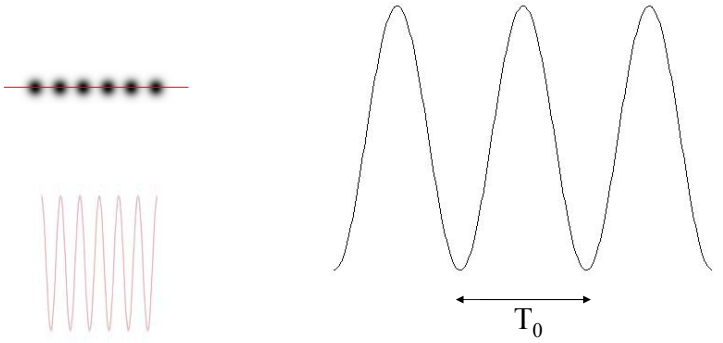
INF 2310 – Digital bildebehandling

En kort midtveis-repetisjon

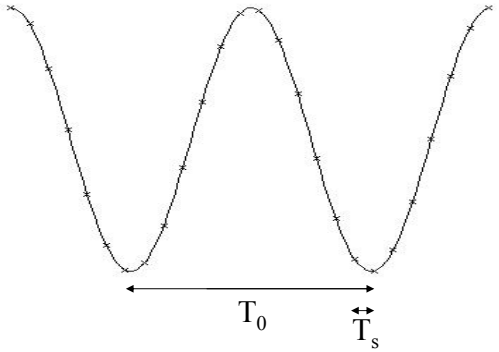
Fritz Albregtsen

Romlig frekvens



- Periode T_0 (f.eks. i mm eller µm)
- Frekvens $f_0 = 1/T_0$

Sampling av kontinuerlige signaler



- Samplingsperiode T_s
- Samplingsfrekvens $f_s = 1/T_s$ (også kalt samplingsrate)
- Hvor ofte må man sample for å kunne rekonstruere signalet?

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.

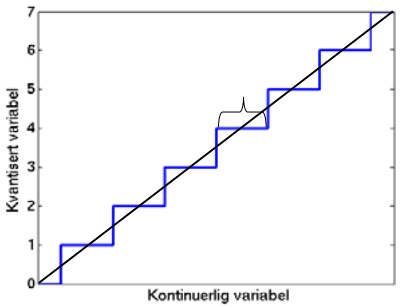
Undersampling/aliasing

- Undersampling (sample med lavere samplingsrate enn Nyquist-kriteriet) medfører **aliasing.**
- Ved undersampling forvrenges frekvensinnholdet og det digitale bildet inneholder ikke de samme frekvenser som det kontinuerlige bildet.
- Sampling av en sinusoid med for lav samplingsrate gir en diskret sinusoid med lavere frekvens.
- Aliasfrekvens er gitt ved

  $f_a = f_s - f$ når $f < f_s < 2f$

Kvantisering

- Hvert piksel lagres med *n* biter.
- Pikselet kan da inneholde heltallsverdier fra 0 til 2^n-1
- Kvantiseringsfeil
  - Σ (over bildet) av hvert piksels avrundingsfeil



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.

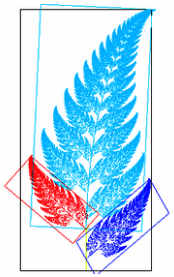
Affine transformer

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

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

- Affine transformer beskrives ved:

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

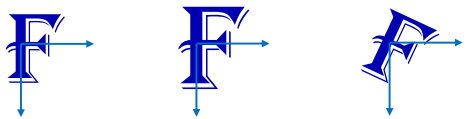


- På matriseform:

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

Eksempler på enkle transformer - I

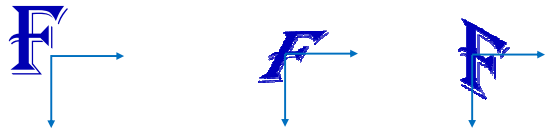
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Identitet	1	0	0	1	0	0	$x' = x$ $y' = y$
Skalering	s_1	0	0	0	s_2	0	$x' = s_1x$ $y' = s_2y$
Rotasjon	cosθ	-sinθ	0	sinθ	cosθ	0	$x' = x\cos\theta - y\sin\theta$ $y' = x\sin\theta + y\cos\theta$



