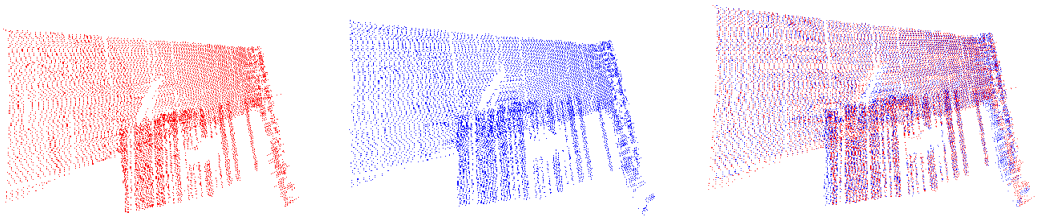
De registratie van puntenwolken is een niet voor de hand liggend probleem. De basisimplementatie van Iterative Closest Point werkt uitstekend voor puntenwolken die goed met elkaar overlappen. Echter, voor puntenwolken die in deze thesis worden gegenereerd is dit meestal niet het geval omdat de opstelling vrij kan rondbewegen en er dus geen garantie is dat de puntenwolken sterk overlappen. Het algoritme kan meer robuust worden gemaakt door ook de geometrie te beschouwen (zoals besproken in Sectie 6.7.2). Deze sectie verdiept zich verder in de resultaten en beschrijft waarom er zich nog problemen voordoen bij de registratie. Zo kan het algoritme convergeren naar meerdere lokale minima en is er niet altijd genoeg context in de scène aanwezig om correcte resultaten terug te geven.

Meerdere lokale minima

Als twee puntenwolken sterk overlappen zal ICP robuust werken, zeker in het geval van een goede initiële registratie. Doordat in deze thesis de framerate van de opname vrij hoog ligt ($\pm 10\ frames/s$), zal er al een relatief goede initiële registratie aanwezig zijn. Een voorbeeld van twee sterk overlappende scènes werd al gegeven in Figuur 6.17. De centroids van beide puntenwolken worden op elkaar gelegd en zorgen voor een goede starttoestand van het algoritme.

Stel nu dat twee puntenwolken elkaar maar gedeeltelijk overlappen, zoals bijvoorbeeld Figuur 7.3a en Figuur 7.3b. Beide beschrijven op het eerste zicht dezelfde scène, toch verschillen ze beide, zij het minimaal, van elkaar en kan het nefaste gevolgen hebben op het resultaat. Figuur 7.3a is het model en bevat links iets meer informatie als Figuur 7.3b, dewelke een data puntenwolk is die moet worden geregistreerd in het linkse model. Het

probleem bij zo'n scène is dat er te weinig diepte in zit. Het algoritme weet niet hoe de data puntenwolk correct op het model moet worden geplaatst. Er zijn meerdere mogelijkheden waar het algoritme een optimaal resultaat teruggeeft. Zo kan de data puntenwolk iets meer naar rechts of iets meer naar links worden geplaatst van het juiste resultaat. Hij pakt de oplossing met het beste lokale minima (zodat de gemiddelde kwadratische afstand minimaal is), maar dit wil niet zeggen dat dit het juiste resultaat is. Als resultaat geeft het algoritme Figuur 7.3c. Op eerste zich lijkt dit een correct resultaat. Toch is dit niet het goede resultaat. Het algoritme probeert zoveel mogelijk punten in elkaar te schuiven en ziet niet dat de rode puntenwolk iets meer naar links moeten liggen dan de blauwe. Let op de arm rechts in de twee puntenwolken. Men zou kunnen zeggen dat deze extra diepte toevoegt aan de scène en er m.a.w. voor zorgt dat het algoritme wel extra context heeft om juist te convergeren. Dit is echter niet waar. De hoeveelheid punten in de arm is in verhouding veel kleiner dan de hoeveelheid punten links in het vlak. Hierdoor zal het algorime een kleinere gemiddelde kwadratische afstand geven indien de punten van het vlak op elkaar liggen, dan dat de punten van de arm op elkaar liggen. Daarnaar is er ook weer niet zoveel diepteverschil in de arm om genoeg invloed te hebben op de gehele wolk.



(a) Model (b) Data (c) Geregistreerd

Figuur 7.3: Probleem registratie bij meerdere minima. De data puntenwolk kan op meerdere mogelijkheden geregistreerd worden in het model.

Het is dus duidelijk dat niet voor elke scène het registratie algoritme goed zal werken. Er moet gekozen worden voor een scène die genoeg diepteverschil bevat om de deels overlappende puntenwolken in elkaar te laten passen. Daarnaast is het ook belangrijk dat de geometrie berekeningen zo nauwkaurig mogelijk worden uitgevoerd. Deze helpen namelijk om het diepteverschil extra uit te buiten.

Een volledige oplossing voor het probleem zou het gebruik zijn van feature detection algoritmes. Het ICP algoritme kan zijn correspondenties dan baseren op de aanwezigheid van corresponderende features. Er bestaan krachtige feature detection algoritmes op de markt die snel zijn en makkelijk te implementeren om het algoritme uit te breiden. Zo kan het algoritme toch robuust worden om ook moeilijkere scènes in elkaar te registreren. Doordat feature detection een heel ander onderzoeksdomein is, wordt er in deze thesis niet dieper op in gegaan.

