Universiteit Hasselt

Bachelorproef

Augmented Reality for Workbenches

Auteur:
Nick Michiels
(0623764)

Promotor
Prof. Dr. Philippe Bekaert
Begeleiders
Tom Cuypers
Yannick Francken

Eindwerk voorgedragen tot het behalen van de graad van bachelor in de informatica/ICT/kennistechnologie

Academiejaar 2008-2009

Abstract

Deze thesis beschrijft technieken om een interactieve werkbank te maken. De interactie werkt op basis van herkenning en manipulering van objecten. Meer specifiek het herkennen van een blad papier om er vervolgens op te projecteren. Dit alles in een realtime omgeving. De realisatie gebeurt in drie fases: een calibratie stap, een segmentatie stap en een computer visie stap.

De calibratie is een essentiële stap. Hij is verantwoordelijk voor het werken in de juiste ruimte. Als een object in het camerabeeld wordt waargenomen, kan het pas gemanipuleerd worden met een projectie als ook geweten is waar het overeenkomstige object zich in het projectievlak bevindt. Het doel van calibratie is een mapping te voorzien tussen camera- en projectorcoördinaten. Deze thesis stelt hiervoor twee technieken voor. Een eerste techniek maakt gebruik van graycode patterns. Het projecteren van een aantal patronen geeft informatie om de mapping te verkrijgen. Naast deze techniek wordt er een tweede oplossing aangeboden, namelijk het teniet doen van degradaties op het camerabeeld door deze te corrigeren. Via een radiale en projectieve correctie wordt vanuit het camerabeeld terug overgegaan naar een overeenkomstige scene in projectorcoördinaten. Het tweede onderwerp van deze thesis is segmentatie. Segmentatie wordt gebruikt om het papier te herkennen. Twee besproken technieken hiervoor zijn Hough transformatie en RANSAC. Beide worden gebruikt voor detectie van bladranden. Na de herkenning van het papier zal in stap 3 computer visie worden gebruikt om het papier te tracken. Dit tracken gebeurt met een particle filter, meer bepaald een condensation filter.

Na implementatie zal blijken dat RANSAC een robuuste manier is om de lijnen van het blad te detecteren. Als eenmaal het blad is herkend zorgt de condensation filter voor een snelle detectie van translatie en/of rotatie. Met een beperkt aantal iteraties zal de condensation filter snel werk leveren en hierbij enkel nog segmentatie nodig hebben om de nauwkeurigheid terug bij te stellen. Het voordeel van de besproken methodes is dat ze stuk voor stuk een goede kandidaat zijn voor implementatie op de GPU. Hierdoor kunnen de huidge resultaten in de toekomst nog enorm worden geoptimaliseerd.

Woord vooraf

Veel dank aan iedereen die mij gesteund heeft bij het schrijven van deze thesis. Hierbij wil ik uitdrukkelijk mijn promotor Philippe Bekaert en begeleiders Tom Cuypers en Yannick Francken bedanken voor hun achtergrondkennis en vele goede tips. Ik wil hen ook bedanken voor de mogelijkheid tot praktische realisatie van mijn bachelorthesis.

Ook mijn ouders wil ik graag bedanken om mij de mogelijkheid te geven om deze bachelorthesis te kunnen schrijven en mij al die jaren de kans te geven om verder te studeren.

Vervolgens bedank ik ook mijn vriendin die me gedurende heel mijn bachelor opleiding gesteund heeft.

Tot slotte wil ik ook nog al mijn studiegenoten bedanken bij wie ik altijd terecht kon met problemen.

Inhoudsopgave

Lijst van figuren

Hoofdstuk 1

Inleiding

Het doel van deze thesis is een basis te ontwikkelen voor interactieve communicatie met een werktafel. Algemeen zal met behulp van één gecalibreerde camera bepaalde objecten op de werktafel worden herkend en vervolgens aangepast door er op te projecteren. Om dit te realiseren is er een calibratie, segmentatie en computervisie techniek nodig. Deze thesis zal zich niet fixeren op het herkennen van verschillende soorten objecten, maar eerder een basis proberen te leggen voor een augmented reality systeem. Voorlopig wordt er enkel een papier herkend.

1.1 Doelstelling

Door het gebruik van één camera, één projector en een tafel willen we een papier kunnen herkennen op de tafel. De projector moet op dit papier kunnen projecteren. Een kritisch punt hierbij is de realtime verwerking ervan.

1.2 Augmented Reality

Binnen het domein van augmented reality tracht men nieuwe computer interfaces te ontwikkelen die veel intuïtiever zijn dan de reeds bestaande. In plaats van informatie te tonen op computerschermen gaat men de data tonen waar men ze verwacht, namelijk in de echte wereld. Door de informatie op een intuïtieve manier te versmelten met een echte wereld zal de gebruiksvriendelijkheid enorm hoog liggen [Wag07]. Een vaste definitie van augmented reality bestaat niet, maar één die veel gebruikt wordt, is deze van Ronald T. Azuma [Azu97a]. Volgens deze definitie moet een augmented reality system aan de volgende karakteristieken voldoen:

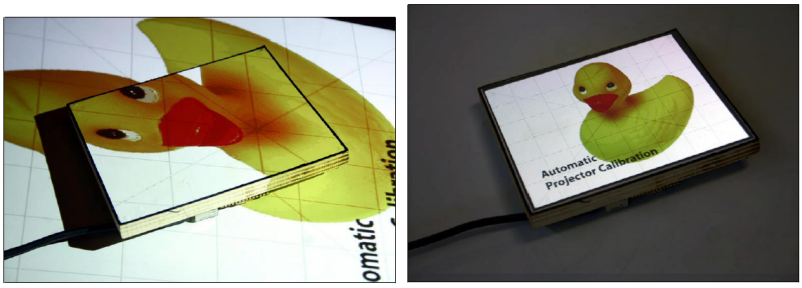
1. Combinatie van de reële wereld met een virtuele wereld.
2. Interactie moet realtime kunnen.
3. Functioneren in 3D.

Films zoals Jurassic Park zijn een samensmelting van een reële en virtuele wereld in 3D. Toch valt deze niet onder de noemer van augmented reality omdat het geen interactieve media is [Azu97b].

Enkele voorbeelden van augmented reality systemen zijn de gekleurde lijn die het traject van de puck laat zien bij ijshockey; de simulaties van vliegtuigen en auto's; medische hulp bij operaties waar de chirurg niet met het blote oog kan opereren en toepassingen voor onderhoud van grote machines [Wik97]. Binnen augmented reality komt het domein computer visie ook veel aan bod. Objecten moeten eerst worden herkend alvorens deze te kunnen aanpassen.

1.3 Bestaand werk

Deze thesis is geïnspireerd door het onderzoek van Johnny C. Lee. Hij heeft een augmented reality systeem ontworpen om allerlei objecten in een 3D wereld te herkennen en te wijzigen. Met behulp van infrarood ledjes en lichtsensoren gaat hij de plaats van het object te weten komen en zo met een projector het object aanpassen door er een figuur op te projecteren. Zijn onderzoek is te lezen in Automatic Projector Calibration with Embedded Light Sensors [Lee04a]. Een voorbeeld van deze techniek kunt u zien in Figuur 1.1. Het grote verschil tussen zijn onderzoek en het onderzoek in deze thesis is dat hij infrarood trackers gebruikt. In deze thesis zal de herkenning van objecten gebeuren via één camera en zonder aangebrachte markers op het object. Een tweede verschil is dat zijn techniek niet realtime werkt. In deze thesis wordt gepoogd de projectie real time uit te voeren.



Figuur 1.1: Onderzoek van Johnny Lee. In zijn onderzoek maakt hij gebruik van infrarood markers die hij met behulp van een infrarood camera gaat detecteren. Op basis van de vier punten kan hij bepalen in welk vlak en op welke grootte hij moet projecteren. De rechtse afbeelding geeft het resultaat [Lee04a].

1.4 Overzicht

Deze thesis is onderverdeeld in 4 hoofdstukken. Het volgende hoofdstuk zal enkele bestaande technieken toelichten met betrekking tot calibratie van de camera en objectherkenning. Deze technieken zullen dan in een verder hoofdstuk (praktische implementatie) al dan niet worden toegepast. Vooral de toepasbaarheid en de efficiëntie van de technieken zullen worden getoetst, alsook een discussie over het globale proces. Vervolgens zal er in een klein hoofdstuk conclusies worden getrokken. In een afsluitend hoofdstuk zullen toekomstige toepassingen van deze thesis worden besproken.

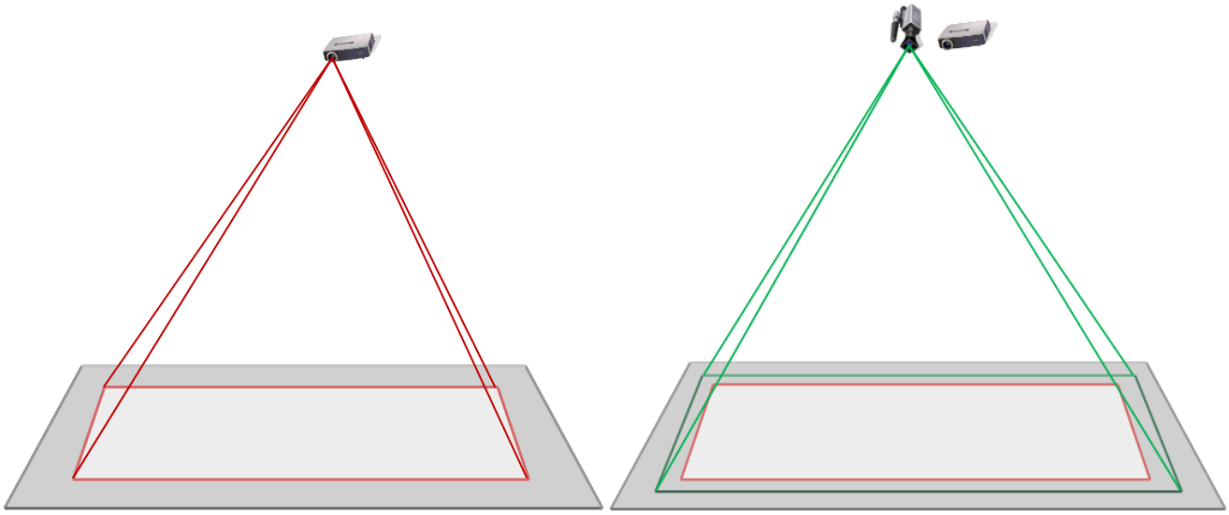
Hoofdstuk 2

Gerelateerd werk

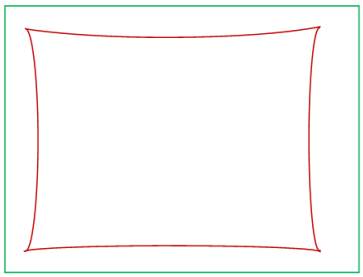
2.1 Calibratie

Binnen dit domein van augmented reality is calibratie een essentiële stap. Er zijn namelijk twee coördinatenstelsels aanwezig: het coördinatenstelsel van het geprojecteerde beeld (of nog het coördinatenstelsel dat overeenkomt met de werktafel) en het coördinatenstelsel van het inkomende camerabeeld. Binnen deze sectie wordt er beschreven hoe deze twee stelsels op elkaar kunnen gemapped worden. Het probleem wordt eerst uitgebreid geschetst. In de secties die daarop volgen zullen enkele technieken worden aangehaald om deze calibratie te verwezenlijken.

Het probleem De opstelling bestaat uit een projectievlak (bv. een tafel), een projector en een camera. Voor de eenvoudigheid wordt er beschouwd dat de camera en projector loodrecht op het projectievlak staan. De projector projecteert op de werktafel. De camera is zo geplaatst dat het hele geprojecteerde beeld waarneembaar is. De opstelling is geïllustreerd in Figuur 3.1. De groene rechthoek op het projectievlak komt overeen met het camerabeeld, de rode met het geprojecteerd beeld. In de realiteit zal het geprojecteerde beeld binnen het camerabeeld niet mooi rechthoekig zijn. Dit fenomeen is afkomstig van de camera. De bolvormigheid van de lens zal het beeld een radiale verstoring geven. Deze radiale verstoring noemt men ook wel een fishbowl effect. Het beeld dat binnenkomt, zal eerder lijken op Figuur 2.2.



Figuur 2.1: De projector projecteert het beeld op de werktafel zoals in de linkse figuur. De camera filmt het projectievlak, hierbij rekening houdend dat het rode vlak (het geprojecteerde beeld uit de linkse afbeelding) volledig binnen het groene vlak ligt.



Figuur 2.2: Omdat de lens van de camera bolvormig is, zal er op elke frame van de camera een radiale verstoring aanwezig zijn. De groene rechthoek is de frame afkomstig van de camera. De rode lijnen geven aan hoe het geprojecteerd beeld zichtbaar is op het camerabeeld. Er is dus duidelijk een radiale vervorming aanwezig.

Uit deze opstelling kunnen twee coördinatenstelsels gedefinieerd worden. Beschouw eerst het geprojecteerde beeld. Het centrum van dit assenstelsel ligt in de linkerbovenhoek. De Y-as loopt van links naar rechts over een bereik van $[0, 800[$. De X-as loopt van boven naar onder over een bereik van $[0, 600[$. Dit is ook het coördinatenstelsel dat uiteindelijk gebruikt zal worden. Het zijn namelijk de posities waarop de beelden worden gerendered.

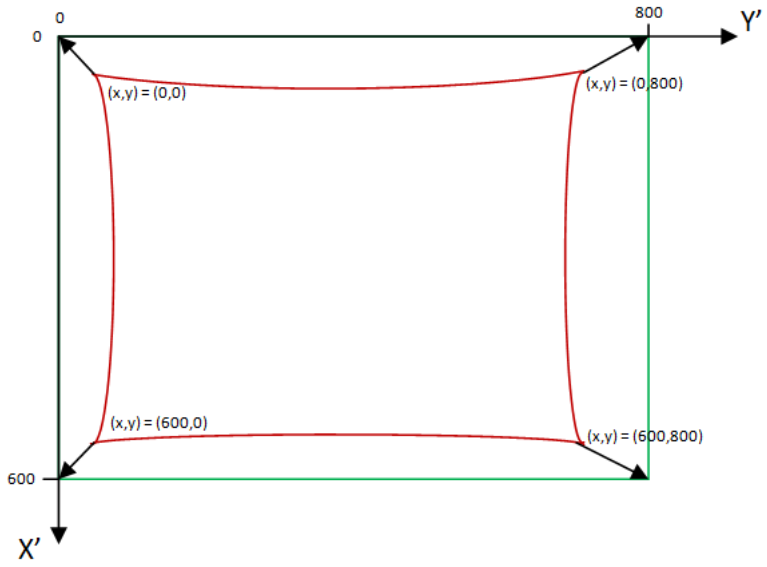
Veronderstel nu een tweede coördinatenstelsel, waarvan de oorsprong in de linkerbovenhoek van het camerabeeld ligt. Ook hier loopt de horizontale as Y' van links naar rechts over een bereik van $[0, 800[$ en de verticale as X' van boven naar onder over een bereik van $[0, 600[$.

De bedoeling van calibratie is voor elk punt van het camerabeeld te bepalen met wel punt het overeenkomt in het projectiebeeld. De waarneembare punten van het projectiebeeld binnen

het camerabeeld moeten dus met een functie gemapped worden over het hele bereik om zo voor elk punt de mapping van coördinaten te bekomen. Dit is gevisualiseerd in Figuur 2.3.

De radiale verstoring zal de calibratie aanzienlijk complexer maken. Het is niet voldoende om enkel de vier hoekpunten te nemen en deze vervolgens uit te rekken over het hele frame. Dit zal een te grote afwijking veroorzaken in de tussenliggende punten. Er zal dus gekozen worden voor een niet lineaire techniek.

In de volgende secties zullen enkele technieken worden beschreven om deze mapping te verwezenlijken.



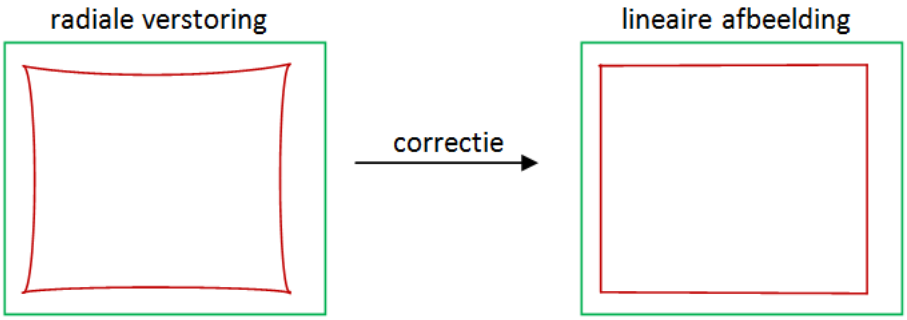
Figuur 2.3: Het geprojecteerde beeld heeft bijvoorbeeld een resolutie van 800 op 600, maar ook het camerabeeld heeft deze resolutie. Het probleem is dat het rode vlak maar een deel van het groene vlak is en in theorie hebben deze twee dezelfde resolutie. Met behulp van een calibratie techniek moeten we dus een mapping uitkomen die het rode vlak uitrekt over het groene vlak. Elk punt van het inkomend beeld zal dus een mapping krijgen op een punt in projectorcoördinaten.

2.1.1 Interpolatie tussen de vier hoekpunten

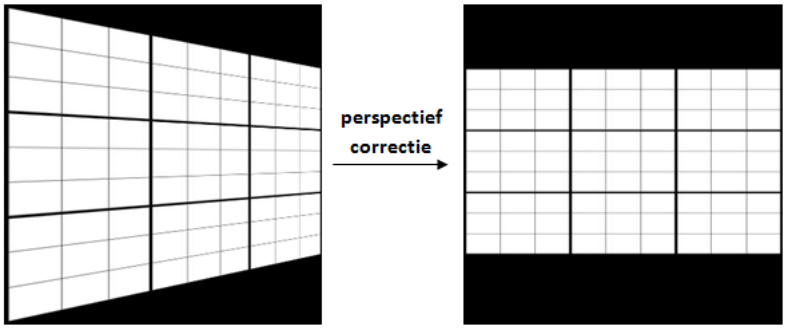
Calibratie met behulp van interpolatie geeft op een vrij eenvoudige manier de mapping. Bij deze werkwijze worden enkel de 4 hoekpunten van de projectie gebruikt. Deze zijn gemakkelijk te bepalen uit het opgenomen beeld met behulp van lijndetectie algoritmes (zie sectie 2.2). Deze gaan vervolgens omgevormd worden in projectorcoördinaten. Als dit is gebeurd, worden al de tussenliggend waardes geïnterpoleerd. Deze techniek kan een goede benadering geven voor elk punt in het beeld. Toch zal het zoals eerder aangehaald meestal een te grote afwijking geven. Vooral als de lens een te groot fishbowl effect teweeg brengt.

