INF 2310 – Digital bildebehandling

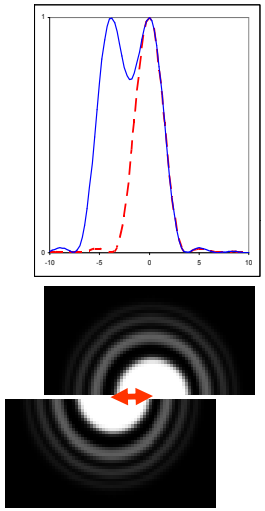
Oppsummering FA, mai 2015:

Avbildning	F1
Sampling og kvantisering	F2
Geometriske operasjoner	F3
Filtrering i bildedomenet	F6, F7
Morfologiske operasjoner	F13
Farger og fargerom	F14

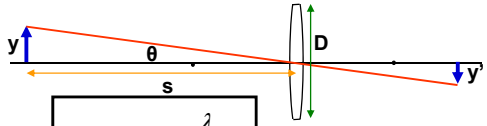
Rayleigh-kriteriet

- To punkt-kilder kan adskilles hvis de ligger slik at sentrum i det ene diffraksjonsmønstret faller sammen med den første mørke ringen i det andre.
  - Vinkelen mellom dem er da gitt ved

    $\sin\theta = 1.22\,\lambda / D$ radianer.
  - Dette er "Rayleigh-kriteriet".
  - *Vi kan ikke se detaljer som er mindre enn dette.*



Hvor små detaljer kan en linse oppløse?



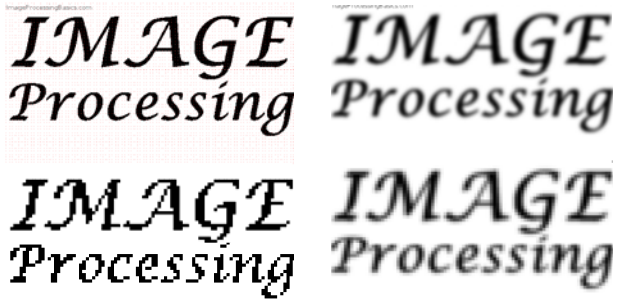
- Vinkeloppløsningen er gitt ved $\sin\theta = 1.22\frac{\lambda}{D}$
- Tangens til vinkelen θ er gitt ved $tg(\theta) = \frac{y}{s}$
- For små vinkler er sin(θ) = tg(θ) = θ, når vinkelen θ er gitt i radianer.
- **=> Den minste detaljen vi kan oppløse:** $\frac{y}{s} = 1.22\frac{\lambda}{D} \Rightarrow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

Samplingsteoremet (Shannon/Nyquist)

- Anta at det kontinuerlige bildet er båndbegrenset, dvs. det inneholder ikke høyere frekvenser enn f_{max}
- Det kontinuerlige bildet kan rekonstrueres fra det digitale bildet dersom samplingsraten $f_s = 1/T_s$ er større enn 2 f_{max} (altså $T_s < ½T_0$)
- 2 f_{max} kalles Nyquist-raten
- I praksis oversampler vi med en viss faktor for å kunne få god rekonstruksjon

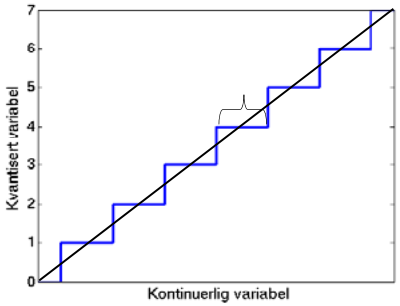
Anti-aliasing

- Ved *anti-aliasing* fjerner/demper vi de høyere frekvensene i bildet **før** vi sampler



Kvantisering

- Hvert piksel lagres vha. *n* biter
- Pikselet kan da inneholde heltallsverdier fra 0 til 2^n-1
- Eks 3 biter:



Kvantiseringsfeil

- Kvantiseringsfeil
  - Summen av hver piksels avrundingsfeil
  - Kan velge intervaller og tilhørende rekonstruksjonsintensiteter for å minimere denne => Ikke nødvendigvis uniform fordeling
- Sentrale stikkord:
  - Lagringsplass
  - Behov for presisjon/akseptabelt informasjonstap
  - Hardware-kompleksitet, eller fysiske begrensninger
- Merk: Fremvisning og videre analyse av det kvantiserte bildet kan stille ulike krav til presisjon

Geometriske operasjoner

- Endrer på pikslenes posisjoner
- Første steg i denne prosessen:
  - Transformer pikselkoordinatene (x,y) til (x',y'):

$$x' = T_x(x,y)$$

$$y' = T_y(x,y)$$

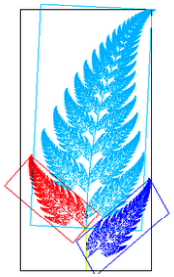
  - T_x og T_y er ofte gitt som polynomer.
- Siden pikselkoordinatene må være heltall, må vi deretter bruke interpolasjon til å finne pikselverdien (gråtonen) i den nye posisjonen.

