
INF2310 – 1. februar 2017 – Ukens temaer

(Kap 2.4.4 og 2.6.5 i DIP)

- Geometriske operasjoner
 - Lineære / affine transformer
- Resampling og interpolasjon
- Samregistrering av bilder

Geometriske operasjoner

- Endrer på pikslenes posisjoner
- Transformerer pikselkoordinatene (x,y) til (x',y') :

$$x' = T_x(x,y)$$

$$y' = T_y(x,y)$$

- T_x og T_y ofte gitt som polynomer

Merk: Her er det ikke pikselverdiene, men piksel-koordinatene x og y som endres.

Anvendelser

- Forstørre deler av bilder for visuell inspeksjon («zooome»)
- Rette opp geometriske feil som oppstår under avbildningen
 - Rotasjon
 - Fiskeøyelinse
 - (Radar)avbildning av terreng
 - Generell linsekorrigerings
 - ...
- Samregistrere bilder
 - .. fra ulike sensorer (f.eks. CT, MR, US)
 - .. tatt på ulike tidspunkt / vinkler
 - .. med kart i en bestemt kartprojeksjon
 - Eks ansiktsgjenkjenning: Finne ansiktene i et bilde og transformere bildet slik at ansiktene i bildet blir på samme sted, orientering og i samme størrelse som i referansebildene
- Generere bilder fra andre kameravinkler
- Spesialeffekter

Affine transformer

- Eksempler på affine transformasjoner:
 - Translasjon (forflytning)
 - Rotasjon
 - "Shearing"
 - Refleksjon
 - Skalering
 - **Kombinasjoner av disse (!)**

Affine transformer kont.

- Transformerer pikselkoordinatene (x,y) til (x',y') :

$$x' = T_x(x,y)$$

$$y' = T_y(x,y)$$

- Affine transformer beskrives ved:

$$x' = a_0x + a_1y + a_2$$

$$y' = b_0x + b_1y + b_2$$

- På matriseform:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{eller} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ b_0 & b_1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

Egenskaper ved affine transformer

- Rette linjer bevares
- Parallelle linjer forblir parallelle
- Lineær transform med påfølgende translasjon
- Kan uttrykkes ved enkel matrisemultiplikasjon
 - Og følgelig svært enkle generelle inverse

Eksempler på enkle transformer

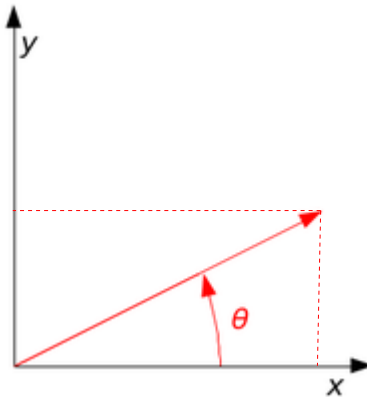
Transformasjon	a_0	a_1	a_2	b_0	b_1	b_2	Uttrykk
Translasjon med Δx og Δy	1	0	Δx	0	1	Δy	$x' = x + \Delta x$ $y' = y + \Delta y$
Skalering med faktor s	s	0	0	0	s	0	$x' = sx$ $y' = sy$
Rotasjon med θ	$\cos\theta$	$-\sin\theta$	0	$\sin\theta$	$\cos\theta$	0	$x' = \cos\theta x - \sin\theta y$ $y' = \sin\theta x + \cos\theta y$
Horisontal "shear" med faktor s	1	s	0	0	1	0	$x' = x + sy$ $y' = y$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Litt mer om rotasjon

Hva vil vi skal skje med hver av disse basisvektorene ved en rotasjon med θ ?

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

Sammenslåing av affine transformere

$$\begin{bmatrix} \text{transl.} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \begin{bmatrix} \text{rot.} \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

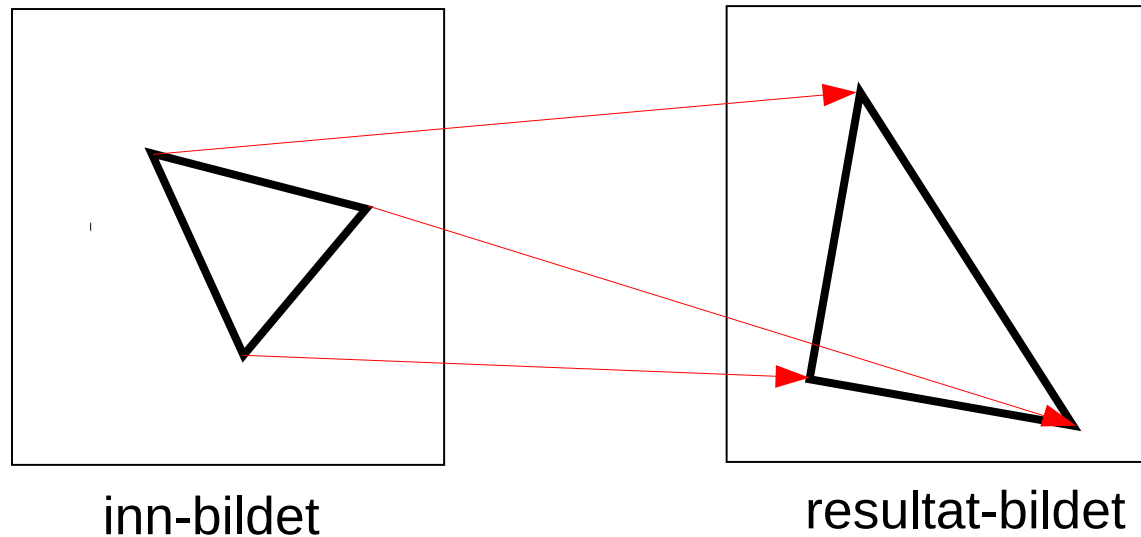
$$\begin{bmatrix} \text{rot.} \end{bmatrix} \begin{bmatrix} \text{transl.} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \text{transl. \& rot.} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix}$$

(etc.)