2.1.2 Interpolatie tussen de vier hoekpunten na antidegradatie

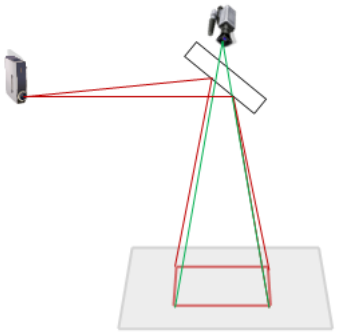
In essentie is deze methode dezelfde als de vorige. Het enige verschil is dat de afwijking zo klein mogelijk wordt gehouden door eerst nog een beeldrestoratie toe te passen. Zoals eerder aangehaald zorgt de camera voor een radiale verstoring. Door een model op te stellen voor degradatie kan er een inverse filtering worden toegepast om zo terug een lineaire afbeelding te verkrijgen [Har03a]. Dit proces is geïllustreerd in Figuur 2.4. Het is echter niet enkel de radiale verstoring die zorgt voor een te grote afwijking. In een ideale omgeving zal camera en projector in exact hetzelfde punt in de ruimte liggen. Het ligt voor de hand dat dit nooit geometrisch kan worden gerealiseerd. In de praktijk kan dit worden nagebootst met een beam splitter [Wik09]. Een opstelling met een beam splitter wordt besproken in Figuur 2.6. Indien de camera en projector niet in hetzelfde punt samenvalt, zal er typisch ook een projectie verstoring aanwezig zijn op het beeld. Ter illustratie Figuur 2.5. Beide modellen zullen nu worden besproken. Een algemene bespreking van degradatie modellen is beschreven door Rafael C. Gonzalez en Richard E. Woods [GW08a].



Figuur 2.4: Het beeld dat binnenkomt, heeft typisch een radiale verstoring. In de figuur wordt het camerabeeld in het groen voorgesteld en de projectie in het rood. Met behulp van een radiale correctie wordt dit zogenaamde fishbowl effect weggewerkt.



Figuur 2.5: Naast de radiale verstoring zal er ook een projectieve verstoring aanwezig zijn. Deze vervorming uit zich doordat *normaal* parallelle lijnen in de scene niet meer parallel zijn in de vervormde scene [Har03b]. Door middel van een omgekeerde degradatie wordt er een correctie bekomen.



Figuur 2.6: Om een projector en camera in de praktijk te laten samenvallen, kan gebruik worden gemaakt van een beam splitter. Een beam splitter kan beschouwd worden als een half doorlaatbare spiegel. De spiegel wordt onder een hoek van 45 graden gedraaid met de projector. De projectie zal via de spiegel weerkaatst worden op de werktafel. De camera kan vervolgens aan de achterkant van de spiegel geplaatst worden. Hij zal geen last ondervinden van weerspiegeling en door de achterkant door kunnen kijken alsof het glas is.

2.1.3 Radiale correctie

Model voor radiale verstoring [Har03a] Stel dat $(\tilde{x}, \tilde{y})$ de coördinaten zijn van het ideale beeld dat we willen verkrijgen en (x_d, y_d) de coördinaten van het gedegradeerde beeld. De projectie van het gedegradeerde punt is afhankelijk van het ideale punt en de radiale verplaatsing. Dus, de radiale verstoring is gemodelleerd als volgt

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \tag{2.1}$$

waarbij $(\tilde{x}, \tilde{y})$ het ideale beeld voorstelt, (x_d, y_d) het radiaal gedegradeerd beeld, $\tilde{r}$ de radiale afstand $\sqrt{\tilde{x}^2 + \tilde{y}^2}$ van het centrum van de radiale verstoring en $L(\tilde{r})$ de verstoringfactor, welke een functie is van enkel de $\tilde{r}$.

Model voor radiale correctie Om het beeld te corrigeren moet de omgekeerde bewerking worden toegepast. Op pixel niveau geeft dit

$$\tilde{x} = x_c + L(r)(x_d - x_c) \quad \tilde{y} = y_c + L(r)(y_d - y_c) \tag{2.2}$$

waarbij (x_c, y_c) het centrum van de radiale verstoring is, met $r^2 = (x_d - x_c)^2 + (y_d - y_c)^2$.

Keuze van verstoringfunctie en centrum van verstoring De verstoringfunctie $L(r)$ is enkel gedefinieerd voor positieve waardes van r en $L(0) = 1$. Meestal wordt de verstoringfunctie gegeven door een Taylor ontwikkeling $L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + \cdots$. De coëfficiënten voor de radiale correctie $\{k_1, l_2, k_3, \ldots, x_c, y_c\}$ zijn eigen aan de interne calibratie van de camera zelf. Meestal wordt het centrum van het videobeeld gekozen als centrum van verstoring [Har03a]. De waardes van de Taylor ontwikkeling kunnen met behulp van trail and error worden afgeschat

tot visueel een mooi resultaat wordt bekomen. Een betere manier is de benadering wiskundig te berekenen vanuit de beelden die binnenkomen om zo de fout minimaal te houden [Lab06]. Dit kan verwezenlijkt worden door vooraf een vast schaakbordpatroon te definiëren waarvan de afmetingen gekend zijn. Dit patroon wordt in een sequentie van beelden getoond in zoveel mogelijk posities en rotaties. Op basis van deze beelden wordt het patroon gezocht en vervolgens bepaald hoeveel afwijking het vertoont met het origineel patroon. Hieruit kan een goede benadering van de verstoring worden gevonden. Des te meer beelden worden gebruikt, des te nauwkeuriger de verstoring zal worden benaderd.

2.1.4 Projectieve correctie

Alvorens een model kan worden opgesteld voor de correctie, moet eerst een model worden opgesteld voor projectieve transformaties. Het is een model beschreven door Felix Klein [Kle39] en maakt gebruik van de homogene representatie van lijnen.

Homogene representatie van lijnen Een lijn in het vlak wordt voorgesteld met de vergelijking $ax + by + c = 0$. Verschillende keuzes van a, b en c resulteren in verschillende lijnen. Een lijn kan dus uniek voorgesteld worden door een vector (a, b, c). De omgekeerde relatie is echter niet uniek. Zo zijn de lijnen $ax + by + c = 0$ en $(ka)x + (kb)y + (kc) = 0$ hetzelfde voor een constante k, waarbij deze constante niet gelijk is aan 0. De twee vectoren (a, b, c) en $k(a, b, c)$ stellen dus dezelfde lijn voor. Een equivalente klasse van vectoren met deze relatie wordt een homogene vector genoemd. Een verzameling van equivalente klasses van vectoren in $\mathbb{R}^3$ zonder $(0, 0, 0)$ vormt de projectie ruimte $\mathbb{P}^2$ (vector $(0, 0, 0)$ wordt niet meegerekend omdat het geen lijn voorstelt) [Har03c].

Projectieve transformatie 2D projectie geometrie is de studie van eigenschappen van de projectie ruimte $\mathbb{P}^2$ die invariant blijven onder een groep van transformaties. Deze groep van transformaties worden ook wel *projectivities* genoemd. Een *projectivity* is een omkeerbare afbeelding van punten in $\mathbb{P}^2$ naar punten in $\mathbb{P}^2$. Dus anders gesteld is het een afbeelding van lijnen op lijnen.

Definitie 2.1.1 *Een projectivity is een omkeerbare afbeelding h van $\mathbb{P}^2$ naar $\mathbb{P}^2$ zodat de drie punten x_1, x_2 en x_3 op dezelfde lijn liggen als en slechts als $h(x_1)$, $h(x_2)$ en $h(x_3)$ ook op dezelfde lijn liggen.*

Andere namen voor een projectivity zijn *collineatie*, *projectieve transformatie* en *homography*. Hier zal voor de duidelijkheid de term projectieve transformatie worden gebruikt. Een meer algebraïsche definitie zal in tegenstelling tot definitie 2.1.1 meer werken in termen van coördinaten in plaats van lijnen. Deze definitie is gebaseerd op het volgende resultaat.

Theorema 2.1.1 *Een afbeelding $h : \mathbb{P}^2 \to \mathbb{P}^2$ is een projectieve transformatie als en slechts als er een reguliere 3×3 matrix H bestaat zodat voor elk punt in $\mathbb{P}^2$, voorgesteld met een vector x, $h(x) = Hx$ naar waar evalueert.*

Dit theorema zegt dat de afbeelding slechts een projectieve transformatie is als er een matrix H kan worden gevonden zodat de lineaire transformatie Hx gelijk is aan het beeld van $h(x)$. Als resultaat hiervan kan er een alternatieve definitie worden opgesteld voor een projectieve transformatie.

Definitie 2.1.1 *Een projectieve transformatie in het vlak is een lineaire transformatie, toegepast op homogene 3-vectoren, voorgesteld met een reguliere* 3×3 *matrix:*

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.3}$$

of in het kort: $x' = Hx$.

Merk op dat de matrix H in deze vergelijking mag veranderd worden door een vermenigvuldiging met een extra schaal factor, verschillend van 0, zonder invloed te hebben op de projectieve transformatie. Daarom wordt H ook wel een *homogene matrix* genoemd. Een projectieve transformatie projecteert elke afbeelding in een geprojecteerde equivalente afbeelding en laat daarbij alle projectie kenmerken invariant.

Model voor projectieve correctie In een afbeelding is er een projectieve verstoring aanwezig als evenwijdige lijnen in de scene, niet evenwijdig zijn in de afbeelding [Har03b]. Deze lijnen convergeren naar een eindig punt. We weten nu dat de afbeelding in de originele scene gerelateerd zijn via een projectieve transformatie, meer bepaald is de originele scene verstoord via een projectieve transformatie. Het is mogelijk om deze transformatie ongedaan te maken door de inverse transformatie te berekenen op de afbeelding. Het resultaat hieruit is vervolgens een afbeelding waarvan de objecten in het vlak zichtbaar zijn met hun correcte geometrische vorm. In deze sectie stellen we een model op om de projectieve transformatie punt gebaseerd uit te voeren.

Om dit model op te stellen moeten er paren van punten worden gezocht die in de scene en in de afbeelding overeenkomen. Stel de niet homogene coördinaten van een paar overeenkomstige punten x en x' in de scene en in het afbeeldingsvlak, respectievelijk (x, y) en (x', y'). Hier wordt er gebruik gemaakt van niet-homogene coördinaten omdat deze direct gebruikt worden vanuit de afbeelding. De projectieve transformatie uit 2.3 wordt dan als volgt geschreven [Har03b].

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Of nog, omdat er voor elk punt twee vergelijkingen zijn,

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13} \tag{2.4}$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23} \tag{2.5}$$

Vier van zulke punten resulteren in acht vergelijkingen, welke genoeg zijn om de matrix H op te lossen op een insignificante vermenigvuldigingsfactor na. Als enige beperking geldt dat de vier punten zo moeten gekozen worden dat er geen enkele 3 punten collinear ten opzichte

van elkaar liggen. Om alle onbekenden van H te vinden hoeft enkel nog de volgende matrix te worden opgelost

$$\left[\begin{array}{cccccccccc} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y & -x'_1 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y & -x'_2 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y & -x'_3 & 0 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y & -x'_4 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y & -y'_1 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y & -y'_2 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y & -y'_3 & 0 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y & -y'_4 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \tag{2.6}$$

De oplossingen kunnen bijvoorbeeld worden bekomen door er een triangulaire matrix van te maken en vervolgens de waardes voor H af te lezen. Let op de laatste vergelijking is de keuze van de insignificante vermenigvuldigingsfactor van H. Hier is er voor gekozen om de som van alle elementen van H te laten resulteren in 1. Door het berekenen van de inverse H^{-1} kan de omgekeerde transformatie worden toegepast en zo pixelgebaseerd het gecorrigeerde beeld worden opgesteld.

2.1.5 Calibratie met behulp van geprojecteerde patronen

De tot nu toe besproken technieken kunnen allemaal een vrij grote afwijking hebben als gevolg van afrondingen en benaderingen. Zo zal de bepaling van de camera-eigenschappen altijd moeten worden benaderd met behulp van statistische gegevens en modellen. De projectieve correctie zal enkel goed werken als de 4 snijpunten nauwkeurig kunnen worden bepaald. Een zéér nauwkeurige en veelgebruikte techniek is het gebruik van graycode patronen.

Een sequentie van patronen worden geprojecteerd op de tafel en zullen waargenomen worden door één of meerdere camera's. Eén graycode patroon bestaat uit zwarte en witte strepen die de projectie verdelen in fijnere gebieden naarmate de sequentie vordert [Lee04b]. Per pixel geeft de groep van geprojecteerde patroon zo een graycode, waarvan elke bit staat voor een kleurwaarde in het patroon. Elke pixel of groep van pixels zal nu overeenkomen met een eigen uniek codewoord. Een overzicht van de pixels met hun overeenkomstige graycode is gegeven in Figuur 2.7. Zo heeft pixel 17 bijvoorbeeld een code 00110. Een voorbeeld van een sequentie is de volgende: Figuur 2.9 [JSB09].

Figuur 2.7: Deze figuur geeft de graycodes weer van een ééndimensionale functie over een bereik van 32 pixels. De graycode voor pixel 25 is bijvoorbeeld 01010.

Voor elke pixel die binnenkomt uit het camerabeeld kan er een graycode worden opgesteld. Voor deze graycode bestaat er een mapping op de overeenkomstige coördinaten van dezelfde pixel in het patroon. Hij mapt dus een pixel van het inkomend beeld op de overeenkomstige pixel in het projectorbeeld. Deze mapping is degene die we willen bekomen voor de calibratie. De mapping zal nu concreter worden gedefinieerd. Stel het projectorbeeld wordt gedefinieerd zoals in Figuur 2.8. Dan zal er voor elke pixel een mapping bestaan van twee graycodes naar 1 coördinaat. Er is namelijk 1 graycode nodig voor de horizontale as en 1 voor de verticale as. Dit geeft

$$[(a_1, a_2, \ldots, a_{k-2}, a_{k-2}), (b_1, b_2, \ldots, b_{l-2}, b_{l-2})] \Rightarrow (x_1, y_1) \tag{2.7}$$

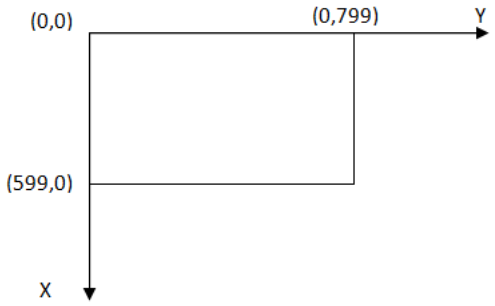
met $(a_1, a_2, \ldots, a_{k-2}, a_{k-2})$ respectievelijk de bit planes van most significant naar least significant voor de X-as en $(b_1, b_2, \ldots, b_{l-2}, b_{l-2})$ deze voor de Y-as. Het punt (x_1, y_1) is een pixel op het projectorbeeld. Op dezelfde wijze kunnen we voor elke pixel in het camerabeeld een graycode opstellen om zo de volgende mapping te verkrijgen

$$(x_2, y_2) \Rightarrow [(a_1, a_2, \ldots, a_{k-2}, a_{k-2}), (b_1, b_2, \ldots, b_{l-2}, b_{l-2})] \tag{2.8}$$

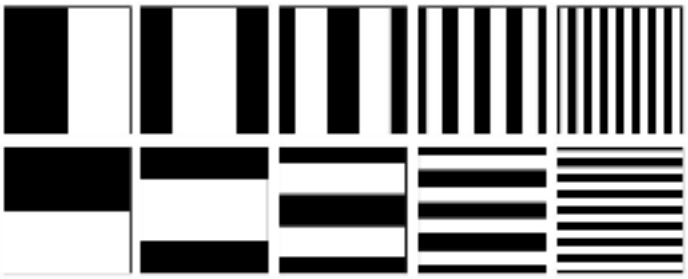
Ook hier zijn $(a_1, a_2, \ldots, a_{k-2}, a_{k-2})$ en $(b_1, b_2, \ldots, b_{l-2}, b_{l-2})$ de bit planes van most significant naar least significant voor respectievelijk de X en Y as. Het punt (x_2, y_2) is een pixel van het camerabeeld. Uit deze twee mappings kan er nu een brug worden gemaakt die een nieuwe mapping aanmaakt. Dit keer is het een mapping tussen de twee coördinatenstelsels zelf. Door mappings met eenzelfde $[(a_1, a_2, \ldots, a_{k-2}, a_{k-2}), (b_1, b_2, \ldots, b_{l-2}, b_{l-2})]$ te koppelen, krijgen we

$$(x_2, y_2) \Rightarrow (x_1, y_1) \tag{2.9}$$

Er bestaat nu een mapping van cameracoördinaten naar projectorcoördinaten.



Figuur 2.8: Coördinatenstelsel van het projectiebeeld.

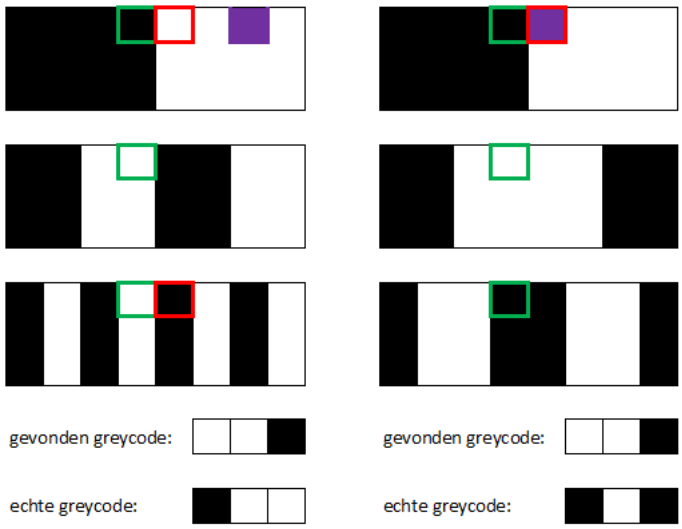


Figuur 2.9: Voorbeeld van graycode patronen die als sequentie kunnen worden geprojecteerd. De bovenste rij wordt gebruikt voor de horizontale as te coderen, de onderste rij voor de verticale as [Lee04b].

Keuze van graycode patroon In plaats van een traditionele binaire verdeling van het beeld, is het beter om de grenzen van de patronen niet altijd op dezelfde plaats te laten samenvallen. Dit kan catastrofale fouten voorkomen omdat nu de pixel maar éénmalig fout kan gekozen worden op een grens of in het ergste geval 2 maal fout. Wat hoogstens ± 1 plaats in het diepste niveau laat verspringen. Dit wordt geïllustreerd in Figuur 2.10. De groene pixel is degene die gecodeerd moet worden. Het linkse gedeelte bevat een binair patroon waardoor de groene pixel voortdurend met een rand van wit en zwart samenvalt. De rode rand geeft aan welke pixel gekozen wordt in het algoritme. In dit geval wordt de pixel twee maal fout gekozen, namelijk in de bovenste en de onderste. Dit zal resulteren in graycode 110. De juiste code zou 011 zijn. In dit geval zal de pixel 3 plaatsen verschuiven, aangegeven met de paarse pixel. Let op, op grotere schaal kan deze afwijking veel groter zijn en zelfs helemaal aan de andere kant van de projectie liggen. In het linkse patroon zal de groene pixel maar tweemaal verkeerd gekozen kunnen worden. Namelijk in de bovenste en in de onderste. Dit geeft hoogstens een verschuiving van twee pixels. In het voorbeeld is er 1 pixel verschuiving aanwezig. Ook op grotere schaal zal de groene pixel maximaal twee maal op een grens liggen. Zie Figuur 2.7 voor een voorbeeld van de tweede methode met 32 pixels.
Er kan ook gekozen worden om niet te werken met een sequentie van patronen maar enkel 1 patroon te projecteren voor de verticale as en 1 voor de horizontale. Dit kan worden gedaan met een gradiëntpatroon. De patroon gaat van onder naar boven respectievelijk van links naar rechts geleidelijk overgaan van wit naar zwart. Zo heeft elke pixel zijn eigen unieke intensiteitswaarde. Deze techniek zal de berekeningen aanzienlijk vereenvoudigen, maar dit gaat gepaard met een serieus verlies in nauwkeurigheid. Als er bijvoorbeeld 800 pixels moeten worden gecodeerd moet het bereik van intensiteit in 800 verdeeld. Het is eenvoudig in te zien dat er dikwijls verkeerde intensiteitswaardes worden waargenomen. De graad van correctheid waarmee deze techniek kan worden toegepast is sterk afhankelijk van de kwaliteit van de camera.

