INF 2310 – Digital bildebehandling

FORELESNING 6

FILTRERING I BILDE-DOMÈNET – I

Fritz Albregtsen

Naboskaps-operasjoner
Konvolusjon og korrelasjon
Kant-bevarende filtre
Ikke-lineære filtre

GW Kap. 3.4 - 3.5 + Kap. 5.3

Lokale operasjoner

- Vi skal se på teknikker i bilde-domenet
  - Teknikker i frekvens-domenet kommer senere
- Bilde-domenet refererer til mengden av piksler som utgjør det digitale bildet.
- Bildeplan-metoder opererer på disse pikslene og gir:

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

g(x,y) er ut-bildet

f(x,y) er inn-bildet

T er en operator på en **omegn** rundt *(x,y)*

Bruksområder - filtrering

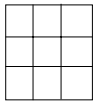
- Av de mest brukte operasjoner i bildebehandling
- Brukes som et ledd i pre-prosessering for mange bildeanalyse-problemer
  - Støyfjerning / støyreduksjon
  - Kant-deteksjon
  - Deteksjon av linjer eller andre spesielle strukturer

Omgivelser/naboskap/vindu

- Kvadrater/rektangler er mest vanlig.
- Av symmetri-hensyn oftest odde antall piksler.
- Omgivelsen kan ha flere dimensjoner *(x,y,z,λ,t)*
- Enklest transform får vi når omgivelsen er 1×1 piksel.
  - *T* er da gråtonemapping der ny pikselverdi bare avhenger av pikselverdien i punktet *(x,y).*
  - Hvis *T* er den samme over hele bildet, har vi en global transform.
- Hvis vinduet er større enn 1×1, har vi en lokal transform (selv om *T* er posisjons-invariant).

Naboskap+Transform=Maske

- Naboskap/vindu/omgivelse/neighborhood:
  - De pikslene rundt et punkt i inn-bildet som *T* opererer på.
- Transform/Operator/Algoritme:
  - Opererer på pikslene i et naboskap.
- Maske/Template/filer:
  - Et geometrisk array/matrise av vekter eller koeffisienter
- Resultatet lagres i et **nytt** bilde.



Eksempel:
beregn gjennomsnitt

1	1	1
1	1	1
1	1	1

Filter-egenskaper - Additivitet

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

H er filteret, og f_1 og f_2 er vilkårlige bilder.

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

Filter-egenskaper - Linearitet

$$H[af_1(x,y) + bf_2(x,y)] = aH[f_1(x,y)] + bH[f_2(x,y)]$$

H er filteret, a og b er konstanter, og f_1 og f_2 er vilkårlige bilder.

- Hva betyr dette:
  - Hvis vi skalerer bildene før filtreringen?

Filter-egenskaper - Homogenitet

$$H[af_1(x,y)] = aH[f_1(x,y)]]$$

- Hvis H er en lineær operator, er responsen på et konstant multippel av en vilkårlig input lik konstanten multiplisert med responsen på input.

Filter-egenskaper – Posisjons-invarians

$$H[f_1(x-l, y-m)] = g(x-l, y-m)$$

for alle *f(x,y)* og for vilkårlige *(l,m).*

- Responsen i et vilkårlig punkt i et bilde avhenger bare av bildets lokale verdi, ikke av posisjonen.

1-D Konvolusjon

$$g(x) = T[f(x)] = \sum_{i=-w}^{w} h(i) f(x-i)$$

- *h* er filteret, *f* er et 1-D bilde eller signal
- Merk at:

$$\begin{aligned} g(x) &= \sum_{i=-w}^{w} h(i) f(x-i) \\ &= h(-w) f(x-(-w)) \\ &+ h(-w+1) f(x-(-w+1) \\ &\vdots \\ &+ h(w-1) f(x-w+1) \\ &+ h(w) f(x-w) \\ &= \sum_{i=x-w}^{x+w} h(x-i) f(i) \end{aligned}$$

Og det er den siste formen vi bruker for utregning.