Performantie

De performantie van de registratie hangt af van een aantal instellingen van het ICP algoritme. Een belangrijke invloed is het aantal samples dat wordt genomen. Het is computationeel niet aan te raden om de volledige puntenwolk te gebruiken. Doordat de puntenwolk al vrij dense is kunnen ook subsets worden gebruikt. Daarnaast kan er gekozen worden om puur op basis van afstand te werken of ook extra geometrie parameters te gebruiken. Het berekenen van de geometrie parameters (normalen, curvature) heeft een sterke impact op de snelheid van het algoritme. Als laatste is het ook van belang een goede afweging te maken in de keuze van de grootte van de neighbordhood voor de geometrie berekeningen in functie van de tijd en kwaliteit ervan. Hoe groter de neighborhood, hoe correcter de geometrie berekening, maar hoe meer processing time er nodig is.

Tabel 7.2 geeft een overzicht van de performantie van het algoritme. Hieruit kan u afleiden dat het aantal gebruikte samples de grootste invloed heeft op de snelheid van het algoritme. Als het aantal gebruikte samples verdubbelt, gaat het algoritme drie keer vertragen (voor 1211 samples ±21 s, waar voor 2821 samples ±62 s). Het is ook duidelijk dat de geometrie berekeningen de grootste bottleneck vormt binnen het algoritme. De standaard ICP berekeningen werken daartentegen wel behoorlijk snel. De grootte van neighborhood (in de Figuur aangeduid met k) heeft ook een impact op de snelheid. Normaal is een neighborhood van 50 voldoende om de geometrie correct te berekenen, maar hoe groter deze wordt genomen hoe nauwkeuriger de berekeningen.

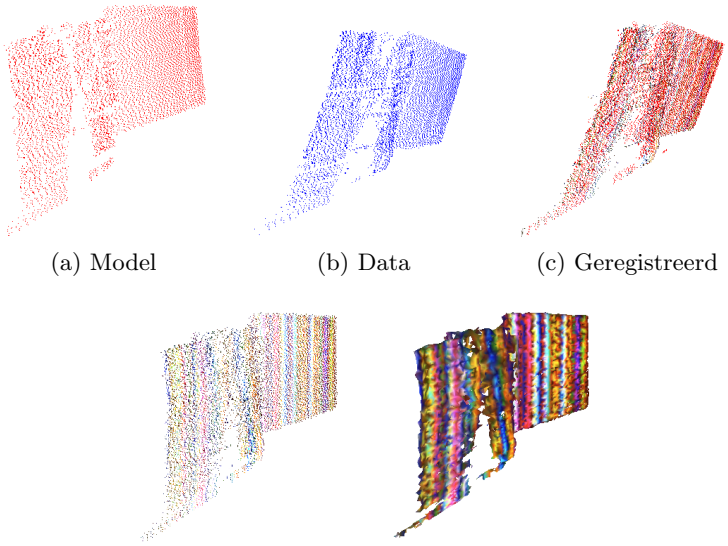
Tabel 7.2: Vergelijkingen snelheden registratie voor verschillende sample grootte en neighborhood

Aantal samples	k	Registration	Geometry	Total algorithm
9404	100	47.637 s	407.522 s	459.064 s
5642	100	17.164 s	142.844 s	179.306 s
2821	100	4.384 s	55.398 s	62.161 s
1411	100	1.313 s	22.191 s	21.023 s
1411	75	1.173 s	15.127 s	18.230 s
1411	50	1.164 s	8.740 s	12.281 s

Discussie

Een gewone implementatie van het ICP algoritme, eventueel uitgebreid met extra geometrische parameters is niet voldoende om elke scène fatsoenlijk te kunnen registreren. Scènes met weinig diepte hebben het meeste kans om te falen. In een aantal gevallen werkt het algoritme goed en geeft het de gewenste resultaten, maar in veel gevallen faalt het. Het maakt geen gebruik van feature detection algoritmes, maar kan eventueel wel

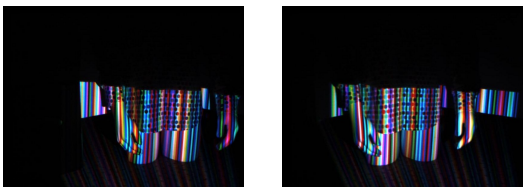
toegevoegd worden. Deze feature detection algoritmes kunnen het een robuust algoritme maken. Daarnaast zou er ook gebruik gemaakt kunnen worden van convolutie. Door de datawolk over de modelwolk te convuleren kan de best mogelijke translatie worden gevonden. Dit kan versneld worden m.b.v. een Fast Fourier Transformation. Deze oplossing is gelijkaardig aan het Extenden Gaussian Images algoritme uit Sectie 4.4. In Figuur 7.4 en Figuur 7.5 vindt u nog twee uitgebreide voorbeelden van de registratie.



