INF1040 – Digital representasjon
Oppsummering 2008 – del II

Fritz Albregtsen

Lydintensitet

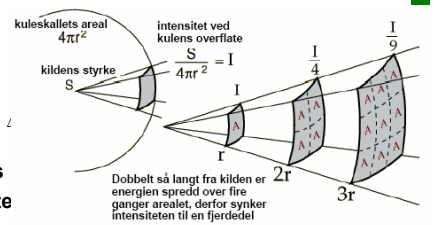
- Vi kan høre lyder over et stort omfang av intensiteter:
  - fra høreterskelen, $I_0 = 10^{-12}$ W/m², til Smerteterskelen,10 W/m²
- Oftest angir vi ikke absolutt lydintensitet i W/m²
  - Vi angir intensiteten i forhold til høreterskelen I_0
  - Vi bruker en logaritmisk skala, decibel-skalaen, forkortet dB

$$\beta(dB) = 10\log_{10}\left(\frac{I}{I_0}\right)$$

  - Logaritmen med basis 10 til et tall x, er den potensen vi må opphøye basistallet 10 i for å få tallet x.
  - Eksempel1: $\log_{10}(100) = 2$ fordi $100 = 10*10 = 10^2$

Forholdet mellom lydintensiteter

- Lyd fra en punktkilde brer seg utover
  - Dobler vi radien, fordeles energien over en fire ganger så stor kuleflate.
  - Med en ti ganger så stor radius fordeles energien over en 100 ganger så stor flate



- Du flytter deg 10 ganger så langt unna en lydkilde. Hvor mye endres lydintensiteten i dB?

$$\Delta\beta = 10\log_{10}\left(\frac{1}{100}\right)dB = 10(-2)\,dB = \underline{-20db}$$

- Et flue gir en lydintensitet på 40 dB. Hva blir lydintensiteten i dB fra 100 fluer?

$$\beta_2 = \beta_1 + 10\log_{10}\left(\frac{100}{1}\right) = 40dB + 10(2)\,dB = \underline{\underline{60db}}$$

Dobbelt så høyt ?

- "Det anslås at F-35 vil ha et støynivå over 20 desibel høyere enn F-16."
- En endring på 10 desibel **oppleves** som en dobling av støyen.
- 20 desibel svarer til en økning med en faktor 100 i **målt intensitet.**

Lydens svingninger - frekvens

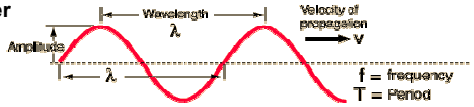
- Hvordan lyden høres ut, avhenger av hvor raske svingninger den inneholder.
- Antall svingninger per sekund er **frekvensen** til en tone.
  - Frekvensen til en tone måles i Hz (≡ s^{-1})
- Vi hører lydbølger
  - med frekvenser mellom 18 Hz og 20 kHz
- De fleste lyder består av flere rene toner med ulik frekvens.
- Oktav: et toneintervall der forholdet mellom de to frekvensene er 1:2 (en oktav opp) eller 2:1 (en oktav ned).
- 12 halvtonetrinn i en oktav
  - $f_2 = f_1 \cdot 2^{(1/12)}$ $f_7 = f_1 \cdot 2^{(1/2)}$ $f_{12} = f_1 \cdot 2$

Bølgelengde og lydhastighet

- Amplituden til en bølge er det maksimale utslaget.
- Hastigheten til en bølge er lik produktet av bølgelengde og frekvens

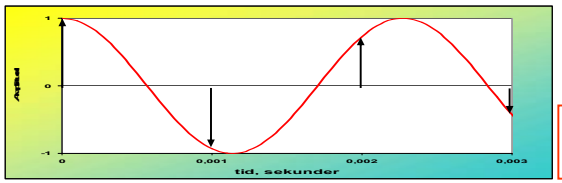
  $v = f \lambda$

  (gjelder generelt, både for lydbølger og elektromagnetiske bølger)
- Bølgelengden λ er den fysiske lengde mellom to punkter på samme sted i svingningen.
  - Bølgelengden for hørbare lydbølger
    - » går fra λ = 17 m for f = 20 Hz
    - » til λ = 1.7 cm for f = 20 000 Hz.
  - En A på f = 440 Hz har en bølgelengde på λ = 75 cm,
  - En A som er en oktav lavere (f = 220 Hz) har λ = 1.5 m.



Diskret representasjon av et kontinuerlig lydsignal

- Alle lydsignaler må samples og kvantiseres.



Hvor presist?

Kvantiserings-nivåer

Hvor ofte? - Samplingsraten

- Prosessen som gjør signalet *diskret i tid* kalles sampling.
- Å sample et signal vil si å plukke ut verdien til signalet på tidspunkter med gitte intervaller.
- Lydintensiteten må så skaleres og kvantiseres for å kunne lagres (f.eks. som un-signed byte som gir verdiene 0-255).

Samplingsintervall og samplingsfrekvens

- Tidsavstanden mellom to sampler, gitt i sekunder (eller millisekunder) kalles *samplingsintervallet*.
- Hvis samplingsintervallet er det samme mellom alle sampler sier vi at vi har en *uniform sampling*.
- *Samplingsfrekvens* (også kalt *samplingsrate*) angis i Hz, og er det inverse av samplingsintervallet.
- Samplingen gjøres av en *A/D-omformer* ved at lydintensiteten (i praksis mikrofonspenningen) holdes i et lite tidsintervall.