Et 1-D eksempel

h=[1 2 3]

Indeks for h: -1 0 1

f=[4 4 4]

Indeks for f: x=1 2 3

$$g(x) = \sum_{i=x-w}^{x+w} h(x-i) f(i)$$

g(1) = h(1)f(0) + h(0)f(1) + h(-1)f(2) = 3×f(0)+2×4+1×4 = 12
g(2) = h(1)f(1) + h(0)f(2) + h(-1)f(3) = 3×4 + 2×4 + 1×4 = 24
g(3) = h(1)f(2) + h(0)f(3) + h(-1)f(4) = 3×4 + 2×4 + 1×f(4)=20
f(0) og f(4) er ikke definert, la oss anta at de er 0.
× er multiplikasjon

Merk at vi må speilvende h før multiplikasjon.

Utregning av 1-D konvolusjon

$$g(x) = \sum_{i=-w}^{w} h(i) f(x-i) = \sum_{i=x-w}^{x+w} h(x-i) f(i)$$

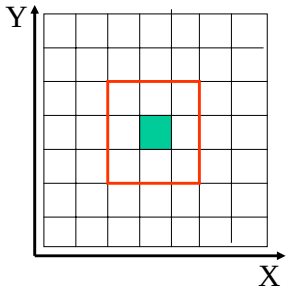
- For å regne ut resultatet av en konvolusjon for posisjon *x*:
  - Speilvend masken, legg den over bildet slik at minst en posisjon overlapper med bildet.
  - Multipliser hvert element i masken med underliggende pikselverdi. Summen av produktene gir verdien for *g(x)* i posisjon *x*.
- For å regne ut resultatet for alle posisjoner:
  - Flytt masken piksel for piksel og gjenta operasjonene over.
- Merk: for symmetrisk *h* spiller det ingen rolle om vi speilvender masken eller ikke.

2-D konvolusjon

- Ut-bildet er gitt ved

$$g(x,y) = \sum_{j=-w_1}^{w_1} \sum_{k=-w_2}^{w_2} h(j,k) f(x-j, y-k)$$

$$= \sum_{j=x-w_1}^{x+w_1} \sum_{k=y-w_2}^{y+w_2} h(x-j, y-k) f(j,k)$$

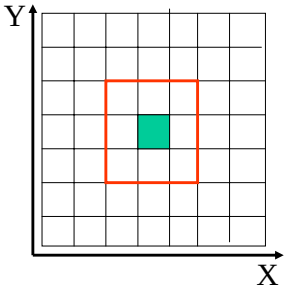


- h er et *m×n* filter: $m=2w_1+1$, $n=2w_2+1$
- *m* og *n* er vanligvis oddetall.
- Ofte har vi kvadratisk vindu *(m=n).*
- Ut-bildet er et veiet sum av inn-pikslene som omgir *(x,y).* Vektene i filteret er gitt ved *h(j,k).*
- Ut-bildets pikselverdi i neste posisjon finnes ved at filteret flyttes ett piksel, og summen beregnes på nytt.

Eksempel: middelverdi

- For hvert piksel, beregn middelverdien av pikslene i et 3×3 vindu rundt piksel *(x,y)*

$$g(x,y) = \frac{1}{9}[f(x-1,y-1) + f(x,y-1) + f(x+1,y-1)$$
$$+ f(x-1,y) + f(x,y) + f(x+1,y)$$
$$+ f(x-1,y+1) + f(x,y+1) + f(x+1,y+1)]$$
$$= \frac{1}{9} \sum_{v=-1}^{1} \sum_{w=-1}^{1} f(x-v, y-w)$$



f(x,y) og 3×3 vindu

- ***Merk orienteringen av X og Y !!!***

Utregning av 2-D konvolusjon

$$g(x,y) = \sum_{j=x-w_1}^{x+w_1} \sum_{k=y-w_2}^{y+w_2} h(x-j, y-k) f(j,k)$$

- For å regne ut resultatet av en konvolusjon for posisjon *(x,y)* :
  - Roter masken 180 grader, legg den over bildet slik at minst en posisjon overlapper med bildet.
  - Multipliser hvert element i masken med underliggende pikselverdi.
  - Summen av produktene gir verdien for *g(x,y)* i posisjon *(x,y).*
- For å regne ut resultatet for alle posisjoner:
  - Flytt masken piksel for piksel og gjenta operasjonene over.
- Vi bruker notasjonen

  g = h * f

  der * er konvolusjons-operatoren

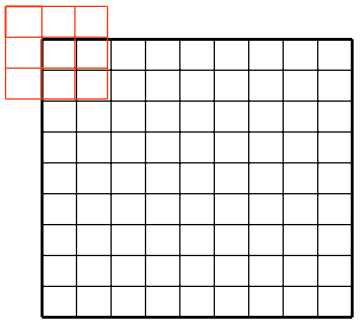
Praktiske problemer

- Kan ut-bildet ha samme piksel-representasjon som inn-bildet?
- Trenger vi et mellom-lager?
- Hva gjør vi langs bilde-randen?

