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 (Mustererkennung)
WS 05/06

Musterlösung 6

Aufgabe 6.1: Affininvariante Fourierdeskriptoren von Polygonzügen

Berechnung des Flächenschwerpunktes:

$$\mathbf{x}_s = \frac{1}{3} \cdot \frac{\sum\limits_{i=0}^{3} |\mathbf{x}_i, \mathbf{x}_{i+1}| \cdot (\mathbf{x}_i + \mathbf{x}_{i+1})}{\sum\limits_{i=0}^{3} |\mathbf{x}_i, \mathbf{x}_{i+1}|} =$$

$$= \frac{1}{3} \cdot \frac{\begin{vmatrix} 1 & 4 \\ 2 & 2 \end{vmatrix} \cdot \begin{pmatrix} 5 \\ 4 \end{pmatrix} + \begin{vmatrix} 4 & 5 \\ 2 & 3 \end{vmatrix} \cdot \begin{pmatrix} 9 \\ 5 \end{pmatrix} + \begin{vmatrix} 5 & 2 \\ 3 & 3 \end{vmatrix} \cdot \begin{pmatrix} 7 \\ 6 \end{pmatrix} + \begin{vmatrix} 2 & 1 \\ 3 & 2 \end{vmatrix} \cdot \begin{pmatrix} 3 \\ 5 \end{pmatrix}}{-6+2+9+1} = \begin{pmatrix} 3 \\ 5/2 \end{pmatrix}$$

Berechnung der Parameter:

$$t_0 = 0$$
$$t_1 = \begin{vmatrix} 1-9 & 4-9 \\ 2-5/2 & 2-5/2 \end{vmatrix} /2 = 3/4$$
$$t_2 = 3/4 + \begin{vmatrix} 4-3 & 5-3 \\ 2-5/2 & 3-5/2 \end{vmatrix} /2 = 6/4$$
$$t_3 = 6/4 + \begin{vmatrix} 5-3 & 2-3 \\ 3-5/2 & 3-5/2 \end{vmatrix} /2 = 9/4$$

T kann als Summe der Flächen von zwei Dreiecken über die Determinante berechnet werden, etwa:

$$T = \begin{vmatrix} 4-1 & 5-1 \\ 2-2 & 3-2 \end{vmatrix} /2 + \begin{vmatrix} 5-1 & 2-1 \\ 3-2 & 3-2 \end{vmatrix} /2 = 3$$

Damit Berechnung der $e_{k,i}$: $e_{k,i} = e^{-j2\pi k t_i/3}$

$$e_{k,0} = 1$$
$$e_{k,1} = e^{-j\pi k/2} = (-j)^k$$
$$e_{k,2} = e^{-j\pi k} = (-1)^k$$
$$e_{k,3} = e^{-j3\pi k/2} = j^k$$

Fourierkoeffizienten:

Mit $t_{i+1} - t_i = 3/4 \quad \forall\, i$ folgt:

$$\mathbf{X}_k = \frac{3}{(2\pi k)^2} \cdot \frac{4}{3} \cdot \sum_{i=0}^{3} (\mathbf{x}_{i+1} - \mathbf{x}_i)(e_{k,i+1} - e_{k,i}) =$$
$$= \frac{1}{(\pi k)^2} [(2\mathbf{x}_0 - \mathbf{x}_1 - \mathbf{x}_3) e_{k,0} + (2\mathbf{x}_1 - \mathbf{x}_0 - \mathbf{x}_2) e_{k,1} + (2\mathbf{x}_2 - \mathbf{x}_1 - \mathbf{x}_3) e_{k,2} + (2\mathbf{x}_3 - \mathbf{x}_2 - \mathbf{x}_0) e_{k,3} =$$

