Bildverarbeitung und Algorithmen

Prof. Dr. Wolfgang Konen

Geometrische Transformationen

Inhalt

- **Einführung**
- Transformationen und Homogene Koordinaten
- Interpolation der Grauwerte
  - nächster Nachbar
  - bilinear
- Nichtlineare Transformationen
- Warping-Algorithmus

Geometrische Bildtransformationen
Einführung (1)

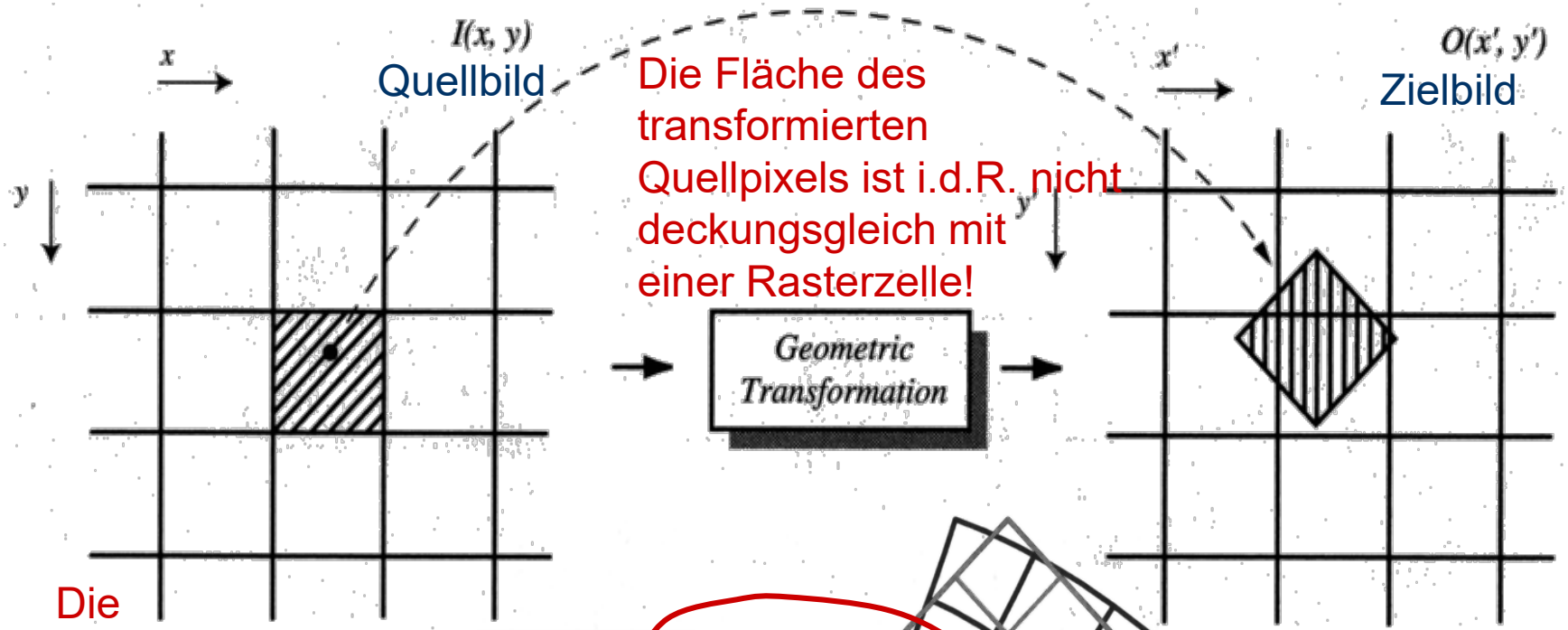
- Geometrische Transformationen sind übliche Operationen in der Computergrafik.
- Bei der Bildanalyse werden geometrische Transformationen zur Korrektur von Verzerrungen (z.B. verursacht durch das Kameraobjektiv) angewandt, beziehungsweise allgemein als Normierungsoperation vor einer Mustererkennung oder Objektvermessung eingesetzt.
- Grundsätzlich wird bei der geometrischen Transformation eines Bildes jeder Pixelwert des Quellbildes **I**(x,y) auf eine neue Position (x', y') im Zielbild abgebildet. Die Nachbarschaftsverhältnisse bleiben dabei erhalten!
- Die Zielposition eines transformierten Pixels liegt i.A. nicht exakt auf einer Rasterposition im Zielbild:
  - Die Zielkoordinaten (x',y') sind Gleitkommazahlen.
  - Um das transformierte Pixel im Zielbild abbilden zu können, muss interpoliert werden.

Geometrische Bildtransformationen
Einführung (2)

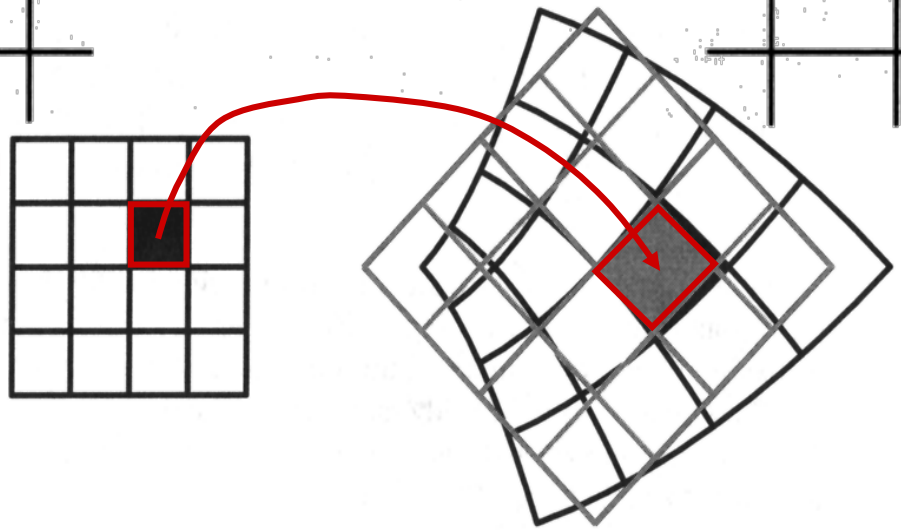
- Geometrische Bildtransformationen verändern nicht nur die Position eines Pixels, sondern oft auch die Bildfläche (Anzahl der Rasterpunkte), die das Pixel im Zielbild ausfüllen soll.
  - Je nachdem, ob das Quellpixel im Zielbild genau eine, weniger als eine oder mehr als eine Rasterzelle ausfüllt, ist der Flächenvergrößerungsfaktor der Transformation genau 1,0 , < 1,0 oder > 1,0 .

- **Lineare** geometrische (affine) Transformationen haben einen konstanten (ortsinvarianten) Flächenvergrößerungsfaktor.
  Beispiele: Translation, Rotation, Skalierung.
- **Nicht-lineare** geometrische Transformationen (*warping transformations*) haben einen variierenden (ortsvarianten) Flächenvergrößerungsfaktor.
  Beispiele: Perspektivische Abbildung, Linsenverzerrung.

Illustrationen zur geometrischen Transformation eines Quellpixels in das Zielbild



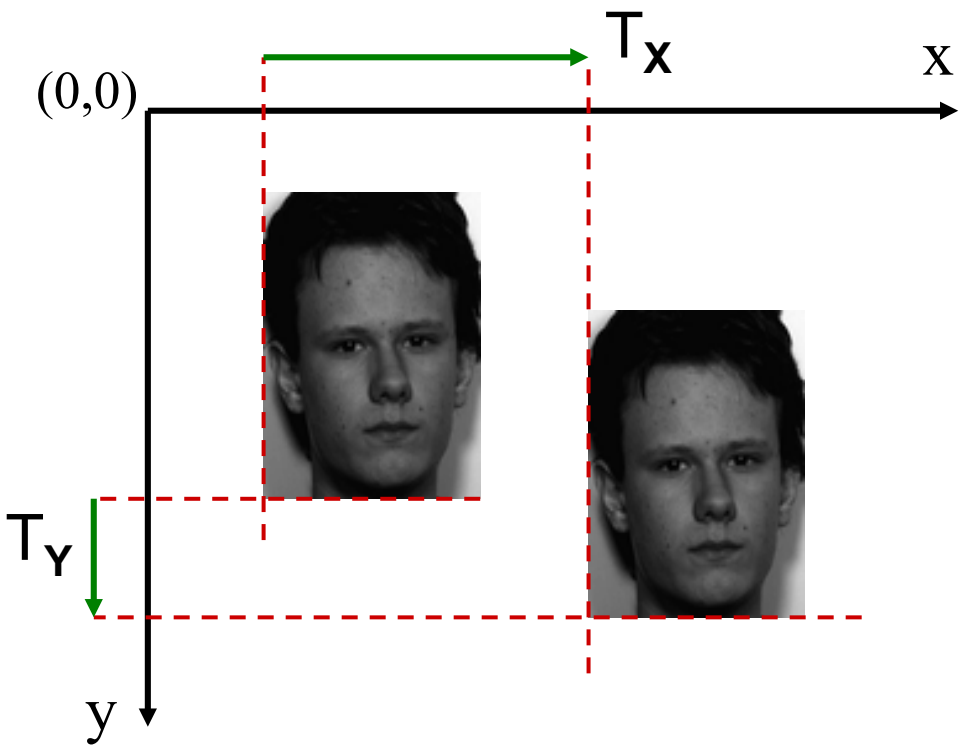
Die Flächenvergrößerung einer beliebigen Transformation wird durch lokale Annäherung einer linearen Abbildung bestimmt.



