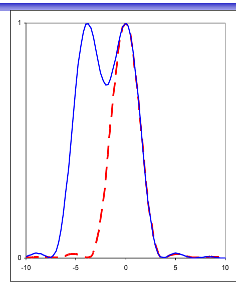
INF 2310 – Digital bildebehandling

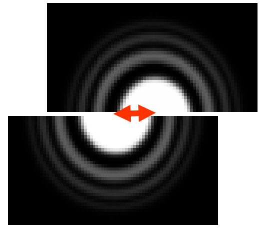
Oppsummering II, mai 2012:

Avbildning	F1
Sampling og kvantisering	F2
Geometriske operasjoner	F3
Gråtone- og histogramoperasjoner	F4,5
Filtrering i bilde-doménet	F6,7
(Filtrering i frekvens-doménet)	F8,9
(Kompresjon og koding av bilder)	F11,12
Segmentering ved terskling	F13
(Morfologiske operatorer)	F14
Farger og fargerom	F15

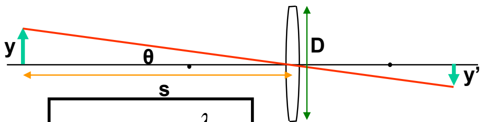
Rayleigh-kriteriet



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

$$\boxed{\frac{y}{s} = 1.22\frac{\lambda}{D}} \Rightarrow \boxed{y = 1.22\frac{s\lambda}{D}}$$

Samplingsteoremet (Shannon/Nyquist)

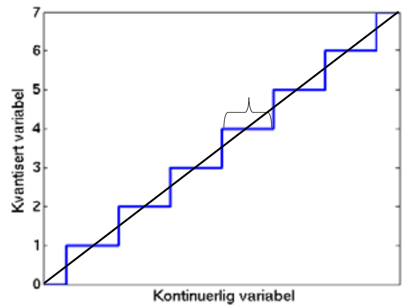
- Anta at det kontinuerlige bildet er båndbegrenset, dvs. det inneholder ikke høyere frekvenser enn f_{max}
- Det kontinuerlige bildet kan rekonstrueres fra det digitale bildet dersom samplingsraten $f_s=1/T_s$ er større enn 2 f_{max} (altså T_s < ½T_0)
- 2 f_{max} kalles Nyquist-raten
- I praksis oversampler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

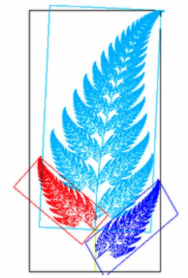
- Siden pikselkoordinatene må være heltall, må vi deretter bruke interpolasjon til å finne pikselverdien (gråtonen) i den nye posisjonen.

Affine transformer

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

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

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



- Affine transformer beskrives ved:

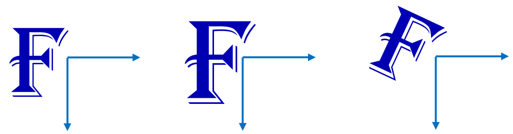
$$x' = a_0x + a_1y + a_2$$
$$y' = b_0x + b_1y + b_2$$

- På matriseform:

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

Eksempler på enkle transformer - I

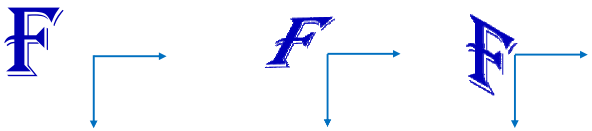
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Identitet	1	0	0	1	0	0	$x' = x$ $y' = y$
Skalering	s_1	0	0	0	s_2	0	$x' = s_1x$ $y' = s_2y$
Rotasjon	cosθ	-sinθ	0	sinθ	cosθ	0	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$