Affine transformer

- Transformerer pikselkoordinatene (x,y) til (x',y'):

$$x' = T_x(x,y)$$

$$y' = T_y(x,y)$$



- Affine transformer beskrives ved:

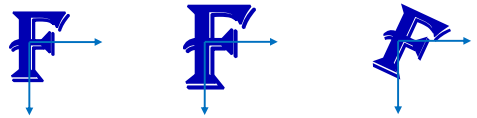
$$x' = a_0x + a_1y + a_2$$
$$y' = b_0x + b_1y + b_2$$

- På matriseform:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{eller} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ b_0 & b_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

Eksempler på enkle transformer - I

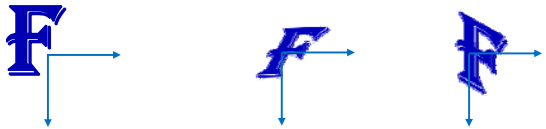
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Identitet	1	0	0	0	1	0	x' = x y' = y
Skalering	s_1	0	0	0	s_2	0	x' = s_1x y' = s_2y
Rotasjon	cosθ	-sinθ	0	sinθ	cosθ	0	x' =xcosθ-ysinθ y' =xsinθ+ycosθ



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Eksempler på enkle transformer - II

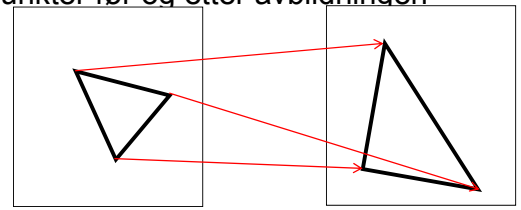
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Translasjon	1	0	Δx	0	1	Δy	x' = x+Δx y' = y+Δy
Horisontal "shear" med faktor s_1	1	s_1	0	0	1	0	x' = x+s_1y y' = y
Vertikal "shear" med faktor s_2	1	0	0	s2	1	0	x' = x y' = s_2x+y



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Alternativ måte å finne transformkoeffisientene

- En affin transform kan bestemmes ved å spesifisere tre punkter før og etter avbildningen



- Med disse tre punktparene kan vi finne de 6 koeffisientene; a_0, a_1, a_2, b_0, b_1, b_2
- Med flere enn 3 punktpar velger man den transformasjonen som minimerer (kvadrat-)feilen summert over alle punktene.

Forlengs-mapping

```
for all x',y' do g(x',y') = 0
a0 = cos θ
a1 = -sin θ
b0 = sin θ
b1 = cos θ

for all x,y do
  x' = round(a0x+a1y)
  y' = round(b0x+b1y)
  if (x',y') inside g
    g(x',y') = f(x,y)
end
```

Eksempel:
Enkel rotasjon ved transformen:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Flytter de posisjonstransformerte pikselposisjonene til nærmeste pikselposisjon i utbildet.

Skriver innbildets f(x,y) inn i g(x', y')

Baklengs-mapping

```
a0 = cos (-θ)
a1 = -sin (-θ)
b0 = sin (-θ)
b1 = cos (-θ)

for alle x',y' do
  x = round(a0x'+a1y')
  y = round(b0x'+b1y')
  if (x,y) inside f
    g(x',y') = f(x,y)
  else
    g(x',y')=0
end
```

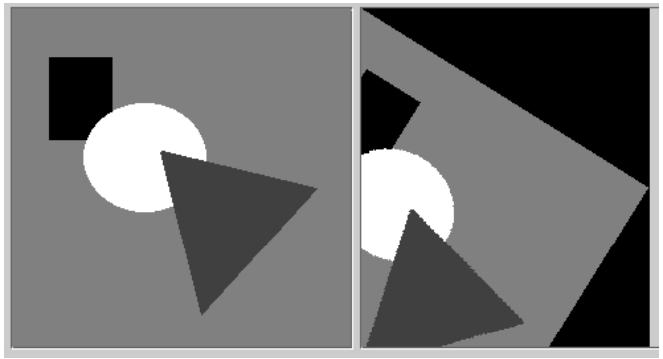
Samme eksempel som ved forlengs-mappingen.

NB: (x,y) rotert med θ ga (x',y')
(x',y') rotert med -θ gir (x,y)

Resample bildet.
Her; for hvert utbilde-piksel, invers-transformér,
og velg verdien til nærmeste piksel fra innbildet.

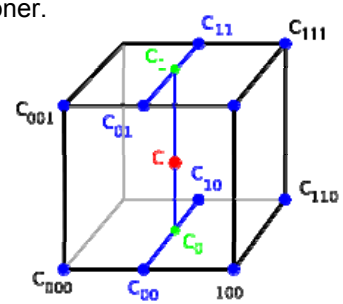
For hver pikselposisjon i ut-bildet:
Hent pikselverdi fra innbildet.

Baklengs-mapping, forts.



