IN2070 - Filtrering i bildedomenet

23. februar 2022

- Naboskaps-operasjoner
- **Konvolusjon** og korrelasjon
- Lavpassfiltrering og kant-bevaring

Filtrering

- Et generelt verktøy for behandling av digitale bilder
- En av de mest brukte operasjonene i bildebehandling
- Brukes ofte som en del i bl.a.:
  - Bildeforbedring
  - Bildeanalyse

  til å bl.a.:
  - Fjerne eller redusere støy (**denne forelesningen**)
  - Forbedre skarpheten (**neste forelesning**)
  - Detektere kanter (**neste forelesning**)

Romlig filtrering

Filteret er definert av en matrise, f.eks.:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

3×3 middelverdifilter

Matrisens **størrelse** og **filterkoeffisienter** (vektene) bestemmer resultatet av filtreringen

Eksempel på konvolusjon

Tenker å konvolvere gitt følgende filter og bilde:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

3×3 middelverdifilter

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

Innbilde f

Eksempel på konvolusjon

Tenker å konvolvere gitt følgende filter og bilde:

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

3×3 middelverdifilter

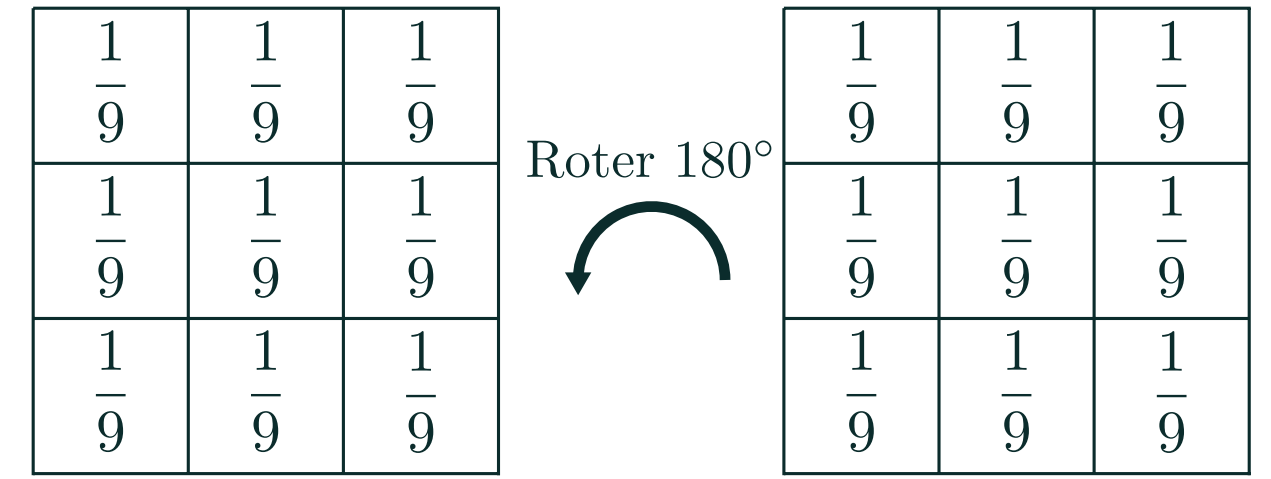
0	0	0	0	0	0
0	1	3	2	1	0
0	5	4	5	3	0
0	4	1	1	2	0
0	2	3	2	6	0
0	0	0	0	0	0

Innbilde f

Anta bildet er 0 utenfor det definerte 4×4 området.

Eksempel på konvolusjon

Steg 1: Roter filteret 180 grader



3×3 middelverdifilter 3×3 middelverdifilter

Her blir det roterte filteret lik filteret før rotering på grunn av symmetri

Eksempel på konvolusjon

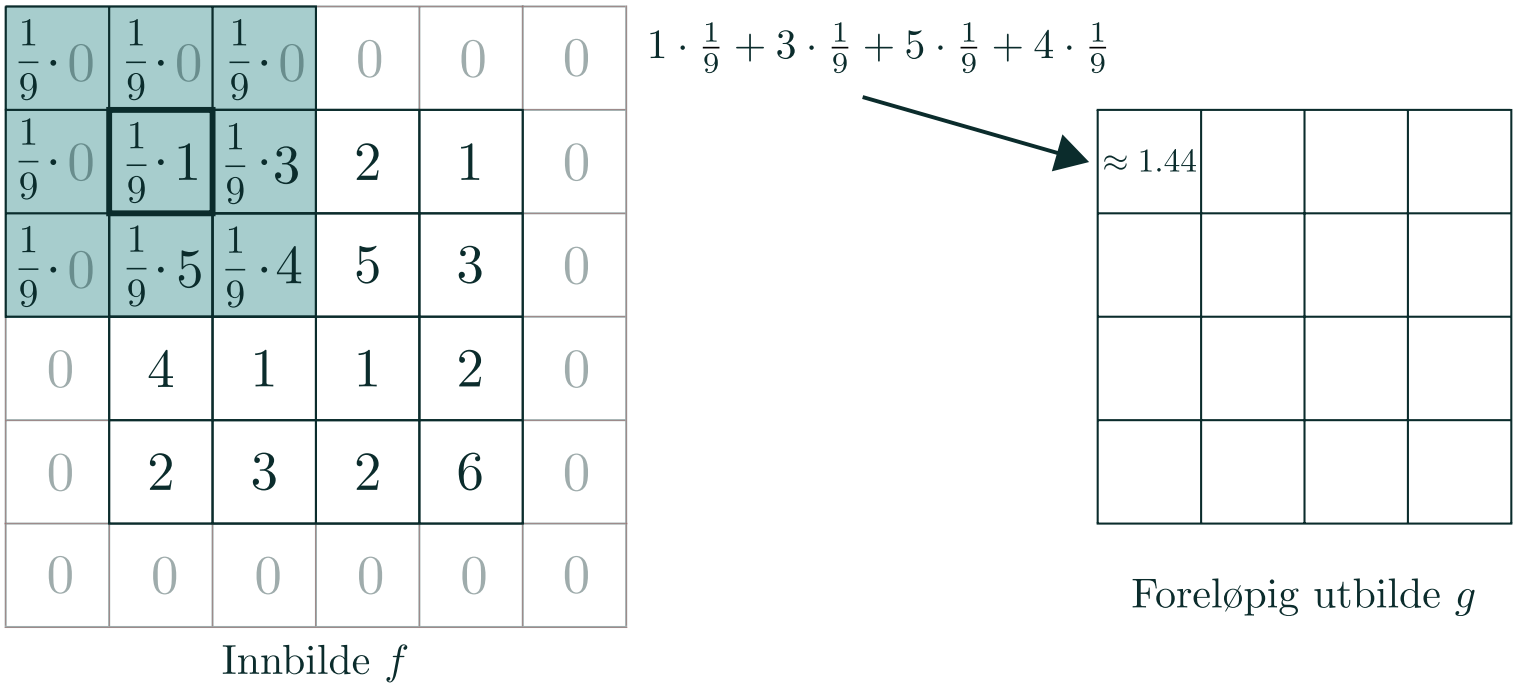
Steg 2: Legg det roterte filteret over første posisjon der filteret og bildet overlapper

$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	0	0	0
$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 1$	$\frac{1}{9}\cdot 3$	2	1	0
$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 5$	$\frac{1}{9}\cdot 4$	5	3	0
0	4	1	1	2	0
0	2	3	2	6	0
0	0	0	0	0	0

Innbilde f

Eksempel på konvolusjon

Steg 3: Multipliser filterets vekter med verdiene av de overlappende pikslene i bildet. Resultatet, eller responsen, er summen av disse produktene:



Eksempel på konvolusjon

Steg 4: Gjenta steg 3 for neste overlapp. Fortsett til det ikke er flere overlapp.