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

Eksempler på enkle transformer - II

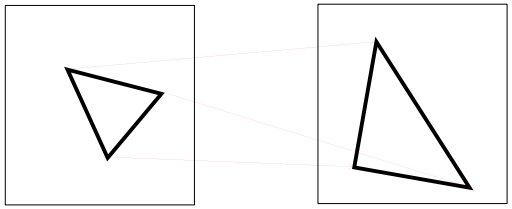
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Translasjon	1	0	Δx	0	1	Δy	$x' = x+\Delta x$ $y' = y+\Delta y$
Horisontal "shear" med faktor s_1	1	s_1	0	0	1	0	$x' = x+s_1y$ $y' = y$
Vertikal "shear" med faktor s_2	1	0	0	s2	1	0	$x' = x$ $y' = s_2x+y$



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

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 (mer om dette senere)

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
```

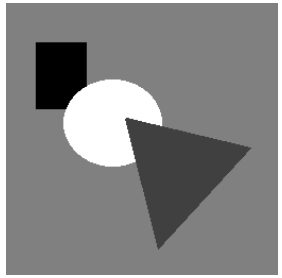
Eksempel:
Enkel rotasjon ved transformen:

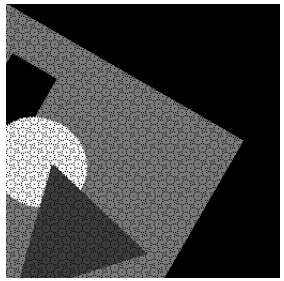
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Flytter de posisjonstransformerte pikselposisjonene til nærmeste pikselposisjon i utbildet.

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

Forlengs-mapping, forts.





Problemer:

. Ikke alle utpiksler får verdi (hullene i bildet)

• Unødig beregning av pikselkoordinater som allikevel ikke blir synlige (ender utenfor utbildet)

• Samme utbilde-piksel kan bli satt flere ganger

Baklengs-mapping

$a_0 = \cos(-\theta)$
$a_1 = -\sin(-\theta)$
$b_0 = \sin(-\theta)$
$b_1 = \cos(-\theta)$

