INF2310 – Digital bildebehandling

FORELESNING 15

REPETISJON

Ole Marius Hoel Rindal

Gråtonetrasformasjoner
Histogramtransformasjoner
2D diskret Fourier-transform (2D DFT)
Filtrering i Fourierdomenet
Kompresjon og koding
Segmentering

Gråtonetransformasjoner

Repetisjon av histogrammer I

- Gråtonehistogram:

  h(i) = antall piksler i bildet med pikselverdi i

$$\sum_{i=0}^{G-1} h(i) = n \times m$$

- Det normaliserte histogrammet

$$p(i) = \frac{h(i)}{n \times m}, \quad \sum_{i=0}^{G-1} p(i) = 1$$

- Det kumulative histogrammet

$$c(j) = \sum_{i=0}^{j} h(i)$$

Lineær - transformasjon

- Lineær strekking

$$T[i] = a\,i \;+\; b$$

$$g(x,y) = a\,f(x,y) \;+\; b$$

- *a* regulerer kontrasten, og *b* "lysheten"
- *a*> *1*: mer kontrast
- *a*< *1*: mindre kontrast
- *b* : flytter alle gråtoner *b* nivåer
- "Negativer": *a*= *-1* , *b* = maxverdi for bildetype

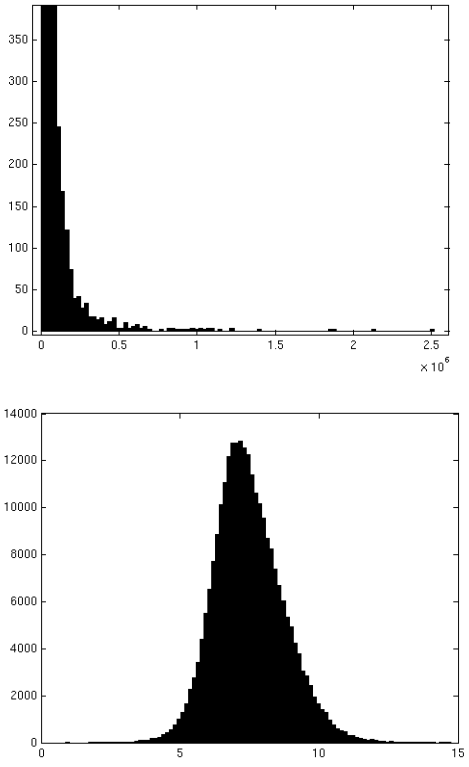
Ikke-lineær transform

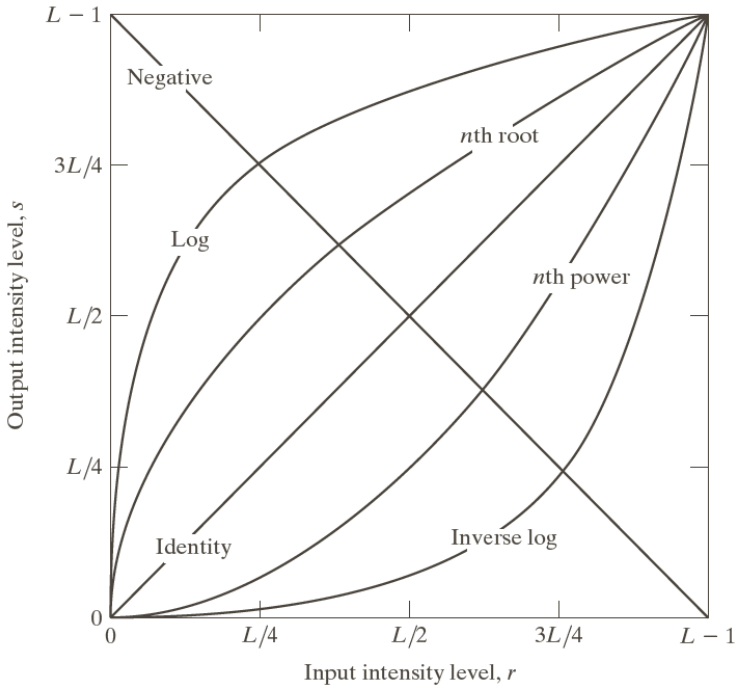
- Logaritmisk skalering
  - Eks: Desibel og radarbilder, Fourier-transform
- Eksponentiell skalering
- Gamma-skalering
- Stykkevis-lineær skalering

- Hva gjøres med kontrasten i de mørke og lyse delene av bildet etter slike skaleringer ?
  - Tegn skisse av funksjonene og se på Δf mot Δg

Logaritmiske transformasjoner

- Hvilken av transformasjonene til høyre er brukt her?





(Fig 3.3 i DIP)

"Power-law" (gamma)-transformasjoner

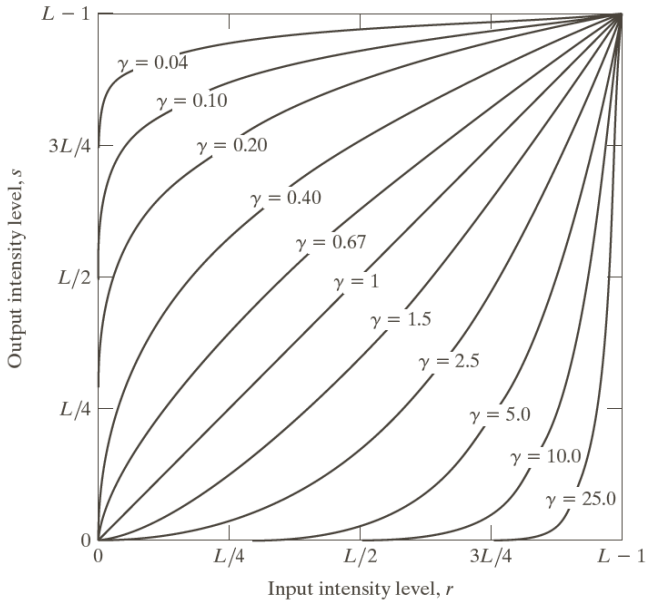
- Mange bildeproduserende apparater har et input/output-forhold som kan beskrives som:

$$s = ci^{\gamma}$$

der s er ut-intensiteten ved en input i

  - $\gamma < 1$: den mørke delen av skalaen strekkes ut
  - $\gamma = 1$: identitets-transform
  - $\gamma > 1$: den lyse delen av skalaen strekkes ut

(Fig 3.6 i DIP)



- Generell kontrast-manipulasjon
  - Brukervennlig med kun én variabel

Histogramutjevning (histogram equalization)

- Maksimal kontrast:
  Alle pikselverdier like sannsynlige
  - Histogrammet er uniformt (flatt)
- Ønsker en transformasjon av bildet slik at det transformerte bildet har uniformt histogram
  - Dvs. at bildet har like mange piksler for hver gråtone
- Tilnærmer ved å flytte på histogramsøyler
- Trenger en oversikt over
  hvor hver søyle skal flyttes: T[i]

Algoritme for histogramutjevning

- For et n×m bilde med G gråtoner:
  - Lag array p, c og T av lengde G med initialverdi 0
- Finn bildets normaliserte histogram
  - Gå igjennom bildet piksel for piksel.
  - Hvis piksel har intensitet i, la p[i]=p[i]+1
  - Deretter skalér, p[i] = p[i]/(n*m), i=0,1,…,G-1
- Lag det kumulative histogrammet c
  - c[0] = p[0], c[i] = c[i-1]+p[i], i=1,2,....,G-1
- Sett inn verdier i transform-array T

  T[i] = Round((G-1)*c[i]), i=0,1,.....,G-1

- **Gå igjennom bildet piksel for piksel,**

  Hvis inn-bildet har intensitet i,
  sett intensitet i ut-bildet til s= T[i]

Histogramtilpasning

- Histogramutjevning gir tilnærmet flatt histogram
- Kan hende at vi ønsker å spesifisere annen form på resultathistogrammet:

1. Gjør histogramutjevning på innbildet, finn $s = T(i)$
2. Spesifiser ønsket nytt histogram $g(z)$
3. Finn den transformen T_g som histogramutjevner $g(z)$, og finn inverstransformen T_g^{-1}
4. Inverstransformer det histogramutjevnede bildet fra punkt 1 ved $z = T_g^{-1}(s)$

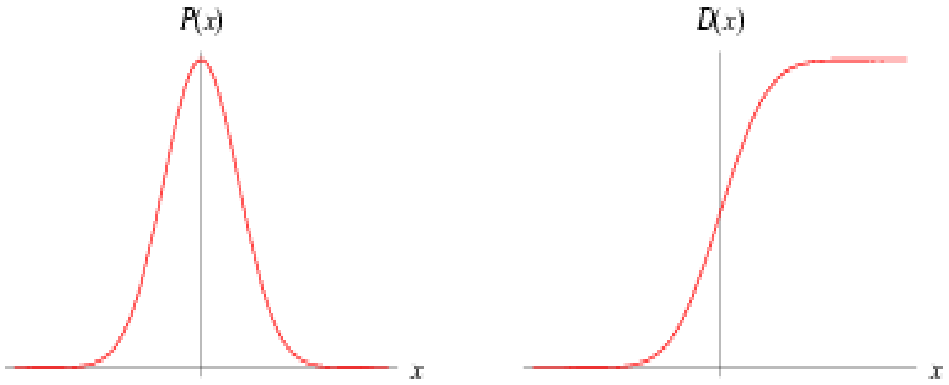
Algoritme - histogramspesifikasjon

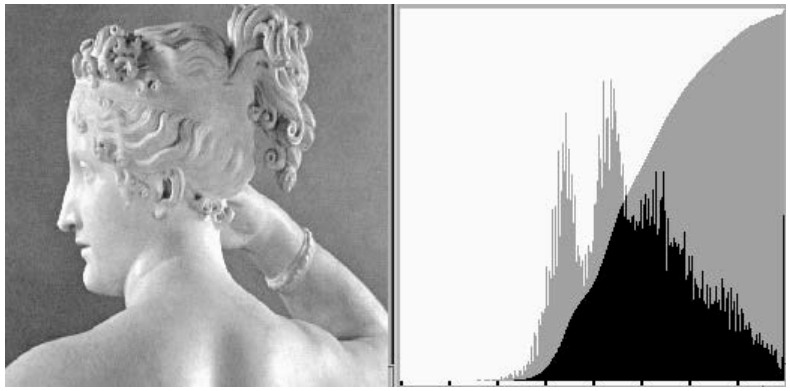
- Finn normalisert histogram, $p_r(i)$, for inputbildet, f(r).
- Lag det kumulative histogrammet c(i).
- Sett s(i) = Round((G-1)* c[i]), i=0,1,.....,G-1

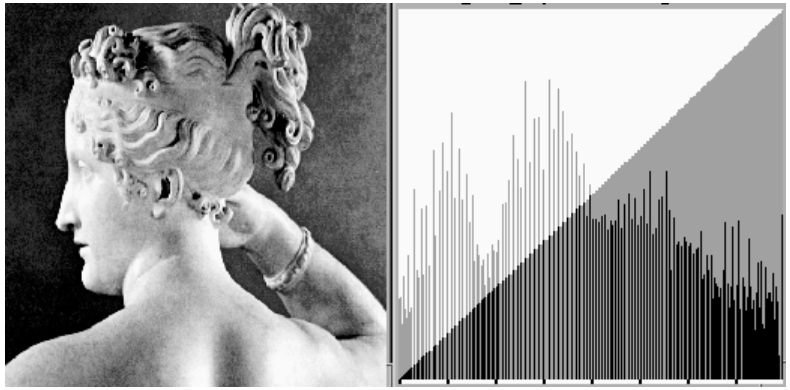
- Gitt ønsket histogram, $p_z(i)$, for bildet g(z).
- Beregn kumulativt spesifisert histogram, skalér, avrund til nærmeste heltall i [0,G-1], og lagre $G_z(q)$.