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

Eksempler på enkle transformer - II

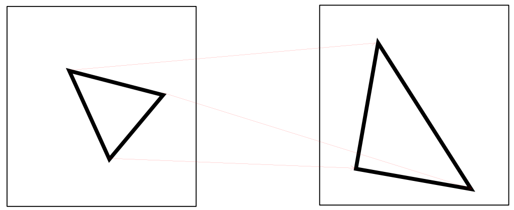
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Translasjon	1	0	Δx	0	1	Δy	$x' = x+\Delta x$ $y' = y+\Delta y$
Horisontal "shear" med faktor s_1	1	s_1	0	0	1	0	$x' = x+s_1y$ $y' = y$
Vertikal "shear" med faktor s_2	1	0	0	s2	1	0	$x' = x$ $y' = s_2x+y$



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

Alternativ måte å finne transformkoeffisientene

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



inn-bildet **resultat-bildet**

- 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

a_0 = cos θ
a_1 = -sin θ
b_0 = sin θ
b_1 = cos θ

for all x,y do
  x' = round(a_0x+a_1y)
  y' = round(b_0x+b_1y)
  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

a_0 = cos (-θ)
a_1 = -sin (-θ)
b_0 = sin (-θ)
b_1 = cos (-θ)

for alle x',y' do
  x = round(a_0x'+a_1y')
  y = round(b_0x'+b_1y')
  if (x,y) inside f
    g(x',y') = f(x,y)
  else
    g(x',y')=0
end

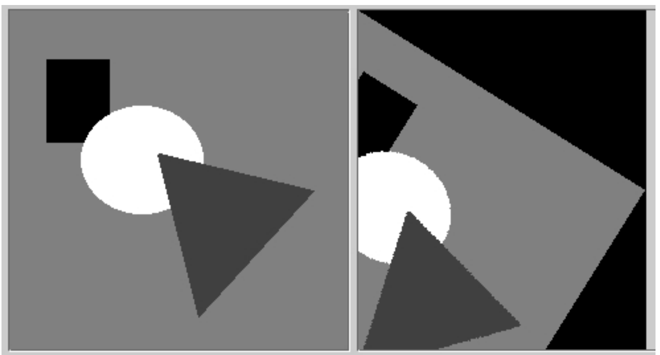
Samme eksempel som ved forlengs-mappingen.

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 nærmeste piksel fra innbildet.

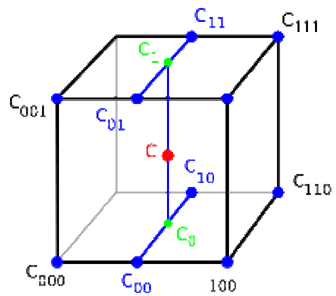
For hver pikselposisjon i ut-bildet: Hent pikselverdi fra innbildet.

Baklengs-mapping, forts.



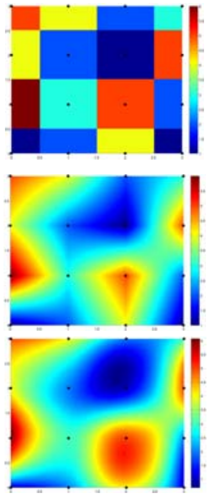
Trilineær interpolasjon

- 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 2x2=4 piksler.
  - Derivert er ikke kontinuerlig over bilde-flaten.
- Bi-kubisk interpolasjon gir glattere flater.
  - Er mer regnekrevende.
  - Bruker 4x4=16 piksler.



Normalisert histogram

- Vi har at $\sum_{i=0}^{G-1} h(i) = n \times m$
- Det normaliserte histogrammet:

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

- *p(i)* kan ses på som en sannsynlighetsfordeling for pikselintensitetene
  - "Uavhengig" av antall piksler i bildet

Kumulativt histogram

- Hvor mange piksler har gråtone mindre enn eller lik gråtone j?

