Prof. Dr. Matthias O. Franz

UMBILDUNG

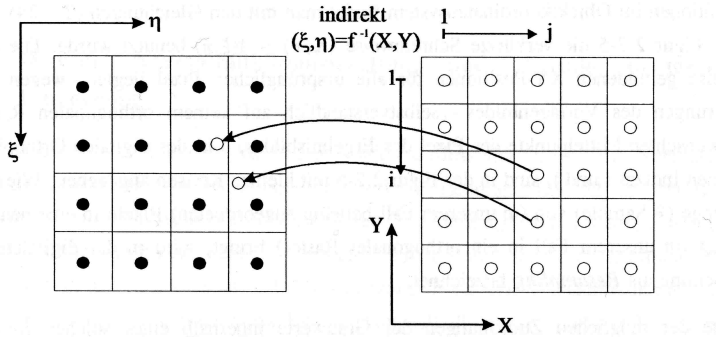
Übungsblatt 1, Computer Vision

Affine Transformation von Bildern

Mit einer affinen Transformation in der Ebene der Form

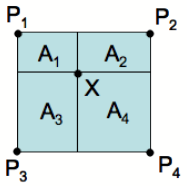
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_{10} \\ a_{20} \end{pmatrix} \quad \text{bzw.} \quad \vec{x}' = A\vec{x} + \vec{a}_0$$

lassen sich viele in der Praxis wichtigen Bildtransformationen beschreiben (z.B. Drehungen, oder Skalierungen).



[Kraus, 2004]

Die häufigste Methode zur Durchführung solcher Bildtransformationen ist die *indirekte Umbildung* (s. Diagramm). Sie beginnt mit einer zunächst leeren Bildmatrix (oft größer gewählt als das Ausgangsbild). Anschließend werden die Koordinaten der Pixelmittelpunkte mit der inversen Transformation auf ihre ursprüngliche Position im Ausgangsbild zurückgerechnet und der dortige Grauwert ausgelesen. Da die Ausgangspositionen fast nie auf einem Pixelmittelpunkt liegen, muß entweder der Grauwert des nächstliegenden Pixelmittelpunktes genommen (im unten gezeigten Beispiel wäre das P_1) oder *bilinear interpoliert* werden.



Für die bilineare Interpolation wird ein gewichteter Mittelwert aus den Grauwerten der 4 Nachbarpunkte berechnet (s. Zeichnung): jeder Nachbarpunkt wird dabei mit dem diagonal gegenüberliegenden Flächenanteil gewichtet, d.h. P_1 mit A_4, P_2 mit A_3, P_3 mit A_2 und P_4 mit A_1. Der interpolierte Grauwert an der Stelle X ist dann

$$g_X = A_4 g(P_1) + A_3 g(P_2) + A_2 g(P_3) + A_1 g(P_4).$$

Damit erhalten die Nachbarpunkte ein um so höheres Gewicht, je näher sie an X liegen.

Aufgaben:
a. Schreiben Sie eine Python-Funktion, mit der sich ein Bild affin transformieren läßt. Übergabeparameter sind die Matrix A, der Verschiebungsvektor $\vec{a}_0$ und das Eingangsbild, zurückgegeben wird das affin verzerrte Ausgangsbild. Sehen Sie dabei vor, daß der Benutzer zwischen Nächster-Nachbar- und bilinearer Interpolation wählen kann. Alle Pixel, deren Ausgangsposition außerhalb des Eingangsbildes liegen, werden auf 0 gesetzt. Die Funktion soll sowohl Grauwert- als auch Farbbilder verarbeiten können.

b. Führen Sie folgende affine Transformationen durch geeignete Wahl der Matrix A aus: (1) Drehen Sie ein beliebiges Bild (z.B. *gletscher.jpg*) um 30°; (2) verkleinern Sie es um den Faktor 0.7; (3) verkleinern Sie es in x-Richtung um den Faktor 0.8, vergrößern Sie es in y-Richtung um den Faktor 1.2; (4) Dehnen Sie das Bild entlang der Diagonalen um 1.5, stauchen Sie senkrecht dazu um 0.5; (5) Entzerren Sie das Objekt zu Füßen der Botschafter im Bild *ambassadors.jpg*. Vergleichen Sie dabei die Ergebnisse für Nächste-Nachbar- und bilineare Interpolation.

Nützliche Python-Funktionen: (1) zur Darstellung von Bildern können Sie die Funktion `imshow()` aus dem Python-Paket *matplotlib* verwenden; (2) Bilder können am bequemsten mit `skimage.data.imread()` aus dem Paket `scikit-image` eingelesen werden; (3) Wenn Bilder in Python eingelesen werden, haben sie meist den Datentyp *uint8*, für den nur eine begrenzte Funktionalität zur Verfügung steht. Wandeln Sie daher die Bilder nach dem Einlesen zuerst mit der Funktion `numpy.astype()` in ein Fließkommaformat um und skalieren Sie das Bild zwischen 0 und 1; (4) Die Inverse einer Matrix wird mit der Funktion `numpy.linalg.inv()` berechnet.