IN2070 - Filtrering i bildedomenet

2. mars 2022

- Høypassfiltrering: Bildeforbedring og kantdeteksjon
- Førstederivert; gradient-operatorer
- Andrederivert; Laplace-operatoren og LoG-operatoren
- Cannys kantdetektor

Høypassfiltre

- Slipper gjennom høye frekvenser og demper/fjerner lave frekvenser
- Effekt:
  - Demper langsomme variasjoner
  - Fremhever skarpe kanter, linjer og detaljer
- **Typiske mål:** Forbedre skarpheten, detektere kanter
- Utfordring: Kan risikere å fremheve støy!

Høypassfiltrering med konvolusjon

- Summen av vektene i konvolusjonsfilteret er typisk 0
- Får positive og negative pikselverdier i utbildet
- Kan være uheldig å alltid bruke resultatet etter konvolusjonen med et høypassfilter, $g(x, y)$, sin absoluttverdi, $|g(x, y)|$
- For framvisning: Gjør $g(x, y)$ positiv med å addere med en konstant og skalér resultatet til et ønsket intervall
- Antar her nullutvidelse og bruker alle posisjoner med overlapp!

Høypassfiltrering ved konvolusjon (kursorisk)

Når summen av vektene i konvolusjonsfiltret er 0, blir også summen av utbildets piksler lik 0

$$\sum_{x=-a}^{M-1+a}\sum_{y=-b}^{N-1+b}(f*h)(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b}\sum_{x=-a}^{M-1+a}\sum_{y=-b}^{N-1+b}h(s,t)f(x-s,y-t)$$

$$\overset{\text{antar nullutvidelse!}}{=} \sum_{s=-a}^{a}\sum_{t=-b}^{b}\sum_{x=s}^{M-1+s}\sum_{y=t}^{N-1+t}h(s,t)f(x-s,y-t)$$

$$= \sum_{s=-a}^{a}\sum_{t=-b}^{b}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}h(s,t)f(m,n)$$

$$= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\left(\sum_{s=-a}^{a}\sum_{t=-b}^{b}h(s,t)\right)$$

$$= 0$$

Punkt-deteksjon

- Eksempel på høypassfilter:

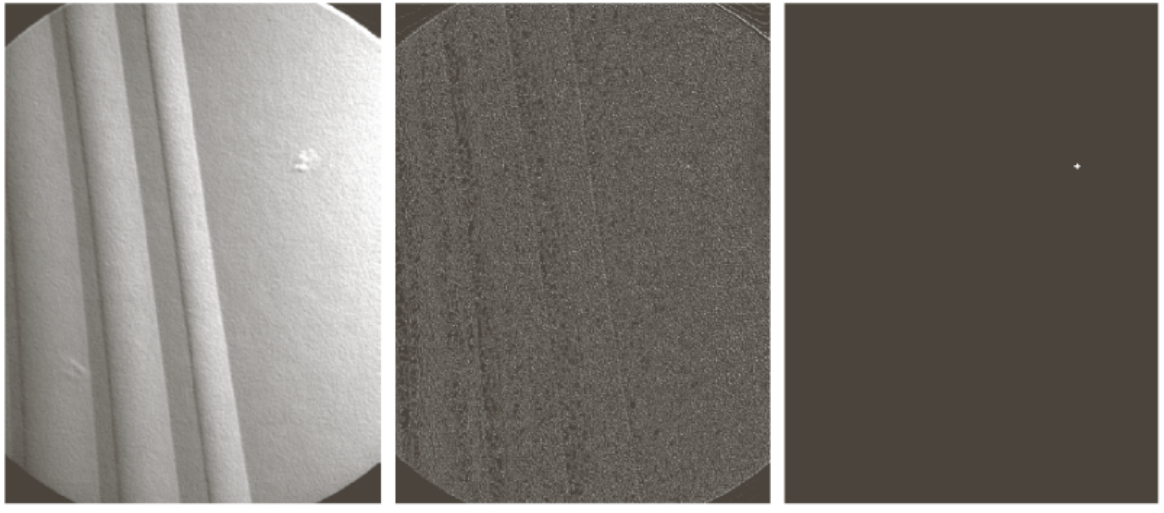
$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

- Dette filteret kan bl.a. brukes til deteksjon av isolerte punkter:
  - Isolerte punkter vil skille seg ut med høy respons (i absoluttverdi)
  - For passende terskel T finner vi de detekterte punktene der $|g(x, y)| \geq T$

Eksempel: Punkt-deteksjon 1

Deteksjon av pore i turbinblad:

1	1	1
1	−8	1
1	1	1



a
b c d

FIGURE 10.4
(a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

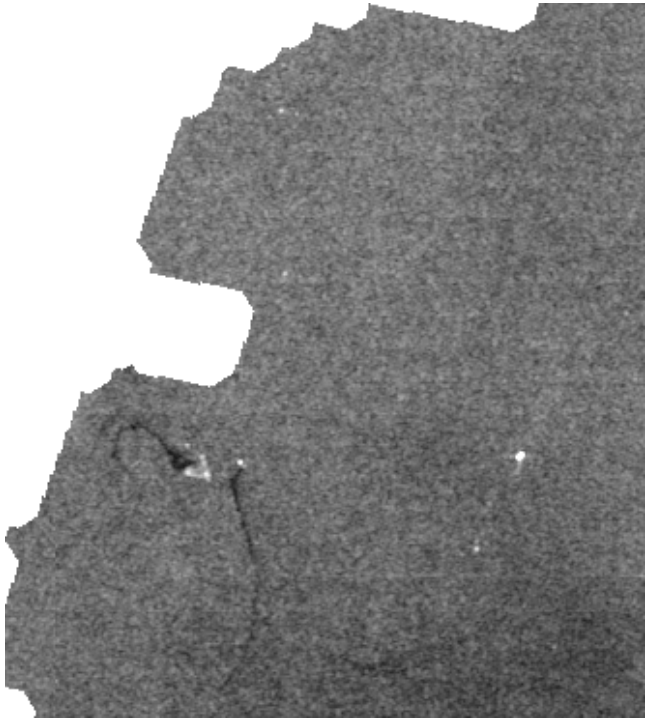
Eksempel: Punkt-deteksjon 2

Deteksjon av skip i radar-bilde over sjø:

- De små lyse punktene er skipene
- Filteret

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

vil gi flere høye responser for hver skip
(nesten like høy respons i kanter og spesielt hjørner)



Bildeforbedring med høypassfiltrering

- Konvolusjonsfilteret kan også brukes til bildeforbedring:

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

- Tanke bak bildeforbedringen:
  - Filtreringen detekterer starten og slutten av kanter
  - Andre områder blir omtrent lik 0
  - Ved å *addere* filtreringen til originalen får vi sterkere kanter og bildet virker skarpere

Eksempel: Bildeforbedring ved høypassfiltrering

Øke skarpheten i et bilde:

Eksempel: Bildeforbedring ved høypassfiltrering

Øke skarpheten i et bilde:

Mulig løsning:

1. Filtrer bildet med:

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

Eksempel: Bildeforbedring ved høypassfiltrering

Øke skarpheten i et bilde:

Mulig løsning:

1. Filtrer bildet med:

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

2. Summér filtrert bilde med innbilde:

Bildeforbedring ved høypassfiltrering

- Konvolusjonsfilteret vi har sett på kan uttrykkes slik:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right)$$

$$\text{Høypass} = \text{Identitet} - \text{Lavpass}$$

Bildeforbedring ved høypassfiltrering

- Konvolusjonsfilteret vi har sett på kan uttrykkes slik:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right)$$

$$\text{Høypass} = \text{Identitet} - \text{Lavpass}$$

Konvolusjon er distributiv: $f * (g + h) = f * g + f * h$

Bildeforbedring ved høypassfiltrering

- Konvolusjonsfilteret vi har sett på kan uttrykkes slik:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right)$$

$$\text{Høypass} = \text{Identitet} - \text{Lavpass}$$

Konvolusjon er distributiv: $f * (g + h) = f * g + f * h$

- Bildeforbedringen vi har sett på tilsvarer tre steg:

Bildeforbedring ved høypassfiltrering

- Konvolusjonsfilteret vi har sett på kan uttrykkes slik:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right)$$

$$\text{Høypass} = \text{Identitet} - \text{Lavpass}$$

Konvolusjon er distributiv: $f * (g + h) = f * g + f * h$

- Bildeforbedringen vi har sett på tilsvarer tre steg:
  1. Lavpassfiltrér med 3×3 middelverdifilter
  2. Subtrahér resultatet fra original bildet
  3. Skalér opp med 9

