

---

# INF2310 – Digital bildebehandling

## **FORELESNING 6**

### **FILTRERING I BILDEDOMENET – I**

Andreas Kleppe

Naboskaps-operasjoner  
Konvolusjon og korrelasjon  
Lavpassfiltrering og kant-bevaring

G&W: 2.6.2, 3.1, 3.4-3.5, deler av 5.3  
og «Matching by correlation» i 12.2

# Filtrering

---

- Generelt verktøy for behandling av digitale bilder.
- Av de mest brukte operasjonene i bildebehandling.
- Brukes ofte som et ledd i bl.a.:
  - Bildeforbedring
  - Bildeanalyse (spesielt i pre-prosesseringen)
  - Deteksjon av linjer og andre spesielle strukturer
- ... til bl.a. å:
  - Fjerne/ redusere støy
  - «Forbedre» skarpheten
  - Detektere kanter

# Konvolusjon

---

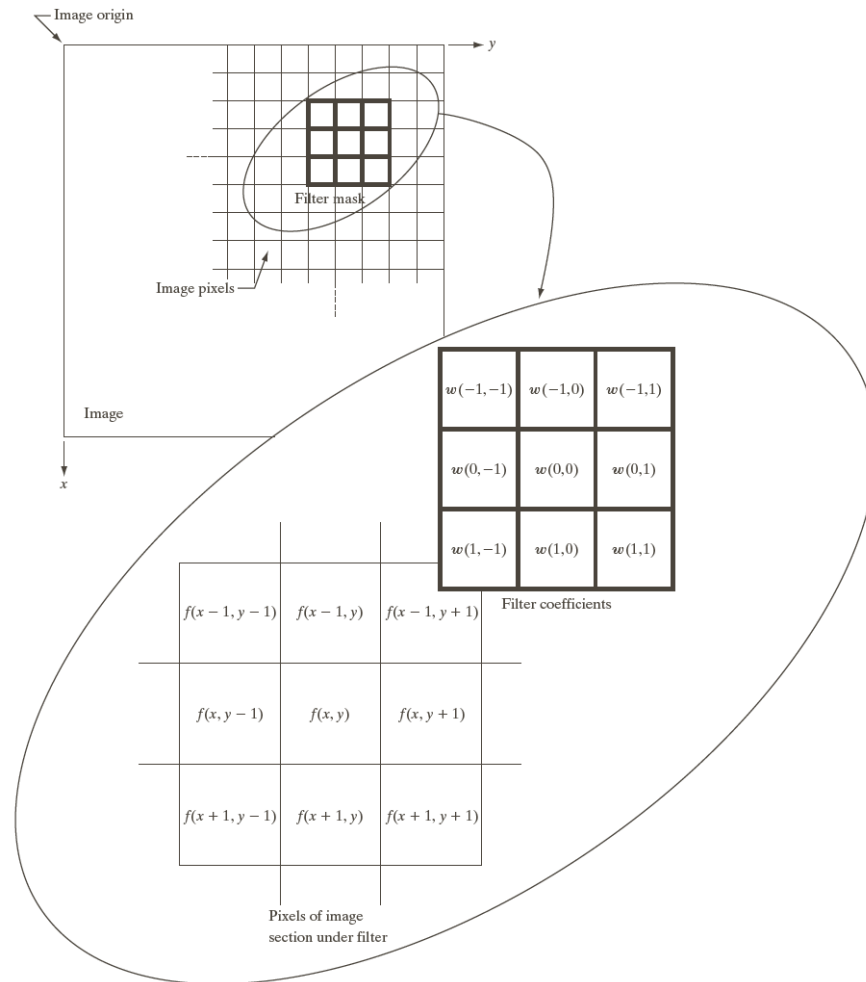
I konvolusjon er filteret definert av en matrise, f.eks.:

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

3x3-middelverdifilter

Matrisens størrelse og verdier bestemmer filtreringen.

# Konvolusjon



**FIGURE 3.28** The mechanics of linear spatial filtering using a  $3 \times 3$  filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

# Konvolusjonseksempel

---

**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  $f$

Anta at bildet er 0 utenfor det definerte 4x4-området.

# Konvolusjonseksempel

---

**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

# Konvolusjonseksempel

---

**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-bildet  $f$

# Konvolusjonseksempel

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

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

$$1 \cdot 1/9 = 1/9 \approx 0,1$$

Inn-bildet  $f$

0,1					

Foreløpig ut-bilde  $g$



# Konvolusjonseksempel

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

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

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

$$1 \cdot 1/9 + 3 \cdot 1/9 = 4/9 \approx 0,4$$

Inn-bildet  $f$

0,1	0,4				

Foreløpig ut-bilde  $g$

# Konvolusjonseksempel

... fjorten steg 3 senere:

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

1	$3 \cdot 1/9$	$2 \cdot 1/9$	$1 \cdot 1/9$
5	$4 \cdot 1/9$	$5 \cdot 1/9$	$3 \cdot 1/9$
4	$1 \cdot 1/9$	$1 \cdot 1/9$	$2 \cdot 1/9$
2	3	2	6

Inn-bildet  $f$

$$\begin{aligned} & 3 \cdot 1/9 + 2 \cdot 1/9 + 1 \cdot 1/9 + \\ & 4 \cdot 1/9 + 5 \cdot 1/9 + 3 \cdot 1/9 + \\ & 1 \cdot 1/9 + 1 \cdot 1/9 + 2 \cdot 1/9 \\ & = 22/9 \approx 2,4 \end{aligned}$$

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-bilde  $g$

# Konvolusjonseksempel

... 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-middelverdifilteret

\*

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

Inn-bildet  $f$

=

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  $g$

# Konvolusjon

- Konvolusjon av et filter  $h$  og et bilde  $f$  er:

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

For å forenkle notasjonen, antar disse formlene at:

- $h$  har odde lengder,  $m = 2a + 1$  og  $n = 2b + 1$ .
- Senterpikselen er naboskapetets origo. Konvolusjon krever ikke disse antagelsene.

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

- Responsen i  $(x, y)$  er en **veiet sum av inn-bildets pikselverdier**.
  - Konvolusjon er en lineær operasjon på inn-bildets verdier.
    - Ikke noe konstantledd.

# Beregne en konvolusjon

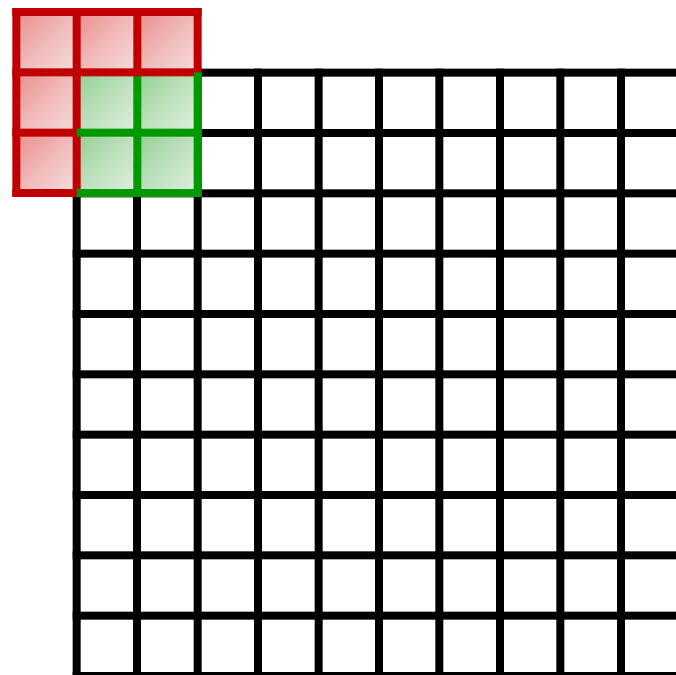
---

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

- For å regne ut responsen i posisjon  $(x, y)$ :
  1. Roter konvolusjonsfilteret 180 grader.
    - Unødvendig dersom filteret er symmetrisk.
  2. Legg det roterte filteret over bildet slik at origo overlapper posisjon  $(x, y)$  i bildet.
  3. Multipliser hver vekt i det roterte konvolusjonsfilteret med underliggende pikselverdi.
  4. Summen av produktene gir responsen,  $g(x, y)$ .
- For å regne ut responsen i alle posisjoner:
  - Flytt konvolusjonsfilteret over bildet og beregn responsen for hver posisjon med overlapp.