Graycodes hebben een $O(\log_2(n))$ relatie tussen het aantal patronen en het aantal pixels n om elke pixel uniek te bepalen. Voor een resolutie van 800 op 600 zijn er 9.64 of afgerond 10 patronen nodig (Enkel 60 patronen nodig om het hele continent van de Verenigde Staten tot op millimeter nauwkeurig te coderen). Graycodes werken ook vrij goed onder slechte condities. Enkel de LSB's van de graycode wordt aangetast als de kwaliteit van de nauwkeurigste patronen

slecht wordt. Dit is een zéér belangrijke eigenschap want zo zal met een beperkt aantal graycodes er toch nog een degelijke calibratie worden gevonden.



Figuur 2.10: Deze figuur toont aan waarom het beter is alternatieve graycodes patronen te gebruiken in plaats van binaire graycode patronen. De linkse figuur gaat binaire patronen gebruiken. De groene pixel is de pixel waarvoor de graycode wordt gezocht, de rode is de pixel die wordt gekozen. Doordat de randen tussen zwart en wit voortdurend blijven samenvallen, kan de pixel meermaals fout worden gekozen. Zo zal in dit geval de pixel tweemal fout worden gekozen waardoor de gevonden pixel (paarse) drie plaatsen van zijn correcte plaats afwijkt. In de rechtse figuur worden gebieden van dezelfde kleur twee aan twee samengenomen. Hierdoor zal de groene pixel maximaal twee maal op een rand vallen waardoor hij ook maar twee maal fout kan worden gekozen. Dit geeft een maximale afwijking van 2 pixels. In dit voorbeeld is er maar één pixel afwijking.

2.2 Segmentatie

Binnen segmentatie probeert men vanuit afbeeldingen attributen te vinden over regio's en objecten die aanwezig zijn in de afbeelding. Het resultaat van de segmentatiestap is dus geen afbeelding, maar zijn wiskundige modellen die beschrijven wat er op de afbeelding te zien is. In deze thesis is vooral het detecteren van lijnen interessant en zal deze sectie zich hier vooral op toespitsen. In de eerste plaats zal er een edge detection algoritme nodig zijn om edges te vinden. Als eenmaal de edges zijn gevonden, moet een segmentatie algoritme de parameters van de lijn bepalen die het beste met de edges overeenkomen. Edge detection kan uitgevoerd worden met een convolutiefilter. Voor segmentatie zullen er twee algoritmes worden besproken: Hough en RANSAC.

2.2.1 Edge Detection

Edge detection zorgt voor een afbeelding waarop enkel nog edges zichtbaar zijn. Een manier om dit te doen is met een highpass convolutiefilter. Deze zorgt ervoor dat detail uit een beeld wordt gedestilleerd. Meer bepaald zullen sterke overgangen in intensiteit worden behouden. Kenmerkend voor een edge is dat deze een sterke overgang in intensiteit bevat.

Convolutiefilter Een convolutiefiltering is een lineaire spatiale filtering die voor elke pixel een omgeving beschouwt en deze omgeving gebruikt om een nieuwe pixelwaarde te vinden voor de desbetreffende pixel [GW08b]. In dit geval wordt een omgeving van $3X3$ gebruikt. Elke pixel in de omgeving krijgt een bepaald gewicht toegekend en dit gewicht bepaald hoe hard de pixel doorweegt op het resultaat. Belangrijk om op te merken is dat bij convolutie eerst de omgeving 180° wordt gedraaid. Algemeen wordt de convolutie van een omgeving $w(x, y)$ (ter grootte van aXb en de afbeelding $f(x, y)$ gegeven door de expressie

$$w(x,y)\bigstar f(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x-s,y-t) \tag{2.10}$$

waarbij de mintekens zorgen voor de rotatie van 180°. $w(x, y)$ ziet er bijvoorbeeld uit zoals Figuur 2.11.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figuur 2.11: Een algemene spatiale filter

Highpass convolutie filter Om specifiek detail uit een afbeelding te halen wordt vaak gebruik gemaakt van de Laplacian [GW08b],[GW08c]. De Laplacian voor een afbeelding $f(x, y)$ is gedefinieerd als

$$\bigtriangledown^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.11}$$

De Laplacian is de tweede afgeleide en kan worden geschreven in discrete vorm als volgt: voor x

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \tag{2.12}$$

en voor de y

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \tag{2.13}$$

Voor de twee variabelen samen geeft dit

$$\bigtriangledown^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{2.14}$$

Deze laatste vergelijking omgezet in filtervorm geeft Figuur 2.12a. Als de diagonalen meegerekend worden, kan deze herschreven worden tot Figuur 2.12b.

0	1	0
1	-4	1
0	1	0

(a) Normale Laplacian

1	1	1
1	-8	1
1	1	1

(b) Uitgebreide Laplacian

Figuur 2.12: Laplaciaanse spatiale filters voor de detectie van detail in een beeld.

Filters specifiek voor lijndetectie Hieronder valt de Sobel filter die gebaseerd is op de gradiënt operator [GW08c]. De filters zijn onderverdeeld per richting: horizontaal, verticaal, $+45°$ en $-45°$. Deze kunnen worden gebruikt om specifiek lijnen te zoeken van een bepaalde oriëntatie. Figuur 2.13a is bijvoorbeeld een filter om horizontale lijnen te detecteren.

-1	-2	-1
0	0	0
-1	-2	-1

(a) Horizontaal

-1	0	-1
-2	0	-2
-1	0	-1

(b) Verticaal

0	1	2
-1	0	1
-2	-1	0

(c) $+45°$

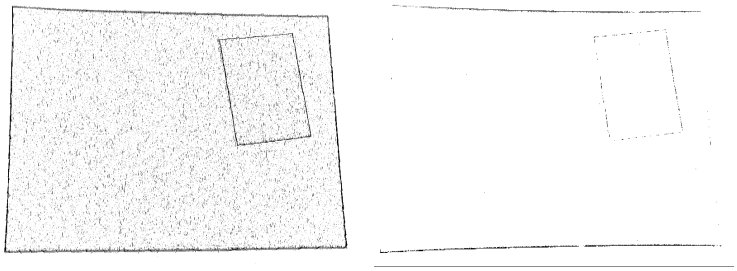
-2	-1	0
-1	0	1
0	1	2

(d) $-45°$

Figuur 2.13: Sobel spatiale filters voor de detectie van lijnen met een bepaalde oriëntatie. De oriëntatie van lijnen die worden gedetecteerd zijn respectievelijk horizontale, verticale, $+45°$ en $+45°$.

2.2.2 Hough

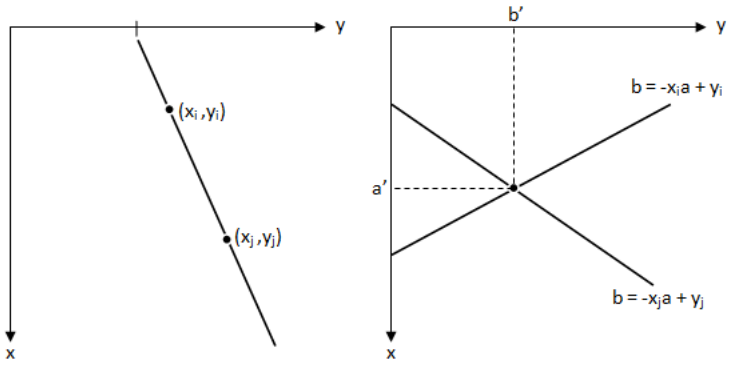
Met de Hough transformatie is het mogelijk om op een globale manier lijn detectie toe te passen. Met globaal wordt bedoeld dat er geen weet is van eventuele interessante gebieden waar objecten kunnen liggen. Het enige wat gebruikt wordt is een edge afbeelding. Een edge afbeelding is een beeld waar het detail uit een afbeelding wordt gehaald. Bij het detecteren van een papier zullen de randen ervan zichtbaar zijn op de edge afbeelding. Dit zijn de randen die in de segmentatiestap moeten worden gevonden. Figuur 2.14 toont twee voorbeelden van een dergelijke edge afbeelding.



(a) Edge image met ruis (b) Edge image zonder ruis

Figuur 2.14: Edge afbeeldingen waar de randen geaccentueerd worden.

Wat is de Hough transformatie Beschouw een punt (x_i, y_i) in het xy-vlak en een lijn $y_i = ax_i + b$ die het punt (x_i, y_i) intersecteert [GW08d]. Voor een variabele a en b kan er een oneindig aantal lijnen in het vlak xy worden gevonden van de vorm $y = ax + b$. Als nu niet x en y als variabel worden beschouwd, maar wel a en b, dan kan, voor (x_i, y_i), de vergelijking worden herschreven in $b = -x_i a + y_i$. Deze voorstelling van een lijn is nu in het ab-vlak. Dit vlak wordt ook wel de parameter ruimte genoemd. De omvorming van het xy-vlak naar ab-vlak is een Hough transformatie. Met een tweede punt (x_j, y_j) komt in de parameter ruimte een tweede lijn overeen. Deze zal de vorige lijn snijden (tenzij ze parallel liggen) in een punt (a', b'). Deze coördinaten a en b zullen in het xy-vlak een lijn beschrijven die door de punten (x_i, y_i) en (x_j, y_j) loopt. Meer bepaald zullen alle punten op deze lijn in parameter space het punt (a', b') intersecteren. Dit is geïllustreerd in Figuur 2.15.

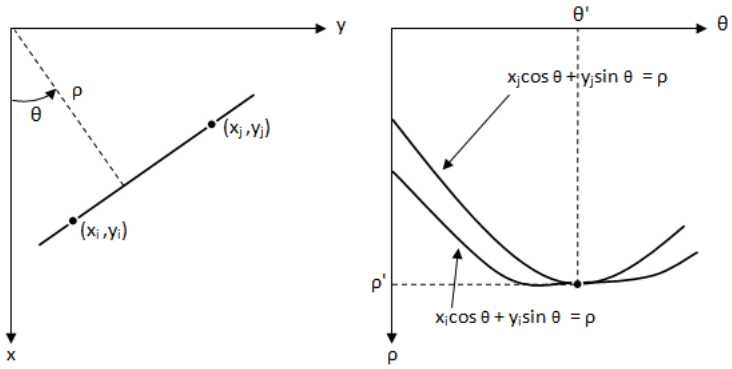


Figuur 2.15: Links het xy-vlak en rechts de parameter ruimte.

Alternatieve parameter ruimte Een probleem met de algemene vergelijking van een lijn is dat deze niet gebruikt kan worden voor verticale lijnen. Om deze beperking tegen te gaan kunnen alternatieve lijnvoorstellingen worden gebruikt. Eén manier is gebruik maken van de homogene voorstelling van een lijn [Har03c]. Hier wordt niet verder op in gegaan. Een tweede manier is het gebruik van de (ρ, θ) parameterruimte. De representatie is als volgt

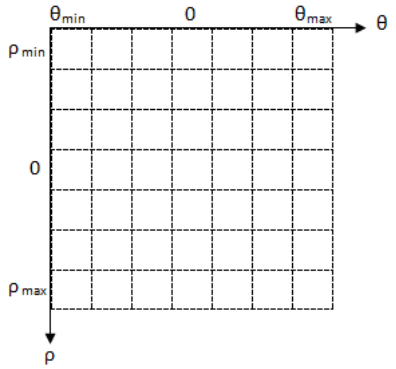
$$x \cos\theta + y \sin\theta = \rho \tag{2.15}$$

Hierbij is θ de hoek die de lijn maakt met de positieve x-as. Dus voor $\theta = 0°$ geeft dit een horizontale en voor $\theta = 90°$ een verticale lijn. ρ geeft de afstand aan van de oorsprong tot de lijn. Dit concept wordt getoond in Figuur 2.16.



Figuur 2.16: Links (ρ, θ) parametrisatie van een lijn uit het xy-vlak en rechts de overeenkomstige parameter ruimte.

Het voordeel van de Hough transformatie is dat de parameterruimte gemakkelijk kan verdeeld worden in cellen waarbij (ρ_{min}, ρ_{max}) en $(\theta_{min}, \theta_{max})$ het bereik is van de parameters: $-90° \preceq \theta \preceq 90°$ en $-D \preceq \rho \preceq D$, met D de maximale afstand van een lijn. De cellen worden weergegeven in Figuur 2.17. In het begin worden al deze cellen op 0 geïnitialiseerd. Voor elk wit punt (voorgrond) (x_k, yk) uit het inputbeeld zal nu de vergelijking $\rho = x_k \cos\theta + y_k \sin\theta$ worden opgelost voor elke mogelijke waarde θ. De ρ die hieruit volgt wordt afgerond naar de dichtst bijzijnde bestaande cel voor de ρ-as. De gevonden cel zal vervolgens met 1 worden verhoogd. Na het toepassen van dit algoritme voor elk wit punt zullen de cellen met de hoogste waardes staan voor de parameters van een lijn die door de meeste punten loopt.



Figuur 2.17: Indeling van de parameter ruimte $\rho\theta$ in discrete cellen. Dit is nodig om het algoritme performant te houden. Afbeelding eigendom van [DIM4]

2.2.3 Heuristische methode: RANSAC

RANSAC of RANdom SAmple Consensus is een algoritme van Fischler en Bolles [FB81]. Het is een stochastisch iteratief algoritme om de parameters van een wiskundig model te schatten uit een aantal punten, de zogenaamde outliers. Het resultaat van het algoritme is niet het exacte wiskundige model dat wordt gezocht, maar de beste benadering die gevonden is. Het algoritme gaat iteratief mogelijke inliers (punten die tot het wiskundige model horen) kiezen. Door deze gekozen inliers wordt het wiskundig model gefit. Een laatste stap is het vergelijken van het gekozen wiskundig model met het gezochte wiskundig model. De mate waarin deze twee modellen overeenkomen bepaalt de nauwkeurigheid van het algoritme. Des te meer iteraties er worden gedaan, des te nauwkeuriger het algoritme zal worden.

RANSAC voor lijndetectie In ons geval zijn we op zoek naar lijnen. Er wordt dus getracht om de vergelijking $y = ax + b$ terug te vinden waarvoor de meeste punten uit de outliers voldoen aan deze vergelijking. Wegens afrondingen zal in de praktijk een punt zelden precies op de lijn liggen. Hiervoor is het noodzakelijk om een threshold in te stellen. Elk punt waarvan de loodrechte afstand tot de lijn binnen een threshold t ligt zal worden geclassificeerd als inliers, alle punten er buiten als outliers. De threshold t kan worden ingesteld, afhankelijk van de input. Hier wordt later dieper op ingegaan.

Er worden random twee punten gekozen uit de verzameling van punten S. Voor deze twee punten kan een vergelijking worden opgesteld. Naarmate er meer andere punten binnen een threshold t van deze lijn liggen, zal de lijn in waarde stijgen. Al de punten op de lijn worden geklasseerd als inliers en worden voorgesteld door de verzameling S_i. Als het aantal punten in S_i groter is als een bepaalde threshold T, eindigt het algoritme en is de beste lijn gevonden. Als het aantal punten in S_i kleiner is als T wordt het algoritme terug herhaald. Na N keer proberen wordt de lijn met de meeste inliers gekozen als beste lijn.

Hoeveel keer twee punten kiezen uit de verzameling Meestal is het niet nodig om elke combinatie te proberen. In de plaats wordt er een waarde N gekozen, hoog genoeg om te verzekeren, met een kans p, dat minstens 1 van de gekozen combinaties s vrij is van punten die er niet zouden mogen bijhoren. Meestal wordt p gekozen op 0.99. Veronderstel nu dat w de kans is dat elk geselecteerd punt een inlier is, en dus $\epsilon = 1 - w$ de kans is dat er een outlier bijzit, dan zijn er zeker N iteraties noodzakelijk, waar $(1 - w^s)^N = 1 - p$, zodat

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}. \tag{2.16}$$

N hoeft niet enkel afhankelijk te zijn van de kans p. Er kan ook gekozen worden om het algoritme een bepaalde tijd te laten uitvoeren. Stel bijvoorbeeld dat RANSAC maximaal 20 ms mag duren. Er kunnen zoveel iteraties als nodig worden uitgevoerd totdat de 20 ms volledig zijn opgebruikt. De beste lijn die is gevonden wordt vervolgens als resultaat gegeven.

2.3 Object Tracking

2.3.1 Particle filter - Conditional Algorithm for Visual Tracking

Het doel van een particle filter is om een geometrisch object te zoeken en/of te volgen in een scene. De toestand van een huidig frame wordt gebruikt om deze van een volgend frame te voorspellen. Dit gebeurt door een aantal particles te genereren die elk een transformatie van het object weerspiegelen. De particle die de echte toestand het best benadert zal gekozen als resultaat voor de volgende frame. De mate waarin het object overeenkomt met de echte wereld zal een meetstaaf zijn voor een gewicht dat aan een particle wordt gehangen. Door de concentratie van particles te verhogen rond particles met een groter gewicht zal voornamelijk daar worden gekeken waar het object zich in een vorige toestand bevond. Een algemene beschrijving van particle filters wordt gegeven door P. Li, T. Zhang en A. Pece [PLP03]. De notaties in deze sectie zijn gebaseerd op deze van T. Cuypers [Cuy07].

