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 4

Aufgabe 4.1: Schnelle Transformation $\mathbb{C}T$

a) Wir wenden die MT Transformation auf den Vektor $\mathbf{x} = (1, 2, 0, 1)^T$ an:
$\widetilde{\mathbf{x}} = MT((1, 2, 0, 1)^T) = (\widetilde{x_0}, \widetilde{x_1}, \widetilde{x_2}, \widetilde{x_3})^T$
$\widetilde{x_0} = \max(\max(1, 0), \max(2, 1)) = 2$
$\widetilde{x_1} = \min(\max(1, 0), \max(2, 1)) = 1$
$\widetilde{x_2} = \max(\min(1, 0), \min(2, 1)) = 1$
$\widetilde{x_3} = \min(\min(1, 0), \min(2, 1)) = 0$
$\widetilde{\mathbf{x}} = (2, 1, 1, 0)^T$

b) $MT((1, 2, 0, 1)^T) = MT((1, 0, 1, 2)^T) = (2, 1, 1, 0)^T$
und $(1, 0, 1, 2)^T$ ist nicht durch zyklische Translation aus $(1, 0, 1, 2)^T$ entstanden.

c) Wegen der Kommutativität der Funktionen $f_{1,2}$ liegen in einer Äquivalenzklasse von $\mathbb{C}T$ nicht nur die mittels zyklischer Translation aus dem Repräsentanten gewonnenen Vektoren sondern auch ihre gespiegelten (Beweis bei Übungsblatt 3, Aufgabe 1).

Daher ist vollständigkeit für vierdimensionale ternäre Vektoren im Allgemeinen nicht gegeben.

Bei vierdimensionalen binären Vektoren lässt sich jede Spiegelung eines Vektors auch durch eine zyklische Translation ausdrücken weshalb das Vorhandensein der gespiegelten Vektoren in der Äquivalenzklasse die Vollständigkeit nicht mehr beeinträchtigt. Daher ist $\mathbb{C}T$ auf vierdimensionalen binären Vektoren vollständig.

(Die Tatsache $\forall x \exists i : \psi(x) = \tau_i(x)$ lässt sich einsehen, wenn man zwischen der Anzahl der Einsen in einem vierdimensionalen binären Vektor unterscheidet: Für 0, 1 , 3 und 4 Einsen ist die Tatsache trivial einzusehen. Bei 2 Einsen erhält man: Für $(1, 0, 0, 1)^T$ und $(0, 1, 1, 0)^T$ gilt: $\psi(x) = x = \tau_0(x)$. Für $(1, 1, 0, 0)^T$ und $(0, 0, 1, 1)^T$ gilt: $\psi(x) = \tau_2(x)$. Und für $(1, 0, 1, 0)^T$ und $(0, 1, 0, 1)^T$ gilt $\psi(x) = \tau_1(x)$.)

Aufgabe 4.2: Fourierreihe

1. per Definition invariant unter Aufpunktverschiebung

2. invariant unter Translation da unabhänig von c_0 und Aufpunktverschiebung:
   $\frac{c_6 e^{i6t_0\omega}}{(c_3 e^{i3t_0\omega})^2} = \frac{c_6}{c_3^2}$,$\omega := \frac{2\pi}{T}$

3. invariant unter Translation und Aufpunktverschiebung:
   $c_1^2 e^{i2\varphi} c_{-2} e^{-i2\varphi} = c_1^2 c_{-2}$

4. invariant unter Translation, Aufpunktverschiebung und Rotation:
   $c_2 e^{i\varphi} e^{ikt_0} c_2 e^{i\bar{\varphi}} e^{ikt_0} = c_2 \bar{c_2}$,$k = \frac{2\pi}{T} \cdot 2$

5. invariant unter Translation:
   c_1 und c_2 sind invariant unter Translation.

6. invariant unter Aufpunktverschiebung und Translation:

   $c_6 e^{i6\cdot\frac{2\pi}{T}t_0} e^{-i(\phi_4+\varphi_0\cdot 4-\phi_1-\varphi_0)\cdot 2} = c_6 e^{i6\varphi_0} e^{-i(\phi_4-\phi_1)-i6\varphi_0} = c_6 e^{-i(\phi_4-\phi_1)}$,$\frac{2\pi}{T}t_0 = \varphi_0$

7. invariant unter Aufpunktverschiebung:
   c_0 und $\frac{c_6}{c_3^2}$ sind invariant gegen Aufpunktverschiebung.

