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

Forelesningsplan

	14 15 16 17 18 19 20		
Januar	21 22 23 24 25 26 27	Introduksjon	Fritz/Andreas
	28 29 30 31 01 02 03	Sampling og kvantisering	Fritz
Februar	04 05 06 07 08 09 10	Geometriske operasjoner	Fritz
	11 12 13 14 15 16 17	Gråtonemapping	Fritz
	18 19 20 21 22 23 24	Histogrambaserte operasjoner	Fritz
	25 26 27 28 01 02 03	Naboskaps-operasjoner I	Andreas
Mars	04 05 06 07 08 09 10	Naboskaps-operasjoner II	Andreas
	11 12 13 14 15 16 17	Midtveis-repetisjon	Fritz/Andreas
	18 19 20 21 22 23 24	**Midtveis-eksamen, 19.03.2013**	
	25 26 27 28 29 30 31	*Påske, ingen forelesning*	
April	01 02 03 04 05 06 07	*Påske, ingen forelesning*	
	08 09 10 11 12 13 14	Fourier I	Andreas
	15 16 17 18 19 20 21	Fourier II	Andreas
	22 23 24 25 26 27 28	Kompresjon og koding I	Andreas
Mai	29 30 01 02 03 04 05	Kompresjon og koding II	Andreas
	06 07 08 09 10 11 12	Segmentering	Fritz
	13 14 15 16 17 18 19	Morfologi	Andreas
	20 21 22 23 24 25 26	Fargerom og fargebilder	Fritz
	27 28 29 30 31 01 02	Repetisjon	Fritz/Andreas
Juni	03 04 05	**Eksamen, 04.06.2013 (4 timer)**	

Filtrering

- Vi skal se på filtrering i bildedomenet (eng.: *spatial domain*).
  - Etter påske tar vi filtrering i (det diskrete) Fourier-domenet.
- **Bildedomenet**: Domenet der et digital bilde er representert som en matrise av piksler.
  - Eksempel:

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

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

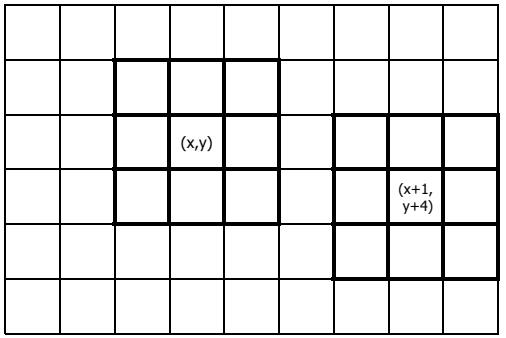
Ut-bildet g

Bruksområder

- 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

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)*:

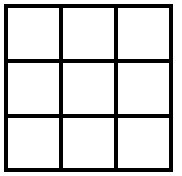


Naboskap: I praksis

- Kvadrater/rektangler er mest vanlig.
  - Av symmetrihensyn er bredden/høyden oftest odde.
    - Da er naboskapets senterpunkt i et piksel.
    - Når annet ikke er spesifisert så er senterpunktet naboskapets **origo**.
- Spesialtilfelle: Naboskapet er 1×1:
  - *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 1×1, 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.
- Romlig filter: Naboskap + Operator
  - Også kalt *maske*, *kjerne* og bare *filter*.
    - og *vindu*, men vi vil bare bruker det om et (muligens ikke-rektangulært) delbilde.
  - Mange filtre kan representeres som en **matrise av vekter eller koeffisienter**.



3x3-naboskap

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

3x3-middelverdifilter: Glatter/utsmører/«blurrer» inn-bildet

Filter-egenskap: Additivt?

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

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

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

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

Filter-egenskap: 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 et vilkårlig piksel

- Hva betyr dette:
  - Hvis vi skalerer et bilde før filtrering?

Filter-egenskap: 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 et 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 et 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**.