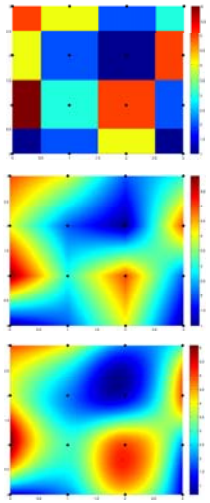
Trilineær interpolasjon

- Utvidelsen fra 2D til 3D kalles *trilineær* interpolasjon, og er en lineær interpolasjon mellom resultatene av to bilineære interpolasjoner.
- **Resultatet er uavhengig av rekkefølgen.**



Interpolasjon – en sammenligning

- Nærmeste nabo gir 2D trappefunksjon.
  Diskontinuitet midt mellom punktene.
- Bi-lineær interpolasjon bruker 4 piksler.
  Derivert blir ikke kontinuerlig
  over bilde-flaten.
- Bi-kubisk interpolasjon gir glattere flater.
  Er mer regnekrevende: 4x4=16 piksler.



Utregning av 2-D konvolusjon

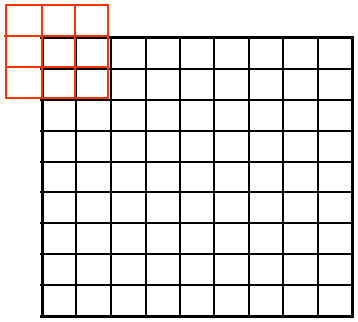
$$g(x,y) = \sum_{j=x-w_1}^{x+w_1} \sum_{k=y-w_2}^{y+w_2} h(x-j, y-k)\, f(j,k)$$

- For å regne ut resultatet av en konvolusjon for posisjon (*x,y)* :
  - Roter konvolusjonsfilteret 180 grader
  - Legg filteret over bildet slik at origo overlapper posisjon (x,y) i bildet.
  - Multipliser hver vekt i filteret med underliggende pikselverdi.
  - **Summen av produktene gir verdien for *g(x,y)* i posisjon (*x,y*).**
- For å regne ut resultatet for alle posisjoner:
  Flytt filteret piksel for piksel og gjenta operasjonene over.
- Vi bruker notasjonen g = h * f

Praktiske problemer

- Kan ut-bildet ha samme piksel-representasjon som inn-bildet?
- Trenger vi et mellom-lager?
- Hva gjør vi langs bilde-randen?

- Anta at bildet er M x N piksler
- Anta at filteret er m x n
  ($m=2m_2+1$, $n=2n_2+1$)
- Uberørt av rand-effekt:
  (M-m+1)x(N-n+1)
  3x3: (M-2)x(N-2)
  5x5: (M-4)x(N-4)



Hva gjør vi langs randen?

Alternativer:

1. Sett g(x,y)=0
2. Sett g(x,y)=f(x,y)
3. Trunker ut-bildet
4. Trunker konvolusjons-masken h
5. Utvid bildet ved ”reflected indexing”
6. ”Circular indexing”

Et lite tips om konvolusjon

- Når vi konvolverer et filter med et bilde:
  - Er vi interessert i å lage et nytt bilde med samme størrelse som input-bildet.
  - Vi bruker en av teknikkene fra forrige foil.
- Når vi konvolverer en filter-kjerne med en annen filter-kjerne:
  - Vi vil lage effektiv implementasjon av et stort filter ved å kombinere enkle, separable filtre.
  - Vi beregner resultatet for alle posisjoner der de to filter-kjernene gir overlapp.

Egenskaper ved konvolusjon

- Kommutativ

  f*g = g*f
- Assosiativ

  (f*g)*h = f*(g*h)
- Distributiv

  f*(g+h) = (f*g) + (f*h)
- Assosiativ ved skalar multiplikasjon

  a(f*g) = (af)*g = f*(ag)
- Kan utnyttes i sammensatte konvolusjoner !

Lavpass-filtre

- Slipper gjennom lave frekvenser, og demper eller fjerner høye frekvener.
  - Høye frekvenser = skarpe kanter, støy, detaljer.
- Effekt: ”blurring” eller utsmøring av bildet
- Utfordring: bevare kanter samtidig som homogene områder glattes.

Ikke-uniformt lavpass-filter

- Uniforme lavpass-filtre kan implementeres raskt.
- Ikke-uniforme filtre, for eksempel:
  - 2D Gauss-filter:

$$h(x,y) = \exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right)$$

  - Parameter σ er standard-avviket(bredden)
  - Filterstørrelse må tilpasses σ

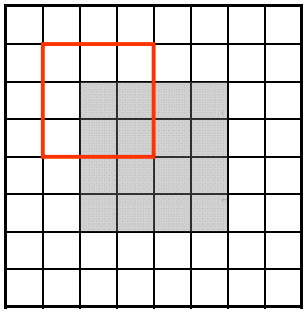
Rang-filtrering

- Vi lager en en-dimensjonal liste av alle piksel-verdiene innenfor vinduet.
- Vi sorterer listen i stigende rekkefølge.
- Responsen i (x,y) er pikselverdien i en bestemt posisjon i listen, eller en veiet sum av en bestemt del av listen.
- Dette er ikke-lineære filtre.

