Kwantisatie en ordening van coëfficiëntbanden in een coderingsalgoritme voor beelden

Instituut voor Perceptie Onderzoek
Postbus 513 - 5600 MB Eindhoven

17.07.1995

Rapport no. 1063

Kwantisatie en Ordening van Coëfficiëntbanden in een Coderingsalgoritme voor Beelden

L.J.A. Brünken

Kwantisatie en Ordening van Coëfficiëntbanden in een Coderingsalgoritme voor Beelden*

L.J.A Brünken

14 maart '95 - 7 juli '95

*Verslag van een Stage op het Instituut voor Perceptie Onderzoek, Eindhoven

Voorwoord

In de opleiding tot Ingenieur in de Elektrotechniek aan de Technische Universiteit Eindhoven zijn enkele stages van 200 uren opgenomen. Dit verslag is geschreven naar aanleding van zo'n 200-uurs stage aan het Instituut voor Perceptie Onderzoek (IPO) te Eindhoven. Op dit instituut, met ongeveer honderd medewerkers, vindt in de visuele groep onderzoek plaats naar subjectieve of perceptieve beeldkwaliteit. Het woord subjectief slaat op hoe de mens de beeldkwaliteit ervaart, afgezien van allerlei artistieke aspekten zoals bijvoorbeeld de scéne inhoud. Een van de onderdelen van dit onderzoek gaat uit naar het coderen van beelden. Hierbij wordt als maatstaaf de perceptieve beeldkwaliteit genomen. Er loopt onder andere een AIO-project waarbij gebruikt wordt gemaakt van Hermite transformaties (Hoofdstuk 3) om beelden te comprimeren. Bij dit project is de stage gelopen.

1 Inhoud

2 Inleiding

In figuur 1 is een algemeen schema gegeven voor het coderen van beelden. Geheel links staat het originele beeld,

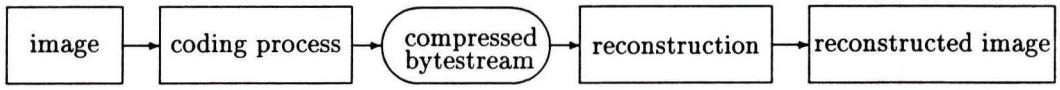


Figure 1: Algemeen schema voor het coderen van beelden

daarna het coding proces. Dit proces bestaat uit het weghalen van redundante informatie, comprimatie met behulp van correlaties tussen verschillende coëfficiënten, kwantisering enzovoort. Dit coderingsproces resulteert in een gecomprimeerde bytestream, waarna de reconstructie volgt. Deze reconstructie volgt globaal de omgekeerde weg van het coding proces. Aan het eind staat het gereconstrueerde beeld. Het algemeen doel van een beeldcompressiealgoritme is om bij een gegeven compressiefactor een zo hoog mogelijke beeldkwaliteit te halen.

Op het IPO wordt een coderingsmethode ontwikkeld waarbij perceptief belangrijke beeldstructuren (randen, lijnen) op expliciete wijze gerepresenteerd worden. Elke klasse van beeldstructuren wordt door een aantal parameters (coëfficiënten) beschreven. Compressie van een dergelijke beeldbeschrijving wordt gerealiseerd door kwantisatie van alle coëfficiënten, verwijdering van redundante informatie (door middel van predictiealgoritmen) en entropiecodering. De stage op het IPO bestond uit twee delen:

1. Het schrijven van uitbreidingen van twee bestaande programma's (polyp en polyb) die gebruikt worden bij het coderen en decoderen van beelden.
2. Het doen van experimenten met behulp van de uitgebreide programma's.

ad 1. Polyq wordt gebruikt voor kwantisatie van coëfficiënten. Elke 'klasse' van parameters vereist specifieke kwantisatiestappen.
Polyb wordt gebruikt voor de omzetting van de coëfficiënten in een bytestream, welke efficient gecodeerd kan worden. Vooral de ordening van alle coëfficiënten in zo'n bytestream heeft grote invloed op de uiteindelijke compressiefactor.
Voor polyq geldt dat er niet voldoende kwantisatie mogelijkheden per 'klasse' zijn en voor polyb geldt dat er meer bytestream ordeningen nodig zijn.

ad 2. Met de nieuwe opties zijn nieuwe strategiën mogelijk. Van deze nieuwe strategieën is nu bekend of ze beter zijn dan de oude mogelijkheden en welke het beste zijn.

Verder zijn nu bekend wat de optimale waarden van een aantal andere parameters zijn. Zoals bijvoorbeeld de spreidingsfactor van een Gaussisch window wat gebruikt wordt.
Het meten bestond voornamelijk uit het meten van haalbare signaal-ruisverhoudingen (PSNR) bij zowel lage als extreem hoge compressiefactoren.

3 Hermite Transformatie

Bij beeldcodering worden beelden meestal ontbonden in een aantal 'zinvolle' patronen (meestal zijn dit bepaalde frequentiepatronen). Een dergelijke ontbinding is meestal lokaal, zoals bijvoorbeeld bij de DCT transformatie, die gebruikt maakt van rechthoekige windows (8×8) om beelden te lokaliseren. De window vermenigvuldiging moet een aantal malen herhaald worden, om het gehele beeld te beschrijven. De grootte van dit window bepaalt het aantal patronen die de data weergeven. De vorm van het window bepaalt het gewicht dat aan elk pixel wordt toegekend.
Na de window vermenigvuldiging wordt de data uit de vermenigvuldiging verder verwerkt. De parameters voor de verwerking van deze data moeten van te voren vast gesteld zijn. Voor het vaststellen van deze parameters kan worden geput uit de theorie. Omdat dit zeer moeilijk is, wordt het menselijk oog vaak[1] als graadmeter gebruikt.
Een nadeel van de rechthoekige niet-overlappende widows is dat bij een hoge compressie het gereconstueerde beeld blokkerig overkomt (zogenaamde "blocking artifacts"). Deze zijn erg storend voor het menselijk oog. In de Hermite transformatie worden dan ook Gaussische windows gebruikt. Deze hebben de vorm van een Gauss-functie, zie figuur 2.

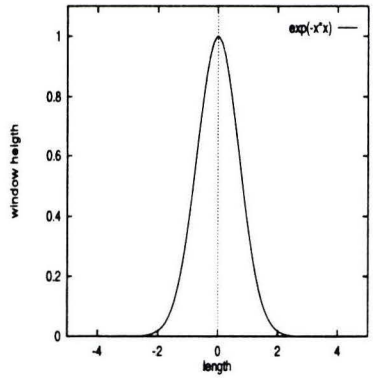


Figure 2: Algemene vorm van Gauss-functie en window.

Bij hoge compressie treden nu minder snel "blocking" artifacten op. Wel is het zo dat een Hermite transformatie tot gereconstrueerde beelden leidt die minder scherp zijn (door het ontbreken van de hogere frequentiebanden).