Bildeforbedring ved høypassfiltrering

- Konvolusjonsfilteret vi har sett på kan uttrykkes slik:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} = 9\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \right)$$

$$\text{Høypass} = \text{Identitet} - \text{Lavpass}$$

Konvolusjon er distributiv: $f * (g + h) = f * g + f * h$

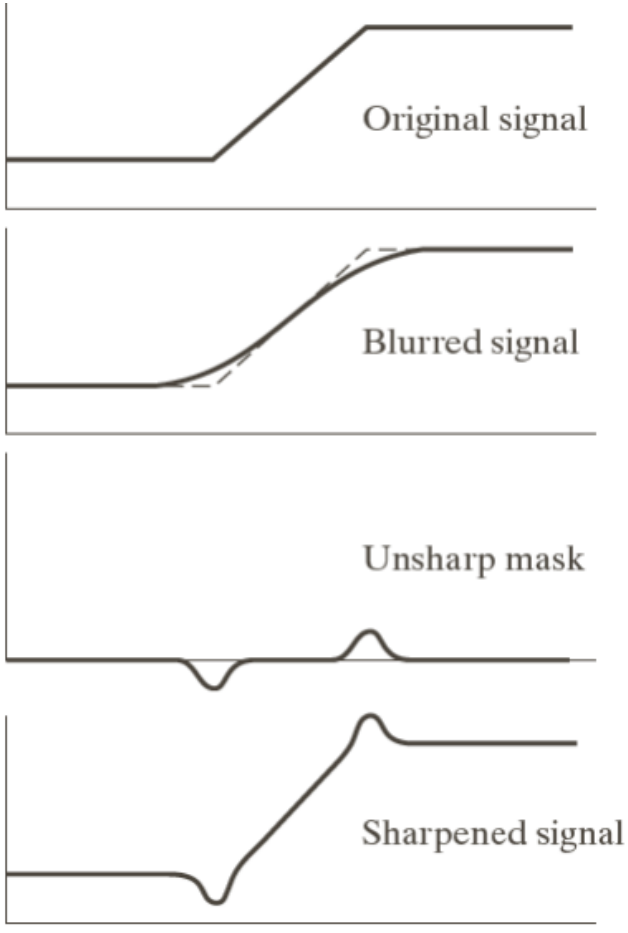
- Bildeforbedringen vi har sett på tilsvarer tre steg:
  1. Lavpassfiltrér med 3×3 middelverdifilter
  2. Subtrahér resultatet fra original bildet
  3. Skalér opp med 9
- Dette er en form for ”highboost”-filtrering

Konvolusjon er assosiativ ved skalar multiplikasjon:

$$a(f * g) = f * (ag)$$

”Unsharp masking” og ”highboost”-filtrering

Gitt et bilde (original):

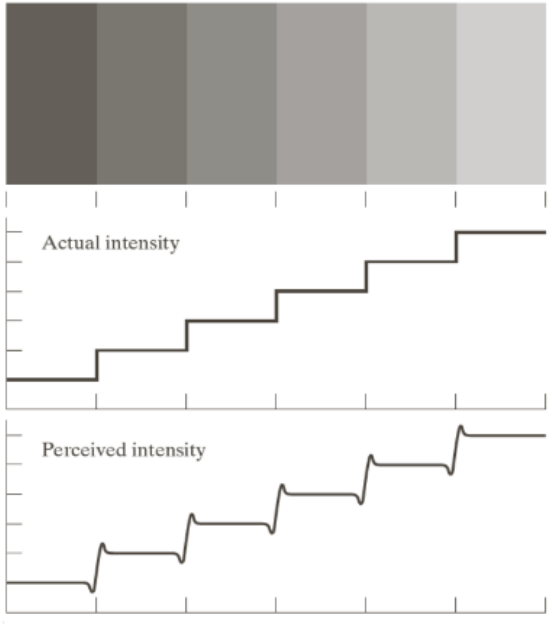


1. Lavpassfiltrér
2. Beregn differansen: original - lavpassfiltrert
3. Resultatet er: original + k·differansen
   - **Unsharp masking:** $k = 1$
   - **Highboost:** $k > 1$

Fra introduksjonsforelesningen:

Nevrale prosessorer i netthinna

- Forsterker kanter
- Stimulering av én del undertrykker stimulering av en annen del
- Øker kontrasten ved overgang mellom uniforme regioner
- Kalles for *Mach-bånd*



30

Eksempel: Unsharp-masking

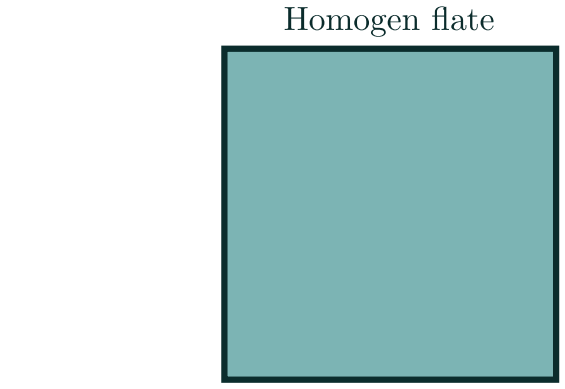
1. Lavpassfiltrering $\rightarrow$ uskrapt bilde
2. Subtrahér uskrapt bilde fra originalen, original - lavpass = høypass
3. Adder differansen til originalen for å fremheve kanter

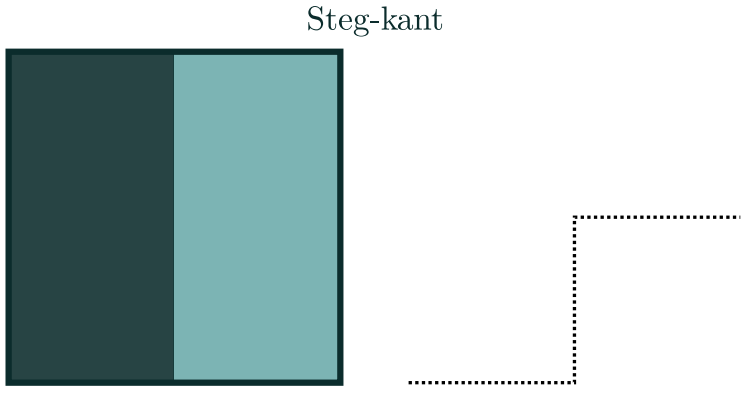
Motivasjon for kantdeteksjon

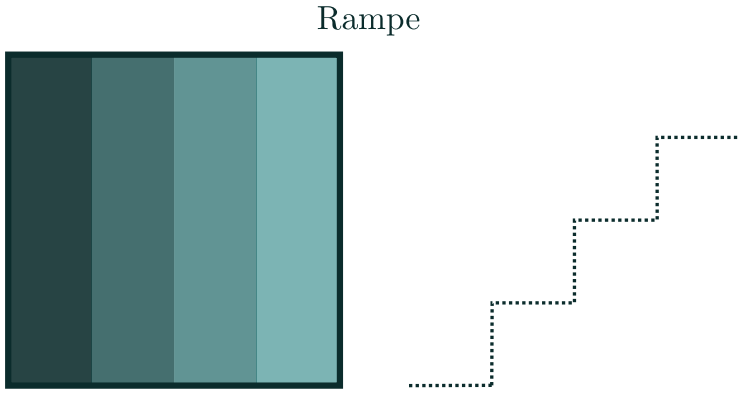
- Meste av informasjonen i et bilde finnes ved kantene til objektene/regionene i bildet
  - Kanter kan innbære bl.a. intensitets-kanter, farge-kanter og tekstur-kanter
- Biologiske synssystemer er basert på kant-deteksjon

Intensitets-flater, -kanter og -linjer

- **Homogen flate:** Et område der alle pikselverdiene er like
- **Kant:** Overgang mellom to områder med forskjellig pikselverdi
  - Steg-kant: Overgang over én pikselverdi
  - Rampe: Overgang over flere pikselverdier med konstant intensitetsendring





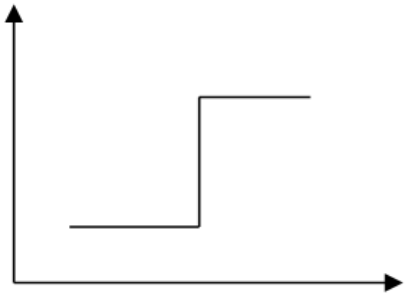


Intensitets-flater, -kanter og -linjer

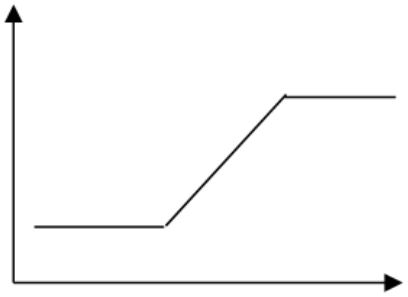
- ”Kant” brukes også om skillepunktet mellom de to områdene;
  - Forskjellige måter å modellere hvor skillet er
  - For steg-kanter:
    - Et alternativ er midt mellom nabopiksler som tilhører forskjellige områder
    - I segmentering ønsker man ofte å finne første pikslen på den siden som tilhører objekt
  - Merk at en linje består av to kanter:

  - Finner ofte i praksis strukturer som ligner steg-kant, rampe og linjer

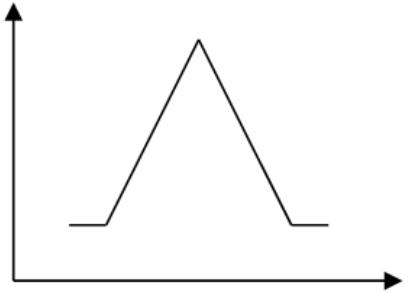
Kant-typer



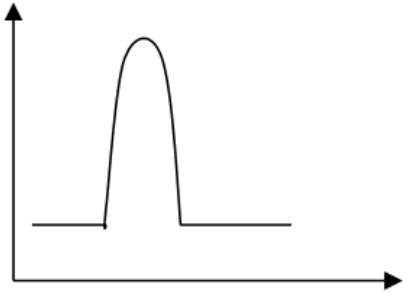
Steg-kant



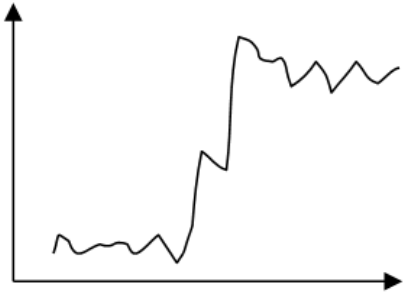
Rampe



Tak-kant



Linje
(litt glattet)



Steg-kant med støy

Digital derivasjon

- En kant kjennetegnes ved endring i intensitetsverdi
- Den **deriverte** av en funksjon $f(x)$ er definert som:

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

  og angir hvor mye f endrer seg i punktet x
- Derivasjon er ikke definert i diskrete tilfeller, men vil kan approksimere den ved å la f.eks la $h \geq 1$ i definisjonen av derivert og bruke *differansen* mellom nærliggende piksler

Derivasjon av bilder

- Et digitalt bilde er en to-variabel, diskret funksjon
- En kontinuerlig funksjon $f(x, y)$ kan deriveres mhp x og y
  - Kalles **partiell-derivasjon** mhp x og y
  - Betegnes henholdsvis $\frac{\partial f(x,y)}{\partial x}$ og $\frac{\partial f(x,y)}{\partial y}$
- **Gradient:** vektoren av de to partiell-deriverte,

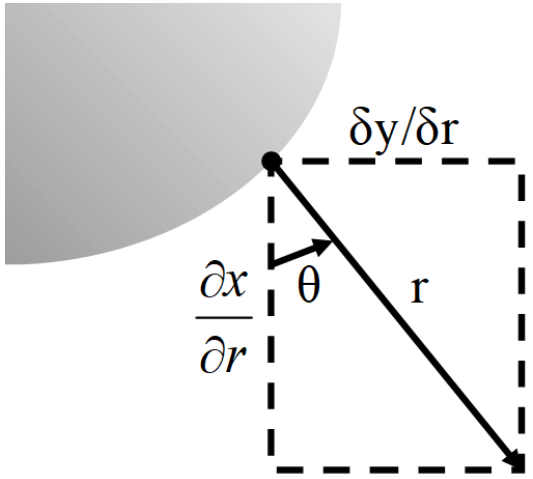
$$\nabla f(x, y) = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix}$$

Gradient i et kontinuerlig bilde

Gradienten peker i retningen der funksjonen forandrer seg mest

- Retningsderiverte til f i retning θ (langs r) er:

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r}$$
$$= \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$



- Retningsderivert er størst når:

$$\frac{\partial}{\partial\theta}\left(\frac{\partial f}{\partial r}\right) = 0$$

- Vinkelen θ_g der retningsderivert er størst når

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

Gradient i et kontinuerlig bilde

- (fort. forrige slide) retningsderivert er størst når vinkelen θ_g oppfyller:

$$\frac{\partial f}{\partial y}\cos\theta_g = \frac{\partial f}{\partial x}\sin\theta_g$$

$$\frac{\sin\theta_g}{\cos\theta_g} = \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}$$

$$\tan\theta_g = \frac{g_y}{g_x}$$

- **Retningen** gradienten peker i der funksjonen forandrer seg mest:

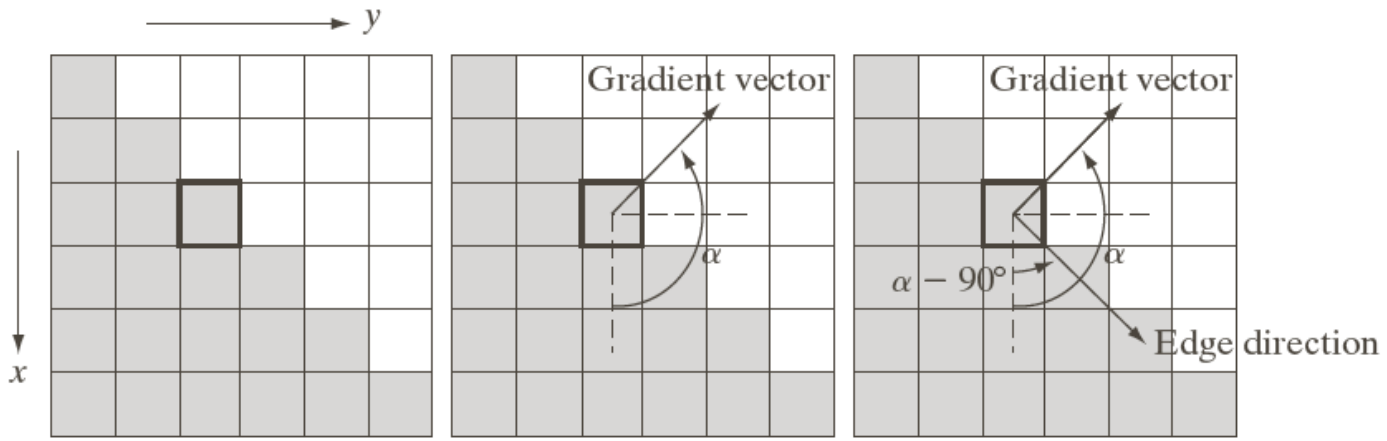
$$\theta_g = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- ... og størrelsen på forandringen, **gradient-magnituden**, er:

$$\frac{\partial f}{\partial r_g} = \sqrt{g_x^2 + g_y^2}$$

Gradient ⊥ Kant

Gradienten peker i retningen der funksjonen forandrer seg mest og kanten går vinkelrett på gradienten:



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

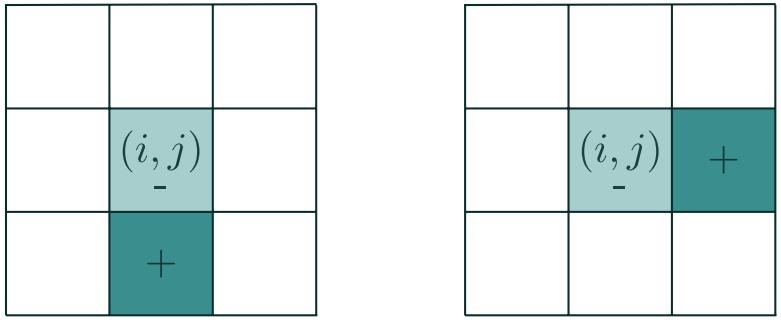
Digitale gradient-tilnærminger

- Ønsker å tilnærme gradienten med differanser mellom nærliggende piksler
- Kan bruke to konvolusjonsfiltre som tilnærmer hver sin gradient-komponent:
  - To slike konvolusjonsfiltre kalles en **gradient-operator**
  - Konvolusjonsfiltrene betegnes ofte som h_x og h_y
  - h_x tilnærmer partiell-derivert i x-retning ved å beregne differansen i vertikal retning av nærliggende piksler (h_y beregner differensen i horisontal retning)
- Mange muligheter!

Digitale gradient-tilnærminger

- Asymmetrisk 1D-operator:

$$g_x(i,j) = f(i+1,j) - f(i,j)$$
$$g_y(x,y) = f(i,j+1) - f(i,j)$$

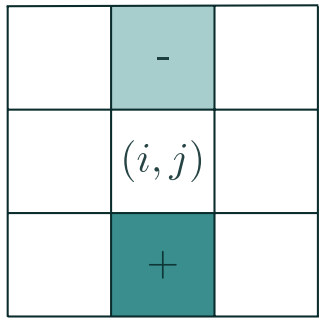


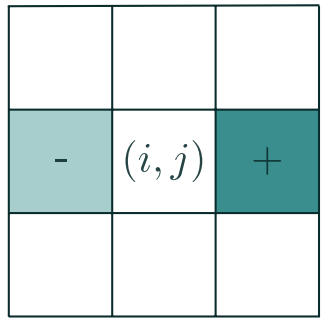
- Her vil gradient-komponentene bli positive for en kant med intensitet som øker nedover og fra venstre mot høyre i bildet
- Ulemper med denne operatoren:
  - Hver av gradient-estimatorene refererer til et punkt midt mellom to piksler
  - $x-$ og $y-$estimatet refererer ikke til samme punkt i bildet