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

- Normalisert kumulativt histogram:

$$\frac{c(j)}{n \times m}$$

(Sannsynligheten for at en tilfeldig valgt piksel er mindre eller lik gråtone j)

Lineær avbilding

- Lineær strekking

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

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

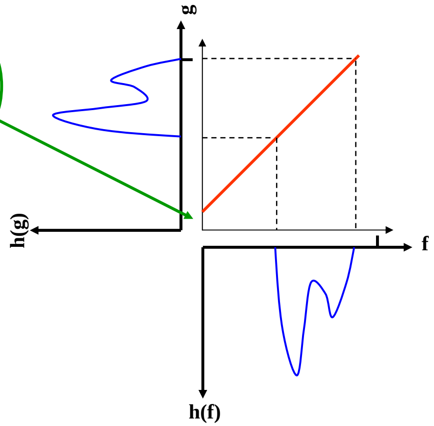
- *a* regulerer kontrasten, og *b* "lysheten"
- *a>1*: mer kontrast
- *a<1*: mindre kontrast
  - Q: Når og hvordan påvirker *a* middelverdien?
- *b*: flytter alle gråtoner *b* nivåer
- Negativer: *a=-1* , *b* =maxverdi for bildetype

Endre "lysheten" (brightness)

- Legge til en konstant ***b*** til alle pikselverdiene

$$g(x,y) = f(x,y) + b$$

  - Hvis ***b > 0***, alle pikselverdiene øker, og bildet blir lysere
  - Hvis ***b < 0***, bildet blir mørkere
  - Histogrammet flyttes opp eller ned med ***b***
  - ***Middelverdien endres!***



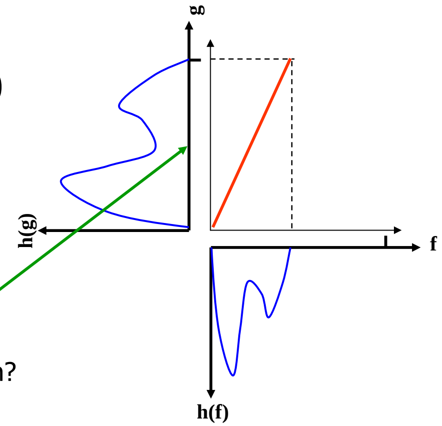
Endre kontrasten

- Multiplisere hver pikselverdi med en faktor a:

$$g(x,y) = a\, f(x,y)$$

  - Hvis ***a > 1***, kontrasten øker
  - Hvis ***a < 1***, kontrasten minker

- Eks: Bruke hele intensitetsskalaen
- **Q:** Hva skjer med middelverdien?



Justering av μ og σ²

- Gitt inn-bilde med middelverdi μ og varians σ^2
- Anta en lineær gråtone-transform T[i]=ai+b
- Ny middelverdi μ_T og varians σ_T^2 er da gitt ved

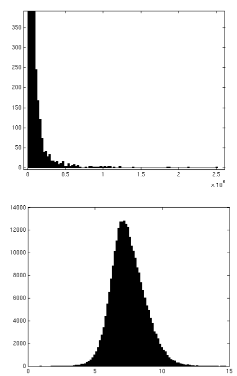
$$\mu_T = \sum_{i=0}^{G-1} T[i]\, p(i) = a\mu + b$$

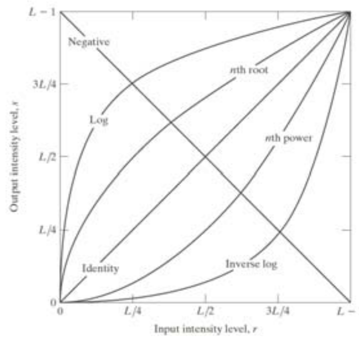
$$\begin{aligned}\sigma_T^2 &= \sum_{i=0}^{G-1} T[i]^2 p(i) - \left(\sum_{i=0}^{G-1} T[i]\, p(i)\right)^2 \\ &= \sum_{i=0}^{G-1}\left(a^2 i^2 + 2aib + b^2\right) p(i) - \left(\sum_{i=0}^{G-1}(ai+b)\, p(i)\right)^2 \\ &= a^2\left(\sum_{i=0}^{G-1} i^2 p(i) - \left(\sum_{i=0}^{G-1} i\, p(i)\right)^2\right) = a^2\sigma^2\end{aligned}$$

- Dvs.
  $a = \sigma_T/\sigma$, $b = \mu_T - a\mu$
- Vi kan altså
  - velge nye μ_T og σ_T^2,
  - beregne a og b,
  - anvende T[i]=ai + b på inn-bildet
  - og få et ut-bilde med riktig μ_T og σ_T^2