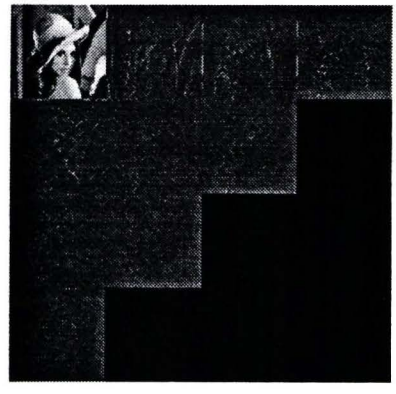


Figure 3: Hermite beelddecompositie van orde '3', ts=4, σ=1.2, window length=7, beeld Lena

In figuur 3 staat een voorbeeld van een zogenaamde multiband image of beelddecompositie. De informatie is verspreid over verschillende (frequentie) banden. Het aantal banden is afhankelijk van de ingestelde orde. De banden bestaan

[1] En natuurlijk op het IPO ook

uit verschillende oplopende frequenties. De laagste frequenties staan in de band links bovenaan. De frequenties lopen dan van links naar rechts en van boven naar beneden op. De frequentiebanden in de onderste diagonaal bevatten dus de hoogste frequenties.
Na een rotatie (na de tweede polyt in figuur 4) die beelden lokaal roteert zijn de banden zijn in drie 'klassen' te verdelen. De band met de laagste frequentie wordt de DC band (geheel links bovenaan) genoemd. De rij banden rechts ervan, de 1D banden. De tweede rij en alles eronder zijn de de 2D banden. De DC band bestaat uit DC coëfficiënten, de 1D banden uit 1D low en 1D high coëfficiënten, de 2D banden uit 2D coëfficiënten. Het verschil tussen 1D low en 1D high is dat de 1D low coëfficiënten veel minder energie hebben dan de 1D high coëfficiënten. Hierdoor is de invloed van de 1D low coëfficiënten op de kwaliteit van het gereconstrueerde beeld veel kleiner.

Na de Hermite transformatie ziet het beeld er dus uit zoals in figuur 3. Het aantal banden in een dergelijke beelddecompositie hangt af van de orde van de uitgevoerde transformatie. Na het uitvoeren van een Hermite transformatie moeten de coefficienten nog verdert bewerkt worden om tot een zgn. gecomprimeerde bytestream te komen. Dit houdt onder andere in dat de coëfficiënten moeten worden gekwantiseerd (voor compressie) en dat de frequentiebanden in een bepaalde volgorde in de bytestream moeten komen. Deze en nog meer bewerkingen worden in het volgende hoofdstuk beschreven.

4 Cantata

Bij het IPO maken ze gebruik van het programma Cantata om een beeld te comprimeren. Met dit programma kan men verschillende subprogramma's representeren met behulp van blokjes. De in- en uitvoer van elk blokje wordt gerepresenteerd als een lijn. Zie figuur 4. De parameters van elk programma zijn per blokje in te stellen via een user interface.

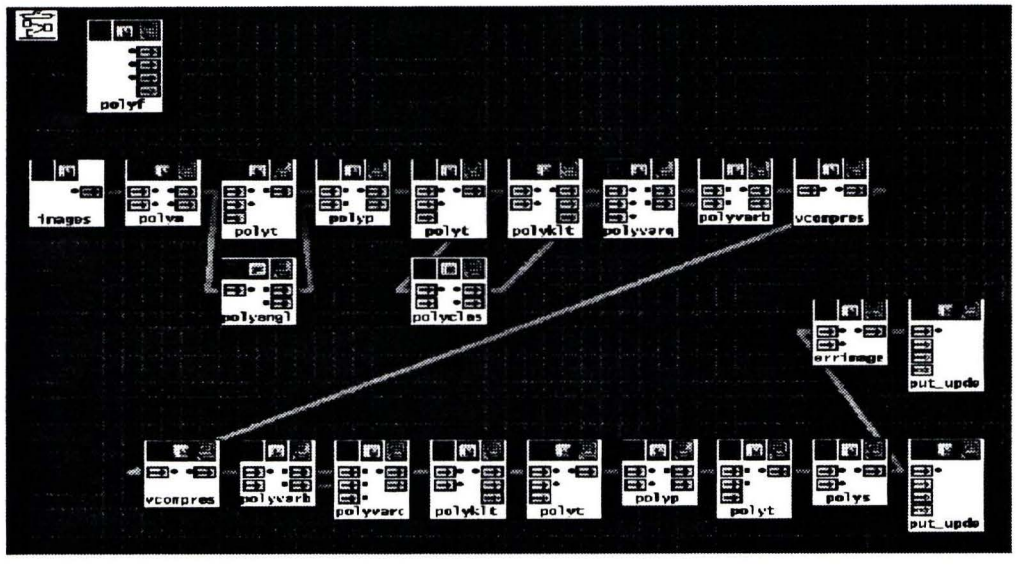


Figure 4: Cantata met alle gebruikte subprogramma's

Geheel links bovenaan in figuur 4 staat het originele plaatje (images). Tussen de bovenste rij en de onderste rij loopt een lijn, welke de gecomprimeerde bytestream voorstelt. De meeste blokjes in de bovenste rij komen ook voor in de onderste rij. Dit komt omdat de bytestream via dezelfde weg teruggetransformeerd moet worden. Geheel rechts onderaan komt het gereconstrueerde plaatje weer tevoorschijn (na polys). Aangezien de reconstructie de omgekeerde weg van de codering moet volgen, ziet men een bijna perfecte symmetrie in de volgorde van blokjes in de codering en de decodering. Hierna volgt van elk van de blokjes een korte beschrijving.

- **images** : Hieruit komt het originele plaatje. Alle resultaten in dit verslag zijn verkregen met het plaatje "Lena" (8 bytes per pixel grijswaarde afmeting 512x512 pixels).
- **polyf** : In dit blokje wordt de filterbank gevormd die nodig is voor het uitvoeren van een zogenaamde cartesische Hermite transformatie. Hier worden onder andere de spreidingsparameter sigma, de window lengte en de orde van de transformatie ingegeven.
- **polya** : Hier worden de filters uit polyf toegepast op het originele beeld. Uit polya komt dus een multiband beelddecompositie (zie figuur 3).
- **polyangle** : Dit blokje berekent voor elke windowpositie de hoek waarin zich de meeste signaalenergie bevindt. Door deze hoek apart te coderen en alle lokale signalen over deze hoek te roteren kan een extra energiecompactie worden verkregen.
- **polyt** : Polyt voert een linaire transformatie uit op de coëfficiënten van de cartesische Hermite transformatie. Hierdoor wordt de cartesische vorm van de Hermite transformatie omgezet naar de polaire vorm. Het voordeel van deze polaire vorm is dat er makkelijker signaalrotaties op gedaan kunnen worden.

- **polyp** : Hier wordt via een rotatie op de coëfficiënten van de polaire Hermite transformatie de meeste energie naar de 1D-banden verplaatst. Dit is aantrekkelijk doordat meer energie over minder banden verspreid wordt. Hierdoor is bijvoorbeeld een hogere kwantisatie van de 2D banden minder slecht voor de signaal-ruis verhouding en dus ook voor de perceptieve kwaliteit. Nadat in polyp een rotatie is uitgevoerd moet de polaire beelddecompositie weer worden terug-getransformeerd naar de cartesische vorm (nogmaals polyp)

- **polyklt** : In polyklt wordt een KLT-transformatie uitgevoerd, die tot doel heeft een nog hogere energiecompactie te bereiken (door gebruikt te maken van correlaties tussen de verschillende coëfficiëntbanden). De uitvoer van polyklt is een beelddecompostie die direct naar polyclass en polyvarq gaat.

