Computervisie gebaseerde positie en orientatie tracking

Steven MAESEN

promotor :
Prof. dr. Philippe BEKAERT

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica afstudeerrichting multimedia

universiteit hasselt

Samenvatting

In deze thesis stellen we een nieuw inside-looking-out optisch tracking systeem voor. Voor dit systeem maken we enkel gebruik van goedkope markeringen en een gewone CCD camera die bevestigd wordt op het te tracken object.
We tonen aan dat we de oriëntatie onafhankelijk van de translatie of positie kunnen berekenen d.m.v. vluchtpunten. We stellen ook een nieuwe Hough transformatie voor die lijnen parametriseren a.h.v. cirkels. Deze implementatie van het algoritme van Hough werkt veel sneller dan de oorspronkelijke implementatie wanneer we lijnpatronen in relatief kleine puntensets zoeken. Positie tracking doen we aan de hand van puntovereenkomsten in opeenvolgende frames. Hierdoor kan de opstelling in principe oneindig groot gemaakt worden terwijl de kost van het uitbreiden beperkt blijft.
De algoritmes die we voorstellen, hebben we zowel in een virtuele als in een reële testopstelling geïmplementeerd. Beide implementaties geven een snel en robuust resultaat dat zeker wat betreft de accuraatheid van de oriëntatie kan concurreren met vele (dure) commerciële trackers die er op dit moment op de markt zijn.

Woord vooraf

Bij deze zou ik graag iedereen willen bedanken die mij gesteund heeft bij het schrijven van deze thesis en de voorbije academiejaren. Eerst en vooral wil ik mijn thesis promotor en begeleider Philippe Bekaert bedanken voor zijn hulp en ideeën die deze thesis mogelijk gemaakt hebben. Ik wil hem ook speciaal bedanken voor zijn tijd die hij in het nakijken en verbeteren van deze thesis heeft gestoken.
Ik wil ook Tom Haber, Cedric Vanaken en Bert De Decker bedanken voor hun hulp bij mijn stage om een echte testopstelling te bouwen.
Anderzijds wil ik ook mijn kameraden bedanken die ik op het LUC/Uhasselt heb leren kennen en waarmee ik 4 jaar lang hard gestudeerd, veel gelachen en vooral ook veel plezier mee gehad heb.
Ten slotte mogen mijn ouders hier ook niet ontbreken die mij de kans gegeven hebben om aan deze universiteit te studeren en mee geholpen hebben aan mijn academische loopbaan.

Inhoudsopgave

Hoofdstuk 1

Inleiding

Computervisie is de wetenschap die bestudeert hoe computers de inhoud van beelden kunnen verstaan. 'Verstaan' betekent hier dat er data uit de beelden gehaald en geïnterpreteerd wordt voor een speciaal doel. Eén van de vele takken in dit vakgebied is 'Tracking'. Tracking betekent het volgen van een bepaald object in een sequentie van beelden, indien we te maken hebben met een optisch trackingssysteem. In het volgend hoofdstuk zullen we ook kort een overzicht geven van andere soorten trackers.
Het precies tracken van objecten is een belangrijk onderwerp in onderzoeksgebieden zoals virtual reality en augmented reality. Hier moet de positie en oriëntatie van een persoon of een object dat deel wil nemen in de virtuele wereld nauwkeurig berekend worden zodat deze persoon zich in een andere realiteit waant.

1.1 Doelstelling

Het doel van deze thesis is het maken van een robuuste inside-looking-out optische tracker. We trachten ook een goedkoop systeem (in tegenstelling tot vele andere trackers) te bouwen door enkel gebruik te maken van een gewone CCD camera en goedkope markeringen in de scène zoals gekleurde plakkertjes of led-strips. Het trackingalgoritme moet ook snel genoeg zijn om elk beeld van de camera te kunnen verwerken en best nog CPU-tijd over hebben om deze gevonden transformatie nuttig te gebruiken.

1.2 Overzicht van de thesis

In hoofdstuk 2 geven we eerst een kort overzicht van verschillende soorten trackingsystemen die momenteel in gebruik zijn. Daarna bespreken we meer specifiek de technieken achter de optische tracker.
Hoofdstuk 3 geeft dan een overzicht van de stappen die onze optische tracker zal ondergaan. Ook wordt hier de opstelling van de scène beschreven.
In hoofdstuk 4 bespreken we hoe we de geplaatste markeringen kunnen herkennen in het invoerbeeld. We zien hier ook hoe dat we de optical flow van de markeringen kunnen bepalen en welke invloed lensdistortie heeft op de positie van de markeringen.
Uit deze set van markeringen gaan we in hoofdstuk 5 de lijnpatronen zoeken. Dit kunnen we doen door gebruik te maken van het algoritme van Hough. In dit hoofdstuk stellen we ook een nieuwe parametrisatie voor waardoor de Hough transformatie heel wat sneller wordt voor een relatief klein aantal markeringen.
In hoofdstuk 6 gaan we de oriëntatie berekenen uit de vluchtpunten van de lijnen die we in de vorige stap gevonden hebben en dit onafhankelijk van de translatie. We tonen hoe we de rotatiematrix kunnen opstellen a.h.v. de gevonden vluchtpunten en bewijzen dat we met 2 vluchtpunten al voldoende informatie hebben om de rotatie te bepalen.
In de laatste stap van de tracker (hoofdstuk 7) bepalen we de translatie die de camera ondergaan heeft. Hiervoor maken we gebruik van de berekende rotatiematrix uit de vorige stap en overeenkomstige markeringen in opeenvolgende frames.
In hoofdstuk 8 bespreken we eerst onze virtuele en reële testopstelling die we geconstrueerd hebben om ons trackingsysteem te testen. Daarna testen we hoe goed ons systeem functioneert in deze testopstellingen. We zien ook de vooruitgangen die we geboekt hebben op het vlak van snelheid, robuustheid en accuraatheid in de verschillende implementaties.
Ten slotte in hoofdstuk 9 geven we een overzicht van de technieken en resultaten die we onderzocht hebben. Ook geven we aan in welke richting we nog verder kunnen zoeken om het systeem te verbeteren.

Hoofdstuk 2

Stand van zaken

In dit hoofdstuk geven we een overzicht van bestaande oplossingen voor het trackingprobleem. In §2.1 geven we eerst een overzicht van trackingssystemen die momenteel gebruikt worden met hun voor- en nadelen. Hier zullen we ook een aantal commerciële systemen vergelijken met elkaar. Vervolgens zullen we in sectie 2.2 motiveren waarom we een inside-looking-out optisch trackingssysteem construeren. In sectie 2.3 bespreken we wat het camera calibratie probleem is en proberen we mogelijke oplossingen hiervoor te classificeren.

2.1 Overzicht soorten trackers

Tracking is het proces om de positie en de oriëntatie van een bewegend object te bepalen over een bepaalde tijdspanne. Er zijn heel wat oplossingen voor het trackingprobleem. We zullen een aantal soorten trackers bespreken die momenteel in omloop zijn. Hierbij baseer ik mij voornamelijk op een overzicht gegeven op een computergraphics conferentie in 2001 in de Verenigde Staten [1].

2.1.1 Akoestische tracking

Akoestische trackers werken meestal met ultrasonische geluidsgolven (frequentie boven de 20.000 Hz) om een afstand te meten. Deze zijn onhoorbaar voor het menselijk oor.

De algemene opbouw

Een zender en ontvanger in een akoestisch systeem kunnen hun onderlinge afstand berekenen. Laat ons veronderstellen dat in dit geval de zender vast blijft staan in de wereld (we kunnen ook de ontvanger vast nemen, waardoor hier de zender en ontvanger van rol wisselen). Hierdoor weten we dat de zender op een bol ligt rond de ontvanger met straal hun onderlinge afstand. Als we dan een tweede ontvanger nemen dan weten we dat de zender op een cirkel ligt (snijpunt van de 2 bollen). Door nog een derde ontvanger toe te voegen blijven er nog 2 mogelijke posities over waarvan meestal 1 onmiddellijk wegvalt. Om een 3D positie te bepalen zijn er dus 3 ontvangers en 1 zender nodig (of 3 zenders en 1 ontvanger). Wil men echter ook de oriëntatie weten dan heeft men 3 zenders en 3 ontvangers nodig.

Technieken

Om de afstand tussen zender en ontvanger te bepalen, maken de meeste akoestische trackers gebruik van de technieken Time of Flight (TOF) en Phase Coherence. Deze methodes zijn beiden gebaseerd op de snelheid van het geluid. Hierdoor komen onmiddellijk al een paar problemen naar voor. Geluid is zeer traag tegenover bijvoorbeeld licht, waardoor er een onoverkomelijke vertraging is gelijk aan de tijd nodig om geluidssignaal van zender naar ontvanger te laten gaan. De snelheid van een geluidssignaal is daar bovenop niet constant. Het hangt af van factoren zoals temperatuur en luchtdruk.

Time of Flight (TOF) Bij de TOF methode wordt de tijd t gemeten die het signaal nodig heeft om van zender naar ontvanger te gaan. Hierdoor kan men de afstand d tussen deze twee berekenen met behulp van de snelheid van het geluid v:

$$d[m] = v[m/s].t[s]$$

Phase Coherence De phase coherence methode meet de veranderingen in fase om hieruit een verandering in afstand te kunnen halen. Elk signaal is een som van harmonische functies van de vorm $A\cos(\omega t - \phi)$, met ϕ de faseverandering. Als men de golflengte λ of de snelheid c en frequentie f (want $c = \lambda f$) kent, kan men uit deze faseverandering ϕ_{delay}(t.o.v. de fase

aan de zender kant) als volgt de verplaatste afstand δ berekenen

$$\delta[m] = \lambda[m].\frac{\phi_{delay}[radialen]}{2\pi[radialen]}$$

of

$$\delta[m] = \frac{c[m/s]}{f[Hz]}.\frac{\phi_{delay}[radialen]}{2\pi[radialen]}$$

De absolute afstand tussen zender en ontvanger is dan $d' = d + \delta$.

We zien dat een sinusfunctie zich om de 2π radialen herhaalt, dus zien we ook dat een fase van $\phi + (n.2\pi)$ gelijk is aan ϕ. Hierdoor is de verplaatste afstand niet eenduidig bepaalt. Dit wordt meestal opgelost door de kleinste afstandsverandering te nemen bij een hoge update rate, behalve als de vorige situatie iets anders aangeeft(zoals een zeer hoge snelheid).

Voor- en nadelen

Voordelen	Nadelen
eenvoudige technieken	niet heel nauwkeurig (temperatuur, vochtigheid, druk kunnen resultaten beïnvloeden)
veel in gebruik	lage resolutie
relatief goedkoop	storing mogelijk van andere apparaten rond deze frequentie
	geen al te grote obstakels tussen zender en ontvanger

Tabel 2.1: Voor- en nadelen van akoestische tracking

2.1.2 Inertiële tracking

Inertiële trackers maken gebruik van versnellingsmeters om veranderingen in positie te bepalen en gyroscopen om veranderingen in oriëntatie te meten. Deze passieve trackers baseren zich op de tweede wet van Newton $F = ma$ en equivalent voor rotatie $M = I\alpha$. Deze trackers zijn niet gebonden aan

een bepaalde plaats omdat ze niet refereren naar een vast punt in de wereld. Deze bewegingsvrijheid heeft wel tot gevolg dat de absolute positie niet behouden blijft omdat de fouten zich blijven opstapelen en er geen manier van corrigeren is. Ook zeer kleine verandering in versnelling worden slecht opgemeten. Bijvoorbeeld als men heel traag stopt en de tracker heeft dit nie opgemeten dan blijft hij kleine verschuivingen aangeven terwijl het object stil staat. Dit fenomeen noemt men drift.
Het is duidelijk dat om een 6D positie te bepalen, er 3 versnellingsmeters voor positie en 3 gyroscopen voor oriëntatie nodig zijn.

Versnellingsmeters

Omdat versnelling a niet onmiddellijk gemeten kan worden, meet men de kracht F uitgeoefend op een lichaam met massa m. Hieruit berekent men de versnelling a door de wet van Newton $F = ma$, dus $a = \frac{F}{m}$. Deze versnelling wordt dan omgezet naar verplaatsing r door dubbele integratie:

$$r = \int \int a.dt^2$$

We bekijken nu hoe de versnelling berekend wordt. Zoals eerder gezegd, is de versnelling gelijk aan de verhouding tussen de kracht op een lichaam en de massa van dat lichaam. Dus men hangt een gekende massa aan een veer. Wanneer er geen versnelling is, heeft de massa geen verplaatsing. Wanneer het trackingapparaat wel een versnelling ondergaat, zal door inertie de massa achter blijven. Hierdoor meet de versnellingsmeter een verplaatsing van de massa, die evenredig is met de eigenlijke versnelling. Deze verplaatsing wordt in de praktijk omgezet naar een bruikbaar elektrisch signaal m.b.v. een potentiometer of een piezoelektrisch apparaat. Het voltage bij een potentiometer is evenredig met de positie van de massa op de meter. Bij een piezoelektrisch apparaat wordt de massa aan een piezoelektrische kristal gehangen, die op zijn beurt een voltage produceert in verhouding met de grootte van de kracht die er op in werkt (en dus ook de verplaatsing van de massa).

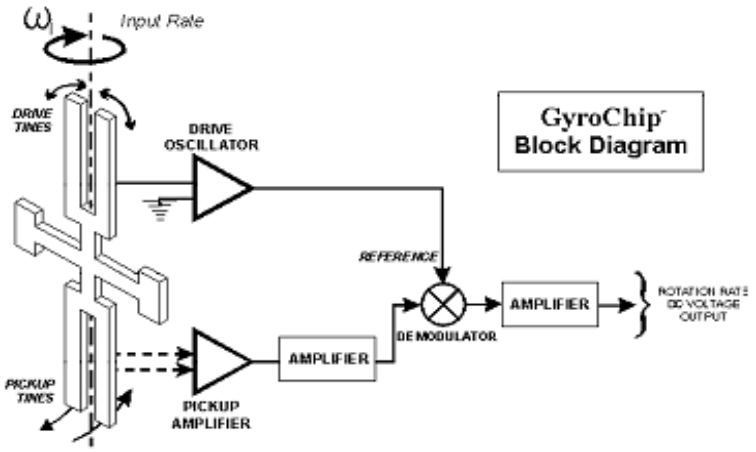
Gyroscopen

Gyroscopen[1] baseren zich op een gelijkaardig principe, namelijk de wet van behoud van hoekimpuls. Wanneer er een draaiende kracht (torsie) uitgeoefend wordt op een draaiende massa, zal zijn rotatieas meedraaien met de

[1]Veel interessante info en demonstraties van gyroscopen: http://www.gyroscopes.org

uitgevoerde torsie. Als deze massa zeer snel draait zal de gyroscoop een sterke weerstand hebben tegen oriëntatiewijzigingen. Hierdoor kan men een rotatie opmeten doordat de behuizing van de meter wel meedraait met de tracker, en de gyroscoop niet. Om een 3D oriëntatie te bepalen heeft men dus 3 gyroscopen nodig.
Deze methodes om inertiële tracking te doen kan men het beste aantonen met mechanische apparaten. Deze zijn in de praktijk (en vooral dan in ons geval waarbij ze op een gebruiker of object bevestigd moeten worden) onhandig doordat ze te groot en zwaar zijn. Daarom worden meestal micromechanische apparaten gebruikt in de vorm van chips.

Een voorbeeld van zo een chip (Figuur 2.1) is de BEI Systron Donner



Figuur 2.1: De BEI Systron Donner Inertial Division's GyroChip.

Inertial Division's GyroChip[2]. Deze maakt gebruik van een vibrerende piezoelektrische quartz stemvork om oriëntatieverschillen op te meten. Men laat de bovenste tanden (de 'drive tines') met een hoge frequentie van en naar elkaar trillen. Hierdoor wordt de vork gevoelig voor torsie rond een as evenwijdig met de tanden. Wanneer de tanden dicht bij elkaar staan roteert de vork sneller, terwijl wanneer ze verder weg staan vertraagt de rotatie van de vork. Bij een verandering van oriëntatie zullen de tanden aan de andere zijden van de vork (de 'pickup tines') op en neer bewegen uit het vlak van de vork. Dit zorgt voor elektrische outputsignalen die versterkt worden en daarna worden omgezet naar een bruikbaar DC signaal. Dit signaal is even-

[2]Een geanimeerd voorbeeld van de werking: http://www.systron.com/tech.asp

redig met de rotatiesnelheid en door deze 1 maal te integreren over de tijd bekomen we de rotatiehoek. Door 3 van deze stemvorken te nemen krijgen we 3 rotatiehoeken en dus de informatie nodig om aan oriëntatietracking te doen.

Voor- en nadelen

Voordelen	Nadelen
tracking gebeurt intern - geen extern apparaat of software meer nodig	last van drift
sterke groei waardoor kost en kwaliteit erop vooruitgaan	op dit moment nog duur
maakt geen gebruik van zenders, enkel het magnetisch veld van de aarde	opstapeling van meetfouten
zeer snel	geen absolute coördinaten
onbeperkt bereik	

Tabel 2.2: Voor- en nadelen van inertiële tracking

2.1.3 Magnetische tracking

Magnetische trackers maken gebruik van magnetische velden om zich te oriënteren en positioneren. Dit kunnen lage frequentie AC velden of gepulseerde DC velden zijn. Zowel de zender als de ontvanger hebben 3 spoelen die orthogonaal op elkaar staan. Hieruit kan men de positie en oriëntatie halen.

Magnetische velden

Bij magnetische tracking wordt er gebruik gemaakt van elektromagnetisme[3]. De magnetische velden worden gegenereerd door spoelen waarop men een spanning zet. De veldsterkte van een magnetisch veld hangt af van het aantal windingen en de lengte van de spoel, maar ook de stroomsterkte. De zender

[3]http://www.ham-radio.nl/cursus/elektromagnetisme.htm

creëert een magnetisch veld door op zijn spoelen een stroom te zetten. Door de stroomsterkte te wijzigen (door bijvoorbeeld wisselstroom of gepulseerde gelijkstroom te gebruiken) bekomt men dus wijzigingen in het magnetisch veld. Door magnetische inductie worden deze wijzigingen in het magnetisch veld opgemeten in de spoelen van de ontvanger. Hier wordt een spanning opgemeten die evenredig is met de oppervlakte van de spoel (constante), de grootte van de verandering in het magnetisch velden en de cosinus van de hoek die de ontvangende spoel maakt met de zendende spoel.

Opstelling

De zender bestaat uit 3 spoelen die een orthogonale basis vormen voor de wereldcoördinaten. Bijvoorbeeld de spoel die naar boven wijst zal overeenstemmen met de Z-as. Op elk van deze spoelen wordt om de beurt een elektrisch signaal gezet. Dit signaal wordt opgemeten in de orthogonale basis (3 spoelen) van de ontvanger. De sterkte van het opgevangen signaal hangt af van de afstand tot de zender (kwadratische afname) en de hoek die de ontvangende met de zendende spoel maakt (cosinus verhouding). Voor elke zendende spoel kan men dus 3 metingen doen en krijgt men dus 9 meetresultaten waaruit de 6 vrijheidsgraden (positie en oriëntatie) kunnen berekend worden.
Bij het opstellen van een magnetische tracker moet er uitgekeken worden voor metalen constructies die het magnetisch signaal kunnen verstoren. Dit komt omdat geleidende materialen zoals ijzer het magnetisch veld vervormen. Het magnetisch veld moet rond metalen objecten "stromen", het kan er niet indringen of doordringen waardoor de intensiteit en de oriëntatie van het veld verandert. Hierdoor kunnen metingen in de ontvanger verkeerd zijn en dus een totaal verkeerde positie en oriëntatie gevonden worden.

Voor- en nadelen

Voordelen	Nadelen
zender en ontvanger moeten elkaar niet 'zien'	storingen in het magnetisch veld door metalen objecten
al lang en veel in gebruik	beperkte werkruimte
goed ontwikkelde technologie	verhoging jitter naarmate men verder van de zender gaat
redelijk goedkoop	latency
robuust	

Tabel 2.3: Voor- en nadelen van magnetische tracking

2.1.4 Mechanische tracking

Een mechanische tracker maakt gebruik van een relatief vast punt, dus een punt waarvan we de coördinaten al kennen. Van hieruit wordt een mechanische verbinding gemaakt naar de punten die men wil tracken. Deze verbinding moet niet rechtstreeks zijn, men kan ook van een punt vertrekken dat men aan het tracken is en van daaruit een verbinding maken met het volgende te tracken punt. De rotatie en afstand tussen twee punten kan men bepalen door bijvoorbeeld tandwielen, potentiometers (zie versnellingsmeters pagina 10) of buig sensors.
In de praktijk is er een zeer grote variatie in mechanische trackers. Deze kunnen gaan van een volledig pak om bijvoorbeeld lichaamsbewegingen te volgen tot een handschoen die de vinger- en handbewegingen registreert tot mechanische armen die bijvoorbeeld een pen volgt in een 3D-ruimte. Deze systemen kan men ruwweg in twee groepen verdelen: de grond en de lichaam gebaseerde trackers. De grond gebaseerde trackers hebben hun vast punt in de wereld (bijvoorbeeld op de grond of op het plafond). Hierdoor is de bewegingsvrijheid van de gebruiker beperkt tot de lengte van de verbinding. In tegenstelling tot de lichaam gebaseerde trackers die hun vast punt op het lichaam zelf hebben. Dit geeft hen geen bewegingsbeperking, maar ze kunnen de gebruiker wel hinderen in zijn bewegingen doordat de apparaten meestal zwaar en/of log zijn.

Voor- en nadelen

Voordelen	Nadelen
grote nauwkeurigheid	relatieve coördinaten t.o.v. een vast punt
zeer hoge update snelheid	draagbare apparaten kunnen zwaar en/of log zijn
geen zichtbaarheid nodig en geen last van metalen objecten	exoskelet kan ongemakkelijk zitten waardoor niet zo vrij bewogen kan worden
wanneer draagbaar, geen beperking op de werkruimte	wanneer niet draagbaar, grote beperking op bewegingsvrijheid
mogelijkheid voor force feedback	

Tabel 2.4: Voor- en nadelen van mechanische tracking

2.1.5 Optische tracking

Optische trackers maken gebruik van licht dat opgevangen wordt door een lichtdetector. Elk punt op de detector stemt overeen met een straal door dit punt en het projectiecentrum waarop de lichtbron ligt.

