INF2310 – Digital bildebehandling

FORELESNING 8

REPETISJON: FILTRERING I BILDEDOMENET

Andreas Kleppe

Filtrering og konvolusjon
Lavpassfiltrering og kant-bevaring
Høypassfiltrering: Bildeforbedring og kantdeteksjon

Filtrering i bildedomenet

- Anvendelsen av en **operator** som beregner ut-bildets verdi i hvert piksel *(x,y)* ved bruk av inn-bildets piksler i et **naboskap** rundt *(x,y)*.

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

g(x,y) angir ut-bildets verdi i *(x,y)* og *f(x,y)* angir inn-bildets piksler i et naboskap rundt *(x,y)*

Ut-bilde | Operator | Inn-bilde

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

Et inn-bilde *f*

Filtrering

(3x3-middelverdifilter, speilende indeksering og bevart bildestørrelse)

2,7	2,9	2,7	2,1
3,1	2,9	2,4	2,2
3,3	3,0	3,0	3,3
2,6	2,2	2,9	3,7

Ut-bildet *g*

2D-konvolusjon

- Konvolusjon av et filter *h* og et bilde *f*:

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

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

  evaluert for alle *(x,y)* slik at hver verdi av *h* overlapper hver verdi av *f*.

For å forenkle notasjonen, antar disse formlene at:

- *h* har odde lengder, $m = 2a+1$ og $n = 2b+1$.
- Senterpikselet er naboskapets origo.

Konvolusjon krever ikke disse antagelsene.

- Responsen i *(x,y)* er en **veiet sum av inn-bildets verdier**.
  - Konvolusjonsfilteret spesifiserer vektene.
  - *h* kan spesifiseres som en matrise!

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3-middelverdifilter

2D-konvolusjons-eksempel

Oppgave: Konvolver følgende filter og bilde:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3-middelverdifilter

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

Inn-bilde *f*

2D-konvolusjons-eksempel

Steg 1: Roter filteret 180 grader.

Ikke nødvendig her ettersom filteret er symmetrisk.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3-middelverdifilteret
er symmetrisk

2D-konvolusjons-eksempel

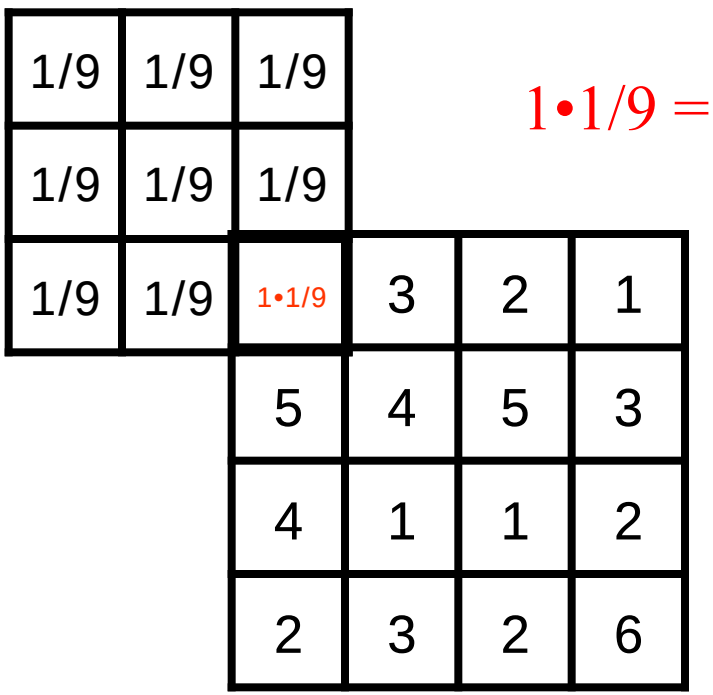
Steg 2: Legg det roterte filteret over først posisjon der filteret og bildet overlapper.

1/9	1/9	1/9			
1/9	1/9	1/9			
1/9	1/9	1•1/9	3	2	1
		5	4	5	3
		4	1	1	2
		2	3	2	6

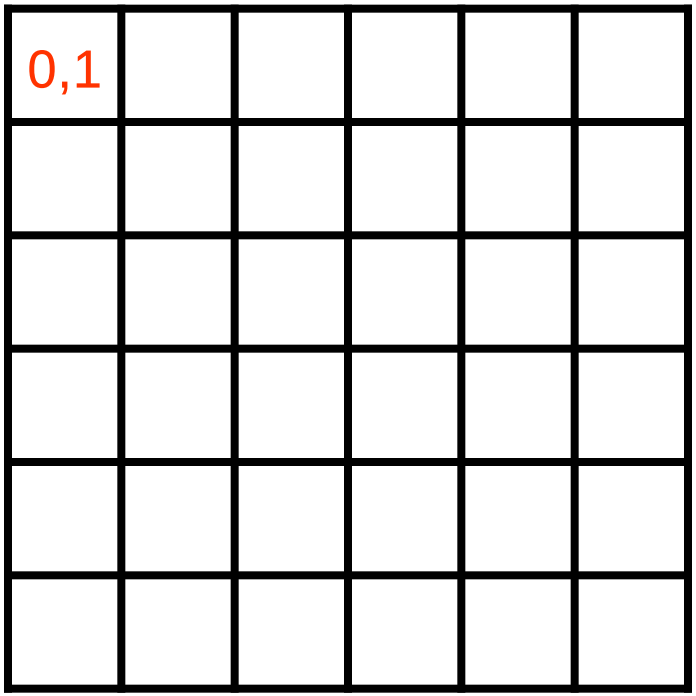
Inn-bilde *f*

2D-konvolusjons-eksempel

Steg 3: Multipliser filterets vekter med verdiene av de overlappende pikslene i bildet.
Responsen er summen av produktene.