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

0	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	0	0
0	$\frac{1}{9}\cdot 1$	$\frac{1}{9}\cdot 3$	$\frac{1}{9}\cdot 2$	1	0
0	$\frac{1}{9}\cdot 5$	$\frac{1}{9}\cdot 4$	$\frac{1}{9}\cdot 5$	3	0
0	4	1	1	2	0
0	2	3	2	6	0
0	0	0	0	0	0

Innbilde f

$$1\cdot\tfrac{1}{9}+3\cdot\tfrac{1}{9}+2\cdot\tfrac{1}{9}+5\cdot\tfrac{1}{9}+4\cdot\tfrac{1}{9}+5\cdot\tfrac{1}{9}$$

≈ 1.44	≈ 2.22		

Foreløpig utbilde g

Eksempel på konvolusjon

Steg 4: Gjenta steg 3 for neste overlapp. Fortsett til det ikke er flere overlapp.

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

Fjorten steg 3 senere:

0	0	0	0	0	0
0	1	3	2	1	0
0	5	4	5	3	0
0	$\frac{1}{9}\cdot 4$	$\frac{1}{9}\cdot 1$	$\frac{1}{9}\cdot 1$	2	0
0	$\frac{1}{9}\cdot 2$	$\frac{1}{9}\cdot 3$	$\frac{1}{9}\cdot 2$	6	0
0	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	0	0

Innbilde f

$$4\cdot\tfrac{1}{9}+1\cdot\tfrac{1}{9}+1\cdot\tfrac{1}{9}+2\cdot\tfrac{1}{9}+3\cdot\tfrac{1}{9}+2\cdot\tfrac{1}{9}$$

≈ 1.44	≈ 2.22	2	≈ 1.22
2	≈ 2.89	≈ 2.44	≈ 1.56
≈ 2.11	3	3	≈ 2.11
≈ 1.11	≈ 1.44		

Foreløpig utbilde g

Eksempel på konvolusjon

Steg 4: Gjenta steg 3 for neste overlapp. Fortsett til det ikke er flere overlapp.

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

... og etter 16 steg 3 senere:

0	0	0	0	0	0
0	1	3	2	1	0
0	5	4	5	3	0
0	4	1	$\frac{1}{9}\cdot 1$	$\frac{1}{9}\cdot 2$	$\frac{1}{9}\cdot 0$
0	2	3	$\frac{1}{9}\cdot 2$	$\frac{1}{9}\cdot 6$	$\frac{1}{9}\cdot 0$
0	0	0	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$	$\frac{1}{9}\cdot 0$

Innbilde f

$1 \cdot \frac{1}{9} + 2 \cdot \frac{1}{9} + 2 \cdot \frac{1}{9} + 6 \cdot \frac{1}{9}$

≈ 1.44	≈ 2.22	2	≈ 1.22
2	≈ 2.89	≈ 2.44	≈ 1.56
≈ 2.11	3	3	≈ 2.11
≈ 1.11	≈ 1.44	≈ 1.67	≈ 1.22

Utbilde g

Vi har ikke flere mulige overlapp mellom bilde og filter $\rightarrow$ Vi er ferdige med konvolusjonen!

Konvolusjon: definisjon

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

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

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

  for alle mulige posisjoner (x, y) slik at hver verdi av h overlapper med minst en verdi av f.
  For å forenkle notasjon, er det antatt at:
  - h har odde lengder
  - Senterpikselen er naboskapets origo
- Responsen (resultatet) i (x, y) er en **vektet sum av innbildets pikselverdier**

Regne ut en konvolusjon

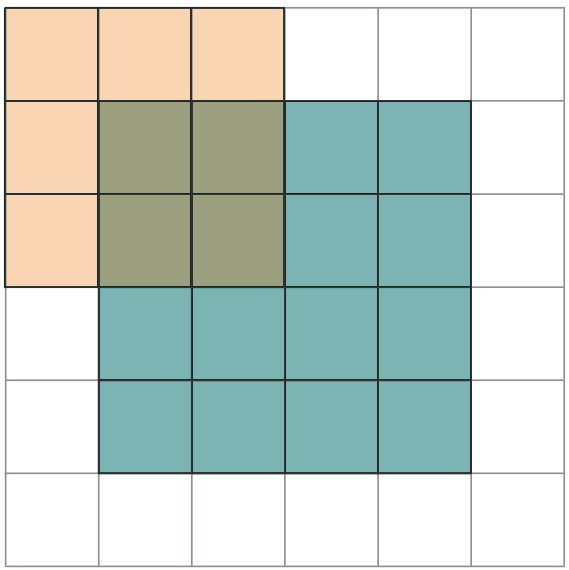
Konvolusjon mellom et filter h og bilde f:

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

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

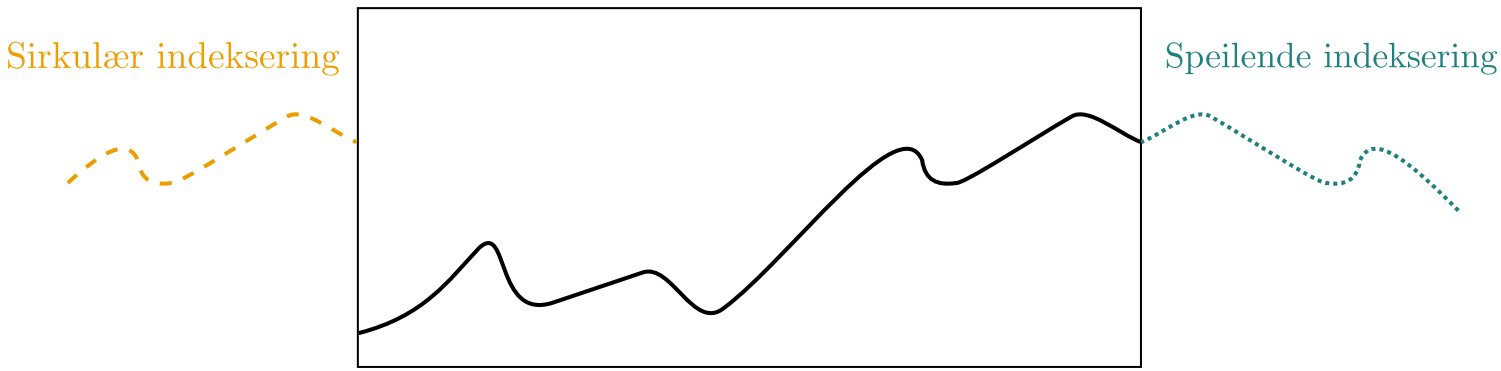
Hvor stort blir utbildet?

