



UNIVERSITEIT
GENT

Faculteit Toegepaste Wetenschappen
Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: Prof. dr. ir. H. Bruneel

Samenvatting van videosequenties m.b.v. sleutelbeelden

door Hiêp Quang Luong

Promotor: prof. dr. ir. W. Philips
Thesisbegeleider: dr. ir. P. De Smet en ir. M. De Geyter

Afstudeerwerk ingediend tot het behalen van de academische graad
van burgerlijk ingenieur in de computerwetenschappen

Academiejaar 2002–2003

Toelating tot bruikleen

De auteur geeft de toelating dit afstudeerwerk voor consultatie beschikbaar te stellen en delen van het afstudeerwerk te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit dit afstudeerwerk.

Hiệp Quang Luong

30 mei 2003

Woord vooraf

Bij het voorleggen van deze scriptie, zou ik graag iedereen willen bedanken die heeft bijgedragen tot de verwezenlijking van dit eindwerk. Ik wens dan ook in de eerste plaats mijn promotor prof. dr. ir. W. Philips danken voor het scheppen van de mogelijkheid dit onderzoek te verrichten. In het bijzonder wens ik ook mijn thesisbegeleiders dr. ir. P. De Smet en ir. M. De Geyter te danken die mij oordeelkundig begeleid hebben bij het maken van deze scriptie. Verder wens ik iedereen binnen telin te bedanken voor hun hulp en tips.

Mijn dank gaat ook naar iedereen die me op één of andere manier steun verleend heeft tijdens het maken van mijn scriptie, in het bijzonder mijn familieleden die me bij moeilijke momenten altijd geholpen hebben. Verder wens ik mijn vrienden te bedanken voor het nalezen van de thesistekst en die er mede voor gezorgd hebben dat deze periode aangenaam en plezant verlopen is. Ook dank ik mijn medestudenten voor de hulp en hun geduldige antwoorden op mijn lastige vragen. En tenslotte dank ik ook de makers van mijn besturingssysteem en mijn PC voor het beperkt aantal crashes tijdens het programmeren en het schrijven van mijn scriptie.

Samenvatting van videosequenties m.b.v. sleutelbeelden

door
Hiệp Quang Luong

Afstudeerwerk ingediend tot het behalen van de academische graad van burgerlijk ingenieur in de computerwetenschappen

Academiejaar 2002–2003

Universiteit Gent
Faculteit Toegepaste Wetenschappen

Promotor: prof. dr. ir. W. Philips

Samenvatting

PC's worden steeds krachtiger. Tevens komen snellere internetverbindingen (b.v. ADSL of kabel) in het bereik van de gewone gebruiker, waardoor deze nu ook toegang heeft tot een enorme hoeveelheid digitale video en audiomateriaal.

Om zijn weg te kunnen vinden in deze informatievloed zou de gebruiker erg gebaat zijn met een inhoudsgebaseerd zoekstelsel: hij zou zo b.v. op zoek kunnen gaan naar alle videobestanden waarin iemand vanaf een brug op een rijdende trein springt, en hoeft dan enkel de desbetreffende scène van deze bestanden te bekijken (zogenoemd content-based access). Het spreekt vanzelf dat dit voorlopig nog toekomstmuziek is: dit vergt immers het kunnen herkennen van objecten, scènes, het kunnen interpreteren van de zoekopdracht van de consument en nog veel meer.

In dit werk zetten we een eerste stap in deze richting door videofragmenten samen te vatten met behulp van sleutelbeelden. De verschillende shots in een beeldsequentie worden gedetecteerd en voorgesteld door één beeld. We bestuderen hierbij een aantal verschillende methoden om aan shotdetectie te doen. Te gelijkende shots worden geclusterd tot scènes (denk b.v. aan dialogen waarbij telkens op de spreker ingezoomd wordt) en uit deze scènes extraheren we de sleutelbeelden. Voor shots met camerabewegingen genereren we panoramische beelden. De sleutelbeelden vormen de samenvatting en worden weergegeven op een gebruikersvriendelijke webpagina, waarbij een klik op een beeld er voor zorgt dat de gewenste beeldsequentie wordt afgespeeld.

Trefwoorden: temporele videosegmentatie, transitie- en cutdetectie, scèneclustering, panoramische beelden, beeldregistratie, shotrepresentatie

Inhoudsopgave

1	Inleiding	1
2	Literatuurstudie: temporele videosegmentatie en scèneclustering	4
2.1	Inleiding	4
2.2	Temporele segmentatie op ongecomprimeerde video	6
2.2.1	Paarsgewijze pixelvergelijking	6
2.2.2	Blokgebaseerde detectie	8
2.2.3	Histogramvergelijking	8
2.2.4	Tweelingsvergelijking	12
2.2.5	Randdetectie	12
2.2.6	Modelgebaseerde segmentatie	15
2.2.7	Spatiaal-temporele snede-analyse	18
2.2.8	Zelf-similariteitsanalyse	26
2.2.9	Andere kenmerken	27
2.2.10	Reductie van de rekentijd	29
2.3	Temporele segmentatie op gecomprimeerde video	31
2.3.1	Beknopte overzicht van MPEG	31
2.3.2	DCT-coëfficiënten	32
2.3.3	DC-beelden	33
2.3.4	Macroblokken	35
2.3.5	Bewegingsvectoren	36
2.3.6	Variabele bitrate	37
2.4	Scèneclustering	38
2.4.1	K-means algoritme	38
2.4.2	Iteratieve boomclustering	39
2.4.3	Temporele correctie en integratie van domeinkennis	41
3	Shotrepresentatie	42
3.1	Inleiding	42
3.2	Selectie van sleutelbeelden	42
3.3	Panoramische beelden	46
3.3.1	Registratie	46
3.3.2	Plaatsing op het canvas	63
3.4	Spatiaal-temporele volumes	67
3.5	Belangrijkheid van een shot of scène	69
3.6	Compacte voorstelling van sleutelbeelden	70
3.6.1	Exhaustieve blokmethode	71

3.6.2	Heuristische plaatsingsmethode	73
3.7	Interactieve webpagina	75
4	Experimentele resultaten	77
4.1	Inleiding	77
4.2	Omgeving	77
4.2.1	Bibliotheken	77
4.2.2	Testmateriaal	78
4.3	Temporele videosegmentatie en scèneclustering	78
4.3.1	Performantiematen	78
4.3.2	Performantie: detectie resultaten	79
4.3.3	Rekentijden van de algoritmen	88
4.3.4	Robuustheid tegen spatiale reductie	89
4.3.5	Scèneclustering	90
4.4	Panoramische beelden	91
4.4.1	Registratie	91
4.4.2	Kwaliteit van een panoramisch beeld	96
5	Besluit	101
A	Drempels	103
B	De Kanade-Lucas-Tomasi tracker	107

Hoofdstuk 1

Inleiding

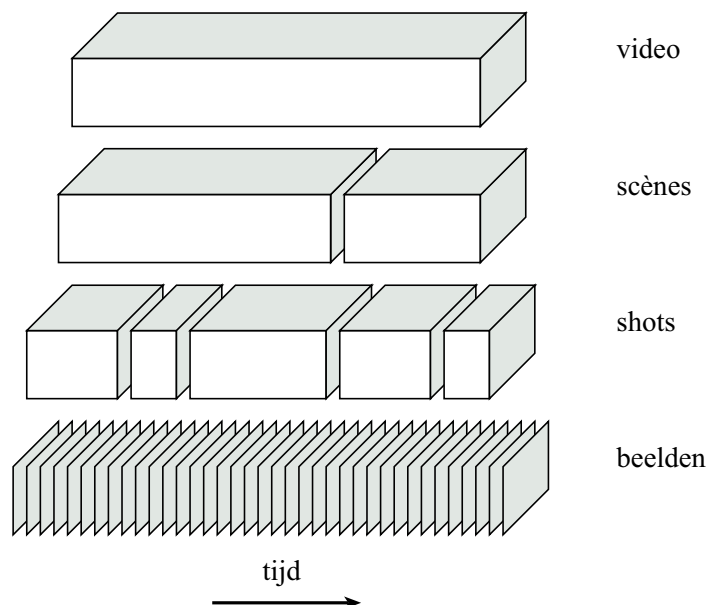
Recente ontwikkelingen in de multimedia compressietechnologie, gekoppeld aan een beduidende groei van computerperformantie en de wereldwijde groei van het internet, leiden tot het gebruik en de beschikbaarheid van een enorme hoeveelheid digitaal video- en audiomateriaal. Toepassingen zoals digitale bibliotheken, digitale televisie, video-op-aanvraag en multimediale informatiesystemen genereren en gebruiken grote hoeveelheden aan videomateriaal.

Om deze enorme informatievloed zo efficiënt mogelijk te kunnen gebruiken is er nood aan een inhoudsgebaseerd zoekstelsel: een systeem dat videodatabanken kan beheren om het opzoeken van relevante videosequenties op basis van hun inhoud mogelijk te maken. De gebruiker zou b.v. op zoek kunnen gaan naar alle videobestanden waarin iemand vanaf een brug op een rijdende trein springt, en hoeft enkel de desbetreffende scène van deze bestanden te bekijken (de zogeheten *content-based access*). Dit is voorlopig nog toekomstmuziek: dit vergt immers het kunnen herkennen van objecten, scènes, verhaallijnen, het kunnen interpreteren van de zoekopdracht van de gebruiker en nog veel meer.

Een inhoudsgebaseerd zoekstelsel bestaat uit twee delen, het syntactisch gedeelte (domeinonafhankelijk) en het semantisch gedeelte (domeinspecifiek). De indexerings van videosequenties voor een inhoudsgebaseerd zoekstelsel gebeurt in vijf stappen: *modellering, segmentatie, extractie, representatie en organisatie*. In een conventioneel databanksysteem, is de toegang tot data gebaseerd op de verschillende attributen of eigenschappen van welgedefinieerde objecten binnen een specifiek domein. Voor ongestructureerde data zoals video, audio of beelden kunnen gelijkaardige attributen gedefinieerd worden. Vanuit het structureel standpunt kan video gemodelleerd worden door segmenten gesynchroniseerde audio van eindige lengtes en door een reeks stilstaande beelden. Dit model is een simpel voorbeeld van de meer algemenere modellen voor heterogene multimedia data-objecten. Video kan ook voorgesteld worden door een reeks basisblokken, ook wel shots genoemd: dit zijn onafgebroken opgenomen segmenten video/audio.

Een tweede stap in de richting van automatische annotatie van digitale videosequenties is temporele videosegmentatie. Het doel is om een videostroom in een set van betekenisvolle en bruikbare segmenten (*shots*) op te splitsen. Deze basiselementen worden gebruikt voor indexerings. Vervolgens worden de verscheidene shots gegroepeerd tot scènes met een meer zinvolle betekenis. Om scènes in hoofdstukken te groeperen, zou de computer de verhaallijn van de film moeten herkennen. Dit is echter nog toekomstmuziek. Temporele videosegmentatie kan het best vergeleken worden met een tekst die in woorden

wordt opgedeeld, waarbij de tekst de videosequentie voorstelt. Hierbij zijn de zinnen de scènes; de woorden (basisblokken) zijn de shots en de letters stellen dan de afzonderlijke beelden voor zoals aangetoond in figuur 1.1.



Figuur 1.1: Structurele splitsing van video in *scènes* en *shots*.

In het extractieproces wordt de inhoud uit de multimediale data geëxtraheerd en voorgesteld door tekst of symbolen. De bekomen informatie wordt dan gerepresenteerd op basis van domeinspecifieke attributen.

Met de huidige traditionele databanktechnologie kan men onvoldoende de aanvragen van multimediale data op een correcte manier behandelen. We kunnen beelden nog niet voldoende efficiënt indexerend en ophalen op basis van een paar sleutelwoorden. Visuele data wordt verschillend ervaren door verschillende personen, waardoor er verscheidene interpretaties van dezelfde data mogelijk zijn. Alle mogelijke interpretaties voorstellen door sleutelwoorden zou leiden tot een explosieve groei van de databank. Om andere methoden te kunnen gebruiken, moet de databank op een andere manier georganiseerd worden. Opvragen van videosequenties kan gebeuren op basis van inhoud. De gebruiker kan vorm- en/of kleurcriteria opgeven, maar ook een schets van de scène behoort tot de mogelijkheden. De gebruiker zou ook op basis van iconen een aanvraag kunnen formuleren. De gebruiker selecteert hiervoor een set van iconen, die sterke gelijkenissen vertonen met de videosequentie (veelal een sterk verkleinde versie van een representatief sleutelbeeld). Aanvragen kunnen tenslotte ook geformuleerd worden door een combinatie van de vorige methoden.

Een samenvatting van videosequenties kan resulteren in een nieuw videosequentie [LPE97, Lie99a]. De nieuwe videosequentie is dan een verkorte versie van het origineel, zoals *trailers* in de filmwereld. Maar in dit werk concentreren we ons vooral op een samenvatting gebaseerd op sleutelbeelden, omdat dit een basis vormt voor het inhoudsgebaseerd

zoeksysteem.

Het hoofddoel van deze scriptie is een programma ontwikkelen die automatisch visueel attractieve samenvattingen kan genereren van videosequenties. Om dit doel te bereiken, moeten we de videosequentie eerst temporeel segmenteren. We bestuderen hierbij een aantal verschillende methoden om aan shotdetectie te doen, dit wordt besproken in hoofdstuk 2. Eens we de verschillende shots hebben, kunnen we de bijbehorende shots clusteren tot scènes.

Om visuele samenvattingen te maken, extraheren we geschikte sleutelbeelden uit de desbetreffende scènes. Shots met camerabewegingen worden voorgesteld door panoramische beelden. Deze sleutelbeelden worden dan op een webpagina geplaatst, zoals beschreven in hoofdstuk 3.

In hoofdstuk 4 wordt een vergelijkende studie gemaakt van de verschillende shotdetectie methoden. Verder wordt de kwaliteit van een panoramisch beeld onderzocht.

Tenslotte trekken we in hoofdstuk 5 enkele conclusies uit dit werk.

Hoofdstuk 2

Literatuurstudie: temporele videosegmentatie en scèneclustering

2.1 Inleiding

Het doel van temporele videosegmentatie is het opdelen van de videosequenties in zinvolle basisblokken, nl. shots. Een shot is een stuk opgenomen videosequentie dat continu is in tijd, ruimte en grafische configuratie, dwz. alles wat ononderbroken opgenomen wordt door één camera.

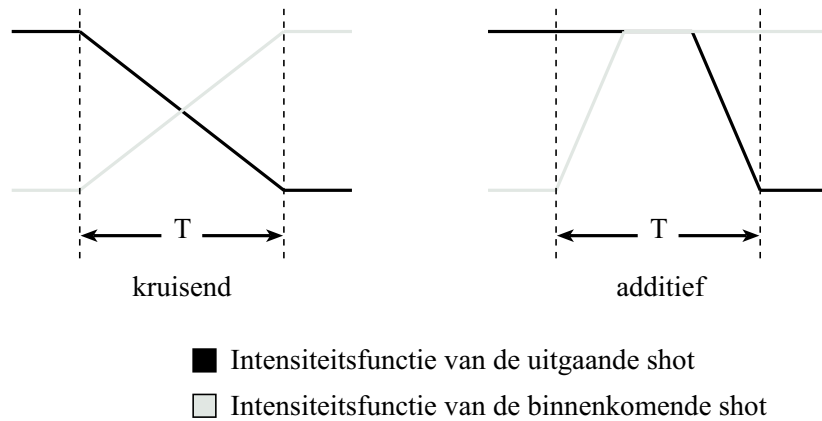
Er zijn twee soorten van shottransities: abrupte en graduele overgangen. Abrupte overgangen worden ook cuts genoemd. Ze overspannen slechts één beeld: wanneer de camera stopt en herstart, zoals aangegeven in figuur 2.1. De cuts zijn de simpelste overgangen. Daarnaast kunnen cinematografische effecten gebruikt worden om twee shots te combineren. Deze overgangen worden gecreëerd op de computer en worden ook graduele overgangen of kortweg transities genoemd. Er worden hoofdzakelijk slechts twee types gebruikt: dissolve en wipe. Er bestaan naast deze twee types nog honderden verschillende types met parameters die men eindeloos kan instellen.



Figuur 2.1: Een videosequentie met een cut tussen beeld 3 en 4.

Bij dissolves vloeit het ene beeld langzaam in het andere beeld, door de intensiteit van de twee beelden te laten variëren. In [Lie01a] worden de twee meestgebruikte soorten dissolves besproken: de kruisende en additieve dissolves, zoals voorgesteld in figuur 2.2. Speciale gevallen van dissolves zijn fade-in, fade-out en morfen. Fade-in (fade-out) wordt bekomen door de intensiteit van het beeld (linear) te doen stijgen (dalen). Morfen is het geleidelijk aan vervormen van een object tot een andere object. Technisch gezien is

morfen een overgang, maar vanuit het semantische standpunt niet, daarom wordt morfen genegeerd bij detectie van transities. Figuur 2.3 toont verschillende dissolves.



Figuur 2.2: Intensiteitsfuncties van twee populaire dissolve-types.

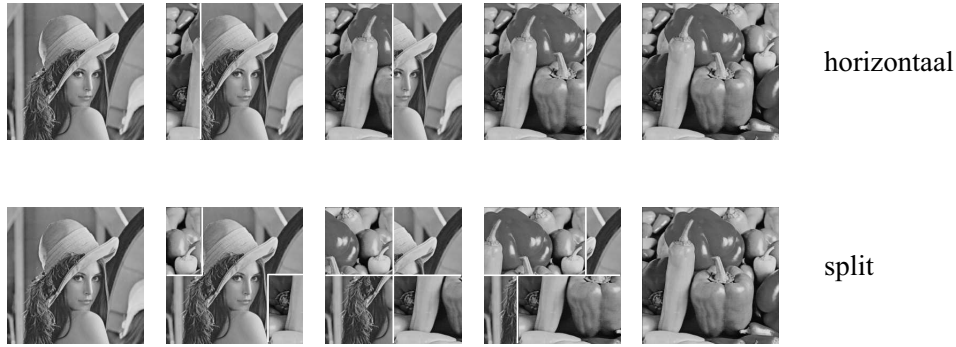


Figuur 2.3: Verschillende soorten dissolves.

Wipes is een totaal andere type van transitie: de nieuwe shot komt tevoorschijn door een bewegende grens in de vorm van een lijn of patroon. Figuur 2.4 toont enkele variaties van wipes.

Meer dan tien jaar onderzoek in temporele videosegmentatie heeft een grote variëteit aan algoritmen opgeleverd. In het begin werd er vooral onderzoek verricht naar cutdetectie, meer recente technieken behandelen moeilijkere probleemgevallen, nl. detectie van graduele overgangen en detectie van camerabewegingen.

In de volgende secties worden de belangrijkste algoritmen besproken. Methoden om geschikte drempels te kiezen, worden uitgelegd in bijlage A.



Figuur 2.4: Twee soorten wipes.

2.2 Temporele segmentatie op ongecomprimeerde video

2.2.1 Paarsgewijze pixelvergelijking

Een simpele manier om een kwalitatieve verandering tussen twee beelden te bepalen is het vergelijken van de corresponderende pixels in de twee beelden. Men berekent hiervoor de som van het absolute verschil van de grijswaarden en vergelijkt deze met een bepaalde drempel δ :

$$D(i, i + 1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y |P_i(x, y) - P_{i+1}(x, y)|}{X \cdot Y} \quad (2.1)$$

waarbij i en $i + 1$ de opeenvolgende beelden zijn met dimensie $X \times Y$. $P_i(x, y)$ stelt de grijswaarde voor van de pixel op coördinaten (x, y) in beeld i . Analoog geldt voor kleurenbeelden:

$$D(i, i + 1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y \sum_c |P_i(x, y, c) - P_{i+1}(x, y, c)|}{X \cdot Y} \quad (2.2)$$

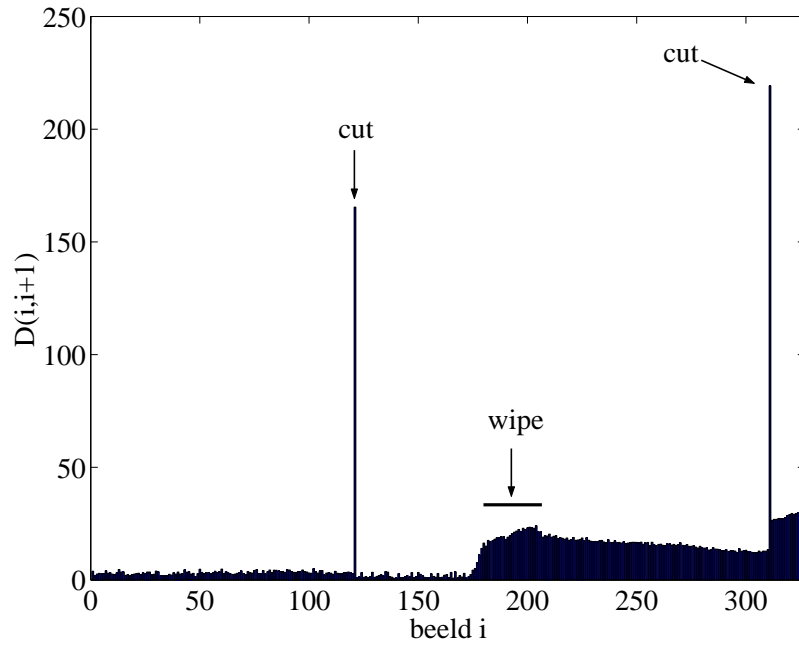
hierbij is c de index van de kleurcomponent (b.v. $c \in \{rood, groen, blauw\}$ in het geval van een RGB-kleurensysteem) en is $P_i(x, y, c)$ de kleurcomponent van de pixel op (x, y) in beeld i .

Een cut tussen twee opeenvolgende beelden wordt gedetecteerd als:

$$D(i, i + 1) > \delta \quad (2.3)$$

Het grootste nadeel van deze methode is dat het niet mogelijk is om een onderscheid te maken tussen een groot verschil in een klein gebied en een klein verschil in een groot gebied. De methode is hierdoor zeer gevoelig voor camera- en objectbeweging.

In figuur 2.5 wordt $D(i, i + 1)$ getoond voor een videosequentie met 2 cuts en 1 wipe. $D(i, i + 1)$ die relatief hoge waarden aannemen maar die niet behoren tot de wipe of de cuts zijn afkomstig van camera- en objectbewegingen.



Figuur 2.5: Een sequentie van $D(i, i + 1)$ van de paarsgewijze pixelvergelijking.

Een verbetering is het tellen van het aantal pixels die van waarde, groter dan een drempel δ_1 , veranderen en deze dan vergelijken met een tweede drempel δ_2 :

$$DP(i, i + 1, x, y) = \begin{cases} 1, & \text{als } |P_i(x, y) - P_{i+1}(x, y)| > \delta_1 \\ 0, & \text{anders} \end{cases} \quad (2.4)$$

$$D(i, i + 1) = \frac{\sum_{x=1}^X \sum_{y=1}^Y DP(i, i + 1, x, y)}{X \cdot Y}$$

Een cut wordt gedetecteerd als het percentage van het aantal veranderde pixels groter ligt dan drempel δ_2 . Hoewel deze techniek irrelevante pixelveranderingen uitfiltert, blijft ze gevoelig voor camera- en objectbeweging. Het effect van deze bewegingen kan een stuk gereduceerd worden door eerst een gemiddeldewaardefilter op de beelden toe te passen: elke pixel wordt vervangen door het gemiddelde van zijn omgeving. Het lineair filtermasker voor een 5×5 -gemiddeldewaardefilter wordt gegeven door:

$$M = \frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (2.5)$$

2.2.2 Blokgebaseerde detectie

In tegenstelling tot de pixelvergelijking, die gebaseerd is op de globale beeldkarakteristiek, gebruiken blokgebaseerde technieken lokale kenmerken om de robuustheid tegen camera- en objectbeweging te verhogen. Elk beeld i wordt verdeeld in b blokken en deze blokken worden vergeleken met de corresponderende blokken in beeld $i + 1$. Het verschil tussen beeld i en $i + 1$ wordt gegeven door:

$$D(i, i + 1) = \sum_{k=1}^b c_k \cdot D_k(i, i + 1) \quad (2.6)$$

waarbij c_k een voorgedefinieerde coëfficiënt is voor blok k en $D_k(i, i + 1)$ een partiële vergelijkingswaarde is van het k -de blok tussen de beelden i en $i + 1$.

De corresponderende blokken worden vergeleken via de waarschijnlijkheidsverhouding:

$$\lambda_k = \frac{\left[\frac{\sigma_{k,i}^2 + \sigma_{k,i+1}^2}{2} + \left(\frac{\mu_{k,i} - \mu_{k,i+1}}{2} \right)^2 \right]^2}{\sigma_{k,i}^2 \cdot \sigma_{k,i+1}^2} \quad (2.7)$$

waarbij $\mu_{k,i}$ en $\mu_{k,i+1}$ de gemiddelde intensiteitswaarden zijn van blok k van de opeenvolgende beelden i en $i + 1$, en $\sigma_{k,i}^2$ en $\sigma_{k,i+1}^2$ de respectievelijke varianties. De blokken waarvoor de waarschijnlijkheidsverhouding groter is dan een drempel δ_1 worden opgeteld:

$$D_k(i, i + 1) = \begin{cases} 1, & \text{als } \lambda_k > \delta_1 \\ 0, & \text{anders} \end{cases} \quad (2.8)$$

Als het aantal gewijzigde blokken groter is dan een drempel δ_2 , is er een cut gevonden.

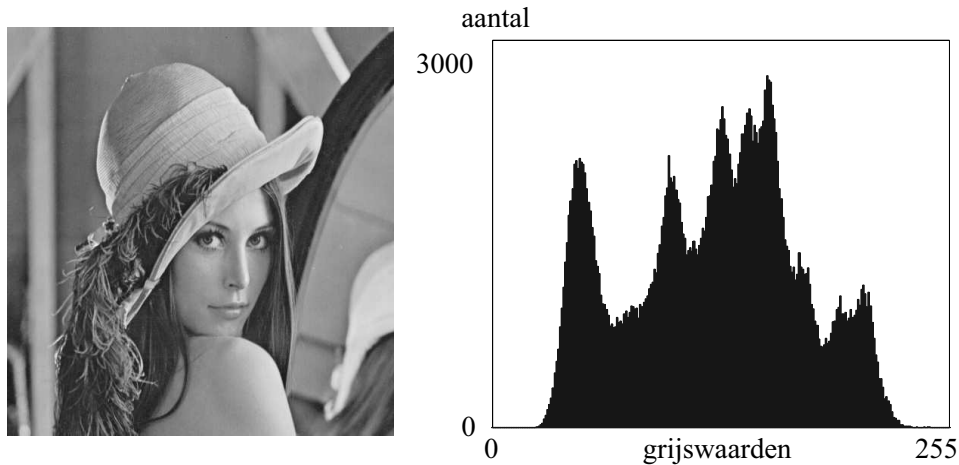
Vergeleken met pixelvergelijking is deze methode toleranter voor trage en kleine objectbeweging, maar is wel complexer en rekenintensiever. Een ander potentieel probleemgeval is dat er geen verandering wordt gedetecteerd indien twee blokken verschillend zijn maar toch dezelfde dichtheidsfunctie hebben. Deze situatie komt zelden voor.

Een andere methode is het kiezen van n niet-overlappende blokken in een beeld, de zogenaamde basisvensters. Voor elk basisvenster wordt de beste overeenkomst gevonden in de respectievelijke omgeving van het vorige beeld op basis van de intensiteitswaarden of kleurwaarden. De basisvensters worden groot genoeg gekozen om ongevoelig te zijn aan veranderingen die geen cut zijn en klein genoeg om de spatiale informatie te behouden. Een niet-lineaire statistische sorteerfilter wordt gebruikt om de gevonden gekoppelde waarden te combineren. Het gewicht dat in vergelijking (2.6) toegekend wordt, zal afhangen van de rang in de gesorteerde lijst. De effecten van camera- en objectbewegingen worden hierdoor sterk gereduceerd. Cuts en transitieën kunnen gedetecteerd worden: cuts worden gevonden via vergelijking met een drempel δ , terwijl transitieën gevonden worden via detectie van monotone stijgingen van de gekoppelde waarden. Deze methode is zeer complex.

2.2.3 Histogramvergelijking

Een stap verder om camera- en objectbeweging te reduceren is het gebruik van histogrammen van de beelden. Het idee achter deze methode is dat twee beelden met dezelfde

achtergrond en met dezelfde (maar toch bewegende) objecten slechts een miniem histogramverschil hebben. Histogrammen zijn ook onverschillig voor rotaties en verschillen maar een beetje als de kijkhoek of schaling verandert. Een nadeel is dat twee beelden dezelfde histogrammen kunnen hebben en toch een totaal andere inhoud bevatten. De kans voor zulke situaties is zeer klein. Een grijswaarde-(kleur-)histogram van beeld i is een n -dimensionele vector H_i en $H_i(j)$ ($j = 1, \dots, n$) het aantal pixels van beeld i met grijswaarde (kleur) j en waarbij n het totaal aantal grijswaarden (kleuren) is. In figuur 2.6 wordt een voorbeeld gegeven van een histogram.



Figuur 2.6: Voorbeeld van een histogram.

Globale histogramvergelijking

Een cut wordt gevonden indien de som van de absolute histogramverschillen van de opeenvolgende beelden groter is dan drempel δ . Het verschil tussen twee opeenvolgende beelden wordt gegeven door:

$$D(i, i + 1) = \sum_{j=1}^n |H_i(j) - H_{i+1}(j)| \quad (2.9)$$

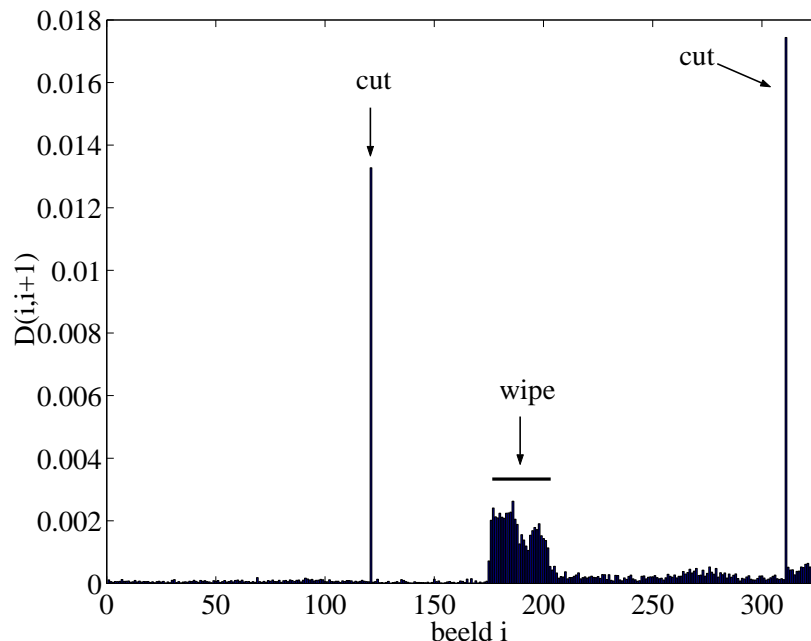
Bij kleurhistogrammen wordt de lengte van het histogram beperkt (3 kleuren van 8 bits geeft een histogram met 2^{24} elementen) door een kleurcode te gebruiken: enkel de twee meest significante bits van elke kleurintensiteit worden gebruikt. De vergelijking van de 64 elementen is genoeg om overgangen te bepalen.

Er kunnen ook gewichten toegekend worden aan elk element van het histogram:

$$D(i, i + 1) = \sum_{j=1}^n \sum_{k \in N_j} W_k \cdot |H_i(j) - H_{i+1}(k)| \quad (2.10)$$

waarbij N_j de omgeving is van element j en W_k het gewicht dat toegekend is aan een buur. Een 3×3 omgeving of 2 burens worden gebruikt respectievelijk in het geval van een 2-dimensionale en een 1-dimensionale histogram.

In figuur 2.7 wordt $D(i, i + 1)$ getoond voor een videosequentie. Bemerkt de verbeteringen ten opzichte van de paarsgewijze pixelvergelijking in figuur 2.5.



Figuur 2.7: Een sequentie van $D(i, i + 1)$ van de globale histogramvergelijking.

Om het verschil van twee histogrammen te berekenen kan ook een χ^2 -test gebruikt worden:

$$D(i, i + 1) = \sum_{j=1}^n \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} \quad (2.11)$$

Als het verschil groter is dan een drempel δ , dan is er een cut gevonden.

Experimentele resultaten hebben aangetoond dat de χ^2 -test niet alleen de globale verschillen tussen twee beelden vergroten, maar ook de verschillen veroorzaakt door camera- en objectbewegingen worden hierdoor vergroot. Daardoor is een χ^2 -test niet noodzakelijk beter dan de lineaire histogramvergelijking van vergelijking (2.9) en vereist hij ook meer rekentijd.

Andere statistische testen zoals de Kolmogorov-Smirnov en de Yakimovski waarschijnlijkheidstest presteren slechter dan de χ^2 -test [SP95, PS97].

Een andere manier om twee histogrammen te vergelijken is via histogramintersectie (voorgesteld in [KSGA96]):

$$D(i, i + 1) = 1 - \text{Intersectie}(H_i, H_{i+1}) = 1 - \frac{\sum_{j=1}^n \min(H_i(j), H_{i+1}(j))}{\sum_{j=1}^n \max(H_i(j), H_{i+1}(j))} \quad (2.12)$$

Voor twee identieke histogrammen is de intersectie 1 (en het verschil 0); terwijl voor twee histogrammen die geen enkel gemeenschappelijke pixel van dezelfde intensiteitswaarde of kleurcomponent hebben, de intersectie 0 is (en het verschil 1).

Lokale histogramvergelijking

Zoals reeds besproken is de globale histogramvergelijking simpel en meer robuust tegen camera- en objectbeweging, maar behoudt het geen spatiale informatie en zal dus bijgevoelen indien twee verschillende beelden gelijkaardige histogrammen opleveren. Blokgebaseerde methoden behouden spatiale informatie, ze presteren beter dan pixelvergelijking maar zijn nog steeds gevoelig aan camera- en objectbewegingen. Door de twee paradigma's samen te voegen, kunnen fout gedetecteerde cuts en transitie door camera- en objectbeweging verder gereduceerd worden terwijl genoeg spatiale informatie behouden wordt om betere resultaten te verkrijgen.

Het verschil tussen de opeenvolgende beelden wordt berekend op de volgende wijze:

$$D(i, i + 1) = \sum_{k=1}^b D_k(i, i + 1) \quad (2.13)$$

$$D_k(i, i + 1) = \sum_{j=1}^n |H_i(j, k) - H_{i+1}(j, k)|$$

waarbij $H_i(j, k)$ de histogramwaarde voorstelt van grijswaarde j van blok k en b het totaal aantal blokken waarin het beeld verdeeld is.

Een ander probleemgeval bij histogramvergelijking is de belichting: valt er een andere belichting op het beeld, dan zullen de histogrammen sterk verschillen. Een selectief HSV-histogram kan gebruikt worden om verschillen, veroorzaakt door intensiteitsverschillen, te reduceren [KC01]. De beeldblokken worden vergeleken in HSV-ruimte (kleurtint, saturatie, helderheid). Het gebruik van kleurtinten maakt het algoritme onverschillig tegenover saturatie- en intensiteitsveranderingen. Kleurtinten zijn wel onstabiel als saturatie en helderheid lage waarden aannemen, vandaar de selectieve vergelijking. Als een pixel veel kleurinformatie bevat (hoge saturatie- en helderheidswaarden, respectievelijk hoge S en V) wordt het geklasseerd in een discrete kleur gebaseerd op kleurtinten, anders op grijswaarden. De selectieve histogrammen $H_{kleurtint}$ en H_{grijs} en het verschil tussen beelden voor blok k met dimensie $X \times Y$ wordt gegeven door:

$$\begin{aligned}
H_{i,k}^{kleurtint}(h) &= \sum_{x=1}^X \sum_{y=1}^Y I_i^{kleurtint}(x, y, h) & H_{i,k}^{grijs}(g) &= \sum_{x=1}^X \sum_{y=1}^Y I_i^{grijs}(x, y, g) \\
I_i^{kleurtint}(x, y, h) &= \begin{cases} 1, & \text{als } S_i(x, y, h) > \delta_s \text{ en } V_i(x, y, h) > \delta_v \\ 0, & \text{anders} \end{cases} \\
I_i^{grijs}(x, y, g) &= \begin{cases} 1, & \text{als } S_i(x, y, g) \leq \delta_s \text{ en } V_i(x, y, g) \leq \delta_v \\ 0, & \text{anders} \end{cases}
\end{aligned} \tag{2.14}$$

$$D_k(i, i+1) = \sum_{h=1}^N \left| H_{i,k}^{kleurtint}(h) - H_{i+1,k}^{kleurtint}(h) \right| + \sum_{g=1}^M \left| H_{i,k}^{grijs}(g) - H_{i+1,k}^{grijs}(g) \right|$$

met h en g de indexen voor respectievelijk kleurtint en grijswaarde; δ_s en δ_v zijn de drempels voor respectievelijk saturatie- en helderheidswaarden.

Er zijn nog talloze varianten mogelijk: andere histogramafstandsmaten (b.v. cosinus-similariteit, ... [OSMM99, HSE⁺95]), andere kleurenruimten (b.v. YCrCb, YUV, YIQ, L*a*b*, Munsell, ... [KSGA96]), enz.

2.2.4 Tweelingsvergelijking

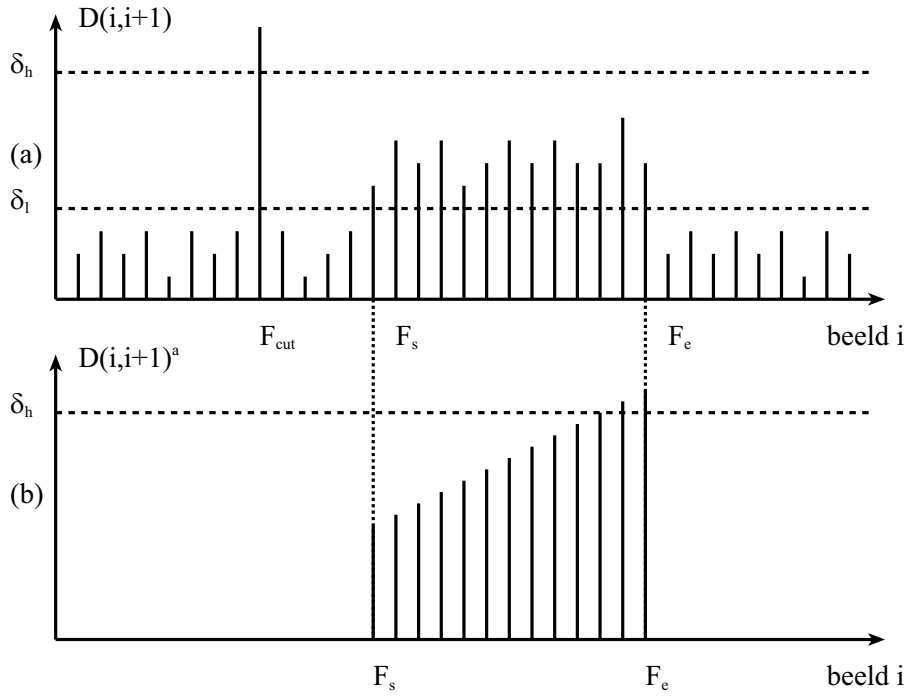
De tweelingsvergelijking is een uitbreiding van de vorige methoden om graduele overgangen te detecteren. Er is een hoge drempel δ_h om cuts te detecteren en een lage drempel δ_l om potentiële graduele overgangen te detecteren. Indien het verschil tussen opeenvolgende beelden groter is dan δ_h , dan is er een cut gedetecteerd, is deze lager dan δ_h en groter dan δ_l , dan markeren we het beeld als startbeeld F_s van de graduele transitie. F_s wordt dan vergeleken met de volgende beelden waarvan het verschil van de opeenvolgende beelden groter is dan δ_l . Dit is een accumulatieve vergelijking omdat tijdens een transitie de verschillen groter worden. Het eindbeeld F_e wordt vastgelegd indien het verschil tussen de opeenvolgende beelden lager ligt dan δ_l . Indien het accumulatief verschil tussen F_s en F_e groter is dan δ_h dan is er een graduele transitie gevonden. Figuur 2.8 toont de werking van de tweelingsvergelijking.

Soms zijn er verschillen van opeenvolgende beelden tijdens een transitie die onder δ_l duiken vooraleer het einde van de transitie bereikt is. Dit kan opgelost worden door een tolerantie in te voeren voor het aantal opeenvolgende verschillen, die onder δ_l liggen, vooraleer de kandidaat F_e vastgelegd wordt.

Bij deze methode is het enerzijds niet mogelijk om de verschillende transities te classificeren; anderzijds faalt zij eveneens bij grote en snelle bewegingen.

2.2.5 Randdetectie

Naast kleuren en grijswaarden kunnen ook andere kenmerken gebruikt worden om cuts en transities te bepalen. In [ZMM95, ZMM99] gebeurt de detectie op basis van contouren of randen. Tijdens een cut of dissolve verschijnen nieuwe randen ver van de oude randen en oude randen verdwijnen ver van nieuwe randen. Een pixel die behoort tot een rand die verschijnt (verdwijnt), wordt een binnenkomende (uitgaande) randpixel genoemd. Om



Figuur 2.8: De werking van de tweelingsvergelijking: (a) is het verschil tussen opeenvolgende beelden en (b) is het accumulatief verschil.

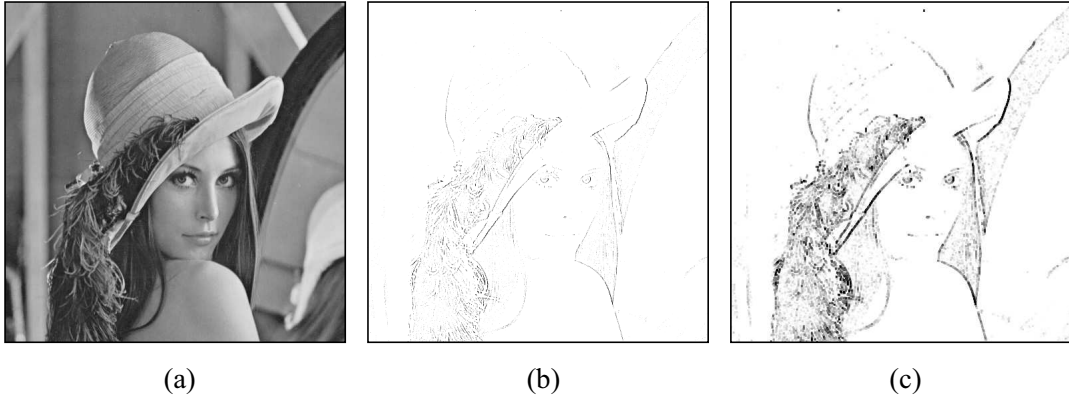
cuts, fades en andere dissolves te classificiëren volstaat het om deze randpixels te tellen. Voor wipes is er nog een extra analyse van de spatiale distributie nodig.

Voordat we de contouren van opeenvolgende beelden gaan vergelijken, moet eerst de camerabeweging gecompenseerd worden: de beelden worden ten opzichte van elkaar geregistreerd. In dit geval is de compensatie van translatie voldoende (uitgebreide technieken worden later besproken in paragraaf 3.3.1) [ZMM95, ZMM99]. De verschuiving tussen de twee beelden worden gekenmerkt door δ_x en δ_y .