Digitale gradient-tilnærminger

- Symmetrisk 1D-operator:

$$g_x(i,j) = f(i+1,j) - f(i-1,j)$$
$$g_y(x,y) = f(i,j+1) - f(i,j-1)$$





- Gradient-estimatorene referer nå til pikselen på posisjon (i,j)
- Gjør dette på to, ortogonale retninger (vertikalt og horisontalt): Tilsvarer å mildt glatte bildet i den ene retningen, for så finne gradient-tilnærmingen i den andre retningen:

$$\begin{pmatrix} 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$$

- Normalt sett uproblematisk, og kan være gunstig også

Gradient-operatorer

- Asymmetrisk 1D-operator ("pixel-difference"-operator):

$$h_x = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad h_y = \begin{pmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Symmetrisk 1D-operator ("separated pixel difference"-operator):

$$h_x = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \quad h_y = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

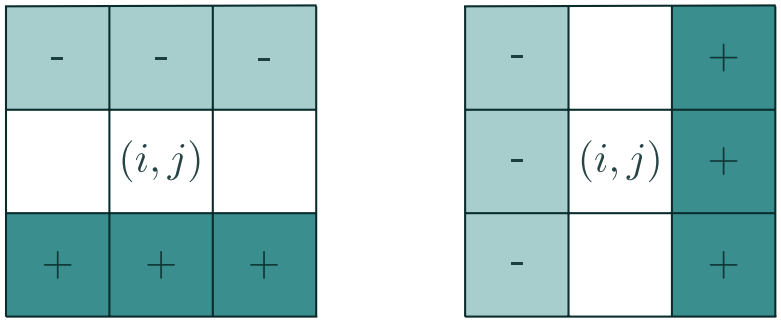
- Roberts-operatoren (Roberts kryssgradient-operator):

$$h_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad h_y = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Filtrene kan avvike med 180 grader fra boka siden vi angir dem til bruk ved konvolusjon. Boka antar de skal brukes til korrelasjon.

Digitale gradient-tilnærminger

- Ulempe ved 1D-operatorene: De er veldig følsomme for støy og støy kan lett bli detektert som kanter!
- En mulig måte å håndtere dette på: Beregn differansen for tre symmetriske par:



- Gradient-estimatorene blir mer robuste mot støy i bildet!

Gradient-operatorer

- Prewitt-operatoren:

$$h_x = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad h_y = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

- Sobel-operatoren:

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

- Frei-Chen-operatoren:

$$h_x = \begin{pmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{pmatrix} \quad h_y = \begin{pmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{pmatrix}$$

Filtrene kan avvike med 180 grader fra boka siden vi angir dem til bruk ved konvolusjon. Boka antar de skal brukes til korrelasjon.

Separasjon av gradient-operatorer

- Seperasjon av Prewitt-operatoren:

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

- Seperasjon av Sobel-operatoren:

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

- Seperasjon av Frei-Chen-operatoren:

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

Egenskaper ved gradient-operatorer

- Operatoren gjøres mindre følsom for støy ved å lavpassfiltrere i én retning og derivere i den andre retninger
- Prewitt: mer følsom for horisontale og vertikale kanter enn for diagonale kanter
- Sobel: motsatt av for Prewitt
- Frei-Chen: gir samme gradient-magnitude hvis kanten ligger langs aksene eller diagonalt

Gradient-beregning

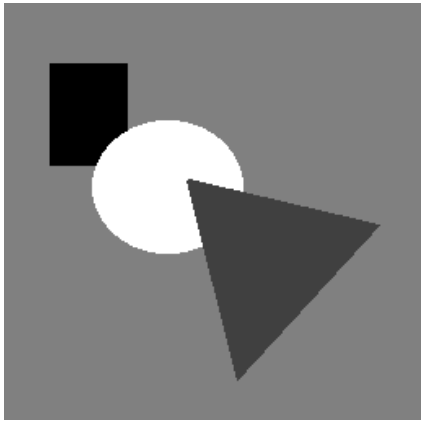
- Horisontale kanter: $g_x = h_x * f$
- Vertikale kanter: $g_y = h_y * f$
- Gradient-magnitude og -retning:

$$M(i,j) = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

$$\theta(i,j) = \tan^{-1}\left(\frac{g_y(i,j)}{g_x(i,j)}\right)$$

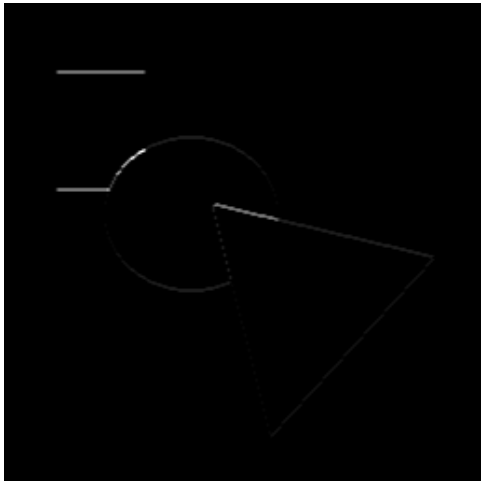
Eksempel: Gradient-beregning med Sobel-operatoren

Innbilde f

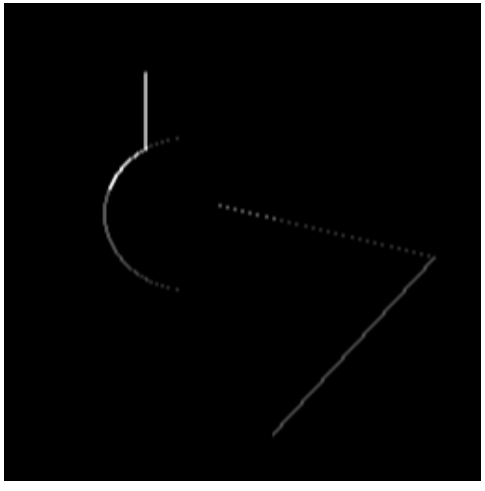


$g_x = f * h_x$

g_x^2

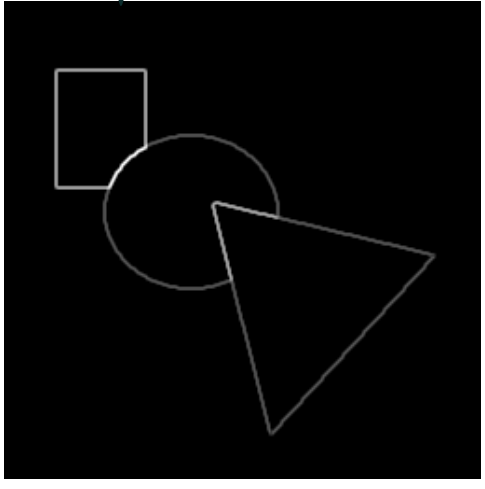


$g_y = f * h_y$



g_y^2

$\sqrt{g_x^2 + g_y^2}$



Større gradient-operatorer

- Gradient-operatorer kan gjøres mer støy-robuste ved å bygge inn mer lavpassfiltrering
- 5×5 Sobel-operator som eksempel:

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

... er resultatet av konvolusjonene

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

Implementasjoner av gradient-operatorer

- Lurt å utnytte separabilitet
- For 5×5-Sobel operatoren på forrige foil:
  - Med 5×5 filtre kreves 50 multiplikasjoner
  - Ved bruk av de fire 3×3-filtrene kreves 36 multiplikasjoner
  - Finnes mange måter å dele opp på, men raskeste er å separere 5×5-filtrene direkte:

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

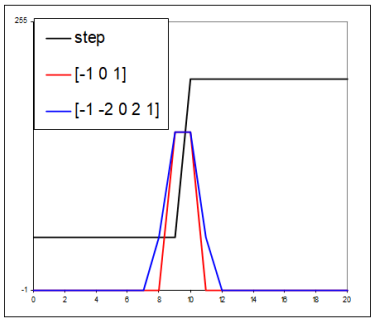
Disse krever kun 20 multiplikasjoner!

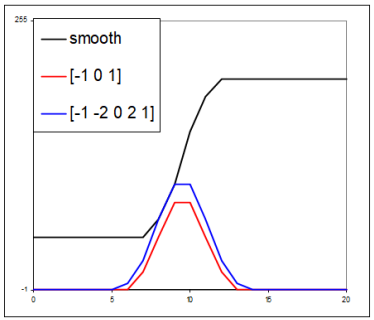
Gradient til kant-deteksjon

- Gradient-magnituden indikerer styrken av kanten i en piksel
- Kantene i gradient-magnituden blir typisk brede
  - Kan være uønsket
  - Bredden avhenger av størrelsen på filtrene og bredden på kanten i bildet
- Mulige fremgangsmåter for å finne en eksakt, tynn kant:
  - Terskle gradient-magnitude og tynne den
  - Finne maksimum i gradient-magnitude (eller bruke andrederivert)

Gradient til kant-deteksjon

- Gradient-magnituden har bred respons, men vi ønsker gjerne en tynn, eksakt kant
- For en steg-kant: Bredden på responsen er avhengig av størrelsen på filteret
- For en bred kant(rampe): Bredden på responsen er avhengig av bredden på kanten
- Maskimum er likt og passende lokalisert!