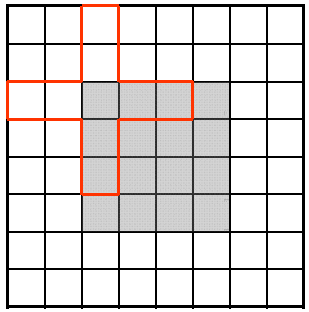
Median-filter

- g(x,y) = median av verdiene i et vindu rundt inn-pikslet.
- Median = den midterste verdien i sortert liste.
- Vindu: kvadrat, rektangel, pluss.
- Rask implementasjon kan gjøres vha. histogram, med histogram-oppdatering etter hvert som vinduet flyttes.
- Et av de mest brukte kant-bevarende støy-filtre.
- Spesielt godt til å fjerne impuls-støy (”salt og pepper”)
- Problemer:
  - Tynne kanter kan forsvinne
  - Hjørner kan rundes av
  - Objekter kan bli litt mindre
- Valg av vindus-størrelse og form er viktig!

Median og hjørner



Med kvadratisk vindu rundes hjørnet av



Med ”pluss”-vindu bevares hjørnet

Høypass-filtere

- Slipper gjennom høye frekvenser.
- Demper eller fjerner lav-frekvente variasjoner.
- Effekt:
  - Fjerner langsomt varierende bakgrunn
  - Framheve kanter, linjer og skarpe detaljer.

Høypass-filtre

- Et høypass-filter må ha positive vekter i midten, og negative vekter lenger ut.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- Vi lar summen av vektene være null
- Hvis vi lar middelverdien av ut-bildet bli null, må noen deler av ut-bildet være <0.
- Det er ingen god ide å benytte $|g(x,y)|$.
- For framvisning, skaler $g(x,y)$ og legg til en konstant slik at vi får positive pikselverdier.

Gradient-operatorer

- Asymmetrisk 1D-operator:

$$h_x(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Symmetrisk 1D-operator:

$$h_x(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

- Roberts-operatoren (også kalt Roberts kryssgradient-operator):

$$h_x(i,j) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

PS: Vi angir konvolusjonsfiltre i den hensikt at de skal brukes til konvolusjon. G&W angir filtermasker som skal brukes til korrelasjon. Filtrene vil derfor avvike med en 180 graders rotasjon.

Gradient-operatorer

- Prewitt-operatoren:

$$h_x(i,j) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

- Sobel-operatoren:

$$h_x(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

- Frei-Chen-operatoren:

$$h_x(i,j) = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \qquad h_y(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$$

PS: Vi angir konvolusjonsfiltre i den hensikt at de skal brukes til konvolusjon. G&W angir filtermasker som skal brukes til korrelasjon. Filtrene vil derfor avvike med en 180 graders rotasjon.

g_x, g_y og gradient-magnituden, G

- Vi finner de horisontale kantene:
  - Beregn $g_x(x,y)=h_x * f(x,y)$
- Vi finner de vertikale kantene:
  - Beregn $g_y(x,y)=h_y * f(x,y)$
- Beregn gradient-magnitude og retning:

$$G(x,y) = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$ **Gradient-magnitude**

$$\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$ **Gradient retning**

Laplace-operatoren

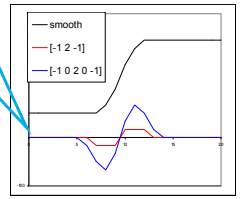
- Laplace-operatoren er gitt ved:

$$\nabla^2(f(x,y)) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Den endrer fortegn der f(x,y) har et vendepunkt.

$|\nabla^2 f|$ har to ekstremverdier per kant

$\nabla^2 f = 0$ markerer kant-posisjon.



- Kantens eksakte posisjon er nullgjennomgangen.
- Dette gir ikke brede kanter.
- Vi finner bare magnitude, ikke retning.

Flere Laplace-operatorer

- Merk at Laplace-operatorene kan uttrykkes som senter-verdi minus et (veiet) middel over et lokalt naboskap.
- 1D $\nabla^2 f(i) = -f(i-1,j) + 2f(i,j) - f(i+1-j) = 3f(i) - \sum_{j=i-1}^{i+1} f(j)$
- 2D "pluss" $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
- 2D "kvadrat" $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Laplace vs. Sobel

Sobel-filtrert
=> bred kant

Laplace-filtrert
=> dobbelt-kant

Fra Laplace til LoG

- Vi gjorde gradient-operatorene støy-robuste ved å bygge inn en lavpassfiltrering.
  Eksempel: Sobel-operator

$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

- Vi kan gjøre det samme med Laplace-operatoren
  - Vi bruker et Gauss-filter G

$$\nabla^2 * (f * G) = (\nabla^2 * G) * f = LoG * f$$

  - Der LoG er resultatet av å anvende

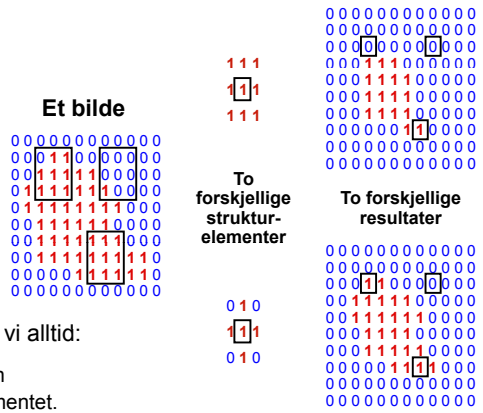
    Laplace-operatoren på en Gauss-funksjon.