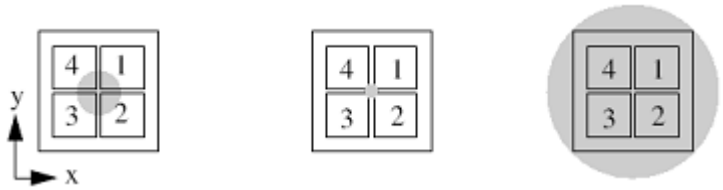
Markeringen

De meeste optische systemen werken met markeringen in de omgeving. Deze kunnen zowel actief als passief zijn, waarbij we actieve markeringen beschouwen als markeringen die door het systeem kunnen bediend worden (bijvoorbeeld aan/uitzetten van lampjes).
Markeringen kunnen zeer uiteenlopende vormen aannemen. Dit kunnen bijvoorbeeld LED's (Light-Emitting Diodes) zijn of infrarode LED's als men zo weinig mogelijk storing van de omgeving wil hebben. Maar dit kunnen even goed gewoon gekleurde stickertjes zijn of goniometrische figuren. Het is duidelijk dat deze laatste markeringen passief zullen zijn. LED's en iLED's kunnen zowel actief als passief zijn. Door bijvoorbeeld ze om een bepaalde tijd aan en uit te doen kan men extra informatie bekomen.
Sommige optische systemen gebruiken geen artificiële markeringen, maar proberen informatie uit de natuurlijke wereld te gebruiken om de positie en oriëntatie te berekenen. Dit kan men doen door bijvoorbeeld vaste punten (zoals hoekpunten) in opeenvolgende beelden te zoeken en hun beweging in het beeld na te gaan.

Detectors

Om de markeringen te registreren heeft een optisch systeem natuurlijk een camera nodig. Dit kan een gewone video of CCD camera zijn of men kan gebruik maken van lichtgevoelige diodes, genaamd fotodiodes. Deze geven een voltage evenredig aan de lichtsterkte die er op valt. Als men een silicium fotodiode neemt kan men infrarood licht opmeten. Het voordeel van de fotodiodes is de verwerkingssnelheid doordat aan de hand van het voltage in een aantal fotodiodes kan men de positie van de invallende lichtstraal berekenen (zie 'QuadCells'). Bij CCD camera's zijn er nog extra beeldverwerkingstechnieken nodig.

Quad cells Quad cells bestaan uit 4 fotodiodes. Als er dus een lichtstraal op de quad cell valt, wordt er in elk van deze diodes een stroom opgewekt naargelang de intensiteit in elke cel. In Figuur (2.2) zien we dat een straal precies in het midden van de quad cell valt. We krijgen dan in elk van de 4 cellen een even groot voltage. Wanneer de straal te klein is zodat hij



Figuur 2.2: Een quad cell. links: de lichtstraal valt precies in het midden van de quad cell. midden: de lichtstraal is te dun om een fotodiode te raken. rechts: de lichtstraal is te groot, er is maximum belichting in alle 4 de cellen

tussen de cellen in valt of te groot is zodat hij alle cellen verlicht, kan er uit de metingen geen informatie gehaald worden i.v.m. de positie. Anders kan men de x- en y-verplaatsing ten opzichte van het middelpunt van de quad cell berekenen aan de hand van:

$$dx = \frac{(i_1 + i_2) - (i_3 + i_4)}{i_1 + i_2 + i_3 + i_4}$$

$$dy = \frac{(i_1 + i_4) - (i_2 + i_3)}{i_1 + i_2 + i_3 + i_4}$$

CCD's (Charge Coupled Devices) Een CCD is een collectie van lichtgevoelige cellen. Dit kan 1D of 2D zijn naargelang wat men wil tracken. Wanneer er op een CCD cel licht valt komen er elektronen vrij. Elke cel houdt bij hoeveel elektronen er gemaakt zijn gedurende een belichtingsperiode. Aan het einde van zo een periode wordt de hoeveelheid elektronen uit elke cel gelezen als een voltage die evenredig is aan de lichtintensiteit. Dit zijn de pixelwaarden van het gegenereerde digitale beeld.
Net zoals bij een gewoon fototoestel moeten we de belichtingsperiode groot genoeg nemen om voldoende licht op te nemen om een beeld te vormen. We mogen deze ook niet te groot nemen omdat er dan een te grote vertraging ontstaat. De grootte van de belichtingsperiode bepaalt dus mee de update rate van de camera.

Technieken

Wanneer men met fotodiodes werkt kan men moeilijk andere informatie gebruiken dan enkel de positie van de invallende lichtstralen. Werkt men met een gewone video camera of een CCD camera dan kan men ook vormen en symbolen herkennen in het beeld voor extra informatie. Dit kunnen gewoon goniometrische figuren zijn waarvan men de vervormingen onder projectie in rekening brengt om oriëntatie en positie te bepalen.
Third Eye Interfaces[4] heeft pas een nieuwe techniek ontwikkeld gebruik makend van microlensdisplays. Iedereen zal deze wel kennen als een speelgoedkaartje waar een andere figuur op staat als je het onder een andere hoek bekijkt. Als men onder elke hoek een totaal verschillende figuur zet kan men met patroon-herkenningssoftware zeer gemakkelijk achterhalen onder welke hoek het object gehouden wordt.

[4]http://www.thirdeyeinterfaces.com

Voor- en nadelen

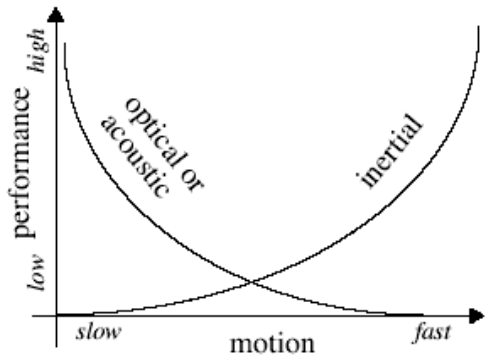
Voordelen	Nadelen
lage kost	occlusieproblemen
hoge beschikbaarheid van componenten (camera, snelle CPU/GPU, ...)	heeft een speciaal ontworpen omgeving nodig (markeringen, weinig storing van andere visuele objecten, ...)
geen magneten en geen last van metalen	nauwkeurigheid (grotere precisie is meestal duur)
eenvoudig te maken en aan te passen	
snelheid	

Tabel 2.5: Voor- en nadelen van optische tracking

2.1.6 Hybride systemen

Hierboven hebben we gezien dat elk fysiek medium wel zijn voor- en nadelen heeft. Daar komt nog bij dat er ook beperkingen zijn op de meetsystemen en dat er beperkingen zijn gerelateerd aan de bewegingen (snelheid, versnelling, ...) van het object in sommige applicaties. Gelukkig hebben niet alle systemen dezelfde beperkingen en kunnen we dus meerdere technieken tegelijk gebruiken om elkaars nadelen te compenseren. Een veel gebruikte combinatie is inertiële-optische tracking (Figuur 2.3).

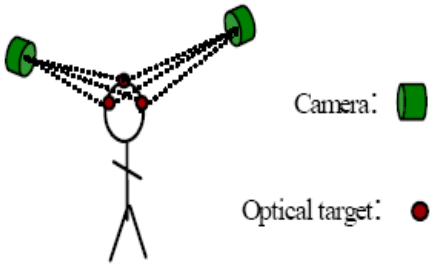
Het grote probleem bij inertiële tracking is de drift tijdens periodes van trage verplaatsing. De tracker geeft dan een kleine verplaatsing aan terwijl dit niet het geval is en de berekende positie 'drijft' dan stilletjes weg van de echt positie. De optische tracker kan dit probleem oplossen want deze heeft een goede nauwkeurigheid bij een trage verplaatsing. Als de inertiële tracker dan toch een kleine verplaatsing aangeeft en de optische ziet het doel niet verschuiven wordt de verschuiving niet in rekening gebracht.
Aan de andere kant is een optisch systeem niet zo goed om grote snelheden en dus verplaatsingen op te meten. Dit komt bijvoorbeeld doordat het licht uitgesmeerd wordt in de richting van de beweging of de sensor kan een referentiepunt uit het vorig beeld niet meer terugvinden in het huidige beeld. Hier kan de inertiële tracker weer een handje toesteken om tot het juiste resultaat te komen.



Figuur 2.3: Vergelijking performantie van inertiële en optische tracking

2.1.7 'Inside looking out' tegen 'Outside looking in'

Er zijn 2 grote groepen van opstellingen van een trackingsysteem, nl. 'Outside looking in' en 'Inside looking out'. Men zou deze 2 technieken ook tegelijk kunnen gebruiken om nog nauwkeuriger te werken en de negatieve effecten van beide technieken te compenseren.



Figuur 2.4: Outside looking in.

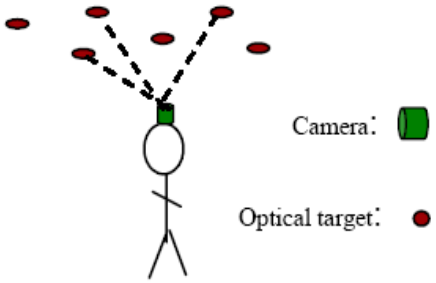
Outside looking In

Hierbij staan de sensors vast in de scène. Ze volgen 1 of meerdere doelen die door de scène bewegen. Hierdoor is de werkruimte beperkt tot het zichtsveld van de sensors. Een ander nadeel is dat men meerdere sensors moet hebben die meestal heel wat duurder zijn dan de doelen. Doelen zijn meestal wel kleiner en makkelijker te bevestigen op een object waardoor het voor de

gebruiker makkelijker is. Men heeft ook altijd maar een constante hoeveelheid beelden te verwerken als er andere doelen in de scène komen omdat 1 sensor meerdere doelen tegelijk kan tracken.

Inside looking Out

Dit is net het omgekeerde als de vorige opstelling. Men bevestigt nu de sensor op het object en bevestigt de doelen vast in de scène. Dit heeft natuurlijk als voordeel dat men de scène makkelijk kan uitbreiden door meer doelen te bevestigen. Deze zijn niet duur en de sensor heeft nog altijd hetzelfde aantal beelden te verwerken waar telkens gewoon meer doelen in 1 beeld zitten. Men heeft wel het nadeel dat men de meestal zwaardere en grotere sensor op het te tracken object moet bevestigen. Ook de beeldverwerking moet hier gebeuren of het object moet verbonden worden met een centrale computer. Dit beperkt dan natuurlijk wel de werkruimte van het object.



Figuur 2.5: Inside looking out.

2.1.8 Overzicht commerciële toepassingen

We geven hier een kort overzicht van commerciële trackingsystemen die nu op de markt zijn:

Product	Fysiek medium	Update snelheid	Latency	Nauwkeurig- (mm)	heid (graden)	Kostprijs (EURO)
Intersense IS-900 Mark 2	akoestisch	180 Hz	4 - 10 ms	4 mm	$0,1°$	8150 - 38.245
Xsens MT6	inertiëel	512 Hz	2 ms	-	$1°$	$1.750 + 1.990$
InertiaCube3 (enkel rotatie)	inertiëel	180 Hz	2 ms	-	$1°$	1.625
Fastrak	magnetisch	120 Hz	4 ms	0,75 mm	$0,15°$	6.020
MotionStar Wired - 6 Sensors, Extended Range Transmitter	magnetisch	144 Hz	4 ms	7,6 - 15 mm	$0,5° - 1°$	21.435
Fakespace Systems BOOM 3C (+ HMD)	mechanisch	$>$ 70 Hz	200 ms	3 mm	$0,1°$	77.350
Shooting Star Technology: ADL-1 head tracker	mechanisch	300 Hz	$<$ 2 ms	5 mm	$0,3°$	1.200
Vicon Peak - Vicon M3 (infrarood camera)	optisch	240 Hz	-	-	-	71.070
3rdTech HiBall 3100	optisch	2000 Hz	$<$ 1 ms	0,4 mm	$0,02°$	vanaf 22.800

Figuur 2.6: Vergelijking verschillende commerciële trackingsystemen

2.2 Motivatie waarom we voor 'inside looking out' optische tracking gekozen hebben

De bedoeling van deze thesis is om een lage kost tracker te bouwen steunend op een nieuwe computervisie techniek. We zien dat de componenten in grote mate overal beschikbaar zijn en tegen een redelijk lage kostprijs. We denken hier vooral aan digitale camera's/webcams, krachtige CPU's en GPU's, ...). We hebben voor de 'Inside looking out' tracking gekozen omdat dit ideaal is voor VR toepassingen. Door dit systeem kunnen we de opstelling eenvoudig uitbreiden en kunnen er meerdere te tracken objecten tegelijk in de opstelling staan.
Nadelen van het gebruik van optische systemen is dat het niet zo nauwkeurig is (vooral door goedkopere camera's), dat we een plaats moeten voorzien om de markeringen aan te brengen en dat deze markeringen zichtbaar moeten zijn in onze werkruimte. Deze nadelen zijn aanvaardbaar voor onze toepassingen. We moeten bijvoorbeeld wanneer we door een wereld lopen niet onze oriëntatie tot op $0,01°$ kennen. Belangrijker voor ons systeem is een lage

jitter en latency. De Line of Sight problemen komen ook niet zo veel voor omdat we in een afgesloten 'VR cave' werken waardoor er maar weinig storing is van andere objecten.

2.3 Camera calibratie

Wanneer we een inside-looking-out optisch trackingsysteem beschouwen trachten we de positie en oriëntatie van de camera te berekenen. Dit komt overeen met het camera calibratie probleem waar we trachten, gegeven 1 of meerdere beelden, de camera parameters te bepalen. Positie en oriëntatie zijn hier de extrinsieke parameters van.

2.3.1 Algemene projectievergelijking

Het doel van camera calibratie is het bepalen van de projectie tussen de 3D wereldcoördinaten en de 2D beeldcoördinaten. Het camera model dat we in deze toepassingen meestal gebruiken is de 'pinhole' camera. Deze heeft een perfecte perspectieve projectie die we kunnen schrijven als een 3×4 projectiematrix P, waarbij

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

Hierbij zijn $[\lambda u, \lambda v, \lambda]$ de beeldcoördinaten met $\lambda \neq 0$ een schaalfactor en $[X, Y, Z, 1]$ de homogene wereldcoördinaten van het geprojecteerde punt. De projectiematrix kan opgesplitst worden in een $3x3$ rotatiematrix R, een translatievector T en een calibratiematrix K als volgt

$$P = K[R|T] \tag{2.2}$$

met

$$K = \begin{bmatrix} \frac{f}{\alpha} & s & u_0 \\ 0 & \frac{f}{\alpha} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

met α gelijk aan de pixelgrootte bij vierkante pixels en s de skew factor die we 0 veronderstellen. De variabelen in de calibratiematrix K stellen

de intrinsieke parameters van het camera model voor. Deze zijn meestal constant in de tijd en hangen enkel af van de fysieke opbouw van de camera. De transformatie die deze camera ondergaan heeft kan opgedeeld worden in een rotatie R en een translatie T die elk 3 vrijheidsgraden hebben. Deze 6 vrijheidsgraden noemen we de extrinsieke parameters van de camera.
Wanneer we nu vergelijkingen (2.1) en (2.2) samenvoegen krijgen we:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = KX' \tag{2.4}$$

Hierbij zijn de coëfficiënten van X' de homogene coördinaten van het punt in het camera-assenstelsel en zijn $[u, v, 1]$ de homogene beeldcoördinaten van dit punt. Dus K is de relatie tussen de homogene beeld- en cameracoördinaten. Het inverse van de calibratiematrix K is

$$K^{-1} = \begin{bmatrix} \frac{\alpha}{f} & 0 & -\frac{\alpha u_0}{f} \\ 0 & \frac{\alpha}{f} & -\frac{\alpha v_0}{f} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Deze zet de homogene beeldcoördinaten terug om naar homogene coördinaten op een schaalfactor λ na.

De meeste camera calibratie methodes veronderstellen de intrinsieke constantes gekend en bepalen enkel de extrinsieke parameters. Nochtans kunnen de intrinsieke parameters ook bepaald worden door puntovereenkomsten zoals bijvoorbeeld in [7] waar men gebruik maakt van de Kruppa's Equations [14]. In ons algoritme gaan we er eveneens vanuit dat de intrinsieke waardes gekend zijn, daarom gaan we in deze thesis hier niet verder op in.
In de volgende secties zullen we trachten de camera calibratie methodes te classificeren op basis van:

- Kenmerk gebaseerde, optical flow gebaseerde en directe methodes
- Continue en discrete tijd algoritmes
- Incrementele methodes
- Opdelen van het camera calibratie probleem

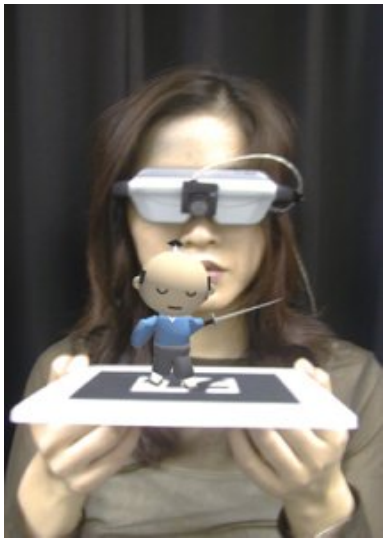
2.3.2 Kenmerk gebaseerde, optical flow gebaseerde en directe methodes

Camera calibratie maakt gebruik van de invoerbeelden die het krijgt van de camera. Hoe deze beelden geïnterpreteerd worden, kan heel verschillend zijn in verschillende systemen.

Kenmerk gebaseerde methodes

Een eerste klasse methodes maakt gebruik van het feit dat bepaalde kenmerken van het eerste beeld overeenkomen met kenmerken in het tweede beeld. Dit kunnen bijvoorbeeld artificiële markeringen zijn die op een vooraf gedefiniëerde manier geplaatst zijn. Andere kenmerken zijn bijvoorbeeld hoekpunten van natuurlijke objecten die zich in de scène bevinden. Deze kenmerken worden dan in een volgend beeld gezocht. Elk van deze puntovereenkomsten heeft dan een vergelijking zoals (2.1). Wanneer we dus voldoende puntovereenkomsten hebben kunnen we de transformatie van het ene camera beeld naar het andere bepalen.
Een algoritme dat gebruik maakt van artificiële markeringen is het HiBall [20] systeem. Dit is een zeer robuust en accuraat optisch tracking systeem (maar ook zeer duur) dat gebruik maakt van een inside-looking-out optische sensor. Deze bepaalt de positie van elk sequentiëel opgelichte infrarode LED die zich in het plafond bevindt. Al deze positie gegevens worden door het SCAAT [22] algoritme omgezet tot een robuuste positie en oriëntatie bepaling.
Kenmerk gebaseerde tracking wordt ook vaak vaak in augmented reality toegepast. Een voorbeeld is de ARToolkit[5] waar een artificiëel patroon getrackt wordt om er in het verrijkte beeld een object op te zetten (Figuur 2.7).
In [16] zien we een voorbeeld van een kenmerk gebaseerd tracking systeem dat geen gebruik maakt van artificiële markeringen in de scène. Er wordt naar hoekpunten in de beelden gezocht die het algoritme dan probeert terug te vinden in het volgende frame. Deze techniek is geschikt voor toepassingen waar men onbekend terrein gaat verkennen. Daarom dat het ook ideaal is voor robot navigatie.
In ons algoritme gaan we zoals het HiBall systeem gebruik maken van een

[5] http://www.hitl.washington.edu/artoolkit/



Figuur 2.7: Kenmerk gebaseerde tracking gebruikt in ARToolkit.

inside-looking-out opstelling waarbij we markeringen in lijnpatronen plaatsen.

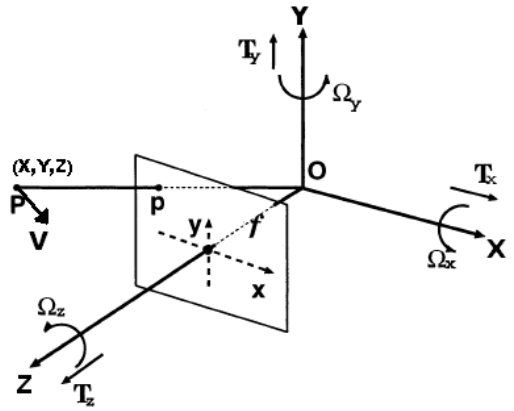
Optical flow gebaseerde methodes

In tegenstelling tot de vorige techniek gaan we niet op zoek naar kenmerken. We bepalen de optical flow van een beeldsequentie. Verschillende methodes om de optical flow te bepalen met hun performatie kan men vinden in [2].
In [6], [12] en [4] wordt de 3D motion berekend uit optical flow met de veronderstelling dat we een statische scène aan het filmen zijn. In dit geval geldt de rigid motion (Figuur 2.8) vergelijking:

$$\vec{V} = \left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}\right)^T = -(\vec{\Omega} \times \vec{P} + \vec{T}) \tag{2.6}$$

Hierbij is $\vec{P}$ een punt in de camera ruimte met $\vec{V}$ zijn snelheid onder de rotatie $\vec{\Omega} = (\Omega_x, \Omega_y, \Omega_z)$ en de translatie $\vec{T} = (T_x, T_y, T_z)$ van de camera. In [6] wordt dan de link gelegd tussen rigid motion $(\vec{\Omega}, \vec{T})$ en de optical flow $\vec{\theta}(x, y)$ (verplaatsing van de pixel (x, y) in het geprojecteerde beeld) als volgt:

$$\vec{\theta}(x, y) = \rho(x, y)A(x, y)\vec{T} + B(x, y)\vec{\Omega} \tag{2.7}$$



Figuur 2.8: Het (X, Y, Z) coördinatensysteem van de camera. P is een punt in de lokale cameraruimte met p zijn projectie. $\vec{T} = (T_x, T_y, T_z)$ en $\vec{\Omega} = (\Omega_x, \Omega_y, \Omega_z)$ de translatie en rotatie van de camera en $\vec{V}$ de snelheid van P onder deze beweging.

met

$$A(x,y) = \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \tag{2.8}$$

$$B(x,y) = \begin{bmatrix} \frac{xy}{f} & -\frac{f+x^2}{f} & y \\ \frac{f+y^2}{f} & -\frac{xy}{f} & -x \end{bmatrix} \tag{2.9}$$

Hierbij is $\rho(x, y)$ ook de diepte (Z-waarde) op image locatie (x, y). Door vergelijking (2.7) met meerdere motion vectoren $\vec{\theta}(x, y)$ op te lossen naar $\vec{T}$ en $\vec{\Omega}$ bekomt men de relatieve transformatie t.o.v. de vorige frame.