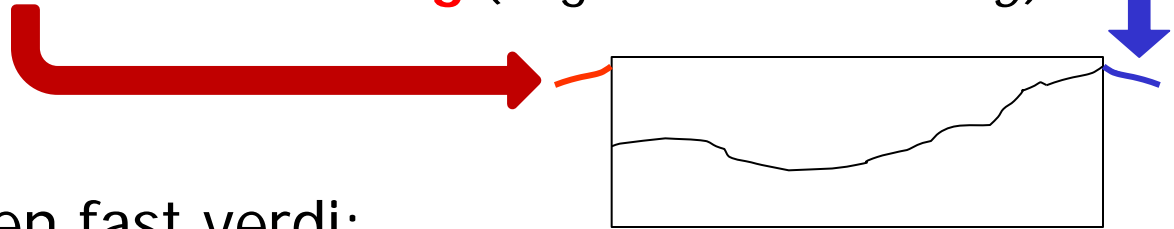
# Praktiske problemer

- Får ut-bildet samme kvantifisering som inn-bildet?
- Kan vi direkte endre inn-bildet eller må vi mellomlagre resultatbildet?
- Hva gjør vi langs bilderanden?
  - Anta at bildet er  $M \times N$  piksler.
  - Anta at filteret er  $m \times n$ .
    - (og at  $m$  og  $n$  er odde)
  - Uberørt av **bilderandproblemet**:  
 $(M-m+1) \times (N-n+1)$ 
    - 3x3-filter:  $(M-2) \times (N-2)$
    - 5x5-filter:  $(M-4) \times (N-4)$



# 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  $g(x,y) = f(x,y)$ .
- Ignorerer posisjonene uten overlapp.
  - Identisk resultat som nullutvidelse for konvolusjonsfiltre.

# Større ut-bilde enn inn-bilde?

---

- Konvolusjons-definisjonen sier at ut-bildet får verdi når filteret og inn-bildet overlapper i minst én piksel.
- Ut-bildet blir da større enn inn-bildet.
  - Dersom filteret er større enn 1x1.
- I praksis: Kun vanlig når man konvolverer to filtre.
  - Antar da at begge konvolusjonsfiltrene er 0 utenfor randen.
    - Når man bruker et konvolusjonsfilteret «antar» man indirekte alltid at det er 0 utenfor randen.
    - Når to filtre konvolveres bør begge bruke denne «antagelsen».



# 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.
- PS: De praktiske problemene vi har nå sett på gjelder **all filtrering**, ikke bare konvolusjon!

# 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!

Merk: Disse egenskapene gjelder generelt bare når man antar **nullutvidelse**.

# Korrelasjon

---

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

- Forskjell fra konvolusjon: **Pluss i stedet for minus.**
  - Det betyr at vi ikke skal rotere filteret!
- Legger korrelasjonsfilterets origo over  $(x, y)$  og multipliserer hver vekt med underliggende pikselverdi. Responsen i  $(x, y)$  er summen av disse produktene.
- Vi kan utføre korrelasjon som en konvolusjon ved å først roterer filteret 180 grader, og visa versa.

# Mønstergjenkjenning

---

- Korrelasjon kan brukes til å gjenkjenne mønster (eng. *template matching*).
  - Filteret er mønsteret/templatet.
    - Enten forhåndsdefinert, f.eks. «idealbilde» av mønsteret,
    - Eller definert som en utsnitt av bildet vi skal filtrere.
- Normaliser med summen av pikselverdiene innenfor filterets naboskap:

$$g'(x, y) = \frac{g(x, y)}{\sum_{s=-a}^a \sum_{t=-b}^b f(x+s, y+t)}$$

- Unngår høyere vektning av lyse piksler.

# Mønstergjennkjennings-eksempel

---

- **Oppgave:** Finn et bestemt objekt i et bilde.
- Templaten må ha samme størrelse og orientering som objektet i bildet (og omtrent samme gråtoner).



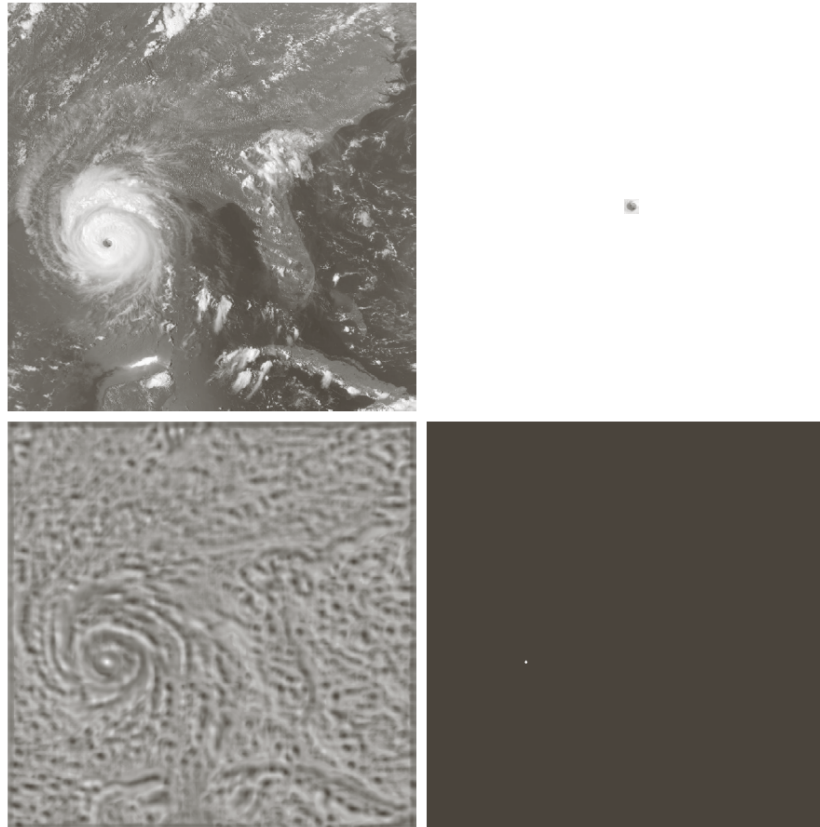
# Korrelasjonskoeffisient

- Mønstergjennkjennning kan bedre gjøres ved bruk av *korrelasjonskoeffisienten*:

$$\gamma(x, y) = \frac{\sum_{s=-at=-b}^a \sum_{t=-b}^b (h(s, t) - \bar{h})(f(x + s, y + t) - \bar{f}(x, y))}{\sqrt{\sum_{s=-at=-b}^a \sum_{t=-b}^b (h(s, t) - \bar{h})^2 \sum_{s=-at=-b}^a \sum_{t=-b}^b (f(x + s, y + t) - \bar{f}(x, y))^2}}$$

- Telleren er en middelve-di-normalisert korrelasjon.
    - Templatets er normalisert med sin (**globale**) middelve-di.
    - Bildet er normalisert med middelve-dien av bildeverdiene i pikslene der templatet og bildet overlapper (**lokal** middelve-di).
  - Nevneren normaliserer korrelasjonsresponsen slik at:
    - $\gamma(x, y)$  ikke påvirkes av den variasjonen rundt  $f(x, y)$ .
    - $\gamma(x, y)$  er alltid i intervallet  $[-1, 1]$ .
- Deler derfor på:  
• Bildet lokale standardavvik.  
• Templatets standardavvik.

