Beeldinterpolatie van voetbalscènes

Auteur:
Gert Geebelen

Promotor:
Prof. dr. Philippe Bekaert

Begeleiders:
Maarten Dumont & Patrik Goorts

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

Academiejaar 2012-2013

Inhoudsopgave

Samenvatting

Deze thesis gaat op zoek naar een geschikte beeldinterpolatietechniek die toelaat om een voetbalscène vanuit een virtueel camerastandpunt naar keuze te kunnen bekijken. Meer bepaald onderzoekt deze thesis de geschiktheid van de visual hull reconstructietechniek en hoe deze techniek kan worden uitgebreid voor een betere benadering van objecten in grote scènes waarin het aantal camera's beperkt is en camera's vaak over een ruime afstand van elkaar verwijderd zijn.

***Sleutelwoorden**: Beeldinterpolatie, beeldsynthese, visual hull, grote sportscènes*

Dankwoord

Graag zou ik mijn begeleiders Maarten Dumont en Patrik Goorts willen bedanken voor hun advies, het opvolgen en het nakijken van deze thesis. Ook mijn promotor Philippe Bekaert zou ik willen bedanken voor zijn enthousiasme waarmee hij onderzoek uitoefent. Dit is onrechtstreeks steeds een grote motivatie geweest om mijn uiterste best te doen. In het bijzonder zou ik ook mijn ouders, broer en vrienden willen bedanken voor hun morele steun.

Hoofdstuk 1

Inleiding

De afgelopen decennia is het in beeld brengen van sportevenementen sterk geëvolueerd door onder andere gebruik te maken van betere camerahardware en het aantal camera's te verhogen. In praktijk is het echter een dure onderneming om alle mogelijke camerastandpunten te ondersteunen. Bij kortstondige, actievolle spelsituaties geven beschikbare camerabeelden soms onvoldoende informatie. Door zelf een geschikt camerastandpunt te kiezen, zouden we betwistbare situaties beter kunnen analyseren. Beeldinterpolatietechnieken laten toe om voor een willekeurig gekozen nieuw camerastandpunt, het overeenkomstige beeld te reconstrueren uit de opgenomen fysieke camerabeelden. De hoeveelheid beschikbare informatie uit de fysieke camerabeelden die we gebruiken om nieuwe standpunten te berekenen, is zeer beperkt in vergelijking met de hoeveelheid informatie die zich binnen het volume van het voetbalstadium bevindt. Het doel van deze thesis is om te onderzoeken welke interpolatietechnieken goed met deze beperkingen kunnen omgaan en hoe we deze technieken kunnen verbeteren, of kunnen combineren zodat ze gezamenlijk een beter resultaat geven. Deze thesis richt zich in het bijzonder op de visual hull als beeldinterpolatietechniek voor grote sportscènes.
In hoofdstuk 2 wordt het begrip interpolatie gedefinieerd en worden enkele wiskundige interpolatiemethoden besproken. Hoofdstuk 3 bespreekt de mogelijke toepassingen van wiskundige interpolatie op beelden en gaat dieper in op verschillende beeldinterpolatietechnieken. Verder in hoofdstuk 3 wordt de geschiktheid van de visual hull 3D-reconstructietechniek voor grote sportscènes gemotiveerd. Hoofdstuk 4 gaat vervolgens uitgebreid in op de visual hull techniek en mogelijke uitbreidingen om de reconstructiekwaliteit te verbeteren.

Hoofdstuk 2

Interpolatie

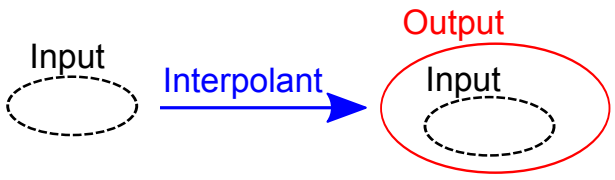
Dit hoofdstuk gaat dieper in op het begrip interpolatie en geeft een overzicht van een aantal interpolatietechnieken die stapsgewijs worden verfijnd tot nieuwe interpolatietechnieken die onder bepaalde condities zoals smoothness, errorgevoeligheid betere resultaten geven en naar het einde toe ook toelaten zelf de interpolatie te controleren door extra constraints te specifiëren.

2.1 Definitie

Interpolatie is het proces waarbij een wiskundige functie, ook wel interpolant genoemd, een verloop heeft door een gespecifieerde inputverzameling van discrete waarden. De interpolant genereert dus een outputverzameling van functiewaarden waarin de discrete inputverzameling bevat zit, zoals afgebeeld in figuur 2.1. De keuze van interpolant bepaalt het tussenliggend pad tussen de inputwaarden.

2.2 Interpolatiecriteria

Welke interpolant de meest geschikte keuze is, hangt af van het verloop van de interpolatiecurve die de gebruiker voor ogen heeft. Zo kunnen er bepaalde



Figuur 2.1: Interpolatieproces.

eisen op de interpolatie worden gedefinieerd zoals:

- smoothness
- correctheid
- controleerbaarheid

2.2.1 Smoothness

Smoothness van een geparametriseerde curve kan naar een smoothe evaluatie van de parameter als naar de smoothe geometrische vorm van de curve verwijzen.

Parametrische smoothness

Parametrische smoothness is een maat voor parametrische continuïteit C. Parametrische continuïteit is de hoogste graad n van differentieerbaarheid waarin de afgeleide functie een continu verloop heeft [Barsky and DeRose [4]], waarbij alle afgeleiden van lagere graden ook continu zijn. In het geval van C_1-continuïteit moeten de raaklijnen in het overgangspunt van de verschillende curvesegmenten, dezelfde grootte en richting hebben. Dit komt overeen met te eisen dat de eerste afgeleide een continu verloop heeft. Des te hoger n, des te smoother de curve zal zijn. Dit geldt zowel voor het curvetraject (=geometrische smoothness), als voor de manier waarop de curve parametrisch geëvalueerd wordt.

Voorbeeld Wanneer we een animerend karakter met een constante snelheid over een curve willen laten bewegen is minimum C_1 continuïteit nodig. De snelheid $v(t) = \frac{d}{dt}(x(t))$, de afgeleide van de positie $x(t)$ naar de tijd, moet continu zijn. Als nog meer parametrische smoothness in evaluatie gewenst is, kan er bijkomende continuïteit in versnelling a opgelegd worden, waarbij $x(t)$ C_2 continu moet zijn met $a(t) = \frac{d}{dt}(\frac{d}{dt}(x(t))$ een continue functie. De parametrische smoothness van een curve die samengesteld is uit het aan elkaar plakken van continue curvesegmenten wordt dus bepaald door de continuïteit in de overgangspunten zoals weergegeven in figuur 2.2.

Geometrische smoothness

Geometrische smoothness is een minder strenge vorm dan parametrische smoothness die enkel smoothness definieert op basis van het curvetraject en

niet op de parametrische evaluatie over dit traject. Parametrische smoothness impliceert dus steeds geometrische smoothness. Geometrische smoothness werd geïntroduceerd omdat parametrisatie van curves geen unieke representatie is van een curve, hetgeen vergelijking 2.1 uitdrukt. Ieder veelvoud $r \neq 0$ van een parametrische representatie $p(t)$ met als domein het oorspronkelijke domein vermenigvuldigt met de inverse van dit veelvoud, $\frac{1}{r}$ vormt een andere parametrische representatie die dezelfde curve representeert.

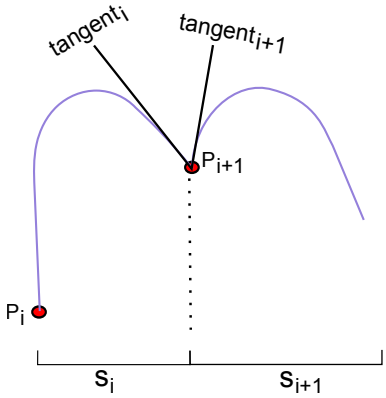
$$p(t), t \in [a, b] \qquad = \qquad p(u) = r * p(t), u \in [\frac{a}{r}, \frac{b}{r}] \tag{2.1}$$

Verschillend geparametriseerde curvesegmenten die geometrisch dezelfde curve voorstellen, zijn ten opzichte van elkaar niet parametrisch continu [Barsky and DeRose [4]]. Er is sprake van geometrische continuïteit als de afgeleide curves er geometrisch hetzelfde uitzien; ze mogen proportioneel van elkaar verschillen en dus ook parametrisch discontinu zijn. De vorm van een curve die enkel geometrisch continu is, is gelijkaardig aan de vorm van een curve die met parametrische continuïteitsconstraints gegenereerd is. In het geometrische geval zal de curve in de overgangspunten proportioneel meer naar de raakvector getrokken worden die het grootst is. De hoogste graad van de afgeleide functie die geometrische continuïteit bezit, bepaalt de geometrische smoothness.

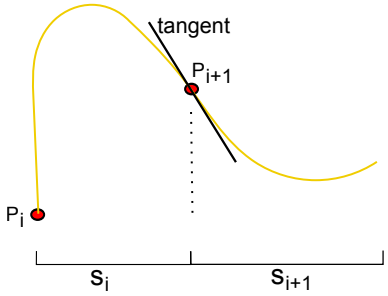
Voorbeeld In het voorbeeld aangehaald bij parametrische smoothness betekent dit dat voor de overgangspunten op de verschillende curvesegmenten de snelheidsvectoren in de overgangspunten tussen verschillende segmenten in dezelfde richting moeten wijzen, maar proportioneel van elkaar mogen verschillen om G_1 continuïteit te bezitten. Dezelfde redenering gaat op voor de versnellingsvectoren om G_2 continu te zijn.

2.2.2 Correctheid van benadering

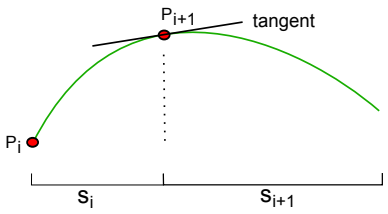
Interpolatie kan gebruikt worden om uit samplewaarden van een bepaalde curve, deze curve te benaderen zodat veel minder data moet worden opgeslagen doordat er geïnterpoleerd wordt. Er zijn oneindig veel interpolatiecurves die door deze samplewaarden gaan en aan de interpolatiedefinitie voldoen. De correctheid van de benadering geeft weer hoe dicht de geïnterpoleerde curve de originele curve benadert met de samplewaarden als input. Dit kan worden uitgedrukt aan de hand van de errorfunctie in 2.3, waarvan de afleiding terug te vinden is in [Bultheel [12]].



(a) C0 continuïteit. Curvesegmenten snijden in het gemeenschappelijk punt.



(b) C1 continuïteit. Curvesegmenten hebben dezelfde tangentiaal in het gemeenschappelijk punt.



(c) C2 continuïteit. Curvesegmenten hebben dezelfde kromming in het gemeenschappelijk punt.

Figuur 2.2: Illustratie van smoothness in de overgang tussen twee curvesegmenten S_i en S_{i+1} met als overgangspunt P_{i+1}.

$$\varepsilon_n(t) = f(t) - b_n(t) \tag{2.2}$$

$$\varepsilon_n = \frac{f^{(n+1)}(\zeta)}{(n+1)!} \prod_{i=0}^{n} (t - t_i) \qquad t, \zeta \in [a, b] \tag{2.3}$$

$b_n(t)$ stelt de interpolant van graad n voor die de originele functie $f(t)$ benadert, gegeven $n+1$ inputwaarden met errorfunctie $\varepsilon_n(t)$ en gegeven dat $f(t)$ $n+1$ keren afleidbaar moet zijn. ζ representeert een onbekend punt in het interval [a,b] waarvan het verband met t om ζ eventueel te berekenen onbekend is [Bultheel [12]]. De reden waarom we deze praktisch minder bruikbare formule toch vermelden, is omdat er andere factoren die de error beïnvloeden uit afgeleid kunnen worden. Zo hangt de interpolatie-error af van:

- $f(t)$, de functie die we willen benaderen. Als de graad n van f toeneemt, zal de noemer $(n+1)!$ toenemen en de error afnemen. Voorwaarden hiervoor zijn dat de inputwaarden t_0 ... t_n onveranderlijk zijn en de afgeleide functies over het domein $[a, b]$ begrensd blijven, zodat de teller begrensd blijft. f(t) dient voor de interpolatie dus voldoende zachte overgangen te hebben zodat de interpolatie-error beperkt blijft en niet divergeert. De afgeleiden van veeltermfuncties zijn begrensd aangezien de $n+1$ graad als afgeleide 0 zal hebben. Als de overige afgeleide functies van hogere graden in de buurt van n van de functie f die we willen benaderen niet al te groot worden, gaan veeltermfuncties goede resultaten geven omdat de functie die we willen benaderen dan een veeltermachtig karakter heeft. Daarnaast bepaalt het gedrag van f in de buurt van het interpolatie-interval in het complexe vlak of het convergentieproces zal convergeren of niet [Bultheel [12]].

- $\prod_{i=0}^{n}(t - t_i)$: de error kan gereduceerd worden door de interpolatiepunten zodanig te kiezen zodat deze factor geminimaliseerd wordt [Amos [2]]. Hierbij speelt de ligging van de interpolatiepunten in het interval $[a, b]$, de verschillende t_i's alsook de ligging van het interval $[a, b]$ een rol omdat $\varepsilon(t)$ afhangt van t.

Als f en b_n veeltermfuncties zijn van dezelfde graad dan stellen ze dezelfde curve voor; veeltermrepresentatie met maximumgraad n door $n+1$ inputpunten is immers uniek volgens het bewijs in [Bultheel [12]]. Overeenkomstig hiermee zal in de errordefinitie $f^{(n+1)} = 0$.

2.2.3 Controleerbaarheid

De controleerbaarheid van de interpolant is een maat in welke het traject van de interpolant kan bijgestuurd worden door extra constraints te specifëren. Verder kan er onderscheid gemaakt worden tussen globale en lokale interpolatiecontrole [Parent [84]]. Bij lokale controle beïnvloedt de herinstelling van een constraint enkel een curvesegment, terwijl bij globale controle het gehele verloop van de curve hierdoor kan wijzigen.

2.3 Veelterminterpolatie

Veelterminterpolatie genereert uit $n+1$ inputpunten een polynoom die exact door deze punten gaat. In de literatuur zijn verschillende methoden die dezelfde polynoom als resultaat geven [Gautschi [34]] waaronder:

- Lagrange veelterminterpolatie
- Newton veelterminterpolatie

2.3.1 Lagrange veelterminterpolatie

Lagrange veelterminterpolatie volgens de parametrische vergelijking 2.4, genereert uit n+1 inputpunten, een polynoom die exact door deze punten gaat. Het resultaat ervan is afgebeeld in 2.6(c).

$$f(t) = \sum_{i=0}^{n} f_i(t) P_i \tag{2.4}$$

$$f_i(t) = \prod_{i \neq j} \frac{t - t_i}{t_i - t_j} \tag{2.5}$$

Vergelijking 2.4 drukt een veeltermfunctie uit in Lagrange notatie, die via inductie geconstrueerd kan worden [Palais and Palais [83]]. De polynoom $f(t)$ is een lineaire combinatie van basispolynomen f_i. P_i stelt een inputpunt voor en is t_i de parameterwaarde per inputpunt met t $\in [t_0, tn]$. f_i stelt een veeltermfunctie voor met maximum graad n. $f(t)$, de polynoominterpolant, is uniek [Bultheel [12]] en is een lineaire combinatie van de basis veeltermfuncties $f_i(t)$ uit vergelijking 2.5 met maximum graad n. Iedere f_i kan gereconstrueerd worden door te eisen dat voor punt P_i, $f_i(t_i) = 1$ en dat in alle andere t-waarden t_j in de overige inputpunten met $i \neq j$, de basispolynoom $f_i(t_j)$ tot 0 evalueert zodat $f(t_i) = P_i$. Deze nulpunten definiëren

een polynoom $f_i(t) = \prod_{i \neq j}(t - t_i)$. Om ervoor te zorgen dat $f_i(t_i) = 1$ schaleren we met $\prod_{i \neq j}(t_i - t_j)$. Door te stellen dat $f_i(t_j) = 0$, zal er geen enkele andere basispolynoom f_j een verandering teweegbrengen in de lineaire combinatie voor de f_i-term voor inputpunt P_i. f is dan de lineaire combinatie van polynoom die door alle inputpunten gaat.

2.3.2 Newton veelterminterpolatie

De veelterminterpolatiemethode van Newton laat op een recursieve wijze toe om uit een veelterminterpolant van graad n de veelterminterpolant van graad $n + 1$ te genereren. Deze methode is dus sneller dan de Lagrange interpolatie als incrementeel inputpunten aan de inputverzameling voor interpolatie zouden toegevoegd worden. De Lagrange veelterminterpolatiemethode zou helemaal opnieuw de berekening moeten starten terwijl Newton veelterminterpolatie recursief kan verdergaan op de reeds gemaakte berekeningen [Yang et al. [110]]. Newton veelterminterpolatie is in parametrische formulevorm uitgedrukt in 2.6:

$$f(t) = D^0 + \sum_{i=1}^{n} D^i \prod_{j=0}^{i-1}(t - t_j) \tag{2.6}$$

met

$$D^m = \begin{cases} \Delta^0(P_{[m,m]}) & m = 0 \\ \Delta^{m+1}(P_{[0,\ldots,m]}) & m > 0 \end{cases} \tag{2.7}$$

$$\begin{cases} \Delta^0(P_{[i,i]}) = P_i \\ \Delta^k(P_{[i,\ldots,k]}) = \frac{\Delta^{k-1}(P_{[i+1,\ldots,k]}) - \Delta^{k-1}(P_{[i,\ldots,k-1]})}{(t_k - t_i)} \end{cases} \tag{2.8}$$

n duidt op een interpolant van graad n gegeven $n + 1$ inputpunten. D^m wordt de m-de gedeelde differentie genoemd [Baker [3]] die recursief wordt opgebouwd uit de gedeelde differenties van de vorige iteratie via de recursieve Δ-functie. De Δ_k beschrijft de Δ-functie met recursiediepte k. Subscript [i,...,k] stelt een geordende lijst van elementen voor met ondergrens i en bovengrens k, waarbij de index i telkens met 1 verhoogd wordt tot k. $P_{[i,\ldots,k]}$ stelt dus een punt voor uit iteratie k en is het i-de inputpunt als $k = 0$.

Voorbeeld Als $n = 2$, dan zijn er dus 3 inputpunten gegeven en heeft de 2-de graads Newton-interpolant de volgende vorm:

$$f(t) = P_0 + D_1(t - t_0) + D_2(t - t_0)(t - t_1) + D_3(t - t_0)(t - t_1)(t - t_2)$$

$$f(t) = P_0 + P_{01}(t - t_0) + P_{012}(t - t_0)(t - t_1) + P_{0123}(t - t_0)(t - t_1)(t - t_2)$$

$$P_{01} = \frac{P_1 - P_0}{t_1 - t_0} \quad P_{12} = \frac{P_2 - P_1}{t_2 - t_1} \quad P_{23} = \frac{P_3 - P_2}{t_3 - t_2}$$

$$P_{012} = \frac{P_{12} - P_{01}}{t_2 - t_0} \qquad P_{123} = \frac{P_{23} - P_{12}}{t_3 - t_1}$$

$$P_{0123} = \frac{P_{123} - P_{012}}{t_3 - t_0}$$

Chebyshev veelterminterpolatie De veelterminterpolatiemethode van Chebyshev gebruikt een betere keuze van inputpunten wanneer de interpolatie gebruikt wordt om een functie te benaderen. Zo wordt de interpolatie-error uit 2.3 gereduceerd door de inputdistributie via een Chebyshev heuristiek te bepalen die $\prod_{i=0}^{n}(t - t_i)$ minimaliseert om vervolgens veelterminterpolatie volgens Lagrange of Newton toe te passen.

2.3.3 Evaluatie

Des te hoger de polynoomgraad des te smoother de veeltermfunctie tussen de verschillende inputpunten zal interpoleren. Toch zijn hier ook enkele nadelen mee verbonden zoals:

- Berekeningskost: wanneer het aantal inputpunten toeneemt kan de polynoomgraad sterk toenemen om aan alle constraints te voldoen, waarvoor veel rekenkracht nodig is om hierdoor een polynoom te berekenen.
- Wiggle-effecten: het op en neergaan over de denkbeeldige lijn tussen de verschillende interpolatiepunten. Dit kan tijdens het interpoleren zeer ongewenst kan zijn. Stel dat we bijvoorbeeld de inputwaarden van een 2 dimensionale computermuis willen interpoleren waarbij de computermuis in een rechte lijn over een vlak beweegt. Dit zou bij een polynoominterpolatie van een hoge graad resulteren in niet-rechtlijnige bewegingen van de muiscursor op het scherm door het wiggle-effect. Wiggle effecten zijn geïllustreed in figuur 2.3(a).
- Runge's fenomeen: is een divergentieverschijnsel dat optreedt bij benadering van curve met behulp van veelterminterpolatie waarbij de

interpolatie-error toch toeneemt ondanks een toename van polynoomgraad gegeven inputpunten die zich op dezelfde afstand van elkaar bevinden. De interpolant zal hierbij almaar grotere wiggle-effecten vertonen naar de randen van het interval toe [Yang et al. [110]] zoals te zien is in 2.4. Voor veelterminterpolatie wordt de interpolatie-error begrensd door de formule uit 2.9.

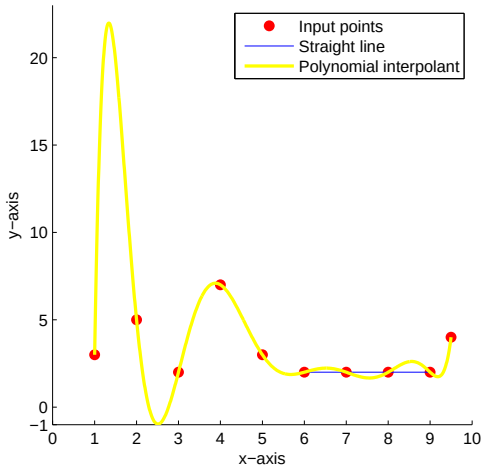
- Globale controle: wanneer er een inputpunt wijzigt heeft dit een invloed op het verloop van de hele curve. We kunnen het verloop niet lokaal controleren.

$$\epsilon(t) \leq max_{t\in[a,b]}|\frac{(t-t_0)(t-t_1)...(t-t_n)}{(n+1)!}|max_{t\in[a,b]}|f^{(n+1)}(t)| \quad (2.9)$$

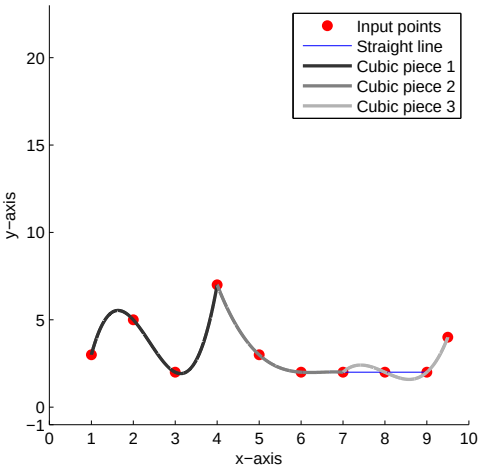
met $n+1$ inputpunten voor een te benaderen polynoom f van graad n over interval $[a,b]$. De error zal dus afnemen als de noemer sneller naar oneindig gaat dan de teller wanneer we n laten toenemen. Runge toonde aan dat dit niet altijd het geval is: de karakteristiek van de te benaderen functie heeft ook een invloed op de interpolatie-error zoals eerder aangehaald in 2.3. Dit verschijnsel kan gecompenseerd worden door de norm van de polynoomfunctie gedefinieerd door $(t-t_0)(t-t_1)...(t-t_n)$ te controleren door de inputpunten volgens een Chebyshev distributie te bemonsteren, of interpolatiemethoden zoals stukjesgewijze veelterminterpolatie, of spline interpolatie, die in de volgende secties beschreven worden, te gebruiken [Amos [2], Knezevic [44]].

2.4 Stukjesgewijze veelterminterpolatie

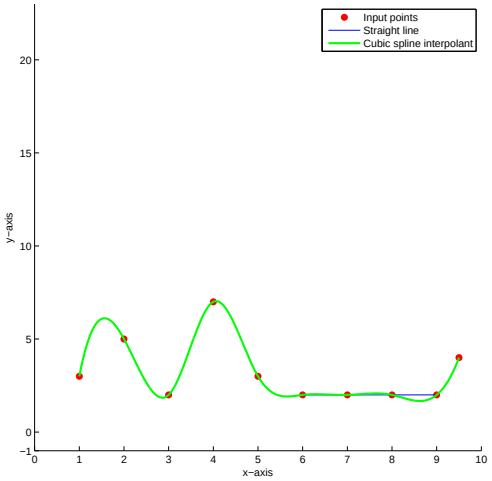
In plaats van een polynoom met voldoende hoge graad die door alle inputpunten gaat zoals bij veelterminterpolatie, zouden ook meerdere polynomen gebruikt kunnen worden die stukjes van de inputdata met elkaar te verbinden. Dit reduceert de complexiteit voor het genereren van een interpolant, waardoor we polynoom interpolanten van lagere orden kunnen gebruiken, waarbij de wiggle-effecten beperkt blijven [Yang et al. [110]].
Het aantal onafhankelijke punten nodig om een polynoom van graad n uit formule 2.4 door een gegeven aantal inputpunten te construeren is gelijk aan $n+1$. Als we dus een lineaire, eerstegraadspolynoom voor interpolatie zouden



(a) Veelterm interpolatie.

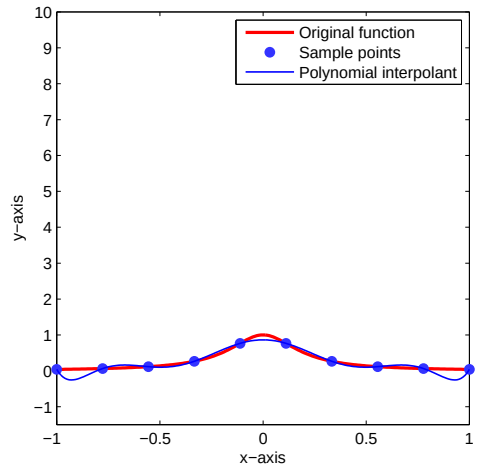


(b) Stukjesgewijze kubische interpolatie.

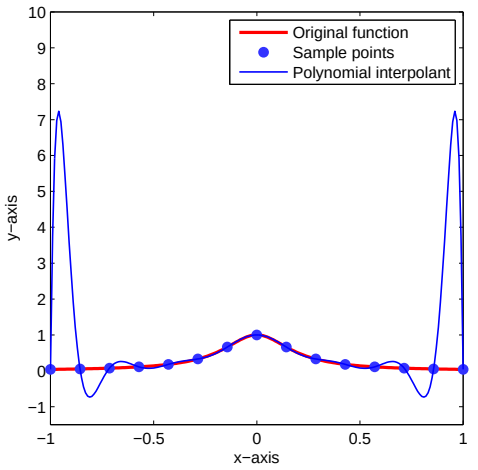


(c) Spline interpolatie.

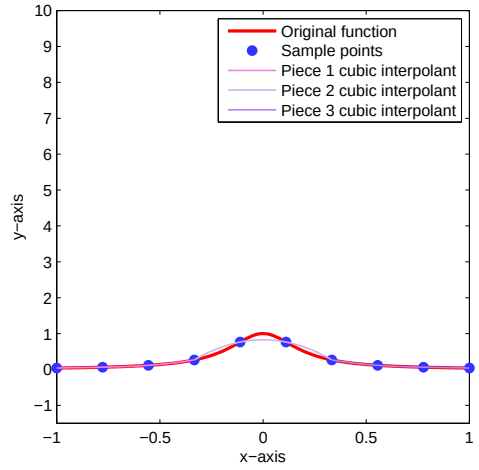
Figuur 2.3: Wiggle effect.



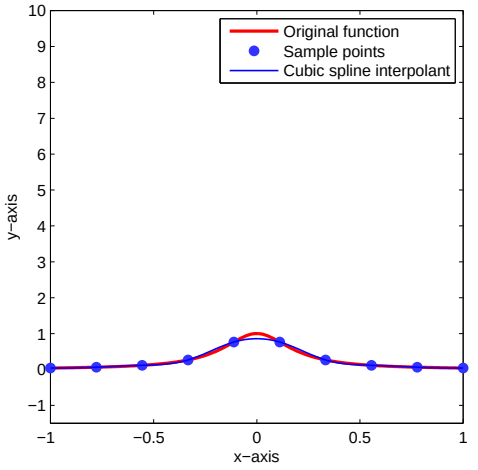
(a) Veelterminterpolatie van graad 9 gegeven 10 sample points.



(b) Veelterminterpolatie van graad 14 gegeven 15 sample points.



(c) Stukjesgewijze kubische veelterminterpolatie gegeven 10 sample points.



(d) Spline interpolatie gegeven 10 sample points.

Figuur 2.4: Runge's fenomeen geïllustreerd door de polynoomgraad van 9 tot 14 te laten toenemen op benadering van functie f met gegeven $f(t) = \frac{1}{1+25t^2}$ en een equidistante bemonstering over het interval $[-1, 1]$. Merk de groei van de interpolatie-error op in de buurt van de randen van het interval wanneer de polynoomgraad toeneemt.

gebruiken, hebben we 2 onafhankelijke inputpunten nodig, bij een kwadratische polynoom 3 etc.
In figuren 2.6(d) en 2.6(b) zijn voorbeelden van stukjesgewijze veelterm interpolanten afgebeeld.

2.4.1 Evaluatie

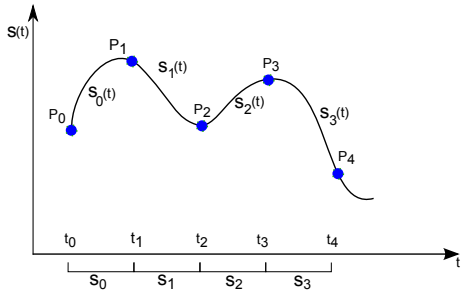
Deze methode heeft minder last van wiggle-effecten en Runge's fenomeen zoals weergegeven in afbeeldingen 2.3(b) en 2.4(c). Daarnaast zijn de stelsels minder complex en is er wel lokale controle mogelijk doordat met stukjes polynomen gewerkt wordt. Toch zijn aan deze methode nog enkele serieuze nadelen verbonden:

- Smoothness in de overgangspunten: de onzachte overgangen ontstaan doordat de afgeleiden per functie in de overgangspunten die op de concatenatie van de verschillende functies liggen, sterk kunnen verschillen.
- Het aantal bemonsteringen nodig voor constructie van de lagere orde polynomen. Hoewel dit aantal bemonsteringen sterk gereduceerd is ten op zichte van veelterminterpolatie toch hebben lagere ordere polynomen net iets te veel samplepoints nodig; onder meer in tijdkritische applicaties zoals in genetwerkte virtuele omgevingen is het noodzakelijk dat er tussen 2 inputpunten smooth kan geïnterpoleerd worden zonder hiervoor te moeten wachten tot er voldoende updatepaketten p gearriveerd zijn om de desgewenste polynoom van graad $p-1$ te construeren. Dit zou immers een ongewenste delay introduceren. Een interpolatietechniek die toelaat smooth in elkaar overgaande stukjesgewijze polynomen uit slechts 2 inputpunten te construeren, is spline interpolatie.

2.5 Spline interpolatie

Een spline curve is een smooth interpolatiecurve die een verzameling is van stukjesgewijze veeltermfuncties. Een curve S noemen we een spline van graad n als de volgende voorwaarden voldaan zijn:

- het domein van S is een gesloten interval: $[a, b]$, S is dus een eindige functie.
- S heeft C^{n-1} continuïteit: alle afgeleiden S' tot en met S^{n-1} zijn gedefinieerd.



Figuur 2.5: Spline interpolatie.

- S gaat door inputpunten P_i die strikt geordend kunnen worden over het domein $[a, b]$: $a = P_0 < P_1 < \ldots < P_n = b$ zodang dat iedere S_i een polynoom is met graad hoogstens n over iedere $[P_i, P_{i+1}]$. De inputpunten worden in spline context ook wel knopen genoemd[Keele [43]].

$$S = \begin{cases} S_0(t), & P_0 \leq t \leq P_1 \\ S_1(t), & P_1 \leq t \leq P_2 \\ \vdots & \\ S_n(t), & P_{n-1} \leq t \leq P_n \end{cases} \tag{2.10}$$

Merk op dat stukjesgewijze kubische interpolatie afgebeeld in figuur 2.6(d) niet aan bovenstaande definitie voldoet vermits de laatste twee voorwaarden er niet gelden: de functie heeft slechts positionele C_0 continuïeit en er is niet in elk opeenvolgend interval $[P_i, P_{i+1}]$ een polynoom gedefinieerd, maar over een groter interval $[P_i, P_{i+3}]$. De lineaire interpolant in 2.6(b) voldoet wel aan de spline definitie en wordt ook wel lineaire spline interpolant genoemd.

2.5.1 Lineaire spline interpolatie

Lineaire spline interpolatie beschrijft een spline die bestaat uit een concatenatie van rechten door de inputpunten. Volgens de spline definitie moet C_0 continuïteit gelden. Een rechte heeft als parametrische vergelijking:

$$f(t) = c_0 + c_1 t \tag{2.11}$$

, waarbij c_i de i-de coëfficient voorstelt en $t \in [0, 1]$. De vergelijking van een rechte tussen twee punten P_0 op $t = 0$ en P_1 op $t = 1$ is.

$$\begin{pmatrix} P_0 \\ P_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} \tag{2.12}$$

Hiervan zijn de coëfficienten de onbekenden, die we kunnen berekenen door het stelsel op te lossen door langs links met de inverse van de matrix die parametrische afbeeldingen van de inputpunten P_0 en P_1 bevat, te vermenigvuldigen:

$$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} P_0 \\ P_1 \end{pmatrix} \tag{2.13}$$

$$\underbrace{\begin{pmatrix} c_0 \\ c_1 \end{pmatrix}}_{M_{coef}} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \end{pmatrix} \tag{2.14}$$

Wanneer we de parametrische vergelijking uit 2.11 in matrixvorm herschrijven krijgen we vergelijking 2.15.

$$f(t) = \begin{pmatrix} 1 & t \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} \tag{2.15}$$

$$f(t) = \underbrace{\begin{pmatrix} 1 & t \end{pmatrix}}_{M_{param}} \underbrace{\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}}_{M_{spline}} \underbrace{\begin{pmatrix} P_0 \\ P_1 \end{pmatrix}}_{M_{constraints}} \tag{2.16}$$

M_{spline} is de basismatrix die de spline-interpolant karakteriseert. Ze drukt de transformatie van de constraintmatrix $M_{constraints}$ naar de polynoomcoëfficienten M_{coef} uit. De blendingmatrix $M_{blending}$ kunnen we berekenen door M_{param} met $M_{blending}$ te vermenigvuldigen. Iedere blending functie b_i, per kolom i in $M_{blending}$ bepaalt het gewicht dat aan de overeenkomstige constraint P_i wordt toegekend tijdens de evolutie van parameter t in het interpolatieproces.

$$f(t) = \underbrace{\begin{pmatrix} 1-t & t \end{pmatrix}}_{M_{blending}} \underbrace{\begin{pmatrix} P_0 \\ P_1 \end{pmatrix}}_{M_{constraints}} \tag{2.17}$$

Voor een lineaire spline met n inputpunten geldt dan:

$$f(t) = \sum_{i=0}^{n-1} b_i(t) P_i \tag{2.18}$$

$$f(t) = \sum_{i=0}^{n-1} (1-t)P_i + tP_{i+1} \tag{2.19}$$

, met P_i, P_{i+1} inputpunten en $t \in [0, 1]$ voor iedere i.

Wanneer we op ieder opeenvolgend paar punten deze lineaire interpolant toepassen krijgen we het resultaat van figuur 2.6(b).

2.5.2 Kubische spline interpolatie

Een kubische polynoom is de laagste orde polynoom die buigpunten kan voorstellen omdat ze C_2 continu is. Tussen hogere graads spline en kubische spline kan het menselijk oog nauwelijks verschillen onderscheiden [Restrepo and Indik [87]]. Bovendien zijn hogere orde polynomen minder efficiënt en hebben ze meer last van oscillatie, waarbij over korte periodes zeer vaak de bewegingsrichting verandert[Yang et al. [110]]. Vandaar dat kubische polynomen vaak bij spline interpolatie gebruikt worden.
Per kubische interpolant zijn volgens vergelijking 2.4 vier onafhankelijke inputpunten nodig om deze op te stellen. Tussen $n+1$ inputpunten bevinden zich n curvesegmenten. Voor elk curvesegment zijn 4 vergelijkingen nodig om de 4 onbekende c-coëfficiënten in hun kubische vergelijking 2.20 op te lossen. Voor $n+1$ inputpunten geldt dat $f(t_i) = P_i$, met $i = 0, 1...n$. De spline definitie zegt dat een kubische spline C_2 continu moet zijn. Voor ieder tussenliggend inputpunt geldt dat $S_{i-1}^{(k)}(t_i) = S_i^{(k)}(t_i)$ waarbij k de k-de afgeleide voorstelt met $k = 0, 1, 2$. Zo zijn er $n+1-2$ dus $n-1$ tussenliggende punten die $3(n-1)$ bijkomende constraints opleggen en hebben we in totaal $4n-2$ constraints. Deze 2 ontbrekende constraints om het stelsel oplosbaar te maken, kunnen op verschillende manieren worden ingevuld:

- natuurlijke eindconditie: een natuurlijke kubische spline stelt dat de eindpunten buigpunten zijn door de tweede afgeleide gelijk te stellen aan 0: $S_0^{(2)}(t_0) = S_{n-1}^{(2)}(t_n) = 0$.

- parabolische uitloper als eindconditie: stelt dat de eindpunten dezelfde tweede afgeleide hebben als hun dichtsbijliggende inputpunt dat deeluitmaakt van hun curvesegment en geen eindpunt is: $S_0^{(2)}(t_0) = S_0^{(2)}(t_1)$ en $S_{n-1}^{(2)}(t_n) = S_{n-1}^{(2)}(t_{n-1})$.

- kubische uitloper als eindconditie: stelt dat de eerste 2 en de laatste 2 intervallen worden samengenomen en gerepresenteerd door 1 kubische curve door de tweede afgeleide in het beginpunt gelijk te stellen aan $S_0^{(2)}(t_0) = 2S_0^{(2)}(t_1) - S_1^{(2)}(t_2)$ en in het eindpunt gelijk te stellen aan $S_{n-1}^{(2)}(t_n) = 2S_{n-1}^{(2)}(t_{n-1}) - S_{n-2}^{(2)}(t_{n-2})$ [Kruger [48]].

- raaklijnrico's als eindconditie: als extra input aan het te oplossen stelsel worden door de gebruiker de richtingscoëfficiënten r van de raaklijnen in de eindpunten gespecifieerd: $S_0^{(1)}(t_0) = r_{begin}$ en $S_{n-1}^{(1)}(t_n) = r_{einde}$.

$$f(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \tag{2.20}$$

2.5.3 Interpolatie m.b.v. controlepunten

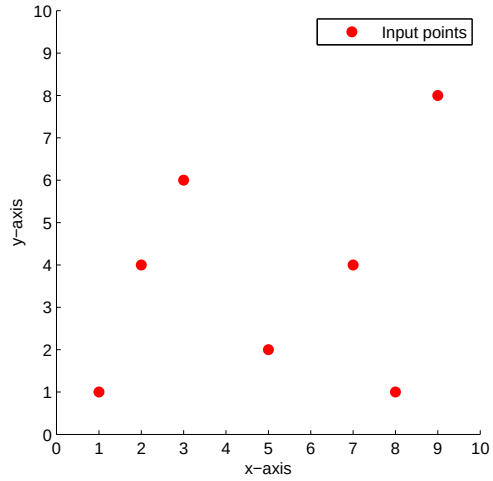
Sommige interpolatiefuncties laten toe om het verloop ervan te controleren, door controlepunten aan de inputverzameling van waarden toe te voegen. Deze controlepunten leggen daardoor extra constraints op het verloop van de interpolant. Een voorbeeld van zo'n interpolant is de beziercurve, afgebeeld in figuur 2.6(f). Deze curve specifieert via de controlepunten de raaklijnen of tangentiale van de inputelementen waartussen geïnterpoleerd wordt.

2.5.4 Evaluatie

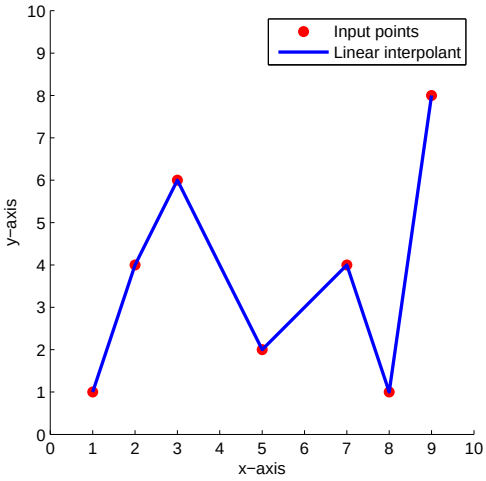
Spline interpolatie laat toe smooth te interpoleren zonder problemen te hebben met Runge's fenomeen zoals afgebeeld in figuur 2.4(d) en weinig last te hebben van wiggle-effecten door gebruik te maken van lagere orden zoals weergegeven in figuur 2.3(c).
Indien spline interpolatie als benaderingsmethode gebruikt wordt, zal de interpolatie-error steeds convergeren doordat de grootte van de error niet afhangt van de hogere orde afgeleiden en dus de karakteristieken van de gegeven functie zoals bij veelterminterpolatie [Amos [2]]. De bovengrens van de lineaire interpolatie-error is $O(h^2)$ en de bovengrens voor kubische spline interpolatie is $O(h^4)$, waarbij $h = \frac{(b-a)}{n}$, de lengte van het subinterval tussen paren van equidistante inputpunten in het interval $[a, b]$ voorstelt, met $n+1$ inputpunten. Dit wil dus zeggen dat als we h halveren dat de bovengrens van de error in het kubische geval met een factor $\frac{1}{2^4}$ verkleint en in het lineaire geval met een factor $\frac{1}{2^2}$ verkleint, waaruit we kunnen concluderen dat kubische spline interpolatie een betere benadering geeft. Spline interpolanten met een nog hogere orde dan 3 geven slechts een zeer kleine verbetering in convergentiesnelheid waardoor ze nauwelijks gebruikt worden[Amos [2]]. Afleidingen van deze errorbovengrenzen zijn terug te vinden in [Amos [2]].
Sommige spline interpolatiemethoden laten ook toe de interpolatie te controleren en lokaal te beïnvloeden. Deze flexibileit maakt spline interpolatie zeer bruikbaar om onder meer zelf wiskundige curves te tekenen in bijvoorbeeld digitale tekenprogramma's en vector graphics.

2.6 Applicatie

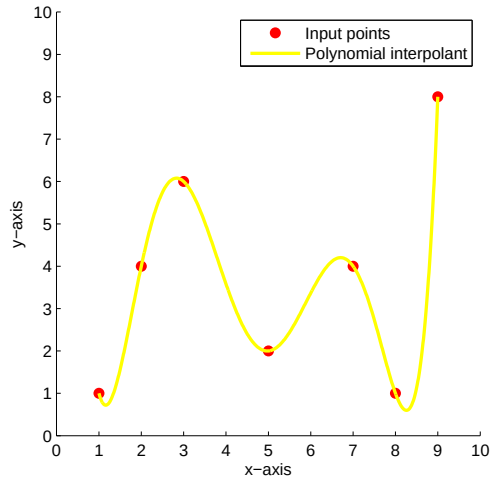
Interpolatie wordt vaak gebruikt om onbekende informatie in te vullen aan de hand van de informatie die gegeven is om zo tussenliggende waarden te genereren die de inputwaarden met elkaar verbinden. De keuze van interpolant hangt af van hoe de overgang moet verlopen in de doelapplicatie: continuïteit, welke orde, welke dimensies we kiezen te interpoleren, smoothness etc. De aangehaalde voorbeelden van interpolanten vormen een kleine deelverzameling van de grote waaier aan mogelijkheden van wiskundige functies om deze overgang te realiseren alsook het grote aantal verschillende fysieke variabelen die we kunnen interpoleren. Bijvoorbeeld positievariabelen interpoleren om een animerend karakter smooth te laten bewegen tussen verschillende gegeven posities, de focuslengte interpoleren om bijvoorbeeld een zoomovergang te creëren, intensiteiten van pixelwaarden in elkaar laten overgaan, tussenliggende camerastandpunten te genereren, camerabeelden vanuit deze tussenliggende camerastandpunten en beelden uit de oorspronkelijke standpunten te vormen, andere wiskundige functies te benaderen door ze te samplen en deze samples als input voor de interpolatiemethode gebruiken etc.



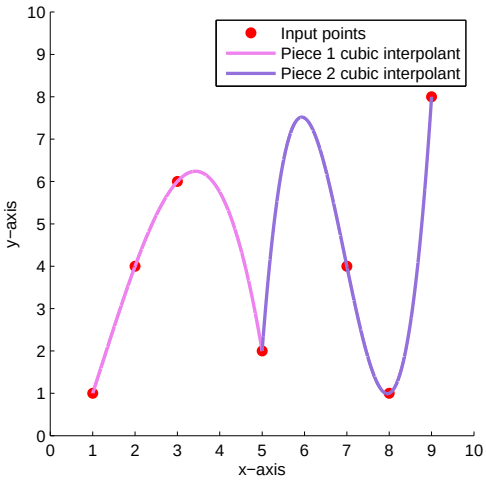
(a) Inputverzameling voor interpolatiefunctie.



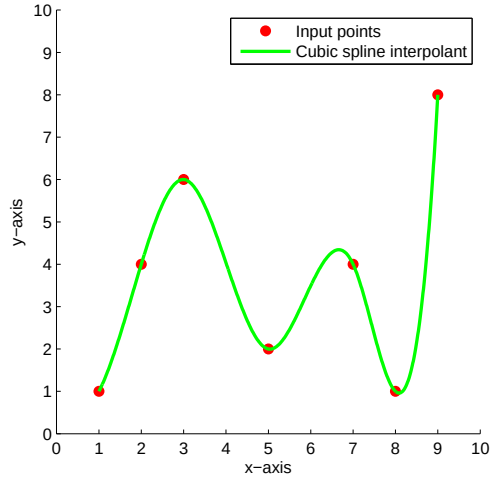
(b) Lineaire spline interpolatie.



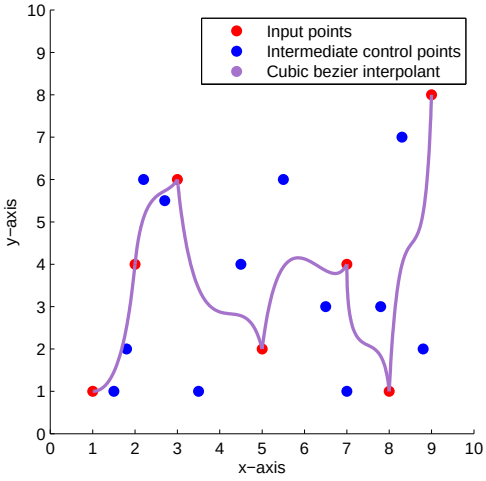
(c) Veelterm interpolatie.



(d) Stukjesgewijze kubische interpolatie.



(e) Kubische spline interpolatie.



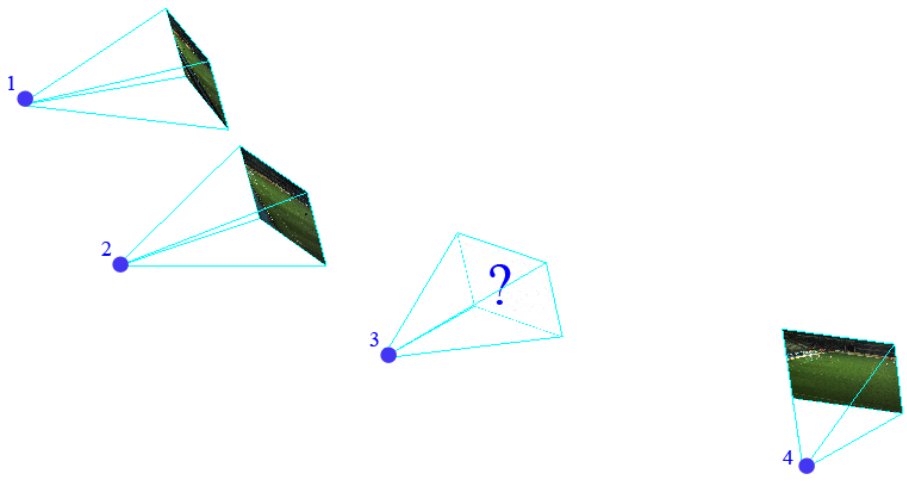
(f) Kubische bezier interpolatie.

Figuur 2.6: Voorbeelden van interpolatiemethoden toegepast op inputdata.

Hoofdstuk 3

Beeldinterpolatie

Beeldinterpolatie gaat uit een verzameling van gegeven inputafbeeldingen een nieuwe afbeelding vanuit een vrijgekozen camerastandpunt of kijkrichting genereren. In afbeelding 3.1 wordt het principe van beeldinterpolatie geïllustreerd. Gegeven zijn afbeeldingen uit camera's 1,2,4. Beeldinterpolatie laat toe om een nieuw tussenliggend beeld te berekenen voor een zelfgekozen virtueel camerastandpunt zoals camera 3 in afbeelding 3.1.



Figuur 3.1: Beeldinterpolatie

Voor het genereren van een nieuw beeld uit gegeven reële beelden kan er een onderscheid gemaakt worden tussen:

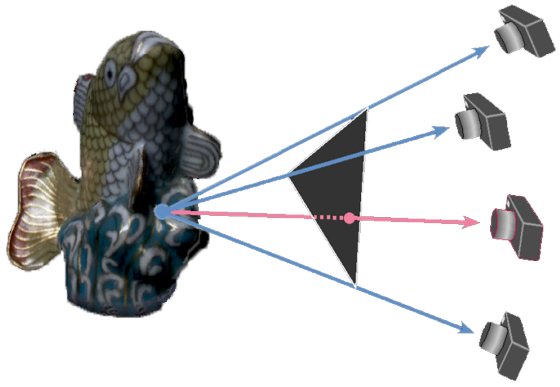
- Beeldgebaseerde rendering

  Bij beeldgebaseerde renderingtechnieken wordt een nieuw beeld rechtstreeks uit reële afbeeldingen gegenereerd. Deze technieken steunen op interpolatie van de inputafbeeldingen of pixel reprojecties van bron - naar doelafbeelding [Kang [41]]. Het voordeel van beeldgebaseerde rendering is dat de performantie niet afhangt van de scene-complexiteit vermits de uitvoertijd enkel afhangt van het aantal pixels.

- Geometriegebaseerde rendering

  Geometriegebaseerde rendering reconstrueert een geometrisch model van de scene uit de inputafbeeldingen dat vervolgens via de grafische kaart gerenderd wordt volgens de eigenschappen van de virtuele camera. Er wordt als het ware een nieuwe foto van de 3D gereconstrueerde scene genomen op de gewenste positie van het nieuwe beeld.

Beide technieken geven als uitvoer een nieuw beeld en kunnen dus ook samen gebruikt worden. Bijvoorbeeld het inkleuren van een gereconstrueerd object via view dependent texture mapping (VDTM) hetgeen afgebeeld is in figuur 3.2.



Figuur 3.2: Illustratie van combinatie van beeld- en geometriegebaseerde rendering: foto's worden via view dependent texture mapping op een geometrisch model geprojecteerd. Gegeven een nieuw camerastandpunt in het roze kijkend naar een bepaald punt. VDTM blendt via gewogen interpolatie de geprojecteerde textures op deze positie met gewichten per cameratexture naargelang de nabijheid van de geprojecteerde rays vanuit deze camerastandpunten tot de nieuwe kijkrichting. Bron afbeelding: [Wood et al. [105]].

3.1 Beeldgebaseerde renderingtechnieken

Beeldgebaseerde renderingtechnieken kunnen geclassificeerd worden naarmate de hoeveelheid geometrische informatie gekend is [Shum and Kang [97]]. We overlopen kort enkele beeldgebaseerde interpolatiemethoden startend van geen geometrie tot almaar stijgend geometriegebruik.

Onder de aanname dat noch geometrische informatie noch kalibratie-informatie beschikbaar is kunnen we geen perspectief correcte interpolant afleiden omdat de calibratieparameters die deze interpolant mee zouden bepalen onbekend zijn. Om uit deze kleine hoeveelheid informatie via interpolatie goede resultaten te bekomen is er veel informatie over de scene die liefst onafhankelijk ten opzichte van de cameraparameters beschreven is. Een mogelijkheid is licht ten opzichte van de scene te parametriseren. De functie die dit beschrijft is de plenoptische (in het Latijn: plenus opticus) functie en is als volgt gedefinieerd [Adelson and Bergen [1]]:

$$P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \longleftrightarrow (\theta, \phi, \lambda, V_x, V_y, V_z, t) \longmapsto L \tag{3.1}$$

De plenoptische functie P beschrijft dus alles wat we van de scene kunnen zien: de radiantie L die op een punt in de scene met positie V_x, V_y, V_z invalt volgens een 3D richting beschreven door de 2 hoeken op een bol (breedtehoek θ en hoogtehoek ϕ) met bepaalde golflengten λ op een bepaald tijdstip t (voor dynamische scenes). Wanneer voldoende lichtinformatie van een scene door bemonstering van de plenoptische functie verzameld wordt, kan via interpolatie niet- bemonsterde lichtinformatie uit de verschillende samples worden ingevuld.

3.1.1 Rendering m.b.v. lichtvelden

In [Levoy and Hanrahan [58]] wordt de plenoptische functie een lichtveld genoemd. Gelijkaardig aan magnetische en elektrische velden die de grootte en richting van de respectievelijk magnetische en elektrische kracht op een bepaalde positie in de ruimte beschrijven, beschrijft een lichtveld de grootte van de radiantie in iedere richting en positie in de ruimte. Het 7-dimensionaal lichtveld kan vereenvoudigd worden naar 4D onder de assumpties dat de lichtkleur (golflengte λ) niet verandert, de scene statisch is (t is constante) en de radiantie constant is. Radiantie is constant in een lege ruimte wanneer licht niet geblokkeerd wordt. Het gevolg van een constante radiantie is dat alle 3D posities op een lijn volgens een 2D richting (θ, ϕ)door een 2D positie en een 2D richting kunnen worden voorgesteld wanneer de camerastandpunten zich buiten een bepaald volume bevinden waar rays niet door het object zelf

geïntersecteerd worden alvorens ze een ander oppervlak op het object raken. Dit geldt wanneer camerastandpunten zich buiten de convex hull van het object bevinden, (zie volgend hoofdstuk).
Een mogelijke parametrisatie (zie [Magnor [67]] voor andere) van een 2D positie en 2D richting is door middel van 2 vlakken, ook wel een light slab genoemd [Levoy and Hanrahan [58]]. Het voordeel van deze parametrisatie is dat lichtvelden rechtstreeks op die manier kunnen worden vastgelegd. Lichtstralen tussen een camera en een object beschrijven dezelfde radiantie en parameters van een lichtveld vermits uit de aannames volgt dat de radiantie constant is langs de lichtstraal. Intensiteitswaarden van pixels die een projectie-afbeelding zijn van een punt in de scene komen dus overeen met de radiantie van een punt uit de scene volgens de lichtstraal die samenvalt met de projectiestraal. Wanneer er voldoende camera's geplaatst zijn rond een object gaat de verzameling lichtstralen die op de oogpunten invallen afkomstig door reflecties van licht op verschillende punten op het object dezelfde verzameling lichtstralen beschrijven als alle lichtstralen die op ieder objectpunt gereflecteerd worden. 4D lichtvelden kunnen dus beschreven worden door enerzijds een vlak bestaande uit array van afbeeldingen van scenepunten geschoten vanuit een ander vlak bestaande uit een array van camera's die het object omringen. Een alternatieve manier is een vlak bestaande uit een array van reflectie-afbeeldingen bij te houden voor ieder vlak van scenepunten [Levoy and Hanrahan [58]].
Om een nieuw beeld te genereren dient iedere lichtstraal vanuit het nieuwe camerastandpunt door iedere pixel in het nieuwe beeld getraceerd te worden. Iedere lichtstraal wordt met de light slab gesneden waardoor ze op dezelfde manier als alle lichtstralen in het 4D lichtveld wordt geparametriseerd. Als deze lichtstraal reeds bemonsterd is, kan de radiantie worden opgezocht in een database aan de hand van de parameters. Indien deze lichtstraal nog niet in het lichtveld is bemonsterd, worden de lichtstralen, voorgesteld door buren in de 2 vlakken, opgezocht. De radiantie voor de gezochte lichtstraal kan dan worden berekend door quadrilineaire interpolatie tussen de 4D parameters die buurstralen voorstellen. Dit is in essentie eigenlijk een lineaire interpolatie van de posities en richtingen van de gevonden naburige lichtstralen. Deze interpolatie geeft een schatting van de radiantie voor de te zoeken lichtstraal.
Wanneer het lichtveld voldoende bemonsterd is, geeft rendering met behulp van lichtvelden doorgaans kwalitatief zeer goede resultaten en deze methode is ook robuust. Deze thesis wenst echter een nieuw beeld te genereren vanuit een beperkt aantal camerastandpunten. De grote hoeveelheid aan beeldinformatie die nodig is om het lichtveld op te slaan maakt deze techniek onbruikbaar voor het toepassingsdomein van deze thesis.

3.1.2 Rendering m.b.v. ongestructureerde lichtgrafen

Een ongestructureerde lichtgraaf is net als een lichtveld een bemonstering van de plenoptische functie met dit verschil dat voor rendering met behulp van lichtgrafen cameraposities niet in een reguliere vlakke grid moeten worden opgesteld, maar vrij gepositioneerd en georiënteerd mogen worden en ook de focuslengten mogen verschillen [Buehler et al. [11]]. De parametrisatie is hetzelfde als bij lichtvelden. Een bijkomende vereiste is dat camera's gekalibreerd zijn [Magnor [67]]. Tijdens het renderproces wordt een camerablending veld gereconstrueerd dat aanduidt welke pixel in welke bemonsterde afbeelding in welke mate bijdraagt tot de kleur van een pixel uit het nieuwe beeld. Per nieuwe lichtstraal worden de beste (naburige) lichtstralen geselecteerd. Die lichtsamples waarvan de hoek het minste afwijkt van deze nieuwe lichtstraal krijgen een groot gewicht toegekend. Vermits kalibratieparameters gekend zijn kunnen deze hoeken berekend worden en werkt deze methode op vrije camerastandpunten. Deze blending lijkt erg op het inkleuren via view dependent texture mapping met dit verschil dat in het geval van VDTM alle pixels van een polygon met dezelfde dichtsbijzijnde camera-afbeeldingen worden ingekleurd en niet met de dichtsbijzijnde lichtstralen die verspreid kunnen zijn over meerdere afbeeldingen. Als er een geometrische benadering van het in te kleuren object gekend is kunnen naast hoeken ook de graad van bemonstering van bemonsterde lichtstralen en visibiliteit in rekening worden gebracht. Wanneer er voor bepaalde rays die naburig zijn een sterke onderbemonstering is ten op zichte van een geometrisch benadering van het object kan een negatief tegengewicht worden toegekend aan zulke stralen zodat ze minder gaan bijdragen. Naburige bemonsterde lichtstralen die de geometrische benadering niet zien worden genegeerd via een vermenigvuldiging met een gewicht van grootte 0.
In [Buehler et al. [11]] worden blendinggewichten niet op iedere pixel in het nieuwe beeld berekend, maar wordt het beeld opgedeeld in driehoeken om projectieve texture mapping en lineaire interpolatie van de grafische hardware te kunnen benutten. Het aantal texture units voor het opslaan van bronafbeeldingen op de grafische kaart om de nieuwe afbeelding in te kleuren via projectieve texture mapping is beperkt [Magnor [67]]. Ongestructureerde lichtgrafen laten toe dat de bemonsteringsdichtheid die bij lichtvelden nodig is voor goede renderingresultaten mag gereduceerd worden omdat er uit reeds gekende geometrische informatie bruikbare informatie zoals visibiliteit, en bemonsteringsinformatie per geometrieoppervlak kan worden afgeleid.

