INF 2310 – Digital bildebehandling

Filtrering i bilde-domenet - 1

Naboskaps-operasjoner
Konvolusjon og korrelasjon
Kant-bevarende filtre
Ikke-lineære filtre
GW Kap 3.4-3.5, + Kap 5.3

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.

Lokale operasjoner

- Vi skal se på teknikker i bilde-domenet
  - Teknikker fra 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

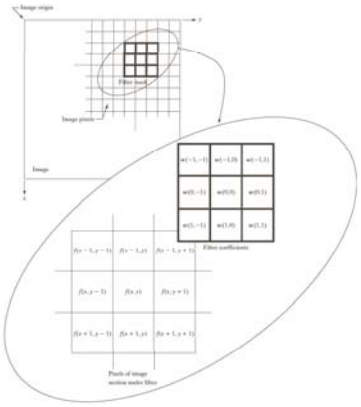
f(x,y) er inn-bildet

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

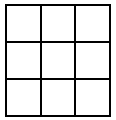
- Hver pikselverdi *g(x,y)* i ut-bildet er en funksjon av pikselverdiene *f(x,y)* i et lokalt område rundt tilsvarende pikselposisjon *(x,y)* i inn-bildet.

Omgivelser/naboskap/vindu

- Kvadratiske/rektangulære vinduer mest vanlig.
- Av symmetri-hensyn oftest odde dimensjon.
- Enkelst transform får vi når vinduet er bare 1x1 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 1x1, har vi en lokal transform.



Naboskap+Transform=Maske



- Naboskap/vindu/omgivelse:
  - De pikslene rundt et punkt i inn-bildet som T opererer på.
- Transform/operator:
  - Operator/algoritme som opererer på pikslene i naboskapet.
  - Sier HVA vi skal gjøre med bildet.
- Maske/filter:
  - En matrise med vekter eller koeffisienter.

1	1	1
1	1	1
1	1	1

Eksempel: et gjennomsnittsfilter

- Resultatene lagres i et nytt bilde.

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 - 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 - 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 – Posisjons-invarians

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

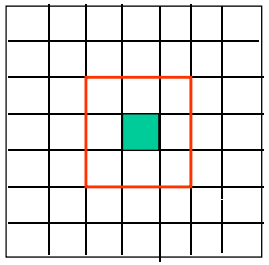
for alle *f(x,y)* og for vilkårlige *(l,m).*

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

Eksempel: middelverdi

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

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



f(x,y) og 3×3 vindu

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:

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

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

Men hva skjer rundt kantene?

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.

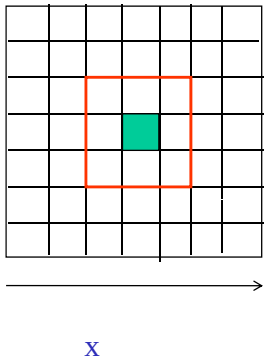
2-D konvolusjon

- 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 med størrelse $m=2w_1+1$, $n=2w_2+1$
- *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 beregner summen på nytt.



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.

Litt notasjon

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

gir oss verdien i ut-bildet for piksel-posisjon (x,y)

- For å konvolvere filteret med hele bildet, bruker vi notasjonen

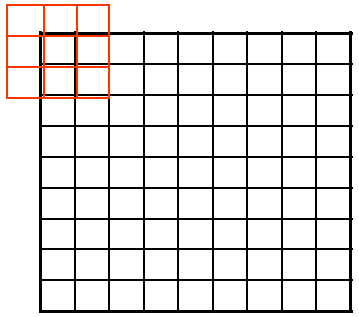
  g = h * f

* er konvolusjons-operatoren

Praktiske problemer

- Kan vi la ut-bildet ha samme piksel-representasjon som inn-bildet? (unsigned byte for eksempel?)
- Trenger vi et mellom-lager?
- Hva gjør vi langs bilde-kantene?
- Anta at bildet er M x N piksler
- Anta at filteret et m x n
  ($m=2m_2+1$, $n=2n_2+1$)
- Uberørt av kant-effekt:
  $(M-2m_2)<(N-2n_2)$

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



Hva gjør vi langs kanten?