(a) Model (b) Data (c) Geregistreerd

(d) Geregistreerd (e) Surface

Figuur 7.4: Uitgebreid voorbeeld van registratie



(a) Model image (b) Data image

(c) Model (d) Data (e) Geregistreerd

Figuur 7.5: Uitgebreid voorbeeld van registratie

Hoofdstuk 8

Conclusie

In deze thesis werd er getracht om algoritmes te vinden die met behulp van een camera en projector de one-shot dieptemappen van een scène kunnen bekomen die al deze dieptemappen kunnen samensmelten in één globaal model. Er mocht geen beperking gelegd worden op de beweging van de camera en de grootte van de scène. Het moest mogelijk zijn om doorheen een kamer rond te lopen en de gebruiker in real-time de kamer incrementeel te laten inscannen tot een 3D model.

Het verkrijgen van de dieptemap gebeurt door gebruik te maken van de actieve structured light techniek. Door de projectie van patronen kunnen correspondenties worden gezocht. Er werden drie grote categorieën besproken en hierbij werd duidelijk dat het algoritme zich moet beperken tot spatial neighborhood patronen. Op elke tijdsinstantie moet het algoritme een dieptemap kunnen bepalen omdat anders de camera niet meer vrij kan rondbewegen. Het gebruik van De Bruijn sequenties zijn zeker aan te raden. Deze zorgen voor een formele opbouw van het patroon zodat problemen worden vermeden. Om problemen met kleuren te vermijden is er geopteerd om correspondenties niet langer te zoeken tussen projector en camera, maar wel tussen twee camera's. In deze thesis is de global optimization dynamic programming techniek geïmplementeerd. Het is een vrij robuust algoritme dat een relatief dense en ruisvrije dieptemap teruggeeft.

Bij het aligneren van puntenwolken werd duidelijk dat voor een nauwkeurige registratie Iterative Closest Point een veel gebruikte techniek is. Er zijn dan ook al veel varianten voor ontwikkeld die het algoritme verfijnen om betere resulaten te verkrijgen. Het is een robuust algoritme dat voor sterk overlappende puntenwolken binnen een beperkte tijd goede resultaten geeft. Doordat, in onze opstelling, de camera constant beelden opneemt en dieptemappen berekent, is er maar weinig verandering tussen twee opeenvolgende beelden, waardoor ICP uitermate geschikt is. In deze thesis wordt het algoritme geoptimaliseerd door het uitbuiten van geometrische parameters van de scène, zoals normalen en change of curvature. Deze aanpak zorgt ervoor dat minder overlappende puntenwolken met genoeg diepte toch kunnen worden geregistreerd.

Bibliografie

[1] E. H. Adelson, J. R. Anderson, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, November/December 1984.

[2] D. Aliaga and Y. Xu. Photogeometric structured light: A self-calibrating and multi-viewpoint framework for accurate 3d modeling. *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.

[4] Stephen T. Barnard and Martin A. Fischler. Computational stereo. *ACM Comput. Surv.*, 14(4):553–572, 1982.

[5] P. Beardsley, J. Vanbaar, and R. Raskar. Augmenting a projector-camera device with laser pointers. Technical Report 2004/035, Mitsubishi Electric Research Laboratories, UNC Chapel Hill, March 2004.

[6] R. Benjemaa and F. Schmitt. Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, page 113, Washington, DC, USA, 1997. IEEE Computer Society.

[7] F. Bernardini and H. Rushmeier. The 3d model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[8] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, Gabriel Taubin, and Senior Member. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5:349–359, 1999.

[9] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[10] Gérard Blais and Martin D. Levine. Registering multiview range data to create 3d computer objects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):820–824, 1995.

[11] K. L. Boyer and A. C. Kak. Color-encoded structured light for rapid active ranging. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(1):14–28, 1987.

[12] Gottesfeld Lisa Brown. A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325–376, 1992.

[13] Brian Carrihill and Robert Hummel. Experiments with the intensity ratio depth sensor. *Computer Vision, Graphics, and Image Processing*, 32(3):337 – 358, 1985.

[14] C. Chen, Y. Hung, C. Chiang, and J. Wu. Range data acquisition using color structured lighting and stereo vision. *Image and Vision Computing*, 15:445–456, 1997.

[15] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992.

[16] Zezhi Chen, Chengke Wu, and Hung Tat Tsui. A new image rectification algorithm. *Pattern Recognition Letters*, 24(1-3):251 – 260, 2003.

[17] D. Chetverikov, D. Svirko, D. Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *In International Conference on Pattern Recognition*, pages 545–548, 2002.

[18] Ingemar J. Cox, Sunita L. Hingorani, Satish B. Rao, and Bruce M. Maggs. A maximum likelihood stereo algorithm. *Comput. Vis. Image Underst.*, 63(3):542–567, 1996.

[19] Pavel Czek. Least trimmed squares in nonlinear regression under dependence. *Journal of Statistical Planning and Inference*, 136(11):3967 – 3988, 2006.

[20] U.R. Dhond and J.K. Aggarwal. Structure from stereo - a review. *IEEE Trans. on Systems, Man, and Cybern.*, 19(6):1489–1510, 1989.