```
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
```

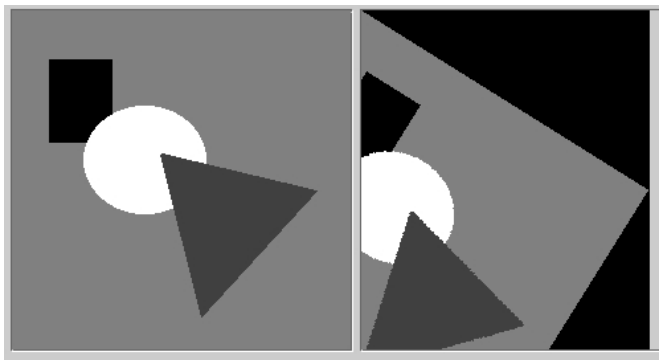
Samme eksempel som ved forlengs-mappingen.

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 nærmeste piksel fra innbildet.

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

Baklengs-mapping, forts.



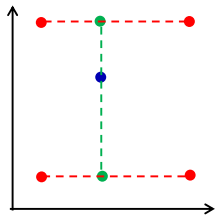
Bilineær interpolasjon - I

- Anta at vi kjenner gråtoneverdien i fire nabo-punkter
- Ønsker å estimere gråtonen i et mellomliggende punkt.
- Gjør to lineære interpolasjoner i én retning først, f.eks i x-retning:

$$f(x,y_1) \approx \frac{(x_2 - x)}{(x_2 - x_1)} f(x_1,y_1) + \frac{(x - x_1)}{(x_2 - x_1)} f(x_2,y_1)$$

og

$$f(x,y_2) \approx \frac{(x_2 - x)}{(x_2 - x_1)} f(x_1,y_2) + \frac{(x - x_1)}{(x_2 - x_1)} f(x_2,y_2)$$



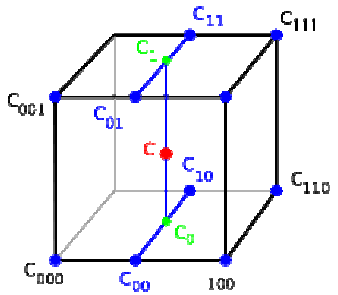
- Så én interpolasjon i ortogonal retning:

$$f(x,y) \approx \frac{(y_2 - y)}{(y_2 - y_1)} f(x,y_1) + \frac{(y - x_1)}{(y_2 - y_1)} f(x,y_2)$$

- Resultatet er uavhengig av rekkefølgen.
- Den interpolerte flaten er kvadratisk (krum), men lineær langs linjer parallelle med aksene.

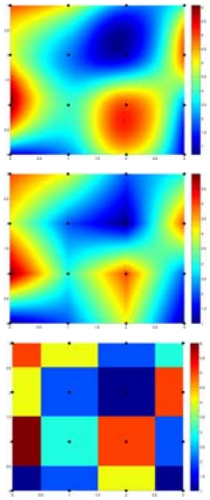
Trilineær interpolasjon

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



Interpolasjon – en sammenligning

- Bi-kubisk interpolasjon gir glattere flater enn bilineær, men er mer regnekrevende; bruker 4x4=16 piksler.
- Bi-lineær interpolasjon bruker 2x2=4 piksler. Derivert er ikke kontinuerlig over flaten.
- Nærmeste nabo gir 2D trappefunksjon, med diskontinuitet midt mellom punktene.



Normalisert histogram

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

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

- p(i) kan ses på som en sannsynlighetsfordeling for pikselverdiene i
  - "Uavhengig" av antall piksler i bildet
- Man kan si en del om bildet ut fra denne sannsynlighets-tetthetsfunksjonen

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 piksel er mindre eller lik gråtone j)

Lineær avbilding

- Lineær strekking

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

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

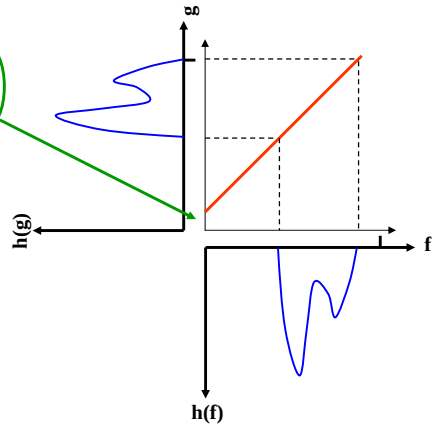
- *a* regulerer kontrasten, og *b* "lysheten"
- *a>1*: mer kontrast
- *a<1*: mindre kontrast
  - Q: Når og hvordan påvirker *a* middelverdien?
- *b*: flytter alle gråtoner *b* nivåer
- Negativer: *a=-1* , *b* =maxverdi for bildetype

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!***



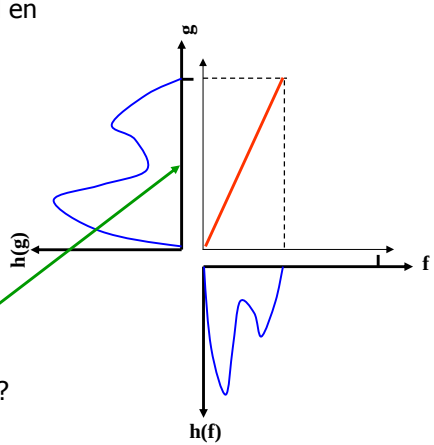
Endre kontrasten

- 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 σ²
- Anta en lineær gråtone-transform T[i]=ai+b
- Ny middelverdi μ_T og varians σ_T^2 er da gitt ved

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

$$\sigma_T^2 = \sum_{i=0}^{G-1} T[i]^2\,p(i) - \left(\sum_{i=0}^{G-1} T[i]\,p(i)\right)^2$$

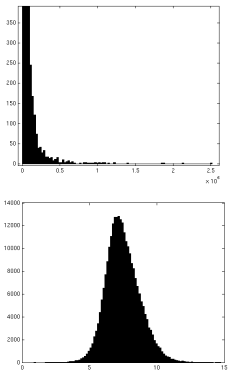
$$= \sum_{i=0}^{G-1} \left(a^2 i^2 + 2aib + b^2\right) p(i) - \left(\sum_{i=0}^{G-1} (ai+b)\,p(i)\right)^2$$