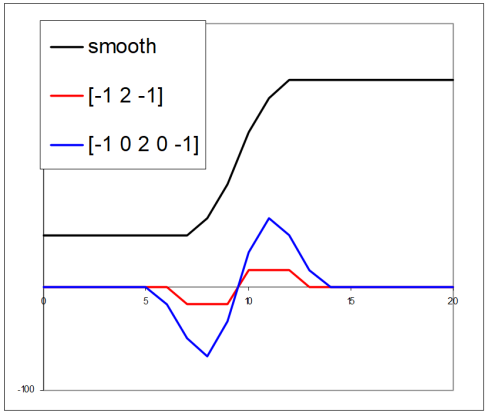
Laplace - operatoren

- Laplace-operatoren er gitt ved:

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

  - Den endrer fortegn der f er et vendepunkt
  - $\nabla^2 f = 0$ markerer kant-posisjon
  - $\left|\nabla^2 f\right|$ har to ekstremverdier pr kant; på starten og på slutten av kanten $\rightarrow$ var dette vi brukte tidligere til å forbedre bildeskarphet!

- **Nullgjennomgang:** Kantens eksakte posisjon
- Finner kant-posisjoner, men ikke retninger



1D-Laplace-operator

- I 1D er $\nabla^2 f$ ekvivalent med den andrederiverte
- Sentrert om indeks i
- Bytter fortegn av konvensjon
- Får at $-\nabla^2 f = -(f[i-1] - 2f[i] + f[i+1])$

Kontinuerlig	Digitalt
$f(x)$	$f[i]$
$f'(x)$	$f[i+1] - f[i]$
$f''(x)$	$f'[i+1] - f'[i] = (f[i+2] - f[i+1]) - (f[i+1] - f[i]) = f[i+2] - 2f[i+1] + f[i]$

2D-Laplace-operator

- Anvend 1D-Laplace operatoren i begge retninger og summér:

$$\begin{aligned} -\nabla^2 f &= -\frac{\partial^2 f}{\partial x^2} - \frac{\partial^2 f}{\partial y^2} \\ &\approx -f(i-1,j) + 2f(i,j) - f(i+1,j) - \\ &\quad f(i,j-1) + 2f(i,j) - f(i,j+1) \end{aligned}$$

- Dette kan beregnes ved å konvolvere f med:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Full 3×3-Laplace - operator

- Hvis vi i tillegg anvender 1D-Laplace tilnærmingen langs begge diagonaler, får vi det som kan beregnes ved:

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

- Dette er filteret vi brukte til:
  - Punkt-deteksjon
  - Øke bildeskarpheten

Laplace på andregradspolynom

- La de lokale intensitetene rundt (x, y) være modellert ved andregradspolynomet:

$$f(m, n) = k_1 + k_2 m + k_3 n + k_4 m^2 + k_5 mn + k_6 n^2$$

  der (m, n) er koordinater relativt til (x, y).
- I et 3×3-naboskap rundt (x, y) har vi intensitetene:

$k_1 - k_2 - k_3 + k_4 + k_5 + k_6$	$k_1 - k_2 + k_4$	$k_1 - k_2 + k_3 + k_4 - k_5 + k_6$
$k_1 - k_3 + k_6$	k_1	$k_1 + k_3 + k_6$
$k_1 + k_2 - k_3 + k_4 - k_5 + k_6$	$k_1 + k_2 + k_4$	$k_1 + k_2 + k_3 + k_4 + k_5 + k_6$

- Den korrekte Laplace-verdien her blir:

$$-\nabla^2 f(x, y) = -\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right) = -2(k_4 + k_6)$$

- Både 4-nabo og 8-nabo Laplace-operatoren gir korrekt estimat:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \frac{1}{3}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

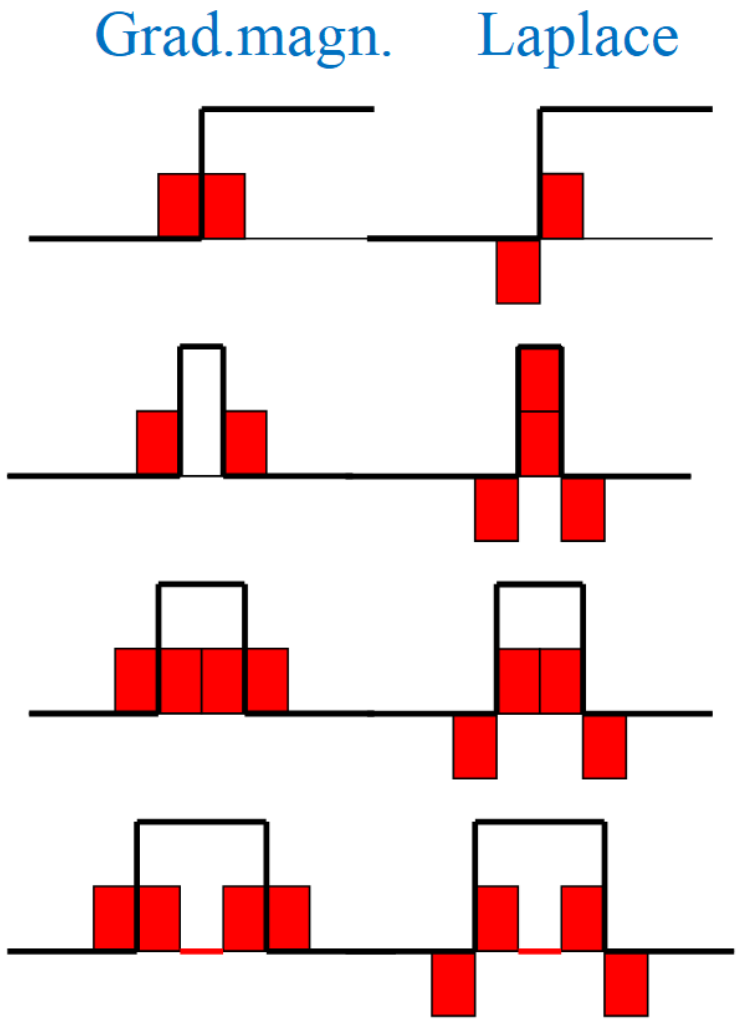
Sobel vs. Laplace

- **Sobel-filtrering:** bred kant

- **Laplace-filtrering:** dobbel kant

Steg-kanter og steg-kantede linjer

- Merk:
  - Grad.-magn. bruker symmetrisk 1D-operator: $f[i+1] - f[i-1]$
  - "Laplace" her er Laplace 1D-operator: $-f[i-1] + 2f[i] - f[i+1]$
- For steg-kanter:
  - Grad.-magn. gir samme respons i første piksel utenfor og første piksel innenfor kanten
  - Laplace gir responser med motsatt fortegn, og riktig kant i nullgjennomgang
- På tvers av en linje (med to steg-kanter):
  - Én piksels linje gir respons på hver side av linja med gradientmagnituden $\rightarrow$ ingen respons
  - Laplace gir riktige kanter i nullgjennomgangene



Andre Laplace-operatorer

- Kan bl.a. finnes ved å bruke gradient-operatorer
- Eksempel på bruk av 3×3 Sobel-operatoren:

$$
\begin{aligned}
-\nabla^2_{5\times 5} &= -\left(\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \right) \\
&= -\left(\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 0 & -2 & 0 & 1 \\ 4 & 0 & -8 & 0 & 4 \\ 6 & 0 & -12 & 0 & 6 \\ 4 & 0 & -8 & 0 & 4 \\ 1 & 0 & -2 & 0 & 1 \end{pmatrix} \right) \\
&= \begin{pmatrix} -2 & -4 & -4 & -4 & -2 \\ -4 & 0 & 8 & 0 & -4 \\ -4 & 8 & 24 & 8 & -4 \\ -4 & 0 & 8 & 0 & -4 \\ -2 & -4 & -4 & -4 & -2 \end{pmatrix}
\end{aligned}
$$

Implementasjon av Laplace-operatorer

- Generelt ikke separable
- Flere Laplace-operatorer kan likevel deles opp i 1D-operasjoner
- F.eks er Laplace-operatoren fra forrige slide ikke separabel, men kan deles opp i 1D-operasjonene

$$\begin{pmatrix}1 & 4 & 6 & 4 & 1\end{pmatrix} * \begin{pmatrix}1 & 0 & 2 & 0 & 1\end{pmatrix}^T + \begin{pmatrix}1 & 0 & 2 & 0 & 1\end{pmatrix} * \begin{pmatrix}1 & 4 & 6 & 4 & 1\end{pmatrix}^T$$