# Korrelasjonskoeffisient-eksempel



a b  
c d

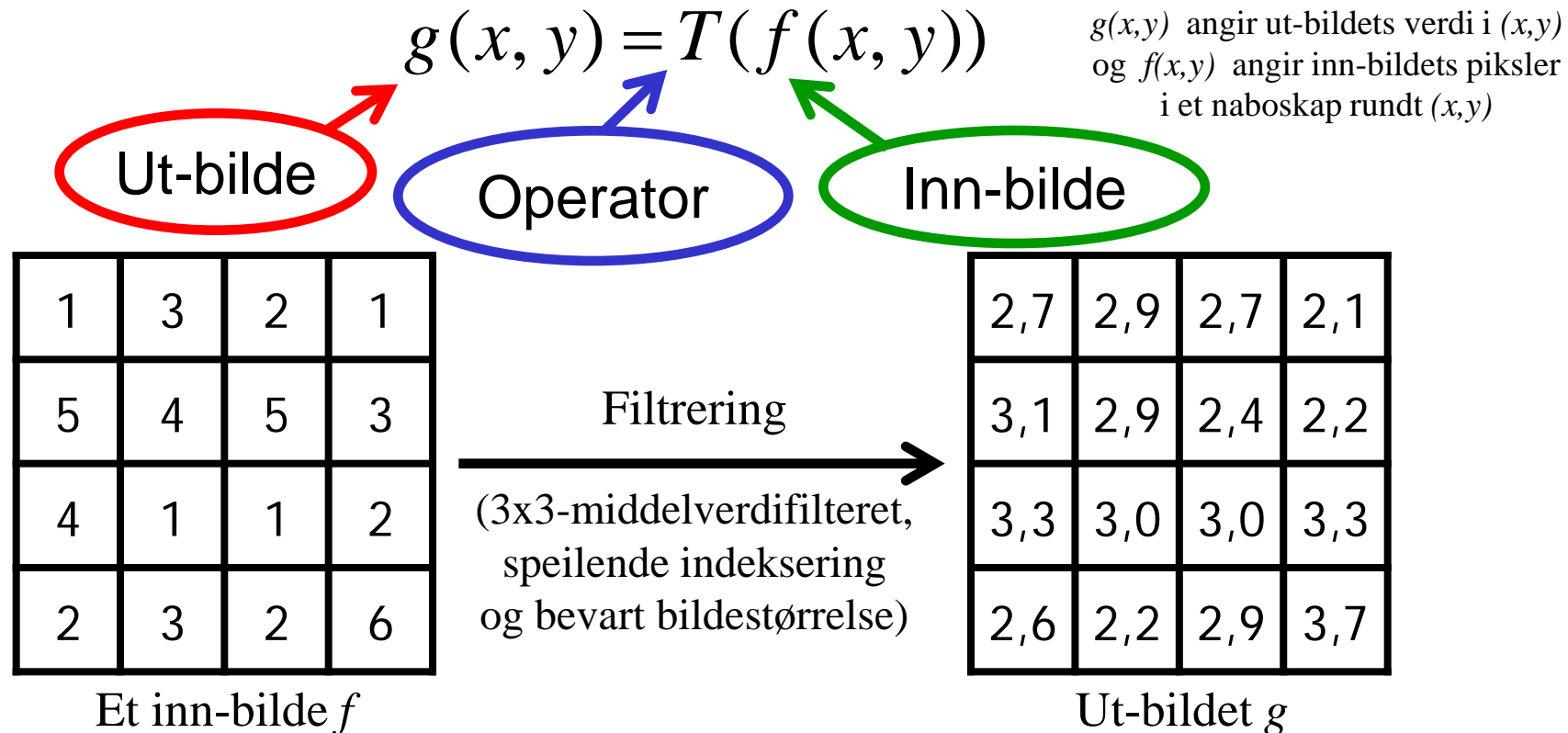
**FIGURE 12.9**

(a) Satellite image of Hurricane Andrew, taken on August 24, 1992. (b) Template of the eye of the storm. (c) Correlation coefficient shown as an image (note the brightest point). (d) Location of the best match. This point is a single pixel, but its size was enlarged to make it easier to see. (Original image courtesy of NOAA.)

- **Q:** Hva hvis templatets størrelse og rotasjon er ukjent?
  - Må det gjøres regnekrevende?

# Filtrering i billedomenet

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

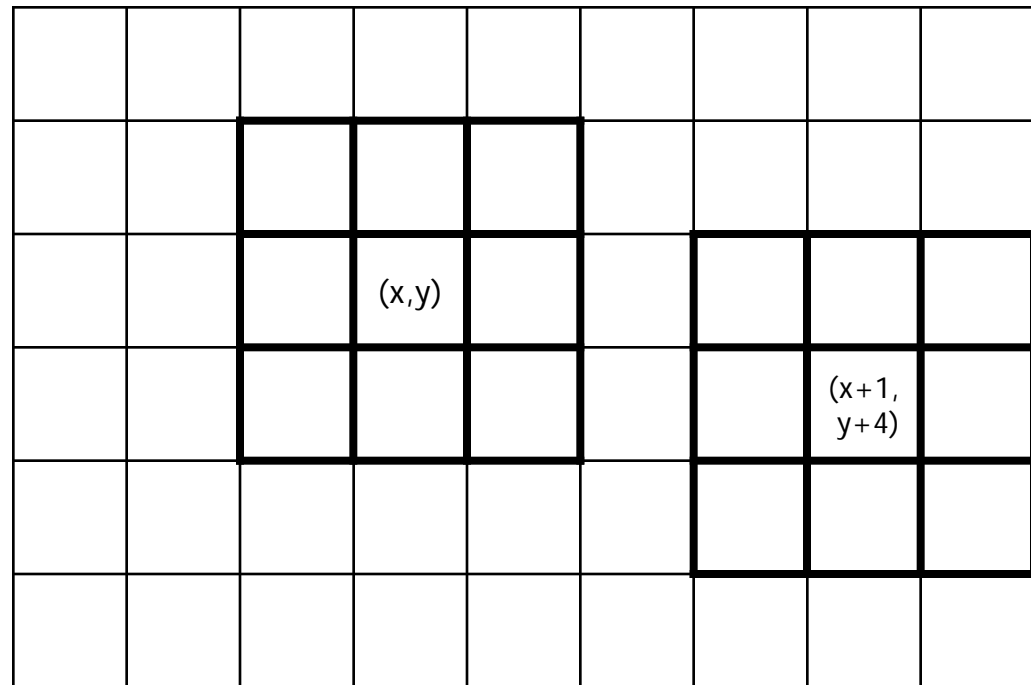




# Naboskap: Definisjon

- Filterets naboskap (eng.: *neighbourhood*) angir **pikslene rundt  $(x,y)$  i inn-bildet** som operatoren (potensielt) benytter.
  - Også kalt *omegn* og *omgivelse*.

- Eksempel:  
Et sentrert  
3x3-naboskap  
rundt  $(x,y)$   
og  $(x+1,y+4)$ :



# Naboskap: I praksis

---

- Kvadrater/rektangler er mest vanlig.
  - Av symmetrihensyn er bredden/høyden oftest odde.
    - Da er naboskapets senterpunkt i en piksel.
    - Når annet ikke er spesifisert så er senterpunktet naboskapets **origo**.
- Spesialtilfelle: Naboskapet er  $1 \times 1$ :
  - $T$  er da en gråtonetransform; ny pikselverdi avhenger bare av den gamle pikselverdien i samme pikselposisjon  $(x,y)$ .
  - Hvis  $T$  er lik over hele bildet, har vi en **global** operator.
- Hvis naboskapet er større enn  $1 \times 1$ , har vi en **lokal** operator (selv om  $T$  er posisjons-invariant).
  - Filtreringen kalles da en **naboskaps-operasjon** (eng. *neighbourhood processing technique*).

# Naboskap + Operator = Filter

- Naboskap:
  - Angir de pikslene rundt  $(x,y)$  i inn-bildet som  $T$  opererer på.
- Operator:
  - Også kalt *transform* og *algoritme*.
  - Opererer på pikslene i et naboskap.
- Filter: Naboskap + Operator
  - Også kalt *maske* og *kjerne*.
    - og *vindue*, men vi vil bare bruke det om et (muligens ikke-rektangulært) delbilde.
  - Eksempel: Et konvolusjonsfilter, f.eks.  $\longrightarrow$


3x3-naboskap

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

$\longrightarrow$  3x3-middelverdifilter

# Filter-egenskap: Additivt?

---

$$T(f_1(x, y) + f_2(x, y)) = T(f_1(x, y)) + T(f_2(x, y))$$

- $T$  er operatoren.
- $f_1$  og  $f_2$  er vilkårlige bilder.
- $(x, y)$  er en vilkårlig piksel.