Nyquist-teoremet

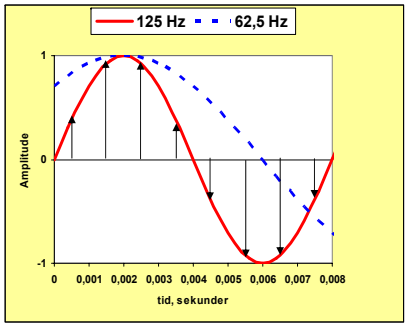
- Hvor ofte må vi måle lydintensiteten i et gitt lydsignal?

Anta at et analogt signal er *bånd-begrenset*, dvs. at det ikke har sinuskomponenter med frekvenser over et maksimum f_{max}. Signalet kan da rekonstrueres eksakt fra de samplene vi har, hvis samplingsfrekvensen $f_s = 1/T_s$ er større enn $2f_{max}$.

- $2f_{max}$ kalles Nyquist-raten.
- Hvis vi sampler med en samplingsfrekvens f_s som er minst $2f_{max}$, så sampler vi i henhold til Nyquist-teoremet.

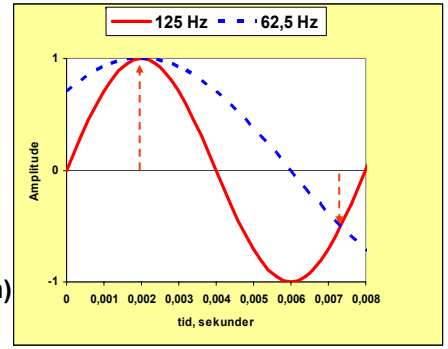
Oversampling

- Vi sampler for eksempel en f =125 Hz sinus med samplingsfrekvens f_s=1 kHz
- Det gir f_s/f = 1 000/125 = 8 sampler pr periode.
- I forhold til Nyqvist-raten er dette en 4X oversampling (svarte piler i figuren).



Undersampling og aliasing

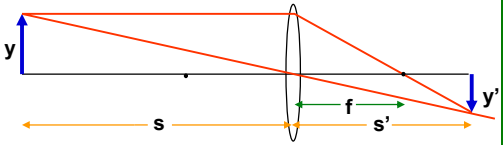
- Aliasing betegner det fenomenet at en sinusoid ved for lav samplingsrate gir opphav til samme diskrete signal som en sinusoid med lavere frekvens.
- Vi sampler for eksempel en f =125 Hz sinus med f_s = 1.5 f = 187.5 Hz.
- Dette gir for eksempel sampler ved t = 0.002 og ved t = 0.733 (stiplede røde piler).
- Rekonstruksjon gir sinus med f_a= 62.5 Hz, (stiplet kurve i figuren)
- Vi har fått en "aliasing".
- Merk at $f_a = f_s - f$ når $f < f_s < 2f$



Objekt-bilde relasjonen

- Objekt-bilde relasjonen:

$$\frac{1}{s}+\frac{1}{s'}=\frac{1}{f} \Rightarrow s' = \frac{s\,f}{(s-f)}$$



- Hvor stort blir bildet?

$$\frac{y'}{s'}=\frac{y}{s} \Rightarrow y' = \frac{ys'}{s}$$

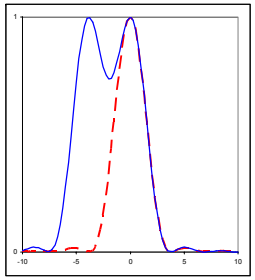
- Setter inn uttrykket vi har for s' finner størrelsen på *bildet* i fokalplanet uttrykt ved s, f og y:

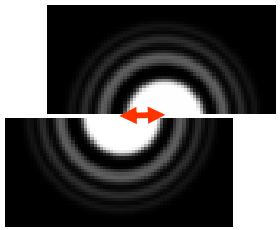
$$y' = \frac{y\,f}{(s-f)}$$

Rayleigh-kriteriet

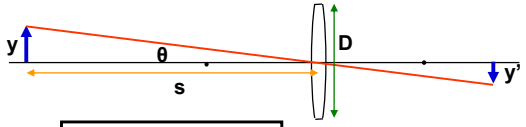
- To lys-punkter kan akkurat adskilles i bildet hvis de ligger slik at sentrum i det ene diffraksjonsmønstret faller sammen med den første mørke ringen i det andre.
- Linse med diameter D, bølgelengde λ.
- La maksimum til den ene falle i første minimum til PSF for den andre.
  - Vinkelen mellom dem er da gitt ved

    sin θ = 1.22 λ / D radianer.
  - Dette er "Rayleigh-kriteriet".
  - *Vi kan ikke se detaljer mindre enn dette.*





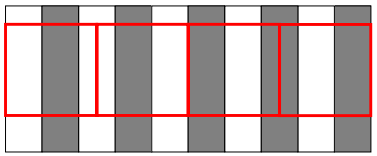
Hvor små detaljer kan linsen oppløse?