Krever da 37 operasjoner istedenfor 49

Effekt av støy på kant-deteksjon

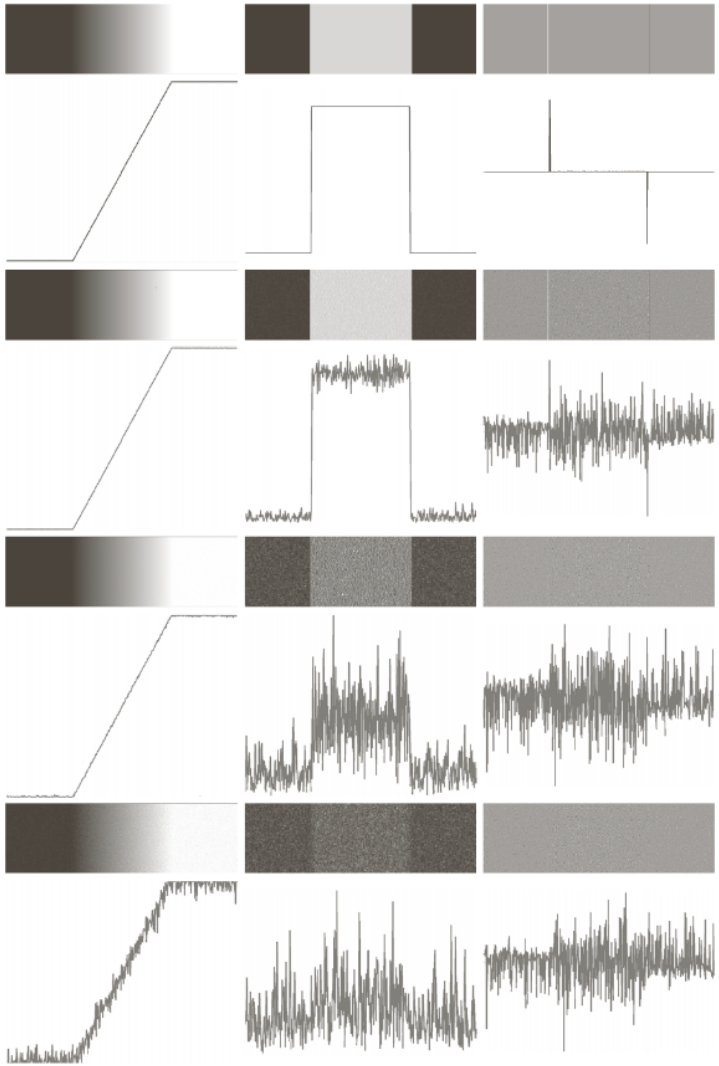


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

Fra Laplace til LoG

- Vi gjorde gradient-operatorene støy-robuste ved å bygge inn lavpassfiltrering

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

- Kan gjøre det samme med Laplace-operator!
- Vanlig å bygge inn et Gauss-filter G med gitt σ:

$$h_{\mathrm{LoG}} = -\nabla^2 * G,$$

som gir oss **Laplacian-of-Gaussian**-operatoren (LoG)!

En 7×7 LoG-operator

Konvolverer vi 3×3-Gauss tilnærmingen (uten skalering med $\frac{1}{16}$),

$$G_{3\times 3} = \begin{pmatrix} 1 & 2 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix},$$

med Laplace-operatoren vi fikk ved å bruke 3×3 Sobel-operatoren

$$-\nabla^2_{5\times 5} = \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix} * \begin{pmatrix} -1 & 0 & 2 & 0 & -1 \end{pmatrix}^T + \begin{pmatrix} -1 & 0 & 2 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix}^T$$

for å få følgende 7×7-LoG - operator:

$$h_{\mathrm{LoG}} = -\nabla^2_{5\times 5} * G_{3\times 3} = \begin{pmatrix} -2 & -8 & -14 & -16 & -14 & -8 & -2 \\ -8 & -24 & -24 & -16 & -24 & -24 & -8 \\ -14 & -24 & 30 & 80 & 30 & -24 & -14 \\ -16 & -16 & 80 & 160 & 80 & -16 & -16 \\ -14 & -24 & 30 & 80 & 30 & -24 & -14 \\ -8 & -24 & -24 & -16 & -24 & -24 & -8 \\ -2 & -8 & -14 & -16 & -14 & -8 & -2 \end{pmatrix}$$

To måter å lage LoG - operatorer

To mulige måter å lage en LoG-operator på:

- Som konvolusjonen av en Laplace-operator og et Gauss-filter
- Som en sampling av LoG funksjonen, som er resultatet av å anvende Laplace-operatoren på Gauss-funksjonen i det kontinuerlige domenet

Disse fremgangsmåtene gir generelt ikke helt like filtre, men begge resulterer i filtre vi kan kalle for LoG-operatorer

Utledning av LoG-funksjonen

1. 2D-Gauss funksjon:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

2. Derivasjon mhp x:

$$\frac{\partial G(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

3. Derivasjon mhp y:

$$\frac{\partial G(x,y)}{\partial y} = -\frac{y}{2\pi\sigma^4}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

4. Andrederivert mhp x:

$$\frac{\partial^2 G(x,y)}{\partial x^2} =$$
$$-\frac{1}{2\pi\sigma^4}\left(1-\frac{x^2}{\sigma^2}\right)\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

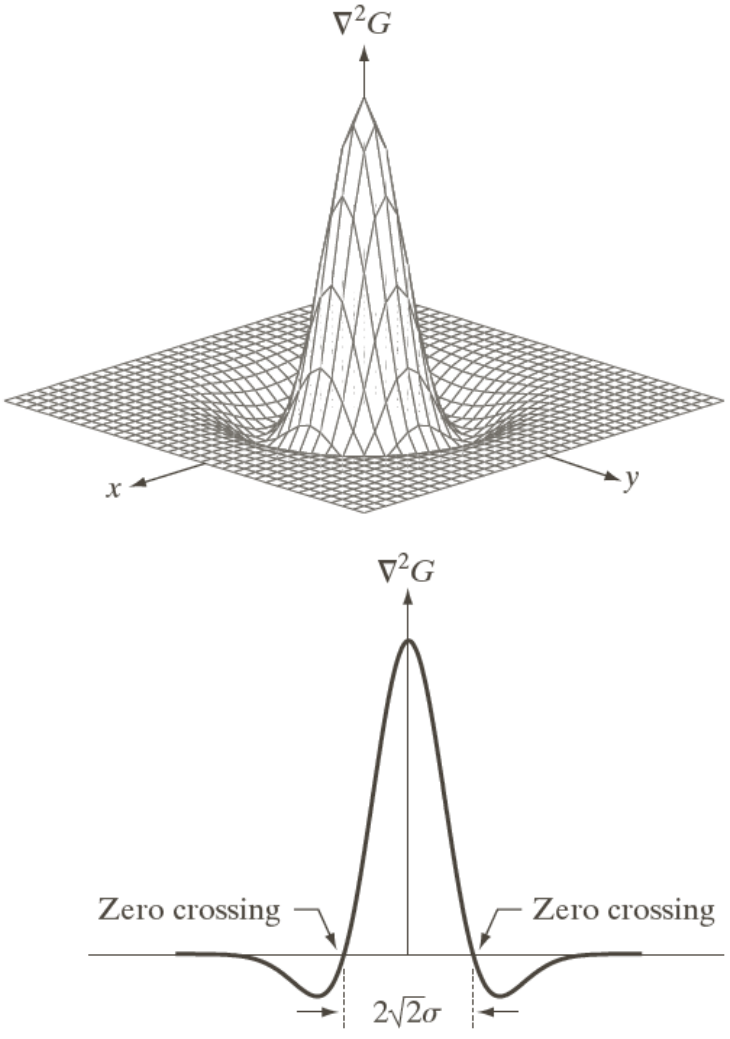
5. Derivasjon mhp y:

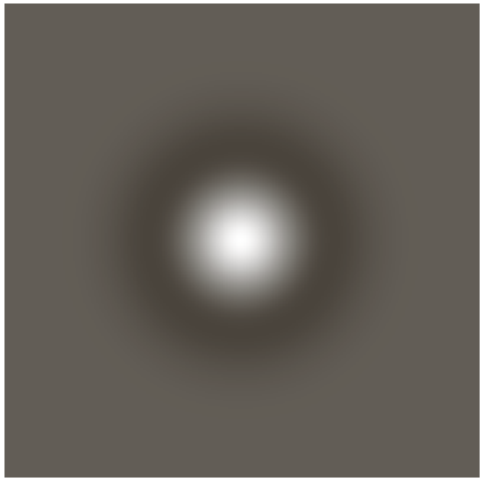
$$\frac{\partial^2 G(x,y)}{\partial y^2} =$$
$$-\frac{1}{2\pi\sigma^4}\left(1-\frac{y^2}{\sigma^2}\right)\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

6. Laplace er summen av disse:

$$-\nabla^2 G(x,y) =$$
$$\frac{1}{2\pi\sigma^4}\left(2-\frac{x^2+y^2}{\sigma^2}\right)\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

LoG-funksjonen





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

FIGURE 10.21
(a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

LoG-operatorer fra LoG-funksjonen

- Får en LoG-operator ved å sample en variant av LoG-funksjonen for heltallige x og y:

$$-\nabla^2 G(x,y) = \frac{1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right)\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right),$$

  der σ er standardavviket til Gauss'en og en verdi man gir som parameter
- I de fleste tilfeller er størrelsen av operatoren lik ca. 8.5σ
  - LoG-funksjonen er omtrent lik 0 utenfor dette området
  - Positive toppen til LoG-funksjonen kalles *kjernen* og $2\sqrt{2}\sigma$ er bredden av denne