Logaritmiske transformasjoner

- Q: 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:

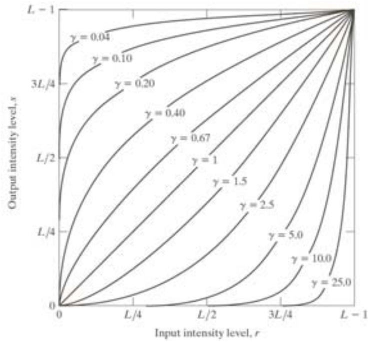
$$s = ci^{\gamma}$$

  der *s* er ut-intensiteten ved en input *i*

- Kan korrigeres ved gråtonetransformen T[i] = $i^{1/\gamma}$

- Generell kontrast-manipulasjon
  - Brukervennlig med kun én variabel

(Fig 3.6 i DIP)



Histogramutjevning (*histogram equalization*)

- Mål: Maksimere kontrasten

–Gjøre histogrammet uniformt (flatt)

↔ Kumulative histogrammet en rett linje

- Middel: Global gråtonetransform; T[i]

–Altså flytte på (hele) histogramsøyler

- Tilnærming ved å spre søylene mest mulig utover det støttede intensitetsintervallet

Algoritme for histogramutjevning

- For et n×m bilde med G gråtoner:

–Lag array h, 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 h[i]=h[i]+1

–Deretter skalér, p[i] = h[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 transformarray T

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

- Gå igjennom bildet piksel for piksel,
Hvis bildet har intensitet i, sett intensitet i utbildet til
s=T[i]

Histogramtilpasning

- Histogramutjevning gir flatt histogram
- Kan 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 inverstransformen T_g^{-1}
4. Inverstransformer det histogramutjevnede bildet fra punkt 1 ved $z=T_g^{-1}(s)$

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 masken 180 grader, legg den over bildet slik at minst en posisjon overlapper med bildet.
  - Multipliser hvert element i masken 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 masken piksel for piksel og gjenta operasjonene over.
- Vi bruker notasjonen

  g = h * f

  der * er konvolusjons-operatoren

Praktiske problemer

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

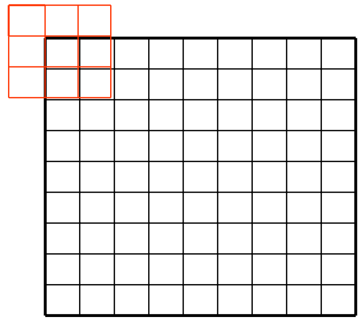
- Anta at bildet er M x N piksler
- Anta at filteret er m x n ($m=2m_2+1$, $n=2n_2+1$)

- Uberørt av rand-effekt:

  $(M-2m_2)x(N-2n_2)$

  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 med en kombinasjon av 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.

Separable filtre

- Geometrisk form: kvadrat, rektangel
- Rektangulære middelverdi-filtere er separable.

$$h(i,j) = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Fordel: et raskt filter.
- Vanlig konvolusjon: n^2 multiplikasjoner og addisjoner.
- 2 stk 1-D konvolusjoner: *2n* multiplikasjoner og addisjoner.

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

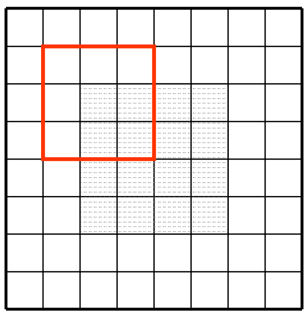
Rang-filtrering

- Vi lager en en-dimensjonal liste av alle piksel-verdiene innenfor vinduet.
- Vi sorterer listen i stigende rekkefølge.
- Vi velger en piksel-verdi fra en bestemt posisjon i den sorterte listen
- Denne piksel-verdien er resultatet av filtreringen, og skrives ut til tilsvarende piksel-posisjon i ut-bildet.