$f_1(x, y) + f_2(x, y)$  angir bildet  $f_1 + f_2$  sine piksler i et naboskap rundt  $(x, y)$  og burde derfor strengt tatt vært skrevet  $(f_1 + f_2)(x, y)$

- Hva betyr dette:
  - Hvis vi skal addere to filtrerte bilder?

# Filter-egenskab: Homogent?

---

$$T(af(x, y)) = aT(f(x, y))$$

- $T$  er operatoren.
  - $a$  er en vilkårlig konstant.
  - $f$  er et vilkårlig bilde.
  - $(x, y)$  er en vilkårlig piksel.
- Hva betyr dette:
    - Hvis vi skalerer et bilde før filtrering?

# Filter-egenskab: Lineært?

---

$$T(af_1(x, y) + bf_2(x, y)) = aT(f_1(x, y)) + bT(f_2(x, y))$$

- $T$  er operatoren.
- $a$  og  $b$  er vilkårlige konstanter.
- $f_1$  og  $f_2$  er vilkårlige bilder.
- $(x, y)$  er en vilkårlig piksel.

- Additiv + Homogen = Lineær

# Filter-egenskap: Posisjons-invarians?

---

$$T(f(x-l, y-m)) = g(x-l, y-m)$$

- $T$  er operatoren.
  - $f$  er et vilkårlig bilde.
  - $(x,y)$  er en vilkårlig piksel.
  - $g(x,y) = T(f(x,y))$  for alle  $(x,y)$ .
  - $(l,m)$  er et vilkårlig posisjons-skift.
- Operatoren bruker ikke pikselposisjonene.
    - Ut-bildets verdi i  $(x,y)$  avhenger kun av inn-bildets **verdier** i naboskapet rundt  $(x,y)$ , **ikke av posisjonene**.

# 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: **Glattning**/utsmøring/«blurring» av bildet.
- Typiske mål: Fjerne støy, finne større objekter.
- **Utfordring**: **Bevare kanter**.



# Middelverdifilter (lavpass)

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

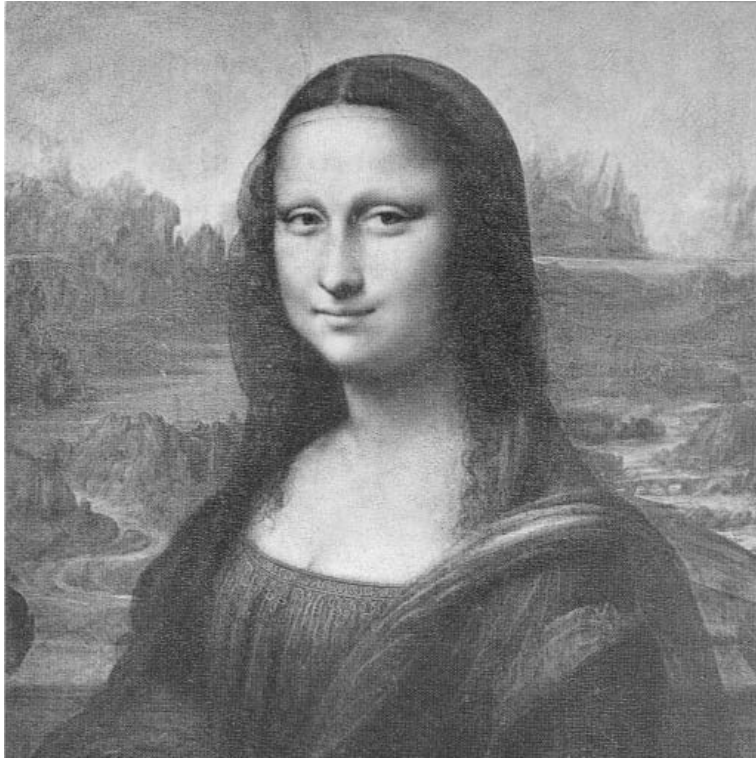
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

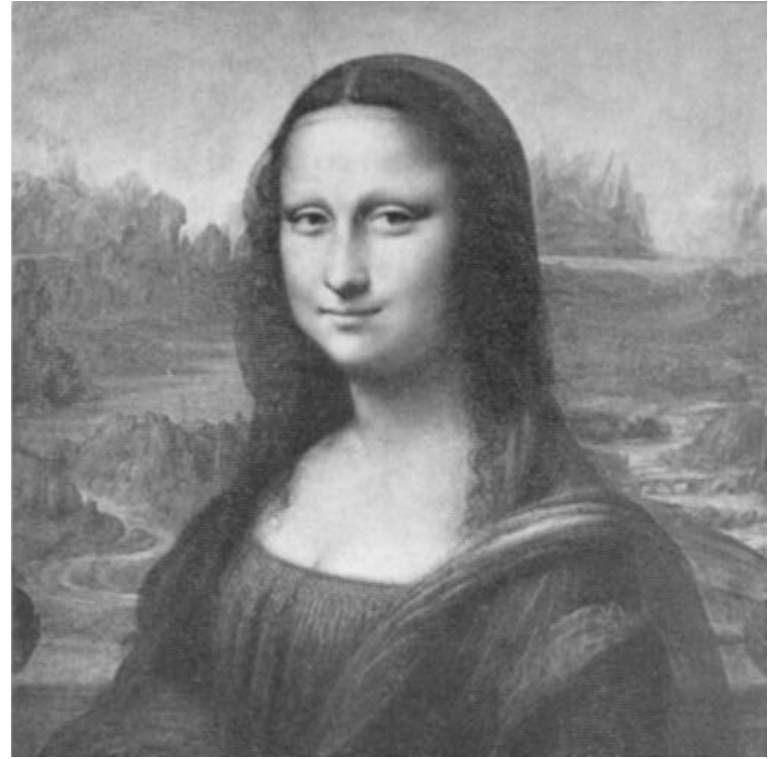
$$\frac{1}{49} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

# Middelverdifilter-eksempel

---



Original



Filtrert med  
3x3-middelverdifilteret

# Middelverdifilter-eksempel

---



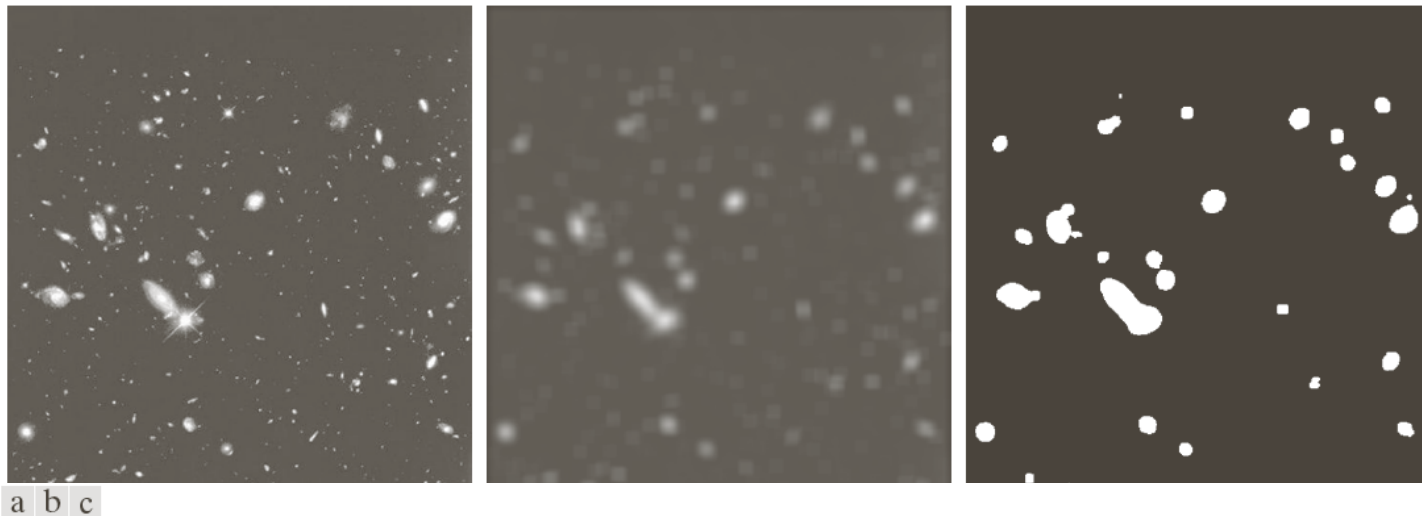
Filtrert med  
9x9-middelverdifilteret



Filtrert med  
25x25-middelverdifilteret

# Middelverdifilter-eksempel

- **Oppgave:** Finne store objekter.
  - I denne oppgaven er objektpikslene lyse.
- **Løsning:** 15x15-middelverdifiltrering + terskling.
  - Filterstørrelsen relaterer seg til hva man legger i «store».



