ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
INSTITUT FÜR INFORMATIK
Lehrstuhl für Mustererkennung und Bildverarbeitung
Prof. Dr.-Ing. Hans Burkhardt

Georges-Köhler-Allee
Geb. 052, Zi 01-029
D-79110 Freiburg
Tel. 0761 - 203 - 8260

Übungen zur Vorlesung
Grundlagen der Bilderzeugung und Bildanalyse WS 05/06

Musterlösung 3

Aufgabe 3.1: $\mathbb{C}T$-Transformation

Gesucht wird die Menge aller bezüglich $\mathbb{C}T$ invarianten Veränderungen eines Musters $\mathbb{I}_{\mathbb{C}T}(\mathbf{x})$ für eine beliebige 4-dimensionale Transformation der Klasse $\mathbb{C}T$.

Nach Kap.3a Seite 34 lässt sich $\mathbb{I}_{\mathbb{C}T(\mathbf{x})}$ als Verieinigung von allen zyklischen Permutationen von $\mathbf{x}$ und allen zyklischen Permutationen des gespiegelten Musters $\psi(\mathbf{x})$ darstellen.

Wir wollen also zeigen:

$$\mathbb{I}_{\mathbb{C}T}({}^4\mathbf{x}) = \mathcal{T}(\mathbf{x}) \cup \mathcal{T}(\psi(\mathbf{x}))$$

$$= \left\{ \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_1 \\ x_0 \\ x_3 \end{bmatrix}, \begin{bmatrix} x_0 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_3 \\ x_0 \\ x_1 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_0 \\ x_3 \\ x_2 \end{bmatrix}, \begin{bmatrix} x_3 \\ x_0 \\ x_1 \\ x_2 \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_0 \end{bmatrix}, \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \right\}$$

$$= \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_8\}$$

Beweis:

Wir benötigen zunächst folgende rekursive Definition für $\widetilde{\mathbf{x}} = \mathbb{C}T(\mathbf{x})$:

$$\widetilde{\mathbf{x}} = \begin{bmatrix} f_1(\widetilde{\mathbf{x}_{1/2}, \mathbf{x}_{2/2}}) \\ f_2(\widetilde{\mathbf{x}_{1/2}, \mathbf{x}_{2/2}}) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}^{(1)}_{1/2}} \\ \widetilde{\mathbf{x}^{(1)}_{2/2}} \end{bmatrix},$$

f_1, f_2 kommutativ.

Der Signalgraph für eine 4-dimensionale Transformation der Klasse $\mathbb{C}T$ hat zwei Schichten. In der ersten Schicht werden das 1. & 3. und das 2. & 4. Element der nullten Schicht (Originalvektor) verknüpft. In der zweiten Schicht werden das 1. & 2. und das 3. & 4. Element der 1.Schicht verknüpft.

Zusammen mit der obigen Definition für die Klasse $\mathbb{C}T$ ergibt sich für den transformierten Vektors $\widetilde{\mathbf{x}}$:
$\widetilde{x_0} = f_1(f_1(x_0, x_2), f_1(x_1, x_3))$
$\widetilde{x_1} = f_2(f_1(x_0, x_2), f_1(x_1, x_3))$
$\widetilde{x_2} = f_1(f_2(x_0, x_2), f_2(x_1, x_3))$
$\widetilde{x_3} = f_2(f_2(x_0, x_2), f_2(x_1, x_3))$

Auf Grund der Kommutativität von f_1, f_2 gilt:
$f_i(f_j(x_0, x_2), f_j(x_1, x_3))$

$= f_i(f_j(x_0, x_2), f_j(x_3, x_1))$
$= f_i(f_j(x_2, x_0), f_j(x_3, x_1))$
$= f_i(f_j(x_2, x_0), f_j(x_1, x_3))$
$= f_i(f_j(x_1, x_3), f_j(x_0, x_2))$
$= f_i(f_j(x_1, x_3), f_j(x_2, x_0))$
$= f_i(f_j(x_3, x_1), f_j(x_0, x_2))$
$= f_i(f_j(x_3, x_1), f_j(x_2, x_0))$

$i, j \in 1, 2$

Es lassen sich also die Positionen von (x_0, x_2) und (x_1, x_3) vertauschen sowie jeweils die Elemente innerhalb dieser Tupel. Auf den ursprünglichen Vektor übertragen, ergeben sich daraus folgende Vektoren, die in die gleiche Äquivalenzklasse abgebildet werden:

$(x_0, x_1, x_2, x_3) = \mathbf{x}_1$
$(x_0, x_3, x_2, x_1) = \mathbf{x}_3$
$(x_2, x_1, x_0, x_3) = \mathbf{x}_2$
$(x_2, x_3, x_0, x_1) = \mathbf{x}_4$
$(x_1, x_0, x_3, x_2) = \mathbf{x}_5$
$(x_1, x_2, x_3, x_0) = \mathbf{x}_7$
$(x_3, x_0, x_1, x_2) = \mathbf{x}_6$
$(x_3, x_2, x_1, x_0) = \mathbf{x}_8$

Diese sind genau die geforderten.