8. Fallunterscheidung:
   $n = 0$, wie in 1.
   $n = 1$, Translation, Aufpunktverschiebung und Rotation Invariant.
   $c_{-n} e^{-in(\phi_{-n}+\varphi)} e^{i\varphi} = c_{-n} e^{-i(n\phi_{-n}+(n-1)\varphi)}$
   $n \notin \{0, 1\}$, Nur Translation Invariant.
   $c_{-n} e^{-in(\phi_{-n}-nt_0\omega)} e^{int_0\omega} = c_{-n} e^{-in(\phi_{-n}-nt_0\omega-t_0\omega)}$,$\omega := \frac{2\pi}{T}$

Aufgabe 4.3: Programmieraufgabe: Transformation $\mathbb{C}T$

1. Es sollten die kommutativen Funktionen für die MT und RT implementiert werden. Eine Lösung könnte folgendermaßen aussehen:

```
function res = pw_add(x,y)
// function res = pw_add(x,y)
//
// elementwise addition of the input arguments
//
// input:    x           vector/ matrix
//
// output:   res         vector/ matrix with sum of elements
//
  res = x+y;

endfunction;

function res = pw_absdiff(x,y)
// function res = pw_absdiff(x,y)
//
// elementwise absolute value of the difference of the
// input arguments
//
// input:    x           vector/ matrix
//
// output:   res         vector/ matrix with absolute
//                       value of the difference
//
  res = abs(x-y);

endfunction;

function res = pw_min(x,y)
// function res = pw_min(x,y)
//
// performs an element wise min op on matrices/ vectors
//
// input:     x,y   vectors/matrices (same size)
//
// output:    res   vector/matrix with values computed
//                  through an element wise min
//
  res = min(x,y);

endfunction;
```

```
function res = pw_max(x,y)
// function res = pw_max(x,y)
//
// performs an element wise max op on matrices/ vectors
//
// input:     x,y   vectors/matrices (same size)
//
// output:    res   vector/matrix with values computed
//                  through an element wise max
//
  res = max(x,y);

endfunction;
```

2. Die implementierten Funktionen dienen als Parameter für die schnelle eindimensionale $\mathbb{C}T$, die so aussehen kann:

```
function res = CT(f1, f2, x)
// function res = CT(f1, f2, x)
//
// input: f1, f2   commutative functions with calling syntax
//                 r = f1(a,b) where a,b,r are equally sized
//                 matrices and r contains the results of
//                 elementwise dyadic operation.
//                 x vector/ matrix: each column will be processed
//                 by the CT.
//
// output: res     vector/ matrix of the columnwise results of
//                 the transformation of x.
  n = size(x,1);
  if n>1,
    i1 = 1: floor(n/2)
    i2 = (floor(n/2)+1):n
    res = [CT(f1,f2,f1(x(i1,:),x(i2,:))) ; CT(f1,f2,f2(x(i1,:),x(i2,:)))]
  else
    res = x
  end;

endfunction;

function res = MT(x)
// function res = MT(x)
//
// perform M-Transformation
//
// input:    x           vector/ matrix to be transformed: each
//                       column will be processed individually.
//
// output:   res         vector/ matrix with the columnwise results of
//                       the transformation of x.
    res = CT(pw_max,pw_min,x);

endfunction;
```

```
// function res = RT(x)
//
// perform R-Transformation
//
// input:    x           vector/ matrix to be transformed: each
//                       column will be processed individually.
//
// output:   res         vector/ matrix with the columnwise results of
//                       the transformation of x.
  res = CT(pw_add,pw_absdiff,x);

endfunction;
```

3. Transformationsergebnisse für die angegebenen Vektoren:

   (a)
   ```
   -->MT( [4, 5, 6, 7, 1, 1, 2, 3]')
    ans  =

     !   7.  !
     !   6.  !
     !   5.  !
     !   4.  !
     !   3.  !
     !   2.  !
     !   1.  !
     !   1.  !
   ```

   (b)
   ```
   -->MT([6, 7, 1, 1, 3, 3, 4, 5]' )
    ans  =

     !   7.  !
     !   6.  !
     !   5.  !
     !   4.  !
     !   3.  !
     !   2.  !
     !   1.  !
     !   1.  !
   ```

   (c)
   ```
   -->RT([2, 3, 4, 5, 6, 7, 8, 9]' )
    ans  =

     !   44. !
     !   4.  !
     !   8.  !
     !   0.  !
     !   16. !
     !   0.  !
     !   0.  !
     !   0.  !
   ```

(d)
```
-->RT([8, 9, 10, 11, 12, 5, 6, 7]' )
 ans  =

  !   68. !
  !   4.  !
  !   8.  !
  !   0.  !
  !   16. !
  !   0.  !
  !   0.  !
  !   0.  !
```

Deutung der Ergebnisse:
Die M-transformierten Vektoren a) und b) stimmen im Ergebnis überein und zeigen die Invarianz der MT gegenüber zyklischer Translation.
Der Vektor d) ist der gleiche Vektor wie c), beaufschlagt mit einer systematischen Störung (die z.B. als Helligkeitsveränderung aufgefasst werden kann) sowie verändert durch eine zyklische Translation. Es wird nur der erste R-Transformationskoeffizient beeinflusst, die anderen bleiben unverändert (vgl. Blatt 3).