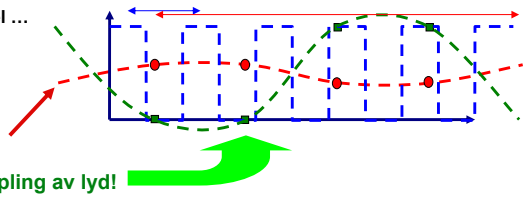


- Rayleigh's kriterium: vinkeloppløsningen er gitt ved $\sin\theta = 1.22\frac{\lambda}{D}$
- Tangens til vinkelen θ er gitt ved $tg(\theta) = \frac{y}{s}$
- For små vinkler er sin(θ) = tg(θ) = θ, når vinkelen θ er gitt i radianer.
- => Den minste detaljen vi kan oppløse: $\frac{y}{s} = 1.22\frac{\lambda}{D} \Rightarrow y = 1.22\frac{s\lambda}{D}$

Et undersamplet Nyquist-stakitt …

- Ta et analogt bilde av et stakitt med 5 sprosser og 5 mellomrom per meter.
  - Vi har en periodisk struktur
    - » f = 5 svingninger per meter
    - » Periode T = 20 cm
- Vi må ha MINST to piksler per periode!
  - Anta nå at pikslene svarer til 25 · 25 cm, dvs f_s = 4 sampler per meter.
  - Finner gjennomsnitt i hvert piksel …
    - » Amplituden reduseres …
    - » Vi får aliasing, $f_a = |f_s - f| = |4 - 5| = 1$
    - » Perioden $T_a = 1/f_a = 1$ meter
- Dette er annerledes enn ved sampling av lyd!
  - Vi får redusert amplitude i forhold til lydsampling
  - Vi får samme aliasingfrekvens
  - Vi kan få faseforskyvning i bildet. Fasen til lyden hører vi ikke!





Viktige begreper

- Et digitalt bilde er en funksjon av to (eller flere) heltallsvariable f(x,y) (x og y er heltall)
- Et 2-dimensjonalt digitalt bilde er en 2-dimensjonal array/matrise. Dette kalles raster-representasjon.
- Hvert element i matrisen kalles en piksel, og angis ved koordinater x og y.
- Tallverdien til hver piksel angir intensiteten til pikselen.
- Lagres pikselverdiene i matriser, trenger vi ikke ta vare på koordinatene.

x

y

1	5	7	3	6	4
3	7	7	6	4	3
2	4	4	3	4	4
3	3	1	1	1	1
3	3	3	9	9	9
5	6	5	5	6	6

Origo (0,0) er ofte oppe i venstre hjørne i bildet. Første piksel kan ha indekser (0,0) eller (1,1)

To representasjoner for bilder

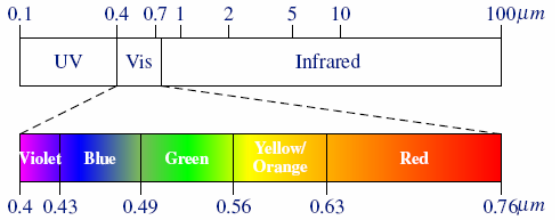
- Lagre alle pikselverdiene (gråtoneverdi, eller fargekomponenter)
  - raster-bilde, en matrise som inneholder pikselverdiene
- Lagre en parametrisert beskrivelse av bildets innhold
  - vektor-bilde, en liste med objekt-parametre
    - » Krever at bildet deles opp i objekter, og at hvert objekt beskrives ved en rekke parametre.
    - » Forutsetter
  1. Enten at bildet er forholdsvis enkelt (skisser, tegninger, CAD, kart, …)
  2. Eller at det brukes veldig mange parametre for å generere noe som ligner på et naturlig bilde (”virtual reality”).
  - Det er naturlig å la objekter ha overflate-egenskaper (varierende farge, refleksjonsegenskaper, tekstur, etc.).

Fargene i spekteret

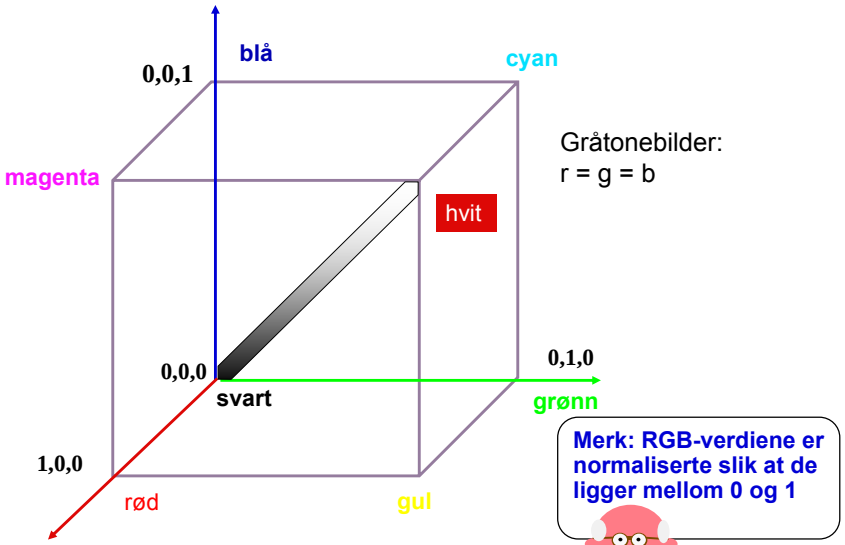
- Vi oppfatter farger mellom 400 og 700 nm (nm = 10^{-9} m)
  - Fiolett: 400 - 446 nm
  - Blå: 446 - 500 nm
  - Grønn: 500 - 578 nm
  - Gul: 578 - 592 nm
  - Oransje: 592 - 620 nm
  - Rød: 620 - 700 nm

CIE har definert primærfargene:

Blå:	435.8 nm
Grønn:	546.1 nm
Rød:	700.0 nm



RGB-kuben

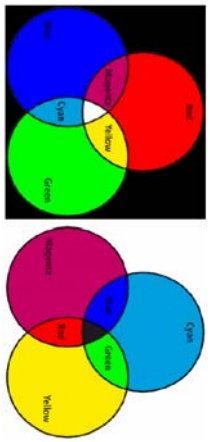


Gråtonebilder:
r = g = b

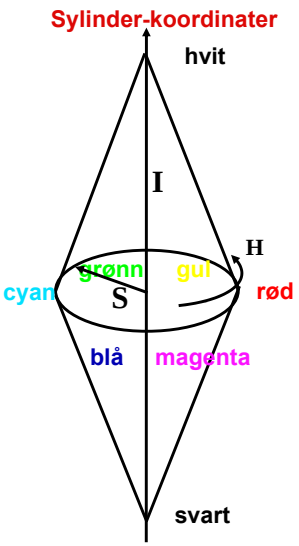
Merk: RGB-verdiene er normaliserte slik at de ligger mellom 0 og 1

RGB og CMYK

- RGB er additiv
  - Starter med svart, legger til lyse farger
  - RGB er vanlig på display, og i kamera-detektorer
- CMYK- modellen er subtraktiv
  - starter med hvitt, trekker fra farger
  - CMYK er vanlig på fargeprintere
  - (K er ekstra komponent for svart).
- Overgang fra RGB til CMY (for hver fargekomponent):
  - Cyan: C = 1 – R (255 – R)
  - Magenta: M = 1 – G (255 – G)
  - Yellow: Y = 1 – B (255 – B)



Intensity, Hue, Saturation (IHS)



- I er en gråtoneskala. Diagonalen i RGB-kuben.
- Hue: fargekomponenten.
  - H er en vinkel som ligger mellom 0 og 2π.
  - Rød velges oftest som startpunkt.
- S er metningskomponenten (0 til 1, radielt)
- Primærfargene ligger ekvidistante rundt sirkelen
  Rød: H = 0, Grønn: H = $2\pi/3$, Blå: H= $4\pi/3$
- Sekundærfargene ligger midt imellom:
  Gul: H = $\pi/3$, Cyan: H = π, Magenta: H= $5\pi/3$,
- Hvis vi skalerer H-verdiene til 8-biter får vi:
  - Primærfargene:
    R: H = 0, G: H = 255/3 = 85, B: H = 255·2/3=170
  - Sekundærfargene:
    Gul: H = 255/6 = 42, Cyan : H = 255/2 = 127,
    Magenta: H=255·5/6 = 213.

YIQ

- NTSC er standard for TV og video i USA. Bruker fargesystemet YIQ.
  - Y beskriver luminans, I og Q er krominanskomponentene.
  - samme signalet brukes både på farge- og gråtoneskjermer.
- Overgangen mellom RGB og NTSC's YIQ :
  - Luminans-komponenten $Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$
  - Hue-komponenten $I = 0.596 \cdot R - 0.274 \cdot G - 0.322 \cdot B$
  - Metnings-komponenten $Q = 0.211 \cdot R - 0.522 \cdot G + 0.311 \cdot B$
- Summen av luminans-koeffisientene er 0.299 + 0.587 + 0.114 = 1.000
  - RGB svart (0,0,0) gir NTSC Y = 0
  - RGB hvit (1,1,1) gir NTSC Y = 1
- Både summen av koeffisientene for I og Q er 0.0
  - RGB grå (g,g,g) gir NTSC I = Q = 0

Fargebilder og fargetabeller

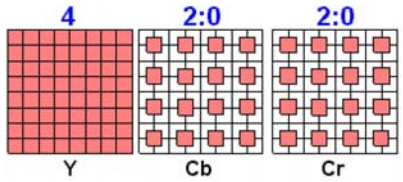
- Hvert piksel i et RGB-bilde kan lagres med f.eks (8+8+8) biter
- Alternativt kan man bruke
  - *b* biter til hvert piksel i bildet og en fargetabell med 2^b linjer.
  - Hver linje i tabellen beskriver en r, g, b-farge med f.eks. 24 biter
  - Tabellen inneholder de 2^b fargene som best beskriver bildet.

M·N·24 biter RGB-bilde **eller** M·N·8 biter gråtone-bilde + 2^8 linjer á 24 biter fargetabell

- I bilde-filen ligger nå pikselverdiene som tall mellom 0 og 2^b.
- Når vi skal vise bildet, slår vi opp i tabellen i samme linje som pikselverdien, og finner r, g, b-verdiene til pikselen.
- Fargetabeller gir fleksibilitet. Hvilket alternativ tar mest plass?

Chroma 4:4:4, 4:2:2, 4:1:1 eller 4:2:0

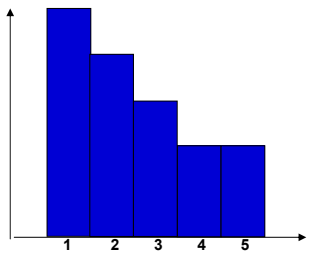
- Subsamplingen i YCbCr
  betegnes med talltrippelet 4:n:n
- n angir hvor mange av 4 originale piksler
  i Cb og Cr som faktisk lagres / sendes.
- 8 · 8 piksler er en "blokk" i et digitalt bilde.
- En "blokk" er grunn-enheten i JPEG-kompresjon
  av digitale bilder.
- JPEG bruker 4:2:0



Hva er et histogram?