Inhalt

- Einführung
- **Transformationen und Homogene Koordinaten**
- Interpolation der Grauwerte
  - nächster Nachbar
  - bilinear
- Nichtlineare Transformationen
- Warping-Algorithmus

Translation



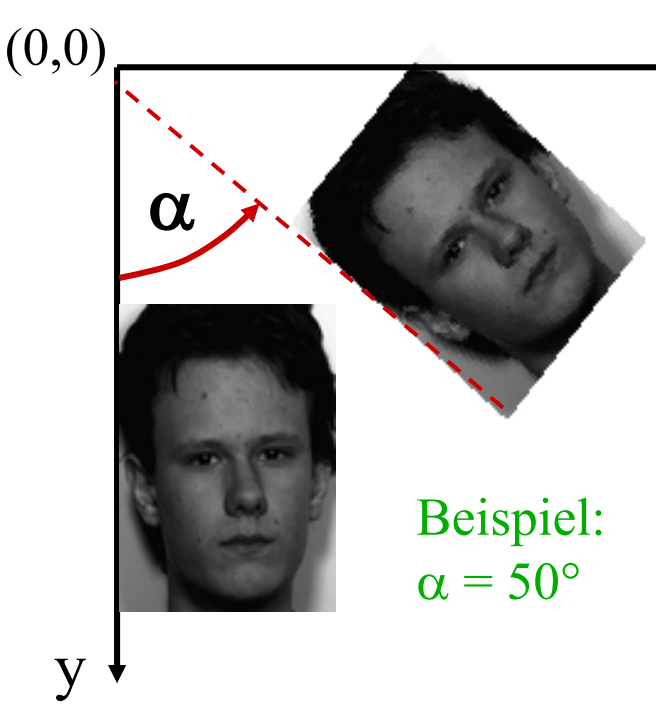
Die Position (x',y') eines Pixels nach einer Verschiebung (Translation) um T_X in positive x-Richtung und T_Y in positive y-Richtung wird wie folgt berechnet:

$$x' = x + T_X$$
$$y' = y + T_Y$$

Die Transformationsgleichung für die Translation in Vektorschreibweise:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_X \\ 0 & 1 & T_Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation



Beispiel:
$\alpha = 50°$

Die Position (x',y') eines Pixels nach einer Drehung (Rotation) um den Ursprung (0,0) mit dem Winkel α wird wie folgt berechnet:

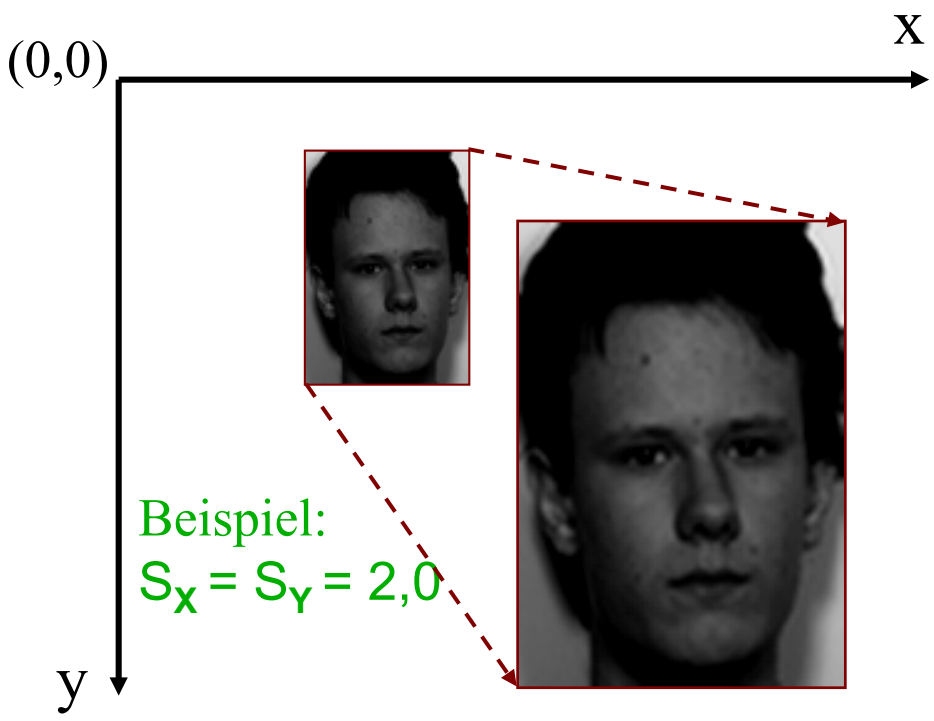
$x' = x \cos \alpha + y \sin \alpha$

$y' = -x \sin \alpha + y \cos \alpha$

Die Transformationsgleichung für die Rotation in Vektorschreibweise:

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

Skalierung



Die Position (x',y') eines Pixels nach einer Skalierung relativ zum Ursprung (0,0), mit dem horizontalen Skalierungsfaktor S_X und dem vertikalen Skalierungsfaktor S_Y wird wie folgt berechnet:

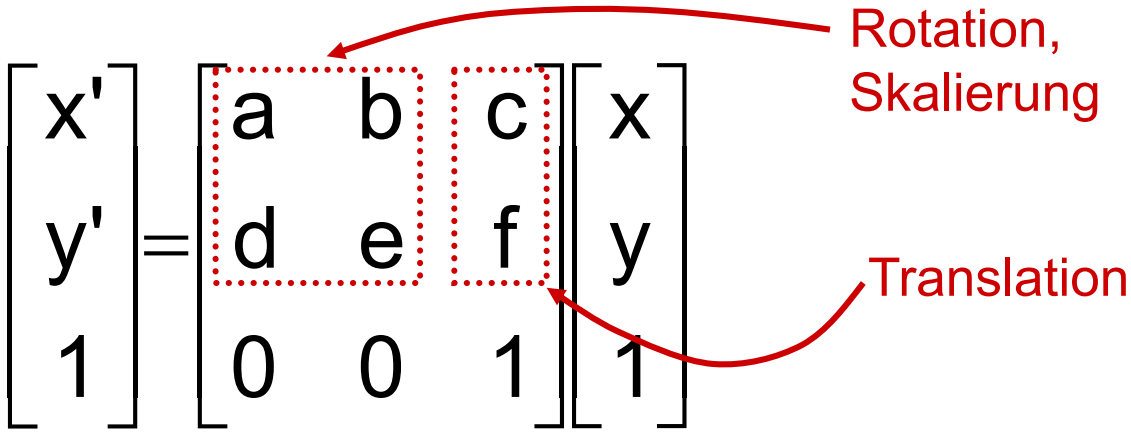
$$x' = x\, S_X$$
$$y' = y\, S_Y$$

Die Transformationsgleichung für die Skalierung in Vektorschreibweise:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine Transformationen in Matrix-Darstellung: Homogene Koordinaten