**FIGURE 3.34** (a) Image of size  $528 \times 485$  pixels from the Hubble Space Telescope. (b) Image filtered with a  $15 \times 15$  averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

# Separable filtre

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

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

- 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.

# Tidsbesparelse ved separasjon

- Vanlig 2D-konvolusjon:  
 $n^2$  multiplikasjoner og  $n^2-1$  addisjoner.
- To 1D-konvolusjoner:  
 $2n$  multiplikasjoner og  $2(n-1)$  addisjoner.
- Andel spart tid ved separasjon  
av  $n \times n$ -konvolusjonsfilter:

$n$	$\Delta$	$\Delta'$
3	0,41	0,33
5	0,63	0,60
7	0,73	0,71
9	0,79	0,78
11	0,83	0,82
13	0,85	0,85
15	0,87	0,87

$$\Delta = \frac{2n^2 - 1 - (2n + 2(n-1))}{2n^2 - 1} = 1 - \frac{4n - 2}{2n^2 - 1} \stackrel{n \text{ stor}}{\approx} 1 - \frac{2}{n} = \Delta'$$

# Konvolusjon ved oppdatering

- Konvolusjonsfiltre med identisk kolonner *eller* rader kan effektivt implementeres ved oppdatering:
  1. Beregn responsen  $R$  for første piksel,  $(x,y)$ .
  2. Beregn responsen i neste piksel  $R_{ny}$  med utgangspunkt i  $R$ :
    - For identisk kolonner: Neste piksel er én til høyre,  $(x,y+1)$ , og ny respons er gitt ved:
$$R_{ny} = R - C_1 + C_n$$
der  $C_1$  er «responsen» for først kolonne når plassert i  $(x,y)$ , og  $C_n$  er «responsen» for siste kolonne når plassert i  $(x,y+1)$ .
    - For identiske rader: Neste piksel er én ned,  $(x+1,y)$ , og ny respons er gitt ved:
$$R_{ny} = R - R_1 + R_n$$
der  $R_1$  er «responsen» for først rad når plassert i  $(x,y)$ , og  $R_n$  er «responsen» for siste rad når plassert i  $(x+1,y)$ .
  3. La neste piksel være  $(x,y)$  og  $R_{ny}$  være  $R$ . Gjenta steg 2.

# Konvolusjon ved oppdatering

---

- $C_1$ ,  $C_n$ ,  $R_1$  eller  $R_n$  beregnes ved  $n$  multiplikasjoner og  $n-1$  addisjoner.
  - Tar totalt  $2n$  multiplikasjoner og  $2n$  addisjoner å finne  $R_{ny}$ .
- Like raskt som separabilitet.
  - Sett bort fra initieringen; finne  $R$  for hver ny rad eller kolonne.
  - Alle «oppdaterbare» konvolusjonsfiltre er separable, men separable konvolusjonsfiltre må ikke være oppdaterbare.
  - Kan kombineres med separabilitet når et 1D-filter er uniformt.
- Uniforme filtre kan implementeres enda raskere.
  - Sett bort fra initiering (finne en kumulativ matrise) så kan hver respons beregnes ved 2 subtraksjoner og 1 addisjon!
  - Kun skalerings av middelveidifiltre er uniforme filtre.



# Gauss-filter (lavpass)

- For heltallsverdier av  $x$  og  $y$ , la:

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

- $A$  settes slik at summen av vektene blir 1.
- $N(0, \sigma^2 I_2)$  med alternativ skalering  $A$ .

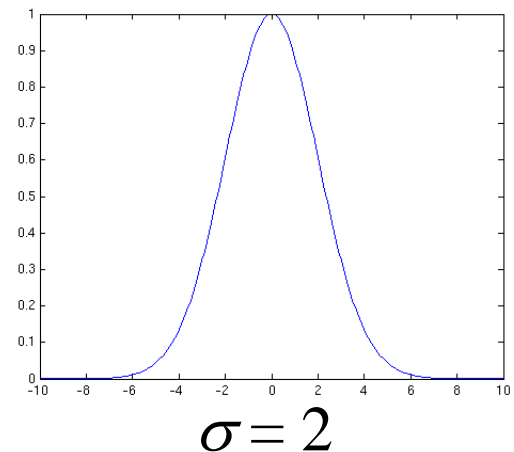
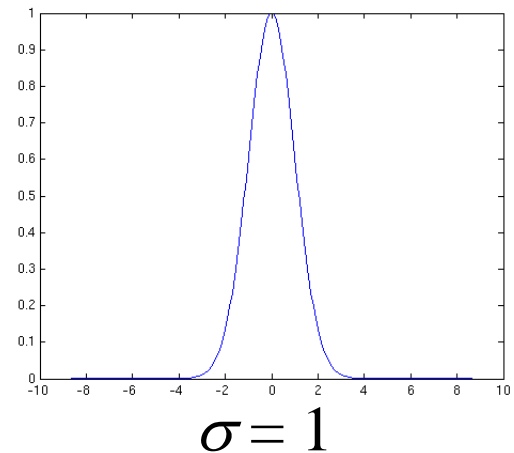
- **Ikke-uniformt lavpassfilter.**

- Parameteren  $\sigma$  er standardavviket og avgjør graden av glatting.

- Lite  $\sigma \Rightarrow$  lite glatting.
- Stort  $\sigma \Rightarrow$  mye glatting.

- Filterstørrelsen  $n \times n$  må tilpasses  $\sigma$ .

- Et Gauss-filter glatter *mindre* enn et middelværdifilter av samme størrelse.



# Approksimasjon av 3x3-Gauss-filter

---


$$\begin{aligned} G_{3 \times 3} &= \frac{1}{16} \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \end{bmatrix} \\ &= \frac{1}{16} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ &= \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \\ &= \frac{1}{16} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \end{aligned}$$

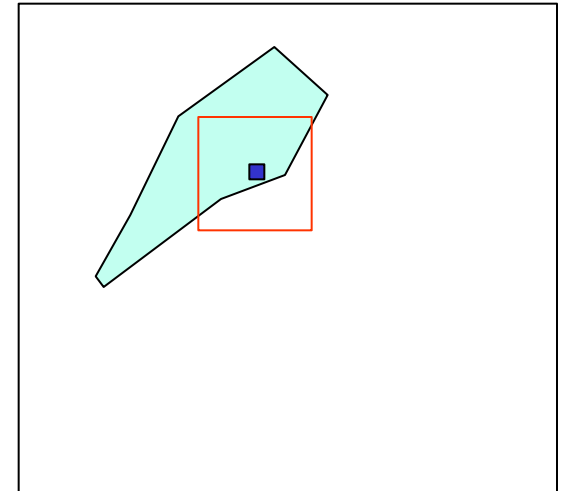
Tilsvarener en

**geometrisk vekting:**

- Vekten til en 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 alle pikslene.
- Vi kan sortere pikslene:
  - Radiometrisk (etter pikselverdi)
  - Både geometrisk (etter pikselposisjon) og radiometrisk



# Rang-filtrering

---

- Vi lager en én-dimensjonal liste av alle pikselverdiene i naboskapet rundt  $(x,y)$ .
- Listen sorteres i stigende rekkefølge.
- Responsen i  $(x,y)$  er pikselverdien i en bestemt posisjon i den sorterte listen.
- Ikke-lineært filter.