- Anta at vi har et gråtonebilde med $m \cdot n$ piksler
  - hvert piksel har ordlengde b biter => vi har 2^b mulige gråtoner.
- Ser på gråtonen i hver piksel, teller opp hvor mange piksler det finnes for hver pikselverdi, og får et *gråtone-histogram*.
- Histogrammet *h(v)* er en tabell over antall forekomster av verdien v.

1	3	2	1
5	4	5	3
4	1	1	2
2	3	2	1



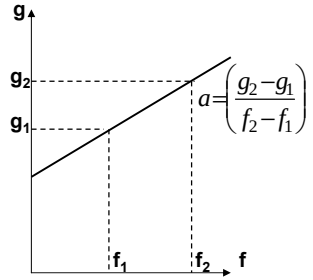
Lineær gråtonemapping

- Vi vil at pikselverdier i området $[f_1, f_2]$ i inn-bildet skal havne i gråtone-området $[g_1,g_2]$ i ut-bildet.
- Dette er en lineær mapping fra f til g

$$g(x,y) = g_1 + \left(\frac{g_2 - g_1}{f_2 - f_1}\right)\left[f(x,y) - f_1\right]$$

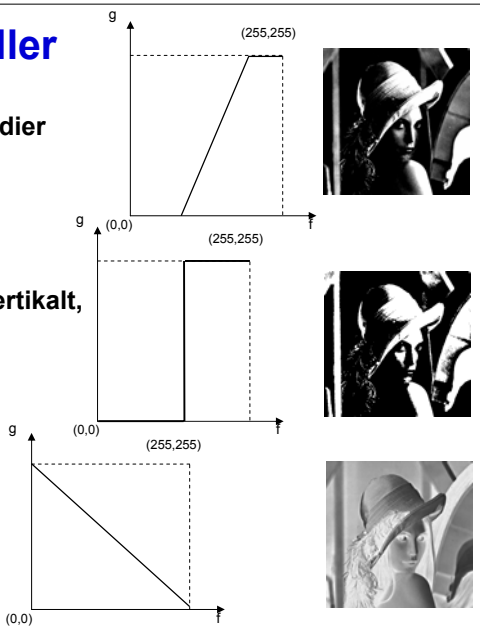
altså en rett linje med stigningstall

$a = (g_2-g_1)/(f_2-f_1)$



Spesialtilfeller

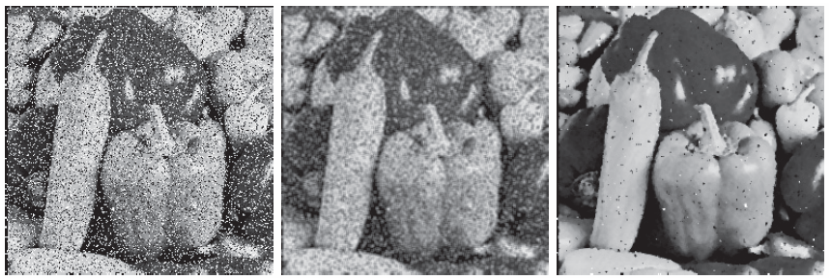
- Et lite område av pikselverdier mappes til å bruke hele gråtone-skalaen.
- Hvis den skrå linjen står vertikalt, svarer det til en terskling av gråtoneverdiene.
- Vi kan invertere gråtonene og få et negativt bilde.



Støyfiltrering – middelverdi eller median

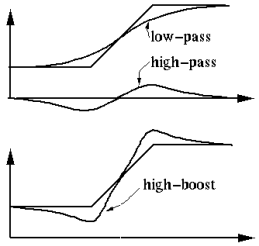
To enkle metoder for å filtrere et bilde:

- Middelverdifilter: erstatt pikselverdien i et punkt med gjennomsnittet av pikselverdiene til nabopikslene.
- Medianfilter: erstatt pikselverdien i et punkt med medianen av pikselverdiene til nabopikslene.
  - Bevarer kanter i bildet bedre enn middelverdifilteret.



Bildeforbedring: ”Unsharp masking”

- Middelverdi-filtrering gjør bildet uskarpt.
- Vi kan subtrahere det uskarpe bildet fra originalen, og addere differansen til originalen.
- Resultatbildet vil virke skarpere enn originalen.
- Alternativ: original + høypass = ”high-boost”



To representasjoner for bilder

- Det er to fundamentalt forskjellige måter å representere et bilde på:

1. Lagre alle pikselverdiene (gråtoneverdi, eller fargekomponenter)
2. *Lagre en parametrisert beskrivelse av bildets innhold*

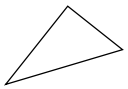
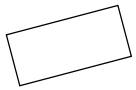
- Den siste metoden krever at bildet deles opp i objekter, og at hvert objekt beskrives ved en rekke parametre.
- Dette forutsetter

1. Enten at bildet er forholdsvis enkelt (skisser, tegninger, CAD, kart, …)
2. Eller at det brukes veldig mange parametre for å generere noe som ligner på et naturlig bilde (”virtual reality”).

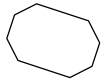
- I det siste tilfellet er det naturlig å la objekter ha overflate-egenskaper (varierende farge, refleksjonsegenskaper, tekstur, etc.).

Vektor-representasjon av enkle geometrier

- Anta at figuren kan ligge hvor som helst innenfor 2^n x 2^m piksler
- Hvor mange biter trengs for å representere en ...