Vervolgens worden de opeenvolgende beelden gefilterd met een gaussiaanse filter met grootte σ en worden de randen bepaald via de Canny-randdetector met een drempel τ voor de gradiëntmagnitude. De bekomen binaire beelden E en E' bevatten enkel de randpixels. Op deze beelden voeren we een morfologische erosie¹ uit om de randen te accentueren. De geërodeerde beelden zijn respectievelijk \tilde{E} en \tilde{E}' . Het proces wordt geïllustreerd in figuur 2.9.

We definiëren ρ_{in} als de fractie van de binnenkomende randpixels: randpixels in \tilde{E}' die op meer dan een vaste Hausdorff-afstand r liggen van de dichtste randpixel in \tilde{E} . Analoge definitie geldt voor de fractie van buitengaande randpixels, ρ_{uit} . De vergelijkingen worden gegeven in (2.15). ρ_{in} is hoog tijdens een fade-in, cut of aan het einde van een dissolve; ρ_{uit} is hoog tijdens een fade-out, cut of aan het begin van een dissolve.

¹Om de Manhattan-afstand tussen de randen te gebruiken, wordt het beeld geërodeerd door een diamant. Bij een euclidische afstand, is het structurelement een cirkel.



Figuur 2.9: Contourextractie van een beeld: (a) het origineel beeld, (b) het binair beeld E na Canny randdetectie en (c) het binair beeld \tilde{E} na morfologische erosie.

$$\rho_{in} = 1 - \frac{\sum_{x=1}^X \sum_{y=1}^Y \tilde{E}[x + \delta_x, y + \delta_y] E'[x, y]}{\sum_{x=1}^X \sum_{y=1}^Y E[x + \delta_x, y + \delta_y]} \quad (2.15)$$

$$\rho_{uit} = 1 - \frac{\sum_{x=1}^X \sum_{y=1}^Y E[x + \delta_x, y + \delta_y] \tilde{E}'[x, y]}{\sum_{x=1}^X \sum_{y=1}^Y E[x, y]}$$

De Hausdorff-afstand tussen puntenset A en B wordt gegeven door:

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

$$h(\tilde{E}', \tilde{E}) \leq r \rightarrow \rho_{in} = 0 \quad (2.16)$$

$$h(\tilde{E}, \tilde{E}') \leq r \rightarrow \rho_{uit} = 0$$

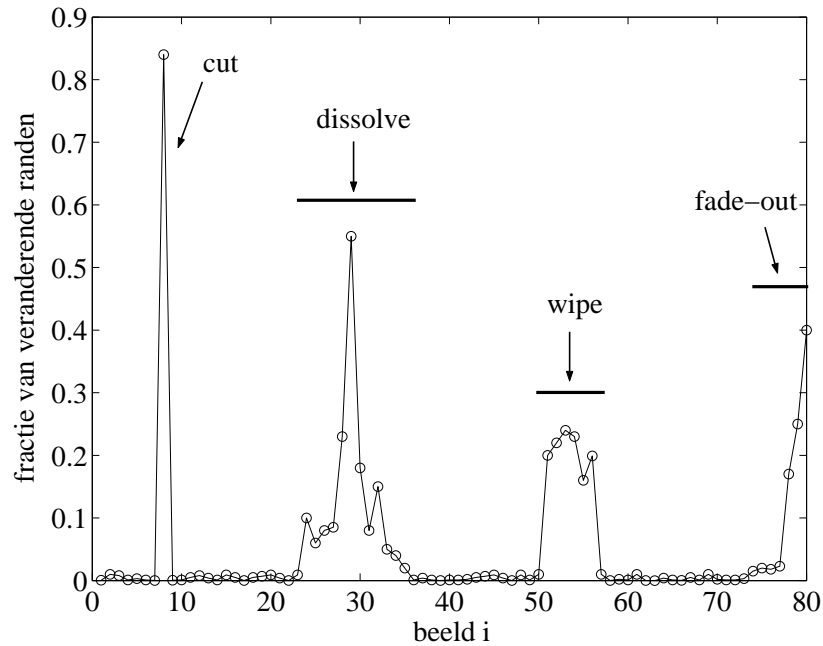
Nog algemener geldt de partiële Hausdorff-afstand: $h_K(A, B) = K_{a \in A} \min_{b \in B} \|a - b\|$, dit is de K -de geordende afstand van een punt in A tot het dichtsbijzijnde corresponderende punt in B .

Om cuts en transities te detecteren, definiëren we een basismaat voor de dissimilariteit tussen twee beelden, de fractie van de veranderende randen: $\rho = \max(\rho_{in}, \rho_{uit})$.

Om pieken te detecteren in het verloop van ρ , worden er twee criteria gehanteerd: ρ moet groter zijn dan een drempel δ én ρ moet een lokaal maximum zijn binnen een venster met een bepaalde breedte. De classificatie van de verschillende overgangen (zie ook figuren 2.10 en 2.11):

- cut: ρ bestaat uit een geïsoleerde piek evenals ρ_{in} en ρ_{uit} .

- dissolve: ρ is groot, ρ_{in} is groot bij het begin en ρ_{uit} is groot op het einde van de dissolve. Bij fade-in is ρ_{in} groter dan ρ_{uit} . Het omgekeerde geldt voor fade-out.
- wipe: hiervoor is er onderzoek naar de spatiale distributie nodig. Het aantal binnenkomende en uitgaande randpixels is geconcentreerd in een strook. Een eenvoudige manier om de spatiale distributie te onderzoeken is het verdelen van het beeld in twee helften (b.v. horizontaal of verticaal) en vervolgens ρ bepalen voor elk gebied. Voor eenvoudige wipes voldoet deze manier van werken.



Figuur 2.10: Verloop van ρ voor verschillende overgangen.

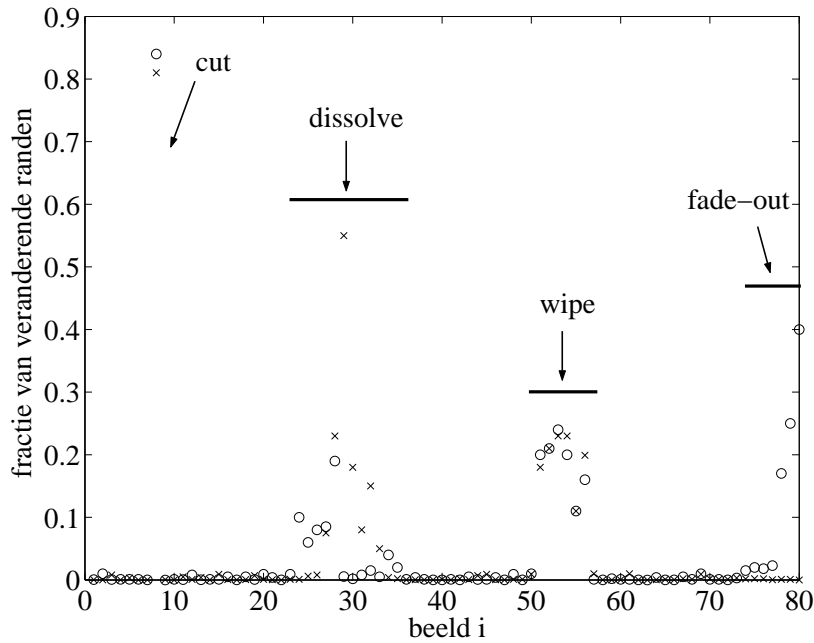
Deze techniek is redelijk robuust tegen verandering van belichting en camerabewegingen, maar is afhankelijk van mogelijke compressie-artifacten. Een potentieel probleemgeval zijn teksten die opeens verschijnen, die foute resultaten geven. De methode is ook complex in vergelijking tot de pixelvergelijking, blok- en histogramgebaseerde methoden.

Een andere methode, die gebaseerd is op hetzelfde principe is, werkt met beeldsegmenten in plaats van met randen [CSZK01]. Hierbij volgen we de gesegmenteerde objecten door de afstand te berekenen tussen de centroiden van de segmenten. Een cut wordt gevonden als het aantal overeenkomsten lager ligt dan een drempel δ .

2.2.6 Modelgebaseerde segmentatie

In deze sectie stellen we modellen op en proberen we de verschillende beelden van de videosequentie zo goed mogelijk in het vooropgestelde model te passen. Door allerlei ruis²

²We denken hierbij aan camera- en objectbeweging, maar ook belichting, compressie-artifacten, filters, enz.



Figuur 2.11: Verloop van ρ_{in} ("x") en ρ_{uit} ("o") voor verschillende overgangen.

zal het model niet 100% correct zijn, daarom moet er steeds een tolerantie ingevoerd worden.

Er zijn twee manieren om aan modelgebaseerde segmentatie te doen:

- het model wordt voorgesteld door een aantal criteria. Voldoen de beelden aan alle of sommige criteria, dan past de beeldsequentie in het model. Hiervoor kan b.v. een Bayes-classificatie op basis van Neyman-Pearson-criterium gebruikt worden.
- de totale set van beelden wordt opgedeeld in twee of meer clusters of klassen. Een beeld behoort tot een bepaalde cluster indien de eigenschappen ervan het dichtst aanleunen bij die van de cluster. Er bestaan veel verschillende methoden om te clusteren: b.v. het gebruik van neurale netwerken met een adaptief leeralgoritme zoals *Learning Vector Quantisation* (LVQ) of de modernere *Support Vector Machines* (SVM) [Lie01a].

In de volgende paragrafen worden enkele modellen voorgesteld.

Mathematische modellen

Een videosequentie wordt wiskundig voorgesteld door $E(x, y, t) = (r, g, b)$, waarbij elk beeld getransformeerd wordt vanuit een 2-dimensionale coördinatenruimte naar een vector in RGB-kleurenruimte. Zijn S_1 en S_2 twee shots die voorgesteld worden door $E(x, y, t)$, dan kan de overgang $E_{1,2}$ tussen deze twee shots mathematisch gedefinieerd worden als:

$$E_{1,2}(x, y, t) = T_{c1}(S_1(T_{s1}(x, y, t))) \otimes T_{c2}(S_2(T_{s2}(x, y, t))) \quad (2.17)$$

waarbij T_{si} spatiale transformaties zijn in de pixelruimte, T_{ci} transformaties zijn in de kleurenruimte en \otimes de operator is die de getransformeerde data van de shots combineert. Er zijn nu drie situaties:

- als T_{si} en T_{ci} identiteitstransformaties zijn, is er een cut gevonden. Identiteitstransformaties zijn transformaties die data op zichzelf laten afbeelden.
- als enkel T_{si} identiteitstransformaties zijn, dan is de overgang bepaald door een verandering in de intensiteitsruimte zoals bij een dissolve of fade.
- als enkel T_{ci} identiteitstransformaties zijn, dan wordt bij de overgang pixels van S_1 vervangen door pixels van S_2 , zoals bij een wipe.

Voor lineaire dissolves wordt de definitie relatief eenvoudig:

$$E_{1,2}(x, y, t) = f_1(t) \cdot S_1(x, y, t) + f_2(t) \cdot S_2(x, y, t), \quad t \in [0; T] \quad (2.18)$$

waarbij voor kruisende dissolves (zie figuur 2.2) geldt:

$$\begin{aligned} f_1(t) &= \frac{T-t}{T}, \quad t \in [0; T] \\ f_2(t) &= \frac{t}{T}, \quad t \in [0; T] \end{aligned} \quad (2.19)$$

en voor additieve dissolves geldt:

$$\begin{aligned} f_1(t) &= \begin{cases} 1, & \text{als } t < c_1 \\ \frac{T-t}{T-c_1}, & \text{anders} \end{cases}, \quad t \in [0; T], \quad c_1 \in]0; T[\\ f_2(t) &= \begin{cases} \frac{t}{c_2}, & \text{als } t < c_2 \\ 1, & \text{anders} \end{cases}, \quad t \in [0; T], \quad c_2 \in]0; T[\end{aligned} \quad (2.20)$$

Voor kruisende dissolves geldt eveneens dat de vorm van de variantiecurve parabolisch is ($\sigma^2 = E[X - \mu]^2$):

$$\sigma^2(t) = (\sigma_1^2 + \sigma_2^2)\alpha^2(t) - 2\sigma_1^2\alpha(t) + \sigma_1^2 \quad (2.21)$$

Het nut van deze mathematische modellen is niet zozeer de detectie van cuts en transitities, maar deze decompositie vormt een basis voor de typologie van de verschillende overgangen. De performantie van de deze mathematische analyse is laag aangezien het model geen rekening houdt met allerlei ruis [FSZ95].

Probabilistisch model

In [KC01] wordt een differentieel model voorgesteld dat gebaseerd is op de probabilistische distributie van het verschil tussen pixelwaarden. Deze methode combineert de volgende factoren:

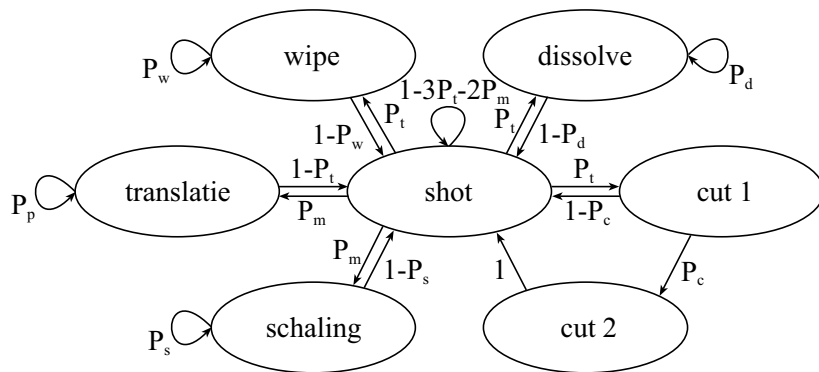
1. een additief gaussiaanse amplitude die ruis modelleert, veroorzaakt door quantizatie, conversie, compressie, enz.

2. een probabilistische distributie voor veranderende pixels, veroorzaakt door camera- en objectbeweging, verandering van kijkhoek en belichting, enz.
3. een transitie-model van shots voor cuts en graduele overgangen.

De probabilliteit van elk beeld voor elke cluster wordt berekend en het beeld wordt geassigneerd aan de cluster waarvoor het beeld de hoogste probabilliteit heeft.

Toestandsmodel

Voor het toestandsmodel gebruiken we een verborgen Markovmodel [AAW01, HLW00] (het zogenaamde *HMM* of *Hidden Markov model*). In figuur 2.12 worden de verschillende toestanden en overgangsprobabiliteiten getoond. Hierbij stellen P_t en P_m de kansen voor dat een beeld respectievelijk behoort tot een transitie of een camerabeweging. De overgangsprobabiliteiten worden getraind door het Baum-Welch algoritme en de overgangen worden aan de hand van verschillende kenmerken van het beeld bepaald door het Viterbi algoritme.



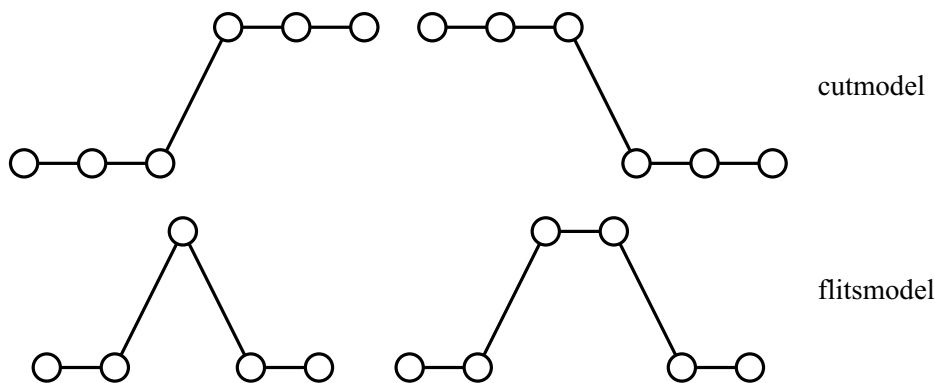
Figuur 2.12: Het Markov toestandsmodel voor videosequenties.

Bij cut zijn er twee toestanden nodig om het onderscheid te maken tussen cuts en flitsen³ [DZ01]. In figuur 2.13 worden de gemiddelde intensiteitsmodellen voorgesteld van een cut en een flits. De flitsen bestrijken meestal slechts één beeld, soms twee beelden. De situatie waarbij de gemiddelde intensiteit dezelfde is voor twee verschillende shots, komt zelden voor.

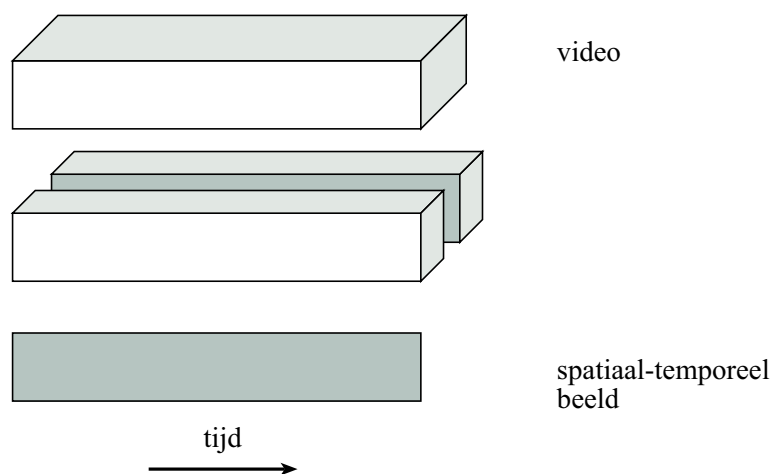
2.2.7 Spatiaal-temporele snede-analyse

Tot nu toe gebeurde de temporele segmentatie steeds door twee beelden te vergelijken en de similariteit ertussen te bepalen. Een videosequentie kan beschouwd worden als een 3-dimensionaal datamodel en een beeld is dan een 2-dimensionale projectie ervan (zie figuur 1.1). We kunnen ook een ander 2-dimensionale projectie nemen van de videosequentie en hierop temporele segmentatie uitvoeren [NPC99a, NPC99b]. De nieuwe projectie levert ons spatiaal-temporele beelden op, zoals aangetoond in figuur 2.14.

³Onder flitsen verstaan we een kortstondig, maar zeer intensieve verandering van de belichting van de omgeving. Voorbeelden zijn bliksemschichten of de flash van een fotooestel.



Figuur 2.13: Het flits- en cutmodel.



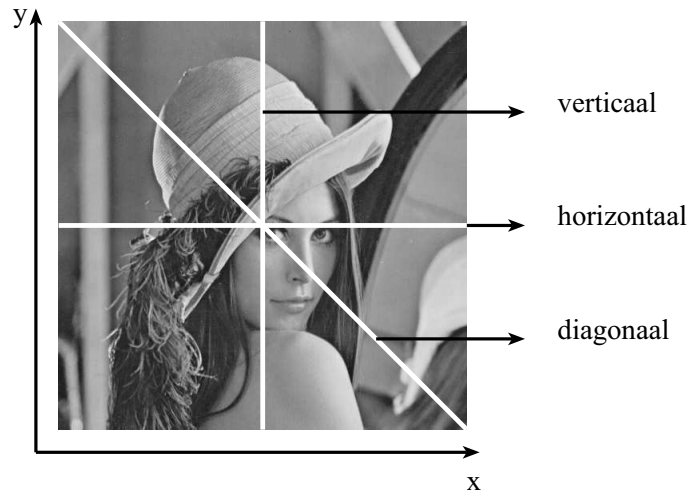
Figuur 2.14: Spatiaal-temporeel beeld.

Een snede is een 1-dimensionale rij uit een beeld en een spatiaal-temporeel beeld is dan een collectie van sneden in de videosequentie op dezelfde positie. We definiëren 3 sneden v , h en d (respectievelijk verticaal, horizontaal en diagonaal) van een beeld E op de volgende manier:

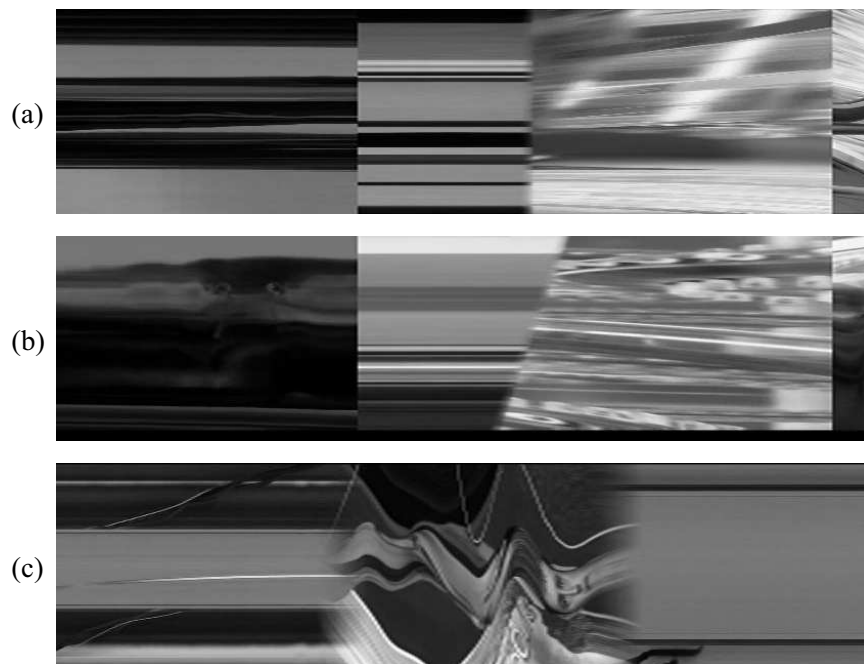
$$\begin{aligned}
 v(i) &= \sum_{p=k_1-j}^{k_1+j} \alpha_p \cdot E(p, i), \quad k_1 = \frac{M}{2} \\
 h(i) &= \sum_{p=k_2-j}^{k_2+j} \alpha_p \cdot E(p, i), \quad k_2 = \frac{N}{2} \\
 d(i) &= \sum_{p=i-j}^{i+j} \alpha_p \cdot E(p, i)
 \end{aligned} \tag{2.22}$$

met $0 \leq p < X$ of Y en $\sum_p \alpha_p = 1$.

Om artifacten en andere ruis te onderdrukken, middelen we de sneden gaussiaans uit, b.v. met $\alpha = [0.2236, 0.5528, 0.2236]$. Voor $j = 0$ wordt enkel de middelste rij of kolom van het beeld gebruikt om de sneden te vormen, zoals geïllustreerd in figuur 2.15. De bekomen spatiaal-temporele beelden worden V , H en D genoemd. Een paar voorbeelden worden getoond in figuur 2.16.















Figuur 2.15: Drie 1-dimensionale sneden v , h en d genomen van het beeld.



Figuur 2.16: Voorbeelden van spatiaal-temporele beelden: (a) een sequentie met 3 cuts, (b) een sequentie met 2 cuts en 1 wipe en (c) een sequentie met 2 dissolves.

Tabel 2.1: Beeldpatronen van overgangen in spatiaal-temporele beelden.

	H	V	D
cut			
verticale wipe			
horizontale wipe			
dissolve			

Om aan cut- en transitiedetectie te doen, moeten we discontinuïteiten, van de overgangen, in textuur en kleur terugvinden in de spatiaal-temporele beelden. Enkele patronen worden voorgesteld in tabel 2.1. Er bestaan tal van manieren om aan textuuronderzoek te doen, maar hier gebruiken we een Markov energiemodel om de exacte lokatie van cuts en wipes te vinden. Voor dissolves onderzoeken we de intensiteitsverandering van de spatiaal-temporele beelden.

Cut- en wipedetectie

Definiëren we $H = [H_r, H_g, H_b, H_y]$, $V = [V_r, V_g, V_b, V_y]$ en $D = [D_r, D_g, D_b, D_y]$ als de spatiaal-temporele beelden in RGB-kleurenruimte en Y^4 -luminantieruimte, dan kunnen we de randinformatie berekenen als:

$$E_{\sigma,\theta}^{H_i} = \bar{G}D_{\sigma,\theta} * H_i \quad (2.23)$$

met $*$ de convolutie-operator en $i \in [r, g, b]$. $\bar{G}D_{\sigma,\theta}$ is de eerste gaussische afgeleide langs de x-as⁵, gegeven door:

$$\begin{aligned} \bar{G}D_{\sigma,\theta}(x, y) &= -\frac{x}{\sigma^2} \bar{G}_{\sigma,\theta}(x, y) \\ \bar{G}_{\sigma,\theta}(x, y) &= \bar{G}_{\sigma}(x', y') \end{aligned} \quad (2.24)$$

waarbij $x' = x \cos \theta + y \sin \theta$ en $y' = -x \sin \theta + y \cos \theta$ is. $\bar{G}_{\sigma}(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ is een gaussische filter met parameter σ .

De berekende textuurkenmerken zijn gebaseerd op de Gabor decompositie. De idee hierachter is om beelden te decomponeren in verschillende spatiale frequentiekanalen. De reële componenten van de omhullende van de kanaalsignalen worden dan gebruikt om een kenmerkenvector te vormen. De complexe gaborbeelden zijn:

$$T_{\sigma_x, \sigma_y, \theta} = \hat{G}_{\sigma_x, \sigma_y, \theta} * H_y \quad (2.25)$$

De gaborfilter $\hat{G}_{\sigma_x, \sigma_y, \theta}(x, y) = \hat{G}_{\sigma_x, \sigma_y}(x', y')$ wordt uitgedrukt als:

$$\hat{G}_{\sigma_x, \sigma_y}(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y} \right) e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)} e^{2\pi j W x} \quad (2.26)$$

⁴De definitie van luminantie is $y = 0.257 \cdot r + 0.504 \cdot g + 0.098 \cdot b + 16$

⁵De x-as bij spatiaal-temporele beelden stelt de dimensie tijd voor.

met $j = \sqrt{-1}$, $W = \sqrt{u^2 + v^2}$ en met (u, v) het centrum van de gekozen frequentie. De waarden σ , σ_x en σ_y vormen de 12-dimensionale kenmerkenvector.

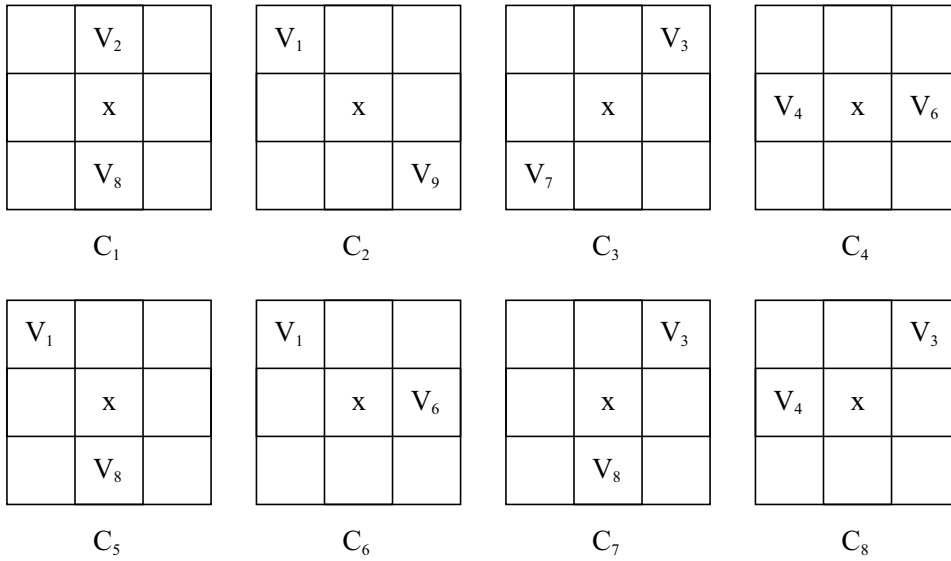
We gebruiken het Markov energiemodel voor de segmentatie: de kans dat een drievoudige pixel $\eta = (\eta_v, \eta_h, \eta_d)$ op een grens ξ van twee geconnecteerde regio's (shots) ligt van $H(k, t)$, $V(k, t)$ en $D(k, t)$, is:

$$p(\eta \in \xi | H, V, D) = p(\eta \in \xi | H_N, V_N, D_N) \quad (2.27)$$

met H_N , V_N en D_N een 3×3 systeemomgeving, getoond in figuur 2.17. Gebaseerd op deze systeemomgeving, definiëren we de acht geconnecteerde componenten $C = \{C_1, C_2, \dots, C_8\}$ (zie figuur 2.18) om η te karakteriseren.

V_1	V_2	V_3
V_4	x	V_6
V_7	V_8	V_9

Figuur 2.17: Systeemomgeving van een pixel in V_N (de figuur is ook van toepassing voor H_N en D_N).



Figuur 2.18: De acht geconnecteerde componenten gebaseerd op de systeemomgeving.

We stellen dat H_N , V_N en D_N onafhankelijk zijn van elkaar, zo kan (2.27) herschreven worden als:

$$p(\eta | H, V, D) = p(\eta_h | H_N) p(\eta_v | V_N) p(\eta_d | D_N) \quad (2.28)$$

met $p(\eta_h)$, $p(\eta_v)$ en $p(\eta_d)$ de probabiliteiten van respectievelijk η_h , η_v en η_d op de shotgrens van H , V en D . Door de Markov-Gibbs equivalent, hebben we:

$$p(\eta_i) = \frac{1}{Z} e^{-U(\eta_i)} \quad (2.29)$$

waarbij Z een constante is om te normaliseren, $i \in \{h, v\}$ en $U(\eta_i)$ de energiefunctie gegeven door:

$$U(\eta_i) = \sum_{c \in C} \beta_c \Gamma_c(\eta_i) \quad (2.30)$$

De energie is de gewogen som van de potentiële energie $\Gamma(\eta_i)$ over alle geconnecteerde componenten, waarbij $\sum_{c \in C} \beta_c = 1$.

Voor classificatie en segmentatie van wipes, definiëren we drie types van energie: $U_{cut}(\eta_i)$, $U_{wipe-}(\eta_i)$ en $U_{wipe+}(\eta_i)$. De energie voor de rode component van het horizontaal spatiaal-temporeel beeld wordt geformuleerd als:

$$\begin{aligned} \begin{bmatrix} U_{cut}^r(\eta_h^r) \\ U_{wipe-}^r(\eta_h^r) \\ U_{wipe+}^r(\eta_h^r) \end{bmatrix} &= 3 \begin{bmatrix} \Gamma_{C_1}^r(\eta_h^r) \\ \Gamma_{C'}^r(\eta_h^r) \\ \Gamma_{C''}^r(\eta_h^r) \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Gamma_{C_1}^r(\eta_h^r) \\ \Gamma_{C_2}^r(\eta_h^r) \\ \Gamma_{C_3}^r(\eta_h^r) \end{bmatrix} - \begin{bmatrix} \Gamma_{C_4}^r(\eta_h^r) \\ \Gamma_{C_4}^r(\eta_h^r) \\ \Gamma_{C_4}^r(\eta_h^r) \end{bmatrix} \\ \Gamma_{C'}^r(\eta_h^r) &= \min_{c \in \{C_2, C_5, C_6\}} \Gamma_c^r(\eta_h^r) \\ \Gamma_{C''}^r(\eta_h^r) &= \min_{c \in \{C_3, C_7, C_8\}} \Gamma_c^r(\eta_h^r) \end{aligned} \quad (2.31)$$

Waarbij η_h^r de pixel voorstelt op (k, t) van het H_r beeld. Voor de andere kleurkanalen g, b, y en snederichtingen V en D gelden analoge formules. De energie U_{cut}^r is laag indien η_h^r gelokaliseerd is op de cut. Gelijkaardig zullen U_{wipe-}^r en U_{wipe+}^r laag zijn als η_h^r op een wipe ligt. De waarden van U_{wipe-}^r en U_{wipe+}^r zijn afhankelijk van de gradiënt van de wipe, die positief of negatief kan zijn. Stel dat $\eta_1 = (k_{\eta_1}, t_{\eta_1})$ en $\eta_2 = (k_{\eta_2}, t_{\eta_2})$ de burens zijn van η_h^r zodat $\{\eta_1, \eta_h^r, \eta_2\}$ de geconnecteerde component C_i vormt, dan wordt de potentiële energie gegeven door:

$$\Gamma_{C_i}^r(\eta_h^r) = \sum_{\theta} \left\{ |E_{\sigma, \theta}^{H_r}(k, t) - E_{\sigma, \theta}^{H_r}(k_{\eta_1}, t_{\eta_1})| + |E_{\sigma, \theta}^{H_r}(k, t) - E_{\sigma, \theta}^{H_r}(k_{\eta_2}, t_{\eta_2})| \right\} \quad (2.32)$$

$\Gamma_{C_i}^g$ en $\Gamma_{C_i}^b$ worden op dezelfde manier berekend en $\Gamma_{C_i}^y$ wordt berekend als:

$$\Gamma_{C_i}^y(\eta_h^y) = \sum_{\theta} \left\{ |T_{\sigma_x, \sigma_y, \theta}(k, t) - T_{\sigma_x, \sigma_y, \theta}(k_{\eta_1}, t_{\eta_1})| + |T_{\sigma_x, \sigma_y, \theta}(k, t) - T_{\sigma_x, \sigma_y, \theta}(k_{\eta_2}, t_{\eta_2})| \right\} \quad (2.33)$$

Combineren we de informatie van de verschillende kleurkanalen en de textuur, dan krijgen we:

$$U_{cut}(\eta_h) = \alpha_c \max_{j \in \{r, g, b\}} U_{cut}^j(\eta_h^j) + \alpha_t U_{cut}^y(\eta_h^y) \quad (2.34)$$

waarbij α_c en α_t twee gewichten zijn voor de kleur- en textuurkenmerken. Analoog geldt voor U_{wipe-} en U_{wipe+} . Figuur 2.19 toont een bekomen segmentatieresultaat, de witte lijnen duiden de gebieden aan van lage energie.



Figuur 2.19: Energieveld van figuur 2.16 (b).

Dissolve detectie

Bij dissolves zijn er geen duidelijke grenzen in de spatiaal-temporele beelden: de geconnecteerde shots hebben wel een gemeenschappelijk overgangsgebied. Gebaseerd op de formules (2.18) en (2.19) voor kruisdissolve wordt de gemiddelde intensiteit van de snede $\mu_i(t)$ in een spatiaal-temporeel beeld i tijdens het interval $0 \leq t \leq T$ gegeven door:

$$\begin{aligned}\mu_i(t) &= f_1(t)\mu_i^{S_1}(t) + f_2(t)\mu_i^{S_2}(t) \\ &= \frac{T-t}{T}\mu_i^{S_1}(t) + \frac{t}{T}\mu_i^{S_2}(t) \\ &= \mu_i^{S_1}(t) + \frac{t}{T}(\mu_i^{S_2}(t) - \mu_i^{S_1}(t))\end{aligned}\tag{2.35}$$

waarbij $\mu_i^{S_j}(t)$ de gemiddelde intensiteit is van de snede van shot j en $i \in \{H, V, D\}$. $\mu_i(t)$ is een rechte met een strikt positieve of negatieve gradiënt zoals in figuur 2.20.

De afgeleide $\dot{\mu}_i(t) = \frac{d\mu_i(t)}{dt}$ wordt gegeven door:

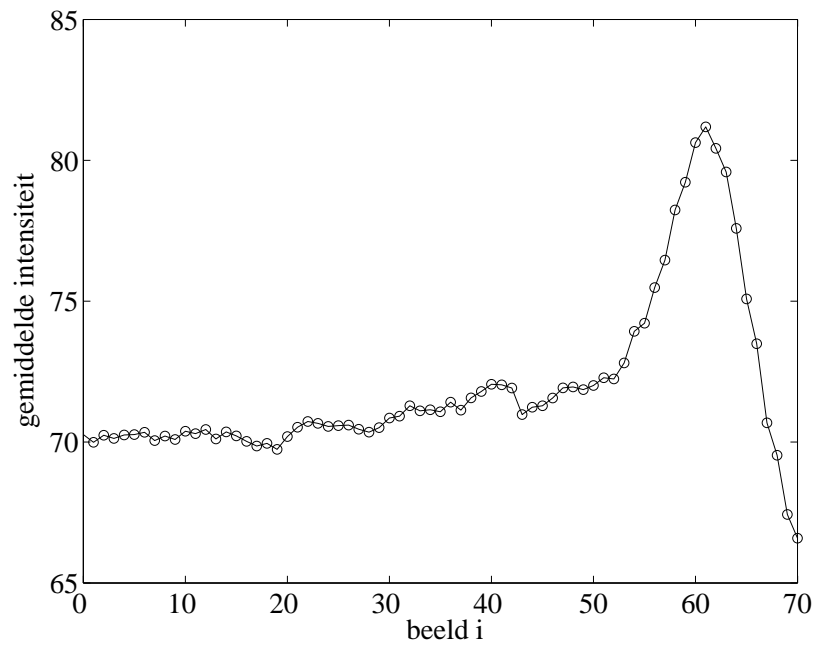
$$\dot{\mu}_i(t) = \frac{\mu_i^{S_2}(t) - \mu_i^{S_1}(t)}{T}\tag{2.36}$$

Als $\mu_i^{S_1}(t)$ en $\mu_i^{S_2}(t)$ onveranderd blijven tijdens de dissolve, dan is $\dot{\mu}_i(t)$ constant. Gelijkaardig geldt voor de variantie $\sigma_i^2(t)$ van een snede tijdens een dissolve:

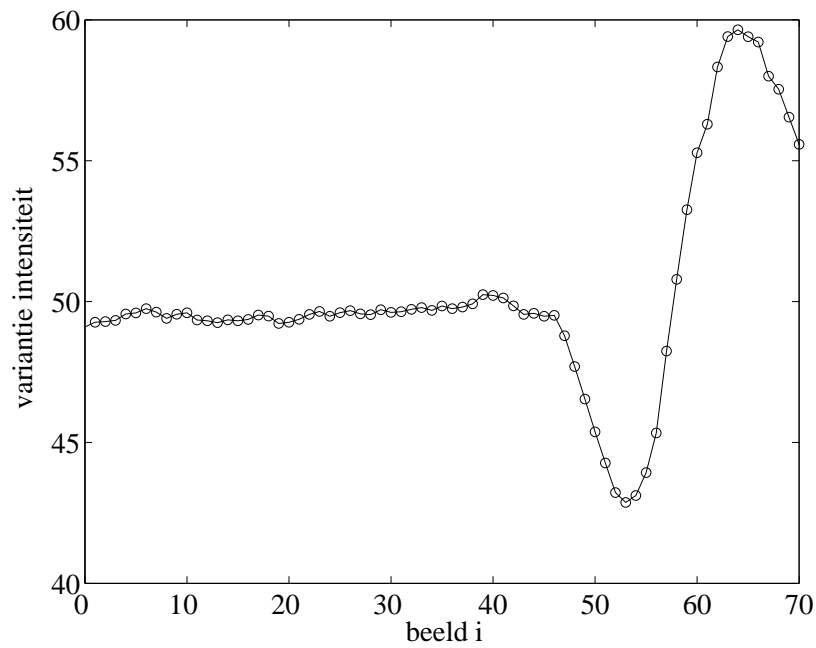
$$\sigma_i^2(t) = (\sigma_{i,S_1}^2(t) + \sigma_{i,S_2}^2(t))\frac{t^2}{T^2} - 2\sigma_{i,S_1}^2(t)\frac{t}{T} + \sigma_{i,S_1}^2(t)\tag{2.37}$$

met $\sigma_{i,S_j}^2(t)$ de variantie van de snede van shot j en $i \in \{H, V, D\}$. Als $\sigma_{i,S_1}^2(t)$ en $\sigma_{i,S_2}^2(t)$ constant blijven tijdens de dissolve, dan is $\sigma_i^2(t)$ een concaaf parabool tijdens $0 \leq t \leq T$ zoals aangetoond in figuur 2.21.

Om dissolves te detecteren, worden (2.36) en (2.37) van een spatiaal-temporeel beeld berekend en een dissolve wordt gevonden indien de gemiddelde intensiteiten constant zijn en als het verloop van de variantie een concave parabolische curve vertoont voor $\delta_1 \leq T \leq \delta_2$, met δ_1 en δ_2 respectievelijk de ondergrens en de bovengrens voor de duur van een dissolve. De veronderstellingen in (2.36) en (2.37) zullen verkeerd zijn indien er beweging aanwezig is. Daarom wordt er een lakser schema voorgesteld:



Figuur 2.20: Analyse van $\mu_i(t)$ met een dissolve tussen beeld 47 en 64.



Figuur 2.21: Analyse van $\sigma_i^2(t)$ met een dissolve tussen beeld 47 en 64.

$$\sum_{i \in \{H, V, D\}} G(\sigma_i^2(t), \mu_i(t)) \geq 2 \quad (2.38)$$

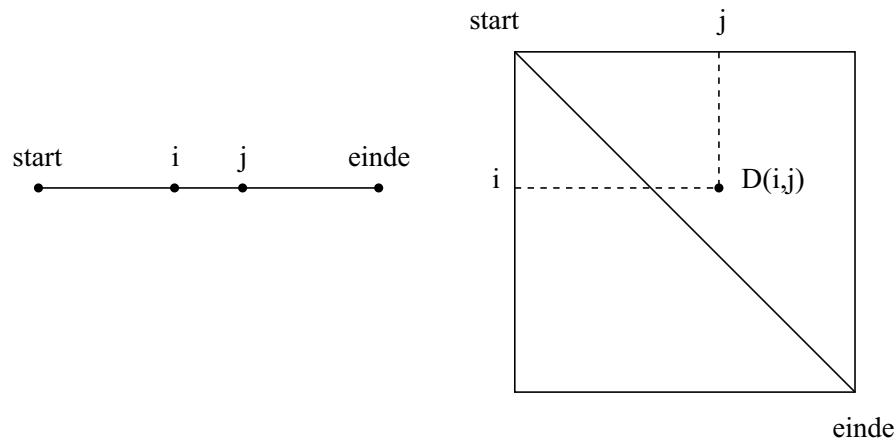
met $G : \mathfrak{R} \times \mathfrak{R} \rightarrow \{0, 1\}$ een logische operator.

Het gebruik van deze methode is vanuit een syntactisch standpunt meer verantwoord dan andere methoden, maar de methode is zeer gevoelig voor bewegingen en ruis. De vals gedetecteerde overgangen worden hoofdzakelijk veroorzaakt door bewegings- en belichtingsverandering. Om het aantal valse overgangen te reduceren, kunnen extra controlestappen (bewegingsdetectie of berekening van de similariteit bij de kandidaatcuts) toegevoegd worden of er kunnen meer dan drie spatiaal-temporele beelden geselecteerd worden, maar dit alles vereist extra rekentijd. Het grootste nadeel van deze techniek is de grote complexiteit, spatiale reductie is nodig om de rekentijd te beperken. Een ander nadeel van deze techniek is dat het niet geschikt is voor realtime⁶-toepassingen, omdat de methode informatie nodig heeft die nog niet beschikbaar zal zijn.

2.2.8 Zelf-similariteitsanalyse

Bij zelf-similariteitsanalyse bouwen we een (dis)similariteitsmatrix op zoals aangegeven in figuur 2.22. We berekenen de zelf-similariteit tussen twee beelden i en j met behulp van de cosinus afstandsmaat van b.v. de respectievelijke histogrammen:

$$D(i, j) = \frac{\sum_{k=0}^n H_i(k) \cdot H_j(k)}{\sqrt{\left(\sum_{k=0}^n H_i^2(k)\right)} \sqrt{\left(\sum_{k=0}^n H_j^2(k)\right)}} \quad (2.39)$$



Figuur 2.22: Opbouw van een similariteitsmatrix.