Een variant van de gewone particle filter, die specifiek kan gebruikt worden voor het volgen van objecten, is het condensation algoritme. Deze is voorgesteld door Michael Isard en Andrew Blake [IB98]. Een particle wordt genoteerd met X. De status van de particle op tijdstip k is X_k. k geeft dus aan in welke succesvolle iteratie het algoritme zit. Er wordt een proposal density function opgesteld, meestal is dit een Gaussiaanse verdeling:

$$\pi(X_k^{(i)}\big|X_{k-1}^{(i)}, Y_{1:k}) = p(X_k^{(i)}\big|X_{k-1}^{(i)}) = N(\hat{X}_k^{(i)}, \hat{P}_k^{(i)}), i = 1, \ldots, N \tag{2.17}$$

Deze verdeling is gedefinieerd voor elke particle $\hat{X}_k^{(i)}$ met een Gaussiaanse verdeling $N(\hat{X}_k^{(i)}, \hat{P}_k^{(i)})$ met $\hat{X}_k^{(i)}$ als gemiddelde en $\hat{P}_k^{(i)}$ als variantie. Het algoritme werkt in het kort als volgt:

1. *Initial step* Maak een verzameling van particles aan met een initiële verdeling. Zet $k = 1$
2. *Sampling step* Maak nieuwe samples en definieer hun gewicht.
   (a) Voor $i = 1, \ldots, N$, sample $X_k^{(i)}$ met de verdelingsfunctie $p(X_k^{(i)}\big|X_{k-1}^{(i)})$.
   (b) Bereken hun gewichten: $w_k^{(i)} = w_{k-1}^{(i)} \frac{p(Y_k\big|X_k^{(i)})p(X_k^{(i)}\big|X_{k-1}^{(i)})}{\pi(X_k^{(i)}\big|X_{k-1}^{(i)}, Y_{1:k}}, i = 1, \ldots, N.$
   (c) Normaliseer de gewichten: $w_k^{(i)} = w_{k-1}^{(i)} p(Y_k\big|X_k^{(i)}), i = 1, \ldots, N.$
3. *Prediction step* Voorspel voor elke particle een mogelijke verandering voor de volgende frame.
4. *Selection step* Particles $X_k^{(i)}$ opnieuw samplen met een kans $w_k^{(i)}$. De verdeling van de particle is opnieuw volgens de kansverdelingsfunctie $p(X_k^{(i)}\big|X_{k-1}^{(i)})$.
5. $k = k + 1$ en ga verder met stap (2)

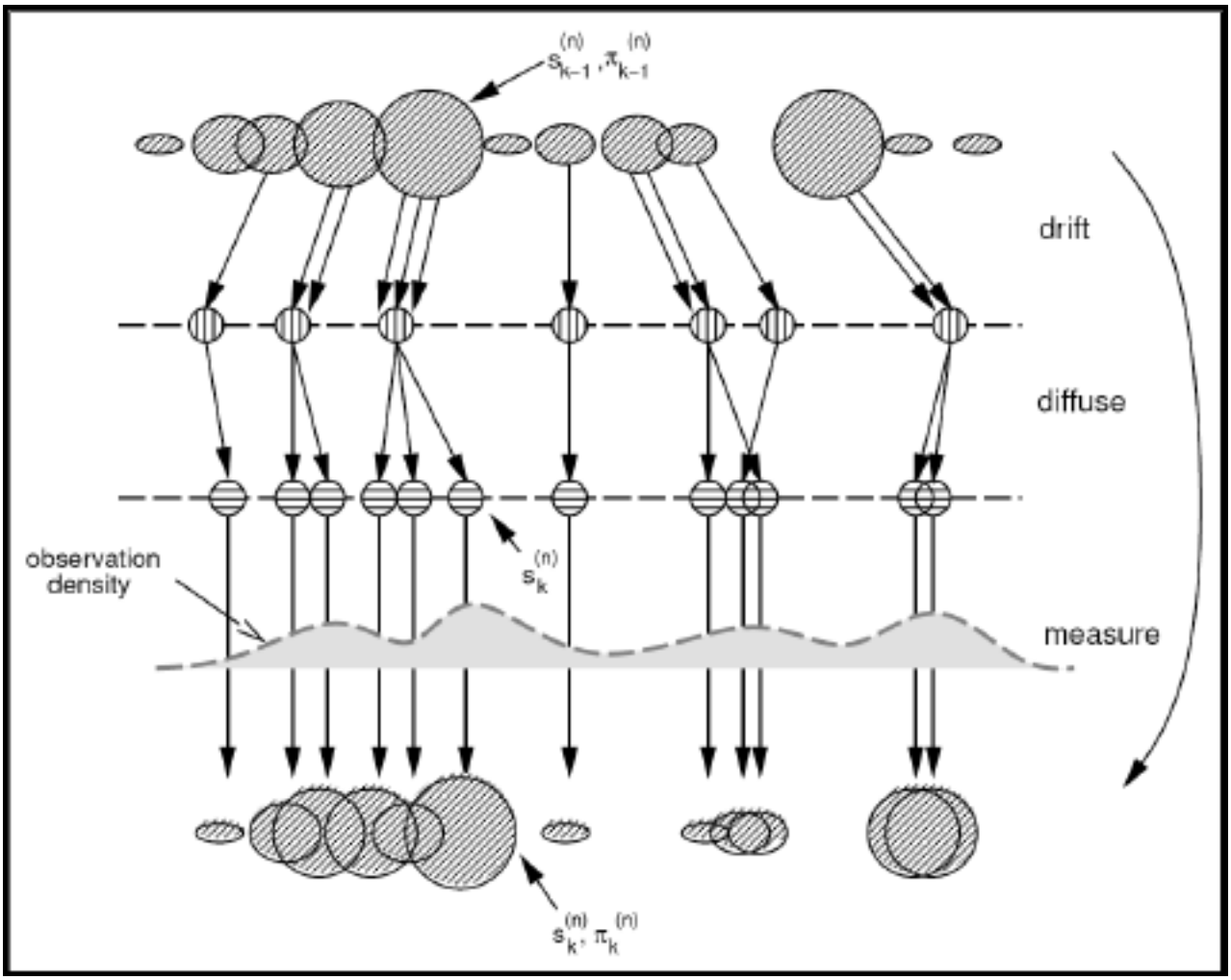
Om dit algoritme te gebruiken voor tracking van objecten wordt het opgedeeld in 3 stappen per frame. De *select*, *predict* en *measure* stap. Alvorens deze 3 stappen iteratief uit te voeren, is er een initialisatie nodig. De stappen zullen geïllustreerd worden met een 1D voorbeeld in Figuur 2.18 [IB98].

Initialisatie Een particle bestaat uit een bepaalde transformatie die specifiek is voor het soort applicatie en het object dat wordt gevolgd. Verder heeft het nog een extra parameter w voor het gewicht. Er wordt een verzameling van N particles aangemaakt. Hun parameters worden uniform random verdeeld. Hun gewicht wordt geïnitialiseerd op de genormaliseerde waarde $\frac{1}{N}$.

Select In deze stap worden er N nieuwe particles gegenereerd. Elke particle bevat de dezelfde parameters als een particle van de vorige frame. Ze worden gekozen aan de hand van het gewicht van de particles uit het vorige beeld. Een particle met een groter gewicht zal dus vaker opnieuw worden gekozen. In de figuur is dit de drift stap. In dit voorbeeld worden particles met een hoger gewicht soms meer als één keer gekozen en particles met een lager gewicht soms niet.

Predict In deze fase wordt voor elke particle een nieuwe toestand voorspeld. Er zal bij de parameters bijvoorbeeld een Gaussiaanse random waarde worden opgeteld. De Gaussiaanse verdeling heeft een gemiddelde 0 en een standaardafwijking σ. De dichtheid van de particles zal hoger zijn op de plaats waar de vorige particles een hoog gewicht had. In de figuur is deze stap voorgesteld met diffusion.

Measure De particles afkomstig uit de predict fase zullen nu worden getoetst met de frames die binnenkomen. Dit is een stap die afhankelijk is van het soort applicatie. Het berekenen van de overeenkomst tussen de particle en het echte beeld is afhankelijk van het object dat wordt getracked. Het gewicht geeft de mate van correctheid aan van de particle. Hoe meer overeenkomst de particle heeft met de echte plaats, hoe hoger het gewicht. Als laatste moeten deze gewichten worden genormaliseerd. In de figuur wordt dit voorgesteld met de measure stap. Elke particle gaat naargelang de goedheid van de overeenkomst met de observatie een groter gewicht toegekend krijgen.



Figuur 2.18: Een stap in het condensation algoritme. In eerste instantie (drift) worden nieuwe particles aangemaakt op basis van de gewichten van de vorige stap. Grotere gewichten zullen dus typisch resulteren in meerdere particles. In tweede instantie (diffuse) zal voor elk nieuw aangemaakte particle een voorspelling worden gedaan. Deze voorspelling is meestal zo gekozen zodat de directe omgeving meer kans heeft om voorspeld te worden. Meestal zal er per frame niet veel verandering plaatsvinden. In een laatste stap (measure) wordt voor elke voorspelde particle nagegaan hoe goed hij overeenkomt met de observatie. Deze mate van overeenkomst zal een indicatie zijn voor het aanmaken van de nieuwe gewichten per particle. [IB98]

Hoofdstuk 3

Implementatie

In het vorige hoofdstuk zijn de algoritmes besproken die kunnen gebruikt worden om deze thesis in praktijk te kunnen verwezenlijken. In dit hoofdstuk zal de aandacht gevestigd worden op specifieke implementatie details van de verschillende algoritmes. Alvorens aan de algoritmes te beginnen zal er eerst een overzicht worden geven van de verschillende stappen die nodig zijn in de implementatie en de opstelling die wij gebruikt hebben.

3.1 Opstelling

In onze opstelling gebruiken we een tafel met een diffuus doek. Het doek zorgt ervoor dat de tafel geen speculaire reflecties heeft. Loodrecht op de tafel wordt een camera en projector gericht. Enkele afbeeldingen van de opstelling zijn gegeven in Figuur 3.1.

Figuur 3.1: De opstelling die wij gebruikt hebben voor de implementatie. Het bevat een tafel met een diffuus doek, een camera en projector loodrecht op de tafel gericht.

3.2 Verschillende stappen

Deze sectie beschrijft welke stappen er nodig zijn om 1 inkomend frame te verwerken. Deze stappen hoeven niet altijd in dezelfde volgorde te worden uitgevoerd. De volgende sectie zal beschrijven welke volgorde in onze implementatie is gebruikt.

- *Radiaal corrigeren*
  Indien de radiale verstoring niet werd opgeheven bij calibratie, moet deze gecorrigeerd worden.

- *Edge detection*
  Een beeld moet met behulp van convolutie worden gefilterd tot een edge image zodat de randen van het papier goed duidelijk zijn.

- *Segmenteren*
  Gegeven een edge image, moet er met behulp van een segmentatie techniek de lijnen van het papier worden bepaald.

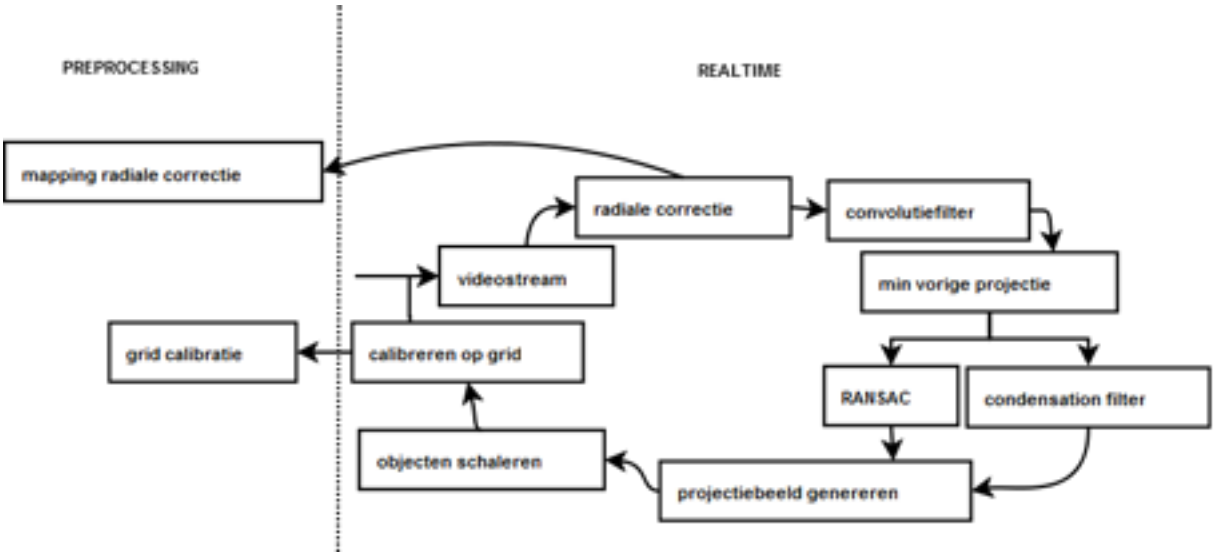
- *Papier tracken*
  Een papier of ander object tracken over de tijd. Zo kan een verplaatsing of rotatie snel worden aangepast door de verplaatsing of rotatie ook toe te passen op het papier in het projectiebeeld.

- *Projectiebeeld genereren*
  De lijnen afkomstig van de segmentatie worden gebruikt om een object op het papier te projecteren. Hiervoor worden de vier snijpunten van de vier lijnen berekend. Deze komen overeen met de hoekpunten van het papier.

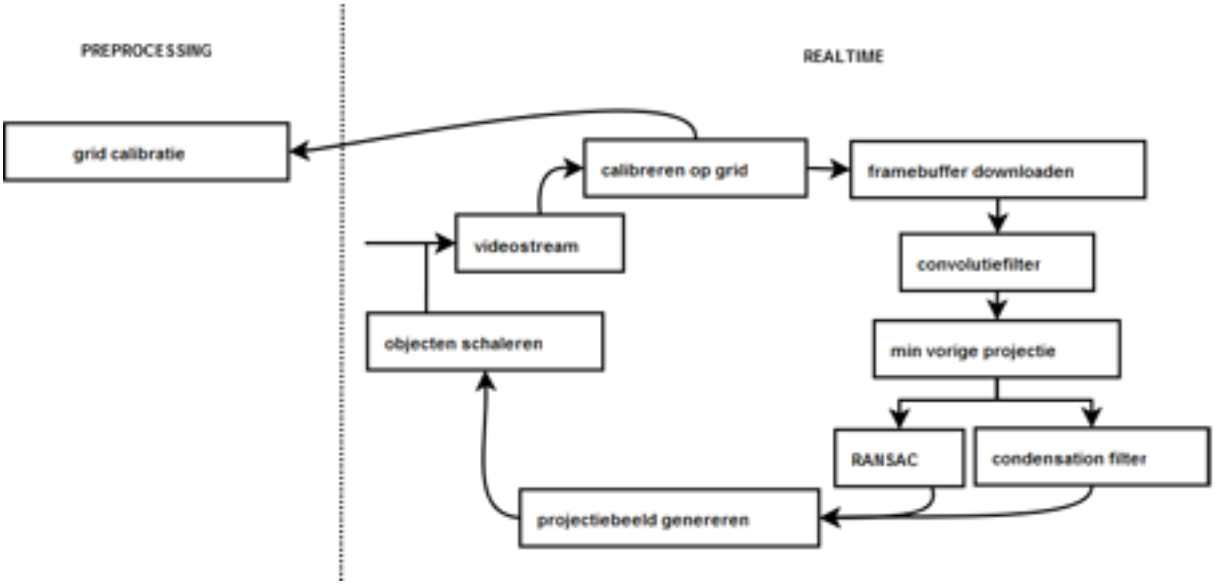
- *Calibreren*
  Een camera moet met een projector kunnen gecalibreerd worden. Dit gebeurt door bijvoorbeeld de mapping te gebruiken die bekomen is vanuit de graycodes.

In onze implementatie zijn twee benaderingen geïmplementeerd. In de eerste benadering wordt er begonnen met radiale correctie. Hierdoor gaan de lijnen worden rechtgetrokken. Er kan dus vervolgens met behulp van convolutie een edge image worden bekomen. Uit deze edge image worden de lijnen gedetecteerd. In eerste instantie zullen de lijnen gedetecteerd worden door middel van segmentatie. Met deze gedetecteerde lijnen wordt een afbeelding opgesteld met het object in dat op het papier moet worden geprojecteerd. Nadat het papier eenmaal is herkend wordt in de daarop volgende frames het tracking algoritme toegepast. De verplaatsing en rotatie afkomstig van het tracking algoritme kunnen toegepast worden op de objecten in het projectiebeeld. Dit gebeurt net zolang de nauwkeurigheid degelijk blijft. Indien de nauwkeurigheid te slecht wordt zal een segmentatieframe de nauwkeurigheid terug maximaliseren. Een eerste belangrijke opmerking is dat het projectiebeeld nog steeds in cameracoördinaten staat. In een volgende stap zal dit dus gemapped worden op projectorcoördinaten. De output hiervan is nu klaar om geprojecteerd te worden. Een tweede belangrijke opmerking is dat ook de beelden die binnenkomen om de mapping van calibratie op te stellen ook eerst radiaal moeten gecorrigeerd worden. Anders zal uiteindelijk de radiale correctie twee keer worden uitgevoerd: één keer in

de eerste stap en nog een keer bij calibratie. Het stroomdiagram van de eerste benadering is weergegeven in Figuur 3.2.
In de tweede benadering gaat het camerabeeld direct gecalibreerd worden naar projectorcoördinaten, inclusief de radiale correctie. Hier gaan we ook direct de radiale correctie in meerekenen. Vervolgens kan er net zoals in de vorige benadering de convolutie en lijndetectie worden toegepast. Ook hier kan in plaats van lijndetectie, het tracking algoritme worden gebruikt. Als vervolgens de afbeelding met het toegevoegde object wordt opgemaakt is deze direct klaar om geprojecteerd te worden. Het stroomdiagram van de tweede benadering is weergegeven in Figuur 3.3.
De eerste benadering wordt beschouwd als de hoofdbenadering. Daarom zullen de volgende secties in deze volgorde worden ingedeeld.



Figuur 3.2: In deze figuur staat de stroom die gevolgd wordt bij de eerste benadering. Aan de linkerkant van de lijn staan de twee preprocessing stappen: het aanmaken van de mapping voor radiale correctie en het aanmaken van een grid waarop een afbeelding kan gecalibreerd worden. Aan de rechterkant van de figuur staat de stroom die realtime wordt uitgevoerd. Deze begint bij de videostream. Dit zijn de frames afkomstig van de camera. Op deze camerabeelden wordt een radiale correctie uitgevoerd. Het resultaat hieruit wordt gebruikt om te filteren tot een edge image. Van deze edge image moeten de objecten uit de vorige frame worden afgetrokken om vervuiling in de huidige frame te voorkomen. Na deze stap kan de stroom in twee richtingen verder gaan. Ofwel gebruikt hij RANSAC voor de lijnen te herkennen en vervolgens op basis hiervan het projectiebeeld genereren. Ofwel gebruikt hij de vorige posities van het blad en zoekt hij met de condensation filter een schatting waar het blad nu ligt. Nadat het projectiebeeld is aangemaakt, moet de objecten hierin nog geschaleerd worden zodat ze het volgende frame niet verstoren. Als laatste moet het beeld op de grid worden geplaatst zodat het gecalibreerd wordt naar projectorcoördinaten. Op dit moment is de cirkel rond en kan hij terug opnieuw beginnen met een nieuw videobeeld.