- **polyclass** : Met behulp van het blokje polyclass wordt een zogenaamd classificatiebeeld aangemaakt. Dit beeld bestaat uit vijf banden, respectievelijk de DC, de 1D low, de 1D high en de 2D band genoemd. De vijfde band is voor overige classificaties bedoeld. Voor de DC-posities geldt dat er weinig energie in alle hogere banden dan de eerste zit. Voor 1D-posities geldt dat er weinig energie in hogere banden dan de 1D-banden, waarbij 1D low veel minder energie heeft dan 1D high. Voor 2D-coëfficiënten geldt dat in alle banden significant veel energie zit. De energie drempels kunnen bepaald worden in de user interface van polyclass.
  Polyclass bepaalt voor elke positie tot welke klasse deze behoort en zet in de juiste band van het classificatie beeld op dezelfde positie een '1' en in de overige banden een '0'. Dit classificatie beeld gaat direct naar polyvarq.

- **polyvarq** : Polyq (het originele programma) kwantiseert de beeldcompositie die uit polyklt komt. Dit houdt het volgende in. Stel de kwantisatiestap is 2. Nu worden de originele waarde van de coëfficiënten die gekwantiseerd worden gedeeld door 2 en afgerond op een hele waarden. Dit gebeurt voor alle coëfficiënten in alle banden. Bij het decomprimeren van het plaatje wordt nu de waarde van de coëfficiënt vermenigvuldigt met 2. Het eindresultaat hoeft natuurlijk niet hetzelfde te zijn als de beginwaarde[2]. Het kwantiseren introduceert dus fouten.

- **polyvarb** : Polyvarb zet een gekwantiseerde beeldcompositie om in een bytestream. Hierin moeten dus alle coëfficiënt-waarden, orientaties en eventueel nog andere informatie betreffende orde, sampling distance etc. worden opgenomen. Om de banden in de bytestream te zetten, zijn verschillende mogelijkheden ter beschikking. Polyvarb is een uitbreiding op het oorspronkelijke programma polyb.

- **vcompress** Dit blokje comprimeert de bytestream uit poly(var)b. Het haalt zoveel mogelijk redundantie uit de bytestream. Hiervoor zijn verschillende algoritmen voorhanden. Het gebruikte algoritme voor alle experimenten is arithmetisch (comp2).

Voor de reconstructie van het originele plaatje zijn vrijwel dezelfde blokken nodig. Men ziet dus de meeste blokjes terug komen in de omgedraaide volgorde na de gecomprimeerde bytestream (na de eerste Vcompress). Het blokje polyclass is vervallen omdat het niet nodig is om nog een classificatie te maken; polys is de tegenhanger van polya.

[2] $\lceil(11/2)\rceil*2=10 \neq 11$

5 Polyvarq

Het nieuwe programma polyvarq is een uitbreiding van polyq. De input en de output van polyvarq zijn:
Input:

1. Input Beelddecompositie
2. Input Classificatiebeeld
3. Kwantiestappen

Output:

1. Output Beelddecompositie
2. Output Classificatiebeeld

Hieronder volgen de verbeteringen gemaakt gedurende de stage. De grootste tekortkoming van polyq is dat de banden niet classificatie afhankelijk gekwantiseerd kunnen worden. Polyvarq kan classificatie-afhankelijk kwantiseren, waarvoor dus het classificatie multiband image als invoer nodig is. Aangezien de kwantisatiestap voor elke band anders kan zijn, kijkt het programma naar een ascii file waar de stappen geordend staan. Zo'n bestand ziet er bijvoorbeeld uit zoals in figuur 5. In deze ascii-file wordt dus per band aangegeven hoe groot de kwantisatiestap is. In figuur

```
  2 2 2 2
1 3 3 3 3
4 4 4 4
4 4 4
4 4
4
```

Figure 5: Kwantisatiestappen ascii file

5 geldt voor de DC-band een stap van 1, de 1D low banden een stap van 2, de 1D high banden een stap van 3 en voor de de 2D banden een stap van 4. De kwantisatiestappen staan in dezelfde configuratie als de banden in de beeldecompositie (figuur 3), behalve de 1D low stappen, die er apart boven staan.
Per classificatie kan de kwantisatiestap opgeschaald worden in de user interface. Indien de schalen allemaal op '1' staan, zijn de kwantisatiestappen voor elke band zoals in de ascii file (figuur 5). Indien men de 1D low coëfficiënten met kwantisatiestap '4' wil kwantiseren, hoeft men allen de schaal in de user interface op '2' te zetten. Zo kan men dus makkelijk in de user interface bepalen wat de kwantisatiestappen voor elke classificatie zijn. Bij de experimenten uitgevoerd gedurende de stage bestond de ascii file geheel uit '1'-en. De waarde van de schalen in de user interface zijn dan automatisch gelijk aan de kwantisatiestappen.

Het programma heeft verder drie kwantisatie modi, te weten :

1. 'data → int' : In deze mode worden de coëfficiënten (floats) gekwantiseerd, en daarna gedeeld door de desbetreffende stapgrootte zodat de gekwantiseerde coefficienten worden afgebeeld op integer waarden.
2. 'int → data' : In deze mode worden de integers weer teruggebracht op hun originele pixel-waarde. Dit lukt niet helemaal want in mode 'data→ int' gaat informatie verloren door afronding.

3. 'data → data': Hier worden de coëfficiëntwaarden gewoon gekwantiseerd. Logischerwijs is dit dus ook niet lossless.

Het gehele kwantisering algoritme ziet er dan ongeveer uit zoals in figuur 6.

```
repeat
  GetNextPixel;
  GetClass(class);
  stap:=GetClassDependedStap(class);
  case mode of
    'data→ int' : pixel:=pixel/stap;
    'int→ data' : pixel:=pixel*stap;
    'data→ data': pixel:=(pixel/stap)*stap;
  end case;
until AllPixelsDone;
```

Figure 6: Kwantisatie algoritme

Bij het kwantiseren kan het voorkomen dat van een positie bijvoorbeeld alle 2D coëfficiënten op nul komen[3]. Een positie die bijvoorbeeld 2D was, kan zo overgaan in een positie die 1D-high is, of zelfs 0D (DC). Met dezelfde reden kan 1D (low en high) overgaan in DC. Polyvarq geeft als output dus naast het gekwantiseerde beeld ook een classificatie multiband image waarin de actuele classificaties staan.

Indien men de classificatie niet wil gebruiken dient men eigenhandig de stappen voor de verschillende klassen gelijk te zetten.

[3] pixelwaarde=2, stap=4 $\overset{kwantisatie}{\Rightarrow}$ integer=0 $\overset{reconstructie}{\Rightarrow}$ reconstructie pixelwaarde =0 !