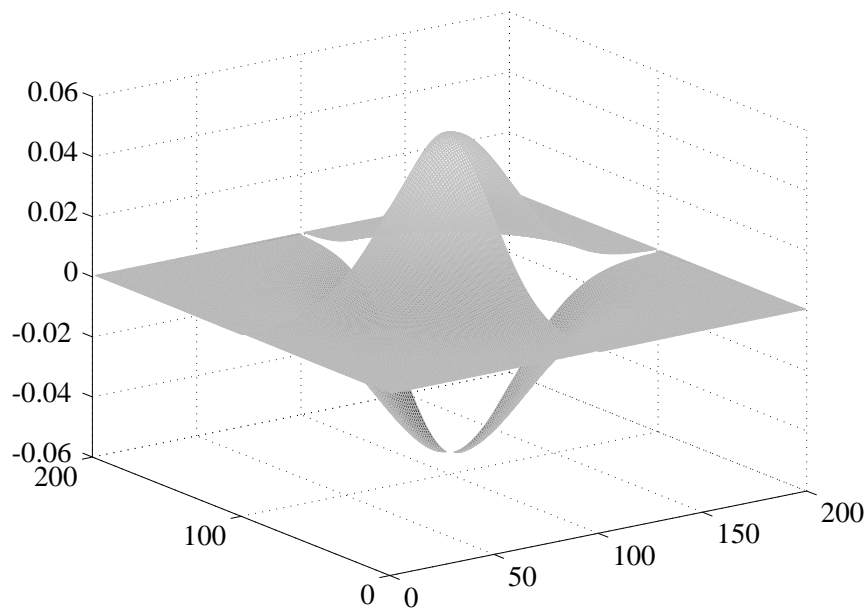
De matrix wordt opgebouwd door de rijen i en kolommen j in te vullen met de afstanden. De matrix is symmetrisch als de gebruikte afstandsmaat symmetrisch is. We berekenen de zelf-similariteit van de beelden die voor (na) het huidig beeld, alsook de kruis-similariteit (tussen de beelden, die voor het huidig beeld komen, en de beelden die

⁶Hiermee bedoelen we de verwerking van videostreamen zonder vertraging die b.v. afkomstig zijn van televisie-uitzendingen of van het internet.

na het huidige beeld komen). Een cut heeft dan een hoge zelf-similariteit en een lage kruis-similariteit. Deze cuts kunnen we detecteren in de matrix door de correlatie langs de hoofddiagonaal te berekenen met behulp van een gaussiaans geruite kern. Een simpel voorbeeld is de 2×2 geruite kern:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (2.40)$$

In figuur 2.23 wordt een voorbeeld van een gaussiaans geruite kern gegeven. De correlatie zal pieken vertonen op plaatsen waar cuts zich bevinden.



Figuur 2.23: Een voorbeeld van een gaussiaans geruite kern.

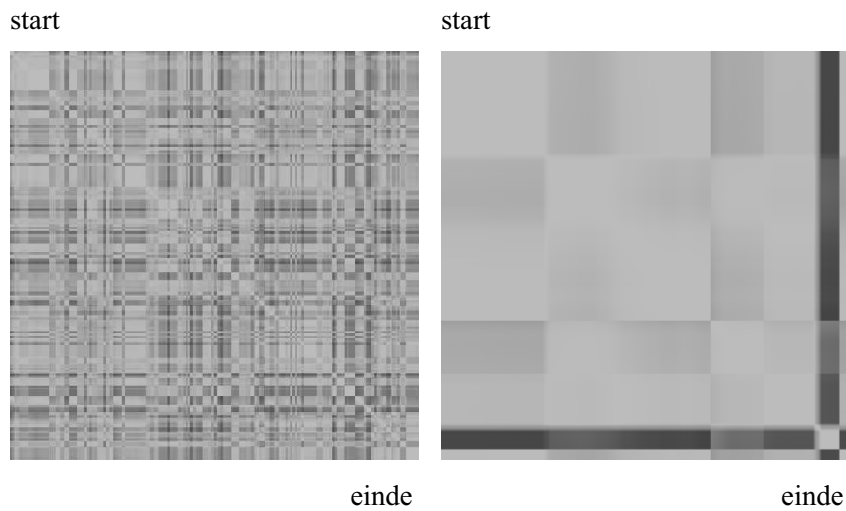
Shots zijn regio's langs de diagonaal met hoge similariteit. Dialogen vertonen een geruit patroon en transities zullen stroken met middelmatige similariteit tussen de shots zijn. Het resultaat van de similariteitsmatrix en het verloop van de correlatie wordt weergegeven in figuren 2.24 en 2.25.

2.2.9 Andere kenmerken

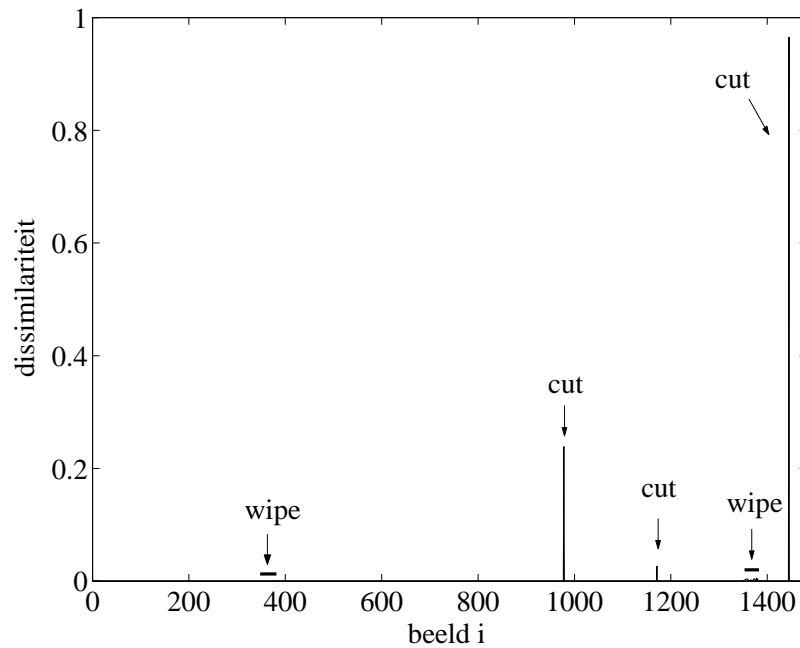
De besproken algoritmen zijn steeds gebaseerd op beeldkenmerken, zoals kleuren, randen, enz. Ook andere kenmerken kunnen aangewend worden om aan temporele videosegmentatie te doen.

Audio

Een videosysteem bestaat niet enkel uit beelden, maar bevat ook een geluidsstroom. Audio kan additionele informatie opleveren met betrekking tot segmentatie waardoor de resultaten verbeterd kunnen worden.



Figuur 2.24: Twee voorbeelden van similariteitsmatrices.



Figuur 2.25: Het verloop van de kerncorrelatie langs de diagonaal van de similariteitsmatrix uit figuur 2.24 (b).

Audio kan opgedeeld worden in 3 categorieën: *stilte*, *spraak* en *muziek*. De kenmerken van audio worden geëxtraheerd op basis van amplitude, frequentie-inhoud, duur, enz. en op basis van deze eigenschappen wordt de geluidsfragmenten ingedeeld in de bovenstaande categorieën. De probabiteit van een overgang is groter als er gelijktijdig een verandering in de geluidsfragmenten is. Stemherkenning kan b.v. worden gebruikt om in een dialoog de verschillende sprekers te onderscheiden.

Semantische segmentatie

Bij semantische segmentatie wordt er rekening gehouden met specifieke kenmerken in een bepaald domein, enkele voorbeelden hiervan zijn:

- segmentatie van nieuwsuitzendingen kan verbeterd worden met behulp van sjablonen: beelden kunnen vergeleken worden met een vooraf gedefinieerde sjabloon. Indien opeenvolgende beelden overeenkomen met eenzelfde sjabloon, behoren ze tot dezelfde shot. Sjablonen kunnen ook scèneclustering enorm verbeteren: gelijkaardige shots zullen in dezelfde sjabloon passen. Voorbeelden waarin sjablonen kunnen gebruikt worden: shots met een nieuwslezer, shots met weerkaarten, enz. In figuur 2.26 worden enkele voorbeelden geïllustreerd. Gebieden van het beeld worden afgebakend op basis van specifieke kenmerken: b.v. bij een shot met een nieuwslezer zal de achtergrond statisch blijven en alle beweging zal beperkt worden tot de ankerpersoon zelf of bij een weerkaart hebben blokken op bepaalde plaatsen van het beeld een bepaalde kleur. Al deze criteria worden in de sjabloon verwerkt.
- bij videobeelden van amateuropnamen kunnen we gebruik maken van de tijdscode die op een vaste plaats in het beeld verschijnt. We kunnen de tijdscode extraheren door tekstherkenning uit te voeren: de structuren MM/DD/YYYY en hh:mm:ss worden ingevuld en een cut wordt gedetecteerd indien de tijdscode niet continu is. Bij digitale videocamera's kunnen de gegevens van de tijdscode zelfs rechtstreeks gehaald worden uit de camera zelf.

Het grote nadeel is natuurlijk dat deze methoden niet algemeen toepasbaar zijn.

2.2.10 Reductie van de rekentijd

Om realtime-toepassingen te maken is het nodig om de rekentijd te beperken. We kunnen repetitieve stukken van een algoritme in hardware implementeren of stukken code optimaliseren voor bepaalde processoren. Een andere manier om de rekentijd te reduceren is het beperken van de te verwerken data. Dit kan zowel spatiaal als temporeel gebeuren.

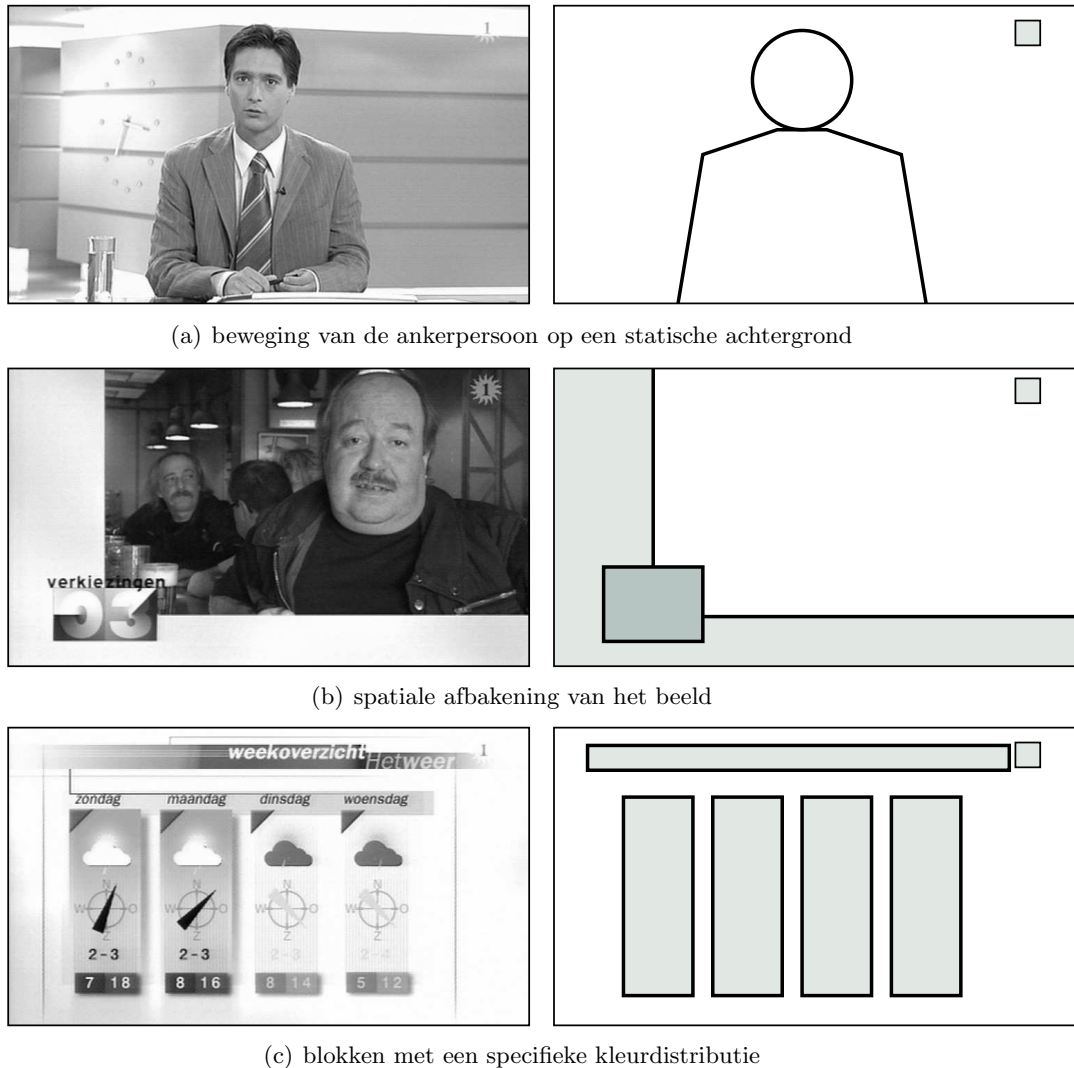
Spatiale reductie

De dimensie (breedte en hoogte) van elke beeld wordt hierbij verkleind. Het aantal te verwerken pixels wordt zo sterk gereduceerd, maar hierdoor gaat ook (kostbare) informatie verloren en zullen er meer valse overgangen gedetecteerd worden.

Temporele reductie

Hierbij worden niet alle beelden van de videosequentie onderzocht, maar wordt er een selectie gemaakt waarbij telkens een reeks beelden worden overgeslagen.

De stap-variabele methode detecteert cuts en transitie door beelden i en j te vergelijken met $j = i + \text{stap}$. Als er geen significant verschil wordt gevonden, worden de beelden die een halve stap verder liggen vergeleken: de beelden $i + \text{stap}/2$ en $j + \text{stap}/2$ worden vergeleken. Anders wordt een binaire zoekmethode gebruikt om de cut te lokaliseren: als i en j opeenvolgend zijn en het verschil van de respectievelijke beelden is groter dan een drempel δ_1 , dan is er een cut gedetecteerd, anders wordt het verschil van de tussenliggende



Figuur 2.26: Verschillende sjablonen voor de semantische segmentatie. Bij elk beeld kan ook het logo van de televisiezender in de sjabloon verwerkt worden.

beelden vergeleken in vergelijking tot een andere waarde δ_2 om graduele overgangen te detecteren.

De grootte van de stap bepaalt de performantie van het algoritme: grote stappen zijn efficiënt maar detecteren vaker valse overgangen, te kleine stappen kunnen graduele overgangen missen.

Meerfase techniek

Deze techniek detecteert cuts en transitie in meerdere fasen. In de eerste fase worden de potentiële cuts en transitie gemarkeerd met behulp van spatiale en temporele reductie van de videosequentie. In de volgende fase worden enkel de mogelijke cuts en transitie in detail onderzocht.

Het grote nadeel is dat deze methode niet geschikt is voor realtime-toepassingen, maar de methode kan wel gebruikt worden bij statische videobestanden.

2.3 Temporele segmentatie op gecomprimeerde video

Tegenwoordig wordt video overall gecomprimeerd, zowel in archivering als in videostreamen. Voorbeelden zijn films op DVD of (S)VCD, videostreamen op internet (Quicktime, Windows Media, RealMedia, ...), digitale televisie, multimedia databanken, enz.

In de vorige sectie hadden we temporele segmentatie uitgevoerd op ongecomprimeerde video. Dit vergt een decoder en het vereist soms veel rekentijd om videostreamen naar een RGB-formaat om te zetten. Het probleem kan worden vermeden door de decoder in hardware te implementeren, maar deze oplossing is niet altijd voorhanden. Daarom werken we rechtstreeks op de gecomprimeerde stromen of bestanden. De voordelen hiervan zijn dat een set van kenmerken (b.v. DCT-coëfficiënten, bewegingsvectoren, ...) reeds vooraf berekend zijn en gewoon gecodeerd zitten in het bestand of in de stroom, er komt geen decoder aan te pas wat de rekencomplexiteit reduceert en gecomprimeerde video heeft een veel lager datadebiet wat de rekentijd verkleint.

In deze sectie geven we een overzicht van de segmentatie op basis van de kenmerken van MPEG⁷-systemen. Merk op dat de meeste onderdelen ook bruikbaar zijn voor andere compressieschema's en dat er nog andere technieken bestaan, b.v. op basis van subbandcodering. We geven eerst een korte inleiding op het MPEG-compressieschema en vervolgens bespreken we de temporele segmentatie op de verschillende onderdelen. Om robuuste segmentatie-algoritmen te ontwikkelen worden vaak verschillende technieken gecombineerd.

2.3.1 Beknopte overzicht van MPEG

De MPEG-compressie is een veelgebruikte internationale standaard voor videocompressie. De standaard gebruikt twee basistechnieken: macroblok-gebaseerde bewegingscompensatie om de temporele redundantie te verkleinen en blokgebaseerde compressie door transformatie om de spatiale redundantie te verkleinen. Een MPEG-stroom bevat drie types van beelden I (*Intra*), P (*Predicted*) en B (*Bi-directional*) die gecombineerd worden in een repetitief patroon, de zogenaamde *group of picture (GOP)*:

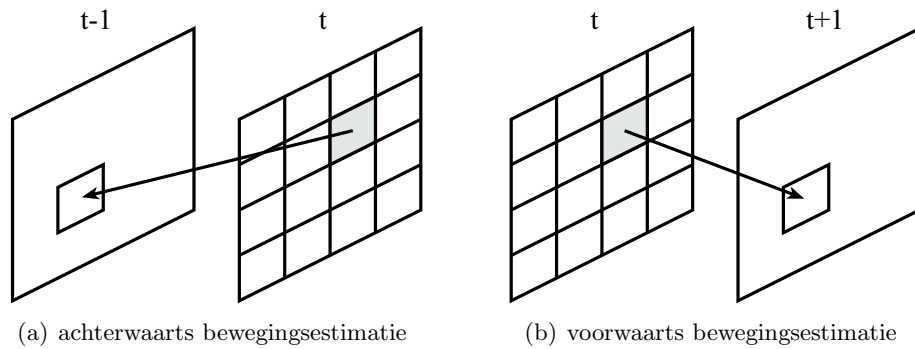
$$\underbrace{IBBPBBPBBPBBIBB\dots}_{GOP} \quad (2.41)$$

We kunnen deze structuur algemeen aannemen zonder verlies aan informatie.

De predictieve relaties tussen de verschillende types van beelden:

- voorwaarts: verwijzing vanuit B -beeld naar het dichtsbijzijnde voorgaande I - of P -beeld en een verwijzing vanuit een P -beeld naar het dichtsbijzijnde voorgaande I - of P -beeld. De schatting van de bewegingsinformatie gebeurt van tijdstip t naar $t + 1$ zoals aangetoond in figuur 2.27.
- achterwaarts: verwijzing vanuit B -beeld naar het dichtsbijzijnde nakomende I - of P -beeld. De estimatie gebeurt van tijdstip t naar $t - 1$ zoals geïllustreerd in figuur 2.27.
- geïnterpoleerd: verwijzing vanuit B -beeld in beide richtingen naar het dichtsbijzijnde voorgaande en nakomende I - of P -beeld.

⁷*Moving Picture Experts Group*



Figuur 2.27: De predictieve relaties: de pijlen tonen de berekende bewegingsvectoren.

I -beelden vormen onafhankelijke entiteiten in de videosequentie. Dit impliceert dat I -beelden als volledige beelden gecomprimeerd worden en dat ze de sleutelpunten vormen in de gecomprimeerde data. De gebruikte codeertechnieken zijn discrete cosinustransformatie (DCT), quantizatie, variabele-woordlengte codering en Huffman entropiecodering. De eerste DCT-coëfficiënt wordt de DC-term genoemd en is 8 keer de gemiddelde intensiteit van het respectievelijk blok.

Macroblokken van P -beelden kunnen gecodeerd worden met een voorwaartse bewegingscompensatie naar het dichtbijzijnde voorgaande referentiebeeld (I of P), macroblokken van B -beelden kunnen dan zowel voorwaarts als achterwaarts of geïnterpoleerd refereren. Bewegingscompensatie wordt later in detail uitgelegd in paragraaf 3.3.1. Na bewegingscompensatie van de macroblokken worden de residuele fouten met DCT geëncodeerd. Bij encoding wordt bij elke macroblok van B - en P -beelden getest wat de beste oplossing is: bewegingscompensatie of intracodering (zoals bij I -beelden). De twee mogelijke types voor P -beelden zijn intracodering en voorwaartse predictie; voor B -beelden zijn er vier mogelijkheden: intracodering, voorwaartse, achterwaartse en geïnterpoleerde predictie.

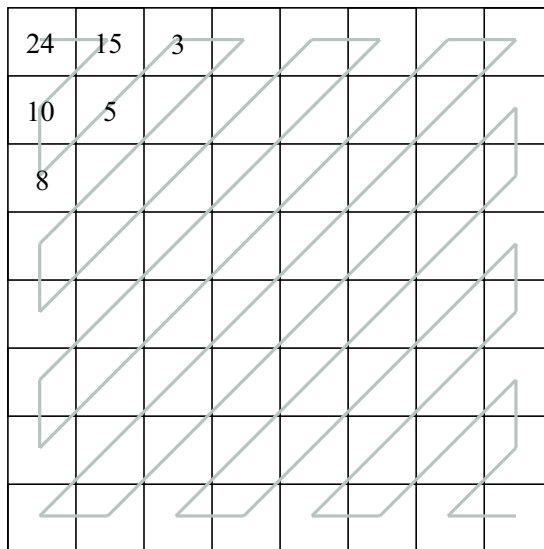
Belangrijk om te weten is dat de resultaten van de volgende technieken afhankelijk zijn van de gebruikte MPEG-encoder. De MPEG-standaard definieert wel de basisconcepten, maar niet de implementatie. Voor meer informatie over MPEG verwijzen we naar gespecialiseerde literatuur [MPFL97].

2.3.2 DCT-coëfficiënten

Deze techniek bestaat erin de DCT-coëfficiënten onderling te correleren. Deze coëfficiënten komen in 8×8 blokmatrices voor zoals getoond in figuur 2.28. Belangrijk om te weten is dat de elementen bij zigzag-scan (of verticale scan) min of meer gesorteerd staan: naarmate het einde van de scan komen er meer (irrelevante) nullen in voor. Daarom wordt er meestal gewerkt met slechts een subset van de DCT-coëfficiënten van een deel van blokken om een representatieve vector te construeren voor een beeld:

$$V_i = \{c_1, c_2, c_3, \dots, c_k\} \quad (2.42)$$

De afstandsmaat tussen twee beelden wordt dan gedefinieerd door een genormaliseerd inwendig product van twee vectoren:



Figuur 2.28: Een 8×8 DCT-matrix met zigzag-scan van de elementen.

$$\Psi = 1 - \frac{|V_i \bullet V_{i+\phi}|}{|V_i| |V_{i+\phi}|} \quad (2.43)$$

met ϕ het aantal beelden tussen de twee beelden die met elkaar vergeleken worden, gewoonlijk is dit de afstand tussen twee I -beelden⁸ omdat de DCT-coëfficiënten rechtstreeks uit de I -beelden gehaald kunnen worden. Ψ wordt dan met een drempel δ vergeleken om overgangen te detecteren.

Een alternatieve methode is de paarsgewijze vergelijkingstechniek (besproken in 2.2.1), maar dan toegepast op de DCT-coëfficiënten van de corresponderende blokken in plaats van pixelintensiteiten. Het verschil tussen de beelden wordt gegeven door:

$$D(i, i + \phi) = \frac{1}{64} \sum_{k=1}^{64} \frac{|c_{l,k}(i) - c_{l,k}(i + \phi)|}{\max(c_{l,k}(i), c_{l,k}(i + \phi))} \quad (2.44)$$

met l het respectievelijk blok. De detectie van cuts verloopt geheel analoog aan de paarsgewijze pixelvergelijking.

Er zijn voor- en nadelen aan verbonden aan deze techniek: enerzijds is de rekentijd (dankzij een beperkte subset van de coëfficiënten) veel kleiner dan de paarsgewijze pixelvergelijking, maar anderzijds kan het verlies van temporele resolutie tussen de I -beelden valse positieven veroorzaken.

2.3.3 DC-beelden

DC-beelden worden gevormd door de eerste DCT-coëfficiënt (de zogenaamde DC-term) van elke DCT-matrix te nemen. DC-beelden zijn spatiaal gereduceerde versies van de

⁸Bij *Motion-JPEG* kan dit berekend worden op elk beeld.

originele beelden: de (i, j) pixel van het DC-beeld is het gemiddelde van de (i, j) blok van het origineel beeld (zie figuur 2.29).



Figuur 2.29: Voorbeeld van een DC-beeld: (a) origineel beeld en (b) DC-beeld.

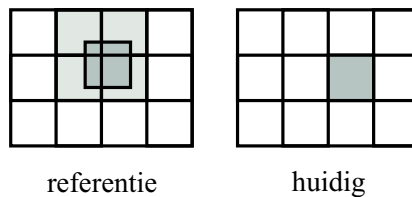
We kunnen alle technieken van temporele segmentatie op ongecomprimeerde video analoog gebruiken indien we ook de DC-beelden van B - en P -beelden construeren. De spatiale reductie van DC-beelden in vergelijking tot de originele beelden bedraagt 64, en zoveel zal het ook sneller zijn dan de technieken besproken in paragraaf 2.2.

Het volgend algoritme zal snel de DC-termen reconstrueren van B - en P -beelden:

1. de DC-term van het gerefereerde beeld (DC_{ref}) wordt benaderd door een gewogen gemiddelde van de DC-termen van de blokken aangeduid door de bewegingsvectoren (zie figuur 2.30):

$$DC_{ref} = \frac{1}{64} \sum_{\alpha \in E} N_{\alpha} \cdot DC_{\alpha} \quad (2.45)$$

waarbij DC_{α} de DC-term is van blok α , E is de collectie van alle blokken die overlapt worden door het referentieblok en N_{α} is het aantal pixels in blok α die overlapt worden door het referentieblok.



Figuur 2.30: Estimatie van de DC-term: het verband tussen DC_{ref} en de huidige DC-term.

2. de benaderde DC-termen van de voorspelde beelden worden toegevoegd aan de geëncodeerde DC-termen van het residueel beeldverschil DC_{res} om de DC-termen van B - en P -beelden te vormen:

$$\begin{aligned}
DC_a &= DC_{res} + DC_{ref} \\
DC_b &= DC_{res} + \frac{1}{2}(DC_{ref_1} + DC_{ref_2})
\end{aligned}
\tag{2.46}$$

waarbij DC_a enkel de voorwaartse of achterwaartse predictie voorstelt en DC_b voor geïnterpoleerde predictie geldt.

2.3.4 Macroblokken

We herhalen dat er twee types van macroblokken bestaan voor P -beelden, namelijk intragecodeerd en voorwaarts predictief. Voor B -beelden zijn er vier types: intragecodeerd, voorwaarts, achterwaarts en geïnterpoleerd predictief. Het desbetreffende macroblok is van het type met de laagste kost.

We definiëren drie verhoudingen:

$$R_p = \frac{\kappa(\eta)}{\kappa(\phi)}, \quad R_b = \frac{\kappa(\beta)}{\kappa(\phi)}, \quad R_f = \frac{\kappa(\phi)}{\kappa(\beta)}
\tag{2.47}$$

waarbij η de set van intragecodeerde macroblokken is, analoog zijn ϕ en β respectievelijk van het type voorwaartse en achterwaartse predictie. $\kappa(g)$ is de kardinaliteit (aantal elementen) van de set g .

We overlopen nu de verschillende situaties. Als er een cut is bij een P -beeld, dan zal de encoder meer intragecodeerde macroblokken gebruiken dan voorwaartse predictie. Dit impliceert dat een cut hier overeenstemt met een piek in R_p . Bij een cut bij B -beelden, bevatten de B -beelden hoofdzakelijk macroblokken geëncodeerd met achterwaartse predictie. Een cut stemt dan overeen met een piek in R_b . Bij I -beelden hebben de opeenvolgende B -beelden, die het I -beeld vlak voorafgaan, voornamelijk macroblokken geëncodeerd met voorwaartse predictie. Een cut stemt dan overeen met opeenvolgende pieken in R_f . Figuur 2.31 illustreert dit aan de hand van referenties in GOP.

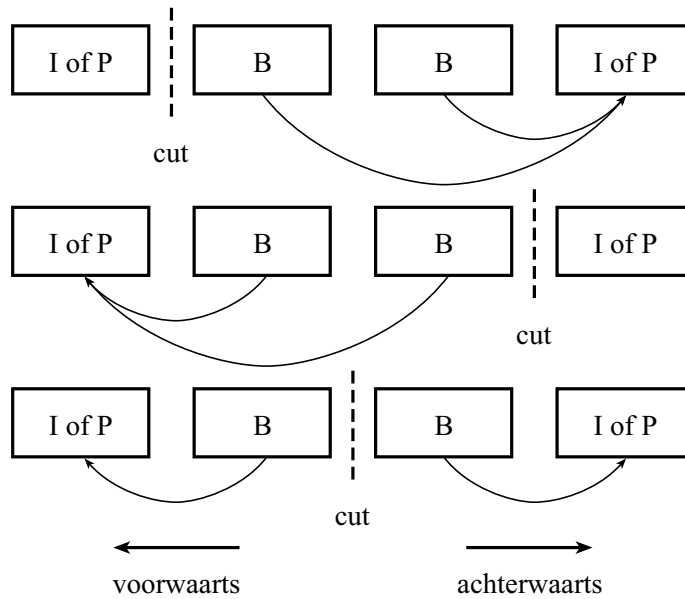
Om dissolves te detecteren, wordt er gekeken naar het type van B -beelden [JYL00]. Het basisprincipe is dat bij een dissolve, de macroblokken refereren naar het dichtsbijzijnde I - of P -beeld. Nemen we algemeen aan dat er steeds twee aansluitende B -beelden tussen I - of P -beelden zitten (zie GOP in (2.41)), dan zal het eerste B -beeld voorwaarts refereren en het tweede B -beeld achterwaarts refereren. We definiëren de voorwaartse macroblok verhouding λ als:

$$\lambda = \begin{cases} \frac{\kappa(\phi)}{\kappa(\phi) + \kappa(\beta)}, & \text{als } \kappa(\phi) + \kappa(\beta) \neq 0 \\ n.v.t., & \text{anders} \end{cases}
\tag{2.48}$$

λ varieert telkens van zeer laag (0) naar zeer hoog (1) bij een dissolve. We assigneren drie predictietoestanden aan B -beelden:

$$\begin{aligned}
T_0 &: (\kappa(\phi) + \kappa(\beta) = 0) \vee (1 - \delta_r \leq \lambda \leq \delta_r) \\
T_+ &: \lambda > \delta_r \\
T_- &: \lambda < 1 - \delta_r
\end{aligned}
\tag{2.49}$$

$$\delta_r > 0.5$$



Figuur 2.31: Referenties van B -beelden bij aanwezigheid van een cut.

met δ_r de drempel die beslist of het B -beeld voorwaarts of achterwaarts predictief is. Bij een dissolve krijgen we b.v. de volgende sequentie: $[T_+, T_-, T_+, T_-, \dots, T_+]$ (zie figuur 2.32). Een bijkomende eis voor dissolves is dat de spatiale distributie van de voorwaarts predictieve macroblokken verspreid moet zitten in het hele B -beeld. Voor wipes is de distributie van de verschillende macrobloktypes wel spatiaal gelokaliseerd in een beperkte strook.

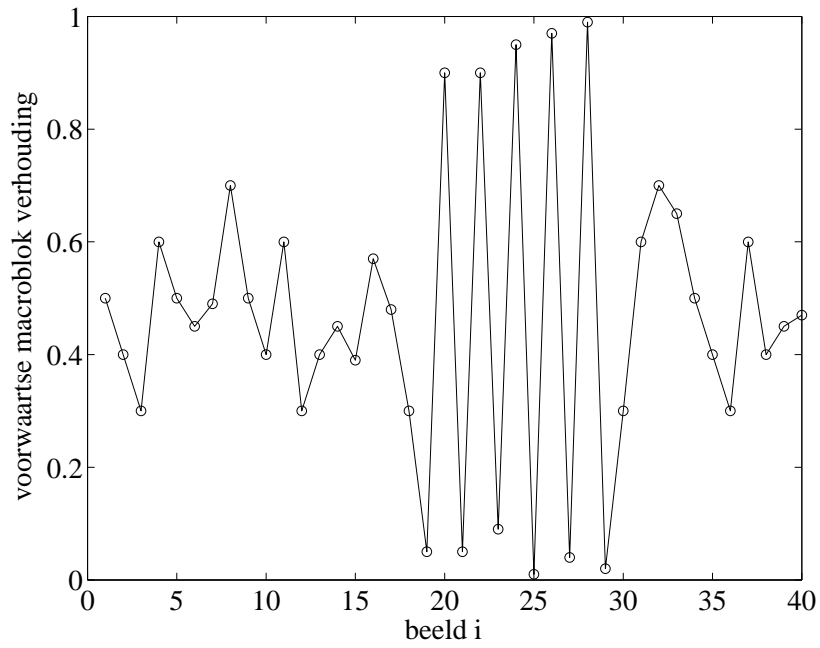
Andere technieken, die gebruik maken van macroblokken, baseren zich op hetzelfde principe [CI01].

2.3.5 Bewegingsvectoren

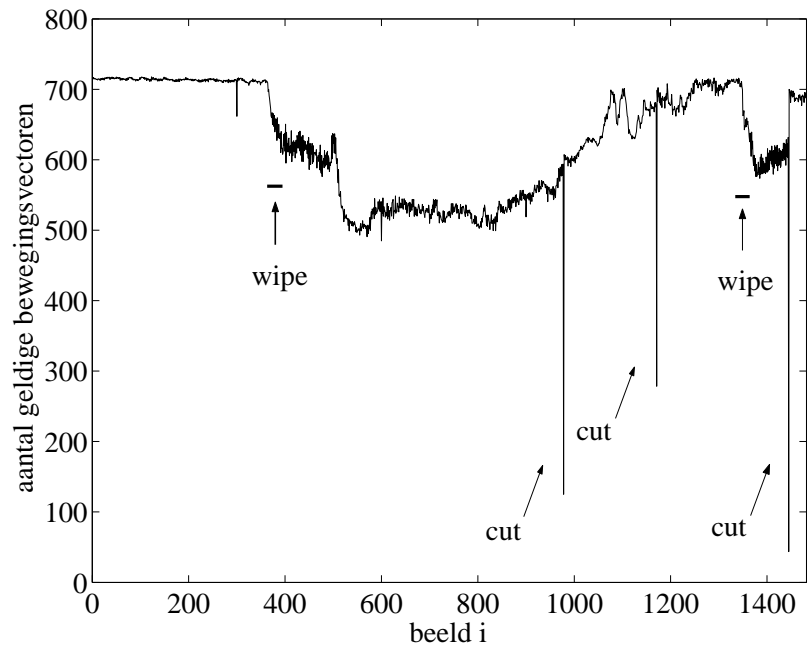
Los van intensiteitswaarden en distributies, maken we hier gebruik van bewegingsvectoren, een hoger niveau kenmerk dat de relatie beschrijft tussen meerdere beelden. Hoe we bewegingsvectoren kunnen berekenen, wordt later besproken in paragraaf 3.3.1.

Net zoals bij macroblokken beperken de technieken zich tot B - en P -beelden. Algemeen vertoont het veld van de bewegingsvectoren in één shot relatief continue veranderingen, terwijl de continuïteit verbroken wordt bij een cut. Een continuïteitsmaat voor een sequentie van bewegingsvelden geldt als een alternatief criterium om cuts te detecteren.

Laat η het aantal geldige bewegingsvectoren zijn voor B - of P -beelden en δ een drempel laag genoeg gekozen zodat $\eta < \delta$ een effectieve cut-indicator is voor of na een B - of P -beeld, zoals geïllustreerd in figuur 2.33. Dit algoritme faalt in het detecteren van transitities: aangezien de verschillen in de opeenvolgende beelden van de sequentie langzaam variëren zullen de meeste bewegingsvectoren geldig blijven.



Figuur 2.32: Voorwaartse macroblok verhouding λ voor een sequentie met een dissolve tussen beeld 19 en 29.



Figuur 2.33: Aantal geldige bewegingsvectoren η uitgezet voor een videosequentie.

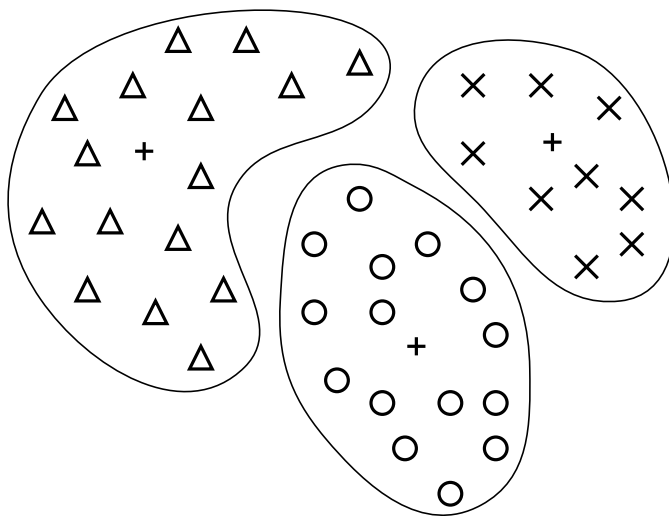
2.3.6 Variabele bitrate

Door het debiet van de informatie te observeren kunnen we cuts detecteren: een grote verandering van de bitrate tussen opeenvolgende *I*- en *P*-beelden duidt op de aanwezigheid

van een cut. Deze detectie is echter louter indicatief: b.v. shots met veel actie vragen immers ook veel meer allocatie van bits. De techniek is ook robuuster bij videos geëncodeerd in een hoge bitrate.

2.4 Scèneclustering

De bedoeling van scèneclustering is om shots, die (semantisch) bij elkaar horen, te groeperen tot één entiteit. Voorbeelden zijn dialogen (waarbij telkens op de spreker ingezoomd wordt), shots opgenomen in dezelfde omgeving, enz. Figuur 2.34 toont het basisprincipe van clustering.



Figuur 2.34: Gelijkaardige shots worden gegroepeerd tot scèneclusters.

Hieronder bespreken we enkele belangrijke methoden.

2.4.1 K-means algoritme

Op basis van beeldkenmerken gaan we de verschillende shots groeperen in clusters. Meestal gebruiken we hiervoor kleuren, clustering op andere kenmerken (zoals vormen) gebeurt op dezelfde manier.

Er zijn verschillende varianten van het K-means clusteringsalgoritme, die gebaseerd zijn op de volgende twee eigenschappen: 1. elke klasse of cluster heeft een centrum, welke het midden is van alle elementen behorende tot die klasse en 2. elk element behoort tot de klasse waarvan het centrum het dichtst bij het element ligt.

Het basialgoritme bestaat uit de volgende stappen:

1. voer temporele segmentatie uit (zie paragrafen 2.2 en 2.3). We bekommen dan s shots.
2. initialiseer de K clustergemiddelden $\mu_1, \mu_2, \dots, \mu_K$. De kenmerkenvector van een shot wordt berekend als het gemiddelde van de kenmerkenvector van alle beelden

behorende tot één shot. De kenmerkenvector kan b.v. geconstrueerd worden door middel van kleurhistogrammen. De shots worden willekeurig in de verschillende klassen opgedeeld, maar zorg dat de afstanden tussen de clustergemiddelden voldoende groot zijn.

3. herhaal tot er geen veranderingen meer zijn die de variantie van de klassen reduceren:
 - elk element wordt naar de klasse verplaatst waarvan het clustergemiddelde het dichtst bij het desbetreffende element ligt.
 - voor elke klasse i wordt μ_i herberekend op basis van de elementen die tot die klasse behoren.

De afstand tussen het clustergemiddelde en een element kan op verschillende manieren berekend worden:

- euclidisch: meestal wordt deze rechtlijnige afstandsmaat gekozen. Voorbeelden zijn $|a - b|$ of $\sqrt{a^2 - b^2}$.
- niet-euclidisch: de afstanden zijn niet rechtlijnig, maar volgen wel bepaalde regels. Voorbeelden zijn de Manhattan-afstand (dit is het minimum aantal horizontale en verticale stappen tussen twee elementen) en de cosinus similariteitsmaat:

$$\frac{\sum x \cdot y}{\sqrt{(\sum x^2)} \sqrt{(\sum y^2)}}$$

Het grote nadeel van dit algoritme is dat we K op voorhand moeten bepalen. Een te kleine K zorgt ervoor dat shots gegroepeerd worden die niet samenhangen en een veel te grote K zorgt ervoor dat de initiële afstanden tussen de clusters te klein zijn om goede resultaten te behalen. Indien K groter is dan het aantal clusters dat we nodig hebben, dan zullen er lege of dode clusters optreden: deze clusters hebben geen belang in het eindresultaat.

Het K-means algoritme is eveneens geschikt voor shotdetectie. Hiervoor moeten we elk individueel beeld laten clusteren; een shot bestaat dan uit een reeks beelden die tot dezelfde cluster behoren. Het grote probleem is dat ook beelden uit een transitie (met een onstabiele inhoud) geclusterd worden, die eigenlijk compleet nutteloos zijn.

2.4.2 Iteratieve boomclustering

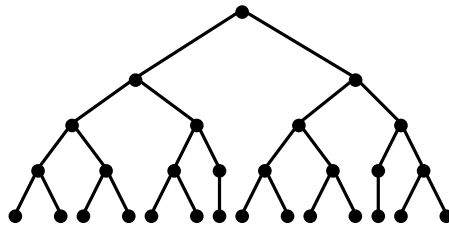
Iteratieve clustering is gebaseerd op dezelfde principes als het K-means clusteringsalgoritme, maar het grote verschil is dat we hier niet gebonden zijn aan een vast aantal clusters.

Het iteratief algoritme verloopt op de volgende wijze:

1. voer temporele segmentatie uit (zie paragrafen 2.2 en 2.3). We bekommen dan s shots.
2. initialiseer de clustergemiddelden $\mu_1, \mu_2, \dots, \mu_s$ als het gemiddelde van de kenmerkenvector van alle beelden behorende tot één shot. De kenmerkenvector kan b.v. geconstrueerd worden door middel van kleurhistogrammen. De clustergemiddelden worden aan een lege lijst toegevoegd.

3. voeg de clusters k en l , die het dichtst bij elkaar liggen, samen tot één cluster k' . $\mu_{k'}$ wordt berekend als het gewogen gemiddelde van μ_k en μ_l , waarbij de gewichten afhangen van de kardinaliteit van clusters k en l . Vervolgens worden clusters k en l verwijderd uit de lijst en $\mu_{k'}$ wordt aan de lijst toegevoegd.
4. herhaal stap 3 tot slechts één cluster overblijft.

Het resultaat is een boom (zoals in figuur 2.35) waarbij de clusterafstanden in de knopen bewaard worden. Om de clusters terug te vinden moeten we vanuit de top vertrekken en de clusterafstand vergelijken met een drempel δ . Is de clusterafstand kleiner dan δ , dan behoren alle eindknopen van de deelboom tot dezelfde cluster.



Figuur 2.35: Een boom met de clusterafstanden in de knopen en waarbij de eindknopen de aparte shots voorstellen.