- Anta at bildet er M x N piksler
- Anta at filteret er m x n ($m=2m_2+1$, $n=2n_2+1$)

- Uberørt av rand-effekt:
  $(M-2m_2)x(N-2n_2)$

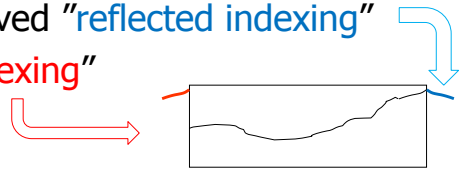
  3x3: (M-2)x(N-2)
  5x5: (M-4)x(N-4)



Hva gjør vi langs randen?

Alternativer:

1. Sett g(x,y)=0
2. Sett g(x,y)=f(x,y)
3. Trunkér ut-bildet
4. Trunkér konvolusjons-masken h
5. Utvid bildet ved "reflected indexing"
6. "Circular indexing"



Et lite tips om konvolusjon

- Når vi konvolverer et filter med et bilde:
  - Er vi interessert i å lage et nytt bilde med samme størrelse som input-bildet.
  - Vi bruker en av teknikkene fra forrige foil.
- Når vi konvolverer en filter-kjerne med en annen filter-kjerne:
  - Vi beregner resultatet for alle posisjoner der de to filter-kjernene gir overlapp.

Korrelasjon

$$g(x,y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j,k)\, f(x+j, y+k)$$

- Merk forskjell fra konvolusjon:
  NB! pluss i stedet for minus.
- Ved konvolusjon roterer vi filteret 180 grader.
- Ved korrelasjon trenger vi ikke dette:
  - vi legger filterkjernen sentrert om piksel *(x,y)* og multipliserer hvert enkelt element med underliggende pikselverdi
- Vi kan utføre korrelasjon som en konvolusjon, hvis vi først roterer filteret 180 grader.

Korrelasjon

$$g(x,y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j,k)\, f(x+j, y+k)$$

- Anvendelse: mønstergjenkjenning eller *template matching*.
- Mønster/template kan være en del av et bilde.
- h(i,j)≥0
- Normaliser ved

$$g'(x,y) = \frac{g(x,y)}{\sum_{j=-n_2}^{n_2} \sum_{k=-m_2}^{m_2} f(x+j, y+k)}$$

for å unngå høyere verdier for lyse piksler.

Korrelasjonskoeffisient

- Alternativt, beregn korrelasjons-koeffisienten

$$\eta(x,y) = \frac{\sum_i \sum_j \left[h(i,j) - \bar{h}\right]\left[f(x+i,y+j) - \bar{f}\right]}{\sqrt{\sum_i \sum_j \left[f(x+i,y+j) - \bar{f}\right]^2} \sqrt{\sum_i \sum_j \left[h(i,j) - \bar{h}^2\right]^2}}$$

- Denne er normalisert både i forhold til middelverdien til filteret, $\bar{h}$ og i forhold til middelverdien til den lokale omegnen i bildet, $\bar{f}(x,y)$

Eksempel – template matching

- Finn et objekt i et bilde.
- Filteret er templaten.
- Templaten må ha samme størrelse og orientering som objektet i bildet (og omtrent samme gråtoner).

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 !