6 Polyvarb

Polyvarb is een uitbreiding van polyb. Er zijn twee grote verbeteringen aangebracht:

1. Het orientatiebeeld wordt nu ook in de bytestream opgenomen.
2. Het classificatie beeld kan nu gebruikt worden om te bepalen hoe de precieze structuur van de bytestream wordt.

De input en output van polyvarb zijn:
Input:

1. Gekwantiseerde Beelddecompositie
2. Classificatiebeeld

Output:

1. Bytestream

Bij de decodering/reconstructie zijn de in- en output omgedraaid. Verder is het classificatie beeld niet nodig indien gebruikt wordt gemaakt van classificatie onafhankelijke ordeningen/methoden. Polyvarb zet de banden van een beeldecompositie om in een bytestream. Ook de orientatieband en eventueel ook de classificatie komen in de bytestream.
Het classificatiebeeld bestaat uit 5 banden die voornamelijk (totaal 80 % '0'-en) uit nullen bestaan. Om het aantal bytes die nodig zijn om de classificatie te representeren terug te brengen, zijn de vijf banden teruggebracht naar een band. In deze ene band staat een '0' indien de bijbehorende pixel alleen een DC coëfficiënt heeft, een '1' voor 1D low, een '2' voor 1D high, etc. Dit getal is gelijk aan het nummer van de band van het oorspronkelijke classificatie multiband image. Ter verduidelijking staat de programmacode voor deze conversie in figuur 7 in C. Eventueel kan er over deze ene band nog een predictie algoritme gehaald worden. Dit is **niet** gedaan in de eigenlijke programma code.

```
for (i=0; i< 5 ; i++)
  clpntr[i]=(float *)class->imagedata+i*nsize;

for (i=0 ; i< 5 ; i++)
  (for (k=0;k< nsize ; k++)
    if (*(clpntr[i]+k) == 1) *(clpntr[0]+k)=i;
  )
```

Figure 7: Programma code voor classificatie conversie

Bij de orientatie band wordt een predictie algoritme gebruikt. Hierbij wordt op het verschil van twee opeenvolgende orientaties de volgende functie toegepast:

$$f(x) = \begin{cases} x & \text{voor } -\frac{nr-orien}{2} \leq x \leq \frac{nr-orien}{2} \\ x - \frac{nr-orien}{2} & \text{voor } \frac{nr-orien}{2} < x \\ x + \frac{nr-orien}{2} & \text{voor } -\frac{nr-orien}{2} > x \end{cases} \quad (1)$$

waarin nr-orien het maximum aantal orientaties voorstelt. Functie 1 staat ook bekend onder First Order DPCM. Deze functie zorgt ervoor dat een stap tussen twee opeenvolgende hoeken zo klein mogelijk is[4]. Het grootste getal

[4]De rij hoeken (maximum aantal hoeken = nr-orien =4) 230213 wordt 2(+1)(+1)(+2)(-1)(+2)

dat nu kan voorkomen in het orientatie gedeelte van de bytestream is nu $\frac{nr-orien}{2}$.
De DC band wordt doorgegeven door telkens het verschil van opeenvolgende DC-coëfficiënten in de bytestream te zetten. De volgorde van de rest van de banden worden doorgegeven met verschillende methoden (in te stellen via de user interface) Hier worden de oude en de nieuwe methoden kort uitgelegd. De volgende methoden waren reeds in de originele polyb beschikbaar:

1. **DC+AC coëfficiënten per band**. Hier worden per band de coëffciëntwaarden in de bytestream gezet.
2. **DC+AC coëfficiënten per positie**. Hier worden per positie de coëfficiëntwaarden uit alle banden in de bytestream gezet.
3. **DC+non-zero AC coëfficiënten per positie**. Per positie wordt bekeken of de laatste waarden van de AC-coëfficiënten uit nullen bestaat. Zo ja, dan wordt er in plaats van die rij nullen een 'end-of-block' teken in de bytestream gezet. Bij het reconstrueren wordt er speciaal op het 'end-of-block' teken gelet.
   Aangezien er bij hoge kwantisatie er veel nullen in het multiband image voorkomen, zullen er veel rijen nullen voorkomen. En omdat er over het algemeen minder hoge coëfficiëntwaarden in de hogere banden voorkomen, zullen de rijen nullen zich juist daar manifesteren.
4. **All non-zero coëfficiënten per positie**. Nu wordt ook gekeken of de rij nullen al begint in de DC band.
5. **DC band + 1D coëfficiënten**. Hier worden geen 2D coëfficiënten doorgegeven. Deze methode werkt per band (vergelijkbaar met 1).

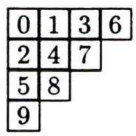
De volgende methoden zijn nieuw:

6. **DC+AC per position classification depended**. Hier kijkt men naar de classificatie van elke positie. Eerst worden alle DC-coëfficiënten doorgegeven, daarna alle waarden van alle 1D low coëfficiënten, dan 1D high coëfficiënten en als laatste alle waarden van de 2D coëfficiënten. Bij de reconstructie moeten de juiste waarden weer bij in de juiste positie in de juiste band komen. Hiervoor moet men dus weten hoeveel waarden elke positie nodig heeft. Voor DC posities geldt dat er een waarde is, voor 1D posititities geldt dat het aantal waarden gelijk is aan de orde plus een DC coëfficiëntswaarde. 2D posities hebben nog meer waarden nodig. De classificatie is dus nodig, waaruit blijkt dat de classificatie voor de pixels in de bytestream moet komen.
7. **DC+AC per position non-zero class depended**. Hetzelfde als hierboven met als bijzonderheid dat indien een positie in de laatste banden allemaal '0'-en als waarde heeft dan wordt die rij '0'-en weergegeven door de waarde '128' (EOB).
8. **DC+AC per band classification depended**. Ook hier kijkt men naar de classificatie van elke positie. Men gaat eerst van alle 1D low coëfficiënten de eerste 1D band doorgeven, daarna van alle 1D low coëfficiënten de tweede 1D band, etc. Daarna zijn de 1D high posities aan de beurt met dezelfde methode en dan de 2D posities. Bij de reconstructie is dus ook hier de classificatie nodig.
9. **DC+AC bands +class + orien**. Hier worden alle waarden in het multiband image doorgegeven zonder ergens op te letten. Tussen de DC band en de rest van de banden komen de classificatie en de orientatie[5].

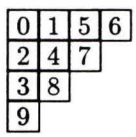
De orientaties worden indien mogelijk direct na de DC-band doorgegeven. Anders worden zij voor de coëfficiënten doorgegeven. Dit omdat het makkelijker is om de hogere waarden bij elkaar te hebbben voor te (redundantie) comprimeren. De orde en andere informatie worden geheel aan het begin van de bytestream gezet, dit omdat deze informatie nodig is om de goede lengtes en aantal te lezen banden te bepalen.
De volgorde van de 1D en 2D banden in de bytestream, kan ingesteld worden in de user interface (coefficients order). We hebben drie mogelijkheden.