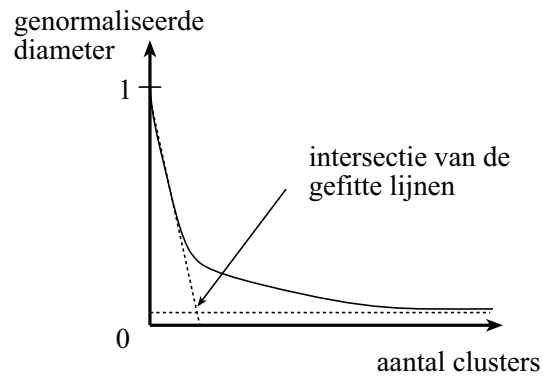
Voor clusterafstanden⁹ kunnen we verschillende methoden gebruiken:

- gemiddelde linkafstand: de clusterafstand is de gemiddelde afstand van alle elementen van cluster k tot alle elementen van cluster l . Dit is hetzelfde als de afstand tussen de twee centrumgemiddelden, indien de afstanden euclidisch zijn.
- maximum of verste buurmethode: de dissimilariteit tussen twee clusters is de grootste dissimilariteit tussen een element van cluster k en cluster l , dit wil zeggen de grootste afstand tussen de elementen van de verschillende clusters.
- minimum of dichtste buurmethode: analoog aan maximum of verste buurmethode.
- inwendige groepsafstand: de clusters worden gevormd door de variantie van de elementen van de clusters te minimaliseren.
- Ward's methode: de clusterafstand is de som van de euclidische kwadratische afstand tussen alle elementen uit de verschillende clusters.

We selecteren de optimale drempel δ door de knik te vinden in de curve van de genormaliseerde clusterdiameter. Bij het verkleinen van het aantal clusters zullen op een gegeven moment irrelevante shots samen geclusterd worden waardoor de diameter van de clusters enorm zal toenemen. Een typisch verloop van de curve wordt gegeven in figuur 2.36.

Evenals het K-means algoritme is ook het iteratief algoritme bruikbaar voor shotdetectie. De eindknopen zijn dan de individuele beelden.

⁹Niet te verwarren met de gewone afstandsmaat tussen twee elementen !



Figuur 2.36: Het verloop van de gemiddelde genormaliseerde diameter in functie van het aantal clusters met de aanduiding voor de optimale drempel δ .

2.4.3 Temporele correctie en integratie van domeinkennis

Bij clustering kan het voorkomen dat twee shots gegroepeerd worden tot één scène, die totaal niet bij elkaar horen. De waarschijnlijkheid dat twee shots, die temporeel ver uit elkaar liggen, bij elkaar horen is echter klein. Daarom kan er als bijkomende voorwaarde gesteld worden dat twee shots tot dezelfde scène behoren indien ze ook temporeel dicht bijeen liggen.

Met integratie van de domeinkennis, kunnen we de samenvatting van videosequenties semantisch verbeteren. Dit gebeurt met behulp van sjablonen (zie paragraaf 2.2.9): shots die totaal verschillend zijn, maar toch bij elkaar horen, zullen dan in één scène gegroepeerd worden (een voorbeeld hiervan zijn de verschillende weerkaarten in een nieuwsuitzending). We kunnen ook bepaalde (onbelangrijke) scènes excluseren uit onze samenvatting (b.v. volledig zwarte beelden, ...).

Hoofdstuk 3

Shotrepresentatie

3.1 Inleiding

Het doel van dit hoofdstuk is om een gebruiksvriendelijke webpagina te ontwikkelen die in staat is om een videosequentie hiërarchisch te doorzoeken. Hiërarchisch doorzoeken van een videosequentie is te verkiezen boven het sequentieel doorzoeken¹ wegens de grote tijdswinst. Naarmate we in hiërarchie dalen wordt de zoekruimte kleiner en krijgen we meer details van de videosequentie. Een typische hiërarchie in dalende volgorde is: de volledige videosequentie, scènes en shots (zie figuur 1.1).

In dit hoofdstuk bespreken we hoe de samenvatting van een videosequentie met behulp van sleutelbeelden aan de gebruiker kan voorgesteld worden. We stellen elke shot (en scène) voor door één of meerdere representatieve sleutelbeelden te selecteren. Voor shots met camerabewegingen, opteren we voor panoramische beelden. De panoramische beelden geven een overzicht van de volledige shot weer op één beeld. Nadat we de sleutelbeelden geëxtraheerd hebben uit de shots en scènes, gaan we ze zo efficiënt mogelijk weergeven op een webpagina.

De geselecteerde sleutelbeelden vormen eveneens de basis voor indexering van multimedia databanken.

In de volgende secties gaan we na hoe de sleutelbeelden worden geselecteerd en vervolgens worden voorgesteld aan de gebruiker.

3.2 Selectie van sleutelbeelden

Bij het selecteren van de sleutelbeelden, halen we de meest representatieve sleutelbeelden uit elke shot, dit zijn de sleutelbeelden die de meeste relevante informatie bevatten over de shots. Als we voor elke shot slechts één sleutelbeeld kiezen, dan hebben we de volgende mogelijkheden:

- eerste of laatste beeld: deze beelden zijn meestal niet representatief als de inhoud van de shot varieert of als ze resten bevatten van mogelijke transities (indien de grenzen van de transities niet voldoende nauwkeurig bepaald zijn).

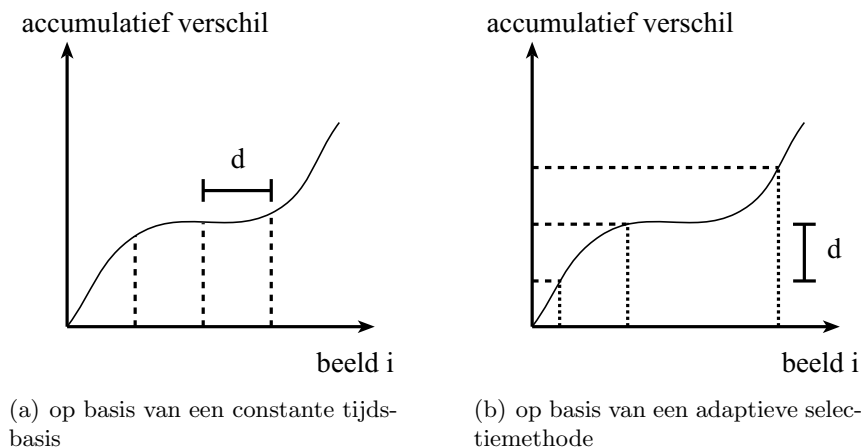
¹Bij analoge video kunnen we als hulp bij het doorzoeken gebruik maken het snel vooruit- en achteruitspoelen. Bij digitale video kunnen we in software ook gebruik maken van een tijdsbalk.

- middelste beeld: deze beelden zijn meestal wel representatief, maar de methode faalt weer als de inhoud varieert.
- het beeld dichtst bij het gemiddelde: hierbij berekenen we het gemiddelde van een kenmerkenvector (gebaseerd op histogrammen of andere kenmerken), het beeld die het dichtst bij het gemiddelde ligt wordt gekozen als sleutelbeeld. Deze beelden zijn meestal representatief. Maar als de inhoud varieert, zullen ze onvoldoende informatie bevatten.

Voor scènes kiezen we uit de set van sleutelbeelden van de shots die behoren tot de desbetreffende scène. We kiezen hiervoor het sleutelbeeld van de grootste shot.

We kunnen voor elke shot ook meer dan één sleutelbeeld kiezen, dit zal vooral ten goede komen voor shots met een variërende inhoud. We stellen enkele manieren voor om n sleutelbeelden uit een shot te halen:

- we verdelen de shot met lengte l in $n + 1$ gelijke stukken en de sleutelbeelden worden geselecteerd op de plaatsen $\left\lceil \frac{l}{n+1} \right\rceil$, $\left\lceil \frac{2 \cdot l}{n+1} \right\rceil$, \dots , $\left\lceil \frac{n \cdot l}{n+1} \right\rceil$. De sleutelbeelden staan temporeel op een constante afstand van elkaar, namelijk $\frac{l}{n+1}$. Figuur 3.1 toont ons de selectie van de sleutelbeelden. Op de figuur zien we dat de kans bestaat dat we meerdere gelijkaardige sleutelbeelden kiezen, zodat we geen extra informatiewinst halen.
- met een adaptieve methode selecteren we sleutelbeelden in dezelfde shot die veel van elkaar verschillen. We berekenen hiervoor het accumulatief verschil tussen de beelden (dit kan met behulp van kleurhistogrammen gebeuren) en we selecteren de beelden die op een constant accumulatief verschil liggen zoals aangegeven in figuur 3.1. De gekozen sleutelbeelden zijn gevarieerder en bieden ons meer informatie over de shot.



Figuur 3.1: Selectie van drie sleutelbeelden in functie van het accumulatief verschil.

Het verschil tussen de twee bovenstaande methoden wordt geïllustreerd in figuur 3.2. Hierbij gaat de voorkeur naar de adaptieve methode omdat die meer informatie bevat dan



(a) op basis van een constante tijdsbasis

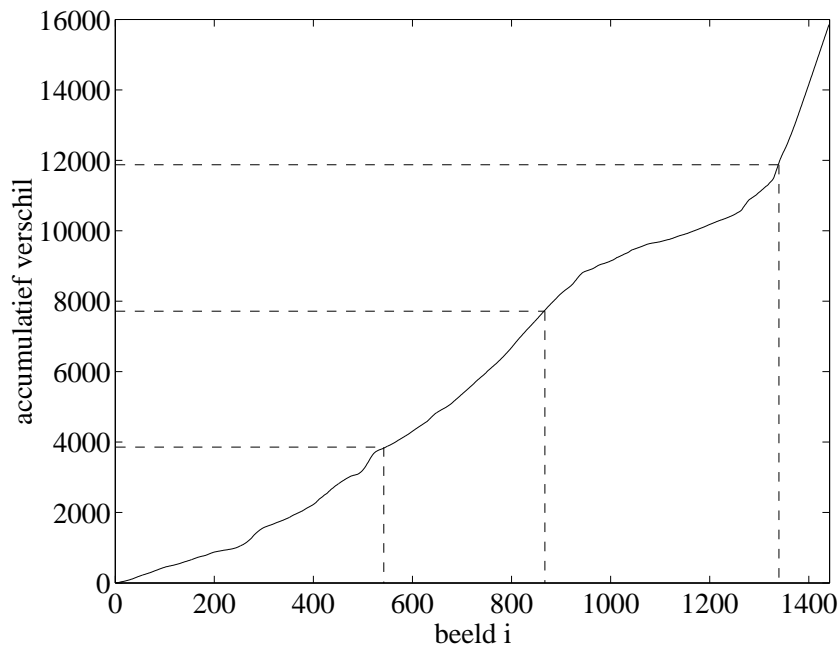


(b) op basis van een adaptieve selectiemethode

Figuur 3.2: Voorbeeld van een selectie van drie sleutelbeelden. De adaptieve methode biedt ons meer informatie over de shot.

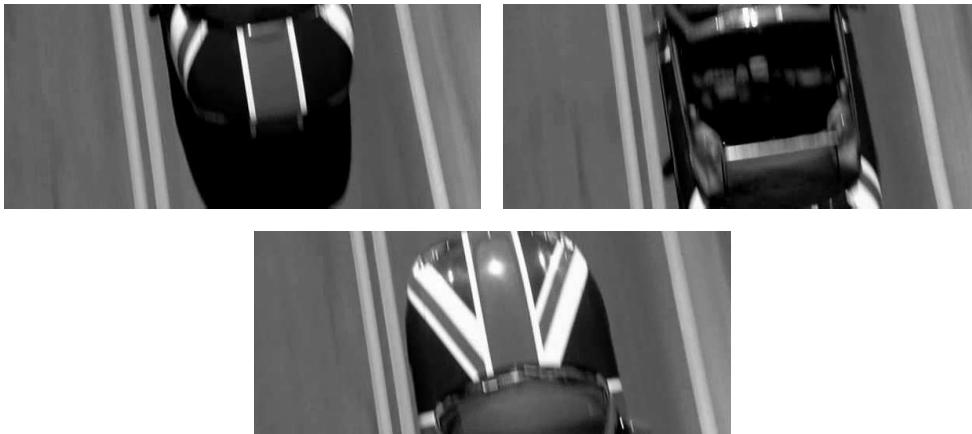
de methode met de constante tijdsbasis. Het verloop van het accumulatief verschil voor figuur 3.2(b) wordt gegeven in figuur 3.3.

Indien we meerdere sleutelbeelden selecteren voor scènes, kunnen we ook gebruik maken van de clusterinformatie: b.v. de beelden die op een variërende afstand van het clustergemiddelde liggen.



Figuur 3.3: Het verloop van het accumulatief verschil voor figuur 3.2(b).

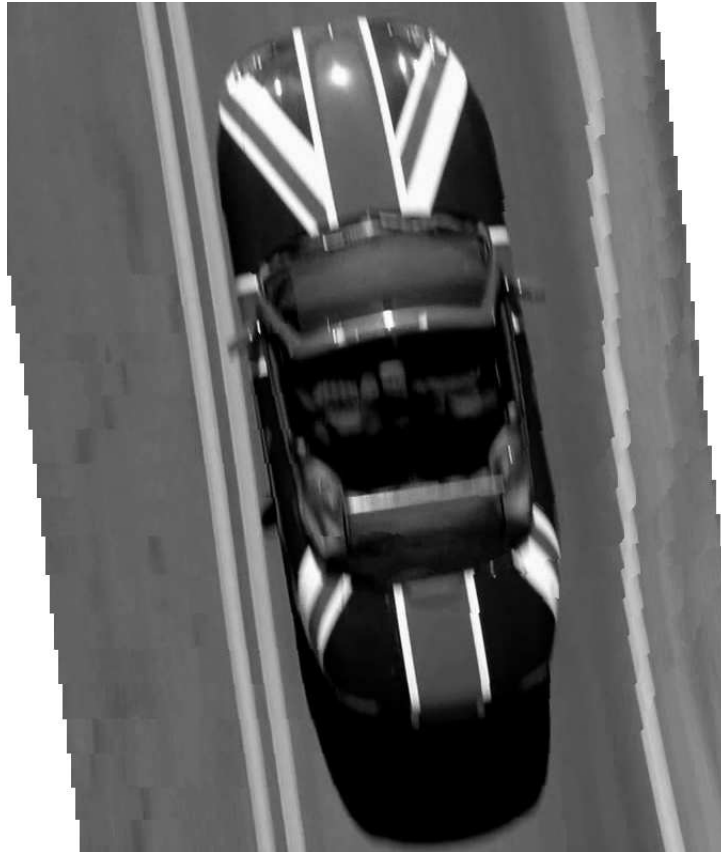
Toch zijn meerdere sleutelbeelden soms niet genoeg om een shot efficiënt voor te stellen zoals geïllustreerd in figuur 3.4. Dit probleem komt vaak voor als er camerabewegingen zijn. We kunnen de shots dan efficiënter voorstellen door één panoramisch beeld. We bespreken dit in de volgende paragraaf.



Figuur 3.4: Inefficiënte voorstelling van een shot met camerabeweging door drie sleutelbeelden.

3.3 Panoramische beelden

Panoramische beelden kennen we het best als langwerpige foto's of als virtuele 3D-werelden op het internet. Maar panoramische beelden hebben nog veel andere toepassingen zoals het creëren van hoge resolutie-beelden, videocompressie (*MPEG-4 sprite coding*), reconstructie van omgevingen onafhankelijk van hun grootte of bereik, . . . Zo'n panoramisch beeld biedt ons veel meer informatie aan dan één of meerdere sleutelbeelden en wordt daarom gebruikt om een shot met camerabeweging voor te stellen. In figuur 3.5 wordt het panoramisch beeld gegeven van de shot die voorgesteld werd in figuur 3.4.



Figuur 3.5: Een panoramisch beeld. De inefficiënte voorstelling wordt gegeven in figuur 3.4.

De constructie van panoramische beelden verloopt in twee fasen: eerst wordt de camerabeweging geëstimeerd door de twee opeenvolgende beelden ten opzichte van elkaar te registreren, vervolgens wordt de pixelinformatie geselecteerd uit de overlappende regio's om het canvas te vullen.

3.3.1 Registratie

Bewegingsmodellen

Alle registratiemethoden baseren zich op een bewegingsmodel. We proberen de camera-operaties te bepalen door één dominante beweging terug te vinden in de shot en vervol-

gens identificiëren we de dominante beweging met de parameters van het vooropgestelde model. We stellen enkel 2-dimensionale transformaties op in dit werk (3-dimensionale bewegingsmodellen zijn ook mogelijk met een projectie op een 2-dimensionaal vlak).

De meest eenvoudige bewegingstransformatie is de translatie:

$$\begin{pmatrix} x \\ y \end{pmatrix}_i = \begin{pmatrix} x \\ y \end{pmatrix}_{i-1} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \quad (3.1)$$

Het beeld wordt over (d_x, d_y) verschoven ten opzichte van het vorige beeld. Komt er rotatie bij, dan wordt het model bepaald door:

$$\begin{pmatrix} x \\ y \end{pmatrix}_i = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}_{i-1} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \quad (3.2)$$

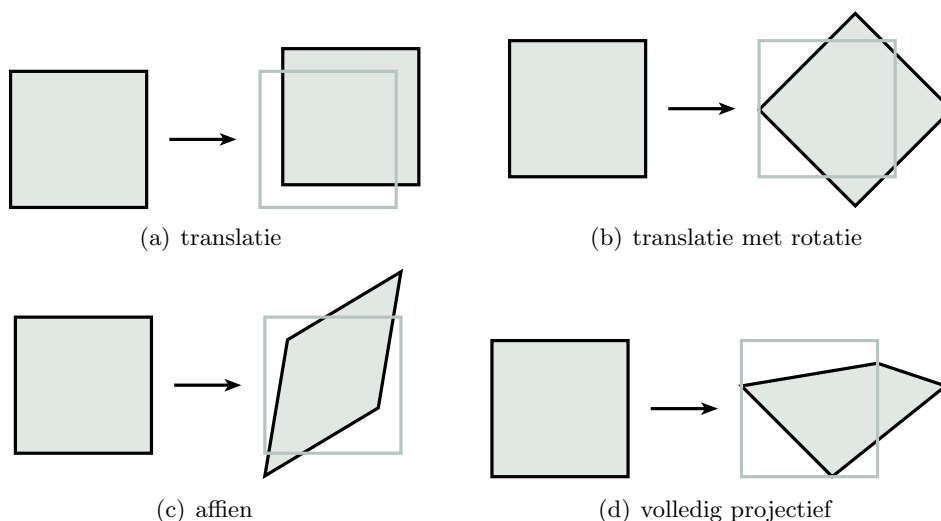
met θ de rotatiehoek. Een 2-dimensionaal affine transformatie wordt gegeven door:

$$\begin{pmatrix} x \\ y \end{pmatrix}_i = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}_{i-1} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \quad (3.3)$$

De volledig 2-dimensionale projectieve transformatie wordt gegeven door:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix}_i = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}_{i-1} \quad (3.4)$$

waarbij de triplet (x, y, w) naar cartesische coördinaten $(\frac{x}{w}, \frac{y}{w})$ wordt teruggebracht met w als schalingsfactor. De transformatiematrix heeft slechts acht vrijheidsgraden: twee transformatiematrices zijn equivalent als ze scalaire veelvouden zijn van elkaar, deze redundantie kan vermeden worden door $a_{33} = 1$ te stellen. De transformaties worden geïllustreerd in figuur 3.6.

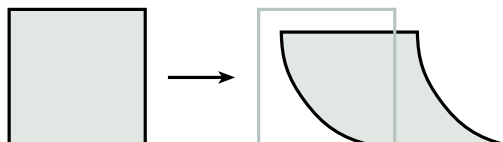


Figuur 3.6: Vier 2-dimensionale transformaties van de eerste orde.

Er bestaan nog tal van complexere bewegingsmodellen, ze zijn van hogere polynomiale orde zoals de parabolische transformatie [SSO99]:

$$\begin{pmatrix} x \\ y \end{pmatrix}_i = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{pmatrix} \begin{pmatrix} x \\ y \\ x^2 \\ y^2 \\ xy \end{pmatrix}_{i-1} + \begin{pmatrix} d_x \\ d_y \end{pmatrix} \quad (3.5)$$

Figuur 3.7 illustreert de parabolische transformatie.



Figuur 3.7: De parabolische transformatie.

De registratie tussen twee beelden kan goed beschreven worden door bewegingsmodellen van de eerste orde. Maar over een langere periode (een reeks van beelden) kan een gesofisticeerde bewegingsmodellering nodig zijn, zeker als de geobserveerde objecten een complexe dieptestructuur (d.i. de afstand tot de camera) hebben. Hiervoor zijn hoge orde polynomiale modellen nodig.

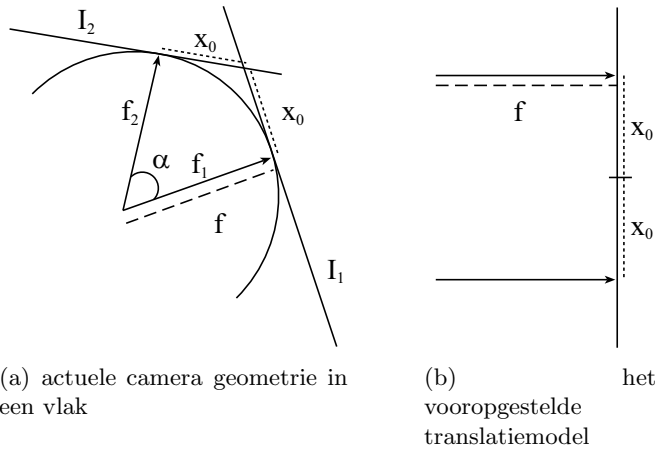
De parameters van het bewegingsmodel zijn niet dezelfde als de intrinsieke camera-parameters. Figuur 3.8 illustreert dit met een vereenvoudigd camerarotatie in een vlak en het daarvoor vooropgestelde bewegingsmodel. Beeldvlakken die verkregen worden vanuit een projectiepunt zijn bolvormig, het bewegingsmodel daarentegen veronderstelt dat de beeldvlakken coplanair zijn. Dit verklaart de grote-hoek lensvervorming. De cameraparameters (b.v. de sferische hoekrotatie (α, β)) kunnen berekend worden indien de brandpuntsafstand f gekend is. Gedetailleerde berekeningen worden gegeven in [Dav98]. De grote-hoek lensvervorming wordt geïllustreerd in figuur 3.9, de grote-hoek lensvervorming is het bolvormig effect van het landschap.

We bespreken nu enkele methoden om de parameters uit het bewegingsmodel te schatten.

Bewegingsvectoren - blokgebaseerde bewegingsestimatie

De MPEG-standaard (zie paragraaf 2.3.1) gebruikt 2-dimensionale bewegingsvectoren om een betere compressie te behalen. Bewegingsvectoren van hogere orde is theoretisch mogelijk, maar wordt in de praktijk nauwelijks gebruikt. De bewegingsvector, bepaald door (d_x, d_y) , wordt voor elk blok onafhankelijk berekend. De berekeningen voor de voorwaartse, achterwaartse en geïnterpoleerde estimatie gebeuren op analoge wijze.

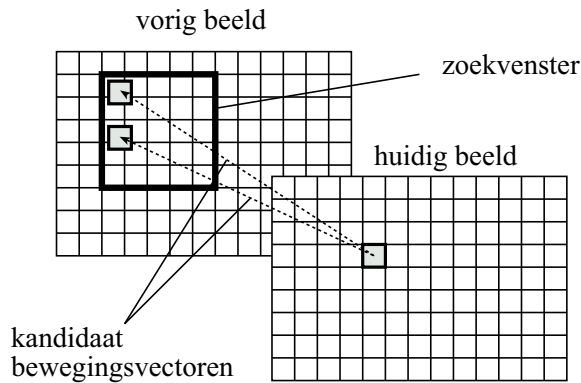
Om de optimale bewegingsvector te bepalen, definiëren we een zoekruimte of zoekvenster. De zoekruimte is meestal een vierkant $[-X, X]$ zoekvenster, gecentreerd rond de positie van het beschouwd blok zoals in figuur 3.10. X wordt groot genoeg gekozen zodat de reële bewegingsvector binnen het zoekvenster ligt. Maar hoe groter X , hoe meer



Figuur 3.8: De intrinsieke cameraparameters.



Figuur 3.9: De grote-hoek lensvervorming in een panoramisch beeld.



Figuur 3.10: Het bepalen van de optimale bewegingsvector in een zoekvenster.

rekentijd het algoritme vraagt (de rekentijd is kwadratisch in functie van X !). Er zijn immers $(X + 1)^2$ kandidaat bewegingsvectoren die geëvalueerd moeten worden. Voor elke kandidaat bewegingsvector wordt de kostfunctie berekend. De kandidaat met de minimale kost wordt dan gekozen als de bewegingsvector van het blok. Een populaire kostfunctie is

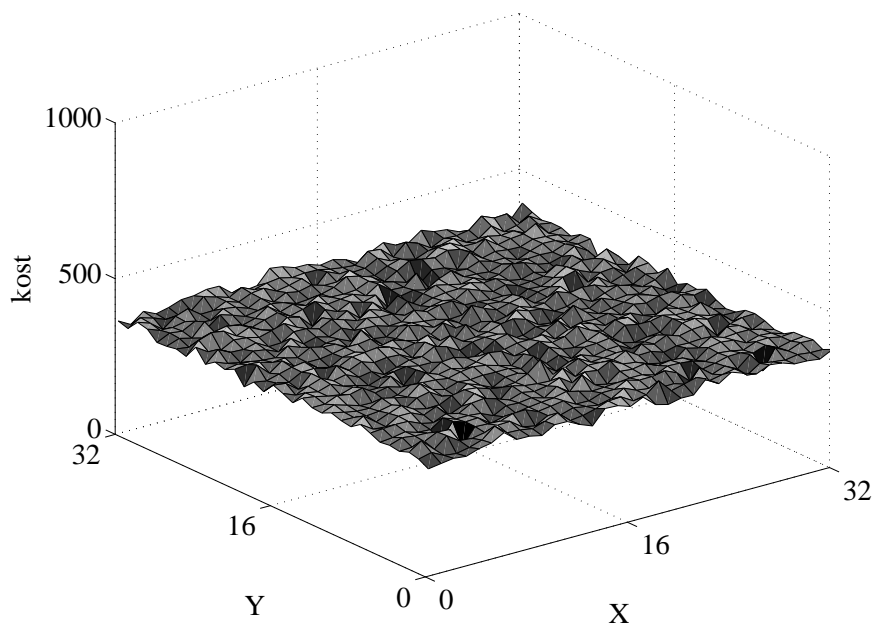
de gemiddeld kwadratische fout (of het verschil) van de pixelintensiteiten:

$$E(d_x, d_y) = \frac{1}{n^2} \sum_{k,l=-n/2}^{n/2} [I_i(k + x_k, l + y_k) - I_{i-1}(k + x_k + d_x, l + y_k + d_y)]^2 \quad (3.6)$$

met (x_k, y_k) het middelpunt van het beschouwd blok en $|d_x| \leq X$, $|d_y| \leq X$ en n is de dimensie van het vierkant blok. Analoog kunnen we ook het gemiddeld absolute verschil als kostfunctie berekenen:

$$E(d_x, d_y) = \frac{1}{n^2} \sum_{k,l=-n/2}^{n/2} |I_i(k + x_k, l + y_k) - I_{i-1}(k + x_k + d_x, l + y_k + d_y)| \quad (3.7)$$

In de literatuur worden nog tal van andere kostfuncties voorgesteld [Sme02]. De evaluatie met behulp van een kostfunctie wordt geïllustreerd in figuur 3.11.



Figuur 3.11: De 2-dimensionale kostfunctie of beeldverschil energie voor een $[-32, 32]$ zoekvenster.

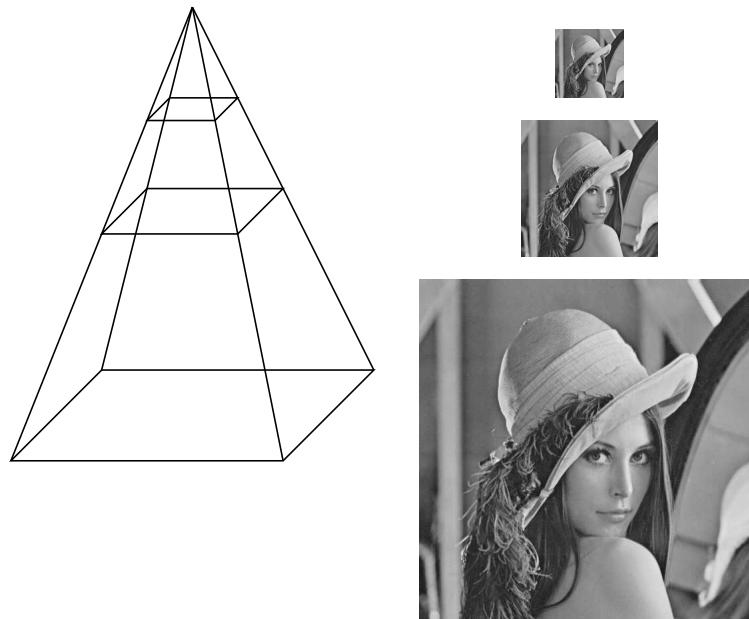
Er kunnen verschillende zoekmethoden gebruikt worden om de bewegingsvector te bepalen. De zoekmethoden worden gedefinieerd door een stapgrootte (om van een kandidaat bewegingsvector naar de volgende te gaan) en een zoekpad (b.v. zigzag (van links naar rechts en van boven naar onderen) of in een spiraal (om de nulvector te bevoordelen), ...).

Er zijn wel nadelen verbonden aan deze basistechniek (met betrekking tot camerabeweging):

- de bewegingsvector voor elk blok wordt onafhankelijk van zijn burens berekend. Hierdoor kunnen de bewegingsvectoren zeer verschillend zijn dan die van hun burens terwijl de blokken hetzelfde bewegingspad volgen. Dit komt b.v. vaak voor in grote uniforme vlakken waarbij heel veel kandidaat bewegingsvectoren zijn die een zeer lage kost hebben. De resultaten zijn dan zeer onbetrouwbaar.
- de volledige zoekruimte voor elk blok aftasten en elke kostfunctie berekenen kan optimale oplossingen geven, maar is uitermate rekenintensief.

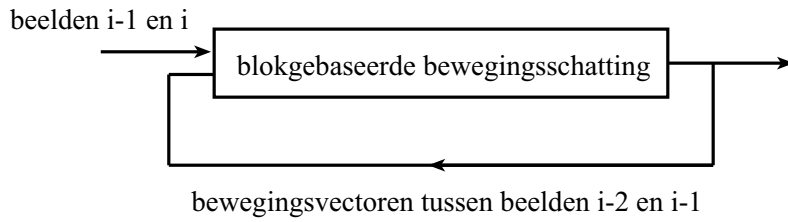
Er zijn tal van varianten van de blokgebaseerde bewegingsestimatie die betere resultaten opleveren [Sme02].

Een variant van de blokgebaseerde bewegingsestimatie is de hiërarchische zoekmethode. Deze methode werkt met gereduceerde versies (in dimensie) van de originele beelden. De initiële bewegingsschatting begint met het beeld met de kleinste resolutie, aan de top van de piramide (zie figuur 3.12). Het resulterend bewegingsveld wordt als input gegeven bij afdaling in de hiërarchie. Het zoekvenster per blok kan bij het volgend niveau nauwer gekozen worden rond de reeds gevonden initiële kandidaat bewegingsvector. De belangrijkste voordelen van deze methode is de lagere rekentijd en de stabiele bewegingsvectoren.



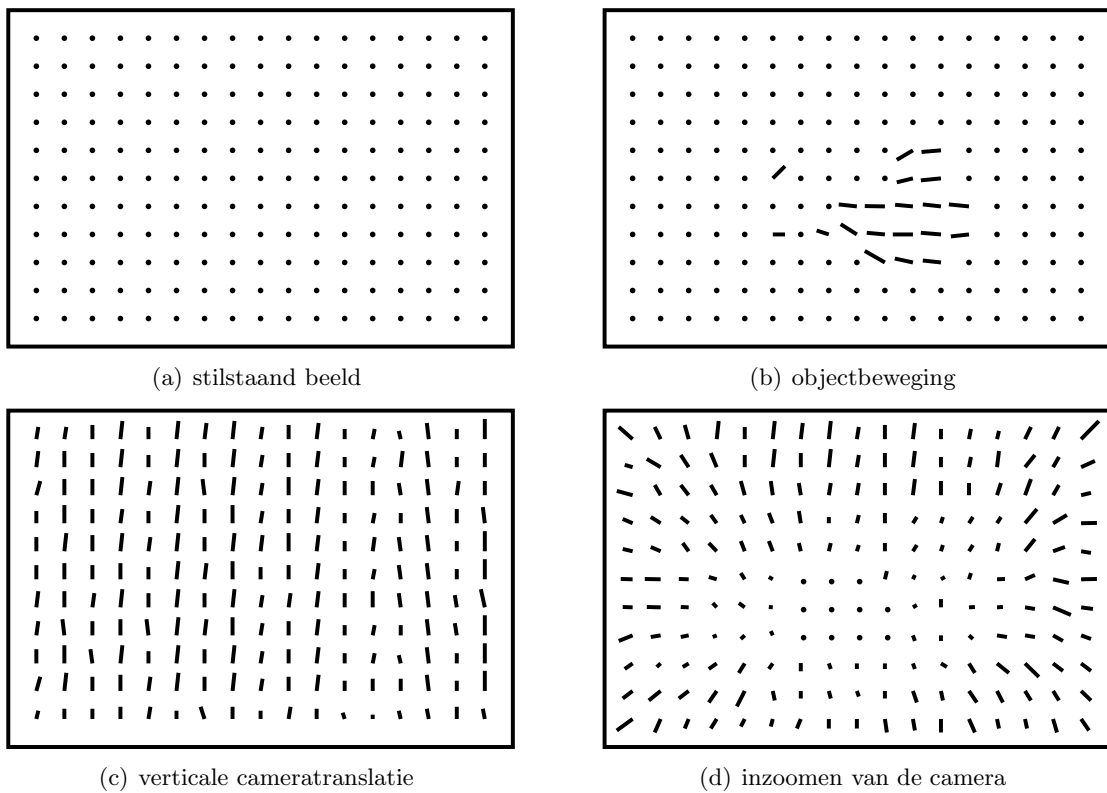
Figuur 3.12: Hiërarchische zoekmethode.

Om de snelheid nog op te voeren kunnen nog andere aanpassingen gemaakt worden: de zoektocht naar de bewegingsvector met de laagste kost kan vroegtijdig afgebroken worden als de kost D kleiner is dan een drempel δ . Dit geeft wel een suboptimale oplossing. En bij de schatting van de bewegingsvelden kunnen we de informatie van de schatting tussen beeld $i - 2$ en $i - 1$ in een teruggekoppelde lus gebruiken voor de schatting tussen beeld $i - 1$ en i zoals aangetoond in figuur 3.13.



Figuur 3.13: Terugkoppeling bij de blokgebaseerde bewegingsschatting: de bewegingsvectoren tussen de vorige beelden worden als extra input ingevoerd.

De resultaten, die we tot nu toe hebben, zijn bewegingsvelden. Figuur 3.14 toont enkele bewegingsvelden. Uit deze velden zullen we de camerabewegingen halen.



Figuur 3.14: Bewegingsvelden.

Voor een cameratranslatie zullen de meeste vectoren dezelfde richting hebben, de dominante richting. Dit impliceert dat de meeste vectoren parallel zijn aan de modale vector (van de dominante richting):

$$\sum_{b=1}^N |\theta_b - \theta_m| \leq \Theta_p \quad (3.8)$$

hierbij is θ_b de richting van de bewegingsvector van blok b en is N het totaal aantal blokken. De drempel Θ_p wordt zeer klein gekozen. De modale vector stelt dan de translatie

voor.

We kunnen ook het in- en uitzoomen van de camera detecteren. Tijdens de zoom hebben de verticale componenten van de bovenste en onderste rij een tegengesteld teken [FSZ95]. Hetzelfde geldt voor de horizontale componenten van de linkse en rechtse kolom. We hebben dus een zoom gevonden als de volgende voorwaarden voldaan zijn:

$$\begin{aligned} |v_k^{boven} - v_k^{onder}| &\geq \max\left(|v_k^{boven}|, |v_k^{onder}|\right) \\ |h_k^{links} - h_k^{rechts}| &\geq \max\left(|h_k^{links}|, |h_k^{rechts}|\right) \end{aligned} \quad (3.9)$$

waarbij v_k en h_k respectievelijk de verticale en horizontale componenten zijn van de k -de bewegingsvector.

Een andere en snelle manier om de translatie te bepalen gebeurt via een 2-dimensionaal histogram van de bewegingsvectoren. Eerst worden alle bewegingsvectoren uitgemiddeld met hun burens en wordt het histogram berekend (met als histogramcoördinaten de x- en y-richting). De translatie wordt voorgesteld door een piek in het 2-dimensionaal histogram.

In [FSZ95] gebeurt de analyse van bewegingsvectoren op basis van de Hough transformatie.

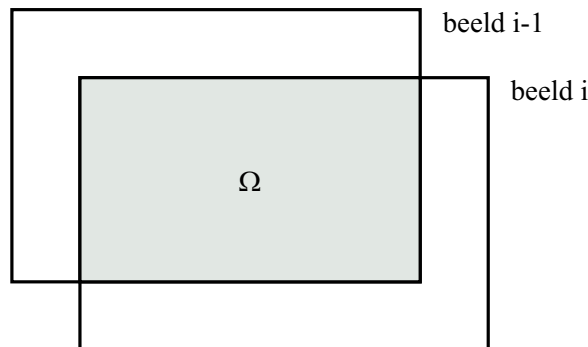
Bewegingsvectoren hebben het grote voordeel dat ze reeds gecodeerd zitten in een MPEG-bestand, maar zijn soms onstabiel. Vooral grote objectbewegingen kunnen de schatting van de camerabeweging zeer moeilijk maken.

Globale minimalisatie van de kwadratische fout

Bij alignatie van twee beelden, wordt de kwadratische intensiteitsfout van het overlappend gebied Ω (zie figuur 3.15) geminimaliseerd. De kwadratische intensiteitsfout wordt gegeven door:

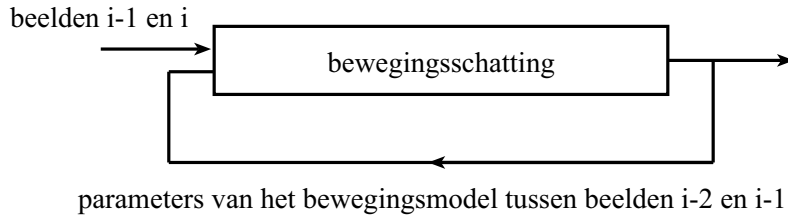
$$E(d_x, d_y) = \frac{1}{\kappa(\Omega)} \sum_{(x,y) \in \Omega} [I_i(x, y) - I_{i-1}(x + d_x, y + d_y)]^2 \quad (3.10)$$

met $\kappa(\Omega)$ het aantal pixels in Ω . De kwadratische fout wordt dan geminimaliseerd [Sze96, NG01].



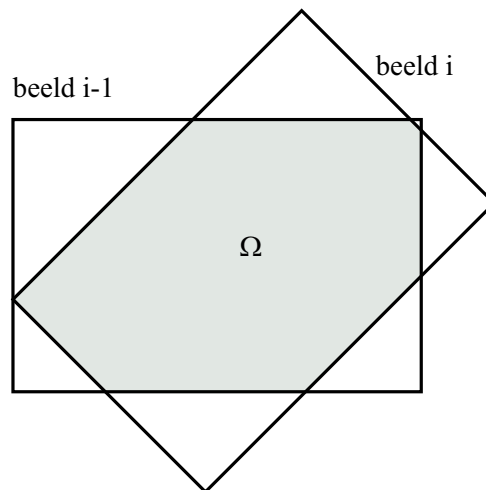
Figuur 3.15: Het overlappingsgebied waarvoor de kwadratische fout wordt berekend.

Om de rekentijd te beperken wordt er gebruik gemaakt van een vierkant $[-X, X]$ zoekvenster. In een videosequentie kan ook de informatie van de bewegingsschatting, tussen de vorige beelden, gebruikt worden in een teruggekoppelde lus zoals aangetoond in figuur 3.16. Hierdoor kan het zoekvenster kleiner gekozen worden en zal het algoritme veel minder rekentijd vragen.



Figuur 3.16: Terugkoppeling bij de bewegingsschatting.

Het algoritme is uitbreidbaar voor rotatie (zie figuur 3.17). Het volledig beeld i wordt geroteerd over een hoek θ (zie vergelijking (3.2)). De schatting van θ kan eveneens teruggekoppeld worden als inputwaarde (zie figuur 3.16).



Figuur 3.17: Het overlappingsgebied bij rotatie.

Deze techniek blijft echter vatbaar voor objectbewegingen. Andere kostfuncties kunnen gebruikt worden om de globale fout te minimaliseren, b.v. een 2-dimensionale genormaliseerde kruiscorrelatie functie [Bro92] of uitbreidingen naar kleurenruimtes (b.v. RGB):

$$E(d_x, d_y) = \frac{1}{\kappa(\Omega)} \sum_{(x,y) \in \Omega} \sum_c [P_i(x, y, c) - P_{i-1}(x + d_x, y + d_y, c)]^2 \quad (3.11)$$

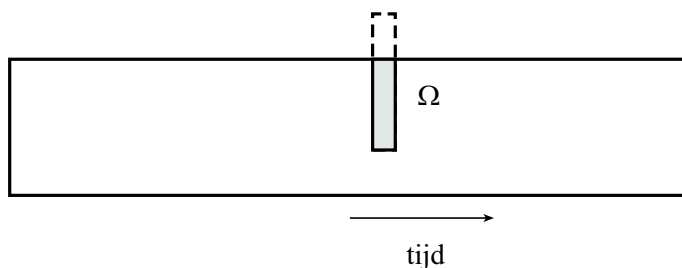
met $P_i(x, y, c)$ de kleurcomponent c ($c \in \{\text{rood}, \text{groen}, \text{blauw}\}$) van de pixel op (x, y) in beeld i .

Camera bewegingsdetectie via snede-analyse

We vertrekken van horizontaal en verticaal spatioal-temporele beelden (voorbeelden zie figuur 2.16). Elke kolom van een spatioal-temporeel beeld stelt de snede voor van een beeld. We verschuiven de sneden van beeld i ten opzichte van de sneden van beeld $i - 1$ zoals aangegeven in figuur 3.18 en berekenen de kwadratische fout van de intensiteiten:

$$\begin{aligned} E_h(d_x) &= \frac{1}{\kappa(\Omega_h)} \sum_{y \in \Omega_h} [S_h(i, y) - S_h(i - 1, y + d_x)]^2 \\ E_v(d_y) &= \frac{1}{\kappa(\Omega_v)} \sum_{y \in \Omega_v} [S_v(i, y) - S_v(i - 1, y + d_y)]^2 \end{aligned} \quad (3.12)$$

met Ω het overlappingsgebied en $\kappa(\Omega)$ het aantal pixels die de verschoven snede gemeenschappelijk heeft met het spatioal-temporeel beeld. De translatie (d_x, d_y) wordt gevonden door E_h en E_v te minimaliseren.



Figuur 3.18: Het overlappingsgebied bij verschuiving van de sneden.