$1 \bullet 1/9 = 1/9 \approx 0{,}1$



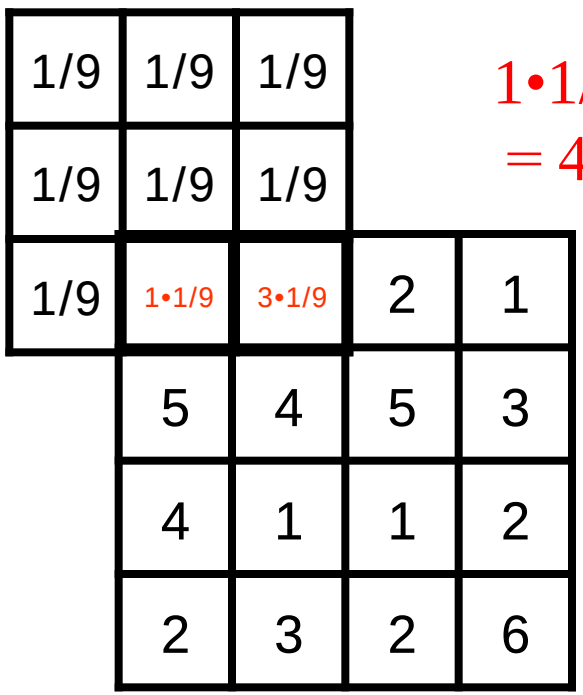
Inn-bilde *f*

Foreløpig ut-bildet *g*

2D-konvolusjons-eksempel

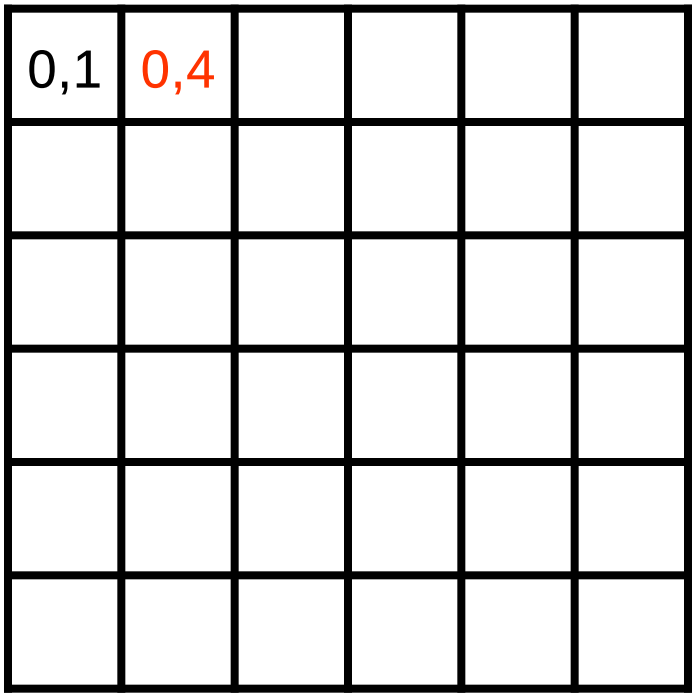
Steg 4: Gjenta 3 for neste overlapp. Ikke flere: ferdig!

Steg 3: Multipliser filterets vekter med verdiene av de overlappende pikslene i bildet og summer.



Inn-bilde *f*

$1 \bullet 1/9 + 3 \bullet 1/9 = 4/9 \approx 0{,}4$



Foreløpig ut-bildet *g*

2D-konvolusjons-eksempel

... fjorten steg 3 senere:

Steg 3: Multipliser filterets vekter med verdiene av de overlappende pikslene i bildet og summer.

$3 \cdot 1/9 + 2 \cdot 1/9 + 1 \cdot 1/9 + 4 \cdot 1/9 + 5 \cdot 1/9 + 3 \cdot 1/9 + 1 \cdot 1/9 + 1 \cdot 1/9 + 2 \cdot 1/9 = 22/9 \approx 2,4$

1	3•1/9	2•1/9	1•1/9
5	4•1/9	5•1/9	3•1/9
4	1•1/9	1•1/9	2•1/9
2	3	2	6

Inn-bilde *f*

0,1	0,4	0,7	0,7	0,3	0,1
0,7	1,4	2,2	2,0	1,2	0,4
1,1	2,0	2,9	2,4		

Foreløpig ut-bildet *g*

2D-konvolusjons-eksempel

... og etter tjue steg 3 til:

Steg 4: Gjenta 3 for neste overlapp. Ikke flere: **ferdig**!

Løsningen er:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3-middelverdifilter

$*$

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

Inn-bilde f

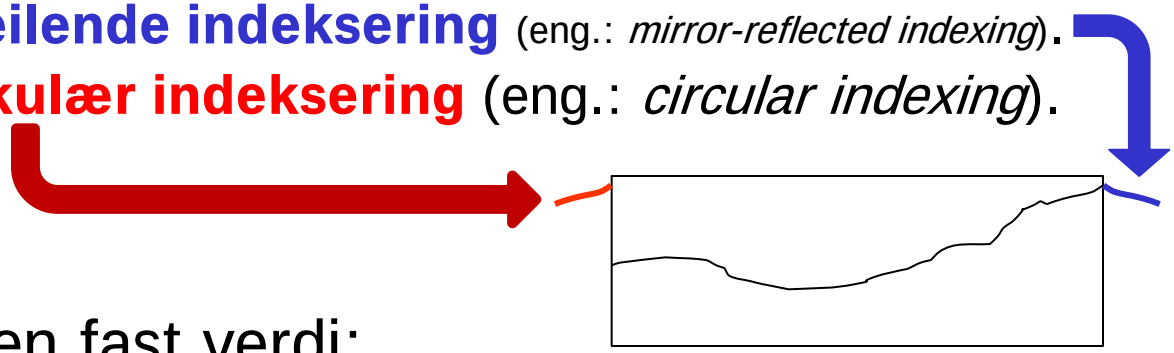
$=$

0,1	0,4	0,7	0,7	0,3	0,1
0,7	1,4	2,2	2,0	1,2	0,4
1,1	2,0	2,9	2,4	1,6	0,7
1,2	2,1	3,0	3,0	2,1	1,2
0,7	1,1	1,4	1,7	1,2	0,9
0,2	0,6	0,8	1,2	0,9	0,7

Ut-bildet g

Hva gjør vi langs bilderanden?