1D-konvolusjon

- Konvolusjon av et 1D-filter *h* og et 1D-bilde *f*:

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

evaluert for alle *x* slik at hver verdi av *h* overlapper hver verdi av *f*.

For å forenkle notasjonen, antar denne formelen at:
- *h* har odde lengde, $m = 2a+1$.
- Senterpunktet er naboskapets origo.

Konvolusjon krever ikke disse antagelsene.

- Konvolusjon er altså en **lineær filtrering**.

$$g(x) = T(f(x)) = (h * f)(x)$$

  - Også posisjons-invariant og uten konstantledd.

- ***h* kan spesifiseres som en tabell**!
  - En slik tabell kaller vi et **konvolusjonsfilter**.

1/3	1/3	1/3

3-middelverdifilter

1D-konvolusjons-eksempel

Oppgave: Beregn konvolusjonen av følgende 1D-filter *h* og 1D-bilde *f*:

```
h = [ 1  2  3]
Indeks for h:      -1  0  1
f = [ 6  4  5]
Indeks for f:       0  1  2
```

1. Speile bildet *f*.
2. Føre det speilvendte 1D-bildet *f* over *h*.
3. For hver posisjon *x* med overlapp i minst ett piksel, beregn *g(x)* som summen av produktene av de overlappende pikslene.

Løsning: 1D-konvolusjon: $g(x) = \sum_{s=-a}^{a} h(s) f(x-s)$ uttrykt med ord

× angir multiplikasjon.

```
g(-1) = h(-1)f(-1-(-1)) + h(0)f(-1-0) + h(1)f(-1-1)
      = h(-1)f(0) + h(0)f(-1) + h(1)f(-2) = 1×6    + 2×f(-1) + 1×f(-2) = 6
g(0)  = h(-1)f(1) + h(0)f(0)  + h(1)f(-1) = 1×4    + 2×6     + 1×f(-1) = 16
g(1)  = h(-1)f(2) + h(0)f(1)  + h(1)f(0)  = 1×5    + 2×4     + 3×6     = 31
g(2)  = h(-1)f(3) + h(0)f(2)  + h(1)f(1)  = 1×f(3) + 2×5     + 3×4     = 22
g(3)  = h(-1)f(4) + h(0)f(3)  + h(1)f(2)  = 3×f(4) + 2×f(3)  + 3×5     = 15
Altså er ut-bildet g = [ 6 16 31 22 15]
    Indeks for g:      -1  0  1  2  3
```

f(-2), f(-1), f(3) og f(4) er ikke definert, vi antar her at de er 0.

Beregningsformel for 1D-konvolusjon

- Merk at:

$$\begin{aligned}(h * f)(x) &= \sum_{s=-a}^{a} h(s) f(x-s) \\ &= h(-a) f(x-(-a)) \\ &\quad + h(-a+1) f(x-(-a+1)) \\ &\quad \vdots \\ &\quad + h(a-1) f(x-a+1) \\ &\quad + h(a) f(x-a) \\ &= \sum_{s=x-a}^{x+a} h(x-s) f(s)\end{aligned}$$

- Den siste formen brukes normalt for utregning.
  - Tilsvarer å speile filteret i stedet for 1D-bildet og føre filteret over bildet (i stedet for føre 1D-bildet over filteret).

Utregning av 1D-konvolusjon

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

- For å regne ut resultatet av en konvolusjon for posisjon *x* (kalt **responsen i *x***):
  1. Speilvend konvolusjonsfilteret og legg den over 1D-bildet slik at konvolusjonsfilteret origo overlapper posisjon *x* i 1D-bildet.
  2. Multipliser hver vekt i det speilede konvolusjonsfilteret med underliggende pikselverdi.
  3. Summen av produktene gir verdien for *g(x)* i posisjon *x*.
- For å regne ut responsen i alle posisjoner:
  - Flytt konvolusjonsfilteret over 1D-bildet og beregn responsen for hver posisjon med overlapp.
- Merk: Vi trenger ikke speilvende symmetriske konvolusjonsfiltre.