[21] Chitra Dorai, Gang Wang, Anil K. Jain, and Carolyn Mercer. Registration and integration of multiple object views for 3d model construction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):83–89, 1998.

[22] N.G. Durdle, J. Thayyoor, and V.J. Raso. An improved structured light technique for surface reconstruction of the human trunk. In *IEEE Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 874–877, 1998.

[23] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*, pages 188–189. MIT Press, Cambridge, MA, USA, 1993.

[24] Olivier D. Faugeras, Quang-Tuan Luong, and Stephen J. Maybank. Camera self-calibration: Theory and experiments. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 321–334, London, UK, 1992. Springer-Verlag.

[25] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. pages 726–740, 1987.

[26] Andrew W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13-14):1145 – 1153, 2003. British Machine Vision Computing 2001.

[27] H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *Society of Industrial and Applied Mathematics Review*, 24(2):195–200, 1982.

[28] A. Fusiello, E. Trucco, and A. Verri. Epipolar rectification. World Wide Web, 2010. `http://profs.sci.univr.it/~fusiello/demo/rect/`.

[29] G. Godin, M. Rioux, and R Baribeau. Three-dimensional registration using range and intensity information. *Proc. SPIE: Videometrics III*, 2350, 1994.

[30] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, chapter 9. Morphological Image Processing; 10. Image Segmentation Image Restoration and Reconstruction, pages 627–794. Pearson International Edition, Upper Saddle River, New Jersey 07458, 3th edition, 2008.

[31] Olaf Hall-Holt and Szymon Rusinkiewicz. Stripe boundary codes for real-time structured-light range scanning of moving objects. In *Eighth International Conference on Computer Vision (ICCV)*, July 2001.

[32] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[33] Samuel W. Hasinoff, Sing Bing Kang, and Richard Szeliski. Boundary matting for view synthesis. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 11*, page 170, Washington, DC, USA, 2004. IEEE Computer Society.

[34] Kwang ho Bae and D. D. Lichti. Automated registration of unorganised point clouds from terrestrial laser scanners. In *In: International Archives of Photogrammetry and Remote Sensing, Vol. XXXV, Part B5, Proceedings of the ISPRS working group V/2*, pages 222–227, 2004.

[35] Hugues Hoppe, Tony DeRose, Tom Duchamp, John Mcdonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *COMPUTER GRAPHICS (SIGGRAPH 92 PROCEEDINGS)*, pages 71–78, 1992.

[36] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[37] Berthold K. P. Horn, H.M. Hilden, and Shariar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOURNAL OF THE OPTICAL SOCIETY AMERICA*, 5(7):1127–1135, 1988.

[38] H. Hügli and G. Maïtre. Generation and use of color pseudo random sequences for coding structured light in active ranging. In *Proceedings SPIE of Industrial Inspection*, volume 1010, pages 75–82, 1988.

[39] S. Inokuchi, K. Sato, and F. Matsuda. Range imaging system for 3-d object recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 806–808, 1984.

[40] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[41] Michael Isard and Andrew Blake. Icondensation: Unifying low-level and high-level tracking in a stochastic framework. pages 893–908, 1998.

[42] M. Ito and A. Ishii. A three-level checkerboard pattern (tcp) projection method for curved surface measurement. *Pattern recognition*, 28(1):27–40, 1995.

[43] T. Jebara, A. Azarbayejani, and A. Pentland. 3d structure from 2d motion. *Signal Processing Magazine, IEEE*, 16(3):66–84, May 1999.

[44] A. E. Johnson and Sing Bing Kang. Registration and integration of textured 3-d data. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, page 234, Washington, DC, USA, 1997. IEEE Computer Society.

[45] Hakan Kjellerstrand. de bruijn sequence. World Wide Web, 2010. `http://www.hakank.org/comb/debruijn.cgi`.

[46] Graphics & Media Lab. Gml c++ calibration toolbox. World Wide Web, 2006. `http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html`.

[47] Visual Computing Lab. Meshlab. World Wide Web, 2010. `http://meshlab.sourceforge.net/`.

[48] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[49] Peihua Li and Tianwen Zhang. Visual contour tracking based on particle filters. *Image and Vision Computing*, 21:111–123, 2002.

[50] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[51] Ameesh Makadia, Alexander IV Patterson, and Kostas Daniilidis. Fully automatic registration of 3d point clouds. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1297–1304, Washington, DC, USA, 2006. IEEE Computer Society.

[52] M. Maruyama and S. Abe. Range sensing by projecting multiple slits with random cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(6):647–651, 1993.

[53] T. Masuda, K. Sakaue, and N. Yokoya. Registration and integration of multiple range images for 3-d model construction. In *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, page 879, Washington, DC, USA, 1996. IEEE Computer Society.

[54] Nick Michiels. Augmented reality for workbenches. Hasselt University, Bachelorthesis, 2009.

[55] T. P. Monks and J. N. Carter. Improved stripe matching for colour encoded structured light. In *CAIP '93: Proceedings of the 5th International Conference on Computer Analysis of Images and Patterns*, pages 476–485, London, UK, 1993. Springer-Verlag.