Alternativer:

1. Sett g(x,y)=0
2. Sett g(x,y)=f(x,y)
3. Trunker ut-bildet (lager det mindre)
4. Trunker konvolusjons-masken h
5. Utvid bildet ved speiling om kant (”reflected indexing”)
6. ”Circular indexing”
   - Anta at bildet repeterer seg selv uendelig i alle retninger
   - Merk: dette bruker vi i Fourier-transform

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 vil lage en effektiv implementasjon av et stort filter ved en kombinasjon av enkle, separable filtre.
  - 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:
  - pluss i stedet for minus.
- Minus for konvolusjon gjør at vi roterer filteret 180 grader.
- For korrelasjon trenger vi ikke dette, vi legger filterkjernen sentrert om piksel *(x,y)* og multipliserer hvert enkelt element.
- Merk: vi kan utføre korrelasjon ved en konvolusjons-funksjon 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 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 operasjoner!

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 gjennomsnittsverdiet 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

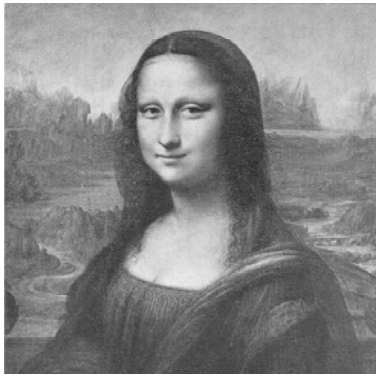
- Slipper gjennom lave frekvenser (se forelesninger om Fourier) og demper høye frekvenser.
  - Høye frekvenser = skarpe kanter, støy, detaljer.
- Effekt: ”blurring” eller utsmøring av bildet.
- Utfordring: bevare kanter samtidig som homogene områder glattes.

Middelverdi-filtere (lavpass)

- 3×3: $\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$
- 5×5: $\frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{bmatrix}$
- 7×7: $\frac{1}{49}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$

- Alle koeffisienter er like.
- Skaler resultatet ved å normalisere med summen av filterkoeffisientene.
- Størrelsen på filteret avgjør graden av glatting.
  - Lite filter: lite glatting, kanter bevares bedre.
  - Stort filter: mye glatting og utsmørt bilde.

Filtrerte bilder, middelverdifilter



Original

Filtrert 3x3-filter

Filtrerte bilder, middelverdifilter

Filtrert 9x9-filter

Filtrert 25x25-filter

Separable filtre

- Et separabelt filter kan splittes i mindre enklere filtere (for eksempel to 1-dimensjonale 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 1-D konvolusjoner: $2n$ multiplikasjoner og addisjoner.

Tidsbesparelse

- Vanlig konvolusjon: n^2 multiplikasjoner og addisjoner.
- 2 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^2-1 addisjoner å beregne resultatet R for et nxn 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 i første kolonne i filteret, og C_n er tilsvarende for siste kolonne i filteret.
- Det tar n multiplikasjoner og (n-1) addisjoner å finne hhv. C_1 og C_n.
  - Dvs. totalt 2n+2n operasjoner for å finne R_{ny}.
- Oppdatering er like raskt som separabilitet.
- For uniforme filtere kan vi også droppe alle multiplikasjoner.

Ikke-uniformt lavpass-filter

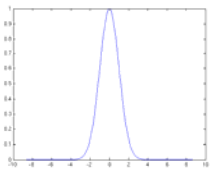
- Uniforme lavpass-filtre kan implementeres raskt.
- Et vanlig ikke-uniformt filter er Gauss-filteret:
  - 2D Gauss-filter:

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

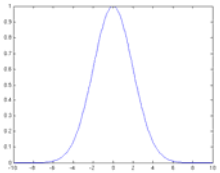
  - Parameter σ er standard-avviket (bredden)
  - Filterstørrelse må tilpasses σ

Effekten av σ

- σ liten: lite glatting
- σ stor: mye glatting
- Men mindre enn med "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 filter-jungelen:
  - Vi kan jo to piksel-populasjoner i vinduet.
  - Da er det suboptimalt å bruke alle pikslene.
  - Vi kan sortere pikslene:
    - Radiometrisk
    - Geometrisk
    - Både radiometrisk og geometrisk