Cannys algoritme

1. Lavpassfiltrer med Gauss-filter (med gitt σ).
2. Finn gradient-magnituden og gradient-retningen.
3. Tynning av gradient-magnitude ortogonalt på kant.
   - F.eks.: Hvis en piksel i gradient-magnitude-bildet har en 8-nabo i eller mot gradient-retningen med høyere verdi, så settes pikselverdien til 0.
4. Hysterese-terskling (to terskler, T_h og T_l):
   a. Merk alle piksler der $g(x,y) \geq T_h$
   b. For alle piksler der $g(x,y) \in [T_l, T_h)$:
      - Hvis (4 eller 8)-nabo til en merket piksel, så merkes denne pikselen også.
   c. Gjenta fra trinn b til konvergens.

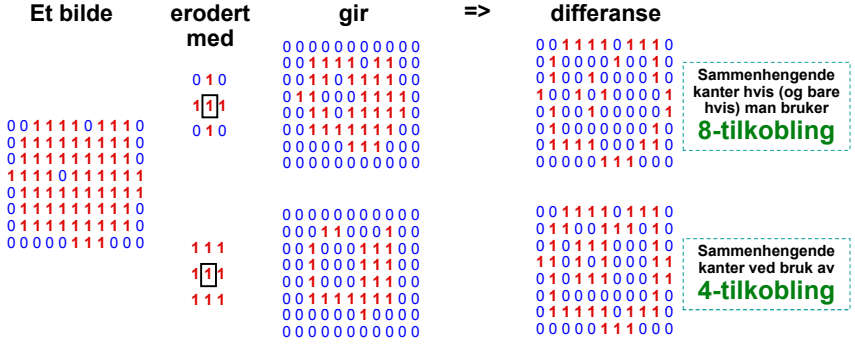
Erosjon: Passer strukturelementet til det binære bildet?

- Vi flytter strukturelementet rundt over et binært bilde.
- Strukturelementet **passer** i posisjonen (x,y) i bildet hvis alle elementer ≠ 0 i strukturelementet overlapper en pikselverdi ≠ 0 i bildet.
- I denne sammenhengen vil vi alltid:
  - Ignorere pikselverdier som overlapper 0 i strukturelementet.
    - 0 markerer jo «ikke er med i naboskapet».
  - Anta at piksler utenfor bildet er 0.



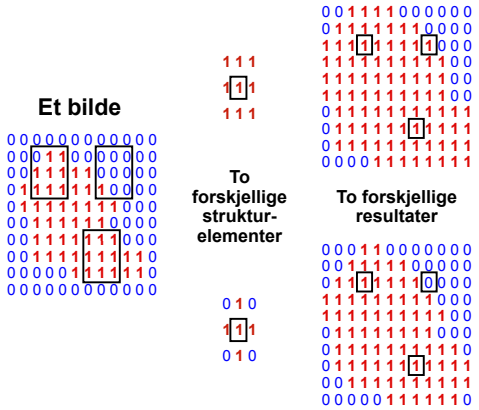
Anvendelse av erosjon: Kantdeteksjon

- Erodering fjerner piksler langs omrisset av et objekt.
- Vi kan finne kantene av objektene i bildet ved å subtrahere et erodert bilde fra originalbildet: **g = f - (f ⊖ S)**
- Det benyttede strukturelementet avgjør kantens tilkoblingstype:



Dilasjon: Treffer strukturelementet det binære bildet?

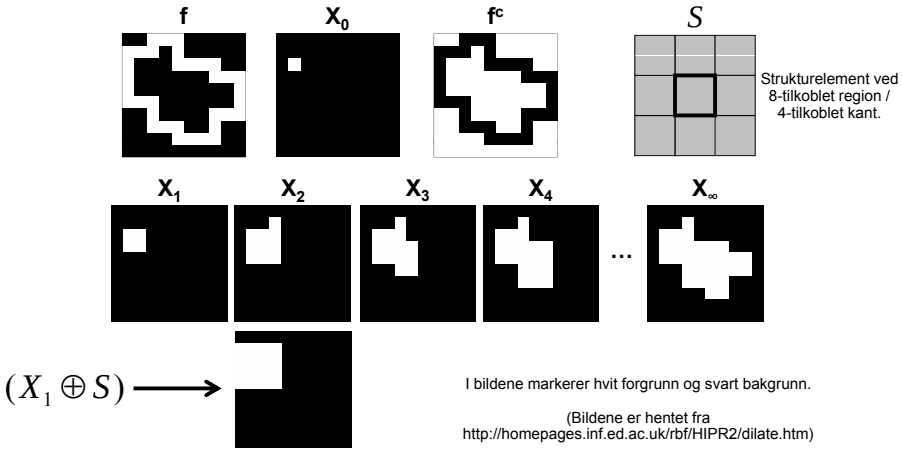
- Vi flytter strukturelementet rundt over et binært bilde.
- Strukturelementet **treffer** i posisjonen (x,y) i bildet hvis et element ≠ 0 i strukturelementet overlapper en pikselverdi ≠ 0 i bildet.
- Her reflekterer vi (roterer 180°) strukturelementet før vi flytter det rundt.