Directe methodes

Horn stelde in o.a. [13] verschillende 'directe' methodes voor om de 3D motion parameters terug te vinden. In deze methodes wordt er geen gebruik gemaakt van puntovereenkomsten of schattingen van optical flow, maar van spatiotemporele gradients van de lichtintensiteit van het beeld. In de paper geeft hij enkel een oplossing voor 3D motion in een statische scène voor een aantal speciale gevallen, nl.

- Pure rotatie

- Pure translatie
- Willekeurige bewegingen met een gekende rotatie

In [12] geeft men ook een directe versie van hun optical flow algoritme, gezien in de vorige sectie. Bij deze methode werden geen restricties geplaatst op de bewegingen.

2.3.3 Continue en discrete tijd algoritmes

Technieken om 3D motion te berekenen kunnen ook opgesplitst worden in continue-tijd en discrete-tijd methodes. Discrete-time algoritmes volgen een aantal kenmerken over een aantal beelden. Gegeven genoeg viewpoints en genoeg overeenkomstige punten, dan is de oplossing van het rigid motion probleem overbepaald en kan deze dus op een robuuste manier bekomen worden door bijvoorbeeld een Least Squares oplossing. In [8] stelt men zo een methode voor om door puntovereenkomsten tussen 2 frames de motion parameters te bepalen.
Continue-tijd algoritmes, ook wel *Instantaneous Time* methodes genoemd, berekenen 3D motion en diepte uit bijvoorbeeld optical flow zoals in 2.3.2. De optical flow kan bijvoorbeeld bepaald worden met correlatie of blok-matching waarbij elke blok vergeleken wordt met dichtbij gelegen blokken. Kenmerken bepalen en terugzoeken is een andere manier om aan de optical flow te komen. Kenmerken worden ook in discrete-tijd methodes gebruikt zoals we hierboven gezien hebben, maar in dit geval gebruiken we niet de positie van de kenmerken, maar zijn snelheid. Heeger en Jepson stellen zo een continue-tijd methode voor in [12] d.m.v. optical flow.
Het echte verschil tussen discrete- en continue-tijd methodes is dat discrete-tijd methodes gebruik maken van de verplaatsing van een kenmerk tussen 2 frames, terwijl de continue-tijd methodes de ogenblikkelijke snelheid van een kenmerk in dit frame proberen te schatten. Dit betekent ook dat bij continue-methodes de veranderingen klein moeten zijn in functie van de tijd om een goede benadering te hebben.

2.3.4 Incrementele methodes

Andere mensen hebben incrementele methodes voorgesteld om 3D motion te bekomen door gebruik te maken van grote hoeveelheden gegevens over een bepaalde tijd. Het doel van deze methodes is om de error op de uitkomst stapsgewijs te verkleinen. Ullman [19] stelt een discrete-tijd, kenmerk gebaseerde, incrementele methode voor. Het voordeel van deze methode is dat deze ook kan omgaan met nonrigid objecten die traag veranderen in de tijd. In [3] is een kenmerk gebaseerd camera calibratie algoritme dat gebruik maakt van de uitgebreide Kalman filter. Eerst wordt er initiële depth map gekozen waar de 3D motion parameters uit berekend worden. De depth map wordt dan aangepast aan de gevonden motion parameters. Dit proces wordt herhaald totdat de motion parameters convergeren naar een juiste oplossing.
De uitgebreide Kalman filter wordt ook in het SCAAT [22] algoritme gebruikt in de HiBall tracker [20]. Dit algoritme wordt gebruikt om een constante waarde (of een waarde constant veranderend in de tijd) te schatten uit noisy invoerdata. Het algoritme bestaat uit 2 grote stappen, nl. de Time Update ('Predict') en de Measurement Update ('Correct'). Het voordeel aan deze methode is dat men niet alle invoerdata op voorhand moet hebben om de waarde te schatten. Hoe meer invoerdata de filter krijgt, hoe nauwkeuriger de schatting wordt. Voor meer informatie over de Kalman filter verwijzen we u naar [21].
Een nadeel aan incrementele methodes is dat deze meestal een goede initiële gok nodig hebben om efficiënt te kunnen werken en dat over het algemeen meerdere iteraties doen een grotere rekenkost met zich meebrengt. Het voordeel is wel dat deze methodes meestal zeer robuust zijn in 'noisy' data.

2.3.5 Opdelen van het camera calibratie probleem

De meeste camera calibratie technieken proberen niet alle camera parameters tegelijk te schatten. Men maakt gebruik van projectieve eigenschappen die onafhankelijk zijn van een aantal parameters. Zo kunnen we m.b.v. de eigenschappen een aantal parameters onafhankelijk van anderen berekend worden. Een voorbeeld van zo een eigenschap is een vluchtpunt [5]. Een vluchtpunt is het snijpunt op oneindig van evenwijdige lijnen. Bij een projectie kan dit punt op oneindig geprojecteerd worden op een punt in het beeld. Denk maar aan een foto van een (evenwijdig) treinspoor dat naar 1 punt lijkt te gaan, het vluchtpunt. Omdat dit punt in de wereld op oneindig ligt is het dus

onafhankelijk van de translatie die de camera ondergaat. In onze implementatie hebben we ook van deze eigenschap gebruik gemaakt. Zo zullen we in hoofdstuk 6 zien hoe we uit de positie van 2 vluchtpunten de rotatiematrix kunnen bepalen onafhankelijk van de translatie.
In [6] proberen ze eerst de translatie-richting te schatten, onafhankelijk van de diepte en rotatie. Dit doen ze door snelheidsmetingen in 5 punten (m.b.v. optical flow) te bundelen in een restfunctie. Deze restfunctie berekent een waarde per translatie-richting hoe goed deze translatie overeenkomt met de gemeten optical flow. De richting met de kleinste waarde is dus de gezochte translatie. Daarna berekenen ze de rotatie onafhankelijk van de diepte uit de translatie en uiteindelijk de diepte en de lengte van de translatie.
De parameters van de camera calibratie worden meestal appart berekend omdat deze vergelijkingen eenvoudiger zijn dan alle parameters tegelijk berekenen uit de algemene projectievergelijking (2.1). Toch moet er ook opgelet worden voor meetfouten in een parameter omdat deze meestal fouten vergroten in andere parameters die uit deze parameter berekend worden.

2.3.6 Evaluatie van camera calibratie methodes

Bij het evalueren van camera calibratie methodes zijn er een aantal criteria belangrijk:

- Stabiliteit
  De camera calibratie mag niet te gevoelig zijn voor meetfouten. Best wordt er gebruik gemaakt van heel wat redundante data. Methodes die slechts lokaal werken zijn ook redelijk kwetsbaar voor fouten. Ook de invloed van onvermijdelijke ruisdata moet tot een minimum herleid worden door bijvoorbeeld veel echte data te gebruiken zodat enkele ruispunten relatief weinig invloed hebben.

- Algemeenheid
  Sommige algoritmes kunnen het calibratie probleem slechts gedeeltelijk oplossen of enkel onder speciale omstandigheden. Bijvoorbeeld het gebruik van artificiële markeringen beperkt het gebruik ervan tot gebouwde opstellingen. Andere algoritmes kunnen misschien dan weer niet goed tegen andere bewegende objecten in de scène. Er zijn ook methodes die enkel de rotatie kunnen vinden, terwijl anderen slechts de translatie kunnen vinden indien de rotatie en/of diepte gekend is.

- Efficiëntie
  Veel camera calibratiesystemen werken zeer goed, zijn zeer robuust en kunnen in heel wat omgevingen werken, maar moeten door hun complexiteit off-line uitgevoerd worden. In sommige toepassingen is dit geen probleem, maar in deze thesis trachten we een real-time tracker te bouwen. Daarom zijn niet alle camera calibratietechnieken geschikt om voor tracking te gebruiken. In de meeste gevallen zijn iteratieve methodes in hoog dimensionele ruimtes redelijk traag en vragen ze meestal voor een goede initiële gok. Aan de andere kant zijn parallelle systemen veel efficiënter omdat deze indien nodig over meerdere pc's verspreid kunnen worden.

- Uitbreidbaarheid In een trackingsysteem is het soms belangrijk om makkelijk uitgebreid te kunnen worden. Wanneer men bijvoorbeeld met een vast calibratie object werkt kan men hier niet zo ver vandaan gaan. Indien we gebruik maken van een kenmerk gebaseerde methode is het voor de uitbreidbaarheid het best dat deze in een herhalend patroon bevestigd worden. Hierdoor kan men relatieve verplaatsingen berekenen, maar heeft men geen absolute houvast meer (buiten bijvoorbeeld enkele vaste calibratie patronen).
  Zoals we later zullen zien hebben wij een zeer makkelijk uit te breiden systeem gebouwd dat toch een bijna absolute rotatie behoudt.

2.4 Motivatie trackingalgoritmes

In sectie 2.2 hebben we reeds onze keuze voor een Inside-looking-out optisch trackingssysteem gemotiveerd. Hier zullen we onze gekozen algoritmes (zie §3.2) motiveren en classificeren.
Eerst en vooral hebben we gekozen voor een kenmerk gebaseerd methode te gebruiken met artificiële markeringen. Deze markeringen kunnen we goed herkennen en dit zorgt er tegelijk voor dat er minder ruis op de invoerdata zit. Het beperkt aantal markeringen zorgt er ook voor dat de verwerking van deze data efficiënter kan gebeuren dan bij vele mogelijke kenmerken. De markeringen geven ons ook een houvast in de wereld zodat er betere absolute waarden kunnen bekomen worden. Nadeel van de artificiële markeringen is natuurijk dat deze geplaatst moeten worden, maar dit valt wel mee omdat we toch met een afgesloten VR omgeving werken.

Ons algoritme werkt ook in discrete tijdsstappen. We maken enkel gebruik van puntovereenkomsten tussen opeenvolgende frames, geen snelheden van de markeringen.
Onze methode werkt niet incrementeel, alles wordt in 1 iteratie berekend. Eventueel zouden we in de toekomst op zoek kunnen gaan naar een methode om uit de gevonden translatie, de rotatie te verbeteren. Dit kan dan recursief een aantal iteraties uitgevoerd worden zodat we het resultaat incrementeel verbeteren.
Ten slotte zullen we het camera calibratie probleem ook opdelen in het bepalen van de rotatie en het berekenen van de ondergane translatie. We zullen zoals in [5] de rotatie onafhankelijk van de translatie bepalen d.m.v. vluchtpunten. Daarna zullen we hieruit de translatie bepalen m.b.v. puntovereenkomsten.

Hoofdstuk 3

Overzicht opbouw van de tracker

Zoals we in het voorgaande hoofdstuk gezien hebben is er al heel wat onderzoek gedaan naar camera calibratie met verschillende opstellingen. Niet al deze technieken zijn geschikt om aan tracking te doen. Eerst en vooral moet het tracking algoritme snel (minstens real-time) en automatisch zijn. Wij zullen veronderstellen dat de intrinsieke parameters (focal lenght, pixel grootte, ...) op voorhand gekend zijn. De extrinsieke parameters zullen we berekenen aan de hand van vluchtpunten zoals in [15]. De positie zullen we tracken d.m.v. puntovereenkomsten in opeenvolgende frames te beschouwen, samen met de gegeven afstand tussen 2 opeenvolgende LED's op een lijn.
Eerst en vooral zullen we de opstelling van de scène moeten definiëren om te weten wat de camera voor beelden kan binnen krijgen. In §3.2 geven we dan een overzicht van de stappen die onze optische tracker zal doorlopen bij het verwerken van deze beelden.

3.1 Opstelling scène

In de ruimte waar we een object willen tracken zullen we markeringen plaatsen. Deze kunnen gewone gekleurde plakkertjes of gekleurde LED's zijn. LED's zijn natuurlijk wat duurder, maar zorgen voor een betere segmentatie van de markeringen. We bevestigen markeringen met eenzelfde kleur in evenwijdige lijnen, bijvoorbeeld op het plafond. De afstand tussen 2 lijnen is niet van belang, maar we moeten er wel op letten dat er in elk beeld best

3 lijnen zichtbaar moeten zijn voor een robuuste tracking. 2 lijnen zijn voldoende voor een snijpunt te berekenen, maar dit is ook zeer gevoelig voor kleine veranderingen in de lijnvergelijkingen. We moeten er ook op letten dat de afstand tussen de LED's onderling kleiner is dan de afstand tussen de lijnen. We veronderstellen ook dat de afstand tussen 2 naburige markeringen op een lijn gekend is. Dit is nodig om de translatie te kunnen berekenen. Een tweede kleur van evenwijdige lijnen worden loodrecht op de voorgaande geplaatst. Elke set van gekleurde lijnen zal overeenstemmen met een as in de wereld. Op het plafond zullen dit waarschijnlijk de X- en Z-as zijn. Een derde kleur kan eventueel nog aangebracht worden indien we ook de Y-as willen toevoegen aan de berekeningen.
Op het te tracken object zullen we een gewone CCD camera bevestigen. Deze zal de markeringen in de scène filmen en de beelden naar het trackingsysteem sturen. We veronderstellen dat de intrinsieke parameters op voorhand gekend zijn. Deze waarden zijn normaal constant voor een camera/lens omdat we veronderstellen dat de camera geen variabele zoom, veranderend diafragma of automatische scherpstellingsfunctie heeft.

3.2 Overzicht trackingalgoritme

Het trackingalgoritme krijgt de beelden van de camera als input binnen die dan verwerkt worden om er de extrinsieke parameters uit te halen. Deze verwerking kan opgedeeld worden in een aantal stappen die in de volgende hoofdstukken in detail behandeld worden:

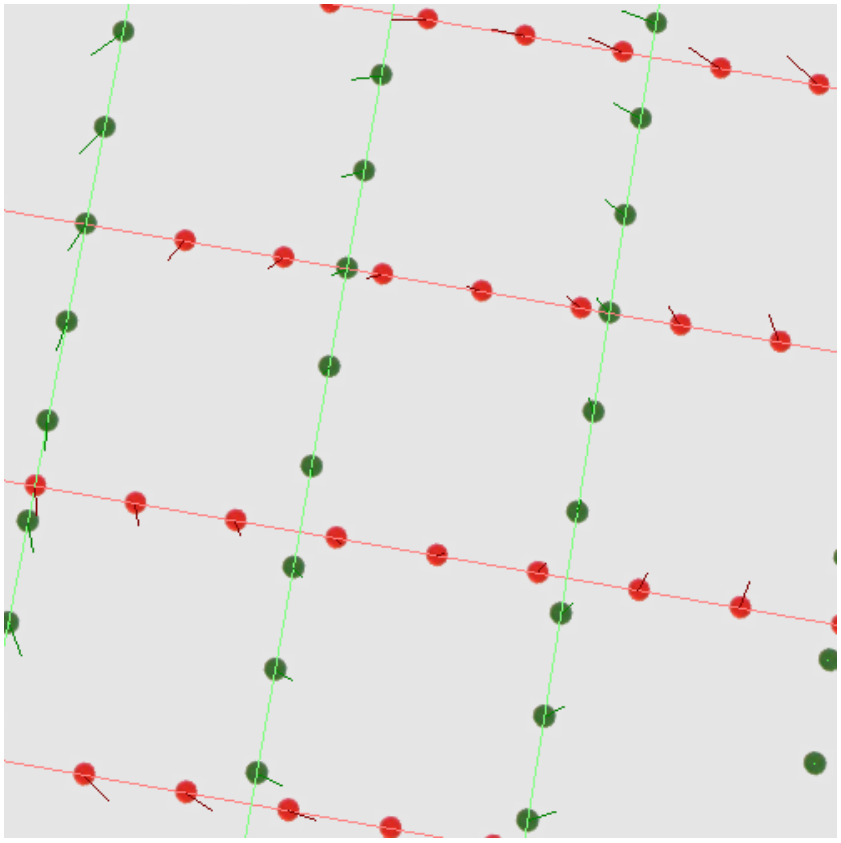
- Hoofdstuk 4: Detectie van markeringen en optical flow
  Hier zullen we de gebruikte technieken bespreken om de geprojecteerde markeringen te herkennen in het beeld alsook de optical flow berekenen.

- Hoofdstuk 5: Herkenning van lijnpatronen
  In de constructie van de scène zien we dat de markeringen lijnen vormen. Om deze lijnpatronen te herkennen, maken we gebruik van het algoritme van Hough dat we voor deze tracker geoptimaliseerd hebben voor real-time gebruik. Ook stellen we hier een nieuwe parametrisatie voor de Hough transformatie voor dat gebruik maakt van cirkels en dat voor kleine sets van punten veel efficiënter werkt.

- Hoofdstuk 6: Berekening van oriëntatie uit vluchtpunten
  In dit hoofdstuk zullen we aantonen hoe we de oriëntatie van de camera kunnen bepalen aan de hand van vluchtpunten.

- Hoofdstuk 7: Bepaling van translatie
  Wanneer we de oriëntatiematrix kennen kunnen we de translatie berekenen a.h.v. overeenkomstige punten in opeenvolgende camera-aanzichten.

Ten slotte evalueren we in hoofdstuk 8 hoe goed ons systeem werkt in een virtuele en een reële testopstelling.



Figuur 3.1: Een verwerkt beeld van de virtuele scène. Lichte puntjes duiden het midden van elke gedetecteerde markering aan. De lichte lijnen dwars door het beeld zijn de gevonden lijnpatronen in het beeld. De donkere strepen bij elke markering stellen de berekende optical flow voor van de markeringen.

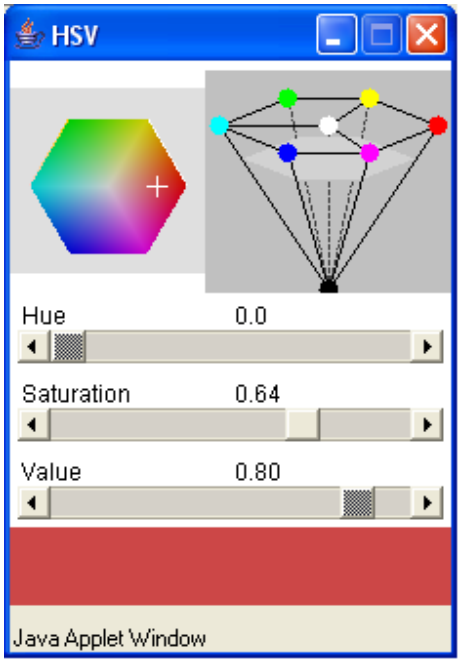
Hoofdstuk 4

Detectie van markeringen en optical flow

De eerste stap in de verwerking van de beelden is natuurlijk het terugvinden van de geprojecteerde markeringen. De pixels die bij de markeringen horen worden teruggevonden aan de hand van hun vooraf bepaalde kleur (§4.1). In §4.2 zullen we deze dan samen voegen zodat enkel de middelpunten worden meegenomen voor verdere verwerking. Voor latere stappen is het ook belangrijk om te weten welke punten overeenkomen in opeenvolgende beelden. Hierbij gebruiken we optical flow (§4.3) om een markering in het volgend beeld terug te vinden. Ten slotte bekijken we in sectie 4.4 ook welk effect lensdistortie op de markeringen heeft.

4.1 Kleurdetectie van markeringen

Het detecteren van een bepaalde markering gebeurt aan de hand van zijn kleur. Hoewel de camera RGB beelden maakt, is het niet zo eenvoudig om een kleur te herkennen aan zijn RGB waarden. Vooral een verandering van lichtintensiteit verandert relatief weinig aan de waargenomen kleurwaarde, maar zijn overeenkomstige RGB waarde verandert wel aanzienlijk. Daarom is het beter om de kleurdetectie in een andere kleurruimte te doen.
Eén van de kleurruimten die wel goed het onderscheid maakt tussen de lichtintensiteit en kleur is HSV. Zoals men kan zien op figuur 4.1 kan men de HSV ruimte voorstellen als een zeshoekige piramide. De kleurwaarde (Hue) kan men voorstellen als de hoek van de kleurpositie rond de verticale as, waarbij



Figuur 4.1: Geometrische interpretatie van de HSV kleurruimte. Deze Java Applet is te vinden op http://www.cs.rit.edu/~ncs/color/a_spaces.html.

rood de waarde 0 heeft. De Saturatie (oftewel verzadiging) geeft aan hoeveel kleur er in de kleurpositie aanwezig is. Dit wordt voorgesteld als de relatieve afstand [0..1] tot de verticale as. De derde waarde ('Value') tenslotte geeft de lichtintensiteit aan en kan men zien als de hoogte van de kleurpositie.
Om te zien of de kleur van een pixel overeenkomt met een van de markeringskleuren moeten we in de HSV kleurruimte dus enkel kijken of de hue binnen een bepaalde drempelwaarde ligt en de saturatie niet te klein is. De lichtintensiteit is van geen belang. We hebben dus enkel nog een conversiefunctie nodig om van RGB naar HSV te gaan. Deze functie gedefiniëerd in [17] gaat als volgt.
We veronderstellen dat min en max respectievelijk de minimum en maximum waarde zijn van de RGB-kleur. Wanneer $min == max$ hebben we een grijswaarde en is de kleurwaarde dus ongedefiniëerd. De verzadiging is dan 0 en de intensiteit van de grijswaarde is gelijk aan max. In het andere geval

is de kleurwaarde:

$$\begin{cases} min == R: & Hue = 3 - \frac{G-B}{max-min} \\ min == G: & Hue = 5 - \frac{B-R}{max-min} \\ min == B: & Hue = 1 - \frac{R-G}{max-min} \end{cases} \tag{4.1}$$

De verzadiging is in dit geval gelijk aan $\frac{max-min}{max}$ terwijl de intensiteit gelijk is aan max.