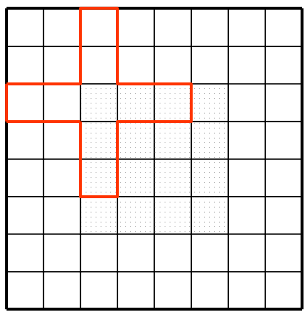
Median-filter

- 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.
  - Hvorfor er dette lurt?
- 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.

Noen gradient-operatorer - I

- "Pixel difference"

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

- "Separated pixel difference"

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

- Roberts-operatoren

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

Noen gradient-operatorer - II

- Prewitt-operatoren

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

- Sobel-operatoren

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

- Frei-Chen-operatoren

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

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

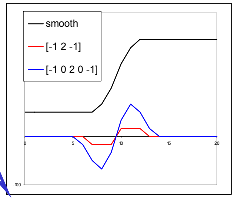
$$\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$ Gradient retning

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 infleksjons-punkt / vendepunkt.
  - $|\nabla^2 f|$ har to ekstremverdier idet vi passerer en kant
  - $\nabla^2 f = 0$ markerer kant-posisjon.



- Kantens eksakte posisjon finnes ved 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

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

- Vi kan gjøre det samme med Laplace-operatoren
  - Vi bruker et Gauss-filter G
  - Og siden konvolusjon er kommutativ, får vi

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

  - Der LoG er resultatet av å anvende Laplace-operatoren på en Gauss-funksjon.

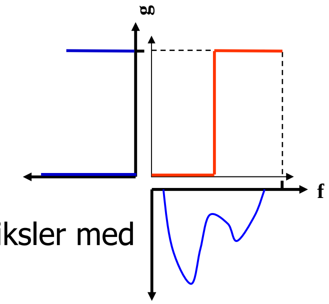
Canny's algoritme

- I Canny's algoritme gjøres kant-lokalisering slik:
  - Tynning (non-maximal suppression): Hvis et piksel har en nabo med høyere pikselverdi, settes pikselverdien ned.
  - Hysterese-terskling (to terskler T_h og T_l)
    - Vanskelig å finne en god gradient-terskel for hele bildet.
    1. Merk alle piksler der $G > T_h$
    2. Scan alle piksler der $G \in [T_l, T_h]$
    3. Hvis et slikt piksel er nabo til et merket piksel, så merkes dette pikselet også.
    4. Gjenta fra trinn 2 til konvergens.

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:

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



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

Klassifikasjonsfeil ved terskling

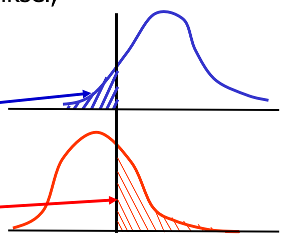
- Anta at histogrammet er en sum av to fordelinger *b(z)* og *f(z)*, *b* og *f* er ***normaliserte*** bakgrunns- og forgrunns-histogrammer.
- La *F* og *B* være ***a priori sannsynlighet*** for bakgrunn og forgrunn (*B+F=1*)
- Det normaliserte histogrammet til bildet kan da skrives

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

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

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

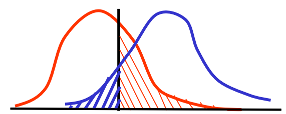
$$E_F(t) = \int_{t}^{\infty} b(z)dz$$



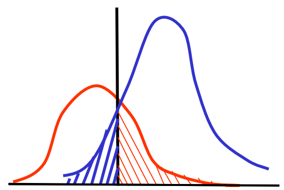
Den totale feilen

- Vi har funnet andelen feilklassifikasjon i hver fordeling.
- Den totale feilen finner vi ved å multiplisere med a priori sannsynlighetene for forgrunn og bakgrunn:

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



- Legges terskelen veldig høyt eller veldig lavt, blir feilen stor.
- Det er rimelig å anta at feilen har et minimum for en bestemt verdi *t = T*.