# Median-filter (lavpass)

---

- $g(x,y)$  = median pikselverdi i naboskapet rundt  $(x,y)$ .
- Median = den midterste verdien i den sorterte listen.
  - Så et medianfilter er et rangfilter der vi velger midterste posisjon i den sorterte 1D-listen.
- Et av de mest brukte kant-bevarende støyfiltrene.
- Spesielt god mot impuls-støy («salt-og-pepper-støy»).
- Ikke uvanlig med ikke-rektangulære naboskap, f.eks. +
- Problemer:
  - Tynne linjer kan forsvinne.
  - Hjørner kan rundes av.
  - Objekter kan bli litt mindre.
- Valg av størrelse og form på naboskapet er viktig!

# Middelverdi eller median?

---



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



Etter middelverdifiltrering



Etter medianfiltrering

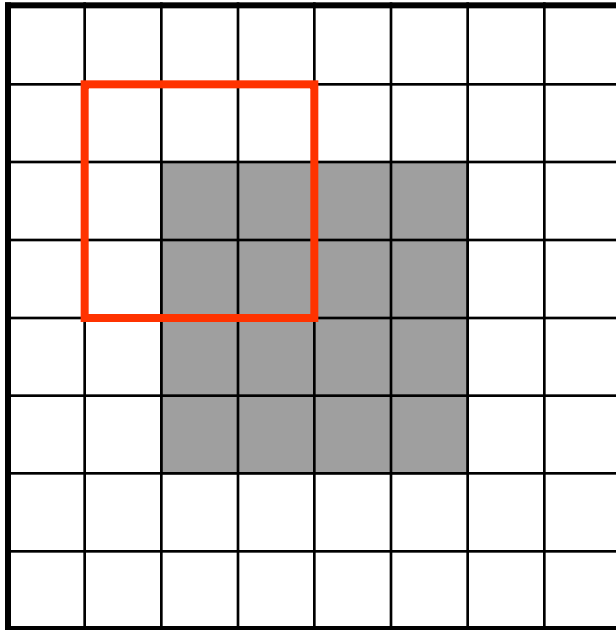
# Middelverdi eller median?

---

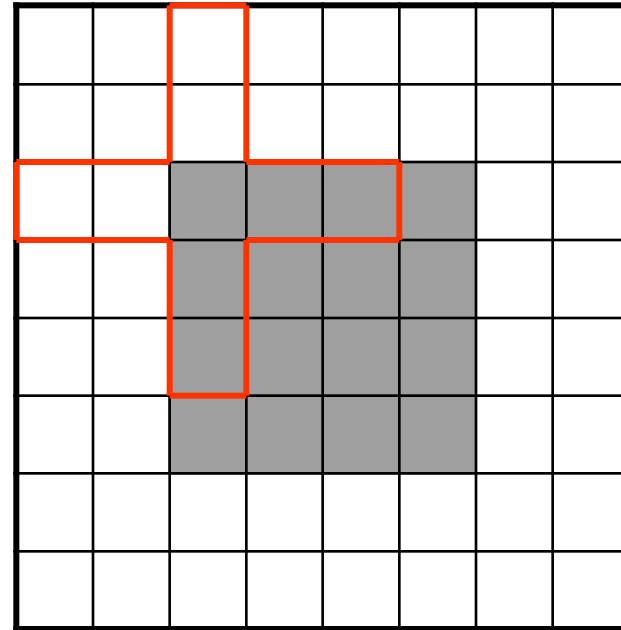
- Middelverdifilter: Middelverdien innenfor naboskapet.
  - Glatter lokale variasjoner og støy, men også kanter.
  - **Spesielt god** på lokale variasjoner, som kan være **mild støy i mange pikselverdier**.
- Medianfilter: Medianen innenfor naboskapet.
  - Bedre på visse typer støy og til å bevare kanter, men dårligere på lokal variasjon og annen type støy.
  - Fungerer **spesielt godt på salt-og-pepper-støy**.

# Medianfiltrering og hjørner

---



Med kvadratisk naboskap  
avrundes hjørnene



Med pluss-formet naboskap  
bevares hjørnene



# Raskere medianfiltrering (kursorisk)

---

- Å sortere pikselverdiene innenfor naboskapet er regnetungt.
- Med bit-logikk kan medianen av  $N$  verdier finnes i  $O(N)$ .
  - Avhenger også av antall biter i representasjonen av en verdi.
  - For  $m \times n$  -naboskap er  $N = n^2$ .
  - Danielsson: "Getting the median faster", CVGIP 17, 71-78, 1981.
- **Oppdater histogrammet** av pikselverdiene i naboskapet for hver ny inn-piksel-posisjon  $(x,y)$  gir  $O(n)$  for  $m \times n$  -naboskap.
  - Avhenger også av antall mulige pikselverdier *eller* middelveien av absoluttdifferansen mellom horisontale (eller vertikale) nabopiksler i det filtrerte bildet.
  - Huang, Yang, and Tang: "A fast two-dimensional median filtering algorithm", IEEE TASSP 27(1), 13-18, 1979.
- Oppdater histogrammet ved bruk av oppdatert kolonnehistogram (gjenbraker lik informasjon fra forrige rad) gir  $O(1)$ .
  - Må avhenge av antall mulige pikselverdier.
  - Finnes i OpenCV programbibliotek (se <http://nomis80.org/ctmf.html>).
  - Perreault and Hébert: "Median filtering in constant time", IEEE TIP 16(9), 2389-2394, 2007.

# Alfa-trimmet middelvefilter (lavpass)

---

- $g(x,y)$  = middelveien av de  $mn-d$  midterste verdiene (etter sortering) i  $m \times n$  -naboskapet rundt  $(x,y)$ :

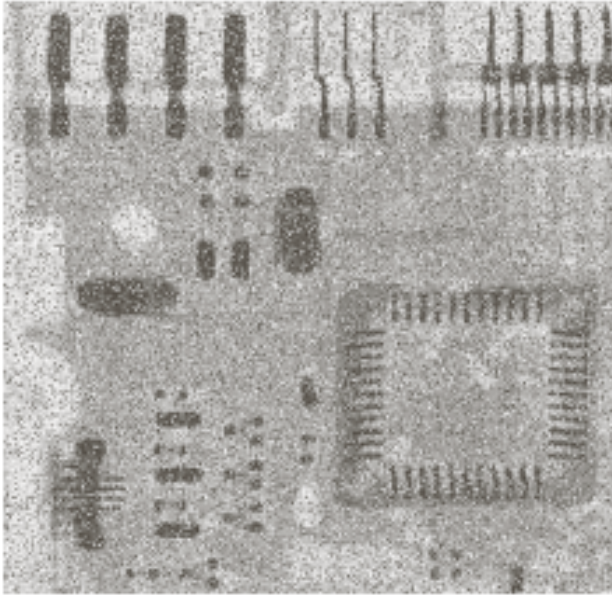
$$g(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{x,y}} f(s, t)$$

der  $S_{x,y}$  angir pikselposisjonene til de  $mn-d$  midterste pikselverdiene etter sortering.

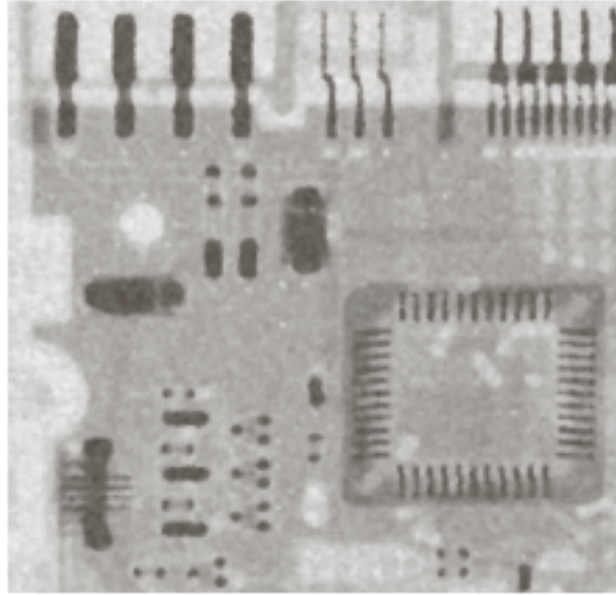
- God egnet ved **flere typer støy**, f.eks. salt-og-pepper-støy og lokale variasjoner.
- **Q**: Hva hvis  $d=0$ ? Eller  $d=mn-1$ ?