[5]Een nadeel van deze methode is dat erg veel coëfficiënten worden doorgegeven, waardoor de redundante compressie erg veel CPU tijd vergt

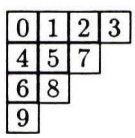
- Initial ordening:



- ZigZag (DCT like):



- DC,1D and 2D diagonal:



De figuren geven de volgorde van de banden in de bytestream weer, van band 0 (de DC band) tot en met band 9.

De volgorde van de verschillende onderdelen in de bytestream is : DC, eventueel de classificatie[6], de orientatie en daarna de DC-banden in een bepaalde ordening (zie hierboven) en met een bepaalde methode.

De classificatie afhankelijke omzettingen zijn **niet** lossless. Dit komt doordat bij de classificering pixels met heel weinig (maar wel wat) energie in de hogere banden (1D en 2D) toch geclassificeerd worden als DC. Deze lage waarden in 1D en 2D banden worden dus niet doorgegeven in de bytestream. In tegenstelling tot de modes waarbij alle pixels in de banden worden doorgegeven (ongeacht de classificatie). Hetzelfde geldt ook voor pixels die als 1D (low of high) worden geclassificeerd maar toch wat energie hebben in de 2D banden. De drempels hoeveel energie bepalend is voor welke classificatie, kunnen worden ingesteld in de user interface van polyclass.

[6]Niet bij classificatie onafhankelijke methoden

7 Experimenten

Om beter inzicht te krijgen in de effecten van verschillende parameters in het codeerschema en effecten van codeerstragieën, bestaat het tweede gedeelte van de stage uit het doen van experimenten/metingen met het in hoofdstuk 4 beschreven compressiealgoritme. Bij deze experimenten/metingen wordt telkens een parameter verandert en de rest van de parameters vastgehouden. De parameter die telkens wordt verandert is ook de titel van de paragraaf.

7.1 Orientaties

Het programma/blokje polyangle vindt per windowpositie een hoek waarin de meeste energie ligt. Nu is de vraag of het nodig is om de hoek weer te geven in graden of dat het niet veel uitmaakt dat er grotere stappen tussen de hoeken zit. Om dit te testen zijn dus metingen uitgevoerd waarbij het aantal hoeken gevariëerd werd (1,2,4,8,16,32 en 64 hoeken). In figuur 8 staan de hoeken voor de situatie waarbij het maximaal aantal hoeken vier is. Alle hoeken liggen tussen de 0° en de 180°. Dit komt doordat de hoeken geen richting hoeven te hebben, dat wil zeggen dat 45° gelijk is aan 225°.

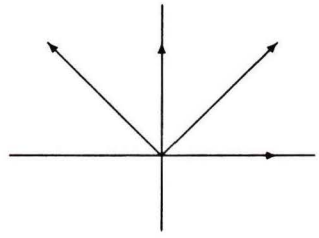


Figure 8: Vier hoekrichtingen

De resultaten van de metingen staan in figuren 9 en 10. In figuur 9 staat de signaal ruis verhouding (PSNR) uit tegen het maximaal aantal hoeken, voor sampling afstand (ts) 4 en 8. In figuur 10 staat links het aantal benodigde bytes in de gecomprimeerde bytestream (na vcompress) voor de orientaties. Rechts in figuur 10 staat hetzelfde als links maar nu voor ts=8.

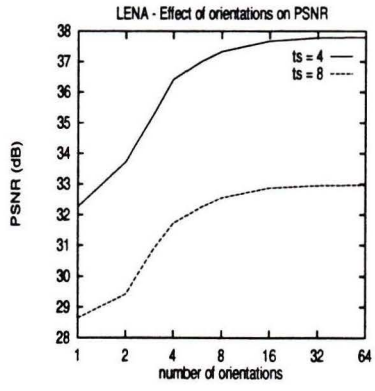


Figure 9: Signaal ruis verhouding tegen het maximale aantal hoeken

Zoals in figuur 9 te zien is, lijkt het het beste om het maximale aantal hoeken op 8 of 16 te zetten. Na 16 hoeken gaat de signaal-ruis verhouding nog maar marginaal omhoog, terwijl er toch meer bytes nodig zijn (figuur 10).

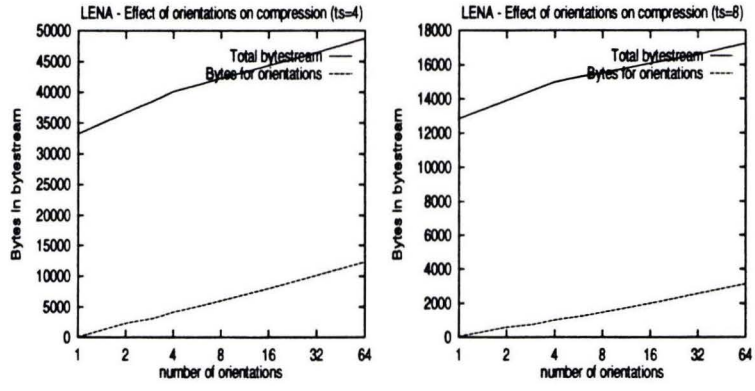


Figure 10: Aantal bytes in bytestream tegen het maximale aantal hoeken

Verrassend is dat het aantal hoeken precies logaritmisch oploopt met het aantal bytes.

7.2 Sigma

Een andere parameter is sigma, ofwel de spreidingsparameter van het Gaussische window dat in de Hermite transformatie wordt gebruikt om beelden te lokaliseren. Zie figuren 11 en 12. Sigma bepaalt dus de breedte van het Gaussisch window dat over het plaatje schuift. Bij een lage sigma (figuur 11 links) krijgen de pixels tussen twee windows in te weinig gewicht (zie figuur 11 rechts) waardoor zogenaamde 'speckles' in het gereconstrueerde beeld optreden. Bij een te hoge sigma (figuur 12 links) is zullen sommige pixels te veel gewicht krijgen (figuur 12 rechts).

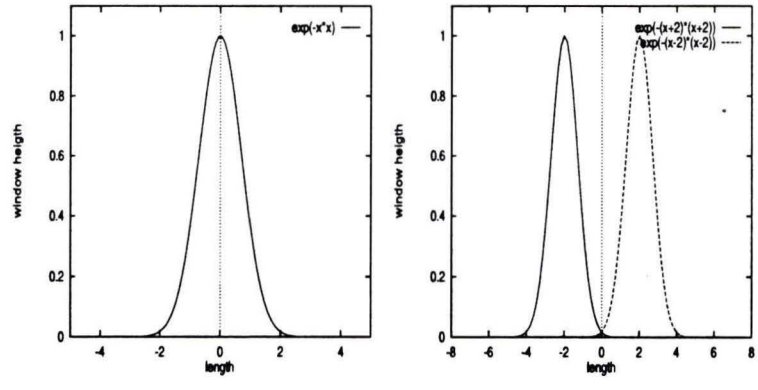


Figure 11: Gauss-functies met kleine sigma's

In figuur 13 staat sigma uitgezet tegen de signaal ruis verhouding voor verschillende ordes en verschillende sampling distances (links: ts=4 en rechts: ts=8). Alle metingen voor de bepaling van een optimale sigma zijn gedaan met een 1D-coder. Hierbij zijn dus alle coëfficiënten DC of 1D. We leiden hieruit af dat voor ts=8 (figuur 13 rechts) een sigma van ongeveer 2.4 optimaal is. Voor ts=4 (figuur 13 links) geldt een sigma van ongeveer 1.2 als optimum.
Hoe groter de sampling afstand, hoe hoger de waarde van sigma moet zijn om een optimale dekking te krijgen.