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:

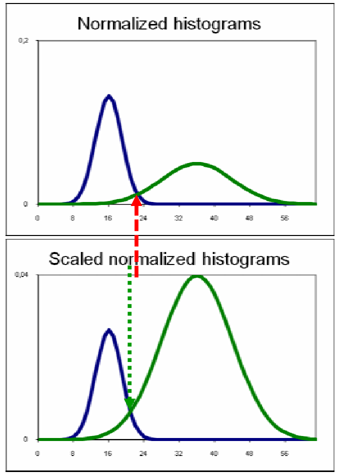
VIKTIG !!!

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

- Merk at dette er en generell løsning som gir minst feil.
- Det er ingen restriksjoner mht. fordelingene *b* og *f* !!

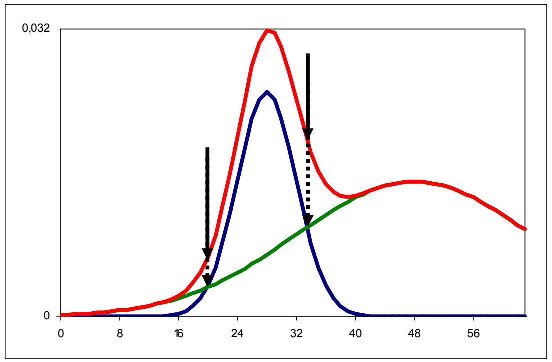
Hvilket histogram ?

- Det er IKKE skjæringen mellom de normaliserte histogrammene vi er ute etter!
- Det er skjæringen mellom **de a priori-skalerte** normaliserte histogrammene som gir riktig terskelverdi !!!



Forskjellige standardavvik ?

- Hvis standardavvikene i de to Gauss-fordelingene er forskjellige
  - og skjæringspunktene mellom fordelingene (skalert med a priori sannsynlighet) ligger innenfor gråtoneskalaen i bildet
- En terskelverdi for hvert skjæringspunkt.
- Det er bare mellom de to tersklene at flertallet av pikslene er bakgrunnspiksler!



Hvor ligger optimal terskel?

- Vi har en annengradsligning i T:

$$\left(\sigma_B^2 - \sigma_F^2\right)T^2 + 2\left(\mu_B\sigma_F^2 - \mu_F\sigma_B^2\right)T + \sigma_B^2\mu_F^2 - \sigma_F^2\mu_B^2 + 2\sigma_B^2\mu_F^2 \ln\left(\frac{B\sigma_F}{F\sigma_B}\right) = 0$$

- Hvis standard-avvikene i de to fordelingene er like ($\sigma_B = \sigma_F = \sigma$) får vi en enklere ligning:

$$2(\mu_B - \mu_F)T - (\mu_B + \mu_F)(\mu_B - \mu_F) + 2\sigma^2 \ln\left(\frac{B}{F}\right) = 0$$

$$\Updownarrow$$

$$T = \frac{(\mu_B + \mu_F)}{2} + \frac{\sigma^2}{(\mu_B - \mu_F)} \ln\left(\frac{F}{B}\right)$$

- Hvis *a priori* sannsynlighetene F og B er omtrent like (eller hvis σ=0) har vi en veldig enkel løsning:

$$\boxed{T = \frac{(\mu_B + \mu_F)}{2}}$$

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

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

- Gjenta de to punktene ovenfor til terskelen ikke flytter seg mer.
- Dette kalles Ridler og Calvard's metode
  - Hvilke betingelser må være oppfylt for at metoden skal virke?