Deze techniek geeft snelle resultaten, maar is veel minder robuust en minder betrouwbaar dan andere technieken. Vooral objectbewegingen vormen hier een groot probleem. De betrouwbaarheid kan verhoogd worden door meer spatioal-temporele beelden te beschouwen. Snede-analyse kan gebruikt worden als snelle initiële schatting voor andere technieken.

Fase correlatie en Mellin-transformatie

De beweging van objecten kunnen de kwaliteit van de registratie tussen beelden sterk degraderen. De fase correlatie-techniek is echter niet onderhevig aan deze objectbewegingen [Dav98].

Deze methode berekent de omgekeerde translatie $(d'_x, d'_y) = (-d_x, -d_y)$ tussen elk beeldpaar $\{I_{i-1}, I_i\}$. De Fouriergetransformeerden $\{\mathcal{I}_{i-1}, \mathcal{I}_i\} = \{\mathcal{F}[I_{i-1}], \mathcal{F}[I_i]\}$ van elk beeldpaar kunnen gerelateerd worden met de faseverschuiving:

$$\begin{aligned} I_i(x, y) &= I_{i-1}(x - d'_x, y - d'_y) \\ \mathcal{I}_i(\xi, \eta) &= e^{-j2\pi(\xi d'_x + \eta d'_y)} \cdot \mathcal{I}_{i-1}(\xi, \eta) \end{aligned} \quad (3.13)$$

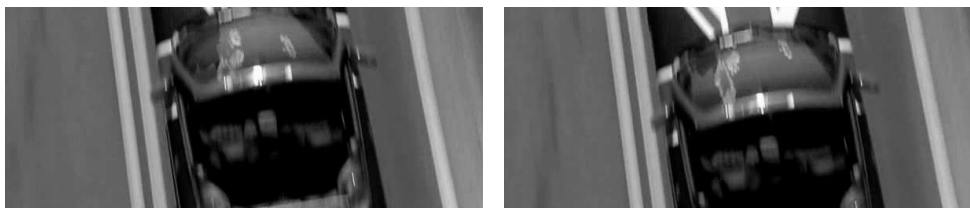
De faseverschuiving $e^{-j2\pi(\xi d'_x + \eta d'_y)}$ kan berekend worden als:

$$e^{-j2\pi(\xi d'_x + \eta d'_y)} = \frac{\mathcal{I}_{i-1}^*(\xi, \eta) \mathcal{I}_i(\xi, \eta)}{|\mathcal{I}_{i-1}^*(\xi, \eta) \mathcal{I}_i(\xi, \eta)|} \quad (3.14)$$

waarbij \mathcal{I}^* de complex toegevoegde is van \mathcal{I} . De inverse Fouriertransformatie van de faseverschuiving is een deltafunctie op (d_x, d_y) in het correlatievlak:

$$\delta(x - d'_x, y - d'_y) = \mathcal{F}^{-1} \left[e^{-j2\pi(\xi d'_x + \eta d'_y)} \right] \quad (3.15)$$

Bij objectbeweging zullen er meerdere pieken bevinden in het correlatievlak: de energie wordt verdeeld over deze pieken. De deltafunctie van de cameratranslatie zal de meeste energie bevatten (op voorwaarde dat de objecten kleiner zijn dan de zichtbare achtergrond) en de positie van deze hoofdpiek wordt niet beïnvloed door objectbewegingen. Figuren 3.19, 3.20 en 3.21 illustreren de deltafunctie.



Figuur 3.19: Translatie tussen twee opeenvolgende beelden.

De Mellin-transformatie is een uitbreiding van de fase correlatie-techniek om de rotatiehoek θ te berekenen. De relatie tussen de beelden wordt gegeven door:

$$\begin{aligned} I_i(x, y) &= I_{i-1}(x \cos \theta + y \sin \theta - d'_x, -x \sin \theta + y \cos \theta - d'_y) \\ \mathcal{I}_i(\xi, \eta) &= e^{-j2\pi(\xi d'_x + \eta d'_y)} \cdot \mathcal{I}_{i-1}(\xi \cos \theta + \eta \sin \theta, -\xi \sin \theta + \eta \cos \theta) \end{aligned} \quad (3.16)$$

We zien dat de magnitude van het spectrum van \mathcal{I}_i een geroteerde versie is van \mathcal{I}_{i-1} . De rotatie kan gevonden worden door de spectra van \mathcal{I}_{i-1} en \mathcal{I}_i om te zetten in poolcoördinaten:

$$|\mathcal{I}_i(r, \vartheta)| = |\mathcal{I}_{i-1}(r, \vartheta - \theta)| \quad (3.17)$$

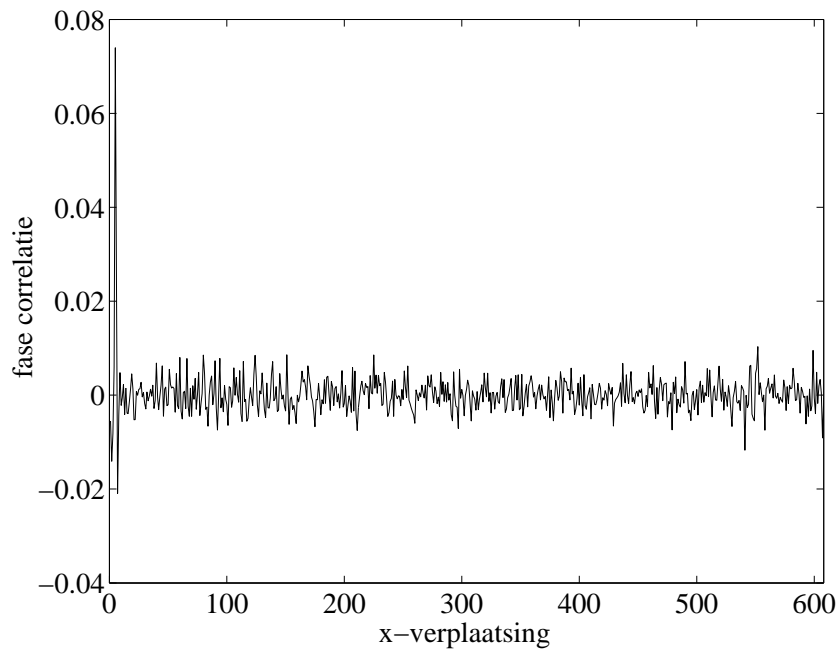
Een rotatie van de spectra komt overeen met een translatie in poolcoördinaten. Nadat de spectra op rotatie geëaligneerd zijn (rotatie over $-\theta$), kunnen de translatieparameters (d_x, d_y) worden gevonden in de spectra van \mathcal{I}_{i-1} en \mathcal{I}_i in cartesische coördinaten. De schaalfactor kan eveneens berekend worden als we gebruik maken van een log-polaire transformatie.

Een kleine beperking van deze techniek, is dat de opeenvolgende beelden meer dan 50% moeten overlappen. Dit vormt echter geen probleem in een videosequentie.

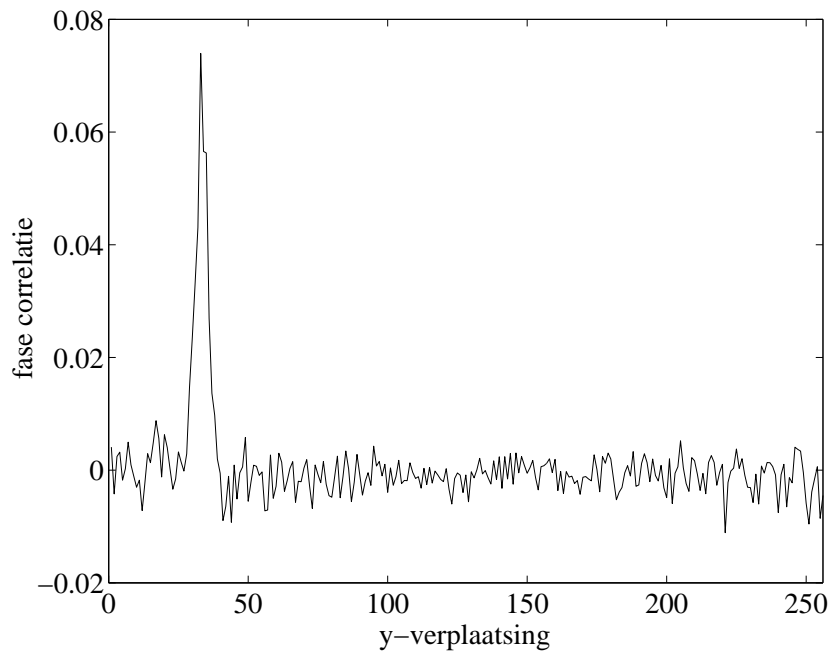
Kanade-Lucas-Tomasi tracking

De Kanade-Lucas-Tomasi tracker modelleert de beweging van een kenmerk (b.v. de vorm) als pure translatie, op basis van de minimalisatie van de som van het kwadratisch intensiteitsverschil in de opeenvolgende beelden.

Bij de estimatie van de camerabeweging, gebruiken we niet meer het volledig beeld, maar controlevensters met goede kenmerken (b.v. hoeken). In de eerste fase moeten we



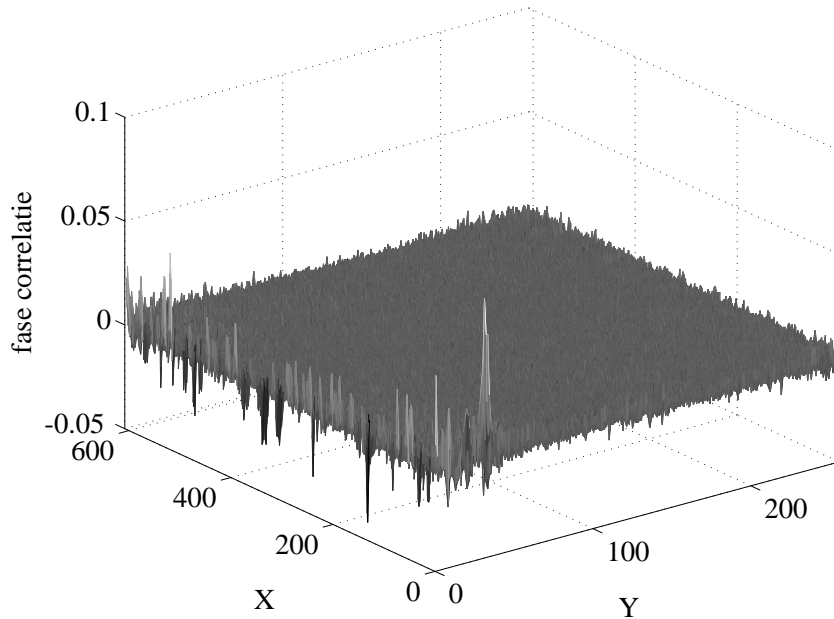
(a) de horizontale translatie



(b) de verticale translatie

Figuur 3.20: Translatie van figuur 3.19 geëstimeerd op basis van fase correlatie: $(d_x, d_y) = (4, 32)$. De fase correlatie vertoont sterke pieken op de translatiecoördinaten.

goede controlevensters of controlepunten selecteren. Goede controlepunten zijn punten die een betrouwbare oplossing hebben voor de volgende vergelijking:



Figuur 3.21: De magnitude van de energie van het volledig vlak voor figuur 3.19.

$$Z\mathbf{d} = \mathbf{e} \quad (3.18)$$

waarbij Z de volgende 2×2 -matrix is:

$$Z = \iint_W \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})w(\mathbf{x})d\mathbf{x} \quad (3.19)$$

en \mathbf{e} de volgende 2×1 -vector is:

$$\mathbf{e} = 2 \iint_W [I_i(\mathbf{x}) - I_{i-1}(\mathbf{x})] \mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} \quad (3.20)$$

De volledige afleiding van de Kanade-Lucas-Tomasi tracker wordt gegeven in bijlage B. Voorbeelden van gekozen controlepunten wordt geïllustreerd in figuur 3.22. De vensters moeten groot genoeg gekozen worden om de gevoeligheid voor ruis te verkleinen. Anderzijds vragen grote vensters veel meer rekentijd. De controlevensters worden goed verspreid gekozen in het beeld om zo onafhankelijk mogelijke resultaten te bekomen. Daarom wordt er een minimumafstand tussen de vensters vastgelegd.

In de tweede fase proberen we dezelfde controlevensters terug te vinden in het volgend beeld. Voor n controlevensters krijgen we n translatievergelijkingen (zie vergelijking (3.1)). We schatten de k parameters van het bewegingsmodel uit deze n vergelijkingen met $n \gg k$. We kunnen de k onbekenden berekenen met Newton-Raphson, maar deze methode kan divergeren. Een betere methode om de parameters te fitten, maakt gebruik van de volgende matrixbewerkingen:



Figuur 3.22: Keuze van controlepunten met behulp van Kanade-Lucas-Tomasi tracker.

$$\begin{aligned}
 MX &= U \\
 X &= (M^T M)^{-1} M^T U
 \end{aligned}
 \tag{3.21}$$

waarbij X de vector vormt met de onbekende parameters, b.v. de zes parameters voor affine transformatie, en X is meteen de kleinste kwadraten oplossing. De matrix M voor de affine transformatie wordt gevormd door de coördinaten van de gekozen controlepunten (x_i, y_i) in beeld $i - 1$:

$$M = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}
 \tag{3.22}$$

De matrix V voor de affine transformatie wordt gevormd door de coördinaten van de gevolgde controlepunten (u_i, v_i) in beeld i :

$$V = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ \vdots & \vdots \\ u_n & v_n \end{bmatrix} \quad (3.23)$$

Bij het volgen van de controlepunten kan het gebeuren dat een controlepunt *verloren* gaat, de tracker vindt het punt niet meer terug om diverse redenen: er is te veel ruis, het punt ligt buiten het beeld of gaat verloren door objectbewegingen. De kans bestaat dat de tracker verkeerde resultaten geeft (verkeerde coördinaten voor de gevolgde controlepunten). Daarom worden in de praktijk m controlepunten geselecteerd met $m \gg n$. Van de m controlepunten worden de n beste resultaten behouden (via een gesorteerde lijst).

Het grote voordeel van deze techniek is de flexibiliteit om de meer complexe bewegingsmodellen te berekenen. Deze techniek kan theoretisch gebruikt worden voor hogere orde bewegingsmodellen, maar zal in de praktijk onnauwkeurige resultaten opleveren voor deze bewegingsmodellen. De techniek is zeer snel, maar ook zeer gevoelig voor objectbewegingen als de objecten groter zijn dan het controlevenster.

Een voorbeeld van een panoramisch beeld dat gebruik maakt van een affien bewegingsmodel wordt gegeven in figuur 3.23. De andere besproken methoden zijn niet geschikt, te traag of niet nauwkeurig genoeg.

Globale vs. lokale registratie

Er zijn twee manieren om beelden te registreren: lokale of sequentiële registratie en globale of canvasregistratie.

Lokale registratie gebeurt door telkens twee beelden met elkaar te vergelijken. De projectiematrix op het canvas wordt gegeven door:

$$P_i = \prod_{j=2}^i A_{j,j-1} \quad (3.24)$$

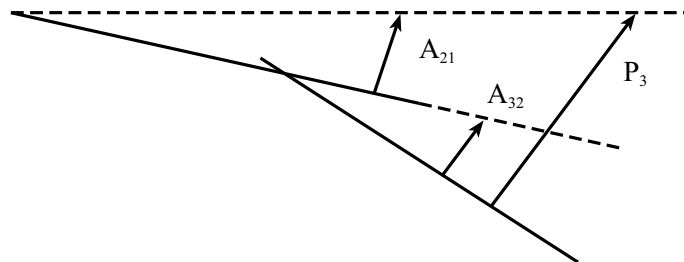
met $A_{j,j-1}$ de registratiematrices en $A_{1,0} = I$ de eenheidsmatrix. De lokale registratie wordt geïllustreerd in figuur 3.24.

Er kunnen fouten optreden bij de registratie: door objectbeweging, door onnauwkeurigheden (b.v. quantizatie op pixelniveau), ... Omdat de lokale registratie tussen twee beelden geen informatie opneemt van registratie tussen andere beelden, kunnen de fouten propageren doorheen de videosequentie, zoals aangetoond in figuur 3.25. Als er dus één slechte registratie voorkomt, dan wordt de sequentie doorbroken (het panoramisch beeld is niet meer vloeiend en de discontinuïteiten zijn duidelijk zichtbaar).

Bij globale registratie registreren we de beelden ten opzichte van het canvas en hernieuwen we het canvas na elke registratie. Het grote voordeel is dat het beeld geregistreerd wordt op de globaal beste plaats, met als resultaat een vloeiend panoramisch beeld. Er is wel een probleem bij globale registratie: door accumulatie van de fouten, kan het soms onmogelijk zijn om het beeld nog te fitten. Er ontstaat een *patstelling* zoals aangetoond in figuur 3.26. De globale registratie vraagt ook meer rekentijd omdat het aantal mogelijkheden veel groter is dan bij lokale registratie (het canvas is veel groter dan een beeld).

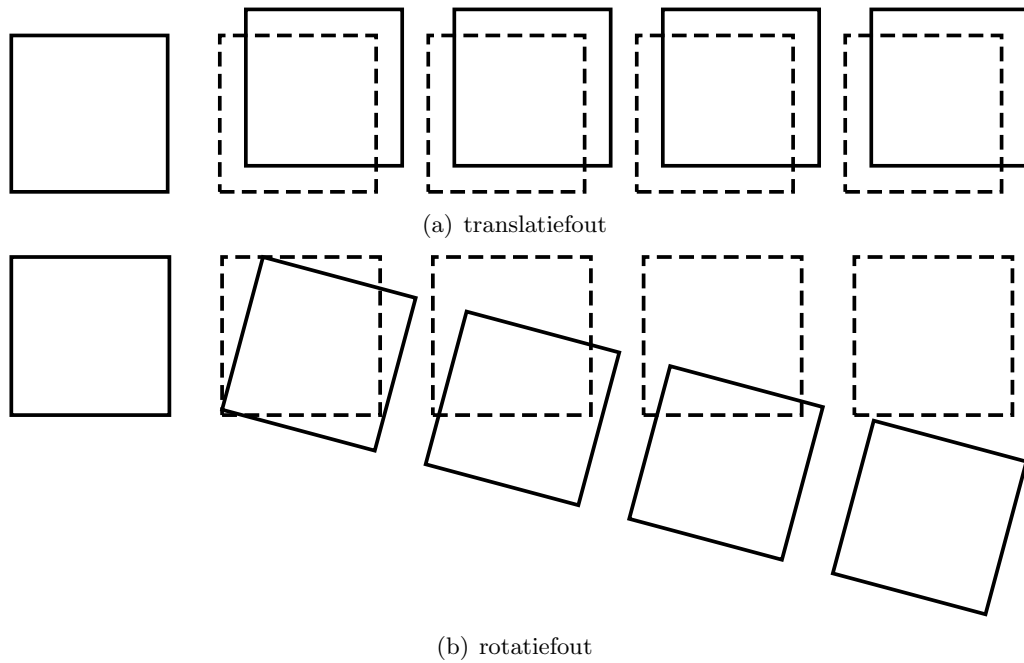


Figuur 3.23: Een panoramisch beeld geconstrueerd met behulp van de Kanade-Lucas-Tomasi tracker (affien bewegingsmodel).

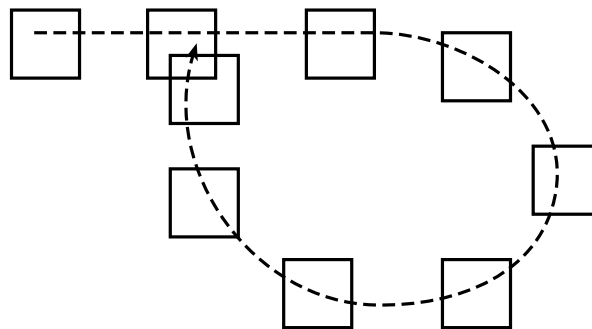


Figuur 3.24: De sequentiële registratie en het verband tussen de projectiematrix P en de registratiematrices A .

Er bestaan enkele algoritmen om de registratiemethode verbeteren [NG01]. Een ver-

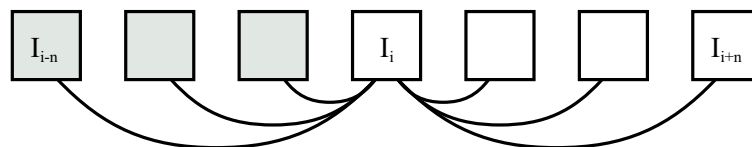


Figuur 3.25: Accumulatie van de fouten.



Figuur 3.26: Situatie waarbij het onmogelijk kan zijn om het beeld nog te fitten.

betering voor de lokale registratie is het vensteralgoritme (zie figuur 3.27). De registratie voor beeld I_i is het gewogen gemiddelde van alle relevante registraties in het venster $[i - n, i + n]$. Een registratie is relevant als de som van de kwadratisch intensiteitsverschillen onder een drempel δ ligt. De methode levert een vloeiend panoramisch beeld op, maar is veel complexer dan gewone lokale registratie.



Figuur 3.27: Het vensteralgoritme met de lokale registraties voor beeld I_i .

Het multifase algoritme registreert de beelden in verschillende fasen in stijgende complexiteit. We registreren het beeld eerst met het translatiemodel (zie vergelijking (3.1)). Eens we de parameters van de translatie bepaald hebben, kunnen we het model verfijnen met rotatie en zoeken we de rotatiehoek (zie vergelijking (3.2)). En zo doen we verder voor de schalingsfactor, de affiene parameters en de projectieve parameters. Een hoger niveau wordt pas gekozen als er een significante verbetering optreedt in de foutmeting E :

$$E_{\text{totaal}} = \min(\min(\min(\min(E_t, E_r \cdot c), E_s \cdot c), E_a \cdot c), E_p \cdot c) \quad (3.25)$$

met E_t , E_r , E_s , E_a en E_p respectievelijk de foutmeting voor translatie, rotatie, schaling, affiene en projectieve registratie. De constante c wordt groter dan 1 gekozen (typisch $1.05 < c < 1.10$) om de complexiteit te beperken.

De gecombineerde methode maakt gebruik van lokale en globale registratie. Eerst wordt lokale registratie toegepast, daarna gebruikt globale registratie de berekende parameters van de lokale registratie als initiële schatting. Hiermee combineren we de goede eigenschappen van beide registraties: de gecombineerde methode is stukken sneller dan de globale registratie en het panoramisch beeld vertoont veel minder discontinuïteiten dan bij de lokale registratie.

Het verschil tussen lokale en globale registratie wordt geïllustreerd in figuur 3.28. Globale registratie levert duidelijk een kwalitatief beter beeld op dan lokale registratie voor onstabiele camerabewegingen.

3.3.2 Plaatsing op het canvas

Er bestaan verschillende methoden pixelinformatie te selecteren voor het canvas. Het doel is om de overgangen tussen de verschillende geregistreerde beelden zo vloeiend mogelijk te maken, de discontinuïteiten worden dus geminimaliseerd. Hierbij kunnen we informatie (de kleurcomponenten) van pixels gebruiken die afkomstig zijn van één of meerdere beelden. Bij pixelinformatie uit één beeld, selecteren we eerst het beeld uit de overlapping en gebruiken we de pixels van dat beeld. Bij pixelinformatie uit meerdere beelden, maken we gebruik van de informatie van alle pixels die elkaar overlappen.

Informatie uit één beeld

Eerste, middelste of laatste beeld Bij selectie van het eerste beeld, vullen we het canvas met het eerste beeld. Bij de daaropvolgende beelden wordt er naar de afzonderlijke pixels gekeken: is er overlapping met het reeds gevulde canvas, dan wordt er niets ondernomen. Is er geen overlapping, dan worden de pixels van het beeld op het canvas geplaatst. Deze methode wordt getoond in figuur 3.29.

De selectie van het laatste beeld gebeurt op analoge wijze (figuur 3.30).

De selectie van het middelste beeld gebeurt eveneens op dezelfde manier (figuur 3.31). Deze laatste techniek wordt ook wel Voronoi-mozaïek genoemd.

Bij deze methoden zijn de discontinuïteiten duidelijk zichtbaar zoals in figuur 3.32. Indien er objectbeweging is of als de belichting verandert, zullen deze distorsielijnen nog duidelijker worden.

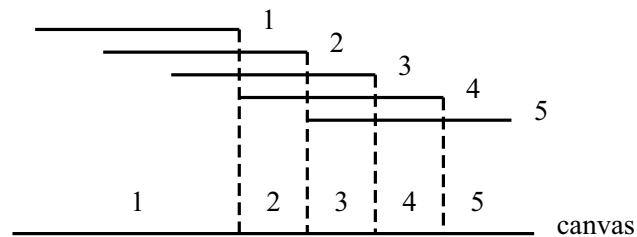


(a) Lokale registratie



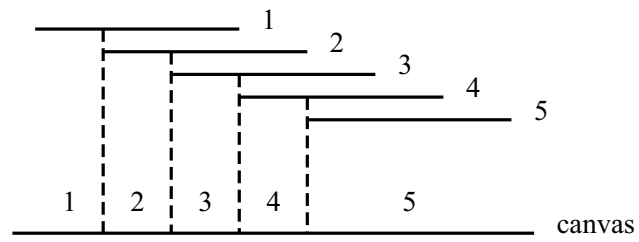
(b) Globale registratie

Figuur 3.28: Het verschil tussen lokale en globale registratie.

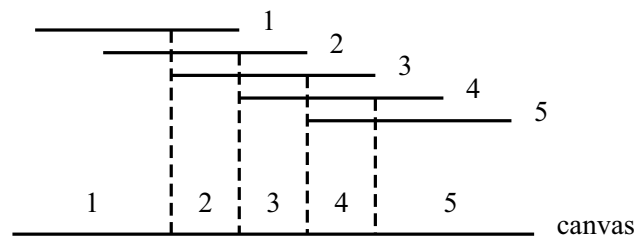


Figuur 3.29: De beeldinformatie wordt geselecteerd uit het eerste beeld van de overlapping bij het vullen van het canvas.

Segmentatie We proberen de distorsielijnen van de beeldovergangen te minimaliseren door gebruik te maken van segmentatie. De discontinuïteiten in het overlappingsgebied tussen het beeld en het canvas worden sterk verminderd als de grens tussen het beeld en het canvas niet te onderscheiden is. Een grens die valt op het pad met het laagste



Figuur 3.30: De beeldinformatie wordt geselecteerd uit het laatste beeld van de overlapping bij het vullen van het canvas.



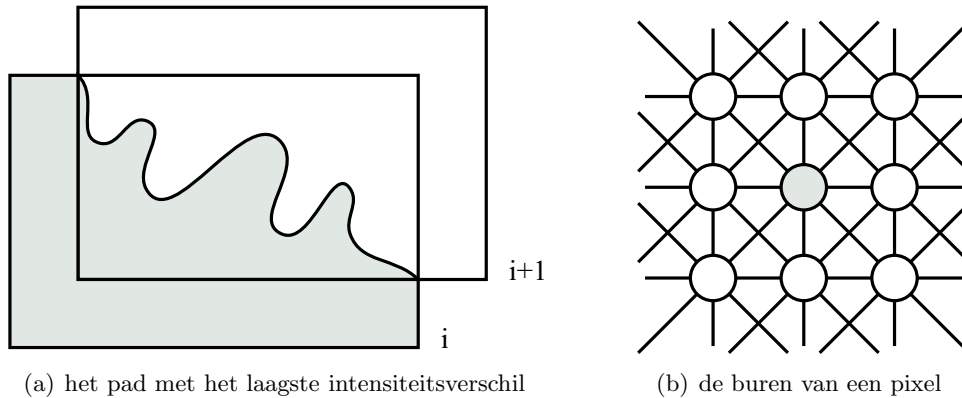
Figuur 3.31: Constructie van de Voronoi-mozaïek: de beeldinformatie wordt geselecteerd uit het middelste beeld van de overlapping bij het vullen van het canvas.



Figuur 3.32: Uitvergroting van het canvas met informatie uit het eerste beeld. De beeldovergangen zijn duidelijk zichtbaar.

intensiteitsverschil zal de minste discontinuïteiten opleveren tussen de verschillende regio's

in het panoramisch beeld. Dit pad wordt geïllustreerd in figuur 3.33. Het pad kan worden berekend met Dijkstra kortste pad algoritme. De graaf wordt opgebouwd met behulp van de volgende transformaties: we beschouwen elk pixel als knoop en de buren worden met elkaar verbonden met takken (zie figuur 3.33). Aan elke tak wordt het absolute intensiteitsverschil van de pixels toegekend als kost. Het pad loopt van de ene hoekpunt naar de tegenoverstaande hoekpunt van het overlappingsgebied zoals aangetoond op de figuur.



Figuur 3.33: Segmentatie van het canvas.

Dit pad vermijdt gebieden waar het overlappingsgebied inconsistent is, dus ook de gebieden waar er objectbeweging is. Hierdoor is het mogelijk dat kleine bewegende objecten niet op het panoramisch beeld staan. Deze techniek vraagt erger veel rekentijd: als het overlappingsgebied $X \times Y$ groot is, dan bestaat de graaf uit $X \cdot Y$ knopen en is de complexiteit van het Dijkstra-algoritme ongeveer $\mathcal{O}(8 \cdot X \cdot Y)$. Snellere (heuristische) algoritmen zijn nodig om het probleem op te lossen. De aanpassing bestaat er in bepaalde zoekrichtingen te favoriteren en enkel in de diepte te zoeken. Het resultaat is suboptimaal, maar de rekentijd is veel kleiner. Het panoramisch beeld, bekomen met de heuristische methode, wordt gegeven in figuur 3.34. Er is veel minder distorsie dan figuur 3.32.

Informatie uit meerdere beelden

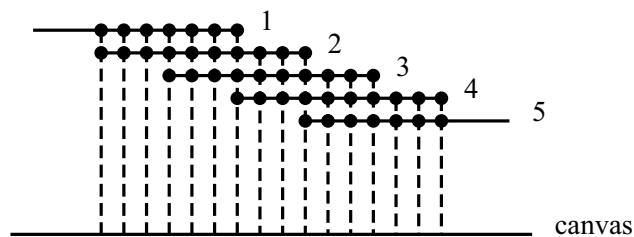
Uitmiddeling van de overlappende pixels Bij uitmiddeling is de pixelintensiteit op het canvas het gemiddelde intensiteit van alle pixels die elkaar overlappen. Het principe wordt geïllustreerd in figuur 3.35. Het panoramisch beeld wordt gegeven in figuur 3.36. Er is veel verlies van scherpheid en kwaliteit, bij objectbewegingen is het beeld wazig en krijgen we een soort spoekeffect. Een beter resultaat kan bekomen worden door een gewogen gemiddelde te nemen.

De uitmiddeling kan ook enkel toegepast worden op de beeldovergangen nadat we de pixelinformatie uit één beeld geselecteerd hebben zoals we in de vorige paragraaf beschreven hebben. Door uitmiddeling zal het beeld in sommige regio's minder scherp zijn, maar de beeldovergangen zijn wel vloeiender.

Selectie op basis van de mediaan De intensiteit van een pixel op het canvas is de mediaan van alle overlappende pixels. Het eindresultaat wordt gegeven in figuur 3.37. Net zoals bij de uitmiddeling is het beeld niet scherp en van lage kwaliteit. Het panoramisch



Figuur 3.34: Uitvergroting van het canvas bekomen na segmentatie.



Figuur 3.35: De pixels van het canvas is het gemiddelde van alle overlappende pixels.

beeld vertoont een soort schilderseffect. Een groot nadeel is dat het canvas enkel kan gevuld worden als alle registraties gebeurd zijn. Dit impliceert dat het b.v. niet mogelijk is om globale registratie te gebruiken.

3.4 Spatiaal-temporele volumes

Een spatiaal-temporeel volume is een projectie van een 3-dimensionale voorstelling van een shot op een vlak. Meestal wordt het volume geconstrueerd door het eerste beeld van de shot te nemen en deze uit te breiden met de horizontale en verticale spatiaal-temporele beelden (de zijvlakken). In figuur 3.38 wordt een voorbeeld getoond van een spatiaal-temporeel volume.

Er zijn een paar voordelen verbonden aan spatiaal-temporele volumes: de lengte van de shot is zichtbaar voor de gebruiker, de gebruiker kan zien of de shot veel (camera)beweging bevat, enz. Spatiaal-temporele volumes zijn visueel attractief voor de gebruiker, maar



Figuur 3.36: Uitvergroting van het canvas bekomen na uitmiddeling van alle overlappende pixels.



Figuur 3.37: Uitvergroting van het canvas bekomen na mediaanselectie van alle overlappende pixels.

hebben geen nut voor de computer.



Figuur 3.38: Een constructie van een spatioaal-temporeel volume.

3.5 Belangrijkheid van een shot of scène

Bij elke samenvatting moeten we de belangrijkste onderdelen accentueren, ook bij de samenvattingen van videosequenties. We geven aan elke shot of scène een belangrijkheidsfactor. Om de belangrijke shots te accentueren, vergroten we hun sleutelbeelden. Voor het omgekeerde doen we hetzelfde: de onbelangrijke shots worden verwijderd of worden in de samenvatting voorgesteld door een kleinere sleutelbeeld.

De belangrijkheidsfactor van een shot hangt van 2 zaken af: 1. de duur of lengte van een shot en 2. de repetitiviteit of de uniciteit van een shot [UFGB99, UF99]. Het spreekt voor zich dat shots minder belangrijk zijn als ze kort zijn of als ze veel voorkomen (b.v. in een dialoog).

Om de belangrijkheidsfactor van een shot te berekenen, moeten we eerst het gewicht voor het cluster (of scène) i berekenen:

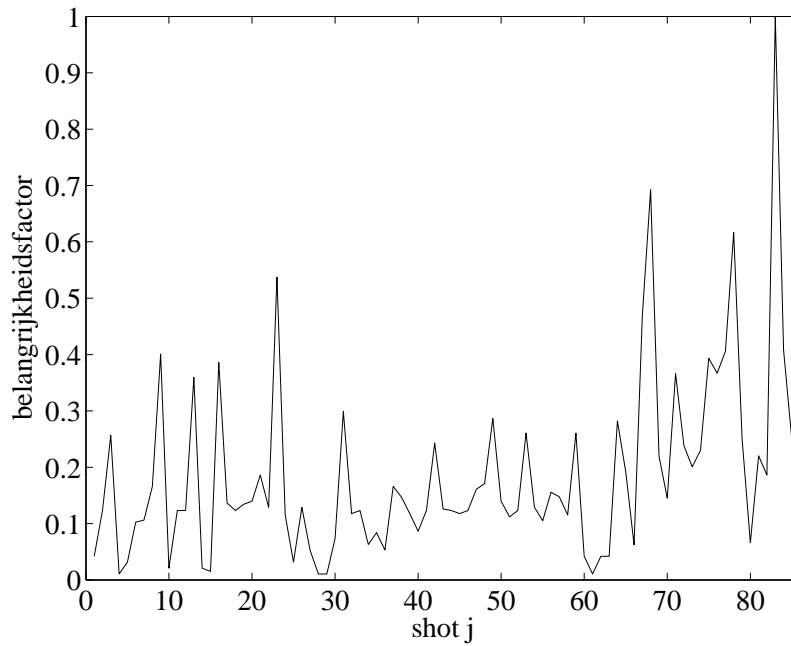
$$W_i = \frac{S_i}{\sum_{j=1}^c S_j} \quad (3.26)$$

met c het totaal aantal clusters en S_i de totale som van de lengtes van de shots van scène i .

De belangrijkheidsfactor I van shot j van scène k wordt dan gegeven door:

$$I_j = L_j \log \frac{1}{W_k} \quad (3.27)$$

met L_j de lengte van shot j . Voor scène i is de belangrijkheidsfactor S_i , we moeten dus enkel rekening houden met de duur. De genormaliseerde belangrijkheidsfactor wordt uitgezet voor de verschillende shots van een videosequentie in figuur 3.39.



Figuur 3.39: De genormaliseerde belangrijkheidsfactor voor de verschillende shots.

Nu we de belangrijkheidsfactor hebben berekend kunnen we de sleutelbeelden schalen: als $I_j \leq \delta$, houden we geen rekening met de shot. Anders als $I_j > \delta$, nemen we de shot op in onze samenvatting. Een goede keuze voor δ is $\frac{1}{8} \max(I_j)$, dan is δ aangepast aan de inhoud.

Grote sleutelbeelden trekt de aandacht van de gebruiker op belangrijke shots of scènes. Als de belangrijkheidsfactor b.v. in $[\frac{1}{8} \max(I_j), \frac{1}{4} \max(I_j)]$, $[\frac{1}{4} \max(I_j), \frac{1}{2} \max(I_j)]$ of $[\frac{1}{2} \max(I_j), \max(I_j)]$ ligt, dan krijgt het sleutelbeeld respectievelijk de kleinste, middelmatige of grootste dimensie toegewezen.

Een uitbreiding van de belangrijkheidsfactor wordt gegeven door:

$$I_j = L_j \log \frac{\sum_t A_t p_t(j)}{W_k} \quad (3.28)$$

waarbij A_t een voorgedefinieerde versterkingsfactor is voor categorie t en $p_t(j)$ is de probabilmiteit dat een shot j tot categorie t behoort.

We kunnen de belangrijkheidsfactor I_j ook nog van andere kenmerken van een shot laten afhangen: hoeveelheid beweging, kleuren, ...

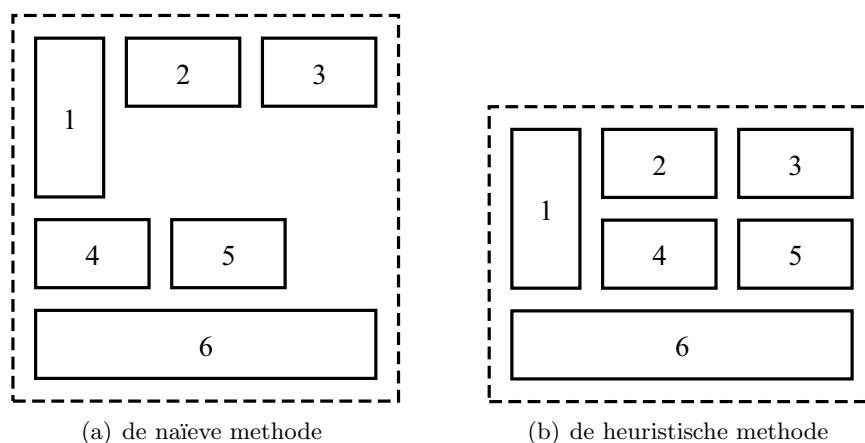
3.6 Compacte voorstelling van sleutelbeelden

In deze sectie bespreken we hoe een samenvatting aan de hand van sleutelbeelden visueel attractief kan gemaakt worden voor de gebruiker. Eens de sleutelbeelden geselecteerd zijn, worden ze in een n -dimensionale ruimte geschikt om een visuele samenvatting te vormen van de videosequentie. Hier bespreken we enkele methoden om beelden in een 2-

dimensionale regio te plaatsen op een ruimtelijk efficiënte manier. De samenvatting wordt voorgesteld in een „stripboek”-stijl, waardoor ze geschikt is om op papier uit te printen of op een webpagina te tonen.

De sleutelbeelden hebben verschillende dimensies (zie paragraaf 3.5) waardoor het iets moeilijker is om ze efficiënt te plaatsen. Er bestaan tal van heuristische en optimalisatie-technieken die suboptimale of optimale oplossingen bieden voor het probleem. Een voorbeeld van een naïeve methode wordt geïllustreerd in figuur 3.40(a). De sleutelbeelden worden naast elkaar geplaatst. Als het sleutelbeeld niet meer in de rij past, dan worden de beelden in een volgende rij geplaatst.

Optimalisatietechnieken, zoals dynamisch programmeren, garanderen de optimale oplossing. Figuur 3.40(b) toont een optimale oplossing. Maar deze technieken zijn niet heel flexibel en de indeling hoeft niet geoptimaliseerd te worden voor de volledige videosequentie. Daarom stellen we een paar technieken voor die gemakkelijker en flexibeler zijn dan de optimalisatietechnieken en die een betere indeling geven dan de naïeve methode.



Figuur 3.40: Verschillende geproduceerde indelingen.

3.6.1 Exhaustieve blokmethode

Bij de exhaustieve blokmethode wordt de volledige ruimte optimaal verdeeld in blokrijen of subgebieden. Per iteratie wordt één blokrij gevuld. Het proces eindigt wanneer alle beelden geplaatst zijn. Elke blokrij wordt gerasterd, waarbij de rastereenheid even groot is als het kleinste sleutelbeeld. Voor het algoritme worden de volgende veronderstellingen gemaakt:

- de dimensies van de sleutelbeelden f_1, f_2, \dots, f_N zijn veelvoud van het kleinste sleutelbeeld.
- de toegelaten blokhoogten zijn $\{r_1, r_2, \dots, r_M\}$ (van de kleinste tot de grootste hoogte van alle sleutelbeelden) en de blokbreedte is een vaste waarde W .
- een kostfunctie $c(x, y)$ wordt opgesteld om een beeld met grootte x in de overblijvende ruimte van grootte y te passen. Een typische kostfunctie is het verschil tussen x en y , maar eender welke andere kostfunctie kan ook gebruikt worden.

- een vulregel wordt gegeven: b.v. van boven naar onder en van links naar rechts.

Het algoritme om één blokrij te vullen, bestaat uit de volgende stappen:

1. initialiseer door het startbeeld door $s = 1$ te stellen.
2. zoek de optimale vulling voor een blokrij met hoogte r en met n sleutelbeelden:
 - (a) laat de blokhoogte r per iteratie een andere waarde aannemen uit een reeks $\{r_1, r_2, \dots, r_M\}$.
 - (b) zoek alle sequenties $\{q_1, q_2, \dots, q_L\}$ die in de blokrij passen. De sequentie q_{n_i} bevat alle sleutelbeelden tussen s en $s + n_i$. Ga terug naar stap (a) als er geen sequenties zijn (b.v. als het sleutelbeeld groter is dan de blokhoogte).
 - (c) we berekenen de gewichtsfactor w_i voor elke sequentie q_i :

$$w_i = \frac{1}{n_i} \sum_{j=1}^{n_i} c(f_{s+j-1}, \lambda_{ij}) + \tau_i$$

$$\lambda_{ij} = \begin{cases} W \cdot r, & \text{als } j = 1 \\ W \cdot r - \sum_{k=1}^{j-1} f_{s+k-1}, & \text{anders} \end{cases} \quad (3.29)$$

waarbij τ_i een additionele gewichtsfactor is bepaald door de overblijvende witte ruimte; λ_{ij} is de overblijvende ruimte voor het j -de element van sequentie i .

- (d) herhaal stappen (a), (b) en (c) voor een andere blokhoogte r tot de optimale sequentie q_{i^*} voor de blokrij gevonden wordt:

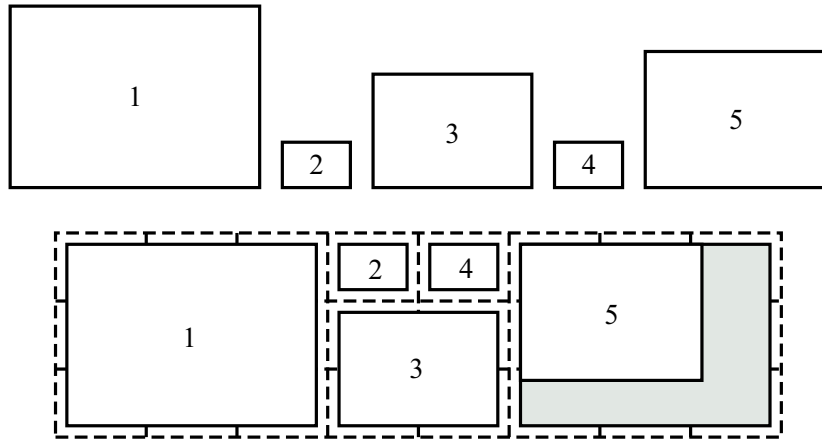
$$i^* = \arg \min_i (w_i) \quad (3.30)$$

n wordt dan gelijk gesteld aan n_{i^*} .