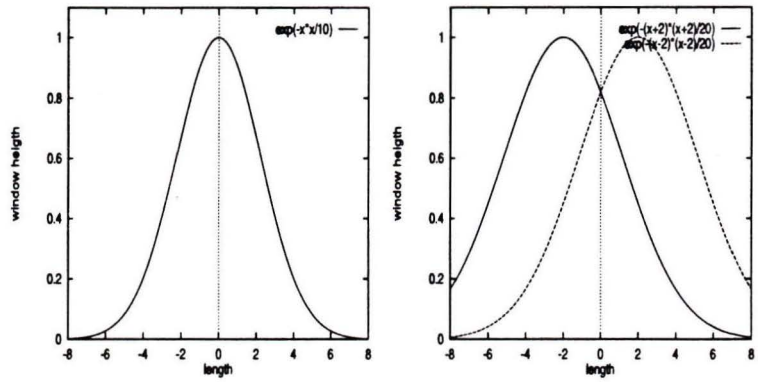


Figure 12: Gauss-functies met grote sigma's

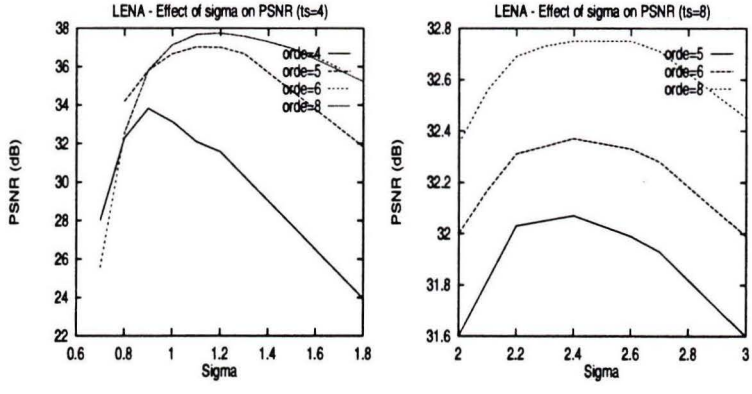


Figure 13: Signaal ruis verhouding tegen sigma voor ts=4 en 8

7.3 Polyvarq Strategie

Doordat we nu elke klasse van beeldposities (DC, 1D low, 1D high en 2D) apart willen kwantiseren, hebben we vier aparte kwantisatie stappen. Het is nu makkelijk om een kwantisatie strategie te hebben. Deze strategie moet richtlijnen geven om aan een zo gunstig (hoge PSNR, kort bytestream) mogelijke codering te komen. Hier is dus een 2D-coder gebruikt.
In de volgende 6 grafieken (figuur 14 en 15) staan telkens in de titel de kwantisatie stappen voor DC en 1D low (1Dl), en voor elke lijn in de grafiek de stappen voor 1D high en 2D gegeven als een paar onder de lijn als (1Dh, 2D).

In de eerste drie grafieken (figuur 14) staan de verschillende kwantisatie stappen uitgezet tegen de signaal ruis verhouding. Hierbij valt op dat voor het bereiken van een PSNR van bijvoorbeeld 33.2 er verschillende mogelijkheden zijn. Zoals (DC,1Dl,1Dh,2D)= (1,10,1,5), (1,10,3,3), (3,10,1,5), (3,10,3,1) en (5,10,1,3).

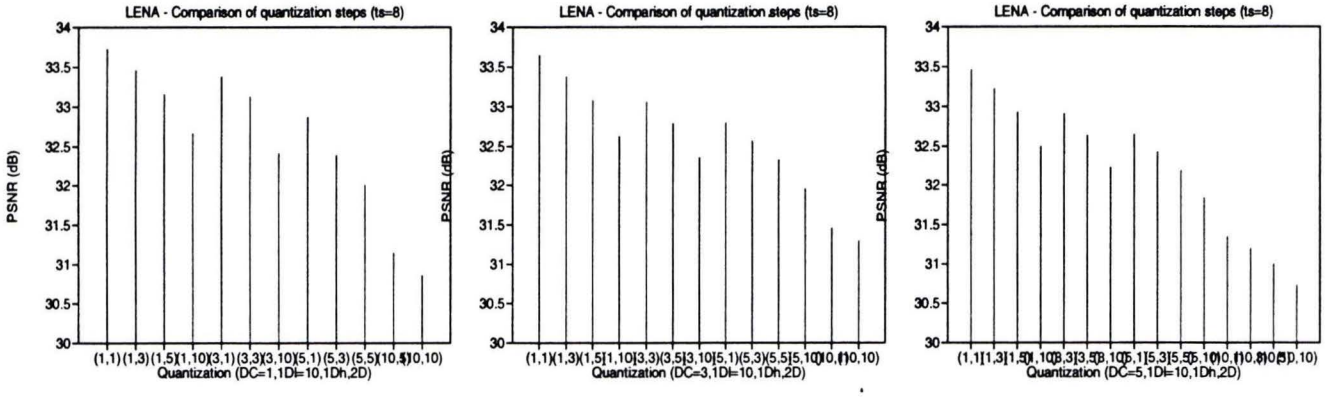


Figure 14: Signaal ruis verhouding tegen verschillende kwantisatievariaties

Wat er al uit de metingen bleek, is dat de 1D low coëfficiënten weinig (±0.5 dB) bijdragen aan de signaal ruis verhouding. Daarom is bij alle metingen als kwantisatie stap voor 1D low '10' gekozen. Hierdoor komen alle 1D low coëfficiënten op '0' te staan. De 1D low posities worden DC posities. Een nadeel van deze hoge kwantisering is dat de 1D low posities niet meer de zogenaamde "blocking" artifacten verminderen. In feite is een stap van '3' al groot genoeg om de 1D low coëfficiënten op nul te krijgen, maar om misverstanden te voorkomen is de groote van '10' gekozen. Indien men juist de perceptieve kwalitteit wil verbeteren zijn juist deze 1D low coëfficiënten nodig. Kwantiseer ze juist dan met een kleine stap.

In figuur 15 is de gecomprimeerde bytestream uitgezet tegen de verschillende kwantsaties. Hier valt op dat wanneer de 2D coëfficiënten met stapgrootte '1' worden gekwantiseerd, dat dit dan erg veel bytes kost. Dit is in te zien door te realiseren dat er veel 2D banden[7] zijn met elk veel coëfficiënten[8]. Door deze vele coëfficiënten vrijwel niet te kwantiseren, wordt de bytestream veel groter. Indien men kijkt naar de signaal ruis verhouding verlaging die een zwaardere (stap='3') kwantisering van de 2D coëfficiënten veroorzaakt, ziet men dat deze niet groot (±0.3) is. Een verkorting van de bytestream met ±15% leidt dus tot een zeer kleine daling van de PSNR. We komen dus tot de volgende richtlijn: Kwantiseer de 2D coëfficiënten liefst met grotere stappen dan '1'.