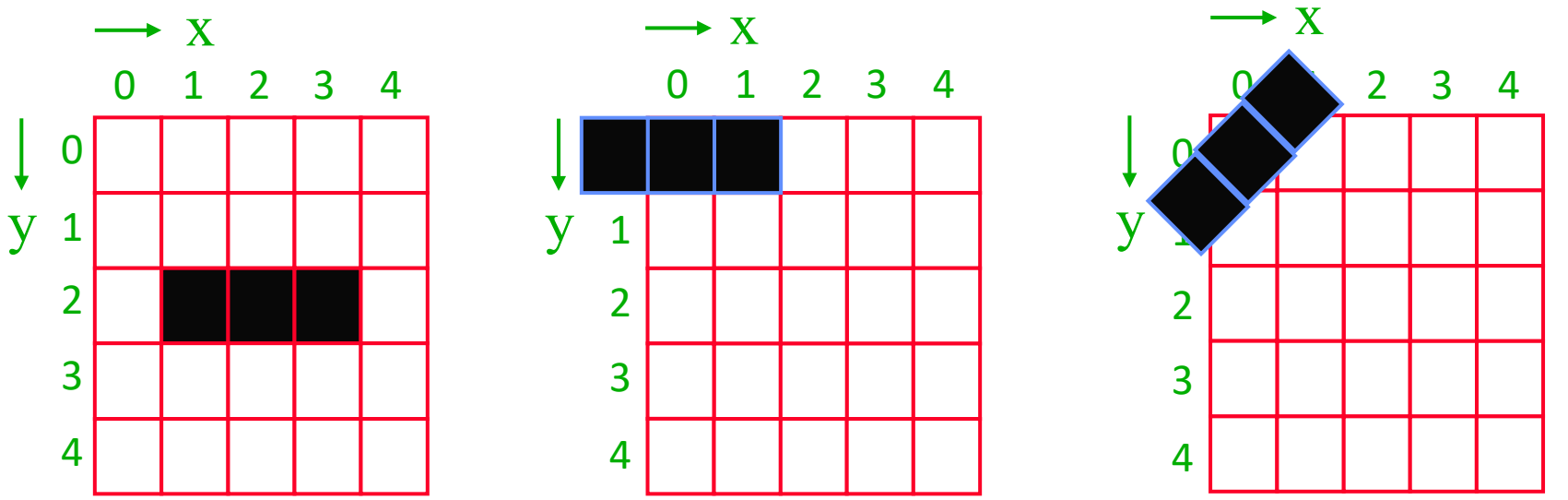
- Zweidimensionale lineare Transformationen lassen sich einheitlich durch eine 3x3 Matrix darstellen.
- Die Transformation eines Punkts (x, y) zu einem neuen Punkt (x', y') durch eine beliebige Folge von Translationen, Rotationen und Skalierungen wird beschrieben durch:



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Zwei oder mehrere affine Transformationen, die hintereinander ausgeführt werden sollen, lassen sich Verketten (Konkatenation) und durch die Anwendung **einer** Transformationsmatrix ausführen.

Affine Transformation, Beispiel (1):
Rotation und Skalierung einer Linie aus drei Pixeln



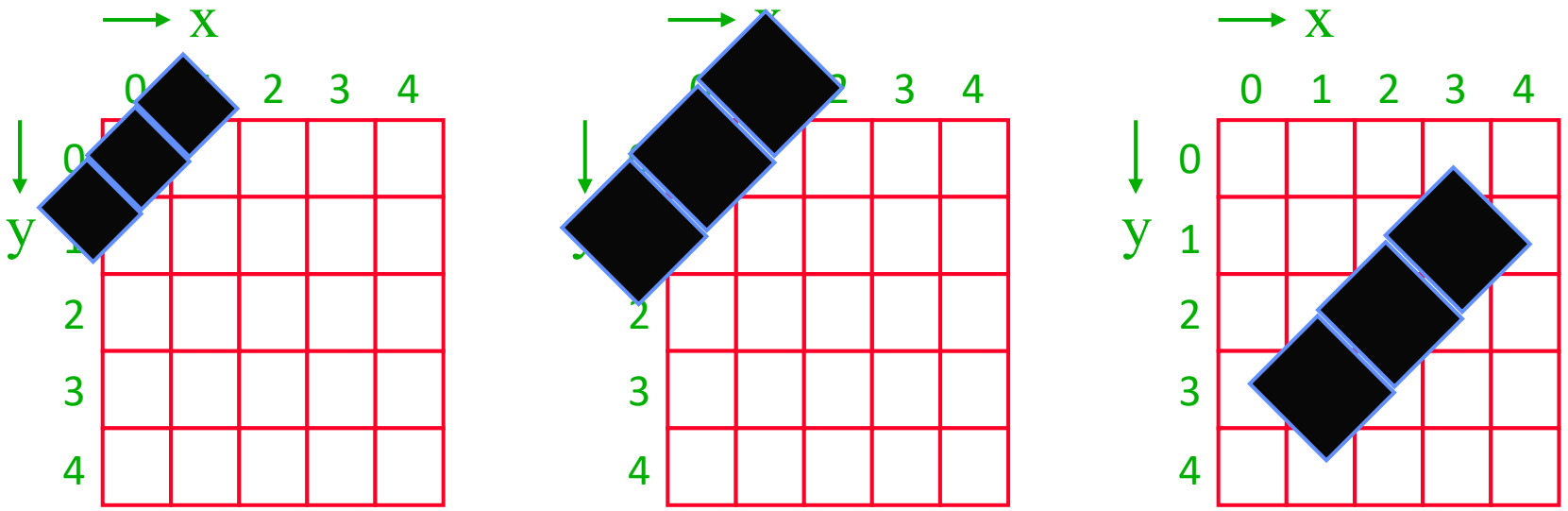
$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation (T_X=-2 / T_Y=-2)

$$\begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Drehung um α=45°

Affine Transformation, Beispiel (2):
Rotation und Skalierung einer Linie aus drei Pixeln



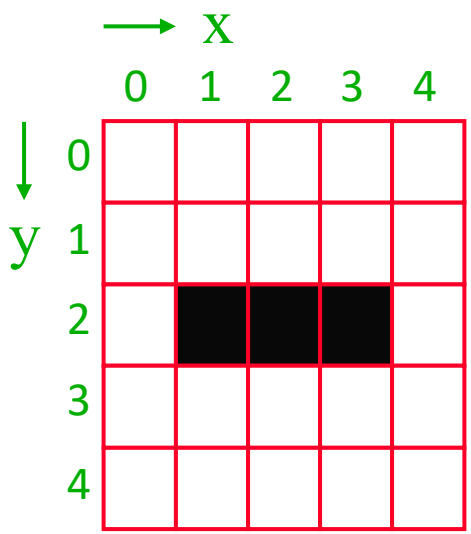
$$\begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Skalierung (S_X=1,4142 / S_Y=1,4142) Translation (T_X=2 / T_Y=2)

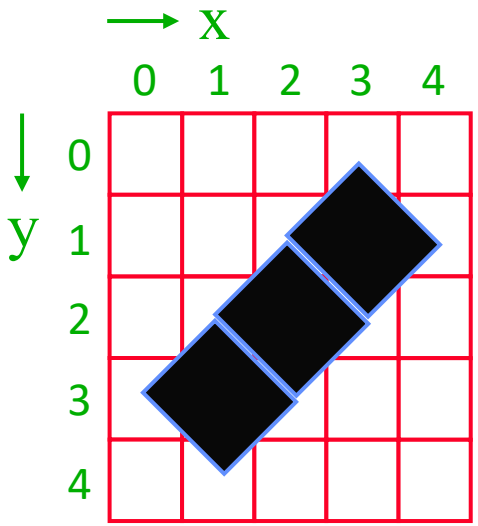
Affine Transformation, Beispiel (3):

Rotation und Skalierung einer Linie aus drei Pixeln

Verkettung der Teiltransformationen



$$\begin{bmatrix} 1 & -1 & -2 \\ 1 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -2 \\ 1 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

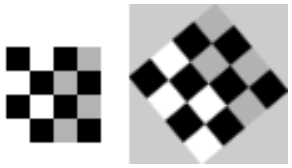
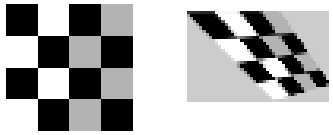
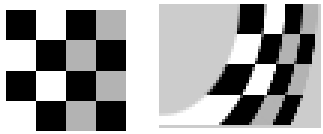
Verkettung: Multiplikation der Matrizen in der Reihenfolge der Transformationen