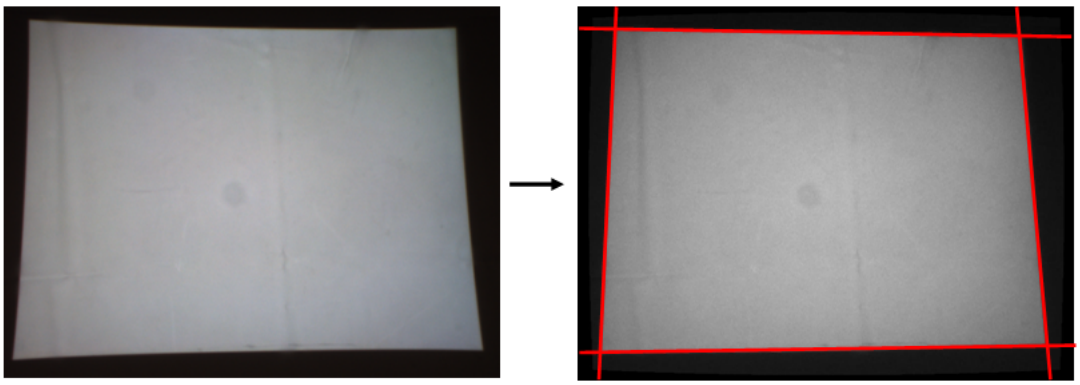
Figuur 3.3: De stroom van benadering 2 is bijna analoog als deze van benadering 1 uit Figuur 3.2. We zullen hier dan ook enkel de verschillen bespreken. Benadering 2 maakt geen gebruik van een radiale correctie, maar gaat deze direct samen met de calibratie opheffen. Hierdoor is er als preprocessing stap enkel nog het aanmaken van de calibratie mapping. Het camerabeeld afkomstig van de videostream gaat nu direct op de grid worden geplaatst. Hierdoor wordt er direct in de juiste ruimte verdergewerkt. Doordat de calibratie op de GPU gebeurt, moet deze nog gedownload worden. Dit wordt gedaan door de framebuffer op te vragen. De rest van de stroom verloopt analoog als deze van benadering 1. Enkel op het laatste kan nu direct het projectiebeeld worden geprojecteerd.

3.3 Stap 1 - Radiaal corrigeren

Radiale correctie is een noodzakelijke stap. Toch is het niet noodzakelijk deze zelf te implementeren. Als graycode patronen worden gebruikt om de calibratie toe te passen zal ook direct de radiale verstoring worden opgeheven. Toch kan het soms gewenst zijn om deze stap afzonderlijk uit te voeren. Later in de resultaten zullen we merken dat het zelfs aan te raden is om deze stap afzonderlijk te doen. Indien er een rechtstreekse correctie bij calibratie wordt gebruikt, zal helemaal in het begin de frame gecalibreerd worden. Deze calibratie zal op de GPU gebeuren. Hierdoor moet, voordat er verder wordt gegaan met het proces, eerst de framebuffer worden gedownload van de GPU. De koppeling tussen processor en GPU gebeurt in 3 stappen. De systeembus aan plus minus 10666 MB/s, PCIExpress aan 4000 MB/s voor versie 1 en de videogeheugenbus aan plus minus 141696 MB/s. Deze waardes zijn natuurlijk erg afhankelijk van het systeem dat wordt gebruikt. Maar het is duidelijk dat PCIExpress een relatieve bottleneck is in dit proces. Een afbeelding van 800 op 600 (met 3 bytes voor de RGB kleurwaarden), of nog 1.440 MB, heeft 0.36 ms nodig om te verplaatsen tussen CPU en GPU. De uitwisseling kan best geminimaliseerd worden. De extra berekeningen die moeten gebeuren om de afbeelding

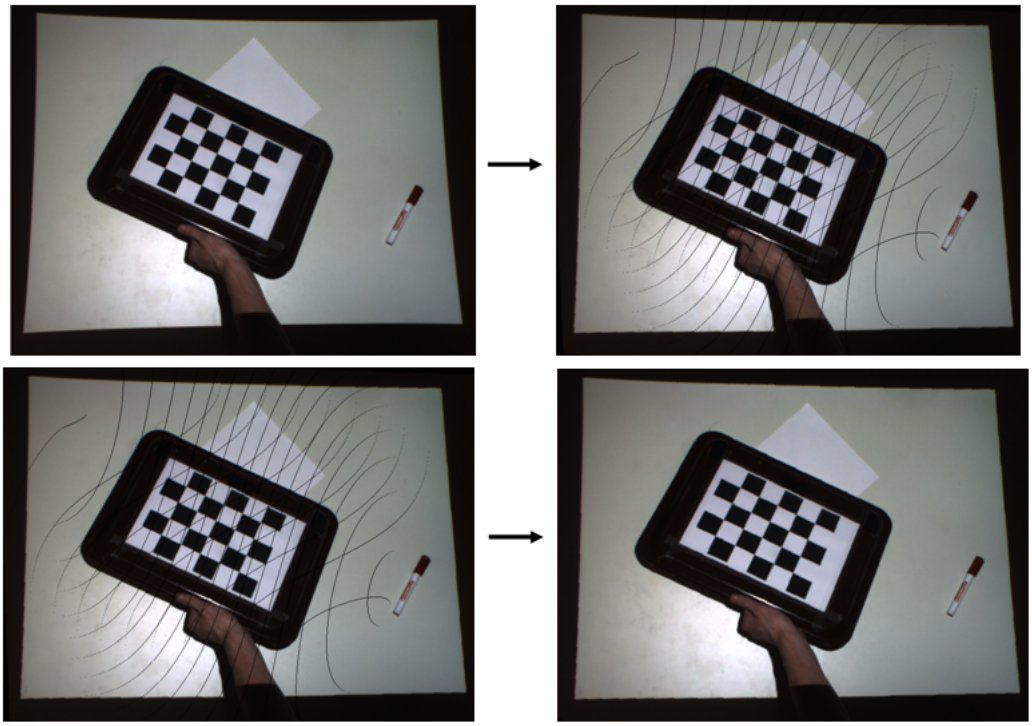
gebruiksklaar te maken loopt in onze applicatie op tot 200 ms. Eens de radiale correctie is uitgevoerd, zijn de randen van het beeld recht getrokken. Hierdoor kan er in cameracoördinaten verder worden gewerkt aan lijndetectie, zonder dat het beeld eerst moet worden gecalibreerd. Het voordeel is dat enkel op het laatste de calibratie moet gebeuren, zonder de framebuffer te downloaden.

In een eerste implementatie hebben we ons volledig gebaseerd op de theorie uit 2.1.3. Deze maakt gebruik van een Taylorontwikkeling voor de verstoringfunctie. Met behulp van trail and error worden deze waardes geschat, tot het visueel bevredigend resultaten geeft. Het nadeel is dat voor elke nieuwe calibratie er eerst een trail and error stap vooraf gaat. Een voorbeeld van deze implementatie is weergegeven in Figuur 3.4.



Figuur 3.4: Een voorbeeld van radiale correctie volgens de eerste implementatie. De Taylor verstoringfunctie is met behulp van Trail and Error geschat tot het een visueel correct resultaat geeft. De linkse afbeelding is radiaal verstoord. De randen van de projectie zijn duidelijk gekromd. De rechtse afbeelding is dezelfde afbeelding na radiale correctie. De rode lijnen zijn hulplijnen om aan te tonen dat de randen van projectie nu recht zijn.

In plaats van te werken met visueel afgeschatte waardes, zijn we in implementatie 2 verder gaan werken met de resultaten uit de GML Toolbox [Lab06]. Het probleem hierbij is dat GML Toolbox geen Taylor ontwikkeling als resultaat geeft, maar wel een verstoringsmatrix en een camera matrix. We hebben dus ook nood aan een nieuw algoritme om hiermee overweg te kunnen. Dit algoritme is niet geheel zelf geïmplementeerd, maar steunt deels op een implementatie van H. Plotter [Plo05]. Deze implementatie maakt enkel gebruik van de camera en verstoringsmatrix die we te weten komen uit GML Toolbox. Het resultaat is weergegeven in het bovenste gedeelte van Figuur 3.5. Zoals we op de afbeelding kunnen zien, is het beeld een beetje vergroot. Als gevolg zijn er onbekende punten ontstaan. Er moet dus nog een interpolatie gebeuren van omliggende punten om de zwarte (onbekende) punten op te vullen. In het onderste gedeelte van Figuur 3.5 worden deze punten geïnterpoleerd met een eigen implementatie van nearest neighbour interpolatie.



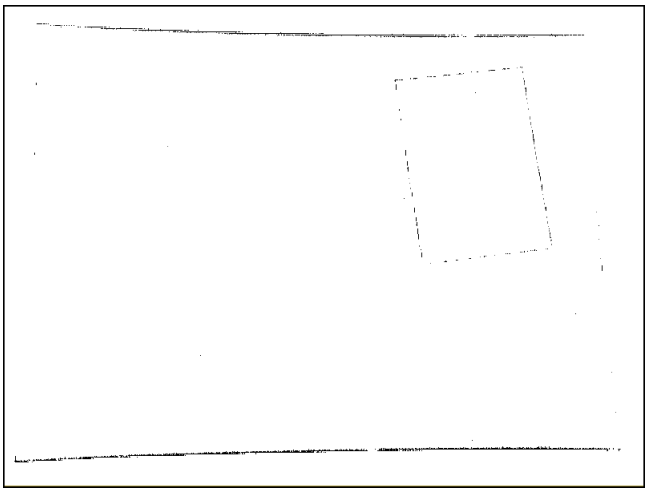
Figuur 3.5: Een voorbeeld van radiale correctie volgens de tweede implementatie. Nu wordt er gebruik gemaakt van GML Toolbox om de verstorings en camera matrix te vinden. Op basis van deze matrices wordt de radiale correctie toegepast. De afbeelding linksboven is een verstoorde afbeelding. Het resultaat na radiale correctie is de afbeelding rechtsboven. Doordat centraal in het beeld de pixels een beetje uit elkaar worden getrokken, zullen er enkele punten onbekend blijven. In implementatie 1 was dit niet het geval omdat daar de afbeelding werd samengedrukt. Deze onbekende punten worden geïnterpoleerd in de onderste twee afbeeldingen met nearest neighbour.

3.4 Stap 2 - Edge detection

De keuze van filter is bij deze stap het moeilijkste. De kwaliteit van de edge image is sterk afhankelijk van de kwaliteit van de camera. In ons geval werkt de convolutiefilter uit Figuur 3.6 het beste om enkel het blad over te houden in het beeld. We gebruiken dergelijke grote waardes omdat deze een mooie balans geven tussen zo goed als geen ruis en zoveel mogelijk edges. Lagere getallen in de filter zouden er voor zorgen dat bijna geen edges zichtbaar zijn en hogere geven heel veel ruis waardoor de segmentatie geen onderscheid maar kan maken tussen edges en ruis. Een toepassing van deze filter op een voorbeeld is weergegeven in Figuur 3.7. Zoals u kunt zien, zijn naast het papier ook de randen van het projectiebeeld zichtbaar. Om deze niet als storende factor te laten werken op het lijndetectie algoritme kan er aan de convolutiefilter ook een margin worden meegegeven. Alles binnen deze margin zal weggeknipt worden uit het beeld.

−5	−5	−5
−5	39	−5
−5	−5	−5

Figuur 3.6: De convolutiefilter die in onze implementatie wordt gebruikt om de edge afbeelding te bekomen

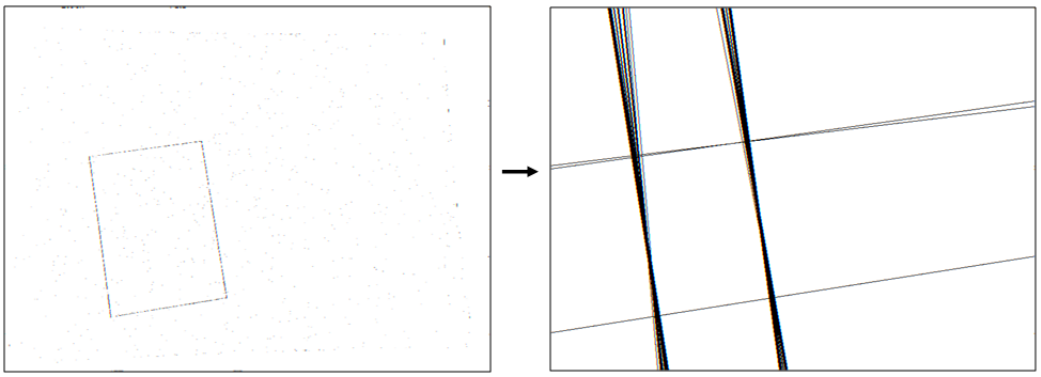


Figuur 3.7: De convolutiefilter uit Figuur 3.6, toegepast op een voorbeeld. Ook de randen van het projectiebeeld zijn zichtbaar. Deze worden weggeknipt door een margin in te stellen.

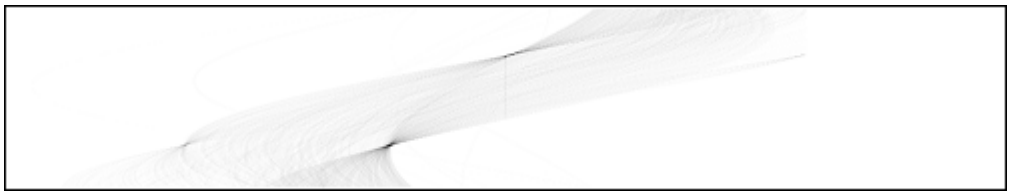
3.5 Stap 3 - Segmenteren

3.5.1 Hough

Voor de Hough transformatie hebben we ons gedeeltelijk gebaseerd op een implementatie van Yuan-Liang [Tan09b]. We nemen een bereik van $-90°$ tot $90°$ voor θ. Dit omdat het centrum van de lijnparametrisatie in het midden van de afbeelding ligt. Stel D de afstand van de linker bovenhoek van de afbeelding tot de rechter onderhoek. In ons geval is dit $D = \sqrt{800^2 + 600^2}$. Dan laten we ρ variëren van $\frac{-D}{2}$ tot $\frac{D}{2}$. De afstand kan negatief zijn doordat de oorsprong van het assenstelsel centraal in het beeld ligt. Bij negatief zal de lijn links in beeld liggen, bij positief rechts. Figuur 3.8 geeft een voorbeeld van lijndetectie met behulp van de Hough transformatie. De Hough transformatie die bij deze figuur hoort is weergegeven in Figuur 3.9.



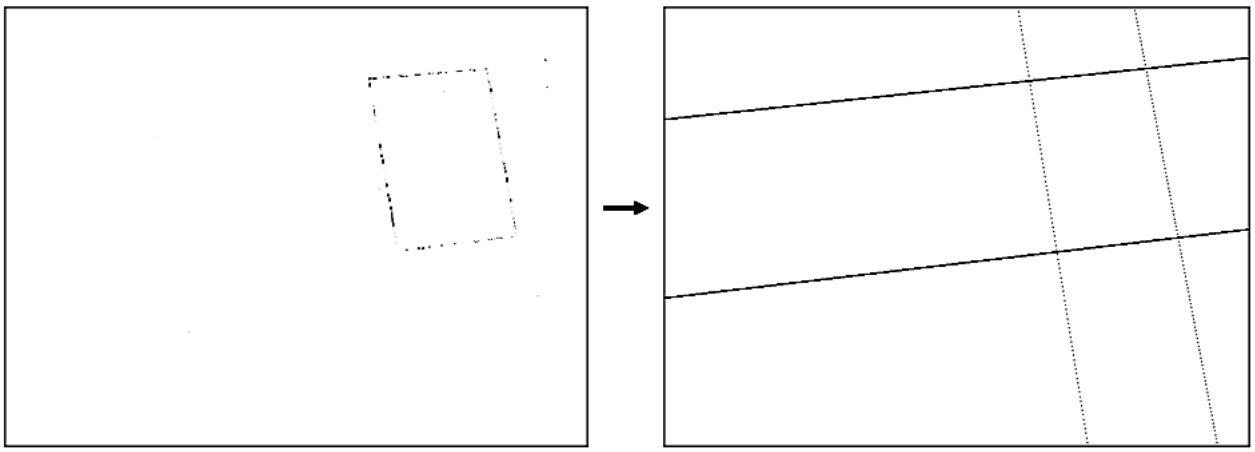
Figuur 3.8: Lijn detectie met behulp van de Hough transformatie toegepast op een detailbeeld.



Figuur 3.9: De Hough transformatie van Figuur 3.8. De lijnen met meer punten zullen een grotere uitstrijk hebben in de Hough transformatie. Hierdoor zullen er meer lijnen in de buurt boven de threshold vallen en zullen er meer lijnen op ongeveer dezelfde plaats worden gevonden. Lijnen met maar heel weinig punten zullen soms zelfs helemaal niet worden gevonden. Dit verklaart waarom in Figuur 3.8 de twee verticale lijnen heel veel resultaten geven en de horizontale zo goed als geen.

3.5.2 RANSAC

RANSAC is geïmplementeerd volgens het algoritme uit 2.2.3. Het is voldoende om 100 iteraties uit te voeren per lijn die wordt gezocht. Van deze 100 iteraties wordt vervolgens de beste lijn gekozen en al de punten die op de lijn liggen worden verwijderd uit de verzameling van punten. Voor het vinden van een papier is het voldoende om op deze manier 4 lijnen te zoeken. Dit algoritme toegepast op een voorbeeld geeft Figuur 3.10.



Figuur 3.10: Ransac toegepast op een detailbeeld van het papier.

3.6 Stap 4 - Papier tracken

Als eenmaal de segmentatie stap is uitgevoerd weten we de afmetingen van het papier. Om nu niet in elke frame de segmentatiestap opnieuw te moeten uitvoeren, kan het papier ook getracked worden. Hiervoor hebben we een condensation filter geïmplementeerd. De parameters die gebruikt worden om een blad te volgen zijn translatie in de horizontale en verticale richting en rotatie. Deze particle filter bevat 4 stappen: initialisatie, select, predict en measure. De specifieke implementatie details zullen voor elke stap nu in het kort worden besproken.

Initialisatie Deze komt grotendeels overeen met de initialisatie die besproken is in sectie 2.3.1. Er worden N aantal particles aangemaakt. Maar nu wordt elke particle niet random gekozen in het beeld, maar geïnitialiseerd op de positie en rotatie waar het blad zich nu bevindt. De rotatie van elke particle begint altijd met 0. Elke particle krijgt een gewicht van $\frac{1}{N}$ toegekend.

Select Ook deze komt overeen met de selection stap uit sectie 2.3.1. De particles worden dus gekozen naargelang het gewicht van de particles uit vorige frame of uit de initialisatie. In onze applicatie worden de particles niet random geselecteerd op basis van een kansverdeling volgens de gewichten, maar gaan de nieuw gegenereerde N particles verdeeld worden over elk gewicht dat voorkomt. Dus elke particle met een bepaald gewicht zal zijn aandeel van de particles krijgen naargelang zijn procentueel aandeel in het totale gewicht.

Predict Net zoals in sectie 2.3.1 is besproken, zullen de parameters van de particles (in ons geval horizontale translatie, verticale translatie en rotatie) Gaussiaans verdeeld worden. Dit wordt verwezenlijkt door een Gaussiaanse random waarde te genereren en vervolgens deze op te tellen met de parameter van de particle die wordt voorspeld. De Gaussiaanse verdeling kan worden benaderd door de som van meerdere uniform verdeelde random waardes. Neem bijvoorbeeld 3 maal een random waarde van -1 tot 1 en tel deze vervolgens met elkaar op. Het

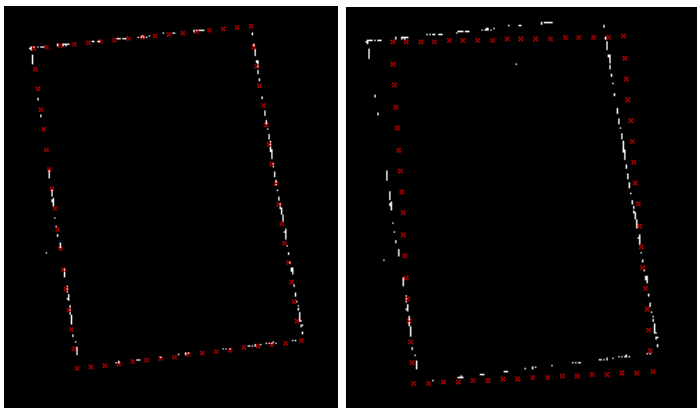
gevonden getal zal dan tussen -3 en 3 liggen. De kansfunctie $P(x)$ van de verschillende waardes is de volgende:

x	-3	-2	-1	0	1	2	3
$P(x)$	1	3	6	7	6	3	1

Met andere woorden zal meer in de directe omgeving worden gekeken dan verder. Per frame zal het blad namelijk niet veel verschuiven.