Alternativ måte å finne transformkoeffisientene

- En affin transform kan bestemmes ved å spesifisere tre punkter før og etter transformasjonen



- Med disse tre punktparene kan vi finne de 6 koeffisientene; $a_0, a_1, a_2, b_0, b_1, b_2$
- Med flere enn 3 punktpar velger man den transformasjonen som minimerer (f.eks. kvadrat-)feilen summert over alle punktene (mer om dette senere)

Transformer med høyere ordens polynomer

- Bilineære transformer beskrives ved:

$$x' = a_0x + a_1y + a_2 + a_3xy$$

$$y' = b_0x + b_1y + b_2 + b_3xy$$

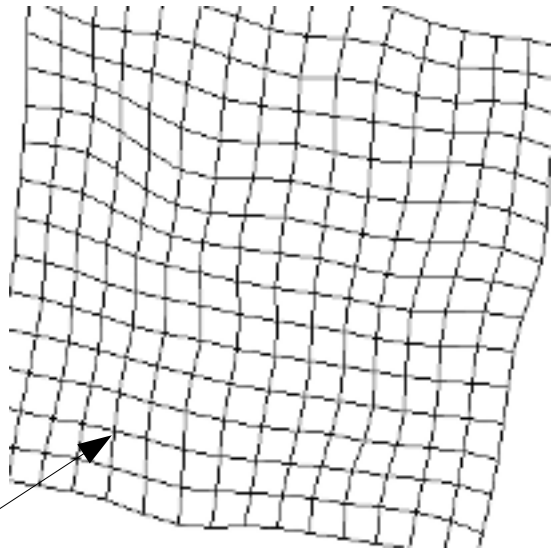
- Kvadratiske transformer:

$$x' = a_0x + a_1y + a_2 + a_3xy + a_4x^2 + a_5y^2$$

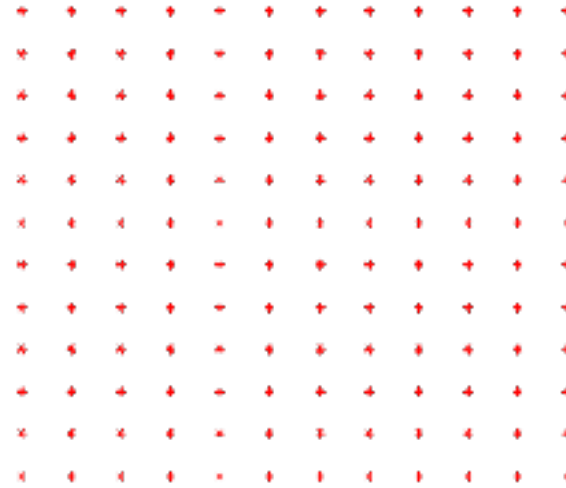
$$y' = b_0x + b_1y + b_2 + b_3xy + b_4x^2 + b_5y^2$$

- Polynomer av høyere orden gir muligheter for å korrigere for mer «komplekse» avbildningsfeil

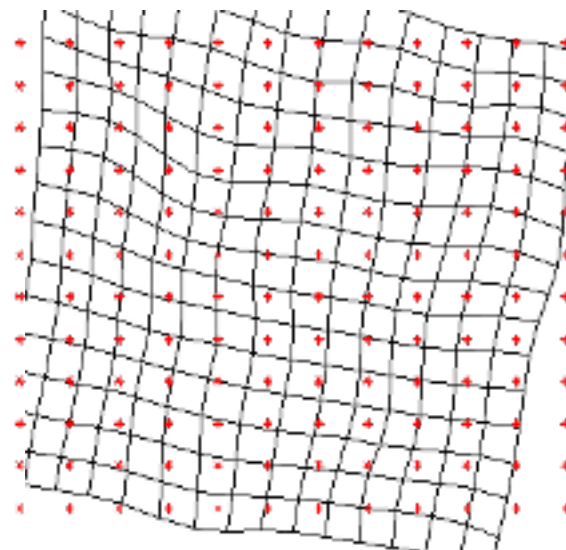
Resampling



I illustrasjonen:
Har kun sample/piksel-verdier
der linjene krysser



Vil gjerne ha bildet samlet
på et rektangulært grid
→ resample



Ikke glem at sammenhengen mellom
romlig oppløsning og samplingsrate
(samplingsteoremet) fortsatt gjelder!