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-konvolusjon

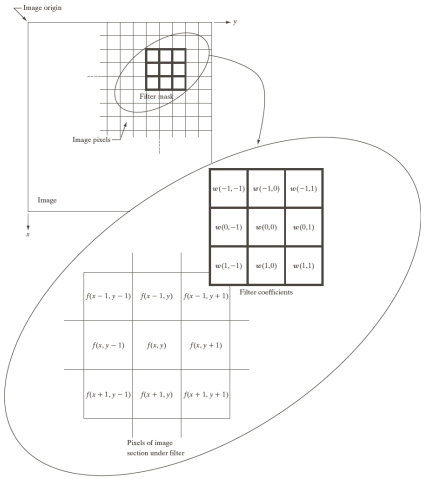


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

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.

1/9	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

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

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.

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

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

0,1	0,4				

Inn-bilde f Foreløpig ut-bildet g

2D-konvolusjons-eksempel

... tretten steg 3 senere:

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

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

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

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		

Inn-bilde f 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

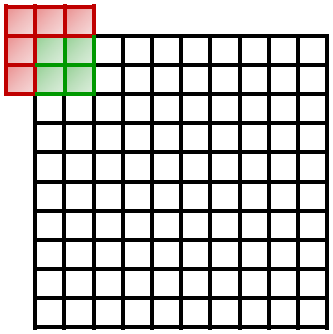
Utregning av 2D-konvolusjon

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

- For å regne ut responsen i posisjon (x,y):
  1. Roter konvolusjonsfilteret 180 grader og legg den over bildet slik at origo overlapper posisjon (x,y) i bildet.
  2. Multipliser hver vekt i det roterte konvolusjonsfilteret med underliggende pikselverdi.
  3. Summen av produktene gir verdien for $g(x,y)$ i posisjon (x,y).
- For å regne ut responsen i alle posisjoner:
  - Flytt konvolusjonsfilteret over bildet og beregn responsen for hver posisjon med overlapp.
- Merk: Vi trenger ikke speilvende symmetriske konvolusjonsfiltre.

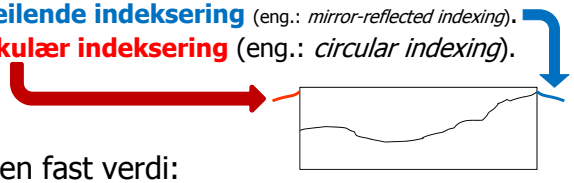
Filtrering: Praktiske problemer

- 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 x N piksler.
  - Anta at filteret er m x n.
    - (og at m og n er odde)
  - Uberørt av **bilderandproblemet**: (M-m+1)x(N-n+1)
    - 3x3-filter: (M-2)x(N-2)
    - 5x5-filter: (M-4)x(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).
- Ignorer posisjonene uten overlapp.
  - Identisk resultat som nullutvidelse for konvolusjonsfiltre.

Utvide inn-bildet?

- Konvolusjons-definisjonen sier at ut-bildet får verdi når filteret og inn-bildet overlapper i minst ett piksel.
- Dette tilsvarer å utvide inn-bildets størrelse.
- 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.
- Merk: Dette gjelder all filtrering, ikke bare konvolusjon!

2D-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 verken trenger å rotere filteret eller bildet!
- Legger konvolusjonsfilterets 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.
- Mønsteret/templatet kan være en del av bildet.
- 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 vekting av lyse piksler.

Mønstergjenkjennings-eksempel

- **Oppgave**: Finn et bestemt objekt i et bilde.

- Templaten må ha samme størrelse og orientering som objektet i bildet (og omtrent samme gråtoner).