Normalisering av filtre

- Hvis vi konvolverer [1 1 1] med seg selv, får vi

$$[1\ 1\ 1] * [1\ 1\ 1] = [1\ 2\ 3\ 2\ 1]$$

- Fortsetter vi, får vi

$$[1\ 1\ 1] * [1\ 2\ 3\ 2\ 1] = [1\ 3\ 6\ 7\ 6\ 3\ 1]$$

- Ved slike ikke-normaliserte filterkjerner, øker gjennomsnittsverdien i det filtrerte bildet.
- Vanligvis normaliserer vi filterkjernen slik at gjennomsnittsverdien i bildet bevares.

$$\frac{1}{3}[1\ 1\ 1]$$

$$\frac{1}{9}[1\ 2\ 3\ 2\ 1]$$

$$\frac{1}{27}[1\ 3\ 6\ 7\ 6\ 3\ 1]$$

Lavpass-filtre

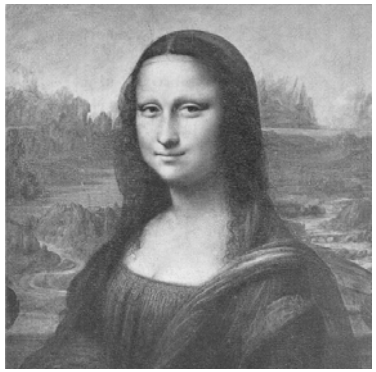
- Slipper gjennom lave frekvenser (forel. 8 & 9) og demper eller fjerner høye frekvenser.
  - Høye frekvenser = skarpe kanter, støy, detaljer.
- Effekt: ”blurring” eller utsmøring av bildet
- Utfordring: bevare kanter samtidig som homogene områder glattes.

Middelverdi-filtere (lavpass)

- Alle koeffisienter er like
- Skaler med summen av filterkoeffisientene
- Størrelsen på filteret avgjør graden av glatting
  - Stort filter: mye glatting (utsmørt bilde)
  - Lite filter: lite glatting, men kanter bevares bedre

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

Filtrerte bilder, middelverdifilter



Original

Filtrert med 3x3-filter

Filtrerte bilder, middelverdifilter

Filtrert med 9x9-filter

Filtrert med 25x25-filter

Separable filtre

- Geometrisk form: kvadrat, rektangel
- Rektangulære middelverdi-filtere er separable.

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

- Fordel: et raskt filter.
- Vanlig konvolusjon: n^2 multiplikasjoner og addisjoner.
- 2 stk 1-D konvolusjoner: 2n multiplikasjoner og addisjoner.

Tidsbesparelse ved separasjon

- Vanlig konvolusjon:
  n^2 multiplikasjoner og addisjoner.
- 2 stk 1-D konvolusjoner:
  2n multiplikasjoner og addisjoner.
- Besparelse ved separasjon av n×n filter:

$$\Delta=\frac{n^2-2n}{n^2}=1-\frac{2}{n}$$

n	Δ
3	0.33
5	0.60
7	0.71
9	0.77
11	0.82
13	0.85
15	0.87

Lavpass-filtrering ved oppdatering

- Det tar n^2 multiplikasjoner og n^2 addisjoner å beregne responsen R for et n×n **uniformt** filter (ser bort fra skaleringen).
- Hvis filteret flyttes ett piksel, blir ny respons

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

  der C_1 er summen av produktene under første kolonne i filteret, og C_n er tilsvarende for siste kolonne i filteret.
- Det tar n multiplikasjoner og n addisjoner å finne hhv. C_1 og C_n .
  - Dvs. totalt $2n+2n$ operasjoner for å finne R_{ny}.
- Oppdatering er her like raskt som separabilitet.
- For uniforme filtere kan vi også droppe alle multiplikasjoner.

Ikke-uniformt lavpass-filter

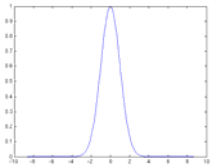
- Uniforme lavpass-filtre kan implementeres raskt.
- Ikke-uniforme filtre, for eksempel:
  - 2D Gauss-filter:

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

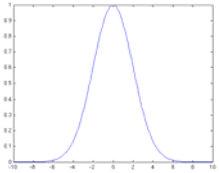
  - Parameter σ er standard-avviket(bredden)
  - Filterstørrelse må tilpasses σ

Effekten av σ

- σ liten: lite glatting
- σ stor: mye glatting
  - •Men mindre glatting enn med et ”flatt” middelverdifilter



σ=1



σ=2

Approksimasjon av Gauss-filtere

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

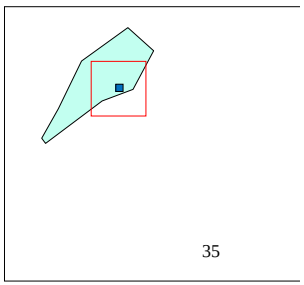
$$= \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix} * \begin{bmatrix}1 & 1\\1 & 1\end{bmatrix}$$