En noe naiv fremgangsmåte: Forlengings-mapping

```
for all x',y' do g(x',y')=0
```

```
a0 = cos θ
```

```
a1 = -sin θ
```

```
b0 = sin θ
```

```
b1 = cos θ
```

```
for all x,y do
```

```
  x' = round(a0x+a1y)
```

```
  y' = round(b0x+b1y)
```

```
  if (x',y') inside g
```

```
    g(x',y') = f(x,y)
```

```
end
```

Eksempel:

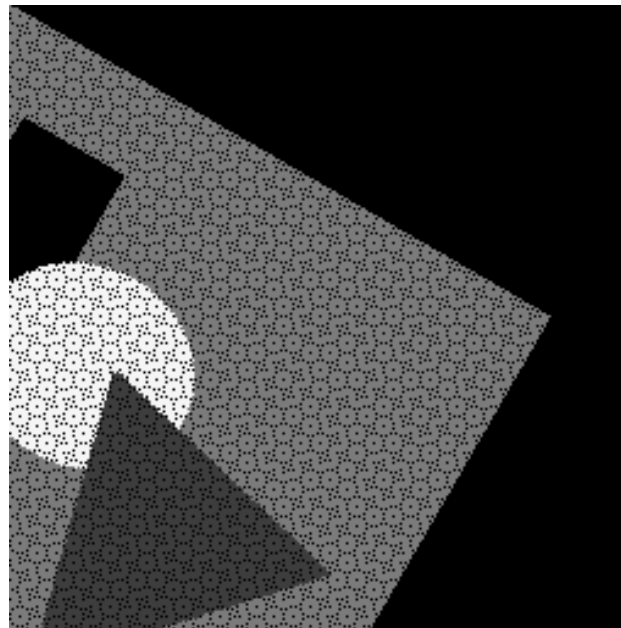
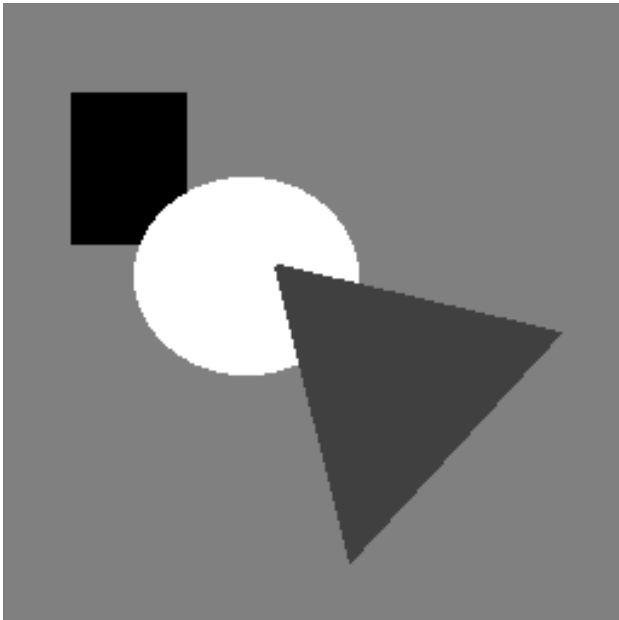
Enkel rotasjon ved transformen:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Flytter de posisjonstransformerte
pikselverdiene
til nærmeste pikselposisjon i utbildet

Skriver innbildets f(x,y) inn i g(x', y')

Forlengs-mapping, forts.

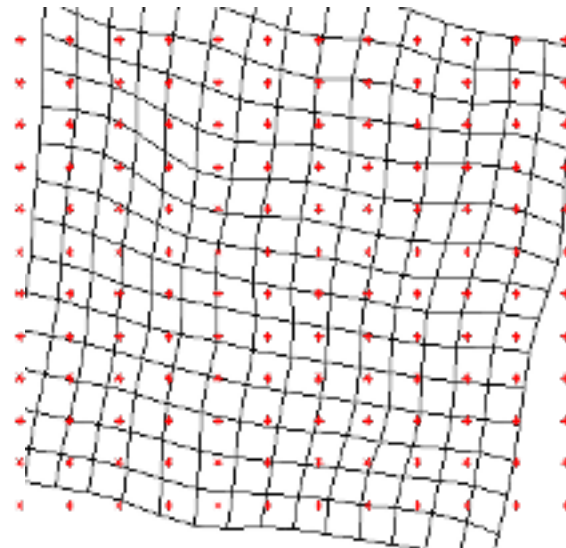


Problemer:

- Ikke alle utpiksler får verdi (hullene i bildet)
- Unødig beregning av pikselkoordinater som allikevel ikke blir synlige (ender utenfor utbildet)
- Samme utbilde-piksel kan bli satt flere ganger

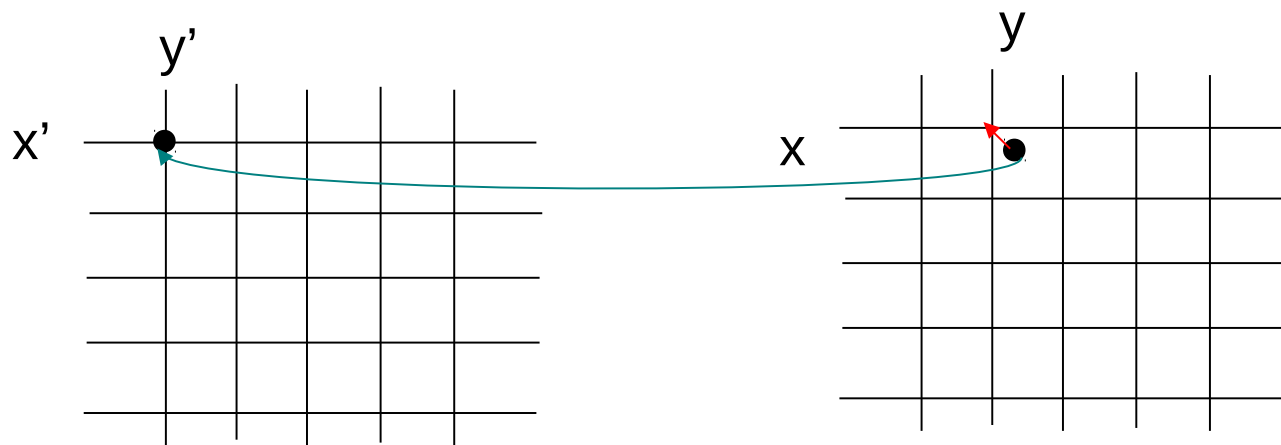
«Baklengs-mapping»