[56] P. J. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *SMA '97: Proceedings of the 1997 International Conference on Shape Modeling and Applications (SMA '97)*, page 130, Washington, DC, USA, 1997. IEEE Computer Society.

[57] OpenCV. Camera calibration and 3d reconstruction. World Wide Web. `http://opencv.willowgarage.com/documentation/cpp/camera_calibration_and_3d_reconstruction.html`.

[58] OpenCV. Opencv. World Wide Web. `http://opencv.willowgarage.com/wiki/`.

[59] Jordi Pagès, Joaquim Salvi, and Joan Battle. Pattern codification strategies in structured light systems. *Pattern recognition*, 37(4):827–849, 2004.

[60] Jordi Pagès, Joaquim Salvi, Christophe Collewet, and Josep Forest. Optimised de bruijn patterns for one-shot shape acquisition. *Image Vision Comput.*, 23(8):707–720, 2005.

[61] Jordi Pagès, Joaquim Salvi, and C. Matabosch. Robust segmentation and decoding of a grid pattern for structured light. In *Proc. 1St Iberian Conf. on Pattern Recognition and Image Analysis*, pages 689–696, LNCS 2652, Mallorca, Spain, 2003.

[62] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*, pages 448–452. Prentice Hall, 1982.

[63] Inc. Point Grey Research. Point grey grasshopper ccd firewire camera. World Wide Web, 2010. `http://www.ptgrey.com/products/flea2/flea2_firewire_camera.asp`.

[64] J. L. Posdamer and M. D. Altschuler. Surface measurement by space-encoded projected beam systems. *Computer Graphics and Image Processing*, 18(1):1 – 17, 1982.

[65] Kari Pulli. Multiview registration for large data sets. In *SECOND INTERNATIONAL CONFERENCE ON 3D DIGITAL IMAGING AND MODELING*, pages 160–168, 1999.

[66] Kari Antero Pulli. *Surface reconstruction and display from range and color data.* PhD thesis, 1997. Chairperson-Shapiro, Linda G.

[67] Ruwen Schnabel Rol and Wahl Reinhard Klein. Efficient ransac for point-cloud shape detection.

[68] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 438–446, New York, NY, USA, 2002. ACM.

[69] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *INTERNATIONAL CONFERENCE ON 3-D DIGITAL IMAGING AND MODELING*, 2001.

[70] F. Sadlo, T. Weyrich, R. Peikert, and M. Gross. A practical structured light acquisition system for point-based geometry and texture. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, pages 89–145, 20–21 June 2005.

[71] J. Salvi, J. Batlle, and E. Mouaddib. A robust-coded pattern projection for dynamic 3d scene measurement. *Pattern Recognition Letters*, 19(11):1055 – 1065, 1998.

[72] Romeil Sandhu, Samuel Dambreville, and Allen Tannenbaum. Particle filtering for registration of 2d and 3d point sets with stochastic dynamics.

[73] T. Sato. Multispectral pattern projection range finder. In *Proceedings of the Conference on Three-Dimensional Image Captuer and Applications II, SPIE*, volume 3640, pages 28–37, San Jose, California, 1999.

[74] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[75] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR '05: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 195–202, June 2003.

[76] K. N. Sivarajan and R Ramaswami. Lightwave networks based on de bruijn graphs. In *IEEE/ACM Transactions on Networking*, volume 2, pages 70–79, 1994.

[77] Ioannis Stamos. 3d photography: Point based rigid registration. World Wide Web. `http://www.compsci.hunter.cuny.edu/~ioannis/registerpts_allen_notes.pdf`.

[78] Changming Sun. Uncalibrated three-view image rectification. *Image and Vision Computing*, 21(3):259 – 269, 2003.

[79] Tomas Svoboda, Daniel Martinec, Tomas Pajdla, Jean-Yves Bouguet, Tomas Werner, and Ondrej Chum. Multi-camera self-calibration. World Wide Web, 2005. `http://cmp.felk.cvut.cz/~svoboda/SelfCal/index.html`.

[80] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, New York, NY, USA, 1994. ACM.

[81] P. Vuylsteke and A. Oosterlinck. Range image acquisition with a single binary-encoded light pattern. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(2):148–164, 1990.

[82] Michael W. Walker, Lejun Shao, and Richard A. Volz. Estimating 3-d location parameters using dual number quaternions. *CVGIP: Image Underst.*, 54(3):358–367, 1991.

[83] Michael Waschbüsch, Stephan Wrmlin, Daniel Cotting, Filip Sadlo, and Markus Gross. Scalable 3d video of dynamic scenes. *The Visual Computer*, 21(8–10):629–638, September 2005.

[84] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95–041, Computer Science, UNC Chapel Hill, 1995.

[85] L. Xu, Z. J. Zhang, H. Ma, and Y. J. Yu. Real-time 3d profile measurement using structured light. *Journal of Physics: Conference Series, International Symposium on Instrumentation Science and Technology*, 48:339–343, 2006.

[86] M. Young and E. Beeson. Viewpoint-coded structured light. In *IEEE '07: Computer Society Conf. on Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007.