Typisierung analytischer Transformationen

Euclidean / Procrustes	x'	$=$	$ax + by + t_x$	, $a = s\cos\alpha$
	y'	$=$	$-bx + ay + t_y$	, $b = s\sin\alpha$
Affine / 1^{st} order polynomial	x'	$=$	$a_0 + a_1x + a_2y$	
	y'	$=$	$b_0 + b_1y + b_2y$	
Bilinear	x'	$=$	$a_0 + a_1xy + a_2x + a_3y$	
	y'	$=$	$b_0 + b_1xy + b_2x + b_3y$	
Perspective	x'	$=$	$(a_0 + a_1x + a_2y)/(c_0x + c_1y + 1)$	
	y'	$=$	$(b_0 + b_1x + b_2y)/(c_0x + c_1y + 1)$	
2^{nd} order polynomial / Biquadratic	x'	$=$	$a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy$	
	y'	$=$	$b_0 + b_1x + b_2y + b_3x^2 + b_4y^2 + b_5xy$	
General polynomial	x'	$=$	$\sum_i \sum_j a_{ij}x^iy^j$	
	y'	$=$	$\sum_i \sum_j b_{ij}x^iy^j$	

Typisierung analytischer Transformationen

- **Eigenschaften, Aussehen**

Euklid / Procrustes	Geraden bleiben Geraden, Quadrate bleiben Quadrate	
Affin	Geraden bleiben Geraden, Parallelen bleiben parallel, Rechteck → Parallelogramm	
Bilinear	Geraden werden zu Kurven	
perspektivisch (projektiv)	Geraden bleiben Geraden, Parallelen → Geraden mit gemeinsamen Fluchtpunkt	
Biquadratisch	Geraden werden zu Kurven	

Typisierung analytischer Transformationen

- **Anzahl Kontrollpunkte ≥ Anzahl Freiheitsgrade/2**

Euklid / Procrustes	Kombination Translation + Rotation + globale Skalierung	mind. 2 Kontrollpunkte
Affin	Kombination Translation + Rotation + x-Skalierung + y-Skalierung (also Scherung)	mind. 3 Kontrollpunkte
Bilinear		mind. 4 Kontrollpunkte
perspektivisch (projektiv)		mind. 4 Kontrollpunkte
Biquadratisch		mind. 6 Kontrollpunkte

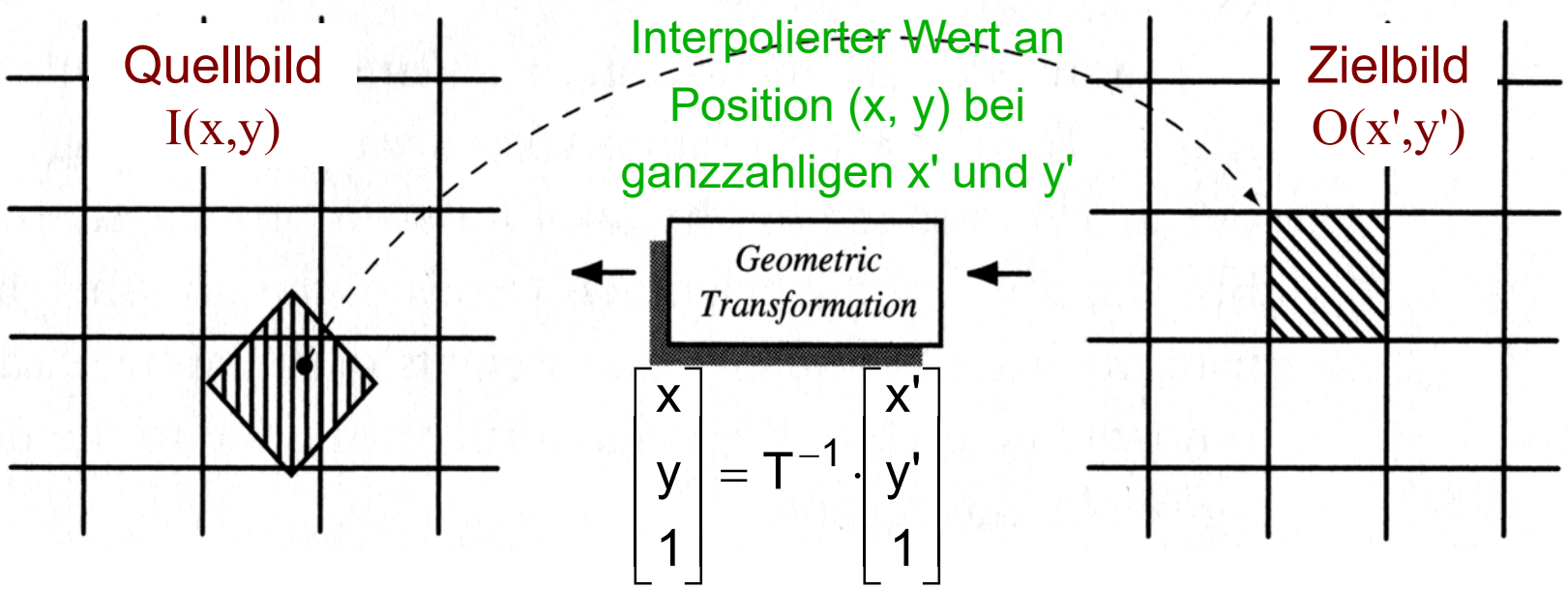
Inhalt

- Einführung
- Transformationen und Homogene Koordinaten
- **Interpolation der Grauwerte**
  - **nächster Nachbar**
  - **bilinear**
- Nichtlineare Transformationen
- Warping-Algorithmus

Interpolation bei geometrischen Bildtransformationen (1)

Wieso besser als source-to-target?

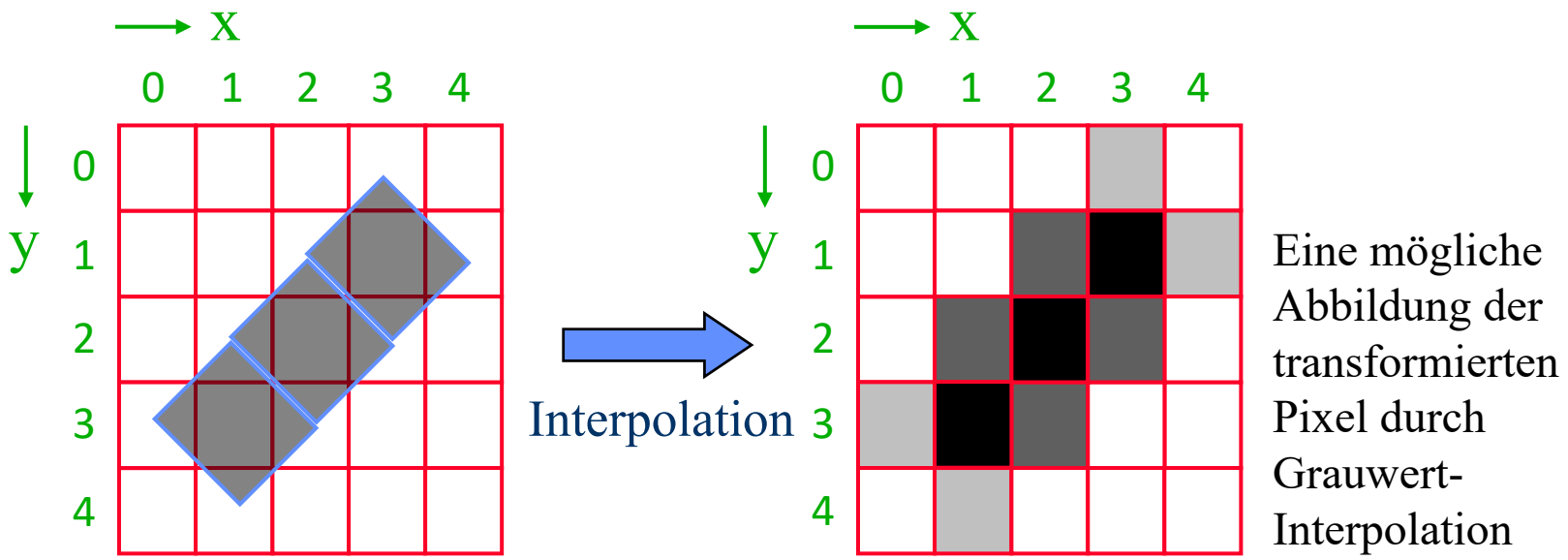
- Bei der geometrischen Transformation von Rasterbildern wird meist eine Ziel-zu-Quelle (target-to-source, backwards) Abbildung implementiert.
- Im Zielbild wird jedes Pixel mit einem Wert besetzt, der durch Interpolation aus einer Pixelnachbarschaft im Quellbild errechnet wird (*resampling*).