3.1.3 Stereo

Andere beeldgebaseerde methoden [Chen and Williams [20]], [Seitz and Dyer [94]], [Stich et al. [100]] die lichtstralen niet parametriseren en een nieuw beeld door middel van pixelreprojectie of interpolatie berekenen, vereisen betrouwbare correspondentie-, of diepte-informatie. Deze informatie kan via stereotechnieken berekend worden. Binoculaire stereotechnieken construeren als resultaat een dieptemap uit 2 beelden voor de pixels die in beide afbeeldingen voorkomen. Deze dieptemap kan worden berekend eens de corresponderende pixels in beide beelden gevonden zijn. Correspondenties worden berekend op basis van een vergelijking van pixelintensiteitswaarden. Een voorbeeld om de kost voor overeenkomst van 2 pixels te bepalen is door het verschil van de absolute waarden tussen kandidaatpixels te berekenen. In [Scharstein and Szeliski [92]] worden lokale als globale optimalisatiemethoden besproken om de pixels te matchen waarvoor de kost lokaal, of globaal over de hele afbeelding gezien, minimaal is. Het zoeken van correspondenties kan tot een zoektocht over een lijn vereenvoudig worden door beeldgeometrie-informatie te benutten [Hartley and Zisserman [38]]. Een correspondentie moet zich namelijk op de projectieafbeelding $e_{i,j}$ van een viewing ray r_i, gevormd door een pixel p_i op het beeldvlak I_i, volgens projectiematrix P_j op het beeldvlak I_j bevinden, met $i \neq j$. De projectieafbeelding van een viewing ray uit het ene beeld op het andere beeld wordt ook wel epipolaire lijn genoemd.
Uit correspondenties kan op 2 manieren diepte worden berekend: via dispariteiten en via triangulatie. Beide technieken zijn geïllustreerd in figuur 3.3. Dispariteitsberekening die de verplaatsing tussen overeenkomstige pixels over verschillende afbeeldingen beschrijft, veronderstelt voor diepteberekening gerectificeerde afbeeldingen, d.w.z. dat de beeldvlakken gealigneerd zijn in eenzelfde vlak. Wanneer beelden softwarematig gerectificeerd worden, is camerakalibratie vereist [Hartley and Zisserman [38]]. In een gerectificeerde opstelling is de verplaatsing tussen overeenkomstige pixels omgekeerd evenredig met de diepte van het 3D punt waarvan de correspondenties de projectieafbeelding zijn. De verplaatsing doet zich in dit geval enkel in de x-richting voor vermits beelden gealigneerd zijn tot horizontale epipolaire lijnen over beide afbeeldingen. Een andere manier om diepte te berekenen uit correspondenties is via triangulatie waarbij de projectiematrices via camerakalibratie moeten gekend zijn.
Het triangulatieproces berekent het snijpunt van de reprojectiestralen van corresponderende pixels (aangeduid in 3.3(b) met groen vierkant) in 3D. Dit levert het punt op waarvan de corresponderende pixels de projectieafbeeldingen zijn. In praktijk zullen deze reprojectiestralen door afrondingsfouten elkaar vaak niet snijden. Er kan dan via een numerieke benaderings-

methode zoals SVD(singular value decomposition) een oplossing gevonden worden. Dit is het punt dat op een minimale afstand ligt tussen beide kruisende rechten.
Voor de berekening van een volledige en nauwkeurige dieptemap zijn betrouwbare correspondenties nodig en dus ook een smalle camerabaseline. Anders kunnen occlusies optreden waarbij er voor pixels uit het ene beeld geen overeenkomst bestaat in het andere beeld omdat de beelden te verschillend zijn. Een ander probleem is dat correspondenties moeilijk overweg kunnen met speculaire regionen, die positie-afhankelijk zijn. Wanneer betrouwbare correspondentie- en diepte-informatie via stereo kan worden berekend, kan deze informatie gebruikt worden voor het genereren van een nieuw beeld via pixelreprojectie. Pixelreprojectiemethoden kunnen worden ingedeeld naargelang hun mappingrichting (geïllustreerd in figuur 3.4):

- voorwaartse mapping

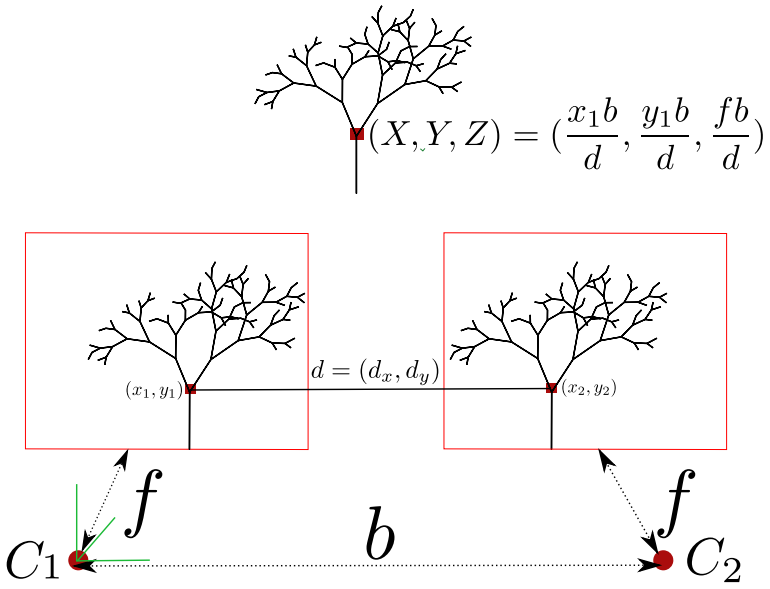
  Voorwaartse mapping transformeert punten uit de inputafbeeldingen via de image warping-vergelijking [McMillan [73]] naar een nieuwe doelafbeelding. Deze vergelijking is volledig gedefinieerd wanneer projectiematrices en de 3D dieptes van de pixels in de inputafbeeldingen gekend zijn. Bronpixels kunnen hierbij naar een subpixel gemapt worden en toekenning van deze bronpixel via dichtste buurmethode aan een doelpixel kan duplicaten veroorzaken (many-to-1 mapping) en holtes indien sommige doelpixels niet worden ingevuld. Zelfs indien de mapping 1 op 1 zou zijn dan nog zouden er holtes kunnen ontstaan door magnificatie en disocclusie [Kang et al. [42]]. Wanneer het nieuw camerastandpunt dichter bij de scene staat zullen de pixels nauwkeurigere informatie bevatten dan wanneer het camerastandpunt ver van de scene verwijderd is. Dit houdt dus in dat pixels in ver verwijderde camerastandpunten voor meer scene-informatie instaan. De bijdragen van pixels kan zo tussen bron- en doelafbeeldingen verschillen en er zullen gaten optreden wanneer ver van de scene verwijderde bronpixels op dichterbij liggende doelpixels gemapt worden omdat dichterbij liggende pixels meer detailinformatie bevatten dan verder verwijderde pixels, waarbij er een informatietekort ontstaat in het dichterbij liggend beeld om alle pixels in te vullen. Disocclusies, pixels in de doelafbeelding die zichtbare informatie van een deel van de scene bevatten, die niet zichtbaar is in de bronafbeelding, kunnen ook gaten veroorzaken [Kang et al. [42]].

- achterwaartse mapping

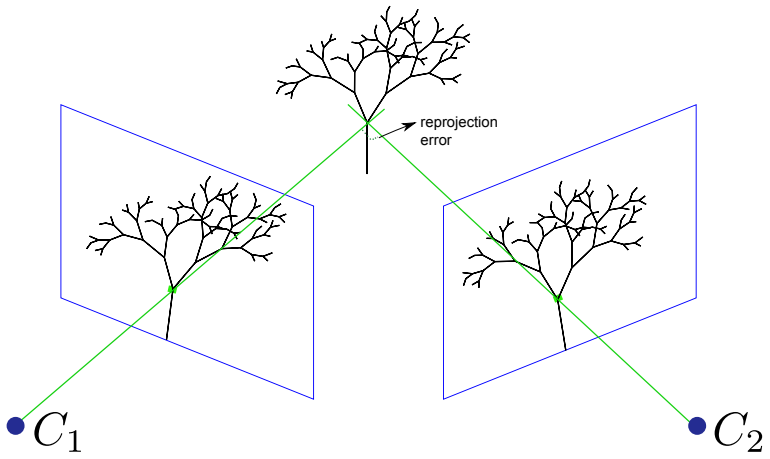
  Bij inverse, of achterwaartse mapping worden pixels in de gewenste

doelafbeelding gemapt naar corresponderende pixels in de bronafbeeldingen[Kang et al. [42]] die de inkleuring van de doelpixel bepalen. De corresponderende pixel kan worden geselecteerd als de dichtsbijzijnde pixel in de bronafbeelding tot het doelcamerastandpunt die via warping naar de doelpixel mapt. Op die manier worden gaten vermeden, ten koste van een zoekoperatie naar de dichtsbijzijnde pixel in de bronafbeelding. Wanneer de doelpixel geoccludeerd is in de bronafbeelding zal de zoekoperatie een verkeerd resultaat opleveren.

Beeldinterpolatie tussen 2 camera's in [Chen and Williams [20]] veronderstelt dat diepte-informatie gekend is door gebruik te maken van synthetische data. Met behulp van diepte worden dichte correspondenties tussen bronafbeeldingen berekend via voorwaartse mapping tussen bronafbeeldingen onderling. Vervolgens kunnen de verplaatsingsvectoren tussen correspondenties worden berekend die deze voorwaartse mapping vectorieel voorstellen. Tussenliggende beelden worden dan gegenereerd door lineaire interpolatie van deze verplaatsingsvectoren en blending van bronpixels. Deze lineaire interpolatie is op zich ook een voorwaartse mapping waardoor ze ook dient rekening te houden met gaten. Bovendien zal deze interpolatie niet steeds een fysiek correcte representaties opleveren[Chen and Williams [20]]. Multiview stereo technieken maken van meer camera's gebruik dan binoculaire stereo en hebben vaak als doel een 3D model of meerdere dieptemappen te reconstrueren [Seitz et al. [95]]. Een voorbeeld van een beeldgebaseerde beeldinterpolatietechniek die meerdere dieptemappen reconstrueert en zeer goede resultaten geeft op reële inputbeelden is [Zitnick et al. [112]]. In deze techniek worden correspondenties tussen afbeeldingen gezocht om uit de dispariteiten tussen gesegmenteerde regio's een dieptemap te reconstrueren. Een nadeel van deze techniek voor grote scenes is dat om goede resultaten te bereiken de baselines voor camera's binnen een bepaalde grens van 150 pixels (op een breedteresolutie van 1024) van elkaar moeten staan. Bovendien neemt de fout van diepteberekening via stereocorrespondenties kwadratisch toe met de afstand [Gallup et al. [33]]. Dit zou kunnen verholpen worden door de camera baseline en de resolutie te verhogen. Het verhogen van de camerabaseline zal in de techniek van [Zitnick et al. [112]] echter problemen geven voor het vinden van correspondenties.

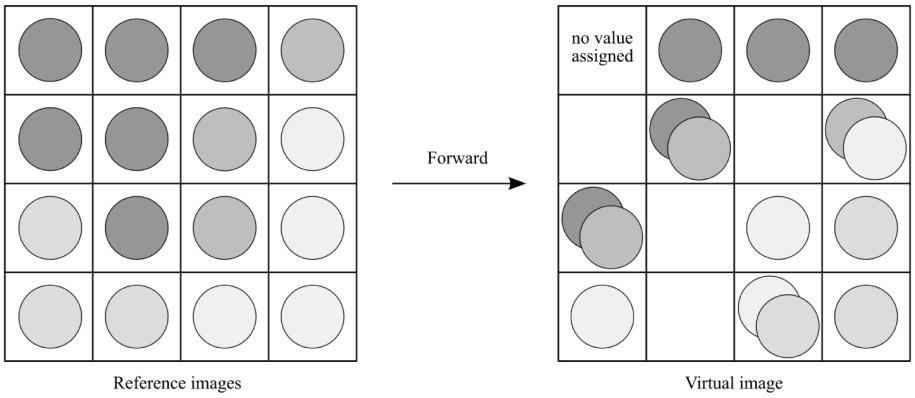


(a) Diepterecon structie via stereodispariteiten bij gerectificeerde beeldvlakken (rode rechthoeken). De diepte Z van een punt P, dat geprojecteerd wordt op corresponderende pixels (aangeduid met rode vierkanten) $p_1 = (x_1, y_1)$ en $p_2 = (x_2, y_2)$, is gelijk aan $Z = \frac{bf}{d}$, waarbij b, de baseline-afstand tussen beide camerastandpunten voorstelt, f de focuslengte, i.e. de afstand van het cameraprojectiecentrum tot het beeldvlak en d is de dispariteit tussen een gevonden correspondenties.

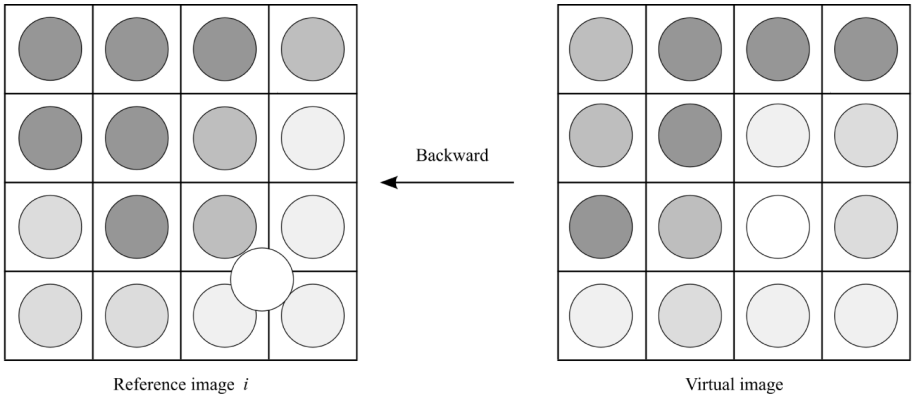


(b) Diepterecon structie van niet-gerectificeerde beeldvlakken via triangulatie.

Figuur 3.3: Diepterecon structie via stereo.



(a) Voorwaartse mapping.



(b) Achterwaartse mapping.

Figuur 3.4: Pixelreprojectie: illustratie van effecten van voorwaartse en achterwaartse mapping. Gridvierkanten stellen pixels voor, cirkels de toegekende kleurwaarden. Bron afbeelding: [Maunder [72]].

3.2 3D reconstructietechnieken

Een andere manier om een beeld te interpoleren is door middel van 3D reconstructie en het kiezen van een nieuw camerastandpunt. Een 3D multiview stereo reconstructietechniek waarvoor geen correspondenties nodig zijn en die goed presteert onder brede camerabaselines is de visual hull [Magnor [67]]. Alvorens dieper op de visual hull in te gaan, definiëren we een hull-techniek, ook wel closure genoemd [Erné [30]]:

Definitie 3.2.1. Een closure of hull-operator is een operator met de volgende eigenschappen:

- Uitbreidend: $O \subseteq Hull(O)$

  Dit houdt in dat de hull van een object O, het object volledig moet bevatten.

- Monotoon: $S \subseteq O \implies Hull(S) \subseteq Hull(O)$

  De hull-operator toegepast op een deelverzameling S van een object O is ook een deelverzameling van de hull operator toegepast op de volledige verzameling O.

- Idempotent: $Hull(Hull(O)) = Hull(O)$

  Het toepassen van de operator op zijn resultaat, moet hetzelfde resultaat geven. De operator is dus telkens 'even krachtig'.

3.3 Conclusie

In dit hoofdstuk werd onderzocht in welke mate wiskundige interpolatie toepasbaar is op beelden, i.e. wanneer de scene zodanig bemonsterd en geparametriseerd is zodat wiskundige interpolatie fysische correcte representaties toelaat en de informatie die door middel van interpolatie moet worden ingevuld, beperkt is zoals bij rendering door middel van lichtvelden. Beeldinterpolatie omvat alle mogelijke manieren om een nieuw beeld te synthetiseren en is dus een veel ruimer begrip dan het louter toepassen van wiskundige interpolatie op beelden. Tijdens de bespreking van verschillende beeldinterpolatiemethoden werd een onderscheid gemaakt tussen beeldgebaseerde renderingtechnieken en geometrisch gebaseerde renderingtechnieken. Stapsgewijs werden verschillende beeldgebaseerde interpolatietechnieken overlopen gaande van veel camera's (lichtvelden) tot gebruik van minder camera's zoals

bij pixelreprojectiemethoden. De belangrijkste beweegreden voor het opteren voor de visual hull als de aangewezen methode voor beeldinterpolatie van grote sportscene, is omdat deze methode goede resultaten toelaat wanneer relatief weinig camera's gegeven zijn en camera's door een brede camera-baseline van elkaar gescheiden zijn omwille van de grote omvang van een sportstadium.

Hoofdstuk 4

Visual hull

De visual hull techniek laat toe beelden te interpoleren door de objecten in 3D te reconstrueren en een nieuw camerastandpunt te kiezen. Shape from silhouette (SFS) technieken reconstrueren uit gegeven silhouette-afbeeldingen een vorm die consistent is met de gegeven silhouette afbeeldingen. De maximale vorm die consistent is met de gegeven silhouette-afbeeldingen wordt de visual hull genoemd. Gewenst is dat de visual hull het oorspronkelijk object zo goed mogelijk benadert voor een realistische reconstructie en dus ook een realistische beeldinterpolatie. De mate waarin dit mogelijk is en wat deze techniek precies inhoudt, wordt in dit hoofdstuk besproken. Het hoofdstuk begint met een theoretische basis over visual hulls en breidt zich naar het einde toe uit naar een bespreking van enkele praktisch bruikbare algoritmen voor visual hull reconstructie. De theoretische terminologie, definities en bewijzen uit dit hoofdstuk zijn sterk geïnspireerd op het werk van Laurentini [Laurentini [52]].

4.1 Definitie

De visual hull VH(O,V) van een object O ten opzichte van een viewregio V, dit is de regio waarin camera's op alle mogelijke posities met mogelijke kijkrichtingen worden opgesteld, is als volgt gedefinieerd:

Definitie 4.1.1. VH(O,V) = verzameling van alle punten $P_k \in E_3$, de Euclidische 3D ruimte, waarbij voor ieder punt voor alle mogelijke camerastandpunten $C_i \in V$ moet gelden dat iedere viewing ray $r_j = \overrightarrow{C_i P_k}$, dit is de halfrechte startend in C_i door het punt P_k, minstens 1 punt van het object O bevat. Dus ieder mogelijke r_j moet het object steeds snijden of raken.

Via een bewijs van de 3 eigenschappen die een hull operator volgens de definitie in hoofdstuk 2 moet bezitten, tonen we aan dat de visual hull een hull-techniek is.

Theorem 4.1.2 (**Hull Theorem).** *De visual hull is een hull operator.*

- $O \subseteq VH(O,V)$

  Bewijs. Ieder punt dat deel uitmaakt van het object voldoet aan de definitie van VH(O,V). Het te reconstrueren object O zal dus steeds een subset zijn van VH(O,V). □

- $X \subseteq Y \implies VH(X,V) \subseteq VH(Y,V)$

  Bewijs. Stel $X \subseteq Y$ en $\exists$ punt P $\in$ VH(X,V) waarbij $P \nsubseteq VH(Y,V)$. Volgens de definitie van VH(Y,V) mag er dan geen viewing ray bestaan door P die Y snijdt. Dit is in contradictie met het gestelde vermits $X \subseteq Y$ en dus alle viewing rays die X snijden ook Y moeten snijden. Bijgevolg kan zo een punt P niet bestaan. Hieruit volgt dat de stelling dat $\exists P \in VH(X,V)$ waarbij $P \nsubseteq VH(Y,V)$ fout is en er dus moet gelden dat $VH(X,V) \subseteq VH(Y,V)$. □

- VH(VH(O,V), V) = VH(O,V)

  Bewijs. Stel $\exists$ punt P $\in$ VH(VH(O,V)) waardoor alle viewing rays VH(O,V) snijden en O niet allemaal snijden. Dan zou er een viewing ray bestaan door VH(O,V) die O niet snijdt. Zo'n viewing ray kan niet bestaan vermits bij definitie VH(O,V) net de verzameling van punten is waarvan alle viewing rays door deze punten object O zullen snijden. Bijgevolg bestaat er zo geen P $\in$ VH(VH(O,V), V) die $\notin$ VH(O,V). Er bestaat ook geen punt $Q \in VH(O,V)$ waardoor alle viewing rays O snijden en VH(O,V) niet allemaal snijden vermits O $\subseteq$ VH(O,V). Er is dus geen punt Q $\in$ VH(O,V) die $\notin$ VH(VH(O,V),V). Hieruit volgt dat VH(VH(O,V), V) en VH(O,V) dezelfde verzameling van punten moeten beschrijven. □

Theorem 4.1.3 (**Maximale Silhouette Consistente Shape Theorema).** *VH(O,V) is de maximale vorm die silhouette-equivalent is met het te reconstrueren object O.*

Dit theorema houdt dus in dat wanneer we het object in de scene met deze visual hull of 3D vorm zouden vervangen, de silhouette-afbeeldingen geschoten vanuit de gegeven view regio van de visual hull een perfecte match moeten zijn met de silhouette-afbeeldingen van het oorspronkelijke object.

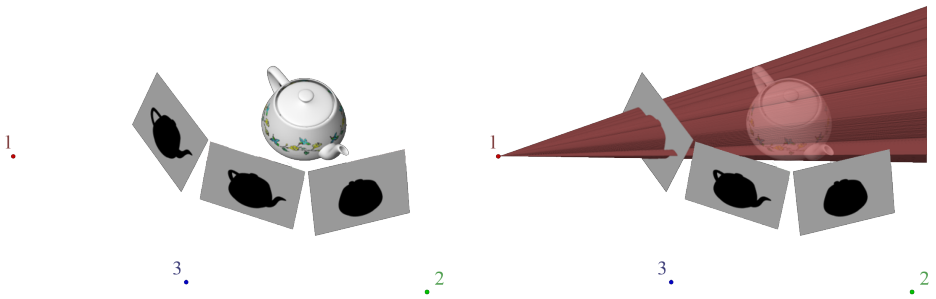
Bewijs. Om de silhouette-equivalentie van VH(O,V) met O aan te tonen moet gelden dat:

- De projectie van elk punt dat deel uitmaakt van VH(O,V) in een camerastandpunt $C \in V$ maakt deel uit van de silhouette afbeelding van O geschoten vanuit C.
- De projectie van elk punt dat deel uitmaakt van O in een camerastandpunt $C \in V$ maakt deel uit van de silhouette afbeelding van VH(O,V) geschoten vanuit V.

De eerste voorwaarde volgt bij constructie rechtstreeks uit de definitie vermits iedere hergeprojecteerde ray door een punt VH(O,V) ook door minstens 1 punt van O gaat waaruit volgt dat de afbeelding van alle punten $\in$ VH(O,V) ook een afbeelding zijn van het silhouette van O voor elk camerastandpunt C. De tweede voorwaarde volgt uit de eigenschap dat O $\subseteq$ VH(O,V). Aangezien O bevat zullen projecties van O steeds binnen een silhouette afbeelding van VH(O,V) vallen. Bij contradictie kan de maximale consistentie van VH(O,V) worden aangetoond. Stel dat VH(O,V) niet maximaal consistent is en er nog punten bestaan die geen element zijn van VH(O,V), maar toch equivalent zouden zijn met de silhouetten van O. Voor alle punten M $\notin$ VH(O,V) bestaat er een viewing ray die O niet snijdt, hetgeen volgt uit de definitie. Bijgevolg kunnen zulke punten M geen deel uitmaken van een silhouette van S vanuit een beschouwd camerastandpunt C, want de viewing ray vanuit C snijdt S niet. Deze punten M kunnen geen deel uitmaken van een vorm die silhouette equivalent is met O en kunnen dus niet bestaan. Hieruit kunnen we concluderen dat de verzameling van punten VH(O,V) maximaal silhouette consistent en equivalent is O. □

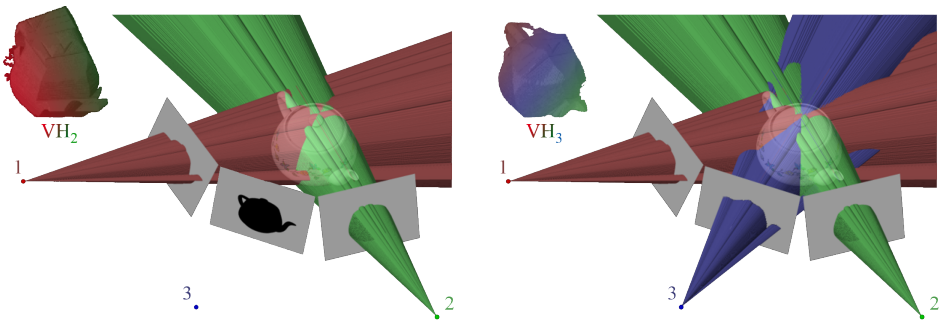
Theorem 4.1.4 (Beste Approximatie Uit Silhouetten Theorema). *De visual hull VH(O,V) is de beste geometrisch conservatieve benadering van object O die uit gebruik van enkel silhouette informatie kan worden bekomen ten op zichte van view regio V.*

Bewijs. Een silhouette-afbeelding per camerastandpunt beschrijft de afbeelding van de viewing rays voor het beschouwde camerastandpunt door het beeldvlak die O steeds zullen snijden, of raken. De punten die het snijpunt



(a) Gegeven silhouette-afbeeldingen van 3 camera's.

(b) Toevoeging van gereprojecteerd silhouettevolume vanuit eerste camera.



(c) Toevoeging van gereprojecteerd silhouettevolume vanuit tweede camera.

(d) Toevoeging van gereprojecteerd silhouette volume vanuit derde camera.

Figuur 4.1: Illustratie van visual hull reconstructie uit 3 silhouette-afbeeldingen.

zijn van alle rays vanuit ieder camerastandpunt door de silhouettevorm vanuit dat camerastandpunt, voldoen aan de definitie en zijn dus ook consistent met iedere silhouette-afbeelding. De visual hull is dan de maximale doorsnede van alle gereprojecteerde silhouettevolumes uit alle mogelijke camerastandpunten en -kijkrichtingen die deel uitmaken van de beschouwde view regio zoals weergegeven in figuur 4.1. Deze maximale doorsnede bevat geen punten buiten een silhouette reprojectievolume en dus een silhouette. Hieruit volgt dat de visual hull de beste geometrisch conservatieve benadering is van een object O die uit volume-intersecties van silhouette-afbeeldingen kan worden berekend. Met conservatieve benadering wordt bedoeld dat het oorspronkelijke object steeds gegarandeerd bevat zal zijn in de visual hull: O $\subseteq$ VH(O,V). □

Volgens de definitie VH(O,V) is de visual hull afhankelijk van:

- het object O
- de view regio V

4.2 View regio's

Volgens de definitie VH(O,V) hangt de visual hull af van de view regio. Grotere view regio's laten meer camerastandpunten, meer silhouette-afbeeldingen en meer constraints op de visual hull punten toe om vanuit iedere mogelijke view aan de definitie te voldoen en dus ook een betere benadering van het object O toe.

Lemma 4.2.1. $V' \subseteq V \implies VH(O,V) \subseteq VH(O,V')$

In praktijk is de bruikbare view regio vaak beperkt vooral wanneer deze zich ook binnen de convex hull van het te reconstrueren object bevindt. In een reëele omgeving passen camera's soms niet in de concave holtes van een object. Omwille van deze praktische beperkingen wordt er een onderscheid gemaakt tussen een visual hull geconstrueerd vanuit standpunten buiten de convex hull van het te reconstrueren object en visual hull geconstrueerd vanuit alle standpunten buiten het object. In het eerste geval spreken we van de visual hull van het object(VH), ook wel de externe visual hull (EVH) genoemd [Laurentini [52]] en in het laatste geval spreken we van de interne visual hull van het object(IVH).

4.3 Visual hull

De visual hull of externe visual hull van een object is de visual hull gereconstrueerd vanuit alle mogelijke camerastandpunten en kijkrichtingen buiten de convex hull van het te reconstrueren object:

Definitie 4.3.1. VH(O) = VH(O, $E^3 - CH(O)$)

, waarbij E^3 de Euclidische 3D ruimte is en $CH(O)$ de convex hull van het object O voorstelt (zie convex hull hoofdstuk in bijlage).

De view regio $E^3 - CH(O)$, waarvan een voorbeeld is afgebeeld in figuur 4.3(c), is oneindig. View regio's buiten de convex hull, die het object voor 360 graden omringen, bevatten alle silhouette-informatie die buiten de convex hull van het object kan worden waargenomen. Als we ∞ aantal camerastandpunten in iedere view regio veronderstellen, bevatten zulke omringende view regio's op verschillende afstanden, evenveel informatie voor visual hull reconstructie, tenminste als er geen andere objecten zijn die de rechtstreekse view op O kunnen verhinderen.

Theorem 4.3.2 (Unieke VH theorema)**.** *Gegeven 1 object O. Zij $V_c \subset E^3 - CH(O)$ een willekeurige deelregio buiten CH(O) die O volledig omringt, i.e. een aaneensluitende regio die een ruimte afbakent waarin O volledig bevat is. Er bestaat een unieke visual hull voor alle view regio's V_c. VH(O) = VH(O,V_c) = unieke visual hull.*

Bewijs. Bovenstaand theorema geldt als er kan bewezen worden dat uit 2 willekeurige view regio's buiten de convex hull van O, die O volledig omringen enkel dezelfde, unieke visual hull kan gereconstrueerd worden. Zij V_1, V_2 zo 2 willekeurige V_c - regio's en zij C_1 een camerastandpunt in V_1 en C_2 een camerastandpunt in V_2 zoals afgebeeld in figuur 4.2(b).

- Camerastandpunten C_1 en C_2 hebben steeds een ongehinderde viewing ray naar object O. Dit geldt uit de gegevens omdat er buiten de convex hull zich geen obstakels tussen view regio's kunnen bevinden: $O \subseteq CH(O)$ en O is het enige object. Anders zouden andere objecten een bepaalde view kunnen verhinderen en zouden de existentiële veronderstellingen in het vervolg van dit bewijs niet steeds waar zijn.

- $\forall$ punt $P_1 \in$ VH(O,V_1) $\wedge$ $\forall \overrightarrow{C_1P_1}$, $\exists$ $C_2 \in \overleftrightarrow{C_1P_1}$. Toepassing van de definitie houdt dus in dat $\forall P_1 \in$ VH(O,V_1) $\implies$ $P_1 \in$ VH(O,V_2) en dus VH(O,V_1) $\subseteq$ VH(O,V_2).

- $\forall$ punt $P_2 \in$ VH(O,V_2) $\wedge$ $\forall \overrightarrow{C_2P_2}$, $\exists$ $C_1 \in \overleftrightarrow{C_2P_2}$. Toepassing van de definitie houdt dus in dat $\forall P_2 \in$ VH(O,V_2) $\implies$ $P_2 \in$ VH(O,V_1) en dus VH(O,V_2) $\subseteq$ VH(O,V_1).
- VH(O,V_1) $\subseteq$ VH(O,V_2) $\wedge$ VH(O,V_2) $\subseteq$ VH(O,V_1) $\iff$ VH(O,V_1) = VH(O,V_2)

□

Uit dit theorema volgt dat de view regio $E^3 - CH(O)$ zonder informatieverlies vanuit theoretisch standpunt herleid kan worden naar een viewregio $V_c \subset E^3 - CH(O)$ die het object volledig omringt. View regio's die het object omringen bevatten evenveel informatie indien er geen overige objecten een rechtstreekse kijk op het object kunnen verhinderen en beschrijven dus dezelfde verzameling van viewing rays.

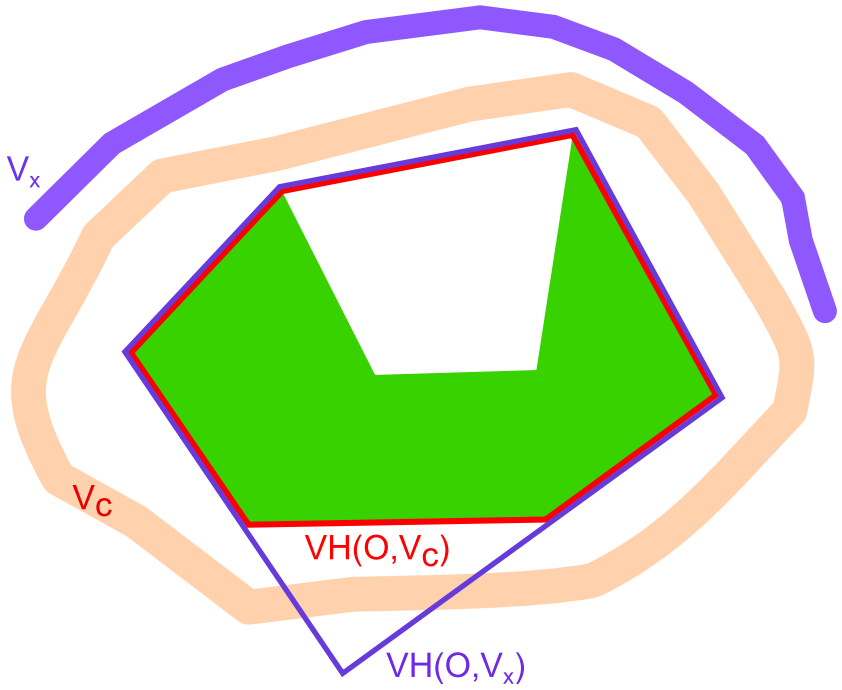
De voorwaarde dat de viewpoints CH(O) volledig moet omringen is strikt om van een unieke VH(O) te kunnen spreken. Volgende opmerkingen halen aan dat zowel inkrimpingen als uitbreidingen van een view regio V_c niet steeds in de unieke VH(O) zullen resulteren.

- VH(O,V_c) $\subseteq$ VH(O,V_x)

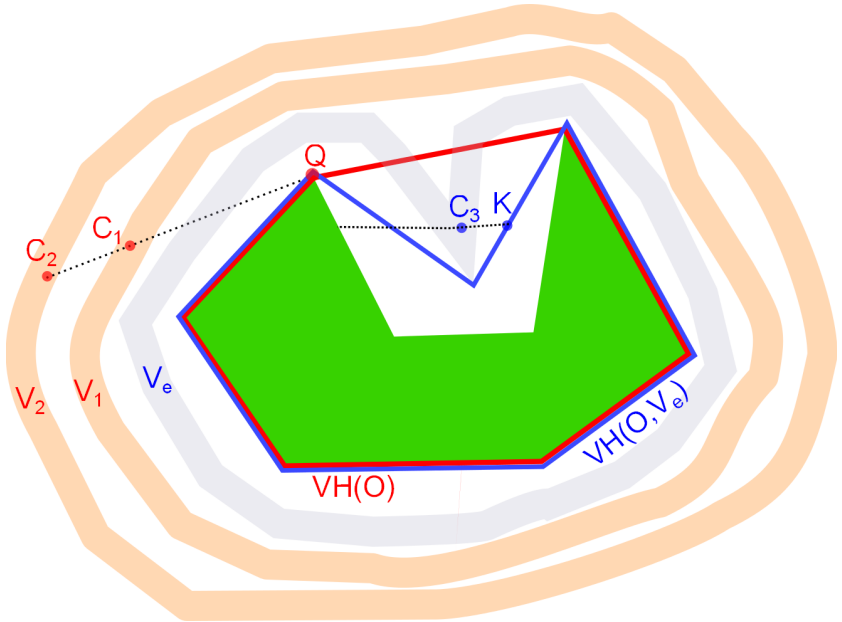
  Uit lemma 4.2.1 volgt dat deelregio's $V_x \subset V_c$ niet steeds de unieke visual hull kunnen reconstrueren zoals afgebeeld in 4.2(a). Inkrimpingen zijn mogelijk voor die verzameling deelregio's V_x die dezelfde verzameling van meest beperkende constraints beschrijft als VH(O,V_c) en waarvoor de unieke visual hull gereconstrueerd kan worden.

- Uniciteitseigenschap geldt zolang viewpoints $V_c \notin$ VH(O).

  Zij V_e een extensie van een $V_c - viewregio$ die VH(O) betreedt. In dat geval zal er niet steeds een ongehinderde viewing ray bestaan naar een willekeurig camerastandpunt buiten de convex hull omdat het object zelf deze viewing ray kan onderbreken. Dit is geïllustreerd in afbeelding 4.2(b) waarin een camerastandpunt C_3 dat gelegen is in CH(O) een viewing ray beschrijft die geen deel uitmaakt van bijvoorbeeld VH(O,V_1) omdat O de rechte C_1K onderbreekt. VH(O,V_1) en VH(O,V_e) beschrijven niet dezelfde verzameling van viewing rays. Volgens de afbeelding beschrijven ze ook niet dezelfde visual hulls en kunnen de meer uitgebreide view regio's de visual hull uitdiepen. Tijdens de bespreking van interne visual hulls wordt de motivering achter deze laatste opmerking besproken.

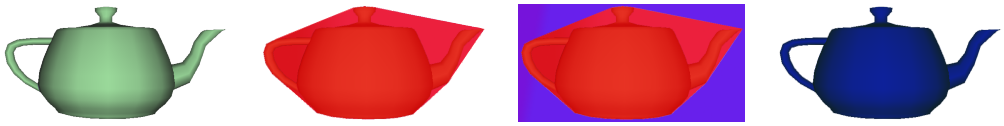


(a) O in het groen, CH(O)=VH(O,V_c) in het rood, VH(O,V_x) in het paars. VH(O,V_c) $\subseteq$ VH(O,V_x).



(b) CH(O)=VH(O,V_c(rood) met V_1,V_2 2 V_c regio's. Viewregio V_e met corresponderende visual hull VH(O,V_e) in het blauw.

Figuur 4.2: Illustratie van de strikte voorwaarden voor de uniciteitseigenschap van VH(O).



(a) O (groen). (b) CH(O) (rood). (c) Geldige camera-standpunten in view regio (E^3 - CH(O)) (paars). (d) VH(O) (blauw).

Figuur 4.3: Visual hull en convex hull van theepotobject.

De view regio V_c kan in theorie nog steeds een oneindig aantal camera-standpunten bevatten. Sommige objecten zijn wel aan de hand van een eindig aantal silhouetten exact reconstrueerbaar. Bovengrenzen van het nodige aantal camerastandpunten voor exacte reconstructie van VH(O) bij bepaalde types van objecten die maar een eindig aantal views voor exacte reconstructie van VH(O) nodig hebben, zijn terug te vinden in [Laurentini [53]],[Petitjean et al. [85]]. Algoritmen die de optimale kijkrichtingen voor visual hull reconstructie benaderen via heuristieken zijn onderzocht in [Bottino and Laurentini [8]], [Shanmukh and Pujari [96]], [Bottino et al. [9]]. In de volgende sectie worden de reconstrueerbare objecten overlopen waarvoor geldt dat O = VH(O).

4.3.1 Reconstrueerbare objecten

Met behulp van de visual hull methode kunnen de volgende objecten worden gereconstrueerd:

- convexe objecten
- concave objecten die silhouette-actief zijn t.o.v. $E^3 - CH(O)$

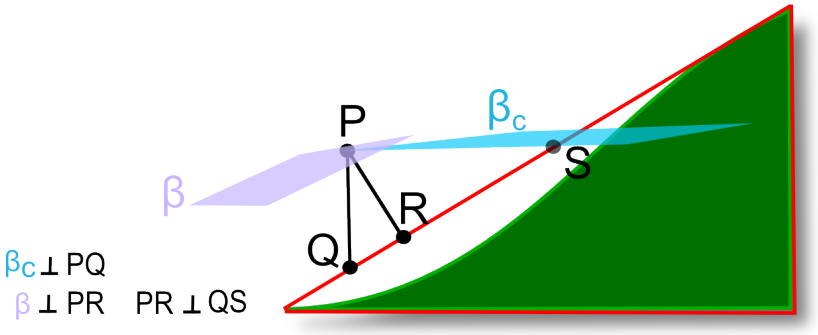
Convexe objecten

Een object O is convex $\Leftrightarrow O \cap CH(O) = \emptyset$. Als er bewezen kan worden dat $VH(O) \subseteq CH(O)$ geldt dan volgt hieruit dat alle convexe objecten reconstrueerbaar moeten zijn via de visual hull techniek.

Lemma 4.3.3. *$VH(O) \subseteq CH(O)$*

Bewijs. VH(O) $\subseteq$ CH(O) kan herleid worden naar de volgende equivalente predikaatuitdrukkingen:

- $\forall P(P \in VH(O) \implies P \in CH(O))$



Figuur 4.4: Illustratie bij bewijs dat CH(O) $\subseteq$ VH(O). Object O is aangeduid in het groen. CH(O) omlijnt in het rood.

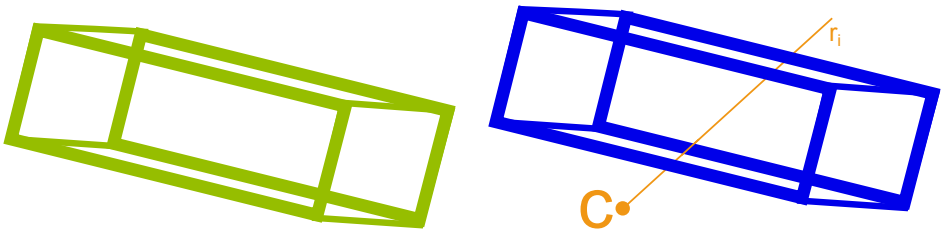
- $\forall P(P \notin CH(O) \implies P \notin VH(O))$ (contrapositie)

Zij P een willekeurig punt buiten de convex hull: $P \notin$ CH(O). Er bestaat steeds een kijkvlak β door P loodrecht op het lijnsegment gevormd door P en het punt Q op de convex hull het dichtst bij P gelegen dat CH(O) en dus O niet kan snijden. In dat geval zal er steeds een viewing ray bestaan in het vlak β die O niet zal snijden en volgt er uit de definitie van VH(O) dat P $\notin$ VH(O).
Via contradictie kan bewezen worden dat zo'n vlak β steeds bestaat. Stel bij contradictie dat een kijkrichting vanuit C in vlak β_c, waarbij subscript c duidt op het contradictievoorstel, toch CH(O) zou snijden in een punt S. Dit punt S kan niet hetzelfde punt voorstellen als Q omdat S $\in \beta_c$ in punt P loodrecht staat op PQ. Aangezien S $\in$ CH(O) en Q $\in$ CH(O) verschillende punten zijn, bestaat er een punt R gelegen op de lineaire tussen QS dat volgens de definitie CH(O) ook in de convex hull moet liggen. Punt R kan in $\triangle$PQS zodanig gekozen worden zodat PR $\perp$ QS. Vermits PR, loodrecht staat op QS is het de kortste afstand van QS tot P. Dit is strijdig met het gegeven dat stelde dat de kortste afstand van P tot CH(O) Q is, terwijl de afstand tot R korter is. Deze contradictie toont aan dat de stelling dat vlak β_c de convex hull kan snijden fout moet zijn omdat er in dat geval telkens bij constructie een kortere afstand tot P kan gevonden worden. Uit deze contradictie volgt dat er steeds een vlak β loodrecht op de rechte gevormd door P en het punt het dichtst bij P gelegen op de CH(O) moet bestaan dat CH(O) niet snijdt hetgeen bewezen moest worden. In afbeelding 4.4 is het vlak dat CH(O) niet zal snijden en loodrecht staat op PR aangeduid met letter β. $\square$

Wanneer in praktijk een view regio V_x het object niet volledig omringt, of uit een beperkt aantal camerastandpunten bestaat, zal de convex hull in sommige gevallen niet steeds in de visual hull VH(O,V_x) bevat zitten omdat kijkvlak β niet steeds een deelverzameling zal zijn van de gekozen view regio. Zo kunnen punten buiten de convex hull niet steeds door een kijkvlak geëlimineerd worden wegens een beperkt aantal kijkvlakken voor de gegeven view regio. VH(O, V_x) $\subseteq$ CH(O) zal dus in praktijk niet altijd gelden zoals afgebeeld is in figuur 4.2(a).

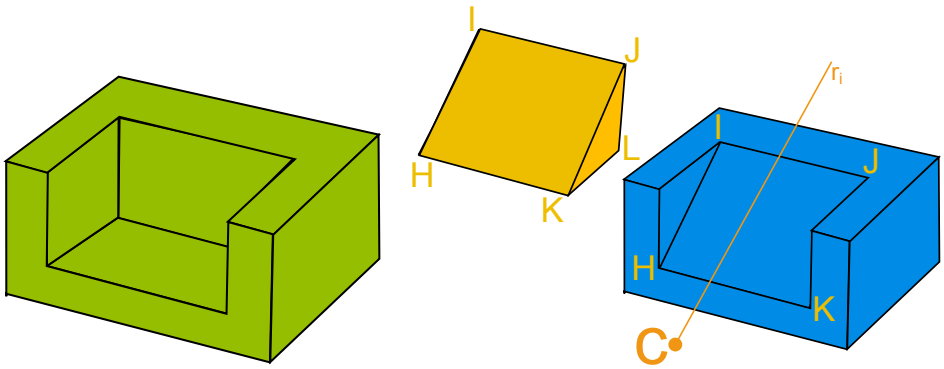
Concave objecten silhouette-actief t.o.v. $E^3 - CH(O)$

Een concaaf object is silhouette-actief indien een willekeurig wijziging in het oppervlak ook een wijziging in een silhouette-afbeelding voor een camera in $E^3 - CH(O)$ inhoudt. Concave niet-silhouette-actieve oppervlakten laten een wijziging van het oppervlak toe zonder dat hierdoor een silhouette-afbeelding in $E^3 - CH(O)$ wordt beïnvloed. Voor concave objecten die enkel bestaan uit silhouette-actieve oppervlakten t.o.v camerastandpunten buiten CH(O) zullen de object oppervlakten samenvallen met de VH(O)-oppervlakten omdat het object volledig door zijn silhouetten kan gedefinieerd worden en dus maximaal consistent is met deze silhouetten. Het visual hull oppervlak is begrensd door die viewing rays die inconsistent zijn in een of meerdere silhouette-afbeelding en buiten een silhouete vallen. Wanneer alle rays vanuit standpunten buiten CH(O) het object met zijn concaviteit kunnen uitsnijden dan kan het object aan de hand van de visual hull techniek gereconstrueerd worden. Lege oppervlakten waarvoor dit niet gaat vanaf standpunten buiten CH(O) zijn niet silhouette-actief t.o.v. $E^3 - CH(O)$ en kunnen eender welke vorm aannemen zonder de silhouete-afbeelding van het object te beïnvloeden [Laurentini [52]]. Voorbeelden van silhouette-actieve en silhouette niet-actieve oppervlakten zijn te zien in figuur 4.5.



(a) Silhouette-actief concaaf object O (groen).

(b) VH(O) van object in 4.5(a) (blauw).



(c) Niet-silhouette-actief concaaf object O (groen).

(d) VH(O) van object in 4.5(c) (blauw).

Figuur 4.5: Voorbeelden van een silhouette-actief object en silhouette niet-actief object en hun overeenkomstige visual hulls. Merk op dat silhouette-inconsisente rays r_i in figuur 4.5(b) vanuit een willekeurig camerastandpunt C alle concave regio's kunnen wegsnijden omdat het inputobject in 4.5(a) enkel uit silhouette-actieve oppervlakten bestaat en zo de visual hull van dit object wel gelijk moet zijn aan het object zelf. In 4.5(d) is de regio IHJKL niet silhouette-actief en kan dus ook niet via r_i-rays worden uitgesneden uit het inputobject 4.5(c). Vermits het inputobject uit afbeelding 4.5(c) niet volledig uit silhouette-actieve oppervlakten is opgebouwd verschilt de VH(O) van O.

4.4 Interne visual hull

De interne visual hull van een object is de visual hull gereconstrueerd vanuit alle mogelijke camerastandpunten en kijkrichtingen buiten het te reconstrueren object:

Definitie 4.4.1. IVH(O) = VH(O, $E^3 - O$)

4.4.1 Reconstrueerbare objecten

Met behulp van de interne visual hull die ook viewregio's binnen CH(O) toelaat kunnen de volgende objecten gereconstrueerd worden:

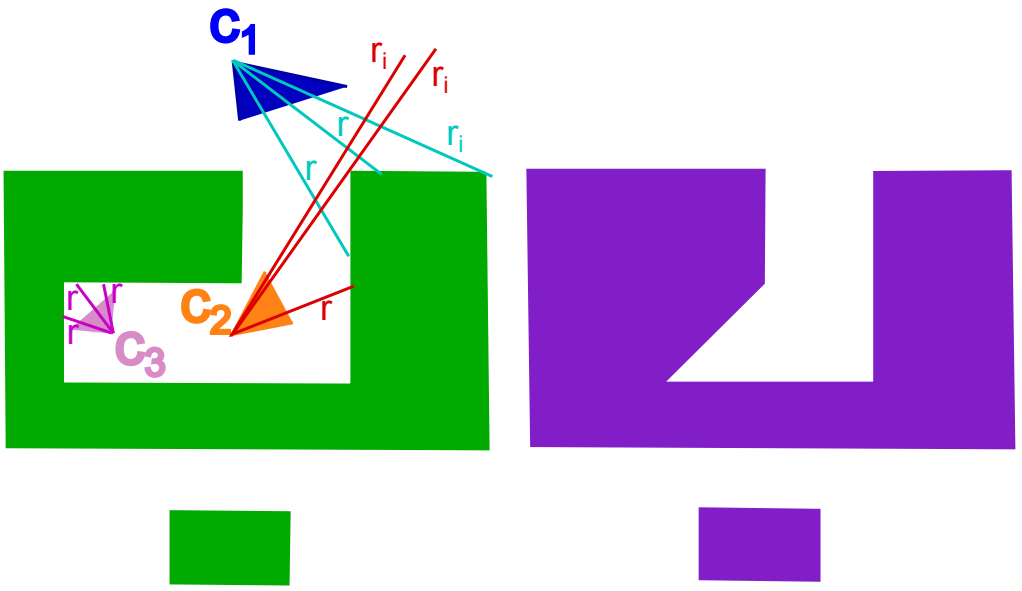
1. VH-objecten
2. Concave objecten silhouette-actief t.o.v. $E^3 - O$

Visual hull reconstrueerbare objecten

De view regio $E^3 - CH(O)$ zit bevat in de view regio $E^3 - O$. Er geldt dus volgens lemma 4.2.1 dat IVH(O) $\subseteq$ VH(O) en alle visual hull reconstrueerbare objecten ook door de interne visual hull reconstrueerbaar zijn.

Concave objecten silhouette-actief t.o.v. $E^3 - O$

Het IVH(O)-oppervlak is begrensd door die viewing rays die inconsistent zijn in een of meerdere silhouette-afbeeling en buiten een silhouete vallen. Doordat bij interne visual hulls de camerstandpunten zich binnen CH(O) uitbreiden kan het aantal inconsistent rays toenemen zolang er een ray bestaat die zonder door het object zelf gehinderd te worden CH(O) snijdt. Voor diepe concave objecten zal zo'n ray niet steeds bestaan. Dit wordt geïllustreerd in figuur 4.6



(a) Diep concaaf object O (groen).

(b) IVH(O) silhouette-actief t.o.v. $E^3 - O$(paars).

(c) VH(O) silhouette-actief t.o.v. $E^3 - CH(O)$ (blauw).

(d) CH(O) (rood).

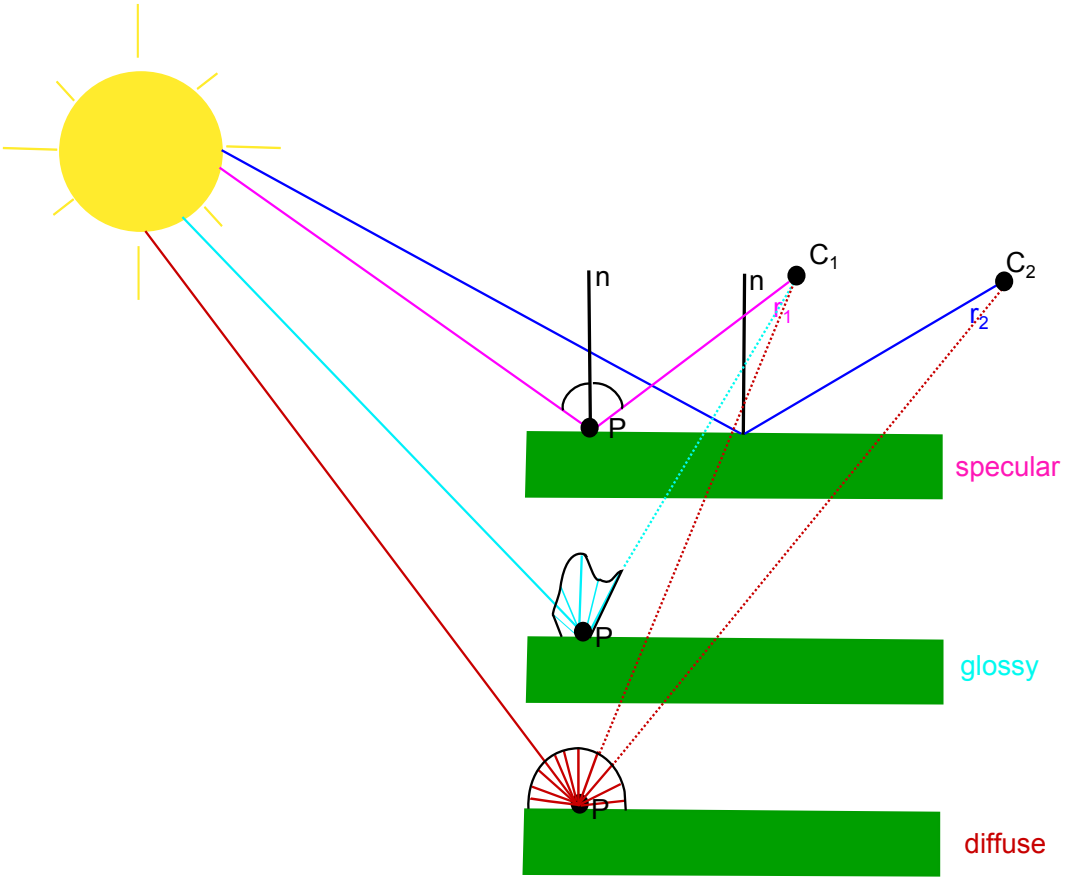
Figuur 4.6: Illustratie van hulls bij diep concaaf object. Merk op dat de interne visual hull niet gelijk is aan het object O doordat in diepconcave regio's er geen inconsistente rays aanwezig zijn die het object verder kunnen uitdiepen. Alles rays r in diep concave regionen zullen consistent zijn, in het object terechtkomen in alle silhouette-afbeeldingen en over de hele view angle een volume reprojecteren. Een voorbeeld hiervan is weergegeven door de consistente rays r vanuit camerastandpunt C_3. Camerastandpunt C_2 beschrijft een standpunt buiten CH(O) waar inconsistente rays vanuit dit standpunt de visual hull zullen uitdiepen, daar consistente rays r geen verdere uitdieping tot gevolg hebben. Camerastandpunt C_1 beschrijft een camerastandpunt binnen CH(O) in een concave regio die nog wel reconstrueerbaar is via de interne visual hull techniek doordat er nog inconsistente rays bestaan die de CH(O) ongehinderd kunnen snijden en zo voor verdere uitdieping zorgen.

4.5 Visual hull in de praktijk

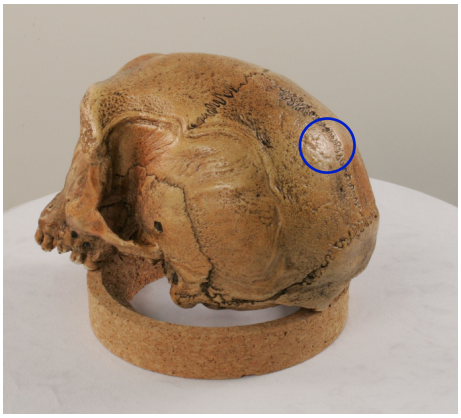
4.5.1 Voordelen

- Robuust

  Het reconstructieproces van visual hulls uit gegeven silhouetteafbeeldingen wordt niet door oppervlakte-eigenschappen beïnvloedt. Onderliggend vereist de visual hull techniek segmentatie op basis van kleuren en zo is de robuustheid van de visual hull techniek ook van de robuustheid van de segmentatie afhankelijk. Zolang een betrouwbare segmentatie van het te reconstrueren object mogelijk is, is de visual hull techniek robuust omdat deze enkel vorminformatie gebruik in tegenstelling tot vele andere technieken die ook kleurinformatie gebruiken en waar bijvoorbeeld speculaire en glanzende oppervlakken wel de robuustheid van de reconstructie kunnen beïnvloeden doordat de positie van de speculaire highlight verschilt naargelang de kijkrichting [jin Yoon and so Kweon [39]]. Bij een speculaire reflectie wordt het licht weerkaatst volgens een uitgaande richting die een spiegeling is van de invallende lichtrichting rond de normaal waarbij ingaande en uitgaande richting dezelfde hoekgrootte hebben ten opzichte van de normaal n [Dutre et al. [27]]. Verschillende oppervlakken met hun reflecties zijn afgebeeld in figuur 4.7. De speculaire lichtstraal r_1 die kaatst vanuit een punt P naar het projectievlak van camera C_1 zal afbeelden in een witte (sub)pixel voor punt P. Vanuit een verschillend camerastandpunt C_2 zal deze speculaire reflectie een andere intensiteit hebben doordat de speculaire reflectierichting r_1 niet op camera C_2 zal invallen op de pixel die een afbeelding is van punt P wegens de speculaire reflectiewet en de aanname dat C_2 verschilt van C_1. De zichtbaarheid van de speculaire reflectie is afhankelijk van de kijkrichting en zal in het beeld van C_2 een afbeelding zijn van een ander punt dat deel uitmaakt van het inputobject. Meer bepaald dat punt waar de lichtstraal vanuit de lichtbron zal kaatsen volgens een speculaire reflectierichting die op het projectievlak van camera C_2 zal invallen. Glanzende objecten hebben een iets hogere kans dat intensiteiten tussen punten die de afbeelding zijn van eenzelfde punt uit de ruimte, gelijkaardig zullen zijn. Bij pure diffuse objecten zal een punt afgebeeld in verschillende beelden steeds dezelfde intensiteitswaarden hebben omdat diffuse projectiestralen in alle richtingen worden weerkaatst en zo alle diffuse lichtstralen op alle camerastandpunten over de hemisfeer worden afgebeeld.