- For i=0,1,…,G-1, finn q slik at $G_z(q)$ er nærmest mulig s(i), og lagre alle disse matchene i en array $T_{ny}(i)$.
  - Hvis flere q gir samme match, velg den minste.
- Kombiner så de de to transformene til en ny mapping.

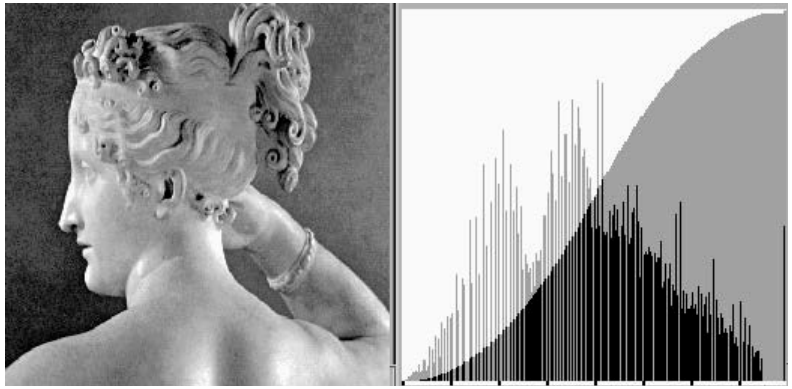
Tilpasning til Gauss-profil







Histogram-utjevnet

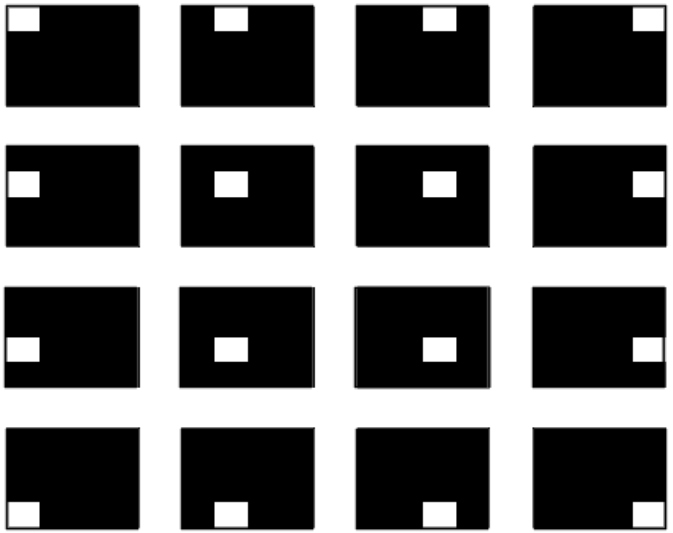


Tilpasset Gauss-form

2D diskret Fourier-transform (2D DFT)

Standardbasis for matriser

Eksempel: Standardbasis for 4x4



- Et gråtonebilde representeres vanligvis som en matrise av gråtoneintensiteter.
- Dette tilsvarer å bruke den såkalte *standardbasisen* for matriser.
- Eksempel: 4x4-gråtonebilder
  - Standardbasisen er de 16 4x4-matrisene vist til venstre.
  - En vektet sum av disse matrisene kan unikt representere enhver 4x4-matrise/gråtonebilde.

Undereksempel:

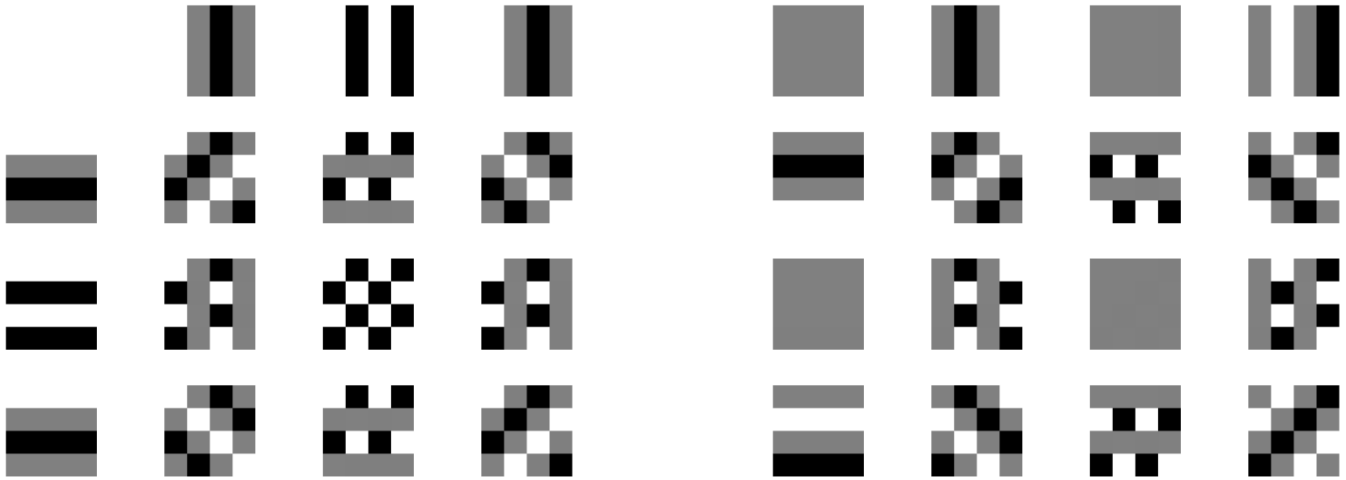
1	3	2	1
5	4	5	3
4	1	1	2
2	3	2	6

$= 1*$  $+ 3*$  $+\ldots+ 6*$ 

I bildene er sort 0 og hvitt 1.

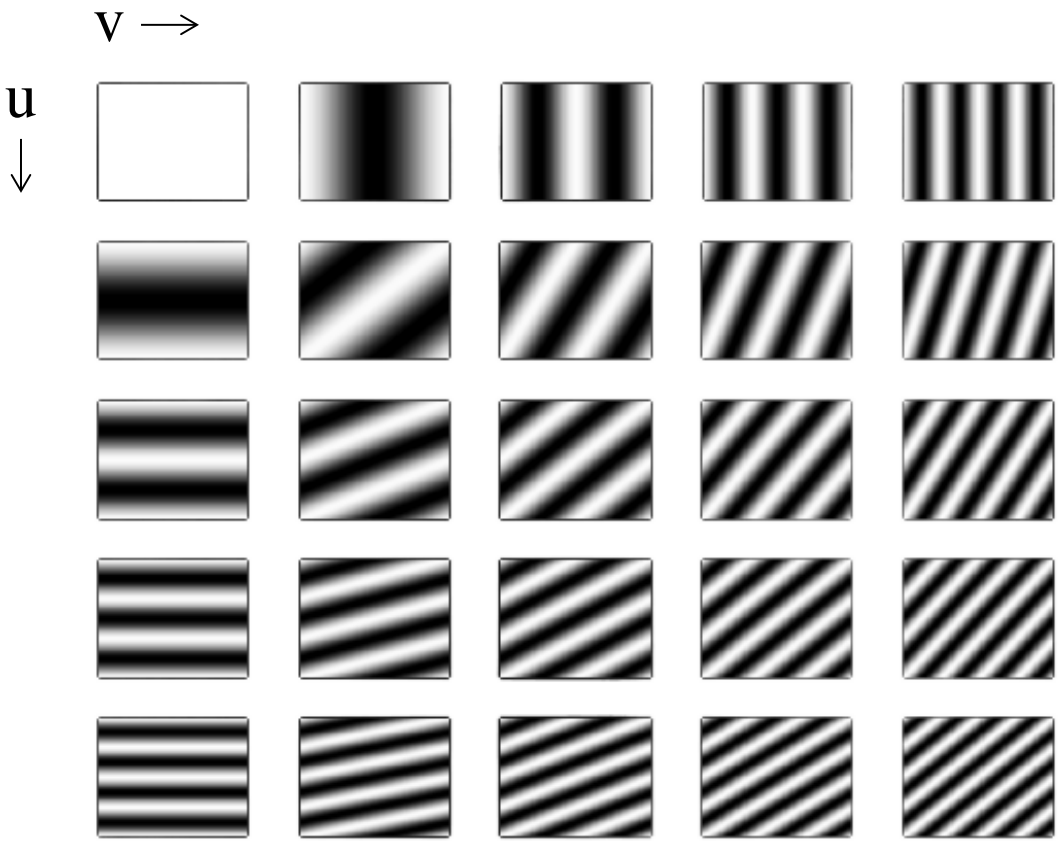
Alternativ basis

- Det finnes mange andre basiser for matriser.
  - Muligheten til å **unikt** representere **enhver** matrise ligger i *basis*.
- **2D DFT** bruker én slik basis som er basert på **sinuser og cosinuser med forskjellige frekvenser**.
  - Disse sinusene og cosinusene er faste for en gitt bildestørrelse (MxN) og kan representeres som hver sin mengde av MN MxN-bilder;



(i bildene er sort -1, grått er 0 og hvitt er 1)