Fortsatt vil vi:
- Ignorere pikselverdier som overlapper 0 i strukturelementet.
- Anta at piksler utenfor bildet er 0.

Anvendelse av dilasjon: Region-fylling

- La X_0 inneholde et punkt i regionen som skal fylles.
- Iterativt beregn $X_k = (X_{k-1} \oplus S) \cap f^c$ inntil konvergens:



I bildene markerer hvit forgrunn og svart bakgrunn.

(Bildene er hentet fra http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm)

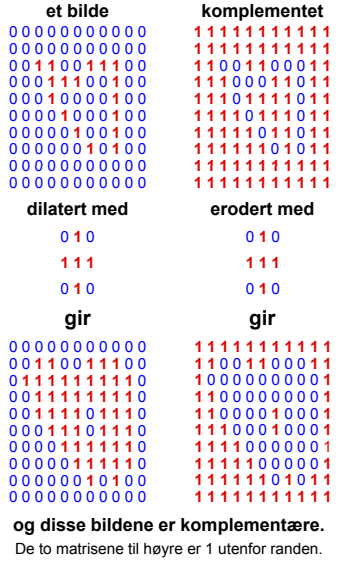
Dualitet

- **Dilasjon og erosjon er duale** med hensyn til komplementering og reflektering (180° rotasjon), dvs. at dilasjon og erosjon kan uttrykkes ved hverandre:

$$f \oplus S = \left(f^c \ \theta \ \hat{S}\right)^c$$

$$f \ \theta \ S = \left(f^c \oplus \hat{S}\right)^c$$

- For å dilatere f med symmetrisk S kan vi erodere komplementet til f med S, og ta komplementet av resultatet.
  - Tilsvarende for å erodere.
- => Dilasjon og erosjon kan utføres av samme prosedyre, forutsatt at vi kan rotere et strukturelement 180° og finne komplementet til et binært bilde.



Dilasjon: Andre egenskaper

- **Dilasjon er kommutativ.**

$$f \oplus S = S \oplus f$$

  - Selv om det er en konvensjon at første operand er bildet og andre er strukturelementet, så har dette altså ingen betydning.
- **Dilasjon er assosiativ.**

$$f \oplus (S_1 \oplus S_2) = (f \oplus S_1) \oplus S_2$$

  - Hvis S kan dekomponeres, dvs. at S er S_1 dilatert med S_2, kan vi spare en del regnetid, spesielt hvis S_1 og S_2 er én-dimensjonale. Eksempel:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Erosjon: Andre egenskaper

- Erosjon er **IKKE** kommutativ:

$$f \ \theta \ S \neq S \ \theta \ f$$

- Erosjon er heller **IKKE** assosiativ, men suksessiv erosjon av bildet f med A og så med B er ekvivalent med erosjon av bildet f med A **dilatert** med B:

$$(f \ \theta \ A) \ \theta \ B = f \ \theta \ (A \oplus B)$$

  - Passer det med denne tidligere påstanden? «Hvis s_2 er formlik s_1, men dobbelt så stort, så er $f \ominus s_2 \approx (f \ominus s_1) \ominus s_1$»

Åpning

- **Erosjon** av et bilde **fjerner** alle strukturer som ikke kan inneholde strukturelementet, og «**krymper**» alle andre strukturer.
- Hvis vi **dilaterer** resultatet av en erosjon med samme strukturelement, vil de strukturene som «**overlevde**» erosjonen bli omtrentlig **gjenskapt**.
- Dette er en **morfologisk åpning**;

$$f \circ S = (f \ \theta \ S) \oplus S$$

- Navnet kommer av at operasjonen kan skape en åpning (et mellomrom) mellom to strukturer som bare henger sammen ved en tynn «bro», uten å krympe disse to strukturene i noen betydelig grad.
  - Bare erosjon kan også skape en slik åpning/mellomrom, men vil også krympe begge strukturene.

Lukking

- **Dilasjon** av et bilde **utvider** strukturer, **fyller** i **hull og innbuktninger** i omrisset.
- Hvis vi **eroderer** resultatet av en dilasjon med samme strukturelement, vil strukturene **stort sett** få **gjenskapt** sin opprinnelige størrelse og form, men **hull og innbuktninger** som ble fylt igjen ved dilasjonen vil **ikke gjenoppstå**.
- Dette er en **morfologisk lukking**;

$$f \bullet S = (f \oplus \mathrm{S}) \theta \ S$$

- Navnet kommer av at operasjonen kan lukke en åpning mellom to strukturer som bare er adskilt med et lite gap, uten at de to strukturene vokser i noen betydelig grad.
  - Bare dilasjon kan også lukke en slik åpning, men vil også forstørre begge strukturene.