- Utvid inn-bildet:
  - **VANLIG**: Med 0-ere (nullutvidelse, eng.: *zero padding*).
  - Med en annen fast verdi.
  - Med nærmeste pikselverdi (eng.: *replicate*).
  - Ved bruk av **speilende indeksering** (eng.: *mirror-reflected indexing*).
  - Ved bruk av **sirkulær indeksering** (eng.: *circular indexing*).



- Sett ut-bildet til en fast verdi:
  - F.eks. g(x,y) = 0 eller g(x,y) = f(x,y).

- Ignorer posisjonene uten overlapp.
  - Identisk resultat som nullutvidelse for konvolusjonsfiltre.

Hvor stort skal ut-bildet være?

- Trunkér ut-bildet
  - Bare behold piksler der hele filteret er innenfor inn-bildet.
- Behold inn-bildets størrelse
  - Bare behold piksler der filterorigo er innenfor inn-bildet.
  - Vanlig når man filtrerer et bilde.
  - Langs randen må vi gjøre en antagelse, se foilen to før.
- Utvid inn-bildets størrelse
  - Behold alle piksler der filteret og «inn-bildet» har overlapp.
  - Svært uvanlig utenom for konvolusjon av to filtre.
  - Langs randen må vi gjøre en antagelse, se foilen to før.
- Merk: Dette gjelder all filtrering, ikke bare konvolusjon!

Lavpassfiltre

- Slipper gjennom lave frekvenser
  og demper eller fjerner høye frekvenser.
  - Lav frekvenser = trege variasjoner, store trender.
  - Høye frekvenser = skarpe kanter, støy, detaljer.
  - ... mye mer om frekvens i Fourier-forelesningene.
- Effekt: Glatting/utsmøring/«blurring» av bildet.
- Typiske mål: Fjerne støy, finne større objekter.
- Utfordring: Bevare kanter.

Middelverdifilter (lavpass)

- Beregner middelverdien i naboskapet.
  - Alle vektene er like.
  - Vektene summerer seg til 1.
    - Gjør at den lokale gjennomsnittsverdien bevares.
- Størrelsen på filteret avgjør graden av glatting.
  - Stort filter: mye glatting (utsmørt bilde).
  - Lite filter: lite glatting, men kanter bevares bedre.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{49}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$$

Separable filtre

- Et filter kalles separabelt hvis **filtreringen kan utføres som to sekvensielle 1D-filtreringer**.
- Fordel: **Raskere filtrering**.
- Geometrisk form: Rektangel (inkludert kvadrat).
- Middelverdifiltre er separable: For 5x5-naboskap:

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

- Beregne én respons for $n \times n$-konvolusjonsfiltre:
  - 2D-konvolusjon: n^2 multiplikasjoner og n^2-1 addisjoner.
  - To 1D-konvolusjoner: $2n$ multiplikasjoner og $2(n-1)$ addisjoner.

Approksimasjon av 3x3-Gauss-filter

$$G_{3\times3} = \frac{1}{16}\begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix}$$

$$= \frac{1}{16}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

$$= \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

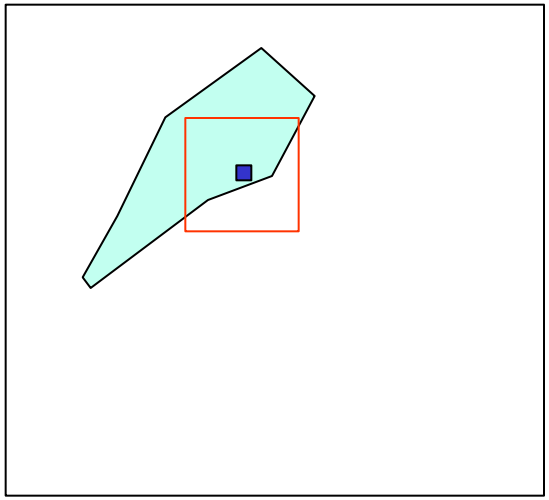
$$= \frac{1}{16}\begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

Tilsvarer en **geometrisk vekting**:

- Vekten til et piksel er en funksjon av avstanden til *(x,y)*.
- Nære piksler er mer relevante og gis derfor større vekt.

Kant-bevarende støyfiltrering

- Ofte lavpassfiltrerer vi for å **fjerne støy**, men ønsker samtidig å **bevare kanter**.
- Det finnes et utall av «kantbevarende» filtre.
- Men det er et system:
  - Tenker at vi har flere piksel-populasjoner i naboskapet rundt *(x,y)*, f.eks. to:
  - Sub-optimalt å bruke all pikslene.



- Vi kan sortere pikslene:
  - Radiometrisk (etter pikselverdi)
  - Både geometrisk (etter pikselposisjon) og radiometrisk

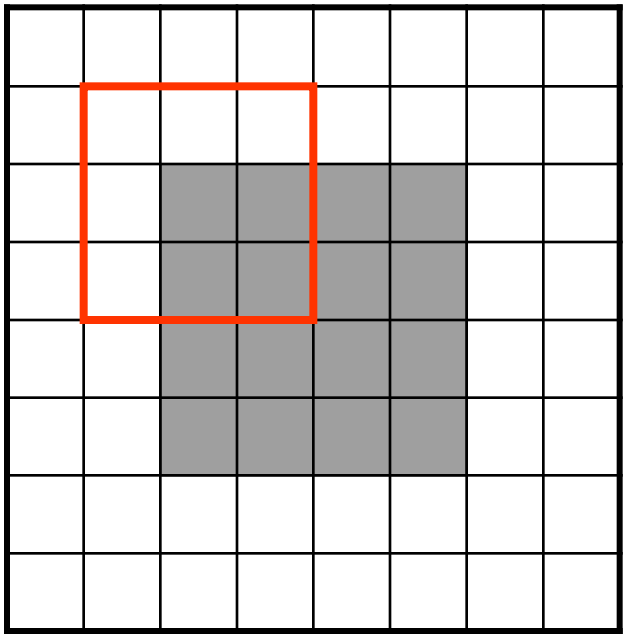
Middelverdi eller median?