Cosinus-bilder for større bilder



… til $v = N\text{-}1$

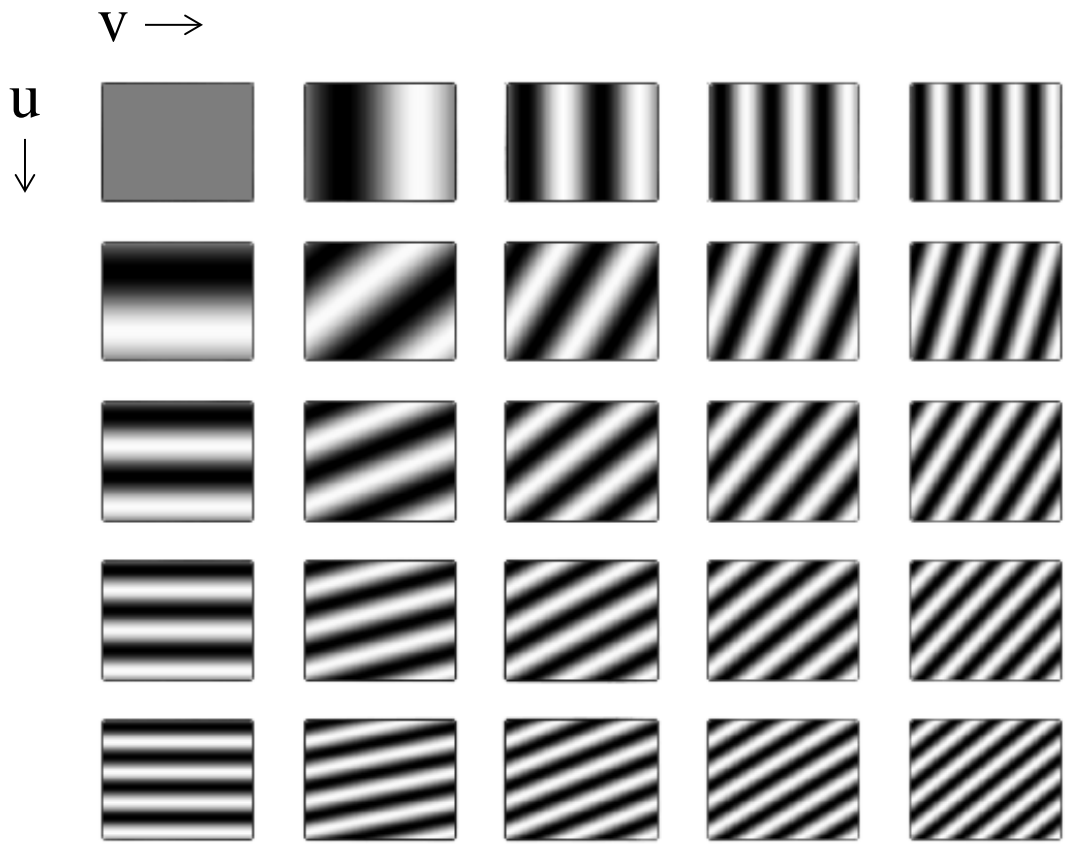
.

.

til $u = M\text{-}1$

I bildene er sort -1 og hvitt 1.

Sinus-bilder for større bilder



… til $v = N\text{-}1$

.

.

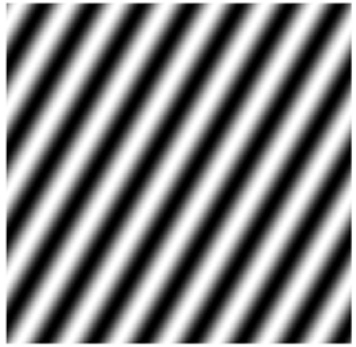
til $u = M\text{-}1$

I bildene er sort -1 og hvitt 1.

Beregning av 2D DFT for en gitt frekvens

- Koeffisienten til 2D DFT av tidligere eksempelbilde for frekvens (0,1):
- real(F(0,1)) = sum($\begin{bmatrix} 1 & 3 & 2 & 1 \\ 5 & 4 & 5 & 3 \\ 4 & 1 & 1 & 2 \\ 2 & 3 & 2 & 6 \end{bmatrix}$ x $\begin{bmatrix} 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}$) = 2
- imag(F(0,1)) = sum($\begin{bmatrix} 1 & 3 & 2 & 1 \\ 5 & 4 & 5 & 3 \\ 4 & 1 & 1 & 2 \\ 2 & 3 & 2 & 6 \end{bmatrix}$ x $\begin{bmatrix} 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & 1 \end{bmatrix}$) = 1
- Altså er *F(0,1) = 2+j*.

Grunnleggende om 2D DFT

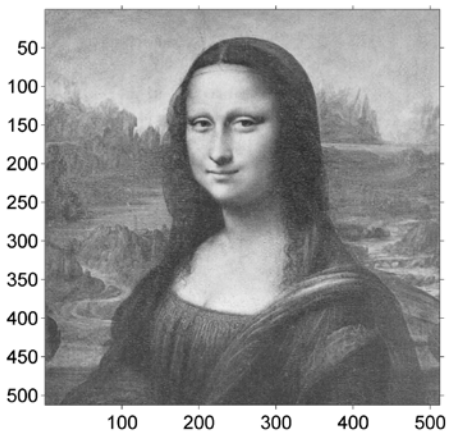
- real(F(u,v)) = sum( x )

  realdelen til 2D DFT-en i frekvens (u,v) = sum(bildet punkt-multiplisert cosinus-bildet for frekvens (u,v))

- Tilsvarende for imaginærdelen og sinus-bildet.
- **Hvert punkt** i 2D DFT-en beskriver altså noe ved **hele bildet**.

Eksempel og filtrering i Fourier-domenet

Eksempel: 5x5-middelverdifiltrering og konvolusjonsteoremet



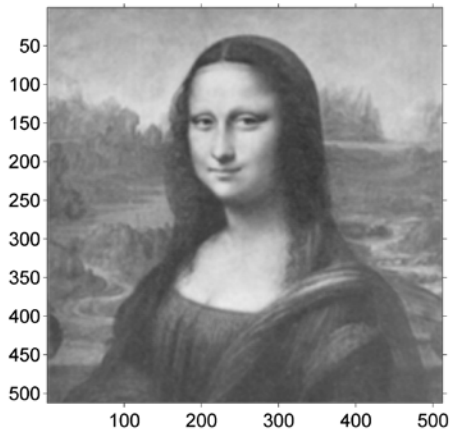
★

$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

Nullutvidet til 512x512

=

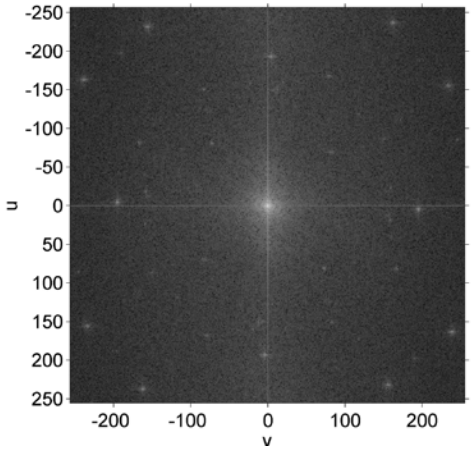
(sirkulær indeksering og bevart bildestørrelse)