- **Alt. 1:** Beregn pikselverdier bare der hele filteret er innenfor innbildet.
  Hvis bildet er $M \times N$ og filteret er $m \times n$ (m, n odde), vil utbildet få størrelse: $(M - m + 1) \times (N - n + 1)$.
- **Alt. 2:** Behold innbildets størrelse:
  - Filterorigo innenfor innbildet
  - Langs bilderanden må det gjøres et valg:
    - **Utvide** bildet ved å legge til flere verdier rundt det
    - Endre filterets form og størrelse



Hva gjør vi langs bilderanden?

- Utvid innbildet:
  - Med 0-ere (nullutvidelse, zero padding)
  - Med en annen, konstant verdi som f.eks bildets gjennomsnitt
  - Ved nærmeste pikselverdi
  - Ved bruk av speilende indeksering
  - Ved bruk av sirkulær indeksering



- Sett pikselet til utbildet til en fast verdi, f.eks $g(x, y) = 0$ eller $g(x, y) = f(x, y)$
- Ignorer posisjonene uten fult overlapp (men mindre utbilde)

Egenskaper ved konvolusjon

- Kommutativ:

$$f \circledast g = g \circledast f$$

- Assosiativ:

$$(f \circledast g) \circledast h = f \circledast (g \circledast h)$$

- Distributiv:

$$f \circledast (g + h) = f \circledast g + f \circledast h$$

- Assosiativ ved skalar multiplikasjon:

$$a(f \circledast g) = (af) \circledast g = f \circledast (ag)$$

Disse gjelder kun hvis man antar nullutvidelse!

Kan brukes i sammensatte konvolusjoner

Korrelasjon: definisjon

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

- Ulikt fra konvolusjon at det er pluss istedenfor minus $\rightarrow$ betyr vi ikke skal rotere filteret!
- Tilsvarende som konvolusjon: legger korrelasjonsfilterets origo over (x, y) og multipliserer hver vekt med underliggende pikselverdi. Responsen i (x, y) er summen av disse produktene.

Mønstergjenkjenning

- Korrelasjon kan brukes til å gjenkjenne mønster
  - Forhåndsdefinert korrelasjonsfilter
  - Et utsnitt av bildet vi skal filtrere
  - Trene en algoritme til å velge optimalt filter til å detekere egenskaper ved objekter (f.eks ved konvolusjonsnettverk)
- Normaliser ved summen av pikselverdier:

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

Mønstergjenkjenning: eksempel

Finn bestemt objekt i bildet:

Templaten, objektet vi ønsker å finne

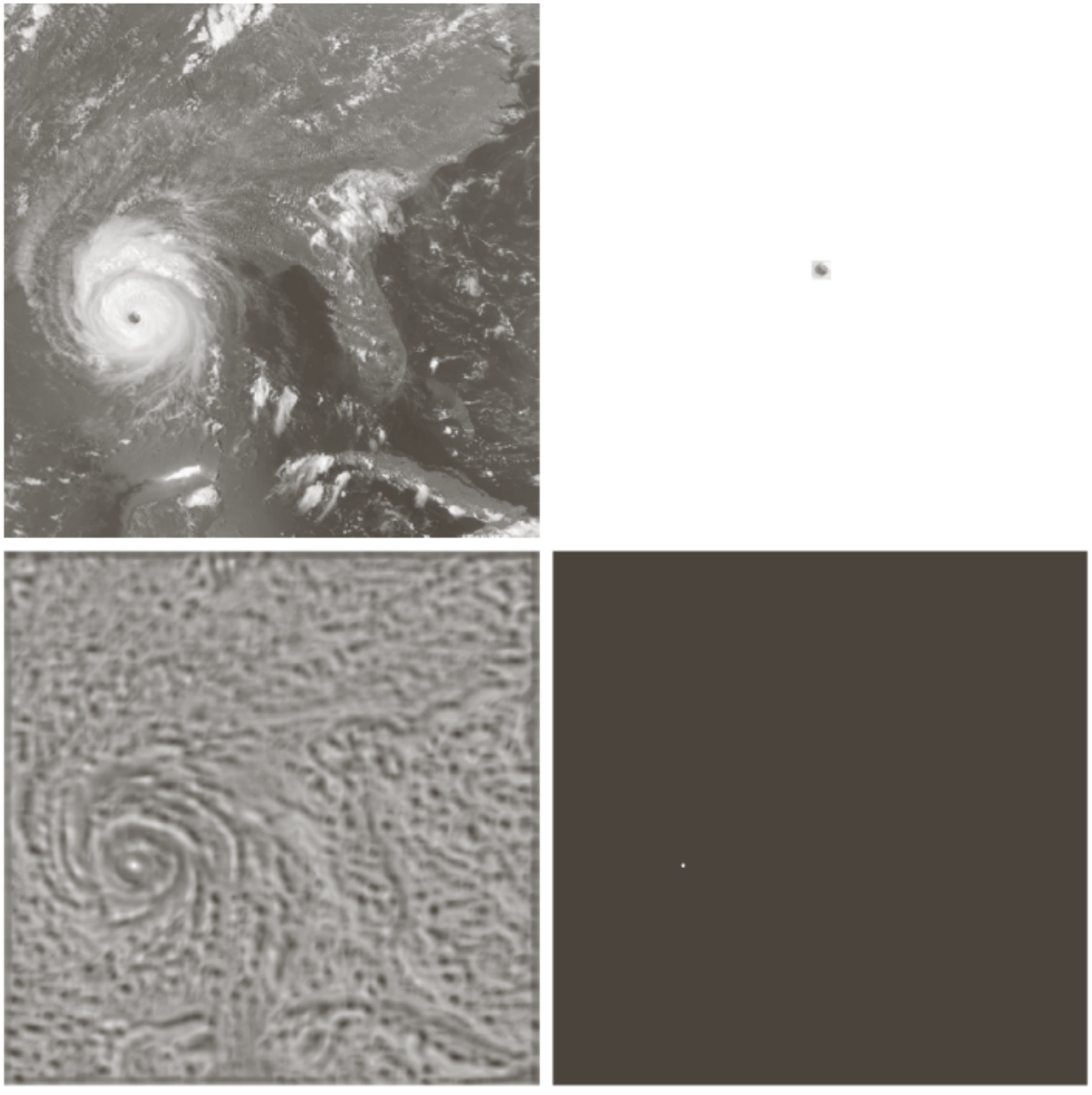
Korrelasjonskoeffisient

Korrelasjonskoeffisient:

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

- $\gamma(x,y)$ er alltid i intervallet $[-1,1]$
- Deler på: bildets lokale standardavvik og templatens 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.)

