Prof. Dr. Matthias O. Franz

Affine Transformation von Bildern

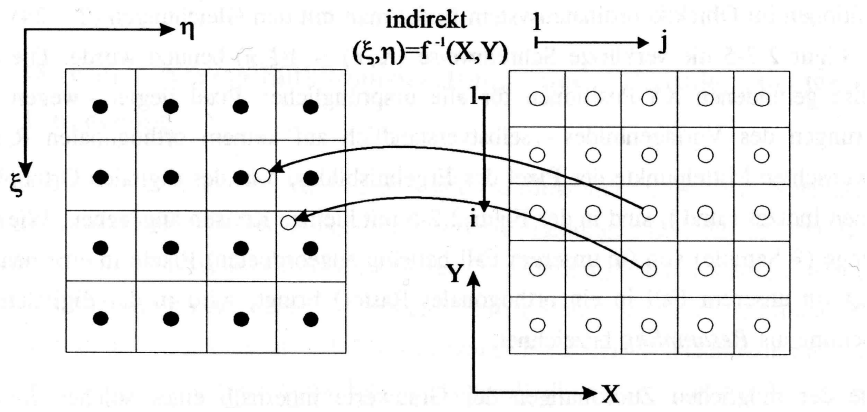
Übungsblatt 1, Computer Vision

Bitte geben Sie Ihre Ergebnisse in Form eines Jupyter-(Lab)-Notebooks ab!

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).



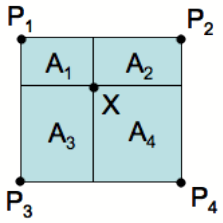
[Kraus, 2004]

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.

Bitte beachten Sie, dass die y-Achse im Bild entgegen der sonst üblichen Konvention nach unten statt nach oben zeigt, d.h. bei der in der Mathematik übliche Konvention handelt es sich um ein *rechtshändiges* Koordinatensystem, während in Bildern ein *linkshändiges* Koordinatensystem verwendet wird. Damit Operationen wie Drehungen oder Verschiebungen

das erwartete Ergebnis zeigen, ist es sinnvoll, die Transformation zwischen links- und rechtshändigem Koordinatensystem explizit zu implementieren. Bei der indirekten Umbildung heißt das, dass wir die Gesamttransformation in drei explizite Schritte unterteilen, die hintereinander ausgeführt werden: 1. Transformation linkshändiges KS (Zielbild) in rechtshändiges KS; 2. inverse affine Transformation; 3. Transformation rechtshändiges KS in linkshändiges KS (Ausgangsbild). Die Gesamttransformation ergibt sich dann mathematisch durch die Multiplikation der entsprechenden Matritzen.

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, aber implementieren Sie zunächst nur die einfachere Nächster-Nachbar-Interpolation. Alle Pixel, deren Ausgangsposition außerhalb des Eingangsbildes liegen, werden auf 0 gesetzt. Die Funktion soll sowohl Grauwert- als auch Farbbilder verarbeiten können.

b. Wie sieht die affine Transformation von einem linkshändigen Koordinatensystem in ein rechtshändiges Koordinatensystem aus? Wie deren Umkehrung? Implementieren Sie eine Gesamttransformation zur affinen Umbildung, bestehend aus der Folge 1. Transformation links- in rechtshändiges KS; 2. inverse affine Transformation; 3. Transformation rechts- in linkshändiges KS. Testen Sie Ihre Funktion, indem Sie ein beliebiges Bild (z.B. *gletscher.jpg*) um 20 Pixel diagonal nach rechts unten verschieben, wobei der zugehörige Translationsvektor in rechtshändigen Koordinaten gegeben ist.

c. Ergänzen Sie nun Ihre Funktion um die bilineare Interpolation. Testen Sie Ihre Interpolation, indem Sie ein kleines Testbild (ca. 30×30 Pixel groß) um den Faktor 10.5 vergrößern. Wie sehen die vergrößerten Pixel bei den beiden Interpolationsmethoden aus?

d. 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 in x-Richtung um den Faktor 0.8, vergrößern Sie es in y-Richtung um den Faktor 1.2; (3) Dehnen Sie das Bild entlang der Diagonalen um 1.5, stauchen Sie senkrecht dazu um 0.5; (4) Entzerren Sie das Objekt zu Füßen der Botschafter im Bild *ambassadors.jpg*. Vergleichen Sie dabei die Ergebnisse für Nächste-Nachbar- und bilineare Interpolation. Hierbei ist es hilfreich, iterativ vorzugehen, d.h. zuerst das Bild grob zu entzerren und dann schrittweise zu verfeinern, jeweils durch *trial and error*.

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.