Inn-bildet med tydelig salt-og-pepper-støy

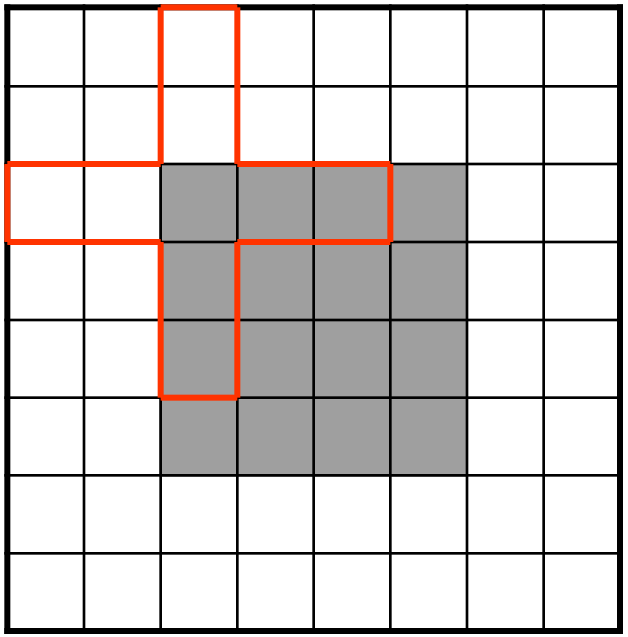
Etter middelverdifiltrering

Etter medianfiltrering

Medianfiltrering og hjørner



Med kvadratisk naboskap
avrundes hjørnene



Med pluss-formet naboskap
bevares hjørnene

Kant-bevarende støyfiltrering

- Trimmet middelverdifilter:
  - Alpha-trimmet middelverdifilter (radiometrisk sortering):
    g(x,y) = middelverdien av de *mn-d* midterste verdiene (etter sortering) i *m*x*n*-naboskapet rundt *(x,y)*.
  - *K* Nearest Neighbour-filter (radiometrisk sortering):
    g(x,y) = middelverdien av de *K* pikslene i naboskapet rundt *(x,y)* som ligner mest på *(x,y)* i pikselverdi.
  - *K* Nearest Connected Neighbour-filter (også geometrisk):
    K Nearest Neighbour-filter med uendelig stort naboskap og der de valgte pikslene er tilkoblet ut-posisjonen *(x,y)*.
  - Max-homogenitet-filter (også geometrisk):
    g(x,y) = middelverdien av det mest homogene sub-naboskapet.
  - Symmetrisk nærmeste nabo-filter (også geometrisk):
    g(x,y) = middelverdien av de mest lignende fra hvert symmetrisk piksel-par rundt *(x,y)*.
- MinimalMeanSquareError-filter: Nær middelverdien når lokal varians tilsvarer anslått støyvarians (en parameter), ellers nær uendret.

Høypassfiltre

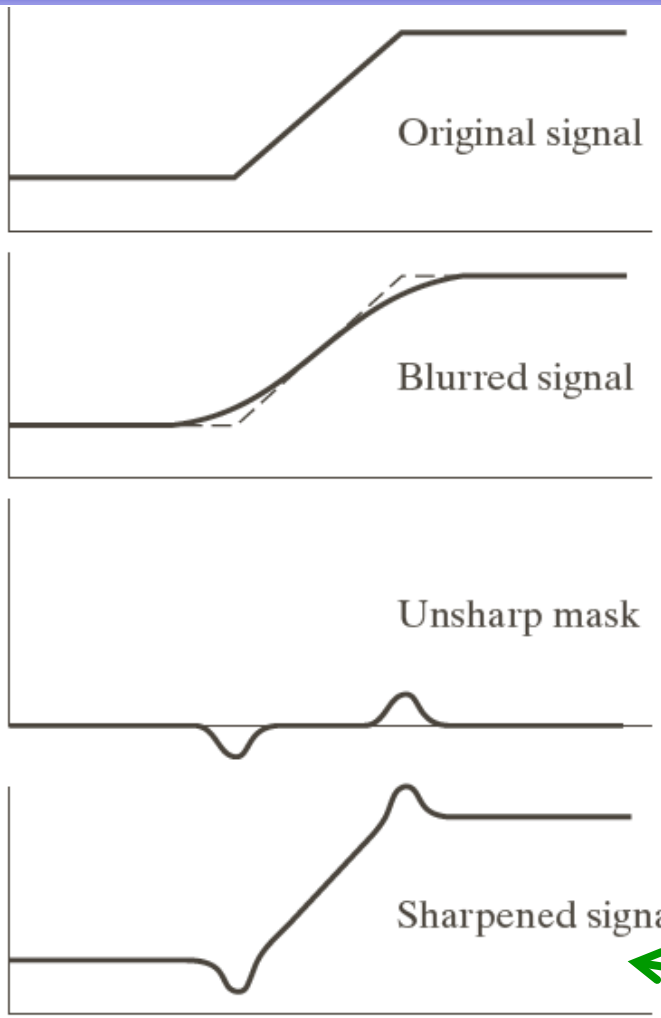
- Slipper gjennom høye frekvenser,
  og demper eller fjerner lave frekvenser.
  - Typisk fjernes den aller laveste frekvensen helt,
    d.v.s. at homogene områder får ut-verdi 0.

- Effekt:
  - Demper langsomme variasjoner, f.eks. bakgrunn.
  - Fremhever skarpe kanter, linjer og detaljer.

- Typiske mål: «Forbedre» skarpheten, detektere kanter.

- Q: Hva skjer med støy?