$= \frac{1}{(\pi k)^2}\left[\begin{pmatrix}-4\\-1\end{pmatrix}\cdot 1+\begin{pmatrix}2\\-1\end{pmatrix}\cdot(-j)^k+\begin{pmatrix}4\\1\end{pmatrix}\cdot(-1)^k+\begin{pmatrix}-2\\1\end{pmatrix}\cdot j^k\right] =$

$= \frac{1}{(\pi k)^2}[1-(-1)^k]\cdot\left[-\begin{pmatrix}4\\1\end{pmatrix}+j^k\begin{pmatrix}-2\\1\end{pmatrix}\right]$

Somit verschwinden Fourierkoeffizienten mit geradem Index $\mathbf{X}_{2z} = \mathbf{0}, z \in Z$.

Sonst ergibt sich:

$X_{4z+1} = \frac{2}{\pi^2(4z+1)^2}\begin{pmatrix}-4-2j\\-1+j\end{pmatrix} = \begin{pmatrix}U_{4z+1}\\V_{4z+1}\end{pmatrix}$ und

$X_{4z-1} = \frac{2}{\pi^2(4z-1)^2}\begin{pmatrix}-4+2j\\-1-j\end{pmatrix} = \begin{pmatrix}U_{4z-1}\\V_{4z-1}\end{pmatrix}$

Insbesondere haben wir für $p=1$: $\mathbf{X}_1 = \frac{2}{\pi^2}\begin{pmatrix}-4-2j\\-1+j\end{pmatrix} = \begin{pmatrix}U_1\\V_1\end{pmatrix}$

Daraus ergibt sich für die Deskriptoren $Q_k = \frac{U_kV_1^*-V_kU_1^*}{U_1V_1^*-V_1U_1^*}$ mit $U_1V_1^*-V_1U_1^* = \frac{4}{\pi^4}\cdot 12j$:

$Q_{2z} = 0$

$Q_{4z+1} = \frac{\pi^4}{4\cdot 12j}\cdot\frac{4}{\pi^4(4z+1)^2}[(-4-2j)(-1-j)-(-1+j)(-4+2j)] = \frac{1}{(4z+1)^2}$

$Q_{4z-1} = \frac{\pi^4}{4\cdot 12j}\cdot\frac{4}{\pi^4(4z-1)^2}[(-4+2j)(-1-j)-(-1-j)(-4+2j)] = 0.$

Die Deskriptoren von $\mathbf{P}_2$ stimmen also mit denen von $\mathbf{P}_1$ überein, und deshalb sind die beiden Vierecke äquivalent unter affinen Abbildungen.

Aufgabe 6.2: Invarianten

z.z.:

$$\tau\nabla I(\mathbf{x}) = \nabla I(\tau\mathbf{x})$$

Wir können also zunächst den Gradientenvektor des Originalvektors bilden und diesen zykl. translieren oder den Originalvektor zuerst zykl. translieren und dann den Gradientenvektor bilden.

Beweis:
Wegen der Invarianz bzgl. den zyklischen Tranlationen τ von $I(\mathbf{x})$ gilt:

$$\nabla I(\mathbf{x}) = \nabla I(\tau\mathbf{x}) \tag{1}$$

Außerdem gilt:

$$\frac{d}{dx_i}I(\tau\mathbf{x}) = (d_{i+1}I)(\tau\mathbf{x}) \tag{2}$$

Einmal wird nach der Variablen x_i abgeleitet und das andere mal nach dem $i+1$ - Argument von I. Die Ableitung nach dem i-ten Argument im translierten Vektor entspricht also der Ableitung nach dem $i-1$-ten Eintrag im Originalvektor. Mit Hilfe von (1) und (2) kann man nun leicht ausrechnen:

$$\nabla I(\tau\mathbf{x}) = \tau^{-1}(\nabla I)(\tau\mathbf{x})$$

und somit

$$\tau\nabla I(\tau\mathbf{x}) = \tau\nabla I(\mathbf{x}) = \nabla I(\tau\mathbf{x})$$

q.e.d.