[7]Bij een orde van 4 zijn er al 10 2D banden
[8]Bij een ts van 4 zijn er 16384 coëfficiënten in elke band

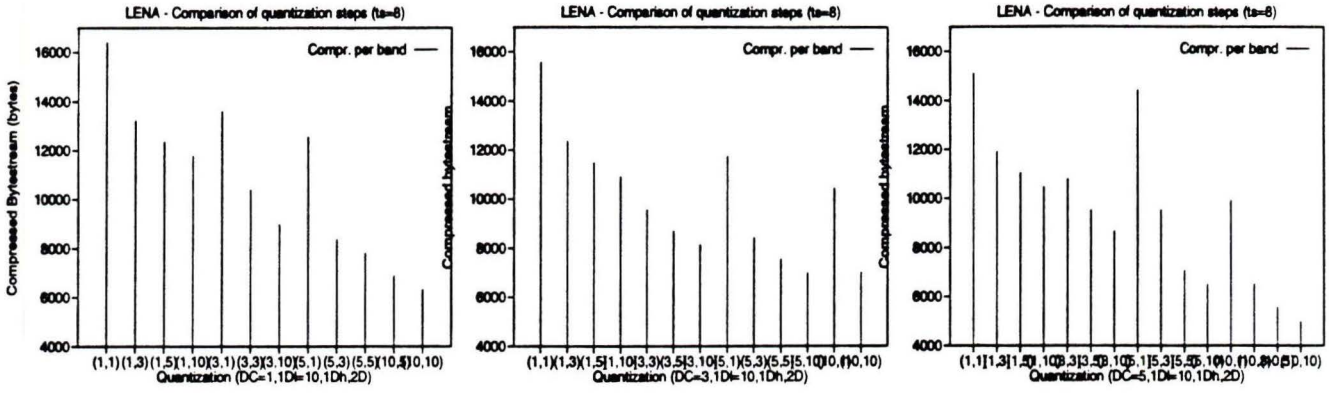


Figure 15: Gecomprimeerde bytestream tegen een aantal kwantisaties

7.4 Polyvarb Strategie

Om te kijken welke methoden het beste zijn om een gekwantiseerde beeldcompositie om te zetten in een bytestream , zijn er metingen voor een aantal bytstreamcoderingsmethoden gedaan bij verschillende kwantisatie stappen. Bij deze metingen is bij verschillende bytestream ordeningen de lengte van de gecomprimeerde bytestream (na vcompress) gemeten. Dit voor verschillende kwantisatiestappen en dus ook verschillende signaal-ruis verhoudingen.

Vergelijking van ordening
Om te bepalen welke ordening nu het beste is, en of het winst oplevert om classificatie afhankelijke ordeningen te gebruiken, is voor verschillende ordeningen de compressie ratio ($\frac{512 \times 512}{vcompress\ bytestream}$) uitgezet tegen de signaal ruis verhouding (figuur 16).
Uit figuur 16 is af te leiden dat voor hogere compressie de classificatie afhankelijke ordeningen een hogere PSNR bij gelijke compressie geven. Voor lagere compressie ratio's zijn de klassieke ordeningen beter. De oorzaak zit in het volgende: Voor hogere compressies zitten er meer '0'-en in het multiband image, en door de classificatie hoeven deze nullen niet worden meegegeven aan de bytestream want de pixels zullen veelal terugvallen naar 1D of DC. Voor lagere compressies is het gebruikt van een classificatiebeeld niet zinvol. Veel posities zullen 1D en 2D zijn, waardoor de winst klein wordt. Doordat het doorgeven van het classificatiebeeld zelf ook veel bytes kost[9] , is een klassieke methode beter.

In figuur 17 staan dezelfde parameters tegen elkaar uitgezet. We zien hier dat de classificatie afhankelijke ordeningen al bij veel lagere compressies beter zijn. Dit komt omdat de sampling distance nu '4' is in plaats van '8'. Dit heeft als gevolg dat er veel meer pixels zijn, en meer pixels met alleen een DC-waarde. Hierdoor is de winst groter dan bij ts=8.
Indien nu met grotere stappen wordt gekwantiseerd, is de winst nog veel groter. Dit komt doordat er veel meer posities terugvallen naar DC en 1D. Richtlijn: De classificatie afhankelijke methoden zijn vrijwel altijd beter.

Vergelijking van compressies
Voor de compressies van de verschillende ordeningen is te verwachten dat naarmate er meer '0'-en in de bytestream uit polyvarb zitten, de compressie van vcompress beter wordt. Om te controleren of de inhoud van de bytestream uit polyvarb ook klopte met de verwachtingen zijn nog de volgenden metingen uitgevoerd. In figuur 18 staan voor drie verschillende band-ordeningen de compressies van vcompress gegeven tegen de resulterende bytestream's.
Hierin is te zien dat 'Comp per pos' en 'Comp per band' veel beter te comprimeren is dan 'Comp per pos non-zero'. Dit is te verklaren doordat in 'Compr per pos non-zero' veel minder nullen zitten. Immers indien voor een pixel geldt dat de rest van de waarden in de banden nullen zijn, deze niet voorkomen in de bytestream. Deze bytestream wordt dan zo kort dat met behulp van de lage compressie we toch op een even lange nieuwe byestream uitkomen als de ander twee methoden in figuur 18.

[9] bij ts=4 kost het 16384 bytes

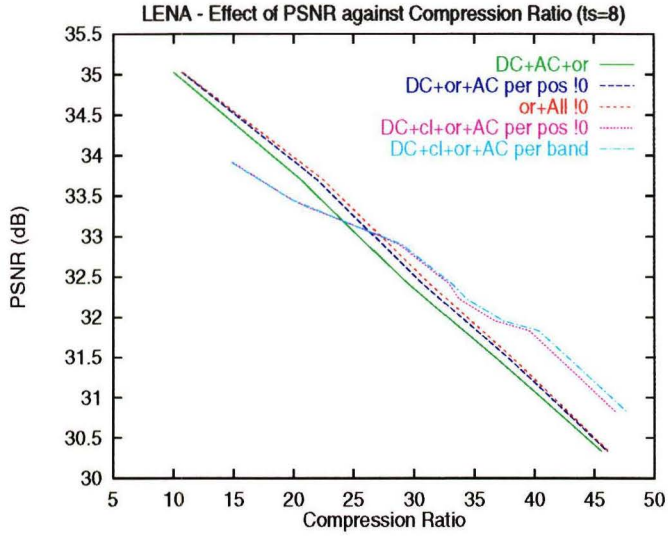


Figure 16: Compressie ratio uitgezet tegen PSNR voor verschillende soorten bytstream ordeningen