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

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

Kant-bevarende støyfiltrering

- Vi filtrerer for å fjerne støy i bildene.
- Det finnes et utall av ”kantbevarende” filtre.
- Men det er et system i dette:
- Vi kan f.eks. ha to piksel-populasjoner i vinduet
- Da er det sub-optimalt å bruke alle pikslene
- Vi kan sortere pikslene
  - Radiometrisk (etter gråtone)
  - Geometrisk (etter posisjon)
  - Både radiometrisk og geometrisk



Rang-filtrering

- Vi lager en én-dimensjonal liste av alle piksel-verdiene innenfor vinduet.
- Vi sorterer listen i stigende rekkefølge.
- Vi velger en piksel-verdi fra en bestemt posisjon i den sorterte listen
- Denne piksel-verdien er resultatet av filtreringen, og skrives ut til tilsvarende piksel-posisjon i ut-bildet.

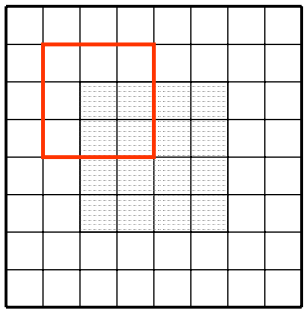
Median-filter

- g(x,y) = median av verdiene i et vindu rundt inn-pikslet.
- Median = den midterste verdien i sortert liste.
- Vindu: kvadrat, rektangel, pluss.
- Rask implementasjon kan gjøres vha. histogram, med histogram-oppdatering etter hvert som vinduet flyttes.
- Et av de mest brukte kant-bevarende støy-filtre.
- Spesielt godt til å fjerne impuls-støy (”salt og pepper”)
- Problemer:
  - Tynne kanter kan forsvinne
  - Hjørner kan rundes av
  - Objekter kan bli litt mindre
- Valg av vindus-størrelse og form er viktig!

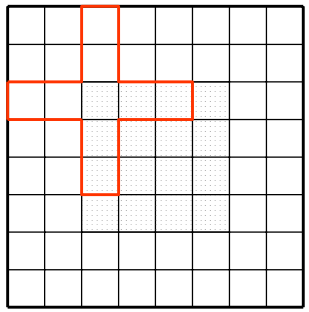
Middelverdi eller median ?

- Middelverdi-filter: beregn middelverdi i vindu.
  - God støy-reduksjon, blurring av kanter.
- Median-filter: finn medianen i vinduet.
  - Dårligere støyreduksjon, bedre kant-bevaring.
  - Fungerer spesielt godt på “salt-og-pepper” støy.

Median og hjørner



Med kvadratisk vindu runder hjørnet av



Med ”pluss”-vindu bevares hjørnet

Raskere median-filtre

- For å finne medianen i et (2r+1)x(2r+1) vindu må vi sortere pikselverdiene og finne midterste verdi i listen.
- Dette filteret er ikke-lineært og ikke-separabelt.
- Kompleksiteten per piksel er $\mathcal{O}(r^2)$ når $0 \le f(x,y) < 2^b$.
- Oppdatering av vindus-histogram reduserer dette til $\mathcal{O}(r)$.
  Huang, Yang, and Tang: ”A fast two-dimensional median filtering algorithm”, IEEE T-ASSP 27(1), 13-18, 1979.
- Medianen finnes raskere ved bit-logikk enn ved sortering.
  Danielsson: ”Getting the median faster”, CVGIP 17, 71-78, 1978.
- Oppdatering av vindus-histogram der informasjon bevares fra linje til linje reduserer kompleksiteten til $\mathcal{O}(1)$.
  Perreault and Hébert: ”Median filtering in constant time”.
  Finnes i OpenCV programbibliotek (se http://nomis80.org/ctmf.html)

Trimmet median

- $g_1(x,y)$ = middelverdi av de mn-d midterste verdiene i et mxn vindu etter sortering:

$$g_1(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{x,y}} g_r(s,t)$$

- $g_2(x,y)$ = middelverdi av piksler innenfor vinduet med gråtone-verdi innenfor et intervall omkring median-verdien.