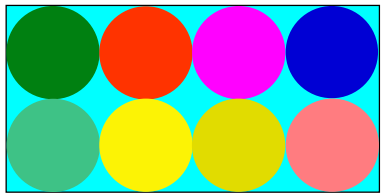
- trekant  tre hjørner => 3(n+m) biter
- rektangel  to hjørner => 2(n+m) biter

Alternativ: Ett hjørne + lengde + bredde + rotasjonsvinkel

- N-kant  N hjørner => N(n+m) biter
- Sirkel  x, y, r => (n+m) + int($\log_2$(r)) +1 biter
- Ellipse  x,y,a,b,θ => (n+m)+2int(log(max(a,b)))+10 biter

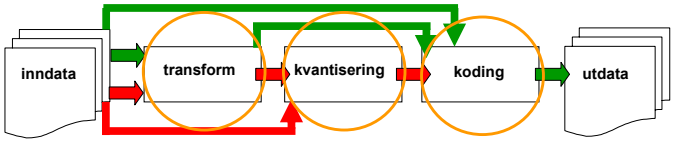
Eksempel

- Gitt 1024 x 512 piksels RGB-bilde
- Inneholder 8 like store sirkler med radius = 128



- Raster-representasjon: 1024 x 512 x 3 byte = 1.5 MiB
- Vektor-representasjon:
  - 8 sirkler: 8 x (10 + 9 +7 + 24) biter = 400 biter
  - bakgrunn: 10 + 9 + 24 biter = 33 biter

De tre stegene i kompresjon



- Mange kompresjons-metoder er basert på å representere dataene på en annen måte, altså transformer av original-dataene.
  - Differansetransform
  - løpelengder/run-length,
- Hvis vi kvantiserer original - dataene, så kan ikke dette reverseres.
- Koding bygger ofte på sannsynlighetsfordelinger,
  - Estimert ved normaliserte histogrammer.
- Transformer og koding er alltid reversible.
- Kvantisering gir alltid et tap av presisjon.

Kompresjonsrate og redundans

- Vi vil lagre en gitt informasjonsmengde ved bruk av færre data.
- Redundante data må bort.
- Kompresjonsraten angis som

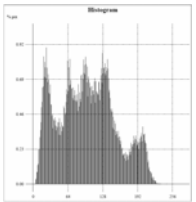
$$CR = i/c,$$

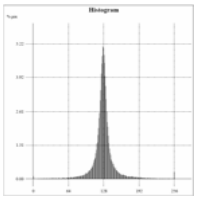
i er antall biter per sampel originalt

c er antall biter per sampel i det komprimerte datasettet.

Differansetransform

- Gitt en linje i bildet med gråtoner
  $f_1,.....f_N$, $0 \le f_i \le 2^b-1$.
- Transformer (reversibelt) til
  $g_1 = f_1$, $g_2 = f_2-f_1$, $g_3 = f_3-f_2$, ... , $g_N = f_N-f_{N-1}$
- Vi trenger nå b+1 biter hvis vi skal tilordne like lange binære koder til alle mulig verdier.
- I differansehistogrammet vil de fleste verdiene samle seg rundt 0.
- Naturlig bit-koding av differansene er IKKE optimalt.





Løpelengde-transform (run-length)

- Gitt en sekvens av pikselverdier fra en linje i et bilde, f.eks
  333333555555555544777777 (24 byte)
- Vi ser at flere piksler etter hverandre er like
  - Hvert slikt ”run” har en pikselverdi og en lengde
- Vi kan lagre en sekvens av tall-par (f_i,r_i)
  - f er pikselverdien i et ”run”
  - r er lengden til et ”run” (run-length)
- Tilsammen trenger vi her 4 tallpar
  - (3,6), (5,10), (4,2), (7,6) istedenfor hele sekvensen av 24 tall ovenfor.
- Løpelengdetransformen er reversibel !

Naturlig binær-koding

- **Naturlig binær-koding:**
  - Alle kode-ord er like lange.
  - Kjenner vi noen eksempler på dette?
- **Eks: vi har 8 mulige verdier**

Symbol nr.	1	2	3	4	5	6	7	8
Symbol	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Kode c_i	000	001	010	011	100	101	110	111

- Naturlig binærkoding er bare optimal hvis alle verdiene i sekvensen er like sannsynlige.

Gjennomsnittlig antall biter pr. symbol

- Anta at vi konstruerer en kodebok med kodeord $c_1,....c_N$ slik at symbol s_i kodes med kodeordet c_i.
- b_i er ord-lengden (angitt i biter) av kodeordet c_i.
- Gjennomsnittlig antall biter pr. symbol for denne koden er:

$$R = b_1 p_1 + b_2 p_2 + ... + b_N p_N = \sum_{i=1}^{N} b_i p_i$$

Entropi, H

- Entropi er gjennomsnittlig informasjon pr. symbol.

$$H = -\sum_{i=0}^{2^b-1} p(s_i)\log_2(p(s_i))$$

- **Entropien setter en øvre grense for kompresjonsraten** hvis vi koder hvert symbol for seg.
- **Entropien er en nedre grense for det gjennomsnittlig antall biter pr. symbol**

$$H \le R$$

Framgangsmåte - Huffman-koding

Gitt en sekvens med N symboler:

1. Sorter symbolene etter sannsynlighet, slik at de minst sannsynlige kommer sist.
2. Slå sammen de to minst sannsynlige symbolene i en gruppe, og sorter igjen etter sannsynlighet.
3. Gjenta 2 til det bare er to grupper igjen.
4. Gi kodene 0 og 1 til de to gruppene.
   - Kode 0 til den mest og 1 til den minst sannsynlige av de to
5. Traverser bakover, og legg til 0 og 1 i kodeordet for de to minst sannsynlige gruppene i hvert steg.