Measure Deze stap is afhankelijk van het object dat wordt gezocht. In onze implementatie zoeken we een papier. Gegeven zijn de vier hoekpunten van het papier en een edge image van de volgende frame. We gaan een reeks punten aanmaken die zich zouden bevinden op de rand van het papier als we een bepaalde translatie en rotatie toepassen. Door eerst de vier bekende punten te transformeren met de parameters van de prediction kunnen we X tussenliggende punten aanmaken. Deze punten worden verwacht op de lijn te liggen, dus ze zullen zichtbaar moeten zijn op het edge image. Het edge image geeft niet perfect de lijnen weer maar eerder fragmenten ervan. Met andere woorden zal niet elk punt een overeenkomst hebben. Het procentueel aantal punten van de prediction die overeenkomen met het aantal witte punten op het edge image zal gedefinieerd worden als het gewicht van de particle. Een voorbeeld van deze measure stap is gegeven in Figuur 3.11. De particle dat uiteindelijk het hoogste gewicht heeft, en dit gewicht boven een bepaalde threshold ligt, zal gekozen worden om geprojecteerd te worden. Indien de threshold niet gehaald wordt, zal er terug worden overgegaan op een segmentatiestap om de nauwkeurigheid terug bij te stellen.

Omdat in veel frames het blad blijft stil liggen, wordt er eerst een particle aangemaakt zonder translatie of rotatie. Als deze al voldoet aan de threshold, zal waarschijnlijk het blad niet verschoven zijn en wordt de projectie van de vorige frame behouden.



Figuur 3.11: Deze figuur geeft de punten weer die gecheckt worden in de measure stap van de condensation filter. De witte punten komen overeen met de edges van de nieuwe frame (de observation) en de rode punten komen overeen met de prediction. Het procentueel aantal rode punten dat overeenkomt met witte punten bepalen het gewicht van de onderzochte particle. De linker figuur zal een groot gewicht toegekend krijgen. In de rechter figuur heeft de prediction een te grote rotatie waardoor er niet veel punten zullen overeenkomen, dit resulteert in een laag gewicht.

3.7 Stap 5 - Projectiebeeld genereren

Uit de vier lijnen die gevonden zijn in de vorige stap, kunnen de hoekpunten van het papier worden berekend. Op basis van deze hoekpunten kan er vervolgens een beeld worden opgesteld dat we willen projecteren op het papier. In eerste plaats projecteren we een polygoon met als hoekpunten de hoekpunten van het blad. Een volgende stap zou zijn het projecteren van een afbeelding. Als we de projectie toepassen stoten we op een probleem. De projectie van een object zal namelijk de volgende frame verstoren.

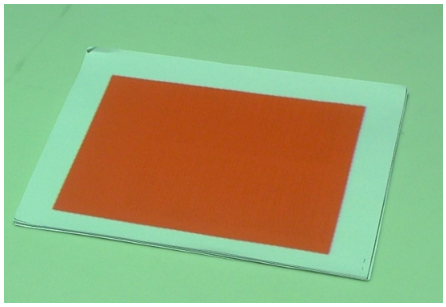
Pollutie van vorige frame Bij lijndetectie rijst er zich nog een probleem. Als in een vorige frame het totaal proces gelukt is, en er dus correct een afbeelding op het papier wordt geprojecteerd, zal in de frame die daarop volgt een slecht detailbeeld worden gevonden. Dit omdat in de edge image ook het detail van de geprojecteerde afbeelding aanwezig is. Dit probleem kan opgelost worden door iteratief het geprojecteerde beeld af te trekken van het nieuwe detailbeeld. Zo zal er niet gekeken worden naar plaatsen waar al iets geprojecteerd is. Nu als de projectie zo genomen wordt dat deze exact overeenkomt met het papier zal dus ook het detail van het papier genegeerd worden. Om dit te vermijden is het best de projectie een klein beetje kleiner nemen zodat de rand van het papier nog zichtbaar is. De geprojecteerde afbeelding wordt bepaald door de vier hoekpunten. Deze kunnen met behulp van transformatie matrices geschaald worden naar het middelpunt van het papier. Het middelpunt verkrijgen we door de coördinaten van de vier hoekpunten uit te middelen. Het is niet voldoende om enkel een schaalmatrix te gebruiken. Deze gaat de punten schalen naar de oorsprong en niet naar het middelpunt van papier. Er moet een

combinatie van een schaalmatrix en translatiematrix worden gebruikt, namelijk

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & -t_x \\ 0 & -t_y \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & x_y \end{bmatrix} \begin{bmatrix} 0 & t_x \\ 0 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.1}$$

met (x', y') de nieuwe coördinaten van het punt (x, y), (t_x, t_y) de coördinaten van het middelpunt van het papier en (s_x, s_y) de schaalfactoren voor respectievelijk de x en y coördinaat. Kort samengevat gaat het papier naar de oorsprong worden verplaatst zodat het middelpunt samenvalt met de oorsprong. Vervolgens wordt de schalering uitgevoerd. Als laatste stap wordt het papier terug op zijn originele positie geplaatst.

Na de punten geschaleerd te hebben, kan het object geprojecteerd worden. Dit geeft Figuur 3.12.

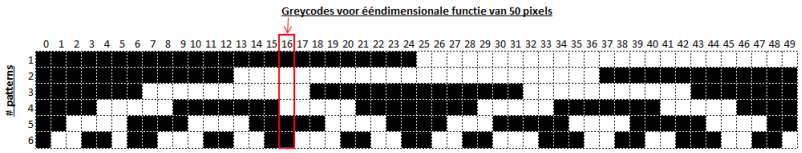


Figuur 3.12: Projectie van een object op de hoekpunten van het papier, nadat deze geschaald zijn. Als deze projectie van de volgende frame wordt afgetrokken zal het detailbeeld van de volgende frame niet meer verstoord zijn.

3.8 Stap 6 - Calibreren

3.8.1 Graycodes

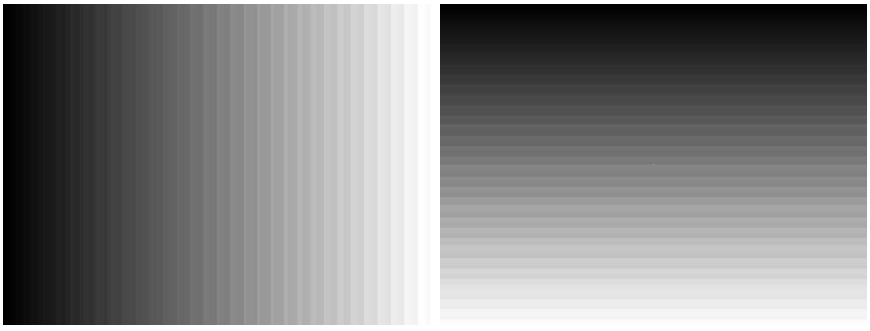
Het aanmaken van binaire patronen is voor een resolutie, bestaande uit een macht van 2, geen probleem. Het beeld kan tot op pixel niveau gehalveerd worden. Als we echter werken met een beeld van 800 op 600, wat in onze implementatie het geval is, is het moeilijker om te blijven delen door 2. Daarom hebben wij gekozen voor een bottom-up alternatief. We starten vanaf pixelniveau en gaan per patroon het aantal pixels van een gebied verdubbelen. Ook hier gaan we gebieden van dezelfde kleur twee aan twee samen nemen om het veelvuldig samenvallen van grenzen te vermijden. Zo zullen we uiteindelijk uitkomen op een binair patroon met een resolutie van 1024 op 1024, maar afgesneden op 800 en 600. Figuur 3.13 toont een voorbeeld waar de graycodes bottom-up zijn opgesteld.



Figuur 3.13: Een voorbeeld waar de resolutie van de patronen geen macht van 2 zijn. De patronen zijn aangemaakt, beginnende van laag 7. De graycode voor bijvoorbeeld pixel 16 zal 0011000 zijn.

Test de correctheid Er bestaat een test om na te kijken of de graycodes correct zijn opgesteld. Elk horizontaal of verticaal punt kan voorgesteld worden door zijn overeenkomstige graycode. Een dergelijke graycode is een binair getal. Bijvoorbeeld de graycode die bij kolom 49 hoor is 0101001. Deze graycode kan omgevormd worden naar zijn overeenkomstige binaire code (Herinner: de binaire code is een alternatief patroon) [Tan09a].
Stel een graycode $g_1 g_2 g_3 \dots g_{n-1}, g_n$. Er wordt begonnen met de n^{de} bit en bereken $\sum_n = (\sum_{i=1}^{n-1} g_i) \bmod 2$. Als $\sum_n$ gelijk is aan 1, moet g_n worden vervangen door $1 - g_n$, anders laat het onveranderd. Vervolgens herhaal de formule voor de graycode zonder de laatste bit. Dus bereken nu $\sum_{n-1} = (\sum_{i=1}^{n-2} g_i) \bmod 2$. Herhaal dit tot alle bits zijn afgegaan. De resulterende nummer $d_1 d_2 d_3 \dots d_{n-1} d_n$ is de binaire code die werd gezocht.

Door de intensiteitswaarde van elke pixel in te stellen op de decimale waarde van het overeenkomstige binaire getal zouden we een beeld moeten bekomen waar er een mooie overgang is van wit aan de ene naar zwart aan de andere kant. De intensiteitswaarde kan best genormaliseerd worden om geen overflow te krijgen van intensiteiten binnen het beeld. Een dergelijk beeld wordt ook wel een ratio image genoemd. Figuur 3.14 toont de ratio afbeeldingen, afkomstig van de applicatie.

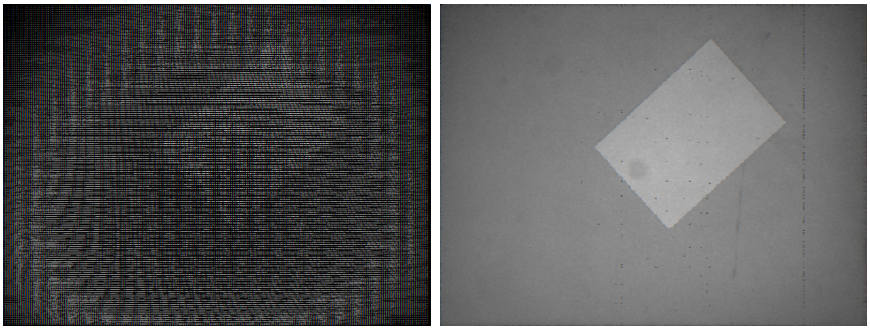


(a) Ratio Verticaal (b) Ratio Horizontaal

Figuur 3.14: De ratio images voor de verticale en horizontale graycodes. Deze beelden zijn opgesteld door de graycodes om te vormen naar hun oorspronkelijke binaire code. Als de decimale waarde van de binaire code wordt genormaliseerd naar een intensiteitswaarde moet de ratio afbeelding mooi gradieel overgaan van zwart naar wit of van wit naar zwart.

3.8.2 Mappen van een camerabeeld naar een projectorbeeld

Als we de mapping van punten hebben bekomen[1], kunnen we handmatig, voor elk punt in het inputbeeld, het outputbeeld samenstellen. Dit zou dan een gecalibreerd beeld geven bijvoorbeeld Figuur 3.15a. Zoals u ziet, zijn er nog veel zwarte punten aanwezig. Dit zijn allemaal punten waarvoor de calibratie via graycodes geen mapping heeft gevonden. Om toch alle punten te weten te komen, wordt elke onbekende mapping nearest neighbour geïnterpoleerd. Dit geeft het resultaat voorgesteld in Figuur 3.15b.

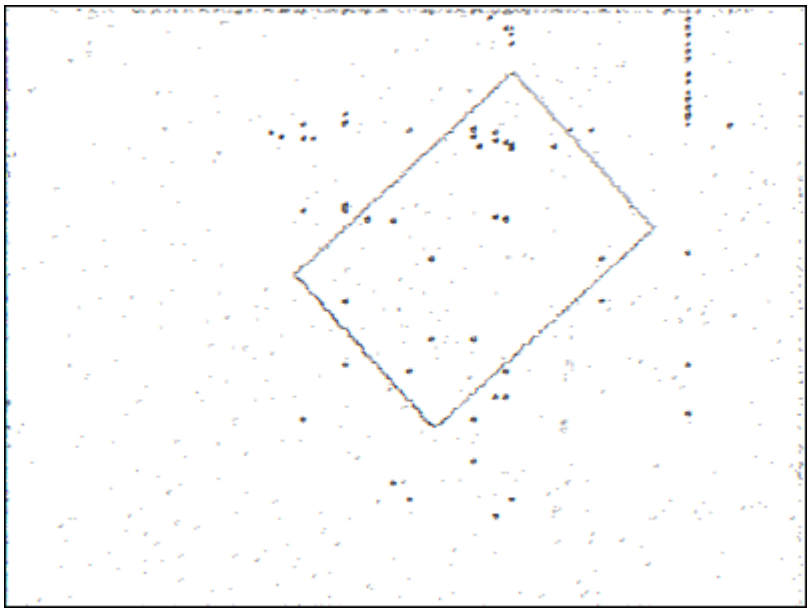


(a) (b)

Figuur 3.15: Afbeelding (a) is het gecalibreerde beeld. Voor elk zwart punt is er in de mapping geen overeenkomstige coördinaat gevonden. Deze punten worden in afbeelding (b) nearest neighbour geïnterpoleerd.

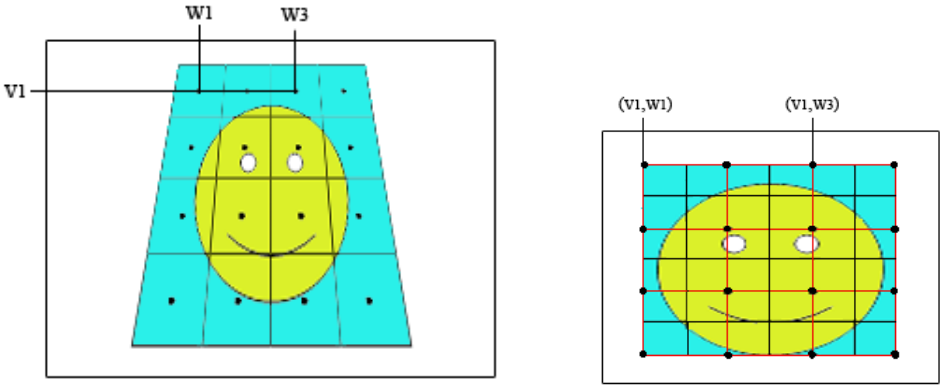
Deze methode heeft echter een probleem. Figuur 3.15b geeft een degelijk resultaat voor de calibratie. Daarenboven werkt de mapping snel. Waarom zal dit dan toch problemen geven? Als we benadering 2 gebruiken zal in eerste instantie het beeld gecalibreerd worden. Na deze calibratie zal er lijndetectie worden toegepast. De meeste lijndetectie algoritmes verwachten als inputbeeld een edge image. Dit is een beeld waar de abrupte overgangen sterk geaccentueerd worden. Als we van het geïnterpoleerd beeld een edge image maken, geeft dit Figuur 3.16. We zien dat er veel ruis aanwezig is. Als de ruis mooi verdeeld is zal dit geen sterke invloed hebben op het lijndetectie algoritme. Maar nu hebben we aan de rechterkant een samenhang van ruis die juist op één lijn ligt, waardoor het algoritme dit als lijn gaat detecteren. Het is niet duidelijk waarom deze ruis aanwezig is.

[1]Dit is in ons geval onder de vorm van een 2D tabel voor de x-as en een 2D tabel voor de y-as. De positie in de tabel komt overeen met de coördinaat van het input beeld. De waarde in de tabel is de coördinaat waarop het punt moet worden gemapped



Figuur 3.16: Edge image van figuur (3.15b). Er is te veel ruis aanwezig voor lijndetectie toe te passen. Meer specifiek er is te veel gecorreleerd ruis aanwezig. De lijndetectie zal eerder een lijn fitten door de bolletjes aan de rechterkant die ook op één lijn liggen.

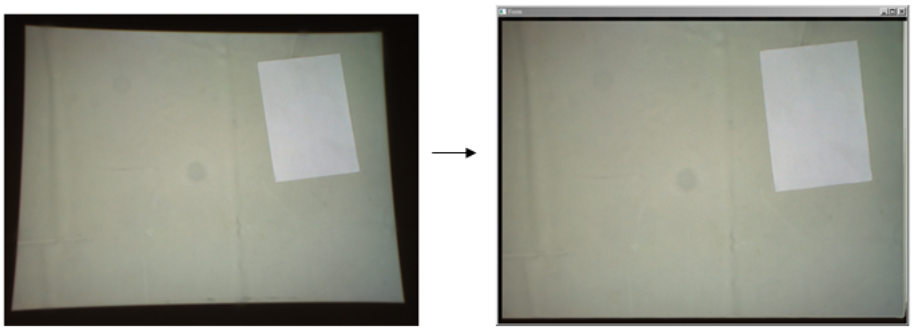
Lineaire Interpolatie op GPU Wij hebben ervoor gekozen om te interpoleren op de GPU. Dit werkt sneller en gaat betere resultaten geven omdat we nu lineair interpoleren. We maken gebruik van OpenGL om dit te verwezenlijken. Het camerabeeld wordt als texture op een grid geplaatst. Dit grid bevat al de bekende mappings. De onbekende mappings zullen vervolgens lineair geïnterpoleerd worden. Naargelang het aantal bits dat gebruikt wordt voor een graycode, zal het inputbeeld verdeeld zijn in vakjes van dezelfde graycode. Stel bijvoorbeeld Figuur 3.17a. Hier zijn graycodes gebruikt van 3 bits groot. Dit resulteert in 16 velden. De bedoeling is om een grid op te stellen bestaande uit quads van dezelfde grootte als de vakjes in Figuur 3.17a. Als nu het camerabeeld als texture ingeladen wordt op de grafische kaart, kunnen de hoekpunten van de quads gekoppeld worden met de texturecoördinaten. Deze texturecoördinaten kiezen we zo, zodat het centrum van een vakje in Figuur 3.17a (bijvoorbeeld (v_1, w_1)) samenvalt met een hoekpunt van een quad. Alle punten tussen de hoekpunten van een quad zal lineair geïnterpoleerd worden met de juiste pixels uit de texture. De grid is voorgesteld in Figuur 3.17a. De rode vakjes komen overeen met de quads in het grid. Door het plaatsen van de grid in een display list hoeft enkel realtime het camerabeeld op de grafische kaart worden geplaatst om deze direct op de gewenste wijze renderen.