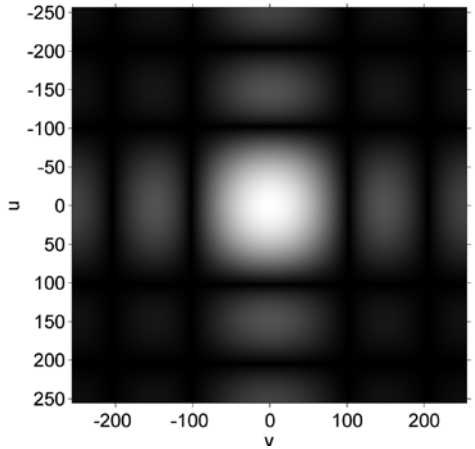
2D DFT ↓ ↑ 2D IDFT

2D DFT ↓ ↑ 2D IDFT

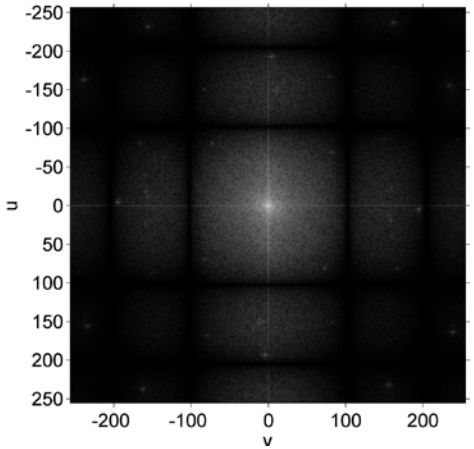
2D DFT ↓ ↑ 2D IDFT



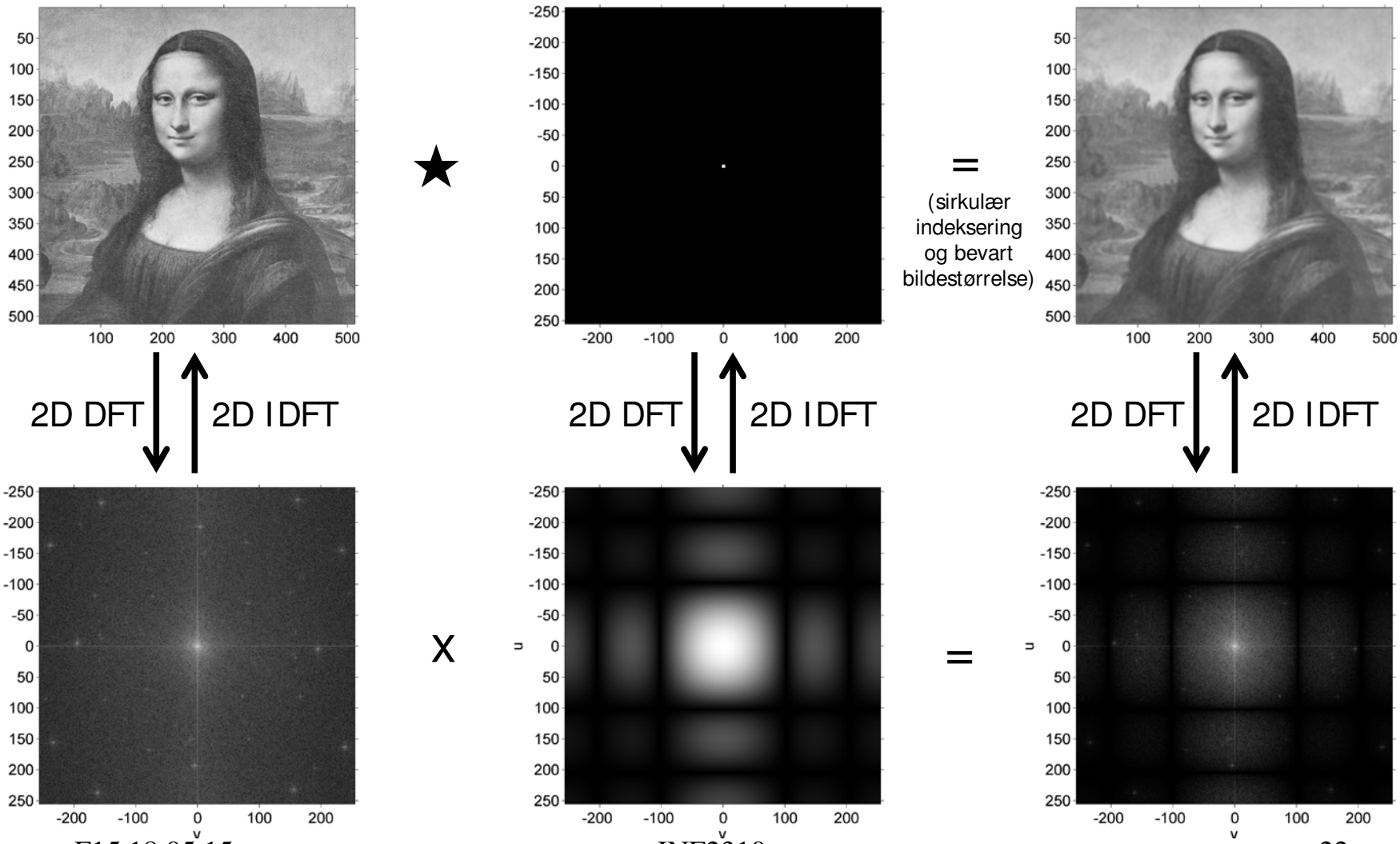
x



=



Eksempel: 5x5-middelverdifiltrering og konvolusjonsteoremet



Anvendelse av konvolusjonsteoremet

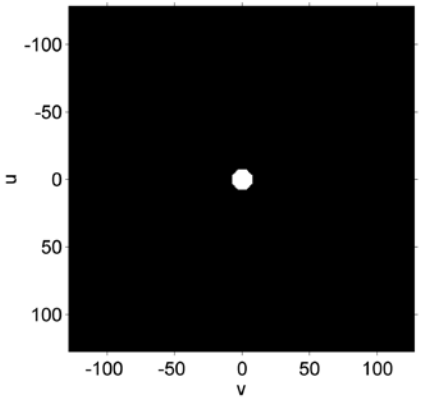
- Design av konvolusjonsfiltre
  med bestemte frekvensegenskaper.
  - Designe konvolusjonsfilteret i Fourier-domenet
    slik at vi har bedre kontroll på dets frekvensegenskaper.

- Analyse av konvolusjonsfiltre.
  - 2D DFT-en til et konvolusjonsfilter gir oss innblikk i hvordan filteret vil påvirke de forskjellig frekvenskomponentene.

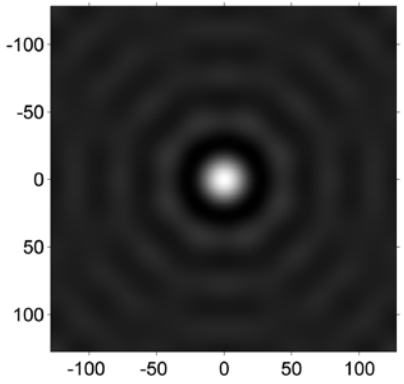
- Rask implementasjon av større konvolusjonsfiltre.

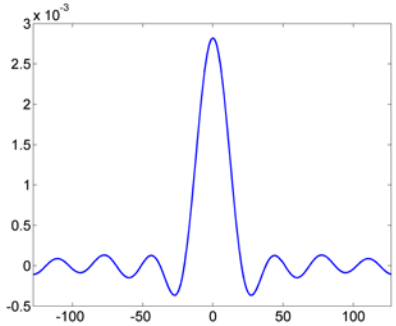
Filter-design i Fourier-domenet

Romlig representasjon av ideelt lavpassfilter



2D IDFT

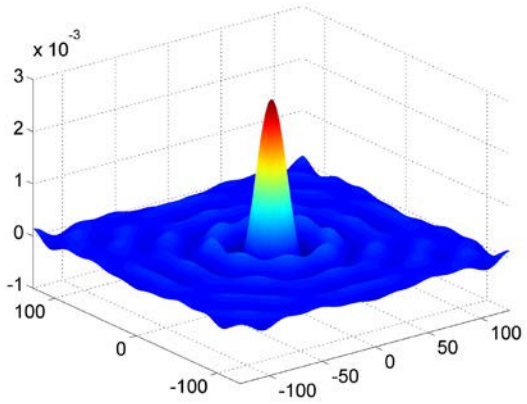




Den romlige representasjonen er en trunkert sinc-funksjon. Sinc-funksjonen er definert som:

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

og sinc(0) = 1.



- Vi får *ringing* i bildet.
  - Husk også tommelfingerregel fra forrige forelesning:
    Smal/bred struktur i bildet ⇔ Bred/smal struktur i Fourier-spekteret

Eksempel: Ideelt lavpassfilter

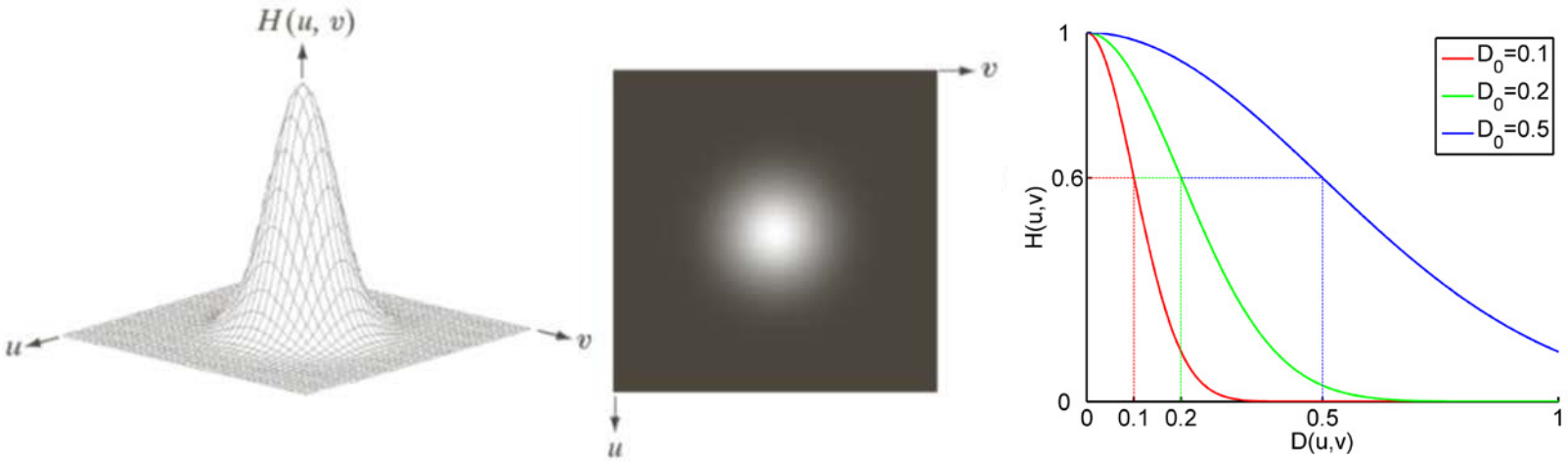
Original

Filtrert med $D_0 = 0,2$

Filtrert med $D_0 = 0,3$

I god nok oppløsning kan striper/ringinger sees ut fra markante kanter i de to filtrerte bildene. Det er dette vi kaller *ringing*.

Gaussisk lavpassfilter

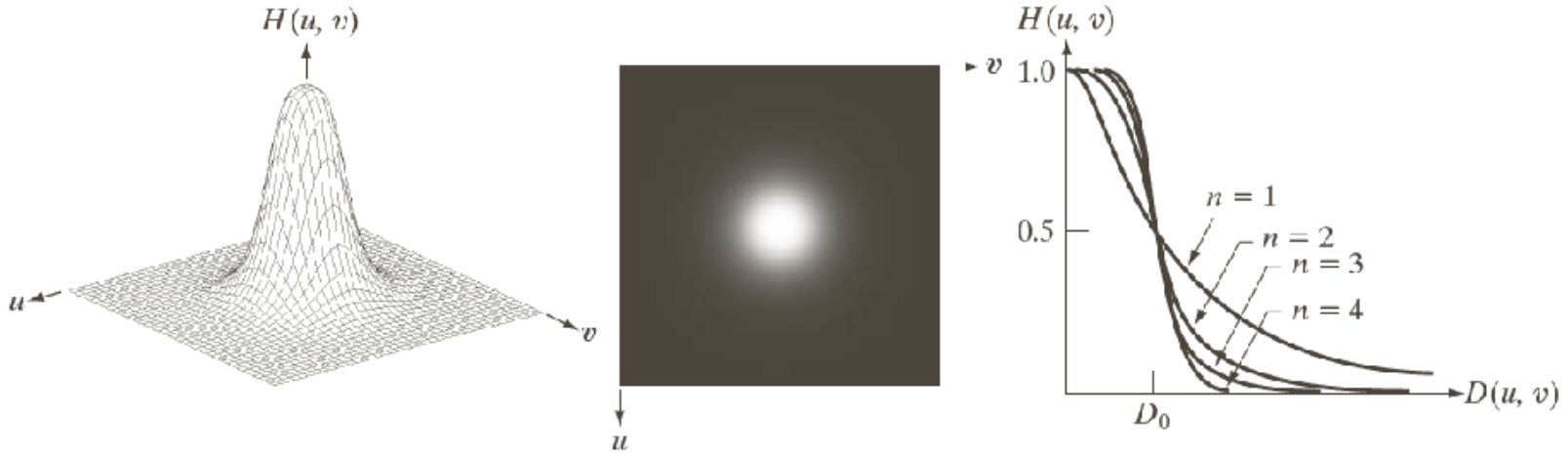


a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Husk tommelfingerregelen:
Smal/bred struktur i bildet ⇔ Bred/smal struktur i Fourier-spekteret

Butterworth lavpassfilter



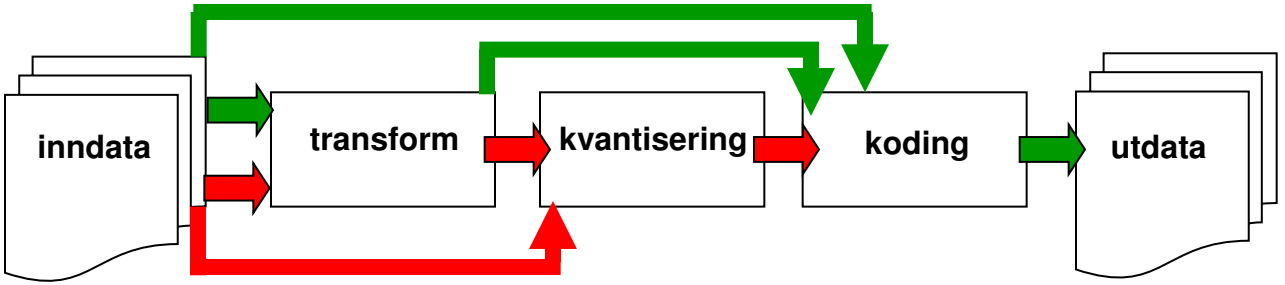
a b c

FIGURE 4.44 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

Kompresjon og koding

Kompresjon

- Kompresjon kan deles inn i tre steg:
  - **Transform** - representer bildet mer kompakt.
  - **Kvantisering** - avrund representasjonen.
  - **Koding** - produser og bruk en kodebok.