3. vermeerder s met n .
4. herhaal stap 2 tot alle sleutelbeelden geplaatst zijn, als $s = N$.

Figuur 3.41 illustreert de procedure voor het plaatsen van de sleutelbeelden in een blokrij. De grootte van sleutelbeeld 5 wordt groter gemaakt om de witte ruimte te minimaliseren.

Er zijn enkele nadelen verbonden aan deze techniek: de oplossing is suboptimaal, de dimensies van de sleutelbeelden kunnen slechts uit veelvouden bestaan van het kleinste sleutelbeeld, in een blokrij kan de volgorde van de sleutelbeelden omgekeerd zijn, ... Daarom wordt een ander heuristische methode opgesteld.



Figuur 3.41: De sleutelbeelden worden in een blokrij ($r = 3$ en $W = 8$) geplaatst.

3.6.2 Heuristische plaatsingsmethode

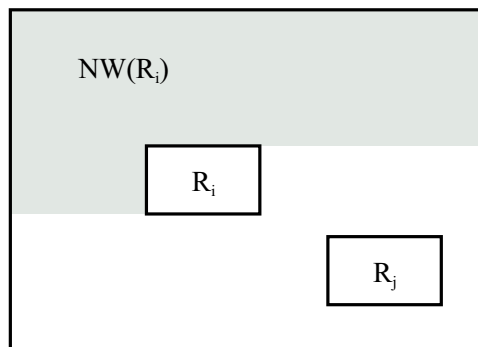
We stellen nu een heuristische methode op die aan de volgende voorwaarden beantwoordt:

- de volgorde wordt gerespecteerd:

$$NW(R_i) \cap R_j = \emptyset, \quad i < j \quad (3.31)$$

waarbij $NW(R_i)$ het noordwestelijk gebied is van R_i zoals aangetoond in figuur 3.42.

- de dimensies van de sleutelbeelden zijn willekeurig (zo kunnen we ook panoramische beelden in de visueel compacte samenvatting integreren).
- de methode moet snel berekenbaar zijn.



Figuur 3.42: De volgorde wordt gerespecteerd: de gebieden $NW(R_i)$ en R_j mogen geen overlapping vertonen.

De heuristische methode verloopt op de volgende wijze:

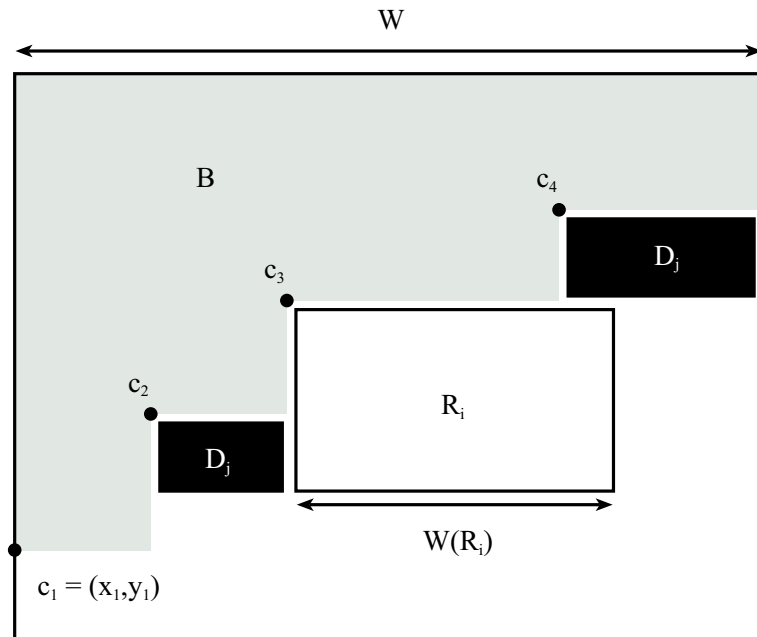
1. initialiseer het bezette gebied $B = \emptyset$, $(x_1, y_1) = (0, 0)$ en $N_{stap} = 1$. Het bezette gebied B heeft een soort trapvorm zoals aangetoond in figuur 3.43 en kan dus gekenmerkt worden door de staphoeken $c_j = (x_j, y_j)$, $j = 1, \dots, N_{stap}$.
2. herhaal voor alle sleutelbeelden R_i , $i = 1, \dots, N$:

- (a) bereken voor elke staphoek c_j de dode ruimte D_j zoals in figuur 3.43. De dode ruimte D_j bestaat uit twee regio's, waar geen nakomende beelden kunnen geplaatst worden ten gevolge van voorwaarde (3.31). Als er geen genoeg ruimte is om R_i te plaatsen (als $W(R_i) > W - x_j$), dan is $D_j = \infty$, waarbij $W(R_i)$ de breedte is van sleutelbeeld R_i en W de totale breedte is van het canvas.
- (b) zoek de plaatsing met de minimale dode ruimte D_{j^*} waarvoor het volgende geldt:

$$j^* = \arg \min_j |D_j| \quad (3.32)$$

met $|D_j|$ de oppervlakte van de dode ruimte D_j .

- (c) plaats R_i aan de j -de hoek c_j .
- (d) herbereken het bezette gebied $B = B \cup R_i \cup D_{j^*}$ en de corresponderende staphoeken $c_j = (x_j, y_j)$ en N_{stap} .



Figuur 3.43: Plaatsing van het i -de beeld in de j -de staphoek $c_j = (x_j, y_j)$ van de bezette regio B . Het bezette gebied B wordt gekenmerkt door de staphoeken c_j .

De performantie λ van de plaatsing wordt berekend door de verhouding van de hoogte H_h (bekomen door de heuristische methode) op de hoogte H_n (bekomen door de naïeve

methode) te berekenen. In [TAT97] wordt aangetoond dat λ afhankelijk is van de verhouding ρ van de vensterbreedte op de gemiddelde beeldbreedte. Gemiddeld is $\lambda = 0.9$ met $\rho = 10$. Er wordt enkel veel winst gemaakt als er veel variërende beelddimensies zijn (b.v. afkomstig van panoramische beelden).

3.7 Interactieve webpagina

De samenvatting van een videosequentie is een set van relevante sleutelbeelden van de scènes. Deze sleutelbeelden kunnen opgeslagen worden in een database en vormen zo de basis voor de indexering van het inhoudsgebaseerd zoekstelsel. Webpagina's kunnen de sleutelbeelden ophalen en aan de gebruiker presenteren (b.v. met behulp van SQL² met PHP³ of Microsoft Access met ASP⁴, enz.).

De grote voordelen van webpagina's zijn de interactieve gebruiksvriendelijkheid en de mogelijkheid tot het hiërarchisch doorzoeken van shots. Het hiërarchisch doorzoeken is te verkiezen boven sequentieel doorzoeken, het bespaart de gebruiker immers veel tijd. De gebruiker daalt af in hiërarchie (b.v. van scènes naar shots) door op een sleutelbeeld te klikken. Klikte de gebruiker op de gewenste shot, dan wordt het gewenste stuk van de videosequentie afgespeeld.

Er zijn tal van mogelijkheden om de samenvatting visueel attractief te maken voor de gebruiker met betrekking tot de webpagina:

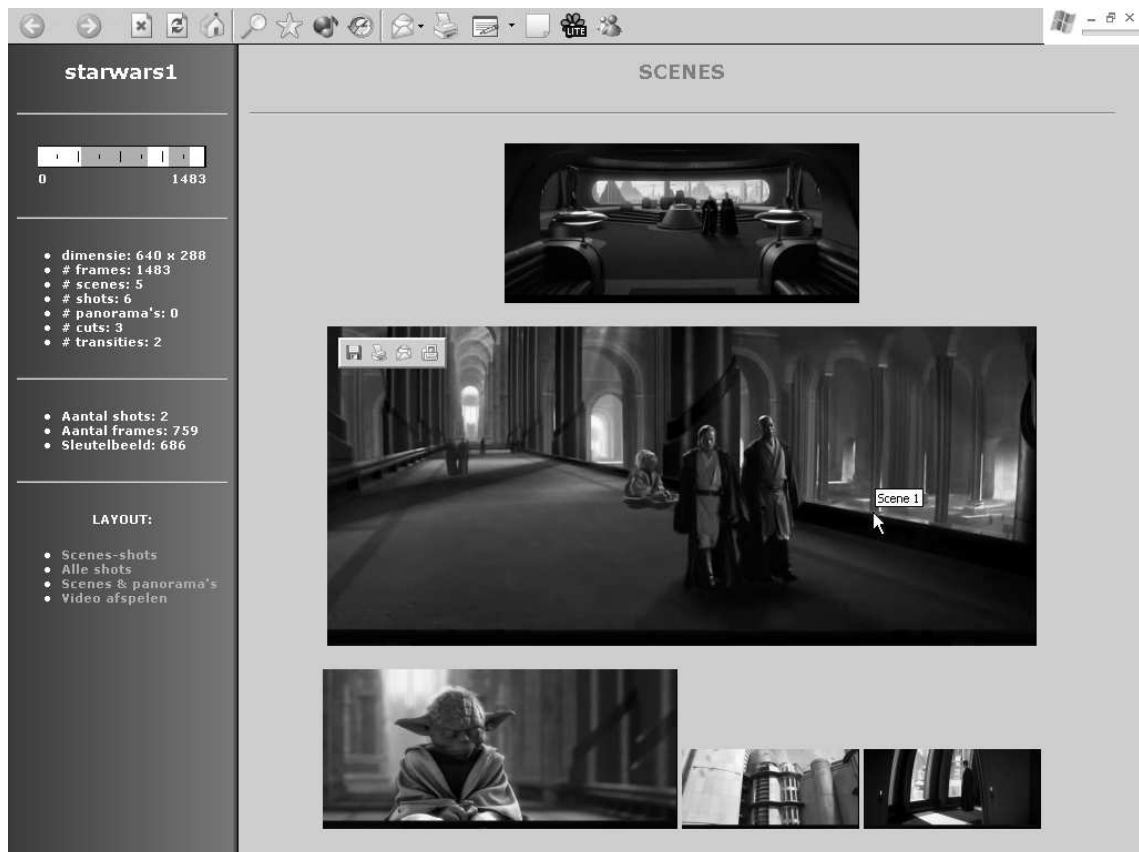
- een tijdslijn die gesynchroniseerd is met de sleutelbeelden als de gebruiker met de muis over zo'n sleutelbeeld gaat. Zo weet de gebruiker steeds waar de scènes of shots gesitueerd zijn in de videosequentie. Omgekeerd kan de gebruiker op een willekeurige positie van de tijdsbalk klikken en zal de webpagina de sleutelbeelden van de shots tonen.
- een venster met extra informatie over de scènes of shots zal verschijnen indien de gebruiker met de muis over een sleutelbeeld gaat. De extra informatie kan van syntactische aard zijn, b.v. de lengte, de kleurinformatie, de positie in de videosequentie, enz. Maar de informatie kan ook van semantische aard zijn, b.v. de titel, het bijschrift, de inhoudsbeschrijving, de namen van eventuele personen of objecten die in de shot voorkomen, ... De semantische informatie wordt manueel ingevoerd of kan door objectherkenning, gezichtsherkenning, stemherkenning, tekstherkenning, ... automatisch ingevuld worden.
- visuele iconen of watermerken kunnen de samenvatting nog aantrekkelijker maken. Deze iconen kunnen zowel op als naast het sleutelbeeld geplaatst worden en kunnen meer informatie geven over de shots of de overgangen zelf. Voor de overgangen kunnen b.v. iconen gebruikt worden om het type van transitie (wipe, dissolve, ...) aan te duiden. Voor de shots kunnen iconen b.v. het genre aanduiden.

Er zijn tal van andere uitbreidingen mogelijk.

²Structured Query Language

³PHP: Hypertext Preprocessor

⁴Active Server Pages



Figuur 3.44: Een webpagina met de visuele samenvatting van een videosequentie.

Hoofdstuk 4

Experimentele resultaten

4.1 Inleiding

We maken in dit hoofdstuk een vergelijkende studie tussen de verschillende besproken algoritmen. Daarvoor wordt een selecte verzameling aan videosequenties verzameld. Op deze testset passen we vervolgens de verschillende technieken op toe en bepalen we de performantiematen voor elk algoritme. We kunnen dan de verschillende algoritmen vergelijken aan de hand van deze performantiematen.

We beginnen dit hoofdstuk met een omschrijving van de gebruikte omgeving en testmateriaal. Vervolgens bespreken we de resultaten van de experimenten die we gedaan hebben met temporele videosegmentatie en scèneclustering. Daarna onderzoeken we de registratie voor panoramische beelden gegeven en bespreken we de kwaliteit van panoramische beelden.

4.2 Omgeving

De programmeertaal is (ANSI) C/C++. De programmeeromgeving is Dev-C++ (versie 4.9.8.0 op <http://www.bloodshed.net>) en de gebruikte compiler is gcc 2.9.5.

4.2.1 Bibliotheken

De algoritmen werden in een C/C++ bibliotheek geïmplementeerd en daarnaast werd er ook gebruik gemaakt van enkele bestaande bibliotheken:

- om video in te lezen wordt er gebruik gemaakt van een MPEG-bibliotheek¹ (versie 0.3.1 op <http://libmpeg2.sourceforge.net/>). De bibliotheek ondersteunt MPEG-1 en MPEG-2, wel enkel video en geen systemen (dus zonder audio). Via een interface wordt de video in blokken van N aantal beelden gedecodeerd naar een RGB-formaat om hierop verder bewerkingen op uit te voeren. De bibliotheek staat gescheiden in de module-directorie `/libmpeg`.

¹De oorspronkelijk gebruikte bibliotheek met ongecomprimeerde AVI's (met PPM-gecodeerde beelden) werd vervangen door de MPEG-bibliotheek wegens beperking van opslagruimte en geheugen.

- er wordt ook gebruik gemaakt van een bestaande Kanade-Lucas-Tomasi-tracker implementatie (versie 1.1.5 op <http://robotics.stanford.edu/~birch/klt/>). De bibliotheek berekent de KLT-tracking informatie zoals beschreven in paragraaf 3.3.1. De bibliotheek staat gescheiden in de module-directorie `/libklt`.
- browsers ondersteunen niet elk beeldformaat, en ook geen PPM (Portable PixelMap formaat). Daarom wordt er gebruik gemaakt van een bibliotheek (versie 0.3.4 op <http://pngwriter.sourceforge.net/>) die de RGB-beelden kan exporteren naar PNG (Portable Network Graphics). De bibliotheek maakt ook gebruik van twee andere bibliotheken: `libpng` (versie 1.2.5 op <http://www.libpng.org/>) en `zlib` (versie 1.1.4 op <http://www.gzip.org/zlib/>). De bibliotheek staat gescheiden in de module-directorie `/libpng`.

4.2.2 Testmateriaal

De gebruikte videosequenties worden naar MPEG-2 formaat² geconverteerd met behulp van Cinema Craft Encoder. De keuze om zelf geen artificiële overgangen te maken (b.v. in Adobe Premiere) wordt bewust gedaan om de algoritmen te testen in reële situaties. Er worden 17 videosequenties geselecteerd voor de testset, goed voor ongeveer 1485 seconden. In tabel 4.1 staan de gegevens van de gebruikte testset.

De videosequenties zijn afkomstig van de volgende bronnen:

- Austin Powers 3 (`austin`): de videosequentie bevat actiescènes met o.a. explosies.
- Bowling For Columbine (`bowling`): de videosequentie is van het genre documentaire.
- Ice Age (`iceage`): de videosequentie is van het genre animatiefilm.
- Monty Python and The Holy Grail (`monty0`, `monty1`): de videosequenties bevatten fade-ins en fade-outs.
- Worldchampionship Snooker 2003 (`snooker0`, `snooker1`): de videosequentie is van het genre sport.
- Star Wars 2 (`starwars0`, `...`, `starwars9`): de videosequenties bevatten wipes.

4.3 Temporele videosegmentatie en scèneclustering

4.3.1 Performantiematen

De performantie van shotdetectie kan gekenmerkt worden door drie maten: het aantal shotgrenzen die correct gedetecteerd zijn $\kappa(C)$, het aantal shotgrenzen die gemist worden bij de detectie $\kappa(M)$ en het aantal vals gedetecteerde overgangen $\kappa(V)$. Het verband tussen $\kappa(C)$ en $\kappa(M)$ ligt vast: $\kappa(C) = \# \text{ overgangen} - \kappa(M)$. Een techniek heeft een goede performantie als ze een hoge $\kappa(C)$ en een lage $\kappa(V)$ heeft.

Om de verschillende technieken met elkaar te kunnen vergelijken, definiëren we de volgende performantiematen:

²geëncodeerd in 3-pass VBR (variabele bitrate) aan gemiddeld 3000 kbits/sec: minimaal 1000 kbits/sec en maximaal 9000 kbits/sec. Deze parameters beperken het verlies aan kwaliteit voor de gebruikte videosequenties.

Tabel 4.1: De testset.

	dimensie	# beelden	# cuts	# transities	# shots
austin	608×256	4077	88	0	89
bowling	576×304	2533	18	0	19
iceage	576×304	4595	65	11	77
monty0	560×304	5562	76	1	78
monty1	560×304	3856	45	2	48
snooker0	640×480	4688	11	13	25
snooker1	640×480	4337	57	0	58
starwars0	640×272	328	2	1	4
starwars1	640×272	1483	3	2	6
starwars2	640×272	1816	14	1	16
starwars3	640×272	300	0	1	2
starwars4	640×272	471	1	1	3
starwars5	640×272	231	0	1	2
starwars6	640×272	242	1	1	3
starwars7	640×272	210	0	1	2
starwars8	640×272	588	1	1	3
starwars9	640×272	300	0	1	2
totaal	-	35617	382	38	437

- sensitiviteit of doeltreffendheid: het percentage van het aantal teruggevonden overgangen. De verhouding wordt gegeven door:

$$sensitiviteit = \frac{\kappa(C)}{\kappa(C) + \kappa(M)} \quad (4.1)$$

- precisie of doelmatigheid: het percentage van het aantal relevante overgangen. De verhouding wordt gegeven door:

$$precisie = \frac{\kappa(C)}{\kappa(C) + \kappa(V)} \quad (4.2)$$

Een techniek heeft een hogere performantie dan een andere techniek als ze een hogere sensitiviteit en precisie heeft.

Andere performantiematen kunnen ook gebruikt worden, zoals *fall-out* (dit is de verhouding van $\kappa(V)$ op het totaal aantal beelden zonder de relevante overgangen), *E*-maat (om sensitiviteit of precisie te accentueren), enz [OSMM99].

4.3.2 Performantie: detectie resultaten

Cutdetectie

We maken een vergelijkende studie van de volgende algoritmen voor cutdetectie:

1. paarsgewijze pixelvergelijking op basis van kleuren (zie vergelijking 2.2).
2. paarsgewijze pixelvergelijking met gebruik van een 5×5 gemiddeldewaardefilter (zie vergelijking 2.5).
3. globale histogramvergelijking op basis van kleuren (zie vergelijking 2.9).
4. spatiaal-temporele snede-analyse met spatiale reductie van hoogte en breedte tot 6.25% (zie paragraaf 2.2.7). Deze grote spatiale reductie zal resulteren in een slechtere performantie, maar zal daarentegen een acceptabele rekentijd nodig hebben.
5. zelf-similariteitsanalyse met 2×2 convolutiekern (zie paragraaf 2.2.8).
6. paarsgewijze vergelijking van DCT-coëfficiënten op alle beelden (zie paragraaf 2.3.2). We onderzoeken hiermee het gedrag van DCT-coëfficiënten in functie van temporele segmentatie.
7. cutdetectie op basis van de histogrammen van de DC-beelden (zie paragraaf 2.3.3). In [PS97] worden het globaal intensiteitshistogram en de rij- en kolomhistogrammen van DC-beelden berekend. De rij- en kolomhistogrammen, respectievelijk X_i en Y_j , van een beeld met $M \times N$ DCT-blokken worden gedefinieerd als:

$$\begin{aligned}
 X_i &= \frac{1}{M} \sum_{j=1}^M c_{0,0}(i, j) \\
 Y_j &= \frac{1}{N} \sum_{i=1}^N c_{0,0}(i, j)
 \end{aligned}
 \tag{4.3}$$

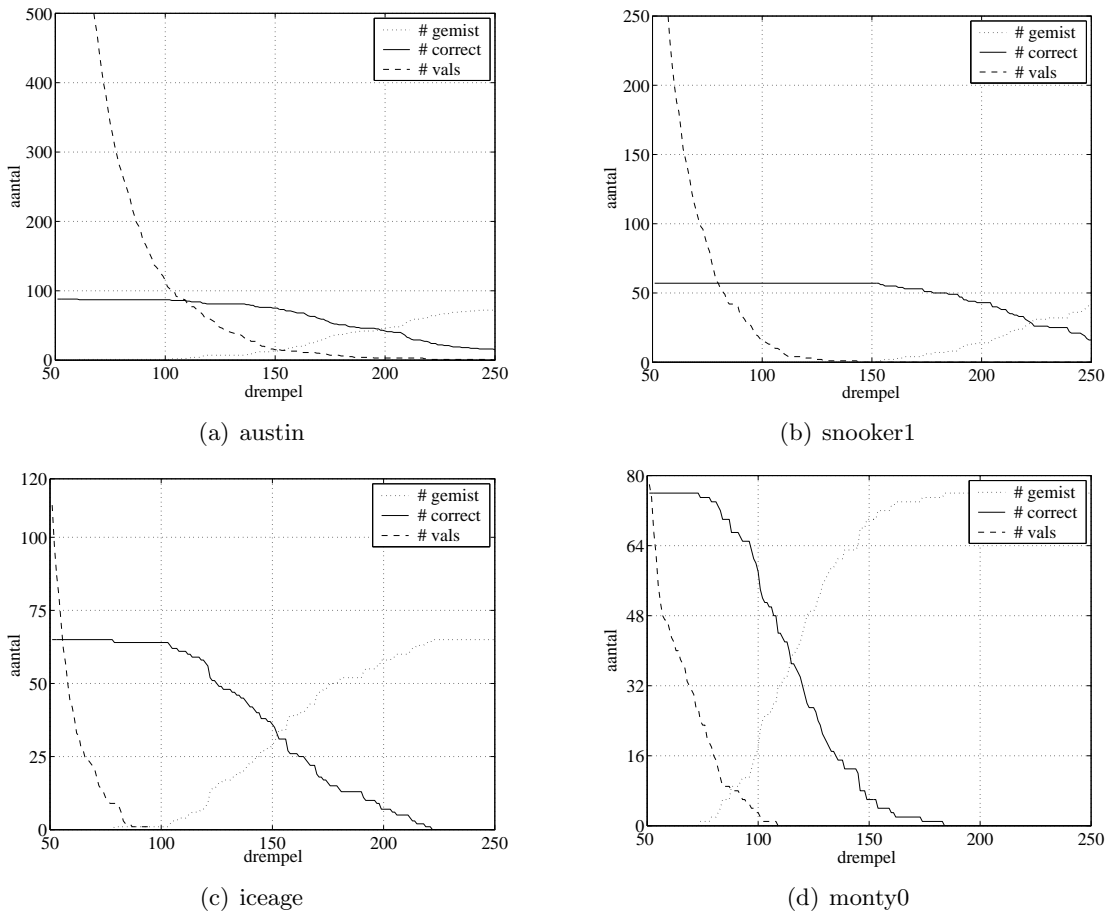
met $c_{0,0}(i, j)$ de DC-component van blok (i, j) . Het intensiteitshistogram representeert de globale informatie van het beeld, terwijl de rij- en kolomhistogrammen lokale informatie bevatten. Deze drie histogrammen worden gaussiaans uitgemiddeld om de performantie te verbeteren. Een χ^2 -test wordt gebruikt om de histogramverschillen te berekenen. De χ^2 -test levert ons drie coëfficiënten op en voor elk van deze drie coëfficiënten berekenen we de significante probabilmiteit:

$$P(\chi^2|\nu) = \frac{1}{2^{\frac{\nu}{2}} \Gamma(\frac{\nu}{2})} \int_{t=\chi^2}^{\infty} t^{\frac{\nu}{2}-1} e^{-\frac{t}{2}} dt
 \tag{4.4}$$

waarbij ν het aantal vrijheidsgraden voorstelt. De probabiliteiten worden vergeleken met een drempel δ om cuts te detecteren.

8. cutdetectie op basis van bewegingsvectoren (zie paragraaf 2.3.5).

We tonen eerst aan dat elke videosequentie *anders* is, dit wil zeggen dat alle technieken telkens andere resultaten geeft voor elke videosequentie. Dit betekent dat b.v. de paarsgewijze pixelvergelijking heel goede resultaten oplevert voor de videosequentie *austin*, maar niet voor *iceage*. Om dit te illustreren worden $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$ van enkele videosequenties uitgezet in figuur 4.1 voor de verschillende globale drempels van de

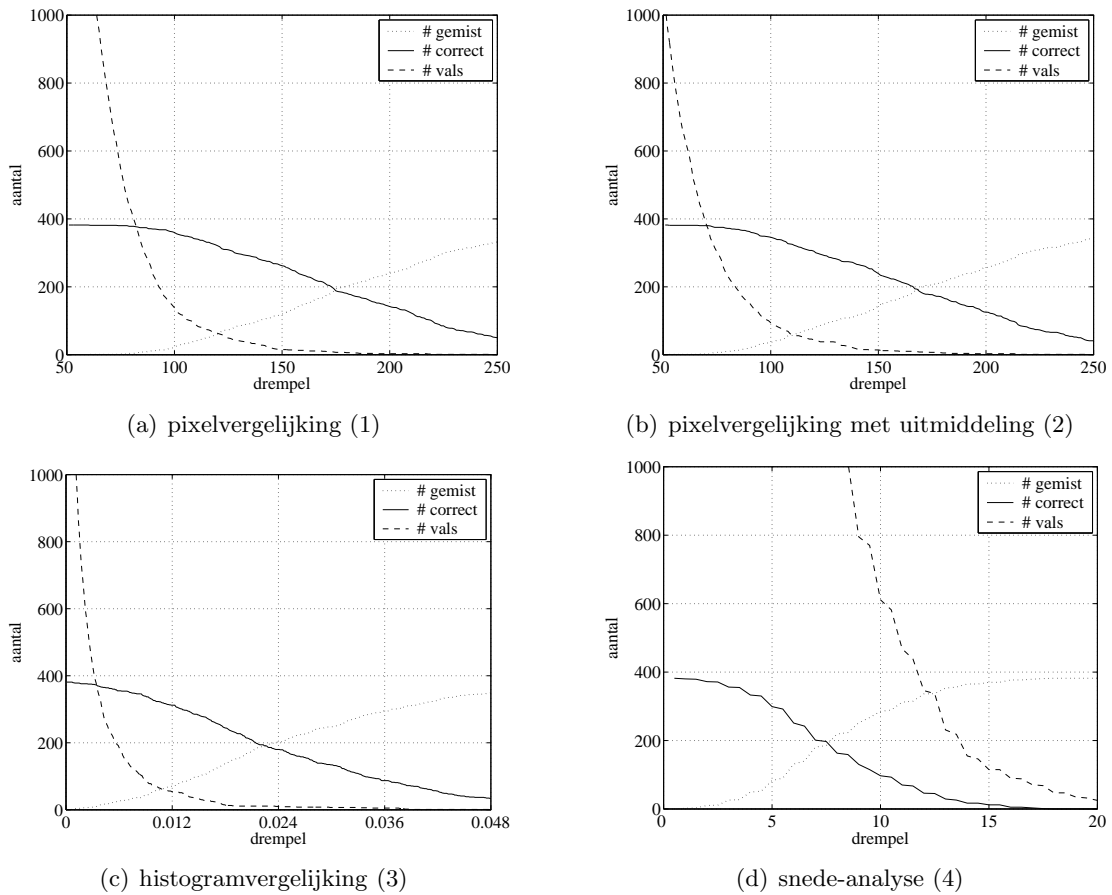


Figuur 4.1: De curven $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$ uitgezet in functie van de globale drempelwaarde. Elke videosequentie gedraagt zich anders op de drempelwaarden.

paarsgewijze pixelvergelijking (1).

De curven $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$ worden nu in functie van de globale drempel uitgezet voor de acht technieken in figuren 4.2 en 4.3. Deze figuren bieden ons reeds veel informatie aan. Het verloop van de drie curven zijn bij alle technieken ongeveer hetzelfde (bij DC-histogramvergelijking (7) zijn de curven trapvormig wegens quantizatie in de implementatie). Op de grafieken kunnen we ook reeds de precisie afleiden: hoe meer de curve $\kappa(V)$ ($\#$ vals) rechts ligt, hoe lager de precisie. Daarom kunnen we vaststellen dat snede-analyse (4), DCT-vergelijking (6), DC-histogramvergelijking (7) en bewegingsvectoren-analyse (8) een lage precisie hebben.

In figuren 4.5 en 4.6 tonen de grafieken van de precisie in functie van de sensitiviteit. We hebben geöpteerd voor deze type grafieken, omdat twee grafieken van sensitiviteit en precisie in functie van de globale drempel ons niet veel meer informatie geven dan de de curven $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$. Als werkingsgebied definiëren we het gebied waarin de techniek een hoge precisie en hoge sensitiviteit heeft. Het normale werkingsgebied wordt aangegeven in figuur 4.4.



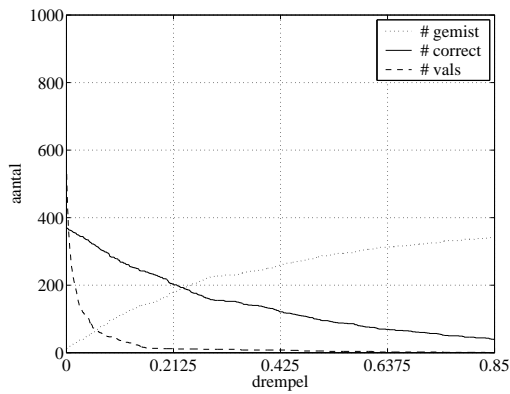
Figuur 4.2: De curven $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$ uitgezet in functie van de globale drempelwaarde voor de verschillende technieken (deel 1).

Wat ons opvalt is de slechte performantie van snede-analyse (4), DCT-vergelijking (6) en bewegingsvectoren-analyse (8). Voor snede-analyse kunnen we de performantie opvoeren, maar hierdoor zal de rekestijd sterk stijgen. De bijzondere vorm van de curve is te verklaren doordat de curve $\kappa(V)$ zeer hoge waarden aanneemt. Omdat de DCT-vergelijking en de bewegingsvectoren-analyse zo slecht presteren, is het een noodzaak dat bij temporele segmentatie van gecomprieeerde video verschillende technieken te combineren om de performantie te verhogen.

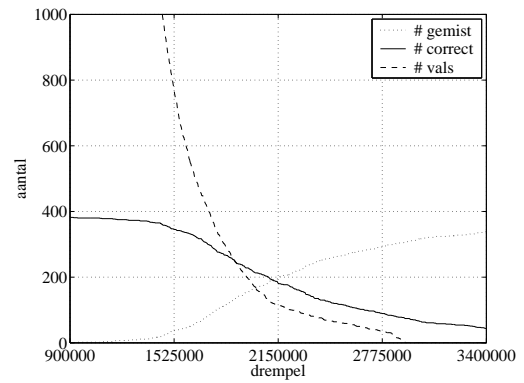
Pixelvergelijking met uitmiddeling (2) heeft de beste performantie, maar het verschil met pixelvergelijking (1), histogramvergelijking (3) en zelf-similariteitsanalyse (5) is heel klein. De optimale drempelwaarde hangt meestal af van wat we willen bereiken: een hoge sensitiviteit of een hoge precisie of beide. Meestal ligt de optimale drempelwaarde in het werkingsgebied.

Automatische drempelselectie met piekdetectie In bijlage A zijn er tal van mogelijkheden om de drempelkeuze te automatiseren en verbeteren, we onderzoeken nu wat de invloed van piekdetectie is. Dit doen we aan de hand van pixelvergelijking met uitmiddeling (1) en histogramvergelijking (3).

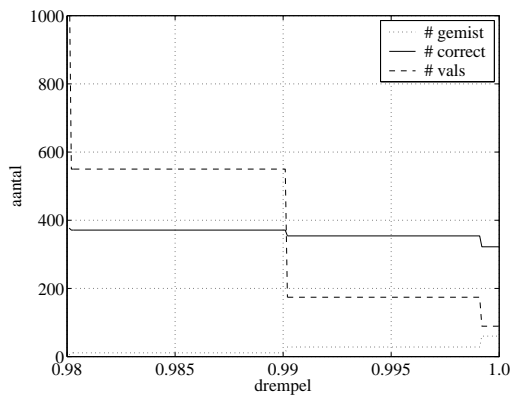
De precisie is uitgezet in functie van de sensitiviteit in figuur 4.7 voor verschillende



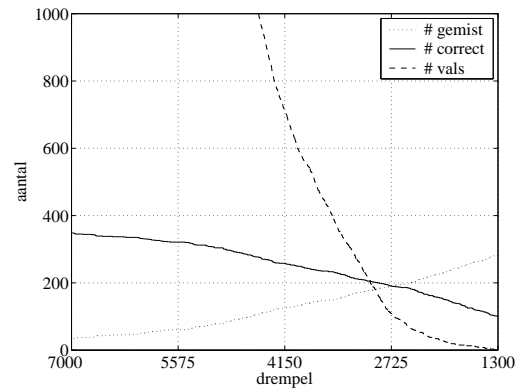
(a) zelf-similariteitsanalyse (5)



(b) DCT-vergelijking (6)

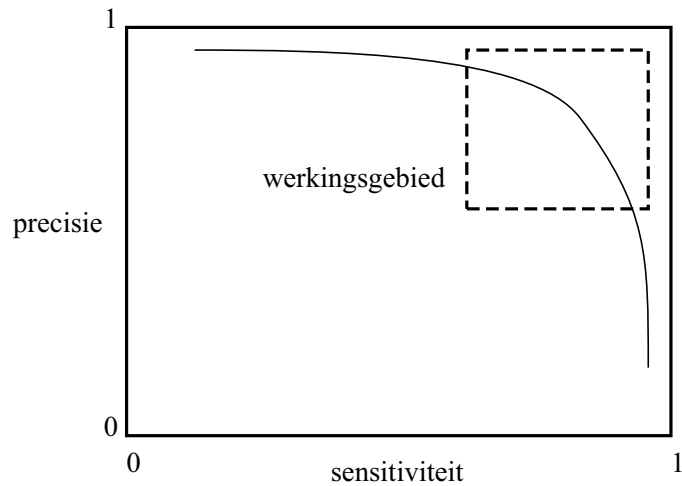


(c) DC-histogramvergelijking (7)

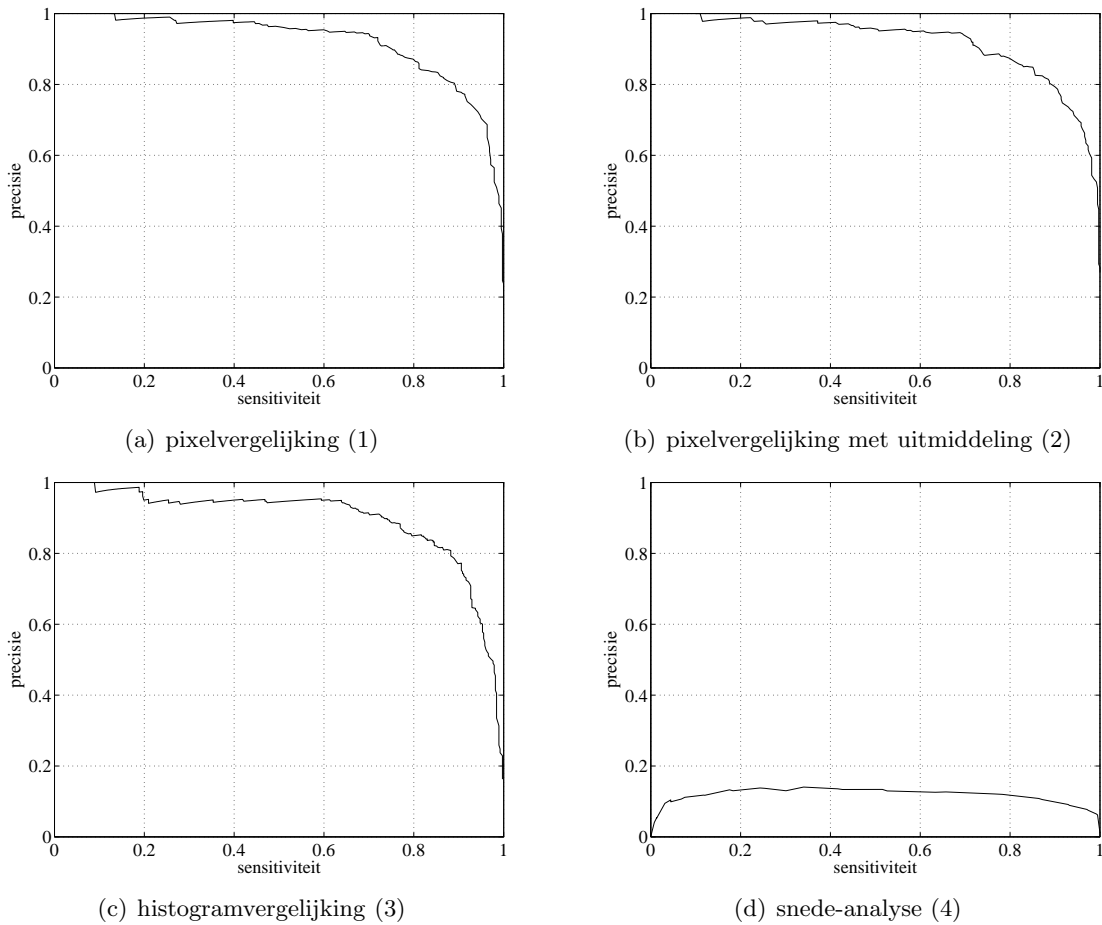


(d) bewegingsvectoren-analyse (8)

Figuur 4.3: De curven $\kappa(C)$, $\kappa(M)$ en $\kappa(V)$ uitgezet in functie van de globale drempelwaarde voor de verschillende technieken (deel 2).



Figuur 4.4: Het werkingsgebied in functie van sensitiviteit en precisie.



Figuur 4.5: De precisie uitgezet in functie van de sensitiviteit voor de verschillende technieken (deel 1).

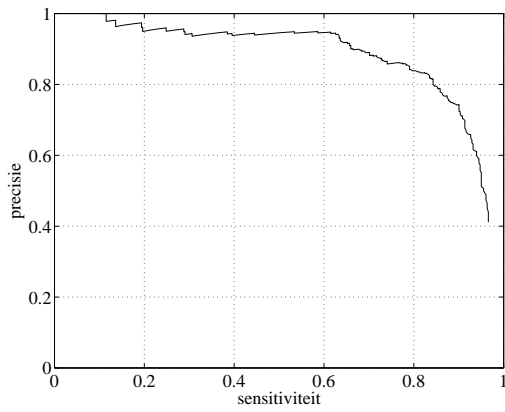
waarden p (het aantal keer dat de grootste testwaarde groter moet zijn dan de tweede grootste testwaarde in het glijdende venster) en venstergrootten n . We zien dat piekdetectie een slechtere performantie geeft in vergelijking met globale drempels. Dit is te verklaren door de aanwezigheid van de talrijke kleine ruispieken.

We zouden de resultaten nog kunnen verbeteren indien we de data door een laagdoorlaatfilter laten gaan zodat de ruis vermindert en een ander globale drempel (die veel lager ligt dan de globale drempel gebruikt in de vorige methode) gebruiken. Piekdetectie heeft als voordeel dat ze op alle technieken kan gebruikt worden zonder drempels manueel te hoeven instellen.

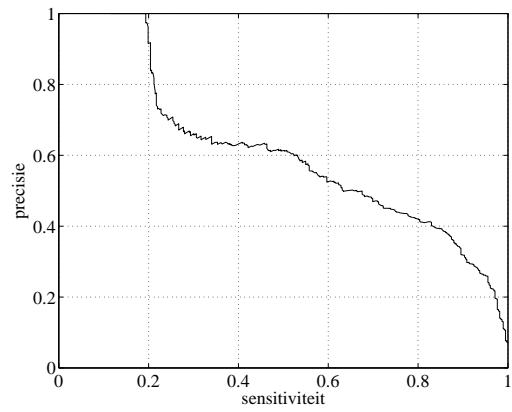
Automatische drempelselectie met een variant van gaussiaanse ruismodellering

In bijlage A wordt de drempel voor gaussiaanse ruismodellering gegeven door $\delta = \mu + \alpha\sigma$. Uit experimenten blijkt dat deze drempel een heel lage performantie heeft, daarom definiëren we de nieuwe drempel als: $\delta = \alpha\mu$. We illustreren dit voor de pixelvergelijking met uitmiddeling (2) in figuur 4.8 en voor de histogramvergelijking (3) in figuur 4.9.

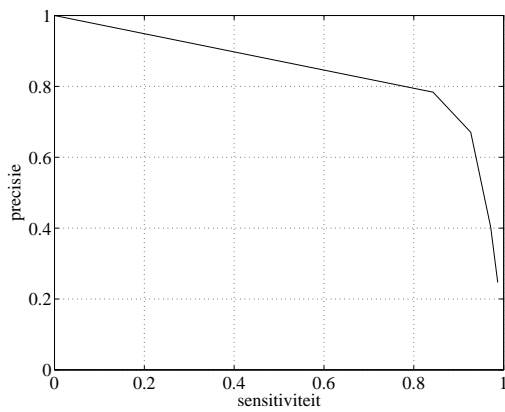
We zien dat voor kleine venstergrootten ($n = 5$) de gaussiaanse ruismodellering heel goede resultaten geeft. De gaussiaanse ruismodellering heeft een hoge sensitiviteit bij



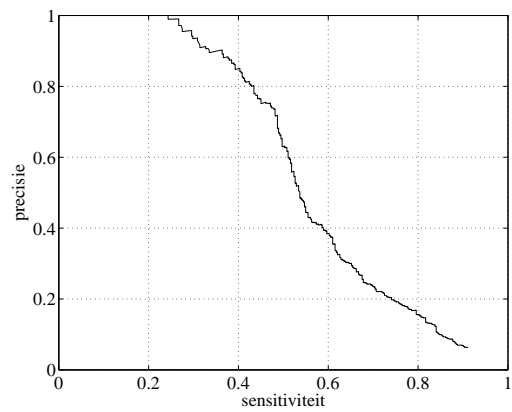
(a) zelf-similariteitsanalyse (5)



(b) DCT-vergelijking (6)

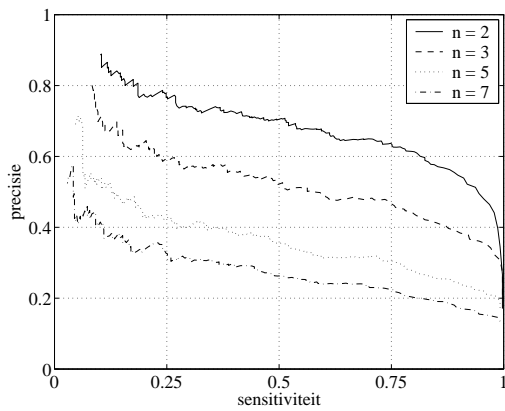


(c) DC-histogramvergelijking (7)

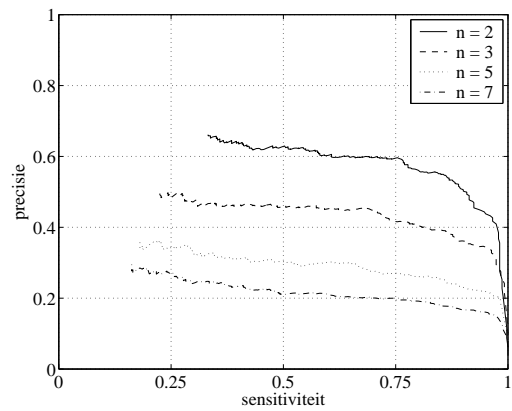


(d) bewegingsvectoren-analyse (8)

Figuur 4.6: De precisie uitgezet in functie van de sensitiviteit voor de verschillende technieken (deel 2).