Filtrering i bildedomenet

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

der $g(x, y)$ er utbilde og $T[f(x, y)]$ operator i et *naboskap* innbildets piksler rundt (x, y)

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

Innbilde f

Filtrering

3×3 middelverdifilter, speilende indeksering

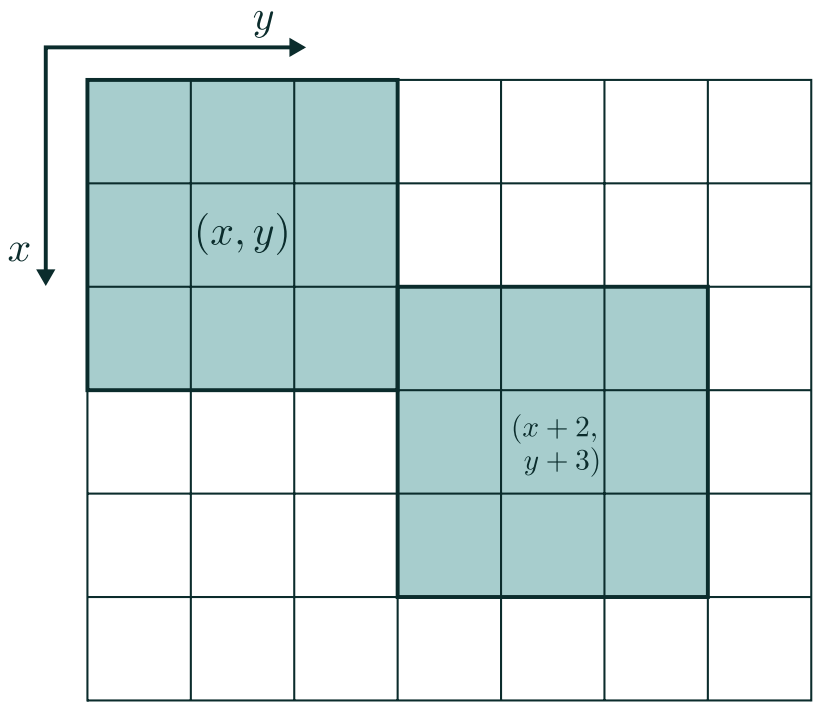
≈ 2.67	≈ 2.89	≈ 2.67	≈ 2.11
≈ 3.11	≈ 2.89	≈ 2.44	≈ 2.22
≈ 3.33	3	3	≈ 3.33
≈ 2.56	≈ 2.22	≈ 2.89	≈ 3.67

Utbilde g

Naboskap

Filterets naboskap: Pikslene rundt gitt posisjon (x, y) i innbildet som operatoren baserer sin utregning på

Eksempel: sentrert 3×3 naboskap rundt (x, y) og $(x + 2, y + 3)$

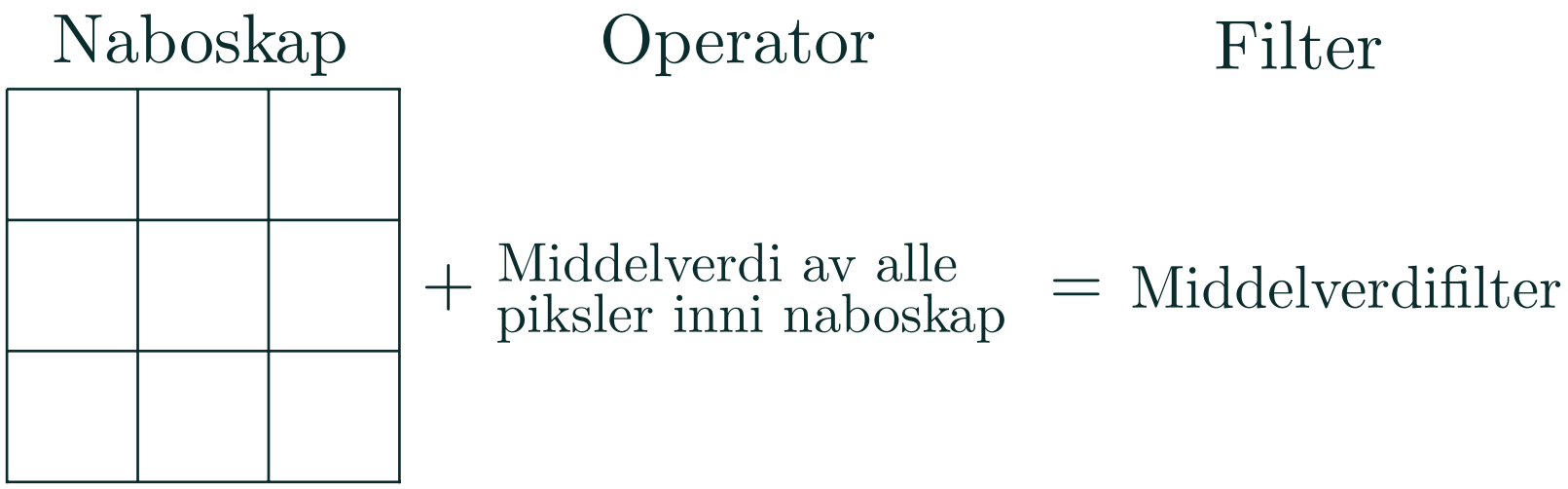


Naboskap: I praksis

- Kvadrater/rektangler er mest vanlig
  - Bredden og høyde er ofte odde pga symmetrihensyn
  - Når annet ikke er spesifisert, så er **senterpunktet** naboskapets origo
- Spesialtilfelle ved 1×1 naboskap:
  - T blir en gråtonetransformasjon; ny pikselverdi er kun avhengig av den gamle pikselverdien i samme posisjon (x, y)
  - T er en **global** operator hvis den er lik over hele bildet
- Hvis naboskapet er større enn 1×1, har vi en **lokal** operator. Filtrering kalles da en **naboskaps-operasjon**.

Naboskap + Operator = Filter

- **Naboskap:** Angir pikslene rundt (x, y) som T operer på i innbildet
- **Operator:** Operer på pikslene i et gitt naboskap
- **Filter (maske/kjerne):** Naboskap + operator



Filter-egenskap: Additivt

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

f_1 og f_2 er vilkårlige bilder, $(f_1 + f_2)(x, y)$ betyr addisjon mellom piksler innad i naboskap og T er operatoren Nyttig å vite når vi

skal addere to filtrerte bilder. Nok å først addere bildene før filtrering, så filtrere!

Filter-egenskap: Homogent

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

f er et vilkårlig bilde, a vilkårlig konstant og T er operatoren

Vi kan enten multiplisere med en skalar før eller filtrering og fortsatt få samme resultat

Filter-egenskap: Lineær

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

f_1 og f_2 er vilkårlige bilder, a og b vilkårlige konstanter og T er operatoren

Additiv + Homogen = Lineær

Filter-egenskap: Posisjons-invariant

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

f er et vilkårlig bilde, l, m vilkårlige skift i posisjon og T er operatoren

Utbildets verdier i (x, y) er kun avhengig av innbildets *pikselverdier* i naboskapet om (x, y), og ikke posisjonene!