$\mathbf{T}^{-1}$ ist die Inverse der Transformationsmatrix für die Quelle-zu-Ziel Abbildung $\mathbf{T}$

Interpolation bei geometrischen Bildtransformationen (2)

- Ein transformiertes Pixel überdeckt (ganz oder teilweise) ein oder mehrere Rasterzellen im Zielbild.
- Ein einfaches Verfahren zur Interpolation der Grauwerte im Zielbild beruht darauf, den Grauwert aus allen beteiligten Quellbild-Pixeln zu berechnen, wobei diese proportional zu ihren Flächenanteilen gewichtet werden.



Interpolation bei geometrischen Bildtransformationen (2)

- Die in der Praxis am häufigsten eingesetzten Interpolationsverfahren bei Bildtransformationen sind:
- Nächster-Nachbar-Interpolation (nearest neighbor)
- Bilineare Interpolation (bilinear)

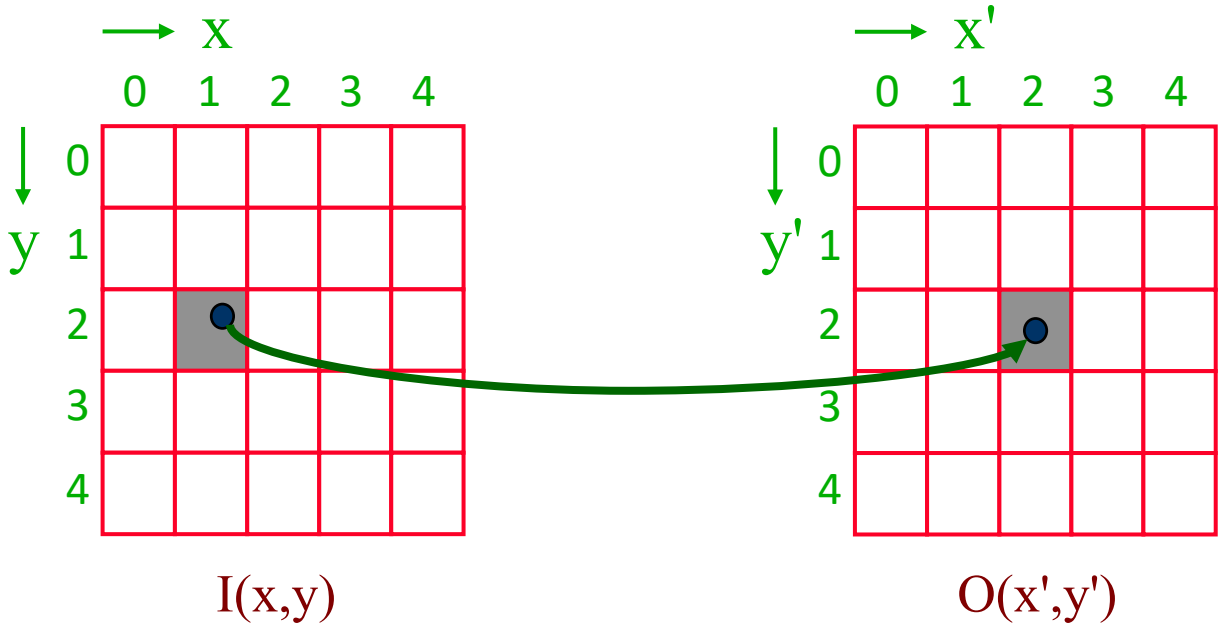
  Interpolation 1. Ordnung aus 4-er Bildnachbarschaft
- Bikubische Interpolation (bicubic)

  Interpolation 2. Ordnung aus 16-er Bildnachbarschaft

Wieso nicht biquadratisch? – kommt später!

Nächster-Nachbar-Interpolation (nearest neighbor)

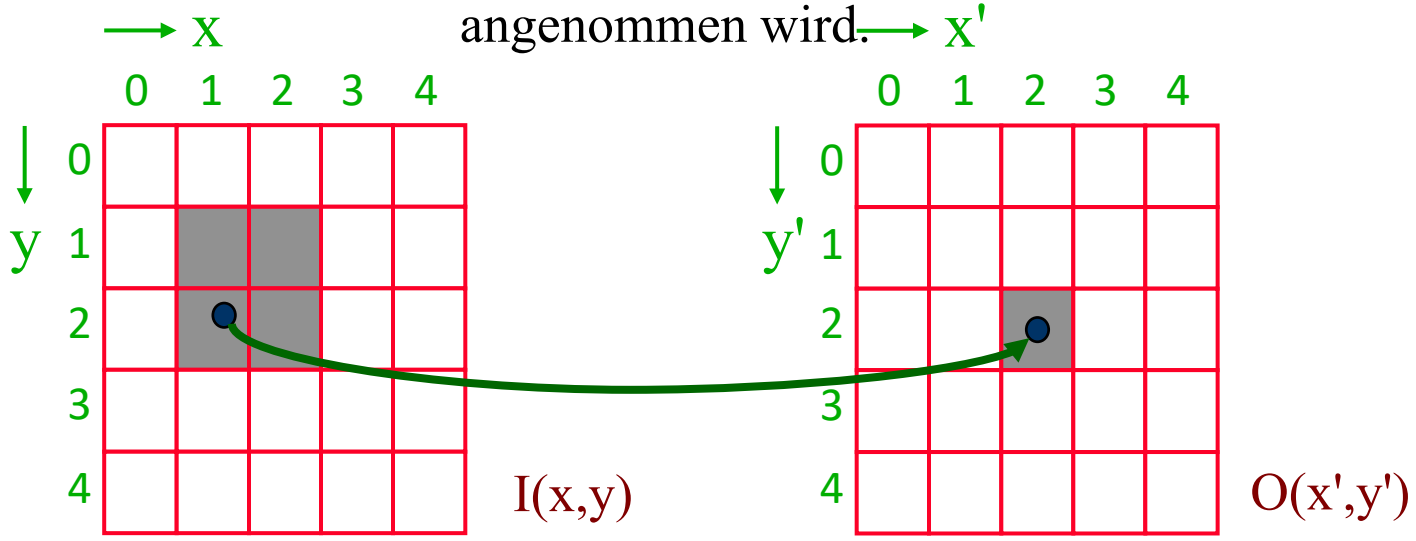
Die Nächster-Nachbar Interpolation weist dem Punkt (x',y') den Grauwert des nächsten Punkts I(x,y) im diskreten Raster zu.



$$O(x',y') = I(\text{ROUND}(x), \text{ROUND}(y))$$

Bilineare Interpolation

Die bilineare Interpolation berechnet den Grauwert aus den 4 nächsten Nachbarpunkten im Raster, wobei ein lokal linearer Grauwertverlauf angenommen wird.



$$O(x',y') = (1-a)(1-b)\, I(n, m) + a\,(1-b)\, I(n+1, m) + (1-a)\, b\, I(n, m+1) + a\, b\, I(n+1, m+1)$$

$a = x - n$
$b = y - m$

$n = \text{floor}(x)$*
$m = \text{floor}(y)$
*größte ganze Zahl kleiner x

Effekt der verschiedenen Interpolationsverfahren
Beispiel: Bildvergrößerung (magnification)

bikubische Interpolation

bilineare Interpolation

nächster Nachbar

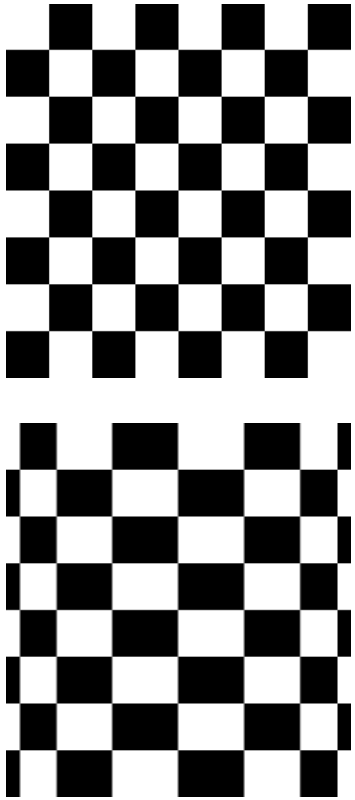
Bildskalierung um Faktor 2

Inhalt

- Einführung
- Transformationen und Homogene Koordinaten
- Interpolation der Grauwerte
  - nächster Nachbar
  - bilinear
- **Nichtlineare Transformationen**
- Warping-Algorithmus