Unsharp masking og highboost-filtrering



G&W fig. 3.39

- Gitt et bilde (original):
  (til venstre: et 1D-bilde av en rampe)

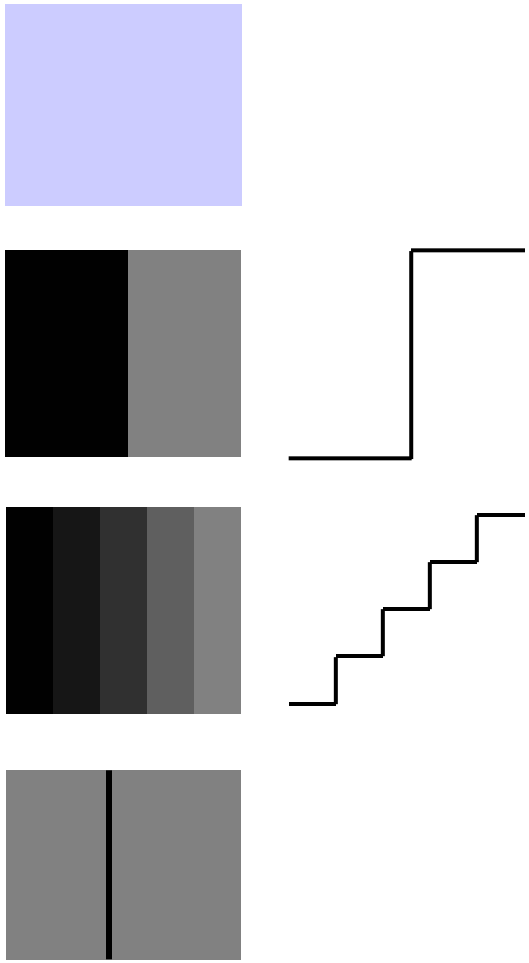
1. Lavpassfiltrer.
   (til høyre er originalen stiplet)

2. Beregn differansen:
   original – filtrering

3. Resultatet er:
   original + k • differansen
   - *k* er en positiv konstant.
   - **Unsharp masking**: *k* = 1 (brukt til høyre)
   - **Highboost-filtrering**: *k* > 1

Intensitets-flater, -kanter og -linjer

- Homogen flate: Et område der alle pikselverdiene er like.
- Kant: Overgangen mellom to områder med forskjellig middelverdi.
  - Steg-kant: En-piksels overgang.
  - Rampe: Fler-piksels overgang med konstant intensitetsendring (d.v.s. konstant gradient).
- «Kant» brukes også om skillepunktet mellom de to områdene.
  - Forskjellige måter å modellere hvor skiller er.
  - For steg-kanter:
    - Et alternativ er midt mellom nabo-piksler som tilhører forskjellig områder.
    - I segmentering ønsker man typisk å finne første piksel på siden som tilhører objektet.
- Merk at en linje består av to kanter.
  - Hver kan f.eks. være steg-kant eller rampe.
- Idealstrukturer er nyttig for modellering, men i praksis finner vi oftest strukturer som bare ligner.



Digital derivasjon

- En kant kjennetegnes ved **endring i intensitetsverdi**.
  - Siden en intensitetskant er overgangen mellom to områder med forskjellig middelverdi, så må intensiteten endres i kanten.

- Den **deriverte** av en funksjon *f(x)* er definert som:

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

  og angir stigningstallet til *f* i punktet *x*,
  så */f(x)/* angir hvor mye *f* endrer seg i punktet *x*.

- Den deriverte er **ikke definert** for diskrete funksjoner, men vi kan **tilnærme** den ved å la $h \geq 1$ i definisjonen.
  - Tilnærme v.b.a. **differanser mellom nærliggende piksler**.

Derivasjon av bilder

- Et digitalt bilde er en to-variabel, diskret funksjon.

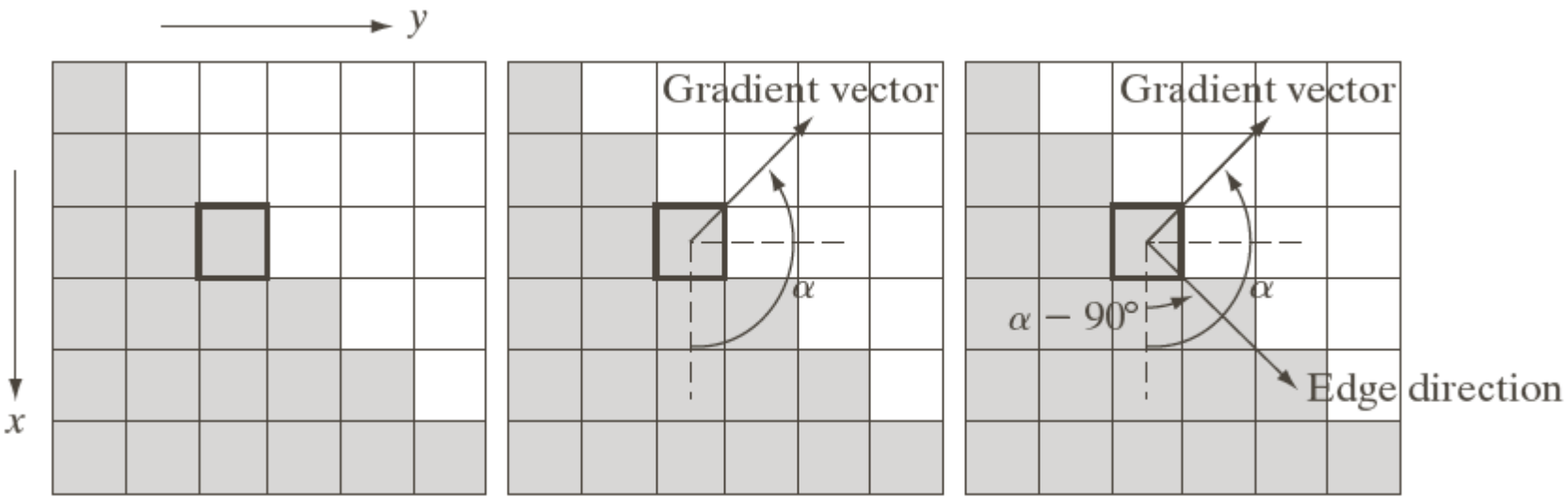
- En kontinuerlig funksjon *f(x,y)* kan deriveres m.h.p. *x* og *y*.
  - Kalles å partiell-derivere m.h.p. *x* og *y*.
  - Betegnes henholdsvis $\partial f(x,y)/\partial x$ og $\partial f(x,y)/\partial y$

- Vektoren av de to partiell-deriverte kalles gradienten og betegnes ∇f:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Gradient $\perp$ Kant

- **Gradienten** peker i retningen der funksjonen **øker 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.