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

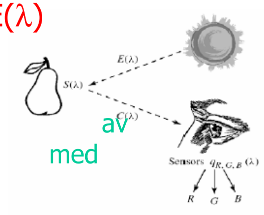
- Gitt et NxM pikslers bilde med G gråtoner.
- Finn bildets histogram, h(k), k= 0,1,2,..,G-1.
- Finn bildets normaliserte histogram: $p(k) = \frac{h(k)}{MN}, \quad k = 0,1,2,\ldots,G-1$
- Beregn kumulativt normalisert histogram: $P_1(k) = \sum_{i=0}^{k} p(i), \quad k = 0,1,2,\ldots,G-1$
- Beregn kumulativ middelverdi, μ(k): $\mu(k) \equiv \sum_{i=0}^{k} ip(i), \quad k = 0,1,2,\ldots,G-1$
- Beregn global middelverdi, μ: $\mu \equiv \sum_{i=0}^{G-1} ip(i)$
- Beregn variansen mellom klassene, $\sigma_B^2(k)$: $\sigma_B^2(k) = \frac{[\mu(k) - \mu P_1(k)]^2}{P_1(k)(1 - P_1(k))}$
- Finn terskelen, T, der $\sigma_B^2(k)$ har sitt maksimum.
- Beregn separabilitetsmålet, η(T): $\eta(T) = \frac{\sigma_B^2(T)}{\sigma_{Tot}^2}, \quad 0 \le \eta(T) \le 1$

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).

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:

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

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

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

RGB primærfarger

- Commision Internationale de l'Eclairage, (CIE) (The International Commision of Illumination)

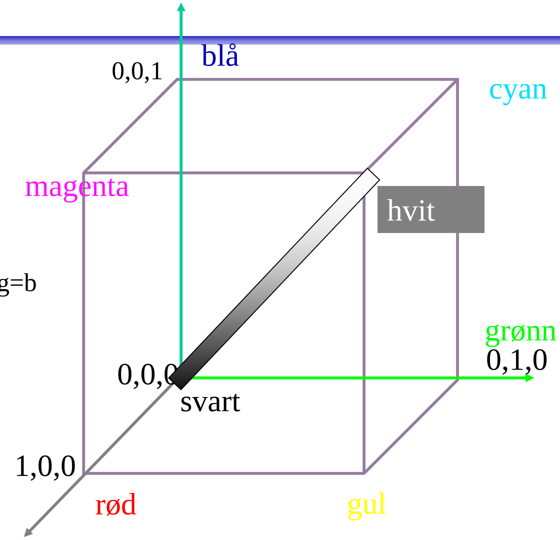
  har definert primærfargene:
  - Blå: 435.8 nm
  - Grønn: 546.1 nm
  - Rød: 700 nm

Beskrivelse av farger

- En farge kan beskrives på forskjellige måter (kalles 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

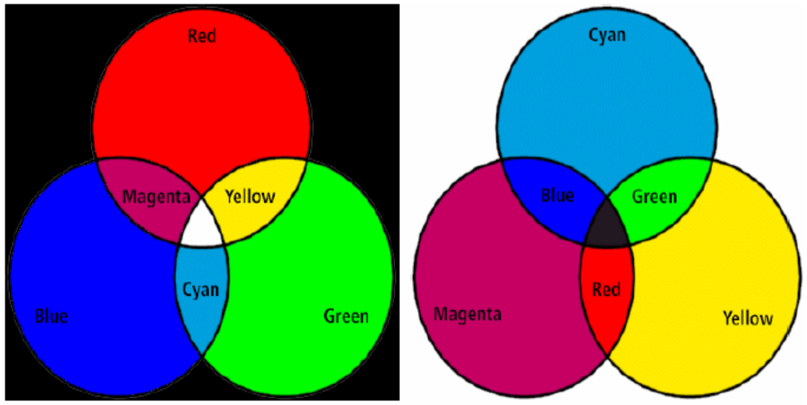
RGB-kuben

Gråtonebilder: r=g=b

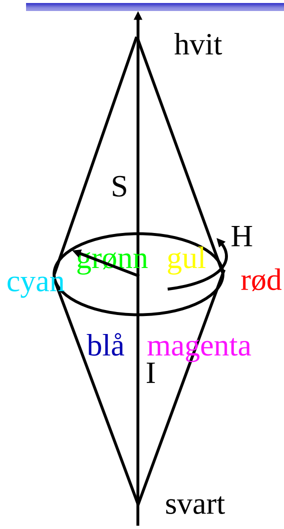


RGB og CMY

- RGB og CMY er i prinsippet sekundærfarger



Hue, Saturation, Intensity (HSI)