- Løp igjennom «utbilde»-pikselposisjonene (røde prikker på figuren) og finn ut hvilke verdier vi har der



Interpolasjon – hvilke intensitetsverdier har bildet mellom sample-punktene?

Baklengs-mapping



Enkleste løsning:

Nullte-ordens interpolasjon eller nærmeste nabo-interpolasjon

$$g(x',y') = f(\text{round}(x) , \text{round}(y))$$

[Intensiteten til g blir da ALLTID en av verdiene til f]

Baklengs-mapping med nærmeste-nabo-interpolasjon

```
a0 = cos (-θ)
a1 = -sin (-θ)
b0 = sin (-θ)
b1 = cos (-θ)

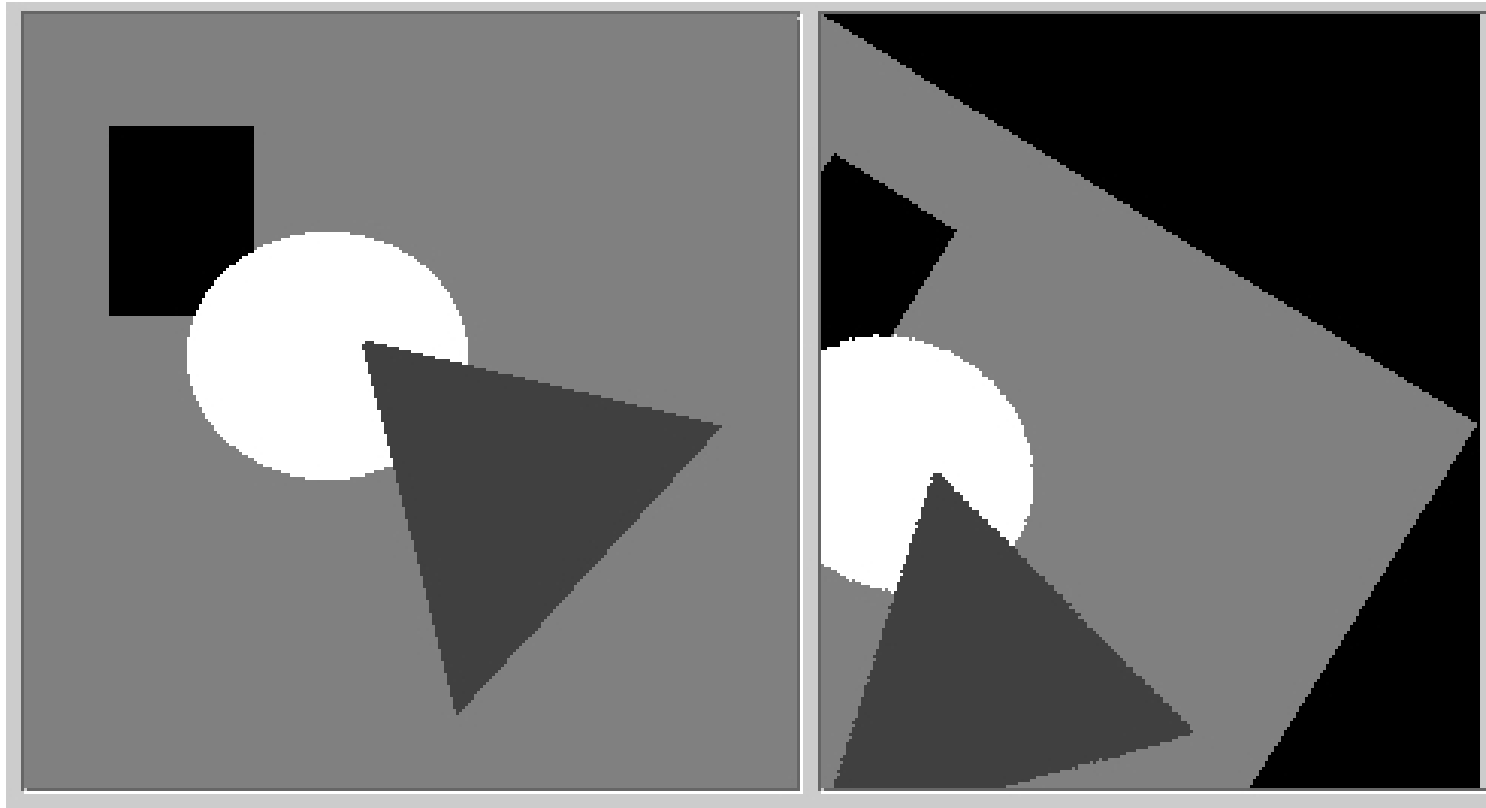
for alle x', y' do
  x = round(a0x' + a1y')
  y = round(b0x' + b1y')
  if (x, y) inside f
    g(x', y') = f(x, y)
  else
    g(x', y') = 0
end
```

Resample bildet. Her; for hvert utbilde-piksel, invers-transformér, og velg nærmeste piksel fra innbildet.