Gradient-operatorer

- Asymmetrisk 1D-operator:

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

  - Også kalt «pixel difference»-operatoren.

- Symmetrisk 1D-operator:

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

  - Også kalt «separated pixel difference»-operatoren.

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

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

NB: Vi angir konvolusjonsfiltre i den tanke at de skal brukes til konvolusjon.

G&W angir filtermasker som skal brukes til korrelasjon.

Filtrene vil derfor avvike med en 180 graders rotasjon.

Gradient-operatorer

- Prewitt-operatoren:

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

- Sobel-operatoren:

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

- Frei-Chen-operatoren:

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

NB: Vi angir konvolusjonsfiltre i den tanke at de skal brukes til konvolusjon.

G&W angir filtermasker som skal brukes til korrelasjon.

Filtrene vil derfor avvike med en 180 graders rotasjon.

Større gradient-operatorer

- De symmetriske gradient-operatorene kan gjøres mer støy-robuste ved å bygge inn mer lavpassfiltrering.

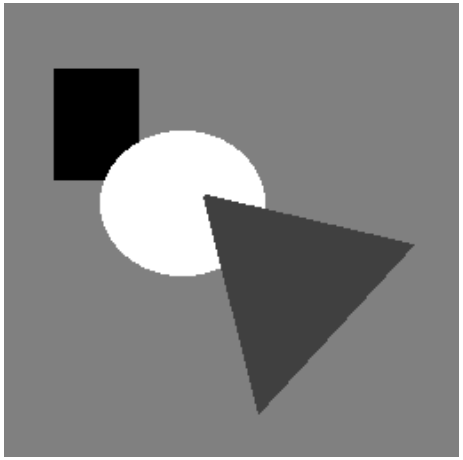
- Eksempel: Følgende 5×5-Sobel-operator:

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

er resultatet av konvolusjonene:

$$h_x(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad h_y(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

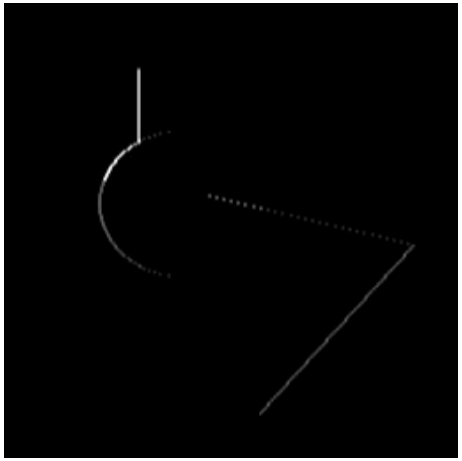
Eksempel: Gradient-beregning med Sobel-operatoren



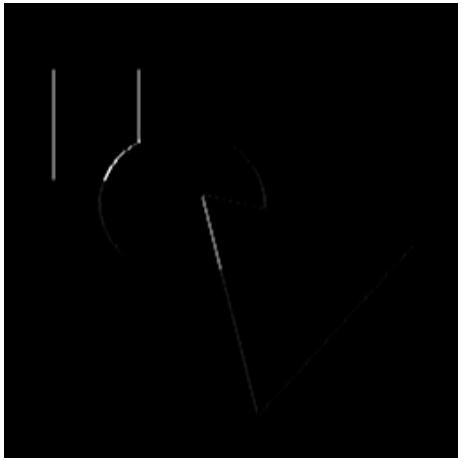
Inn-bilde f

$g_x = f*h_x$

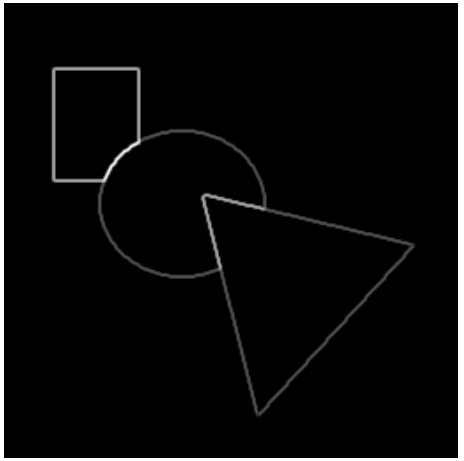
g_x^2



$g_y = f*h_y$



g_y^2

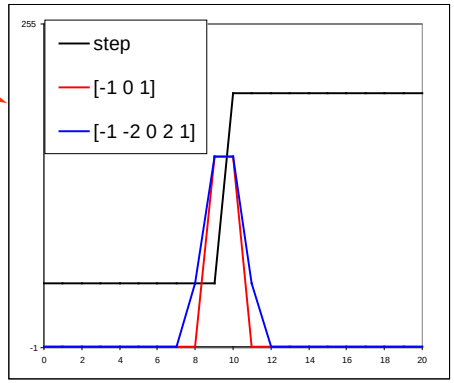


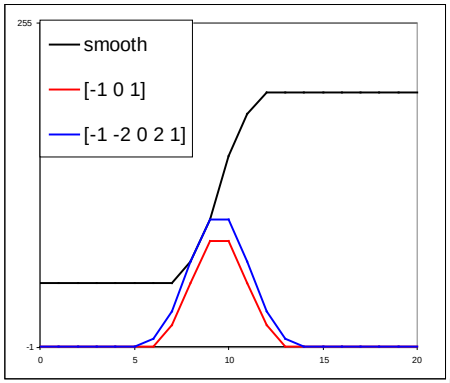
$(g_x^2+g_y^2)^{1/2}$

Merk:
Hvert bilde er skalert ved å dele på sitt maksimum.
De negative verdiene i g_x og g_y er satt til 0.