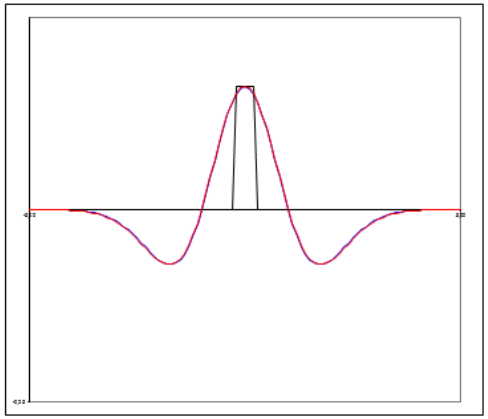
Bruk av LoG-operator

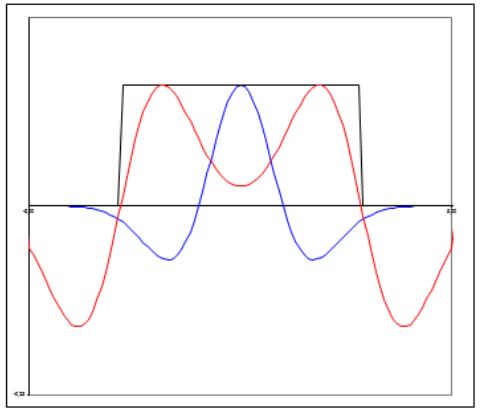
- Laplace-operator detekterer kanter, men er følsomme for støy
- Ofte må man lavpassfiltrere før Laplace-filtrering
- En LoG-operator gjør begge operasjoner i ett
- Fungerer ellers som en Laplace-operator:
  - I homogene områder vil en LoG - operator gi respons i 0
  - Den vil ha positiv respons på den ene siden av kanten, ideelt sett 0 i kantskillet og har negativ respons på den andre siden
  - **Nullgjennomganger angir kanter**
- LoG-operatorer med varierende σ fungerer også godt som ”blob”-detektorer

Kantdeteksjon ved LoG-nullgjennomganger

Tommelfinger-regel for deteksjon av strukturer: *LoG-kjernen må være smalere enn strukturen*

- Hvis strukturen er mindre enn halvparten av LoG-kjernen: nullgjennomgangene vil havne utenfor kantskillene
- Hvis strukturen er større enn halvparten av LoG-filteret: nullgjennomgangene er nøyaktig lik kantskillene
- Et sted imellom: Avhenger av diskretiseringen og tilnærmingen av LoG-filteret

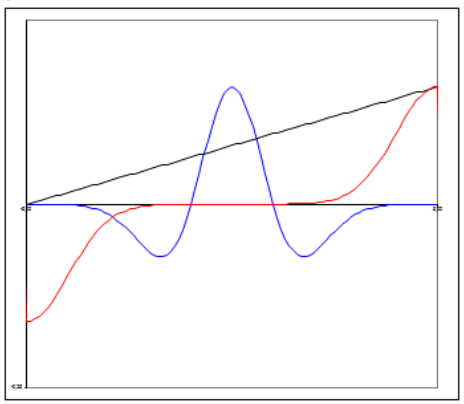


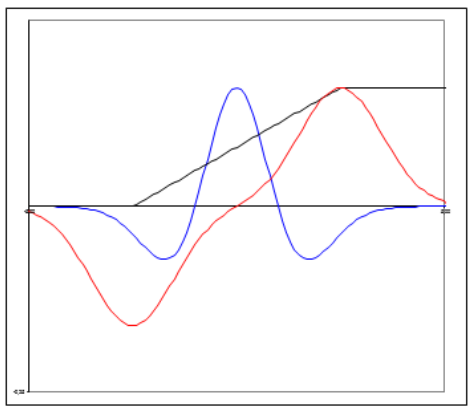


Kantdeteksjon ved LoG-nullgjennomganger

Tommelfinger-regel for deteksjon av ramper: LoG-filteret må være større enn rampen

- Rampen er bredere enn LoG-filteret: Ingen nullgjennomgang
- Ellers, vil nullgjennomgangen være midt på random
- På grunn av støy, må LoG-filteret være betydelig større enn rampen

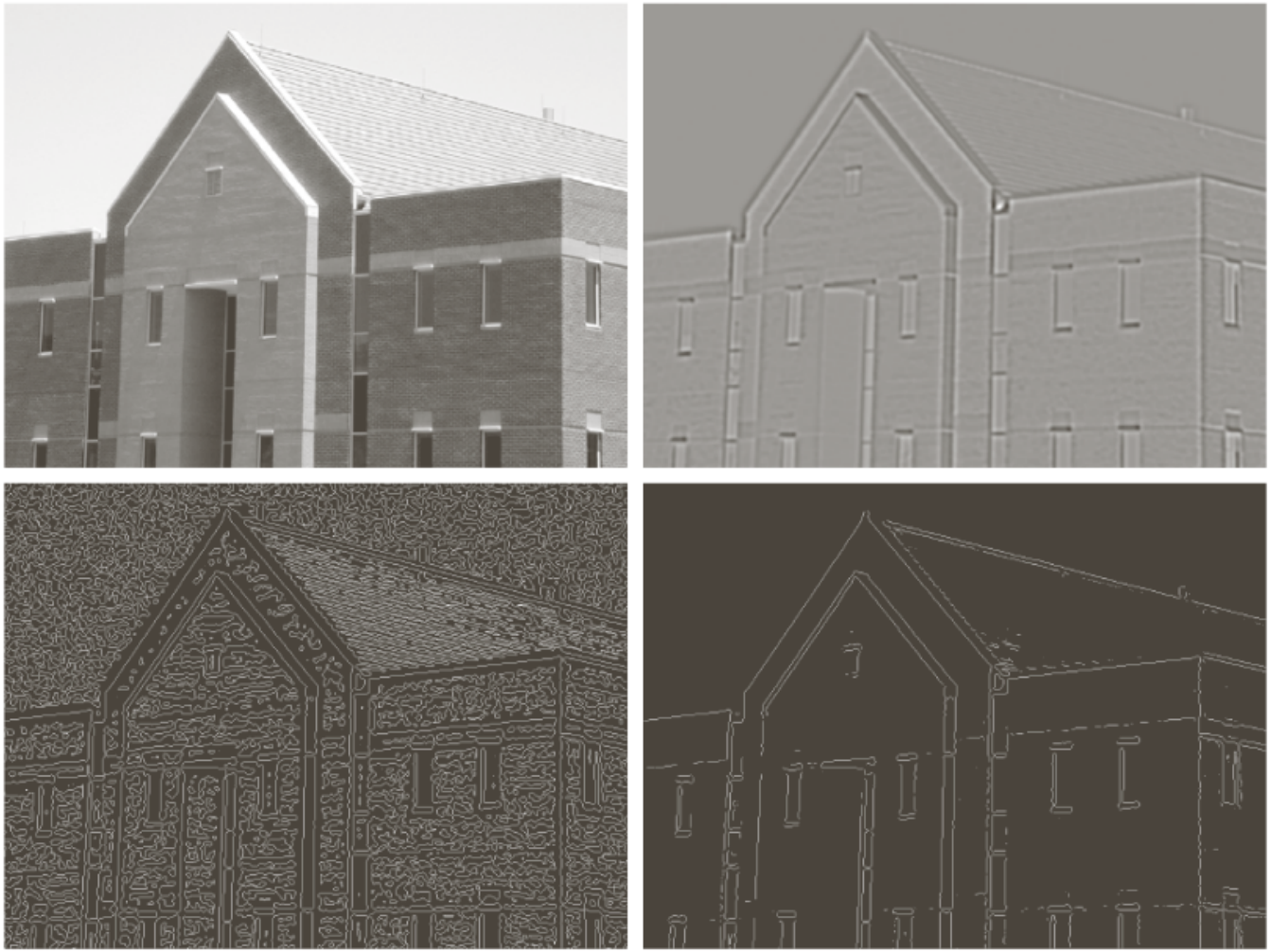




Velg kjerne- og filterstørrelse med omhu! Angis av standardavviket til Gauss-funksjonen, som gir bredden av LoG-kjernen og antyder størrelsen på LoG filteret

Eksempel: LoG-kantdeteksjon

Finne fremtredende kanter:



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

Robust kantdeteksjon

Vanligvis tre steg i robust kantdetektor:

1. **Støy-reduksjon**: Forsøker å fjerne så mye støy som mulig uten å glatte ut kantene for mye (lavpassfiltrering)
2. **Kant-filtrering:** Finner kantene (høypassfiltrering)
3. **Kant-lokalisering:** Etterbehandler resultatet fra kant-filtreringen for å finne eksakt kant-posisjoner (respons på kant bør helst være én piksel tykk og lokalisert der kanten faktisk befinner seg i innbildet)

Hva kjennetegner en god kantdetektor?