- Kompresjon kan gjøres:
  - **Eksakt / tapsfri** (eng.: *lossless*) – følg de grønne pilene.
    - Kan da eksakt rekonstruere det originale bildet.
  - **Ikke-tapsfri** (eng.: *lossy*) – følg de røde pilene.
    - Kan da (generelt) ikke eksakt rekonstruere bildet.
    - Resultatet kan likevel være «godt nok».
  - Det finnes en mengde ulike metoder innenfor begge kategorier.

Ulike typer redundans

- Psykovisuell redundans. → Mer generelt: **Irrelevant informasjon**: Unødvendig informasjon for anvendelsen, f.eks. for visuell betraktning av hele bildet.
  - Det finnes informasjon vi ikke kan se.
    - Eksempler på enkle muligheter for å redusere redundansen: Subsample eller redusere antall biter per piksel.
- Interbilde-redundans.
  - Likhet mellom nabobilder i en tidssekvens.
    - Eks.: Lagre noen bilder i tidssekvensen og ellers bare differanser.
- Intersampel-redundans.
  - Likhet mellom nabopiksler.
    - Eks.: Hver linje i bildet kan løpelengde-transformeres.
- Kodings-redundans.
  - Enkeltsymboler (enkeltpiksler) blir ikke lagret optimalt.
  - Gitt som gjennomsnittlig kodelengde minus et teoretisk minimum.
    - Velg en metode som er «grei» å bruke og gir liten kodingsredundans.

Entropi

- Gjennomsnittlig informasjonsinnhold i sekvensen, også kalt gjennomsnittlig informasjon per symbol, er da:

$$H = \sum_{i=0}^{G-1} p_i I(s_i) = -\sum_{i=0}^{G-1} p_i \log_2 p_i$$

Hvis $p(s_i)=0$ lar vi det tilhørende entropibidraget, $0\log_2 0$, være 0.

- H er entropien til sekvensen av symbolene.

- **Entropien setter en nedre grense for hvor kompakt sekvensen kan representeres.**
  - Gjelder bare hvis vi koder hvert symbol for seg.

Eksempel: Huffman-koding

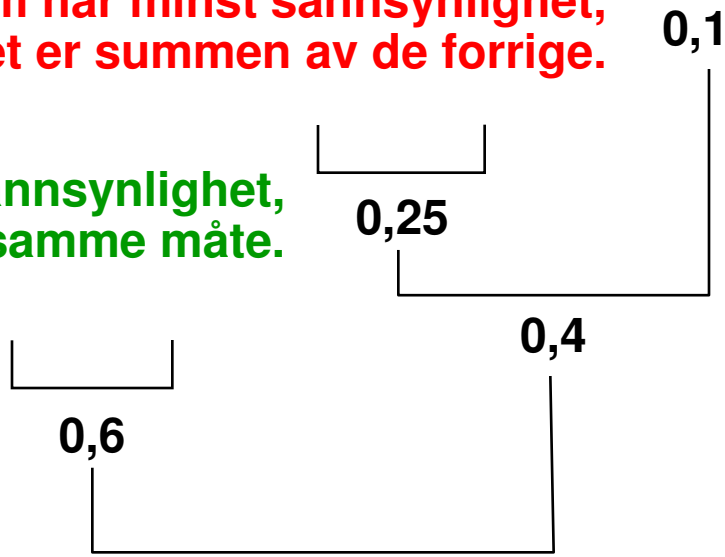
- La oss finne Huffman-koden til modellen som består av følgende seks begivenheter med sannsynligheter:

Begivenhet	A	B	C	D	E	F
Sannsynlighet	0,3	0,3	0,13	0,12	0,1	0,05

Slå sammen de to gruppene som har minst sannsynlighet, Den nye gruppens sannsynlighet er summen av de forrige.

Finn de to som nå har minst sannsynlighet, og slå dem sammen på samme måte.

Fortsett til det er bare to igjen.

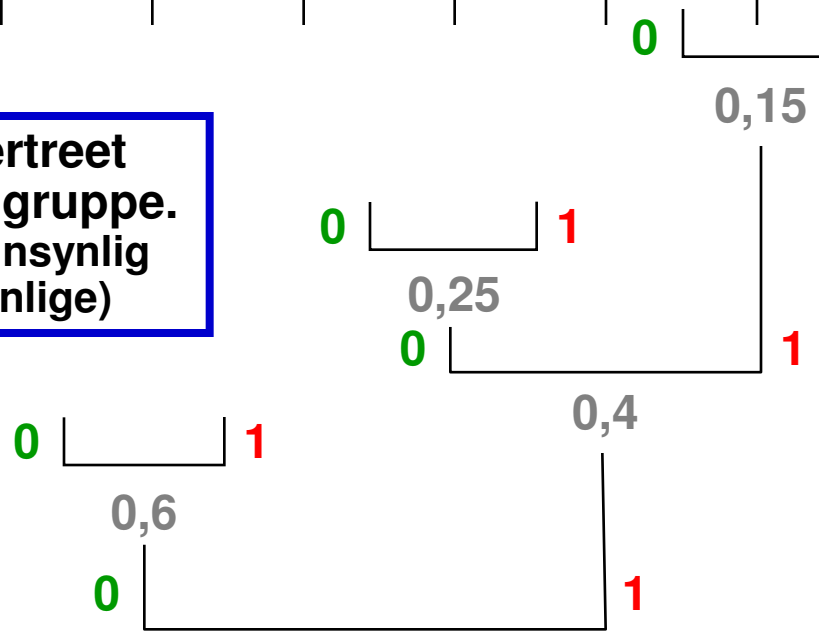


Eksempel: Huffman-koding

- La oss finne Huffman-koden til modellen som består av følgende seks begivenheter med sannsynligheter:

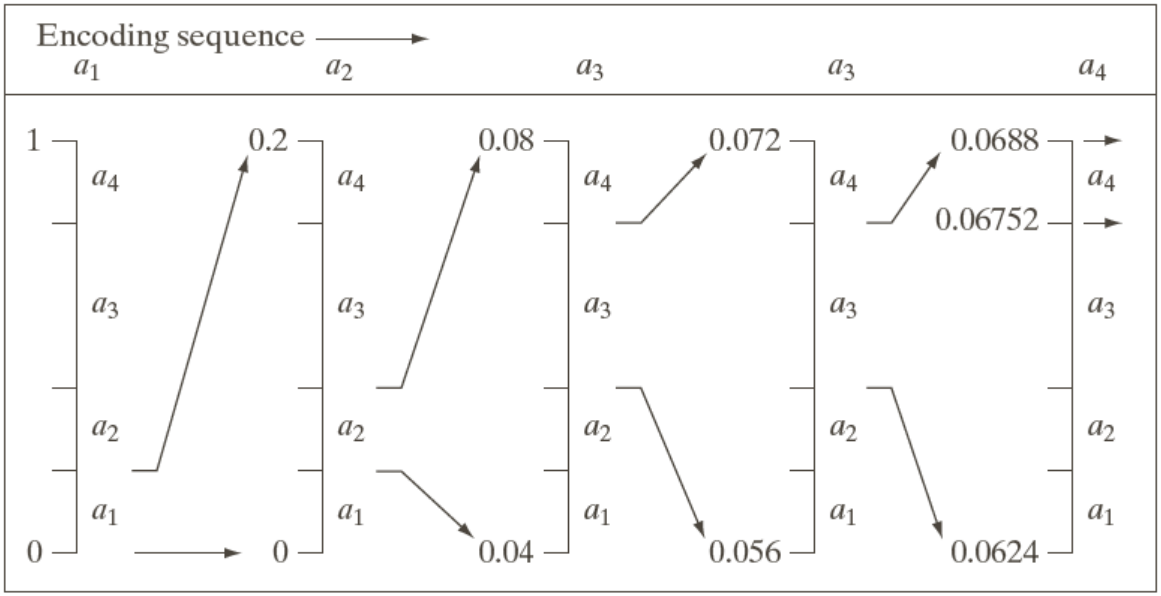
Begivenhet	A	B	C	D	E	F
Sannsynlighet	0,3	0,3	0,13	0,12	0,1	0,05

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



Eksempel: Aritmetisk koding

- Sannsynlighetsmodell: $P(a_1) = P(a_2) = P(a_4) = 0{,}2$ og $P(a_3) = 0{,}4$
- Melding/symbolsekvens: $a_1 a_2 a_3 a_3 a_4$



- a_1 ligger i intervallet [0, 0,2)
- $a_1 a_2$ ligger i intervallet [0,04, 0,08)
- $a_1 a_2 a_3$ ligger i intervallet [0,056, 0,072)
- $a_1 a_2 a_3 a_3$ ligger i intervallet [0,0624, 0,0688)
- $a_1 a_2 a_3 a_3 a_4$ ligger i intervallet [0,06752, 0,0688)

Representasjon av intervall

- Eksempel: Intervallet er [0,5232, 0,53184).
  - Finner første forskjellig bit i binærrepresentasjonen:
    - Skriver **$0{,}5232_{10}$** på binærform:

$$
\begin{aligned}
2 * 0{,}5232 &= 1{,}0464 && \Rightarrow d_1 = \mathbf{1},\ \text{rest} = 0{,}0464\\
2 * 0{,}0464 &= 0{,}0928 && \Rightarrow d_2 = \mathbf{0},\ \text{rest} = 0{,}0928\\
2 * 0{,}0928 &= 0{,}1856 && \Rightarrow d_3 = \mathbf{0},\ \text{rest} = 0{,}1856\\
2 * 0{,}1856 &= 0{,}3712 && \Rightarrow d_4 = \mathbf{0},\ \text{rest} = 0{,}3712\\
2 * 0{,}3712 &= 0{,}7424 && \Rightarrow d_5 = \mathbf{0},\ \text{rest} = 0{,}7424
\end{aligned}
$$