Siehe [1] (Kap.4.1., Seite 37) für eine Herleitung für $\mathbb{I}_{\mathbb{C}T}(\mathbf{x})$ wo $\mathbf{x}$ beliebig lang ist.

[1] Hans Burkhardt, *Transformationen zur lageinvarianten Merkmalgewinnung*, Habilitationsschrift, Universität Karlsruhe, 1979.

Aufgabe 3.2: R-Transformation

Die R-Transformation $T(\mathbf{x})$ eines Objekts $\mathbf{x} \in \mathbb{R}^{2^n}$ ist rekursiv gegeben durch

$$T(\mathbf{x}) := \begin{cases} \mathbf{x} & falls\ n = 0 \\ \begin{pmatrix} T(\mathbf{x}_{1|2} + \mathbf{x}_{2|2}) \\ T(\left|\mathbf{x}_{1|2} - \mathbf{x}_{2|2}\right|) \end{pmatrix} & sonst. \end{cases}$$

1. Wir zeigen mittels vollständiger Induktion, dass die R-Transformation für positive $k \in \mathbb{R}^+$ homogen ist, d.h. $T(k\mathbf{x}) = kT(\mathbf{x})$.

   Für den Basisfall n = 0 verwenden wir die Definition von $T(\mathbf{x})$, also $T(\mathbf{x}) = \mathbf{x}$ bzw $T(k\mathbf{x}) = k\mathbf{x}$.

   Basisfall: n=0

   $T(k\mathbf{x}) = k\mathbf{x} = kT(\mathbf{x})$

   Bei dem Induktionsschritt können wir zunächst k im Argument von T ausklammern und anschließend die Induktionsvorraussetzung $T(k\mathbf{x}) = kT(\mathbf{x})$anwenden.

   Ind. Schritt: n → n+1

   $$T(k\mathbf{x}) = \begin{pmatrix} T(k\mathbf{x}_{1|2} + k\mathbf{x}_{2|2}) \\ T(\left|k\mathbf{x}_{1|2} - k\mathbf{x}_{2|2}\right|) \end{pmatrix} = \begin{pmatrix} T(k(\mathbf{x}_{1|2} + \mathbf{x}_{2|2})) \\ T(k\left|\mathbf{x}_{1/2} - \mathbf{x}_{2|2}\right|) \end{pmatrix} \overset{nachI.V.}{=} kT(\mathbf{x})$$

2. Im zweiten Teil war mittels vollständiger Induktion zu zeigen, dass sich eine Änderung des Gleichanteils nur auf den nullten Koeffizienten der Transformierten auswirkt, d.h $T(\mathbf{x} + r(1, ..., 1)^T) = T(\mathbf{x}) + r2^n\mathbf{e}_0$.

   Im Basisfall nutzen wir die Defintion von T für $n = 0$ und die Tatsache, dass $2^0 = 1$ ist. Zudem ist anzumerken, dass für $n = 0$ $\mathbf{x} \in \mathbb{R}$ ist und somit $\mathbf{x}$ nur einen Komponente besitzt.

   Basisfall: n=0

   $T(\mathbf{x} + r) = \mathbf{x} + r = \mathbf{x} + r2^0$

   Im Induktionsschritt nutzen wir die Spaltung des Vektors $\mathbf{x}$ und somit auch des Vektors $(1, ..., 1)^T$ in zwei Vektoren der halben Länge aus. Da $\mathbf{x}_{1|2}$, $\mathbf{x}_{2|2}$ die Länge 2^n besitzen können wir die Induktionsvoraussetzung anwenden. Nach Induktionsvoraussetzung wirkt sich die Änderung nur auf den nullten Koeffizienten der oberen Hälfte des transformierten Vektors aus und da sich die untere Hälfte nicht verändert, insgesamt nur auf den nullten Koeffizienten der Tranformierten.