Gradient til kant-deteksjon

- Gradient-magnituden har «bred respons», men vi ønsker eksakt, tynn kant.
- For en steg-kant:
  - Bredden på responsen er avhengig av størrelsen på filteret.
- For en bred kant (glattet med [1 2 3 2 1]):
  - Bredden på responsen er avhengig av bredden på kanten.
- **Maksimumet er likt og fornuftig lokalisert!**
  - Bruke den andrederiverte til å finne maksimumene?



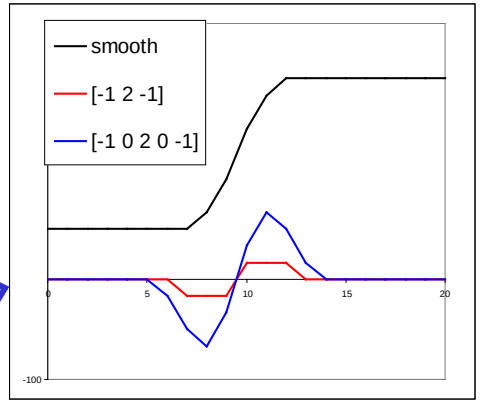


Laplace-operatoren

- Laplace-operatoren er gitt ved:

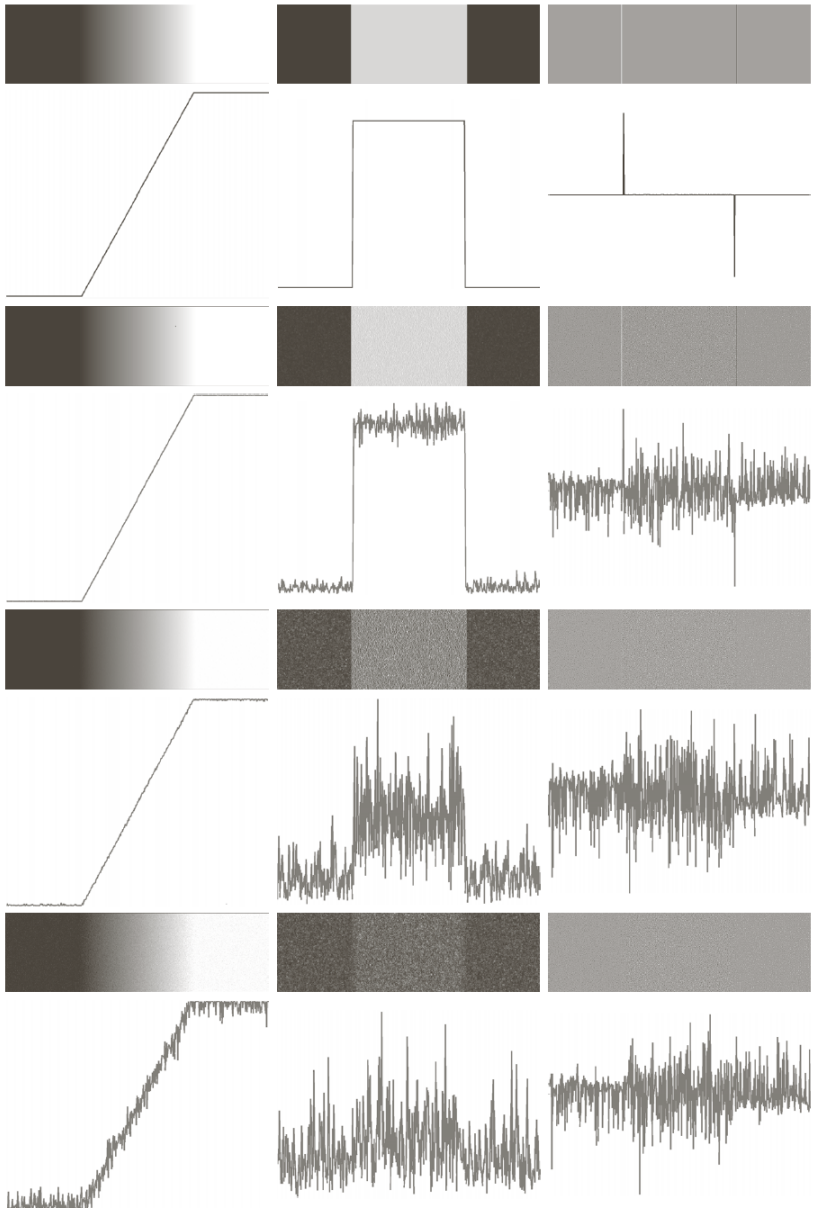
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  - Den endrer fortegn der f et vendepunkt.
  - $\nabla^2 f = 0$ markerer kant-posisjon.
  - $|\nabla^2 f|$ har to ekstremverdier per kant; på starten og på slutten av kanten.
    - Derfor brukte vi den tidligere til å forbedre bildeskarpheten!



- Kantens eksakte posisjon er **nullgjennomgangen**.
- Dette gir tynne kanter.
- Vi finner bare kant-posisjoner, ikke kant-retninger.

Også lavpassfiltrere?



Signal Derivert Laplace

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.