Rang-filtrering

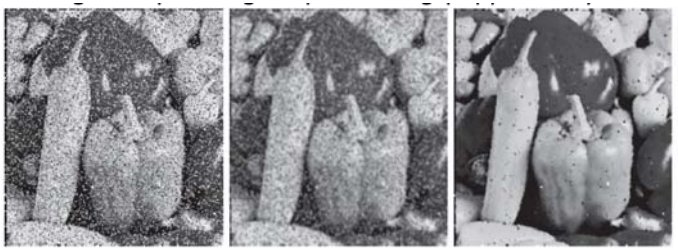
- Vi lager en en-dimensjonal liste av alle piksel-verdiene innenfor vinduet.
- Vi sorterer listen i stigende rekkefølge.
- Vi velgen en piksel-verdi fra en bestemt posisjon i den sorterte listen
- Denne piksel-verdien er resultatet av filtreringen, og skrives ut til tilsvarende piksel-posisjon i ut-bildet.

Median-filter

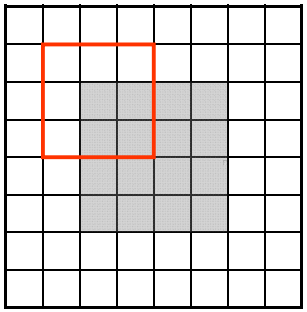
- g(x,y) = median av verdiene i et vindu rundt inn-pikslet.
- Median = den midterste verdien i sortert liste.
- Vindu: kvadrat, rektanger, pluss.
- Rask implementasjon kan gjøres vha. histogram, med histogram-oppdatering etter hvert som vinduet flyttes.
- Et av de mest brukte kant-bevarende støy-filtre.
- 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 vindus-størrelse og form viktig!

Middelverdi eller median?

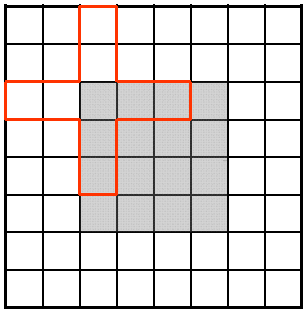
- Middelverdi-filter: beregn middelveri i vindu.
  - God støy-reduksjon, men blurring av kanter.
- Median-filter: finn medianen i vinduet.
  - Dårligere støyreduksjon, bedre kant-bevaring.
  - Fungerer spesielt godt på "salt-op-pepper" støy.



Median og hjørner



Med kvadratisk vindu
rundes hjørnet av



Med "pluss"-vindu
bevares hjørnet

Median ved histogram-oppdatering

1. Skal oppdatere median-verdien mens vi flytter et vindu med *2m+1* rader og *2n+1* kolonner rundt i bildet. Sett *th=(2m+1)(2n+1)/2.*
2. Sett viduet ved venstre bildekant, sorter og finn mediamed *MED* og antall piksler med gråtone *f≤MED, LE_MED*
3.

```
 for j=-m to m do
begin
    --H[f(x-n,y-j)]
    if (f(x-n,y-j)<MED) then --LE_MED
end
```

4. Flytt vinduet et piksel til høyre:

```
 for j=-m to m do
begin
    ++H[f(x+n,y-j)]
    if (f(x+n,y-j)<MED) then ++LE_MED
end
```


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 $n \log_2 n$

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.

KNN-filteret

- KNN = K Nærmeste Nabo
- Ut-verdi = gjennomsnitt (eller median) av de K pikslene i vinduet som ligger nærmest 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.
  - Velger vi for liten K, fjerner vi lite støy
  - For stor K fjerner tynne linjer og hjørner

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

Adaptiv filtrering

- Adaptive filtere beregner gråtone-statistikk fra et vindu rundt hvert piksel og lar filter-parametere avhenge av dette.
- Et piksel kan betraktes som støy dersom det er signifikant forskjellig fra sine naboer:

```
if abs( f(x,y) - middelverdi )  >  t then
   g(x,y) := middelverdi
else
   g(x,y) := f(x,y)
endif;
```

- Men hvordan setter vi *t* og finner hva som er "signifikant forskjellig"?

Sigma-filtret

- Ut-verdi=middelverdien av alle piksler innen vinduet hvis gråtonerverdi ligger i intervallet $f(x,y)\pm t\sigma$, der t er en parameter og σ er estimert i et homogent område i bildet.

- Fjerner ikke isolerte støy-piksler.

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….
- Vær obs på kant-effekter.
- Ikke-lineære filtere
- Planer framover:
  - Ulike typer filtre
  - Gradient-operatorer
  - Kantdeteksjon