Aufgabe 6.3: Programmieraufgabe: Gruppenmittel und DFT

1.

$$\begin{aligned} I(0) &= |X(0)|^2 \\ &= (x_0 + x_1 + x_2)^2 \\ &= (x_0^2 + x_1^2 + x_2^2) + 2(x_0x_1 + x_1x_2 + x_2x_0) \\ &= M\{x_0^2 + 2x_0x_1\}, \end{aligned}$$

wobei mit M der Mittelungsoperator über die Gruppe der zyklischen Translationen bezeichnet wurde.

$$\begin{aligned} I(1) &= |X(1)|^2 = X(1)X(1)^* \\ &= X(1)X(2) = (x_0 + x_1\omega + x_2\omega^2)(x_0 + x_1\omega^2 + x_2\omega) \\ &= x_0^2 + x_1^2 + x_2^2 + (x_0x_2 + x_1x_2 + x_2x_0)(\omega + \omega^2) \\ &= (x_0^2 + x_1^2 + x_2^2) - (x_0x_1 + x_1x_2 + x_2x_0) \\ &= M\{x_0^2 - x_0x_1\} \end{aligned}$$

und $I(2) = I(1)$, da $X(2) = X(1)^*$.

Die $I(k)$ sind also gleichwertig zu den zwei Invarianten $M\{x_0^2\} =: J_0$ und $M\{x_0x_1\} =: J_1$, da man diese durch Linearkombination der $I(k)$ erhält.

2. Eine Musterimplementation ist folgende:

```
function res =  monom_feature(v,m)
//-------------------------------------------------------------------------
// function res =  monom_feature(v,m)
//
// Function that computes invariant features of a vector v with
// the monomial kernel specified in m
//
// input:    v       vector, for which the translation invariant
//                   feature should be computed
//
//           m       integer vector of length n = lenght(v) representing
//                   the monomial kernel v(1)^d1*v(2)^d2*...*v(n)^dn .
//                   The number in m(i) represents the exponent, to wich
//                   v(i) should be raised.
//
// output:   res     the invariant feature
//
//-------------------------------------------------------------------------

//make colunm vectors
v=v(:);
m=m(:);
```

```
//ensure length is the same
if length(v) ~= length(m)
   error("Dimensions of m and v must match!");
end;

// create matrices of right size and order

onevec=ones(m);
m = m*onevec';
v = v*onevec';
v = w_DI(v')';

// raise
 res = v.^m;

//make product and sum
tmp = prod(res,1);
res=sum(prod(res,1));

endfunction;


function res = w_DI(x)
//----------------------------------------------------------------------
// function res = w_DI(x)
//
// perform w_DI-transformation
//
//          x           matrix to be transformed
//
// output:  res         transformation of x.
//                      result is cyclic translation of the rows of x
//                      such that res has x's diagonal elements in
//                      the first column.
//----------------------------------------------------------------------

res = x;

for i = 2:size(x,1)
  res(i,:) = [res(i,i:$) res(i,1:(i-1))];
end;

endfunction;
```

3. Testen der Funktion:

```
//Aufrufskript
v = [5 2 1]';

// Invariante über das Betragsquadrat der DFT
```

```
inv_1 = (abs(dft(v,-1)))^2;

// Invariante über Gruppenmittel (zykl. Verschiebung)
inv_2(1) = monom_feature(v,[2 0 0]') + 2*monom_feature(v,[1 1 0]');
inv_2(2) = monom_feature(v,[2 0 0]') - monom_feature(v,[1 1 0]');
inv_2(3) = monom_feature(v,[2 0 0]') - monom_feature(v,[1 1 0]');

//print
disp(inv_1);
disp(inv_2);
```

liefert das Ergebnis:

```
-->disp(inv_1);

!   64. !
!   13. !
!   13. !

-->disp(inv_2);

!   64. !
!   13. !
!   13. !
```