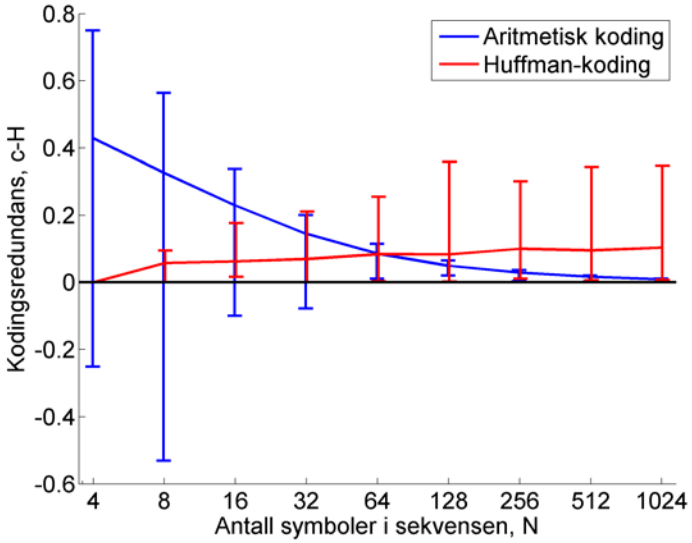
    - Skriver **$0{,}53184_{10}$** på binærform:

$$
\begin{aligned}
2 * 0{,}53184 &= 1{,}06368 && \Rightarrow d_1 = \mathbf{1},\ \text{rest} = 0{,}06368\\
2 * 0{,}06368 &= 0{,}12736 && \Rightarrow d_2 = \mathbf{0},\ \text{rest} = 0{,}12736\\
2 * 0{,}12736 &= 0{,}25472 && \Rightarrow d_3 = \mathbf{0},\ \text{rest} = 0{,}25472\\
2 * 0{,}25472 &= 0{,}50944 && \Rightarrow d_4 = \mathbf{0},\ \text{rest} = 0{,}50944\\
2 * 0{,}50944 &= 1{,}01888 && \Rightarrow d_5 = \mathbf{1},\ \text{rest} = 0{,}01888
\end{aligned}
$$

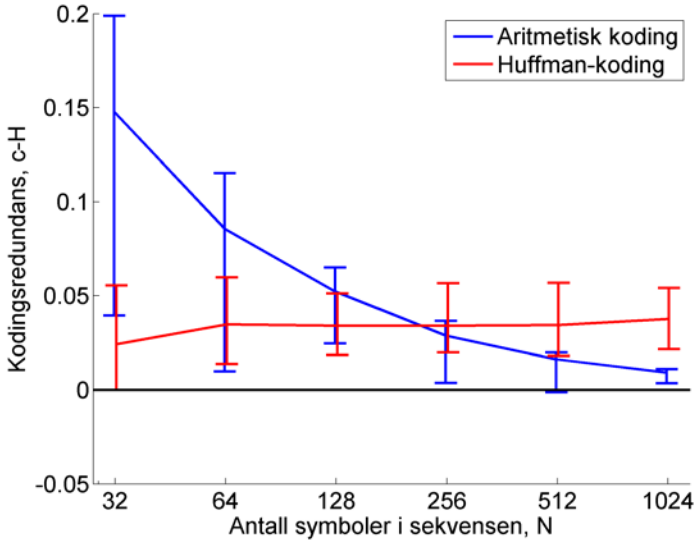
  - Siden den siste resten til den øvre grensa er større enn 0 må **$0{,}10001_2 = 0{,}53125_{10}$** være et tall i intervallet.
  - $0{,}53125_{10}$ er det tallet i intervallet med kortest binær-representasjon.

Aritmetisk koding vs Huffman-koding

- Aritmetisk koding: Bedre kompresjon jo lenger symbolsekvensen er.
- Huffman-koding: Bedre kompresjon jo flere symboler i alfabetet.
- Aritmetisk koding komprimerer typisk litt bedre enn Huffman-koding, men er mer regnekrevende å utføre.
  - For vanlige bilder, dvs. med relativt få symboler i alfabetet og (potensielt sett) mange symboler i sekvensen.



Med G=4 symboler i alfabetet.



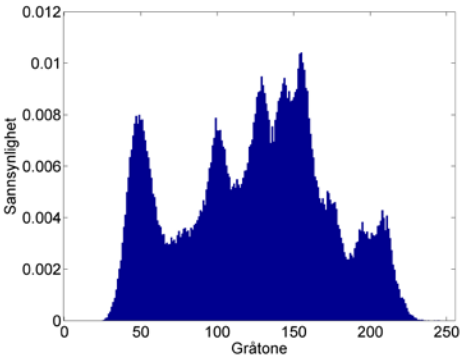
Med G=20 symboler i alfabetet.

Differansetransform

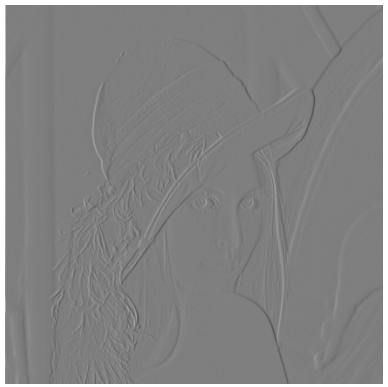
- Utnytter at horisontale nabopiksler ofte har ganske lik gråtone.

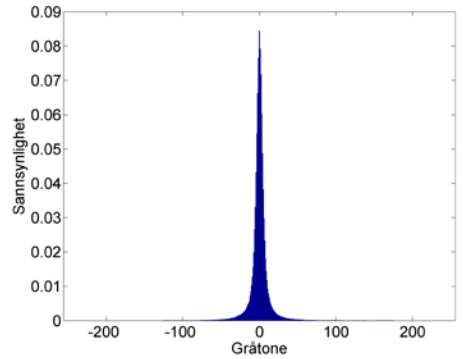
- Gitt en rad i bildet med gråtoner:
  $f_1, \ldots, f_N$ der $0 \leq f_i \leq 2^b-1$
- Transformer (reversibelt) til
  $g_1 = f_1, \; g_2 = f_2-f_1, \; \ldots, \; g_N = f_N-f_{N-1}$

- Merk at: $-(2^b-1) \leq g_i \leq 2^b-1$
  - Må bruke b+1 biter per g_i hvis vi skal tilordne like lange kodeord til alle mulig verdier.
- Ofte er de fleste differansene nær 0.
  - Naturlig binærkoding av differansene er ikke optimalt.



Entropi ≈ 7,45 => CR ≈ 1,1





Entropi ≈ 5,07 => CR ≈ 1,6

Løpelengde-transform

- Ofte inneholder bildet objekter med lignende gråtoner, f.eks. svarte bokstaver på hvit bakgrunn.
- Løpelengde-transformen (eng.: *run-length transform*) **utnytter når horisontale nabopiksler har samme gråtone**.
  - Merk: Krever ekte likhet, ikke bare omtrent like.
  - Løpelengde-transformen komprimerer bedre ettersom kompleksiteten i bildet blir mindre.
- Løpelengde-transformen er reversibel.
- Hvis pikselverdiene til en rad er:

  33333355555555544777777 (24 tall)
- Så starter løpelengde-transformen fra venstre og finner tallet 3 gjentatt 6 ganger etter hverandre, og returnerer derfor tallparet (3,6). **Formatet er: (tall, løpelengde)**
- For hele sekvensen vil løpelengdetransformen gi de 4 tallparene:

  (3,6), (5,10), (4,2), (7,6) (merk at dette bare er 8 tall)
- Kodingen avgjør hvor mange biter vi bruker for å lagre tallene.

Eksempel: LZW-transform

- Alfabetet: a, b og c med koder 0, 1 og 2, henholdsvis.
- Meldingen: ababcbababaaaaabab (18 symboler)
- LZW-sender: ny streng = sendt streng pluss neste usendte symbol
- LZW-mottaker: ny streng = nest siste streng pluss første symbol i sist tilsendte streng

Ser	Sender	Senders liste	Mottar	Tolker	Mottakers liste
		a=0, b=1, c=2			a=0, b=1, c=2
a	0	ab=3	0	a	
b	1	ba=4	1	b	ab=3
ab	3	abc=5	3	ab	ba=4
c	2	cb=6	2	c	abc=5
ba	4	bab=7	4	ba	cb=6
bab	7	baba=8	7		

» Vi mottar kode 7, men denne koden finnes ikke i listen!

» Fra ny-streng-oppskriften vet vi at kode 7 ble laget ved: ba + ?

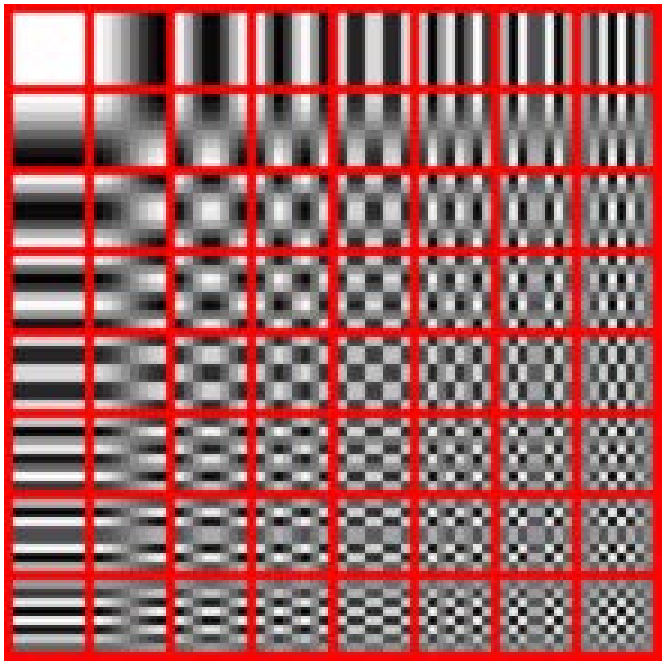
» Siden kode 7 nå sendes, må: ? = b => 7 = ba + b = bab

2D diskret cosinus-transform

- Grunnpilaren i ikke-tapsfri JPEG-kompresjon er 2D DCT:

$$F(u,v) = \frac{2}{\sqrt{MN}} c(u)c(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi u}{M}\left(x+\frac{1}{2}\right)\right] \cos\left[\frac{\pi v}{N}\left(y+\frac{1}{2}\right)\right], \qquad c(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{hvis } \xi = 0 \\ 1 & \text{ellers} \end{cases}$$

- Sterkt relatert til 2D DFT.
- I JPEG transformerer vi 8x8-blokker så vi bruker bare de 64 «8x8-cosinus-bildene»:
  - For hvert bilde vi går til høyre eller ned så økes den tilhørende frekvenskomponenten med 0,5.
  - Husk: I 2D DFT økte frekvenskomp. med 1, som lagde par med like cosinus-bilder.
  - Husk: I 2D DFT hadde vi også noen sinus-bilder.
  - 2D DCT-koeffisientene beregnes analogt med det vi gjorde for 2D DFT; summere punktproduktet mellom 8x8-blokken og hvert «cosinus-bilde».
  - 2D DCT beregnes hurtig ved å forhåndsberegnes de 64 «8x8-cosinus-bildene».