[87] B. Zhang, Y. F. Li, and Y. H. Wu. Self-recalibration of a structured light system via plane-based homography. *Pattern Recogn.*, 40(4):1368–1377, 2007.

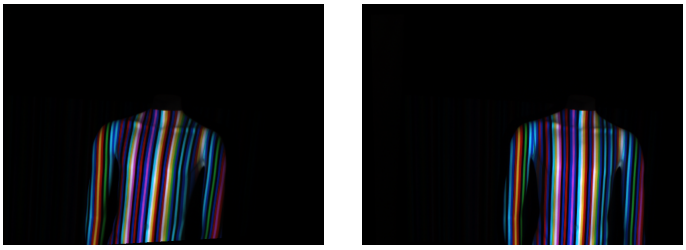
[88] Li Zhang, Brian Curless, Aaron Hertzmann, and Steven M. Seitz. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multi-view stereo. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 618, Washington, DC, USA, 2003. IEEE Computer Society.

[89] Li Zhang, Brian Curless, and Steven M. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *The 1st IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 24–36, June 2002.

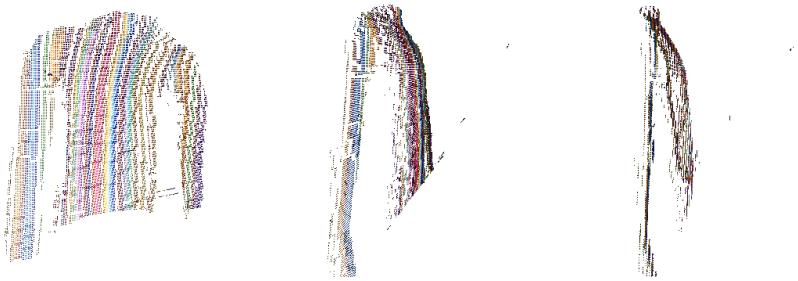
[90] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977 – 1000, 2003.

Bijlage A Voorbeelden reconstructie

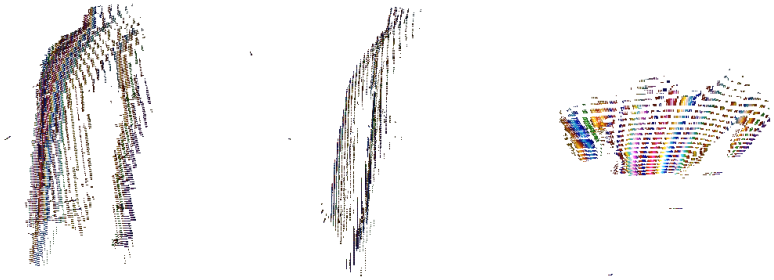
Scène 1 - Paspop



(a) Input left (b) Input Right



(c) Front (d) Left (e) Left

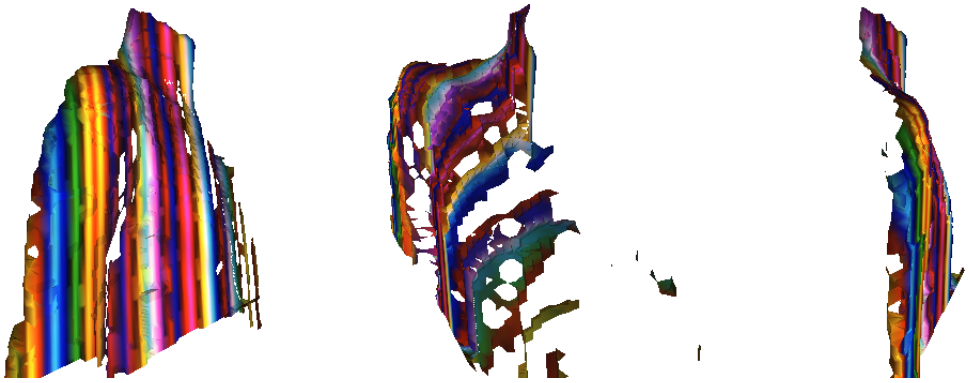


(f) Right (g) Right (h) Top

Figuur 1: 3D reconstructie. (a) en (b) zijn respectievelijk de linker en rechter opnamebeelden. (c) tot en met (g) verschillende point of views op de gereconstrueerde 3D puntenwolk.