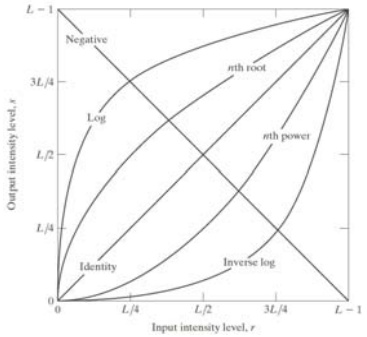
$$= a^2 \left(\sum_{i=0}^{G-1} i^2\,p(i) - \left(\sum_{i=0}^{G-1} i\,p(i)\right)^2\right) = a^2\,\sigma^2$$

- 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

- Hvilken av transformasjonene til høyre er brukt her?





(Fig 3.3 i DIP)

Power-law (gamma)-transformasjoner

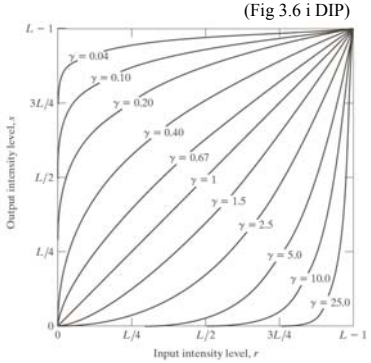
- Mange bildeproduserende apparater har et input/output-forhold som kan beskrives som:

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

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)



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 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 $z=T_g^{-1}(s)$

Lokale operasjoner i bilder

- Vi skal bare se på teknikker i bilde-domenet
- Bilde-domenet refererer til mengden av piksler som utgjør det digitale bildet.
- Bildeplan-metoder opererer på disse pikslene og gir:

$$g(x,y) = T[f(x,y)]$$

g(x,y) er ut-bildet

f(x,y) er inn-bildet

T er en operator på en **omegn** rundt *(x,y)*

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

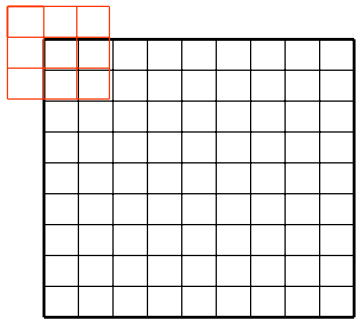
  der * er konvolusjons-operatoren

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

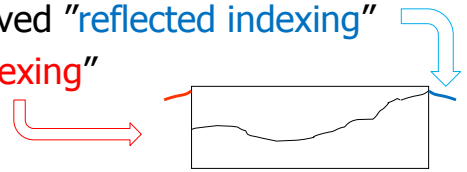
  3x3: (M-2)x(N-2)
  5x5: (M-4)x(N-4)



Hva gjør vi langs randen?

Alternativer:

1. Sett g(x,y)=0
2. Sett g(x,y)=f(x,y)
3. Trunkér ut-bildet
4. Trunkér 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.

Ikke-uniformt lavpass-filter

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

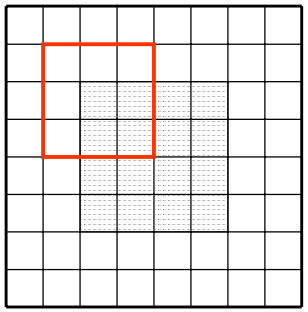
$$h(x,y) = \exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right)$$

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

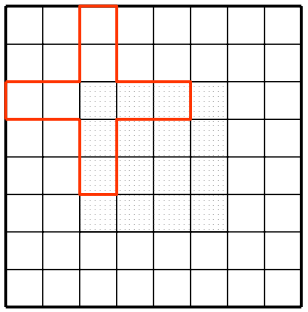
Median-filter

- 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.

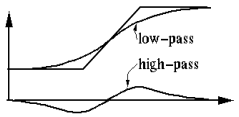
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- 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.

"High-boost" filtrering

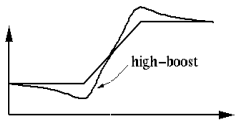
- Høypass-filtering kan utføres ved
  Høypass = Original – Lavpass

$$\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}=\begin{bmatrix}0&0&0\\0&9&0\\0&0&0\end{bmatrix}-\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$



- Adder høypass-resultatet til originalen, og få et "High-boost" bilde

$$\begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}+\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}=\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}$$



- Legg forskjellige vekter på de to bildene, og få et "veiet high boost" bilde.

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

"Unsharp masking"

- Middelverdfilter => uskarpt bilde.
- Subtraher uskarpt bilde fra originalen.
- Adder differanse til originalen.
- Resultatet er skarpere enn originalen.

Digitale gradient-operatorer

- Vi husker at den deriverte av *f(x)* er gitt ved