Samme eksempel som ved forlengs-mappingen.

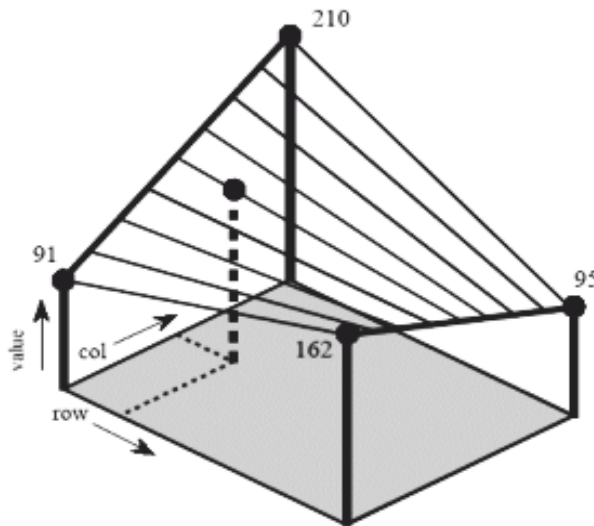
Husk: Hvis (x,y) roteres med θ og gir (x', y'), tilsvarer det at hvis (x', y') roteres med $-\theta$ får vi (x,y). [Kan selvfølgelig også bare inverttere transformmatrisen for å få koeffisientene.]

Baklengs-mapping, forts.



Første-ordens interpolasjon/ bilineær interpolasjon

- Intensiteten blir en kombinasjon av pikselverdiene i de fire pikslene som omgir punktet
- Bidragene fra hver av disse vektet med avstanden
- Interpolere i x- og y-intervallene mellom 0 og 1:
 $f(x, y) \approx f(0, 0)(1 - x)(1 - y) + f(1, 0)x(1 - y) + f(0, 1)(1 - x)y + f(1, 1)xy$



Praktisk algoritme:

$$x_0 = \text{floor}(x), \quad y_0 = \text{floor}(y)$$

$$x_1 = \text{ceil}(x), \quad y_1 = \text{ceil}(y)$$

$$\Delta x = x - x_0$$

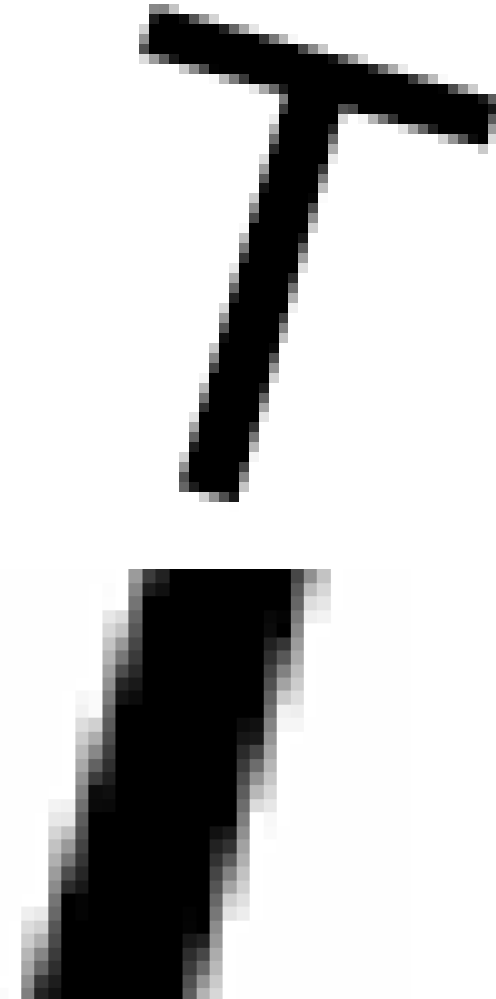
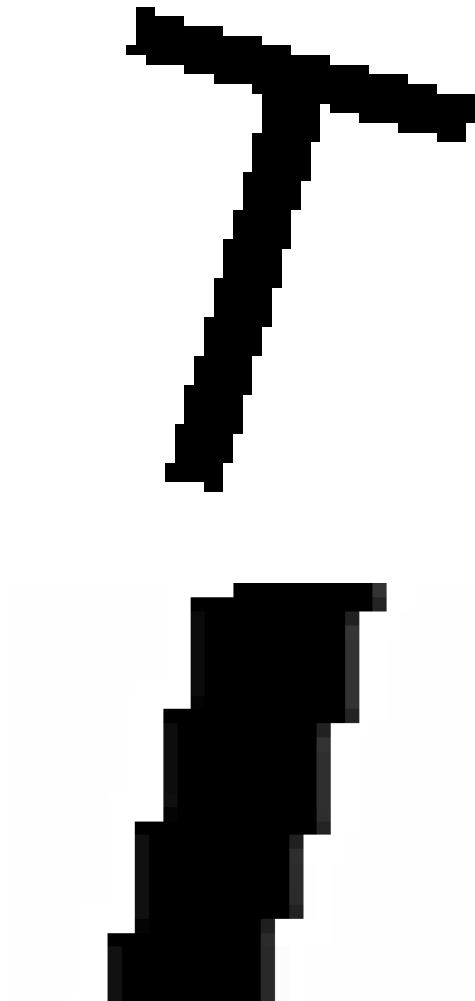
$$\Delta y = y - y_0$$