(a) Verschillende oppervlakken met hun reflecties.



(b) Speculaire reflectie omringt in het blauw op schedelafbeelding vanuit een camerastandpunt C_1.

(c) Afbeelding vanuit nieuw camerastandpunt C_2 met een verschillende speculaire reflectie ten op zichte van C_1 en verschillende en andere intensiteitswaarden in blauwe regio.

Figuur 4.7: Vergelijking van reflecties bij verschillende oppervlakken. Intensiteitswaarden zijn niet steeds hetzelfde over alle afbeeldingen van een gegeven punt P.

- Efficiënte algoritmen

  Voor de berekening van de visual hull zijn verschillende real-time algoritmen beschikbaar: [Li [59]], [Matusik et al. [70]], [Matusik et al. [71]], [Wu et al. [106]].

- Veel toepassingen

  Silhouetten vormen een sterke visual cue [Koenderink [45]]. Dit is een visuele aanwijzing die toelaat om objecten te herkennen en te onderscheiden. Denk maar aan een toneelvoorstelling waarbij silhouetten van acteurs achter een wit doek op het doek geprojecteerd worden. Hierbij zijn verschillende objecten nog vaak goed te onderscheiden hetgeen visual hulls naast 3D reconstructie ook bruikbaar maakt voor objectherkenning [Laurentini [54]]. Wanneer de visual hulls VH(O) van 2 objecten van elkaar verschillen moeten de objecten van elkaar verschillen vermits $O \subseteq VH(O)$. Deze eigenschap maakt het nuttig om de visual hull techniek ook te combineren met andere reconstructietechnieken omdat zoekregio's voor bijvoorbeeld stereo matching [Scharstein and Szeliski [92]] beperkt kunnen worden tot die pixels die binnen de visual hull projecteren. Alle andere pixels kunnen tijdens het correspondentieprobleem genegeerd worden. Andere toepassingen naast viewsynthese en objectherkenning zijn shape acquisition [Cheung [21]] en collision detection [de Decker et al. [26]].

4.5.2 Nadelen

- Approximatie

  Een eerste notie van benadering treedt op aangezien de visual hull techniek inherent beperkt is in kracht voor objecten met niet-silouette-actieve oppervlakken. De visual hull techniek kan in theorie objecten die enkel bestaan uit silhouette-actieve oppervlakken met een precisie naar keuze reconstrueren. Een tweede vorm van approximatie hangt samen met de hoeveelheid verschilllende silhouette-informatie beschikbaar is. Zoveel mogelijk verschillende vorminformatie over het object is gewenst. Het aantal silhouette-afbeeldingen nodig hangt af van de geometrische complexiteit van het te reconstrueren object zoals het aantal vlakken en of deze al dan niet gekromd zijn [Laurentini [53]]. Wanneer slechts weinig verschillende camerastandpunten gebruikt worden is er weinig vorminformatie beschikbaar en zal de visual hull vaak een zeer

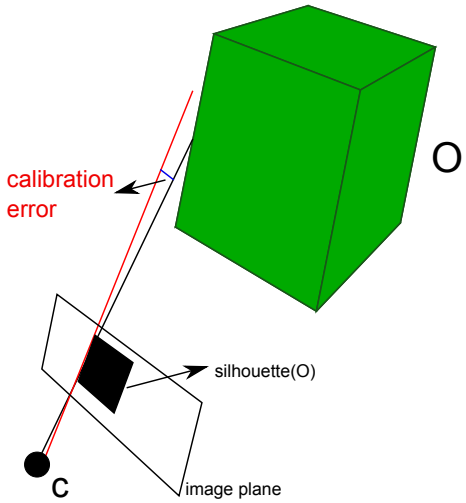
grove benadering zijn van het te reconstrueren object. Dit kan in het spatiaal domein opgelost worden door meer camera's in de ruimte te plaatsen en zo meer vorminformatie te verkrijgen of in het tijdsdomein door meer vorminformatie te extraheren uit verschillende poses van het object doorheen de tijd [Cheung [21]]. Om extra silhouette-informatie doorheen de tijd te verkrijgen, is het wel nodig dat ofwel het object beweegt ofwel de camera's bewegen. Een derde vorm van benadering kan ontstaan door geometrische fouten in kalibratie en segmentatie. Fouten kunnen ontstaan door beeldruis en over- en ondersegmentatie door schaduwen, speculaire reflecties en moeilijke te onderscheiden regio's, bijvoorbeeld bij doorzichtige objecten zoals afgebeeld in figuren 4.8 en 4.9.

- Artefacten

  Tijdens visual hull reconstructie kunnen ghost artefacten optreden [Center and Wexler [15]]. Dit kan zich voordoen wanneer zeer weinig camera's gebruikt worden waarbij afzonderlijke regio's kunnen ontstaan die geen objectinformatie omhullen. In [Bogomjakov and Gotsman [6]] worden deze artefacten tegengegaan door een kleine hoeveelheid diepteinformatie in rekening te brengen.

(a) Doorschijnende sculptuur van een paard waarvan transparantie van rode regio uitvergroot is afgebeeld [Campbell et al. [13]].

(b) Segmentatie aangeduid in het wit van een paard. Segmentatie-errors treden op in rode doorschijnende regio[Campbell et al. [13]].



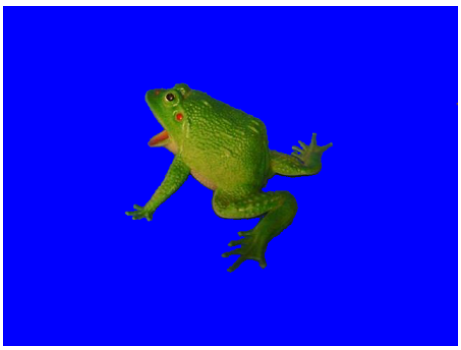
(c) Illustratie van calibratie-error in cameramatrices die een reprojectie-error tot gevolg heeft.

Figuur 4.8: Geometrische errors in calibratie en segmentatie-errors bij doorschijnende regio's.

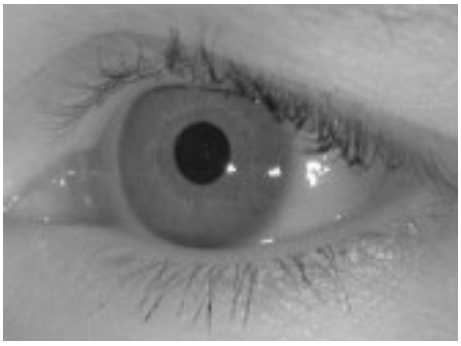
(a) Kikker met schaduw.

(b) Segmentatie van kikker met schaduw. Achtergrond in het blauw. Segmentatie-errors in schaduwregio.

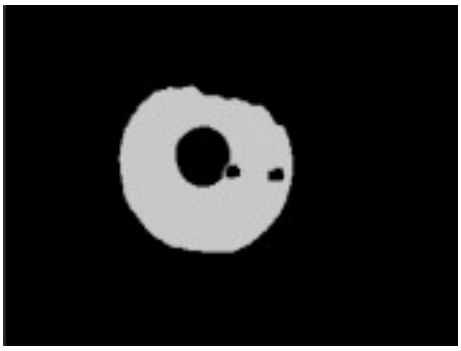
(c) Kikker zonder schaduw.



(d) Correcte segmentatie wanneer er geen schaduw aanwezig is.

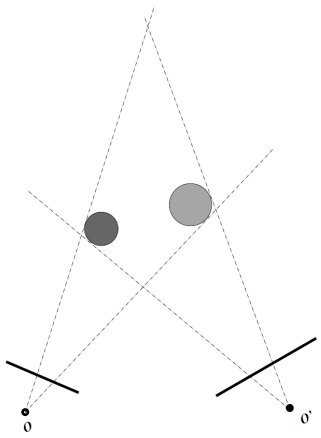


(e) Oog met 2 opvallende speculaire reflecties.

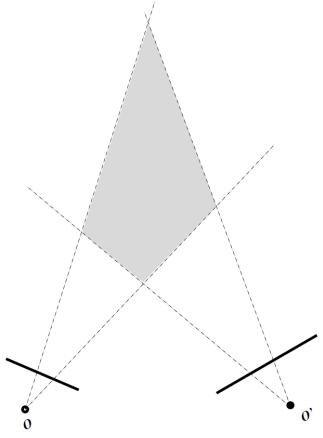


(f) Segmentatie van iris-regio aangeduid in het wit met holtes op posities van speculaire reflecties.

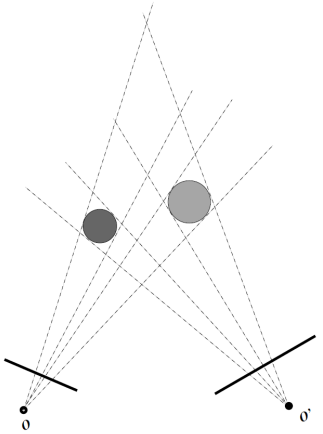
Figuur 4.9: Segmentatie-errors door schaduwen en speculaire reflecties. Afbeeldingen a-d [Chandrand [16]], e-f [Pundlik et al. [86]].



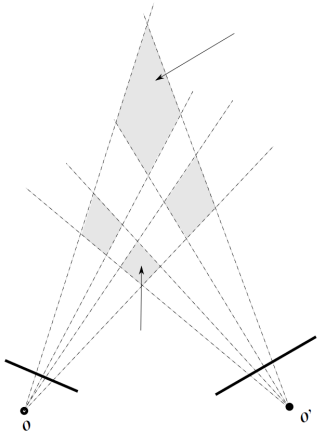
(a) Convex hull reconstructie van object O dat bestaat uit 2 cirkels. De intersectie van de kijkvlakken die O volledig bevatten, bepalen CH(O).



(b) CH(O)-benadering t.o.v. 2 views



(c) Visual hull reconstructie.



(d) VH(O)-benadering t.o.v 2 views waarbij ghost artefacten met pijlen zijn aangeduid.

(e) Voorbeeld met ghost artefacten achter de gereconstrueerde objecten.

Figuur 4.10: Illustratie van ghost-artefacten bij visual hull reconstructie. Afbeeldingen a-d [Center and Wexler [15]], e [Bogomjakov and Gotsman [6]].

4.6 Representaties

In deze sectie worden visual hull algoritmen overlopen op basis van hun onderliggende visual hull representaties waarin de meeste algoritmen in kunnen worden onderverdeeld. In het bijzonder ligt de nadruk op de voor- en nadelen die met deze representaties verbonden zijn en ook de hierop steunende algoritmen al dan niet voordelig beïnvloeden. Er wordt een onderscheid gemaakt in volgende 4 representaties:

- Volumegebaseerde visual hull
- Oppervlaktegebaseerde visual hull
- Beeldgebaseerde visual hull
- Hybride representatie van visual hull

4.6.1 Volumegebaseerde visual hull

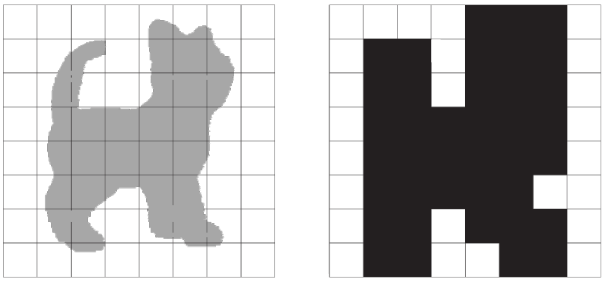
Volumegebaseerde visual hull reconstructie tracht het visual hull volume te reconstrueren. Typisch wordt er als initialisatiestap een volume gebruikt waarin het te reconstrueren object zeker bevat zit zoals de scene waarin het te reconstrueren object zich bevindt [Dyer [28]]. Dit volume wordt discreet onderverdeeld in een grid van voxels. Een voxel is de afkorting voor volume pixel en is een kubus, het 3d-equivalent van een pixel. De visual hull wordt gereconstrueerd door de voxels die deel uitmaken van de visual hull onaangetast te laten en diegene die niet deel uitmaken van de visual hull te verwijderen. Er kan een onderscheid gemaakt worden in de aanpak voor het wegsnijden van voxels:

- Benutten van beste-approximatie eigenschap (jaren 80)

  De visual hull is de maximale doorsnede van alle silhouette reprojectievolumes zoals afgebeeld in figuur 4.1. Zodra voxels tijdens een silhouettereprojectie niet bevat zijn in het reprojectievolume worden ze weggesneden. Incrementeel per silhouette afbeelding wordt de visual hull zo almaar meer benaderd [Eisert [29]] zodat enkel de voxels in de doorsnede van alle reprojectievolumes niet worden verwijderd. Het aantal intersectieberekeningen is groot en hangt af van de scene complexiteit en het aantal silhouette-afbeeldingen. Dit maakt deze methode minder geschikt wanneer efficiëntie vereist is.

- Benutten van silhouette-consistentie eigenschap

  Overloop alle voxels en doe voor iedere voxel een consistentietest. Tijdens deze test wordt een voxel geprojecteerd in elke silhouette-afbeelding. Indien de 2D-projectie buiten een silhouette valt wordt de voxel weggesneden bijvoorbeeld door ze transparant te maken [Li [59]]. Dit proces wordt geïllustreerd in afbeelding 4.12. De efficiëntie van silhouette-consistentiemethoden hangt af van de resolutie van het startvolume en niet van de scenecomplexiteit. De toevoeging van extra silhouette-afbeeldingen veroorzaakt ook minder overhead in efficiëntie dan de volume-intersecties uit de beste-approximatiemethoden.
  De consistentietest per voxel in een 3D gridstructuur kan volgens [Niem and Buschmann [81]] vereenvoudigd worden tot een consistentietest per voxelkolom waarbij een projectie van de voxelkolom op de afbeelding volledig bepaald is door de onderste en de bovenste voxel van die kolom.
  De projectie van een voxel kan resulteren in een bounding box, een convex hull afgelijnd door de perspectiefprojectie van de 8 voxelvertices op het projectievlak, die niet exact overeenkomt met de grootte van een pixel zoals geïllustreerd in figuur 4.13.
  In [Ladikos et al. [51]] wordt dit probleem bij benadering opgelost door de gegeven silhouette-afbeelding te super- of te downsamplen zodat de geprojecteerde voxelgrootte overeenkomt met een pixelgrootte bij de silhouette-consistentietest.
  In [Cheung et al. [22]] wordt een voxel als consistent gelabeld indien ze in iedere afbeelding voldoet aan de sparse pixel occupancy test (SPOT). Deze voorwaarde, houdt in dat een voxel silhouette-consistent is ten op zichte van een afbeelding zodra een minimum aantal pixels uit een uniform gedistribueerde deelverzameling van pixels binnen de convex hull van de geprojecteerde voxel in het silhouette bevat zijn. Via de grootte van de deelverzameling van pixels en de striktheid van SPOT-conditie, bepaald door het aantal pixels uit de deelverzameling die aan de conditie moeten voldoen, kan de fout voor valse voxelindelingen gecontroleerd worden. De posities van de verzameling pixels gekozen binnen een voxelprojectieregio kunnen in een lookup table met de desbetreffende voxel geassocieerd worden in een preprocessingfase tenminste wanneer noch de camera's noch de reconstructiegrid verplaatst worden [Ladikos et al. [51]]. De scene zelf mag wel dynamisch zijn. In een postprocessingstap kunnen alle voxels die geen deel uitmaken van het objectoppervlak worden verwijderd voor een efficiëntere rendering. Zulke voxels kunnen geïdentificeerd worden door te kijken of er tenminste 1 van zijn



Figuur 4.11: Illustratie van quantisatie-artefacten aan visual hull randen. Bron afbeelding: [Milne et al. [78]].

6 buren verwijderd is. Zoja maakt de voxel deel uit van het visual hull-oppervlak.

Het voordeel van volumetrische reconstructiemethoden is dat ze op eenvoudige en stabiele algoritmen steunen die bovendien makkelijk parallelliseerbaar zijn op de GPU [Ladikos et al. [51]] en toelaten complexe objecten zoals bomen en haren voor te stellen en te reconstrueren [Li [59]]. Een groot nadeel volumetrische reconstructiemethoden zijn de discretisatie-artefacten die ontstaan door discretisatie van het volume door voxels:

- aliasing

  Aliasing kan ontstaan door onderbemonsterde volumeresoluties [Esteban and Schmitt [31]] waardoor een voxel kan instaan voor meerdere signalen/pixels uit het beeld. Informatie verspreid over meerdere pixels wordt zo vermengd onder de representatie van 1 voxel.

- quantisatie
  Quantisatie kan zich aan de visual hull randen voordoen doordat het visual hull volume soms slechts gedeeltelijk bevat zit in de voxel. Dit wordt weergegeven in afbeelding 4.11.

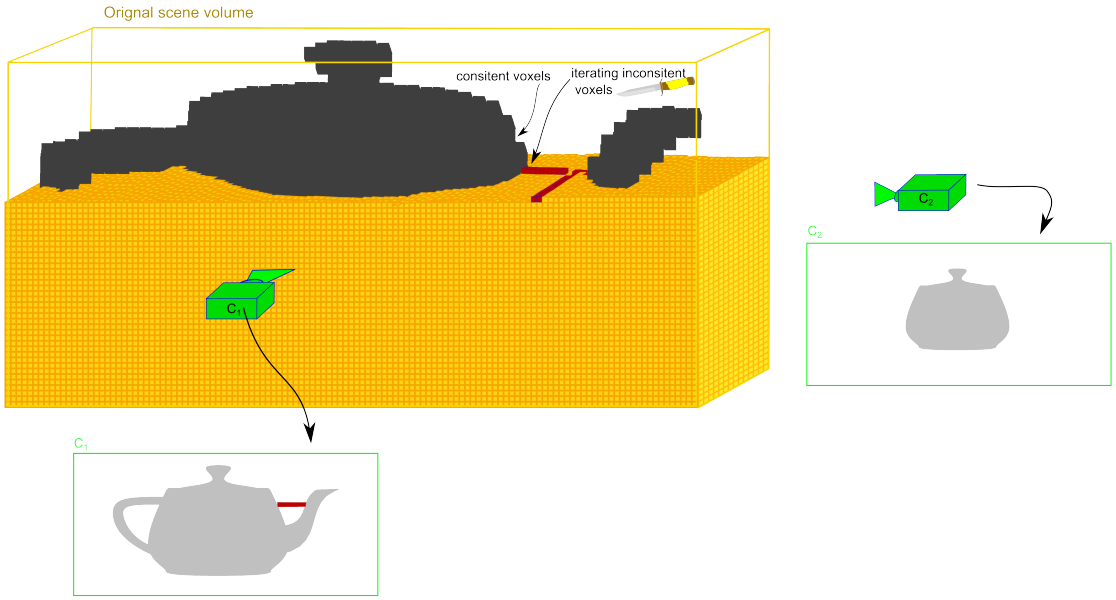
Aliasing en quantisatie kunnen worden beperkt door een grotere volumeresolutie zodat een minder groot volume in een voxel gekwantiseerd wordt en verschillende pixels niet door 1 voxel gerepresenteerd worden. De precisie van de reconstructie kan naar keuze worden bepaald. Hoge kwaliteit gaat echter ten koste van snelheid en geheugengebruik.

Optimalisaties Efficiëntie in uitvoering en opslag kan enigszins worden verhoogd wanneer er grote samenhorige regio's zijn in het object en hierbij de

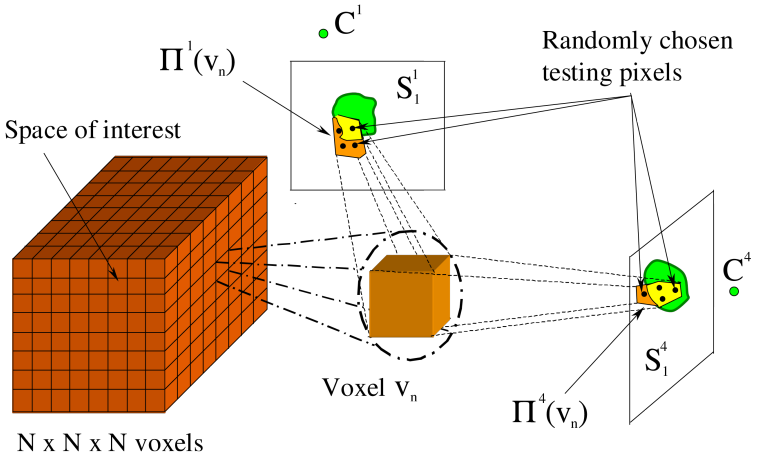
voxelgrootte te laten variëren naargelang de regiogrootte. Hiërarchische volumedecompositie zoals octrees [Szeliski [101]] afgebeeld in figuur 4.14 benutten deze eigenschap. Een octree is een boomdatastructuur die vertakt naar 8 kinderen in de overgang naar een volgend niveau in de boom; een binaire vertakking voor iedere dimensie geeft $8 = (2^3)$ vertakkingen. Octree's zijn dus de 3D-equivalenten van binaire bomen. Tijdens de octree-reconstructie wordt iedere knoop die een voxel met welbepaalde grootte voorstelt gekarakteriseerd als zijnde:

- vol: een substantie van VH(O,V) zit volledig in de voxel bevat. Volle knopen worden niet meer opgesplitst.
- leeg: de voxel bevindt zich volledig buiten VH(O,V). Lege knopen worden niet meer opgesplitst.
- gedeeltelijk vol: de voxel bevat een gedeelte van VH(O,V) maar is niet volledig gevuld. Gedeeltelijke volle knoppen worden recursief verder opgesplitst totdat alle suboctanten vol of leeg zijn of als een vooraf gedefinieerde maximumdiepte bereikt is.

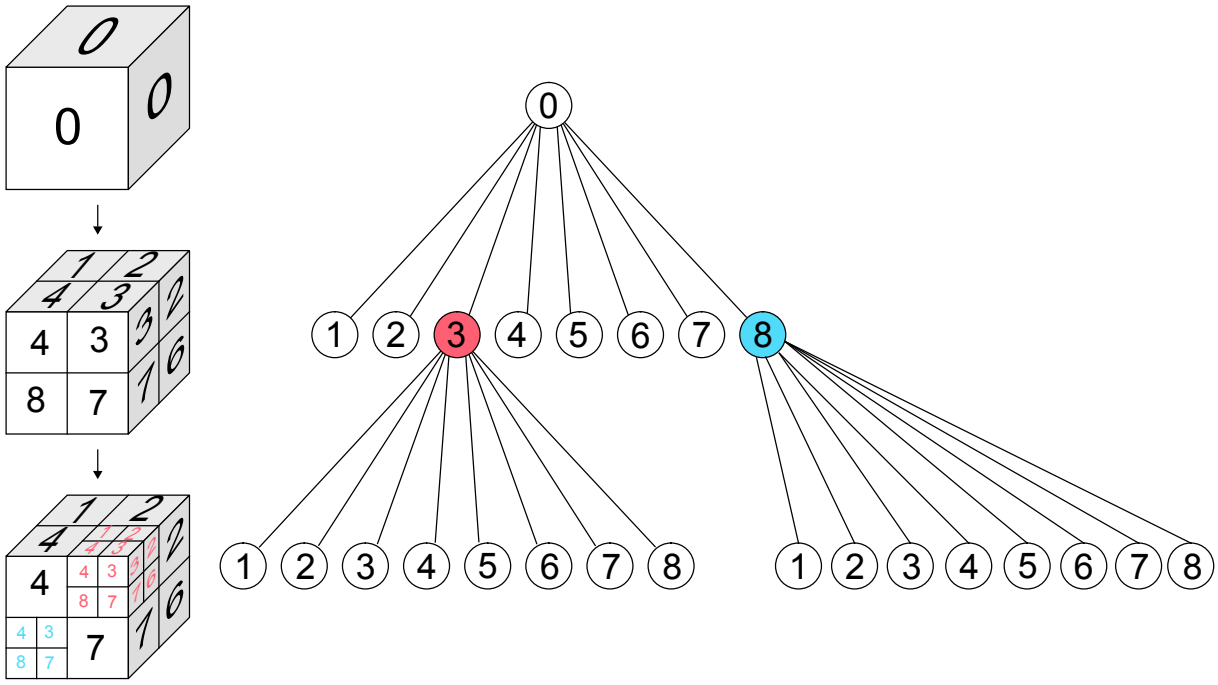
Initieel wordt de root van een octree geïnitialiseerd als een voxel zo groot als de scene. Het verdere verloop van de octree-reconstructie wordt bepaald door de voxelkarakteristieken. Dit proces is afgebeeld in figuur 4.14. Door de bladeren in de octree incrementeel verder te vertakken in kleinere voxels kan de reconstructie progressief verfijnd worden.



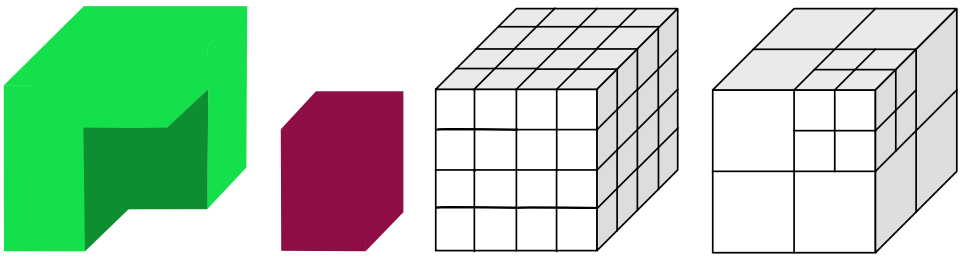
Figuur 4.12: Illustratie van voxel carving. De te onderzoeken voxels die als volgende in de iteratie aan de beurt zijn, zijn in het rood aangeduid. Sommige zijn consistent met de silhouette-afbeelding in C_2, maar allen zijn niet consistent vanuit C_1 en mogen dus verwijderd worden. De grijze voxels zijn wel consistent met beide afbeeldingen en maken dus deel uit van de VH(O,V) met V = $\{C_1, C_2\}$.



Figuur 4.13: De convex hull van de perspectiefprojectie van een voxel op 2 afbeeldingen is in het zwart omlijnd met hierin zwarte stippen die de random gekozen pixels in deze regio voorstellen. De groene regio op iedere silhouette-afbeelding stelt het silhouette van een gezicht voor. De gele regio bevat de pixels die in het silhouette bevat zijn. De oranje regio bevat de pixels die buiten het silhouette vallen. De verhouding van het aantal pixels in de gele regio tot het aantal pixels in de oranje regio bepaalt of er al dan niet aan de SPOT-conditie voldaan is [Cheung [21]].

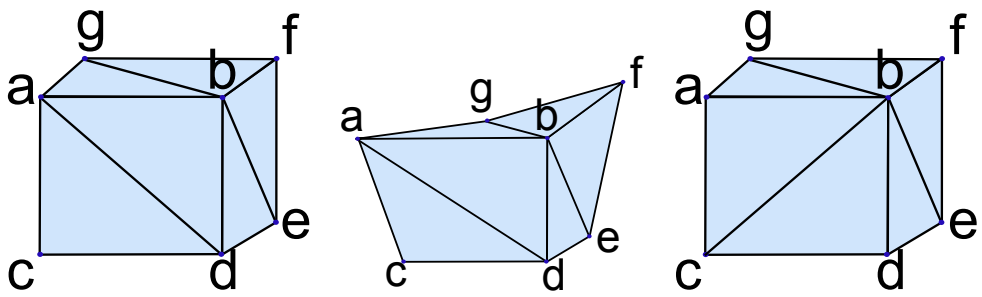


Figuur 4.14: Illustratie van octreesubdivisie toegepast op voxels.



(a) Object O. (b) Scene - O. (c) Aantal voxels benodigd bij gridreconstructie. (d) Aantal voxels benodigd bij octree reconstructie.

Figuur 4.15: Illustratie van het efficiënter opslaan bij gebruik van octrees. Voor reconstructie van object O zijn er in het geval van een reguliere grid 32 voxels nodig waarvan er 20 consistent zijn met O. Wanneer octrees gebruikt worden zijn er slechts 15 voxels nodig waarvan er 11 deel consistent zijn met O.



(a) Gegeven topologie en geometrie van object O.

(b) Topologisch equivalent met O, maar geometrisch niet equivalent.

(c) Geometrisch equivalent met O, maar topologisch niet equivalent met 4.16(a) wegens verandering van edge AD in BC.

Figuur 4.16: Illustratie van het verschil tussen topologie en geometrie.

4.6.2 Oppervlaktegebaseerde visual hull

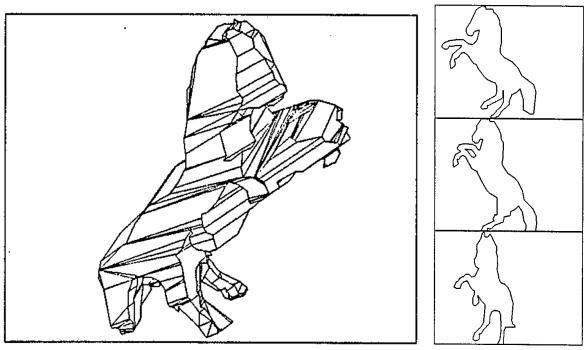
Oppervlaktegebaseerde visual hull technieken proberen enkel het buitenste oppervlak van VH(O,V) te reconstrueren. De motivatie hierachter is dat enkel het oppervlak visueel zichtbaar is na rendering en de volume-inhoud binnen dit oppervlak berekeningsoverhead is aangezien deze toch visueel verborgen is. Om de oppervlakte van de visual hull te berekenen hoeven enkel de silhouetteranden in iedere afbeelding onderzocht te worden vermits zij de buitenste grens van de maximale doorsnede van silhouetterereprojecties en dus het oppervlak van VH(O,V) bepalen. Een pixel maakt deel uit van de silhouetterand wanneer er 1 of meer van zijn 8 buurpixels tot de achtergrond behoren. De correctheid van het gereconstrueerde oppervlak kan beoordeeld worden naargelang:

- geometrie

  De geometrie beschrijft de positionele informatie van de mesh: waar in de ruimte een oppervlak aanwezig is. Twee meshes zijn geometrisch equivalent als de ene mesh via een rigide beweging, dit is een translatie of rotatie uit de andere kan worden bekomen.

- topologie

  De topologie beschrijft de connectiviteitsinformatie van de mesh: welke vertices of nodes met elkaar verbonden zijn. Twee meshes zijn topologisch equivalent als er een homeomorphische transformatie tussen hen bestaat [Zeeman [111]]. Een homeomorphische transformatie is een



Figuur 4.17: Oppervlaktereconstructie van VH(O,V) uit volume-intersecties van gereprojecteerde contouren. In de 3 kleinere silhouette-afbeeldingen zijn de contouren gehighlighted. De polyhedron in de grootste figuur is het resultaat na reconstructie van het VH(O,V)-oppervlak bekeken vanuit een nieuw camerastandpunt tegenovergesteld aan de camera's die de 3 gegeven silhouette-afbeeldingen vastlegden.

continue bijectieve (1 op 1 mapping) functie. Dit wil dus zeggen dat er een vervorming, een combinatie van uitrekkingen en inkrimpingen van de ene mesh in de andere mesh bestaat zonder dat hierbij snij-operaties (veroorzaken discontinuïteiten) en lijmoperaties (veroorzaken een niet-bijectie) aan te pas komen.

De meeste oppervlaktegebaseerde algoritmen die we zullen bespreken, onderscheiden zich in de manier waarop uit contourinformatie een oppervlakterepresentatie wordt afgeleid en focussen zich op de geometrische correctheid. Sommige recentere algoritmen [Lazebnik et al. [57]],[Franco and Boyer [32]] trachten ook de topologie van VH(O,V) te reconstrueren mits enkele veronderstellingen zoals smoothness van het te reconstrueren object.

3D intersectiemethoden

In [Baumgart [5]] wordt de geometrie van het oppervlak van VH(O,V) benaderd door iedere contour, die op zich een 2D polygon vormt in de silhouetteafbeelding, te reprojecteren in de ruimte. De intersecties van al deze reprojectievolumes vormt een polyhedron die het oppervlak van VH(O,V) zal benaderen zoals afgebeeld in figuur 4.17.

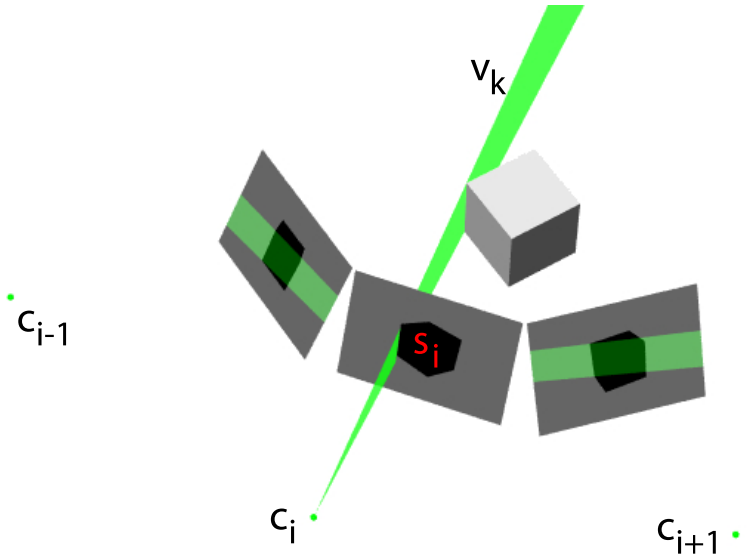
2D intersectiemethoden

Intersecties in 3D kunnen gereduceerd worden tot simpelere intersecties in 2D omdat het silhouettereprojectievolume geen willekeurige polyhedron in een Euclidische ruimte voorstelt, maar een polyhedron met een projectieve

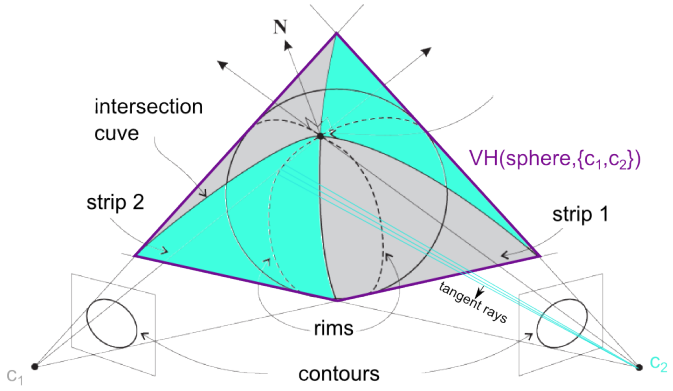
structuur. De structuur van een silhouettereprojectievolume hangt enkel af van het object en de positie cameraprojectiecentrum en niet van de positionering van het beeldvlak vermits een silhouette-afbeelding de doorsnede van het beeldvlak met het reprojectievolume voorstelt waarbij ieder punt in de doorsnede op dit vlak dezelfde vaste schalering heeft ondergaan. Zo zal de informatie op beeldvlakken met een verschillende afstand van eenzelfde projectiecentrum slechts een schaalfactor van elkaar verschillen en zal de positie van het beeldvlak de vorm van het silhouetereprojectievolume niet beïnvloeden. Silhouettecontourinformatie vanuit een gegeven afbeelding I_1 kan met de silhouettecontour uit I_2 worden geïntersecteerd door alle reprojectierays van het silhouette in I_1 te projecteren op I_2 met behulp van de projectiematrix P_2. Deze projecties van rays vanuit C_1 door I_1 op het beeldvlak I_2 worden epipolaire lijnen genoemd. Voor meer informatie en illustraties zie bijlage B betreffende epipolaire geometrie. In [Matusik et al. [71]] wordt in realtime de geometrie van een polyhedrische visual hull berekend. De filosofie achter de werkwijze is dat vlakken van de resulterende visual hull polyhedron wel moeten overeenkomen met een vlak van een van de oorspronkelijke silhouette-reprojectievolumes. Voor iedere silhouette-afbeelding S_i, met i-de index, worden de contouren met een polygoon benaderd en voor iedere rand e_k van de polygon wordt het vlak v_k van het reprojectievolume dat deze rand in 2D encodeert, gemapt in alle andere silhouetten S_j met i $\neq$ j en vervolgens geïntersecteerd met deze silhouetten zoals afgebeeld in figuur 4.18, hetgeen eigenlijk de intersecie van een reprojectievolumeface met alle andere reprojectievolumes voorstelt. De resultaten van deze intersecties F_j definiëren op hun beurt opnieuw een verzameling van polygonen die wanneer ze geherprojecteerd worden volgens iedere P_j^{-1} op het vlak v_k van silhouette S_i dit vlak zullen begrenzen met polygonen Q_j. De hergeprojecteerde polygonen Q_j dienen nog met elkaar geïntersecteerd te worden om het visual hull oppervlak voor vlak v_k te reconstrueren. Na over alle k's in een silhouette-afbeelding S_i te itereren zal het reprojectievolume R_i van S_i met iedere andere silhouetteafbeelding gesneden zijn en zullen die stukjes van vlakken van R_i, die van het oppervlak VH(O,V) deel uitmaken, overblijven. Wanneer dit herhaald wordt voor iedere i zal het totale oppervlak van VH(O,V) gereconstrueerd worden. Intersecties worden in [Matusik et al. [71]] op een efficiënte manier berekend door gebruik te maken van een edge-bin datastructuur.

Topologiegebaseerde methoden

In [Lazebnik et al. [56]] wordt de geometrie en de topologie van de visual hull mesh gereconstrueerd onder de aanname dat contouren niet degeneren, zijden niet uit vlakken bestaan en het object smooth is met genus 0, dit wil zeggen



Figuur 4.18: Illustratie van intersecties van epipolaire lijnen van vlak v_k met andere silhouetten [Matusik et al. [71]].



Figuur 4.19: Illustratie van de verschillende topologische features volgens [Lazebnik et al. [56]] voor 2 camerastandpunten. De rim-mesh bestaat uit de mesh beschreven door de verschillende rims, aangeduid in stippellijnen. De visual hull mesh is de verzameling van de verschillende strips afgebakend door intersectiecurves en de oppervlakte valt in de illustratie niet samen met het objectoppervlak.

dat het object uit 1 geheel bestaat zonder gaten begrensd door een gesloten curve (= genus 0) waarbij men door het object kan kijken. Een voorbeeld van een smooth object van genus 0 is een vaas terwijl een ring een object van genus 1 is. Eens de geometrische posities van de vertices, edges en faces die de topologie definiëren berekend zijn, hetgeen volgens [Lazebnik et al. [56]] overigens in 2D kan worden gedaan met de hulp van epipolaire geometrie en ordeningsconstraints, kan het visual hull oppervlak worden gereconstrueerd. De grote bijdrage van dit onderzoek is dat er een topologische structuur voor de connectiviteit van de visual hull van een smooth object wordt voorgesteld die ons een idee geeft over de resulterende mesh structuur en de punten waarvoor er een geometrie moet worden berekend om het visual hull oppervlak te reconstrueren. In [Lazebnik et al. [56]] worden 2 topologische beschrijvingen voorgesteld: de rim mesh en de visual hull mesh. De reden hiervoor is dat de topologische beschrijving van de rim mesh bijkomende constraints kan opleggen op de zoekregio's voor punten die kunnen deel uitmaken van de topoplogie van de visual hull. We overlopen kort de verschillende topologische structuren aangezien sommige hiervan ook bij andere algoritmen gebruikt worden in de samenstelling van een oppervlakterepresentatie.

Eerst wordt een topologische beschrijving van de contourgeneratoren, de rim-mesh afgeleid. De contourgenerator of rim ten op zichte van een camerastandpunt is de verzameling punten op het objectoppervlak die strikt geraakt worden door de visuele rays vertrekkend vanuit het camerastandpunt door de contour in de silhouette-afbeelding. De contour is dus de projectie-afbeelding van de contourgenerator en is onder de gestelde aannames steeds een smooth space curve zonder excepties [Cipolla and Giblin [23]]. De rim-mesh is topologisch als volgt gedefinieerd:

- vertices: frontier punten. Frontier punten zijn de punten op het oppervlak waar 2 smooth rims elkaar snijden en waar rays die beide het oppervlak in het frontier punt raken elkaar zullen snijden. Het raken van de 2 reprojectievolumes moet hierbij gedefinieerd zijn, vandaar dat object smoothness, hetgeen stelt dat er een raakvlak in ieder punt bestaat, een vereiste is in de aanname.

- edges: rimsegmenten. Een oppervlakterand wordt beschreven door een rimsegment, de rim tussen 2 opeenvolgende frontier points.

- faces: regio's begrensd door 2 of meer edges.

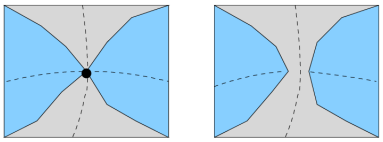
De rim mesh is afgebeeld in figuur 4.19 en bewijsmateriaal omtrent de verschillende topologiën kan worden teruggevonden in [Lazebnik [55]]. De visual hull mesh, afgebeeld in figuren 4.19 en 4.20(a), heeft onder de gestelde aannames de volgende topologie[Lazebnik et al. [56]]:

- vertices: frontier en intersectiepunten. Frontier punten zijn net als bij de rim-mesh de punten op het oppervlak waar 2 rims elkaar snijden. Intersectiepunten, ook wel triple punten genoemd, zijn de punten waar 3 silhouette-reprojectievolumes elkaar snijden.

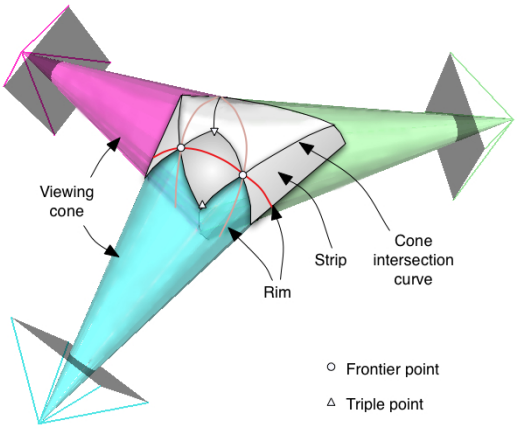
- edges: intersectiecurvesegmenten tussen 2 opeenvolgende vertices. Intersectiecurvesegmenten zijn gedefinieerd als de intersecties tussen 2 reprojectievolumes.

- faces: strips. Een strip is een patch, een door een curve beschreven oppervlak, van het reprojectievolume die alle rays bevat die door de contour gaan en het object oppervlak zullen raken. Een strip is begrensd door intersectiecurves en hoeft dus niet steeds samen te vallen met het oppervlak van het object. Frontier punten op een rim splitsen de reprojectiestrip op dezelfde rim op in aparte faces.

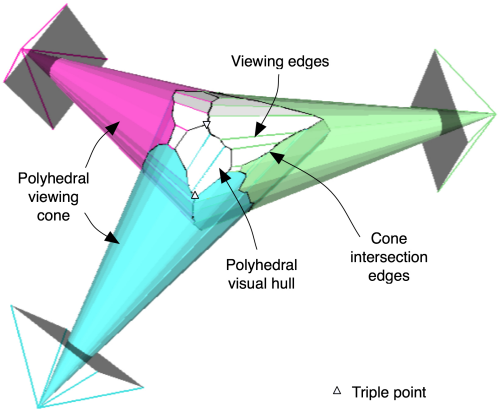
Het voordeel van deze techniek is dat de visual hull gerepresenteerd wordt op basis van zijn intrinsieke projectieve features, dit zijn features die na projectie features blijven zoals frontier punten, intersectiepunten en intersectiecurvesegmenten, in plaats van polygonen en voxels die discretisatie-artefacten kunnen veroorzaken. De projectieve features laten toe om de geometrische berekeningen naar 2D te reduceren en gebruik te maken van multiview geometrie en de contacteigenschappen van de features om ze te identificeren. Voor implementatiedetails zie [Lazebnik et al. [56]]. Na het identificatieproces kan een representatie naar keuze gebruikt worden: beeldgebaseerd via mapping naar een nieuw beeld, of geometriegebaseerd via triangulatie van de topologische features. Een nadeel van de techniek is dat enkele topologische features bijvoorbeeld het kruisen van intersectiecurves in frontier punten in reële data niet steeds robuust te detecteren is. Een efficiëntere variant op het algoritme van [Lazebnik et al. [56]] is terug te vinden in [Lazebnik et al. [57]]. Verschillen met de vorige aanpak zijn dat er geen rim mesh wordt gereconstrueerd, enkel van georiënteerde projectieve geometrie wordt gebruik gemaakt om de visual hull topologie incrementeel te berekenen en dat contouren met grote dichtheid gediscritiseerd mogen worden in plaats van een continue beschrijving zoals kubische B-splines in [Lazebnik et al. [56]]. Voor niet-smooth objecten is de voorgestelde topologie van [Lazebnik et al. [56]] niet steeds reconstrueerbaar omdat frontier punten soms afwezig zijn zoals afgebeeld in figuur 4.20(b). Hierdoor gaan de constraints die een strip afbakenen verloren.
In [Franco and Boyer [32]] wordt er een algemenere topologie op de visual hull gedefinieerd:



(a) Invloed van onnauwkeurigheden op frontier punten en de connectiviteit van strips.



(b) Topologie beschreven door [Lazebnik et al. [56]] voor smooth objecten in een ideaal scenario zonder errors. Merk op dat voor 3 camerastandpunten de strips meer de bol benaderen dan voor de 2 camerastandpunten in figuur 4.19 Er treden nu ook intersectiepunten op. In de illustratie zijn, voor een goed begrip, een rim en aantal vertices weggelaten opdat de figuur niet te onoverzichtelijk wordt.



(c) Visual hull topologie onder aanwezigheid van onnauwkeurigheden zoals ruis en calibratie-errors. Deze figuur beschrijft de robuustere topologie volgens [Franco and Boyer [32]] waarbij strips afgebakend zijn door discrete intersectie-curves.

Figuur 4.20: Illustratie van de invloed van ruis en andere onnauwkeurigheden op de topologie van de visual hull mesh van een bol.

- vertices: intersectiepunten en bounding edge punten. In tegenstelling tot frontier punten zijn intersectiepunten wel stabiel bij kleine calibratie-afwijkingen [Franco and Boyer [32]]. Bounding edge punten zijn het begin- en eindpunt van een bounding edge.

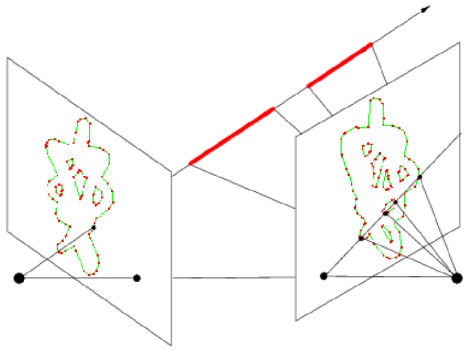
- edges: bounding edges, intersectie edges van contour-reprojectievolume. Bounding edges zijn de edges van VH(O,V) waarvan begin- en eindpunt op een contour projecteren. Ze bestaan uit intervallen op een gereprojecteerde viewing ray door een contourpunt die silhouette-consistent zijn met iedere afbeelding. Dit is geïllustreerd in afbeelding 4.21. Intersectie edges zijn stukjesgewijze lineaire curves die ontstaan na snijding van 2 contour-reprojectievolumes.

- faces: kunnen op een discrete strip gedetecteerd worden aan de hand van de 2D edges op de contouren die ieder een face in 3D representeren. Een discrete strip is een discreet oppervlak binnen een countour-reprojectievolume afgebakend door intersectie edges. Discrete strips ontstaan door discretisatie van randen, het minder smooth zijn van randen en onnauwkeurigheden waardoor strips niet steeds aan elkaar zullen sluiten zoals afgebeeld in figuur 4.20(b).

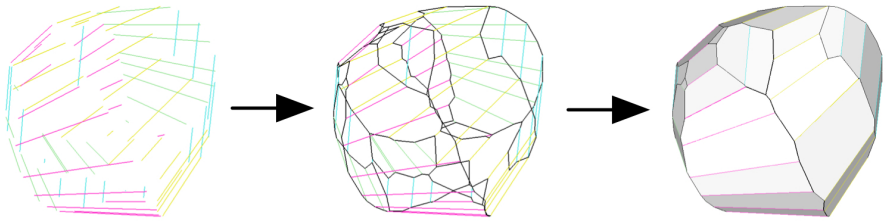
Deze topologie die afgebeeld is in figuur 4.20(c) laat toe om op een robuuste manier een waterdichte polyhedron representatie van VH(O,V) te berekenen. Het stappenplan van het algoritme van [Franco and Boyer [32]] om de topologie en de geometrie te reconstrueren, staat beschreven in afbeelding 4.21. Voor implementatiedetails zie [Franco and Boyer [32]].

Differentiële methoden

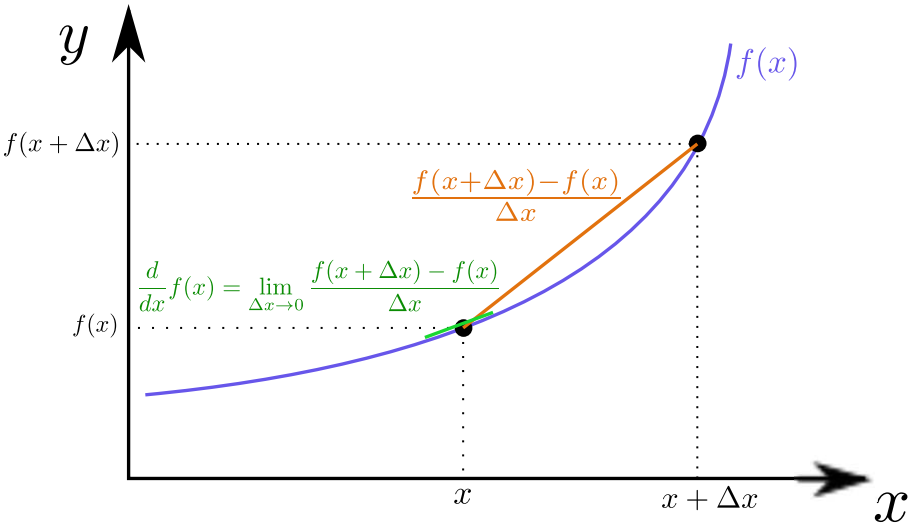
De afgeleide in een punt op een afleidbare 2D curve vertelt wat de helling van de curve is in dit punt zoals afgebeeld in figuur 4.22. Differentiële methoden [Brand et al. [10]],[Chen [18]] beschrijven het oppervlak van VH(O,V) in 3D voor smooth objecten beschreven als de enveloppe van berekende hellingen, i.e. in 3D raakvlakken, van punten die zich op het oppervlak van VH(O,V) bevinden. Des te meer bemonsterde punten des te meer raakvlakken en des te nauwkeuriger dit omhulsel het oppervlak van VH(O,V) zal benaderen. De rays vanuit aan camerastandpunt door de 2D contour van een silhouette-afbeelding zullen het oppervlak van VH(O,V) steeds in minstens 1 punt raken. Deze rays beschrijven lokale 3D raakvlakken in ieder punt op het oppervlak van VH(O,V). De waaier van raakvlakken zijn dus door de contour beschreven. Stel dat er geen gebruik wordt gemaakt van correspondenties en diepte-informatie. Het probleem dat zich voor differentiële



Illustratie van de berekening van viewing/bounding edge intervallen voor 2 camerastandpunten



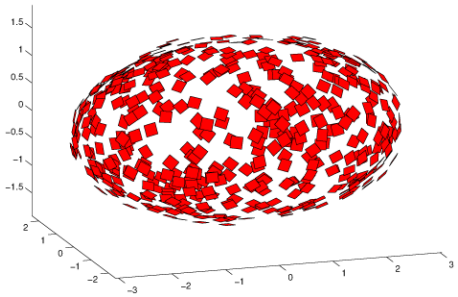
Figuur 4.21: Overzicht van de verschillende stappen voor de berekening van de visual hull volgens [Franco and Boyer [32]].



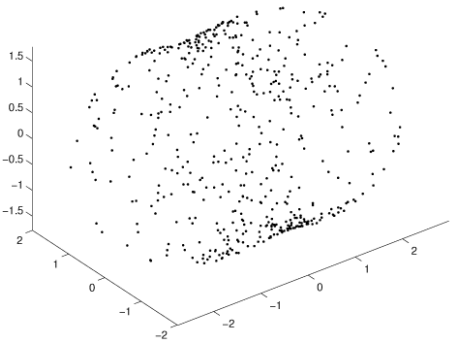
Figuur 4.22: Opeenvolgende differenties $\frac{f(x+\Delta x)-f(x)}{\Delta x}$ benaderen de helling van de curve $f(x)$ en zijn een exacte benadering van de helling in een punt op de curve wanneer $\Delta x \to 0$.

methoden dan stelt is welk raakvlak bij welke punten op het oppervlak van VH(O,V) horen vermits we de 3D posities van deze punten niet kennen en we niet weten welke punten over de verschillende views bekeken dicht bij elkaar liggen. Uit het bestuderen van continuïteiten tussen alle raakvlakken gerepresenteerd door alle contouren kan buurinformatie worden afgeleid door te kijken welke raakvlakken een gelijkaardige helling vertonen zodat er ingeschat kan worden welke contourpunten dicht bij andere contourpunten liggen. Dit probleem kan vereenvoudigd worden in duale ruimte waar raakvlakken gerepresenteerd worden door punten en waar continuïteit tussen de verschillende raakvlakken, beschreven door de kromming of 2de afgeleide in de primaire ruimte, in duale ruimte kan worden vereenvoudig tot een lineaire differentieberekening tussen punten. Deze differentieberekeningen waarbij gebruik kan gemaakt worden van epipolaire geometrie [Chen [18]] stellen op hun beurt tangentinformatie voor maar nu in duale ruimte. Snijpunten in de duale ruimte stellen dus raakvlakken voor die het object meerdere malen raken. Wanneer de nabijheden van de verschillende contouren in duale ruimte zijn afgeschat, kan het oppervlak van VH(O,V) worden berekend door de duale representatie van de tangentbasissen in de duale ruimte [Brand et al. [10]]. In praktijk is kromming informatie erg gevoelig voor meetfouten en ruis. In [Brand et al. [10]] worden er enkele robuustere werkwijzen voorgesteld die hiermee beter overweg kunnen. Het reconstructieproces is geïllustreerd in figuur 4.23.

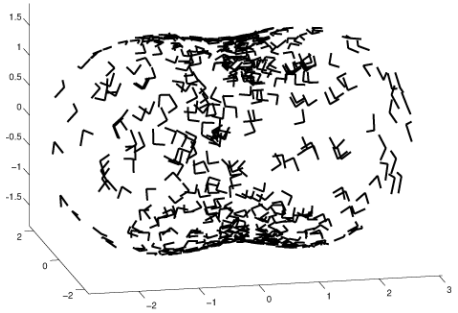
Wanneer het object zichzelf intersecteert kan dit niet door 1 raakvlak beschreven worden. Er zijn in dat geval soms meerdere raakvlakken die met dit intersectiepunt overeenkomen of er is geen oplossing. Dit probleem kan omzeild worden door voor singuliere punten een smooth pad te definiëren dat door deze singuliere punten gaat [Brand et al. [10]]. Voor deze smooth paden door singuliere punten zijn de raakvlakken wel gedefinieerd. Het nadeel ten opzichte van voorgaande technieken is dat voor benadering van oppervlakken meer bemonsteringen nodig zijn in regio's waar de kromming snel verandert. Experimenten van [Chen [18]] tonen aan dat meer afbeeldingen (ongeveer 15 uniform gedistribueerd) nodig zijn opdat deze methode een vergelijkbare accuraatheid bereikt in vergelijking met bijvoorbeeld [Franco and Boyer [32]]. Bij meer camerastandpunten wordt deze methode voor smooth objecten accurater dan [Franco and Boyer [32]] gegeven een B-spline voorstelling van de contouren waarbij subpixel accuraatheid mogelijk is [Chen [18]]. Differentiële methoden zijn eerder geschikt dan andere intersectiegebaseerde methoden wanneer een oppervlak uit beelden van een camera bewegend over een continu traject over de tijd moet worden gereconstrueerd [Lazebnik et al. [57]]. In dat geval zullen een groot aantal rays van reprojectievolumes over opeenvolgende tijdstippen bijna samenvallen en kunnen er numerieke insta-



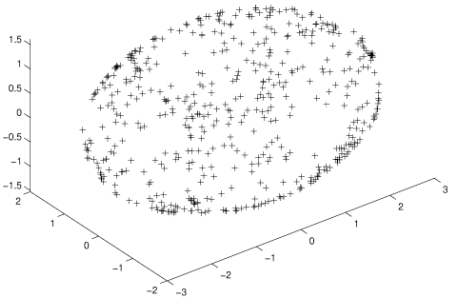
(a) Raakvlakken beschreven door contouren, hier aangeduid op het te reconstrueren ellipsoïde-object.



(b) Duale representatie van raakvlakken zijn punten in duale ruimte.

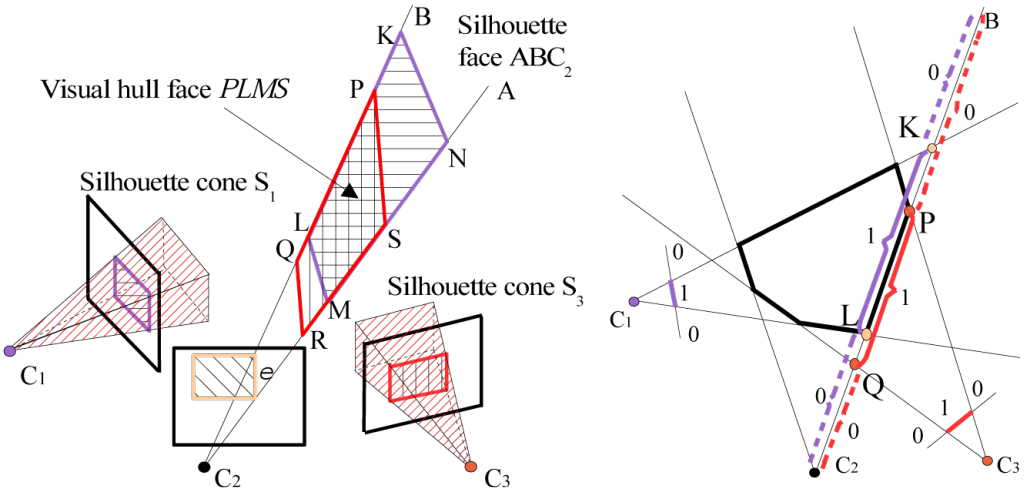


(c) Reconstrueren van tangent basissen door differentieberekeningen in duale ruimte. Deze basissen beschrijven een raakvlak in duale ruimte.