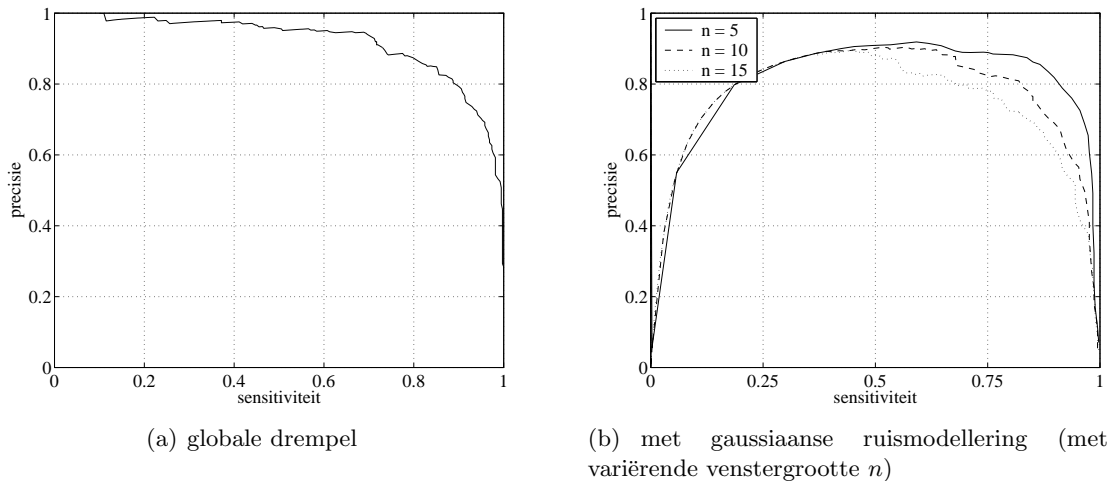
(a) pixelvergelijking met uitmiddeling (2)



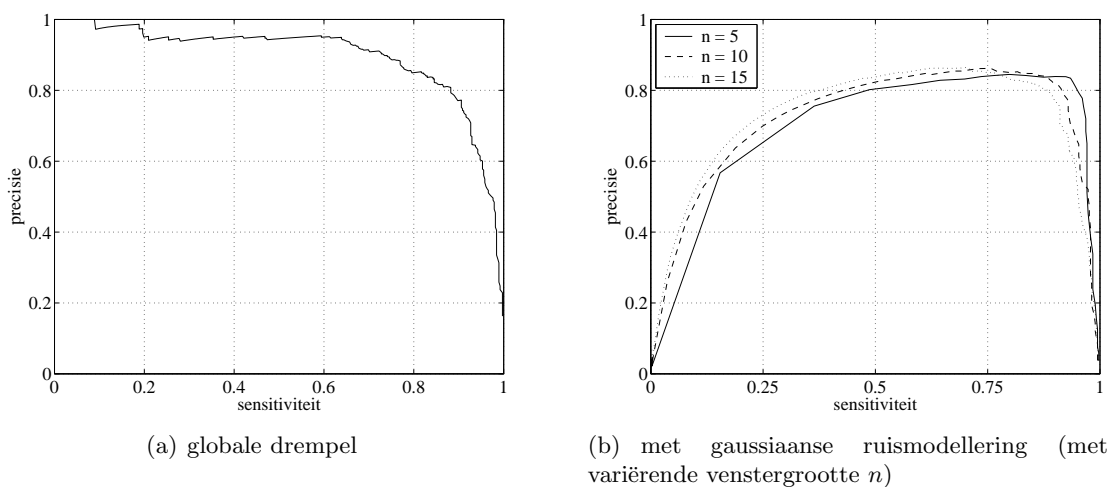
(b) histogramvergelijking (3)

Figuur 4.7: De precisie uitgezet in functie van de sensitiviteit met behulp van piekdetectie en variërende venstergrootte n .

relatief hoge precisie. Ze heeft dus een hoge performantie in het werkingsgebied, zelfs hoger dan bij een globale drempel. De gaussiaanse ruismodellering heeft ook een algemeen lagere precisie in vergelijking met de globale drempel. Het verloop van de curven is te verklaren door het hoge aantal valse detecties $\kappa(V)$ (wederom door die ruispieken). We kunnen dit verbeteren met een laagdoorlaatfilter.



Figuur 4.8: De precisie uitgezet in functie van de sensitiviteit voor pixelvergelijking met uitmid-deling (2).



Figuur 4.9: De precisie uitgezet in functie van de sensitiviteit voor histogramvergelijking (2).

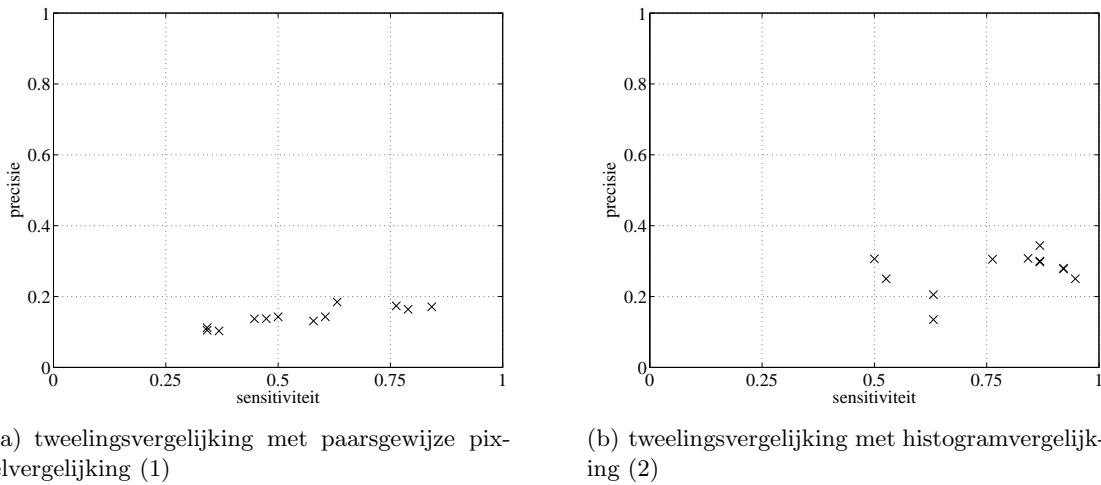
Detectie van transities

Voor detectie van transities, vergelijken we de volgende methoden:

1. tweelingsvergelijking met paarsgewijze pixelvergelijking (zie paragraaf 2.2.4).
2. tweelingsvergelijking met histogramvergelijking.

3. spatiaal-temporele snede-analyse zonder spatiale reductie voor dissolve-detectie (zie paragraaf 2.2.7).
4. spatiaal-temporele snede-analyse zonder spatiale reductie voor wipe-detectie. We passen deze techniek niet toe op ons volledige testset omdat de convolutieberekeningen (vergelijkingen (2.23) en (2.25)) zeer rekenintensief zijn.

Bij de tweelingsvergelijkingen zijn er twee parameters (de drempels) die de detectie niet-lineair beïnvloeden. Daarom is het zeer moeilijk om de optimale drempels te vinden. De performantie voor de twee tweelingsvergelijkingen wordt gegeven in figuur 4.10. De detectie van een transitie is correct indien er meer dan 50% overlapping is met de echte transitie.



Figuur 4.10: De precisie uitgezet in functie van de sensitiviteit van de detectie van transities voor de tweelingsvergelijkingen met variërende drempels.

We kunnen uit de figuren twee conclusies trekken: 1. de performantie ligt bijzonder laag in vergelijking tot cutdetectie-methoden (dit is normaal omdat de detectiemethoden bijzonder gevoelig zijn voor bewegingen) en 2. de tweelingsvergelijking op basis van histogramvergelijking heeft een hogere performantie dan de tweelingsvergelijking op basis van paarsgewijze pixelvergelijking: zoals we in paragraaf 2.2.3 besproken zijn histogramvergelijkingen robuuster tegen bewegingen.

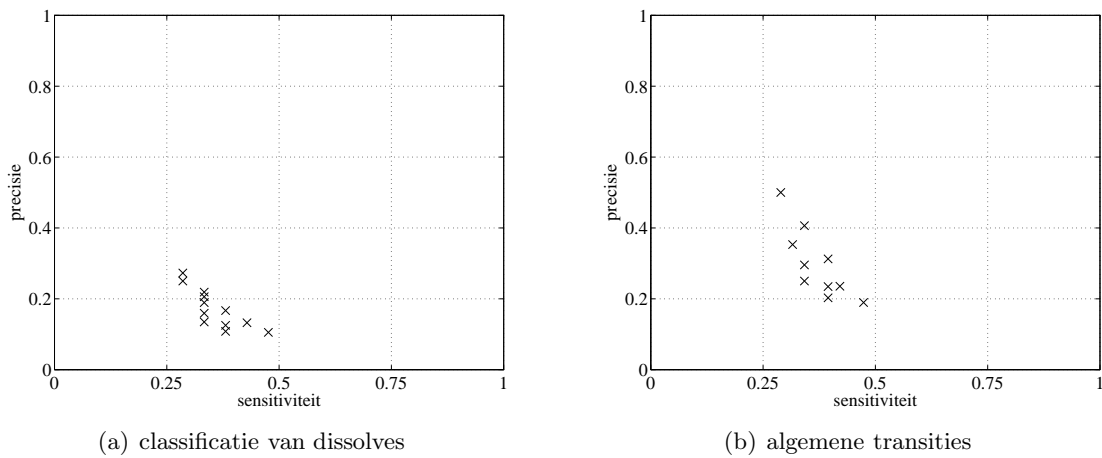
Een van de voordelen bij spatiaal-temporele snede-analyse is dat er onderscheid wordt gemaakt tussen de verschillende types transitie. Omdat de rekentijd meer dan lineair toeneemt met stijgende dimensie (door de convolutieberekeningen), wordt enkel de sensitiviteit van horizontale, verticale en diagonale wipes berekend. De sensitiviteit voor de wipes van de *starwars*-videosequenties bedraagt 72.7%. De sensitiviteit is berekend op slechts 11 wipes, verder onderzoek is dus nodig om een betrouwbaar resultaat te bekomen.

Voor dissolves wordt de volledige testset gebruikt (omdat er geen convolutieberekeningen nodig zijn). De performantie voor de dissolve-detectie wordt gegeven in figuur 4.11 voor de verschillende parameters (de drie parameters om de paraboolvorm te specificeren). De performantie is zeer laag: de dissolve-detectie baseert zich immers op een mathematisch model. Door allerlei ruis en bewegingen klopt het model meestal niet met de werkelijkheid.

Tabel 4.2: De precisiefout per beeld van de transitie voor enkele gebruikte detectiemethoden.

	gemiddelde precisiefout per beeld
tweelingsvergelijking met pixelvergelijking (1)	1.98210
tweelingsvergelijking met histogramvergelijking (2)	0.48072
snede-analyse voor dissolve-detectie (3)	0.52483
snede-analyse voor wipe-detectie (4)	0.42250

We merken op dat de dissolve-detectie toegepast op alle transitie beter scoort dan enkel op de dissolves. Er zijn hier nog verbeteringen mogelijk (zie paragraaf 2.2.7).



Figuur 4.11: De precisie uitgezet in functie van de sensitiviteit voor de dissolve-detectie met verschillende parameters met behulp van spatiaal-temporele snede-analyse.

De precisiemaat stelt de nauwkeurigheid voor van de detectie van de shotgrenzen. Deze nauwkeurigheid wordt bepaald door de gemiddelde fout op de positie van de gedetecteerde shotgrenzen. De resultaten worden gegeven in tabel 4.2. We zien op de tabel dat de tweelingsvergelijking op basis van de histogramvergelijking een kleine fout heeft, daarom is de histogramvergelijking te verkiezen boven de paarsgewijze pixelvergelijking voor de tweelingsvergelijking.

Er is nog veel onderzoek nodig om transitie correct te detecteren. Vooral actiescènes vormen een probleem. De besproken technieken kunnen verbeterd worden door b.v. eerst bewegingscompensatie uit te voeren.

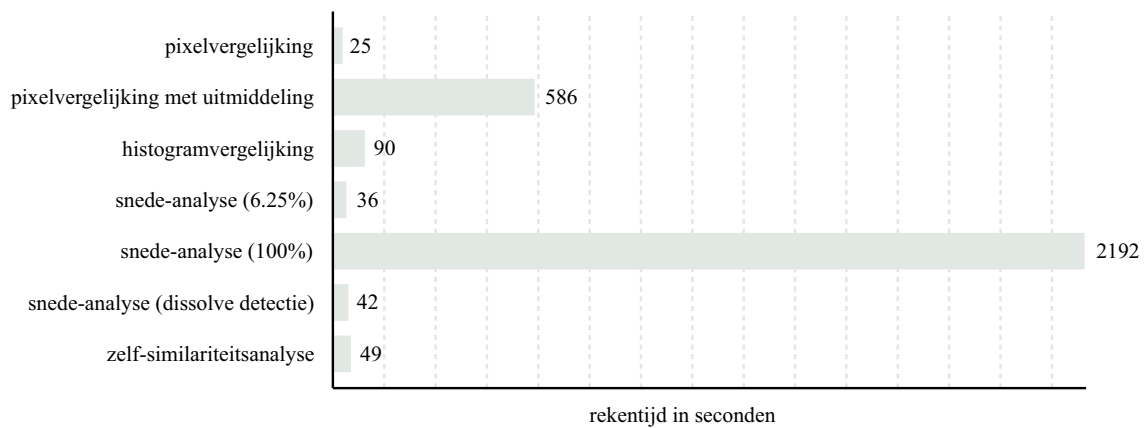
4.3.3 Rekentijden van de algoritmen

Het is belangrijk om te weten wat de snelheid van de technieken zijn om reële toepassingen te ontwerpen. Hoewel er methoden bestaan die de rekestijd reduceren (zie paragraaf 2.2.10) houden we in deze sectie geen rekening mee (de tijden kunnen dus nog drastisch ingekort worden). We houden ook geen rekening met de technieken die normaal

in gecomprimeerde videosequenties gebruikt worden, omdat ze in een praktische implementatie niet veel rekentijd vragen.

De reketijden worden getoond in figuur 4.12. De algoritmen zijn niet volledig geoptimaliseerd: er wordt geen gebruik gemaakt van speciale instructiesets zoals MMX (MultiMedia Extended), SSE (Streaming SIMD Extensions), ... Het snelste algoritme is de paarsgewijze pixelvergelijking met 100 beelden per seconde (zonder decoder-tijd meegerekend). De verwerkingssnelheid van de histogramvergelijking is ongeveer 4.5 beelden per seconde. Dit is te traag voor realtime-toepassingen, maar met allerlei optimalisaties (b.v. met speciale instructiecode) en reductietechnieken (zie paragraaf 2.2.10) kunnen we het algoritme veel sneller maken.

Hoewel spatiaal-temporele snede-analyse zonder spatiale reductie enorm veel reketijd vraagt, biedt deze techniek veel nieuwe perspectieven: we kunnen hiervoor andere (snellere) algoritmen gebruiken voor de textuurmodellering.



Figuur 4.12: De reketijden voor de verschillende algoritmen gemeten op een computer met 1 Gb RAM en AMD Athlon XP 1800+ processor. De videosequentie bevat 300 beelden of ongeveer 12 seconden. Het decoderen neemt ongeveer 22 seconden in beslag.

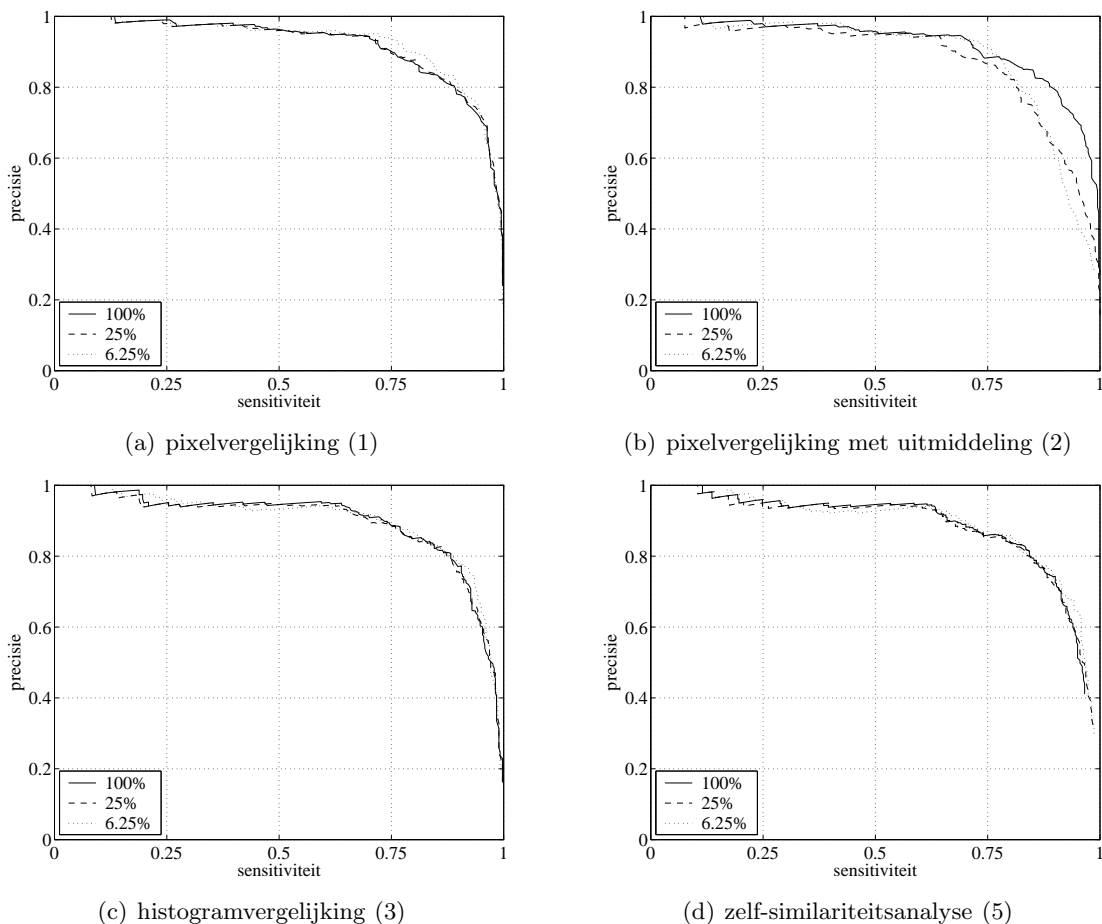
4.3.4 Robuustheid tegen spatiale reductie

In internettoepassingen worden meestal sterk gereduceerde versies van videosequenties gebruikt. Spatiale reductie kan ook gebruikt worden om de reketijd sterk te verkleinen. We onderzoeken nu wat de invloed is van spatiale reductie op de temporele videosegmentatie. We reduceren de dimensies bilineair met respectievelijk 25% en 6.25% (voor een dimensie van 640×480 wordt dit respectievelijk 320×240 en 160×120) en comprimeren in MPEG-2 CBR (constante bitrate) aan 500 kbits/sec. De performantiematen voor verschillende technieken (behalve snede-analyse (8)) worden uitgezet in figuren 4.13 en 4.14.

We zien dat bij pixelvergelijking (1), histogramvergelijking (3), zelf-similariteitsanalyse (5) en DCT-vergelijking (6), de algoritmen met spatiale reductie soms beter presteren dan bij de originele versie. Bij de spatiale reductie wordt de bilineaire methode gebruikt, dit impliceert dat de gereduceerde versies eigenlijk uitgemiddelde versies zijn (daarom komt het randverschijnsel bijna niet voor bij de pixelvergelijking met uitmiddeling (2)). Deze uitmiddeling filtert ook alle ruis en beweging uit, zodat de algoritmen dan weer iets

beter presteren. Een tweede oorzaak van de goede prestaties ligt aan de encoding voor de sterkste reductiefactor: de videosequenties worden geëncodeerd aan een relatief hogere bitrate dan bij de originele videosequentie (origineel (100%): 3000 kbits/sec en gereduceerd (6.25%): 500 kbits/sec). Hierdoor bevatten de gereduceerde beelden meer bits per pixel dan de originele beelden, wat uiteindelijk resulteert in relatief meer nauwkeurige beelden.

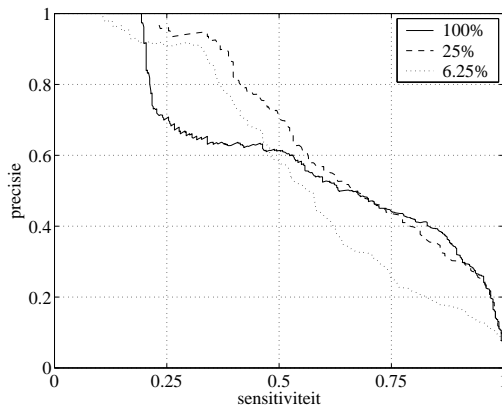
We merken ook op dat bij DC-histogramvergelijking (7), de spatiale reductie het algoritme onbruikbaar maken. De rij- en kolomhistogrammen bevatten dan te weinig informatie om een goede performantie te hebben.



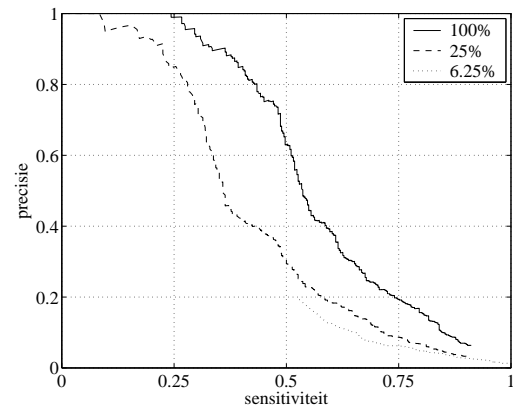
Figuur 4.13: De precisie uitgezet in functie van de sensitiviteit voor de verschillende dimensies (deel 1).

4.3.5 Scèneclustering

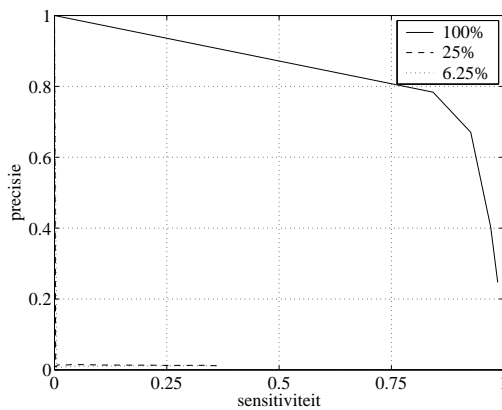
Bij scèneclustering is het heel moeilijk om een performantie aan vast te knopen: de computer kan namelijk nog geen semantisch zinvolle clusters maken. Syntactisch clusteren (volgens welbepaalde regels) gaat wel, maar zelfs als gebruiker hebben we daar soms problemen mee. We geven een voorbeeld ter illustratie: camera volgt persoon *a* in omgeving *A*, die naar omgeving *B* gaat. Vervolgens wordt in een totaal andere scène persoon *b* gevolgd die vertrekt van omgeving *B* naar omgeving *A* gaat. We hebben 2 totaal andere



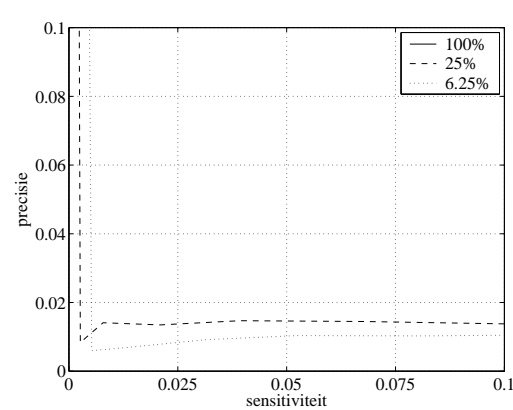
(a) DCT-vergelijking (6)



(b) bewegingsvectoren-analyse (8)



(c) DC-histogramvergelijking (7)



(d) DC-histogramvergelijking (7) ($10\times$ uitvergroot)

Figuur 4.14: De precisie uitgezet in functie van de sensitiviteit voor de verschillende dimensies (deel 2).

scènes, maar worden toch geclusterd tot één scène omdat ze allebei gelijkaardige kenmerken vertonen. Omgekeerd: in een scène wordt een detailopname getoond van een voorwerp. De shot met het voorwerp behoort tot de scène, maar wordt als aparte cluster opgenomen.

De iteratieve boomclustering (in YCrCb-kleurenruimte) met temporele correctie (zie paragrafen 2.4.2 en 2.4.3) heeft een precisie van 100% voor onze testset (alle gegenereerde clusters bevatten enkel shots die bij elkaar horen) en een sensitiviteit van ongeveer 70% (30% van alle shots kunnen nog (beter) geclusterd worden).

4.4 Panoramische beelden

4.4.1 Registratie

Een goed panoramisch beeld is volledig afhankelijk van de registratie tussen de beelden: een afwijking in de registratie is duidelijk zichtbaar in het eindresultaat. Daarom vormt de registratie het belangrijkste onderdeel in de opbouw van panoramische beelden.

We maken een vergelijkende studie van de volgende registratiemethoden voor het translatiemodel (zie paragraaf 3.3.1):

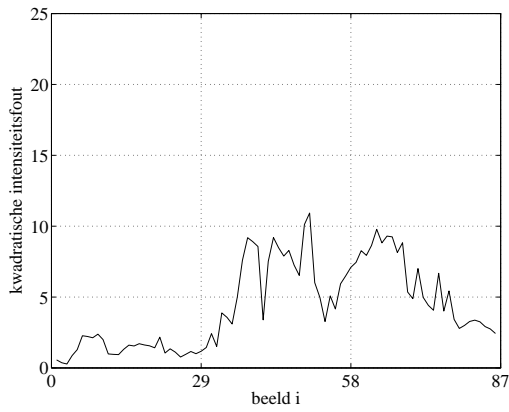
1. detectie van de dominante beweging van bewegingsvectoren op basis van een 2-dimensionale histogram.
2. registratie door globale minimalisatie van de kwadratische intensiteitsfout met terugkoppelingslus (zie vergelijking (3.10)).
3. registratie door globale minimalisatie van de kwadratische kleurenfout met terugkoppelingslus (zie vergelijking (3.11)). De gebruikte kleurenruimte is RGB.
4. bewegingsdetectie met behulp van snede-analyse.
5. registratie op basis van fase correlatie.
6. registratie met behulp van de Kanade-Lucas-Tomasi tracker.

Bij meer complexe bewegingsmodellen (b.v. affien) werkt slechts enkel de Kanade-Lucas-Tomasi tracker voldoende efficiënt van de besproken algoritmen (zie als voorbeeld figuur 3.23).

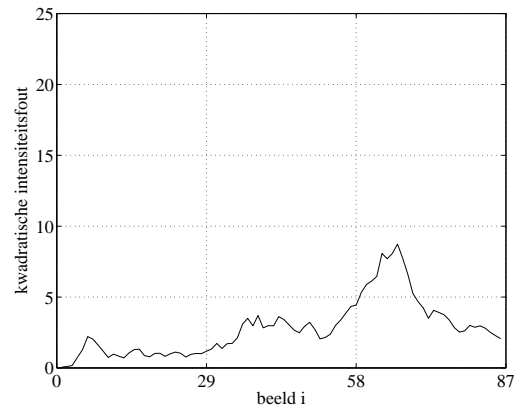
Het gemiddeld kwadratisch intensiteitsverschil per pixel wordt voor de verschillende technieken uitgezet in figuren 4.15 en 4.16.

We zien dat de registratie m.b.v. bewegingsvectoren (1) en de globale minimalisaties (2,3) een relatief lage kwadratische intensiteitsfout in vergelijking tot de andere technieken. In de tweede helft van de shot zijn de gemiddelde intensiteitsfouten hoger dan in de eerste helft, dit komt door de beweging van de persoon in de shot (vergelijkbaar met een groot objectbeweging). Dat is meteen de reden waarom de Kanade-Lucas-Tomasi tracker hier zo slecht presteert (als er objectbeweging is van objecten die kleiner zijn dan het controlevenster, dan heeft de Kanade-Lucas-Tomasi tracker een even goede of zelfs betere performantie dan de globale minimalisaties). Het panoramisch beeld staat in figuur 4.17 voor de globale minimalisatie van de kwadratische intensiteitsfout (1).

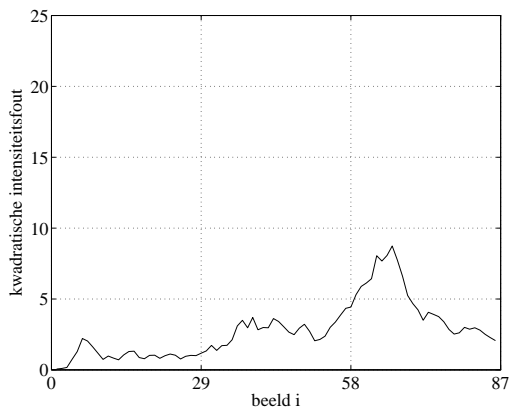
In figuren 4.18, 4.19 en 4.20 worden de verschillende posities uitgezet voor de opeenvolgende beelden van de shot. De exacte posities worden benaderd door de globale minimalisaties (2,3) (die beide dezelfde resultaten geven). In figuren 4.21 en 4.22 worden de resultaten in één grafiek gezet om ze met elkaar te kunnen vergelijken. Een cameratranslatie is meestal een vloeiende beweging, dit moet dus ook gelden voor de curven voor de X - en Y -translatie. Dit is b.v. niet zo voor de registratie op basis van bewegingsvectoren (1). De horizontale translatie voor de Kanade-Lucas-Tomasi tracker wijkt ook heel veel af in vergelijking tot de andere methoden. Dit ligt aan de objectbewegingen. Registratie op basis van spatiaal-temporele snede-analyse geeft ook slechte resultaten, dit is te wijten aan het feit dat objectbewegingen, die op de snede vallen, als cameratranslaties waargenomen worden. De methode kan aanzienlijk robuuster gemaakt worden door veel meer dan twee (horizontale en verticale) sneden te nemen. Registratie m.b.v. fase correlatie is niet altijd robuust: op de figuren is te zien dat sommige registraties niet correct verlopen (dit is



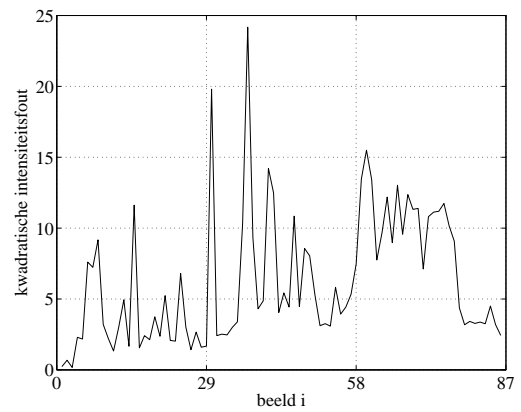
(a) registratie m.b.v. bewegingsvectoren (1)



(b) minimalisatie van de intensiteitsfout (2)

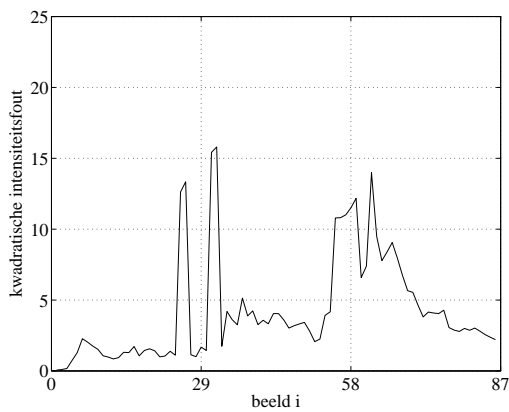


(c) minimalisatie van het RGB-verschil (3)

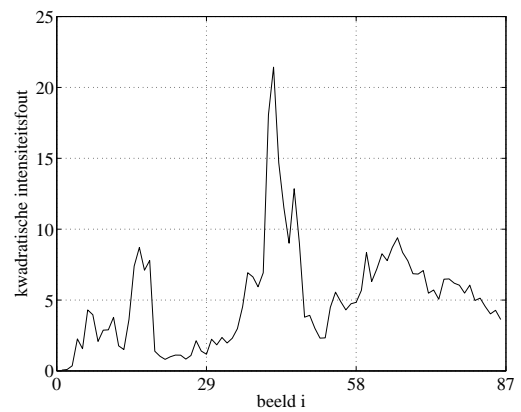


(d) registratie m.b.v. snede-analyse (4)

Figuur 4.15: De gemiddelde kwadratische intensiteitsfout per pixel voor de lokale registraties van *austin* (beeld 2662 - 2749) (deel 1).



(a) registratie m.b.v. fase correlatie (5)



(b) registratie m.b.v. de Kanade-Lucas-Tomasi tracker (6)

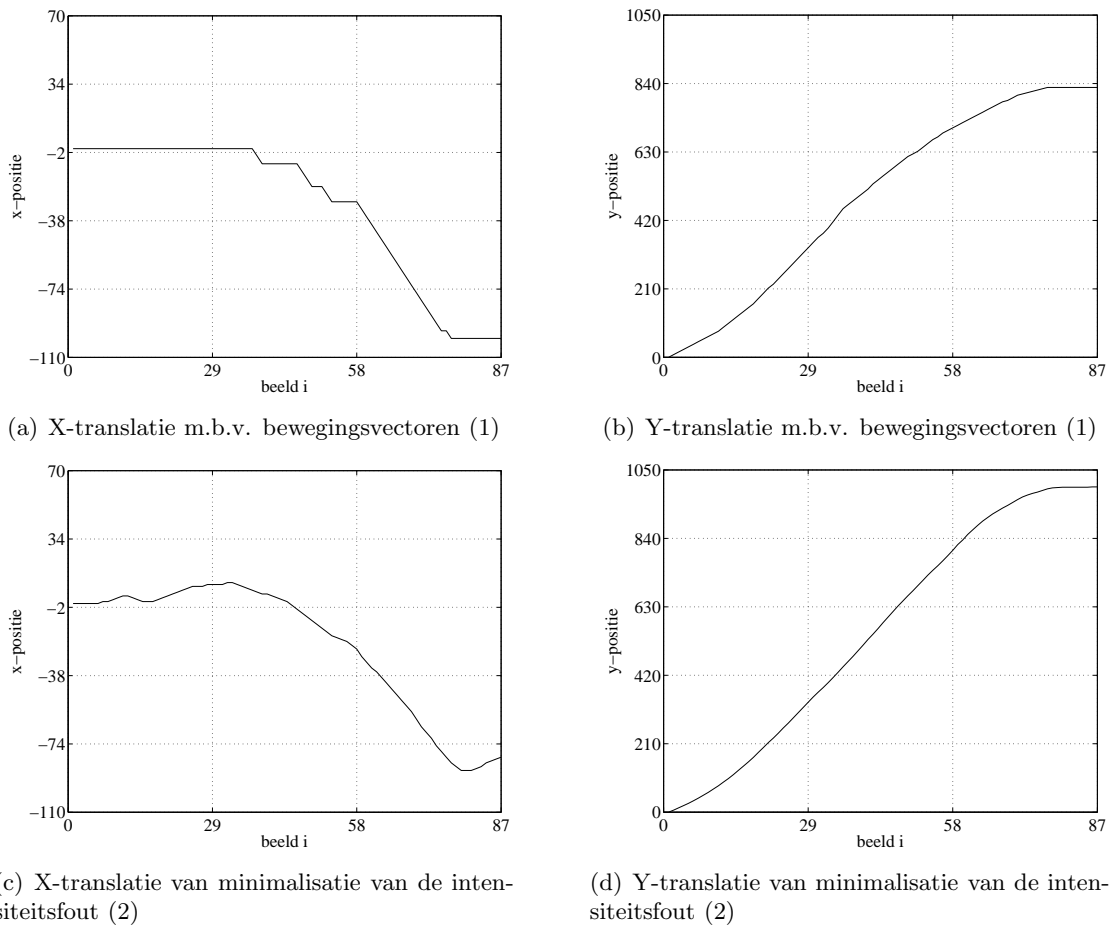
Figuur 4.16: De gemiddelde kwadratische intensiteitsfout per pixel voor de lokale registraties van *austin* (beeld 2662 - 2749) (deel 2).



Figuur 4.17: Het panoramisch beeld van *austin* (beeld 2662 - 2749).

eveneens deels te wijten aan de grote objecten).

We geven ook de rekestijden van de verschillende algoritmen in figuur 4.23. De algoritmen zijn ontworpen zonder optimalisatie op basis van instructiesets (zie paragraaf 4.3.3). Uit de meettijden leiden we af dat terugkoppeling de rekestijd drastisch kan inkorten.

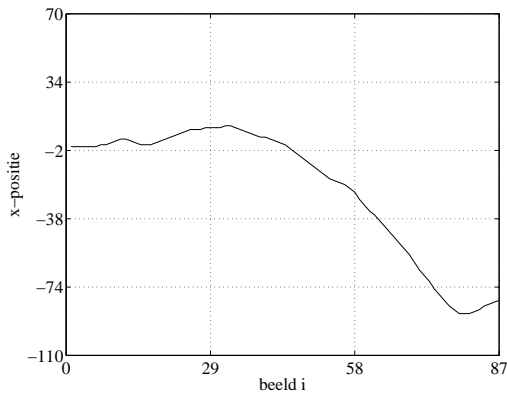


Figuur 4.18: De accumulatieve translatiewaarden voor de opeenvolgende registraties van *austin* (beeld 2662 - 2749) (deel 1).

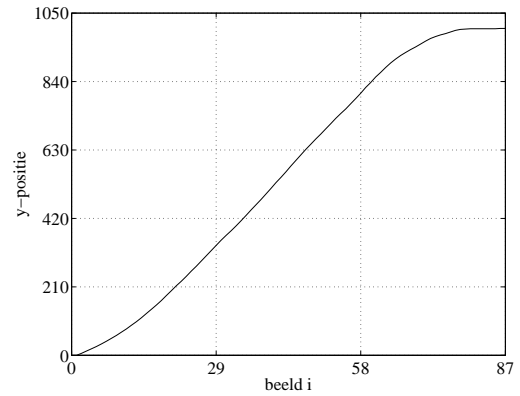
De terugkoppeling zorgt ook voor vloeiende curven voor X - en Y -translatie, dus hebben we tweemaal voordeel om terugkoppeling te gebruiken. Merk op dat de tijden voor de teruggekoppelde technieken gemeten zijn zonder initiële parameterschatting. Dit impliceert dat de actuele rekentijd hoger (de initiële schatting gebeurt door hetzelfde algoritme maar zonder terugkoppeling) of lager ligt (de initiële schatting van de beeldsequentie wordt bepaald door een ander maar sneller algoritme, b.v. op basis van bewegingsvectoren of snede-analyse).

De rekentijd voor de registratie op basis van fase correlatie is heel groot, daarom is deze techniek minder interessant in de praktijk. Registratie m.b.v. spatiaal-temporele snede-analyse is wel zeer snel, maar is daarentegen minder robuust. De registratie door globale minimalisatie van de kwadratische kleurenfout is trager dan de minimalisatie van de kwadratische intensiteitsfout. Beide methoden geven bijna hetzelfde resultaat, daarom is de minimalisatie van de kwadratische intensiteitsfout te verkiezen boven de andere methode.

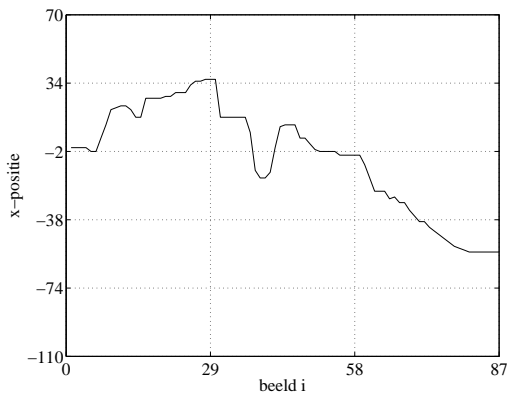
Samengevat kunnen we besluiten dat de registratie m.b.v. de globale minimalisatie van de kwadratische intensiteitsfout zeer goede resultaten oplevert en voor complexe be-



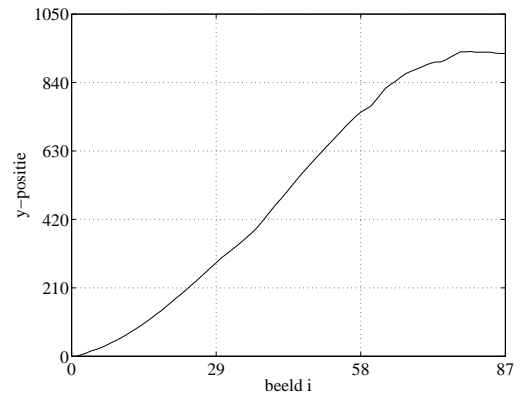
(a) X-translatie van minimalisatie van het RGB-verschil (3)



(b) Y-translatie van minimalisatie van het RGB-verschil (3)



(c) X-translatie van registratie m.b.v. snede-analyse (4)



(d) Y-translatie van registratie m.b.v. snede-analyse (4)

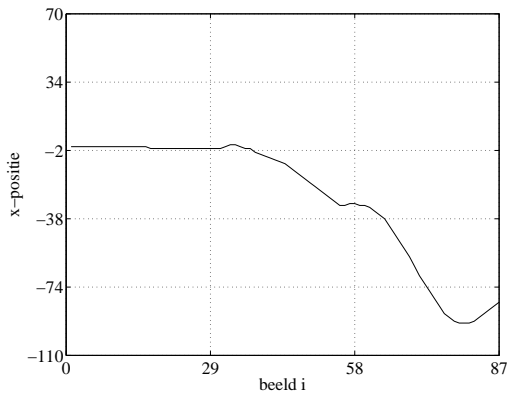
Figuur 4.19: De accumulatieve translatiewaarden voor de opeenvolgende registraties van *austin* (beeld 2662 - 2749) (deel 2).

wegingsmodellen (vanaf rotatie) is de Kanade-Lucas-Tomasi tracker aangewezen op voorwaarde dat er geen bewegingen zijn van objecten, die groter zijn dan het controlevenster. Door de kleine rekentijd van snede-analyse en bewegingsvectoren, zijn beide methoden geschikt voor detectie van camerabewegingen zonder expliciet de registratie nauwkeurig te hoeven berekenen. Deze detectie wordt gebruikt om automatisch shots te selecteren die geschikt zijn om door panoramische beelden voorgesteld te worden.