$$p = f(x_0, y_0) + [f(x_1, y_0) - f(x_0, y_0)] \Delta x$$

$$q = f(x_0, y_1) + [f(x_1, y_1) - f(x_0, y_1)] \Delta x$$

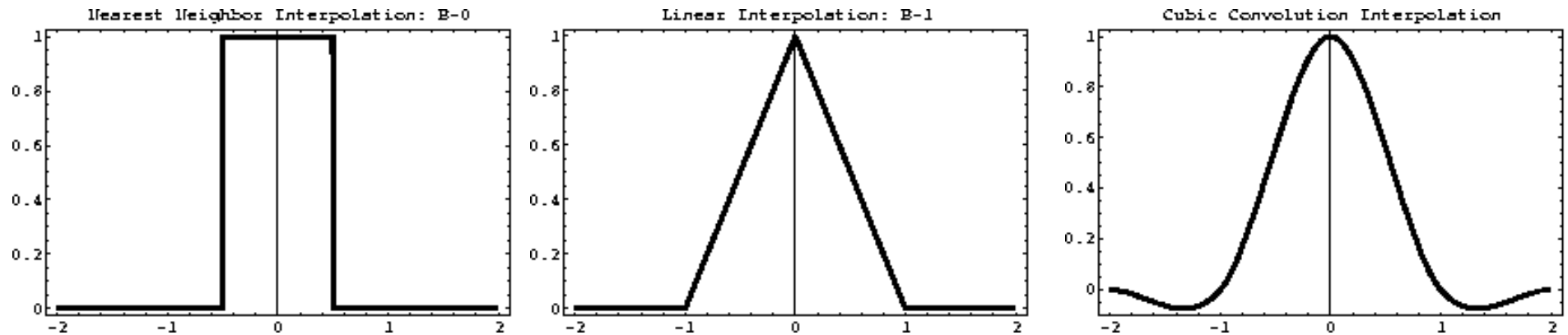
$$f(x', y') = p + (q - p) \Delta y$$

Bilineær interpolasjon, eksempel



Høyere-ordens interpolasjon

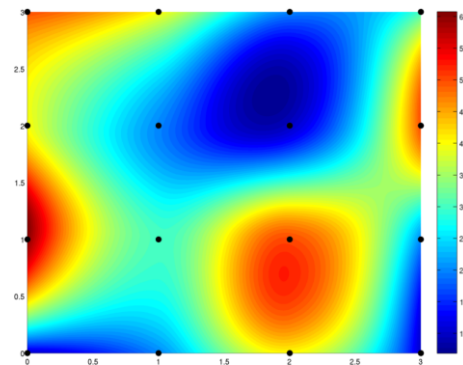
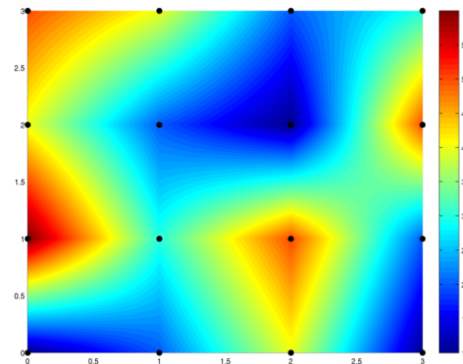
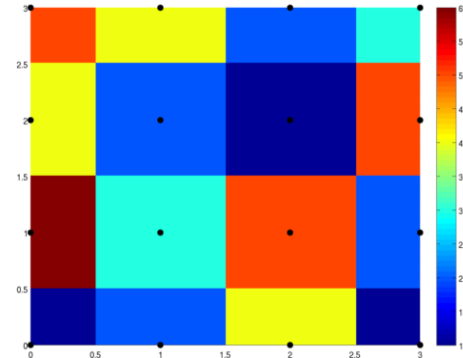
- Kubisk interpolasjon benytter et naboskap på 4×4 piksler
- Interpolasjon kan sees på som (kontinuerlig) konvolusjon med bestemte filtre



(1D-varianter av nærmeste nabo, lineær og kubisk interpolasjonskjerne)

Interpolasjon – en sammenligning

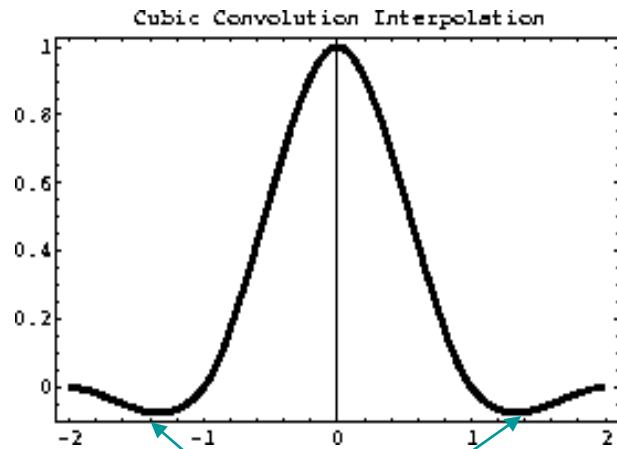
- Nærmeste nabo gir 2D trappefunksjon, med diskontinuitet midt mellom punktene
- Bi-lineær interpolasjon bruker $2 \times 2 = 4$ piksler. Derivert er ikke kontinuerlig over flaten
- Bi-kubisk interpolasjon gir glattere flater enn bilineær, men er mer regnekrevende; bruker $4 \times 4 = 16$ piksler