Eksempel - Huffman-koding

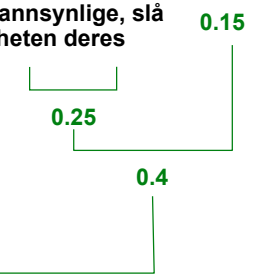
- Gitt 6 begivenheter A, B, C, D, E, F med sannsynligheter

Begivenhet	A	B	C	D	E	F
Sannsynlighet	0.3	0.3	0.13	0.12	0.1	0.05

Slå sammen de to minst sannsynlige, slå også sammen sannsynligheten deres

Finn så de to som nå er minst sannsynlige, og slå dem sammen på samme måte

Fortsett til det er bare to igjen

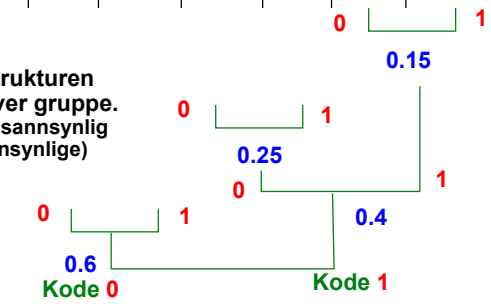


Eksempel - Huffman-koding

- Gitt 6 begivenheter A, B, C, D, E, F med sannsynligheter

Begivenhet	A	B	C	D	E	F
Sannsynlighet	0.3	0.3	0.13	0.12	0.1	0.05

Gå baklengs gjennom strukturen og tilordne 0 eller 1 til hver gruppe.
(F. eks. kode 0 til den mest sannsynlig og kode 1 til den minst sannsynlige)



Kodeboken

- Dette gir følgende kodebok

Begivenhet	A	B	C	D	E	F
Kode	00	01	100	101	110	111

- Med sannsynlighetene 0.3 0.3 0.13 0.12 0.1 0.05 blir gjennomsnittlig antall biter pr. symbol (R) for denne koden:

$$R = b_1 p_1 + b_2 p_2 + ... + b_N p_N = \sum_{i=1}^{N} b_i p_i = 0.6 * 2 + 0.4 * 3 = 2.4$$

- Entropien H er her mindre enn R:

$$H = -\sum_{i=0}^{2^b - 1} p(s_i) \log_2 (p(s_i)) = 2.34$$

Når er Huffman-koding optimal?

- Den ideelle binære kode-ord lengden for symbol s_i er $b_i = -\log_2(p(s_i))$
- Siden bare heltalls ordlengder er mulig, er det bare

$$p(s_i) = \frac{1}{2^k}$$

for heltall k som tilfredsstiller dette.

0	1	2	4
2	2	4	5
2	4	5	5
4	5	6	7

Eksempel: bildeutsnitt og histogram til høyre.

H = - 3 · (1/4) · $\log_2$(1/4) - 4 · (1/16) · $\log_2$(1/16)

= 3 · (1/4) · 2 + 4 · (1/16) · 4 = 2.5 biter per piksel

Og siden alle sannsynlighetene her er $1/2^k$,
er Huffman-koding optimal,
og den gjennomsnittlige ordlengden blir også
R = 2.5 biter per piksel.

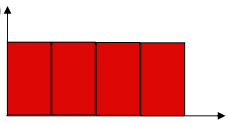
s	h(s)	p(s)
0	1	1/16
1	1	1/16
2	4	¼
4	4	¼
5	4	¼
6	1	1/16
7	1	1/16

Naturlig binærkoding eller Huffman?

- Alle symboler like sannsynlige (flatt histogram)

  => Naturlig binærkoding er optimal

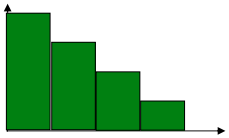
  generelt: R > H (toerpotenser: R=H)



- Ulike sannsynligheter, ikke toerpotenser

  => Huffman-koding gir kodings-gevinst

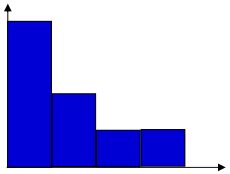
  R > H



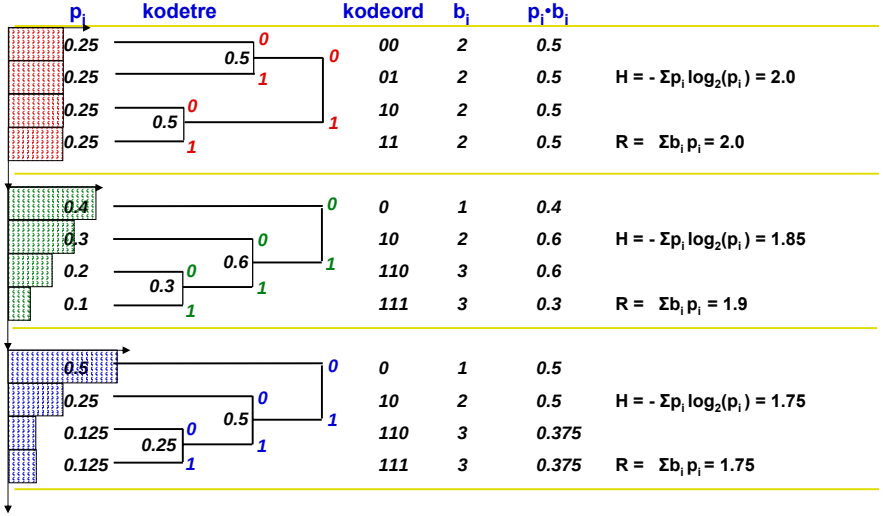
- Alle sannsynlighetene er toerpotenser

  => Huffman-koding er optimal

  R = H



Vi Huffman-koder litt !