$$g_2(x,y) = \frac{\sum_{i=1}^{m\times n} W(|f_i - M|) f_i}{\sum_{i=1}^{m\times n} W(|f_i - M|)}$$

$$M = median\{f_i\},\ i = 1,\ldots,m\times n$$

$$MAD = median\{|f_i - M|\},\ i = 1,\ldots,m\times n$$

$$W(|f_i - M|) = \begin{cases} 1 \text{ hvis } |f_i - M| \le k \times MAD \\ 0 \text{ ellers} \end{cases}$$

K Nearest Neighbor-filteret

- g(x,y) = gjennomsnitt (eller median) av de K pikslene i vinduet som ligner mest på senterpikslet i gråtone-verdi.
- Kan sees som en modifikasjon av middelverdi-filtret (eller median-filtret), der man kun tar med de K mest aktuelle av nabo-pikslene.
- 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

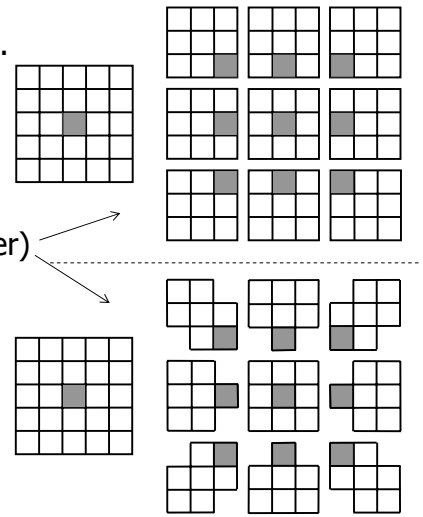
- I (n×n)-vindu :
- K=1: ingen effekt
- K<n: bevarer tynne linjer
- K<(n/2+1)²: bevarer hjørner
  K<(n/2+1)n: bevarer rette kanter

K Nearest Connected Neighbor filtering

- Opererer uten noe vindu
- For alle piksler i bildet:
  - Selected pixel = current pixel
  - While # selected pixels < K
    - Add (4- or 8-) neighbors of recently selected pixel
    - Sort list of pixel values
    - Select pixel most similar to current pixel
  - Compute mean of K selected pixels
  - Dette vil bevare kanter og tynne linjer, uansett form.

Max-homogenitet

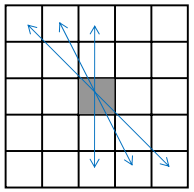
- Ønsker kant-bevarende filter.
- ***Et enkelt triks:***
- Del opp vinduet i flere, overlappende sub-vinduer; slik at alle inneholder senterpikslet (flere muligheter)
- Det mest homogene vinduet inneholder minst kanter.
  - Beregn (μ, σ) i hvert subvindu.
  - Bruk μ fra det sub-vinduet som har lavest σ.
  - Alternativ: |max-min| for σ



Symmetrisk nærmeste nabo (SNN)

- For hvert symmetrisk piksel-par i vinduet:
  - Velg det pikslet som ligner mest på senterpikslet.
- ***Beregn middelverdien av disse pikslene.***

```
for y=0, height-1 do
    for x=0, width-1 do
      sum=0.0; count=0.0; C =f(x,y);
      for v=-r,r do
        for u=-r,r do
          A=f(x+u,y+v); B=f(x-u,y-v)
          if(|C-A|<|C-B|) then
          sum = sum + A;
          else
          sum = sum + B;
          count = count + 1;
        end; end;
    g(x,y) = sum/count;
end;  end;
```



Er det noen ulemper med denne pseudo-koden?

Sigma-filtret

- Ut-verdi=middelverdien av alle piksler innen vinduet hvis gråtonerverdi ligger i intervallet $f(x,y) \pm t\,\sigma$
  - t er en parameter og σ er estimert i homogene områder i bildet, f.eks. som et lavt percentil fra histogram over σ fra alle vinduer.

$$g(x,y) = \frac{\sum_{i=-m}^{m}\sum_{j=-n}^{n} w(i,j)\, f(x+i,y+j)}{\sum_{i=-m}^{m}\sum_{j=-n}^{n} w(i,j)}$$

$$w(i,j) = \begin{cases} 1 \;\; hvis \;\; \left| f(x,y) - f(x+i,y+j) \right| \le t\sigma \\ 0 \;\; ellers \end{cases}$$

- Minner om KNN der filteret selv finner K for hvert vindu.
- Fjerner ikke isolerte støy-piksler.
- Dette kan også fikses!