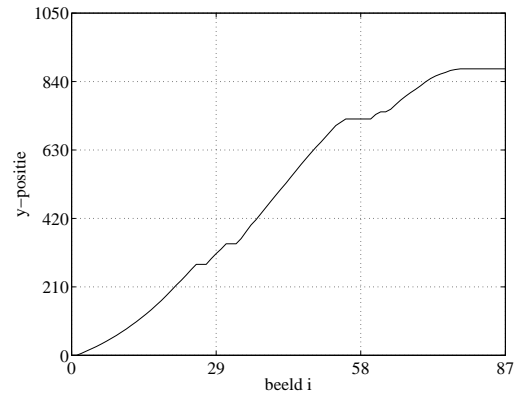
4.4.2 Kwaliteit van een panoramisch beeld

De kwaliteit van een panoramisch beeld hangt van twee factoren af: 1. de registratie en 2. de plaatsing van de pixel op het canvas. Een foute registratie leidt tot een zichtbare discontinuïteit in het panoramisch beeld, daarom moet de registratie zeer nauwkeurig gebeuren (met behulp van globale registratie, zie figuur 3.28).

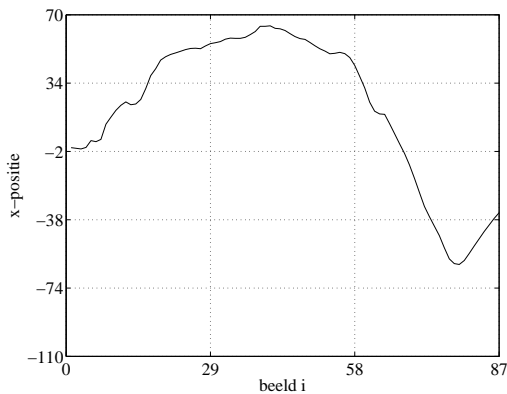
De *afwerking* van het panoramisch beeld gebeurt door een goede selectie van de overlappende pixelinformatie. In figuren 4.24 en 4.25 worden de verschillende keuzes geïllustreerd voor dezelfde shot. Selectie van pixelinformatie uit meerdere beelden (uitmiddeling en mediaan) levert ons onscherpe beelden op. Bij de uitmiddeling van de overlappende pix-



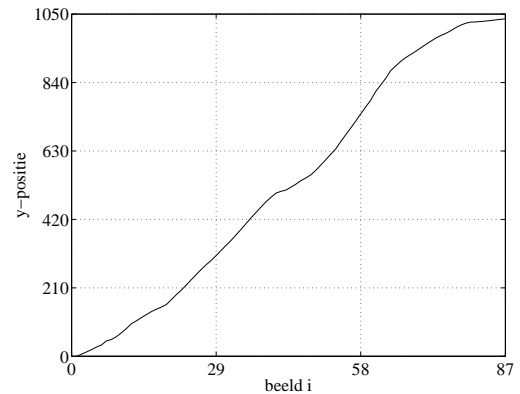
(a) X-translatie van registratie m.b.v. fase correlatie (5)



(b) Y-translatie van registratie m.b.v. fase correlatie (5)



(c) X-translatie van registratie m.b.v. de Kanade-Lucas-Tomasi tracker (6)



(d) Y-translatie van registratie m.b.v. de Kanade-Lucas-Tomasi tracker (6)

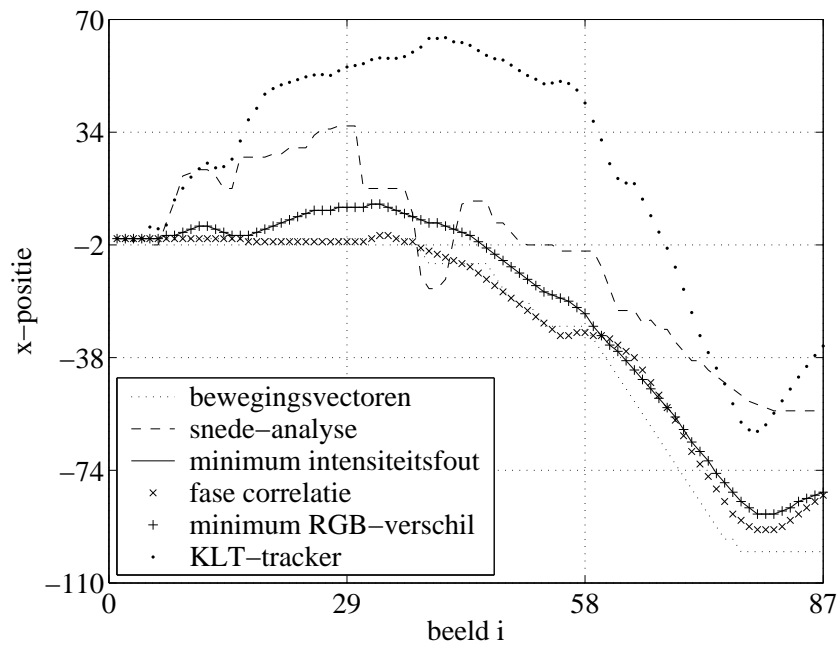
Figuur 4.20: De accumulatieve translatiewaarden voor de opeenvolgende registraties van *austin* (beeld 2662 - 2749) (deel 3).

els wordt b.v. de rechtse streep wazig. Bij de mediaan treedt er op sommige plekken beelddistorsie op.

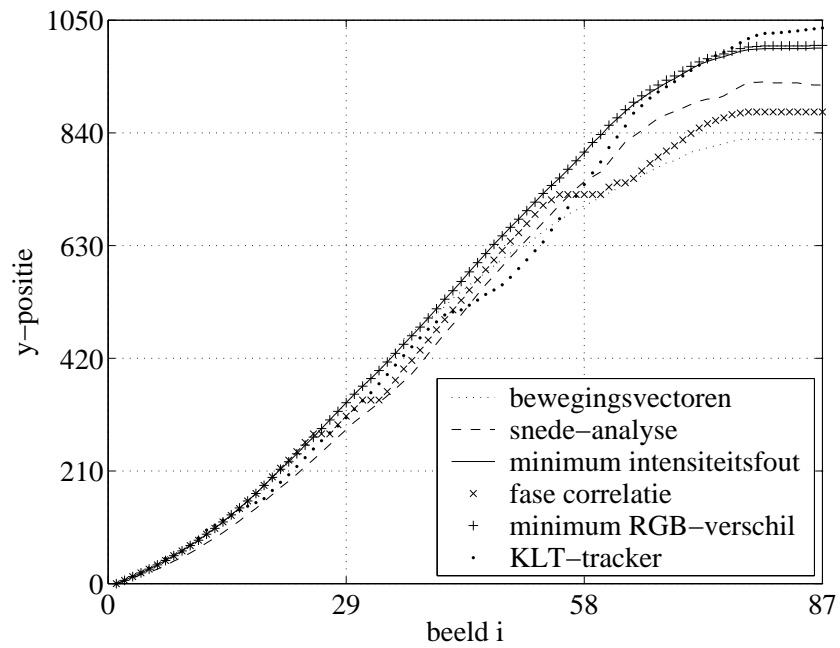
Selectie van pixelinformatie uit één beeld levert ons scherpe beelden van hogere kwaliteit op. Bij selectie van het eerste beeld zien we duidelijk de randen van de afzonderlijke beelden (b.v. aan de rechtse streep). Bij segmentatie wordt een goede oplossing gezocht voor de beeldranden met behoud van scherpte en produceert ook kwalitatief het beste panoramisch beeld.

Er is nog tal van aanpassingen mogelijk om panoramische beelden te verbeteren: bij de registratie kunnen we de complexiteit van de bewegingsmodellen verhogen om alle reële cameraparameters te kunnen schatten. Om goede registraties te maken moet er ook rekening gehouden worden met het dieptezicht van de verschillende objecten: de objecten hebben immers een verschillende afstand tot de camera. Indien we hiermee geen rekening houden, worden de panoramische beelden vervormd door de verschillende dieptezichten.

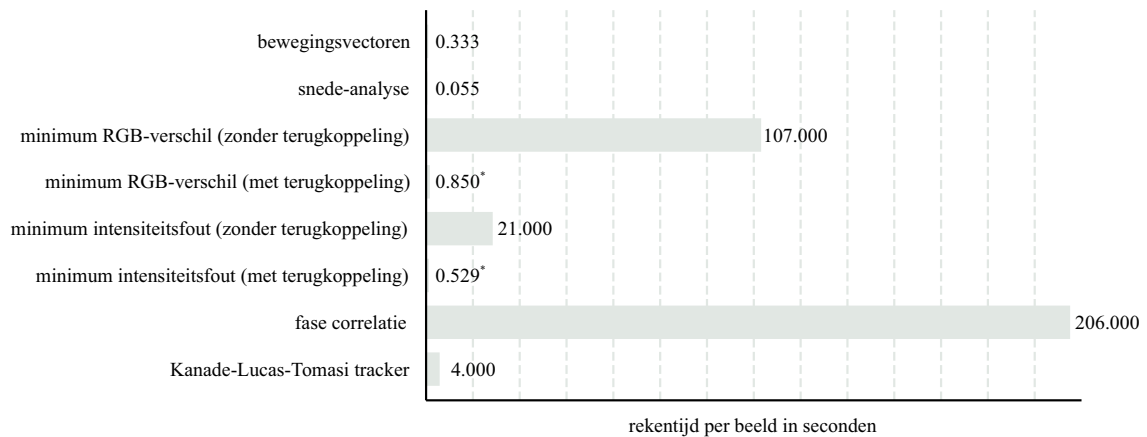
Een ander probleem is de veranderende belichting van de shot. Het kan voorkomen dat de belichting verandert tijdens de shot, in het panoramisch beeld is deze verandering



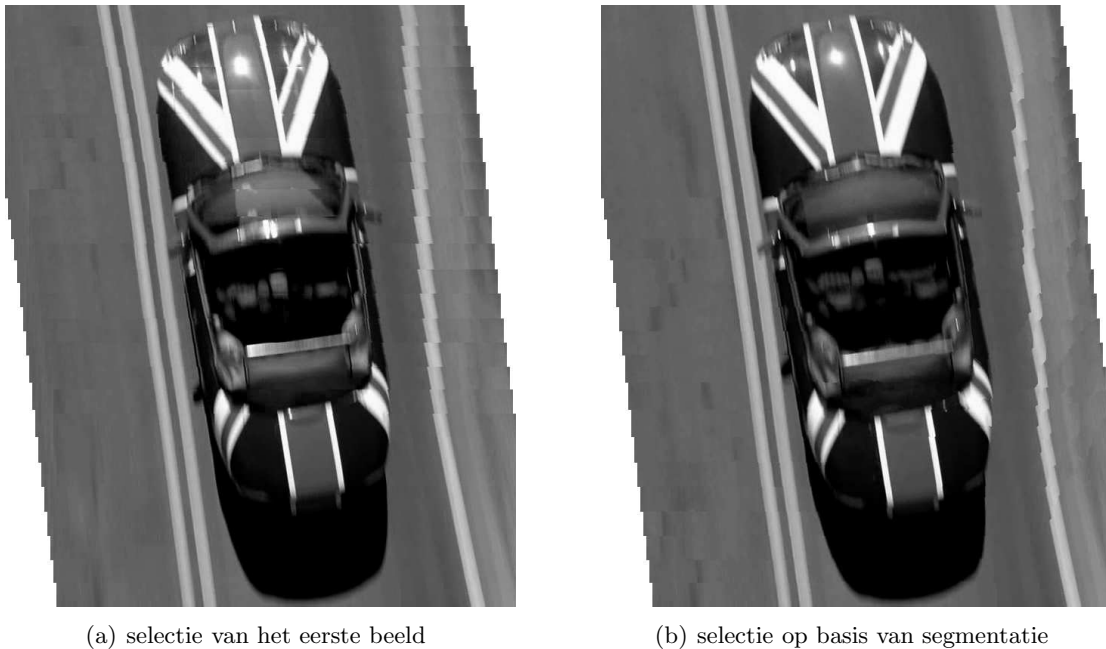
Figuur 4.21: De accumulatieve horizontale translatiewaarden voor de opeenvolgende registraties van *austin* (beeld 2662 - 2749) uitgezet voor de verschillende technieken.



Figuur 4.22: De accumulatieve verticale translatiewaarden voor de opeenvolgende registraties van *austin* (beeld 2662 - 2749) uitgezet voor de verschillende technieken.



Figuur 4.23: De rekestijden (per 608×256 beeld) voor de verschillende algoritmen gemeten op een computer met 1 Gb RAM en AMD Athlon XP 1800+ processor. (* zonder initiële parameterschatting voor de terugkoppellus)



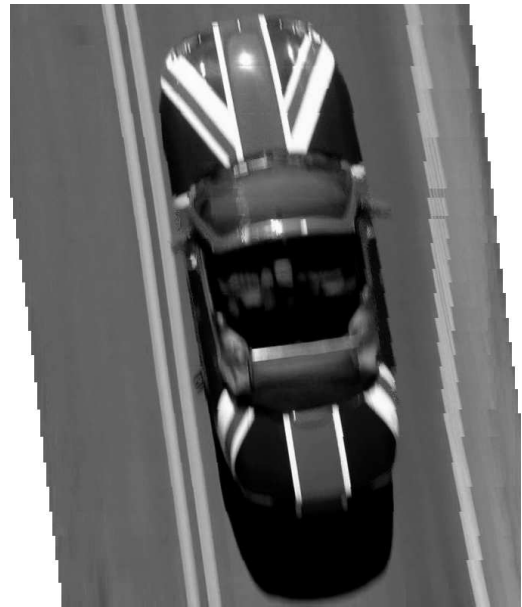
Figuur 4.24: Het panoramisch beeld van *austin* (beeld 878 - 896) (deel 1).

duidelijk zichtbaar indien we de belichting niet corrigeren.

De segmentatie-techniek kan nog verbeterd worden als we (al dan niet bewegende) objecten kunnen onderscheiden van de achtergrond. We kunnen de watershed techniek gebruiken om vooraf objecten te selecteren en ze in hun geheel op het canvas te plaatsen in plaats van op het pixelniveau te werken.



(a) uitmiddeling van de overlappende pixels



(b) selectie op basis van de mediaan

Figuur 4.25: Het panoramisch beeld van *austin* (beeld 878 - 896) (deel 2).

Hoofdstuk 5

Besluit

In deze scriptie werd er onderzocht hoe automatische samenvattingen gegenereerd kunnen worden van videosequenties. Verschillende algoritmen voor shotdetectie en scèneclustering werden bestudeerd, geïmplementeerd en vergeleken. Uit de experimentele resultaten blijkt dat de eenvoudige paarsgewijze pixelvergelijking en de histogramvergelijking goed presteren voor detectie van cuts. Een vergelijkende studie toont aan dat complexe technieken niet superieur zijn voor cutdetectie. Voor de detectie van transities, biedt de spatiaal-temporele snede-analyse de meeste perspectieven.

Uit onderzoek blijkt dat de hiërarchische boomclustering met temporele correctie zeer goede resultaten oplevert met betrekking tot scèneclustering. Deze scèneclustering kan uitgebreid worden door domeinkennis te integreren om zo semantisch „correcte” samenvattingen te maken. Semantische samenvattingen zijn zeker niet uniek: er bestaan immers veel verschillende (even goede) samenvattingen van een boek of film.

Vervolgens hebben we onderzocht hoe een samenvatting zinvol en visueel attractief aan de gebruiker kan worden voorgesteld aan de hand van sleutelbeelden. Deze sleutelbeelden worden rechtstreeks uit de shots geselecteerd of als panoramische beelden gegenereerd uit shots met camerabewegingen. Panoramische beelden hebben een ruim toepassingsgebied: van videocompressie tot reconstructie van hoge resolutiebeelden. Om panoramische beelden te construeren, werden in de eerste fase verscheidene registratiemethoden bestudeerd en vergeleken. Uit de experimenten kunnen we besluiten dat voor relatief eenvoudige cameratranslaties de globale minimalisatie van de kwadratische intensiteitsfout zeer robuuste resultaten geeft. Voor complexere bewegingsmodellen is de registratie op basis van de Kanade-Lucas-Tomasi tracker aangewezen op voorwaarde dat er geen bewegingen zijn van objecten die groter zijn dan het controlevenster.

Vervolgens werd in de scriptie onderzocht hoe de kwaliteit van een panoramisch beeld beïnvloed wordt door de selectie van de pixelinformatie van de overlappende pixels. De resultaten tonen aan dat de segmentatie van de overlappende gebieden goede resultaten geeft. De kwaliteit kan verbeterd worden door gebruik te maken van informatie op objectniveau i.p.v. op pixelniveau. Hiervoor kunnen andere segmentatietechnieken, zoals de watershed techniek, gebruikt worden om objecten te onderscheiden van de achtergrond.

Panoramische beelden kunnen verder uitgebreid worden tot een 3D-reconstructie van een volledige omgeving.

Tenslotte wordt een gebruikersvriendelijke webpagina gegenereerd op basis van de

samenvatting die de gebruiker toelaat om doorheen een grote hoeveelheid videomateriaal hiërarchisch te navigeren. De gebruiker hoeft de videosequenties niet meer sequentieel te doorzoeken om de gewenste shot te vinden.

De webpagina geeft de sleutelbeelden weer en zorgt er voor dat bij een klik op het sleutelbeeld de bijhorende beeldsequentie wordt afgespeeld.

De eerste stap naar inhoudsgebaseerde zoeksystemen is reeds gelegd, maar de weg is nog lang...

Bijlage A

Drempels

Om goede resultaten te bekomen is het van groot belang om goede drempels te kiezen. In de volgende secties worden enkele methoden beschreven.

Globale drempel

Hierbij is de drempel δ een constante tijdens het volledige proces. De optimale waarde moet groot genoeg gekozen worden om geen valse detectie te krijgen en klein genoeg om geen reële positieven te missen. Meestal is er geen ideale drempel en moeten we veel slechte resultaten en weinig goede resultaten tegen elkaar afwegen.

De binaire functie wordt gegeven als:

$$g(i) = \begin{cases} 1, & \text{als } f(i) \geq \delta \\ 0, & \text{anders} \end{cases} \quad (\text{A.1})$$

Deze traditionele methode zal slecht werken als er veel variatie is in de testwaarden $f(i)$. Om een globale drempel te vinden, wordt er meestal een trainingssequentie gebruikt om een (initiële) drempel te bepalen.

Lokale drempel

De drempelwaarde van δ kan (zachtjes) variëren naargelang de sequentie van de testwaarden. De variatie kan vooraf gedefiniëerd zijn of kan afhangen van de informatie die tijdens het proces verzameld wordt.

Automatische methoden om drempels te vinden

De drempels automatisch vinden kan op verschillende manieren gebeuren en wordt verkozen boven het manueel instellen van de drempel, want elk type videosequentie (b.v. actiescène, dialogen, ...) vertoont andere karakteristieken. Hieronder bespreken we kort enkele methoden.

Gaussiaanse ruismodellering We geven hier een toepassing voor de paarsgewijze pixelvergelijking (besproken in paragraaf 2.2.1):

Stel dat alle ruis gaussiaans gemodelleerd kan worden en stel dat σ en μ respectievelijk de standaardafwijking en het gemiddelde van de testwaarden zijn. De probabiliteitsintegraal van de gaussiaanse distributie wordt gegeven door:

$$P(x) = \int_0^x \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \quad (\text{A.2})$$

De ondergrens is 0 omdat alle testwaarden in dit geval positief zijn. De testwaarden afkomstig van shots (en geen overgang) liggen in het bereik $[0, \mu + \alpha\sigma]$ voor een kleine constante α . Bijvoorbeeld voor $\alpha = 3$ zal vergelijking (A.2) 99.9% van de testwaarden bevatten. De drempel δ kan dan gekozen worden als:

$$\delta = \mu + \alpha\sigma \quad (\text{A.3})$$

Gekende distributies Indien de distributies gekend zijn, dan kunnen we een percentage assigneren aan de distributie:

$$c(\delta) = \frac{1}{f} \quad (\text{A.4})$$

waarbij $1/f$ de fractie van de testwaarden zijn die we moesten selecteren (of $1 - 1/f$ voor de inverse selectie) en $c(\delta)$ de cumulatieve distributie is:

$$c(\delta) = \sum_{i=0}^{\delta} p(i) \quad (\text{A.5})$$

Piekdetectie In veel algoritmen wordt er gebruik gemaakt van piekdetectie. Een methode hiervoor die de drempels automatisch kiest, baseert zich op 2 criteria: 1) de testwaarde is het maximum in een venster met lengte l over de testwaarden en 2) de testwaarde moet n keer groter zijn dan de tweede grootste testwaarde in het glijdende venster. Als 1) en 2) vervuld zijn, dan is er een piek gevonden. Een groot voordeel is dat de methode onafhankelijk is van de gebruikte testwaarden.

K-Means variatie De optimale drempel δ tussen de clusters wordt in figuur A.1 gegeven. Stellen we $\mu_l(\delta)$ het gemiddelde van alle testwaarden die lager liggen dan de drempel δ en $\mu_h(\delta)$ het gemiddelde van alle testwaarden die hoger liggen dan de drempel δ . Dan geldt voor de optimale drempel δ :

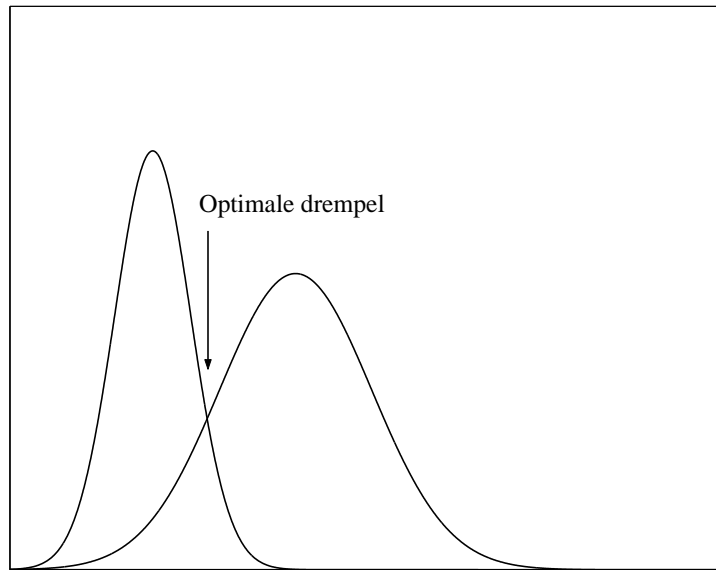
$$\begin{aligned} \forall f(i) \geq \delta : |f(i) - \mu_l(\delta)| &> |f(i) - \mu_h(\delta)| \\ \forall f(i) < \delta : |f(i) - \mu_l(\delta)| &< |f(i) - \mu_h(\delta)| \end{aligned} \quad (\text{A.6})$$

De optimale drempel kan worden gevonden door de volgende vergelijkingen iteratief op te lossen:

$$\begin{aligned}
\mu_l^i &= \frac{\sum_{i \in \text{cluster}_l} f(i)}{\|\text{cluster}_l\|} \\
\mu_h^i &= \frac{\sum_{i \in \text{cluster}_h} f(i)}{\|\text{cluster}_h\|} \\
\delta^{i+1} &= \frac{\mu_l^i + \mu_h^i}{2}
\end{aligned}
\tag{A.7}$$

Met $\|\text{cluster}_j\|$ het aantal elementen in cluster j en $j \in \{l, h\}$.

Deze methode werkt alleen goed als de distributies gelijkend zijn, en niet goed als ze verschillende varianties hebben.



Figuur A.1: De probabiliteitsdistributies met de optimale drempel.

Otsu methode De Otsu methode is een andere manier om een drempel te vinden voor de verschillende clusters. We proberen de drempel zo te bepalen dat de distributies elkaar zo weinig mogelijk overlappen. We kunnen natuurlijk de distributies niet veranderen, maar we kunnen wel de drempel bepalen die hen scheidt. Als we de drempel aanpassen, zal de spreiding van de ene cluster verkleinen en van de andere cluster vergroten. Het doel is om een optimale drempel te kiezen zodat de gecombineerde spreiding geminimaliseerd wordt.

Definiëren we de inwendige variantie als de gewogen som van de varianties $\sigma_j^2(\delta)$ van elke cluster:

$$\begin{aligned}
\sigma_{in}^2(\delta) &= n_l(\delta)\sigma_l^2(\delta) + n_h(\delta)\sigma_h^2(\delta) \\
n_l(\delta) &= \sum_{i=-\infty}^{\delta-1} f(i) \\
n_h(\delta) &= \sum_{i=\delta}^{\infty} f(i)
\end{aligned} \tag{A.8}$$

Hierbij moeten we de inwendige variantie minimaliseren. Om niet elke keer de inwendige variantie volledig opnieuw te moeten berekenen voor elke drempel, gebruiken we de Otsu methode:

$$\begin{aligned}
n_l(\delta + 1) &= n_l(\delta) + n_\delta \\
n_h(\delta + 1) &= n_h(\delta) - n_\delta \\
\mu_l(\delta + 1) &= \frac{\mu_l(\delta)n_l(\delta) + \delta n_\delta}{n_l(\delta + 1)} \\
\mu_h(\delta + 1) &= \frac{\mu_h(\delta)n_h(\delta) - \delta n_\delta}{n_h(\delta + 1)}
\end{aligned} \tag{A.9}$$

met n_δ het aantal elementen die bij een nieuwe drempel van cluster veranderen.

Bijlage B

De Kanade-Lucas-Tomasi tracker

De Kanade-Lucas-Tomasi tracker (zie paragraaf 3.3.1) wordt gebruikt om bepaalde punten of objecten te volgen in een beeldsequentie. Dit gebeurt door goede controlevensters te kiezen en ze terug te vinden op het volgend beeld. De volledige afleiding voor de tracker wordt nu gegeven.

De dissimilariteit tussen twee vensters, één uit beeld I_{i-1} en één uit I_i , wordt gegeven door de volgende symmetrische definitie:

$$\epsilon = \iint_W \left[I_i\left(x + \frac{d_x}{2}, y + \frac{d_y}{2}\right) - I_{i-1}\left(x - \frac{d_x}{2}, y - \frac{d_y}{2}\right) \right]^2 w(x, y) dx dy \quad (\text{B.1})$$

waarbij (d_x, d_y) de translatiecoördinaten zijn, W is het gebied van het venster en $w(x, y)$ is een gewichtsfunctie (meestal gelijkgesteld aan 1).

De Taylorreeks van I_i wordt afgebroken rond punt (a_x, a_y) met het behoud van de lineaire termen:

$$I_i(\xi_x, \xi_y) \approx I_i(a_x, a_y) + (\xi_x - a_x) \frac{\partial I_i}{\partial x}(a_x, a_y) + (\xi_y - a_y) \frac{\partial I_i}{\partial y}(a_x, a_y) \quad (\text{B.2})$$

Stellen we nu $(\xi_x, \xi_y) = (x + d_x, y + d_y)$ en $(a_x, a_y) = (x, y)$, dan krijgen we:

$$I_i(x + d_x, y + d_y) \approx I_i(x, y) + \frac{d_x}{2} \frac{\partial I_i}{\partial x}(x, y) + \frac{d_y}{2} \frac{\partial I_i}{\partial y}(x, y) \quad (\text{B.3})$$

Analoog voor I_{i-1} :

$$I_{i-1}(x - d_x, y - d_y) \approx I_{i-1}(x, y) - \frac{d_x}{2} \frac{\partial I_{i-1}}{\partial x}(x, y) - \frac{d_y}{2} \frac{\partial I_{i-1}}{\partial y}(x, y) \quad (\text{B.4})$$

Stel $\mathbf{x} = [x, y]^T$ en $\mathbf{d} = [d_x, d_y]^T$, dan geldt:

$$\begin{aligned} \frac{\partial \epsilon}{\partial \mathbf{d}} &= 2 \iint_W \left[I_i\left(\mathbf{x} + \frac{\mathbf{d}}{2}\right) - I_{i-1}\left(\mathbf{x} - \frac{\mathbf{d}}{2}\right) \right] \left[\frac{\partial I_i(\mathbf{x} + \frac{\mathbf{d}}{2})}{\partial \mathbf{d}} - \frac{\partial I_{i-1}(\mathbf{x} - \frac{\mathbf{d}}{2})}{\partial \mathbf{d}} \right] w(\mathbf{x}) d\mathbf{x} \\ &\approx \iint_W \left[I_i(\mathbf{x}) - I_{i-1}(\mathbf{x}) + \frac{1}{2} \mathbf{g}^T \mathbf{d} \right] \mathbf{g}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \end{aligned} \quad (\text{B.5})$$

met

$$\mathbf{g} = \begin{bmatrix} \frac{\partial}{\partial x}(I_{i-1} + I_i) \\ \frac{\partial}{\partial y}(I_{i-1} + I_i) \end{bmatrix} \quad (\text{B.6})$$

Om de verschuiving $\mathbf{d} = [d_x, d_y]^T$ te vinden, stellen we de afgeleide gelijk aan nul:

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = \iint_W \left[I_i(\mathbf{x}) - I_{i-1}(\mathbf{x}) + \frac{1}{2} \mathbf{g}^T \mathbf{d} \right] \mathbf{g}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} = 0 \quad (\text{B.7})$$

Herschikken we de termen:

$$\begin{aligned} \iint_W [I_i(\mathbf{x}) - I_{i-1}(\mathbf{x})] \mathbf{g}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} &= - \iint_W \frac{1}{2} \mathbf{g}^T(\mathbf{x}) \mathbf{d} \mathbf{g}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \\ &= -\frac{1}{2} \left[\iint_W \mathbf{g}(\mathbf{x}) \mathbf{g}^T(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \right] \mathbf{d} \end{aligned} \quad (\text{B.8})$$

We moeten met andere woorden de volgende vergelijking oplossen:

$$Z \mathbf{d} = \mathbf{e} \quad (\text{B.9})$$

waarbij Z de volgende 2×2 -matrix is:

$$Z = \iint_W \mathbf{g}(\mathbf{x}) \mathbf{g}^T(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \quad (\text{B.10})$$

en \mathbf{e} de volgende 2×1 -vector is:

$$\mathbf{e} = 2 \iint_W [I_i(\mathbf{x}) - I_{i-1}(\mathbf{x})] \mathbf{g}(\mathbf{x}) w(\mathbf{x}) d\mathbf{x} \quad (\text{B.11})$$

Bibliografie

- [AAW01] A.A. Alatan, A.N. Akansu, and W. Wolf. Multi-modal dialog scene detection using hidden markov models for content-based multimedia indexing. *Multimedia Tools and Applications*, 14:137–151, 2001.
- [AL96] G. Ananger and T.D.C. Little. A survey of technologies for parsing and indexing digital video. *Journal of Visual Communication and Image Representation*, 7(1):28–43, 1996.
- [BBG⁺01] J. Baan, A. Van Ballegooij, J.-M. Geusebroek, D. Hiemstra, J. den Hartog, J. List, C. Snoek, I. Patrasand, S. Raaijmakers, L. Todoran, J. Vendrig, A. De Vries, T. Westerveld, and M. Worring. Lazy users and automatic video retrieval tools in (the) lowlands. In *10th Text Retrieval Conference*, The Netherlands, 2001.
- [BGGU00] J. Boreczky, A. Girgensohn, G. Golovchinsky, and S. Uchihashi. An interactive comic book presentation for exploring video. *SIGCHI conference on Human factors in computing systems*, pages 185–192, April 2000.
- [Bro92] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, January 1992.
- [BSM⁺00] P. Browne, A.F. Smeaton, N. Murphy, N. O’Connor, S. Marlow, and C. Berrut. Evaluating and combining digital video shot boundary detection algorithms. In *Irish Machine Vision and Image Processing Conference*, Belfast Northern Ireland, 2000.
- [BSP00] G. Boccignone, M. De Santo, and G. Percannella. An algorithm for video cut detection in mpeg sequences. *SPIE Conf. on Storage and Retrieval of Media Databases*, pages 523–530, January 2000.
- [CB98] X.U. Cabedo and S.K. Bhattacharjee. Shot detection tools in digital video. *Noblesse Workshop on Non-Linear Model Based Image Analysis*, pages 231–236, July 1998.
- [CF01] M. Cooper and J. Foote. Scene boundary detection via video self-similarity analysis. *IEEE International Conference on Image Processing*, pages 378–381, 2001.
- [CI01] J. Calic and E. Izquierdo. Towards real-time shot detection in the mpeg-compressed domain. In *Workshop on Image Analysis for Multimedia Interactive Services*, Finland, May 2001.

- [CNS02] P. Campisi, A. Neri, and L. Sorigi. Wipe effect detection for video sequences. *14th IEEE International Conf. on Digital signal Processing*, July 2002.
- [CSZK01] S.-C. Chen, M.-L. Shyu, C.-C. Zhang, and R.L. Kashyap. Video scene change detection method using unsupervised segmentation and object tracking. *International Conference on Multimedia and Expo*, pages 57–60, August 2001.
- [DA00] M.S. Drew and J. Au. Video keyframe production by efficient clustering of compressed chromaticity signatures. *8th ACM international conference on Multimedia*, 2000.
- [Dav98] J. Davis. Mosaics of scenes with moving objects. *IEEE Proc. Computer Vision and Pattern Recognition*, pages 354–360, June 1998.
- [DKD98] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. *ACM Multimedia 98*, pages 211–218, 1998.
- [DZ01] H.J. Zhang D. Zhang, W. Qi. A new shot boundary detection algorithm. *IEEE Pacific Rim Conference on Multimedia 2001*, pages 63–70, 2001.
- [DZNL00] M.S. Drew and X. Zhong Z.-N. Li. Video dissolve and wipe detection via spatio-temporal images of chromatic histogram differences. *IEEE Int. Conf. on Image Processing*, 3:929–932, 2000.
- [FEW02] D. Farin, W. Effelsberg, and P.H.N. De With. Robust clustering-based video-summarization with integration of domain-knowledge. In *IEEE International Conference on Multimedia and Expo*, pages 89–92, Lausanne, August 2002.
- [FSZ95] B. Fuhrt, S.W. Smoliar, and H.J. Zhang. *Video and image processing in multimedia systems*. Kluwer Academic Publishers, 1st edition, 1995.
- [GBW01] A. Girgensohn, J. Boreczky, and L. Wilcox. Keyframe-based user interfaces for digital video. *IEEE Computer*, 34(9):61–67, 2001.
- [GFT98] B. Günsel, A.M. Ferman, and A.M. Tekalp. Temporal video segmentation using unsupervised clustering and semantic object tracking. *J. Electronic Imaging (special issue)*, 7(3):592–604, July 1998.
- [GGBZ99] J.M. Gauch, S. Gauch, S. Bouix, and X. Zhu. Real time video scene detection and classification. *Information Processing and Management*, 3:401–420, 1999.
- [GK98] U. Gargi and R. Kasturi. Performance characterization and comparison of video indexing algorithms. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 559–565, June 1998.
- [GL00] Y. Gong and X. Liu. Video summarization using singular value decomposition. *Computer Vision and Pattern Recognition*, pages 174–180, 2000.
- [HLW00] J. Huang, Z. Liu, and Y. Wang. Joint video scene segmentation and classification based on hidden markov model. In *IEEE International Conference on Multimedia and Expo*, New York USA, August 2000.

- [HO00] K.A. Hua and J.H. Oh. Detecting video shot boundaries up to 16 times faster. *8th ACM international conference on Multimedia*, pages 385–387, 2000.
- [HSE⁺95] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):729–736, July 1995.
- [HYK00] S.H. Han, K.J. Yoon, and I.S. Kweon. A new technique for shot detection and key frames selection in histogram space. *Image Processing and Image Understanding 2000*, pages 475–479, January 2000.
- [Jac96] K. Jack. *demystified: A Handbook for the digital engineer*. Hightext publications, 2nd edition, 1996.
- [JDD99] R.C. Jones, D. DeMenthon, and D.S. Doermann. Building mosaics from video using mpeg motion vectors. *7th ACM international conference on Multimedia (Part 2)*, pages 29–32, 1999.
- [JSA00] S.M. Jacob, A.H. Salden, and F.A. Aldershoff. Video summarization by dynamic scale-space and similarity clustering paradigm. *IASTED Conference on Internet and Multimedia Systems and Applications*, 2000.
- [JYL00] S.B. Jun, K. Yoon, and H.-Y. Lee. Dissolve transition detection algorithm using spatio-temporal distribution of mpeg macro-block types. *8th ACM international conference on Multimedia*, pages 391–394, 2000.
- [KC01] I. Koprinska and S. Carrato. Temporal video segmentation: a survey. *Signal Processing: Image Communication*, 16:477–500, 2001.
- [KSGA96] R. Kasturi, S.H. Strayer, U. Gargi, and S. Antani. An evaluation of color histogram based methods in video indexing. *International Workshop on Image Databases and Multi-media search*, pages 75–82, August 1996.
- [Lie99a] R. Lienhart. Abstracting home video automatically. *7th ACM international conference on Multimedia (Part 2)*, pages 37–40, April 1999.
- [Lie99b] R. Lienhart. Comparison of automatic shot boundary detection algorithms. *Storage and Retrieval for Still Image and Video Databases VII 1999*, January 1999.
- [Lie00] R. Lienhart. Dynamic video summarization of home video. *SPIE 3972: Storage and Retrieval for Media Databases*, pages 378–389, January 2000.
- [Lie01a] R. Lienhart. Reliable dissolve detection. *Storage and Retrieval for Media Databases 2001*, pages 219–230, January 2001.
- [Lie01b] R. Lienhart. Reliable transition detection in videos: a survey and practitioner’s guide. *International Journal of Image and Graphics*, 1(3):469–486, 2001.
- [LPE97] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communications of ACM*, 40(12):55–62, December 1997.

- [LS01] D. Lelescu and D. Schonfeld. Video skimming and summarization based on principal component analysis. *Management of Multimedia on the Internet, Lecture Notes in Computer Science*, pages 128–141, 2001.
- [LW00] Z.N. Li and J. Wei. Spatio-temporal joint probability images for video segmentation. *IEEE Int. Conf. on Image Processing*, 2:295–298, 2000.
- [LZ01] R. Lienhart and A. Zaccarin. A system for reliable dissolve detection in videos. In *IEEE ICIP 2001*, Thessaloniki Greece, October 2001.
- [MIP00] M.K. Mandal, F. Idris, and S. Panchanathan. Image and video indexing in the compressed domain: a critical review. *Image and Vision Computing*, 2000.
- [MPFL97] J.L. Mitchell, W.B. Pennebaker, C.E. Fogg, and D.J. LeCall. *MPEG video compression standard*. International Thomson Publishing, 1st edition, 1997.
- [NG01] A. Netzer and C. Gotzman. Mosaicing video sequences. Technical report, Technion - Israel Institute of Technology, 2001.
- [NPC99a] C.W. Ngo, T.C. Pong, and R.T. Chin. Camera breaks detection by partitioning of 2d spatio-temporal images in mpeg domain. *IEEE Multimedia Systems, International Conference on Multimedia Computing and Systems*, 1:750–755, 1999.
- [NPC99b] C.W. Ngo, T.C. Pong, and R.T. Chin. Detection of gradual transitions through temporal slice analysis. *IEEE Computer Vision and Pattern Recognition*, 1:36–41, June 1999.
- [NPZ01] C.W. Ngo, T.C. Pong, and H.J. Zhang. On clustering and retrieval of video shots. *9th ACM international conference on Multimedia*, pages 51–60, 2001.
- [OHL00] J.H. Oh, K.A. Hua, and N. Liang. A content-based scene change detection and classification technique using background tracking. *SPIE Conf. on Multimedia Computing and Networking*, pages 254–265, January 2000.
- [OSMM99] C. O’Toole, A. Smeaton, N. Murphy, and S. Marlow. Evaluation of automatic shot boundary detection on a large video test suite. In *The Challenge of Image Retrieval, 2nd UK Conference on Image Retrieval (CIR’99)*, Newcastle UK, February 1999.
- [PGG99] K.M. Pua, S.E. Gauch, and J.M. Gauch. Vidseek: dynamic multi-dimensional browsing of video archives. In *ACM SIGIR ’99 Workshop on Multimedia Indexing and Retrieval*, pages 1–17, Berkeley, August 1999.
- [PH97] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:338–343, June 1997.
- [PLE01] S. Pfeiffer, R. Lienhart, and W. Effelsberg. Scene determination based on video and audio features. *Multimedia Tools and Applications*, 15(1):59–81, 2001.

- [PLFE96] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, December 1996.
- [PS96] N.V. Patel and I.K. Sethi. Compressed video processing for cut detection. *IEEE Proc. Vision, Image and Signal Processing*, 143:315–323, October 1996.
- [PS97] N.V. Patel and I.K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):583–592, October 1997.
- [SA96] J.S. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(8):814–831, April 1996.
- [Sme02] P. De Smet. *Segmentatie en analyse van digitale beeldsequenties (Segmentation and analysis of digital image sequences)*. doctoraatsproefschrift faculteit toegepaste wetenschappen, Universiteit Gent, 2002.
- [SP95] I.K. Sethi and N.V. Patel. A statistical approach to scene change detection. *SPIE Conf. on Storage and Retrieval for Image and Video Databases*, 3:329–338, February 1995.
- [SS02] A. Shastri and R. Schowengerdt. Airborne video registration for visualization and parameter estimation of traffic flows. In *ISPRS Mid-Term Symposium*, Denver USA, November 2002.
- [SSO99] A. Smolić, T. Sikora, and J.-R. Ohm. Long-term global motion estimation and its application for sprite coding, content description, and segmentation. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(8):1227–1242, 1999.
- [Sze96] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Application*, 16(2):22–30, March 1996.
- [TAT97] Y. Taniguchi, A. Akutsu, and Y. Tonomura. Panorama excerpts: extracting and packing panoramas for video browsing. *5th ACM international conference on Multimedia*, pages 427–436, 1997.
- [TJ02] A. Tiwari and N. Jain. Video segmentation and video content analysis. Technical report, Indian Institute of Technology Kanpur, April 2002.
- [TSKR00] Y.P. Tan, D.D. Saur, S.R. Kulkarni, and P.J. Ramadge. Rapid estimation of camera motion from compressed video with application to video annotation. *IEEE Trans. on Circuits and Systems for Video Technology*, 10(1):133–146, February 2000.
- [UF99] S. Uchihashi and J. Foote. Summarizing video using a shot importance measure and a frame-packing algorithm. *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, 6:3041–3044, 1999.

- [UFGB99] S. Uchihashi, J. Foote, A. Girgensohn, and J. Boreczky. Video manga: generating semantically meaningful video summaries. *ACM Multimedia '99*, pages 383–392, November 1999.
- [WWL98] M. Wu, W. Wolf, and B. Liu. An algorithm for wipe detection. *IEEE International Conference on Image Processing*, 1:893–897, 1998.
- [YS00] A. Yilmaz and M.A. Shah. Shot detection using principal coordinate system. *IASTED Internet and Multimedia Systems and Applications Conf.*, page 168, November 2000.
- [ZMM95] R. Zabih, J. Miller, and K. Mai. Feature-based algorithms for detecting and classifying scene breaks. *ACM Multimedia 95*, pages 189–200, November 1995.
- [ZMM99] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying production effects. *Multimedia Systems*, 7:119–128, 1999.
- [ZT02] J. Zhou and W. Tavanapong. Shotweave: a shot clustering technique for story browsing for large video databases. *Multimedia Data Document Engineering*, pages 299–317, 2002.
- [ZWB⁺99] W. Zhao, J. Wang, D. Bhat, K. Sakiewicz, N. Nandhakumar, and W. Chang. Improving color based video shot detection. *IEEE Int. Conf. On Multimedia Computing and Systems*, 2:752–756, 1999.