4.2 Clustering van markeringspixels

In de vorige sectie zagen we hoe we een markeringspixel kunnen identificeren. Het is niet nodig om elke pixel naar de volgende stap te sturen voor lijndetectie. Bovendien zullen er veel ruispixels onterecht als markeringspixel worden beschouwd. Daarom worden markeringspixels gegroepeerd tot clusters, waarbij enkel het middelpunt van een markering is van belang voor lijndetectie.
Het clusteringalgoritme dat we hier gebruikt hebben, zoekt naar een bounding box rond de markering en werkt als volgt:

1. Begin bij een gevonden markeringspixel en definiëer een bounding box rond deze pixel met grootte 1.

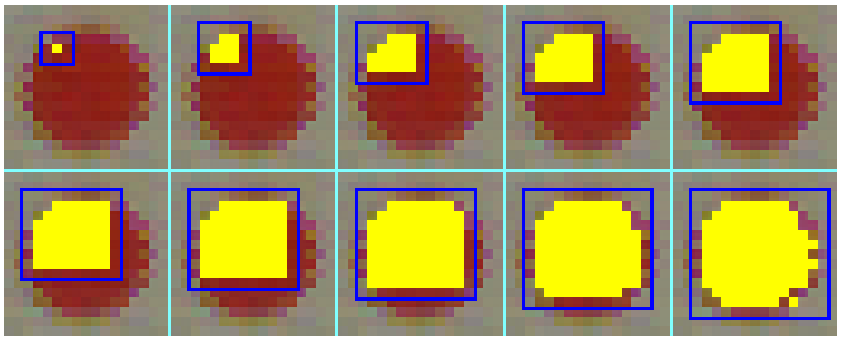
2. Kijk of er aan de bovenste zijde van de box een markeringspixel te vinden is. Indien ja, vergroot de bounding box met 1 naar boven.

3. Doe nu hetzelfde met de linker, rechter en onderste zijde van de bounding box.

4. Zolang de box vergroot is, herhaal vanaf stap 2.

5. Test of er genoeg markeringspunten in de bounding box zitten.
   Indien ja, dan is het gemiddelde van de markeringspixels het midden van de markering. Anders, de pixels behoren waarschijnlijk niet tot een markering, gooi deze ruispixels weg.

De stappen die het clusteralgoritme neemt worden in figuur 4.2 gedemonstreerd op één markering van een invoerbeeld.
Het minimum aantal markeringspunten in een bounding box moet op voorhand opgegeven worden. Buiten testen op dit aantal kan men ook nog testen



Figuur 4.2: Van links boven naar rechts onder worden de iteraties getoond die het clusteralgoritme doorloopt om deze markering te vinden.

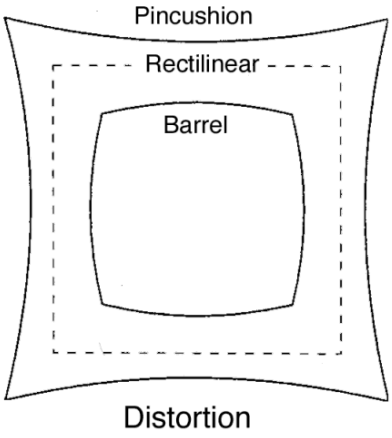
of de vorm ongeveer overeen komt en of de dichtheid van de markeringspixels hoog genoeg is.
Doordat elke markering een minimum aantal pixels in het beeld moet hebben, moeten we niet elke pixel in het beeld nagaan als mogelijke beginpixel voor een cluster. We kunnen het beeld om de X pixels bemonsteren naar markeringspixels, waardoor we het initiëel aantal te onderzoeken pixels delen door X^2. Dit zorgt niet alleen voor een aanzienlijke snelheidswinst, maar zorgt er ook voor dat alleenstaande ruispixels nog minder kans krijgen om gedetecteerd te worden.

4.3 Bepaling van optical flow

In latere stappen is het belangrijk om te weten welke markeringen overeenkomen in opeenvolgende beelden. Daarom moeten we de optical flow van de markeringen bepalen. Dit doen we door voor elke markering niet alleen de positie $P = (x, y)$, maar ook de snelheid $V = (v_x, v_y)$ bij te houden. We voorspellen dan de positie van de markering door $P' = P + V$ en beginnen dan van hieruit te zoeken naar de markering het dichtst gelegen bij P'. De nieuwe snelheid wordt dan gewoon gezet op het verschil tussen nieuwe positie en oude positie.

4.4 Lensdistortie

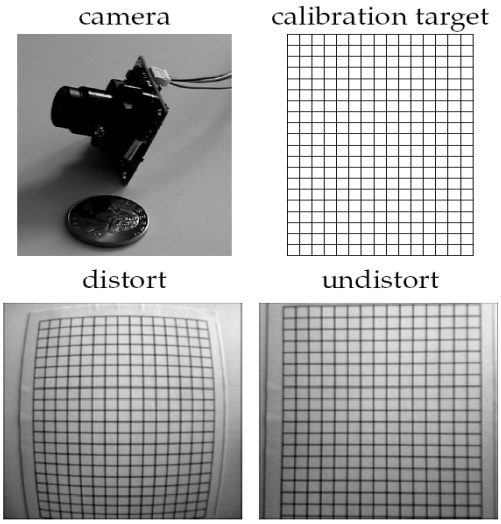
Bij een perfecte gaatjescameramodel worden rechte lijnen in 3D altijd afgebeeld als rechte lijnen in 2D. De meeste cameralenzen zijn niet perfect (zeker bij de goedkopere camera's) en veroorzaken beeldvervormingen. Deze vervormingen ziet men het hardst in de hoeken van het beeld. Er zijn 2 vormen van lensdistorties: Barrel distortie en Pincushion distortie (zie figuur 4.3[1]). Een combinatie van beide kan evenzeer voorkomen.



Figuur 4.3: Barrel en Pincushion distortie.

Een voordeel van lensdistorties is dat ze constant zijn (in afwezigheid van zoom, automatische scherpstelling,...) en enkel afhangen van de camera en meer bepaald de lens. Daarom kunnen we op elk beeld gegenereerd door een bepaalde camera een transformatie toepassen die het beeld terug 'rechttrekken'. Als men dit voor alle pixels in het beeld moet doen kan dit wel een tijdsintensieve operatie worden. In ons geval is het enkel nodig om de coördinaten van de gevonden markeringen te transformeren. In figuur 4.4 zien we een voorbeeld[2] van een sterk vervormd rooster dat terug mooi recht getrokken wordt, zodat niets meer van de vervorming te merken valt.
Het lensdistortiemodel dat wij gebruikt hebben, is gebaseerd op het camera model van Tsai [18]. Hierin toon hij het verband tussen de vervormde

[1] http://www.dvinfo.net/articles/optics/lensdefects.php
[2] http://www.dgp.toronto.edu/~hertzman/courses/csc418/winter_2003/effects/distortion.html



Figuur 4.4: Een door lensdistortie sterk vervormt beeld wordt hersteld.

coördinaten (X_d, Y_d) en de originele coördinaten (X_u, Y_u):

$$X_d + D_x = X_u \tag{4.2}$$
$$Y_d + D_y = Y_u \tag{4.3}$$

met

$$D_x = X_d(k_1 r^2 + k_2 r^4 + \ldots) \tag{4.4}$$
$$D_y = Y_d(k_1 r^2 + k_2 r^4 + \ldots) \tag{4.5}$$
$$r = \sqrt{X_d^2 + Y_d^2} \tag{4.6}$$

Hierbij zijn de parameters k_i de distortie coëfficiënten die we moeten calibreren. Dit doen we door een bekend patroon (een grid) te filmen en dan k_i te schatten tot we het juiste beeld van het patroon terug hebben.

Hoofdstuk 5

Herkenning van lijnpatronen

In de vorige stap hebben we de geprojecteerde posities van de markeringen teruggevonden. In dit hoofdstuk gaan we trachten de lijnpatronen die deze markeringen voorstellen te herkennen. De meest voor de hand liggende methode is natuurlijk de Brute-force methode (§5.1). In de volgende sectie bekijken we dan het algoritme van Hough (§5.2) dat op een intelligentere manier naar lijnen gaat zoeken. In 5.3 stellen we dan een nieuwe Hough transformatie voor die gebruik maakt van cirkels als parametrisatie. Ten slotte vergelijken we deze technieken met elkaar.

5.1 Brute-force methode

De naïeve manier om lijnpatronen te zoeken in een verzameling punten is de brute-force methode:

- Bereken al de lijnen die gevormd kunnen worden tussen 2 markeringen
- Ga na hoeveel punten er op elke lijn liggen
- De lijnen met het hoogste aantal punten erop zijn de gezochte lijnpatronen

Het is duidelijk dat dit geen efficiënte oplossing is. Als er N markeringen gevonden zijn, zijn er daartussen dus $N(N-1)/2$ lijnen. Op elke lijn wordt er dan $(N-2)$ keer getest of een punt op deze lijn ligt. Men ziet onmiddelijk dat de complexiteit van deze methode $O(N^3)$ is.
Dit algoritme kan men eenvoudig versnellen door gebruik te maken van het

feit dat de afstand tussen 2 markeringen op dezelfde lijn altijd kleiner is als de afstand tussen 2 lijnen. Hierdoor beschouwen we enkel de N lijnen gevormd door 2 aanliggende punten en versnellen we het algoritme tot een $O(N^2)$ rekenkost.

5.2 Hough transformatie

Een intelligentere aanpak om lijnen te detecteren is de Hough transformatie [11]. Hough stelt een transformatie van de ruimtelijke coördinaten van een punt naar parameter coördinaten voor, genaamd de Hough transformatie [11]. Het basisidee achter lijndetectie m.b.v. Hough is om de verzameling lijnen te beschouwen die allemaal door een gegeven punt (x_i, y_i) gaan. Deze verzameling lijnen tracht men dan te parametriseren zodat de gezochte lijnen makkelijker te vinden zijn door het snijpunt van de duale parametercurves. Hieronder zullen we een paar veel voorkomende parametrisaties opsommen en zullen we kijken naar de implementatie van het algoritme van Hough en mogelijke optimalisaties.

5.2.1 Parametrisatie door lijnen

Dit is een zeer intuïtieve parametrisatie van elk punt naar een lijn in een duale parameterruimte (a, b). We zullen deze techniek illustreren aan de hand van een voorbeeld.
Door een gegeven punt (x_i, y_i) gaan er oneindig veel lijnen. Elk van deze lijnen kunnen we voorstellen met de vergelijking $y_i = ax_i + b$. Laten we deze vergelijking nu eens omzetten naar de parameterruimte (a, b):

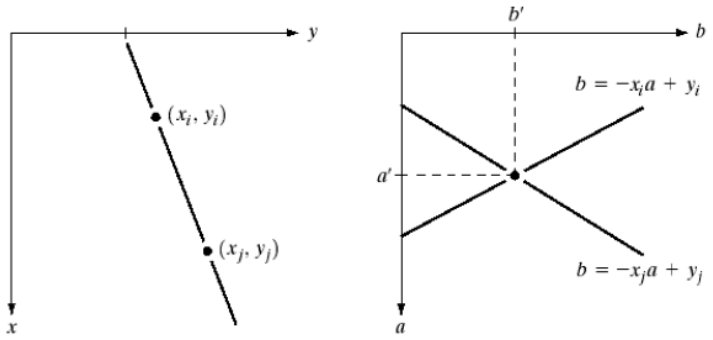
$$b = -x_i a + y_i$$

Merk op dat voor een bepaalde waarde voor a, de waarde van b vast ligt. Hieruit volgt dat elk punt (x_i, y_i) in de duale parameterruimte overeenkomt met een lijn.
Neem nu een 2^e punt (x_j, y_j) met parameter lijn

$$b = -x_j a + y_j$$

en stel nu dat (a', b') het snijpunt is van deze 2 parameter lijnen, dan zien we dat $y = a'x + b'$ de vergelijking is van de rechte door (x_i, y_i) en (x_j, y_j)

(zie Figuur 5.1). Het is ook duidelijk dat hoe meer parameter lijnen er door het snijpunt (a', b') gaan, hoe meer colineaire punten er liggen op de rechte $y = a'x + b'$.



Figuur 5.1: We zien hier de omzetting van wereldcoördinaten naar parameter coördinaten en zien dat de lijn door de 2 punten overeenkomt met het snijpunt van de 2 rechtes in Hough parameterruimte. (Figuur uit [11])

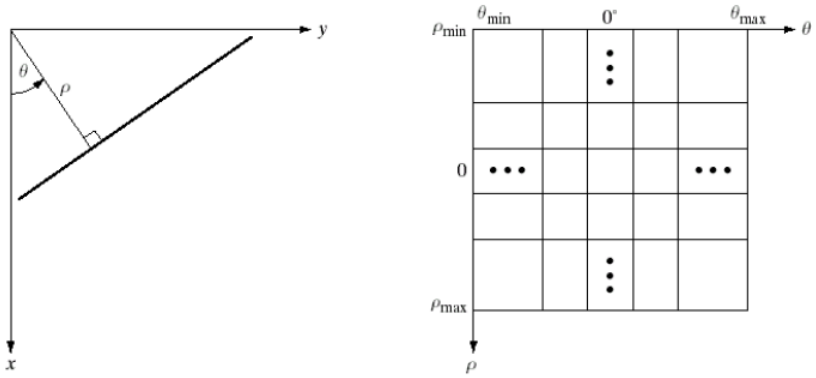
In de praktijk is de omzetting naar parameter ruimte A, B niet vanzelfsprekend omdat rechten evenwijdig met de Y-as niet in a en b uitgedrukt kunnen worden. De grenzen van a en b zijn ook niet strikt bepaald $(a, b \in]-\infty, \infty[)$ waardoor snijpunten ook op oneindig kunnen liggen. Dit zorgt voor moeilijkheden bij de implementatie van het algoritme van Hough zoals we zullen zien in 5.2.3. Daarom wordt er bijna altijd gebruik gemaakt van de volgende parametrisatie.

5.2.2 Parametrisatie door sinus-curves

Een andere veel gebruikte parametrisatie is deze die elke lijn voorstelt door een hoek θ en een afstand ρ zodat

$$x \cos\theta + y \sin\theta = \rho$$

Hierin stellen θ en ρ respectievelijk de hoek tussen de X-as en de loodlijn op de rechte door het nulpunt, en de afstand van het nulpunt tot de rechte voor (zie Figuur 5.2).
In de parameterruimte stelt elk punt nu dus een sinus-vormige curve voor,



Figuur 5.2: Links zien we hoe een rechte in wereldcoördinaten overeenkomt met een punt (θ, ρ) in de parameter ruimte. Aan de rechterzijde zien we hoe we de Hough parameterruimte kunnen opsplitsen in cellen. (Figuur uit [11])

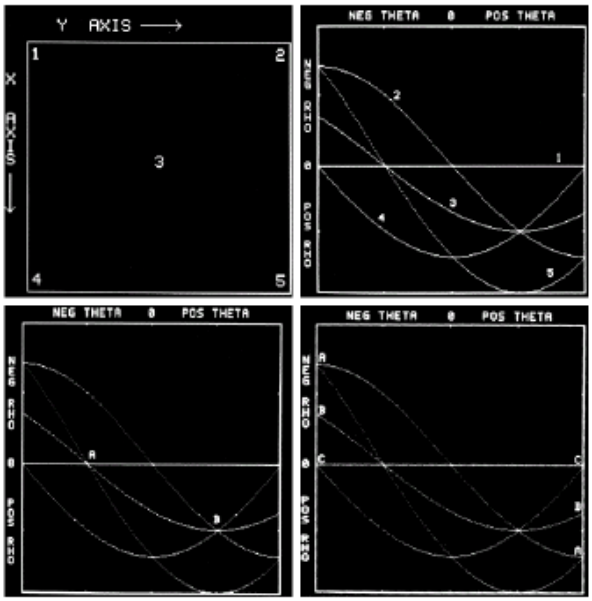
maar de eigenschap dat het snijpunt tussen 2 curves de lijn door de 2 overeenkomstige punten bepaalt, blijft gelden. In tegenstelling tot a en b zijn θ en ρ wel begrensd, nl. $\theta \in [0, \pi[$ en $\rho \in [-\sqrt{2}D, \sqrt{2}D]$ met D de diagonale breedte van het beeld. Deze parametrisatie wordt dus ook meestal gebruikt in de implementatie van het lijndetectie algoritme van Hough .

5.2.3 Lijndetectie algoritme van Hough

Zoals we hierboven gezien hebben is de θ, ρ parametrisatie begrensd. Hierdoor kunnen we het bereik van θ en ρ opdelen in een eindig aantal cellen. Het algoritme gaat nu als volgt:

- Voor elk punt (x_i, y_i) 'tekenen' we een sinus-curve met vergelijking $x \cos\theta + y \sin\theta = \rho$ in het θ, ρ rooster door de overeenkomstige cellen met 1 te verhogen.
- De cellen met de hoogste waarden zijn de lijnpatronen die we zochten.

Deze stappen zien we ook duidelijk in figuur 5.3. De nauwkeurigheid van het algoritme van Hough hangt af van de grootte van de cellen. Om een nauwkeuriger resultaat te bekomen kan men het aantal cellen verhogen (dus grootte van de cellen verkleinen). Dit zorgt niet alleen voor een hogere rekenkost, maar kan het zoeken naar highlights in het rooster moeilijker maken

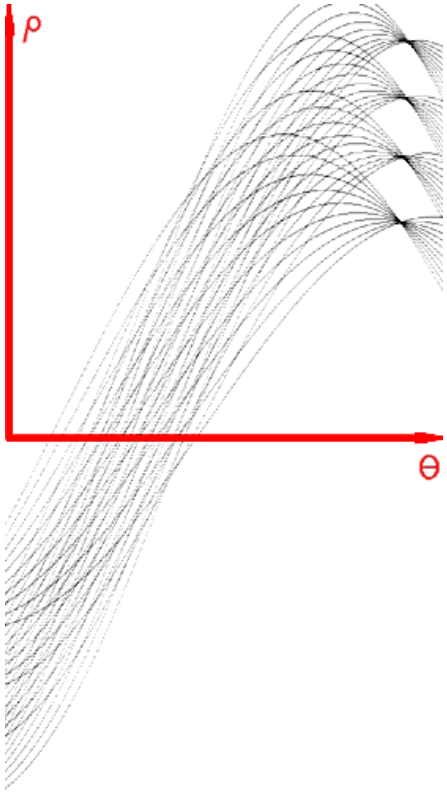


Figuur 5.3: Links boven zien we 5 punten in de XY-ruimte. Voor elk van deze wordt er een harmonische functie getekend, dit zien we rechts boven. Aan de rechterzijde zien we hoe we de Hough parameterruimte kunnen opsplitsen in cellen. Links onder zijn de highlights aangeduid met A en B, deze stemmen overeen met respectievelijk de lijnen door 1, 3 en 5 en door 2, 3, 4. (Figuur uit [11])

omdat deze verspreid liggen over meerdere cellen.
Over het algemeen is het algoritme van Hough een zeer robuuste manier om lijnpatronen te zoeken en is het ook goed bestand tegen ruis. Bij beelden met veel lijnpixels werkt het ook relatief snel doordat de Hough transformatie maar een complexiteit $O(N)$ heeft. Maar hierbij moeten we de constante kost om de curves te tekenen en om de highlights te zoeken niet onderschatten. Zeker in ons geval waar we relatief weinig invoerpunten hebben. Daarom moet er gezocht worden naar optimalisaties van dit algoritme.

5.2.4 Optimalisatie

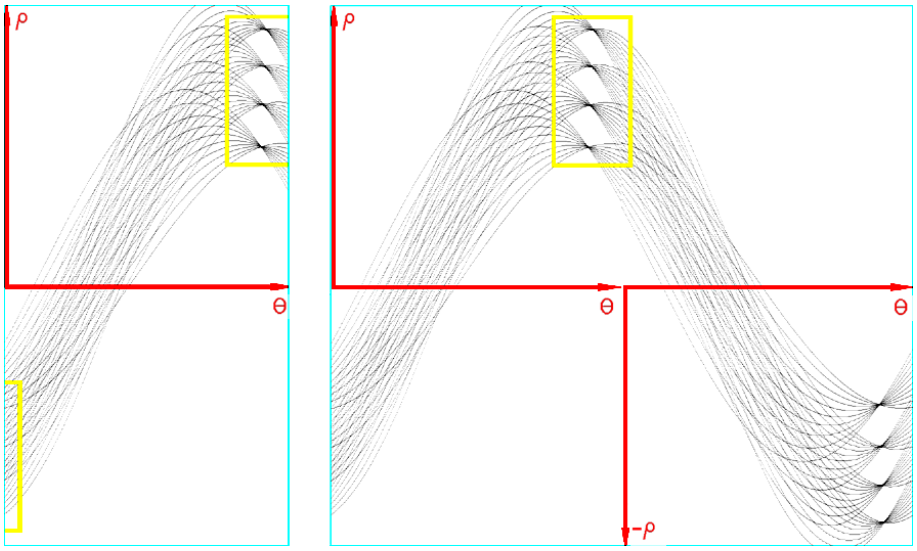
Wanneer we de Hough ruimte bekijken (figuur 5.4) dan zien we dat het grootste gedeelte van de ruimte leeg is of slechts heel weinig punten bevat. De highlights zelf liggen ook redelijk gegroepeerd en verplaatsen weinig bij



Figuur 5.4: Wanneer we de Hough ruimte uittekenen zien we duidelijk dat een zeer groot deel van de ruimte leeg is of zeer kleine waardes bevat. We zien ook dat de snijpunten zich in de rechterbovenhoek bevinden.

kleine wijzigingen van oriëntatie. Dit kunnen we uitbuiten door een kleiner zoekwindow te gebruiken om de curves te tekenen en naar highlights te zoeken. We moeten hierbij wel rekening houden dat de Hough ruimte cyclisch is (zie figuur 5.5). Deze optimalisatie verandert niets aan de complexiteit, maar vermindert de constante rekenkost aanzienlijk.