Rekonstruksjonsfeil i gråtonebilder

- JPEG-kompresjon kan gi **8×8-piksels blokk-artefakter**, **glatting** og **ringinger**.
- Avhengig av vektmatrisen
  - som bestemmer hvor mange koeffisienter som lagres, og hvor presist disse lagres.

Blokk-artefakter

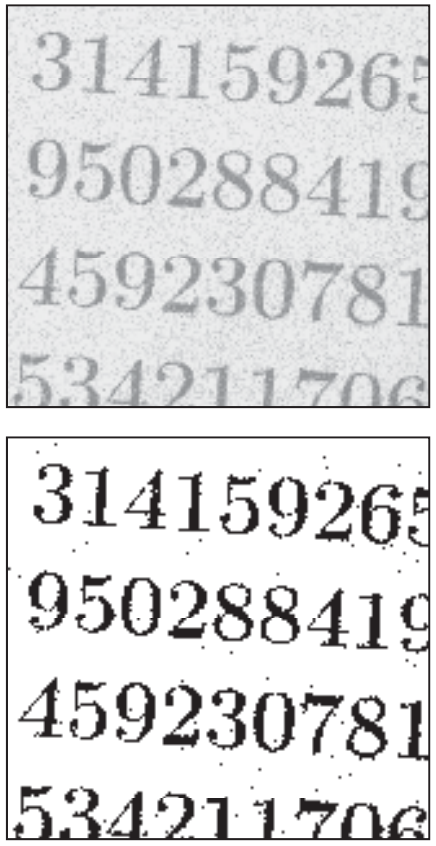
- Blokk-artefaktene øker med kompresjonsraten.

- Øverst: kompresjonsrate = 25
- Nederst: kompresjonsrate = 52

Segmentering

Hva er segmentering?

- Segmentering er en prosess som deler opp bildet i meningsfulle regioner.
- Segmentering er ett av de viktigste elementene i et komplett bildeanalyse-system.
- I segmentering får vi fram regioner og objekter som senere skal beskrives og gjenkjennes.
- I det enkleste tilfellet har vi bare to typer regioner:
  - Forgrunn
  - Bakgrunn



Eksempel:
finne symboler for OCR

To segmenterings-kategorier

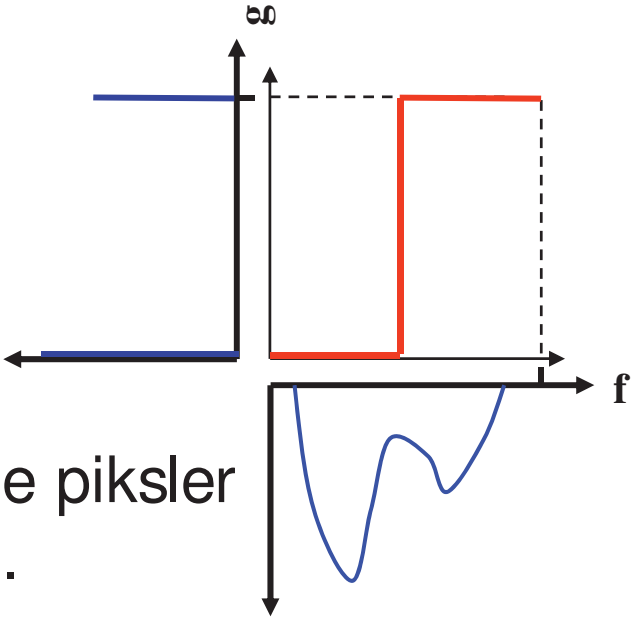
- Vi skiller mellom to kategorier av metoder, basert på hhv. likhet og diskontinuitet mellom pikslene i bildet.

1. Ved terskling og region-basert segmentering får vi fram de pikslene som ligner hverandre.
   Dette gir alle pikslene i objektet.
2. Ved kant-basert segmentering finner vi basis-elementer i omrisset til objektene:
   - Kant-punkter, linje-punkter, hjørne-punkter..
   - I neste steg:
     - Tynner brede kanter
     - Lenker punktene sammen

Terskling

- Hvis vi har grunn til å anta at objektene f.eks. er lysere enn bakgrunnen, kan vi sette en terskel T og lage oss et binært ut-bilde g(x,y) ved mappingen:

$$g(x,y) = \begin{cases} 0 \text{ hvis } & f(x,y) \le T \\ 1 \text{ hvis } & f(x,y) > T \end{cases}$$

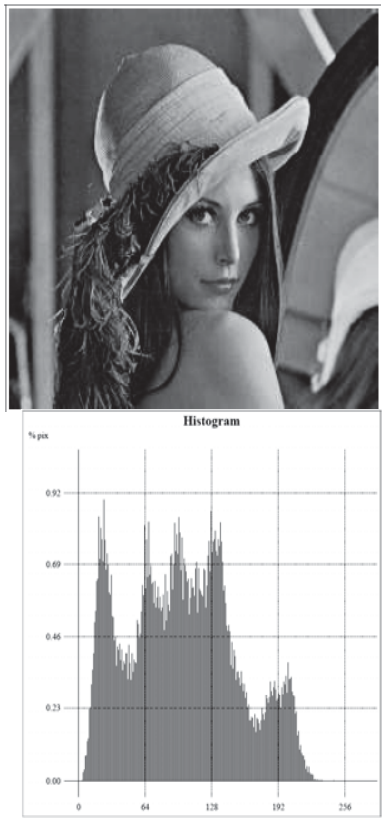


- Da har vi fått et ut-bilde g(x,y) med bare to mulige verdier.
- Med riktig valg av T vil nå de fleste piksler med g(x,y)=1 være objekt-piksler.

Flernivå terskling

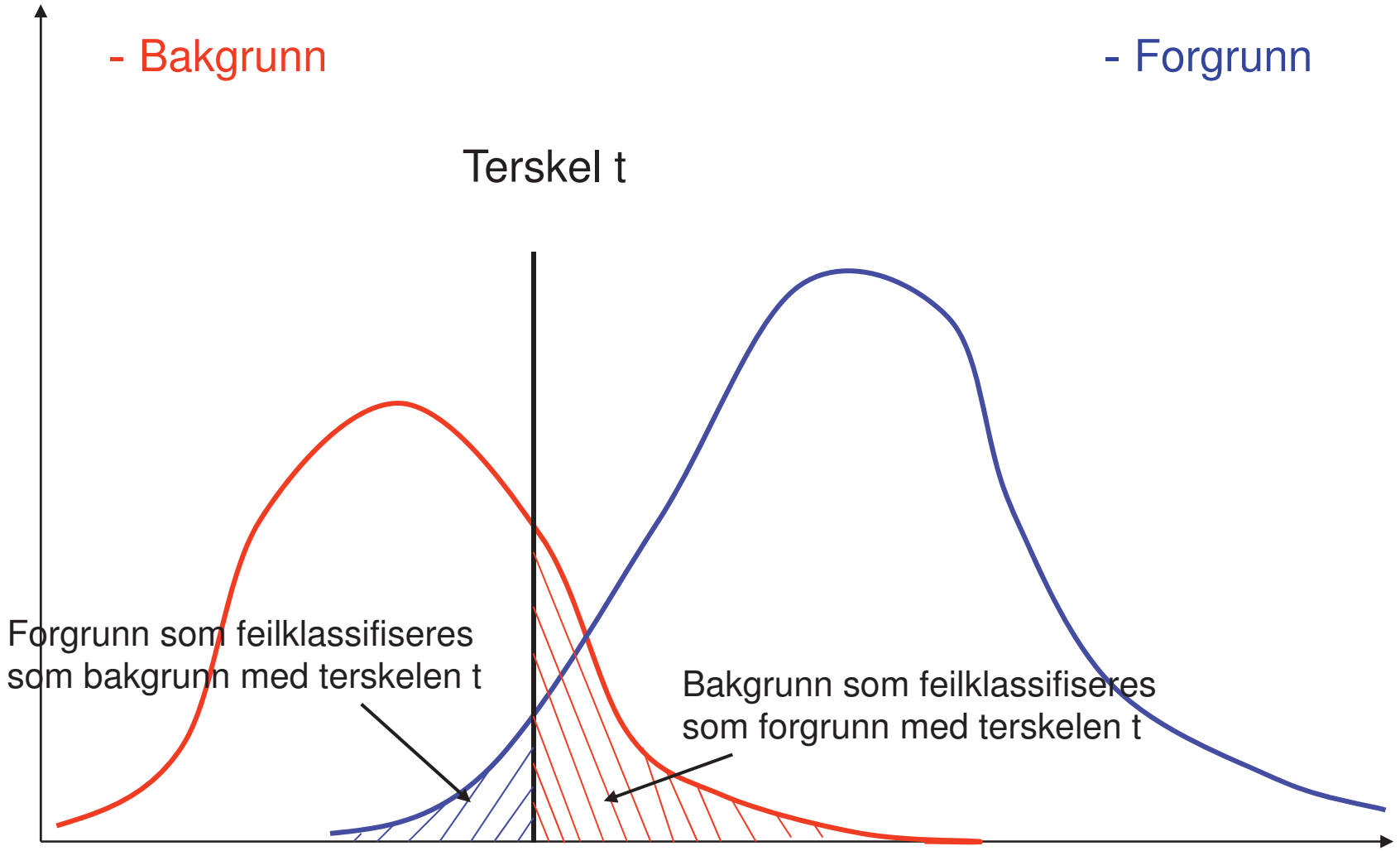
- Har vi flere klasser av objekter med forskjellig intensitet, så kan vi utvide dette til M gråtone-intervaller ved hjelp av M-1 terskler.

$$g(x,y) = \begin{cases} 0 & \text{hvis} \quad 0 \le f(x,y) \le t_1 \\ 1 & \text{hvis} \quad t_1 \le f(x,y) \le t_2 \\ \ldots & \\ M-1 & \text{hvis} \quad t_{M-1} \le f(x,y) \le G-1 \end{cases}$$



- Terskling er et spesialtilfelle av klassifikasjon.
- Jfr. histogram-utjevning med noen få gråtoner.