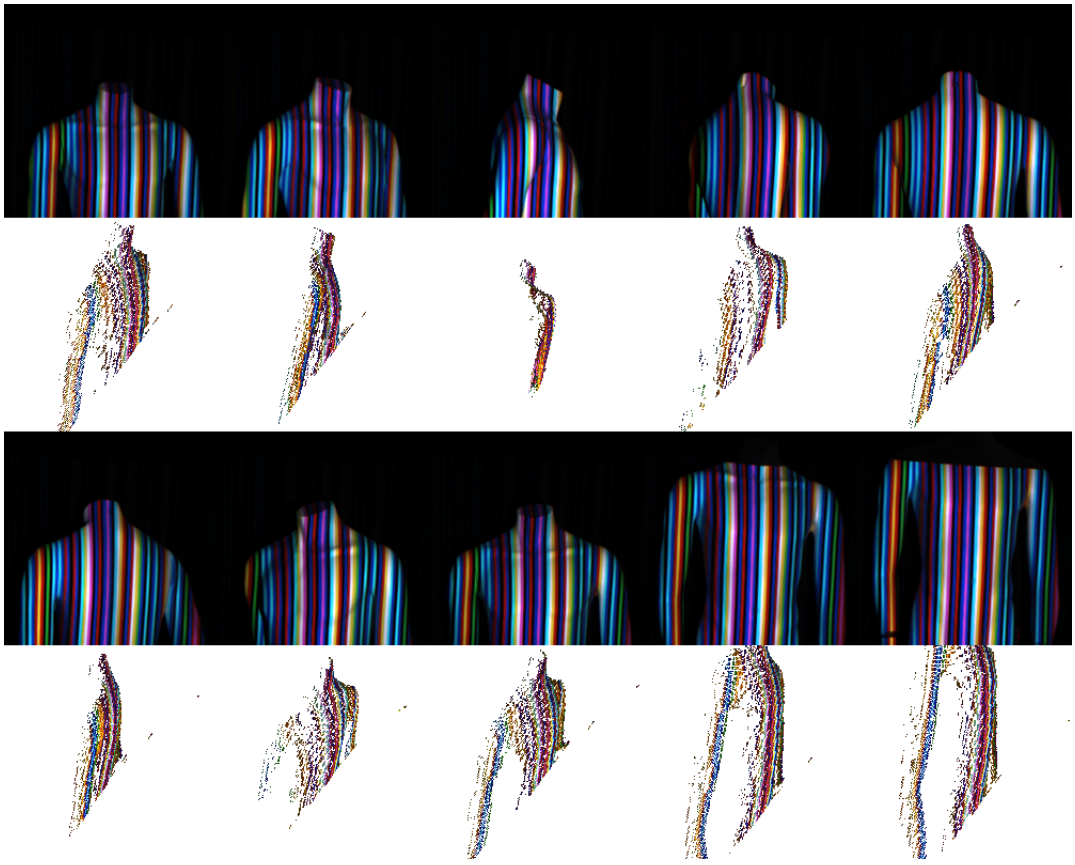
(a) Puntenwolk 1

(b) Puntenwolk 2



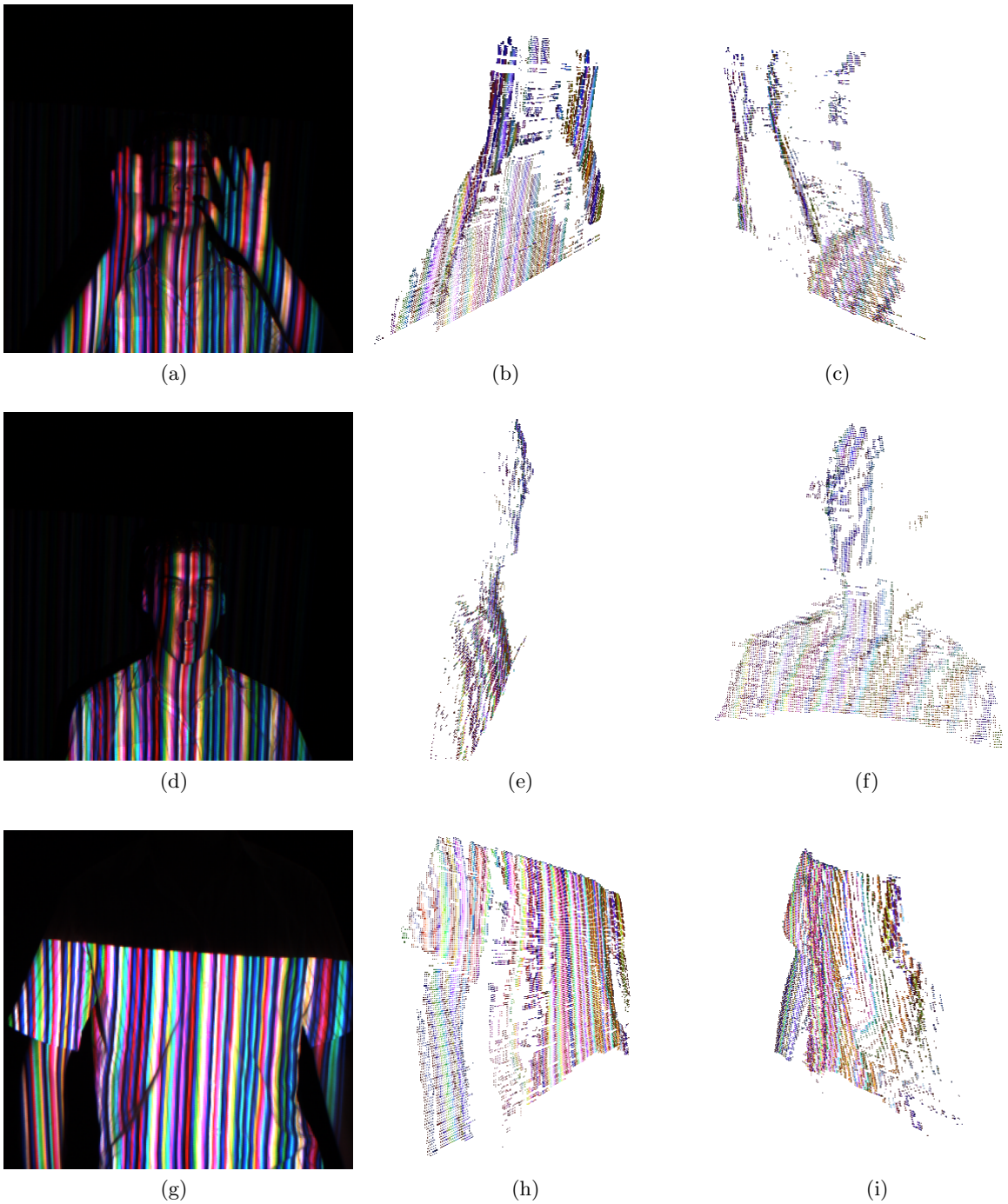
(c) Puntenwolk 3

Figuur 2: Ball pivoting algoritme uitgevoerd in MeshLab. (a), (b) en (c) zijn de surface reconstructions van drie verschillende puntenwolken.



Figuur 3: Gereconstrueerd zijaanzicht van een bewegende paspop. Bovenaan ziet u telkens het opgenomen camerabeeld. Onderaan de gereconstrueerde puntenwolk in zij-aanzicht (de camera is links van de puntenwolk geplaatst). Er is niet altijd evenveel informatie aanwezig om