$$\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

- I våre digitale bilder setter vi h ≥ 1.
- Vi får da en gruppe av operator som approksimerer de ortogonale gradient-komponentene $\delta F(x,y)/\delta x$ og $\delta F(x,y)/\delta y$

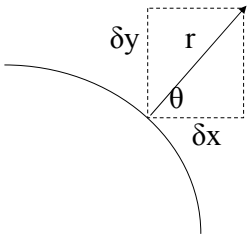
- Noen operatorer gir bare estimat av gradient-magnituden.
- Andre gir også gradient-retningen.

- Gitt to digitale filtermasker H_x og H_y.
  - Disse to konvolveres med bildet *F(i,j)* og måler gradient komponentene g_x og g_y i en omegn om *(i,j)* i bildet.

Gradient i et kontinuerlig bilde

- Gradienten til F langs r i retning θ

$$\frac{\partial F}{\partial r}=\frac{\partial F}{\partial x}\frac{\partial x}{\partial r}+\frac{\partial F}{\partial y}\frac{\partial y}{\partial r}$$



dvs. $$\frac{\partial F}{\partial r}=\frac{\partial F}{\partial x}\cos\theta+\frac{\partial F}{\partial y}\sin\theta$$

- Gradienten er størst når $$\frac{\partial}{\partial\theta}\left(\frac{\partial F}{\partial r}\right)=0$$

- Dvs. for den vinkelen θ_g der

$$-\frac{\partial F}{\partial x}\sin\theta_g+\frac{\partial F}{\partial y}\cos\theta_g=0 \quad\Leftrightarrow\quad \frac{\partial F}{\partial y}\cos\theta_g=\frac{\partial F}{\partial x}\sin\theta_g$$

- Men $\delta F/\delta x$ og $\delta F/\delta y$ er bare de ortogonale gradient-komponentene g_x og g_y i hhv. x- og y-retning i bildet.

Gradient i kontinuerlig bilde - II

- Vi gjentar: Gradienten er størst for den vinkelen θ_g der

$$\frac{\partial F}{\partial y}\cos\theta_g = \frac{\partial F}{\partial x}\sin\theta_g \quad \Rightarrow \quad \frac{\sin\theta_g}{\cos\theta_g} = \frac{\partial F / \partial y}{\partial F / \partial x} \quad \Rightarrow \quad \tan\theta_g = \frac{g_y}{g_x}$$

- Derfor: Retningen til kanten relativt til x-aksen er gitt ved

$$\theta_g = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- Og gradient-magnituden er gitt ved kvadratroten av summen av kvadratene av de to gradient-komponentene:

$$\left(\frac{\partial F}{\partial r}\right) = \left[g_x^2 + g_y^2\right]^{1/2}$$

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

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

Separasjon av gradient-filtre

- Separasjon av Prewitt-operatoren:

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

- Separasjon av Sobel-operatoren:

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

- Separasjon av Frei-Chen:

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

Implementasjoner av en gradient-operator

- Større filtre gir mindre følsomhet for støy.
- Sobel (og de andre operatorene) kan utvides til større enn 3×3.
- Her er en 5×5 Sobel operator:

$$h_x = \begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -8 & -4 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -8 & -4 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix} \quad h_y = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Krever 50 multiplikasjoner

- Denne kan implementeres som