Lavpassfiltre

- Filtre som slipper gjennom *lave frekvenser* og demper/fjerne høye frekvenser
  - **Lave frekvenser:** Trege variasjoner, store trender, myke overganger
  - **Høye frekvenser:** Skarpe kanter, støy detaljer
  - ... dette kommer vi tilbake til i Fourier-forelesningene!
- Effekt fra lavpassfiltre: Glatting/utsmøring/"blurring" av bildet
- Typiske mål: Fjerne støy, finne større objekter

Lavpassfiltre

- Filtre som slipper gjennom *lave frekvenser* og demper/fjerne høye frekvenser
  - **Lave frekvenser:** Trege variasjoner, store trender, myke overganger
  - **Høye frekvenser:** Skarpe kanter, støy detaljer
  - ... dette kommer vi tilbake til i Fourier-forelesningene!
- Effekt fra lavpassfiltre: Glatting/utsmøring/"blurring" av bildet
- Typiske mål: Fjerne støy, finne større objekter
- Utfordring: Vi ønsker å bevare kanter!

Lavpassfilter: Middelverdifilter

- Beregner middelverdien i naboskapet
  - Alle vektene er like
  - Vektene summerer seg til 1 (lokal middelverdi bevares)
- Størrelsen på filteret avgjør graden av glatting
  - Stort filter: Mye glatting (og derav utsmørt bilde)
  - Lite filter: lite glatting, men kanter bevares bedre

3×3 middelverdifilter

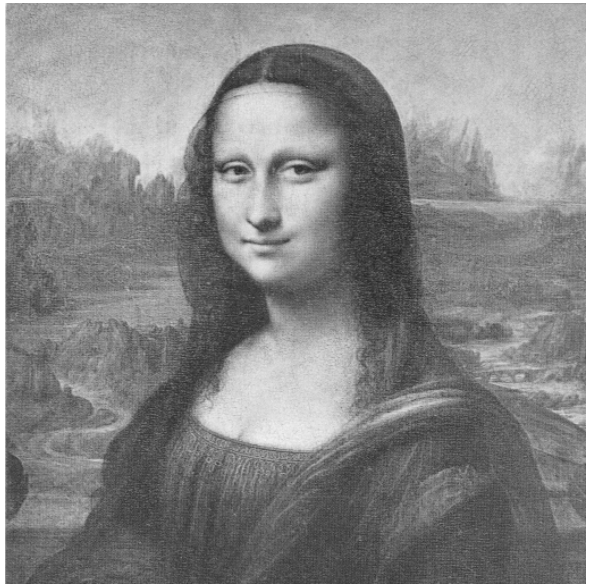
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

4×4 middelverdifilter

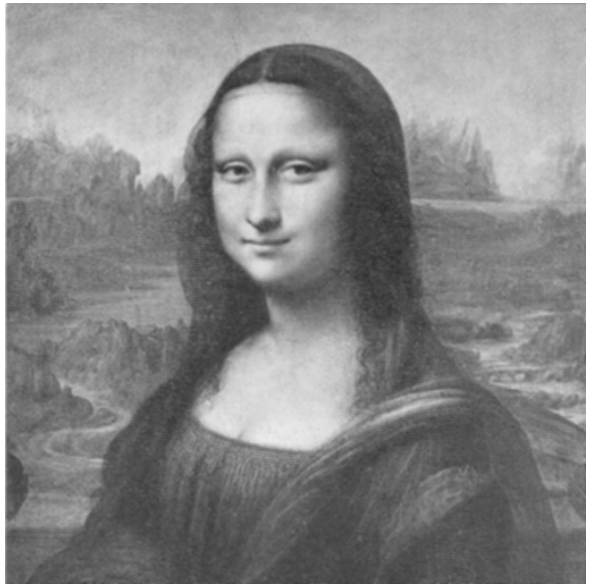
$$\frac{1}{16}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Lavpassfilter: Middelverdifilter, eksempel 1

Original



Filtrert med 3×3 middelverdifilter



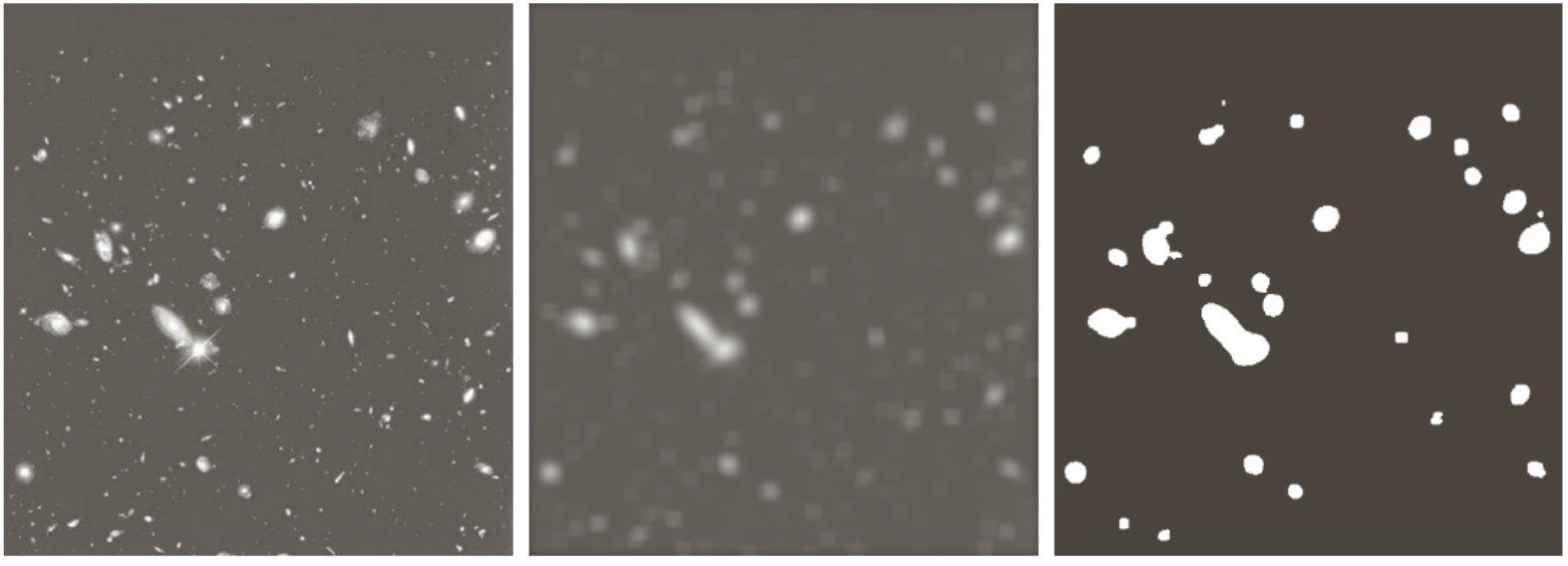
Filtrert med 9×9 middelverdifilter

Filtrert med 25×25 middelverdifilter

Lavpassfilter: Middelverdifilter, eksempel 2

Mål: Finne store objekter

Mulig løsning: 15×15-middelverdifiltrering og terskling



a b c

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

Separable filtre

- **Separabelt filter:** Filtreringen kan utføres som to sekvensielle 1D-filtreringer
- Fordel: Raskere filtrering!
- Middelverdi filtre er separable!
  Eksempel på 5×5 middelverdi filter:

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

- En 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

Konvolusjon ved oppdatering

Konvolusjons med identiske kolonner eller rader kan effektivt implementeres ved oppdatering:

1. Beregn responsen R for første piksel på posisjon (x, y)
2. Beregn responsen i neste piksel R_{ny} med utgangspunkt i R:
   - For identiske kolonner: Neste piksel er én til høyre, $(x, y+1)$ og ny respons er gitt ved:

     $$R_{\text{ny}} = R - C_1 + C_n$$

     det C_1 er responsen for første kolonne når filteret er plassert i (x, y) og C_n er responsen ved siste kolonne når filtrert er plasser i $(x, y+1)$
   - For identiske rader: Neste piksel er én ned, $(x+1, y)$ og ny respons er gitt ved :

     $$R_{\text{ny}} = R - R_1 + R_n$$

     der R_1 er responsen i første rad når filteret er plassert i (x, y) og R_n er responsen i siste rad når den er plassert i $(x+1, y)$.
3. La neste piksel være (x, y) og R_{ny} bli R.
4. Gjenta steg 2 og steg 3 frem til det ikke er fler piksler i innbildet å regne ut respons til.

Konvolusjon ved oppdatering

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

Lavpassfilter: Gauss-filter

- For heltallsverdier av x og y, er Gauss-filteret definert ved

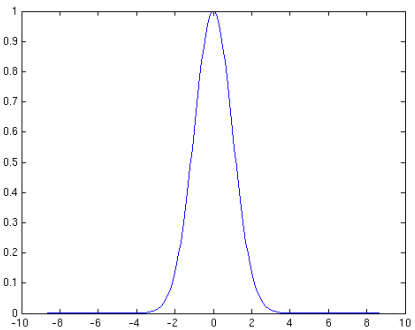
$$h(x, y) = A \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right),$$

  der A settes slik at summen av vektene blir 1
- Ikke-uniformt lavpassfilter

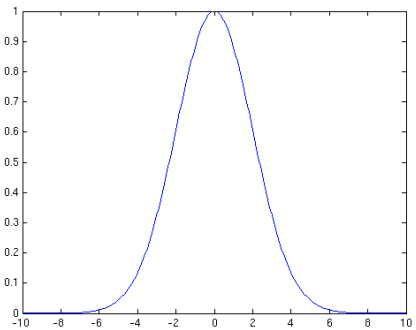
Lavpassfilter: Gauss-filter

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

- Parameteren σ er standardavviket og avgjør graden av glatting:
  - Liten σ gir lite glatting
  - Stor σ gir mye glatting
- Filterstørrelsen $n \times n$ tilpasses ofte σ
- Et Gauss-filter glatter mindre enn et middelverdifilter av samme størrelse



$\sigma = 1$



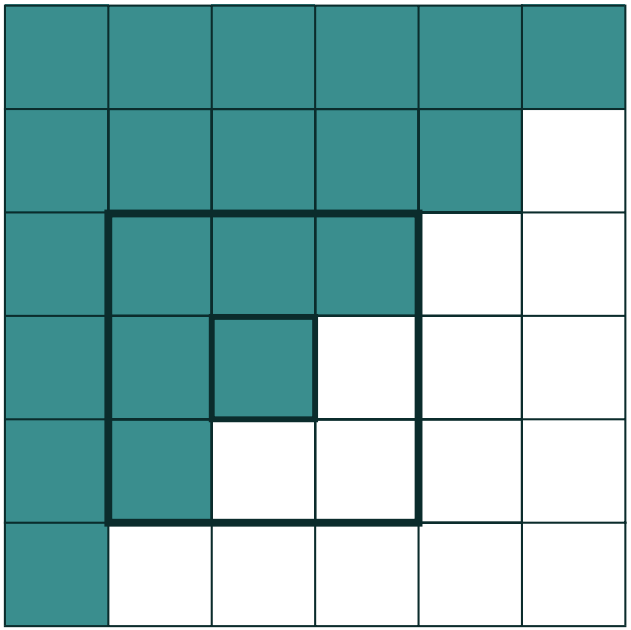
$\sigma = 2$

Approksimasjon av et 3×3 Gauss-filter

$$
\begin{aligned}
G_{3\times3} &= \frac{1}{16}\begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} * \begin{bmatrix}1\\1\end{bmatrix} * \begin{bmatrix}1 & 1\end{bmatrix} \\
&= \frac{1}{16}\begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} \\
&= \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix} \\
&= \frac{1}{16}\begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\end{bmatrix}
\end{aligned}
$$

Kant-bevarende støyfiltrering

- Ofte lavpassfiltrerer vi for å fjerne støy, men ønsker samtidig å bevare kanter
- Finnes mange kantbevarende filtre
- Vi kan lage filtre som sorterer pikslene:
  - Radiometrisk (etter pikselverdi)
  - Geometrisk (posisjon) og radiometrisk
- Bruker dette til å håndtere eventuelt flere populasjoner innad et naboskap:



Lavpassfilter: Rang-filtrering

- Lager en 1D-liste av alle pikslene i naboskapet om (x, y)
- Listen sorteres i stigende rekkefølge
- Responsen i (x, y) er åikselverdien i en bestemt posisjon i den sorterte listen, eller en veiet sum av en del av listen
- Dette er et **ikke-lineært** filter

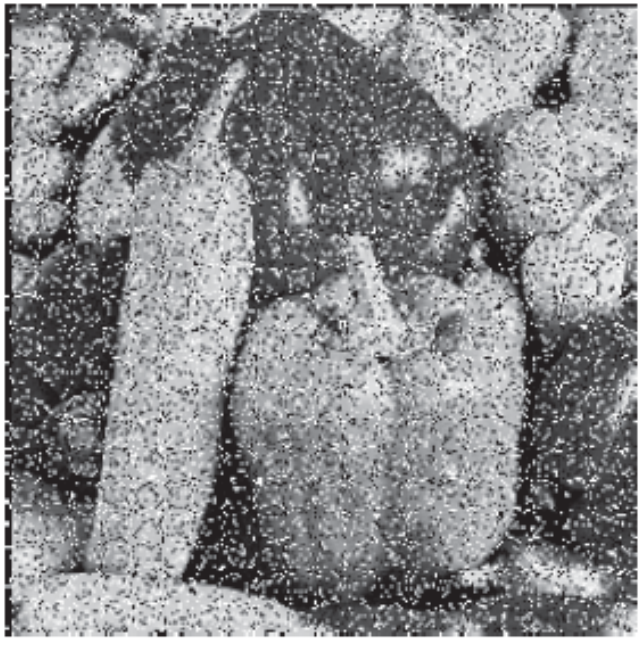
Lavpassfilter: Median-filter

$$g(x, y) = \text{median}[\text{pikselverdier i naboskapet rundt } (x, y)]$$

- **Median:** Midterste veriden i den sorterte listen
  $\rightarrow$ Median filteret er et rang-filter der vi velger midterste posisjon i den sorterte 1D-listen
- Et av de mest brukte kant-bevarende støyfiltre
- Spesielt god mot impuls-støy (”salt-og-pepper” støy
- Vanlig med ikke-rektangulære naboskap, f.eks pluss-formede naboskap
- Utfordringer:
  - Tynne linjer kan forsvinne
  - Hjørner kan rundes av
  - Objekter kan bli litt mindre
- Valg av størrelse og form på naboskapet er også viktig!