(a) (b)

Figuur 3.17: Links het inputbeeld. Dit beeld is ingedeeld in vakjes van dezelfde graycode. Voor elk vakje kan het centrum gemapped worden van cameracoördinaten naar projectorcoördinaten. Voor deze mapping wordt in OpenGL een grid opgemaakt. OpenGL gaat vervolgens elk tussenliggend punt interpoleren waardoor de hele afbeelding zal worden gecalibreerd. Het resultaat geeft afbeelding b. Voor de duidelijkheid staat in afbeelding (a) twee punten (v_1, w_1) en (v_1, w_3) aangegeven. Ook in afbeelding (b) zien we deze terugkomen. Hier stellen ze echter niet de coördinaten voor maar wel de texture coördinaten die aan dat punt zullen worden gekoppeld.

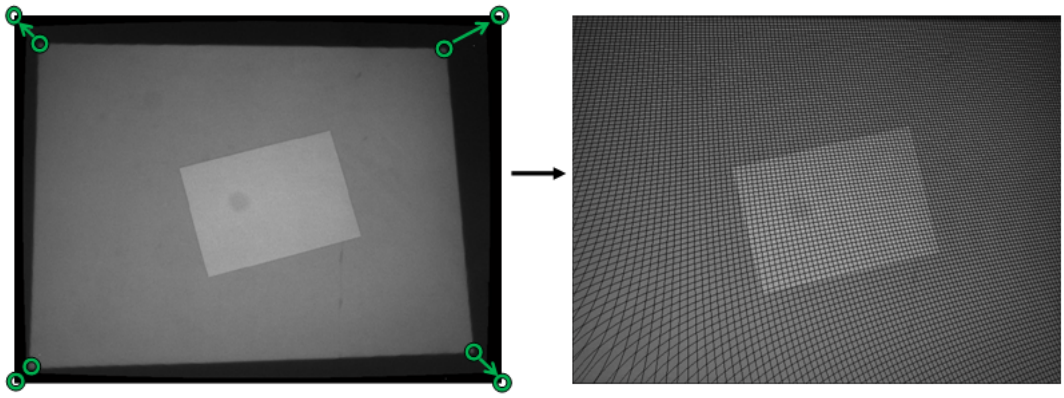
Als we dit toepassen in onze calibratie krijgen we het resultaat dat te zien is in Figuur 3.18. Het voordeel van deze aanpak is dat deze ook goed werkt voor minder bits per graycode. Het voorbeeld is gerendered met een graycode van 6 bits. Wetende dat de het maximaal aantal bits 11 is, is dit een grote hap in de render tijd. Om een grid op te stellen voor een calibratie met diepte 11 moet er voor elke pixel een vertex voorzien worden in het grid. Een graycode van 6 bits geeft maar een grid met 1024 vertices. Dit is weinig ten opzichte van 480000 vertices.



Figuur 3.18: Voorbeeld van calibratie met behulp van OpenGL. Het beeld wordt gerendered op een grid die de mapping verzorgd tussen de twee coördinatenstelsels.

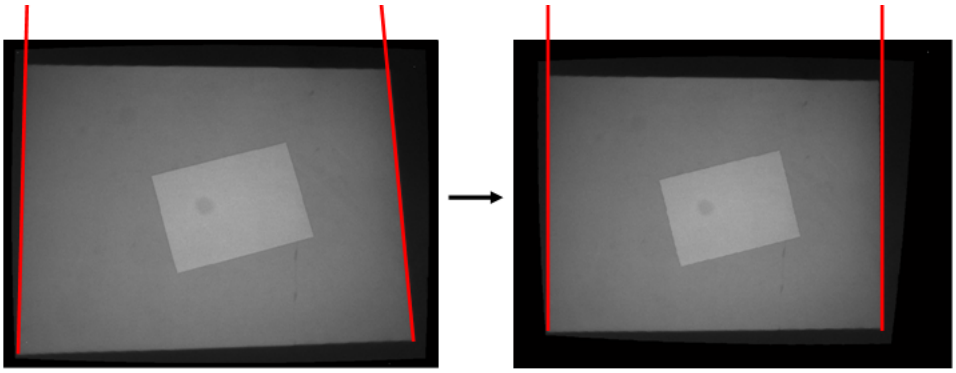
3.8.3 Projectieve correctie

Herinner uit sectie 2.1.4 dat we voor dit algoritme vier gekende mappings moeten hebben om de vergelijkingen te kunnen opstellen. We weten, omdat we het cameravlak en het projectievlak willen laten samenvallen, dat de vier hoekpunten van de projectie moeten overeenkomen met de hoekpunten van de afbeelding. De punten die moeten overeenkomen zijn in Figuur 3.19 aan de linkerzijde aangeduid met groene punten. De vier hoekpunten van de projectie kunnen we met behulp van RANSAC[2] bepalen. Eens we deze punten hebben, kunnen we de acht vergelijkingen met acht onbekenden opstellen. Om de acht onbekende te bepalen moeten we de matrix uit 2.6 oplossen. Dit kan gemakkelijk verwezenlijkt worden door eerst de matrix in triangulaire vorm te zetten en vervolgens van onder naar boven de waardes aflezen en invullen in de vergelijking er boven. Als nu de homogene transformatie op elk punt van de afbeelding toegepast wordt, krijgen we als resultaat Figuur 3.19 rechts. Ook hier krijgen we weer onbekende punten die met behulp van interpolatie moeten worden berekend. Het is ook mogelijk om er voor te zorgen dat het resultaat kleiner is als het beginbeeld. Dit kan door op dezelfde manier te werken, maar alvorens de transformatie toe te passen de inverse matrix te berekenen en deze te gebruiken als transformatie. Een voorbeeld hiervan is te zien in Figuur 3.20. De rode lijnen in de figuur geven aan dat na correctie de randen van de projectie terug evenwijdig lopen.



Figuur 3.19: Toepassing van een projectieve correctie. Links geven de groene punten aan met welke punten ze overeenkomen. Rechts het resultaat.

[2]Met behulp van RANSAC de vier raaklijnen bepalen van de projectie. Vervolgens de vier snijpunten bepalen van deze lijnen.



Figuur 3.20: Toepassing van projectieve transformatie, maar nu met een verkleind resultaat. De rode lijnen geven aan dat de randen van de projectie na correctie terug evenwijdig lopen.

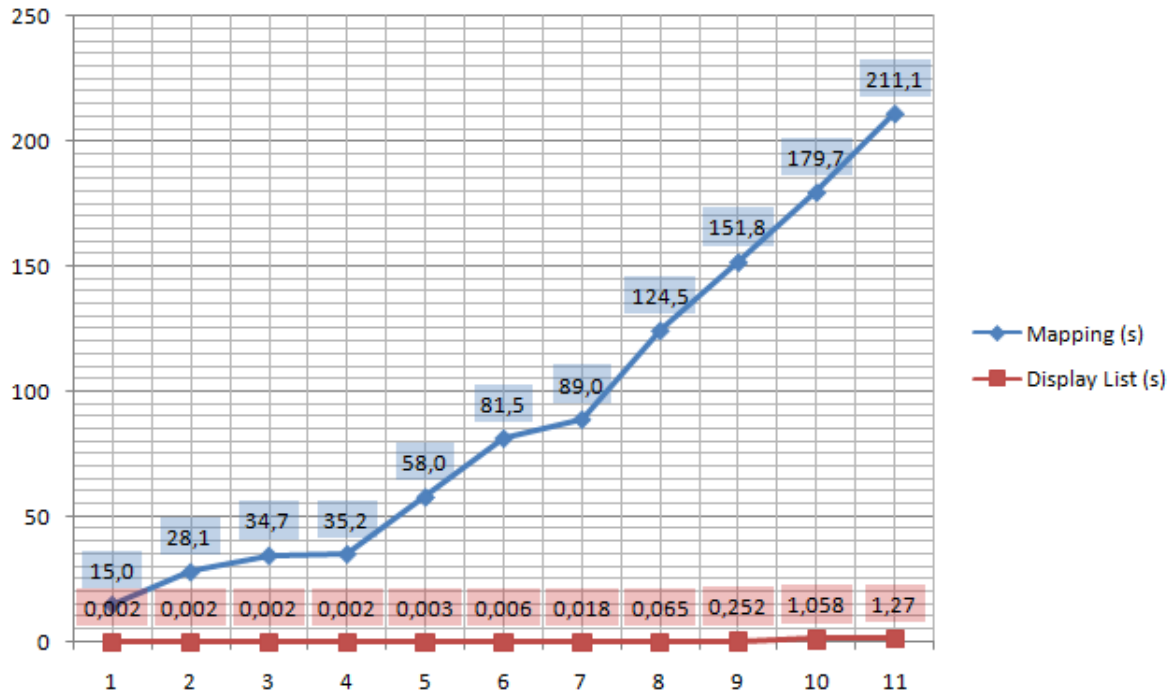
Hoofdstuk 4

Resultaten

In dit hoofdstuk bespreken we eerst de haalbaarheid van elk algoritme afzonderlijk. In het bijzonder toetsen we hun performantie. Als laatste bespreken we de snelheid van het gehele proces. Hierbij zullen we een indeling maken volgens de benadering die we hebben gebruikt. De resultaten zijn berekend op een Intel Core 2 Duo CPU (aan 2.10 GHz) met 4.00 GB RAM systeem. Het gebruikt een ATI Mobility Radeon HD 2600 grafische kaart.

4.1 Calibratie

Graycodes Het voordeel van calibratie met behulp van graycodes is dat deze stap niet realtime uitgevoerd wordt. Het volstaat om éénmalig de mapping te bepalen voor een vaste opstelling en deze voor elke frame te gebruiken. Als zo'n mapping opgeslagen is in een 2D-matrix per coördinaat, zal de opvragingstijd hiervan nihil zijn. Figuur 4.1 geeft een overzicht van de duur van calibratie voor verschillende dieptes van graycodes. Het is opgedeeld in twee verschillende tijden. Een tijd om de mapping van de twee coördinatenstelsels op te stellen en een tweede tijd die aangeeft hoe lang het duur om de display list in OpenGL aan te maken (zodat een inkomende afbeelding zeer snel kan worden gecalibreerd). Zoals u kunt zien is de grafiek van de mapping vrij lineair. Wat betekent dat de tijd van uitvoering met ongeveer een vast aantal zal stijgen naargelang er een bit wordt toegevoegd. Het aanmaken van een display list is geen tijdsintensieve stap. Het aantal bits per graycode zal enkel een invloed hebben op het aantal punten in het grid en niet het aantal keren dat de afbeeldingen worden doorlopen. Het algoritme zal altijd de afbeelding maar één keer moeten doorlopen. Het aantal punten in het grid komt overeen met het aantal gebieden van dezelfde graycode. Dit is dus maximaal 1 punt per pixel. Dit correspondeert met een maximale tijd van ongeveer 1270 ms. Het voordeel van deze techniek is dat voor een klein aantal bits per graycode hij al een heel goede benadering vindt voor de calibratie. Een graycode van 5 bits is al voldoende om een zo goed als correcte calibratie te vinden. Op minder als een minuut kan deze in een voorbereidende stap worden bekomen.



Figuur 4.1: Grafiek die weergeeft hoeveel tijd er nodig is om een calibratie uit te voeren als er een X aantal bits wordt gekozen voor de graycodes.

Radiale Correctie Deze techniek is redelijk performant te implementeren. In onze applicatie heeft hij plus minus 25 milliseconde nodig. Hiervan is 5 milliseconde voor het algoritme zelf en 20 milliseconden voor de nearest neighbour interpolatie. In sommige gevallen kan de interpolatie wel een grote bottleneck zijn met uitschieters tot 500 ms. Meestal is dit niet het geval en met andere woorden kan de radiale correctie realtime gebruikt worden in de applicatie.

Projectieve Correctie Herinner dat voor de projectieve correctie de vier hoekpunten van de projector moeten gekend zijn alvorens men de transformatiematrix kan opstellen. Deze kunnen bekomen worden met een segmentatie techniek uit sectie 2.2. Dit kan als preprocessing stap gebeuren dus heeft geen invloed op de verwerkingstijd van 1 frame. Het probleem dat we hebben in onze implementatie is de onnauwkeurigheid van het resultaat. Zoals u in Figuur 3.20 kan zien, geeft onze projectieve correctie een zwarte band in de rechter bovenhoek. Dit is nog een stuk beeld dat niet bij de projectie hoort. Dit heeft waarschijnlijk te maken met de benadering van de radiale correctie. De radiale correctie is met andere woorden wel goed genoeg om lijnen te herkennen, maar zal niet goed genoeg zijn om ook een projectieve correctie te kunnen toepassen. Deze methode werkt dus niet goed genoeg om te kunnen gebruiken voor calibratie. De tijd die het algoritme nodig heeft (indien de vier hoekpunten al gekend zijn), bedraagt 84 ms. Ook hier zou in een preprocessing de mapping kunnen worden opgesteld in een 2D matrix waardoor de opvraagtijd nihil wordt.

4.2 Segmentatie

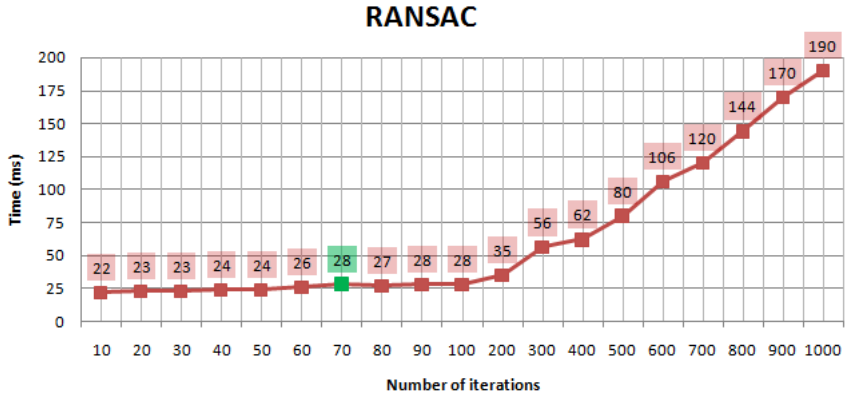
Hough Ook Hough heeft twee grote problemen waaruit we kunnen besluiten dat hij niet in aanmerking komt om realtime te gebruiken.

1. Ten eerste geeft hij enkel maar lijnen terug die heel uitgesproken aanwezig zijn in de afbeelding. Zoals in het vorige hoofdstuk, kan het zijn dat sommige lijnen helemaal niet gevonden worden. Als dan toch al de gewenste lijnen worden gevonden, is de kans groot dat de meer uitgesproken lijnen ook meerdere resultaten geven. Er moet dus ook nog kunnen bepaald worden welke lijnen bijeen horen en dus als één resultaat moeten worden beschouwd.

2. Ten tweede is het een zéér traag algoritme. Momenteel heeft hij 11817 ms of nog plus minus 12 seconden nodig om éénmaal de Hough lijndetectie uit te voeren. Dit is dus véél te traag om realtime toe te passen. De snelheid is sterk afhankelijk van de resolutie van de parameterruimte. In onze implementatie gebruiken we een resolutie van 360 op 500. De parameterruimte wordt opgedeeld in vakjes van gelijke hoekgraad. De resolutie van 500 is de afstand van de lijn tot het middelpunt. Deze is 500 omdat de afstand van het midden van het papier tot een buitenste hoek maximaal 500 bedraagt.

RANSAC Een veel betere keuze voor lijndetectie dan de Hough lijndetectie is RANSAC. Het heeft drie voordelen. Hij zal ten eerste niet tweemaal dezelfde lijn detecteren. Dit omdat de punten die al binnen een offset van een lijn vallen, verwijderd worden uit de verzameling van alle punten. Ten tweede, een detailbeeld met relatief veel ruis zal meestal toch de goede lijnen vinden. Zelfs al wordt er een klein aantal iteraties gedaan. En ten laatste, het algoritme werkt vrij snel. De grafiek in Figuur 4.2 geeft een overzicht van de tijden die nodig zijn om het RANSAC algoritme uit te voeren met een specifiek aantal iteraties. Het groene punt geeft aan vanaf hoeveel iteraties de vier lijnen van het papier zo goed als altijd wordt herkend. Zoals u kunt zien is het eerste deel van de grafiek bij benadering vlak. Er kan dus nog een extra marge worden genomen in het aantal iteraties. De tijd (28 ms) nodig voor het groene punt (70 iteraties) komt overeen met de tijd die nodig is voor 100 iteraties. We kunnen dus even goed 100 iteraties nemen.

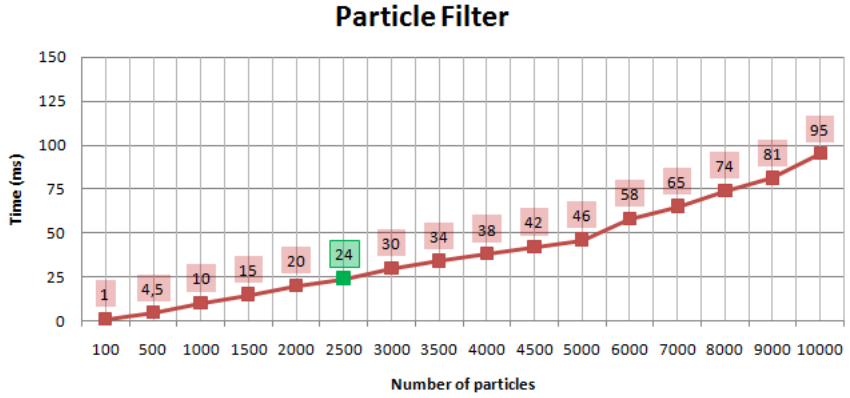
Een mogelijkheid om het RANSAC algoritme te versnellen is het edge beeld te downsamplen. Door deze downsampeling zullen er minder edges overschieten, maar nog genoeg om de lijnen te herkennen. Hierdoor zal RANSAC veel minder edges twee aan twee moeten afgaan waardoor het algoritme sneller zal werken.



Figuur 4.2: Deze grafiek toont de verhouding van het aantal iteraties binnen het RANSAC algoritme en de tijd die hiervoor nodig is. Het groene punt geeft aan vanaf wanneer het algoritme robuust begint te werken.

4.3 Tracking

De snelheid van het condensation algoritme hangt af van het aantal particles dat wordt genomen. De grafiek uit Figuur 4.3 toont de verhouding van het aantal particles ten opzichte van de tijd die nodig is om het algoritme uit te voeren. Het groene punt geeft aan vanaf welke hoeveelheid particles het algoritme goed begint te werken. Ook hier kan best een marge genomen worden om zeker te zijn dat het algoritme goed werkt. We nemen dus standaard ongeveer 3000 particles waardoor het ongeveer 30 ms duur om 1 frame te verwerken.