$$h_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Krever 36 multiplikasjoner

Implementasjoner av en gradient-operator II

- De to 3×3 filtrene er separable, og kan skrives

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

Krever 24 multiplikasjoner

- Eller vi kan skrive

$$h_x = \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$h_y = \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Krever 32 multiplikasjoner

- Raskeste blir

$$h_x = \begin{bmatrix} 1 & 2 & 0 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \quad h_y = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix}$$

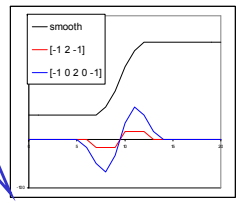
Krever bare 20 multiplikasjoner

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" $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Implementasjoner av en 5×5 Laplace-operator

- Laplace-operatoren

$$\nabla^2 = -\begin{bmatrix} 2 & 4 & 4 & 4 & 2 \\ 4 & 0 & -8 & 0 & 4 \\ 4 & -8 & -24 & -8 & 4 \\ 4 & 0 & -8 & 0 & 4 \\ 2 & 4 & 4 & 4 & 2 \end{bmatrix}$$

Krever 2×25 operasjoner

er ikke separabel.

- Men den kan skrives som

$$\nabla^2 = -\left(\begin{bmatrix} 1 & 0 & -2 & 0 & 1 \\ 4 & 0 & -8 & 0 & 4 \\ 6 & 0 & -12 & 0 & 6 \\ 4 & 0 & -8 & 0 & 4 \\ 1 & 0 & -1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \right)$$

som igjen kan skrives som

$$\nabla^2 = -\left(\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} + \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

Vi ser at

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

kan finnes ved 2 ganger derivasjon med Sobel – filtrene

- Separasjon gir den raskeste implementasjonen: Krever 2×20 operasjoner

$$\nabla^2 = -\left([1 \;\; 0 \;\; -2 \;\; 0 \;\; 1] * [1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]^T + [1 \;\; 4 \;\; 6 \;\; 4 \;\; 1] * [1 \;\; 0 \;\; -2 \;\; 0 \;\; 1]^T \right)$$

Fra Laplace til LoG

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

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

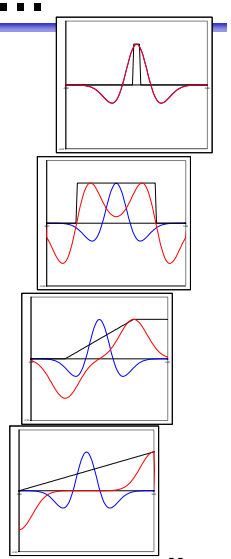
- Vi kan gjøre det samme med Laplace-operatoren
  - Vi bruker et Gauss-filter G
  - Og siden konvolusjon er kommutativ, får vi

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

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

Legg merke til at ...

- En struktur som er smalere enn LoG-kjernen, gir to nullgjennomganger.
- Kjernen bestemmer avstanden mellom dem.
- Hvis strukturen er bredere enn kjernen, men smalere enn filteret, blir kantene riktig detektert.
- På en rampe som er bredere enn kjernen, men smalere enn filteret, finner LoG en nullgjennomgang midt på.
- På ramper som er bredere enn filteret, finner ikke LoG noen nullgjennomgang, bare et null-platå.



Canny's algoritme

- I Canny's algoritme gjøres kant-lokalisering slik:
  - Lavpass-filtrering med et Gauss-filter (gitt σ).
  - Gradient-deteksjon med Prewitt eller Sobel-operatoren.
  - 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.