(d) Gereconstrueerde punten op het oppervlak van VH(O,V) van ellipsoïde door duale van raakvlakken in duale ruimte.

Figuur 4.23: Illustratie van de werkwijze van differentiële methoden, bron van afbeeldingen: [Brand et al. [10]].



Figuur 4.24: Illustratie van visual hull berekening op de gpu. De rechterfiguur stelt het bovenaanzicht voor van de linkerfiguur. De regio's KLMN en PQRS stellen de resultaten voor van de intersectie van face ABC$_2$ voor met de geprojecteerde textures van respectievelijk C_1 en C_3. De dubbelgearceerde regio PLMS is de resulterende polygon waarvoor alle alpha-waarden van geherprojecteerde silhouetten vanuit camerastandpunten C_1, C_3 tot 1 zullen vermenigvuldigen omdat polygon QLMR alpha-waarde 0 heeft voor camera C_1 en polygon PKNS alpha-waarde 0 heeft voor camera C_3. Bron afbeelding: [Li [59]].

biliteiten optreden hetgeen bij differentiële methoden wordt vermeden.

GPU-gebaseerde methoden

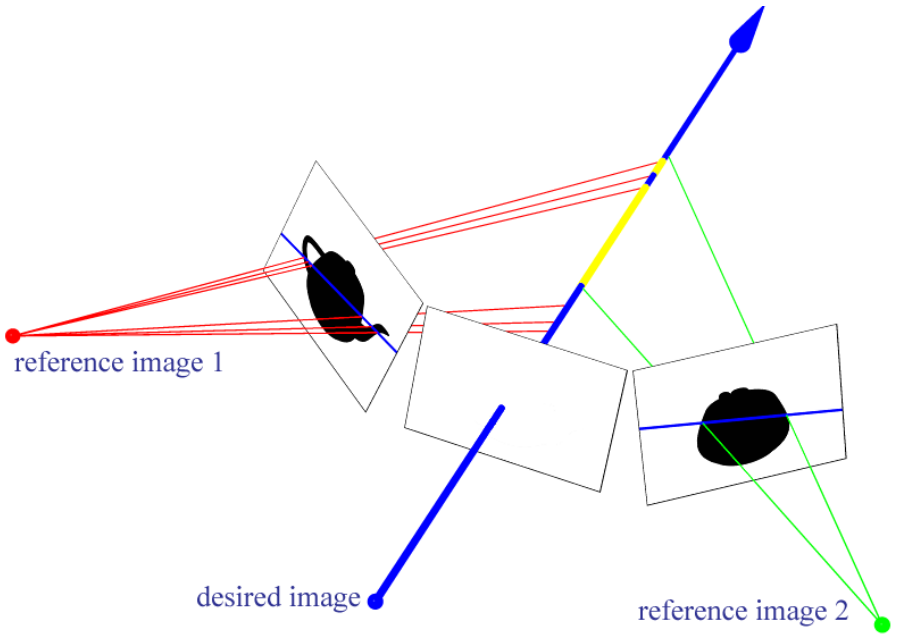
In [Li [59]] wordt een visual hull algoritme voorgesteld dat de aanpak van [Matusik et al. [71]] op de hardware implementeert waardoor berekeningen versneld worden. Het verschil met [Matusik et al. [71]] en voorgaande besproken technieken is dat er geen expliciet 3D model van het oppervlak van VH(O,V) wordt gereconstrueerd, maar een impliciete geometrie die de dieptemappen van de visual hull reconstrueert. Eerst worden alle binair in voor- en achtergrond gesegmenteerde silhouette-afbeeldingen S_i met i-de index als texture unit ingeladen. De alpha-transparantiewaarde van iedere texture is 1 voor de voorgrond en 0 voor de achtergrond. Voor ieder reprojectievolume, beschreven door silhouette S_i, wordt er voor iedere face v_k, beschreven door edge e_k op de i-de contour, een projectie uitgevoerd en wordt de face gerenderd. Iedere face wordt vervolgens met alle andere reprojectievolumes S_j met i $\neq$ j geïntersecteerd door projectieve texture mapping. Iedere texture T_j wordt geprojecteerd op v_k volgens iedere projectiematrix P_j berekend tijdens de calibratie. Door de alpha-test op de grafische kaart te activeren worden enkel die stukken op de face na intersectie met iedere geprojecteerde

texture T_j gerenderd die niet alpha-waarde 0 hebben. De texture mapping zorgt ervoor dat een face in een aantal polygonen met alpha waarde 1 wordt opgedeeld.
Nu dienen nog enkel die polygonen op een face gerenderd te worden die consistent zijn met alle silhouetten. Dit zijn de polygonen die in de doorsnede liggen van alle polygonen op een face. Deze doorsnede kan eenvoudig worden berekend door de alpha-waarden te vermenigvuldigen tijdens de projectieve texture mapping van iedere texture T_j. Alle posities op een face waarvoor er 1 minstens geprojecteerde texture een 0 alpha-waarde heeft zullen dus niet worden gerenderd. Enkel polygonen consistent met al de silhouetten S_j zullen op face v_k overblijven. Nadat dit reconstructieproces voor alle faces k van silhouetten i is uitgevoerd, volgt er in graphische hardware een rasterisatiestap waarin verborgen oppervlakten verwijderd worden. Het resultaat van het reconstructieproces zijn dieptemappen van de zichtbare oppervlakten. Het reconstructieproces voor een face is in afbeelding 4.24 geïllustreerd.

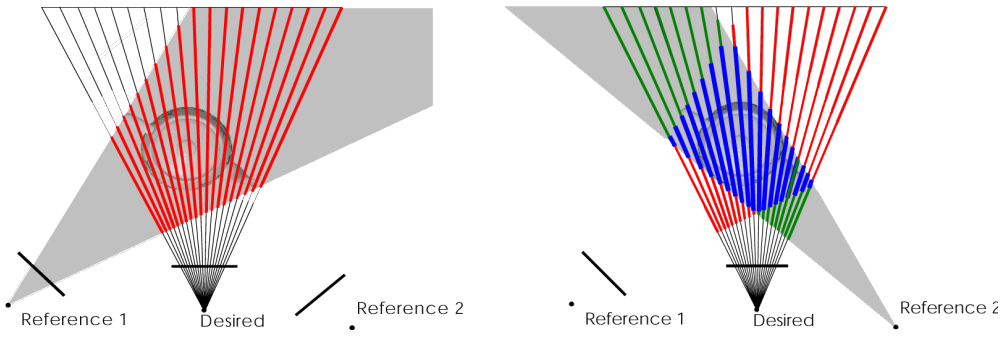
Oppervlaktegebaseerde methoden kunnen accurate tot op de pixel exacte de visual hull representeren [Franco and Boyer [32]] die visueel vaak realistischer en smoother ervaren worden dan de discrete voxelrepresentaties. Geometrische intersectiemethoden die snijpunten tussen rays berekenen, kunnen numerieke instabiliteiten veroorzaken wanneer de reprojectierays elkaar bijna raken bijvoorbeeld wanneer silhouette-afbeeldingen geschoten zijn vanuit camerastandpunten die dicht bij elkaar liggen. Deze instabiliteiten kunnen anomaliën tot gevolg hebben zoals gaten, duplicaten, zelfintersecties[Franco and Boyer [32]] hetgeen resulteert in corrupte mesh modellen [Xia et al. [107]]. De gpu-gebaseerde methode lijdt hier echter niet onder omdat intersecties in beeldruimte berekend worden [Li [59]]. De topologische gebaseerde en differentiële methoden zijn alternatieve methoden die zulke degeneratiegevallen proberen te vermijden. Dit doen ze ten koste van smoothness assumpties en dienen ze soms speciale condities in te bouwen om met extreme gevallen zoals zelfocclusies, t-juncties, cusps en scherpe hoeken [Chen [18]] om te gaan.

4.6.3 Beeldgebaseerde visual hull

Bij beeldgebaseerde visual hull technieken wordt een nieuw beeld rechtstreeks uit de inputafbeeldingen gegenereerd zonder hiervoor een volledig geometrisch model te reconstrueren. In [Matusik et al. [70]] wordt een beeldgebaseerde visual hull representatie afgeleid op een manier die sterk lijkt op het principe van het berekenen van bounding edges eerder besproken bij oppervlaktegebaseerde methoden zoals [Matusik et al. [71]], [Franco and Boyer



Figuur 4.25: Illustratie van de berekening van een beeldafhankelijke visual hull representatie. De blauwe viewing ray vanuit het nieuw camerastandpunt wordt in de silhouetteafbeeldingen geprojecteerd en vervolgens met de silhouetten geïntersecteerd. De gele intervallen op de viewing ray zijn het resultaat van de doorsnede van alle gereprojecteerde intervallen die silhouette-consistent zijn. Bewerkte afbeelding van bron: [Matusik and McMillan [69]].



Figuur 4.26: Bovenaanzicht van een visual hull representatie van het theepotobject vanuit het nieuwe camerastandpunt. De visual hull voor het nieuwe camerastandpunt bestaat uit de blauwe regio die de silhouette-consistente raysegmenten bevat. Bron afbeeldingen: [Matusik and McMillan [69]].

[32]]. In plaats van alle bounding edges, die het resultaat zijn van intersecties van silhouette-faces met alle andere silhouette-afbeeldingen te berekenen, worden in de beeldgebaseerde methode enkel de 3D punten van het oppervlak van de visual hull berekend die vanuit het nieuwe camerastandpunt zichtbaar zullen zijn. Dit resultaat, dat voorgesteld kan worden als een diepte-afbeelding van de visual hull uit het nieuwe camerastandpunt, kan met behulp van ray casting en epipolaire geometrie als volgt worden berekend (zie afbeeldingen 4.25 en 4.26 voor illustraties):

- Projectiefase

  Iedere viewing ray gevormd door een pixel in het gewenste beeld en het nieuwe camerastandpunt wordt in alle gesegmenteerde inputafbeeldingen geprojecteerd, waarbij de projectie-afbeeldingen van deze ray de epipolaire lijn van deze ray in iedere inputafbeelding voorstelt.

- Intersectiefase

  Voor iedere silhouette-afbeelding worden de silhouettecontouren berekend en voorgesteld door een lijst van stukjesgewijze lineaire edges. Vervolgens worden de intersectiecoördinaten van de epipolaire lijn met iedere silhouettecontour berekend. Deze intersectieberekening kan worden versneld door gebruik te maken van het edge-bin of het wedge cache algoritme, beide beschreven in [Matusik and McMillan [69]].

- Reprojectiefase

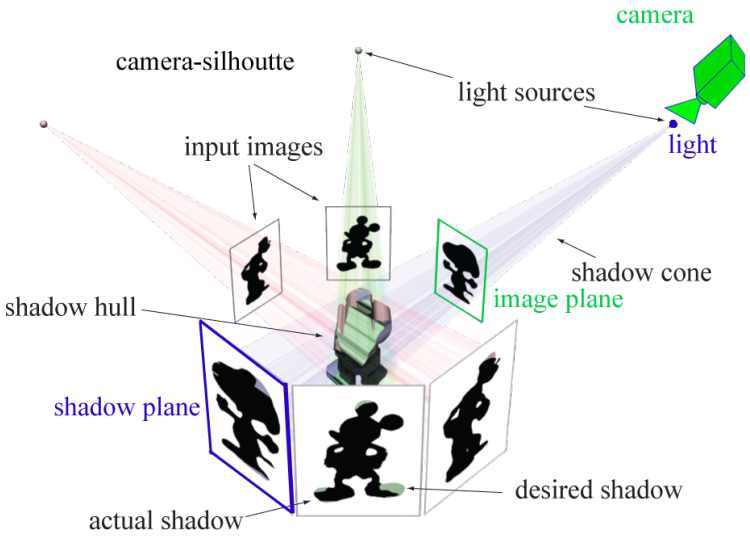
  Alle intersectie-intervallen, ontstaan na intersectie van een geprojecteerde viewing ray met iedere silhouette-afbeeldingen worden gereprojecteerd op de viewing ray. Vervolgens wordt de doorsnede van alle gereprojecteerde intervallen op de viewing ray berekend. Deze doorsnede, geïllustreerd in figuur 4.26 bestaat uit de intervallen op de viewing ray die consistent zullen zijn met alle silhouette-afbeeldingen. Het voorste 3D punt van ieder interval op een viewing ray dat het dichtst bij het nieuwe camerastandpunt ligt, stelt een punt op het oppervlakte van de visual hull voor. De dieptewaarden van al deze voorste punten opgeslagen in een afbeelding stellen de diepte-afbeelding van het visual hull oppervlak ten op zichte van het nieuwe camerstandpunt voor. De nieuwe afbeelding kan worden ingekleurd door de voorste punten in de verschillende inputafbeelding te projecteren en de kleurwaarde van de pixel van dichtsbijzijnde camera waarvoor het voorste punt zichtbaar is, aan de overeenkomstige pixel van de nieuwe afbeelding toe te kennen.

In [Miller and Hilton [77]] worden de silhouettecontouren niet benaderd door lineaire edges, maar wordt iedere contourpixel geïndexeerd in een-look up table volgens de hoek die de pixel maakt met de epipoolafbeelding van het nieuwe camerastandpunt in de gegeven referentie-afbeelding. Dit laat tot op de pixel exacte intersecties toe die efficiënt kunnen worden berekend door de look-up resultaten op te vragen voor de hoek die een epipolaire lijn van een viewing ray r_i in het nieuwe beeld maakt met het epipoolpunt van het nieuwe camerastandpunt in de referentie-afbeelding. Het voorste punt, het punt op de visual hull het dichtst bij de gewenste camera, wordt, zonder gebruik te maken van reprojectie, bepaald door de projectie-invariante kruisverhoudingen van intersectiepunten over de verschillende referentie-afbeeldingen te vergelijken en te sorteren volgens stijgende afstand tot het nieuwe cameraprojectiecentrum. Wanneer het voorste punt gekend is, wordt de afstand van dit punt tot het nieuwe cameraprojectiecentrum berekend en als dieptewaarde op de overeenkomstige pixel van viewing ray r_i opgeslagen. De tot op de pixel exacte beeldafhankelijke visual hull representatie kan worden bekomen door dit proces voor iedere viewing ray in het nieuwe beeld uit te voeren. De 2 besproken beeldgebaseerde visual hull technieken zijn dus in essentie achterwaartse mappingtechnieken van doelafbeelding naar bronafbeeldingen.

De 2D intersecties bij de berekening van beeldgebaseerde visual hulls zijn robuust [Magnor [67]] hetgeen voor 3D intersecties in praktijk niet steeds het geval is zoals eerder werd aangehaald bij oppervaktegebaseerde visual hulls. Bovendien kunnen approximaties zoals een lineaire benadering van silhouetteranden worden vermeden en wordt er geen quantisatietussenstap zoals bij volumegebaseerde methoden gebruikt. Een nadeel van de beeldgebaseerde visual hull methode is dat er geen 3D model gereconstrueerd wordt waardoor deze methode minder geschikt is voor bepaalde toepassingen zoals computergames met geanimeerde 3D modellen.

4.7 Uitbreidingen

De visual hull kan enerzijds verfijnd worden door de hoeveelheid silhouetteinformatie via schaduwen, of temporele informatie te vergroten en anderzijds door naast silhouette-informatie van andere visuele aanwijzingen gebruik te maken zoals textuur- en diepte-informatie. Deze sectie gaat op deze uitbreidingen in.



Figuur 4.27: Illustratie van analogie tussen visual hull reprojectievolumes (beschreven door de camera en beeldvlak in het groen) en schaduw hull reprojectievolumes (beschreven door lichtbron en schaduwvlak in het blauw). Geëditeerde afbeelding van bron: [Mitra and Pauly [79]] waar schaduw hulls voor artistieke doeleinden gebruikt worden om een nieuwe figuur te genereren. In deze illustratie is de samengestelde figuur een mix van Disney-figuren.

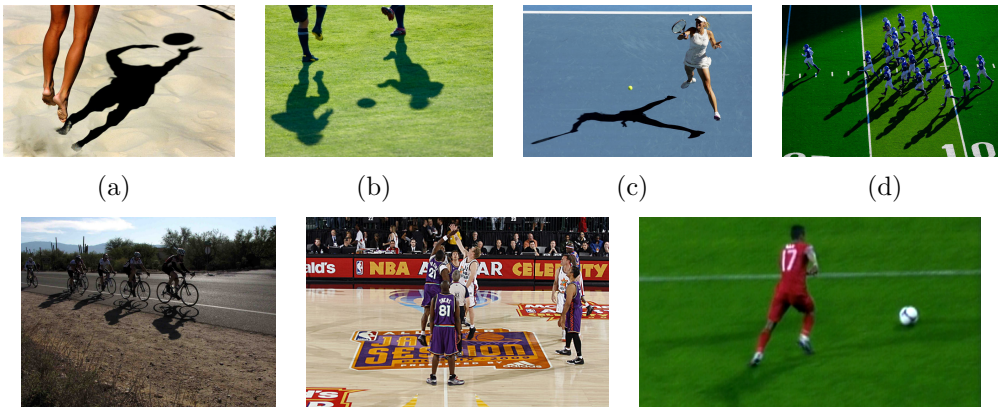
4.7.1 Schaduw hull

Zij O het te reconstrueren object, L een lichtbron, V de regio waarin verschillende lichtbronnen zijn geplaatst. De schaduw hull is als volgt gedefinieerd:

Definitie 4.7.1. SH(O,V) = verzameling van alle punten $P_k \in E_3$ waarbij voor ieder punt voor alle mogelijke lichtbronnen $L_i \in V$ moet gelden dat iedere schaduwstraal $s_j = \overrightarrow{L_i P_k}$, dit is de halfrechte startend in L_i door het punt P_k, minstens 1 punt van het object O bevat. Dus ieder mogelijke s_j moet het object steeds snijden of raken.

De schaduw hull is analoog aan de visual hull [Matsushita [68]] en wordt bekomen door een beeldvlak achter het object te plaatsen ten opzichte van een lichtbron. Camera's worden dus vervangen door lichtbronnen. Schaduwstralen vanuit de lichtbron die het beeldvlak niet belichten zullen zo een silhouette vormen. Deze analogie is afgebeeld in figuur 4.27. De schaduw hull kan dus gecombineerd worden met de visual hull en kan extra constraints op de silhouetten opleggen zonder dat hiervoor camera's moeten worden bijgeplaatst. Voorbeelden van schaduwen in verschillende sportscènes zijn afgebeeld in figuur 4.28. Wanneer meerdere lichtbronnen geplaatst zijn, kan

er veel vorminformatie van een object in 1 beeld worden opgeslagen. Deze lichtbronnen dienen net als camera's gekalibreerd te worden om de posities ervan te achterhalen zodat de schaduwreprojectievolumes kunnen gereconstrueerd worden. In het geval van beweegbare lichtbronnen zoals de zon is een regelmatige herkalibratie nodig. Voor sommige objecten die moeilijk te segmenteren zijn, zoals een fiets en planten, kan rechtstreeks van hun schaduwen gebruik worden gemaakt om betrouwbaardere, gesegmenteerde afbeeldingen te verkrijgen dan state-of-the art segmenatie algoritmes toelaten. Door schaduwinformatie te gebruiken is een hoge kwaliteitsreconstructie van complexe objecten in goed geconditioneerde omgevingen mogelijk [Yamazaki et al. [109]].



Figuur 4.28: Voorbeelden van schaduwen in sportscènes. Bronnen: [Schaduw [88]], [Schaduw [89]], [Schaduw [90]], [Schaduw [91]].

4.7.2 Photo hull

Zij O het te reconstrueren object, V de view regio waarin camera's zijn geplaatst. De photo hull is als volgt gedefinieerd [Kutulakos and Seitz [50]]:

Definitie 4.7.2. PH(O,V) = verzameling van alle punten $P_k \in E_3$ waarvoor er een radiantiefunctie bestaat waarvoor P_k fotoconsistent is met de inputafbeeldingen en voor iedere objectpixel p_m in silhouette S_i bestaat er een punt P_k dat op p_m projecteert. Een punt is fotoconsistent als het niet in de achtergrond projecteert en als voor alle camera's $C_i \in V$ waarvoor het punt zichtbaar is, de radiantie van ray $r_i = C_i P_k$ gelijk is aan de radiantie van de projectie van P_k op het beeldvlak I_i van camera C_i.

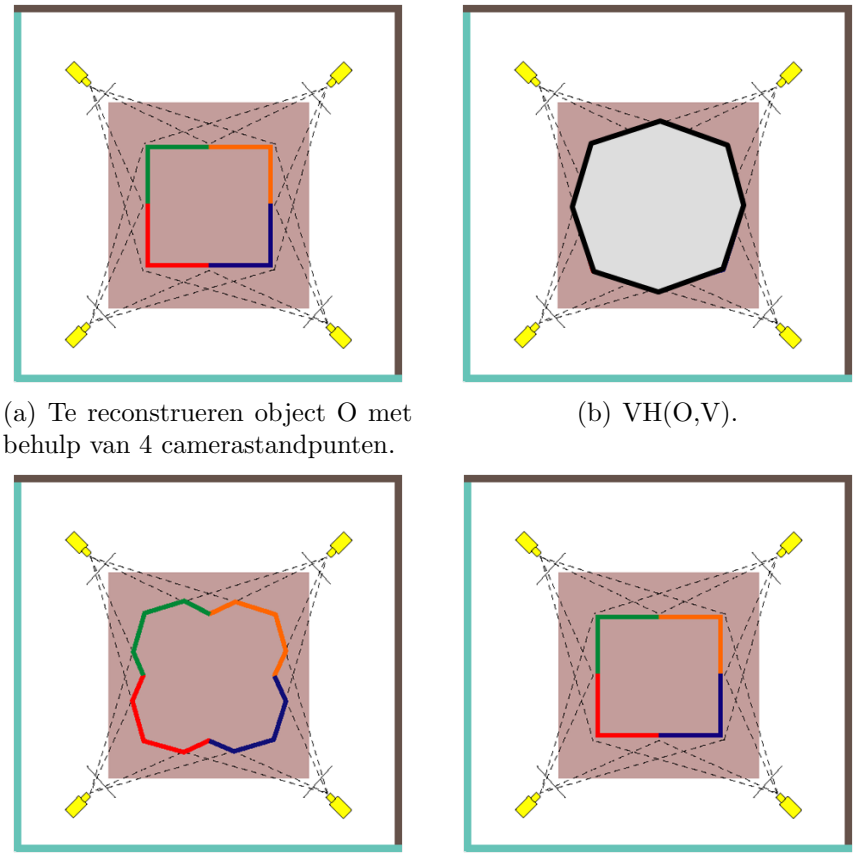
De photo hull is uniek en is de maximaal fotoconsistente vorm met de inputafbeeldingen. Zie [Kutulakos and Seitz [50]] voor bewijzen van deze theorema's. De photo hull is die vorm die wanneer ze gefotografeerd vanuit dezelfde camerastandpunten als het oorspronkelijke object dezelfde gekleurde afbeeldingen als resultaat heeft als de inputafbeeldingen. De visual hull is een speciaal geval van de photo hull waarbij de inputafbeeldingen binair gesegmenteerd zijn. In dat geval staat fotoconsistentie gelijk aan silhouetteconsistentie vermits alle punten binnen een silhouette dezelfde kleur hebben zullen ze geen constraints leggen op andere silhouettereprojectievolumes en zal de photo hull bestaan uit alle punten die op de silhouettepixels projecteren. Wanneer de objectkleuren uit de inputafbeelingen als constraints op silhouette zullen er punten die niet-fotoconsistent zijn niet tot de photo hull behoren. Een voorbeeld hiervan is afgebeeld in figuur 4.29(c). De photo hull is dus een strikter omhulsel dan de visual hull en kan wel concave regionen reconstrueren [Kutulakos and Seitz [50]] zolang projecties van concave regionen een kleurinconstistentie teweegbrengen met andere regionen. Uniform gekleurde objecten met concave holten zijn echter niet via fotoconsistentie reconstrueerbaar.
Een nadeel van deze methode is de vereiste van het bestaan van een radiantiefunctie waarbij de projectie van een objectpunt in de verschillende afbeeldingen waarvoor dit punt zichtbaar is een pixelintensiteit heeft die dezelfde is over alle afbeeldingen. Deze redenering gaat op voor diffuse objecten die de radiantie in alle richting weerkaatsen zoals afgebeeld in figuur 4.7, maar niet voor speculaire en glossy objecten die voor punten zichtbaar vanuit meerdere camera's verschillende intensiteiten hebben en volgens de definitie van PH(O,V) geen deel uitmaken van de photo hull. In principe is dit incorrect vermits zulke inconsistenties mogen voorkomen.
In praktijk kunnen punten voor diffuse objecten, die onder ideale omstandigheden als fotoconsistent worden gemarkeerd, toch verwijderd worden. Dit komt door afwijkingen in intensiteitswaarden van pixels in verschillende afbeeldingen. Hoewel deze pixels de afbeelding zijn van eenzelfde punt, toch kunnen er afwijkingen ontstaan door toedoen van pixelruis en verschillende camera kleurschema's. Om dit te verhelpen wordt vaak een afwijking van intensiteitswaarde getolereerd, of worden de kleuren van de camera's gekalibreerd.
In de literatuur zijn algoritmen beschikbaar die een incrementele berekening van de photo hull op de visual hull ondersteunen voor zowel volumegebaseerde [Kutulakos and Seitz [49]], oppervlaktegebaseerde [Li et al. [61]] als beeldgebaseerde representaties[Slabaugh et al. [98]].

4.7.3 Depth hull

Zij O, het te reconstrueren object, V de view regio waarin camera's zijn geplaatst en D_i de dieptemap bestaande uit de verzameling 3D punten zichtbaar voor i-de camera $C_i \in V$. Zij Z_i^k de projectie van een 3D punt P_k op D_i van camera C_i. De depth hull is als volgt gedefinieerd [Bogomjakov and Gotsman [6]]:

Definitie 4.7.3. DH(O,V) = verzameling van alle punten $P_k \in E_3$ waarbij voor ieder punt P_k voor alle mogelijke camerastandpunten $C_i \in V$ moet gelden dat iedere dieptestraal $d_j = \overrightarrow{Z_i^k P_k}$, dit is de halfrechte startend in Z_i^k door het punt P, minstens 1 punt van het object O bevat. Dus iedere d_j moet het object steeds snijden, of raken.

De depth hull is een uitbreiding van de visual hull die de viewing rays van iedere silhouete startend vanaf hun 3D diepte reprojecteert en intersecteert met rays uit andere silhouette met diepte-informatie. De depth hull is de maximale vorm die diepteconsistent is met de diepte-afbeeldingen van alle silhouetten. In theorie wanneer de diepte-informatie betrouwbaar is zal de depth hull alle zichtbare concave regionen kunnen construeren, ook diepe concave regionen wanneer camerastandpunten binnen deze regionen toegelaten zijn. Wanneer enkel een oneindig aantal camerastandpunten buiten CH(O) is toegelaten, zal de depth hull gelijk zijn aan de interne visual hull [Bogomjakov et al. [7]] omdat diepte-informatie die zichtbaar is buiten CH(O) de zichtbare concave regionen kan reconstrueren. Dit zijn die regio's waarvoor er silhouette-inconsistente rays vanuit camera's binnen CH(O) bestaan die concave regio's bij de interne visual hull uitsnijden zoals afgebeeld in figuur 4.6(b).
Zodra in praktijk de diepte van het hele objectoppervlak is vastgelegd, kan een object met een beperkt aantal camera's accuraat gereconstrueerd worden zoals afgebeeld is in figuur 4.29(d). De diepte-informatie werd in [Bogomjakov et al. [7]] verworven door dieptecamera's met ingebouwde projector die de structured light - techniek gebruiken. Structured light [Scharstein and Szeliski [93]] is een actieve reconstructiemethode toepasbaar in goed geconditioneerde omgevingen waarbij verschillende, welbepaalde gecodeerde infrarood lichtpatronen helpen bij het berekenen van betrouwbare diepte-informatie. Problemen met deze methode ontstaan wanneer meerdere dieptecamera's gelijktijdig een dieptebeeld willen opnemen omdat verschillende lichtpatronen dan met elkaar gaan interfereren. Andere dieptecamera's zoals diegene die diepte uit stereo-informatie halen, zijn minder betrouwbaar vanwege occlusieproblemen waarbij er features in een beeld niet zichtbaar zijn in het tweede beeld en correspondentieproblemen in speculaire en uniform ge-



(a) Te reconstrueren object O met behulp van 4 camerastandpunten.

(b) VH(O,V).

(c) PH(O,V).

(d) DH(O,V).

Figuur 4.29: Vergelijking van verschillende hulls: $O \subseteq DH(O,V) \subseteq PH(O,V) \subseteq VH(O,V)$. Bron afbeelding: [Kutulakos and Seitz [49]].

kleurde regio's. Hardwaregebaseerde dieptecamera's zijn bovendien erg duur in vergelijking met videocamera's.

4.7.4 Temporele visual hulls

In [Cheung [21]] worden de volgende stappen ondernomen om een verfijnde visual hull presentatie af te leiden:

- Reconstructiefase

  Voor een beperkt aantal camera's n worden de raakpunten op het oppervlak van de VH(O,V_n) op k verschillende tijdstippen $t_0, t_1 ... t_k$ berekend. Zo ontstaan er k visual hull puntenwolken, eentje voor ieder

tijdstip. Er is op 3 manieren geëxperimenteerd om een puntenwolk te reconstrueren:

- bounding edge representatie met gekleurde oppervlaktepunten. Per bounding edge die op dezelfde manier als bij [Franco and Boyer [32]] in afbeelding 4.21 wordt berekend worden oppervlaktepunten waar de bounding edge het object raakt gereconstrueerd. Dit wordt gedaan door het punt uit alle camera's met de minimum geprojecteerde kleurvariantie te selecteren, rekening houdend met visibiliteit.
- volumegebaseerde presentatie van visual hull met als punten de oppervlaktevoxels die ingekleurd zijn met gereprojecteerde kleuren uit de silhouette-afbeeldingen
- volumegebaseerde presentatie van photo hull bekomen na verfijning van de visual hull. De centra van de reeds ingekleurde voxels van de photo hull worden als punten voor de puntenwolk gebruikt.

Doordat het object in de scene kan bewegen, gaan de silhouetteafbeeldingen $S_1, S_2 ... S_n$ en de visual hulls op verschillende tijdstippen variëren. Om 1 visual hull representatie te bekomen moeten er een alignering worden berekend. Uit experimenten van [Cheung [21]] blijkt dat de bounding edge representatie kwalitatief de beste resultaten geeft voor de berekening van deze alignering. Bij de voxelrepresentaties is er steeds een approximatie van een oppervlaktepunt. In het geval van de photo hull kunnen er te weinig voxels, of te veel zijn weggesneden hetgeen ongelijkmatigheden tussen puntenwolken kan veroorzaken ten opzichte van de verschillende tijdstippen.

- Aligneringsfase

Voor een rigide object kunnen rigide bewegingen zoals translaties en rotaties van een object ten opzichte van vast gepositioneerde camera's volledig gesimuleerd worden door cameratransformaties zodat niet het object, maar de camera beweegt en we voor het object althans hetzelfde resultaat krijgen wanneer de camera's voor en na deze beweging een beeld zouden opnemen. Voor verschillende objecten die tegelijkertijd bewegen zijn ten opzichte van ieder object cameratransformaties nodig om deze beweging via een camera te simuleren [Xiao et al. [108]]. Voor de 3D puntenwolk van een object dat uit 1 vaste massa is opgebouwd, bestaat er een rotatie en translatie (R_i, T_i) die de puntenwolk, gereconstrueerd uit de camera-afbeeldingen op tijdstip t_i, aligneert met de

camera's op tijdstip t_j. Deze transformatie (R_i, T_i) wordt in [Cheung [21]] berekend door die (R_i, T_i) die de intensiteitsafwijking tussen de gemiddelde kleur van een punt in de puntenwolk met de kleuren in afbeeldingen op tijdstip t_j minimaliseert, te berekenen. Op die manier kunnen de afbeeldingen geschoten op tijdstip t_i toegevoegd worden aan de afbeeldingen op tijdstip t_j door de transformatie (R_i, T_i) op de afbeeldingen op tijdstip t_i toe te passen. Door transformaties van afbeeldingen op latere tijdstippen met het eerste tijdstippen te berekenen, kunnen alle afbeeldingen op latere tijdstippen $t_1 ... t_k$ volgens de transformaties naar het eerste tijdstip t_1 worden overgebracht waarbij alle bewegingen over de tijd via verschillende cameraposities op tijdstip t_1 bevat zijn. Dit levert op tijdstip t_1 k x n silhouetteafbeeldingen.

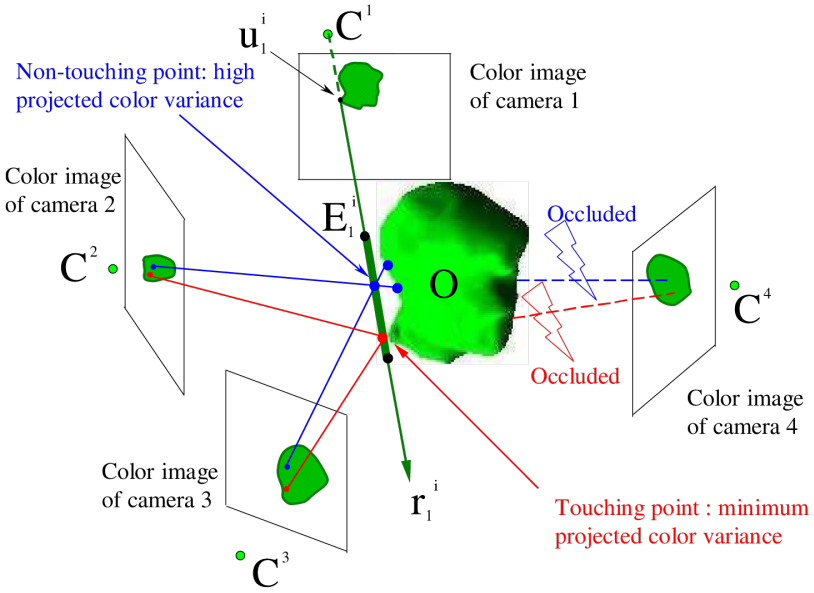
- Verfijningsfase

  Eens alle afbeelding naar tijdstip t_1 getransformeerd zijn, kan de VH(O,V) consistent met alle silhouette-informatie over de tijd gereconstrueerd worden. Het reconstructieproces van de VH(O,V) over de tijd is geïllustreerd in figuren 4.30 - 4.32.

Voor gearticuleerde objecten zoals het menselijk lichaam die uit meerdere beweegbare rigide onderdelen kunnen bestaan, is het een complexer probleem om de visual hull over tijd te berekenen omdat de beweegbare onderdelen niet gekend zijn en gesegmenteerd moeten worden. In [Cheung [21]] wordt hierbij eerst een arbitraire segmentatie gebruikt die verfijnd wordt door de spatiale en temporele coherentie uit de relatieve bewegingen van punten in de puntenwolk af te leiden. Hiervoor is wel voldoende beweging nodig. Eens alle beweegbare delen geïdentificeerd zijn, kan er voor ieder beweegbaar onderdeel m in de puntenwolk voor elk bepaald tijdstip t_i met i $\neq 1$ een transformatie (R_i^m, T_i^m) berekend worden om het deelsegment S_i^m wat met dit onderdeel in iedere overeenkomt te mappen naar een afbeelding van S_i^m op tijdstip t_1. Vervolgens kan VH(O,V) consistent met alle silhouette-informatie over de tijd gereconstrueerd worden.
Dit algoritme van [Cheung [21]] ondervindt nog problemen wanneer de relatieve bewegingen tussen gearticuleerde onderdelen te klein is. Hierdoor kan het zijn dat het algoritme, dat de bewegingen tijdens de aligneringsfase inschat, niet convergeert.

4.7.5 Stereo informatie

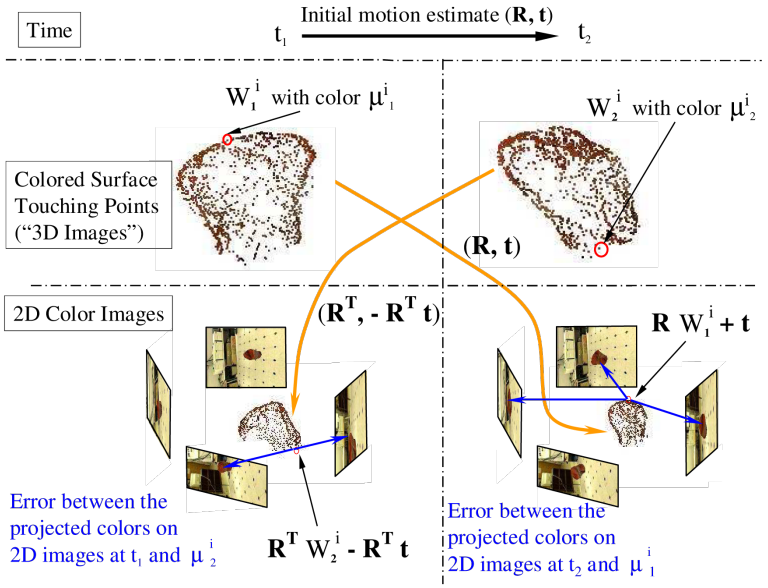
De combinatie stereo-visual hull is interessant op zich omdat beide technieken elkaar kunnen verbeteren:



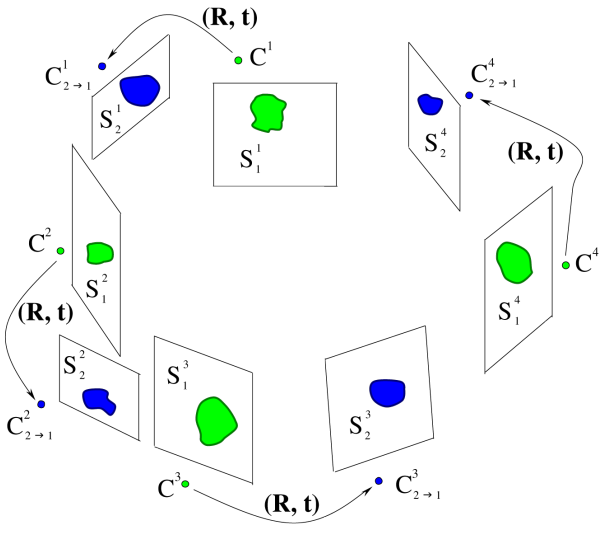
(a) Opbouwen van puntenwolk door detectie van gekleurde oppervlaktepunten. C stellen cameracentra voor. Subscripten stellen de tijdstippen voor. De gerepojecteerde i-de ray van pixel wordt voorgesteld door r. E is de bounding edge voortgebracht door de i-de pixel. Een punt op de contourrand wordt voorgesteld door u.

(b) Resulterende puntenwolken op verschillende tijdstippen.

Figuur 4.30: Reconstructieproces op ieder tijdstip. Bron afbeelding: [Cheung [21]].



Figuur 4.31: Aligneringsproces tussen 2 tijdstippen t_1, t_2. Subscripten stellen de tijdstippen voor. W^i stelt het i-de gekleurde oppervlaktepunt voor met gemiddelde kleur μ^i. R en t stellen rotaties en translaties voor. Bron afbeelding: [Cheung [21]].



(a) Illustratie van verfijning over 2 tijdstippen. Silhouette-afbeeldingen op tijdstip t_2 worden naar tijdstip t_1 getransformeerd. Subscripten stellen de tijdstippen voor, C de cameracentra, S de silhouetteafbeeldingen en (R,t) de transformaties naar tijdstip t_1.

(b) Visual hull resultaat na verfijning.

Figuur 4.32: Verfijningsproces. Bron afbeelding: [Cheung [21]].

- Diepte-informatie uit stereo laat toe om concave regionen van het object te reconstrueren en zo de visual hull te verfijnen.

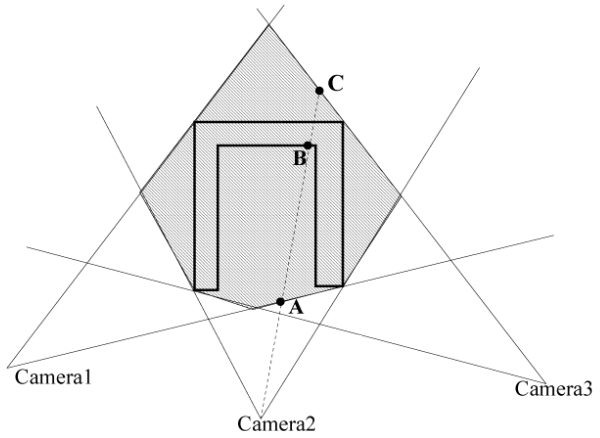
- De visual hull schatting begrenst het dieptebereik per pixel zoals afgebeeld in figuur 4.33. De diepte-ondergrens en -bovengrens per pixel kunnen gebruikt worden om valse correspondenties te identificeren en de zoekregio voor correspondenties over de epipolaire lijnen nog meer te beperken. Zo is er een initiële begrenzing op de mogelijke dispariteit.

- De 3D visual hull kan gebruikt worden om mogelijke occlusies te identificeren door te kijken of het 3D punt op de visual hull, waarvan de ene pixel de afbeelding is, ook zichtbaar is in de andere afbeelding. Dit wil zeggen dat de projectieray van het 3D punt naar de andere afbeelding geen andere 3D punten van de visual hull mag snijden.

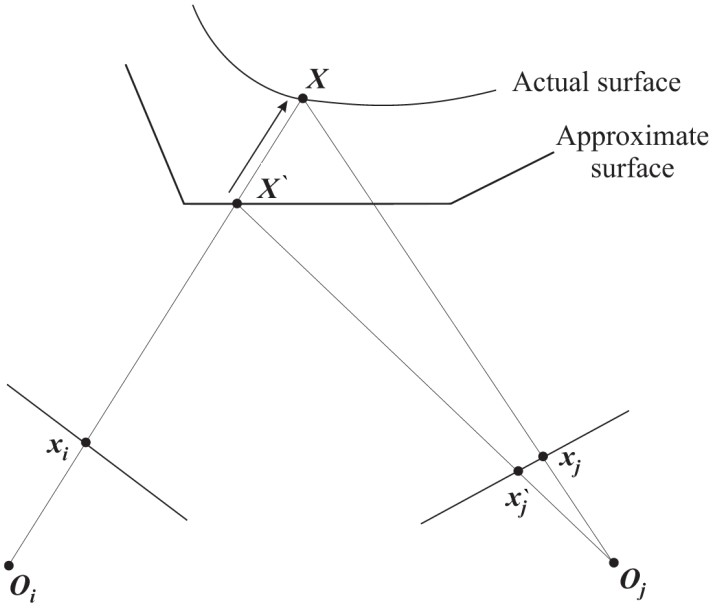
In [Li et al. [60]] wordt een dieptemap met behulp van de visual hull tussen paren van stereo-opgestelde camera's berekend. De visual hull legt constraints op het bereik dat overeenkomstige pixels in gerectificeerde beeldvlakken zich kunnen verplaatsen. Resultaten tonen aan dat deze extra constraints de dieptemap aanzienlijk verbeteren en foute correspondenties die optraden zonder gebruik te maken van de visual hull kunnen elimineren.
Een andere werkwijze voorgesteld door [Lazebnik et al. [56]] en getoond in afbeelding 4.34 probeert het visual hull volume te verbeteren door te zoeken naar 3D punten op de viewing ray r_i door een pixel x_i en een 3D punt op het visual hull oppervlak die dichter bij het objectoppervlak aanleunen. Selecteer dat 3D punt X waarvoor de pixelafbeelding x_j op de epipolaire lijn van deze viewing ray r_i in afbeelding I_j een minimaal intensiteitsverschil heeft met pixel x_i, waarbij i $\neq$ j. De zoektocht naar een fotoconsistentere pixel in een ander beeld kan worden beperkt in 1 richting startend vanaf de pixelafbeelding x'_j van het 3D punt X' op het VH-oppervlak dat we willen verfijnen verdergaand naar pixels die afbeeldingen zijn van 3D punten die verder van de camera verwijderd zijn dan X'. Enkel pixelkandidaten in deze richting moeten op de epipolaire lijn onderzocht worden vermits de visual hull een conservatieve approximatie is die het object gegarandeerd bevat. De kandidaatpixels met een betere pixelintensiteitsovereenkomst gaan het werkelijke punt X op het oppervlak van het te reconstrueren object beter benaderen.

4.8 Conclusie

In dit hoofdstuk werd de visual hull techniek samen met enkele andere hulltechnieken en uitbreidingen besproken. De visual hull techniek reconstrueert



Figuur 4.33: De visual hull (gearceerde regio) van het hoefijzerobject definieert diepte- en dus ook dispariteitsconstraints voor stereo matching. Punt B in de concave regio moet zich tussen A en C bevinden. Bron afbeelding: [Li et al. [60]].



Figuur 4.34: Verfijnen van het visual hull volume via stereo. Door het vergelijken van pixelintensiteiten in stereobeelden over epipolaire lijnen kan het visual hull oppervlak verfijnd worden van (X‘ naar X). Bron afbeelding: [Lazebnik et al. [56]].

uit gesegmenteerde silhouette-afbeeldingen het volume dat maximaal consistent is met alle silhouetten en ook de beste conservatieve approximatie is van het object gegeven silhouette-informatie. Vervolgens werd ingegaan op de objecten die de visual hull kan reconstrueren en wat de invloed is van view regio's op de reconstructiekwaliteit. Een inherente beperking aan de visual hull is dat ze een benadering is en niet alle concave regionen kan reconstrueren omdat concave informatie niet steeds door silhouetten kan beschreven worden. Doordat de techniek niet afhangt van oppervlakte-eigenschappen zijn er ook robuuste algoritmen beschikbaar waaronder volumetrisch gebaseerde methoden. Oppervlaktegebaseerde methoden leveren een accurater 3D model dat visueel meer plausibel is. Oppervlaktegebaseerde methoden zijn minder flexibel wannneer we de visual hull benadering m.b.v. andere informatie zoals textuurinfo willen verfijnen. Beeldgebaseerde visual hulls gebruiken geen tussenliggende representatie of quantisatiestappen waardoor aliasing wordt vermeden. Een nadeel van beeldgebaseerde methodes is dat we telkens per nieuw camerastandpunt de beeldafhankelijke representatie moeten berekenen. Dit is echter een klein nadeel, want bij dynamische scenes is voor iedere visual hull representatie een herberekening nodig. Het visual hull volume kan zonder het aantal camera's te moeten verhogen, worden verfijnd door schaduw-, fotometrische-, diepte-, temporele, of stereo-informatie in rekening te brengen.

4.9 Toepassing

Aangezien voor beeldinterpolatie van sportscenes een zo realistisch mogelijk beeld gewenst is, gaat de voorkeur uit naar een beeldgebaseerde visual hull representatie. Deze representatie is schaalonafhankelijk en gebruikt geen tussenliggende representatie die quantisatie, of aliasing introduceert zoals de volumegebaseerde representatie. Daarnaast laat de dieptemaprepresentatie ook gemakkelijk toe om de de visual hull verder te verfijnen, bijvoorbeeld via stereo correspondenties of fotometrische informatie. Wanneer een 3D model echter gewenst is, zouden we een volumetrische representatie verkiezen boven een oppervlaktegebaseerde representaties omdat deze laatste veel moeilijker te verfijnen is via bijvoorbeeld fotometrische informatie: de topologische structuur van de mesh moet dan wijzigen terwijl bij de volumetrische methode gewoon voxels mogen verwijderd worden zonder dat de structuur van de mesh in het gedrang komt.
Voor sportscenes zal de visual hull, mits voldoende silhouette-informatie, al een redelijke benadering van de te reconstrueren scene zijn omwille van de volgende redenen:

- In sportscenes hebben we vaak te maken met beeldinterpolatie van menselijke objecten. Deze bevatten slechts een beperkt aantal concave regionen.

- Voor betrouwbare diepte-informatie op lange afstanden zijn hoge resolutiebeelden, of bredere camerabaselines nodig [Gallup et al. [33]]. Een brede camerabaseline zal het aantal constraints op het object vergroten, nauwkeurigere diepteberekeningen toelaten en dus, gegeven een beperkt aantal camera's, een betere benadering van het te reconstrueren object geven.

De hoeveelheid silhouette-informatie kan door tijdsinformatie in rekening te brengen vergroot worden zodat meer constraints en dus een betere benadering uit silhouetten mogelijk is. Dit is echter een complexe onderneming omdat sporters constant in beweging zijn en alle gearticuleerde onderdelen afzonderlijk gedetecteerd en gealigneerd moeten worden. Een andere optie om meer silhouette-informatie te bekomen is door gebruik te maken van schaduwinformatie. Deze informatie zal wel niet in iedere sportscene toepasbaar zijn wegens complexe belichting, of onduidelijke schaduwen. Dieptecamera's zijn vrij duur, en voor betrouwbare diepte-informatie zijn hoge resoluties vereist. Gebruik maken van fotometrische informatie heeft ook zijn beperkingen wanneer de spelersuitrusting bijvoorbeeld bestaat uit uniforme regio's. Dit beperkt het aantal constraints op de photo hull en kan leiden tot onderschatting van het te reconstrueren object. In zo'n geval zijn vaak edges de enige features die betrouwbare informatie voor het vinden van correspondenties bevatten.

De volgende hoofdstukken zullen dieper ingaan op de berekening van de beeldgebaseerde visual hull en het matchen van edges als robuuste sparse features om de visual hull dieptemap te verbeteren. In dit hoofdstuk worden enkele nieuwe ideeën voor edge matching onderzocht. Eens dieptewaarden uit de correspondenties berekend zijn, kunnen de overige visual hull dieptewaarden in een later stadium worden verfijnd om zo geleidelijk aan over te gaan naar deze berekende dieptewaarden gezien de notie dat diepte van het objectoppervlak lokaal gezien gradueel toe- of afneemt (mogelijk via minimalisatiemethode).

Hoofdstuk 5

Beeldgebaseerde visual hull

Dit hoofdstuk bespreekt de methode van [Miller and Hilton [77]] voor het berekenen van beeldgebaseerde visual hulls. Vooreerst zal worden toegelicht waarom precies deze methode gekozen werd voor visual hull berekening. Vervolgens wordt de werking van deze visual hull techniek in detail besproken en zal ze geëvalueerd worden op zowel virtuele als reële datasets.

5.1 Motivering

De methode uit [Miller and Hilton [77]] werd uit alle besproken visual hull technieken gekozen omwille van de volgende redenen:

- beeldgebaseerd

  Beeldinterpolatie berekent als output een nieuw beeld uit gegeven inputbeelden. Het gebruik van andere tussenliggende representaties is voor het oplossen van dit probleem eigenlijk overbodig. Bovendien kunnen ze zoals eerder besproken artefacten zoals aliasing introduceren.

- exacte visual hull

  Er wordt geen gebruik gemaakt van geometrische benaderingen. Het algoritme zal een tot op de pixel-exacte visual hull berekenen. De nauwkeurigheid hangt hierbij samen met de nauwkeurigheid van de stappen die de visual hull berekening voorafgaan zoals camerasynchronisatie, cameracalibratie en beeldsegmentatie.

- hoge parallelliseerbaarheid

De visual hull berekeningen voor iedere pixel in het gewenste nieuwe camerastandpunt kunnen volledig onafhankelijk van elkaar uitgevoerd worden.

- 2D-operaties

  Berekeningen in 2D zijn efficiënter dan in 3D.

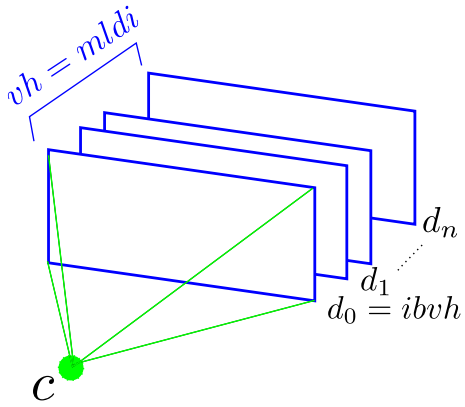
5.2 Methode

Het beeldinterpolatieprobleem voor een object in de scene kan worden herleid naar de volgende subproblemen:

- het achterhalen van het geometrisch voorkomen van het object ten opzichte van het nieuwe camerastandpunt. Dit wordt in deze thesis aangepakt door het berekenen van een dieptemap vanuit het nieuwe camerastandpunt, i.e. de beeldgebaseerde visual hull (sectie 5.2.1 Visual hull berekening).
- het achterhalen van de belichting en de inkleuring van het object (in dit geval de visual hull) in het nieuwe beeld (sectie 5.2.2 Visual hull shading).

5.2.1 Visual hull berekening

De beeldgebaseerde visual hull (ibvh) bevat de voorste punten van de visual hull ten opzichte van een camerastandpunt, hetgeen door een dieptemap kan worden voorgesteld. Op die manier kunnen alle visual hull punten ten opzichte van een camerastandpunt worden voorgesteld om zo de volledige visual hull door beelden ten opzichte van een camerastandpunt te beschrijven zoals geïllustreerd is in afbeelding 5.1. De afbeelding op diepte i (d_i) zal de visual hull punten bevatten die de i-de visual hull intersectie zijn van de reprojectiestralen, die deel uitmaken van het silhouetreprojectievolume vertrekkend van het nieuwe camerastandpunt c, met andere silhouetreprojectievolumes. In 2d komt dit overeen met de i-de visual hull intersectie van de epipolaire lijn met een silhouetcontour. Eerst zullen dus deze 2D intersecties moeten worden berekend. Daarna dienen alle gevonden 2D intersecties in referentiebeelden op een epipolaire lijn ten opzichte van de gewenste 3D beeldstraal te worden gesorteerd om hieruit de intersecties die deel uitmaken van de visual hull te selecteren. Tot slot wordt voor iedere gesorteerde visual hull intersectie een dieptewaarde berekend. Naargelang de intersectievolgorde worden deze dieptewaarde opgeslagen in de, met die volgorde corresponderende, dieptemap.



Figuur 5.1: Multi-layered depth images (mldi). De volledige visual hull kan worden gerepresenteerd door meerdere dieptebeelden d ten opzichte van een camerastandpunt c.

2D intersecties

Iedere reprojectiestraal vanuit het nieuwe camerastandpunt gaande door een pixel in het nieuwe beeld, wordt in alle referentie-afbeeldingen geprojecteerd om de intersectie met andere silhouetten te berekenen. Herinner dat de punten die niet binnen een silhouet vallen eigenlijk de grens bepalen van het visual hull volume dat maximaal consistent is met alle silhouet-afbeeldingen. Deze randpunten bepalen waar de silhouetinconsistenties beginnen en vormen de constraints op het visual hull volume. Het volstaat dus om in elke referentie-afbeelding de intersectie van alle epipolaire lijnen van het nieuw camerastandpunt met de silhouetcontouren te intersecteren. Tijdens deze intersectieberekening van pixels met epipolaire lijnen moet er rekening mee gehouden worden welke delen van deze oneindige lijn eigenlijk een geldige ray zijn vanuit het projectiecentrum van het nieuwe camerastandpunt. Er bestaat enkel een geldige intersectieberekening als de geldige delen van deze ray ook zichtbaar zijn in de beschouwde referentiecamera. Zo zal, in het normale geval wanneer het nieuw camerastandpunt C_{new} en referentiecamera's naast elkaar zijn opgesteld, enkel dat deel van de ray r die vertrekt vanuit het projectiecentrum C_{new} door het beeldvlak tot een punt op oneindig geldig zijn. Bij projectie van deze ray in een andere camera zal dit de vector zijn die start in de epipool en eindigt in het vluchtpunt. Alles in richting -r wordt bij gebruik van normale perspectiefprojectiecamera's niet op het beeldvlak van C_{new} vastgelegd en is dus ongeldig. Het kan echter wel zijn dat delen in richting -r wel op een beeldvlak van een referentiecamera wordt vastgelegd bijvoorbeeld wanneer deze achter camera C_{new} gepositioneerd is. Deze ongeldige segmenten moeten tijdens de intersectieberekening van epi-

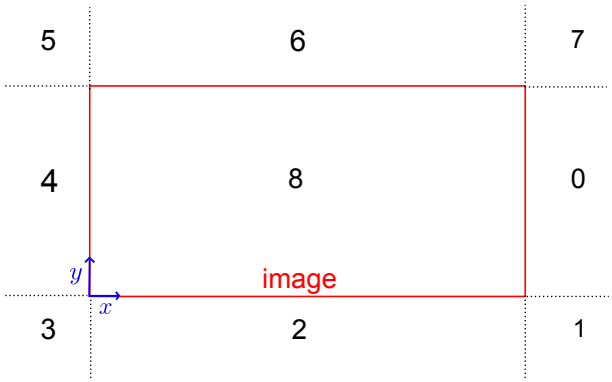
polaire lijnen met een silhouet-contour genegeerd worden vermits ze voor het nieuw camerastandpunt C_{new} een niet-zichtbaar deel van de scene beschrijven in een referentie-afbeelding. De verschillende gevallen van geldige epipolaire segmenten voor de mogelijke camerapositioneringen worden in bijlage B besproken. Een bijzonder geval voor intersectieberekening is wanneer we de visual hull in een referentiebeeld zelf willen berekenen en hierbij ook de informatie in dit referentiebeeld willen gebruiken. Dit kan nuttig zijn om een dieptemap te genereren. De intersectie-berekening met de referentie-afbeelding zelf kan in dit geval worden vervangen door enkel beeldstralen door de silhouet-voorgrond te schieten en deze met de andere referentie-afbeelding te intersecteren. De silhouet-voorgrond zal gelijk zijn aan het visual hull silhouet in het referentiebeeld omdat de visual hull maximaal consistent is met alle silhouet-afbeeldingen.

In [Matusik et al. [70]] worden, voor het berekenen van intersecties tussen epipolaire lijnen en silhouetcontouren, de silhouetcontouren via lijnen benaderd om vervolgens lijn-lijn intersecties te berekenen. In [Miller and Hilton [77]] worden exacte pixel intersecties berekend door alle contourpixels af te gaan en deze in de epipolaire lijnvergelijking $ax + by + c = 0$, waarbij a, b, c de homogene lijnparameters voorstellen. Deze lijnvergelijking verdeelt een vlak in 2 delen en dus kan aan de hand van het teken van $ax + by + c$ bepaald worden aan welke kant van de lijn een punt (x, y) ligt. Het snijpunt van een epipolaire lijn met een contour kan exact worden berekend door over een contour te itereren en achtereenvolgens opeenvolgende contourpixels in de lijnvergelijking in te vullen om tekenveranderingen op te sporen. Bij een tekenverandering kan de exacte intersectie worden berekend door een lijn tussen deze opeenvolgende contourpixels een lijn te construeren en vervolgens een lijn-lijn intersectie te berekenen tussen de epipolaire lijn en de contour subpixel-lijn.

