MMT practicum 2 : Image Compression

Dit practicum wordt per 2 gemaakt (zie reeds vastgelegde groepen), de deadline voor inlevering staat vermeld op Blackboard.

In dit practicum werken we aan image compressie, op een manier die lijkt op JPEG. Je start altijd met een image die vierkant is, en een pixelcount in breedte en hoogte heeft die een veelvoud is van 4 (bv 512x512). Aangezien we slechts een elementaire compressie toepassen werken we met grijswaardenbeelden van 8 bpp. De compressie moet bestaan uit volgende stappen :

- Indeling van het originele beeld in blokken van *4x4* pixels
- Elk blok wordt getransformeerd via de DCT. Voor een oplossing van de DCT voor 4x4 blokken kan je kijken naar de afleiding onderaan deze opgave. Het resultaat van deze transformatie moet op een overzichtelijke manier uitgeschreven kunnen worden (bv in een log file of op console).
- De getransformeerde blokken worden gequantiseerd. De quantisatiefactor(en) is/zijn *instelbaar* ! Je kan voor verschillende coefficienten een andere factor gebruiken. Het resultaat moet ook uitgeschreven kunnen worden.
- Voer een zig-zag scan uit (zoals bij JPEG) om de coëfficiënten in een lijst te steken
- Pas run-length encoding toe op deze lijst – eventueel beperkt tot de factoren die 0 zijn. Deze stap moet aan- en uit te schakelen zijn mbv een flag in de config file (0 of 1, zie verder)
- Schrijf het resultaat uit naar een bestand (binair) – kijk hieronder voor het te gebruiken formaat.

Gebruik niet meer bits dan nodig voor het uitschrijven van de gecodeerde data. Dus, als je bijvoorbeeld runlength encoding optioneel wil maken, voorzie dan slechts 1 bit in de gecodeerde data die aangeeft of RLE gebruikt wordt of niet. Zitten bepaalde getallen bijvoorbeeld in de range [0,63] gebruik dan slechts 6 bits, etc.

De decompressie moet ook geïmplementeerd worden en is in staat om, vertrekkende van een gecomprimeerd bestand, terug een beeld van grijswaarden op te bouwen.

We verwachten twee executables: encoder en decoder. Beiden moeten één argument aanvaarden: Een configuratie bestand waarmee de settings voor de encoder/decoder bepaald kunnen worden.
Het bestand waarmee je begint (ongecomprimeerd) heeft altijd als extensie **.raw**, het gecomprimeerde bestand altijd **.enc** . Uiteraard is het gedecomprimeerde bestand op zijn beurt ook weer een **.raw** file

Het configuratie bestand heet **settings.conf** en moet exact de volgende parameters bevatten, maar niet noodzakelijk in deze volgorde:

```
rawfile=input.raw
encfile=encoded.enc
decfile=decoded.raw
width=512
height=512
rle=1
quantfile=matrix.txt
logfile=log.txt
```

Het matrix.txt bestand bevat een 4x4 quantisatiematrix in dit **exacte** formaat :

```
2 4 8 16
4 4 8 16
8 8 32 64
16 32 64 128
```

Let op !

- Er mogen GEEN andere parameters in de config file staan die niet hierboven vermeld zijn
- Er morgen GEEN command line argumenten gevraagd worden die niet hierboven/hieronder vermeld zijn
- Er mogen GEEN prompts gegeven worden door het programma bij uitvoering.
- Schermoutput moet/mag informatie bevatten over de voortgang, maar niet overdreven (niet de informatie van elk blokje uitschrijven)
- Namen zoals 'settings.conf' en 'matrix.conf' zijn voorbeeldnamen, het echte settingsbestand wordt als argument aan je programma meegegeven, en de naam van het bestand met de quantisatiematrix staat in de settings file.
- Zowel het bestand met settings als dat met de quantisatiematrix kunnen UNIX ('\n') of DOS ('\r\n') tekens gebruiken om regels van elkaar te scheiden. Je programma MOET met beide om kunnen gaan.

Met ImageMagick kan je alle soorten bestanden (JPEG, PNG, ...) naar 8-bit grayscale raw images omzetten. Ook de omgekeerde richting is mogelijk zodat je de data kan zien zonder een viewer te schrijven.