# Alfa-trimmet middelvefilter (lavpass)

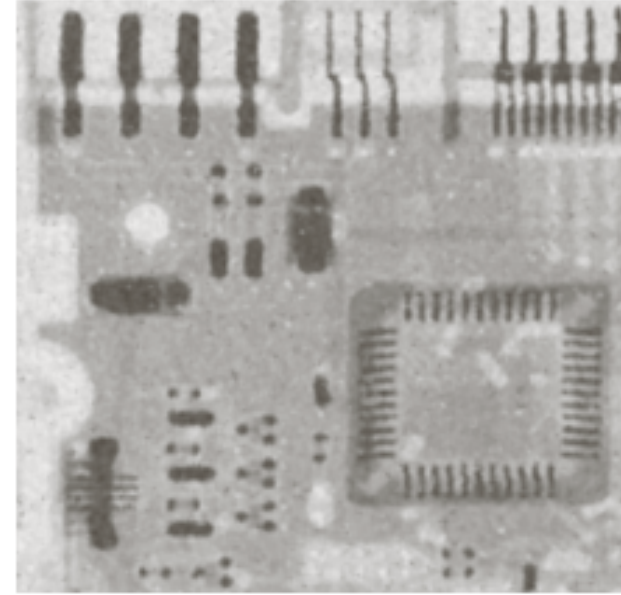
---



Inn-bilde med  
lokal variasjon-støy  
og salt-og-pepper-støy



Etter  
5x5-medianfiltrering



Etter alfa-trimmet  
middelvefiltrering med  
 $d=5$  og 5x5-naboskap

Fra fig. 5.12 i DIP.

# $K$ Nearest Neighbour-filter (lavpass)

---

- $g(x,y)$  = middelveien av de  $K$  pikslene i naboskapet rundt  $(x,y)$  som ligner mest på  $(x,y)$  i pikselverdi.
  - «Ligner mest» = Lavest absoluttdifferanse.
- Er et trimmet middelveidifilter:  
Middelveien av de  $K$  mest like nabopikslene.
- Problem:  $K$  er konstant for hele bildet.
  - Hvis vi velger for liten  $K$ , fjerner vi lite støy.
  - Hvis vi velger for stor  $K$  fjernes linjer og hjørner.

For  $n \times n$ -naboskap der  $n=2a+1$ :

- $K=1$ : ingen effekt
- $K \leq n$ : bevarer tynne linjer
- $K \leq (a+1)^2$ : bevarer hjørner
- $K \leq (a+1)n$ : bevarer rette kanter

# $K$ Nearest Connected Neighbour-filter (lavpass)

---

- Naboskapet er uendelig stort!
- Responsen i  $(x,y)$  beregnes slik:
  - Valgt piksel =  $(x,y)$
  - Liste = <tom>
  - While # valgte piksler  $< K$ 
    - Tilføy (4- eller 8-)naboene til sist valgt piksel i listen.
      - Ikke tilføy piksler som allerede finnes i listen.
    - Sorter listen på pikselverdi.
    - Velg pikselen i listen som er mest lik  $(x,y)$  og fjern den fra listen.
  - Beregn middelveiden av de  $K$  valgte pikslene.
- Tynne linjer, hjørner og kanter blir bevart dersom  $K$  er mindre eller lik antall piksler i objektet.

# MinimalMeanSquareError (MMSE) (lavpass)

---

- Merk: For et naboskap rundt  $(x, y)$  kan vi beregne lokal varians  $\sigma^2(x, y)$  og lokal middelværdi  $\mu(x, y)$ .
- Anta at vi har et estimat på støy-variansen,  $\sigma_\eta^2$
- MMSE-responsen i  $(x, y)$  er da:

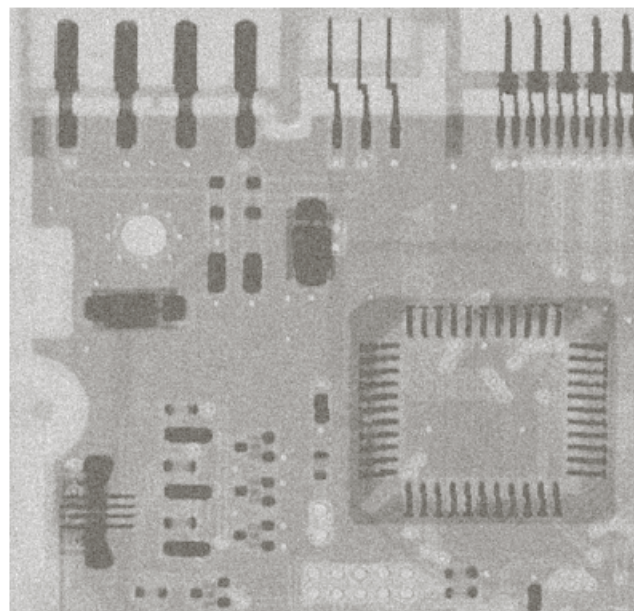
$$g(x, y) = f(x, y) - \frac{\sigma_\eta^2}{\sigma^2(x, y)} (f(x, y) - \mu(x, y))$$

– Hvis  $\sigma_\eta^2 > \sigma^2$  så settes  $g(x, y) = \mu(x, y)$ .

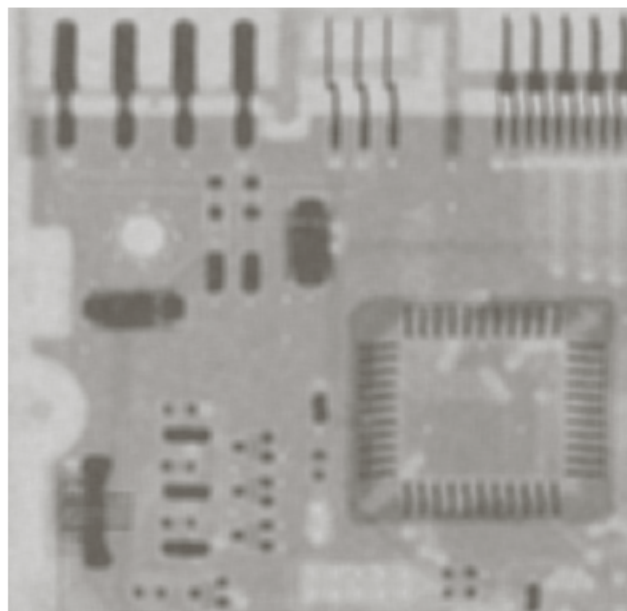
- I «homogene» områder blir responsen nær  $\mu(x, y)$ .
- Nær en kant vil  $\sigma^2(x, y)$  være større enn  $\sigma_\eta^2$  og resultatet blir nær  $f(x, y)$ .

# Minimal Mean Square Error (MMSE) (lavpass)

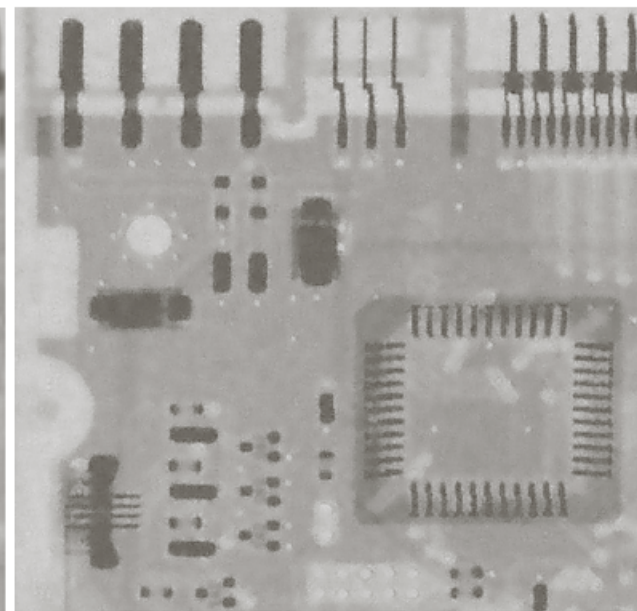
---



Inn-bilde med  
lokal variasjon-støy



Etter  
7x7-middelverdifiltrering



Etter MMSE-filtrering  
med  $\sigma_n^2 = 1000$  og  
7x7-naboskap

Fra fig. 5.13 i DIP.