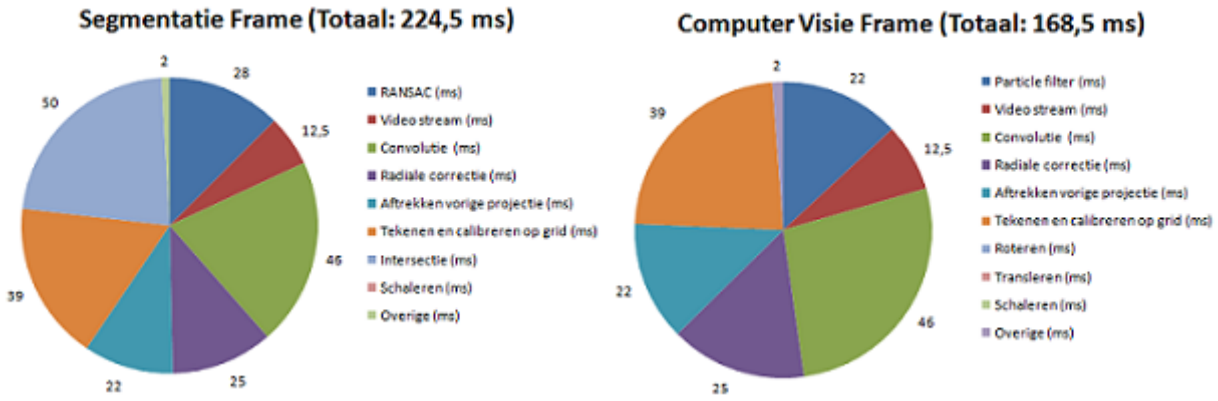
Figuur 4.3: Deze grafiek toont de verhouding van het aantal particles gebruikt in het condensation algoritme en de tijd die nodig is om het algoritme uit te voeren. Het groene punt geeft aan vanaf wanneer het algoritme robuust begint te werken.

4.4 Totaal Proces

We kijken nu naar de duur van het totale proces voor 1 frame. In onze implementatie hebben we twee benaderingen uitgeprobeerd. Deze zullen we nu ook afzonderlijk bespreken.

4.4.1 Benadering 1

Herinner dat in deze benadering de radiale correctie afzonderlijk gebeurt en dat de calibratie pas als laatste stap wordt uitgevoerd. Het lijkt misschien overbodig om al de moeite (zoals het zoeken van de verstoringsmatrix met behulp van de GML Toolbox) te doen om radiale correctie afzonderlijk uit te voeren. Vooral omdat de calibratie met behulp van graycode patronen deze vervorming mee kan opheffen. Toch kan het slim zijn om deze apart uit te voeren. Zoals eerder al vermeld kunnen we, als eerst een radiale correctie wordt uitgevoerd, direct op het camerabeeld verder werken zonder al reeds te calibreren. Dit heeft als voordeel dat niet eerst het beeld moet gecalibreerd worden op het grid en dan er terug moet worden afgehaald om er aan verder te werken. Nu kan gewoon het proces gebeuren en pas achteraf op het grid worden geplaatst zonder dat het gerenderde beeld nog moet worden gedownload. De benadering is opgedeeld in twee soorten frames. Er is een segmentatie frame die met behulp van RANSAC het papier detecteert en de vier hoekpunten ervan berekent. In een tweede soort frame gaat we met behulp van de condensation filter het papier trachten te tracken. Deze laatste frame noemen we de computervisie frame. In Figuur 4.4 worden de tijdsverdelingen van beide soorten frames weergegeven. In deze figuur valt duidelijk op dat de computer visie frame sneller werkt dan de segmentatieframe. Dit is ook wat we verwachtten en de reden waarom we het algoritme toegevoegd hebben. Nochtans is de duur van de particle filter en RANSAC ongeveer dezelfde. Beide rond de 25 ms. Het verschil ligt hem vooral in het feit dat na RANSAC ook nog de vier snijpunten van de vier lijnstukken moeten worden berekend. Dit heeft nog een extra 50 ms nodig.



(a) Segmentatie frame

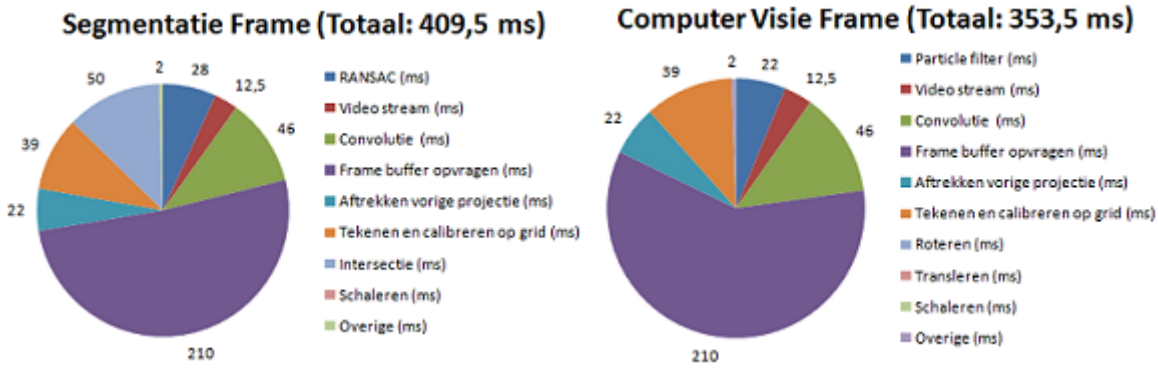
(b) Computer visie frame

Figuur 4.4: De tijdsverdelingen van de verschillende stappen voor benadering 1 in respectievelijk (a) de segmentatieframe en (b) de computer visie frame.

We trachten de particle filter zo goed mogelijk fijn te stellen zodat er maar heel zelden een segmentatie frame moet worden berekend en we zo goed mogelijk het papier proberen te tracken. In de grafiek 4.4b wordt er gebruik gemaakt van ongeveer 2500 particles. Dit kan opgetrokken worden tot bijna 9000 particles om aan dezelfde snelheid te komen als een segmentatie frame. Door bijvoorbeeld het aantal particles op 4000 te zetten, werkt de frame nog altijd sneller en zal hij meer kans hebben om een tracking resultaat te vinden. Hierdoor moet maar om de paar seconden een segmentatie frame uitgevoerd worden.

4.4.2 Benadering 2

De tweede benadering gaat eerst het camerabeeld calibreren naar projectorcoördinaten. Hierdoor wordt er direct verder gewerkt in de goede ruimte. Het nadeel is dat de framebuffer moet worden opgevraagd van het gerenderde camerabeeld op het grid. Het verschil in tijd zal dus sterk afhangen van het verschil in duur van de radiale correctie en het downloaden van de framebuffer. Ook hier zijn er twee soorten frames: het segmentatie frame en computer visie frame. De tijdsverdeling voor beide frames is weergegeven in Figuur 4.5. Alle stappen zijn zo goed als analoog aan deze van benadering 1. Het enige verschil is dat in plaats van radiale correctie nu de frame buffer wordt opgevraagd. Zoals u kunt zien geeft deze een enorme minderwaarde aan de totale tijd. De framebuffer opvragen duurt namelijk 185 ms langer als het radiaal corrigeren. Dit is het duidelijke bewijs dat benadering 1 veel beter is als benadering 2. Hierbij moet wel de nuance gemaakt worden dat het opvragen van de framebuffer gebeurt met Qt [Tro09]. Deze vergt nog veel kopieën naar Qt compatibele afbeeldingsformaten. Rechtstreeks deze framebuffer opvragen zal waarschijnlijk sneller gaan. Toch kan het downloaden van de grafische kaart best vermeden worden.



(a) Segmentatie frame

(b) Computer visie frame

Figuur 4.5: De tijdsverdelingen van de verschillende stappen voor benadering 2 in respectievelijk (a) de segmentatieframe en (b) de computer visie frame.

4.5 Discussie

4.5.1 Bruikbaarheid

Voor een opstelling waar de objecten doorheen de wereld niet snel bewegen en voornamelijk statisch blijven liggen werkt deze applicatie goed. RANSAC is een robuuste manier om objecten te vinden. Het voordeel van de particle filter is dat de herkenningsstap maar éénmaal moet worden uitgevoerd. Er moet dus maar om de paar seconden een segmentatie frame toegepast worden. Bijkomend voordeel is dat enkel RANSAC nadeel ondervindt van de vervuiling in het edge image. De condensation filter heeft hier geen last van. Met andere woorden als een object herkend is kunnen andere objecten het tracking algoritme niet verstoren zolang het te volgen object binnen een beperkte afstand wordt verder bewogen. Dit maakt ook het tracken van het papier zeer robuust.

Anderzijds als men een opstelling heeft waar de objecten wel snel kunnen bewegen zal de bruikbaarheid aanzienlijk dalen. De gebruiker verwacht immers dat de projectie smooth meebeweegt met het object. Dit is in onze applicatie nog niet helemaal het geval. Ook het nog niet wegfilteren van de handen en armen komt de bruikbaarheid niet ten goede. Oplossing is het toepassen van een skin detection algoritme om vervolgens deze weg te filteren.

4.5.2 Snelheid

Over het algemeen werkt de implementatie nog vrij traag. Dit kan het gevolg zijn van het weinig performant implementeren van de algoritmes. Nu zullen er een aantal oplossingen aangeboden worden die het proces kunnen versnellen.

Resolutie De resolutie van het beeld heeft een invloed op de verwerking. In plaats van een resolutie van 800 op 600 zou een resolutie van 600 op 400 ook al voldoende kunnen zijn voor deze doeleinden. Met als gevolg dat er veel minder pixels moeten worden afgelopen, waardoor elk algoritme aanzienlijk gaat stijgen in snelheid. Enkele voorbeelden: afbeeldingen sneller kopiëren, upload en/of download van respectievelijk naar GPU versnelt, minder punten om te calibreren, ... Doordat de implementatie niet snel genoeg is, gaat er een deel van de realtime feeling verloren. Het zou veel intuïtiever zijn als de projectie smooth meebeweegt met de objecten in plaats van de vertraging die er nu op zit.

GPU Een tweede oplossing die gegarandeerd een versnelling teweeg brengt is het programmeren van de algoritmes op de GPU. Algoritmes zoals RANSAC, convolutie, interpolatie, radiale correctie en particle filter zijn perfect paralleliseerbaar. Dit maakt ze uitermate geschikt voor te implementeren als shader op de GPU. Elke particle van de condensation filter kan bijvoorbeeld parallel worden berekend. In feite kan het hele proces van een frame van begin tot eind op de GPU worden berekend. Dit zal een serieuze snelheidswinst opleveren wat de bruikbaarheid zeker ten goede komt. Enkele voorbeelden van shadingtalen zijn Cg [NVi09a] en Cuda [NVi09b].

4.5.3 Uitbreidbaarheid

Voorlopig kan deze applicatie enkel maar papieren van een rechthoekige vorm herkennen. De segmentatiestap zoals RANSAC is perfect uitbreidbaar voor de herkenning van andere wiskundige modellen. Elk model dat wiskundig kan worden voorgesteld met een aantal parameters kan door RANSAC worden herkend. Daarnaast bestaan er ook zeer geavanceerde libraries voor de herkenning van specifieke objecten. Deze kunnen gebruikt in plaats van RANSAC. Net zoals RANSAC kan op dezelfde manier de particle filter worden uitgebreid om meerdere objecten te kunnen herkennen. In de measure stap maakt deze gebruik van hetzelfde wiskundige model als RANSAC.
De calibratie kan worden uitgevoerd voor eender welk vlak. Zo kan er niet enkel een werktafel worden gebruikt, maar kan het ook worden toegepast voor bijvoorbeeld een interactief schoolbord.

Hoofdstuk 5

Conclusie

Door een proces op te stellen, bestaande uit 6 stappen, zijn we er in geslaagd een basis te ontwikkelen voor een interactieve werkbank. Een papier kan op een robuuste manier worden herkend op de werktafel. Beweging van het papier wordt op een snelle manier gedetecteerd met een tracking algoritme. Dit basisprincipe kan worden uitgebreid tot de herkenning van een groot scala van objecten. Van zodra het object in een wiskundig model kan worden beschreven, is het makkelijk implementeerbaar voor segmentatie in RANSAC en voor computervisie in de condensation filter. De snelheid kan geoptimaliseerd worden door een parallele implementatie op de GPU. Dit alles maakt het mogelijk een zéér interactieve werktafel op te bouwen.

Hoofdstuk 6

Toekomstig werk

Binnen dit domein kan er nog heel wat onderzoek gebeuren. Als we enkel kijken naar dit specifiek onderwerp zijn er nog veel uitbreidingen mogelijk. Ten eerste zou het meerdere papieren tegelijk kunnen herkennen. De grootste uitdaging hierbij is te weten welke lijnen bij welk object horen. Ook moet de objectherkenning veel algemener. Naast de herkenning van papieren zou er een groot aantal verschillende objecten herkenbaar moeten zijn. Hiervoor bestaan er al enkele populaire libraries. Een voorbeeld is de AR Toolkit[Lab09]. Verder kan de nauwkeurigheid van detectie veel fijner. Zo zou bijvoorbeeld, als er tekst op het papier staat (of andere pollutie), het papier nog moeten worden herkend. Een belangrijk voorbeeld hiervan is de pollutie die de handen veroorzaken. Als in de huidige implementatie het de handen van de gebruiker in beeld komen, gaat ook het detail van deze handen herkend worden als lijnen. Een oplossing hiervoor is het wegfilteren van handen en armen. Dit kan worden verwezenlijkt met skin detection algoritmes [H.04].

Een tweede grote uitdaging is dezelfde techniek mogelijk te maken in 3D. Voorlopig zijn we beperkt door het vlak van de bureau zelf. De werktafel kan meer functionaliteit bieden als ook objecten die door de gebruiker worden opgepakt kunnen gemanipuleerd worden. Dit zal een hele andere aanpak nodig hebben dan degene in dit werkstuk. Ten eerste zal er al meer als 1 camera gebruikt worden voor calibratie. Er zijn twee camera's nodig om te kunnen calibreren in 3D. Een extra probleem bij het werken in 3D is de focus. De projector zal maar voor een beperkte diepte een scherp beeld projecteren. Als de diepte van het papier is berekend, zal er dus een scherptestelling moeten gebeuren van het geprojecteerde beeld. Als we te maken hebben met manipulatie van objecten in 3D, zal ook de snelheid van het proces nog aanzienlijk moeten verhogen. Doordat de objecten die in 3D worden herkend dikwijls zullen worden opgepakt door de gebruiker, gaan ze niet langer statisch op de tafel liggen, maar zullen ze aan een hogere frame rate moeten worden aangepast. Indien het niet versneld wordt zal het object al weg zijn alvorens de projectie wordt geprojecteerd.

Verdere ontwikkeling van dit onderwerp maakt het mogelijk een interactieve werktafel op te bouwen. Het voordeel hiervan is dat de gebruiker op een zéér intuïtieve manier kan interageren met zijn computer. De computer zal niet meer een onderdeel van de bureau zijn, maar zijn

bureau zal letterlijk de computer worden. Hij kan het als multifunctionele desktop gebruiken. Zo kan er bijvoorbeeld een interactie worden voorzien met een pen. Op die manier kan de gebruiker digitaal op het papier schrijven. Een menu kan voorzien worden om extra opties te voorzien. Denk maar aan het kiezen van een penkleur of het tekenen van voorgedefinieerde objecten of eventueel een printknop die ervoor zorgt dat uw papier wordt afgeprint. Andere functionaliteit die kan voorzien worden is het doorbladeren van een boek of fotoalbum.

Bibliografie

[Azu97a] Ronald T. Azuma. *A Survey of Augmented Reality*, chapter 2 Applications. Hughes Research Laboratories, Malibu, 1997. http://www.cs.unc.edu/ãzuma/ARpresence.pdf.

[Azu97b] Ronald T. Azuma. *A Survey of Augmented Reality*, chapter 1 Introduction, 2 Background and Related Work. Hughes Research Laboratories, Malibu, 1997. http://www.cs.unc.edu/ãzuma/ARpresence.pdf.

[Cuy07] Tom Cuypers. A-multi-camera system for interactive videogames. Master's thesis, Universiteit Hasselt, May 2007.

[FB81] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. comm. assoc. comp. mach, 1981.

[GW08a] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, chapter 5 Image Restoration and Reconstruction. Pearson International Edition, 3th edition, 2008.

[GW08b] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, pages 144–168. Pearson International Edition, 3th edition, 2008.

[GW08c] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, pages 692–700. Pearson International Edition, 3th edition, 2008.

[GW08d] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, pages 733–738. Pearson International Edition, 3th edition, 2008.

[H.04] Zheng H. Blocking adult images based on statistical skin detection. Technical report, MIIRE Group, November 2004.

[Har03a] Richard Hartley. *Multiple View Geometry in computer vision*, pages 189–193. Cambridge University Press, Canberra, Australia, 2003. Radial Distortion.

[Har03b] Richard Hartley. *Multiple View Geometry in computer vision*, pages 32–37. Cambridge University Press, Canberra, Australia, 2003. Projective Distortion.

[Har03c] Richard Hartley. *Multiple View Geometry in computer vision*, pages 25–27. Cambridge University Press, Canberra, Australia, 2003. Homogene Representatie.

[IB98] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[JSB09] Jordi Pagès Joaquim Salvi and Joan Batlle. *Pattern codification strategies in structured light systems.* Spain, 24-01-2009.

[Kle39] F. Klein. Elementary mathematics from an advanced standpoint, 1939.

[Lab06] Graphics & Media Lab. Gml c++ calibration toolbox. World Wide Web, 2006. http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html.

[Lab09] Human Interface Technology Laboratory. Ar toolkit. World Wide Web, 2009. http://www.hitl.washington.edu/artoolkit/.

[Lee04a] Johnny C. Lee. *Automatic Projector Calibration with Embedded Light Sensors.* PhD thesis, Human-Computer Interaction Institute Carnegie Mellon University, Pittsburgh, USA, 2004.

[Lee04b] Johnny C. Lee. *Automatic Projector Calibration with Embedded Light Sensors.* PhD thesis, Human-Computer Interaction Institute Carnegie Mellon University, Pittsburgh, USA, 2004. Graycode patterns.

[NVi09a] NVidea. Cg. World Wide Web, 2009. http://developer.nvidia.com/page/cg_main.html.

[NVi09b] NVidea. Cuda. World Wide Web, 2009. http://www.nvidia.com/object/cuda_home.html.

[Plo05] Harry Plotter. Camera correctie. World Wide Web, 2005. http://www.vf.utwente.nl/õntw0404.

[PLP03] T. Zhang P. Li and A. Pece. Visual contour tracking based on particle filters. 21(1):111–123, May 2003.

[Tan09a] Yuan-Liang Tang. Gray code. World Wide Web, 2009. http://mathworld.wolfram.com/GrayCode.html.

[Tan09b] Yuan-Liang Tang. Hough implementatie. World Wide Web, 2009. http://www.cyut.edu.tw/ỹltang/.

[Tro09] Trolltech. Qt. World Wide Web, 2009. http://doc.trolltech.com/.

[Wag07] Daniel Wagner. *Handheld Augmented Reality*, chapter 1 Introduction 2 Background and Related Work. University of Technology, 2007. http://studierstube.icg.tu-graz.ac.at/thesis/Wagner_PhDthesis_final.pdf.

[Wik97] Wikipedia. Toegevoegde realiteit. World Wide Web, 1997. http://nl.wikipedia.org/wiki/Toegevoegde_realiteit#Voorbeelden.

[Wik09] Wikipedia. Beam splitter. World Wide Web, 2009. http://en.wikipedia.org/wiki/Beam_splitter.