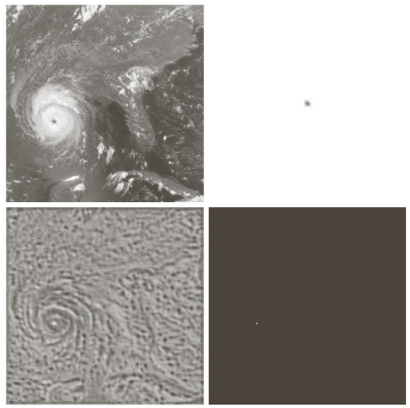
Korrelasjonskoeffisient

- Mønstergjenkjenning kan bedre gjøres ved bruk av *korrelasjonskoeffisient*:

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

  - Telleren er en middelverdi-normalisert korrelasjon
    - Templatets er normalisert med sin (globale) middelverdi.
    - Bildet er normalisert med middelverdien av pikslene der templatet og bildet overlapper (lokal middelverdi).
  - Nevneren normaliserer variansen
    - Templatets standardavvik ganger bildets lokale 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?

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!

Egenskapene:
- Gjelder generelt bare når man antar **nullutvidelse**.
- Gjelder generelt **ikke korrelasjon**.

Lavpassfiltre

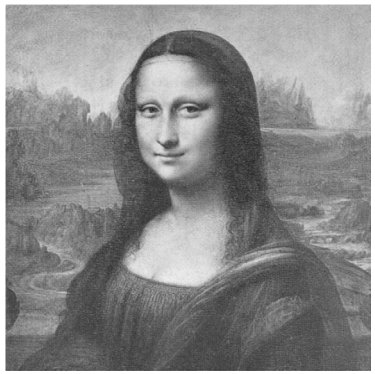
- Slipper gjennom lave frekvenser og demper eller fjerner høye frekvenser.
  - Lav frekvenser = trege variasjoner, store trender.
  - Høye frekvenser = skarpe kanter, støy, detaljer.
  - … mye mer om frekvens i Fourier-forelesningene.
- Effekt: Glatting/utsmøring/«blurring» av bildet.
- Typiske mål: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

Middelverdifilter (lavpass)

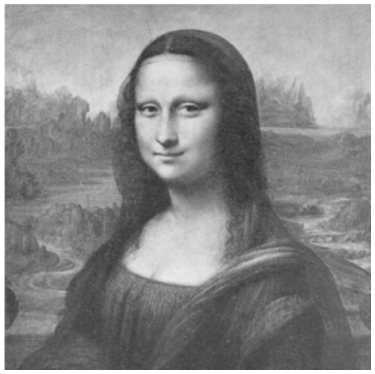
- Beregner middelverdien i naboskapet.
  - Alle vektene er like.
  - Vektene summerer seg til 1.
    - Gjør at den lokale gjennomsnittsverdien bevares.
- Størrelsen på filteret avgjør graden av glatting.
  - Stort filter: mye glatting (utsmørt bilde).
  - Lite filter: lite glatting, men kanter bevares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

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

Middelverdifilter-eksempel



Original



Filtrert med 3x3-middelverdifilter

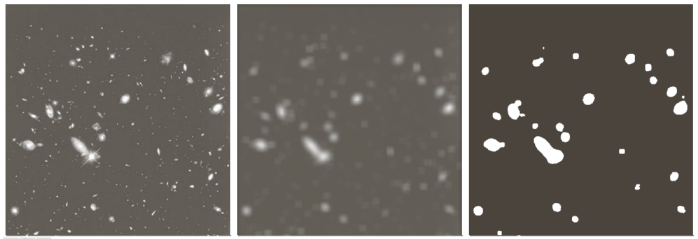
Middelverdifilter-eksempel

Filtrert med 9x9-middelverdifilter

Filtrert med 25x25-middelverdifilter

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».



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

- 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.

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	*Δ*	*Δ′*
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

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

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

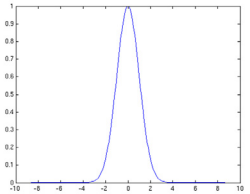
- C_1, C_n, R_1 eller R_n beregnes ved n multiplikasjoner og n-1 addisjoner.
  - Tar totalt $2n$ multiplikasjoner og $2(n-1)$ addisjoner å finne R_{ny}.
- Like raskt som separabilitet.
  - Sett bort fra initieringen; finne R for hver ny rad eller kolonne.
  - Alle «oppdaterbare» konvolusjonsfiltre er 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 skaleringer av middelverdifiltre er uniforme filtre.