Dualitet mellom åpning og lukking

- **Lukking** er en **dual** operasjon til **åpning** med hensyn til komplementering og reflektering (180° rotering), og omvendt:

$$f \bullet S = (f^c \circ \hat{S})^c \qquad f \circ S = (f^c \bullet \hat{S})^c$$

- Lukking kan utføres ved å komplementere bildet, åpne det med det speilvendte (180° rotere) strukturelementet, og ta komplementet av resultatet.
  - Tilsvarende for åpning.
- Vi kan altså utføre begge operasjonene med kode bare for den ene, hvis vi har kode for å speilvende og komplementere et binært bilde.
- **Lukking** er en **ekstensiv** transformasjon (piksler legges til).
- **Åpning** er en **antiekstensiv** transformasjon (piksler fjernes).

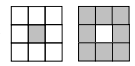
$$f \circ S \subseteq f \subseteq f \bullet S$$

«Hit-or-miss»-transformasjonen

- Tilbake til den opprinnelige situasjonen: Bilde f og strukturelement S
- Men strukturelementet S er nå definert ved et par $[S_1, S_2]$ av binære strukturelementer som ikke har noen felles elementer.
- «Hit-or-miss»-transformasjonen av f med S = $[S_1, S_2]$ er definert som:

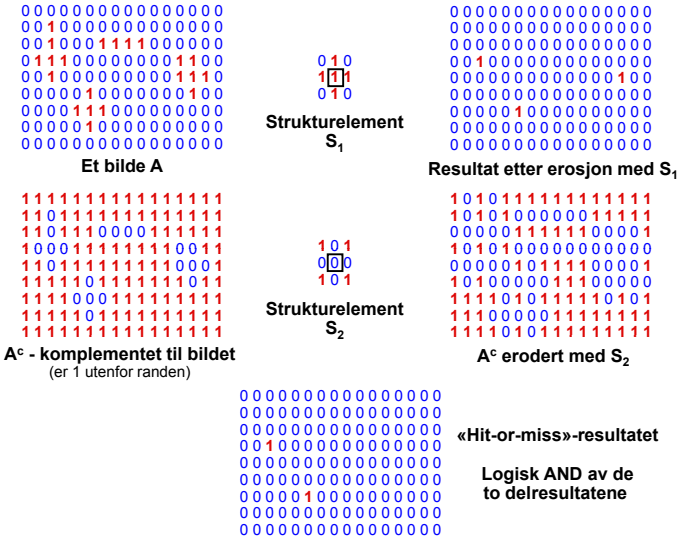
$$f(*)S = f(*)[S_1, S_2] = (f \ \theta \ S_1) \cap (f^C \ \theta \ S_2)$$

- En **forgrunnspiksel** i ut-bildet **oppnås** kun hvis:
  - **S_1 passer forgrunnen** rundt pikselen **og**
  - **S_2 passer bakgrunnen** rundt pikselen.
- Kan brukes til å finne/behandle bestemte mønstre i et bilde, f.eks. til å:
  - Finne bestemte strukturer.
  - Fjerne enkeltpiksler.



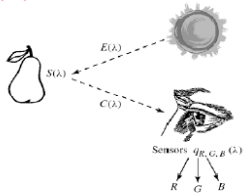
  - Benyttet i "tynning" (om to slides).

Eksempel: «Hit-or-miss»



Tre integraler gir RGB

- Lys fra en kilde med spektralfordeling E(λ)
  - treffer et objekt med spektral refleksjonsfunksjon S(λ).
  - Reflektert lys detekteres av tre typer tapper med spektral lysfølsomhetsfunksjon $q_i(\lambda)$.



- Tre analoge signaler kommer ut av dette:

$$R = \int E(\lambda)\, S(\lambda) q_R(\lambda) d\lambda$$

$$G = \int E(\lambda)\, S(\lambda)\, q_G(\lambda) d\lambda$$

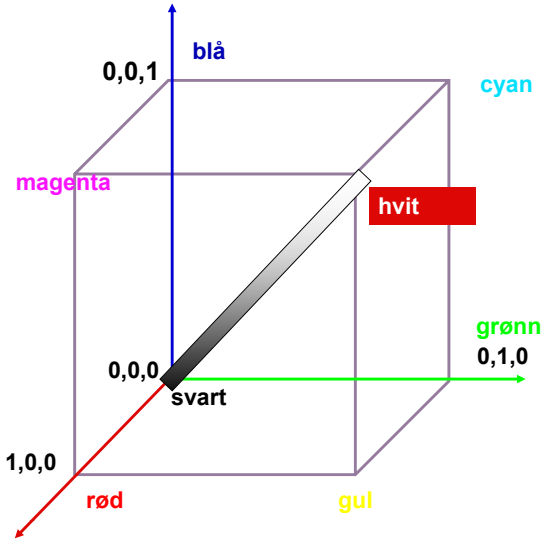
$$B = \int E(\lambda)\, S(\lambda)\, q_B(\lambda) d\lambda$$

Beskrivelse av farger

- En farge kan beskrives på forskjellige måter (fargerom)
  - RGB
  - HSI (Hue, Saturation, Intensity)
  - CMY (Cyan, Magenta, Yellow)
  - pluss mange flere …..
- HSI er viktig for hvordan vi beskriver og skiller farger.
  - I – Intensitet: hvor lys eller mørk er den
  - S – saturation/metning: hvor ”sterk” er fargen
  - H – dominerende farge (bølgelengde)
  - H og S beskriver sammen fargen og kalles kromatisitet