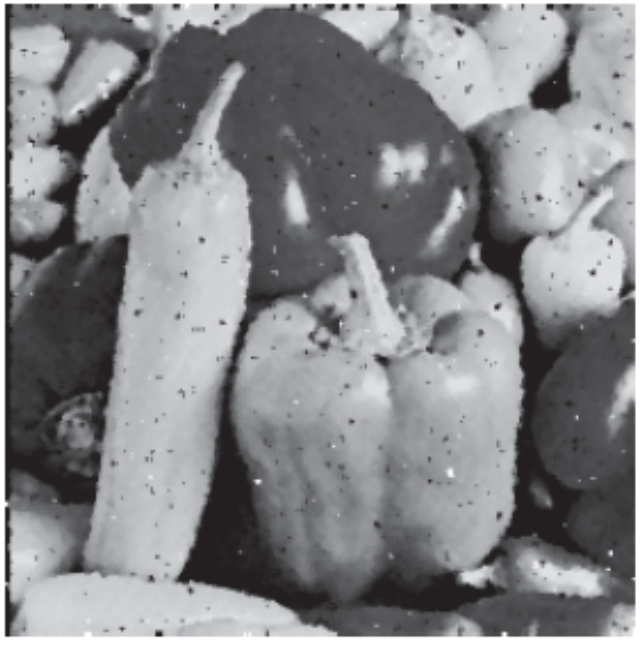
Middelverdi eller median?

Innbilde med salt-pepper-støy



Etter middelverdifiltrering

Etter medianfiltrering

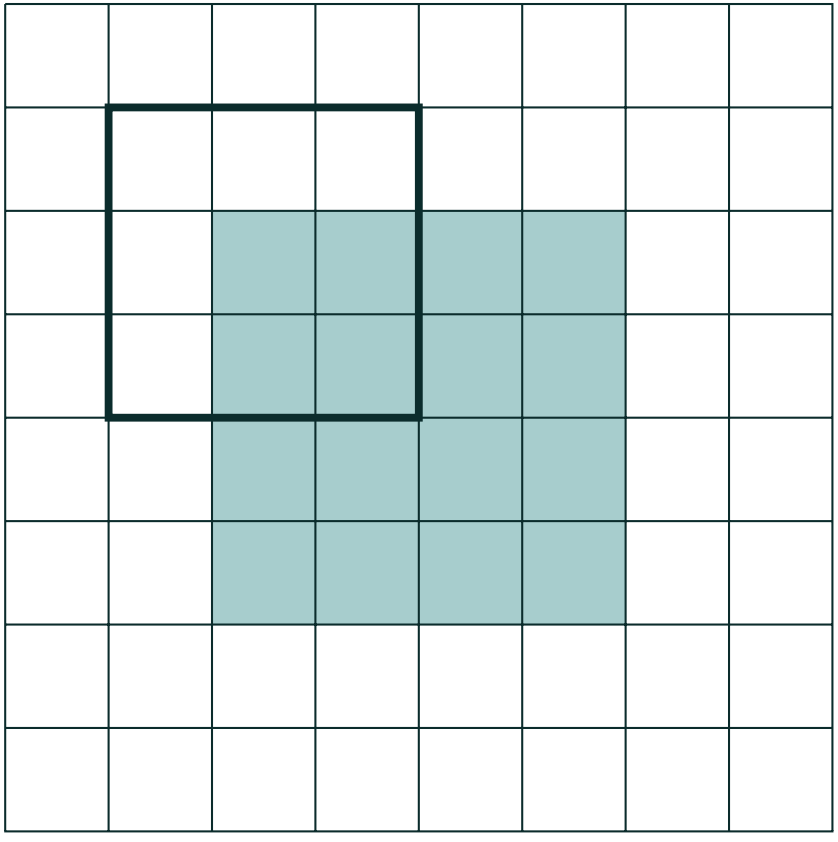


Middelverdi eller median?

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

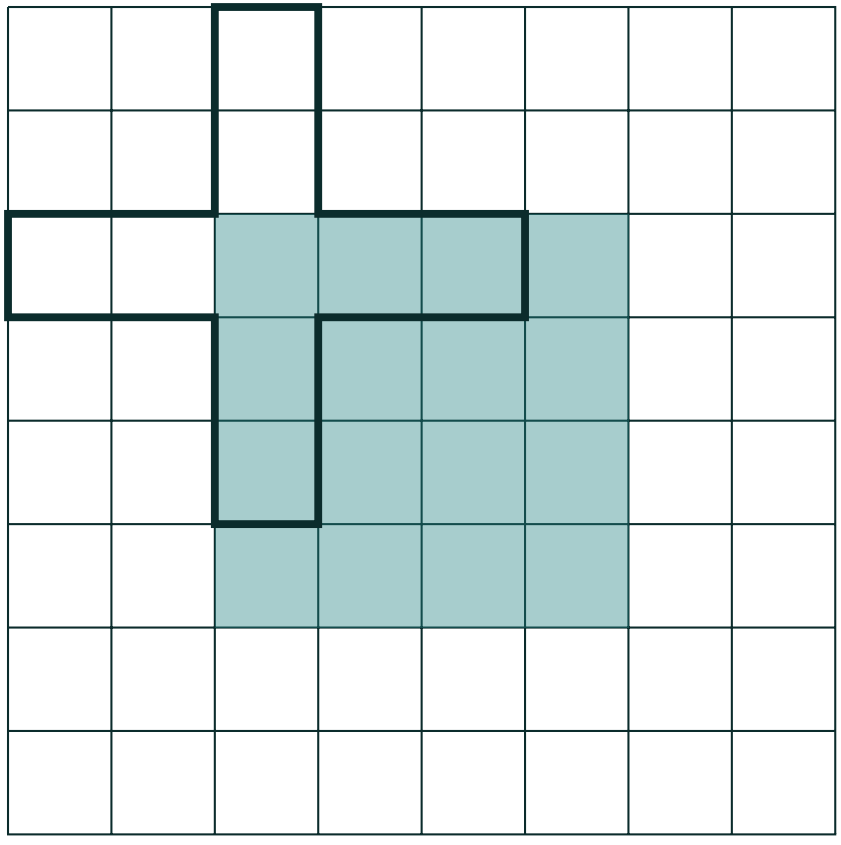
Medianfiltrering og hjørner

Kvadratisk naboskap:



Hjørnene avrundes

Pluss-formet naboskap:



Hjørnene bevares

Lavpassfilter: Trimmet middelverdifilter

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

$S_{(x,y)}$ er pikselposisjonene til de $mn-d$ midterste pikselverdiene etter sortering

- $g(x,y)$ er middelverdien av de $mn-d$ midterste verdiene (etter sortering) i $m \times n$ naboskapet rundt (x,y).
- Godt egnet til flere typer støy, f.eks salt-pepper støy og lokale variasjoner
- Hva blir filteret ved:
  - $d = 0$?
  - $d = mn-1$?