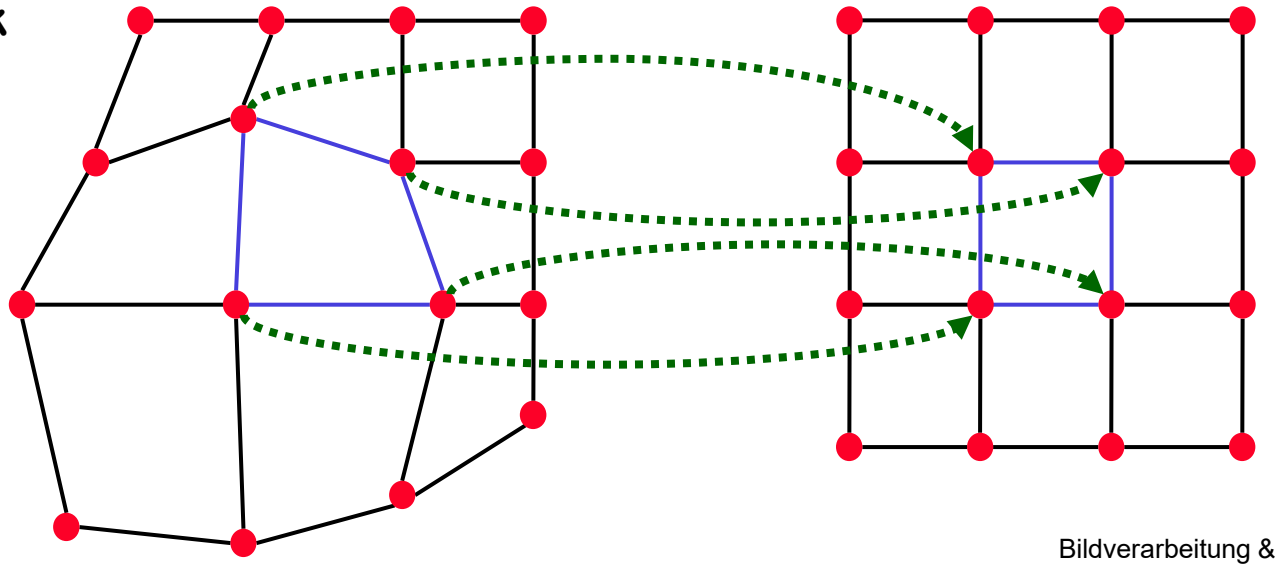
Nicht-lineare geometrische Transformationen

- Bei den nicht-linearen Transformationen variiert die lokale Bildverzerrung (ortsvarianter Flächenvergrößerungsfaktor).
  Beispiele:
  Perspektivische, sphärische und zylindrische Abbildungen.

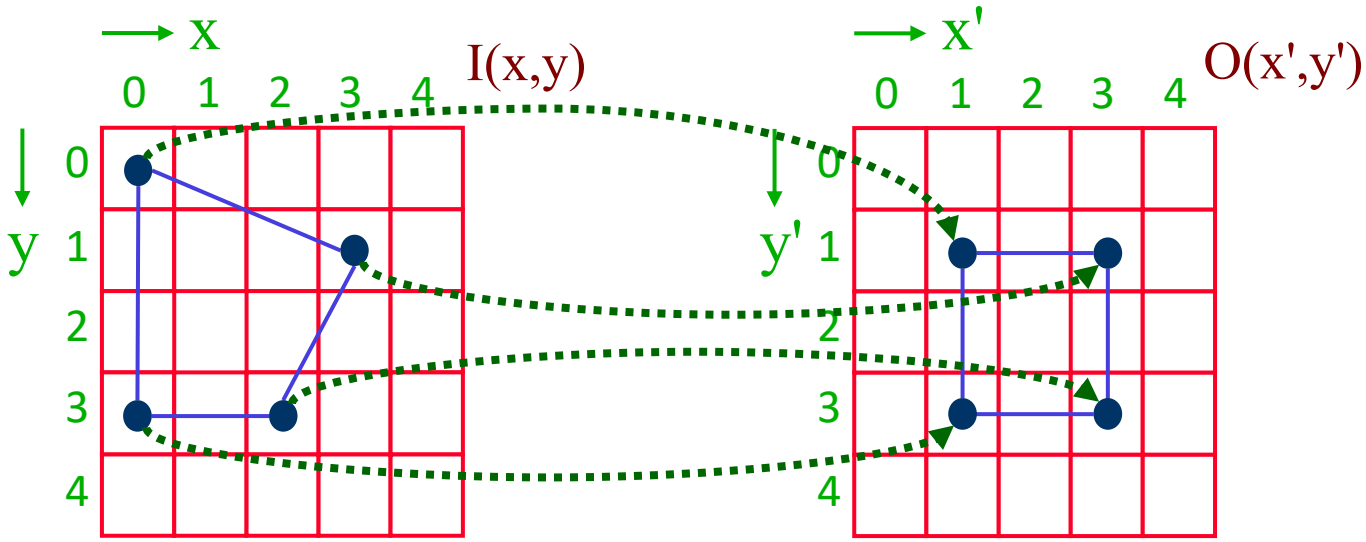


Korrektur von nicht-linearen Bildverzerrungen durch eine stückweise lineare Transformation

- Bei dem sogenannten **Warping** werden für eine Menge von ausgewählten Passpunkten (landmarks, fiducial points) im Quellbild die jeweils korrespondierenden Punkte im Zielbild bestimmt.
- Um eine geometrische Bildverzerrung zu korrigieren kann man ein Gitter von Passpunkten erstellen, das auf ein regelmäßiges (unverzerrtes) Gitter abgebildet werden soll.
- Die gesamte Bildtransformation wird in Regionen aufgeteilt, wobei jede Region im Gitter ein Viereck darstellt.



Herleitung einer stückweise linearen Warping-Transformation



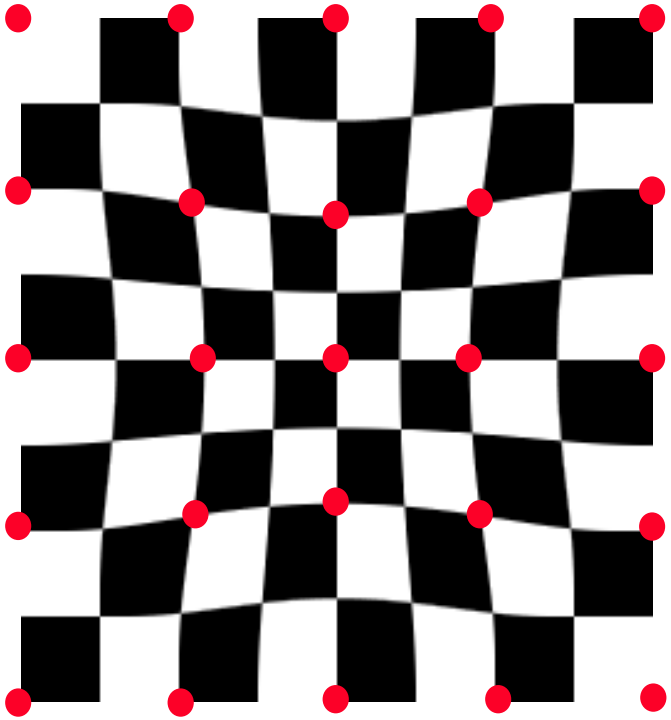
Ein beliebiges Viereck (4-seitiges Polygon) kann mit einer bilinearen Transformation auf ein anderes Viereck (z.B. Quadrat) abgebildet werden:

$$x' = a_1 x + a_2 y + a_3 x y + a_4$$

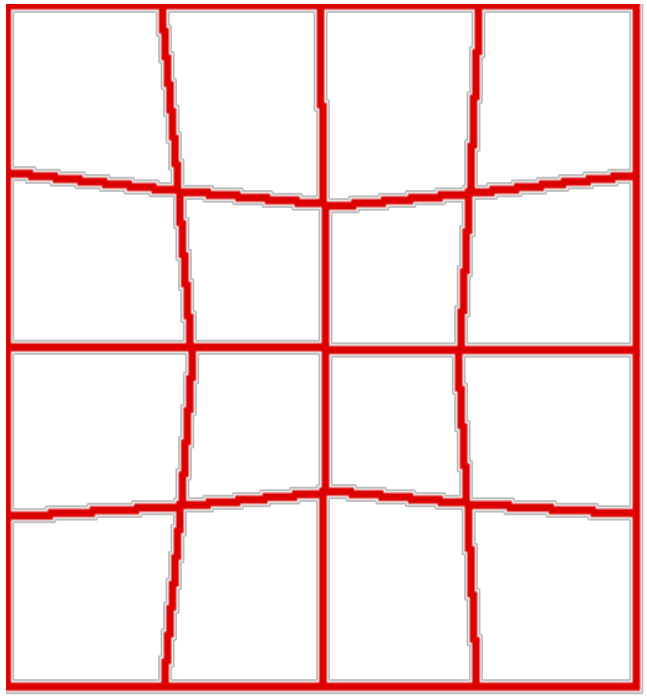
$$y' = b_1 x + b_2 y + b_3 x y + b_4$$

Mit den korrespondierenden 4 Passpunkten erhält man acht Gleichungen. Dieses Gleichungssystem wird nach den 8 Unbekannten ($a_1 ... a_4$, $b_1 ... b_4$) aufgelöst.