   Ind. Schritt: n → n+1

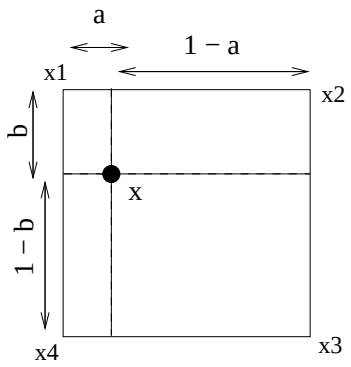
   $$T(\mathbf{x} + r\underbrace{(1, ..., 1)^T}_{2^{n+1}Stück}) = \begin{pmatrix} T(\mathbf{x}_{1|2} + r\underbrace{(1, ..., 1)^T}_{2^n} + \mathbf{x}_{2|2} + r(1, ..., 1)^T) \\ T(\left|\mathbf{x}_{1|2} + r(1, .., 1)^T - \mathbf{x}_{2|2} - r(1, ..., 1)^T\right|) \end{pmatrix}$$

$$= \begin{pmatrix} T(\mathbf{x}_{1|2} + \mathbf{x}_{2|2} + 2r(1,...,1)^T) \\ T(|\mathbf{x}_{1|2} - \mathbf{x}_{2|2}|) \end{pmatrix} \overset{nachI.V.}{=} \begin{pmatrix} T(\mathbf{x}_{1|2} + \mathbf{x}_{2|2}) + 2 * 2^n r e_0 \\ T(|\mathbf{x}_{1|2} - \mathbf{x}_{2|2}|) \end{pmatrix}$$

$$= \begin{pmatrix} T(\mathbf{x}_{1|2} + \mathbf{x}_{2|2}) \\ T(|\mathbf{x}_{1|2} - \mathbf{x}_{2|2}|) \end{pmatrix} + 2^{n+1} r \mathbf{e}_0$$

Aufgabe 3.3: Programmieraufgabe: Rotation mit Bilineare Interpolation

Wir ändern der Program von Aufgabe 2.3, und nutzen Bilineare Interpolation statt der 'Nächste-Nachbar' Regel.



Der Grauwert x lässt sich mit Hilfe von bilinearer Interpolation so berechnen:

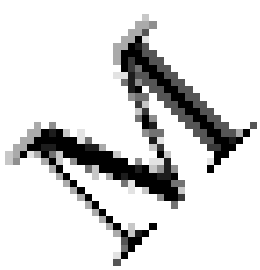
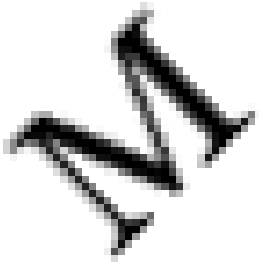
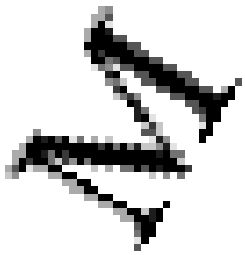
$x = (1-a)(1-b)x_1 + (a)(1-b)x_2 + (a)(b)x_3 + (1-a)(b)x_4$

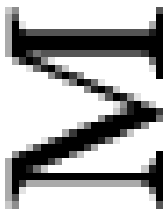
```
function B = rotImgBI(A, phi)
// function p = rotImgBI(A, phi)
//
// Rotation eines Bildes mit Bilinearer Interpolation
//
// input:    A            Bild beliebiger Größe
//           phi          Rotationswinkel
//
// output:   B            gedrehtes Bild (sqrt(2)* so groß wie Orginal)
//


// Konstruiere die Rotationsmatrix
R = [cos(phi) sin(phi); -sin(phi) cos(phi)];

// Hole Größen des Bildes
m = size(A,1);
n = size(A,2);
mhalbe = round(m/2);
nhalbe = round(n/2);

// das Ergebnisbild ist würzel 2 mal so groß
B = ones(floor(m*sqrt(2)),floor(n*sqrt(2)))*255;

// Erweitere das Orginal auf würzel 2 mal Ergebnisbildgröße
As = ones(m*2,n*2)*255;
As(mhalbe:(m+mhalbe-1),nhalbe:(n+nhalbe-1)) = A;

// das Rotationszentrum
```

```
center = [m;n];
center2= [round(m/sqrt(2)); round(n/sqrt(2))];

// Schleife über das Ergebnisbild
for x = 1:size(B,1),
   for y = 1:size(B,2),
      // rotiere Koordinate
      P = (R' * ([x ; y]-center2) + center); // Exact Location of the point
      p = P(1); q=P(2);

      // The factors a and b for bilinear interpolation
      a = p - floor(p);
      b = q - floor(q);

      // grayvalues at the four grid points surrounding the point (p,q)
      x1 = As(floor(p), floor(q));
      x2 = As(floor(p), ceil(q));
      x3 = As(ceil(p), ceil(q));
      x4 = As(ceil(p), floor(q));

      // Compute value by interpolation
      B(x, y) = (1-a)*(1-b)*x1 ...
                  + (1-a)*(b)*x2  ...
                  + (a)*(b)*x3   ...
                  + (a)*(1-b)*x4;

   end;
end;
```

Die Ergebnisse angewendet auf das gegebene Bild für Winkel von 45,60 und 90 Grad.

Nächste Nachbar **Bilineare Interpolation**




Offensichtlich sind die Ergebnisse schöner mit Bilineare Interpolation da die Grauwertübergänge nicht so stark sind.