# Sigma-filter (lavpass) (kursorisk)

- $g(x,y)$  = middelveien av de pikslene i naboskapet rundt  $(x,y)$  som har pikselverdi i intervallet  $f(x,y) \pm t \sigma$ 
  - $t$  er en parameter og  $\sigma$  er estimert i «homogene» områder i bildet, f.eks. som en lav persentil av alle lokale standardavvik (beregnet rundt hver piksel ved bruk av det aktuelle naboskapet).

$$g(x,y) = \frac{\sum_{s=-at=-b}^a \sum_{t=-b}^b h_{x,y}(s,t) f(x+s, y+t)}{\sum_{s=-at=-b}^a \sum_{t=-b}^b h_{x,y}(s,t)}, \quad h_{x,y}(s,t) = \begin{cases} 1 & \text{hvis } |f(x,y) - f(x+s, y+t)| \leq t\sigma \\ 0 & \text{ellers} \end{cases}$$

- Alternativt: Erstatte  $\sigma$  med **lokal median absolute deviation** (MAD):

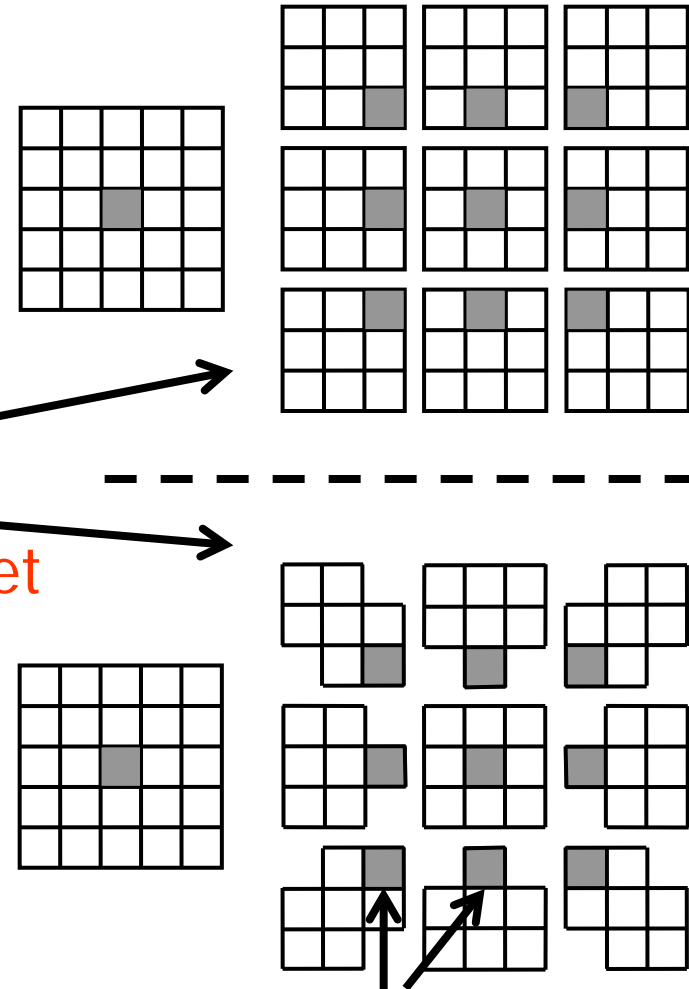
$$\text{MAD}_{x,y} = \text{median}_{s \in [-a,a], t \in [-b,b]} \left( \left| f(x+s, y+t) - \text{median}_{u \in [-a,a], v \in [-b,b]} (f(x+u, y+v)) \right| \right)$$

- Dette filteret kalles *MAD trimmed mean* (MADTM).
- Problem: Ingen av disse filtrene fjerner isolerte støy-pikslar.



# Max-homogenitet (lavpass)

- Ønsker kantbevarende filter.
- *Et enkelt triks:*
- Del opp naboskapet i flere, overlappende sub-naboskap.
  - Alle inneholder senterpikselen.
  - Flere mulige oppdelinger.
- **Det mest homogene sub-naboskapet inneholder minst kanter.**
  - Beregn  $\mu$  og  $\sigma$  i hvert sub-naboskap.
  - La  $(x,y)$  være  $\mu$  fra det sub-naboskapet som gir lavest  $\sigma$ .
  - Alternativ:  $|\max-\min|$  istedetfor  $\sigma$ .

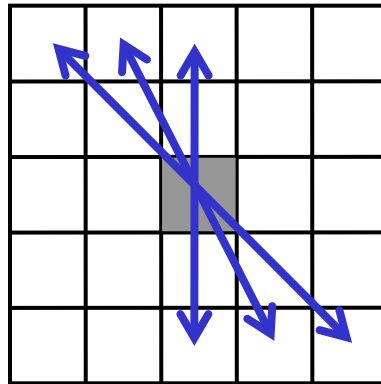


Merk: Sub-naboskapets origo er ikke sub-naboskapets senterpiksel. 57

# Symmetrisk nærmeste nabo (SNN) (lavpass)

---

- For hvert symmetrisk piksel-par i naboskapet rundt  $(x,y)$ :
  - Velg den pikselen som ligner mest på  $(x,y)$  i pikselverdi.



- $g(x,y)$  = middelveien av  $(x,y)$  og de valgte pikslene.
  - Antall verdier som midles v.b.a.  $m \times n$  -naboskap:  $1 + (mn-1)/2$

# Mode-filter (lavpass) (kursorisk)

---

- $g(x,y)$  = hyppigst forekommende pikselverdi i naboskapet rundt  $(x,y)$ .
  - **Q**: Hvordan er dette forskjellig fra median?
- Antall pikselverdier bør være lite i forhold til antall piksler i naboskapet.
  - Brukes derfor sjelden på gråtonebilder.
  - Anvendes mest på segmenterte eller klassifiserte bilder for å fjerne isolerte piksler.
- Kan implementeres v.h.a. histogram-oppdatering.

# Oppsummering

---

- Romlig filter = Naboskap + Operator
  - Operatoren definerer *hvordan* inn-bildet endres.
- Konvolusjon er lineær romlig filtrering.
  - Konvolusjonsfilteret er en matrise av vektorer.
  - Å kunne utføre konvolusjon for hånd på et lite eksempel er sentralt i pensum!
  - Å programmere konvolusjon er sentralt i Oblig1.
- Korrelasjon og korrelasjonskoeffisienten kan brukes til mønstergjenkjenning.
- Lavpassfiltrering kan fjerne støy.

# Lavpassfiltre i denne forelesningen

Navn	Kommentar
Middelverdifilter	Uniformt konvolusjonsfilter.
Gauss-filter	Konvolusjonsfilter med geometrisk vekting.
Rang-filter	Husk medianfilteret! Det bevarer kanter bedre enn middelverdifilteret.
Alfa-trimmet middelverdifilter	Mellomting mellom middelverdi- og medianfilter.
$K$ Nearest Neighbour-filter	$K$ må velges med omhu.
$K$ Nearest Connected Neighbour-filter	Må også være geometriske naboer.
MinimalMeanSquareError-filter	Benytter at den lokale variansen er større i en kant.
Sigma-filter	(kursorisk)
Max-homogenitet	Sub-naboskap tar i bruk geometrien.
Symmetrisk nærmeste nabo-filter	Symmetrisk parring tar i bruk geometrien.
Mode-filter	(kursorisk)