Beispiel: Restauration eines verzerrten Schachbrettmusters (1)



25 manuell gewählte Passpunkte auf dem verzerrten Schachbrettbild



Verzerrtes Gitter

Beispiel: Restauration eines verzerrten Schachbrettmusters (2)

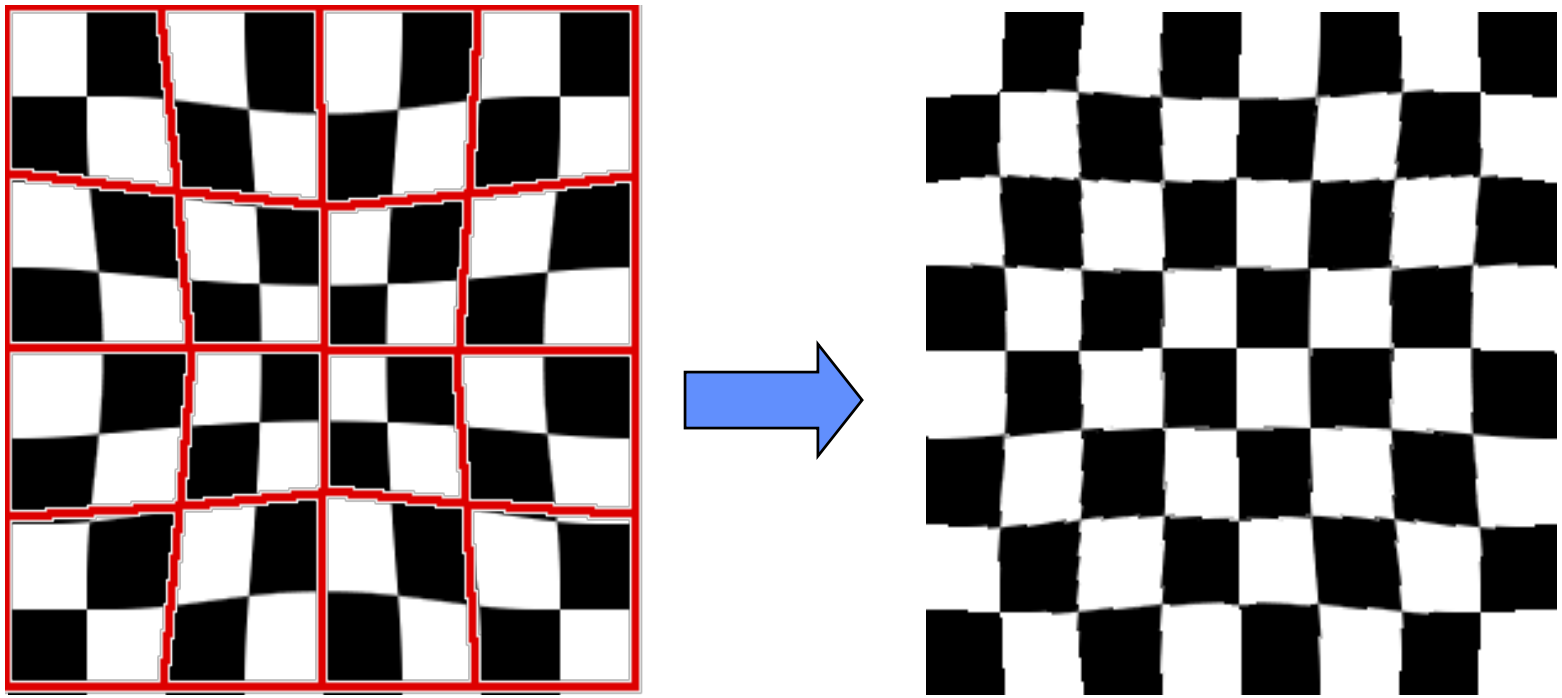
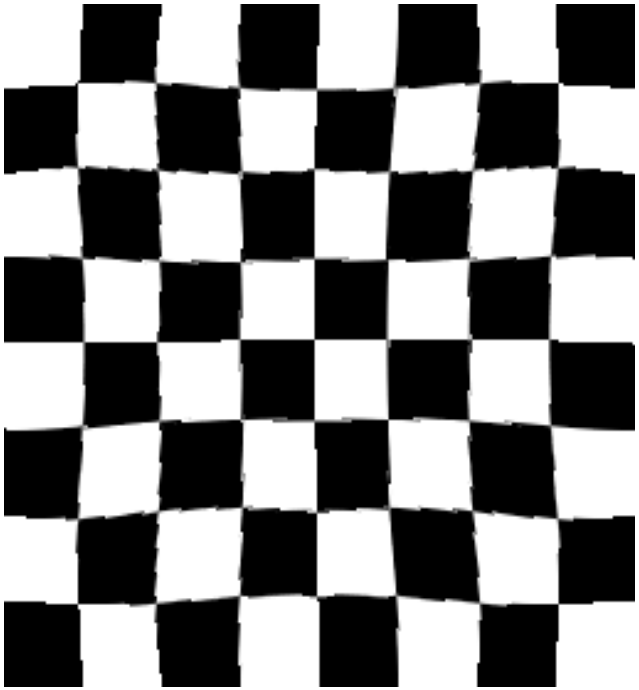
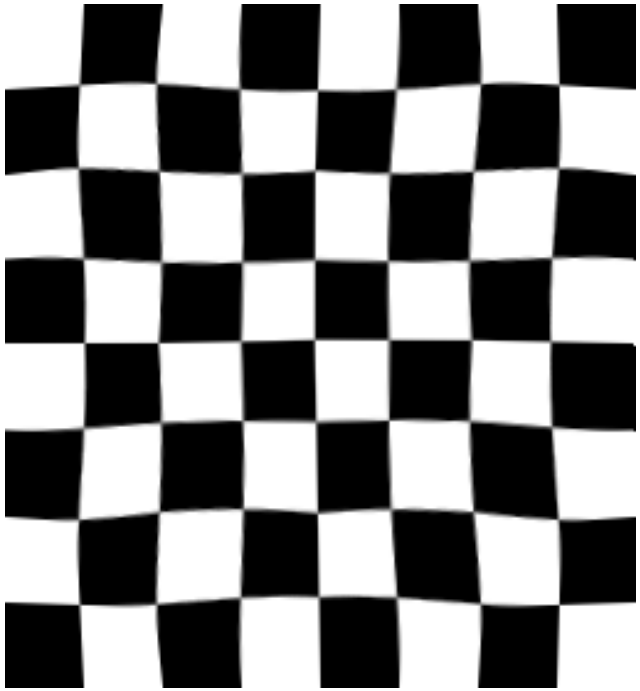


Abbildung eines verzerrten Gitters auf das regelmäßige Normalgitter.

Einfluss des Interpolationsverfahrens beim Warping



"Nächster Nachbar"



"Bi-linear"

Inhalt

- Einführung
- Transformationen und Homogene Koordinaten
- Interpolation der Grauwerte
  - nächster Nachbar
  - bilinear
- Nichtlineare Transformationen
- **Warping-Algorithmus**

Die Schritte des Warping-Algorithmus

- **Schritt 1**: **Passpunkte finden**
  - automatisch oder manuell
- **Schritt 2**: **Suche zu den Passpunkten die beste Transformation**
  - analytisch (z.B. Vertreter aus Klasse der affinen Transformationen)
  - oder stückweise linear für jeden Pixelort **x'** im Zielbild (**Gridding**)
- **Schritt 3**: **Grauwerte der Zielpixel festlegen**
  - durch Interpolation am Quellort (nächster Nachbar, bilinear, bikubisch, ...)