To måter å lage LoG-operatorer

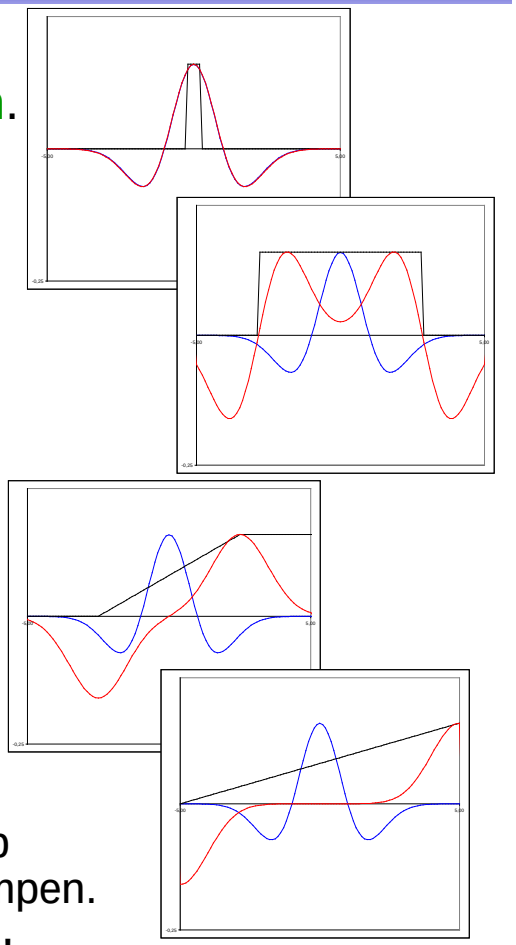
- Ofte lages og implementeres en LoG-operator som konvolusjonen av en Laplace-operator og et Gauss-filter.

- Ofte defineres en **LoG-operator** 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 kaller LoG-operatorer.

Kantdeteksjon ved LoG-nullgjennomganger

- Tommelfingerregel for strukturer:
  LoG-kjernen må være smalere enn strukturen.
  - Strukturen er mindre enn halvparten av LoG-kjernen
    => Nullgjennomgangene er utenfor kantskillene
  - Strukturen er større enn halvparten av LoG-filteret
    => Nullgjennomgangene er nøyaktig kantskillene
  - Et sted i mellom: Avhenger av
    diskretiseringen og tilnærmingen av LoG-filteret.
- Tommelfingerregel for ramper:
  LoG-filteret må være større enn rampen.
  - Rampen er bredere enn LoG-filteret,
    => Ingen nullgjennomgang, bare et null-platå.
  - Ellers: Nullgjennomgang midt på rampen
    (kan få én 0-respons akkurat på midten),
    altså en fornuftig definisjon av kantskillet til rampen.
    - P.g.a. støy krever ofte at nullpasseringen er skarp
      => LoG-filteret må være betydelig større enn rampen.
- => **Velg kjerne- og filterstørrelsen med omhu**!
  - Angis først og fremst av standardavviket til Gauss-funksjonen,
    som gir bredden av LoG-kjernen og antyder størrelsen av LoG-filteret.



Ideen til Canny

- Lag en kantdetektor som er optimal i forhold til følgende tre kriterier:
  - Best mulig deteksjon (alle kanter og bare kanter)
  - God kant-lokalisering
  - Én enkelt respons

- Optimer ved bruk av et bilde med støy.

- Resultat: Følgende enkle algoritme oppnår nesten optimumet:

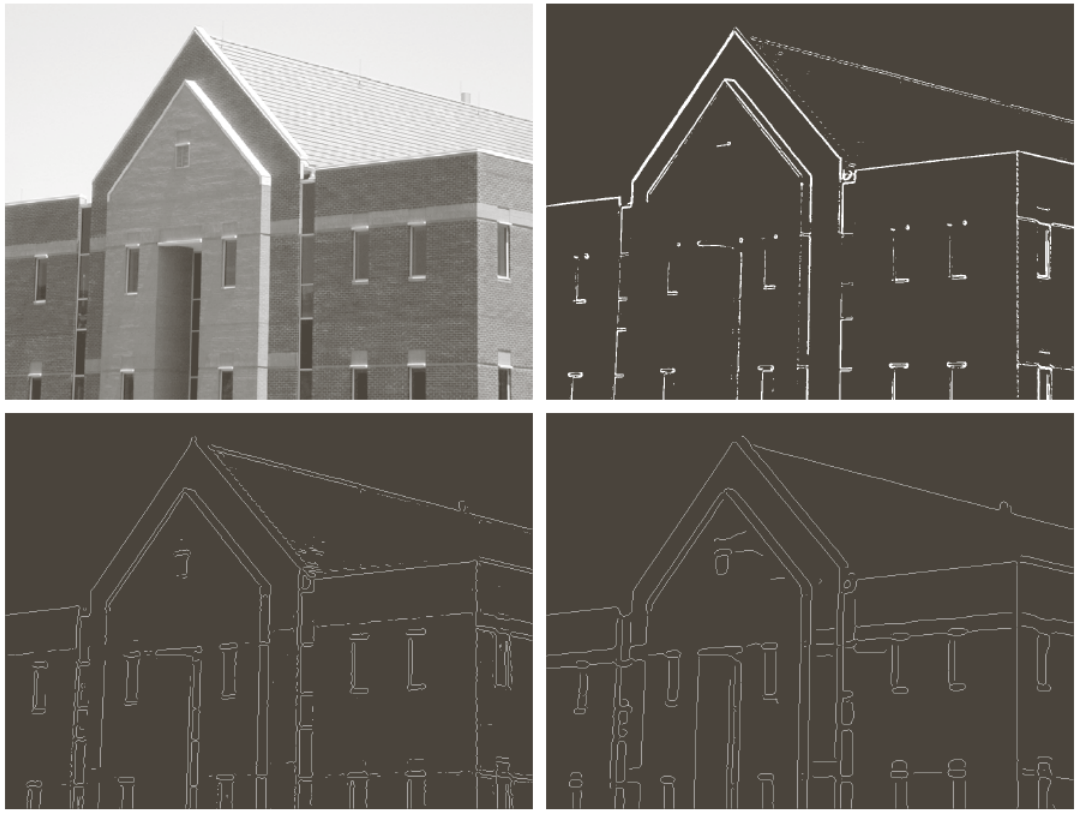
Cannys algoritme

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

Eksempel: Kantdeteksjon

- **Oppgave**: Finn fremtredende kanter.

Inn-bilde | Tersklet glattet grad.mag.



LoG-kantdetektor | Cannys kantdetektor

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.