Zij $l = (a, b, c)$ en zij $l' = (a', b', c')$. Het intersectiepunt P van beide lijnen is dan gelijk aan:

$$P = l \times l' \tag{5.1}$$

, waarbij $\times$ het kruisproduct tussen 2 vectoren voorstelt. Het achtereenvolgens testen op een tekenverandering vereist dat de objectcontouren eerst moeten getraceerd worden in de segmentatie-afbeelding. Deze contourtracering werd geïmplementeerd aan de hand van het algoritme van [Chang et al. [17]] dat in lineaire tijd toelaat zowel inwendige als uitwendige objectcontouren te traceren. Een resultaat van deze procedure op een gesegmenteerde afbeelding wordt getoond in figuur 5.14. In [Miller and Hilton [77]] worden contouren in gelijke bins onderverdeeld om het aantal contourpixels waarover



Figuur 5.2: Illustratie van de 9 verschillende regio's gevormd door het beeldvlak waarin de epipool kan liggen.

geïtereerd moet worden voor intersectieberekenig met een epipolaire lijn, te beperken tot het itereren over de contourpixels die zich in een bepaalde bin bevinden.

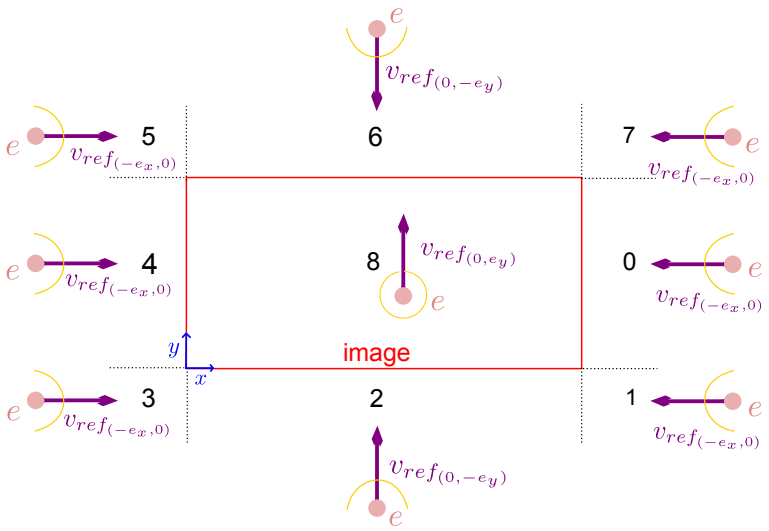
Contourbins

De verdeling van contourbins volgens hoekindexering uit [Miller and Hilton [77]] voor niet-parallelle epipolaire lijnen zal in deze sectie veralgemeend worden uitgelegd ten opzichte van alle mogelijke epipoolprojecties. Verder wordt een contourbin partitionering voor parallelle epipolaire lijnen uitgelegd zodat voor alle camerastandpunten een contourverdeling kan worden gebruikt om de intersectieberekening te versnellen.

Niet-parallelle epipolaire lijnen

In [Miller and Hilton [77]] worden de bins, die het referentiebeeldvlak verdelen, beschreven volgens de hoek die ze maken een referentievector v_{ref} die start vanuit de epipool e op het referentiebeeldvlak. Het beeldvlak bakent volgens [McMillan [73]] 9 regio's af waarin de epipool kan geprojecteerd zijn zoals weergegeven in figuur 5.2.

Opdat de methode en formules voor bin partitionering uit [Miller and Hilton [77]] bij iedere epipoolprojectie bruikbaar zouden zijn, dient de referentievector nauwgezet gekozen te worden naargelang de regio waarin de epipool geprojecteerd werd. Vooreerst zullen we bespreken hoe de referentievector moet worden gekozen. Het belang hiervan zal tijdens de bespreking van de formules worden aangetoond. De referentievector die een vlak in de



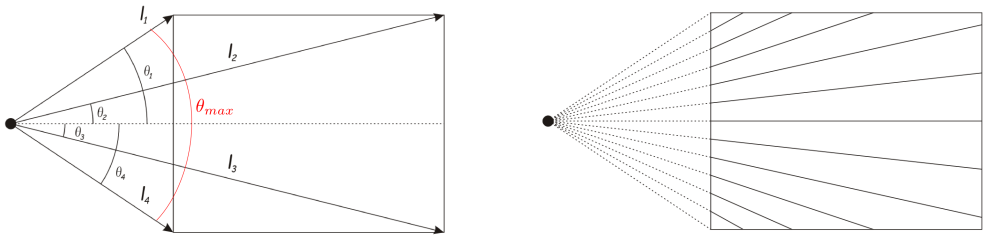
Figuur 5.3: Illustratie van de oriëntatie van de referentielijn v_{ref}, vertrekkend van een epipool e, die per regio als basis gebruikt wordt voor binpartitionering. De gele bogen rond een referentielijn geven de maximale hoekwaaier, i.e. het domein voor de hoeken die pixels in een willekeurig beeldvlak met deze referentielijn kunnen maken, weer.

richting van de vector opsplitst in positieve en negatieve hoeken moet per epipoolregio zodanig gekozen worden dat alle vectoren tussen de epipool en het beeldvlak binnen de hoekwaaier met domein $[-\frac{\pi}{2}, \frac{\pi}{2}]$ van de referentievector vallen, behalve voor het geval waarin de epipool in het referentiebeeld wordt geprojecteerd waar het domein van de hoekwaaier 2π zal zijn en de vector willekeurig kan gekozen worden. Voorbeelden van referentievectoren die hieraan voldoen zijn afgebeeld in figuur 5.3.

Eenmaal het projectiecentrum van het nieuwe camerastandpunt geprojecteerd is in een referentiebeeld en de corresponderende referentievector per regio geconstrueerd is, kan de maximale hoekwaaier worden berekend die een mogelijke epipolaire lijn in het referentiebeeld met de referentievector in een regio kan maken. Zij l_1, l_2, l_3, l_4 de epipolaire lijnen van de epipool tot ieder van de 4 hoekpunten van het referentiebeeld, zoals afgebeeld is in figuur 5.4. Zij $\theta_i = \angle(v_{ref}, l_i)$ de overeenkomstige hoek tussen de i-de lijn l en de referentievector, zijn $\theta_{min} = min(\theta_1, \theta_2, \theta_3, \theta_4)$, $\theta_{max} = max(\theta_1, \theta_2, \theta_3, \theta_4)$. De grootte van de hoekwaaier θ_{range} is gelijk aan:

$$\theta_{range} = \theta_{max} - \theta_{min} \tag{5.2}$$

Wanneer echter niet alle vectoren tussen het referentiebeeld en de epipool binnen de hoekwaaier van een referentievector vallen kan de hoekberekening



(a) Illustratie van de hoekwaaier $\theta_{max} = \theta_1 - \theta_4$ die een referentiebeeldvlak met de referentievector maakt.

(b) Bin constructie over de hoekwaaier θ_{max} tussen l_1 en l_4 met een gekozen aantal bins gelijk aan 12.

Figuur 5.4: Bin partitionering voor niet-parallelle epipolaire lijnen voor het geval dat de epipool projecteert in regio 4 van figuur 5.2. Bewerkte afbeelding uit [Miller and Hilton [77]].

verstoord worden zoals aangetoond wordt in figuur 5.5.

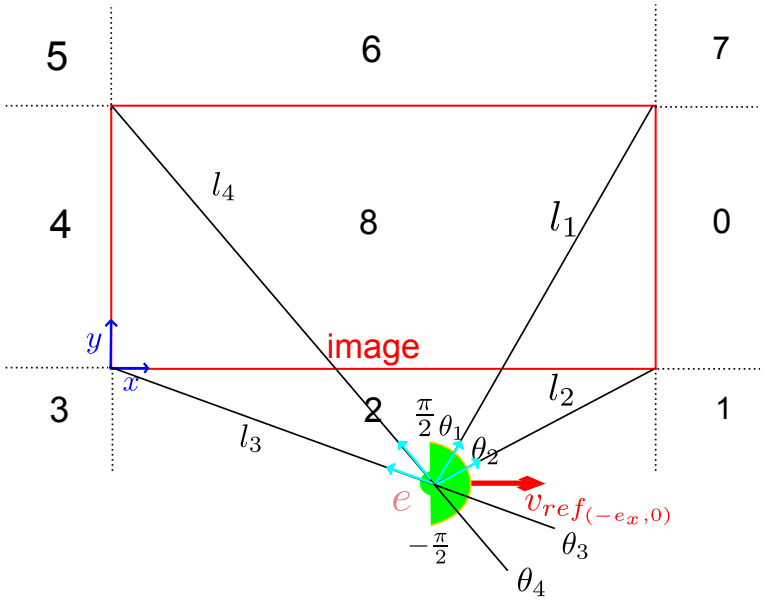
Uit de hoek range θ_{range} kan vervolgens een hoekindexering worden gekozen naarmate het aantal bins waarin we het beeldvlak wensen op te splitsen. Zij B het aantal bins. Dan is de hoek die tussen opeenvolgende bins θ_{incr} gelijk aan:

$$\theta_{incr} = \frac{\theta_{range}}{B} \tag{5.3}$$

Zij l_{min} de lijn tussen een hoekpunt die overeenkomt met θ_{min} en zij l_{max} de lijn die overeenkomt met θ_{max}. Dan kunnen alle binscheidinsglijnen worden gereconstrueerd door de waaier die start in l_{min} telkens met θ_{incr} te incrementeren tot l_{max}. Het corresponderende bin-nummer b_i voor een epipolaire lijn waarmee we de intersectie met een contour wensen te berekenen is dan gelijk aan:

$$b_i = \frac{\theta - \theta_{min}}{\theta_{range}} \tag{5.4}$$

Waarbij θ de hoek voorstelt tussen een geldige epipoolvector en de referentievector. De richting van de geldige epipoolvector kan uit de de epipoolcoördinaten worden afgeleid zoals in bijlage B betreffende epipolaire geometrie wordt uitgelegd. Contours in een referentiebeeld kunnen op basis van de reeds berekende epipoolprojectie van het nieuwe camerastandpunt in het referentiebeeld als preprocessingstap in contoursegmenten worden onderverdeeld door over de contour te itereren, per contourpixel het bin-nummer van de contourpixel te berekenen en een contoursegment van een bin te verlengen zolang er geen verandering in bin-nummer b_i optrad. Aangezien exacte



Figuur 5.5: Illustratie van een foute keuze voor referentievector v_{ref} waarbij de vectoren $\vec{l_3}, \vec{l_4}$ niet in de hoekwaaier van v_{ref} gelegen zijn. Het gevolg hiervan is dat $\theta_{range} = \theta_1 - \theta_4$ terwijl de range van hoeken die het beeldvlak ten opzichte van de epipool beschrijft eigenlijk gelijk moet zijn aan de range beschreven door θ_2 en θ_3.

intersecties tussen epipolaire lijn en contour berekend worden door te kijken naar een tekenverandering wanneer contourpixels in de lijnvergelijking $ax + by + c = 0$ worden ingevuld, dient in het geval dat een epipolaire lijn zou samenvallen met een binscheidingslijn, ook nog een extra overgangspixel met de aangrenzende bin te worden opgeslagen om de tekenverandering waar te nemen. Het resultaat van het onderverdelen van de contour in segmenten per bin is getoond in figuur 5.15.

Parallelle epipolaire lijnen Wanneer het beeldvlak van het nieuw camerastandpunt C_{new} gealigneerd is met een beeldvlak in een referentiecamera C_{ref} dan zal de projectievector $C_{new} - C_{ref}$ die de epipoolprojectie e_{new} van C_{new} op het beeldvlak van C_{ref} beschrijft parallel zijn met dit beeldvak. In dit geval zal de z-coördinaat van e_{new} gelijk zijn aan 0 en zal er geen geldige projectie op het beeldvlak van C_{ref} kunnen worden berekend [McMillan [73]]. Een bin partitionering van parallelle epipolaire lijnen kan worden geconstrueerd door ze te beschrijven ten opzichte van het beeldvlak. De maximale afstand tussen 2 parallelle lijnen over een beeldvlak is gelijk aan de diagonaal over het beeldvlak. Om een verdeling op basis van afstand te doen is het noodzakelijk dat de referentielijn die we gebruiken parallel is met de

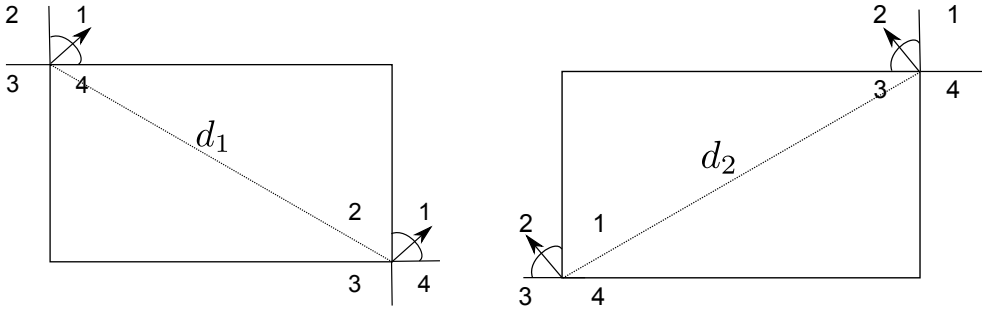
epipolaire lijn. De hoekpunten van de diagonaal d_1 gaande van linksboven naar rechtsonder kunnen beide alle oriëntaties van een epipolaire lijn door de oorsprong over kwadranten 1 en 3 beschrijven. Voor iedere lijn l_{1-3} door de oorsprong van kwadranten 1 en 3 kan op ieder punt van de diagonaal d_1 een overeenkomstige parallelle lijn in het hoekpunt linksboven, of rechtsonder geconstrueerd worden zonder dat hierbij de afstand wordt beïnvloed zoals kan worden afgeleid uit figuur 5.6(a). Bovendien zal iedere lijn l_{1-3} de diagonaal d_1 snijden vermits deze loopt over kwadranten 2 en 4 en niet samen zal vallen met de cartesiaanse assen. Iedere lijn l_{1-3} zal dus gegarandeerd een andere richtingscoëfficient hebben. Alle lijnen l_{1-3} kunnen naargelang de afstand tot een vast hoekpunt op diagonaal d_1 geparametriseerd worden. Dezelfde redenering, geïllustreerd in afbeelding 5.6(b) gaat op voor diagonaal d_2 gaande van rechtsboven naar linksonder en lijnen l_{2-4} over kwadranten 2 en 4. Voor een nieuw camerastandpunt bepaalt het kwadrant van de epipoolvector of de binverdeling moet geparametriseerd worden tegenover een hoekpunt op diagonaal d_1 met dezelfde richtingsvector, of tegenover een hoekpunt op diagonaal d_2 met dezelfde richtingsvector. Wanneer de richting van de epipoolvector evenwijdig is met de cartesiaanse assen, mag eender welk hoekpunt van het beeldvlak voor parametrisatie gekozen worden. Zij B het gewenste aantal bins, zij $|d|$ de lengte van diagonaal waarvan het gekozen hoekpunt deel uitmaakt. De incrementele afstand $dist_{incr}$ van een bin tot een volgende bin is dan gelijk aan:

$$dist_{incr} = \frac{|d|}{B} \tag{5.5}$$

Zij r de richtingsvector in het gekozen hoekpunt parallel aan de epipolaire lijnen. Dan kan het bin-nummer b_i voor een gegeven pixel p in het referentiebeeld als volgt worden berekend:

$$b_i = B\frac{distance(p, r)}{|d|} \tag{5.6}$$

Wanneer de parallelle richting van een epipolaire bemonsteringsvector van een pixel in een nieuw beeld is berekend, dan kunnen contourpixels in bins worden ingedeeld door per contourpixel het bin-nummer te berekenen. Per bin-nummer worden dan achereenvolgens de contourpixels met hetzelfde binnummer opgeslagen. Tussen opeenvolgende bins worden de overgangpixels in de respectievelijke bins mee opgenomen om in het geval dat epipolaire lijnen samenvallen met een binscheidingslijn nog steeds de intersectie van de epipolaire lijn met de contourpixel op de scheidingslijn kan worden berekend via het waarnemen van een tekenverandering zoals eerder werd toegelicht.



(a) Parallelle binverdeling op basis van de epipoolvector in het hoekpunt linksboven of rechtsonder wanneer epipoolvector in kwadranten 1 en 3 projecteert.

(b) Parallelle binverdeling op basis van de epipoolvector in het hoekpunt linksonder of rechtsboven wanneer epipoolvector in kwadranten 2 en 4 projecteert.

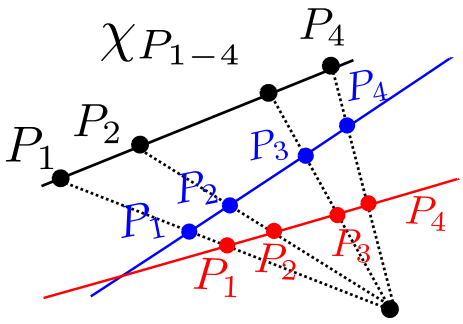
Figuur 5.6: Mogelijke situaties bij parallelle binverdeling van een beeldvlak op basis van de afstand van de diagonaal.

Ordenen van 2D intersecties

Per beeldstraal uit het nieuwe beeld waarvoor de kandidaat visual hull intersecties werden berekend, moeten de 2D intersecties voor deze beeldstraal die gevonden werden in alle referentiebeelden in stijgende afstand ten opzichte van deze beeldstraal worden gesorteerd om in een volgende stap te kunnen uitmaken vanaf welke intersecties silhouet-consistentie voor de beeldstraal ten opzichte van alle referentie-silhouetten geldt om dan te beslissen welke intersecties deel uitmaken van de visual hull. In [Miller and Hilton [77]] worden de 2D intersecties ten opzichte van de beedstraal volledig in 2D gesorteerd zonder dat ze hiervoor moeten worden gereprojecteerd op de beeldstraal. Dit kan worden verwezenlijkt door een projectie-invariante relatieve ordeningsmaatstaf te gebruiken zoals de cross ratio(kruisverhouding) van 4 collineaire punten (zie [Mundy and Zisserman [80]] voor bewijsvoering van de projectie-invariantie. De cross ratio $\chi_{(P_{1-4})}$ van 4 collineaire punten P_1, P_2, P_3, P_4 is gedefinieerd als:

Definitie 5.2.1 (Cross ratio)**.** $\chi_{(P_{1-4})} = \frac{|\overrightarrow{P_1P_2}||\overrightarrow{P_3P_4}|}{|\overrightarrow{P_1P_3}||\overrightarrow{P_2P_4}|}$

Uit de projectie-invariantie van deze ratio volgt dat wanneer we projectiestralen laten vertrekken vanuit 4 punten die zich op 1 lijn bevinden, elke mogelijke lijn door deze projectiestralen 4 snijpunten zal bevatten waarvan de cross ratio van deze 4 punten dezelfde zal zijn ongeacht de lijnkeuze zoals wordt geïllustreerd in figuur 5.7. De keuze van deze lijn stelt een willekeurige perspectiefprojectie voor. Dit houdt dus in dat voor 4 gekozen punten op een 3D beeldstraal gaande door het nieuw camerastandpunt en beeldvlak de



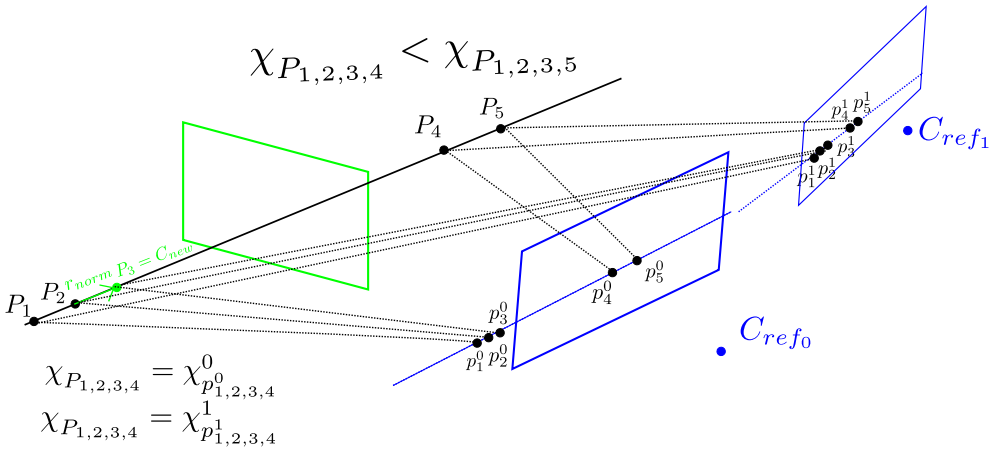
Figuur 5.7: Illustratie van projectie-invariantie van cross ratio. Voor elke lijn afbeelding over projectiestralen is de cross ratio dezelfde.

cross ratio dezelfde zal zijn als de cross ratio van eender welke perspectiefafbeelding van deze 4 punten.

Lemma 5.2.2. *Zij P_{1-4} vier collineaire punten $\in \mathbf{R^3}$. Zij $p^i_{1-4} \in \mathbf{R^2}$ hun respectievelijke perspectiefprojecties in afbeelding I_i. Dan geldt dat $\chi_{(P_{1-4})} = \chi_{(p^i_{1-4})}$*

Punten op een ray in 3D zijn geordend volgens de richting van deze ray. De ordening van ieder punt P_4 op de ray r kan ook worden beschreven aan de hand van de cross ratio door 3 punten P_{1-3} vast te kiezen en als vierde punt, het punt P_4 te kiezen. Op die manier zullen punten P_4 volgens een stijgende cross ratio geordend kunnen worden over de beeldstraal. Uit de bevindingen van lemma 5.2.2 kan deze ordening volledig in 2D over alle referentiebeelden berekend worden wanneer de projecties p^i_{1-3} van de 3 vast gekozen punten P_{1-3} gekend zijn in referentie-afbeelding I_i en als vierde punt een intersectiepunt p^i_4 te kiezen dat de projectie-afbeelding is van een punt P_4 op de ray. Dit wordt geïllustreerd in afbeelding 5.8 waar punten over alle referentiebeelden via een 2D cross ratioberekening kunnen worden geordend over een ray r. De vaste punten P_{1-3} kunnen vrij gekozen worden. Voor latere diepteberekening van een punt ten opzichte van een nieuw camerastandpunt C_{new} zal blijken dat het handig is om het projectiecentrum van C_{new} als vast punt te kiezen en de overige 2 vaste punten relatief hiervan te kiezen op een afstand die een veelvoud is van de genormaliseerde ray $r_{norm} = \frac{r}{|r|}$. Zij P_{1-3} vast gekozen punten op de ray r met

$$\begin{aligned} P_1 &= C_{new} - 2r_{norm} \\ P_2 &= C_{new} - r_{norm} \\ P_3 &= C_{new} \end{aligned} \tag{5.7}$$



Figuur 5.8: De cross ratio tussen 4 colineaire punten is gelijk aan de cross ratio van hun projectie-afbeeldingen. Het sorteren van deze projectie-afbeeldingen impliceert een projectie-invariante sortering over ray r.

Zij p_1^i, p_2^i, p_3^i de projectie-afbeeldingen van punten P_1, P_2, P_3 in beeld I_i, waarbij p_3^i de epipoolafbeelding van C_{new} zal zijn. De ordening van iedere contourintersectie p_4^i in iedere referentie-afbeelding i over ray r kan dan via de cross ratio in 2D worden berekend.

Selecteren van VH-intersecties

Voor het selecteren van VH-intersecties wordt in [Miller and Hilton [77]] aangenomen dat alle referentiecamera's het object, waarvoor de visual hull berekend wordt, zien. Per referentiebeeld worden gevonden intersecties van de epipolaire lijn naargelang hun cross ratio ordening genummerd startend van 1 tot het aantal intersecties die in dit referentiebeeld gevonden werden. Vermits de cross ratio de intersecties over de verschillende afbeelding in stijgende afstand tot het projectiecentrum van een nieuw camerastandpunt C_{new} zal ordenen, zal de eerste intersectie in ieder referentiebeeld corresponderen met een punt waarin de ray het object voor dit standpunt zal enteren. Bijgevolg zal de volgende intersectie het silhouet verlaten. Zo zal elke oneven intersectie in een referentiebeeld het silhouet enteren en elke even intersectie het silhouet in het referentiebeeld verlaten. Gegeven geordende intersecties in ieder referentiebeeld genummerd startend vanaf 1 tot het aantal intersecties in een referentiebeeld. Alle intersecties zijn via de cross ratio geordend en bevatten een teller naargelang het nummer van voorkomen van deze intersectie in een bepaald referentiebeeld. Die intersecties waarvoor de volgende stelling geldt begrenzen het visual hull volume:

Theorem 5.2.3 (VH intersectie selectie theorema). *Alle intersecties i uit de geordende verzameling waarbij de voorwaarde waar wordt dat voor alle referentie-afbeeldingen een oneven aantal silhouet-intersecties heeft plaatsgevonden, maken samen met de direct hieropvolgende $i+1$ intersectie waarvoor de conditie onwaar wordt deel uit van de visual hull.*

Intuïtief valt dit te verklaren door in te zien dat dit de maximale punten zijn waarvoor silhouet-consistentie zal gelden. Vanaf het eerste voorkomen i waarbij het punt op de ray in alle silhouetten zal projecteren zal er een silhouetovergang zijn van buiten een referentiesilhouet naar binnen alle referentiesilhouetten. Zodra er een even-telling vlak na punt i plaatsvindt zal er een overgang zijn binnen alle silhouetten naar buiten een silhouet in een referentiebeeld. Deze punten begrenzen de maximale silhouet-consistentie. Een voorbeeld van visual hull intersectie selectie is afgebeeld in figuur 5.9.

Diepteberekening voor visual hull intersecties

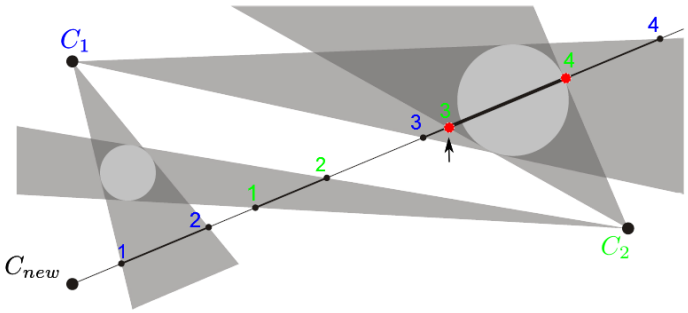
Iedere visual hull intersectie die voor een ray $r_{(x,y)}$ vanuit C_{new} werd geselecteerd wordt genummerd naargelang de volgorde van visual hull intersectie langs deze ray. Zij n dit nummer. Zij n_{max} het maximum aantal gevonden vh-intersectie langs een ray vanuit C_{new}. Dan wordt een multi-layered depth map gereconstrueerd bestaande uit n_{max} dieptemappen. Voor ieder visual hull punt v_n langs ray $r_{(x,y)}$ door pixel (x,y) in de dieptemap wordt de diepte berekend die overeenkomstig met het nummer n van het visual hull punt op de ray wordt opgeslagen in dieptemap n op positie (x,y). De diepteberekening van een 2D visual hull intersectie op $r_{(x,y)}$ ten opzichte van C_{new} kan op voorwaarde dat de vaste punten P_{1-3} zoals in 5.7 gekozen zijn uit de cross ratio als volgt worden berekend:

$$w_n = \frac{\chi_n}{\frac{1}{2} - \chi_n} \tag{5.8}$$

, waarbij w_n het gewicht van genormaliseerde ray $r_{norm_{(x,y)}}$ voor visual hull punt v_n voorstelt. De afleiding van deze formule is terug te vinden in [Miller and Hilton [77]]. De diepte d_n van v_n ten opzichte van C_{new} is dan gelijk aan:

$$d_n = w_n r_{norm_{(x,y)}} \tag{5.9}$$

De eerste dieptemap d_0 zal alle punten bevatten die rechtstreeks zichtbaar zijn vanuit C_{new} , i.e. ibvh. De lokaties (x,y) over de dieptemap waarvoor over alle dieptemappen heen geen vh-intersectie werd geselecteerd, maken geen deel uit van de visual hull.



Figuur 5.9: Voor iedere referentiecamera zijn de tellers, in corresponderende kleuren naargelang de volgorde van voorkomen van de intersectie in een referentiebeeld, weergegeven bij de intersectiepunten op een ray vertrekkend vanuit het nieuw camerastandpunt C_{new}. Vanaf de intersectie die door de pijl wordt aangeduid wordt theorema 5.2.1 waar en zullen de intersecties die deel uitmaken van de visual hull (in het rood aangeduid) geselecteerd worden. Bewerkte afbeelding uit [Miller and Hilton [77]].

5.2.2 Visual hull shading

Bij het inkleuren van de beeldgebaseerde visual hull *ibvh* is het gewenst dat hiervoor pixels uit referentiebeelden gebruikt waarvoor de visual hull punten v ook daadwerkelijk zichtbaar zijn. Dit wil dus zeggen dat v niet geoccludeerd wordt door andere visual hull punten in de richting naar het referentiebeeld. In het geval van occlusies zal immers de kleur in het referentiebeeld kunnen verschillen omdat in dit beeld eigenlijk een ander punt op de scene op geprojecteerd is. Daarom is het wenselijk om eerst een visibiliteitsberekening van iedere visual hull pixel in het gewenste beeld uit te voeren in ieder referentiebeeld. Vervolgens kan dan voor een visual hull pixel een kleur worden berekend op basis van een selectie van referentiebeelden die hetzelfde punt, waarvan de visual hull pixel de afbeelding is, zien.

Visibiliteit

Om na te gaan of een visual hull punt v geoccludeerd door andere visual hull punten wordt in de richting naar een referentiebeeld moet de volledige beeldgebaseerde visual hull ten opzichte van het nieuwe camerastandpunt C_{new} berekend zijn. De eerste dieptemap d_0 volstaat niet voor visibiliteitsberekening omdat deze geen occlusie-informatie van andere visual hull punten in de richting naar referentiecamera C_{ref} bevat. Deze informatie is noodzakelijk want de voorste punten ten opzichte van C_{new} kunnen door occlusies niet gelijk zijn aan de voorste visual hull punten zichtbaar vanuit een referentiecamera C_{ref}. Het berekenen van visibiliteitsinformatie voor een ray

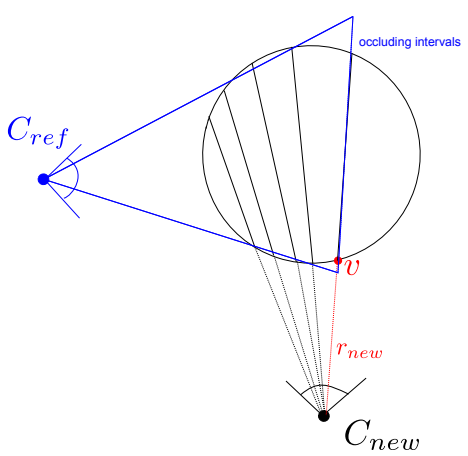
r_{new} vertrekkend vanuit C_{new} door de visual hull silhouet-afbeelding kan in de volgende stappen worden onderverdeeld:

- Occlusiemap berekening voor deze ray ten opzichte van ieder referentiebeeld I_{ref}.
- Visibibiliteitsberekening voor deze ray ten opzichte van ieder referentiebeeld I_{ref}.

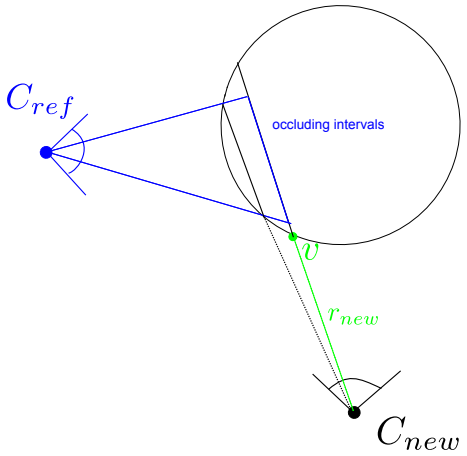
Per ray-occlusiemap berekening
De per ray-occlusiemap $\omega_{occl_{\rightarrow ref}}$ bevat alle visual hull intervallen in de richting naar referentiecamera C_{ref} die de visual hull intervallen op ray r_{new} door pixel p_{new} occluderen. In 3D kunnen deze worden berekend door vertrekkend vanuit C_{ref} alle visual hull intervallen te projecteren op de visual hull intervallen van ray r_{new} zoals weergegeven wordt in figuur 5.10. De afbeeldingen van de mogelijk visual hull intervallen die de ray kunnen occluderen bevinden zich in 2D op de epipolaire lijn $l_{ref \rightarrow new}$ tussen de epipoolafbeelding van C_{ref} op I_{new}, $e_{ref \rightarrow new}$ en p_{new} zoals geïllustreerd is in figuur 5.11. De iteratie-richting hangt hierbij af van het teken van $e_{ref \rightarrow new}$ zoals uitgelegd wordt in bijlage B over epipolaire geometrie. Volgens de correcte iteratierichting kunnen alle opgeslagen visual hull intervallen in de multi layered depth map per pixel k_{new} op de lijn $l_{ref \rightarrow new}$ geprojecteerd worden op de visual hull intervallen in de multi layered depth map op de positie van pixel p_{new}. Het resultaat van deze projectie zal een verzameling diepte-intervallen ten opzichte van pixel p_{new} zijn waarvoor p_{new} in de referentie-afbeelding geoccludeerd werd. Merk op dat hier in tegenstelling tot bij visual hull berekening het hier de referentiecamerastandpunten zijn C_{ref} die geprojecteerd worden in de nieuwe afbeelding I_{new}. Wanneer de per ray occlusiemap voor ieder ray r_{new} berekend is ten opzichte van een camera C_{ref} is de volledige occlusie van de visual hull punten naar de zichtbare punten vanuit c_{new} op een tot de pixel exacte manier beschreven. De totale output van de occlusieberekening bedraagt, gegeven n referentiecameras, n multi-layered depth images die de occlusies van de visual hull punten tussen referentiecamera en nieuwe camera beschrijven opgeslagen per pixel relatief ten opzichte van de nieuwe camera.

Per ray visibiliteitsberekening

Zij ι_{new} de visual hull intervallen op een ray r_{new}. Zij $\iota_{occl_{\rightarrow ref}}$ de geoccludeerde visual hull intervallen veroorzaakt door het object ten opzichte van een referentiecamera C_{ref}. De zichtbare intervallen $\iota_{vis_{\rightarrow ref}}$ ten opzichte van



(a) Visual hull intervallen langs de ray r_{new} worden door andere visual hull intervallen geoccludeerd in richting van C_{ref} aangeduid door de blauwe projectie van de occlusieintervallen op r_{new}. Punt v is niet zichtbaar in referentiebeeld C_{ref}.



(b) Het visual hull punt v langs de ray r_{new} wordt niet geoccludeerd door eerdere visual hull intervallen in richting van C_{ref} aangeduid door de blauwe projectie van de occlusieintervallen op r_{new}. Punt v is zichtbaar in referentiebeeld C_{ref}.

Figuur 5.10: Illustratie van zichtbare en niet-zichtbare visual hull punten in een referentiebeeld. De afbeelding toont het bovenaanzicht van een doorgesneden bol.

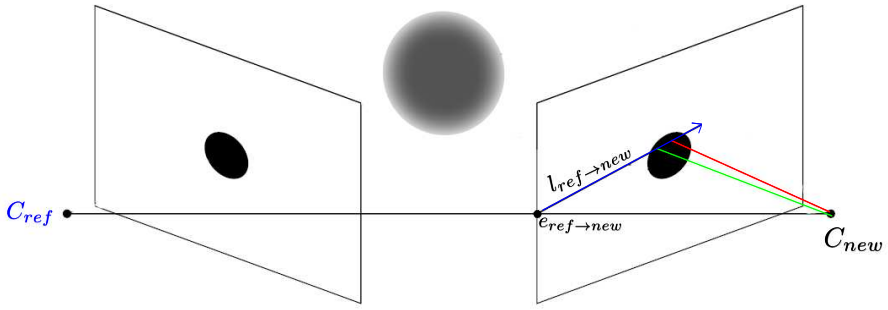
camera C_{ref} zijn dan die visual hull intervallen op r_{new} die niet-geoccludeerd zijn:

$$\iota_{vis_{\rightarrow ref}} = \iota_{new} - (\iota_{new} \cap \iota_{occl_{\rightarrow ref}}) \tag{5.10}$$

De visibiliteitsmappen kunnen gevisualiseerd worden door $\iota_{vis \rightarrow ref}$ van alle rays vanuit C_{new} in de corresponderende referentie-afbeelding te projecteren. In figuur 5.18 zijn enkele voorbeelden van visibiliteitsmappen weergegeven voor 6 referentieviews. De zichtbare visual hull punten in een referentiebeeld zijn in een visibiliteitsafbeelding opgeslagen met dichtste buurpixel afronding.

5.2.3 Inkleuring

Wanneer voor iedere referentiecamera de visibiliteit ten opzichte van de nieuwe camera berekend is, wordt voor iedere pixel in het nieuwe beeld de refentiecamera's geselecteerd waarvoor het voorste visual hull punt van deze pixel zichtbaar is. Vervolgens wordt in iedere kandidaat referentiecamera de ray berekend met dit visual hull punt. De referentiepixel voor inkleuring



Figuur 5.11: De pixels op de epipolaire lijn die loopt tot ray r_{new} zijn de afbeeldingen van de mogelijke occluderende rays. De projectie van de visual hull intervallen in multi layered depth map van iedere pixel op lijn $l_{ref \to new}$ op r_{new} beschrijven de occlusies op ray r_{new} voor camerastandpunt C_{ref}. De groene ray r_{new} is zichtbaar in de referentiecamera terwijl de rode ray r_{new} geoccludeerd wordt.

wordt gekozen in de referentiecamera waarvoor het hoekverschil tussen de ray in het referentiebeeld en de de ray in het nieuwe beeld het kleinste is. Deze pixel bevat de projectie-afbeelding van de belichting van de scene die over alle referentie-afbeeldingen het dichtst in de buurt komt van de gezochte belichtingsrichting.

5.3 Resultaten

Voor de resultaatbespreking zal de besproken beeldgebaseerde visual hull methode worden geëvalueerd over verschillende datasets met verschillende camerabaseline opstellingen en een verschillend aantal gebruikte camera's voor visual hull reconstructie. Alvorens tot deze evaluatie over te gaan worden eerst de gebruikte datasets besproken en worden de resultaten van de verschillende stappen in het visual hull algoritme op een voorbeeld dataset geïllustreerd.

5.3.1 Dataset

Voor evaluatie zal gebruikt gemaakt worden van volgende datasets:

- Beethoven dataset uit [Kolev and Cremers [46]]
  In de Beethoven dataset zijn camera's in wide baseline opstelling van ongeveer 40 graden rond een Beethoven model opgesteld.

- Buddha dataset
  De buddha dataset is een ground truth virtuele dataset die zelf werd gegenereerd door het virtueel schieten van afbeeldingen van een buddha geometrie in opengl. Ground truth segmentaties van het buddhamodel werden gegenereerd door de diffuse en speculaire belichtingseigenschappen van de geometrie volledig op de kleur wit in te stellen en de opengl achtergrond zwart te kiezen zodat in iedere camerastandpunt een binaire segmentatiemasker wordt afgebeeld wanneer voor ieder virtueel camerastandpunt opnieuw afbeeldingen worden geschoten. De ground truth dieptemappen werden gegenereerd door in ieder virtueel camerastandpunt de dieptebuffer offline te renderen naar een textuurafbeelding via het toevoegen van een depth attachment aan een opengl framebuffer object. De waarden van de dieptetextuur liggen tussen [0,1] waarbij 0 overeenkomt met de diepte van het gekozen near clipping plane en 1 overeenkomt met de gekozen diepte voor het far clipping plane. Vervolgens werden de dieptes getransformeerd en beschreven ten opzichte van het camerastandpunt zelf zodat diepte-error berekening tussen ground truth diepte en de visual hull dieptemap mogelijk is.

- Voetbaldataset
  Deze zelfgegenereerde dataset bevat 7 wide baseline beelden uit een opname van een voetbalwedstrijd om zo de visual hull techniek te evalueren over grote sportscenes. Beelden werden gesegmenteerd via achtergrondsubtractie van een beeld met de veldachtergrond.

5.3.2 Illustratie van verschillende stappen van het beeldgebaseerde visual hull algoritme

Aan de hand van de reële Beethoven dataset [Kolev and Cremers [46]] zullen de verschillende stappen van het beeldgebaseerde visual hull algoritme worden uitgelegd.

(a) C_{ref_0}

(b) C_{ref_1}

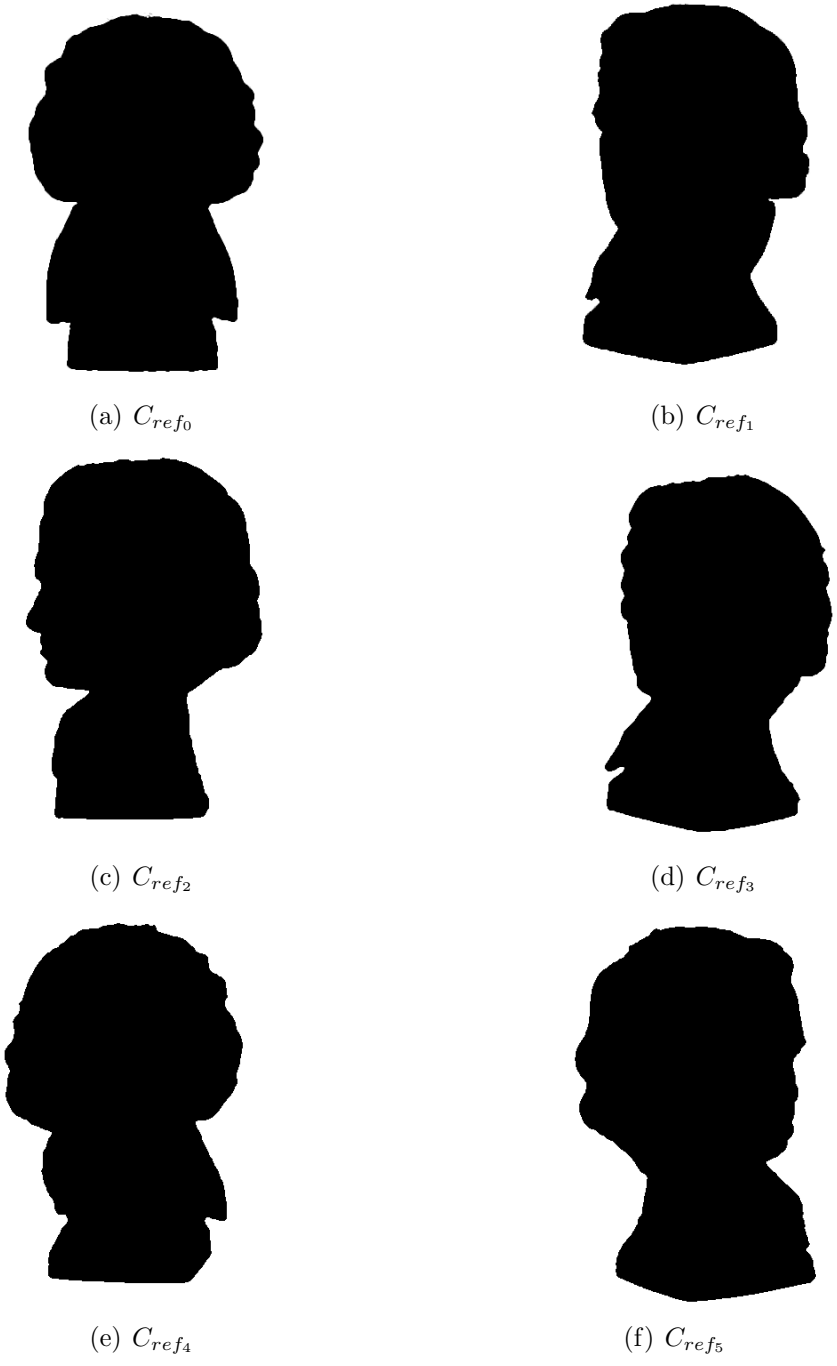
(c) C_{ref_2}

(d) C_{ref_3}

(e) C_{ref_4}

(f) C_{ref_5}

Figuur 5.12: Illustratie van de referentie-afbeeldingen.

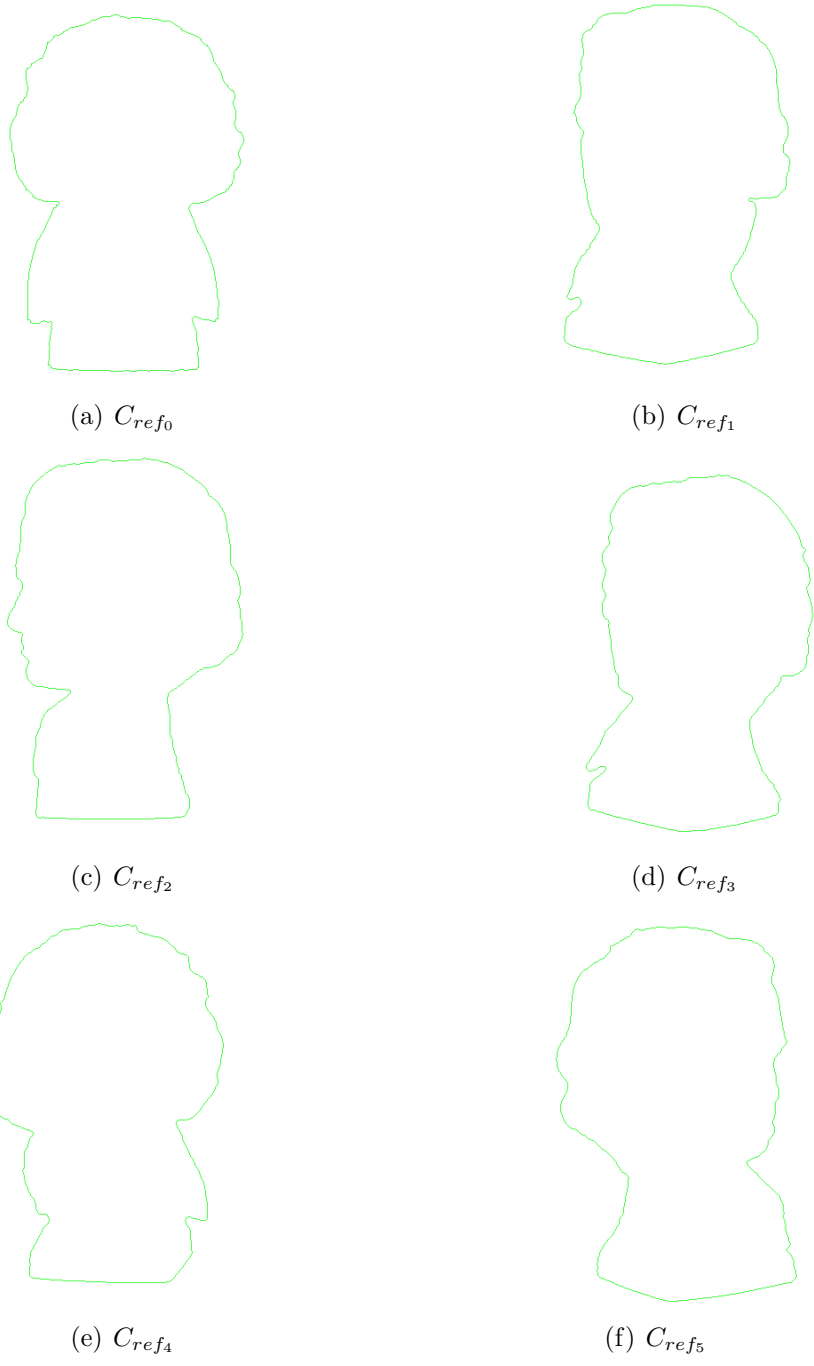


(a) C_{ref_0} (b) C_{ref_1}

(c) C_{ref_2} (d) C_{ref_3}

(e) C_{ref_4} (f) C_{ref_5}

Figuur 5.13: Illustratie van de segmentatie van het Beethoven model in de referentiebeelden.

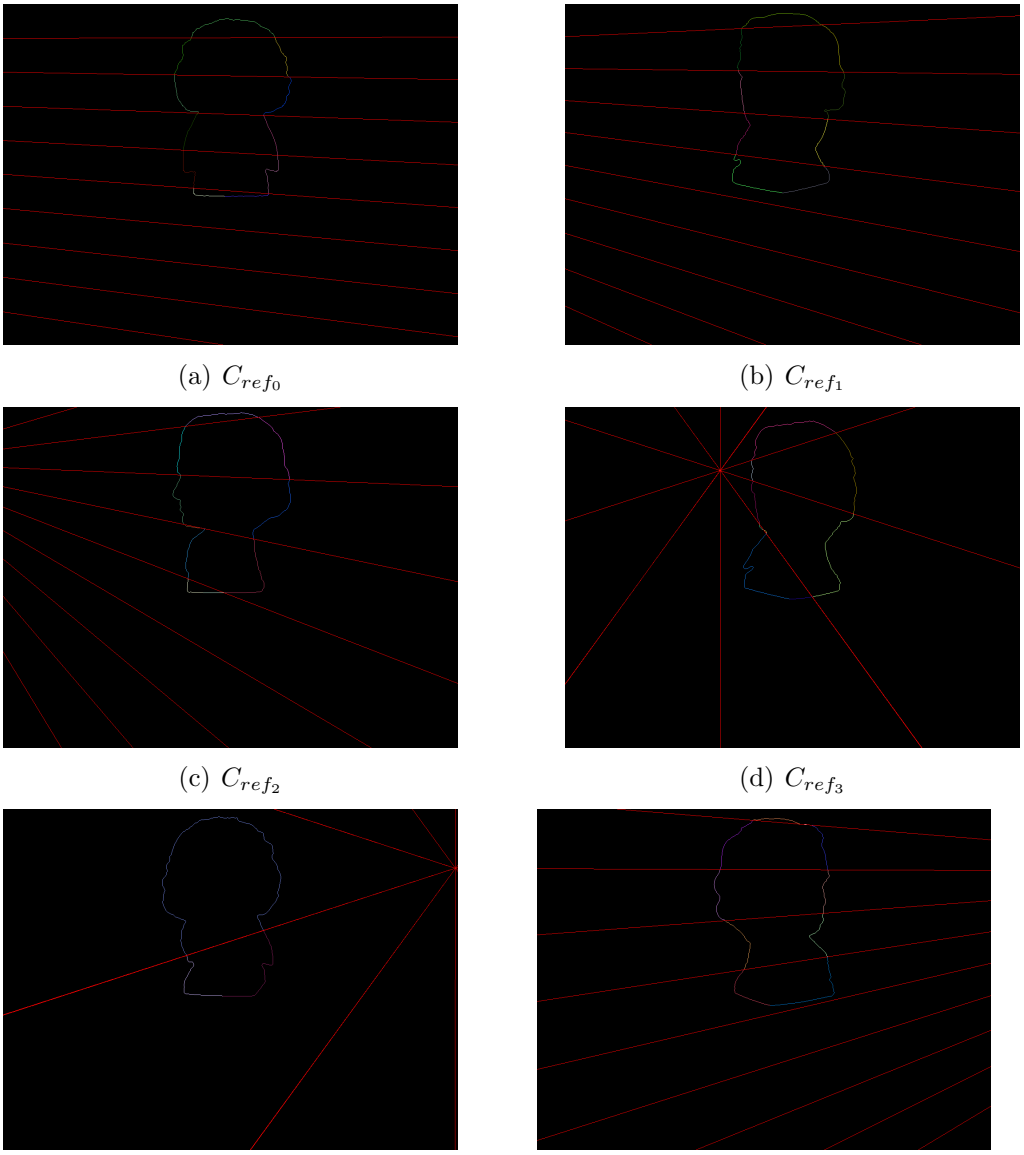


(a) C_{ref_0} (b) C_{ref_1}

(c) C_{ref_2} (d) C_{ref_3}

(e) C_{ref_4} (f) C_{ref_5}

Figuur 5.14: Illustratie van contourtracering volgens het algoritme van [Chang et al. [17]] op de segmentatie-afbeeldingen, aangeduid in het groen.

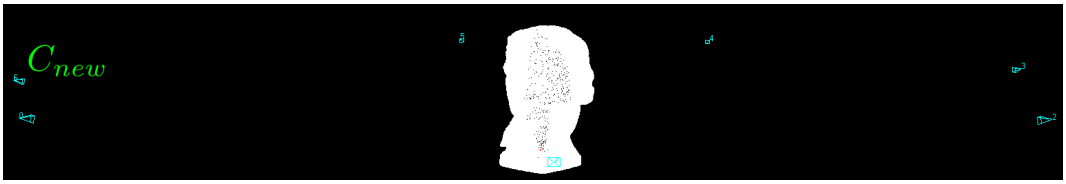


(a) C_{ref_0} (b) C_{ref_1}

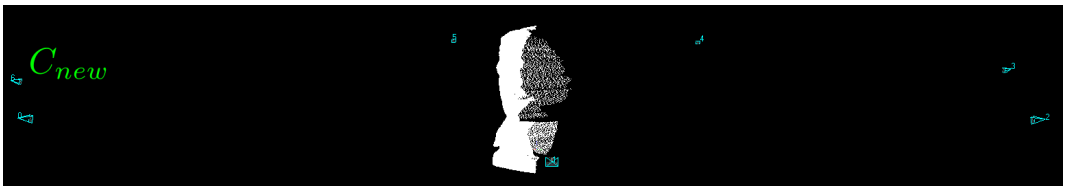
(c) C_{ref_2} (d) C_{ref_3}

(e) C_{ref_4} (f) C_{ref_5}

Figuur 5.15: Illustratie van bin partitionering, aangeduid door rode lijnen, van de object contouren in contoursegmenten per bin. Per bin heeft ieder contoursegment een verschillende kleur. In de bin waar de contourtracering van start gaat, kunnen 2 contoursegmenten voorkomen.



(a) Visual hull puntenwolk geconstrueerd uit de multi layered depth image (mldi) ten opzichte van C_{new}.



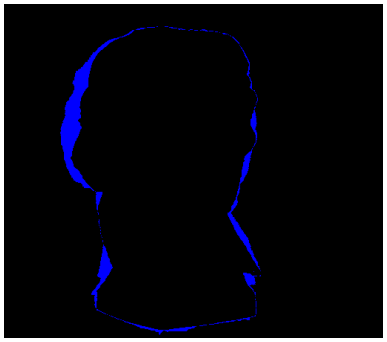
(b) Beeldgebaseeerde visual hull dieptemap (ibvh) ten opzichte van C_{new}. Dit is de eerste laag van multi layered depth image (mldi).

Figuur 5.16: Illustratie van de berekende visual hull bekomen uit 6 referentie-afbeeldingen C_{ref} voor het nieuw camerastandpunt C_{new}.

(a) Illustratie van de fout voor segmentatie tussen segmentatie-afbeelding 5.17(b) in C_{new} en het origineel in C_{new} getoond in figuur 5.19(a). In dit geval is er geen sprake van ondersegmentatie.

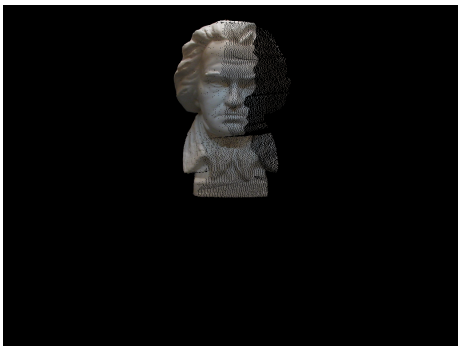
(b) Origineel silhouet in C_{new}.

(c) silhouet-afbeelding van de visual hull in C_{new}.

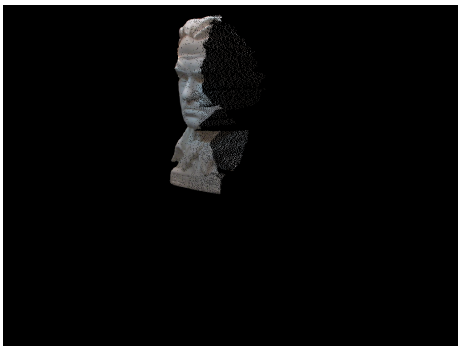


(d) Spatiale overestimatie fout van de visual hull weergegeven in het blauw. Dit zijn de pixels (x,y) waarvoor het visual hull silhouet een overschatting gaf van het silhouet-origineel.

Figuur 5.17: Illustratie van de visual hull bekomen uit 6 referentie-afbeeldingen C_{ref} voor het nieuw camerastandpunt C_{new}.



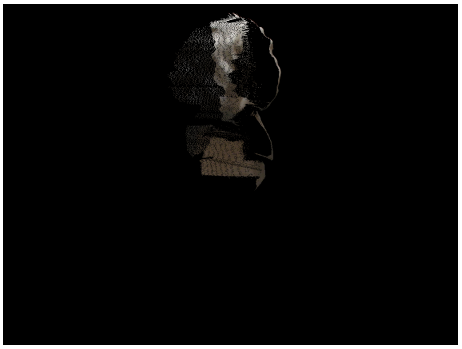
(a) C_{ref_0}



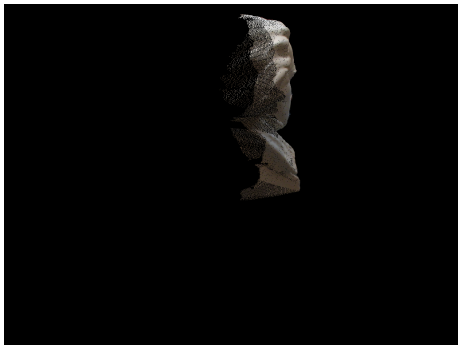
(b) C_{ref_1}

(c) C_{ref_2}

(d) C_{ref_3}



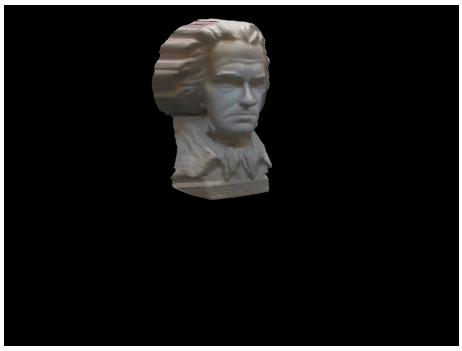
(e) C_{ref_4}



(f) C_{ref_5}

Figuur 5.18: Illustratie van visibiliteitsberekening van het nieuwe camera-standpunt C_{new} ten opzichte van iedere referentie-afbeelding $_{ref}$.

(a) Originele afbeelding in C_{new}



(b) Ingekleurd visual hull resultaat m.b.v. visibiliteitsberekening in figuur 5.18.

(c) Fout in intensiteit per pixel van het visual hull resultaat ten opzichte van het origineel.

Figuur 5.19: Visual hull shading berekend voor het nieuw camerastandpunt C_{new} uit 6 referentie-afbeeldingen C_{ref}.

5.3.3 Visual hull resultaten onder verschillende camerabaselines

In deze sectie wordt de invloed van de camerapositionering van referentiecamera's ten opzichte van een nieuw camerastandpunt C_{new} bestudeerd aan de hand van toetsing met de genereerde ground truth buddha dataset. Tijdens de evaluatie werden 2 verschillende camerabaseline opstellingen gebruikt:

- small camerabaseline
  Camera's zijn over een hoek van 8 graden van elkaar geplaatst.

- wide camerabaseline
  Camera's zijn over een hoek van 40 graden van elkaar geplaatst.