Lavpassfilter: Trimmet middelverdifilter

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

$S_{(x,y)}$ er pikselposisjonene til de $mn-d$ midterste pikselverdiene etter sortering

- $g(x,y)$ er middelverdien av de $mn-d$ midterste verdiene (etter sortering) i $m \times n$ naboskapet rundt (x,y).
- Godt egnet til flere typer støy, f.eks salt-pepper støy og lokale variasjoner
- Hva blir filteret ved:
  - $d = 0$ (middelverdi)
  - $d = mn-1$ (median)

Lavpassfilter: K nearest neighbour-filter

$$g(x,y) = \text{middelverdi}\big[\, K \text{ pikslene i naboskapet rundt } (x,y) \text{ som ligner mest på } (x,y) \text{ i pikselverdi}\big] \,,$$

"ligner mest" i absolutt-differanse

- Er et trimmet middelverdi filter; Middelverdien av de K mest like nabopikslene
- Utfordring; K er konstant for hele bildet
  - For liten K fjerner for lite støy
  - For stor K fjerner 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

Lavpassfilter: K Nearest Connected Neighbour-filter

- Naboskapet er hele bildet
- Responsen i (x, y) beregnes ved:
  1. Valgt piksel ved posisjon (x, y)
  2. Initialiser en tom liste
  3. While (antall valgte piksler) $< K$:
     - 3.1 Legg til (4- eller 8-)naboene til sist valgte piksel i listen
     - 3.2 Sorter listen på pikselverdi
     - 3.3 Velg pikselen i listen som er mest lik (x, y) og fjern den fra listen
  4. Beregn middelverdien av de K valgte pikslene
- Tynne linjer, hjørner og kanter blir bevart dersom K er mindre eller lik antall piksler i objektet

Lavpassfilter: Minimal-Mean-Square-Error (MMSE)

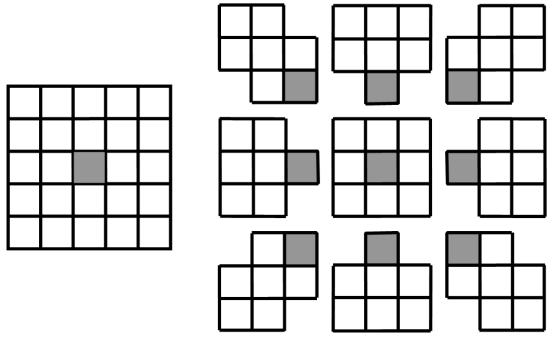
- For et naboskap rundt (x, y) kan vi beregne lokal varians $\sigma^2(x, y)$ og lokal middelverdi $\mu(x, y)$
- Anta vi har et estimat på støy-variansen σ_μ^2, MMSE-responsen i (x, y) er:

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

- I homogene områder blir responsen nær $\mu(x, y)$ (er $\sigma_\mu^2 > \sigma^2$, så settes $g(x, y) = \mu(x, y)$)
- Nær en kant vil $\sigma^2(x, y)$ være større enn σ_μ^2 og resulatet nær $f(x, y)$

Lavpassfilter: Maks-homogenitet

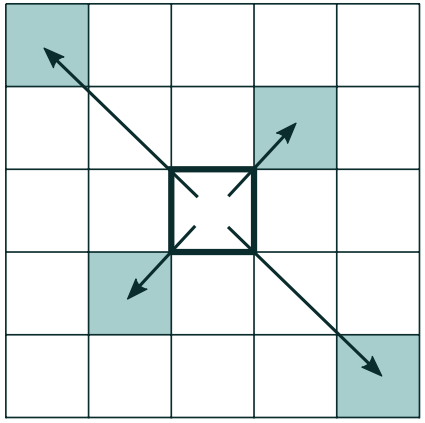
- Ønsker et kantbevarende filter
- Trikset er å dele opp et naboskap inn i flere, overlappende del-naboskap!
  - Alle del-naboskap inneholder senterpikselen
  - Flere mulige oppdelinger



- Det mest homogene del-naboskapet inneholder minst kanter;
  - Beregn μ og σ i hver del-naboskap
  - La $g(x, y)$ være lik μ fra det del-naboskapet som gir lavest σ (evnt $\left|\max(\text{pikselverdi inni naboskap}) - \min(\text{pikselverdi inni naboskap})\right|$)

Lavpassfilter: Symmetrisk Nærmeste Nabo (SNN)

- For hvert symmetrisk pikselpar i naboskapet rundt (x, y): Velg pikselen som ligner mest på (x, y) i pikselverdi



- Responsen blir:

$$g(x, y) = \text{middelverdi}[\text{av de valgte pikslene}]$$

Lavpassfilter: Mode-filter (kursorisk)

$g(x, y) =$ hyppigst forekommende pikselverdi i naboskapet rundt (x, y)

- Antall pikselverdier bør være lite i forhold til antall piksler i naboskapet
  - Brukes derfor sjeldent på gråtonebilde
  - Anvendes mest på segmenterte eller klassifiserte bilder for å fjerne isolerte piksler
- Kan implementeres gjennom histogram-oppdatering

Oppsummering

- Romlig filter = Naboskap + Operator,
  operatoren definerer hvordan innbildet endres
- Konvolusjon er lineær romlig filtrering
  - Konvolusjonsfilteret er en matrise av vakter
  - Å kunne utføre konvolusjon for hånd på et lite eksempel står sentralt i pensum!
  - Å programmere konvolusjon er sentralt i oblig 1!
- Korrelasjon og korrelasjonskoeffsienten kan brukes til mønstergjenkjenning
- Lavpassfiltrering kan fjerne støy

Lavpassfiltre i denne forelesningen

Navn	Kommentar
Middelverdifilter	Uniformt konvolusjonsfilter
Gauss-filter	Konvolusjon med geometrisk vekting
Rang-filter	Bevarer kanter bedre enn middelverdifilteret (f.eks median)
Trimmet middelverdi-filter	Mellomting mellom middelverdi- og medianfilter
K Nearest neighbor filter	K må velges med omhu
K Nearest Connected Neighbor filter	Pikslene må være geometriske naboer
MinimalMeanSquareError filter	Bruker at lokal varians er større i nærheten av en kant
Maks-homogenitet	Sub-naboskaå tar i bruk geometrien
Symmetrisk Nærmeste Nabo filter	Symmetrisk parring som tar i bruk geometrien
Mode-filter	Hyppigst forekommende pikselverdi