Gauss-filter (lavpass)

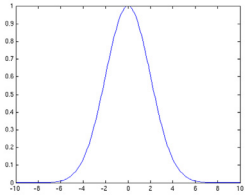
- For heltallsverdier av x og y, la:

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

  - A settes slik at summen av vektene blir 1.
  - $N(0,\sigma^2 I_2)$ med alternativ skalering A.
- **Ikke-uniformt lavpassfilter**.
- Parameteren σ er standardavviket og avgjør graden av glatting.
  - Lite σ: lite glatting.
  - Stort σ: mye glatting.
- Filterstørrelsen $n \times n$ må tilpasses σ.
  - Et 2D-Gauss-filter glatter *mindre* enn et middelverdifilter av samme størrelse.



$\sigma = 1$



$\sigma = 2$

Approksimasjon av 3x3-Gauss-filter

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

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

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

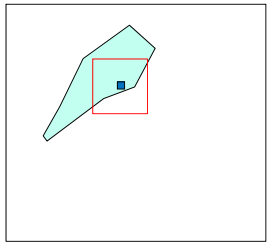
$$= \begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

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

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.

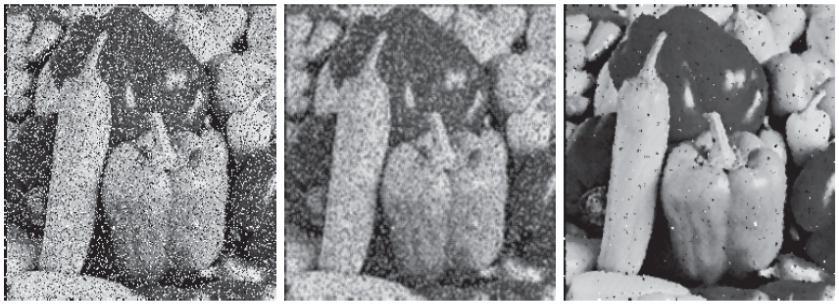
Median-filter (lavpass)

- *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-rektangulære naboskap, f.eks. pluss, ikke uvanlig.
- Problemer:
  - Tynne kanter kan forsvinne.
  - Hjørner kan rundes av.
  - Objekter kan bli litt mindre.
- Valg av størrelse og form på naboskapet er viktig!

Middelverdi eller median?

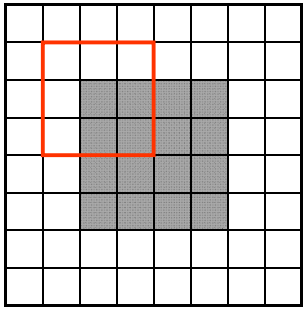
- Middelverdifilter: Middelverdien innenfor naboskapet.
  - Glatter lokale variasjoner og støy, men også kanter.
  - Spesielt god på lokale variasjoner, som kan være mild støy i mange pikselverdier.
- Medianfilter: Medianen innenfor naboskapet.
  - Bedre på visse typer støy og til å bevare kanter, men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

Middelverdi eller median?

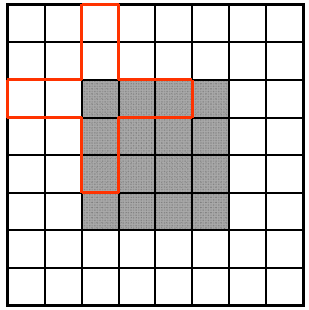


Inn-bildet med tydelig salt-og-pepper-støy | Etter middelverdifiltrering | Etter medianfiltrering

Medianfiltrering og hjørner



Med kvadratisk naboskap
avrundes hjørnene



Med pluss-formet naboskap
bevares hjørnene

Raskere medianfiltrering (kursorisk)

- Å sortere pikselverdiene innenfor naboskapet er tungt.
- Med bit-logikk kan medianen av *N* verdier finnes i *O(N)*.
  - Avhenger også av antall biter i representasjonen av en verdi.
  - For *n*x*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 *n*x*n*-naboskap.
  - Avhenger også av antall mulige pikselverdier *eller* middelverdien av absolutt-differansen 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 (gjenbruker 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.