Voor de verschillende opstellingen werden de volgende fouten onderzocht:

- spatiale fout van het berekende visual hull silhouet in C_{new} ten opzichte van het ground truth silhouet. Deze wordt in iedere afbeelding aangeduid in het blauw. De waarde van e_s is een percentage dat ten opzichte van het aantal object pixels in C_{new} beschrijft hoeveel afwijking er was.

- intensiteitsafwijking per pixel van de ingekleurde visual hull afbeelding met de ground truth afbeelding als maat voor de correctheid van de visual hull shading.

- dieptefout van de visual hull in C_{new} ten opzichte van de ground truth diepte in in C_{new}. De dieptefouten die vermeld worden zijn uitgedrukt in eenheid van genormaliseerde ray afstand. De dieptefout wordt ook aan de hand van een kleurgradiënt van groen naar rood gevisualiseerd om weer te geven waar de afwijking ten opzichte van de ground truth diepte het grootst is (aangeduid in het rood). Deze inkleuring is relatief ten opzichte van de minimum diepte-afwijking gelijk aan 0 en de gevonden maximum diepte-afwijking $e_{d_{max}}$ na dieptevergelijking met de ground truth diepte. Verder wordt ook de gemiddelde diepte-afwijking $e_{d_{avg}}$ berekend.

- segmentatiefout. Voor een niet ground truth dataset zoals de voetbalscene, is het belangrijk te weten hoe betrouwbaar de segmentaties in een beeld zijn. Dit kan worden nagegaan door de segmentatie te maskeren op de afbeelding waardoor de segmentatiefout zichtbaar wordt.

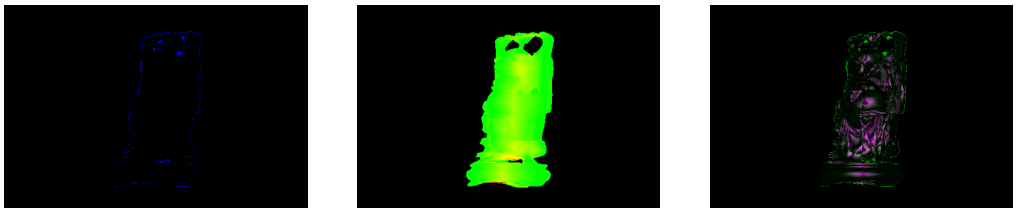
De referentiecamera's in iedere opstelling zijn allen geplaatst rond eenzelfde nieuw camerastandpunt C_{new} dat afgebeeld is in figuur 5.20.

(a) Small baseline: groene buddha. (b) Wide baseline: gele buddha.

Figuur 5.20: Ground truth afbeelding in het gemeenschappelijk nieuw camerastandpunt C_{new} over alle opstellingen.

Bespreking

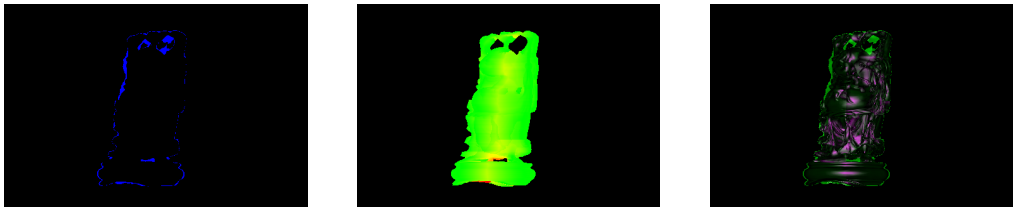
Uit de verschillende cameabaseline opstellingen kan worden afgeleid dat de spatiale error groter is bij een wide cameraopstelling met een brede baseline dan bij een smalle camera baseline. De dieptefoutbij een wide camera baseline is ten opzichte van de ground truth veel kleiner dan bij een small camerabaseline. Dit werd telkens ook bevestigd doordat de dieptefout steeds groter wanneer de camera's die het verst van het nieuw camerastandpunt verwijderd waren, werden genegeerd. Dit valt te verklaren door het feit dat camera's in een brede opstelling vooral de constraints zullen opleggen op de diepte van een visual hull kegel terwijl naburige camera's eerder constraints zullen opleggen in het spatiaal domein vermits ze daar het silhouet afbakenen. De aanwezige intensiteitsfout valt te verklaren door het feit dat het buddha object zeer speculair is en de inkleuring dus sterk zal afhangen van de gekozen richting.



(a) Spatiale fout. (e_s = 0.0251872)

(b) Dieptefout. ($e_{d_{max}}$ = 4.93074, $e_{d_{avg}}$ = 0.142342)

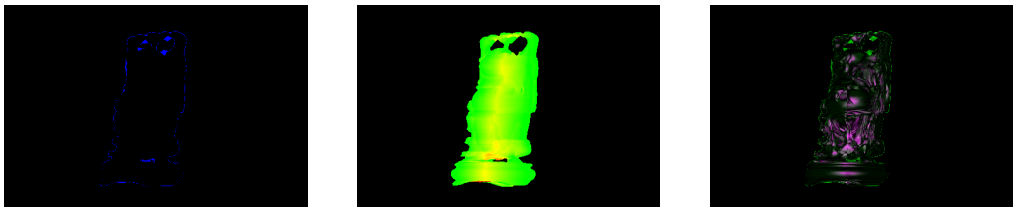
(c) Intensiteitsfout.



(d) Spatiale fout. (e_s = 0.0568839)

(e) Dieptefout. ($e_{d_{max}}$ = 5.06455, $e_{d_{avg}}$ = 0.142257)

(f) Intensiteitsfout.

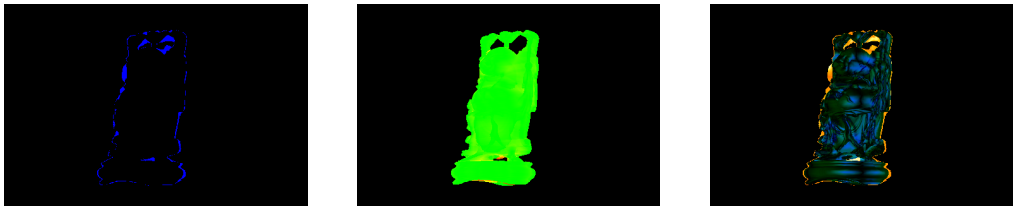


(g) Spatiale fout. (e_s = 0.029144)

(h) Dieptefout. ($e_{d_{max}}$ = 6.01814, $e_{d_{avg}}$ = 0.193614)

(i) Intensiteitsfout.

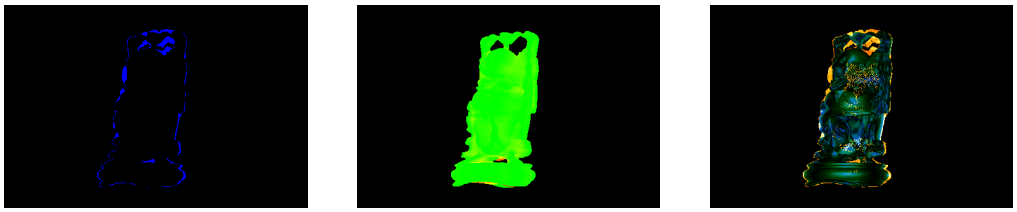
Figuur 5.21: Foutenanalyse van de visual hull in het nieuwe camerastandpunt onder een small camerabaseline van 7 referentiecamera's even verspreid rond C_{new} over een hoek van 8 graden. Voor figuren 5.21(a) - 5.21(c) werden alle 7 referentiecamera's gebruikt. Voor figuren 5.21(d) - 5.21(f) werden 5 camera's gebruikt waarbij de 2 dichtstbijzijnde camera's rond C_{new} zijn genegeerd. Voor figuren 5.21(g) - 5.21(i) werden 5 camera's gebruikt waarbij de 2 verst verwijderde referentiecamera's ten opzichte van C_{new} zijn genegeerd.



(a) Spatiale fout. ($e_s = 0.0635211$)

(b) Dieptefout. ($e_{d_{max}} = 3.24452, e_{d_{avg}} = 0.049793$)

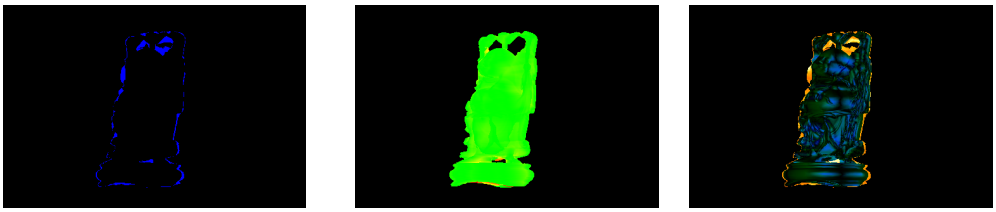
(c) Intensiteitsfout.



(d) Spatiale fout. ($e_s = 0.0735619$)

(e) Dieptefout. ($e_{d_{max}} = 3.24452, e_{d_{avg}} = 0.0598648$)

(f) Intensiteitsfout.



(g) Spatiale fout. ($e_s = 0.0712219$)

(h) Dieptefout. ($e_{d_{max}} = 3.24452, e_{d_{avg}} = 0.0626664$)

(i) Intensiteitsfout.

Figuur 5.22: Foutenanalyse van de visual hull in het nieuwe camerastandpunt onder een wide camerabaseline van 7 referentiecamera's even verspreid rond C_{new} over een hoek van 40 graden. De camera's beschrijven bijna de volledige cirkel rond het buddha object. Voor figuren 5.22(a) - 5.22(c) werden alle 7 referentiecamera's gebruikt. Voor figuren 5.22(d) - 5.22(f) werden 5 camera's gebruikt waarbij de 2 dichtstbijzijnde camera's rond C_{new} zijn genegeerd. Voor figuren 5.22(g) - 5.22(i) werden 5 camera's gebruikt waarbij de 2 verst verwijderde referentiecamera's ten opzichte van C_{new} zijn genegeerd.

Figuur 5.23: Cameraopstelling voor voetbaldataset.

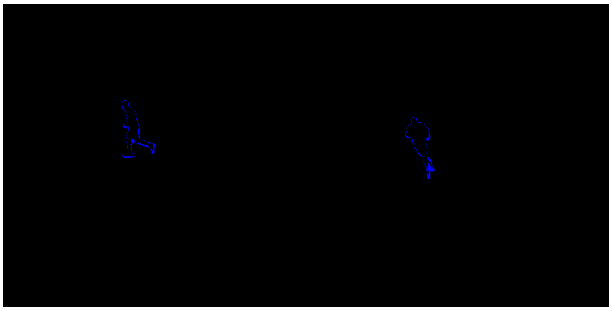
5.3.4 Visual hull resultaten in een sportscene

Voor de evaluatie van de visual hull over een sportscene werd een camera waarvoor het beeld gekend was, gebruikt als nieuw gewenst camerastandpunt om zo de berekende resultaten met de gegeven resultaten te vergelijken. De gebruikte cameraopstelling is weergegeven in figuur 5.23.

Bespreking

In de resultaten van de voetbalscene is er een minder grote intensiteitsfout dan bij de buddha dataset. In een voetbalscene zijn er weinig speculaire reflecties en bovendien is de spelersuitrusting vaak uniform gekleurd. De niet te overziene afwijking rond de 22 procent in spatiale error is grotendeels te verklaren door segmentatiefouten vooral in de buurt van de ledematen. Soms zijn de resultaten voor minder referentiecamera's op het gebied van spatiale fout beter dan wanneer meerdere referentiecamera's gebruikt werden. Dit kan door een aanzienlijke fout in calibratie of segmentatie verklaard worden. Voorgaande resultaten tonen aan dat bij een betrouwbare segmentatie zoals de ground truth segmentatie van het buddhabeeld de spatiale errors nog veel lager zullen zijn zelfs met weinig camera's en zeker wanneer ook meer naburige standpunten beschikbaar zijn. Gezien het beperkt aantal camera's dat voor het genereren van deze resultaten werd gebruikt en de geringe detailinformatie door de grote afstand kunnen we concluderen dat de besproken visual hull methode zeer goede resultaten geeft op basis van de beschikbare informatie.

(a) Fout in segmentatie.



(b) Spatiale fout. ($e_s = 0.255503$)

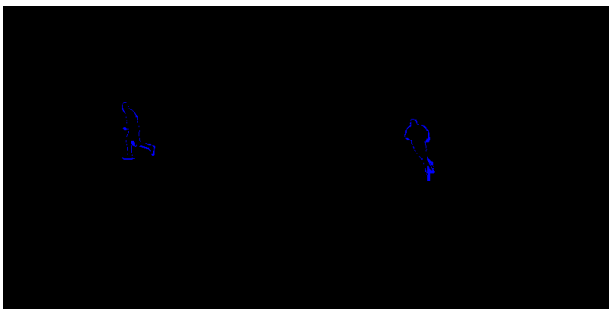
(c) Visual hull resultaat.

(d) Intensiteitsfout.

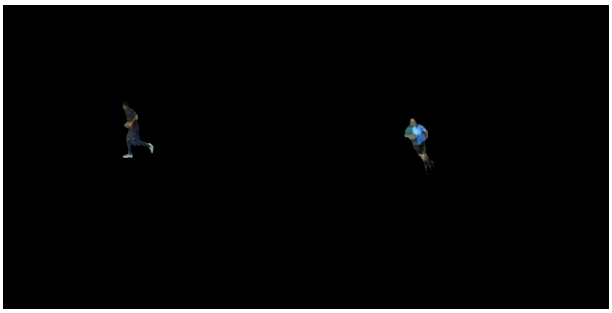
(e) Origineel.

Figuur 5.24: Visual hull berekening in C_{new} op basis van 6 referentie-afbeeldingen.

(a) Fout in segmentatie.



(b) Spatiale fout. ($e_s = 0.251187$)



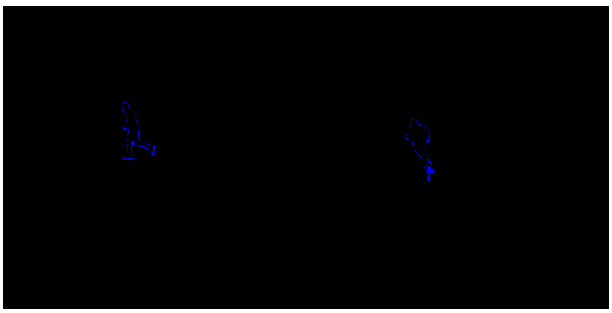
(c) Visual hull resultaat.

(d) Intensiteitsfout.

(e) Origineel.

Figuur 5.25: Visual hull berekening in C_{new} op basis van 4 referentie-afbeeldingen. De dichtstbijzijnde camera's aan weerszijden van C_{new} werden genegeerd.

(a) Fout in segmentatie.



(b) Spatiale fout. ($e_s = 0.220544$)

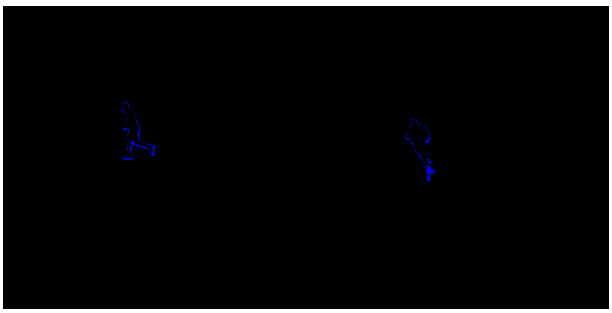
(c) Visual hull resultaat.

(d) Intensiteitsfout.

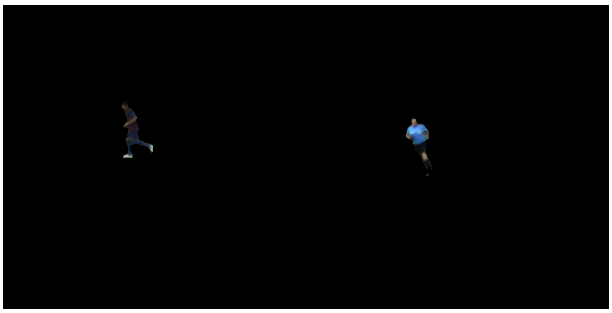
(e) Origineel.

Figuur 5.26: Visual hull berekening in C_{new} op basis van 4 referentie-afbeeldingen. De uiterste camera's aan weerszijden van C_{new} werden genegeerd.

(a) Fout in segmentatie.



(b) Spatiale fout. ($e_s = 0.223565$)



(c) Visual hull resultaat.

(d) Intensiteitsfout.

(e) Origineel.

Figuur 5.27: Visual hull berekening in C_{new} op basis van 4 referentie-afbeeldingen. De camera's 2 posities verder aan weerszijden van C_{new} werden genegeerd.

5.4 Verbeteringen

De besproken visual hull techniek neemt voor visual hull punt selectie aan dat alle camera's het object in de scene moeten zien. In een voetbalscene is dit niet steeds het geval door het grote oppervlak dat een voetbalveld bestrijkt en de beperkte kijkhoek van de camera's. Om deze techniek hierop toepasbaar te maken, zullen over de beelden de objecten moeten geïdentificeerd worden die zichtbaar zijn in een beeld. Dit kan bereikt worden door sparse features op gesegmenteerde objecten met elkaar te matchen. Eens deze identificatie gebeurd is, dient per object de referentiecamera's geselecteerd te worden voor visual hull reconstructie van dit object. Een andere manier om dit algoritme direct toepasbaar te maken op een voetbalscene is door beelden aan elkaar te stitchen [Szeliski [102]] om dan op verschillende posities rond het voetbalplein een panorama beeld opgebouwd uit verschillende camera's te verkrijgen. Deze panoramabeelden kunnen wel alle objecten in de scene zien waardoor visual hull reconstructie op deze panoramabeelden rechtstreeks kan worden uitgevoerd.

5.5 Conclusie

In dit hoofdstuk werd de beeldgebaseerde visual hull techniek uit [Miller and Hilton [77]] als kandidaattechniek voor beeldinterpolatie besproken en uitgebreid zodat ze onder verschillende mogelijke camerapositioneringen en randgevallen correct kan opereren. Uit de resultaatbespreking kunnen we afleiden dat de visual hull techniek goede resultaten geeft op grote scenes zoals voetbalvelden en dat afwijkingen vooral veroorzaakt worden door fouten in segmentatie. Resultaten tonen aan dat de spatiale error van de visual hull silhouet-afbeelding kan worden beperkt door camera's dichter bij elkaar te plaatsen en dat bredere camerabaselines significant de dieptefout reduceren. Een voorwaarde waaraan een scene moet voldoen opdat de besproken visual hull techniek correcte resultaten zou geven, is dat alle camera's het object moeten zien. Om hieraan tegemoet te komen werden een aantal mogelijke trajecten aangehaald die hier een oplossing voor bieden. Verder kan de visual hull benadering nog op andere manieren worden verfijnd om detailregio's zoals het gezicht en de ogen nauwkeuriger te reconstrueren. Op dit probleem wordt in het volgende hoofdstuk dieper op ingegaan.

Hoofdstuk 6

Edge matching

In dit hoofdstuk wordt een nieuwe edge matching methode besproken die op zoek gaat naar correspondenties tussen edges over verschillende afbeeldingen. Eerst zullen enkele karakteristieken van edges besproken worden samen met reeds bestaande edge matching technieken. Vervolgens zal onze methode worden toegelicht en geëvalueerd.

6.1 Edge definitie

Een edgepunt wordt in [Gonzalez and Woods [36]] gedefinieerd als een concept dat een maat is voor de lokale intensiteitsverandering in een punt en zijn omliggende punten. Een edge is een niet lege verzameling van aangrenzende edgepunten met een zeer gelijkaardige intensiteitsverandering. Ieder edgepunt heeft dus een bepaalde richting en sterkte. Zoals terecht opgemerkt wordt in [Kondermann [47]] is deze definitie niet eenduidig omdat ze afhangt van de semantische keuze van de threshold die we nemen alvorens we een pixelovergang met intensiteitsverschil als een edgepixel beschouwen en de mate van gelijkaardigheid die we gebruiken om edgepixels te groeperen tot een edge. In deze context wordt onder een edge een niet lege verzameling van aaneensluitende pixels verstaan die door een edgedetectiemethode na binaire thresholding als edge uitvoer zijn geselecteerd.

6.2 Multiview edge variaties

Overeenkomstige edges kunnen over meerdere beelden verschillen in:

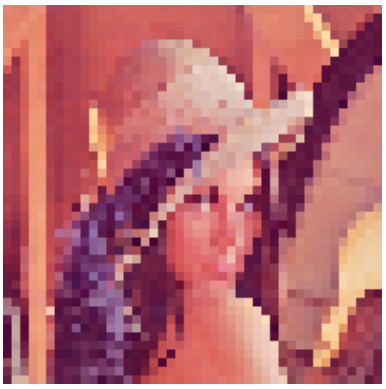
- oriëntatie: naargelang het verschil in cameraoriëntatie kan de kromming van overeenkomstige edges verschillen.
- zin: een edge kan ten opzichte van een corresponderende edge zodanig geroteerd zijn zodat de zin van de edge tijdens de edgedetectie verschilt ten opzichte van de zin waarin de corresponderende edge werd gedetecteerd.
- schalering: verschilt naargelang het verschil in afstand tussen de camerapositionering en de edge. Deze schalering kan niet-uniform zijn wanneer de diepte van edge tot het camerastandpunt varieert.
- translatie: de positie waarin de edge in het beeld voorkomt kan verschillen.
- volledigheid: de start- en eindpunten kunnen verschillen. De ene edge kan een subsegment zijn van de andere edge, bijvoorbeeld bij een occlusie.
- perspectieve vervorming: bijvoorbeeld een vierkantige edge zal in vooraanzicht hoeken afbeelden van 90 graden terwijl bij een rotatie van het vierkant met 45 graden de perspectiefafbeeldingen van de hoeken geen 90 graden meer zullen zijn. Zo kan een cirkel in zij-aanzicht afbeelden tot een lijn.

Ondanks deze veranderingen zal de ordening van edgepixels over verschillende camerastandpunten steeds dezelfde zijn omdat de image warping [McMillan [73]] operatie de volgorde behoudt [Meltzer and Soatto [74]].

6.3 Gerelateerd werk

In verhouding tot het detecteren en matchen van puntfeatures zoals FREAK [Ortiz [82]] en andere [Miksik and Mikolajczyk [76]] is er tot nu toe weinig aandacht besteed aan het matching probleem voor edges over verschillende afbeeldingen [Meltzer and Soatto [74]].
In [Mikolajczyk. et al. [75]] wordt edge matching gebruikt voor het herkennen van objecten. Detectie van SIFT puntfeatures [Lowe [66]] wordt veralgemeend tot schaalonafhankelijke detectie van puntfeatures op edges. Edges

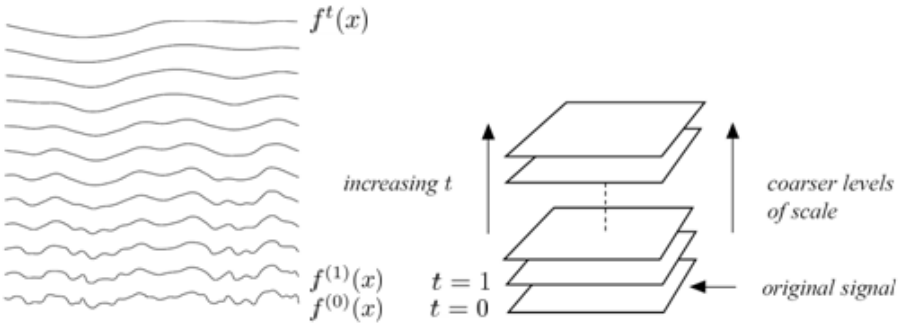
worden op meerdere schaalniveaus gedetecteerd door Canny edge detectie [Canny [14]] op ieder schaalniveau toe te passen. De detectie op meerdere schaalniveaus zou kunnen gesimuleerd worden door het verkleinen van een afbeelding via onderbemonstering. Dit kan ongewenste features introduceren die aanvankelijk niet in de scene aanwezig waren zoals weergegeven wordt in figuur 6.1.



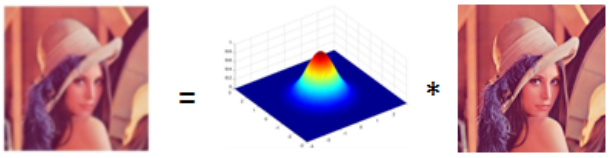
Figuur 6.1: Illustratie van een grotere schaal van de Lena-afbeelding via onderbemonstering. Deze illustratie toont aan dat de beschrijving van een schaalruimte door herbemonstering van een afbeelding extra, ongewenste features introduceert.

De afbeelding vervagen met een schaalgenormaliseerde Gaussiaanse filter [Lindeberg [63]] laat toe een schaalruimte te creëren zonder hierbij extra features te introduceren. In [Lindeberg [63]] wordt aangetoond dat een correcte simulatie van een toenemende schalering enkel met een Gaussiaanse kernel uitvoerbaar is. Intuïtief kan dit verklaard worden door in te zien dat dit de enige kernel is die richtingonafhankelijk (isotroop) en homogeen (bevat nuloplossing) is zodat geen extra structuren aan de afbeelding worden toegevoegd. Het schaalniveau is verwant met verschillende piekbreedten van de Gaussiaan die instelbaar zijn via de standaardafwijking σ. Deze genormaliseerde Gaussiaanse kernel toepassen op een rand simuleert de verschillende representaties van deze rand over verschillende schalen zoals weergegeven is in figuur 6.2.

Dit laat toe om een schaalruimte te creëren waarin we op zoek kunnen gaan naar schaalonafhankelijke features. Deze voorstelling van schaalruimte kan de posities van randen lichtjes doen verschuiven wanneer bijvoorbeeld vervaging over een rand wordt toegepast. Dit kan dus rotatie en positiebeschrijvingen bij descriptie lichtjes beïnvloeden. In [Lindeberg [63]] zijn een aantal alternatieve methode terug te vinden die het mogelijk maken om



(a) Illustratie van het voorkomen van een edge bij toenemende schaal. [Girod [35]]



$$f^{(t)}(x,y) = g^{(t)}(x,y) * f(x,y)$$

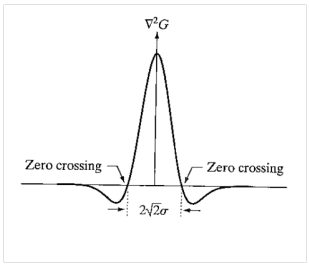
$$g^{(t)}(x,y) = \frac{1}{2\pi t} e^{(-\frac{x^2+y^2}{2t})}$$

(b) Convolutie van Lena-afbeelding met Gaussiaanse kernel g waarbij $t = \sigma^2$

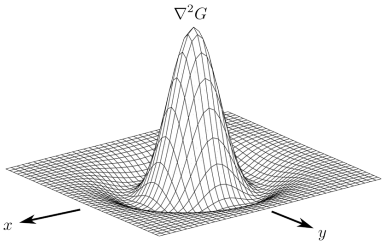
Figuur 6.2: De verandering van het voorkomen van een edge wanneer de schaal toeneemt, kan worden gesimuleerd door edge detectie toe te passen na Gaussiaanse smoothing.

schaalestimaties op een niet-lineaire manier te berekenen via adaptieve vervaging om deze verschuivingen tegen te gaan.
Om robuuste edgefeatures over verschillende schaalniveaus te detecteren, gaat [Mikolajczyk. et al. [75]] op zoek naar edge features waarvan de buurtregio op een of meerdere schalen erg descriptief is. Deze beschrijving kan schaalonafhankelijk gebeuren door de respons van een met schaal mee variërende signaalsveranderingsdetector te onderzoeken. Hiervoor zijn een variatie aan featuredetectoren beschikbaar [Grauman and Leibe [37]] naargelang het type feature waarin we in geïnteresseerd zijn. De schaleringen (bepaald door de standaardafwijking σ van de Gaussiaan) waarvoor er extreme signaalverandering optreedt, worden de karakteristieke schalen een edgepixel genoemd.
In [Mikolajczyk. et al. [75]] wordt een 2D Laplacian of Gaussian (LoG)-detector (ook wel mexican hat filter genoemd [Gonzalez and Woods [36]]) gebruikt.

De kernel van een LoG- filter, afgebeeld in figuur 6.3 bestaat uit circulaire



(a) LoG 1D-voorstelling.



(b) LoG 3D-voorstelling.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

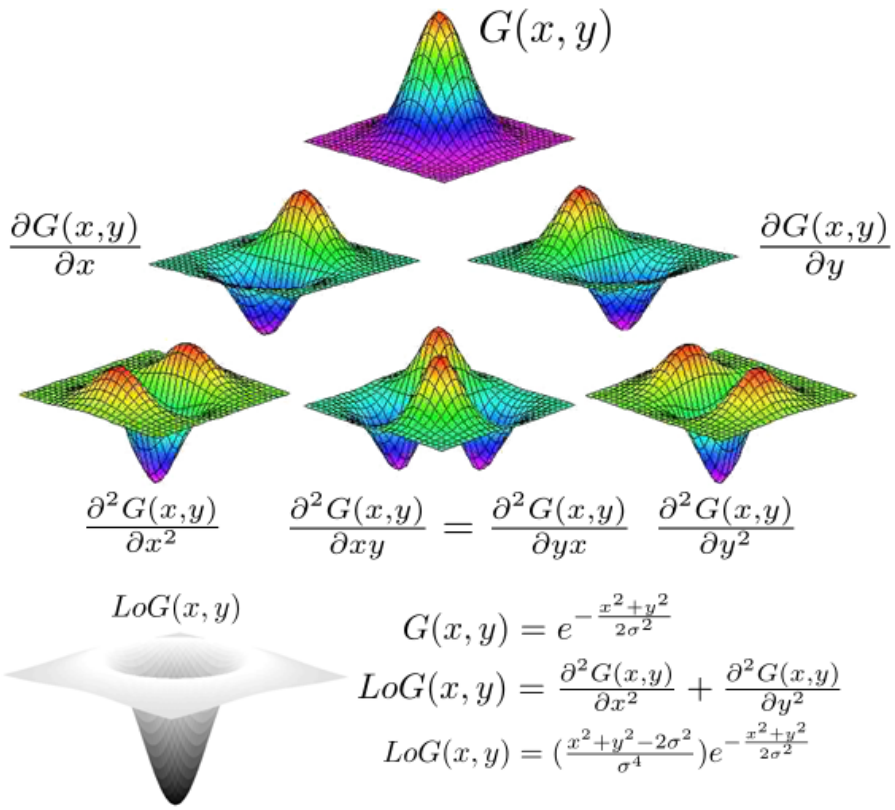
(c) Overeenkomstig filtermasker van LOG-functie in 2d.

Figuur 6.3: Laplaciaan van Gaussiaan filter(LOG-filter) [Gonzalez and Woods [36]]. De voorstellingen beelden -LoG(x,y) af. Bij LoG(x,y) wijst de piek naar beneden zoals in figuur 6.4.

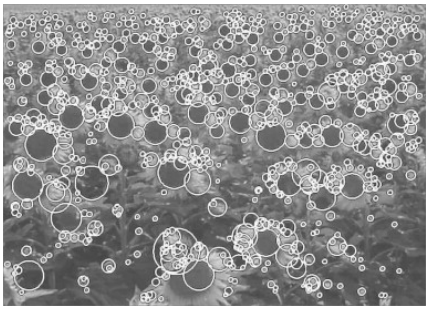
positieve waarden omringt door negatieve waarden, of omgekeerd. De LoG-filter is homogeen (som van de filter is 0) en isotroop. Ze zal het meeste response geven bij convolutie met een buurt met een gelijkaardige circulaire blob feature structuur. Een voorbeeld hiervan is weergegeven in figuur 6.5.

Over verschillende schaalniveaus heeft de LoG-filter de belangrijke eigenschap dat ze een extremum response zal geven wanneer ze op een andere edge botst. Dit zal zich voordoen wanneer het centrum van de LoG-filter de andere rand zal benaderen waarbij convolutie met het filtercentrum de resulterende signaalsterkte een enorme boost zal geven hetgeen zal leiden tot een extremum op een bepaalde schaal wanneer we de signaalwaarden van de filter over de schaal bepaald door σ plotten, zoals weergegeven in figuur 6.6.

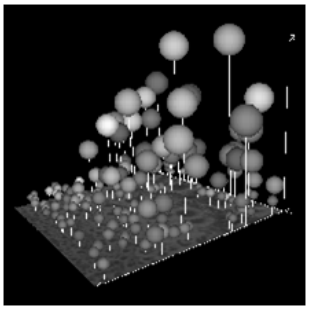
Edgepixels waarvoor extrema over schaal gedetecteerd zijn, worden geaccepteerd, andere edgepixels met een homogene buurt zullen geen extrema vormen over schaal en worden, omdat ze niet descriptief genoeg zijn, genegeerd. In het geval van een step edge, afgebeeld in figuur 6.6(b), waar er een per pixel zuivere edge overgang zal de schaalparameter waarvoor de Laplaciaanoperator een extremum respons (nulpunten van eerste afgeleide van de signaalfunctie naar schaal) geeft, overeenkomen met de afstand tot de naburige edge [Lindeberg [64]]. Edge-overgangen zullen zuiver zijn na het binaire resultaat van Canny edgedetectie. Hierop een LoG-filter toepassen zal bij robuuste features over schaal een extremumrespons op een bepaalde schaal geven waarbij de schaal gelijk zal zijn aan de afstand de edgefeature en naburige edgepixels. Deze afstand vormt tevens een schaalonafhankelijke, betrouwbare regiogrootte voor het bouwen van de descriptievector om later features te matchen. De schaalfactor tussen overeenkomstige features op verschillende schalen kan berekend worden uit de gekozen standaardafwijkingen



Figuur 6.4: Voorstelling van de afgeleiden van de Gaussiaanse functie $G(x, y)$. Op de eerste rij zijn de eerste orde partiële afgeleiden, i.e. de elementen van de Jacobiaanse matrix $J(G(x, y))$, getoond. Op de tweede rij zijn de tweede orde partiële afgeleiden, i.e. de elementen van de Hessiaanse matrix $H(G(x, y))$, getoond. De Laplaciaan van een Gaussiaan is dan gelijk aan de som van de elementen van de tweede orde partiële afgeleiden tegenover iedere onafhankelijke variabele, hetgeen gelijk is aan de som van elementen op de hoofddiagonaal van de Hessiaanse matrix. Bewerking van [de Boer et al. [25]].

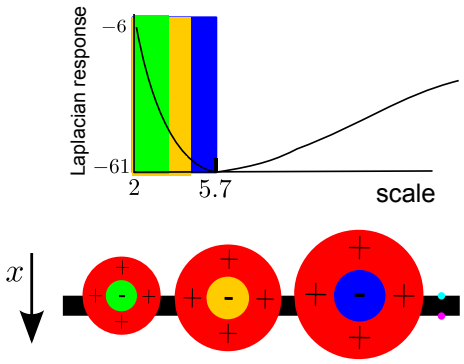


(a) Grootte van een zonnebloemblob wordt bepaald door de extrema over schaalruimte. [Girod [35]]

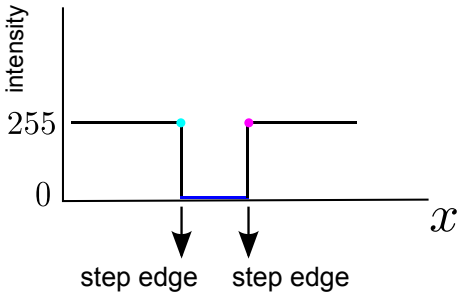


(b) LoG-responses per zonnebloemblob. Spatiale posities van de zonnebloemen zijn af te lezen in het liggend vlak, de responses zijn af te lezen in het rechtopstaand vlak.

Figuur 6.5: Illustratie van blobvormige response van een LoG-filter op een afbeelding bestaande uit zonnebloemen. [Lindeberg [64]]



(a) Laplaciaanfilters op verschillende schalen toegepast op een lijn.

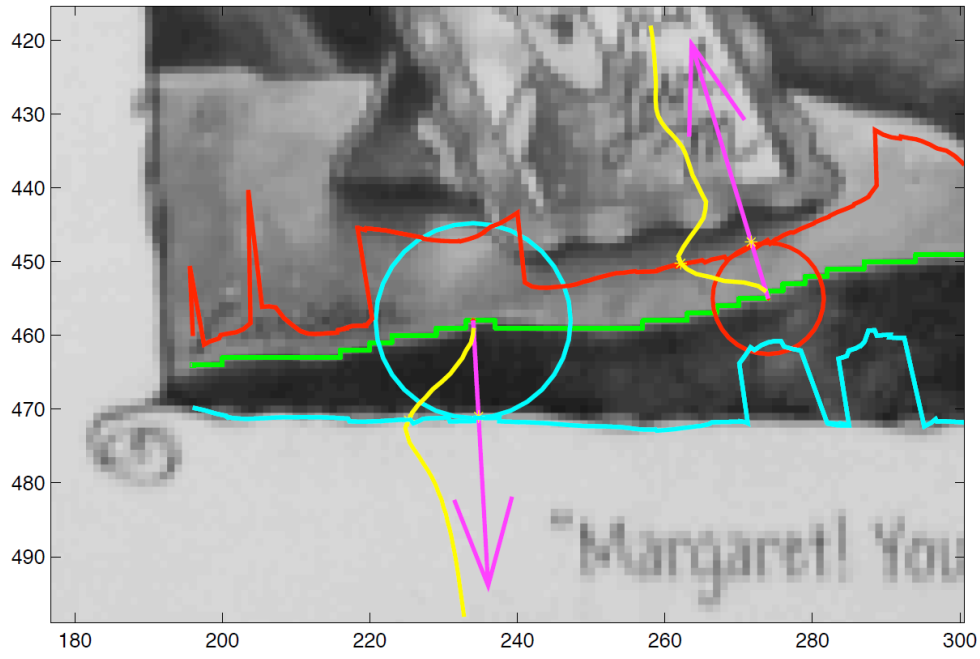


(b) Intensiteitsprofiel van de lijn uit 6.6(a) met ideale step edges.

Figuur 6.6: Illustratie van de Laplaciaan responsie over schaal. De schaal van de extremum respons komt overeen met de afstand naar de volgende rand (aangeduid in het blauw).

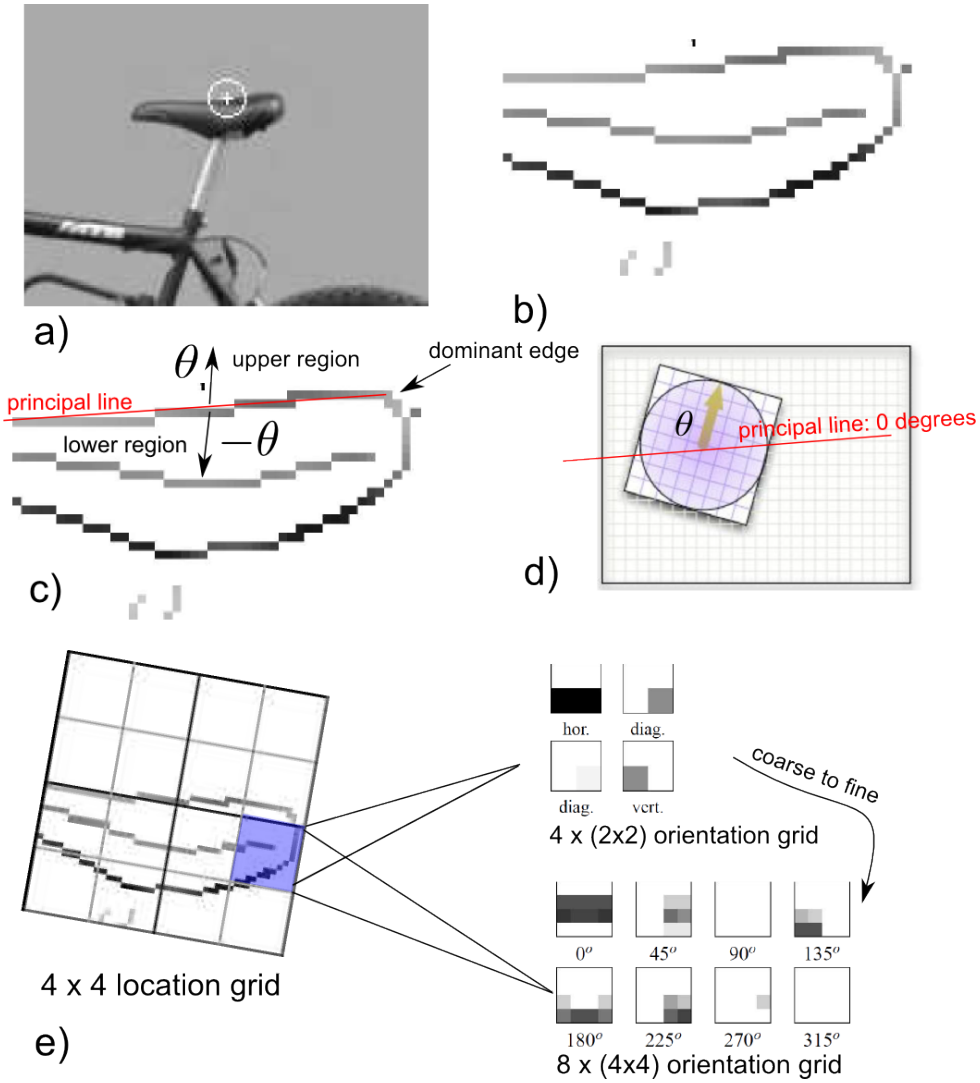
σ en σ' en is gelijk aan $\frac{\sigma}{\sigma'}$ [Grauman and Leibe [37]]. In [Mikolajczyk. et al. [75]] wordt opgemerkt dat bij het beschrijven van de features a.h.v. naburige edges enige voorzichtheid nodig is. Edges uit de voorgrond kunnen beschreven worden ten opzichte van edges in de achtergrond die vanuit verschillende standpunten kunnen verschillen. Om de invloed van de achtergrond op het resultaat te reduceren wordt een buurtregio in 2 opgesplitst aan de hand van de dominante edge. Descriptors worden dan voor iedere subregio berekend waardoor 1 subregio met grote waarschijnlijkheid descripties ten opzichte van de voorgrond zal bevatten. Indien het object eenvoudig te segmenteren is van de achtergrond kan ook geopteerd worden voor een voorgrondsegmentatie en een edgedetectie op de voorgrond zodat achtergrondedges niet in rekening worden gebracht. Descriptieve eigenschappen van een edgepixel, zoals positie en oriëntatie, worden op een rotatie-invariante manier opgeslagen door deze relatief te berekenen ten opzichte van de principiële lijn door de geselecteerde dominante edge in de buurt. De featureregio's rond een edgefeature over schaal worden beschreven door een 3D histogram histo(x,y,θ) van relatieve posities (x,y) en oriëntaties θ van edgepixels in deze regio ten opzichte van een lokatiegridoverlay en oriëntatiegridoverlay zoals afgebeeld in figuur 6.8. De grootte van de grids wordt stilaan verfijnd om matching op een coarse to fine manier door te voeren. Iedere histogram bin waartoe een edgepixel behoort, wordt verhoogd naargelang de met een Gaussiaan gewogen gradiëntwaarde van de edgepixel. Significante edgepixels met een grote gradiëntvector zullen zo meer bijdragen en er zal meer gewicht worden gegeven aan edgepixels in het centrum van de regio vermits deze minder positionele verschuivingen ondergaan [Grauman and Leibe [37]]. Eens edgefeatures gedetecteerd en beschreven zijn in de te onderzoeken afbeelding en in een template afbeelding, worden de descriptievectoren genormaliseerd met hun lengte. Dit maakt de descriptie robuuster voor belichtingsverandering over verschillende afbeeldingen omdat er ook voor gelijkaardige edges een verschil in intensiteitsverandering, gradiënt, kan zijn wanneer de belichting tussen afbeeldingen verschilt. De normalisatie maakt deze intensiteitsveranderingen relatief ten opzichte van het totale intensiteitsverschil in deze regio. Voor edge matching wordt op een coarse-to-fine manier de Euclidische afstand tussen genormaliseerde descriptievectoren berekend waarvan de beschrijvingen meer en meer verfijnen door het kiezen van een fijnere grid naarmate het matchingproces evolueert.
In [Meltzer and Soatto [74]] worden edges gedetecteerd via de multi-scale edgedetectiemethode van [Lindeberg [65]]. Hierbij worden edges over schaal voorgesteld als edgevoxels, die de representatie zijn van een edgepixel met een bepaalde positie en schaal. Tussen edgevoxels worden oriëntaties berekend en wanneer er voxels met een nabijgelegen positie en oriëntatiestructuur

voorkomen worden ze met elkaar gelinkt. Iedere edgevoxel die deel uitmaakt van een edge krijgt zo een positie, schaal, oriëntatie en edgesterkte toegekend. Net als in [Mikolajczyk. et al. [75]] worden er dan robuuste edgefeatures over schaal gedetecteerd. In plaats van descriptievectoren van edgefeatures onafhankelijk met elkaar te matchen, wordt er een linkmechanisme en volgorde op het voorkomen van een edgefeature in de edge gedefinieerd om zo een descriptievector voor een volledige edge te construeren. Voor iedere edgepixel wordt de LoG-filter respons over schaal onderzocht om voor iedere edgepixel een karakteristieke schaal te ontdekken. De Laplaciaanrespons wordt zo voor iedere edgepixel langs de 2 kanten van een edge berekend door de LoG-filter enkel op regio's uit een bepaalde kant toe te passen. Zo wordt een schaalgrens langs iedere kant van de edge berekend die aan weerskanten een descriptieregio afbakent zoals te zien is in figuur 6.7.



Figuur 6.7: Illustratie van de edgedescriptieregio's gebruikt door de edge matching techniek uit [Meltzer and Soatto [74]]. De edge is in het groen afgebeeld en de afbakeningen van de descriptieregio's aan weerszijden van de edge zijn aangeduid door rood - en cyaankleurige enveloppes. Deze enveloppes zijn berekend door per edgepixel het snijpunt van de descriptieregio van een regiokant met de edgenormaal in de edgepixel, die naar deze regiokant wijst, te berekenen. De straal van de descriptiecirkel, wordt zoals in [Mikolajczyk. et al. [75]] bepaald door de descriptieve schaal in een edge pixel, i.e. een extremum in Laplaciaan respons. Boven de edgenormalen, aangeduid via pijlen, zijn de Laplaciaan responses over verschillende schalen weergegeven. Die schaal waarvoor er zich een extremum voordoet, wordt als straal gekozen. Figuur uit [Meltzer and Soatto [74]].

Daarna worden robuuste edge ankerpunten geselecteerd door per kant die edgepixels te selecteren waarvoor in schaalenveloppe een schaalextremum wordt bereikt. De regioafbakening kan vervormen over verschillende camera-standpunten, maar de extrema ervan zouden stabiel moeten blijven [Meltzer and Soatto [74]]. Deze ankerpunten worden vervolgens in volgorde met elkaar gelinkt en voor ieder ankerpunt wordt voor iedere kant een descriptie gevormd. Deze descriptie gebeurt aan de hand van een oriëntatiehistogram over schaal die relatief ten opzichte van de edgepixel de positie en oriëntatie van de gradiëntvectoren op een oriëntatie-onafhankelijke manier beschrijft. In tegenstelling tot [Mikolajczyk. et al. [75]] worden gradiëntvectoren van pixels, gelegen binnen de cirkelstraal proportioneel aan een schaal, in rekening gebracht behalve de pixels op de karakterstieke schaal. Deze worden bewust genegeerd omdat deze een staprand beschrijven waarbij de gradiënt aan weerskanten van de rand constant zal zijn en voor de descriptie geen extra distinctieve informatie biedt. Op die manier wordt er voor weerskanten van een edge een discrete, geordende lijst van beschrijvingen van ankerpunten opgeslagen. Per ankerpunt bevatten deze descripties, op een schaal- en rotatie-onafhankelijke manier, een beschrijving van de relatieve positie en rotatie van dit ankerpunt op ten opzichte van de buurtregio. De lengte van de descriptievector varieert naarmate het aantal ankerpunten op een edge werd geselecteerd.
Tot slot worden descriptievectoren met elkaar gematcht via DTW (dynamic time warping) [Treichler and Agee [104]]. Deze methode laat toe om via dynamisch programmeren [Dasgupta et al. [24]] de optimale aligning en een matchingsscore te vinden tussen 2 discrete signalen waarvan de samplingrate en de lengte mogen verschillen en ook deelsignalen mogen ontbreken. Een speciaal geval van DTW is SW, het algoritme van [Smith and Waterman [99]], [Chen et al. [19]]. Dit algoritme wordt vaak gebruikt voor het aligneren van proteïnesequenties en is iets efficiënter dan DTW. Ten koste van de snelheidswinst is, in tegenstelling tot DTW, een vast alfabet van mogelijke letters die in een sequentie kunnen voorkomen en een vooraf gedefinieerde matchingscore voor ieder letterpaar met een strafscore voor ontbrekende letters, vereist. Alvorens SW toe te passen, dienen de histogrambeschrijvingen omgezet te worden naar verschillende letters uit een vast alfabet. Via het berekenen van de Euclidische afstand tussen histogramdescripties kan een voorafgedefinieerde matchingscore tussen letterparen worden berekend. Beide methodes kunnen op een elegante manier met het volledigheidsprobleem voor edges overweg zodat ook edgesegmenten met elkaar gematched kunnen worden. Aan het verschil in zin dat kan ontstaan tussen edges kan tegemoetgekomen worden door in omgekeerde volgorde met elkaar te linken ankerpunten en een matchingscore voor deze omgekeerde volgorde te berekenen.



Figuur 6.8: Illustratie van descriptie die in [Mikolajczyk. et al. [75]] wordt opgebouwd. In figuur a is een edge feature, met een grootte waarvoor er een extremum respons van de Laplaciaan respons over schaal was, weergegeven. Figuur b toont de gradiëntwaarden van alle edgepixels. Figuur c toont de regioverdeling die gebeurt op basis van de lijn die de meest dominante edge benadert. Hierbij wordt de voorgrond gescheiden van de achtergrond en de meest dominante richting θ berekend die de referentiehoek van de gridoverlay bepaalt, afgebeeld in figuur d. Voor beide regio's wordt vervolgens een descriptievector opgebouwd. Figuur e toont de constructie van de 3D descriptievector (2D positie, 1D rotatie) op basis van gridoverlays over de gradiëntwaarden, waarbij per gridlokatie een histogram wordt opgebouwd. Ieder histogram op een bepaalde lokatie wordt verhoogd met iedere gradiëntwaarde die op deze lokatie werd teruggevonden. Alvorens te verhogen worden de gradiëntwaarden eerst nog met een Gaussiaans window over de grid gewogen. Bewerkte figuur uit [Mikolajczyk. et al. [75]].

6.4 Methode

Nieuw ten opzichte van voorgaand besproken edge matchingmethoden is dat onze methode een edgedescriptie construeert waarbij enkel de geometrische informatie van de edge zelf voor matching gebruikt wordt. Onze techniek kan in 3 stappen worden onderverdeeld:

- edge segmentatie
- edge descriptie
- edge matching

Verderop in deze sectie worden de details achter deze stappen verder ontleed en wordt er besproken hoe deze methodes overweg kunnen met de verschillende mutaties van een edge over verschillende beelden.

6.4.1 Edge segmentatie

Edge segmentatie wordt toegepast om de robuustheid van het matchen van corresponderende edges met variantie in volledigheid te matchen te vergroten en hierbij betrouwbare schaalfactoren af te leiden die kunnen inschatten hoe edges ten opzichte van elkaar geschaleerd zijn.

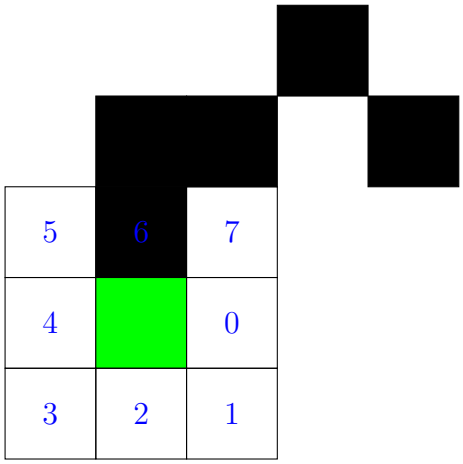
Partitie in edgesegmenten

Om een edge op te splitsen in verschillende edgesegmenten wordt er op zoek gegaan naar die pixels die in de buurt liggen van lokale extrema waarbij de edge, over een grid gezien, een richtingsverandering heeft ondergaan. Beschouw de 3 x 3 grid mask uit tabel 6.1.

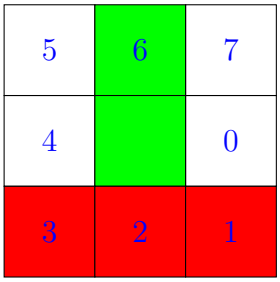
Deze matrix wordt als een overlay op iedere edgepixel geplaatst. De positie van de buurtpixel relatief ten opzichte van de huidige pixel waarover de grid geplaatst is, kleurt het vakje van het vierkant op deze positie. Dit is weergegeven in figuur 6.9 waar in de eerste stap positie 6 wordt ingekleurd in het groen. Een richtingsverandering wordt waargenomen wanneer er een

5	6	7
4		0
3	2	1

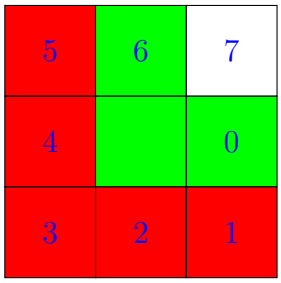
Tabel 6.1: Richtingsveranderingsmasker.



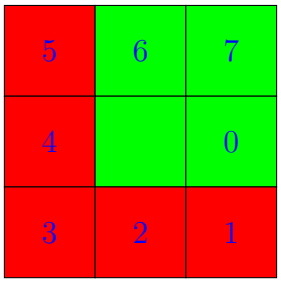
(a) Startsituatie.



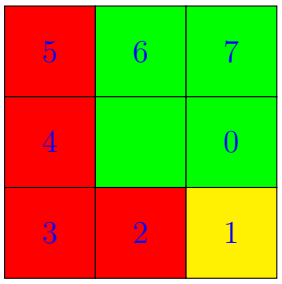
(b) Gridmask bij step = 1 en step = 2.



(c) Gridmask bij step = 3.



(d) Gridmask bij step = 4. Stopcondities op 1,3,5 zijn eerder al gemarkeerd.



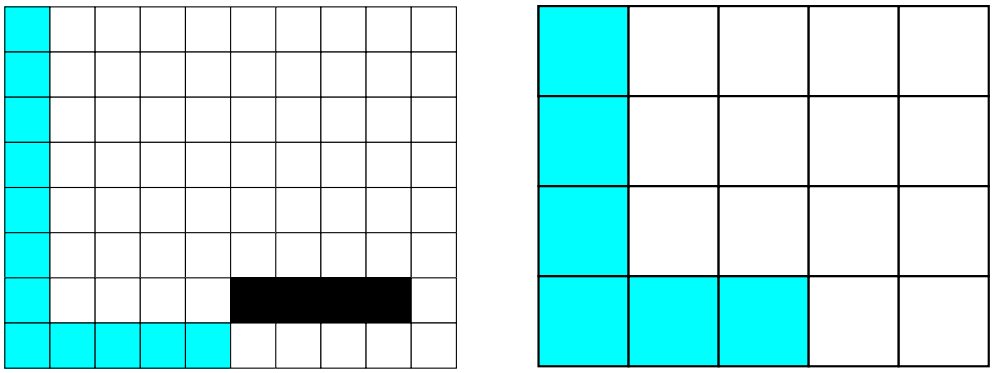
(e) Gridmask bij step = 5.

Figuur 6.9: Illustratie van monotoniciteistalgoritme voor het detecteren van richtingsveranderingen. De iteratie van de 3 x 3 grid over de zwarte edgepixels wordt over de verschillende figuren geïllustreerd. Hun relatieve posities ten opzichte van de grid kleuren een gridpositie, aangeduid in het groen en corresponderende stopcondities aangeduid in het rood. Een schending van monotoniciteit vindt plaats in 6.9(e) waar de pixel rechtsonder in het geel een eerder gemarkeerde stopconditie heeft getriggerd.

lokale rotatie niet gelijk aan 90 graden heeft plaatsgevonden waarbij een pixel wordt bezocht waarvan de nieuw ingeslagen richting indruist tegen de al afgelegde richting. Telkens wanneer een hoek van de grid gekleurd wordt, worden de overige hoeken geselecteerd als stopconditie voor een richting. Bij inkleuring van een niet-hoek wordt telkens de tegenoverstaande zijde als stopconditie ingekleurd. Stopcondities zijn in de richtingveranderingsgrid in het rood weergegeven. Wanneer bij een volgende inkleuring een index van de grid gekleurd wordt die reeds als stopconditie werd gemarkeerd, is er een richtingsverandering opgetreden. De richtingsveranderingsgrid wordt opnieuw gereset en de nieuwe richting wordt op dezelfde manier getracked. Een voorbeeldoperatie van dit algoritme is weergegeven in figuur 6.9.

Schaalfactor

Onder assumptie van een uniforme schaalverandering kan uit de verhouding van de meest gedetailleerde edge e_{max}, i.e. de edge met grootste lengte ten opzichte van de minst gedetailleerde edge e_{min}, i.e. de kortste edge, de schaalverandering s van e naar e' worden berekend: $\lambda = \frac{|e_{max}|}{|e_{min}|}$. Aan de hand van de uniforme schaalberekening λ kan voor een gekozen descriptiestap t voor e_{max}, de overeenkomstige descriptiestap t' van e_{min} berekend worden: $t' = \lambda t$.
Wanneer de edges e en e' in volledigheid verschillen gaat deze berekening niet op zoals afgebeeld wordt in figuur 6.10. Om dit te verhelpen kunnen we edge matching in een bottom-up manier toepassen waarbij edges verdeeld worden in edgesegmenten en we per edgesegment een schaalfactor λ berekenen hetgeen meer robuustheid zal geven omdat edge onvolledigheden enkel de schaalschatting binnen een onvolledig segment zullen beïnvloeden.



(a) Voorbeeld van een edge. (b) Onvolledige corresponderende edge.

Figuur 6.10: Illustratie van verhoging van robuustheid voor het schatten van schaalfactoren wanneer edges gesegmenteerd worden. De globale schaalfactor $\lambda = \frac{16}{6} = 2.6$. De schaalfactor enkel berekend ten opzichte van de cyaan gekleurde segmenten $\lambda_0 = \frac{12}{6} = 2$, hetgeen overeenkomt met de doorgevoerde schalering tussen beide segmenten.