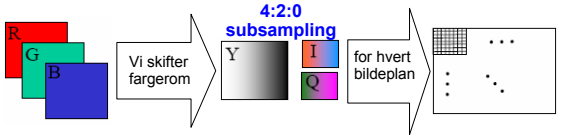


Et eksempel på Lempel-Ziv-koding

- Anta at alfabetet er a, b og c som tilordnes kodene 1, 2 og 3.
- Sender: ny frase = sendt streng + neste usendte symbol
- Mottaker: ny frase = nest siste mottatte streng + første symbol i siste streng
  - Mottakeren mottar kodene 1 2 4 3 5 …. Hva er meldingen?
    - » Mottar 1 = a 1 = a, 2 = b, 3 = c
    - » Mottar 2 = b Ny frase: 4= ab
    - » Mottar 4 = ab Ny frase: 5 = ba
    - » Mottar 3 = c Ny frase: 6 = abc
    - » Mottar 5 = ba Ny frase: 7 = cb
  - Meldingen var altså: ababcba ....
  - Etter hvert øker lengden på frasene I listen, og kompresjonsraten øker.

”Lossy” JPEG-kompresjon av RGB-bilde

- Vi skifter fargerom slik at vi separerer intensitet fra kromasi (sparer plass).
- Bildet deles opp i blokker på 8 · 8 piksler, og hver blokk kodes separat.



- Hver blokk transformeres med DCT (Diskret Cosinus Transform):

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

DCT transform

39.88	6.56	-2.24	1.22	-0.37	-1.08	0.79	1.13
-102.43	4.56	2.26	1.12	0.35	-0.63	-1.05	-0.48
37.77	1.31	1.77	0.25	-1.50	-2.21	-0.10	0.23
-5.67	2.24	-1.32	-0.81	1.41	0.22	-0.13	0.17
-3.37	-0.74	-1.75	0.77	-0.62	-2.65	-1.30	0.76
5.98	-0.13	-0.45	-0.77	1.99	-0.26	1.46	0.00
3.97	5.52	2.39	-0.55	-0.05	-0.84	-0.52	-0.13
-3.43	0.51	-1.07	0.87	0.96	0.09	0.33	0.01

- Informasjonen i de 64 pikslene samles i en liten del av de 64 koeffisientene
  - Mest i øverste venstre hjørne

”Lossy” JPEG-kompresjon – 2

- Transformkoeffisientene skaleres med en vektmatrise og kvantiseres til heltall.

39.88	6.56	-2.24	1.22	-0.37	-1.08	0.79	1.13
-102.43	4.56	2.26	1.12	0.35	-0.63	-1.05	-0.48
37.77	1.31	1.77	0.25	-1.50	-2.21	-0.10	0.23
-5.67	2.24	-1.32	-0.81	1.41	0.22	-0.13	0.17
-3.37	-0.74	-1.75	0.77	-0.62	-2.65	-1.30	0.76
5.98	-0.13	-0.45	-0.77	1.99	-0.26	1.46	0.00
3.97	5.52	2.39	-0.55	-0.05	-0.84	-0.52	-0.13
-3.43	0.51	-1.07	0.87	0.96	0.09	0.33	0.01

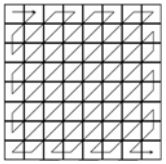
divideres med

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Avrundes til

2	1	0	0	0	0	0	0
-9	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Sikk-sakk-scanning ordner koeffisientene i 1D-rekkefølge.
  - Koeffisientene vil da stort sett avta i verdi utover i rekka
  - Mange koeffisienter er rundet av til null.
  - Løpelengde-transform av koeffisientene
    - » 2,1,-9,3,0,0,...,0 => (2,1)(1,1)(-9,1)(3,1)(0,60)
  - Huffman-koding av løpelengdene.



- Huffman-koden og kodeboken sendes til mottaker eller til lager.
- Gjentas for alle blokker i alle kanaler.

Takk for oppmerksomheten !

- Når alt er over har vi gitt dere
  - 14 forelesninger
  - En 387 siders lærebok
  - Over 500 lysark, 4 obliger og masse oppgaver
- Noe av dette ser dere sikkert nytten av allerede.
- Noe vil dere senere finne at dere har bruk for.
- Men det er sikkert noe dere ikke *kan* …
  - og noe dere ikke vet at dere ikke kan …
  - og mye dere kan som dere ikke har fra oss …

YESS !!!

Ken Musgrave, ”Blessed State” (1988)

- Tenk om du etter eksamen kan si
- Det har vært en fornøyelse å forelese for dere!
- Vi står fortsatt til disposisjon, helt fram til like før eksamen
- *Lykke til videre !*