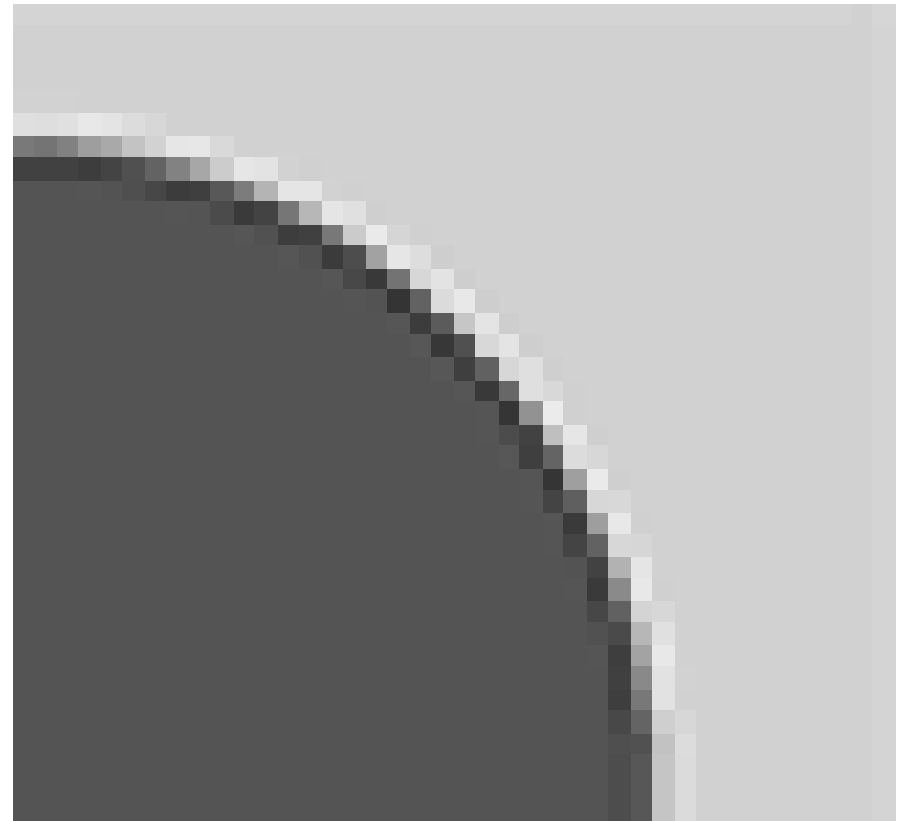
Interpolasjonsfunksjoner i praksis

- Nærmeste nabo:
 - Taggete kanter og større totalfeil.
 - Hver ut-piksel har en verdi fra inn-bildet:
 - Ingen rekvantisering nødvendig, og en fordel hvis man vil bevare visse statistiske egenskaper i bildet (eller hvis bildet er segmenert i ulike klasser)
- Bilineær interpolasjon og høyere-ordens interpolasjon er mer regnekrevende
- Kubisk interpolasjon gir skarpere bilder og har kontinuerlige deriverte, men er (mye) mer regnekrevende enn bilineær interpolasjon, og kan gi opphav til "kant-glorie-effekter"

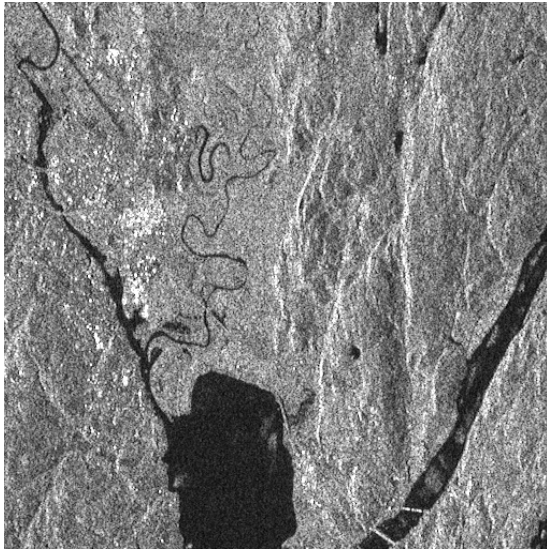
”Kant-glorie-effekter” / ”ringing” ved kubisk interpolasjon