Alpha-trimmet middelverdifilter (lavpass)

- *g(x,y)* = middelverdien av de *mn-d* midterste verdiene (etter sortering) i *m*x*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.

K Nearest Neighbour-filter (lavpass)

- *g(x,y)* = middelverdien av de *K* pikslene i naboskapet rundt *(x,y)* som ligner mest på *(x,y)* i pikselverdi.
  - «Ligner mest» = Lavest absoluttdifferanse.
- Er et trimmet middelverdifilter: Middelverdien 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*×*n*-naboskap der *n=2a+1*:
- K=1: ingen effekt
- K≤n: bevarer tynne linjer
- K≤(a+1)²: bevarer hjørner
- K≤(a+1)n: bevarer rette kanter

K Nearest Connected Neighbour-filtrering (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.
    - Sorter listen på pikselverdi.
    - Velg pikselen i listen som er mest lik (x,y) og fjern den fra listen.
  - Beregn middelverdien 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 middelverdi $\mu(x,y)$.
- Anta at vi har et estimat på støy-variansen, σ_η^2
- Responsen i *(x,y)* er da:

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

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

Sigma-filter (lavpass) (kursorisk)

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

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

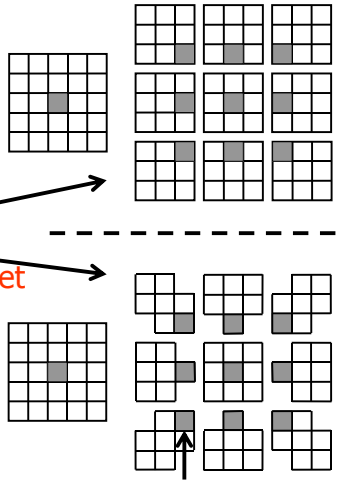
- Alternativt: Erstatte σ med **lokal** *median absolute deviation* (MAD):

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

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

Max-homogenitet (lavpass)

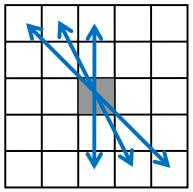
- Ønsker kantbevarende filter.
- ***Et enkelt triks:***
- Del opp naboskapet i flere, overlappende sub-naboskap.
  - Alle inneholder senterpikselet.
  - Flere mulige oppdelinger.
- Det mest homogene sub-naboskapet inneholder minst kanter.
  - Beregn μ og σ i hvert sub-naboskap.
  - La *(x,y)* være μ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



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

Symmetrisk nærmeste nabo (SNN) (lavpass)

- For hvert symmetrisk piksel-par i naboskapet rundt *(x,y)*:
  - Velg det pikselet som ligner mest på *(x,y)* i pikselverdi.



- *g(x,y)* = middelverdien av *(x,y)* og de valgte pikslene.
  - Antall verdier som midles v.b.a. *mxn*-naboskap: *1+(mn-1)/2*

Mode-filter (lavpass) (kursorisk)

- *g(x,y)* = hyppigst forekommende pikselverdi i naboskapet rundt *(x,y)*.
  - 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 vekter.
  - Å kunne utføre konvolusjon for hånd på et lite eksempel er sentralt i pensum!
  - Å programmere 2D-konvolusjon er sentralt i Oblig1.
- Lavpassfiltrering kan fjerne støy.
  - Median bevarer kanter bedre enn middelverdi.
  - Vi har sett flere ikke-lineære, kantbevarende filtre.