Zoals we in hoofdstuk 8 zullen zien zorgt deze optimalisatie ervoor dat het



Figuur 5.5: De linkse figuur toont ons hoe we een cyclisch zoekvenster definiëren in de cyclische Hough ruimte. Aan de rechterzijde zien we dat dit zoekvenster normaal een rechthoek is in de volledige Hough ruimte.

algoritme van Hough op deze toepassing in reële tijd kan draaien. Toch zouden we graag hebben dat het nog iets sneller gaat zodat onze tracker als een achtergrond proces kan draaien voor andere programma's die nuttig gebruik maken van de tracker. We zien dat in vergelijking met het aantal pixels in het beeld er weinig markeringspunten gebruikt worden om lijnen te herkennen. Hierdoor kunnen we ook methodes gebruiken met een hogere complexiteit, bv. $O(N^2)$, maar met een zeer kleine constante kost. Daarom stellen we voor om de highlights bij de Hough transformatie analytisch te bepalen.
De parametrisatie door lijnen in de A, B ruimte heeft als voordeel dat de snijpunten analytisch eenvoudig te berekenen zijn, maar deze ruimte is onbegrensd zodat er snijpunten op oneindig kunnen liggen. Anderzijds is de parameterruimte θ, ρ wel begrensd, maar zijn de snijpunten tussen sinuscurves moeilijk analytisch te bepalen. Daarom stellen we in de volgende sectie een andere parametrisatie voor die beide voordelen verenigt.

5.3 Nieuwe Hough transformatie met cirkels

In deze thesis stellen we hier een nieuwe Hough transformatie voor. Deze bevat de voordelen van zowel de parametrisatie door lijnen als deze door sinus-curves, nl. dat de parameterruimte begrensd is en dat het snijpunt makkelijk analytisch te bepalen is.

5.3.1 Parametrisatie door cirkels

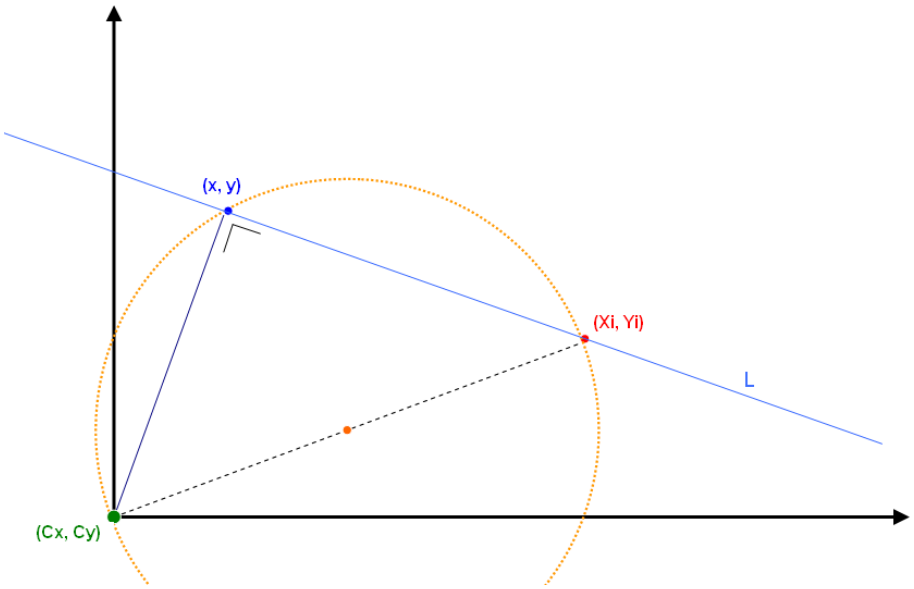
We parametriseren de set van lijnen door een punt (x_i, y_i) door een cirkel met vergelijking

$$\left(x - \frac{x_i + C_x}{2}\right)^2 + \left(y - \frac{y_i + C_y}{2}\right)^2 = \left(\frac{\sqrt{(x_i - C_x)^2 + (y_i - C_y)^2}}{2}\right)^2 \quad (5.1)$$

waarbij (C_x, C_y) een vast gedefiniëerd punt is, bv. $(0, 0)$. Dit wil zeggen dat we voor elk punt (x_i, y_i) een cirkel construeren met als middelpunt het midden tussen (C_x, C_y) en (x_i, y_i) en als straal de helft van de afstand tussen (C_x, C_y) en (x_i, y_i).
De relatie tussen een set van lijnen en de bijbehorende cirkel wordt duidelijk gemaakt op figuur 5.6. Elk punt (x, y) op de cirkel stelt een lijn L voor. Hierbij is (x, y) het punt op L dat het dichtste gelegen is bij het vaste punt. We merken op dat er maar 1 punt het dichtste bij het vaste punt kan liggen en dus ook dat elk punt overeenkomt met precies 1 lijn. De constructie lijkt zeer sterk op deze gebruikt in sectie 5.2.2, maar i.p.v. de hoek θ en de afstand ρ als parameters te gebruiken, gebruiken we nu de positie van dit dichtste punt (x, y). In sectie 5.3.4 zullen we dan ook het verband tussen deze 2 methodes aantonen.

5.3.2 Algoritme

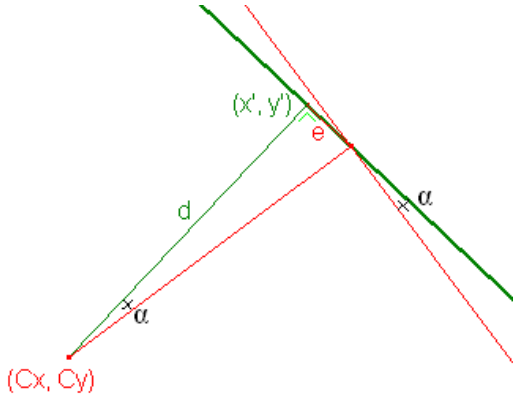
Bij deze parametrisatie doen we in wezen net hetzelfde als bij het vorige algoritme (§5.2.3). We zetten elk punt (x_i, y_i) om naar een cirkel met vergelijking (5.1) en berekenen de onderlinge snijpunten. Hierbij bepalen we de snijpunten niet door ze in een discrete ruimte te tekenen, maar bepalen we ze analytisch. Dit was ondoenbaar wanneer we dit met sinus-curves deden, maar bij cirkels is dit een heel deel simpeler (zie §5.3.3). De plaatsen waar er veel snijpunten bij elkaar liggen, stellen de lijnpatronen voor die we zoeken.



Figuur 5.6: Een set van lijnen kan men parametriseren door een cirkel. Men ziet hier duidelijk dat voor elk punt (x, y) op de cirkel er een lijn door (x_i, y_i) kan geconstrueerd worden.

Door ruis en meetfouten zullen deze snijpunten niet precies op elkaar liggen. Daarom moeten we naburige snijpunten ook meenemen en definiëren we dus een soort van cellen (bins) die de naburige snijpunten verzamelen. De cellen definiëren de posities die lijnen voorstellen met een nog aanvaardbare afwijking. Deze afwijking kunnen we bij voorkeur uitdrukken als de maximale hoek α tussen de gezochtte lijn en een kandidaat lijn omdat het zoeken van een vluchtpunt gevoeliger is voor veranderingen in oriëntatie dan in positie van de lijnen. Het is duidelijk dat posities dicht bij het vast punt gevoeliger zijn voor kleine afwijkingen. Daarom moeten we de grootte van de cellen laten afnemen in functie van de afstand tot het vast punt. Zoals we kunnen zien in figuur 5.7 volgt rechtstreeks uit de driehoeksmeetkunde dat:

$$grootte\ cel = \tan(\alpha) * afstand\ tot\ vast\ punt$$



Figuur 5.7: Wanneer we een maximale afwijkingshoek α hebben kan men de maximale afstand tot het midden van de cel e berekenen d.m.v. een eigenschap uit de driehoeksmeetkunde $\tan\alpha = \frac{e}{d}$.

5.3.3 Snijpunt van 2 cirkels

Hier zullen we aantonen hoe we het snijpunt van 2 cirkels op een redelijk efficiënte manier kunnen berekenen.

We vertrekken vanuit de situatie aangegeven in figuur 5.8, waarbij P_0, P_1, r_0 en r_1 gekend zijn. We controleren eerst of er wel snijpunten zijn door de afstand d tussen P_0 en P_1:

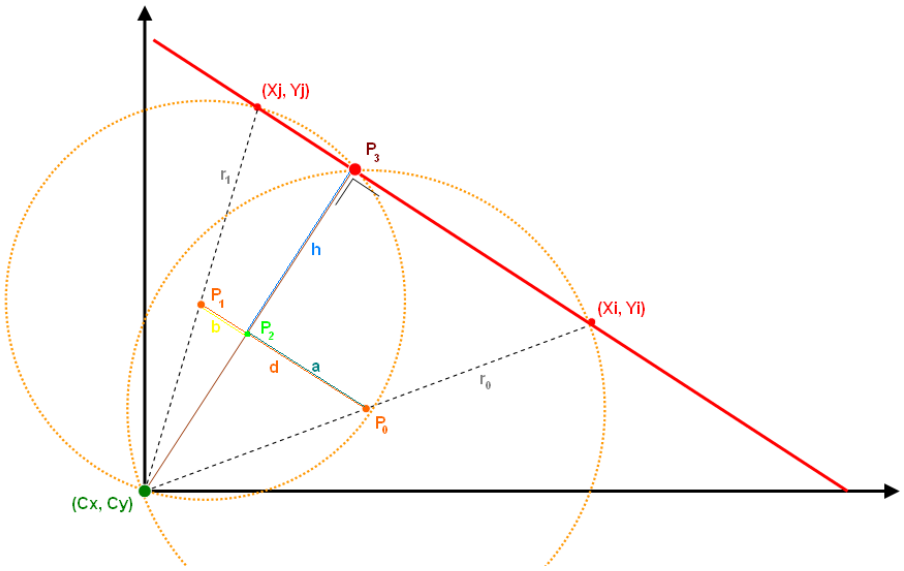
$$d = \|P_1 - P_0\| \tag{5.2}$$

Indien $d > r_0 + r1$ zijn er geen snijpunten. Maar in ons geval weten we al dat er tenminste 1 snijpunt moet zijn, nl. het vast punt (C_x, C_y) waar elke cirkel door moet gaan door constructie. Dus deze test kan niet falen.
Dan gaan we op zoek naar het punt P_2 dat de loodrechte projectie is van de snijpunten P_3 op het lijnstuk $\overline{P_0P_1}$. Dit doen we door de rechthoekige driehoeken $\triangle P_0P_2P_3$ en $\triangle P_1P_2P_3$ te beschouwen zodat door de stelling van Pythagoras:

$$a^2 + h^2 = r_0^2 \tag{5.3}$$
$$b^2 + h^2 = r_1^2 \tag{5.4}$$

en dus dat

$$a^2 - b^2 = r_0^2 - r_1^2 \tag{5.5}$$



Figuur 5.8: Constructie van de cirkels zodat het snijpunt P_3 een lijn door (x_i, y_i) en (x_j, y_j) voorsteld.

We zien ook dat $d = a + b$, zodat we b kunnen vervangen door $d - a$:

$$a^2 - (d^2 - 2ad + a^2) = r_0^2 - r_1^2 \tag{5.6}$$

Als we dit oplossen naar a krijgen we

$$a = \frac{r_0^2 - r_1^2 + d^2}{2d} \tag{5.7}$$

En zo vinden we P_2 dus doordat P_2 tussen P_0 en P_1 ligt op afstand a van P_0:

$$P_2 = P_0 + \frac{a}{d}(P_1 - P_0) \tag{5.8}$$

We weten nu dat 1 van de snijpunten gelijk is aan het vast punt (C_x, C_y) door de constructie van de cirkels. Daardoor kan het andere snijpunt P_3 makkelijk vinden omdat P_2 altijd het midden is tussen de 2 snijpunten en dus dat:

$$P_3 = 2P_2 - C \tag{5.9}$$

Door (5.8) en (5.7) in (5.9) te voegen krijgen we:

$$P_3 = 2P_0 + \frac{r_0^2 - r_1^2 + d^2}{d^2}(P_1 - P_0) - C \tag{5.10}$$

of vereenvoudigd

$$P_3 = (P_0 + P_1) + \frac{r_0^2 - r_1^2}{d^2}(P_1 - P_0) - C \tag{5.11}$$

Hier is dus P_3 het gezochte snijpunt van 2 cirkels (P_0, r_0) en (P_1, r_1) die door een vast punt C gaan zoals in de constructie in sectie 5.3.1.

5.3.4 Verband tussen de cirkels en sinus-curves

Zoals we eerder al hadden aangehaald is er een grote gelijkenis in de constructie van de parametrisatie door cirkels en deze door sinus-curves. In deze sectie zullen we dan ook aantonen dat deze parametrisaties eigelijk dezelfde zijn, maar dan in een verschillende parameterruimte.
Om dit aan te tonen zullen we vertrekken van de parametrisatie van een lijn door een cirkel met als vast punt $(0, 0)$. We kiezen de oorsprong als vast punt omdat dit ook bij de constructie van de sinus-curves gebruikt wordt. Door (C_x, C_y) gelijk te stellen aan $(0, 0)$ in vergelijking 5.1 krijgen we:

$$\left(x - \frac{x_i}{2}\right)^2 + \left(y - \frac{y_i}{2}\right)^2 = \frac{x_i^2 + y_i^2}{4} \tag{5.12}$$

Als we de kwadraten verder uitwerken krijgen we:

$$x^2 - x_i x + \frac{x_i^2}{4} + y^2 - y_i y + \frac{y_i^2}{4} = \frac{x_i^2}{4} + \frac{y_i^2}{4} \tag{5.13}$$

of vereenvoudigd

$$x^2 - x_i x + y^2 - y_i y = 0 \tag{5.14}$$

We zetten deze cirkel nu om naar de parameterruimte θ, ρ door elk punt om te zetten naar zijn poolcoördinaten, d.w.z.

$$\forall (x, y), \exists (\theta, \rho) : x = \rho \cos\theta, y = \rho \sin\theta$$

De vergelijking van de cirkel in poolcoördinaten wordt dan

$$\rho^2 \cos^2\theta - x_i \rho \cos\theta + \rho^2 \sin^2\theta - y_i \rho \cos\theta = 0 \tag{5.15}$$

Wanneer we veronderstellen dat $\rho \neq 0$ (d.w.z. dat de straal van de cirkel groter is als 0) krijgen we dat

$$\rho(\cos^2\theta + \sin^2\theta) - x_i \cos\theta - y_i \cos\theta = 0 \tag{5.16}$$

en omdat we weten dat $\cos^2\theta + \sin^2\theta = 1$ wordt de vergelijking

$$\rho = x_i \cos\theta + y_i \cos\theta \tag{5.17}$$

Zoals we kunnen zien is dit de parametervergelijking van een lijn in de θ, ρ ruimte uit sectie 5.2.2, wat we dus wilden aantonen. Hieruit volgt dat de parametrisatie door cirkels dezelfde robuustheid heeft als de oorspronkelijke Hough transformatie. Maar het analytisch werken met cirkels is eenvoudiger dan met de sinus-curves. Daarom zullen we in de resultaten ook zien dat deze methode veel sneller is voor een kleiner aantal invoerpunten.

Hoofdstuk 6

Berekening van oriëntatie uit vluchtpunten

Uit de vorige stappen hebben we de vergelijkingingen van de lijnen gevonden in de beeldruimte. De lijnen kunnen we indelen in sets op basis van hun kleur van markeringen. Elke set van deze lijnen stellen evenwijdige lijnen voor in de wereld. Door het vluchtpunt van elke set te beschouwen kunnen we de oriëntatie van de camera bepalen. In §6.1 zullen we eerst uitleggen hoe we de vluchtpunten berekenen. Daarna leggen we in sectie 6.2 uit waarom we de oriëntatie van de camera uit vluchtpunten kunnen bepalen zonder dat we voorkennis hebben van de translatie die deze ondergaan heeft. In §6.3 zien we dan hoe we precies de rotatiematrix uit de vluchtpunten berekenen en in 6.4 zien we tenslotte dat we eigenlijk maar 2 vluchtpunten nodig hebben.

6.1 Vluchtpunten berekenen

De lijnpatronen die we uit de vorige stap gevonden hebben stellen de parallelle lijnen voor in de wereld. Het snijpunt van deze is dan ook hun vluchtpunt. We berekenen het 'best fit' vluchtpunt op de volgende manier[1]:

- We berekenen de 'second moment' matrix M:

$$M = \sum_{i=1}^{n} \begin{bmatrix} a_i^2 & a_i b_i & a_i c_i \\ a_i b_i & b_i^2 & b_i c_i \\ a_i c_i & b_i c_i & c_i^2 \end{bmatrix}$$

[1] http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt

waarbij $a_i x + b_i y + c_i = 0$ de vergelijking is van de i-de lijn.
Merk op: M is een symmetrische matrix.

- Bepaal uit deze matrix M de eigenwaarden en eigenvectoren. Dit kan men het beste doen met de Jacobi methode [10] omdat deze zeer goed is voor symmetrische matrices. In mijn implementatie heb ik hiervoor de Newmat C++ matrix library[2] gebruikt.

- De eigenvector die bij de kleinste eigenwaarde hoort is het vluchtpunt van deze set lijnen.

Het vluchtpunt dat we bekomen is een vector met 3 waardes. Wanneer de laatste waarde dicht bij 0 ligt, betekent dit dat de lijnen in het beeld parallel (of toch bijna parallel) lopen. De eerste 2 waardes bepalen dan de richting van de lijnen. Als de laatste waarde niet 0 is dan kan men alle 3 de waardes delen door de laatste en bekomt men de coördinaten van het snijpunt van de lijnen. M.a.w. dit zijn de homogene 2D coördinaten van het vluchtpunt.

6.2 Relatie tussen beeld- en wereldcoördinaten van een vluchtpunt

Voordat we de oriëntatie uit de vluchtpunten kunnen bepalen moeten we natuurlijk eerst weten wat de relatie is tussen de beeldcoördinaten en de wereldcoördinaten van een vluchtpunt. De wereldcoördinaten van een vluchtpunt is de richtingsvector van de betreffende groep parallelle lijnen. Deze is voor elke groep van lijnen (per kleur van markeringen) gegeven. We zullen ook aantonen dat we geen voorkennis moeten hebben over de translatie die de camera ondergaan heeft.
Uit 2.3.1 krijgen we dat:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = K \; [\; R \mid T \;] \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{6.1}$$

[2] http://www.robertnz.net/nm_intro.htm

We vermenigvuldigen nu beide zijden met K^{-1}:

$$\lambda_i K^{-1} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = [\ R \mid T\ ] \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{6.2}$$

We zullen nu bewijzen dat T onbelangrijk is voor ons doel en dus gelijkgesteld kan worden aan $[0, 0, 0]^T$ (en dus weggelaten kan worden) doordat we werken met vluchtpunten:

- We vertrekken van de vergelijking van een rechte in functie van vectoren:
$$L(t) = \vec{p_0} + t\vec{D}$$
- Het punt op oneindig van deze rechte is dan:
$$\vec{p} = \lim_{t\to\infty}(\vec{p_0} + t\vec{D})$$
Dit punt is het vluchtpunt van de lijnen die parallel lopen met deze rechte en zal door projectie afgebeeld worden op het snijpunt van de beelden evenwijdige lijnen.
- Wanneer we dit vluchtpunt in homogene coördinaten uitschrijven krijgen we:
$$\lim_{t\to\infty}(\vec{p_0} + t\vec{D}) = \lim_{t\to\infty} \begin{bmatrix} x_0 + tx_D \\ y_0 + ty_D \\ z_0 + tz_D \\ 1 \end{bmatrix} \tag{6.3}$$
Omdat dit homogene coördinaten zijn kunnen we elke coëfficiënt delen door t:
$$\lim_{t\to\infty}(\vec{p_0} + t\vec{D}) = \lim_{t\to\infty} \begin{bmatrix} x_0/t + x_D \\ y_0/t + y_D \\ z_0/t + z_D \\ 1/t \end{bmatrix} \tag{6.4}$$
Wanneer we deze limiet dan uitvoeren krijgen we:
$$\lim_{t\to\infty}(\vec{p_0} + t\vec{D}) = \lim_{t\to\infty} \begin{bmatrix} x_D \\ y_D \\ z_D \\ 0 \end{bmatrix} \tag{6.5}$$

We zien dat dit de richtingsvector is van de rechte. Deze is hetzelfde voor elke evenwijdige lijn in de wereld.

- Het vluchtpunt na projectie voldoet nog steeds aan:

$$\lambda Y = K[R|T] \begin{bmatrix} x_D \\ y_D \\ z_D \\ 0 \end{bmatrix} \tag{6.6}$$

Zoals we zien (door die 0) is deze vergelijking onafhankelijk van T, dus

$$\lambda Y = KR \begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} \tag{6.7}$$

Wanneer we in deze laatste formule beide zijden vermenigvuldigen met K^{-1}:

$$\lambda_i K^{-1} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = R \begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} \tag{6.8}$$

of:

$$\lambda_i \; K^{-1} \; V_i = R \; D_i \tag{6.9}$$

waarbij V_i de beeldcoördinaten van het vluchtpunt zijn en D_i de overeenstemmende wereldcoördinaten.