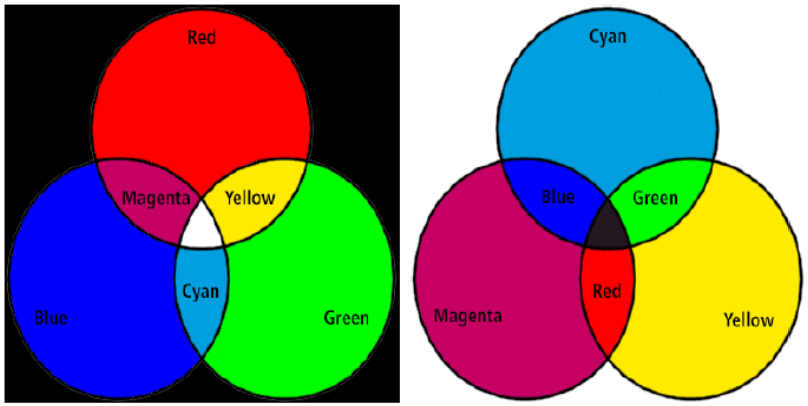
RGB-kuben

Gråtonebilder: r=g=b

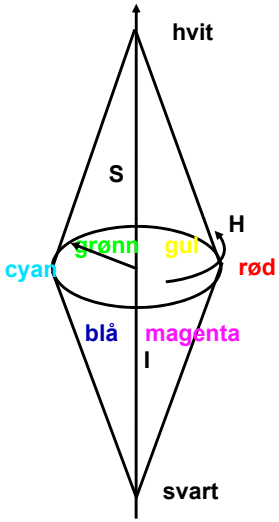


RGB og CMY

- RGB og CMY er i prinsippet sekundærfarger for hverandre.



Hue, Saturation, Intensity (HSI)



- **Hue: ren farge - gir bølgelengden i det elektromagnetiske spektrum.**

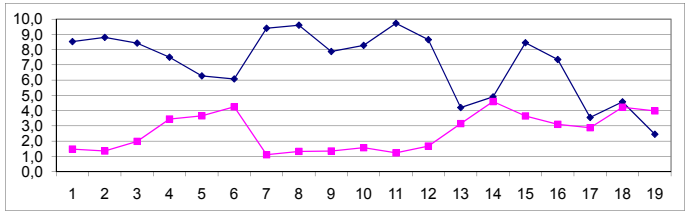
- **H er vinkel og ligger mellom 0 og 2π:**
  Rød: H=0, grønn: H= 2π/3, blå= 4π/3,
  gul: H=π/3, cyan= π, magenta= 5π/3
- **Hvis vi skalerer H-verdiene til 8-bits:**
  Rød: H=0, grønn: H= 85, blå= 170,
  gul: H=42, cyan= 127, magenta= 213.

Fargebilder og fargetabeller

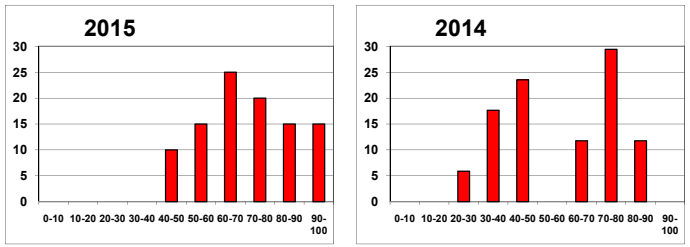
- RGB kan lagres med like mange biter for r, g, b, f.eks (8 + 8 + 8)
- Selv 3 + 3 + 3 = 9 biter gir oss 8 · 8 · 8 = 512 kombinasjoner,
  men bare 8 forskjellige nivåer av rødt, grønt og blått,
  og dermed også bare 8 forskjellige gråtoner.
- Et scene med mange nyanser av én farge vil da se ille ut !
  Hvorfor? Jo fordi denne fargen bare får 8 forskjellige nyanser !
- Det er ikke sikkert at alle de 512 fargene finnes i bildet.
- Alternativt kan man bruke 8 biter og fargetabeller.
- Hver rad i tabellen beskriver en r, g, b-farge med 24 biter.
- Tabellen inneholder de 256 fargene som best beskriver bildet.
- I bilde-filen ligger pikselverdiene som tall mellom 0 og 255.
- Når vi skal vise bildet, slår vi bare opp i samme rad som pikselverdien, og finner de tilsvarende r, g, b-verdiene.

Midtveiseksamen

- Noen oppgaver var vanskeligere enn andre, (μ,σ) for del-oppgaver:



- Dette kullet gjorde det bra, normalisert histogram over normert poengsum:



- Kontakt oss …
  - Hvis du lurer på noe i INF2310-pensum
  - Hvis du tenker på flere kurs i digital bildeanalyse
  - Hvis du tenker på å ta en Master-oppgave

Takk for følget, og lykke til med eksamen !!!