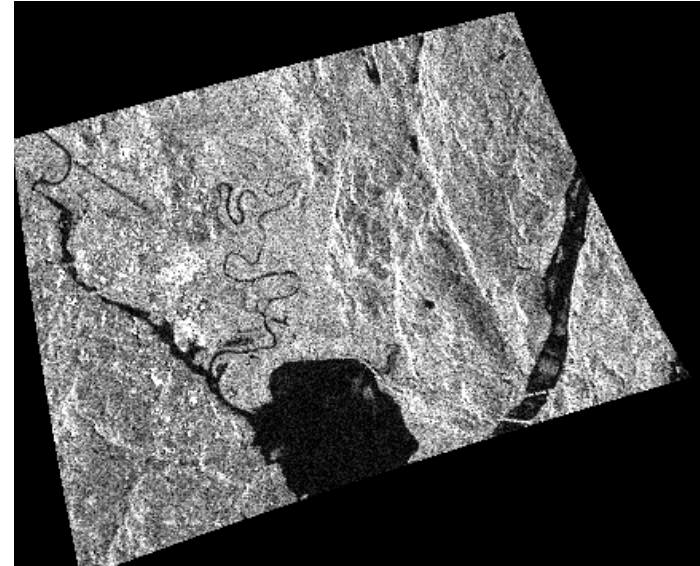
Negative «lobe»-verdier



Bruk av geometriske transformer: Samregistrering av bilder



Original



Transformert



Ønsket bilde å
samregistrere med

Samregistrering II

- Om bildenes kartkoordinater er kjent kan disse benyttes til å finne transformkoeffisientene
- Hvis ikke, brukes gjerne kontrollpunkter:
 - Kontrollpunkter plukkes ut manuelt - lett identifiserbare punkter (landemerker) i begge bildene
 - Affine transformer er unikt spesifisert med 3 punktpar (bestemmer $a_0, a_1, a_2, b_0, b_1, b_2$), bilineære med 4 punktpar (bi-kvadratiske med 6, etc.)
 - I praksis velges ofte mange flere punkter for å få en god transformasjon (se neste side)

Samregistrering III

- Ved flere kontrollpunkter enn nødvendig for å bestemme transformkoeffisientene, benyttes ofte kvadratfeilen som minimeringskriterium
- Gitt M kontrollpunkter $(x_i, y_i), (x_i^r, y_i^r)$ («r» indikerer referansebildet) og anta mappingen $(x_i, y_i) \rightarrow (x'_i, y'_i)$
- Polynomkoeffisientene settes til de som minimerer kvadratfeilen mellom kontrollpunktets sanne koordinater (x_i^r, y_i^r) og de transformerte koordinatene (x'_i, y'_i) :

$$J = \sum_{i=1}^M (x'_i - x_i^r)^2 + (y'_i - y_i^r)^2$$

- "Enkel" lineæralgebra benyttes til å finne eksakt løsning

Samregistrering IV (Minimere kvadratfeilen)

$$J = \sum_{i=1}^M (x'_i - x_i^r)^2 + (y'_i - y_i^r)^2 = J_x + J_y$$

$$J_x = \sum_{i=1}^M (x'_i - x_i^r)^2$$

G og a er her basert på en affin transform

$$\begin{array}{c}
 \mathbf{d} \\
 \left[\begin{array}{c} x_1^r \\ x_2^r \\ \vdots \\ x_M^r \end{array} \right]
 \end{array}
 \quad
 \begin{array}{c}
 \mathbf{G} \\
 \left[\begin{array}{ccc} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_M & y_M & 1 \end{array} \right]
 \end{array}
 \quad
 \begin{array}{c}
 \mathbf{a} \\
 \left[\begin{array}{c} a_0 \\ a_1 \\ a_2 \end{array} \right]
 \end{array}$$

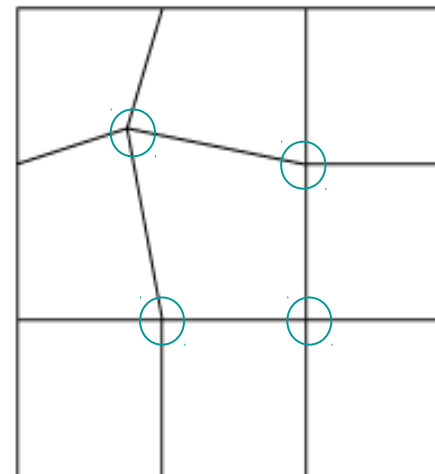
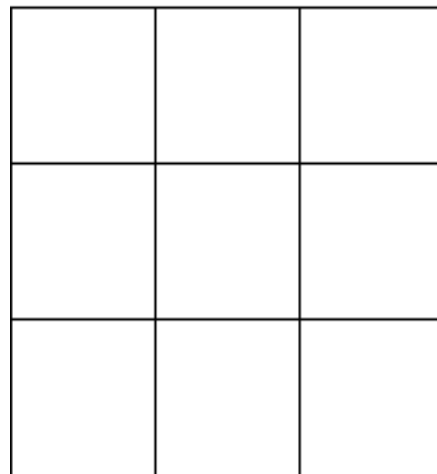
Vi er på jakt etter koeffisientene i denne **a**-vektoren

$$J_x = (d - Ga)^T (d - Ga) = d^T d + a^T G^T G a - 2a^T G^T d$$

$$\frac{\partial J_x}{\partial a^T} = 2G^T G a - 2G^T d = 0 \quad \Rightarrow \quad a = \underbrace{(G^T G)^{-1} G^T d}_{\text{Én enkelt matrise}}$$

Stykkevis transformeringer

- Forskjellige transformeringer for ulike deler av bildet
- Ofte bestemmes et kontrollgrid som styrer hvordan de ulike delene skal endres
- Bilineær transformasjon benyttes ofte:
 - $x' = a_0xy + a_1x + a_2y + a_3$
 - $y' = b_0xy + b_1x + b_2y + b_3$



Hver firkants fire hjørnepunkter bestemmer entydig den bilineære transformen

Oppsummering

- Transform/endring av pikslenes posisjoner (x-,y-koordinater)
 - Fokus på affine transformer
- Resampling
 - Forlengs- og baklengsmapping
 - Interpolasjonsmetoder
 - Nærmeste nabo-interpolasjon
 - Bilineær interpolasjon
 - Høyere-ordens interpolasjoner
- Bruk av geometriske operasjoner til å samregistrere bilder
 - Kontrollpunkter