6.3 Bepaling van de rotatiematrix d.m.v. vluchtpunten

De rotatiematrix R is een 3x3 matrix, dus 9 waarden. Deze kunnen we bepalen met 3 vluchtpunten (3D vectoren) wat ons dus ook 9 vergelijkingen geeft. Een rotatiematrix kan steeds als volgt geschreven worden:

$$R = \begin{bmatrix} H_x & H_y & H_z \\ U_x & U_y & U_z \\ N_x & N_y & N_z \end{bmatrix} \tag{6.10}$$

Hierbij vormen de vectoren H, U en N een orthogonale basis die ook de oriëntatie van de camera bepaalt, nl.:

- de richting H stemt overeen met rechts in het beeld en zal dus afgebeeld worden op de X-richting
- de richting U stelt de richting omhoog voor in het beeld en wordt afgebeeld op de Y-richting
- de richting N is de omgekeerde kijkrichting van de camera en zal dus afgebeeld worden op de Z-richting

We zullen nu aantonen hoe we de rotatiematrix kunnen bepalen uit onze gevonden vluchtpunten. Neem vluchtpunt $V_1 = [u_1, v_1, w_1]^T$ het vluchtpunt dat overeenstemt met de X-richting, dus met de richting $D_1 = [1, 0, 0]^T$. Hetzelfde doen we voor V_2 en V_3, de vluchtpunten van de Y- en Z-richting, die dus overeenstemmen met $D_2 = [0, 1, 0]^T$ en $D_3 = [0, 0, 1]^T$. We kunnen deze vectoren samen zetten in een 3x3 matrix en krijgen dan dat:

$$V = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \tag{6.11}$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.12}$$

In vergelijking (6.9) ontbreekt nu nog enkel de schaalfactor λ_i. Deze bepalen we door het feit dat de transpose van de rotatiematrix gelijk is aan zijn inverse, zoals in [9], (dus $R^T.R = I$) en D_i de genormaliseerde richtingsvectoren zijn (dus $D_i^T.D_i = 1$). Voor het gemak stellen we $U_i = K^{-1}\ V_i$:

$$(\lambda_i U_i)^T.(\lambda_i U_i) \overset{(6.9)}{=} (RD_i)^T.(RD_i) = D_i^T R^T R D_i$$

$$\Downarrow \quad {\scriptstyle (R^T.R=I)}$$

$$\lambda_i^2 (U_i^T U_i) = D_i^T.I.D_i = D_i^T D_i$$

$$\Downarrow \quad {\scriptstyle (D_i^T D_i=1)}$$

$$\lambda_i^2 (U_i^T U_i) = 1$$

$$\Downarrow$$

$$\lambda_i = \pm\sqrt{\frac{1}{U_i^T U_i}} = \pm\frac{1}{|U_i|} \tag{6.13}$$

We leggen nu de relatie tussen de beeld- en wereldcoördinaten van de vluchtpunten door (6.11), (6.12) en (6.13) in de relatie (6.9) in te vullen:

$$\begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} H_x & H_y & H_z \\ U_x & U_y & U_z \\ N_x & N_y & N_z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} \frac{1}{\alpha} & 0 & -\frac{u_0}{\alpha} \\ 0 & \frac{1}{\alpha} & -\frac{v_0}{\alpha} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} H_x & H_y & H_z \\ U_x & U_y & U_z \\ N_x & N_y & N_z \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} H_x & H_y & H_z \\ U_x & U_y & U_z \\ N_x & N_y & N_z \end{bmatrix} = \begin{bmatrix} \lambda_1 \frac{u_1 - u_0 w_1}{\alpha} & \lambda_2 \frac{u_2 - u_0 w_2}{\alpha} & \lambda_3 \frac{u_3 - u_0 w_3}{\alpha} \\ \lambda_1 \frac{v_1 - v_0 w_1}{\alpha} & \lambda_2 \frac{v_2 - v_0 w_2}{\alpha} & \lambda_3 \frac{v_3 - v_0 w_3}{\alpha} \\ \lambda_1 w_1 & \lambda_2 w_2 & \lambda_3 w_3 \end{bmatrix} \tag{6.14}$$

met $\lambda_i = \pm 1 / \sqrt{\frac{u_i - u_0 w_i}{\alpha}^2 + \frac{v_i - v_0 w_i}{\alpha}^2 + w_i^2}$

6.4 Oriëntatie bepalen uit 2 vluchtpunten

Zoals we in formule (6.14) zien heeft men 3 vluchtpunten nodig om de rotatiematrix te bepalen. Doordat H, U en N een orthogonale basis is, zijn er een aantal constraints waaraan deze vectoren moeten voldoen. Namelijk H, U en N zijn genormaliseerd en staan alle 3 loodrecht op elkaar, dus:

$$\begin{cases} H = U \otimes N \\ U = H \otimes N \\ N = H \otimes U \end{cases} \tag{6.15}$$

Deze constraints kunnen we gebruiken om op de data, die we uit de 3 vluchtpunten berekend hebben, normalisatie en orthogonaliteit te forceren en zo dus meetfouten te verkleinen. Maar we kunnen deze constraints ook gebruiken zodat we maar 2 in plaats van 3 vluchtpunten nodig hebben. Neem bijvoorbeeld dat we slechts 2 soorten markeringen op het plafond aanbrengen (zoals in mijn implementatie). Hieruit kunnen we dan de vluchtpunten van X- en Z-richting halen. Met behulp van formule (6.14) kunnen we dan alle

componenten bepalen, behalve H_y, U_y en N_y. Door hier de orthogonaliteit constraint (6.15) op toe te passen krijgen we dat:

$$\begin{cases} H_y = U_x N_z - U_z N_x \\ U_y = H_x N_z - H_z N_x \\ N_y = H_x U_z - H_z U_x \end{cases}$$

Door maar met 2 markeringen te werken kan de werkruimte elke afmeting hebben en de ruimte groter maken doet men dan gewoon door het gewenste stuk markeringen toe te voegen. Anderzijds kan men ook een 3^e markering gebruiken en kan men indien er te weinig informatie is voor 1 van de markeringen overschakelen op enkel de andere 2 markeringen. Zo kan men volledig 360° rond draaien in elke richting zolang er maar minstens 2 soorten markeringen in voldoende mate zichtbaar zijn in de camera.

6.5 Ambiguïteiten

In de berekende waarden van de rotatiematrix (vergelijking 6.14) zitten nog een aantal ambiguïteiten doordat λ_i positief of negatief kan zijn. Voor elk van de 3 vluchtpunten + of - kiezen geeft ons dus 8 ambiguïteiten. Deze bepalen in welk octant we aan het werken zijn. Deze kunnen we eenvoudig oplossen door te kijken naar het vorige frame. We kiezen dan de oplossing die het dichtst bij de vorige oplossing lag. Als beginsituatie kunnen we de gebruiker laten opgeven in welk octant de camera start. Door deze ambiguïteiten kunnen we duidelijk geen absolute oriëntatie garanderen. Indien dit toch gewenst is kunnen er bijkomende markeringen aangebracht worden die deze ambiguïteiten met zekerheid kunnen oplossen.

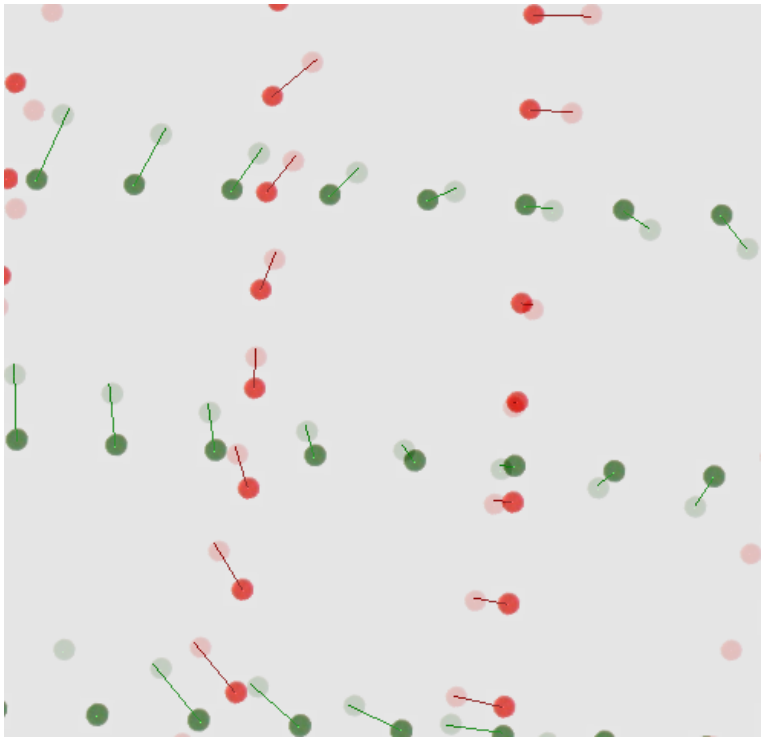
Hoofdstuk 7

Bepaling van translatie

In het vorig hoofdstuk hebben we de rotatiematrix bepaald. In deze stap zullen we de overblijvende extrinsieke parameters berekenen, nl. de translatie van de camera. In hoofdstuk 4 werd beschreven hoe de optical flow van de markeringen berekend kan worden. Uit de overeenkomsten van markeringen in opeenvolgende frames kunnen we de translatie berekenen.
In sectie 7.1 leiden we eerst een formule af die de translatie berekend a.h.v. puntovereenkomsten in de cameraruimtes. Spijtig genoeg hebben we enkel de 2D beeldcoördinaten die de punten in de cameraruimte bepalen op een schaalfactor na. In sectie 7.2 zullen we dan een algoritme beschrijven hoe we de translatie kunnen berekenen uit de optical flow van 2D beeldpunten. Daarna zullen we ook bekijken hoe we de schaalfactoren kunnen calibreren (7.3). Ten slotte doen we in sectie 7.4 een controle of er geen punten ontbreken op een lijn door gebruik te maken van de dubbelverhouding.

7.1 Translatie bepalen uit puntovereenkomsten

We zullen hier eerst de relatie afleiden tussen de translatie van de camera en de verplaatsing van punten in het camera-assenstelsel. We gaan er van uit dat de rotatiematrix gekend is en dat we weten welke punten overeenkomen in de twee beelden (Zie figuur 7.1).
We vertrekken vanuit de algemene vergelijking van een transformatie van de ene basis naar de andere. In dit geval is dat de transformatie van de



Figuur 7.1: De doorzichtige markeringen stellen het vorige frame voor, de volle markeringen is het huidige frame en de verbindingslijnen zijn de optical flow van de markeringen die in de 2 frames gevonden werden.

coördinaten in de camera-assenstesel naar deze in de wereld:

$$X = R_1^T.X_1' + T_1 \tag{7.1}$$
$$X = R_2^T.X_2' + T_2 \tag{7.2}$$

Hierin zijn R_1 en R_2 de rotatiematrices en T_1 en T_2 de positie vectoren van respectievelijk camera 1 en 2. X_1' en X_2' zijn de coördinaten van X in respectievelijk de camera-assenstelsels 1 en 2.
Omdat we de scène statisch geconstrueerd hebben, weten we dat X niet zal wijzingen in de tijd, daarom

$$R_1^T.X_1' + T_1 = R_2^T.X_2' + T_2 \tag{7.3}$$

De translatie die camera 2 gemaakt heeft t.o.v. camera 1 is dan

$$\Delta T = T_2 - T_1 = R_1^T.X_1' - R_2^T.X_2' \tag{7.4}$$

Deze formule geeft aan dat we de translatie eenvoudig kunnen berekenen met de cameracoördinaten van 1 wereldpunt in 2 aanzichten. Spijtig genoeg weten we X_1' en X_2' niet precies liggen omdat we enkel hun geprojecteerde beeldcoördinaten kennen.

7.2 Translatie berekenen uit 2D optical flow

Zoals we in de vorige sectie gezien hebben, hebben we de cameracoördinaten nodig van onze markeringen maar kennen we enkel zijn beeldcoördinaten. We hebben in 2.3.1 reeds gezien dat de calibratiematrix K de homogene beeldcoördinaten $(u, v, 1)$ relateert met een punt X' in cameracoördinaten, waardoor geldt dat

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{\alpha} & 0 & u_0 \\ 0 & \frac{f}{\alpha} & v_0 \\ 0 & 0 & 1 \end{bmatrix} X' \tag{7.5}$$

Als we K dan inverteren krijgen we

$$X' = \lambda \begin{bmatrix} \frac{\alpha}{f} & 0 & -\frac{\alpha u_0}{f} \\ 0 & \frac{\alpha}{f} & -\frac{\alpha v_0}{f} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{7.6}$$

en dus krijgen we dat

$$X' = \lambda \begin{bmatrix} \frac{\alpha(u-u_0)}{f} \\ \frac{\alpha(v-v_0)}{f} \\ 1 \end{bmatrix} = \lambda Y \tag{7.7}$$

Als we dit nu samen nemen met vergelijking (7.4) krijgen we dat we 5 onbekenden hebben, nl. ΔT_x, ΔT_y, ΔT_z, λ_1 en λ_2. We zien wel dat λ_1 dezelfde is als λ_2 in de vorige iteratie. Wanneer we gebruik maken van het feit dat we de afstand tussen opeenvolgende LED's kennen kunnen we λ_2 van elk punt in het tweede beeld berekenen. Hierdoor moeten we in elke iteratie enkel λ_2 berekenen en dan kunnen we de translatievector uit elk punt halen met vergelijking (7.4).

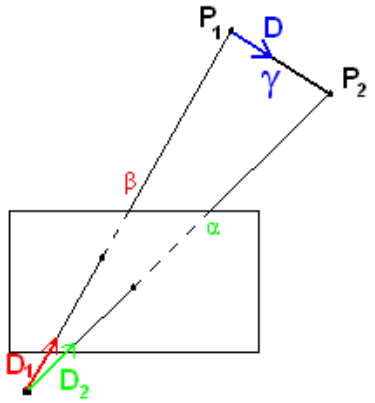
7.3 Calibratie van de schaalfactoren

We zullen nu afleiden hoe we, gegeven 2 opeenvolgende punten op een lijn, de schaalfactoren (en daarmee ook de diepte) berekenen. Bij elk van die punten hoort een homogeen beeldpunt zoals in vergelijking (7.7). Deze komen overeen met een richting D^i in de wereldruimte met

$$\vec{D_i} = R^T Y^i \tag{7.8}$$

Stel dat $\vec{D}$ de opgegeven richting is waarin de markeringen geplaatst zijn in de wereld en γ de afstand tussen 2 LED's in de wereld, dan zien we duidelijk uit figuur 7.2 dat

$$\gamma\vec{D} = \alpha\vec{D_2} - \beta\vec{D_1} \tag{7.9}$$



Figuur 7.2: Gegeven de homogene beeldcoördinaten $\vec{D_1}$ en $\vec{D_2}$ en de richting van P_1 naar P_2 met hun onderlinge afstand γ in de wereld, kunnen we de schaalfactoren α en β berekenen.

De richting $\vec{D}$ krijgen we gegeven uit het feit dat we de kleur van de lijn waar deze 2 punten op liggen kennen en daarmee ook de richting die overeenstemd met deze lijn. Uit de vorige vergelijking kunnen we α en β berekenen. Dit zijn dan onmiddelijk de gezochte schaalfactoren λ uit vergelijking (7.7).
Omdat we weten dat het schatten van de diepte vanuit 1 camerastandpunt zeer onnauwkeurig is kunnen we best deze operatie doen op alle lijnpunten onderling en de waarden uitmiddelen. Ook kunnen we gebruik maken van

het feit dat al deze punten op een 3D lijn liggen en op een vaste afstand van elkaar. Hierdoor kunnen we outliers in de λ's verminderen.
In deze afleiding zijn we er wel vanuit gegaan dat we de afstand tussen 2 opeenvolgende punten met zekerheid kennen als de afstand tussen 2 LED's. Dit kan bijvoorbeeld mis gaan als er een LED in het midden van een lijn ontbreekt.

7.4 Controle op ontbrekende markeringen

Zoals we hierboven al aangegeven hebben moeten we controleren of er in een reeks van equidistante markeringen er geen ontbreekt of bijgekomen is. Dit doen we door gebruik te maken van de dubbelverhouding. De dubbelverhouding is een eigenschap van 3 afstanden tussen 4 punten die bewaard blijft onder een projectie.
Gegeven 4 opeenvolgende LED's A, B, C, D met een onderlinge afstand d (zoals in figuur 7.3), dan is de dubbelverhouding gedefiniëerd als

$$(ABCD) = \frac{(ABC)}{(ABD)} = \frac{\frac{\|CA\|}{\|CB\|}}{\frac{\|DA\|}{\|DB\|}} \tag{7.10}$$

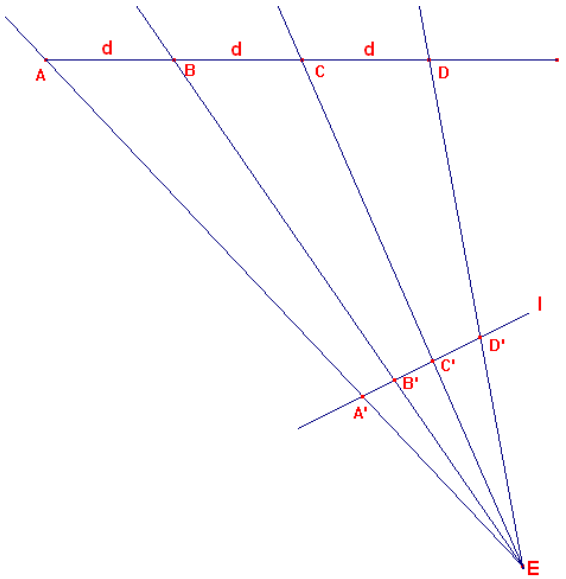
Wanneer we hier nu hun onderlinge afstand d invullen krijgen we

$$(ABCD) = \frac{\frac{2d}{d}}{\frac{3d}{2d}} = \frac{4}{3} = 1.33 \tag{7.11}$$

Eén van de eigenschappen van de dubbelverhouding is dat deze bewaard blijft onder projectie, dus

$$(A'B'C'D') = \frac{4}{3} = 1.33 \tag{7.12}$$

Nu kunnen we dus de lijn aflopen en per koppeltje van 4 punten hun dubbelverhouding berekenen. Indien deze te sterk verschilt, weten we dat de afstand tussen deze punten dus niet dezelfde is en dat er dus een markering moet ontbreken.



Figuur 7.3: Gegeven de dubbelverhouding $(ABCD)$ in de wereld, dan geldt dat de dubbelverhouding van de projectie van deze punten $(A'B'C'D') = (ABCD)$

Hoofdstuk 8

Resultaten en discussie

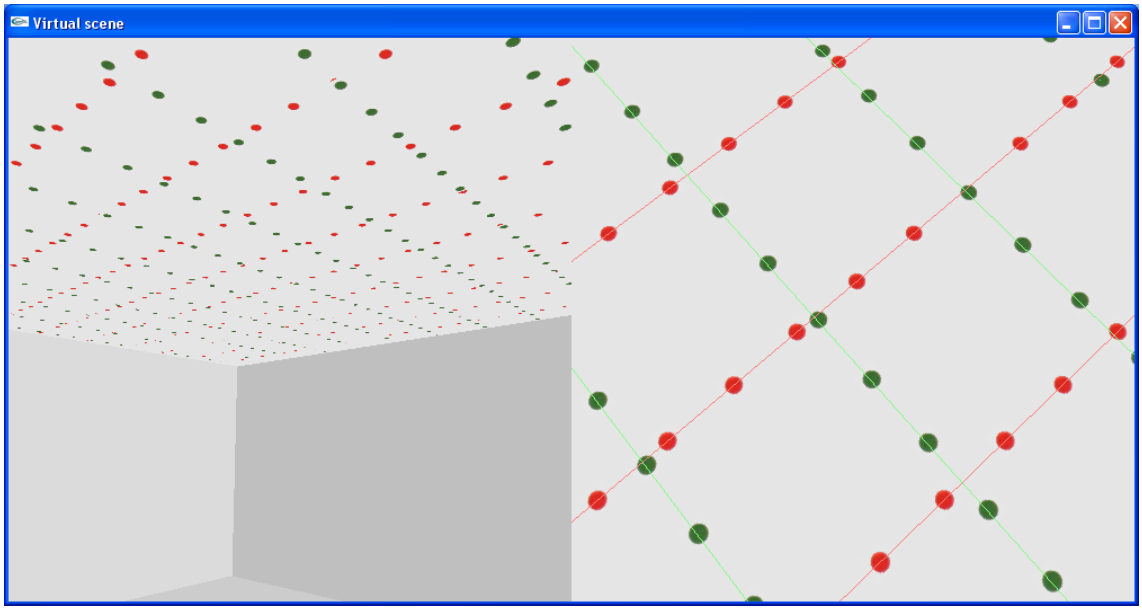
Om de algoritmes en technieken uit de vorige hoofdstukken te testen en beoordelen hebben we deze ook geïmplementeerd. Om deze te testen hebben we ook een testopstellingen gebouwd. Deze zullen we eerst bespreken in sectie 8.1. Vervolgens gaan we de voorgaande algoritmes en hun implementatie beoordelen naar snelheid en accuraatheid.
Alle testen met de virtuele scène werden uitgevoerd op een pc met een 2GHz AMD processor met 1Gb RAM. De scène is in OpenGL geschreven m.b.v. GLUT. Hierbij is het tekenen en het inlezen van de OpenGL schermbuffer (500×500) niet in de tijden meebegrepen. De testen met een echte camera werden gedaan op de computers in de CAVE van het EDM. Dit zijn Dualcore processors (1.5GHz) en 2Gb RAM met een firewire camera aangesloten. Deze camera maakt beelden van 1024×768 aan 30fps, waarbij ook hier het maken en binnen halen van de beelden niet in de tijden is begrepen.
In sectie 8.2 zullen we eerst kijken naar de resultaten in de virtuele opstelling. Hier vergelijken we de verschillende implementaties van het algoritme van Hough en bekijken we de resultaten van de oriëntatie- en positietracker. Daarna kijken we hoe goed onze tracker het doet in de echte wereld (§8.3).

8.1 Testopstellingen

8.1.1 Virtuele testopstelling

Eerst hebben we een virtuele testopstelling gemaakt in OpenGL. Hierin hebben we een 3D virtuele kamer geconstrueerd waar we markeringen op het plafond aangebracht hebben zoals in de constructie in hoofdstuk 3.1. In figuur 8.1 zien we een screenshot van deze implementatie waarbij links de kamer getoond wordt en rechts hetgeen de camera ziet. De virtuele kamer is 8 op 10 meter groot en 2,5 meter hoog. Op het plafond hebben we 2 soorten markeringen aangebracht:

- Rode: deze zijn in lijnen geplakt die evenwijdig lopen met de X-as in de wereld. Ze worden 17 cm van elkaar geplakt en tussen 2 lijnen is er een afstand van 45 cm.
- Groene: deze stellen de lijnen voor evenwijdig met de Z-as. Ze worden op dezelfde afstanden geplakt als de rode markeringen.

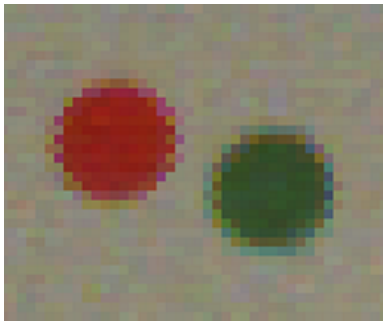


Figuur 8.1: De implementatie van een virtuele opstelling waarbij het rechtse beeld de invoer van het algoritme is.