6.4.2 Edge descriptie

Edges worden beschreven door de lokatie van iedere edgepixel relatief te beschrijven ten opzichte van de edge zelf. Zij e een edge, waarbij $e[i]$ een edgepixel op plaats i voorstelt. Per edge-pixel wordt de signed, relatieve hoek θ tussen 2 2D-vectoren v_b, waarbij b staat voor backward, en v_f, waarbij f staat voor forward, berekend:

$$v_b = e[i] - e[i-t] \tag{6.1}$$

$$v_f = e[i+t] - e[i] \tag{6.2}$$

$$\begin{aligned}
\theta &= \tan^{-1}\left(\frac{|v_b \times v_f|}{v_b \cdot v_f}\right) \\
\theta &= \tan^{-1}\left(\frac{v_{b\perp} \cdot v_f}{v_b \cdot v_f}\right) \\
\theta &= \tan^{-1}\left(\frac{v_b[0]v_f[1] - v_b[1]v_f[0]}{v_b[0]v_f[0] + v_b[1]v_f[1]}\right)
\end{aligned} \tag{6.3}$$

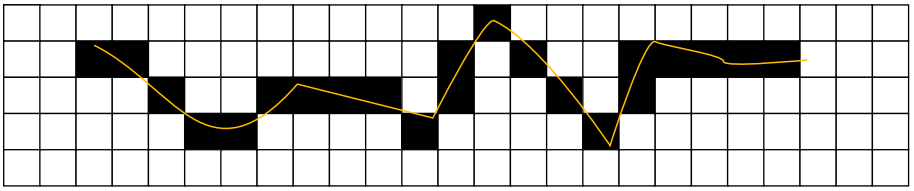
waarbij $\cdot$ het dot product, $_\perp\cdot$ het perpendicular dot product en $\times$ het kruisproduct tussen 2 vectoren voorstellen. Array indices 0 en 1 spreken respectievelijk de x- en y-waarde van de vector aan. In figuur 6.11 wordt een illustratie van deze beschrijving getoond.

Om voor alle pixels een descriptie te kunnen berekenen, moeten de randuiteinden met t pixels worden uitgebreid. Hiervoor wordt de extensievector tussen het randuiteinde en de edgepixel die t pixels voor dit randuiteinde ligt berekend. Zij v_{e_b} de back vectorextensie van edge e, zij $n = |e| - 1$, v_{e_f} de front vectorextensie van edge e:

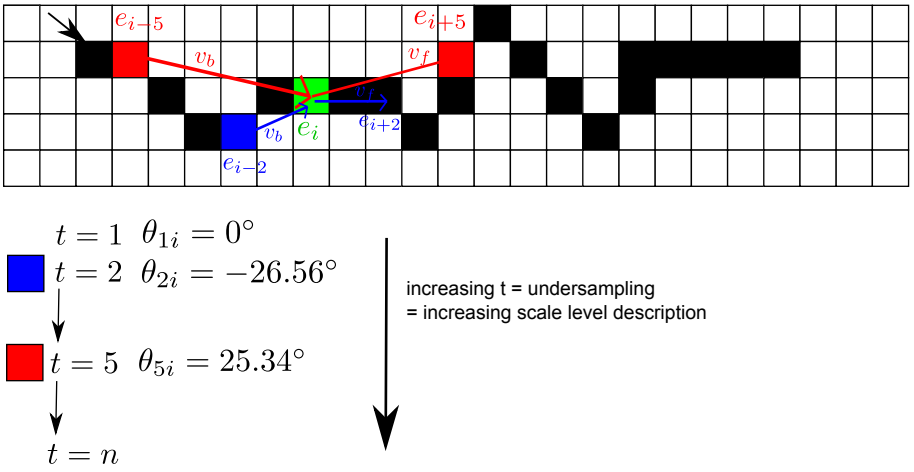
$$v_{e_b} = e[n] - e[n-t] \tag{6.4}$$

$$v_{e_f} = e[0] - e[t] \tag{6.5}$$

Eens deze vectoren berekend zijn, wordt de edge langs beide kanten verlengd door t pixels van de Bresenham-lijn [Joy [40]] die de vector benadert in te kleuren zoals afgebeeld in figuur 6.12 en vervolgens voor descriptie te gebruiken.

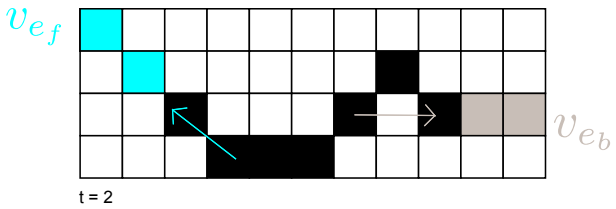


(a) Discretisatie van edge over grid.

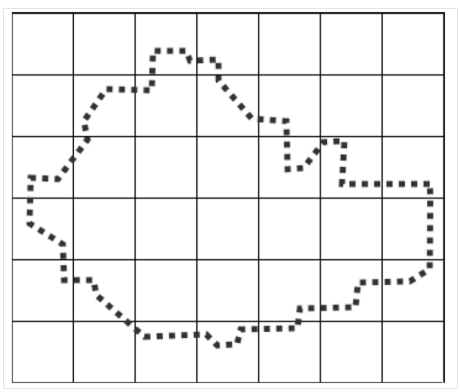


(b) Illustratie van edge descriptie voor edgepixel e_i met descriptiestappen $t = 2$ en $t = 5$.

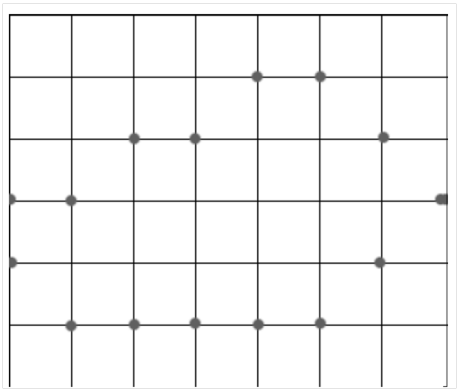
Figuur 6.11: Edge descriptie over meerdere schaalniveaus.



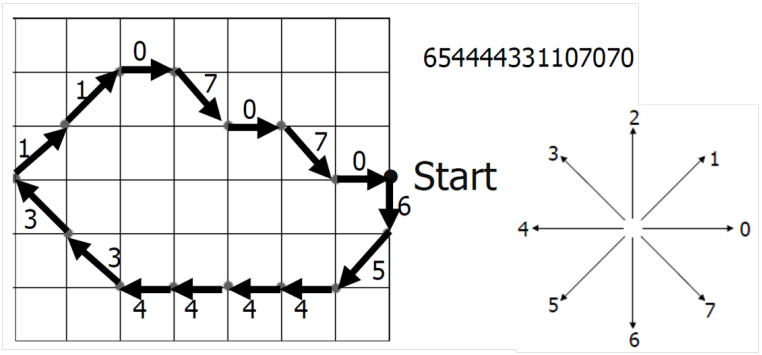
Figuur 6.12: Illustratie van edge extensievectoren voor een descriptiestap $t = 2$.



(a) Onderbemonstering van edge over grid om lokale ruisinvloeden te beperken.

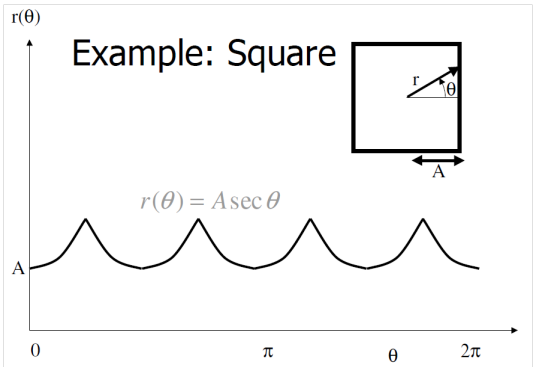


(b) Geselecteerde samples.



(c) Freeman chain code van edge.

Figuur 6.13: Illustratie van Freeman chain code van een edge. Bron: [Lilienthal [62]].



Figuur 6.14: Illustratie van hoek-afstand encodering voor vierkant tegenover centraal punt. Bron: [Lilienthal [62]].

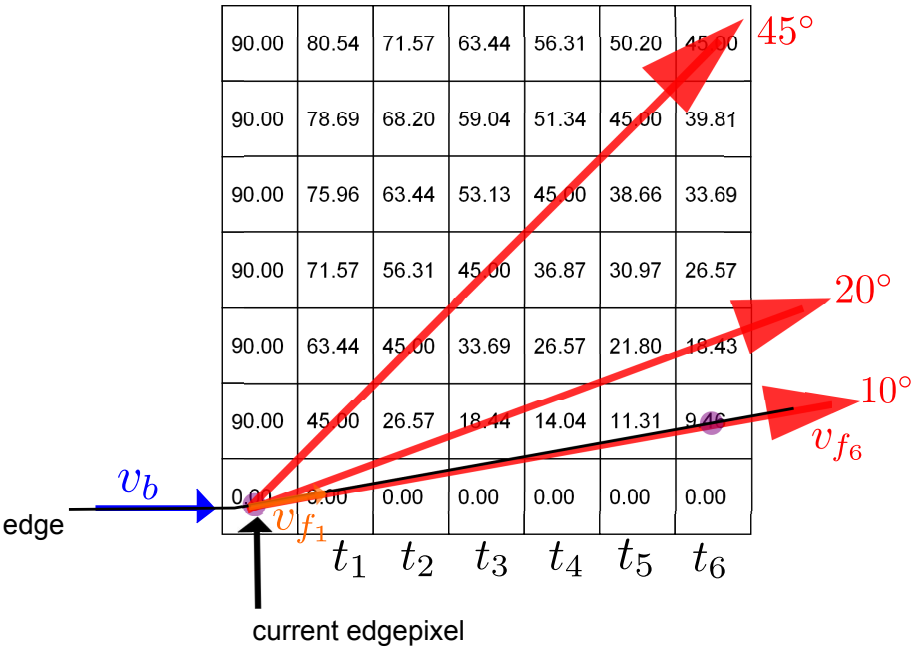
Invariante betrekkingen

In tegenstelling tot Freeman chaincodes [Gonzalez and Woods [36]], geïllustreerd in afbeelding 6.13, is deze edge descriptie in het geval waar t = 1, tot op pixelhoekresolutie rotatie-invariant doordat relatieve hoeken worden opgeslagen. Andere methoden zoals hoek over afstand geometrie-encoderingen slaan de hoek van de raaklijn per edgepixels, of de hoek die een edgepixel maakt met een referentielijn door een centraal punt, zoals weergegeven in figuur 6.14, op naargelang de afstand. Wanneer een betrouwbaar centrumpunt voor een object kan worden berekend, is deze methode rotatie-invariant. In het geval van onvolledige edges en zal de onvolledigheid de lokatie van een betrouwbaar punt kunnen beïnvloeden. Daarnaast is de descriptie aan de hand van raaklijnen erg lokaal en gevoelig voor ruis.
Onze methode biedt ruimte voor descriptie op meerdere detailniveaus. Naast lokale connectiviteit waarbij t = 1, kan de connectiviteit op meerder schaalniveaus in rekening worden gebracht door t te laten variëren. Op die manier bekomen we een rotatie- en schaalinvariante matrix van edgedescripties, afgebeeld in figuur 6.15.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} e_0 & e_1 & \dots & e_n \end{array} \\ \begin{pmatrix} \theta_{10} & \theta_{11} & \dots & \theta_{1n} \\ \theta_{20} & \theta_{21} & \dots & \theta_{2n} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \theta_{n0} & \theta_{n1} & \dots & \theta_{nn} \end{pmatrix} \end{array}$$

Figuur 6.15: Descriptiematrix.

De kolommen van deze matrix beschrijven de graduele hoekevolutie van een pixel ten opzichte van zijn buren en de rijen beschrijven de descriptie op een bepaald connectiviteitsniveau. De nauwkeurigheid van de rotatie-invariante beschrijving is afhankelijk van de afstanden die de vectoren v_b en v_f over de grid afleggen. Deze afstanden bepalen de resolutie van de hoek die deze vectoren over een grid kunnen beschrijven. Voor kleine descriptiestappen zullen de vectoren weinig pixels over de grid bestrijken en zal de hoekresolutie laag zijn. Voor grotere descriptiestappen zal de nauwkeurigheid van de rotatiebeschrijving toenemen. Dit is geïllustreerd in figuur 6.16. Zininvariantie kan eenvoudig worden bekomen door de descriptie van een edge ook in omgekeerde richting op te slaan. Zij d(e_i) de descriptie d van



Figuur 6.16: Illustratie van de invloed van de descriptiestap op de nauwkeurigheid van de rotatiebeschrijving. Voor t = 1, kan in het slechtste geval een rotatie worden gedetecteerd over een hoek van 45 graden. Stel bijvoorbeeld dat een diagonaal aangrenzende pixel rechtsboven instaat voor het bereik van waarden tussen 22.5 en 67.5 graden en dat een lijn met de huidige pixel een hoek van 22.5 graden maakt. Hierbij wordt de diagonaalpixel dus ingekleurd. Op pixelniveau zal een rotatie van de lijn in tegenwijzerzin zich pas uiten wanneer deze lijn over een hoek groter of gelijk aan 45 graden geroteerd wordt waarbij vanaf 67.5 graden de verticale buur boven de huidige pixel zal worden ingekleurd. Zoals te zien is in de figuur zal een rand die over 10 graden geroteerd is, voor de huidige pixel met een descriptiestap van t = 1 een descriptie geven gelijk aan 0 graden. Voor t = 6 kan de rotatie tot op iets minder dan 10 graden nauwkeurig worden beschreven. Voor t=6 kan de rotatiebeschrijving van een corresponderende edge dus maximaal tot 10 graden afwijken, voor t = 1 bedraagt de maximaal toegestane afwijking 45 graden.

edge e_i, waarbij e een willekeurige edge uit image i voorstelt. Zininvariantie kan dan op dezelfde wijze als in [Meltzer and Soatto [74]] worden bekomen door d(e_i) zowel te matchen met d(e_j) uit image j en de omgekeerde descriptie $d_{reverse}(e_j)$.

6.4.3 Edge matching

Voor het zoeken van edgecorrespondenties worden edges bottom-up met elkaar vergeleken. Op het laagste niveau worden eerst de edgesegmenten met elkaar gematched. In een later stadium worden de aangrenzende segmenten waarvoor een match gevonden werd met elkaar gegroepeerd en verlengd zolang de condities voor matching standhouden.

Segment matching

Alvorens descriptievectoren van 2 edgesegmenten met elkaar te kunnen vergelijken wordt eerst de overeenkomstige descriptiestap t' van het minst gedetailleerde segment e_{min} uit de vast gekozen descriptiesprong t van het meest gedetailleerde segment e_{max} berekend aan de hand van de schaalfactor λ: $t' = \lambda t$. Dit resulteert in 2 descriptievectoren $d_t(i_k)$ en $d_{t'}(j_m)$ met een gelijkaardig descriptieniveau. De variabelen i en j stellen de variërende integer pixelindex van een verzameling edgepixels voor in respectievelijke afbeeldingen k en m. Als t' een floating point getal is, zijn subpixel descripties vereist. Zij f het fractionair gedeelte van t' en zij t'_l = floor(t') en t'_u = ceil(t') de naburige integer waarden van t' met $d_{t'_l}(j_m)$ en $d_{t'_u}(j_m)$ de descriptiehoeken op sprongen t'_l en t'_u van de edgepixel index j ten opzichte van zijn buurpixels $j \pm t'_l$ en $j \pm t'_u$. Dan wordt via lineaire interpolatie de descriptie over sprongen t' als volgt berekend:

$$d_{t'}(j_m) = (1 - f)d_{t'_l}(j_m) + f d_{t'_u}(j_m) \tag{6.6}$$

Bij een schalering van een edgesegment zal $|d(t)| \neq |d(t')|$. Met andere woorden, de bemonsteringsschaal van de edge moet nog vereenzelvigd worden. De meest getailleerde edge wordt hiervoor onderbemonsterd met een stapgrootte gelijk aan λ. Voor subpixel descripties wordt ook lineaire interpolatie gebruikt om de descriptiehoek tussen naburige descripties op het nieuwe onderbemonsteringsniveau te berekenen. Zij s de bemonsteringstap voor de meest gedetailleerde edge e_{max}, f het fractionair gedeelte van s_k, s_{k_l} = floor(s_k) en s_{k_u} = ceil(s_k) de naburige integer bemonsteringen. De descriptie op sample op positie s_k is dan gelijk aan:

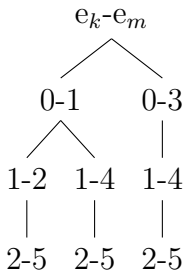
$$d_t(s_k) = (1 - f)d_t(s_{k_l}) + f d_t(s_{k_u}) \tag{6.7}$$

Onder aanname van edge volledigheid en uniforme schalering worden edges gematched op basis van 3 thresholds die voldaan moeten zijn:

- P_c, percentage overeenkomst in connectiviteit: tussen 2 mogelijk corresponderende segmenten wordt het aantal pixels geteld waarbij de

Edge matching: $e_k - e_m$	
e_k-segments	e_m-segments
0	1
0	3
1	2
1	4
2	5

(a) Matching segments voor edges $e_k - e_m$.



(b) Boom.

Figuur 6.17: Voorbeeld van het opbouwen van een matching tree voor corresponderende edgesegmenten.

connectiviteitsafwijking van de hoek kleiner was dan tien graden bij een descriptiestap van 5.

- P_{cdev}, percentage sterke afwijking in connectiviteit: tussen 2 mogelijk corresponderende segmenten wordt het percentage aan edgepixels berekend waarbij de connectiviteitsafwijking groter was dan 30 graden.
- AVG_θ, gemiddelde hoekafwijking over de edge: tussen 2 mogelijk corresponderende segmenten wordt de gemiddelde hoekafwijking berekend. Een gemiddelde afwijking onder 8 graden wordt geaccepteerd.

Wanneer alle condities aanvaard zijn, worden de 2 segmenten als corresponderend aanzien.

Segment merging

Eens matching tussen edgesegmenten voltooid is, worden per edge alle combinaties van mogelijke opeenvolgende segmenten gevormd. Er kunnen meerdere paden van opeenvolgende segmenten zijn vermits eenzelfde segment met meerdere andere segmenten kan matchen. Deze paden worden beschreven door een matching tree. Een voorbeeld hiervan is weergegeven in figuur 6.17. Aan alle bladeren uit de boom worden segmenten toegevoegd indien ze opvolgend zijn aan het reeds bestaande pad in de boom. Ieder pad van wortel naar blad beschrijft een mogelijk sequentie die voor matching moet worden onderzocht.

Nadat deze paden zijn opgebouwd, wordt er opnieuw een segment matching uitgevoerd voor tussenliggende segmenten. Deze tussenliggende segmenten kunnen nu mogelijk wel een match geven als ze eerder nog niet als

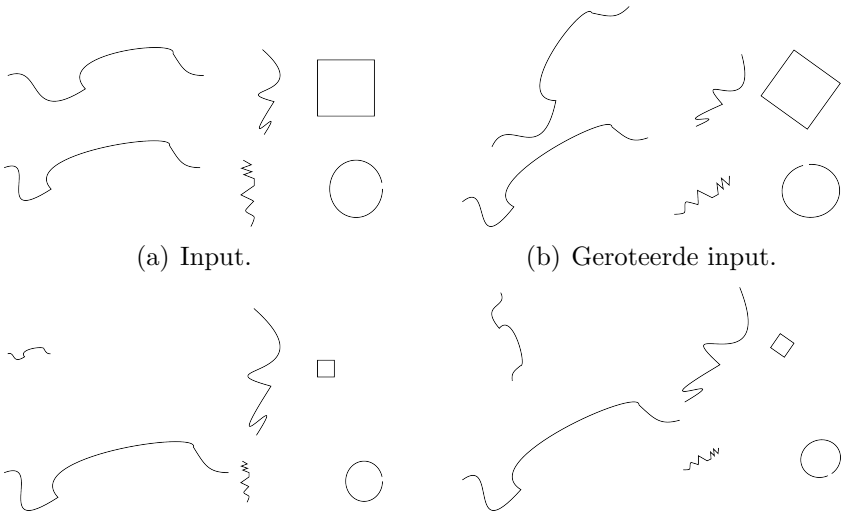
volledig segment aan de segment matching procedure hebben doorlopen. Verder worden edgesegmenten verlengd met behulp van de schaalfactor berekend tussen dit segment en het overeenkomstige segment, zolang de waarde van P_{cdev} klein genoeg blijft. Indien er voor paden van gelinkte edgesegmenten van een edge een bepaald percentage van de edge correspondeert met een edge uit het andere beeld, dan worden deze edges als een match beschouwd.

6.5 Resultaten

In deze sectie wordt de de geïntroduceerde edge matching techniek geëvalueerd aan de hand van de resultaten bij uitvoering op afbeeldingen die corresponderende edges bevatten die in volledigheid, positionering, orëntatie en schaal kunnen verschillen.

6.5.1 Dataset

Onze edge-dataset, waarop we onze techniek zullen evalueren, werd gegenereerd met getekende edges, zoals in figuren 6.18. Vervolgens werd de edge detector van [Topal et al. [103]] met default parameters toegepast om zo edges van 1 pixelbreedte te verkrijgen zoals deze in figuur 6.19. Hierna werden de gedetecteerde edges manueel ingekort door edgepixels weg te snijden om dubbele randen te vermijden. Het resultaat hiervan is getoond in figuur 6.20. Edges verschillen in de dataset dus niet enkel van rotatie, of schalering, ook de volledigheid van edges kan verschillen door verschillende positionering van begin- en eindpunten na het inkorten ervan zoals weergegeven in figuur 6.21. Daarnaast kunnen lichte veranderingen in lokale kromming optreden door onnauwkeurigheden van de gebruikte edgedetectiemethode. Zulke fouten zijn geïllustreerd in figuren 6.22 en 6.23.



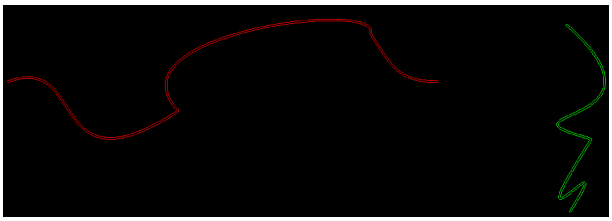
(a) Input.

(b) Geroteerde input.

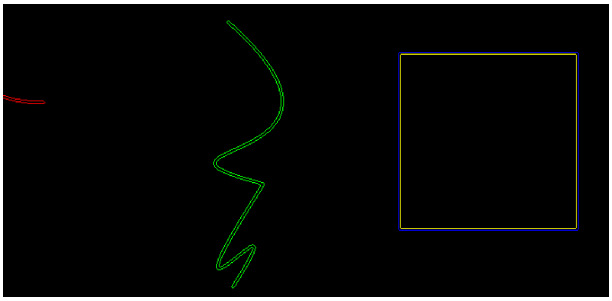
(c) Schaalgetransformeerde input.

(d) Schaalgetransformeerde en geroteerde input.

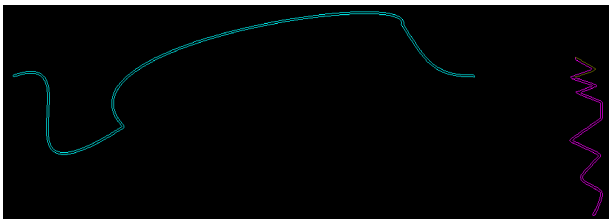
Figuur 6.18: Manueel getekende edges.



(a) Detectie in de linkerbovenhoek.



(b) Detectie in de rechterbovenhoek.



(c) Detectie in de hoek linksonder.

(d) Detectie in de hoek rechtsonder.

Figuur 6.19: Edgedetectie van [Topal et al. [103]] toegepast op figuur 6.18(a).

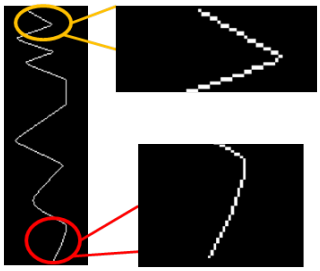
(a)

(b)

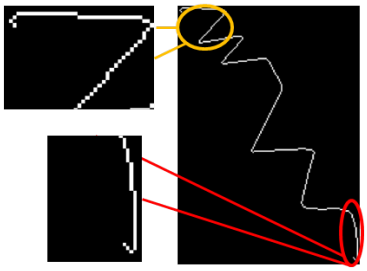
(c)

(d)

Figuur 6.20: Resulterende edges van 1 pixelbreedte die gebruikt zullen worden voor de evaluatie van onze edge matching methode. Deze zijn bekomen uit de gedetecteerde edges door ze in te korten zodat dubbele randen verdwijnen.

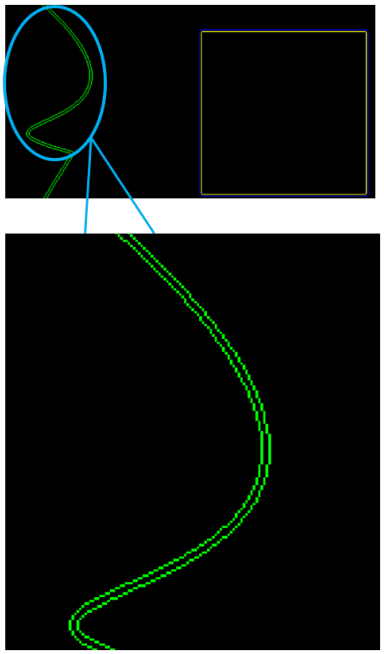


(a) Edge uit figuur 6.20(c).

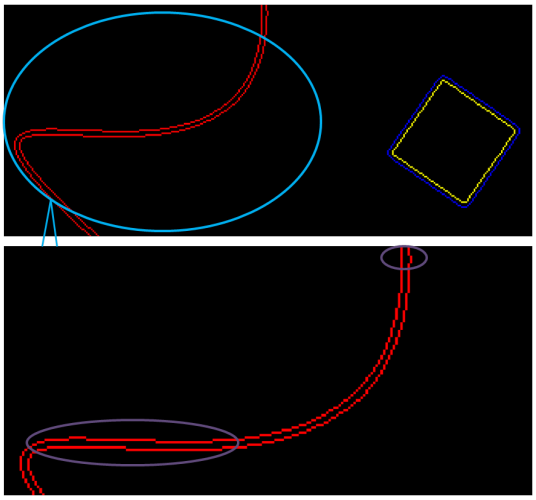


(b) Edge uit figuur 6.20(b).

Figuur 6.21: Voorbeeld van verschillende begin- en eindposities van een overeenkomstige edge na edgedetectie en het bijsnijden van edges.



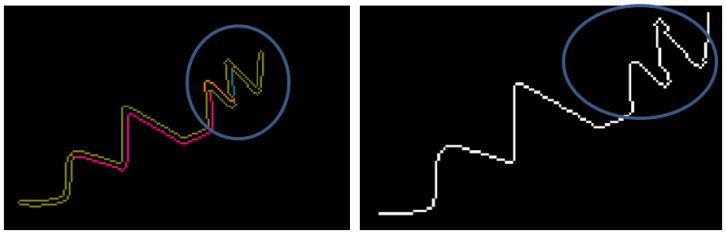
(a) Edge detectie toegepast op figuur 6.20(a).



(b) Edge detectie toegepast op figuur 6.20(d).

Figuur 6.22: De gebruikte edgedetectiemethode van [Topal et al. [103]] is niet steeds richtingconsistent op lokaal niveau wanneer ze wordt toegepast op getransformeerde edges. Geïntroduceerde richtinginconsistenties zijn in afbeelding 6.22(b) omcirkelt in het paars. Op deze plaatsen zijn er richtingsminima en -maxima toegevoegd terwijl deze niet aanwezig waren bij de edgedetectie in figuur 6.22(a).

(a) Zig-zag edge uit figuur 6.20(c).

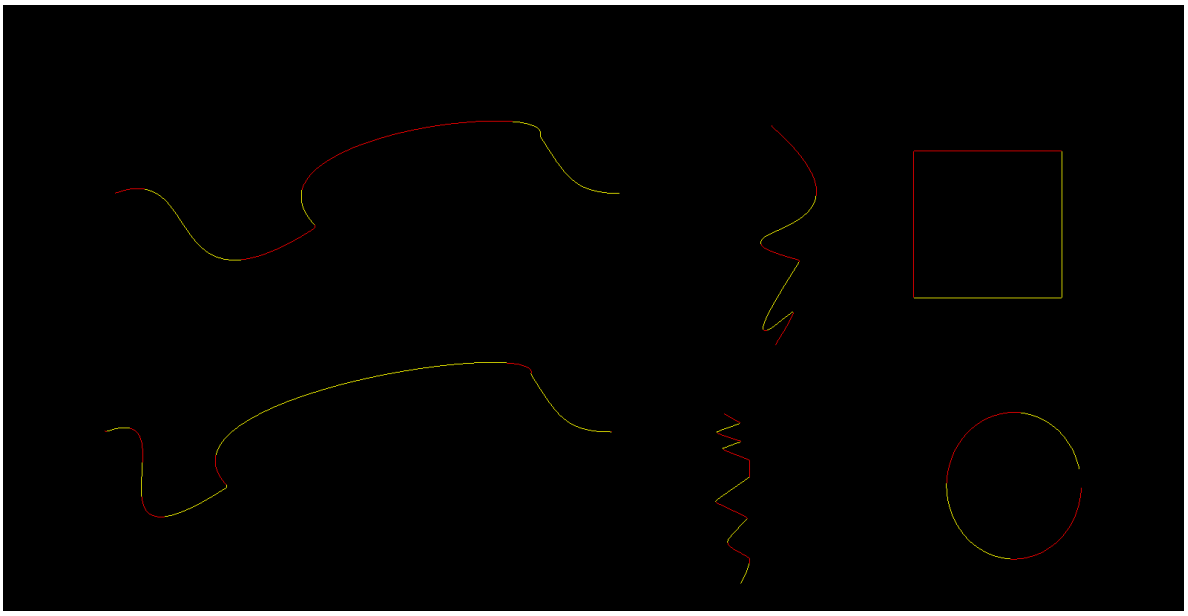


(b) Zig-zag edge uit figuur 6.20(d) met geometrische errors omcirkelt in het blauw.

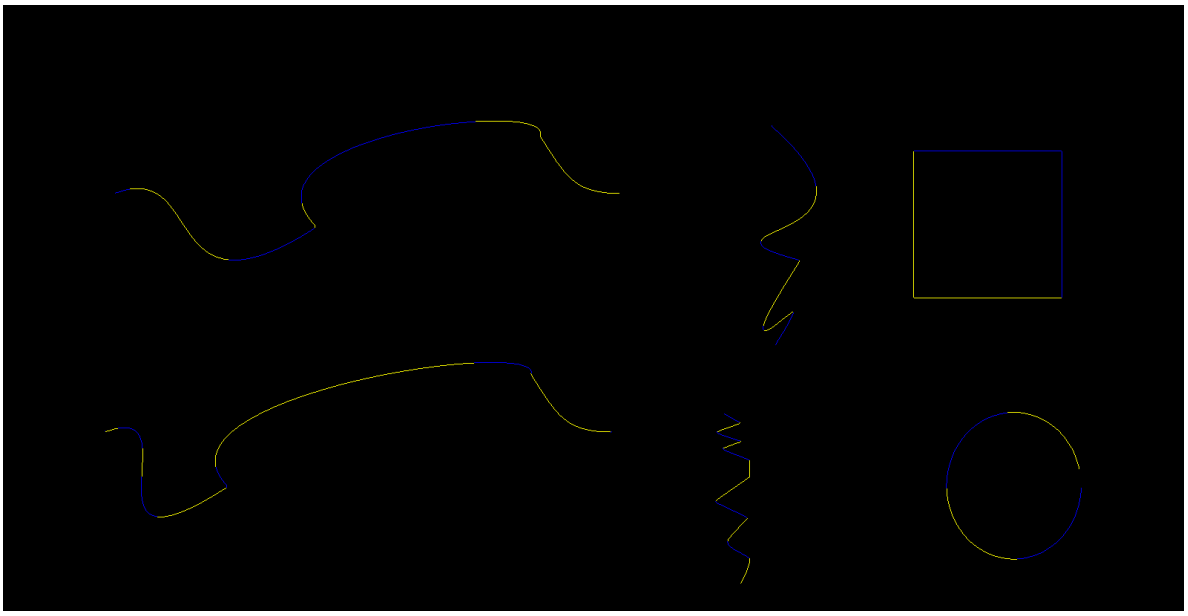
Figuur 6.23: Illustratie van introductie geometrische errors na edgedetectie van de zig-zag edge uit figuur 6.18(d).

6.5.2 Edge segmentatie

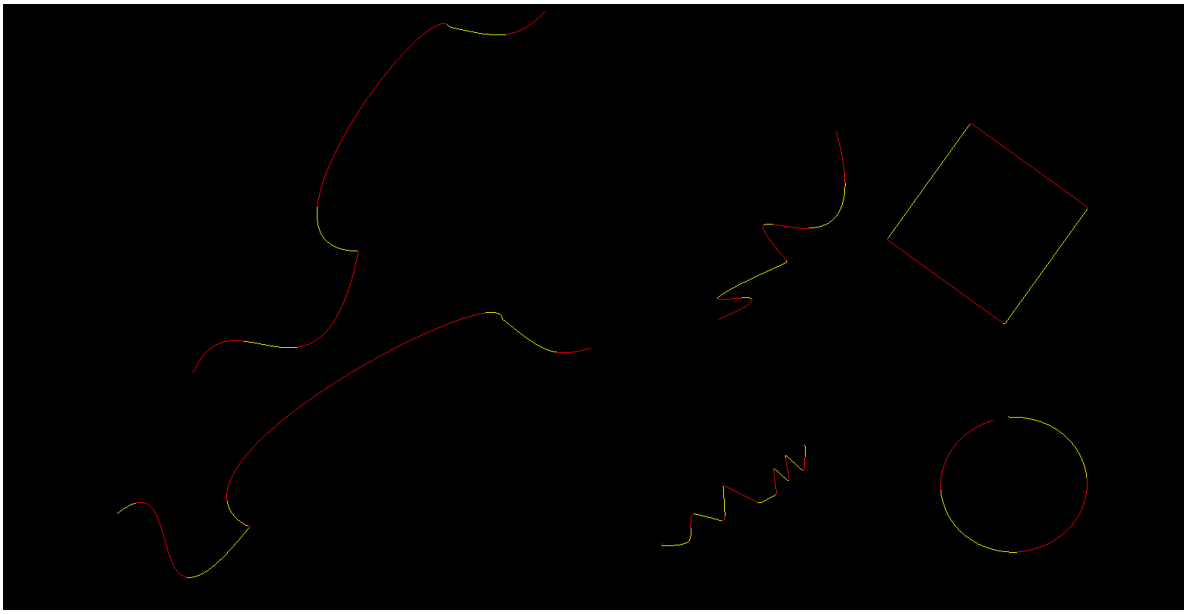
Het edge segmentatiealgoritme dat edges in segmenten verdeelt op basis van richtingsveranderingen wordt in 2 richtingen op iedere edge toegepast om zininvariantie te kunnen garanderen. Figuren 6.24 - 6.31 tonen de resultaten van de toepassing van dit algoritme op de verschillende afbeeldingen uit onze dataset. De kleurcombinatie rood-geel duidt de segmenten aan die respectievelijk gevonden werden na een voorwaartse edge iteratie. De kleurcombinatie blauw-geel duidt de segmenten aan die gevonden werden bij een edge iteratie in omgekeerde zin. Uit figuren 6.24 en 6.26 kunnen we afleiden dat de richtingsveranderingspunten gedetecteerd bij gekromde lijnen, rotatievariant zijn wanneer deze beschreven worden aan de hand van een regular grid. De monotoniciteit van een kromme kan door de buiging verschillen naargelang hoe deze georiënteerd is ten opzichte van de grid, terwijl de monotoniciteit van een rechte ten opzichte van een grid, ongeacht de oriëntatie, behouden zal blijven. Zij R een gedetecteerd richtingsveranderingspunt in een edge e en zij R' een gedetecteerd richtingsveranderingspunt in een andere edge e'. In het geval van rechte lijnen (bijv. polygonsegmenten) zal tussen corresponderende edges over alle oriëntaties gelden dat $R \subseteq R'$, ofwel $R' \subseteq R$ aangezien over een lijn zelf geen nieuwe richtingsveranderingen worden toegevoegd. De afbeelding van een 3D lijn zal steeds een monotoon stijgende lijn zijn over een reguliere grid omdat de persepectiefprojectie colineariteit behoudt [Miller and Hilton [77]]. De verzamelingen R en R' kunnen een verschillende kardinaliteit hebben waarbij $|R|$ niet noodzakelijk gelijk is aan $|R'|$. Bijvoorbeeld wanneer we het vierkant in figuur 6.24 met het vierkant uit figuur 6.26 vergelijken. Bij exacte alignering van lijnen van het vierkant zal volgens het gebruikte segmentatiealgoritme nog geen richtingsverandering optreden omdat nog geen tegenovergestelde richtingen bezocht zijn zoals geïllustreerd wordt in figuur 6.32. In figuur 6.33 wordt getoond dat afwijkingen in smoothness die ontstaan zijn na het toepassen van de edgedetectiemethode uit [Topal et al. [103]] de richtingconsistentie soms verstoort, waardoor er extra edgesegmenten worden toegevoegd. Deze afwijkingen kunnen zich bij iedere edgedetectiemethode voordoen bijvoorbeeld in geval van beeldruis in een edge-afbeelding.
Een schalering van edges doet de richting van een willekeurige edge over een grid niet veranderen. In dat geval kan het segmentatiealgoritme voor willekeurige edges gebruikt worden. In figuren 6.24 en 6.28 zijn dan ook erg gelijkaardige partities in edgesegmenten te zien.



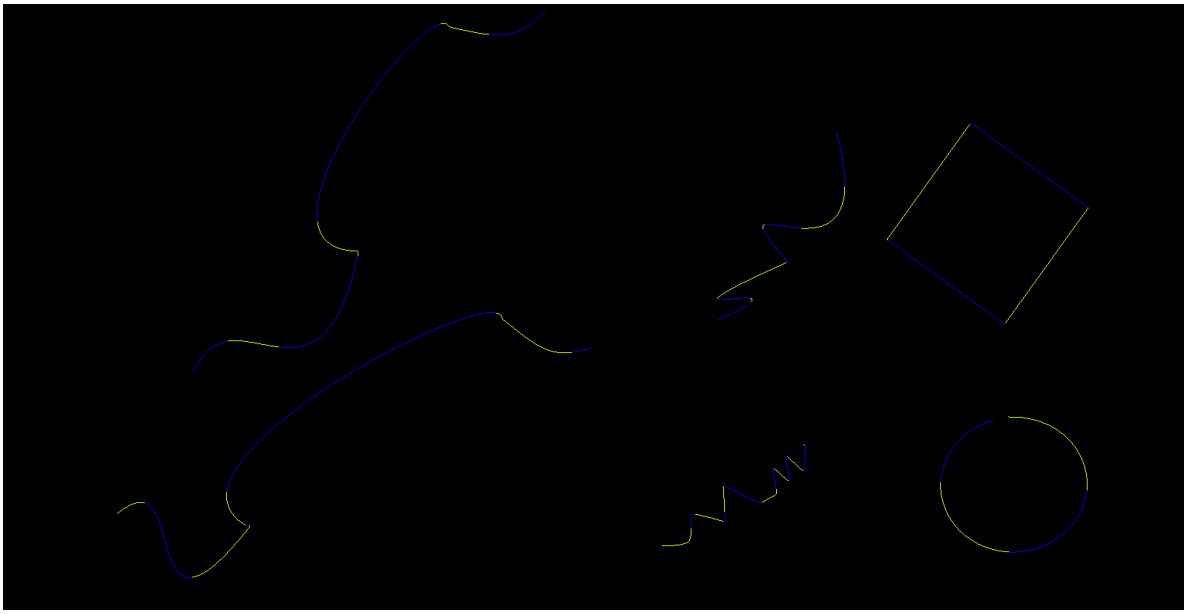
Figuur 6.24: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in voorwaartse richting over iedere edge uit figuur 6.20(a) is verplaatst.



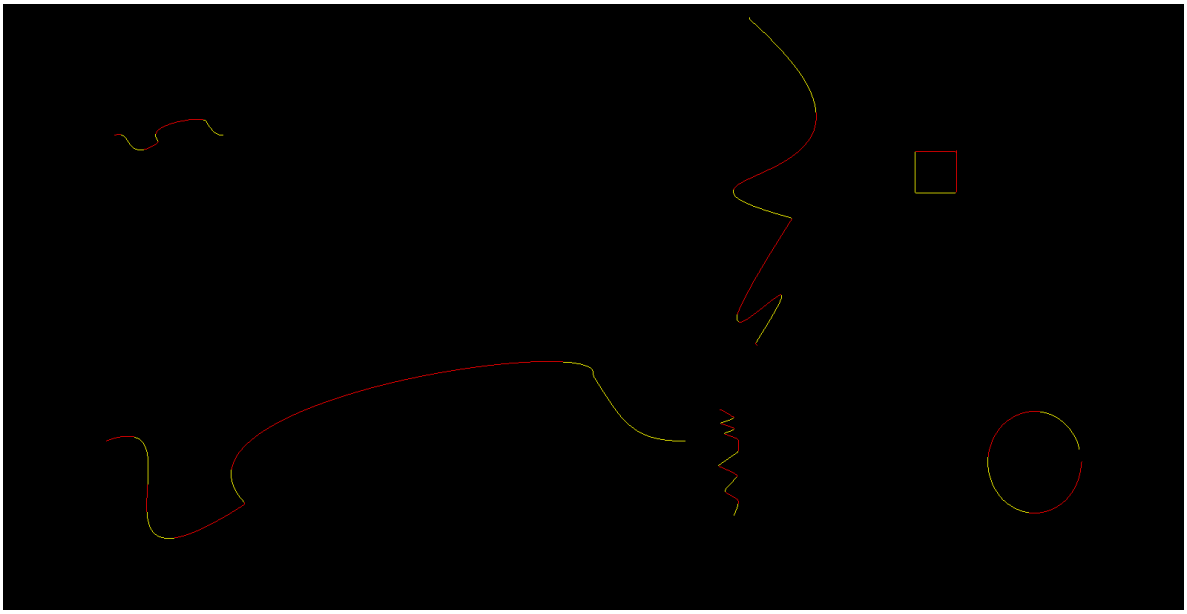
Figuur 6.25: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in achterwaarse richting over iedere edge uit figuur 6.20(a) is verplaatst.



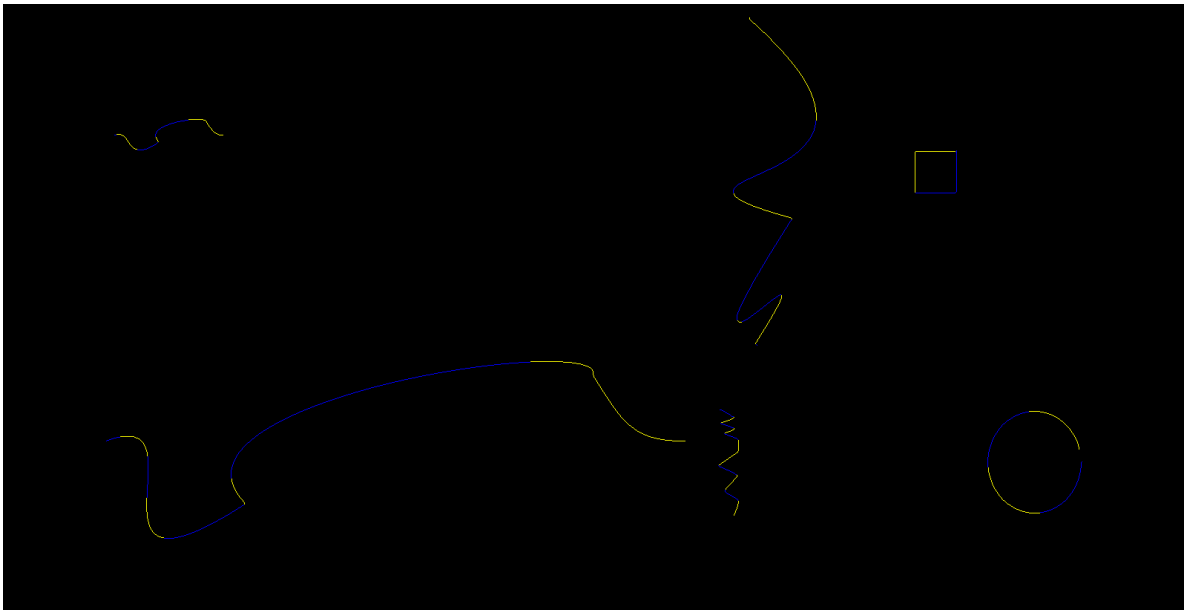
Figuur 6.26: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in voorwaartse richting over iedere edge uit geroteerde figuur 6.20(b) is verplaatst.



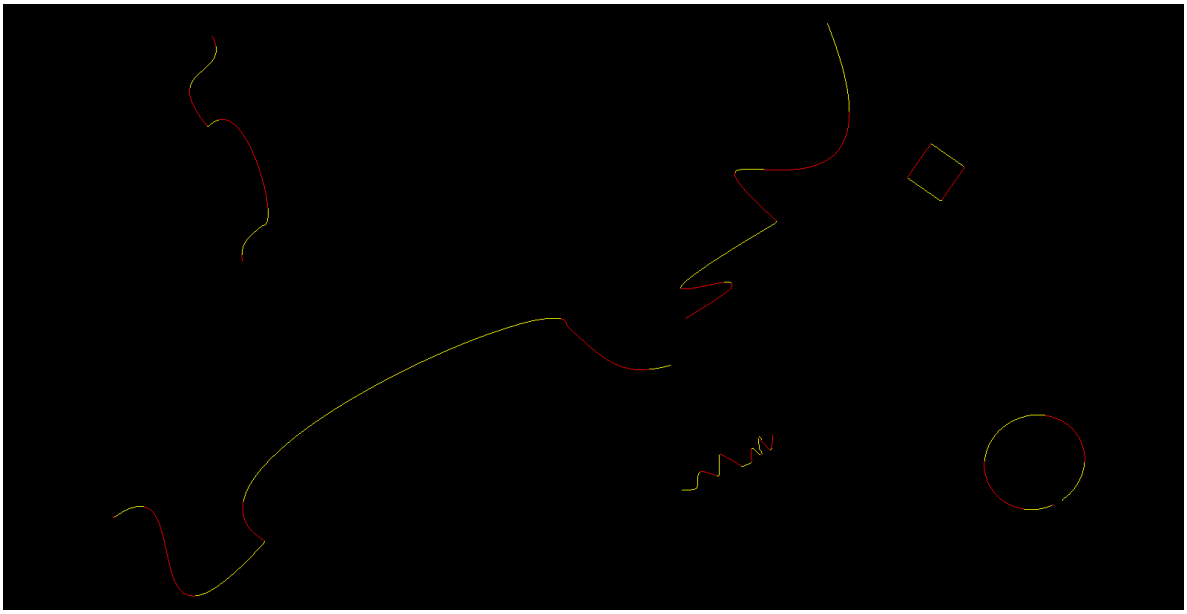
Figuur 6.27: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in achterwaarse richting over iedere edge uit geroteerde figuur 6.20(b) is verplaatst.



Figuur 6.28: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in voorwaartse richting over iedere edge uit geschaleerde figuur 6.20(c) is verplaatst.

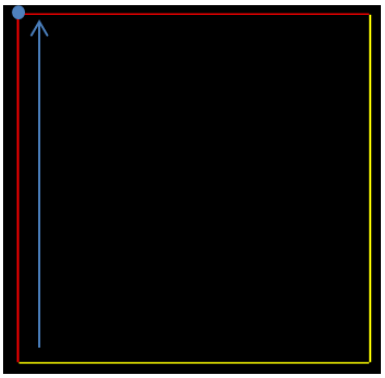


Figuur 6.29: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in achterwaarse richting over iedere edge uit geschaleerde figuur 6.20(c) is verplaatst.

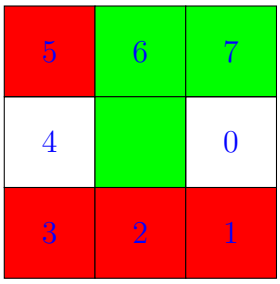


Figuur 6.30: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in voorwaartse richting over iedere edge uit geschaleerde en geroteerde figuur 6.20(d) is verplaatst.

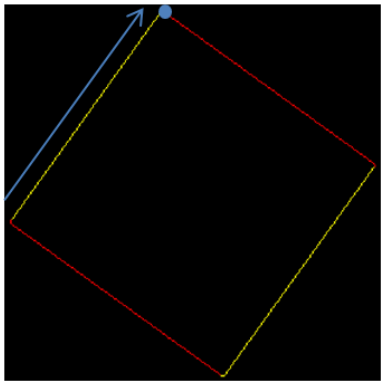
Figuur 6.31: Resultaten van edge segmentatiealgoritme waarbij het gridmasker in achterwaarse richting over iedere edge uit geschaleerde en geroteerde figuur 6.20(d) is verplaatst.



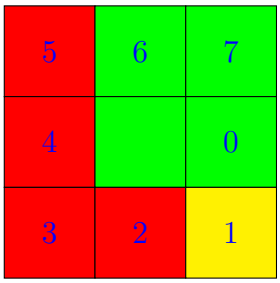
(a) Segmentatie van het vierkant uit figuur 6.24.



(b) Het resultaat van het richtingsveranderingsmasker na het traceren van de linkerzijde in het omcirkelt punt van figuur 6.32(a).

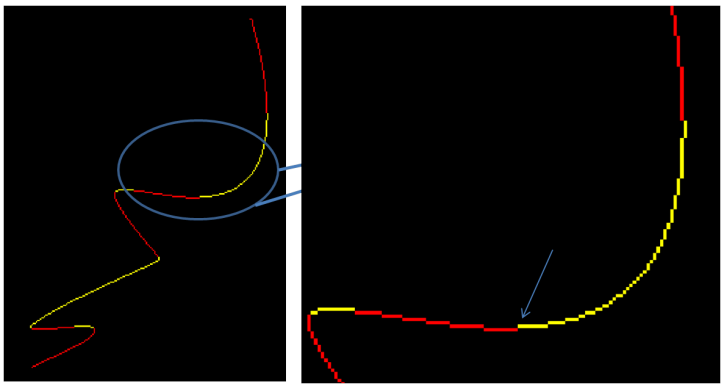


(c) Segmentatie van het vierkant uit figuur 6.26.

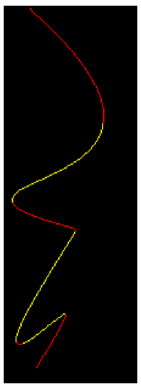


(d) Het resultaat van het richtingsveranderingsmasker na het traceren van de linkerzijde in het omcirkelt punt van figuur 6.32(c).

Figuur 6.32: Illustratie van edge segmentatie bij corresponderende edges die enkel uit lijnen zijn opgebouwd. Het aantal gepartitioneerde segmenten kan verschillen, maar de richtingsveranderingspunten van figuur 6.32(a) $\subseteq$ de richtingsveranderingspunten van figuur uit figuur 6.32(c).



(a) Fouten in smoothness na edgedetectie zoals afgebeeld in figuur 6.22(b) kunnen de edgesegmentatie beïnvloeden en extra segmenten introduceren zoals dit hier gebeurt voor een edge uit figuur 6.20(b).



(b) Edge segmentatie wanneer smoothness behouden blijft voor corresponderende edge uit figuur 6.20(a).

Figuur 6.33: Illustratie van de invloed van toegevoegde richtingsveranderingen door fouten tijdens de edgedetectie.

Tijdcomplexiteit

Voor het verdelen van een edge in segmenten moet het richtingsveranderingsmasker over een edge worden verplaatst in 2 richtingen om zininvariant te zijn. Op iedere positie, moet de positie van de buurtpixel van de huidige pixel in het masker worden onderzocht. Dit kan in constante tijd. Zij n het aantal edgepixels in een afbeelding. De tijdscomplexiteit bedraagt dan $O(2n)$.

Conclusie

Het besproken edge segmentatiealgoritme dat opereert in lineaire tijd is invariant in zin en schaal. De toepassing ervan bij geroteerde edges is echter beperkt omdat bijvoorbeeld de richtingen van gekromde lijnen over een grid op een lokaal pixelniveau niet invariant zijn. Het algoritme kan wel opereren onder rotaties wanneer edges opgebouwd zijn uit rechte lijnen bijvoorbeeld bij polygonen. Hierbij dienen segmenten in sommige gevallen nog gecombineerd te worden om hetzelfde aantal edgesegmenten te bekomen.

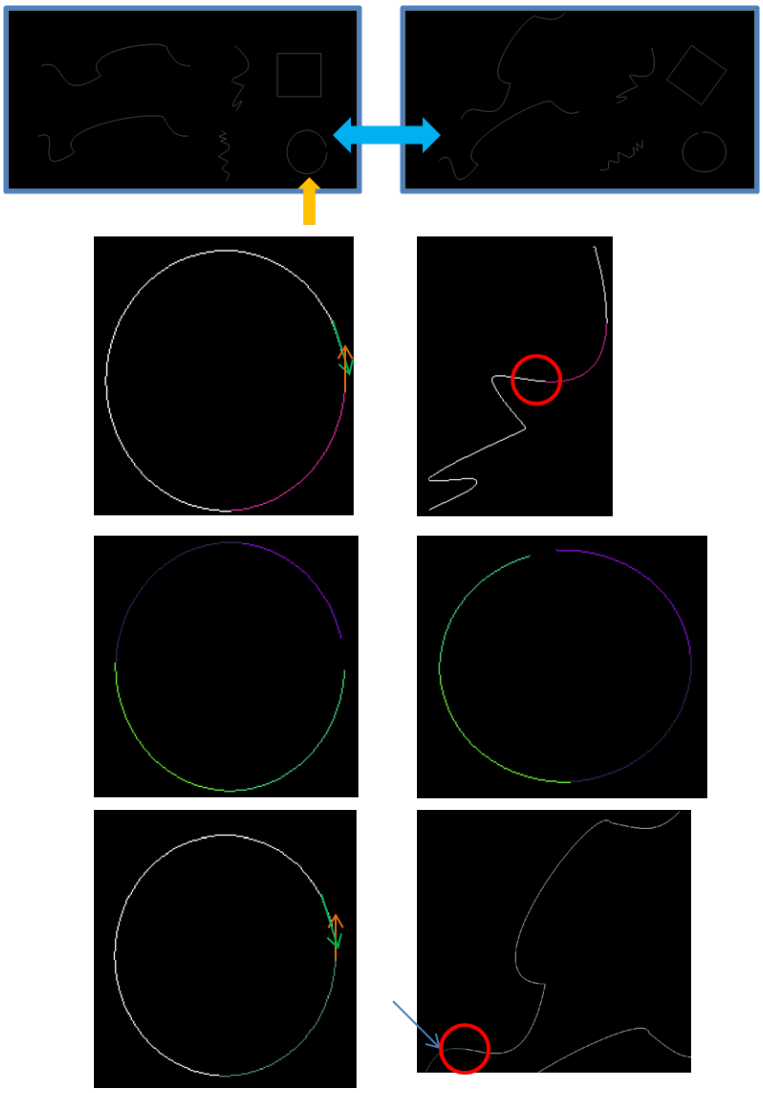
Verbeteringen

Om de edgesegmentatie toepasbaar te maken voor alle type edges onder rotatie, zouden we, in de plaats van de huidige features die gekozen worden op basis van de grid, edge features over schaal kunnen zoeken in de descriptiematrix. Pixels waarbij over schaal veel hoekextrema voorkomen ten opzichte van buurtpixels, komen hiervoor in aanmerking.

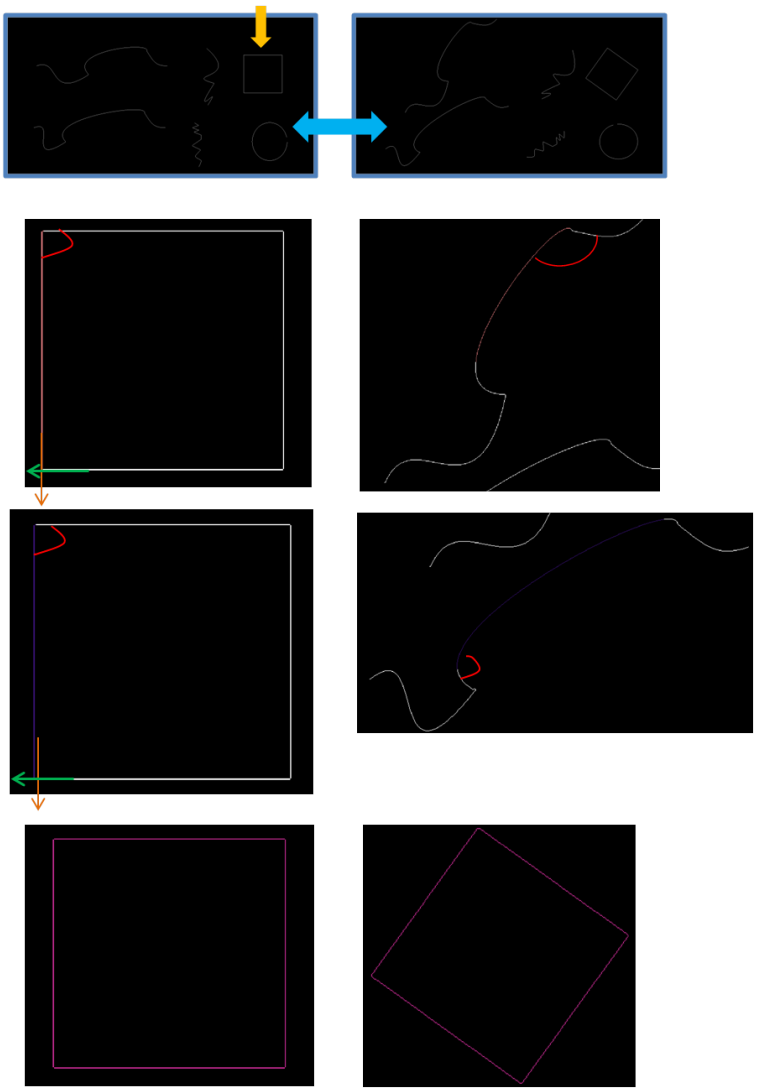
6.5.3 Edge matching

In deze sectie wordt de robuustheid van de matching techniek op verschillende descriptieniveaus over de verschillende edge transformaties geëvalueerd in figuren 6.34 - 6.41. In iedere figuur wordt een edge geselecteerd (aangeduid door de gele pijl in de subfiguur linksboven) waarvoor overeenkomsten in het rechter beeld worden gezocht. Op de volgende rijen wordt in elke figuur ieder pad van segmenten, waarvoor er in een edge uit het andere beeld overeenkomsten gevonden zijn, getoond. Corresponderende kleuren tussen verschillende beelden visualiseren een match tussen segmenten.