Om de foto om te zetten naar een grayscale image van 512 op 512 pixels, waarbij elke pixel 1 byte in beslag neemt:

```
convert foto.jpg -resize 512x512! gray:foto.raw
```

Om een raw grayscale image van 512 op 512 pixels (1 byte per pixel) terug om te zetten naar PNG (*verplicht* PNG, geen JPG):

```
convert -size 512x512 -depth 8 gray:foto.raw fotogray.png
```

Maak voorbeeldbestanden met verschillende quantisatie-matrices om het effect ervan te beoordelen. Gebruik hierbij ook grote/kleine quantisatiewaarden.

Opgelet !

- Het is NIET toegelaten om extra quantisatie-factoren in rekening te brengen, buiten deze gedefinieerd in de matrix. Pas dus zelf geen vermenigvuldiging toe op deze factoren.
- De encoder wordt als volgt opgestart (windows/linux) :
  - **encoder.exe settings.conf** of **encoder settings.conf**
- De decoder wordt als volgt opgestart :
  - **decoder.exe settings.conf** of **decoder settings.conf**
- De genoemde bestandsnamen moeten exact aangehouden worden, teneinde het voor het onderwijsteam mogelijk te maken om de resultaten snel te evalueren.
- Projecten die zich niet houden aan de conventies mbt bestandsnamen en parameters kunnen rekenen op puntenverlies.
- Let op het geheugenbeheer. Hou rekening met memory leaks en hou het geheugengebruik onder controle, want je zal de code moeten herbruiken in het volgende practicum.

Wat is in te leveren ?

- Code van encoder en decoder, **inclusief make file of VS solution!**
- Executable van encoder en decoder in RELEASE mode (let op, test dit voorheen zelf op problemen !)
- Readme bestandje waarin je beschrijft welke features wel/niet geïmplementeerd zijn. Als je zelf artefacten tegenkomt bij het runnen van de applicatie op voorbeeldbestandjes moet je dat hierin duidelijk aangeven. Als we stuiten op bugs waarvan je zelf weet dat ze erin zitten en niet aangegeven zijn volgt een negatieve impact op de score.

Inlevering gebeurt via blackboard. Er zal ook een demonstratie gegeven moeten worden, informatie hierover volgt later.

Volgende tekst geeft een korte uitleg hoe je in ‘t algemeen de DCT kan berekenen.

De uitleg over DCT komt uit het boek H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia by Iain E. G. Richardson.

3.4.2.2 DCT

The Discrete Cosine Transform (DCT) operates on **X**, a block of $N \times N$ samples (typically image samples or residual values after prediction) and creates **Y**, an $N \times N$ block of coefficients. The action of the DCT (and its inverse, the IDCT) can be described in terms of a transform matrix **A**. The forward DCT (FDCT) of an $N \times N$ sample block is given by:

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}} \tag{3.1}$$

and the inverse DCT (IDCT) by:

$$\mathbf{X} = \mathbf{A}^{\mathrm{T}}\mathbf{Y}\mathbf{A} \tag{3.2}$$

where **X** is a matrix of samples, **Y** is a matrix of coefficients and **A** is an $N \times N$ transform matrix. The elements of **A** are:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \qquad \text{where } C_i = \sqrt{\frac{1}{N}}\ (i=0), \qquad C_i = \sqrt{\frac{2}{N}}\ (i>0) \tag{3.3}$$

Equation 3.1 and equation 3.2 may be written in summation form:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \tag{3.4}$$

$$X_{ij} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N} \tag{3.5}$$

In onze implementatie werken we met 4x4 pixel blokken en hebben we dus volgende afleiding :

Example: $N = 4$

The transform matrix **A** for a 4×4 DCT is:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \tag{3.6}$$

The cosine function is symmetrical and repeats after 2π radians and hence $\mathbf{A}$ can be simplified to:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & -\sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{bmatrix} \tag{3.7}$$

or

$$\mathbf{A} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad \text{where} \quad \begin{aligned} a &= \frac{1}{2} \\ b &= \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ c &= \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{aligned} \tag{3.8}$$

Evaluating the cosines gives:

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix}$$