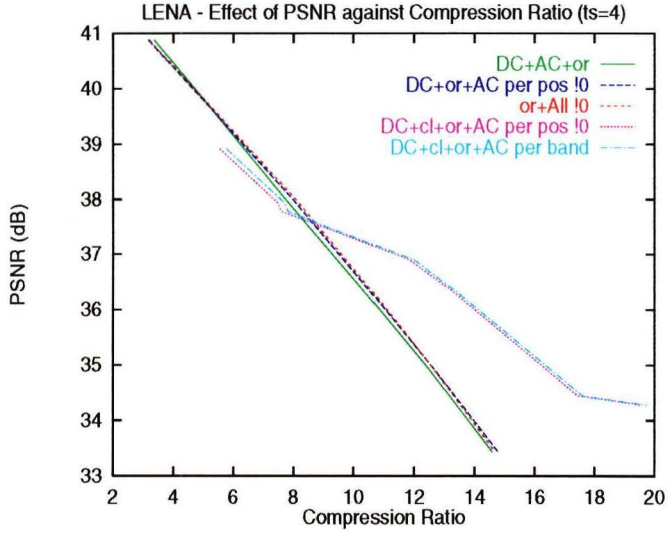


Figure 17: Compressie ratio uitgezet tegen PSNR voor verschillende soorten bytstream ordeningen

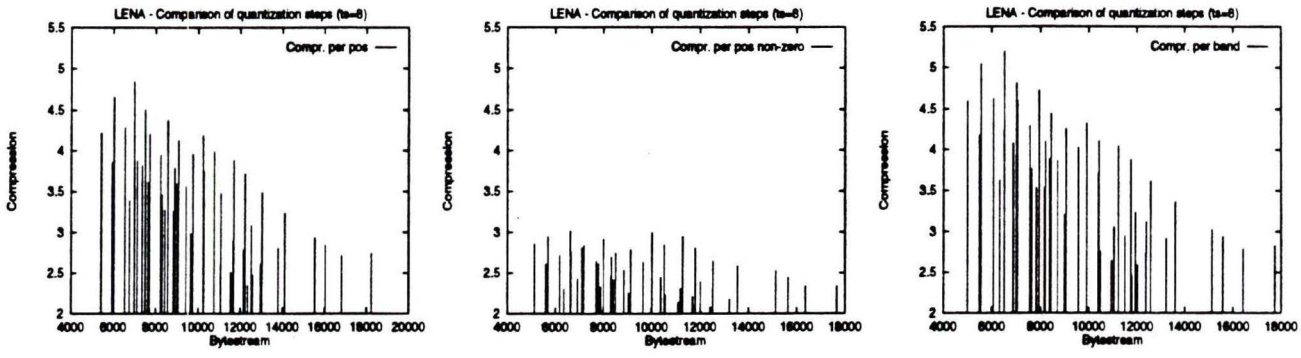


Figure 18: Different compression ratios for different band-ordening

In figuur 19 staat voor de ordening 'DC+AC+class+orien' de compressie. Bij deze ordening is de bytestream uit Polyvarb altijd even lang. Door kwantisatie zitten er veel of weinig nullen in. Hier hangt dus ook de compressiefactor van af. Deze factor is gelijk aan $\frac{Polyvarbbytestream}{vcompressbytestream} = \frac{constante}{x}$, wat dus een mooie grafiek oplevert.

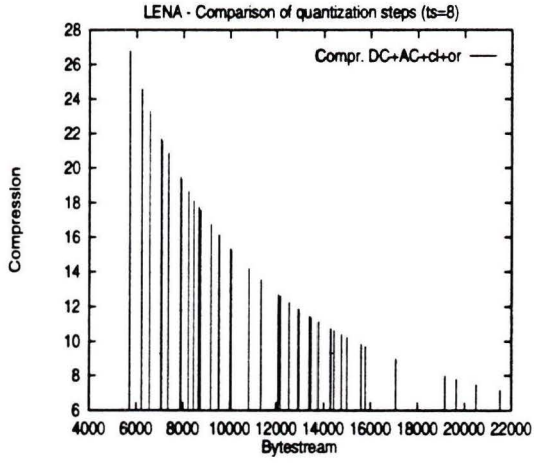


Figure 19: Different compression ratios for DC+cl+or+AC

Voor de volgorde van banden is telkens "DC, 1D and 2D diagonal" genomen. Dit omdat deze volgorde ervoor zorgt dat de hoogste coëfficiënswaarden in het begin van de bytestream zitten. Dit is het gunstigste bij het "per band" ordenen.

8 Conclusies

Tot slot van dit verslag een opsomming van de belangrijkste conclusies gevonden in dit verslag:

- De kwantisatie stap van de 1D low posities kan vrij hoog worden genomen. Deze 1D low coëfficiënten hebben weinig invloed op de PSNR. Een nadeel hiervan is dat er meer "blocking artifacts" tevoorschijn komen.
- Voor de kwantisatie stap van 2D posities zal voor een goede bytestream/PSNR verhouding grote stappen beter zijn dan kleine. Omdat er veel 2D coëfficiënten zijn kost een kwantisering met stap '1' erg veel bytes in de bytestream.
- Voor het maximaal aantal hoeken dat gekozen moet worden in polyangle is '16' al ruim voldoende. Eventueel kan ook voor '8' hoeken gekozen worden. Dit heeft met name nut bij zeer hoge compressiefactoren.
- Voor de sigma hebben we twee waarden gevonden. Voor ts='4' geldt een sigma van $\pm$=1.2 als optimum, terwijl voor ts='8' een sigma van $\pm$2.4 het optimum is.
- Als ordening van de banden in de bytestream kan vrijwel altijd worden gekozen voor een classificatie afhankelijke ordening. Alleen indien men een lage compressie wil, komen de klassieke methoden beter uit de bus.

Om te vergelijken met andere compressiemethoden, is in figuur 20 de compressie factor uitgezet tegen de PSNR. De twee referentie methoden zijn het Zerotree algoritme en het JPEG-DCT algoritme.

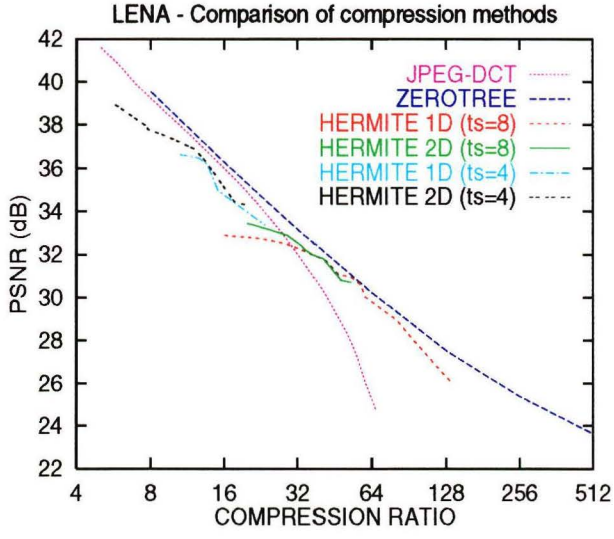


Figure 20: Different compression methods

Uit figuur 20 is af te leiden dat de Hermite transformatie methode goed bruikbaar is ten opzichte van andere compressie methoden. Nadeel is wel dat er steeds verschillende windows en coders worden gebruikt om goede resultaten te verkrijgen. Niet alle parameters staan dus van te voren vast. In het bijzonder voldoet de methode prima bij hogere compressie ratios ten opzichte van het veel gebruikte JPEG-DCT algoritme.