Warping, Schritt 1
Passpunkte finden

- **manuell**
  - o.k. für Bild**be**arbeitung, aber keine (automatisierte) Bild**ver**arbeitung
  - mühsam, wenn viele Bilder
  - ungenau: optional: Passpunkte per Kreuzkorrelation verbessern
  - Weitere Anwendung: Kalibrierung
- **automatisch: möglich, wenn korrespondierende Punkte in 2 Bildern vorliegen**
- **diese Aufgabe heißt Matching oder Registrierung**
- **2 Teilprobleme:**
  - das Auffinden geeigneter Punkte im Bild
    - "salient points", Landmarken
    - Kanten ungeeignet >> Aperturproblem (!)
    - Ecken >> s. Projekt Eckendetektor
  - das genaue Wiederfinden im anderen Bild
    - Template Matching
    - s. Projekt Tracking

Warping, Schritt 2
Geometrische Transformation T

- **Wie genau macht man den Warp in ImageJ/Jama?**

1. T aus Passpunkten berechnen:

   a) wenn T aus bestimmter Klasse (z.B. Euklid) sein soll: beste Transformation im Least-Square-(LS-)Sinne suchen

   b) freier Warp: stückweise-linear interpolieren zwischen Passpunkten

2. T invertieren (T^{-1}) und auf jedes (homogene) Ziel-Pixel anwenden

Warping, Schritt 2
2.1.a) Bestes T mit Least-Square-Fit

- Zu 2.1.a): Seien mehr Passpunkte gegeben als man zur Bestimmung von T braucht. Bsp. Rotationsstreckung:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \Leftrightarrow \quad \begin{cases} x' = ax + by \\ y' = -bx + ay \end{cases} \quad \Leftrightarrow \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x & y \\ y & -x \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Der Trick: Jeder freie Parameter a,b,... taucht genau 1x auf

- Übergang auf mehrere Passpunkte:

$$\begin{pmatrix} x_1' \\ \vdots \\ x_n' \\ y_1' \\ \vdots \\ y_n' \end{pmatrix} = \begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \\ y_1 & -x_1 \\ \vdots & \vdots \\ y_n & -x_n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \quad \Leftrightarrow \quad \mathbf{X'} = \mathbf{Zt}$$

$\mathbf{X'}$ $\mathbf{Z}$

Dieses i.d.R. überbestimmte System für Parametervektor $\mathbf{t} = (a,b)^T$ wird durch den Jama[1]-Befehl

```
Matrix t = Z.solve(Xp);
```

im Least-Square-Sinne gelöst.

[1]Jama: http://math.nist.gov/javanumerics/jama/

Warping, Schritt 2
2.2 gegegebenes T

- Zu 2.2.: aus Parametervektor **t** können wir homogene 3x3-Matrix **T** bilden, z.B.:

$$t = \begin{pmatrix} a \\ b \end{pmatrix} \quad \Rightarrow \quad \mathbf{T} = \begin{pmatrix} a & b & 0 \\ -b & a & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

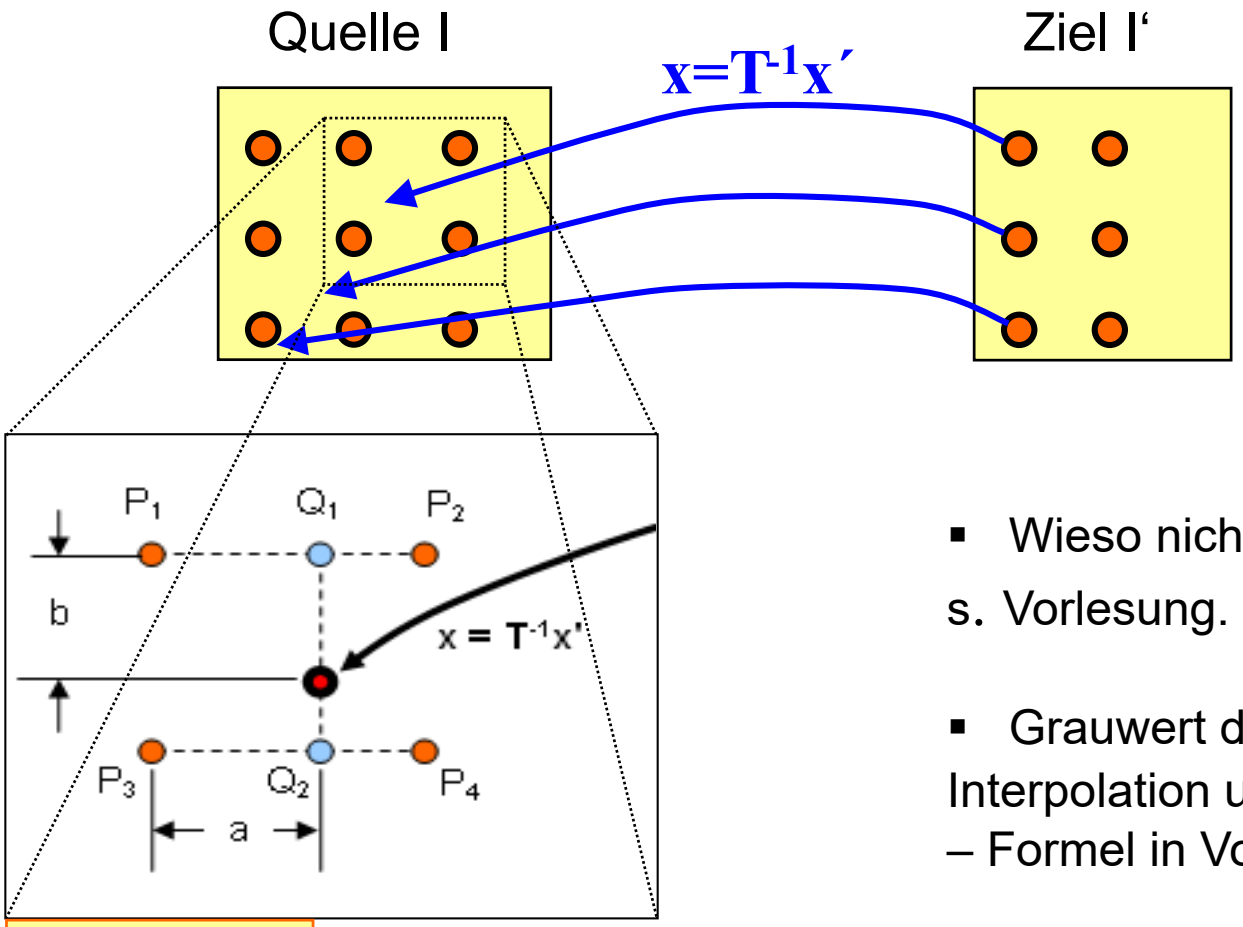
- Jama[1]-Befehle für jedes Zielpixel (x',y')=(i,j)

```
Matrix T_inverse = T.inverse();
double [][] valXP = {{i},{j},{1.0}};
Matrix XP = new Matrix(valXP);
Matrix X = T_inverse.times(XP);
double x = X.get(0,0);
double y = X.get(1,0);
```

[1]Jama: http://math.nist.gov/javanumerics/jama/

Warping, Schritt 3
(Transformationen T und T^{-1} seien gegeben)

- Backwards-Warp: Zielpixelort **x'** wird durch Quellort **x** koloriert



Quelle: aufg_loes.htm

- Wieso nicht Vorwärts-Warp? – s. Vorlesung.

- Grauwert durch z.B. bilineare Interpolation um Quellort **x** herum – Formel in Vorlesung

Anwendungen des Warping-Algorithmus

- Kamerakalibrierung
  - (Kissen- oder Tonnenverzeichnung korrigieren)
- Industrielle Prüfaufgaben für deformierbare Objekte
- Luftaufnahmen Landschaft zur Deckung bringen
  - ... und dann Differenz, um Unterschiede zu sehen (Ökologie)
- Inverse Perspektive
- Medizinische Bildverarbeitung
  - Matching und Registrierung bei multimodalen, multitemporalen Aufnahmen >> s. Projekt Matching
- Zwischenbilder berechnen >> s. Projekt Morphing
- Facial Animation, 3D-MM (Morphable Models) [Blanz, Vetter] >> s. reanim_exercise.mpg
- Image Mosaicing (mehrere Bilder zusammensetzen) >> s. Projekt Panoramic View