Klassifikasjons-feil ved terskling

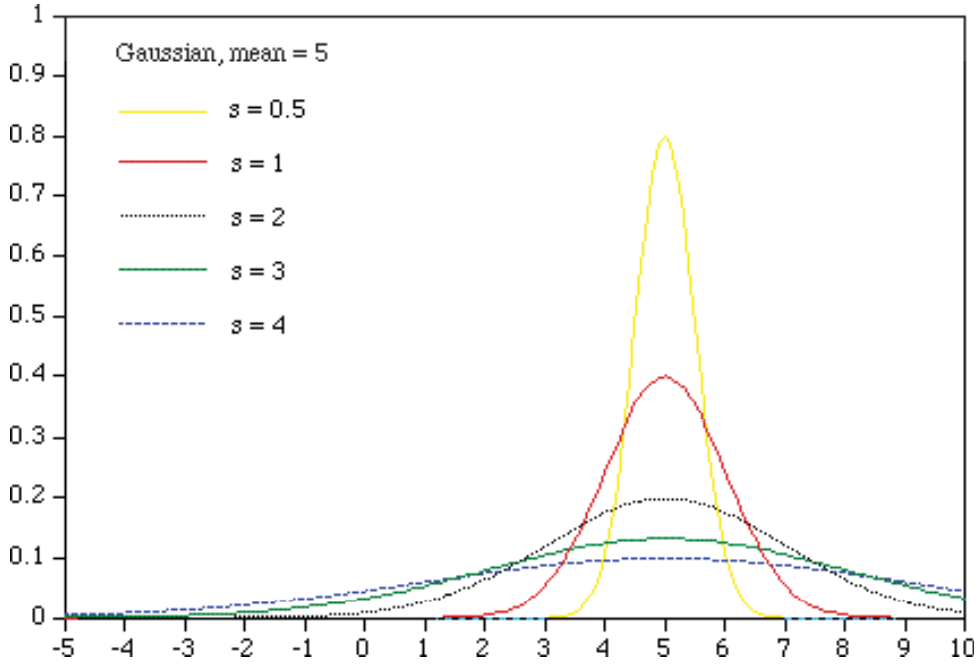


Fordelinger, standardavvik og varians

- En Gauss-fordeling (normalfordeling) er gitt ved
  - middelverdien μ
  - variansen σ^2:

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Varians: σ^2,
- Standardavvik: σ



Klassifikasjonsfeil ved terskling

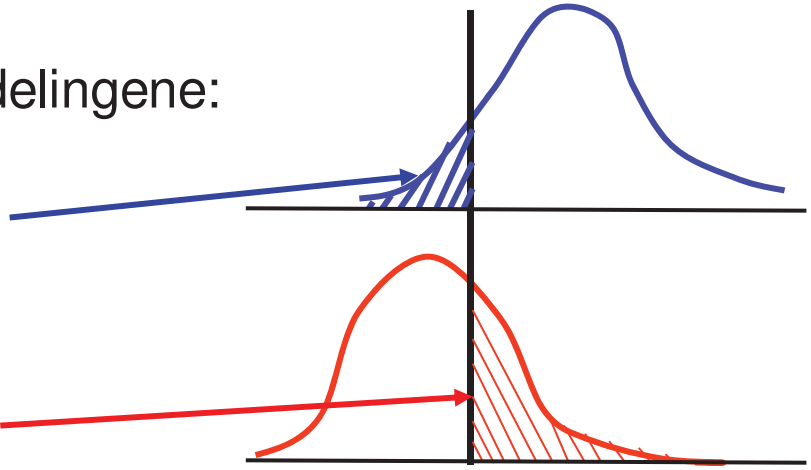
- Anta at histogrammet er en sum av to fordelinger *b(z)* og *f(z)*, *b* og *f* er ***normaliserte*** bakgrunns- og forgrunns-histogrammer.
- La *F* og *B* være ***a priori sannsynlighet*** for bakgrunn og forgrunn (*B+F=1*)
- Det normaliserte histogrammet til bildet kan da skrives

$$p(z) = B \cdot b(z) + F \cdot f(z)$$

- Sannsynlighetene for å feilklassifisere et piksel, gitt en terskelverdi t, finner vi fra de normaliserte fordelingene:

$$E_B(t) = \int_{-\infty}^{t} f(z)dz$$

$$E_F(t) = \int_{t}^{\infty} b(z)dz$$



Finn den T som minimerer feilen

$$E(t) = F\int_{-\infty}^{t} f(z)dz + B\int_{t}^{\infty} b(z)dz$$

- Deriverer E(t) mhp. t vha. Leibnitz regel for derivasjon av integraler.
- Setter den deriverte lik 0 og får:

VIKTIG !!!

$$\frac{dE(t)}{dt} = 0 \Rightarrow \boxed{F \cdot f(T) = B \cdot b(T)}$$

- Merk at dette er en generell løsning som gir minst feil.

- Det er ingen restriksjoner mht. fordelingene *b* og *f* !!

En enkel tersklings-algoritme

- Start med terskel-verdi t = middelverdien til alle pikslene i bildet.
  - Finn middelverdien ($\mu_1(t)$) av alle piksler som er mørkere enn terskelen
  - Finn middelverdien ($\mu_2(t)$) av alle piksler som er lysere enn terskelen.
- La ny terskel-verdi være

$$t = \frac{1}{2}\left(\mu_1(t) + \mu_2(t)\right)$$

- Gjenta de to punktene ovenfor til terskelen ikke flytter seg mer.
- Dette kalles Ridler og Calvard's metode
- Dette gjøres i algoritmen på side 742 i GW.
  - Hvilke betingelser må være oppfylt for at metoden skal virke?
  - Når vil denne metoden svikte?

Samme algoritme: bruk histogrammet!

- Når vi skal terskle et ukjent bilde,
  kjenner vi ikke μ_B eller μ_F (og heller ikke σ_B og σ_F)
- Vi kan iterativt estimere μ_B og μ_F fra bildets histogram
  gitt den terskelen t_k vi bruker:

$$t_{k+1} = \frac{1}{2}\left[\mu_1(t_k) + \mu_2(t_k)\right] = \frac{1}{2}\left[\frac{\sum_{i=0}^{t_k} ip(i)}{\sum_{i=0}^{t_k} p(i)} + \frac{\sum_{i=t_k+1}^{G-1} ip(i)}{\sum_{i=t_k+1}^{G-1} p(i)}\right]$$

- Merk at estimatene $\mu_1(t_k)$ og $\mu_2(t_k)$ finnes fra
  trunkerte fordelinger (trunkert ved terskelen t_k)

Otsu's metode - motivasjon

- Anta at vi har et gråtonebilde med G gråtoner, med normalisert histogram p(i).
- Anta at bildet inneholder to populasjoner av piksler, slik at pikslene innenfor hver populasjon er noenlunde like, mens populasjonene er forskjellige.
- **Målsetting:**

  Vi vil finne en terskel T slik at hver av de to klassene som oppstår ved tersklingen blir mest mulig homogen, mens de to klassene bli mest mulig forskjellige.

  – Klassene er homogene:
  variansen i hver av de to klassene er minst mulig.

  – Separasjonen mellom klassene er stor:
  avstanden mellom middelverdiene er størst mulig.

Otsu's metode; oppsummering

- Gitt et NxM pikslers bilde med G gråtoner.
- Finn bildets histogram, h(k), k= 0,1,2,..,G-1.
- Finn bildets normaliserte histogram:

$$p(k) = \frac{h(k)}{MN}, \quad k = 0,1,2,...,G-1$$

- Beregn kumulativt normalisert histogram:

$$P_1(k) = \sum_{i=0}^{k} p(i), \quad k = 0,1,2,...,G-1$$

- Beregn kumulativ middelverdi, μ(k):

$$\mu(k) \equiv \sum_{i=0}^{k} ip(i), \quad k = 0,1,2,...,G-1$$

- Beregn global middelverdi, μ:

$$\mu \equiv \sum_{i=0}^{G-1} ip(i)$$

- Beregn variansen mellom klassene, $\sigma_B^2(k)$:

$$\sigma_B^2(t) = \frac{\left[\mu(t) - \mu P_1(t)\right]^2}{P_1(t)(1 - P_1(t))}$$

- Finn terskelen der $\sigma_B^2(k)$ har sitt maksimum.
- Beregn separabilitetsmålet, η(t):

$$\eta(t) = \frac{\sigma_B^2(t)}{\sigma_{Tot}^2}, \quad 0 \le \eta(t) \le 1$$

”Minimum feil” terskling

- Kittler og Illingworth (1985) beregner et kriterium for alle mulige terskelverdier:

$$J(t) = 1 + 2\left[P_1(t)\ln\sigma_1(t) + P_2(t)\ln\sigma_2(t)\right] - 2\left[P_1(t)\ln P_1(t) + P_2(t)\ln P_2(t)\right]$$

- For hver t-verdi estimeres alle fem parametrene.
- Beregn J(t) for alle t og finn ***minimum***, eller finn løsning iterativt.
- Kriterie-funksjonen har lokale minima ved endene av gråtoneskalaen.
- En uheldig start-verdi i et iterativt søk kan gi meningsløs terskelverdi.
- Bruk Otsu’s terskelverdi som start-verdi i et iterativt søk.

Adaptiv terskling ved interpolasjon

- Globale terskler gir ofte dårlig resultat.
- Globale metoder kan benyttes lokalt.
- Dette virker ikke der vinduet bare inneholder en klasse !
- Oppskrift:
  - **NIVÅ I:** Del opp bildet i del-bilder.
    - For del-bilder med bi-modalt histogram:
      - Finn lokal terskelverdi $T_c(i,j)$
        og tilordne den til senterpikselet (i,j) i del-bildet.
    - For del-bilder med uni-modalt histogram:
      - Finn lokal terskelverdi ved interpolasjon.
  - **NIVÅ II:** Piksel-for-piksel interpolasjon:
    - Gå gjennom alle piksel-posisjoner
      - bestem adaptiv terskelverdi T(x,y)
        ved interpolasjon mellom de lokale terskelverdiene $T_c(i,j)$.
    - Terskle så hvert piksel (x,y) i bildet i terskelverdiene T(x,y).