- Finner alle og bare de relevante kantene
- Posisjonen til detektert kant samsvarer med der kanten finnes i innbildet
- En kant gir én enkelt respons
- Robust for støy (kompromiss mellom støy-robusthet og kant-lokalisering)

Idéen til Canny

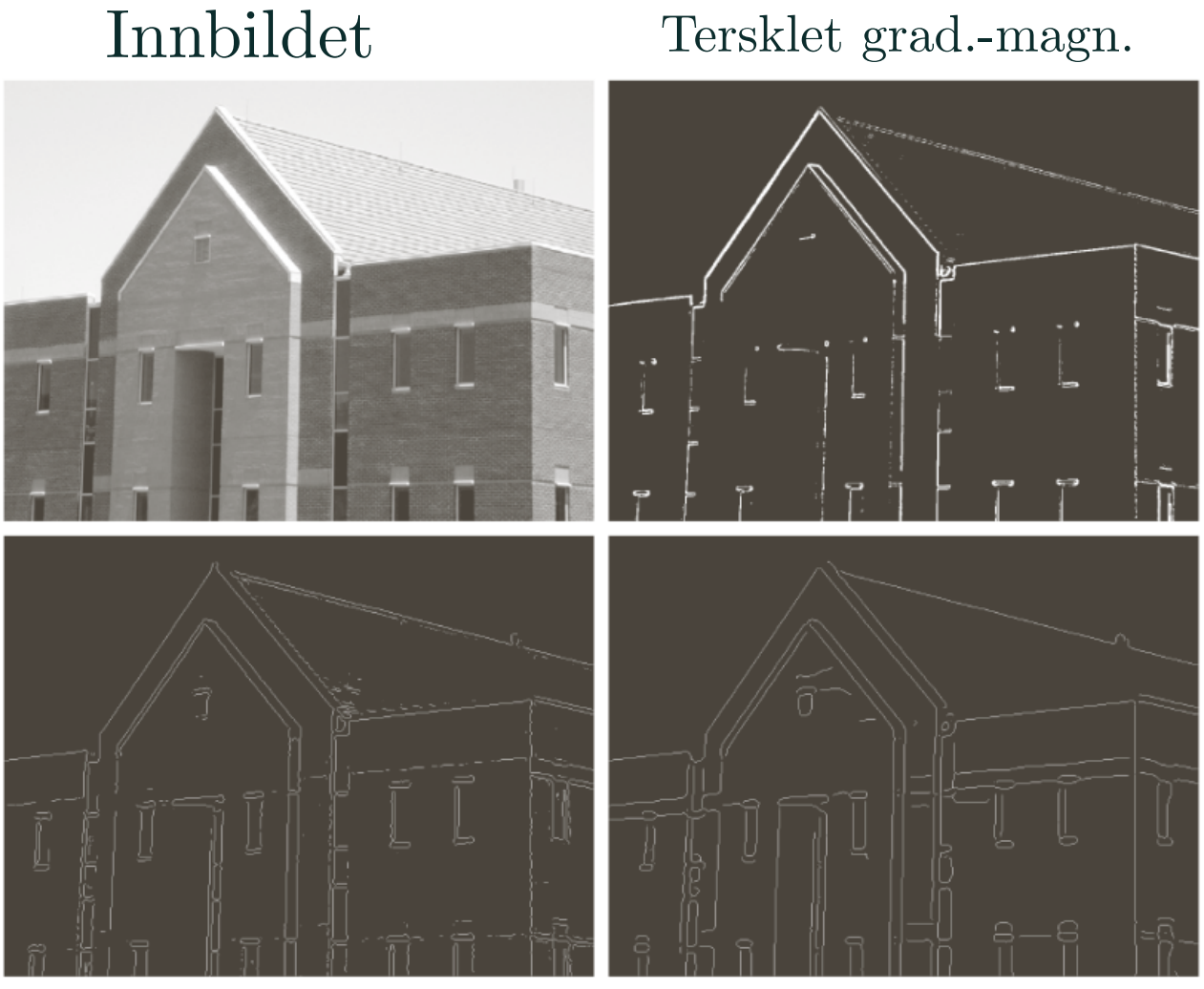
- Lag en kantdetektor som er optimal i forhold til følgende tre kriterier:
  - Best mulig deteksjon (finne alle kanter, og kun kanter)
  - God kant-lokalisering
  - Én enkelt respons
- Optimér ved bruk av et bilde med støy

Cannys algoritme

1. Lavpassfiltrér med 2D Gauss-filter med en bruker-bestemt σ
2. Finn gradient-magnituden og gradient-retningen
3. Tynning av gradient-magnitude ortogonalt på kant
   - Hvis en piksel i gradient-magnitude bildet har en 8-nabo i eller mot gradient-retningen med høyrere verdi, settes pikselverdien til 0
4. Hysterese-terskling med to bruker-bestemte terskler T_h og T_l:
   - 4.1 Merk alle piksler der $g(x, y) \geq T_h$
   - 4.2 For alle piksler der $g(x, y) \in [T_l, T_h)$:
     Hvis pikselen er (4 eller 8)-nabo til en merket piksel, så merkes denne pikselen også
   - 4.3 Gjenta trinnet over frem til det ikke merkes noen piksler

Eksempel: Kantdeteksjon 1

Finn fremtredende kanter:



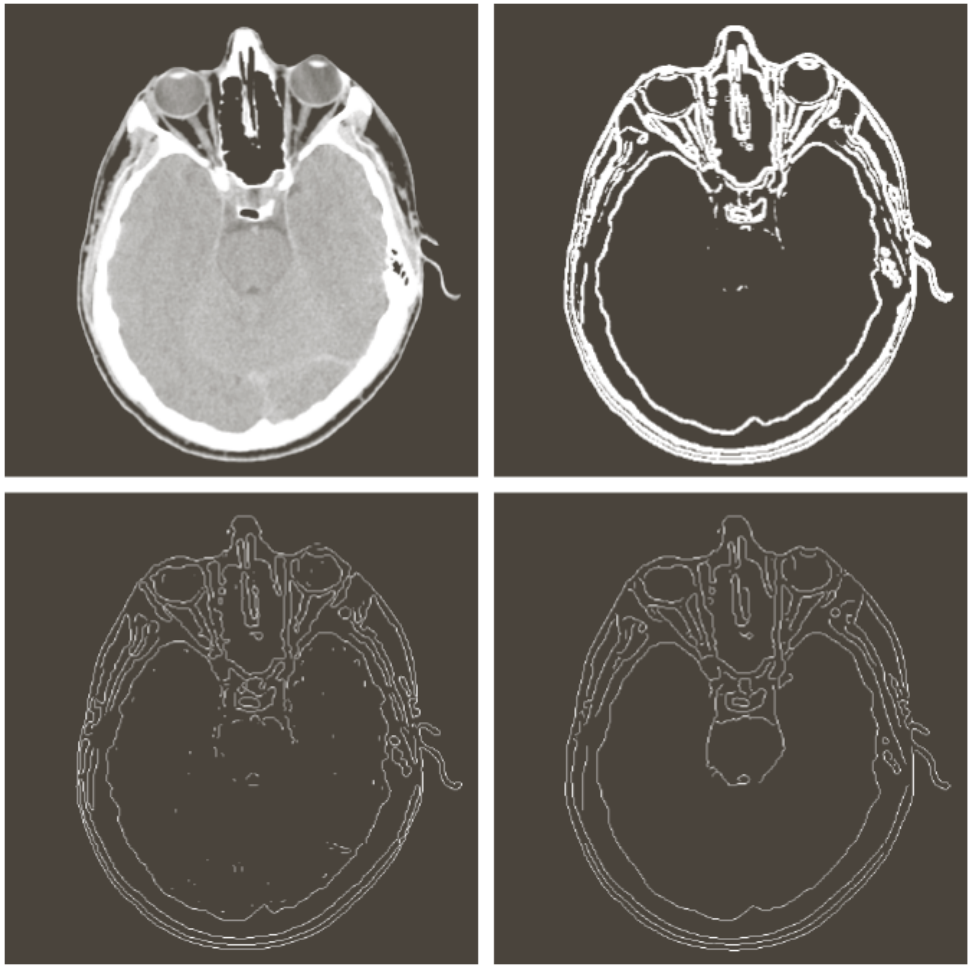
a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm.
Note the significant improvement of the Canny image compared to the other two.

Eksempel: Kantdeteksjon 2

Innbildet

Tersklet grad.-magn.



LoG-kantdetektor

Cannys kantdetektor

Oppsummering

- Noen kant-deteksjons operatorer
- Gradient-operatoren glatter i den ene retningen og gjør kantdeteksjon i den ortogonale retninger
- Gradient-operatorer gir både kant-styrke og -retning
- Laplace - operatorer gir presis lokalisering av kanten, men forsterker støy
- LoG - operatoren er en mer robust versjon av Laplace som inkluderer Gauss-glatting (kjernens og filterets størrelse påvirker resultatet!)
- Cannys kantdetektor gir et godt kompromiss mellom støyreduksjon og kantlokalisering