De camera heeft vaste intrinsieke waardes zoals een brandpuntsafstand van 2.3 mm en een pixelbreedte van 5 μm. Er is ook geen lensdistortie op deze ideale camera en zijn principal point ligt mooi in het midden van het beeld. De camera maakt beelden met een resolutie van 500×500 die d.m.v. de OpenGL functie *glReadPixels()* in BGR formaat ingelezen worden en naar de trackermodule verstuurd worden.
Een virtuele testopstelling laat ons toe om dingen in de scène te veranderen en om de resultaten dan eenvoudig te kunnen vergelijken met de echte waarde.

8.1.2 Opstelling in de reële wereld

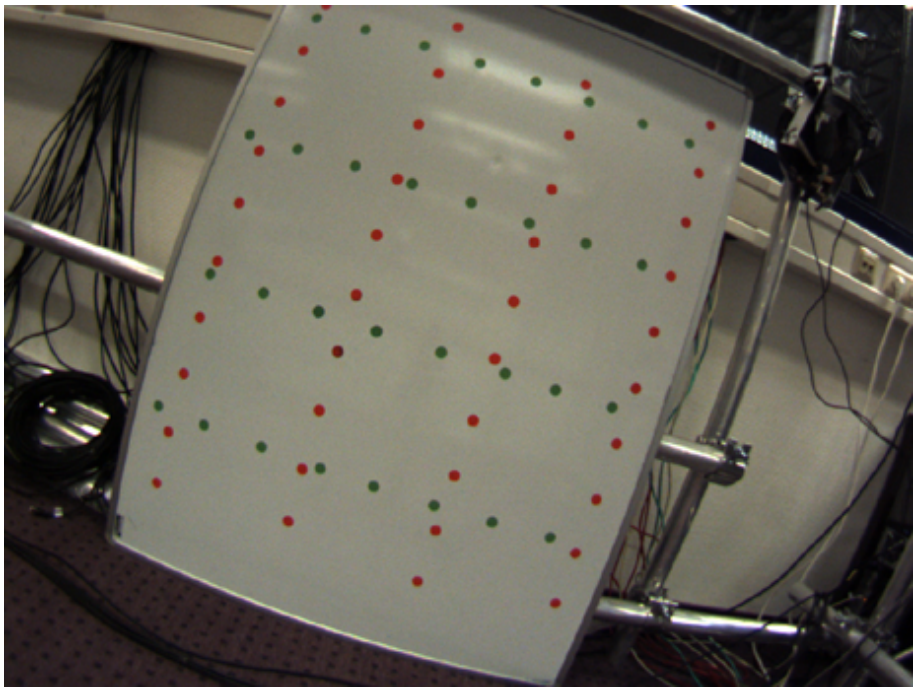
Wanneer we een echte scène gaan bouwen en met een echte camera gaan werken, moeten we er ook rekening mee houden dat er allerlei distorties (lensdistortie, motionblur, ...) kunnen voorkomen. Ook moeten we rekening houden met ruis, lichte kleurschakeringen (zie figuur 8.2) en storende achtergrondvoorwerpen die mogelijk kunnen leiden tot detectie van verkeerde markeringen.
In onze reële-wereld opstelling maken we gebruik van gewone ronde stikker-



Figuur 8.2: Hier hebben we een beeld van markeringen sterk vergroot. We zien dat de markeringen niet uit 1 mooi egale kleur bestaan. Aan de randen zien we dat de kleur plaatselijk zelfs sterk kan variëren.

tjes als markeringen en een wide-angle camera. Hierdoor zien we duidelijk de lensdistortie die we eerst moeten elimineren. De stikkertjes komen uit een papierhandelaar en kostten slechts 0.50 EUR voor 100 stuks waardoor deze opstelling zeer goedkoop uit te breiden is (tot in het oneindige als er genoeg plafond is). In een andere niet-labo context kan er gebruik gemaakt worden van LED-strips. Deze zijn makkelijker te herkennen en zorgen ervoor

dat achtergrond bijna volledig genegeerd kan worden. Figuur 8.3 toont onze opstelling in een invoerbeeld afkomstig van de gebruikte camera.

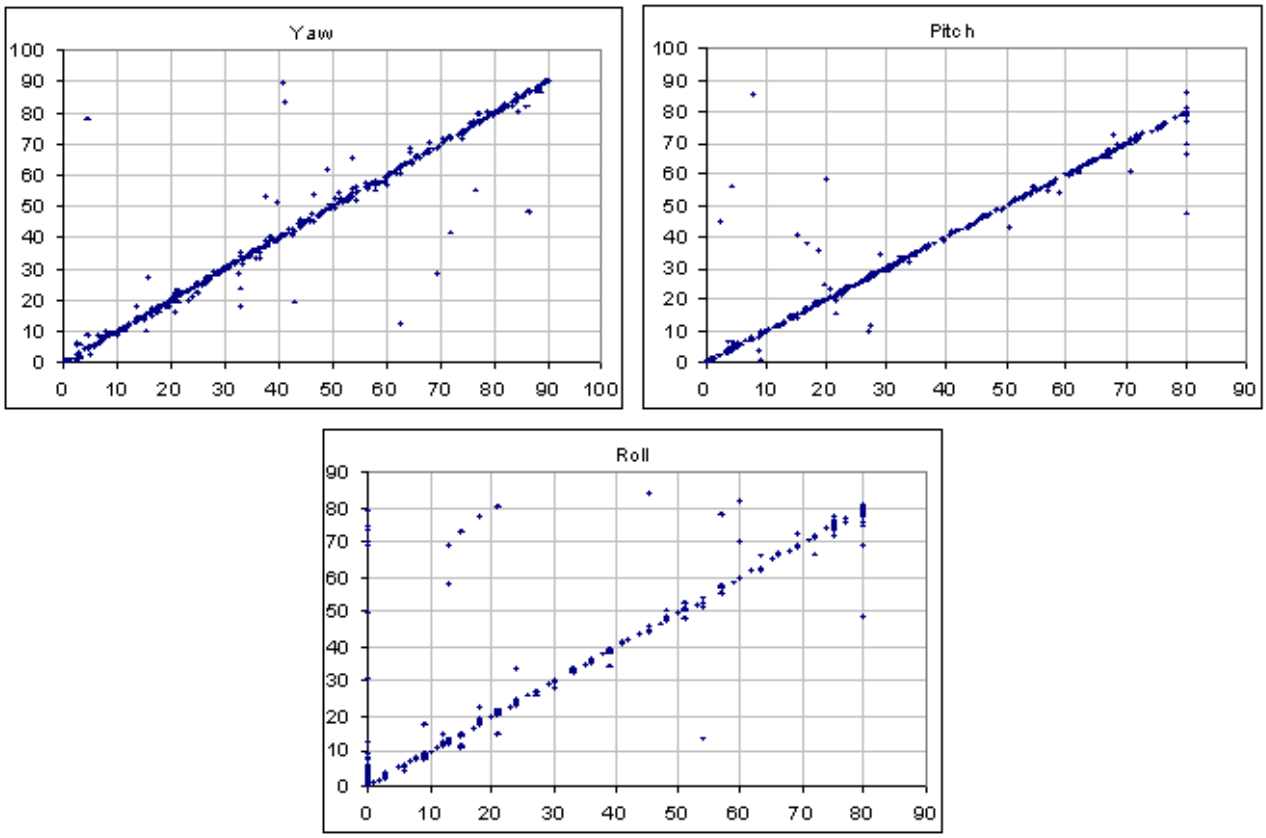


Figuur 8.3: Dit invoerbeeld van de camera toont de opstelling die we gebouwd hebben in de echte wereld.

8.2 Resultaten virtuele testopstelling

8.2.1 Gewone Hough transformatie

In onze eerste implementatie hebben we gebruik gemaakt van de gewone Hough transformatie zoals deze staat uitgelegd in §5.2.2. De nauwkeurigheid van θ nemen we 0.5° en voor ρ 2 pixels. De grafieken in figuur 8.4 geven de berekende waarden voor de Yaw, Pitch en Roll in functie van de echte waarden. Zoals we kunnen zien is komen veel berekende waarden goed overeen met de echte waarden, maar zijn er ook veel te veel outliers. Dit kunnen we verklaren door het feit dat highlights verspreid kunnen liggen over verschillende cellen waardoor een verkeerde lijn soms onterecht als lijn wordt



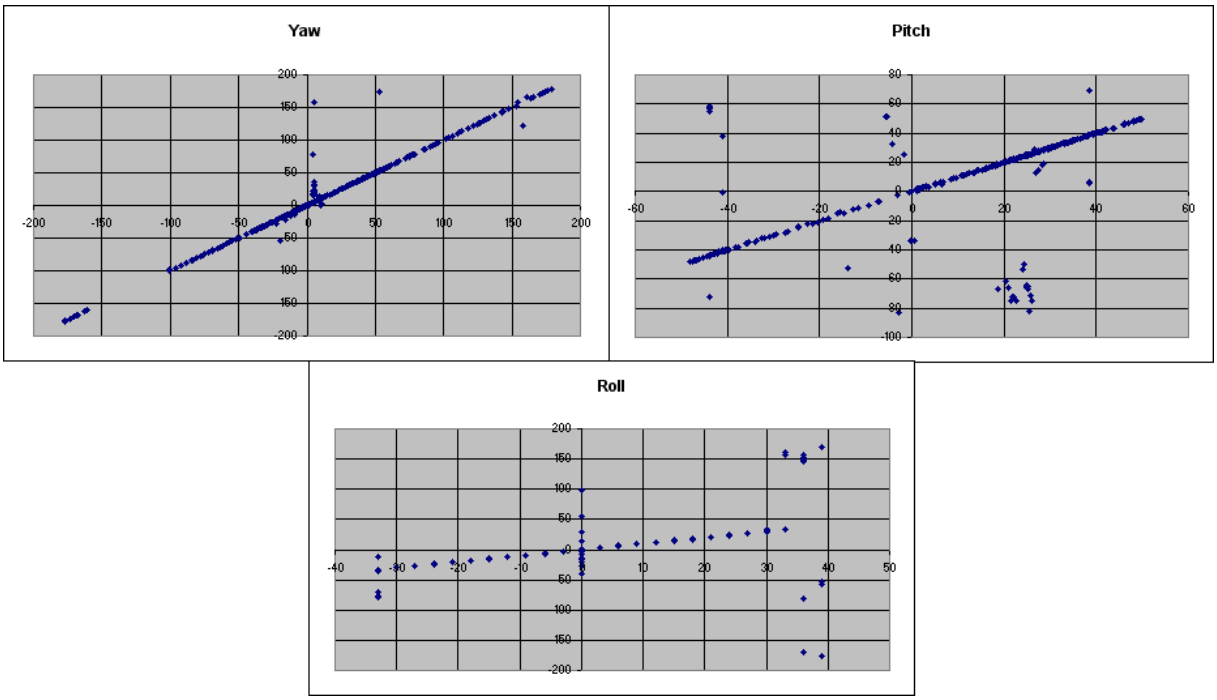
Figuur 8.4: Deze grafieken zetten de berekende Yaw, Pitch en Roll van het gewone algoritme van Hough uit tegenover de echte waarden.

beschouwd. We hebben ook gemerkt dat de berekeningstijd nodig om een beeld te verwerken rond de 30 - 40 ms lag. Dit is te traag voor een real-time tracker. Kortom deze implementatie was niet goed en snel genoeg om een betrouwbare en robuuste tracker te bouwen.

8.2.2 Geoptimaliseerde Hough transformatie met zoekvenster

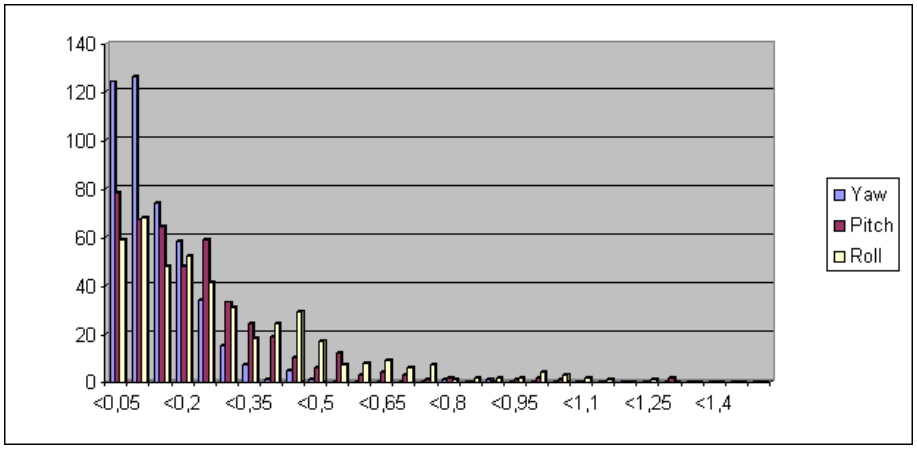
Een tweede poging om een goede oriëntatietracker te bouwen verbeterde de vorige Hough transformatie door beter de highlights te detecteren. Dit deden we door een soort blur-functie toe te passen op de Hough ruimte zodat er

ook naar de naburige cellen werd gekeken. Hierdoor konden we de cellen nog kleiner maken (nl. 0.25°) waardoor het algoritme nauwkeuriger werd. Dit gaat wel ten koste van de snelheid, die hierboven al niet zo snel was. Daarom werd de Hough transformatie versneld door enkel een kleiner cyclisch zoekwindow te beschouwen. Dit gaf redelijk goede oplossingen, zowel in de virtuele als in de reële opstelling.



Figuur 8.5: Deze grafieken zetten de berekende Yaw, Pitch en Roll van het geoptimaliseerde algoritme van Hough met zoekvensters uit tegenover de echte waarden.

In figuur 8.5 zien we de berekende oriëntatiehoeken uitgezet tegen de echte waarden in deze implementatie. We zien opnieuw heel wat outliers. Als we naar de foutenmarge gaan kijken (figuur 8.6) zien we dat de bekomen hoeken in de meeste gevallen slechts een afwijking van 0.5° hebben. We zien ook dat de yaw accurater is dan de pitch en de roll, omdat yaw de rotatie is evenwijdig met het vlak.
De snelheid van dit algoritme hangt wel sterk af van de oriëntatie omdat wanneer de hoeken t.o.v. het vlak groter worden, het zoekwindow meer

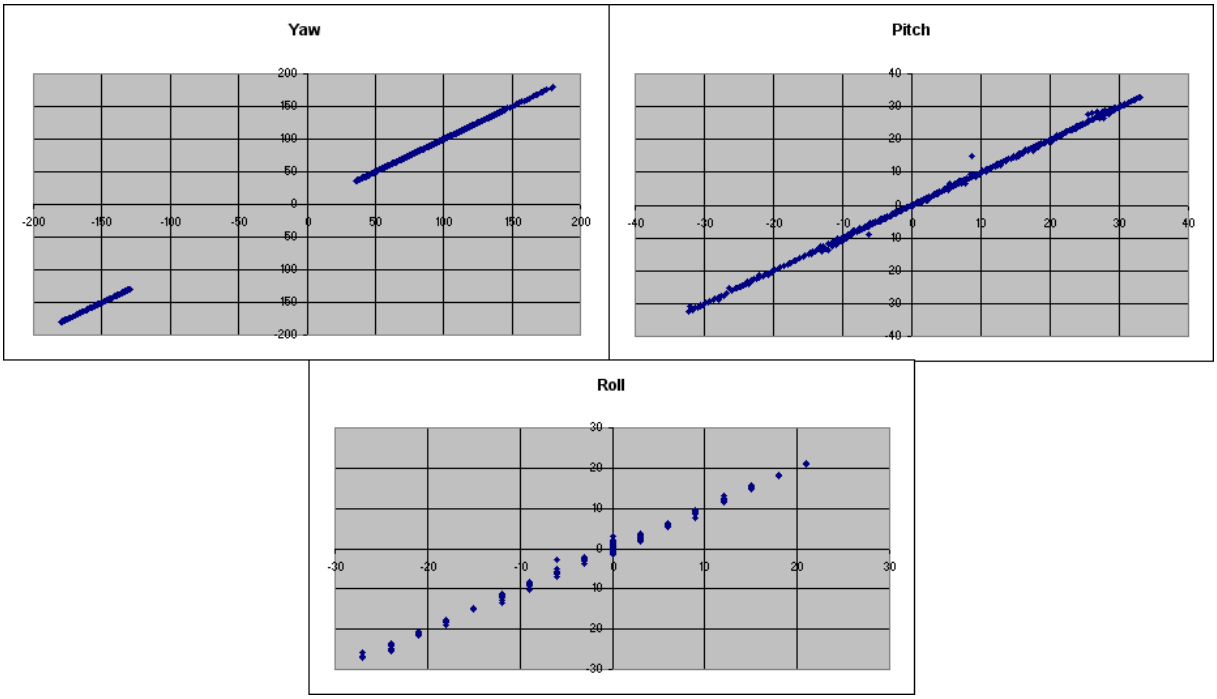


Figuur 8.6: In deze grafiek zien we hoeveel van de 486 metingen er binnen een bepaalde error grens liggen. Een 40-tal outliers zijn niet meegenomen in deze grafiek.

vierkantig en groter wordt. De gemeten snelheden lagen tussen de 6 en 15 ms. Dit is al voldoende voor een robuuste tracker, maar zowel de accuraatheid als de snelheid konden nog beter.

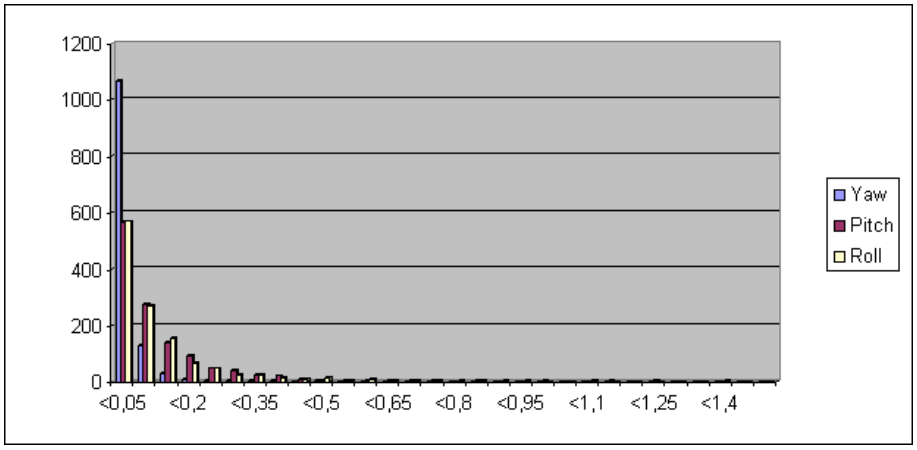
8.2.3 Hough transformatie met cirkels

De enige manier waarop we de vorige implementatie nog aanzienlijk konden versnellen, en tegelijk ook nog wat accurater te maken, is door gebruik te maken van de cirkelparametrisatie en analytische snijpunten. Deze implementatie maakt gebruik van het algoritme uitgelegd in sectie 5.3. Vanaf deze implementatie werd er in de virtuele implementatie gebruik gemaakt van textures voor de markeringen. Dit heeft als gevolg dat het moeilijker is om de doelen te detecteren, maar dit is ook realistischer t.o.v. een echte opstelling. In figuur 8.7 zien we het resultaat van deze implementatie. De foutenmarge op de berekende waarden is gevisualiseerd in figuur 8.8. Daaruit blijkt dat meer dan 97% van de metingen van de Yaw een foutenmarge onder de 0.2° heeft. Voor de rotatiehoeken Pitch en Roll liggen meer dan 80% onder de 0.2° grens.



Figuur 8.7: Deze grafieken zetten de berekende Yaw, Pitch en Roll van de Hough transformatie met cirkels uit tegenover de echte waarden.

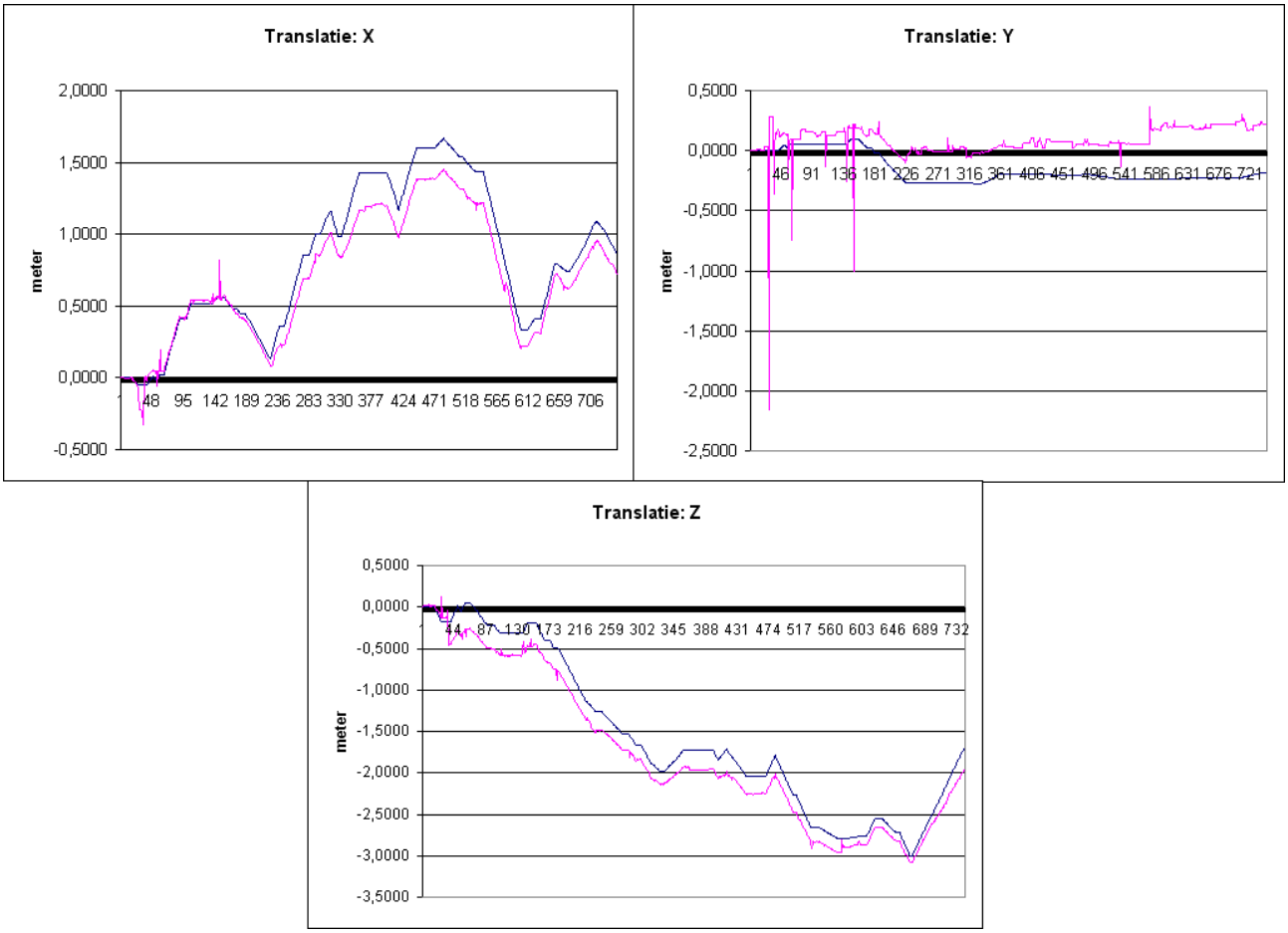
Wat we krijgen is een zeer robuuste implementatie van de tracker die accuraat genoeg is om te concurreren met vele commerciële oriëntatietrackers. Bovendien is deze implementatie zeer snel, nl. het verwerken van een beeld neemt in de virtuele testopstelling niet meer dan 1 ms in.