de 3D vorm er goed in te zien. Zo staat de camera in het derde beeld achter de paspop waardoor het maar minimale informatie laat zien van de links opgenomen puntenwolk.

Scène 2 - Hoofd

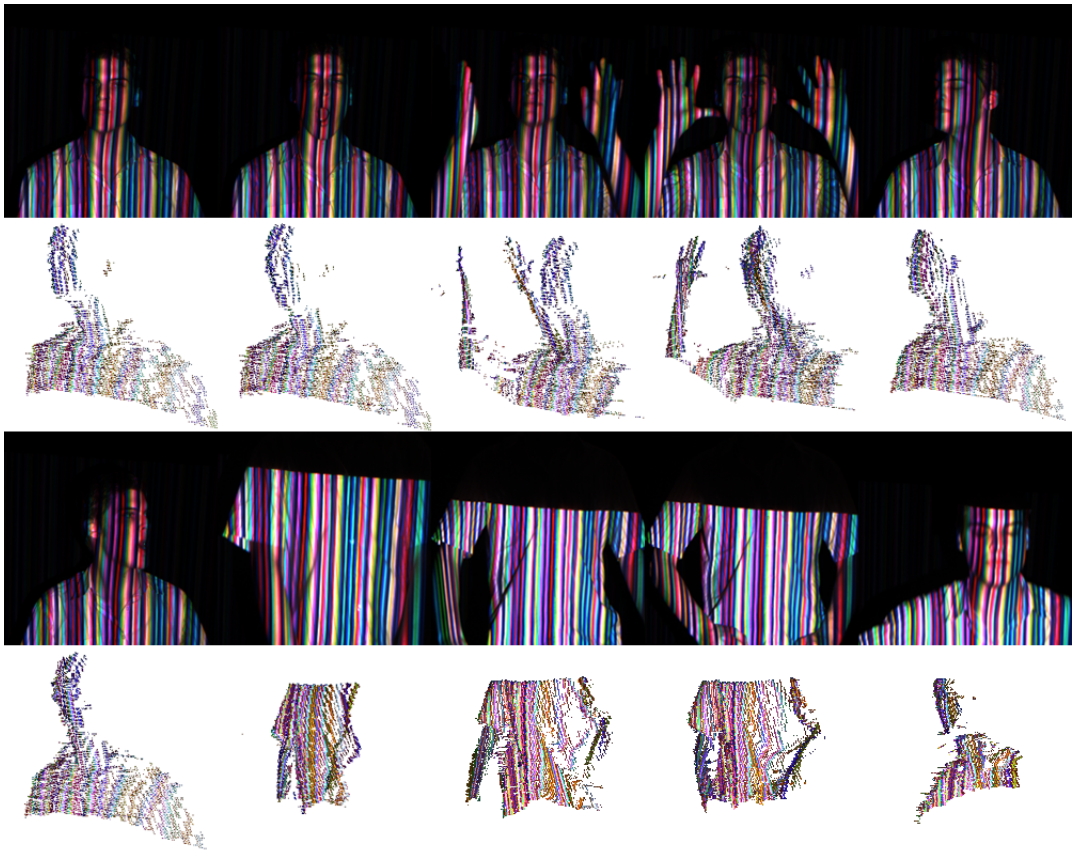


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figuur 4: Scène hoofd: Uitegebreide puntenwolken.



Figuur 5: Gereconstrueerd zijaanzicht van een bewegend hoofd. Bovenaan ziet u telkens het opgenomen camerabeeld. Onderaan de gereconstrueerde puntenwolk in zijaanzicht (de camera bekijkt de puntenwolk van de rechterkant).