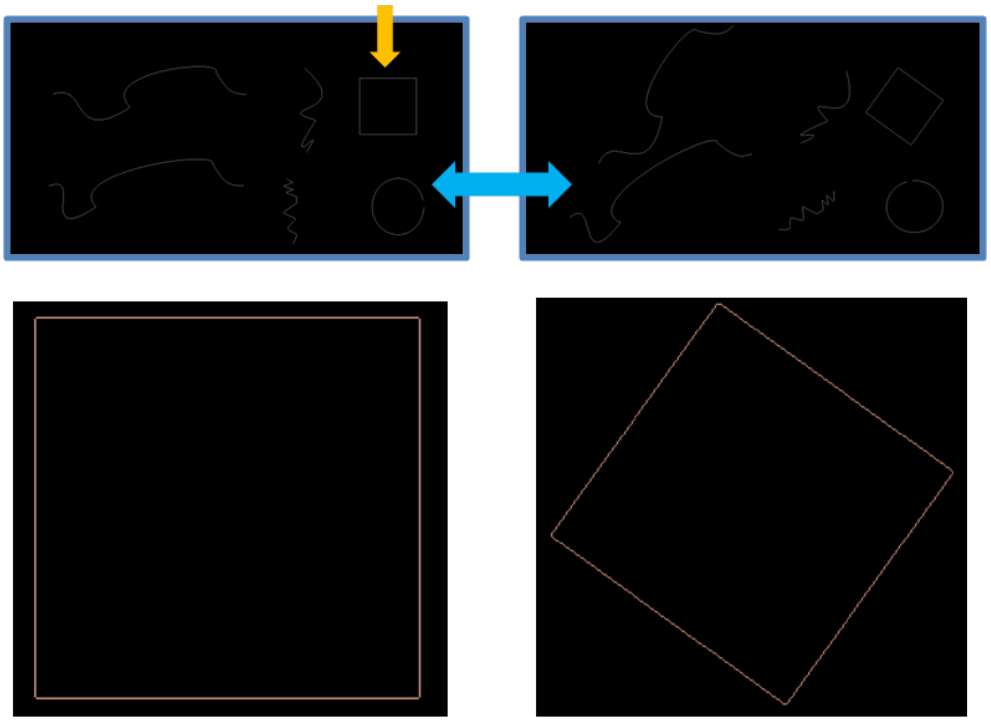
Rotatie



Figuur 6.34: Illustratie van de gevonden overeenkomsten voor de cirkelsegmenten uit de figuur links met de geroteerde edges uit de figuur rechts met een descriptiestap van 12 edgepixels. De tweede rij afbeeldingen tonen aan dat alle corresponderende cirkelsegmenten succesvol werden gevonden. Tweemaal in de figuren op rij 1 en 3 werden valse segmenten als corresponderend aangeduid met een cirkelsegment aan het uiteinde voor de opening van de cirkel. Dit is een gevolg van het gebruik van de extensievectoren voor descriptie. De extensievector introduceert een afvlakking aan het uiteinde van het cirkelsegment (aangeduid in het oranje). Deze afvlakking in connectiviteit is ook terug te vinden bij de aangrenzende regio's van het corresponderend segment (omcirkeld in het rood).

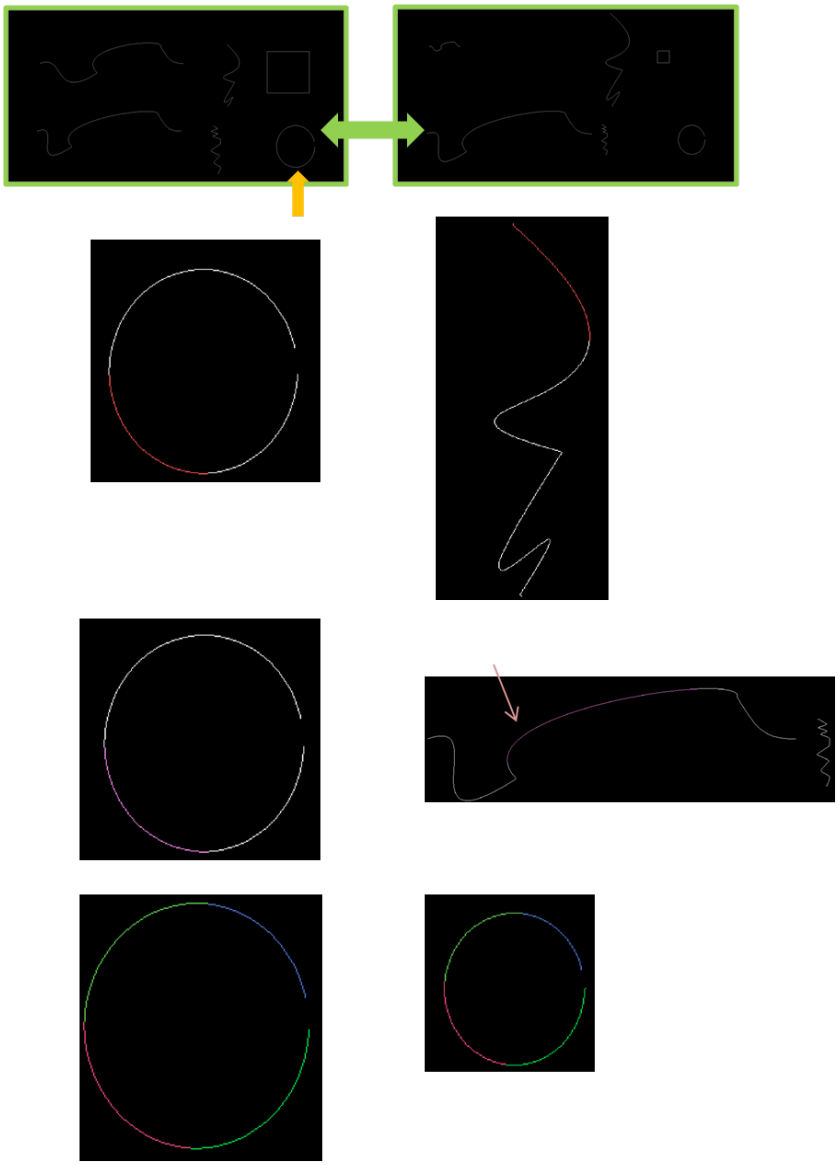


Figuur 6.35: Illustratie van de gevonden overeenkomsten voor het vierkant uit de figuur links met de geroteerde edges uit de figuur rechts met een descriptiestap van 12 edgepixels. Hoewel het vierkant gesloten is, worden voor de begin- en eindsegmenten ook descriptievectoren aan de uiteinden voor descriptie gebruikt. Enkele segmenten op rij 1 en 2 in de rechterkolom vormen met naburige segmenten quasi een hoek van 90 graden en zullen daarom ook als match geselecteerd worden. Het corresponderende vierkant werd volledig teruggevonden zoals te zien is op de laatste rij. De uniforme kleur geeft aan dat er een match heeft plaatsgevonden van segment(en) met andere segment(en) en niet voor alle segmenten meteen een corresponderend segment werd gevonden, hetgeen aan de hand van de edgepartitie in figuur 6.26 te verklaren is. Om tot een volledige correspondentie te komen, zijn in beide afbeeldingen nieuwe tussenliggende segmenten geconstrueerd op basis van de reeds corresponderende segmenten. Wanneer deze tussenliggende segmenten overeenkomen werd het geheel samengevoegd als 1 segment (segment merging), vandaar de uniforme kleur.

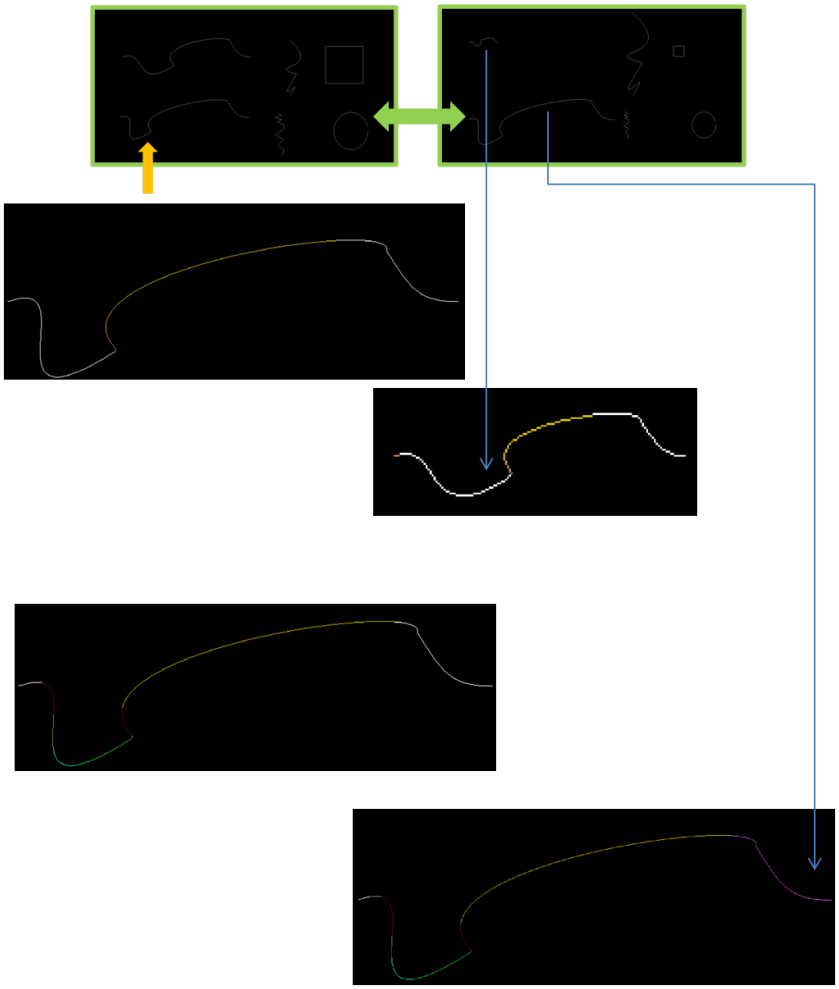


Figuur 6.36: Illustratie van de gevonden overeenkomsten voor het vierkant uit de figuur links met de geroteerde edges uit de figuur rechts met een descriptiestap van 25 edgepixels. In dit geval wordt enkel een overeenkomst voor het vierkant gevonden tussen beide beelden, daar dit met een descriptiestap van 12 niet alleen het geval was. Een grotere descriptiestap zal edges meer op een globaal detailniveau beschrijven, waarbij meer informatie van aangrenzende regio's verderop in het matching proces zal worden geïncorporeerd. Gebruik maken van connectiviteitsinformatie op meerdere niveaus zal de robuustheid vergroten, wat door deze resultaten ook wordt aangetoond.

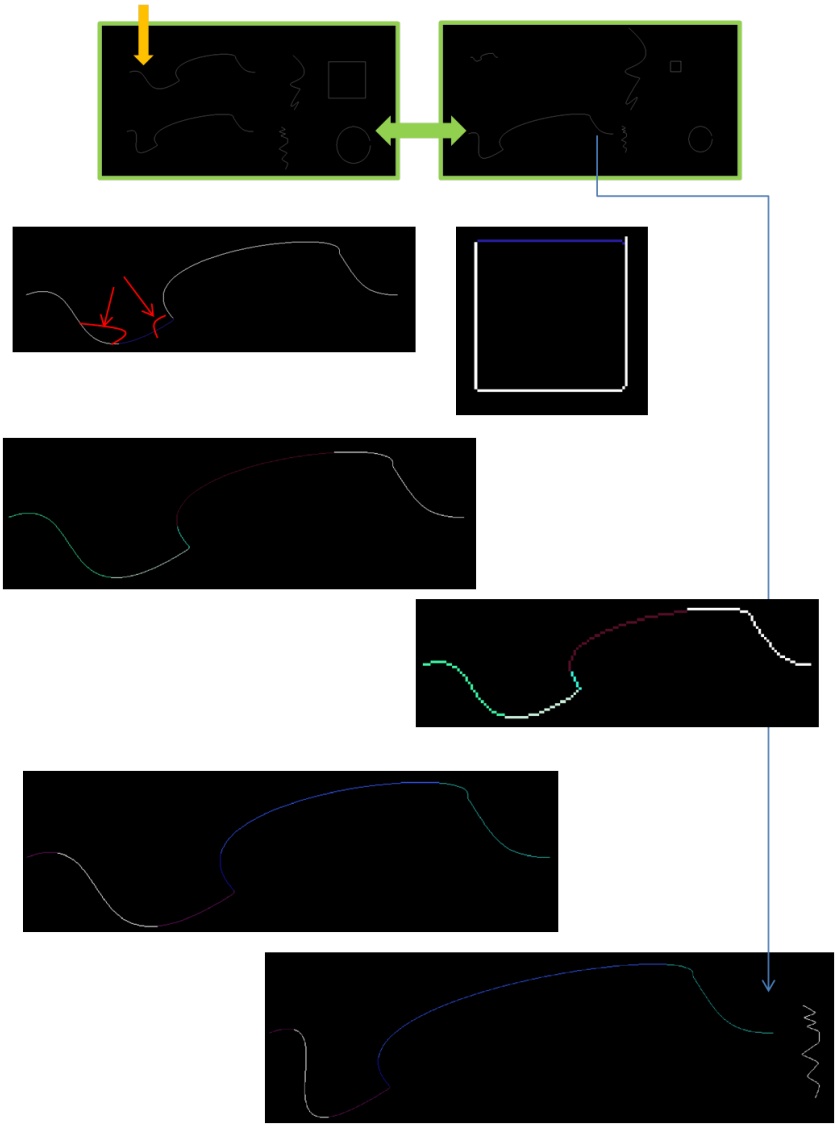
Schalering



Figuur 6.37: Illustratie van de gevonden overeenkomsten voor de cirkelsegmenten uit de figuur links met de geschaleerde edges uit de figuur rechts met een descriptiestap van 7 edgepixels. De onderste rij toont gewenste correspondenties. De corresponderende segmenten op de overige rijen beschrijven lokaal ook een cirkelachtige structuur. De robuustheid van matching zal toenemen wanneer de descriptiestap vergroot wordt omdat dan de cirkelstructuur voor deze segmenten niet meer wordt aangehouden.

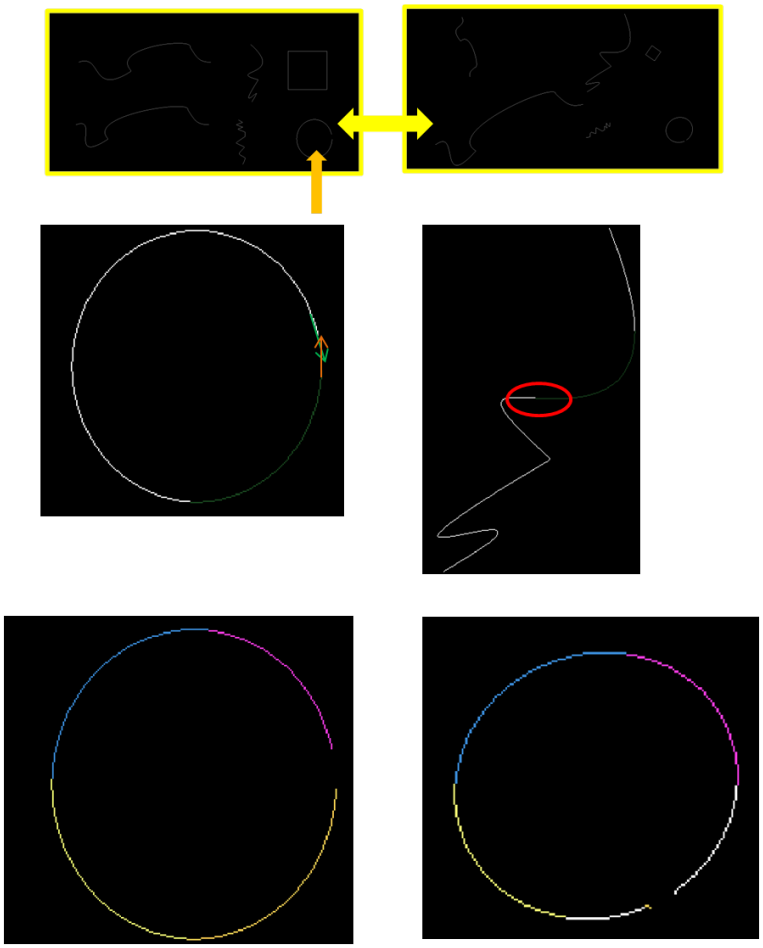


Figuur 6.38: Illustratie van de gevonden overeenkomsten voor de edge linksonder in de linkerfiguur met de geschaleerde edges uit de rechterfiguur met een descriptiestap van 7 edgepixels. Twee edgesegmenten van de edge linksonder werden teruggevonden in de edge linksboven uit de rechterfiguur. Dit is correct vermits enkel de z-vorm aan het linkeruiteinde van de edges linksboven en linksonder verschillend is. Het z- vormige segment werd terecht ook niet als correspondentie gemarkeerd. In de onderste rij zijn bijna alle edgesegmenten teruggevonden. De laatste pixel uiterst rechts van de edge links beneden in de linkerafbeelding, die als segment is toegevoegd door fouten in de randdetectie, gaat met het laatste segment van deze edge in de geschaleerde afbeelding corresponderen (aangeduid in het paars). Deze correspondentie ontstaat door sterke onderbemonstering, waarbij er maar 1 bemonstering wordt genomen gelijk aan de volledige lengte van een edgesegment en enkel de connectiviteit tussen 2 pixels wordt vergeleken. Het correcte segment wat correspondeert met het paarse segment in de geschaleerde afbeelding voldeed met $AVG_\theta = 8.6$, $P_c = 78\%$,$P_{cdev} = 6.3\%$ niet aan de vastgelegde norm en werd hierdoor niet geselecteerd.

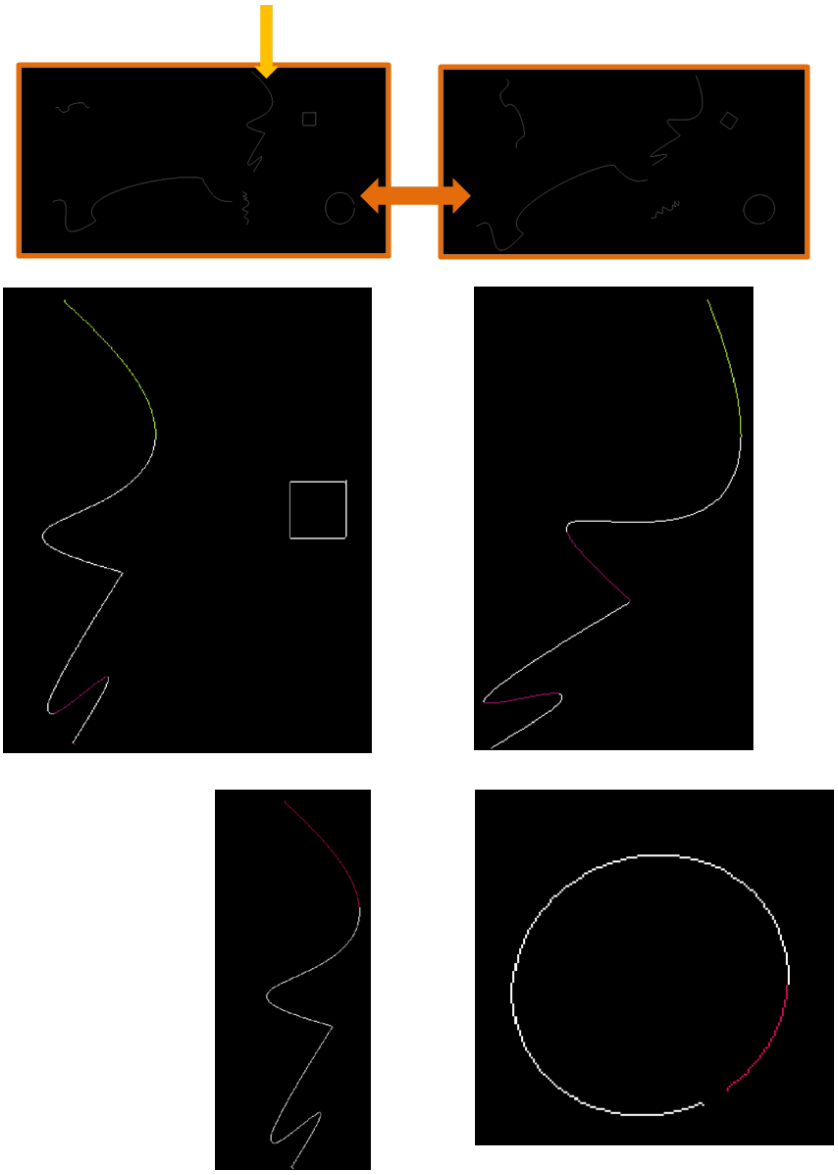


Figuur 6.39: Illustratie van de gevonden overeenkomsten voor de edge linksboven in de linkerfiguur met de geschaleerde edges uit de rechterfiguur met een descriptiestap van 7 edgepixels. Op de eerste rij wordt een stukje in het dal van de edge als match met een zijde van het vierkant geselecteerd. Op de volgende rij zijn de segmenten te zien in de corresponderende werden teruggevonden. De overeenkomst tussen het tweede segment met het tweede segment over schaal bedraagt $AVG_\theta = 7.2$, $P_c = 76.47\%$,$P_{cdev} = 0\%$ en voldoet net als de correspondentie tussen de laatste segmenten waar $AVG_\theta = 5.42$, $P_c = 87.88\%$,$P_{cdev} = 1.5\%$ niet aan de vooropgestelde grenzen. De edge linksboven in het linkerbeeld en de edge linksonder in het rechterbeeld corresponderen, op de z-bocht na. Dit is correct aangezien deze edges enkel daar verschillend zijn.

Rotatie en schalering



Figuur 6.40: Illustratie van de gevonden overeenkomsten voor de cirkel in de linkerfiguur met de geroteerde en geschaleerde edges uit de rechterfiguur met een descriptiestap van 12 edgepixels. Een cirkelsegment wordt gematched met een cirkelvormig bocht van een zig-zag edge die naar het einde toe afvlakt. Mede door de afvlakking die de extensievector teweegbrengt, zal dit segment gaan matchen met het cirkelsegment. Voor overige segmenten werd ook een correspondentie gevonden in de geroteerde en geschaleerde cirkel. De fouten die hier ontstaan zijn te wijten aan de rotatievariantie voor krommen van de gebruikte edgesegmentatiemethode.



Figuur 6.41: Illustratie van de gevonden overeenkomsten voor de zig-zag edge met booguiteinde in de reeds geroteerde linkerfiguur met de geroteerde en geschaleerde edges uit de rechterfiguur met een descriptiestap van 5 edgepixels. Net als in figuur 6.40 wordt het booguiteinde met een cirkelsegment gematched vanwege gelijkenissen in connectiviteit op het gekozen descriptieniveau. Verder wordt voor dit segment ook het corresponderende segment gevonden in de rechterafbeelding en wordt een rechtlijnig segment twee keren gematched met rechtlijnige segmenten die gelegen zijn in v-vormige bochten. Het niet vinden van correspondenties van de overige segmenten is toe te schrijven aan het verschil in edgesegmentatie.

Tijdcomplexiteit

Zij n het aantal edgepixels in een afbeelding. Het construeren van de descriptiematrix die de connectiviteit op verschillende detailniveaus beschrijft kan in 1 iteratie pass over een edge worden berekend door per pixel de connectiviteit met behulp van verschillende descriptiestappen te berekenen. De tijdcomplexiteit van de descriptie per afbeelding bedraagt $O(n)$. Vervolgens moet ieder segment uit een afbeelding met een segment uit een andere afbeelding worden vergeleken. Zij s het maximaal aantal edgesegmenten in een afbeelding. De bovengrens voor deze vergelijking bedraagt dan $O(s^2)$.

Conclusie

Het besproken edge matchingalgoritme dat opereert in kwadratische tijd is invariant in rotatie, zin en schaal. Bij een betrouwbare edgesegmentatie werden praktisch alle corresponderende edgesegmenten teruggevonden en die segmenten die niet gevonden werden, zaten rond de grens van de gebruikte thresholds. Af en toe werd een segment met een fout segment gematched omdat de lokale connectiviteit op het gebruikte descriptieniveau zeer gelijkaardig was. Er werd vervolgens aangetoond dat deze valse matches kunnen worden vermeden door edges op meerdere descriptieniveaus met elkaar te vergelijken.

Verbeteringen

De voorgestelde edge matching techniek kan verbeterd worden door:

- adaptieve thresholding

  Uit de resultaten bleek dat onder betrouwbare edgesegmentatie slechts een klein aantal corresponderende segmenten niet als match werden geselecteerd. De matchberekeningen van deze segmenten bevonden zich wel in de buurt van de vastgelegde thresholds. De thresholding kan worden verbeterd door ze adaptief te kiezen naargelang de mate van nauwkeurigheid van de rotatiebeschrijving in een bepaalde pixel. Zoals eerder besproken hangt de rotatienauwkeurigheid samen met de gekozen descriptiestap en is het dus wenselijk dat thresholds naargelang hun descriptieschaal ook mee variëren. Dit zou betekenen dat de thresholds bij kleine descriptiestappen (bijv. bij een zeer sterk geschaleerde edge) minder begrenzend zullen zijn dan bij grotere descriptiestappen en we in het geval van 2 verschillende descriptiestappen, de matching zullen doorvoeren op basis van de minst restrictieve threshold.

- vermijden van extensievectoren bij gesloten edges

  Bij gesloten edges zijn extensievectoren niet vereist aangezien de descriptie aan het uiteinde van een edge kan gebeuren relatief ten opzichte van het begin van een edge en omgekeerd omdat de edges gesloten zijn.

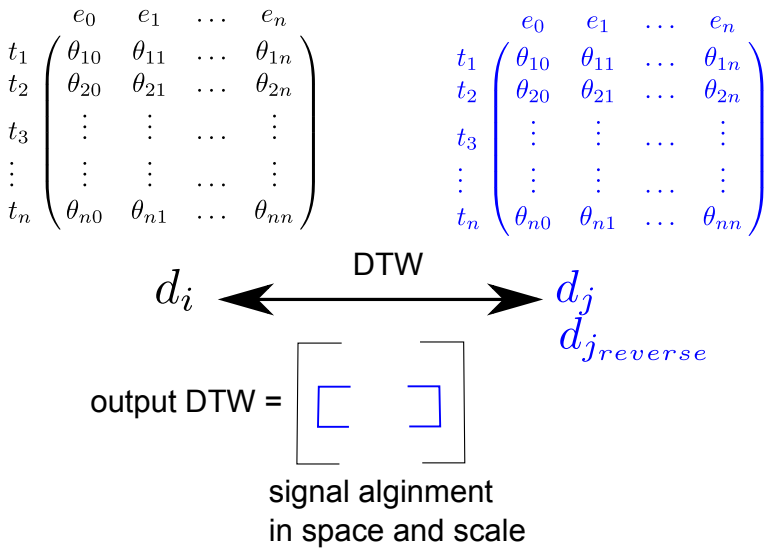
- genormaliseerde descriptie en exacte lengteberekening

  Tot nog toe werden het aantal pixels van een edge als benadering voor de lengte van een edge gebruikt om hieruit een schaalfactor af te leiden. Voor een exacte lengteberekening van een edge moeten het aantal horizontale pixels h, het aantal verticale pixels v en het aantal diagonale pixels d geteld worden [Gonzalez and Woods [36]]. De lengte van een edge is dan gelijk aan: $h + v + \sqrt{2}d$. Vermits diagonale pixels een grotere contributie hebben in afstand, dienen edgepixels herbemonsterd naar een keten van pixels waarvan de tussenliggende pixelafstand telkens 1 is. Deze normalisatie is bijvoorbeeld nodig wanneer een edge zich vooral diagonaal over een grid spreidt en dus minder pixels zal tellen dan een geroteerde versie van deze edge waarbij de spreiding eerder horizontaal is. Zo zullen de tijdelijke out of syncs tijdens het bemonsteren van een descriptievector gereduceerd worden.

6.6 Veralgemening

Het zoeken van correspondenties over de descriptiematrix d kan worden geformuleerd als het vinden van de aligning tussen 2 descriptiematrices die het verschil in descripties over de verschillende schaalniveaus minimaliseert. Beide signalen zullen dus spatiaal gealigneerd worden (kolom-aligning in descriptiematrix) en over schaal (rij-aligning in descriptiematrix). Deze aligning kan via dynamic time warping (DTW [Treichler and Agee [104]]) gevonden worden ondanks verschillen in lengte van descriptievectoren en verschillende bemonsteringen van 2 discrete signalen [Meltzer and Soatto [74]]. In deze aanpak is edgesegmentatie niet meer nodig vermits via DTW- per edgepixel de schaal met ideale aligning van beide discrete signalen berekent, hetgeen ook niet-uniforme schaleringen toelaat.

Zoals aangehaald werd door [Meltzer and Soatto [74]] is geometrische informatie niet invariant voor perspectieve vervorming. Het matchen van edges over een wide baseline is omwille van deze reden voor de huidige methode en de veralgemening ervan nog een open probleem omdat deze de connectiviteitshoeken gaan beïnvloeden. In zulke gevallen kan het beschrijven van relatieve connectiviteitspatronen op verschillende niveaus van een edge ten



Figuur 6.42: Edge matching van descriptiematrices d uit afbeeldingen i,j via dynamic time warping (DTW).

opzichte van geselecteerde features waar er over de verschillende descriptieniveaus een sterke connectiviteitsverandering heeft plaatsgevonden, een mogelijke basis vormen om edges toch aan de hand van geometrische informatie met elkaar te matchen, of om bestaande technieken die buurtinformatie gebruiken te verfijnen.

6.7 Conclusie

In dit hoofdstuk werd een nieuwe edge matching techniek geïntroduceerd. Deze techniek verschilt van de besproken technieken in het gerelateerd werk in het feit dat ze geen gebruik maakt van buurtregio-informatie, maar enkel gebruikt maakt van geometrische informatie voor het matchen van edges. Voor het vinden van edge correspondenties steunt onze techniek op edgesegmentatie enerzijds en een reflectie connectiviteitsdescriptie van een edge anderzijds.
De edgesegmentatie bestudeert de monotoniciteit van edges over een vaste grid met behulp van een 3 x 3 richtingsveranderingsmasker. Uit evaluatie van deze edgesegmentatiemethode blijkt dat deze is opgewassen tegen schaalvariantie, maar enkel rotatie-invariant is voor rechtlijnige edges. Enkele voorname redenen voor het toepassen van edgesegmentatie zijn het toelaten van matching over onvolledige edges bijvoorbeeld bij occlusies en het toelaten om een schaalfactor te kunnen schatten om descripties gelijkwaardig te kunnen bemonsteren.
De reflectieve connectiviteitsdescriptie beschrijft per edgepixel de connectiviteit van deze edgepixel ten opzichte van edge buurtpixels. Deze descriptie kan op verschillende detailniveaus gebeuren door de descriptiestap waartegenover de connectiviteit wordt beschreven, te laten variëren. Resultaten tonen aan dat dit een krachtige manier is om edges op basis van geometrische informatie met elkaar te kunnen vergelijken en dat de descripties op verschillende niveaus de robuustheid tijdens het vinden van correspondenties aanzienlijk kunnen vergroten. Tot slot werd gepostuleerd hoe de huidige connectiviteitsdescriptie in combinatie met een optimalisatietechniek gebruikt zou kunnen worden voor het vinden van edge correspondenties over verschillende afbeeldingen waarbij niet-uniforme schaleringen mogelijk zijn zonder dat hierbij een edgesegmentatie aan vooraf hoeft te gaan. Deze veralgemeende geometrischgebaseerde edge matching techniek is toepasbaar op afbeeldingen waartussen het verschil in perspectieve vervorming beperkt is, aangezien perspectieve vervorming de geometrische informatie van een edge verstoort.

Bijlage A

Convex hull

Een object uit de scene kan ofwel convex of niet-convex zijn. Een niet-convex object wordt ook wel een concaaf object genoemd. Voorbeelden van een object en een concaaf object zijn geïllustreerd in figuur A.1.

A.1 Definitie

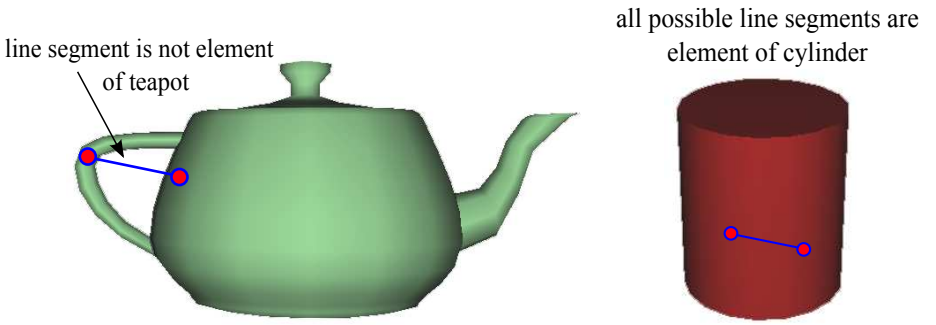
Definitie A.1.1. CH(O), de convex hull van een object O is het minimale object omhulsel waarbij ieder mogelijk lijnsegment tussen iedere mogelijke combinatie van twee punten van het object volledig in dit omhulsel bevat zit.

Uit deze definitie volgt dat in de convex hull alle andere punten bevat zijn en de binnenhoeken van de hullranden steeds convex, kleiner dan 180 graden moeten zijn om aan de definitie te voldoen zoals geïllustreerd is in afbeelding A.2. Wanneer dit niet het geval is, zal er minstens 1 binnenhoek groter zijn dan 180 graden (reflex) en is het object concaaf. Een alternatieve definitie van de convex hull is [Center and Wexler [15]]:

Definitie A.1.2. CH(O) is de doorsnede van alle halfruimten die het object bevatten.

Een halfruimte is elk van de 2 deelruimten die een vlak in 3D voortbrengt. De convex hull wordt dus door een verzameling van vlakken begrensd. De convex hull ten op zichte van een view regio V is [Center and Wexler [15]]:

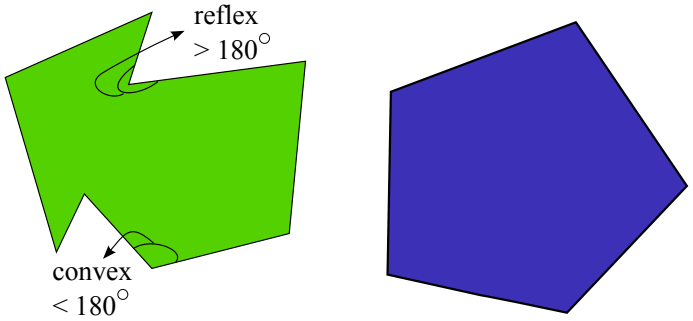
Definitie A.1.3. CH(O,V) is de doorsnede van alle halfruimten die uit silhouete-afbeeldingen van alle camera's $\in$ V worden voortgebracht en het object O bevatten.



(a) Theepot is een concaaf object.

(b) Cilinder is een convex object.

Figuur A.1: Illustratie van het verschil tussen convexe en concave objecten.



(a) Concave polygon: minstens 1 reflex hoek.

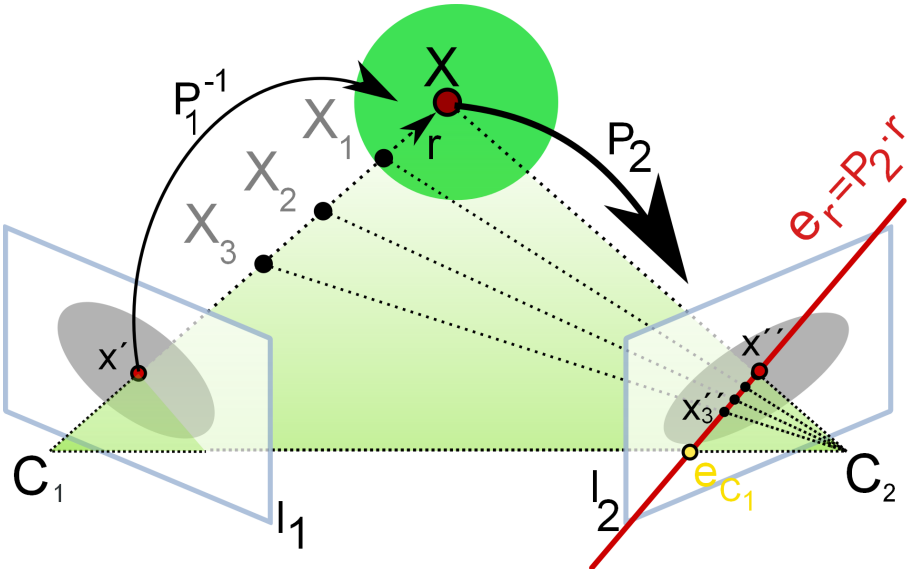
(b) Convexe polygon: allemaal convexe hoeken.

Figuur A.2: Illustratie van het verschil tussen convexe en concave objecten.

Bijlage B

Epipolaire geometrie

De epipolaire geometrie is de geometrie die tussen 2 camera's aanwezig is. Deze geometrie hangt af van de interne parameters van iedere camera en de relatieve positionering van een camera ten opzichte van een andere camera. Meer bepaald beschrijft ze de geometrie van de intersectie van de waaier van vlakken rond de camerabaseline (de lijn tussen de 2 cameraprojectiecentra) met het beeldvlak van iedere camera [Hartley and Zisserman [38]].



Figuur B.1: Illustratie van epipolaire geometrie. De epipolaire lijn van $r = P^{-1}x'$ is $e_r = P_2P^{-1}x'$ in het rood afgebeeld in I_2. In het geel is de epipoolafbeelding van C_1 op I_2 aangeduid. De groene, driehoekvormige, doorzichtige regio visualiseert een epipoolvlak.

Definitie B.0.4 (Epipool)**.** De epipool is de perspectiefprojectie-afbeelding van het projectiecentrum van de ene camera in het beeldvlak van de andere camera.

Definitie B.0.5 (Epipolair vlak)**.** Een vlak dat de camerabaseline bevat.

Definitie B.0.6 (Epipolaire lijn)**.** Een epipolaire lijn is het resultaat van de intersectie van een epipolair vlak met een beeldvlak.

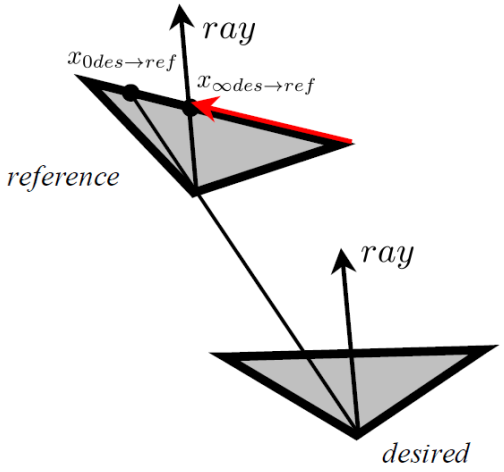
Anders geformuleerd is de epipolaire lijn de projectie-afbeelding van een ray vertrekkend vanuit het projectiecentrum van de ene camera door een pixel in diens beeldvlak afgebeeld in de andere camera. In figuur B.1 wordt de epipolaire geometrie en de verschillende terminologiën geïllustreerd.

B.1 Geldige epipolaire segmenten van een viewing ray

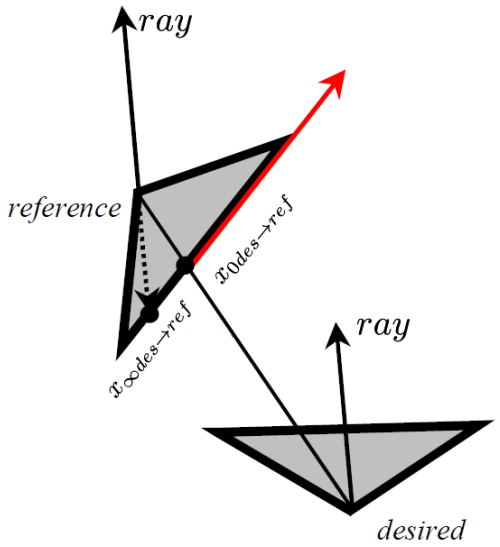
Wanneer de epipolaire geometrie zoals bijvoorbeeld bij de berekening van de beeldgebaseerde visual hul gebruikt wordt voor objectreconstructie aan de hand van 2D intersecties van een viewing ray r_{new} met andere afbeeldingen, mag enkel het geldige deel van de projectie-afbeelding van deze viewing ray met andere afbeeldingen worden geïntersecteerd. Een viewing ray is begrensd door het cameraprojectiecentrum en een punt op oneindig. Zo is het epipolaire segment op zijn beurt begrensd door de projectie-afbeeldingen van het cameraprojectiecentrum en dit punt op oneindig. Dit zijn respectievelijk de epipool x_0 en het vluchtpunt x_∞. De geldige richting van een epipoolsegment is niet steeds gelijk aan x_∞ - x_0, maar is afhankelijk van [Matusik et al. [70]]:

- de positie van het projectiecentrum van de ene camera C_{des} ten opzichte van de andere camera C_{ref}, meer bepaald bevindt camera C_{des} zich voor, of achter het beeldvlak van de camera C_{ref}. Dit is af te leiden uit het teken van de z-coördinaat van de epipoolafbeelding $x_{0_{des\rightarrow ref}}$ van het projectiecentrum op het beeldvlak van camera C_{ref}. Het teken van $x_{0_{z}des\rightarrow ref}$ bepaalt of er van de epipoolafbeelding weg moet worden geïtereerd ($x_{0_{z}des\rightarrow ref} > 0$) of naar de epipoolafbeelding toe ($x_{0_{z}des\rightarrow ref} < 0$) [McMillan [73]].
- de richting van de viewing ray r_{des} ten opzichte van het beeldvlak van camera C_{ref} wordt bepaald door het teken van $x_{\infty_{des\rightarrow ref}}$.

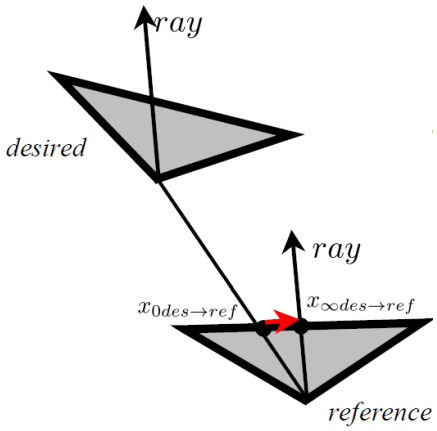
Een overzicht van de verschillende geldige epipolaire segmenten is getoond in figuren B.2 en B.3.



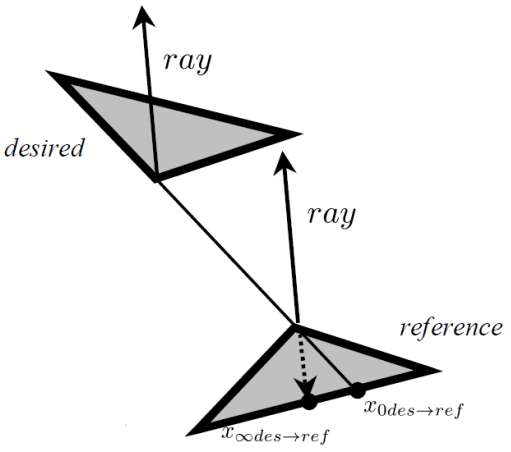
(a) $(x_{0_z des\to ref} < 0)$ en $(x_{\infty_z des\to ref} > 0)$. Geldig segment is de halfrechte die eindigt in $x_{\infty_{des\to ref}}$ met richting $x_{0_{des\to ref}} - x_{\infty_{des\to ref}}$.



(b) $(x_{0_z des\to ref} > 0)$ en $(x_{\infty_z des\to ref} < 0)$. Geldig segment is de halfrechte die start in $x_{0_{des\to ref}}$ met richting $x_{0_{des\to ref}} - x_{\infty_{des\to ref}}$.

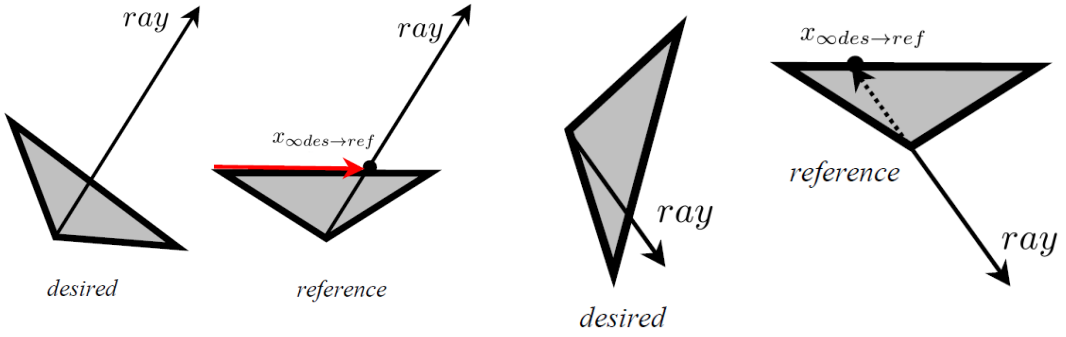


(c) $(x_{0_z des\to ref} > 0)$ en $(x_{\infty_z des\to ref} > 0)$. Segment is het lijnstuk tussen $x_{0_{des\to ref}}$ en $x_{\infty_{des\to ref}}$ met richting $x_{\infty_{des\to ref}} - x_{0_{des\to ref}}$.



(d) $(x_{0_z des\to ref} < 0)$ en $(x_{\infty_z des\to ref} < 0)$. Geen geldig segment.

Figuur B.2: Geldige epipolaire lijnsegmenten in het geval van niet-parallelle epipolaire lijnen $x_{0_z des\to ref} > 0$ of $x_{0_z des\to ref} < 0$ zijn aangeduid in het rood. Bewerkte afbeelding uit [Matusik et al. [70]].



(a) ($x_{\infty_z des \to ref} > 0$). Geldig segment is de halfrechte die eindigt in $x_{\infty_{des \to ref}}$ met richting $x_{\infty_{des \to ref}} - x_{0_{des \to ref}}$.

(b) ($x_{\infty_z des \to ref} < 0$). Geen geldig segment.

Figuur B.3: Geldige epipolaire lijnsegmenten in het geval van parallelle epipolaire lijnen $x_{0_z des \to ref} = 0$ zijn aangeduid in het rood. Bewerkte afbeelding uit [Matusik et al. [70]].

Bibliografie

[1] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, 1991.

[2] Ron Amos. Polynomial interpolation: Error analysis and introduction to splines. 2010.

[3] Kirby Baker. Interpolation for polynomial parametric curves. 2002.

[4] Brian A. Barsky and Anthony D. DeRose. Geometric continuity of parametric curves. Technical Report UCB/CSD-84-205, EECS Department, University of California, Berkeley, Oct 1984.

[5] Bruce Guenther Baumgart. *Geometric modeling for computer vision.* PhD thesis, Stanford, CA, USA, 1974. AAI7506806.

[6] Alexander Bogomjakov and Craig Gotsman. Reduced depth and visual hulls of complex 3d scenes. *Comput. Graph. Forum*, 27(2):175–182, 2008.

[7] Alexander Bogomjakov, Craig Gotsman, and Marcus Magnor. Free-viewpoint video from depth cameras. pages 89–96, 2006.

[8] Andrea Bottino and Aldo Laurentini. What's next? an interactive next best view approach. *Pattern Recognition*, 39(1):126 – 132, 2006. ISSN 0031-3203.

[9] Andrea Bottino, L. Cavallero, and Aldo Laurentini. Interactive reconstruction of 3-d objects from silhouettes. In *WSCG*, pages 230–236, 2001.

[10] Matthew Brand, Kongbin Kang, and David B. Cooper. Algebraic solution for the visual hull. In *CVPR (1)*, pages 30–35, 2004.

[11] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 425–432, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X.

[12] Adhemar Bultheel. *Inleiding tot de numerieke wiskunde*. Acco, 2006. ISBN 90-334-6253-2.

[13] Neill D. F. Campbell, George Vogiatzis, Carlos Hernandez, and Roberto Cipolla. Automatic object segmentation from calibrated images. In *Proceedings of the 2011 Conference for Visual Media Production*, CVMP '11, pages 126–137, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4621-6.

[14] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851. URL `http://dx.doi.org/10.1109/TPAMI.1986.4767851`.

[15] Wexler Chellappa Center and Y. Wexler. View synthesis using convex and visual hulls. In *In Proc. BMVC*, 2001.

[16] Sharat Chandrand. Lecture: Calibrated image acquisition for multi-view 3d reconstruction. 2009.

[17] Fu Chang, Chun jen Chen, and Chi jen Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93:206–220, 2004.

[18] Liang Chen. *3D Model Reconstruction From Silhouettes*. PhD thesis, 2008.

[19] Longbin Chen, Rogerio Feris, and Matthew Turk. Efficient partial shape matching using smith-waterman algorithm. In *In CVPR workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, 2008.

[20] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 279–288, New York, NY, USA, 1993. ACM. ISBN 0-89791-601-8.

[21] German K. M. Cheung. Visual hull construction, alignment and refinement for human kinematic modeling, motion tracking and rendering. 2003. AAI3114280.

[22] German K. M. Cheung, Takeo Kanade, Jean-Yves Bouguet, and Mark Holler. A real time system for robust 3d voxel reconstruction of human motions. In *CVPR*, pages 2714–2720, 2000.

[23] Roberto Cipolla and Peter J. Giblin. *Visual motion of curves and surfaces*. Cambridge University Press, 2000. ISBN 978-0-521-63251-5.

[24] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. *Algorithms*. 2006.

[25] I.H. de Boer, F.B. Sachse, S.Mang, and O. Dössel. Methods for determination of electrode positions in tomographic images. *International journal of bioelectromagnetism*, 2(2), 2000.

[26] Bert de Decker, Tom Mertens, and Philippe Bekaert. Interactive collision detection for free-viewpoint video. In *GRAPP (AS/IE)*, pages 114–120, 2007.

[27] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002. ISBN 1568811772.

[28] Charles R. Dyer. Volumetric scene reconstruction from multiple views. In *Foundations of Image Understanding*, pages 469–489. Kluwer, 2001.

[29] Peter Eisert. *Reconstruction of Volumetric 3D Models*, pages 133–150. John Wiley & Sons, Ltd. ISBN 9780470022733.

[30] Marcel Erné. The universal concept of closure, 2008.

[31] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Comput. Vis. Image Underst.*, 96(3): 367–392, December 2004. ISSN 1077-3142.

[32] Jean-Sébastien Franco and Edmond Boyer. Efficient polyhedral modeling from silhouettes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(3): 414–427, March 2009. ISSN 0162-8828.

[33] David Gallup, Jan-Michael Frahm, Philippos Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. In *CVPR*, 2008.

[34] Walter Gautschi. *Numerical analysis: an introduction*. Birkhäuser, 1997. ISBN 9780817638955.

[35] Bernd Girod. Lecture on scale space feature detection.

[36] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 013168728X.

[37] Kristen Grauman and Bastian Leibe. *Visual Object Recognition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

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

[39] Kuk jin Yoon and In so Kweon. Correspondence search in the presence of specular highlights using specular-free two-band images. In *In ACCV*, pages 761–770, 2006.

[40] Kenneth I. Joy. On-line computer graphics notes. 1999.

[41] S.B. Kang. *A survey of image-based rendering techniques*. Cambridge Research Laboratory technical report series. Digital, Cambridge Research Laboratory, 1997.

[42] S.B. Kang, Y. Li, and X. Tong. *Image-Based Rendering*. Foundations and trends in computer graphics and vision. 2007. ISBN 9781601980182.

[43] Luke Keele. *Semiparametric Regression for the Social Sciences*. Wiley, 2008. ISBN 978-0470319918.

[44] David Knezevic. Polynomial interpolation. Technical report, SEAS Institute, Harvard University, 2005.

[45] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984. ISSN 0301-0066.

[46] Kalin Kolev and Daniel Cremers. Beethoven dataset sampled on hemisphere.

[47] Daniel Kondermann. What is a useful definition of an edge in image processing, 2012.

[48] Kruger. Constrained cubic spline interpolation.

[49] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. Technical report, Rochester, NY, USA, 1998.

[50] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38:307–314, 2000.

[51] Alexander Ladikos, Selim Benhimane, and Nassir Navab. Efficient visual hull computation for real-time 3D reconstruction using CUDA. In *Proceedings of the 2008 Conference on Computer Vision and Pattern Recognition Workshops*, volume 0, pages 1–8, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[52] Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, February 1994. ISSN 0162-8828.

[53] Aldo Laurentini. How many 2d silhouettes does it take to reconstruct a 3d object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[54] Aldo Laurentini. Computing the visual hull of solids of revolution. *Pattern Recognition*, 32(3):377 – 388, 1999. ISSN 0031-3203.

[55] Svetlana Lazebnik. Projective visual hulls, 2002.

[56] Svetlana Lazebnik, Edmond Boyer, and Jean Ponce. On computing exact visual hulls of solids bounded by smooth surfaces. In *CVPR (1)*, pages 156–161, 2001.

[57] Svetlana Lazebnik, Yasutaka Furukawa, and Jean Ponce. Projective visual hulls. *International Journal of Computer Vision*, 74(2):137–165, 2007.

[58] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4.

[59] Ming Li. *Towards Real-Time Novel View Synthesis Using Visual Hulls*. PhD thesis, Universität des Saarlandes, 2004.

[60] Ming Li, Hartmut Schirmacher, Marcus A. Magnor, and Hans-Peter Seidel. Combining stereo and visual hull information for on-line reconstruction and rendering of dynamic scenes. In *IEEE Workshop on Multimedia Signal Processing*, pages 9–12, 2002.

[61] Ming Li, Marcus Magnor, and Hans peter Seidel. Hardware-accelerated rendering of photo hulls, 2004.

[62] Achim J. Lilienthal. Lecture on description and representation, 2011.

[63] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270, 1994.

[64] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1994. ISBN 0792394186.

[65] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, CVPR '96, pages 465–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7258-7.

[66] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. ISSN 0920-5691.

[67] Marcus A. Magnor. Video-based rendering. In *VRCIA*, page 279, 2006.

[68] Yasuyuki Matsushita. *Shadow elimination and interpolation for Computer Vision and Graphics*. PhD thesis, 2002.

[69] Wojciech Matusik and Leonard McMillan. Image-based visual hulls, 2001.

[70] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1-58113-208-5.

[71] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115–126, London, UK, UK, 2001. Springer-Verlag. ISBN 3-211-83709-4.

[72] Nicholas Frank Maunder. Virual view synthesis using visual hulls, 2005.

[73] Leonard McMillan, Jr. *An image-based approach to three-dimensional computer graphics*. PhD thesis, Chapel Hill, NC, USA, 1997. UMI Order No. GAX97-30561.

[74] Jason Meltzer and Stefano Soatto. Edge descriptors for robust wide-baseline correspondence, 2008.

[75] K Mikolajczyk., A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 779–788, 2003.

[76] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching, 2012.

[77] Gregor Miller and Adrian Hilton. Exact view-dependent visual hulls. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 01*, ICPR '06, pages 107–111, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2521-0. doi: 10.1109/ICPR.2006.515.

[78] Phillip Milne, Fred Nicolls, and Gerhard de Jager. Visual hull surface estimation. 2004.

[79] Niloy J. Mitra and Mark Pauly. Shadow art. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 156:1–156:7, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-858-2.

[80] J.L. Mundy and A. Zisserman. *Appendix - Projective Geometry for Machine Vision*. 1992.

[81] Wolfgang Niem and Ralf Buschmann. Automatic modelling of 3d natural objects from multiple views. In *In European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*. Springer, 1994.

[82] Raphael Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4.

[83] Richard S. Palais and Robert A. Palais. *Differential Equations, Mechanics, and Computation*. American Mathematical Society, 2009.

[84] Rick Parent. *Computer Animation, Second Edition: Algorithms and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2007. ISBN 0125320000.

[85] Sylvain Petitjean, Communicated Ming, C. Lin, and Dinesh Manocha. A computational geometric approach to visual hulls, 1997.

[86] Shrinivas Pundlik, Damon Woodard, and Stan Birchfield. Iris segmentation in non-ideal images using graph cuts. *Image and Vision Computing*, 28(12):1671 – 1681, 2010. ISSN 0262-8856.

[87] Juan M. Restrepo and Rob Indik. Interpolation. July 2001.

[88] Schaduw. Olympische spelen sydney, 2000. URL `http://www.silvoldeonline.nl/inschrijven-strandvolleybaltoernooi/a-beach-volleyball-competitor-smashes-a-ball/`.

[89] Schaduw. Australian open, 2008. URL `http://www.nocaptionneeded.com`.

[90] Schaduw. Nba all star celebrity game, 2009. URL `http://www.nba.com/multimedia/photo_gallery/0902/allstar09.celebritygame/content.1.html`.

[91] Schaduw. Ek: Duitsland-portugal, 2012.

[92] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, April 2002. ISSN 0920-5691.

[93] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'03, pages 195–202, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1900-8, 978-0-7695-1900-5.

[94] Steven Seitz and Charles Dyer. View morphing. In *SIGGRAPH 96*, pages 21–30, 1996.

[95] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.19.

[96] K. Shanmukh and Arun K. Pujari. Volume intersection with optimal set of directions. *Pattern Recognition Letters*, 12(3):165 – 170, 1991. ISSN 0167-8655.

[97] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *VCIP*, pages 2–13, 2000.

[98] Greg Slabaugh, Ron Schafer, and Mat Hans. Image-based photo hulls. *3D Data Processing Visualization and Transmission, International Symposium on*, 0:704, 2002.

[99] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[100] Timo Stich, Christian Linz, Georgia Albuquerque, and Marcus Magnor. View and time interpolation in image space. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 27(7):1781–1787, February 2008.

[101] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58(1):23–32, July 1993. ISSN 1049-9660.

[102] Richard Szeliski. Image alignment and stitching: A tutorial. Technical report, MSR-TR-2004-92, Microsoft Research, 2004, 2005.

[103] Cihan Topal, Cuneyt Akinlar, and Yakup Genc. Edge drawing: A heuristic approach to robust real-time edge detection. In *ICPR*, pages 2424–2427. IEEE, 2010. URL `http://ceng.anadolu.edu.tr/CV/EdgeDrawing/EDDemoUpload.aspx`.

[104] J. R Treichler and B. G. Agee. A new approach to multipath correction of constant modulus signals. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 31:459–472, 1983.

[105] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography (presentation slides). In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 287–296, New York, NY, USA, 2000. ISBN 1-58113-208-5.

[106] Xiaojun Wu, Osamu Takizawa, and Takashi Matsuyama. Parallel pipeline volume intersection for real-time 3d shape reconstruction on a pc cluster. In *ICVS*, page 4, 2006.

[107] D. Xia, D. Li, Q. Li, and S. Xu. A novel approach for computing exact visual hull from silhouettes. *Optik*, 122:2220–2226, December 2011.

[108] Jiangjian Xiao, Cen Rao, Mubarak Shah, and Short Presentations. View interpolation for dynamic scenes, 2002.

[109] Shuntaro Yamazaki, Srinivasa G Narasimhan, Simon Baker, and Takeo Kanade. On using shadowgrams for visual hull reconstruction. Technical Report CMU-RI-TR-07-29, Robotics Institute, Pittsburgh, PA, August 2007.

[110] Won Young Yang, Wenwu Cao, Tae-Sang Chung, and John Morris. *Applied numerical methods using MATLAB*. Wiley, 2005. ISBN 9780471698333.

[111] E.C. Zeeman. *An Introduction to Topology: The Classification Theorem for Surfaces*. Mathematics Institute, University of Warwick, 1966.

[112] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, August 2004. ISSN 0730-0301.