MMSE-filteret

- MMSE (MinimalMeanSquareError)-filteret:
- Anta at vi har et estimat på støy-varians, σ_n^2
- Vi estimerer lokal støy i et vindu rundt piksel (x,y): $\sigma^2(x,y)$ og lokal middelverdi $\bar{f}(x,y)$.
- Pikslene i ut-bildet finnes fra:

$$g(x,y) = f(x,y) - \frac{\sigma_n^2}{\sigma^2(x,y)}\left[f(x,y) - \bar{f}(x,y)\right]$$

- I homogene områder blir resultatet nær $\bar{f}(x,y)$
- Nær en kant vil $\sigma^2(x,y)$ være større enn σ_n^2 og resultatet blir nær f(x,y).

Min og Max filtre

- Disse filtrene finner hhv. laveste og høyeste pikselverdi innenfor et vindu.
- Begge gir en ikke-lineær blurring av bildet.
- De krever ikke sortering av pikslene i bildet.
- Raskere enn full sortering

  n-1 sammenligninger mot $n\log_2 n$

Max-Min filter

- Også kalt ”Range-filter”
- Ut-verdi er differansen mellom høyeste og laveste pikselverdi innenfor et lokalt vindu.
- Raskt filter
- Fungerer som en kant-detektor uten retnings-angivelse.
- Kan kombineres med tynning og hysterese-terskling til rask ”pseudo-Canny”.
- Kan brukes til ”high-boost”.
- Bør ikke brukes med store vinduer.

Kombinasjoner av Min og Max filtre

- La $min_w(f(x,t))$ og $max_w(f(x,y))$ bety den minste og største verdien f har innenfor et vindu w sentrert om (x,y).
- La vinduet flytte seg gjennom alle mulige posisjoner i bildet.
- Da vil to-pass operasjonen

  $g_1(x,y)=max(min(f(x,y)))$

  fjerne ”topper” som er mindre enn vinduet.
- Operasjonen

  $g_2(x,y)=min(max(f(x,y)))$

  fyller ”daler” som er mindre enn vinduet.
- For å forsterke alle strukturer som er mindre enn vinduet:

  $g(x,y)=f+a(2f-max(min(f))-min(max(f)))$
- Min og Max-operasjonene på et $m \times n$ vindu er separable i den forstand at de kan utføres i to pass med et $(n \times 1)$ vindu og et $(1 \times m)$ vindu.

Mode-filter

- Ut-verdi = hyppigst forekommende verdi i et vindu rundt inn-pikslet.
  - Hvordan er dette forskjellig fra median?
- Implementeres vha. histogram-oppdatering
- Anvendes mest på segmenterte eller klassifiserte bilder, for å fjerne isolerte piksler.
- Forutsetter noen få mulige verdier, derfor brukes det sjelden på gråtone-bilder.

En oppsummering av filtere og bruk

Merk: vi har ikke gjennomgått alle disse

KATEGORI	METODE	ANVENDELSE: Støy-reduksjon	Bilde-skjerping	Kant-deteksjon
Lavpass filtrering	Homogen lavpass	X		
	Gauss-filter	X		
Høypass filtrering	Gradient-operatorer			X
	Høypass (punkt)			
	“High Boost”		X	
	“Unsharp Masking”		X	
	Laplace-operator			X
	LoG-operator			X
	Canny-detektoren			X
Rang filtrering	Median	X		
	Min og Max			
	Kombinasjoner av Min og Max	X	X	
	Mode	X		
	Range			X
Hybrid filtrering	α-trimmet median	X		
	MMADTM	X		
Adaptiv filtrering	KNN	X		
	KNCN	X		
	Sigma	X		
	Invers gradient	X		
	Gradient-masker	X		
	Max Homogenitet	X		
	MMSE	X		

Oppsummering

- Konvolusjon brukes for å filtrere
  et bilde *f* med en filterkjerne *h*.
- Å kunne utføre konvolusjon (manuelt)
  på et lite eksempel er sentralt i pensum ...
- Å programmere 2D konvolusjon er sentralt i
  ukeoppgave 5 og i Oblig1.
- Vær obs på rand-effekter.
- Median bevarer kanter bedre enn middelverdi.
- Det finnes en mengde ikke-lineære filtre.