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

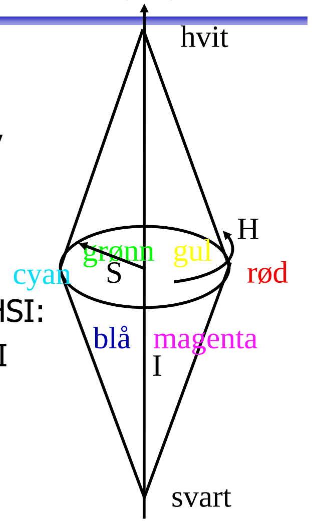
- H er vinkel og ligger mellom 0 og 2π: Rød: H=0, grønn: H= $2\pi/3$, blå= $4\pi/3$, gul: H=$\pi/3$, cyan= π, magenta= $5\pi/3$
- Hvis vi skalerer H-verdiene til 8-bits verdier vil Rød: H=0, grønn: H= 85, blå= 170, gul: H=42, cyan= 127, magenta= 213.

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.

Histogramutjevning av RGB-bilder

- Histogramutjevning på hver komponent (R,G,B) uavhengig av hverandre
  - Ofte dårlig resultat
- Et bedre alternativ er å benytte HSI:
- Transformér bildet fra RGB til HSI
- Gjør histogramutjevning på I-komponenten
- Transformer HSI_{ny} tilbake til RGB



Terskling av fargebilder - I

- Anta at vi har observert samme scene på flere bølgelengder.
- Vi kan da utføre terskling basert på
  - to-dimensjonale
  - tre-dimensjonale
  - eller multi-dimensjonale histogrammer
- Enkel metode:

  1: Bestem terskler uavhengig for hver kanal.

  2: Kombiner alle segmenterte kanaler til ett bilde.
- Dette svarer til at vi har delt opp f.eks. RGB-rommet i bokser.

Terskling av fargebilder - II

- En mer kompleks metode:
- Velg et punkt i det multidimensjonale rommet som referanse, f.eks. (R_0,G_0,B_0)
- Terskle basert på avstand fra dette referansepunktet.

$$d(x,y)=\sqrt{[f_R(x,y)-R_0]^2+[f_G(x,y)-G_0]^2+[f_B(x,y)-B_0]^2}$$

- Slik at

$$g(x,y)=\begin{cases}1\,\text{hvis } d(x,y)\le d_{max}\\0\,\text{hvis } d(x,y)>d_{max}\end{cases}$$

- Dette definerer en kule med radius d_{max} omkring punktet (R_0,G_0,B_0) .
- Kan lett generaliseres til ellipsoide med forskjellige avstands-terskler i R,G,B

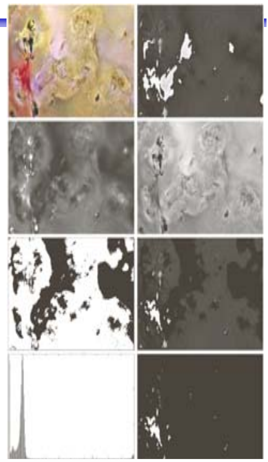
$$d(x,y)=\sqrt{\frac{[f_R(x,y)-R_0]^2}{d_R^2}+\frac{[f_G(x,y)-G_0]^2}{d_G^2}+\frac{[f_B(x,y)-B_0]^2}{d_B^2}}$$

- Merk at da er

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

Terskling i HSI

- Transformer fra RGB til HSI.
- Anta at vi vil segmentere ut de delene av bildet som
  - Har en gitt farge (H)
  - Er over en gitt metnings-terskel (S)
- Lag en maske ved å terskle S-bildet (velg en percentil)
- Multipliser H-bildet med masken.
- Velg et intervall i H som svarer til ønsket farge.
- Husk at H er sirkulær!



a b
c d
e f
g h

FIGURE 6.42 Image segmentation in HSI space. (a) Original. (b) Hue. (c) Saturation. (d) Intensity. (e) Binary saturation mask (black = 0). (f) Product of (b) and (e). (g) Histogram of (f). (h) Segmentation of red components in (a).

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

Takk, og lykke til med eksamen !!!