Figuur 8.8: In deze grafiek zien we hoeveel van de 1269 metingen er binnen een bepaalde error grens liggen. Slechts 2 outliers zijn niet meegenomen in deze grafiek.

8.2.4 Positie tracking

In deze implementatie maken we gebruik van de berekende rotatiematrix om de translatie te berekenen. Deze implementatie gebruikt de bepaling van translatie d.m.v. het algoritme in hoofdstuk 7.
In figuur 8.9 worden de eerste resultaten van de positietracker getoond. We zien dat de berekende translaties de echte redelijk goed volgen. Het valt wel op dat de Y-component, de richting loodrecht op het vlak, meer last heeft van translatieschommelingen dan de andere richtingen. Dit komt omdat de diepte informatie berekent uit 1 camerastandpunt onnauwkeurig is in de kijkrichting. Wel zijn er hier en daar zeer grote uitschieters die nog opgelost moeten worden. We zien ook dat de berekende posities beginnen af te wijken van de echte waardes. Dit zou men kunnen oplossen door af en toe opnieuw te calibreren via een 'vast' punt.
De implementatie lijkt bijna niets te merken van het toevoegen van het positie tracking algoritme. De CPU-tijd komt nauwelijks boven de 1 ms.

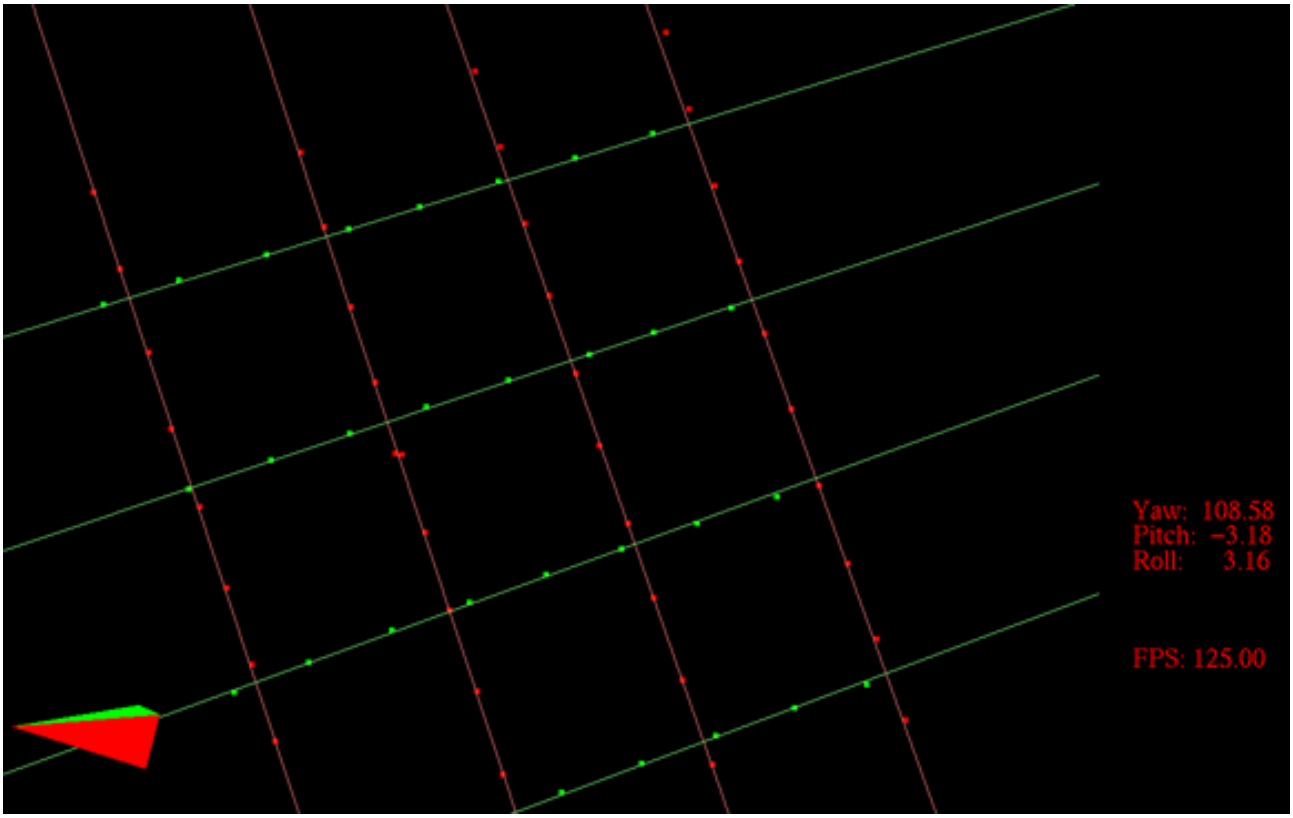


Figuur 8.9: Deze grafieken zetten de berekende (X, Y, Z) positie waardes (roze) uit tegenover de echte waarden (blauw) in functie van de tijd.

8.3 Resultaten reële opstelling

Om het algoritme ook in de praktijk te testen hebben we een echte scène gebouwd (figuur 8.3). Deze wordt gefilmd met een wide-angle camera die beelden van 1024x768 maakt. Omdat we de gevonden hoeken niet kunnen vergelijken met een echte waarde, vergelijken we enkel de snelheden.
Als eerste hebben we de geoptimaliseerde Hough transformatie met zoekvenster geïmplementeerd. Een screenshot van de interface van de implementatie

zien we in figuur 8.10. We zien debug informatie uitgeprint zoals de markeringen en lijnen die we gevonden hebben. We zien ook dat er van de lensdistortie niets meer te merken is. Het algoritme deed er 8 ms over om een camera beeld te verwerken en de oriëntatie uit te schrijven. Maar dit hing zoals de virtuele implementatie (8.2.2) af van de hoek van de camera met het bord zodat de verwerkingstijd ook wel tot 20 ms kon stijgen.
De tweede implementatie had hier geen last van, nl. de Hough transformatie met cirkels. Deze gaf ook zienbaar minder jitter dan de vorige implementatie en had maar een rekenkost tussen de 3 en 5 ms. Dit is al een mooi resultaat voor een goedkope tracker.



Figuur 8.10: De interface van onze implementatie in een reële opstelling.

8.4 Discussie

Een eerste opmerking die we hebben wanneer we de virtuele en reële implementatie vergelijken is het verschil in snelheid van hetzelfde algoritme. De reden waarom de echte implementaties trager werken als deze in de virtuele scène is niet alleen omwille van een grotere resolutie. Door ruis en distorties die bij elke camera voorkomen kunnen ruispixels onterecht als markeringspixels aanzien worden. Het verwerken (clusteren, ...) van deze ruispixels neemt ook enige tijd in beslag voordat deze genegeerd kunnen worden.
We zien ook meer en andere problemen opduiken in de echte opstelling tegenover de virtuele testscène. Een duidelijk voorbeeld hiervan is bijvoorbeeld de lensdistortie waarvan er in de virtuele opstelling geen sprake is. Ook het segmenteren van de markeringen gebeurt moeizamer doordat deze geen mooie egale kleur hebben en sterk afhangen van de lichtintensiteit in de kamer.

De eerste resultaten van de positietracker geven al aan dat we op de goede weg zijn om een goede positietracker te maken. Maar dan moeten de outliers zeker eruit gefilterd worden en moeten de berekende translaties beter uitgemiddeld worden. Ook moet er nog nagedacht worden over calibratiepatronen die de drift proberen tegen te gaan.

Ten slotte willen we de aandacht trekken op de qualitatieve resultaten in de virtuele opstelling. Vooral bij de laatste implementatie van de oriëntatie tracker zien we een zeer robuuste, snelle en vooral accurate tracker. Zeker als we in ons hoofd houden dat dit een zeer goedkope opstelling is zonder speciale apperatuur en dat deze uitvoer nog pure ongefilterde tracker data is. Als we zien dat een dikke 80% van de resultaten maar een foutenmarge heeft kleiner dan 0.2°, dan kunnen we stellen dat onze implementatie kan concurreren met vele trackers die er op dit moment op de markt zijn.

Hoofdstuk 9

Besluit

In deze thesis hebben we een nieuw goedkoop optisch trackingsysteem gebouwd dat zeker op oriëntatie gebied met zijn snelheid, robuustheid en accuraatheid zou kunnen concurreren met andere veel duurdere commerciële trackers. Om dit aan te tonen hebben we zowel een virtuele als een reële opstelling gebouwd die al zeer goede resultaten genereerden. Ook hebben we in deze thesis een nieuwe parametrisatie voor de Hough transformatie voorgesteld die voor kleine hoeveelheden punten een snel resultaat oplevert zonder zijn robuustheid te verliezen.

9.1 Samenvatting

We zullen hier nog even kort de stappen overlopen die we genomen hebben om tot een trackingsysteem te komen en wat we in elke stap geleerd hebben.

- **Hoofdstuk 2: Stand van zaken**
  We zijn begonnen met te kijken naar soorten tracking systemen die het tracking probleem oplossen. Hieruit hebben we besloten dat we een inside-looking-out optisch syteem wilden construeren. Om dit probleem op te lossen hebben we vervolgens gekeken naar camera calibratie systemen en technieken die momenteel gebruikt worden. Deze hebben we dan geclassificeerd in een aantal groepen zoals 'Feature' en 'Optical flow' gebaseerde methodes. We hebben ook gekeken naar incrementele oplossingen en hoe het probleem in een aantal makkelijkere deelproblemen opgesplitst kunnen worden. Ten slotte werden een aantal criteria opgesteld om methodes met elkaar te vergelijken. We merken

ook op dat niet alle camera calibratie methodes even geschikt zijn om een tracker systeem te bouwen omwille van bijvoorbeeld efficiëntie of uitbreidbaarheid.

- **Hoofdstuk 3: Overzicht opbouw van de tracker**
  In dit hoofdstuk hebben we de vereisten van onze tracker gespecifieerd alsook de algemene opstelling van een scène. We hebben hier ook een overzicht gegeven van de stappen die de tracker ondergaat en hoe deze stappen met elkaar samenhangen.

- **Hoofdstuk 4: Detectie van markeringen en optical flow**
  In de eerste stap hebben we besproken hoe dat we de markeringen die we in de scène bevestigd hebben terug kunnen vinden in ons beeld. We hebben hier geopteerd om de kleurdetectie in de HSV kleurruimte te implementeren omwille van zijn beter scheiding tussen kleurwaarde en lichtintensiteit. We hebben hier ook een methode besproken om markeringspixels op een redelijk efficiënte manier te clusteren. Anderzijds hebben we ook gezien hoe we de optical flow van de markeringen kunnen bepalen en hebben we het probleem van lensdistortie besproken.

- **Hoofdstuk 5: Herkenning van lijnpatronen**
  De tweede stap bestaat uit het zoeken van lijnpatronen uit de markeringen die we in de eerste stap gevonden hebben. We zien dat het 'Brute Force' algoritme een te hoge rekenkost nodig heeft en daarom zijn we overgestapt naar de robuustere Hough transformatie. We hebben de Hough transformatie voor lijnpatronen besproken die algemeen in gebruik is en aangetoond dat door ons beperkt aantal punten een efficiëntere en accuratere oplossing mogelijk is. Daarom hebben we nieuwe Hough transformatie voorgesteld die werkt op basis van een cirkelparametrisatie. Ten slotte hebben we ook de relatie tussen de nieuwe Hough transformatie en deze op basis van sinusparametrisatie aangetoond.

- **Hoofdstuk 6: Berekening van oriëntatie uit vluchtpunten**
  Wanneer we de lijnpatronen uit de vorige stap berekent hebben, kunnen we in deze stap de oriëntatie bepalen uit de vluchtpunten van deze lijnen. Eerst en vooral moesten we het 'best fit' vluchtpunt van een set van lijnen berekenen. Vervolgens hebben we dan de relatie tussen de beeld- en wereldcoördinaten van een vluchtpunt bepaald waardoor we

zagen dat deze enkel afhankelijk was van de rotatie van de camera en niet de translatie. Dit wil zeggen dat we de rotatie onafhankelijk van de translatie kunnen berekenen uit vluchtpunten. We hebben ook getoond hoe dat we, gegeven de 3 X-, Y- en Z-vluchtpunten, de rotatiematrix kunnen opstellen. Daarna toonden we ook aan dat we slechts 2 vluchtpunten nodig hadden door gebruik te maken van de eigenschappen van de rotatiematrix. Ten slotte bemerkten we dat de rotatie absoluut bepaald is op een aantal ambiguïteiten na die we konden oplossen door naar het vorige frame te kijken.

- **Hoofdstuk 7: Bepaling van translatie**
  In de laatste stap van de tracker hebben we gezocht naar een manier om de translatie te bepalen, gegeven de rotatiematrix uit de vorige stap. Dit hebben we gedaan door gebruik te maken van puntovereenkomsten in 2 opeenvolgende camerastandpunten. Deze 3D camerapunten zijn enkel bepaald op een schaalfactor na. Daarom moeten we deze schaalfactoren calibreren m.b.v. de vaste afstand tussen 2 opeenvolgende markeringen. Om er zeker van te zijn dat de markeringen op een lijn wel degelijk op elkaar volgen, hebben we een controle voorgesteld die controleert op ontbrekende punten d.m.v. de dubbelverhouding.

- **Hoofdstuk 8: Resultaten en discussie**
  Ten slotte in het laatste hoofdstuk hebben we onze implementatie van de algoritmes uit vorige hoofdstukken uitgetest. We hebben beschreven hoe we zowel een virtuele als een reële testopstelling geconstrueerd hebben. In elk van deze opstellingen hebben we ons trackingssysteem geëvalueerd. Hier hebben we vooral gekeken naar de snelheid, robuustheid en accuraatheid van de geïmplementeerde algoritmes. In de reële opstelling hebben we enkel de snelheid kunnen meten omdat we geen referentie hebben wat de echte waarden op een bepaald moment zijn. We zien vooral dat de laatste implementatie van de oriëntatietracker een zeer robuust resultaat geeft met een foutenmarge van 0.2° waardoor deze zou kunnen concurreren met de huidige commerciële trackers. Ook hebben we gezien dat de snelheid drastisch gestegen is tegenover de eerste implementatie. De positietracker geeft al redelijke resultaten, maar er is nog ruimte voor verbeteringen om een stabielere implementatie te bekomen.

9.2 Toekomstig werk

Vooral bij de methode voor het bepalen van de positie is er nog ruimte voor verbeteringen en uitbreidingen. De uitvoer van de translatie moet stabieler worden. Zo kan er misschien eens gekeken worden of de translatie berekenen over een aantal frames betere resultaten geeft. Ook zouden er 'vaste' markeringen in de wereld gehangen kunnen worden waarvan de positie geschat wordt zodat de drift van de positie tegengegaan kan worden. Deze zouden ook kunnen dienen om de ambiguïteiten van de translatievector te verbeteren. Met deze ambiguïteiten bedoelen we het repetitieve karakter van de markeringen waardoor een verplaatsing gelijk aan de grote van het calibratiepatroon eenzelfde translatievector geeft als dat er niet bewogen wordt. Het plaatsen van een aantal calibratiemarkeringen vergroot gewoon de herhalingsafstand.
Nog een mogelijke uitbreiding is het implementeren van filters die de rauwe tracker output filteren om een smoother verloop van de beweging te hebben. Dit kan de nauwkeurigheid ook ten goede komen, hoewel dit soms wel voor een kleine latency kan zorgen.
Een occlusie detectie systeem kan ook handig zijn om bijvoorbeeld een waarschuwing te geven dat de tracker data onbetrouwbaar kan zijn. Zo kan hiermee rekening gehouden worden door de gemeten waarden te negeren, of indien we meerdere camera's op een object bevestigen kunnen de andere camera's de tracking tijdelijk overnemen van de 'blinde' camera.

Bibliografie

[1] Gary Bishop B. Danette Allen and Greg Welch. Tracking: Beyond 15 minutes of thought. In *SIGGRAPH - course 11*, Los Angeles, California, USA, 2001. ACM Press. Proceedings of the 28st Annual Conference Computer Graphics and Interactive Techniques.

[2] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43 – 77, February 1994.

[3] T J Broida and R Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(1):90–99, 1986.

[4] Anna R. Bruss and Berthold K.P. Horn. Passive navigation. Technical report, Massachusetts Institute of Technology, 1981.

[5] B. Caprile and V. Torre. Using vanishing points for camera calibration. *Int. J. Comput. Vision*, 4(2):127–140, 1990.

[6] Yang Chunke, Shunichiro Oe, and Kenji Terada. Estimation of three-dimensional motion information from optical flow using subspace method. *Transactions of The Institute of Electrical Engineers of Japan*, 121(7):1187–1194, Juli 2001.

[7] Olivier D. Faugeras, Quang-Tuan Luong, and Stephen J. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pages 321–334, 1992.

[8] Olivier D. Faugeras and Stephen J. Maybank. Motion from point matches: multiplicity of solutions. In *Proceedings of the Workshop on Geometry and Robotics*, pages 172–193, London, UK, 1989. Springer-Verlag.

[9] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed. in C)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[10] Menno Genseberger. *Domain decomposition in the Jacobi-Davidson method for eigenproblems*. PhD thesis, Universiteit Utrecht, 2001.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[12] David J. Heeger and Allan D. Jepson. Subspace methods for recovering rigid motion i: algorithm and implementation. *Int. J. Comput. Vision*, 7(2):95–117, 1992.

[13] Berthold K. P. Horn and Jr. E. J. Weldon. Direct methods for recovering motion. *Artificial intelligence at MIT: expanding frontiers*, pages 436–471, 1990.

[14] Stephen J. Maybank and Olivier D. Faugeras. A theory of self-calibration of a moving camera. *Int. J. Comput. Vision*, 8(2):123–151, 1992.

[15] T. Drummond R. Cipolla and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. In *Proc. British Machine Vision Conference*, University of Nottingham, 1999. Proceedings of the 10th British Machine Vision Conference.

[16] Parvaneh Saeedi, Peter D. Lawrence, and David G. Lowe. 3d motion tracking of a mobile robot in a natural environment. In *ICRA*, pages 1682–1687, 2000.

[17] Alvy Ray Smith and Eric Ray Lyons. Hwb - a more intuitive hue-based color model. *journal of graphics tools*, 1(1):3–17, 1996.

[18] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. pages 221–244, 1922.

[19] Shimon Ullman. Maximizing rigidity: The incremental recovery of 3-d structure from rigid and rubbery motion, June 1983.

[20] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. Highperformance wide-area optical tracking: The hiball tracking system, 2001.

[21] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[22] Greg Welch and Gary Bishop. SCAAT: Incremental tracking with incomplete information. *Computer Graphics*, 31(Annual Conference Series):333–344, 1997.

Nuttige links

- Veel interessante info en demonstraties van gyroscopen
  http://www.gyroscopes.org
- Een geanimeerd voorbeeld van de werking van de Inertial GyroChip
  http://www.systron.com/tech.asp
- Elektromagnetisme
  http://www.ham-radio.nl/cursus/elektromagnetisme.htm
- Nieuw optisch tracking systeem op basis van microlensdisplays
  http://www.thirdeyeinterfaces.com
- Kenmerk gebaseerde tracking in ARToolkit
  http://www.hitl.washington.edu/artoolkit/
- Java applets die de omzetting tussen verschillende kleurruimtes visualiseren
  http://www.cs.rit.edu/~ncs/color/a spaces.html
- Lensdistorties
  http://www.dvinfo.net/articles/optics/lensdefects.php
  http://www.dgp.toronto.edu/~hertzman/courses/csc418/winter_2003 /effects/distortion.html
- Snijpunt tussen cirkels
  http://astronomy.swin.edu.au/~pbourke/geometry/2circle/
- Best fit vluchtpunten berekenen
  http://www-2.cs.cmu.edu/~ph/869/www/notes/vanishing.txt
- Newmat C++ matrix library
  http://www.robertnz.net/nm_intro.htm

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen en uw akkoord te verlenen.

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Computervisie gebaseerde positie en orientatie tracking
Richting: **Master in de informatica** Jaar: **2006**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Deze toekenning van het auteursrecht aan de Universiteit Hasselt houdt in dat ik/wij als auteur de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij kan reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

U bevestigt dat de eindverhandeling uw origineel werk is, en dat u het recht heeft om de rechten te verlenen die in deze overeenkomst worden beschreven. U verklaart tevens dat de eindverhandeling, naar uw weten, het auteursrecht van anderen niet overtreedt.

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

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

Ik ga akkoord,

Steven MAESEN

Datum: