Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik
FH Zentralschweiz

IPCV: Kompression I + II - Übung

Prof. Dr. Josef F. Bürgler

I.BA_IPCV

Die Aufgaben sind zusammen mit dem Lösungweg in möglichst einfacher Form darzustellen. Numerische Resultate sind mit einer Genauigkeit von 4 Stellen anzugeben. Skizzen müssen qualitativ und quantitativ richtig sein.

Sie sollten im Durschnitt 75% der Aufgaben bearbeiten. Abgabetermin ihrer Übungsaufgaben ist die letzte Vorlesungsstunde in der Woche nachdem das Thema im Unterricht besprochen wurde.

1: Von Hand zu lösen

Skizzieren Sie die die Funktionen $\cos(0\cdot x)$, $\cos(1\cdot x)$, $\cos(2\cdot x)$ und $\cos(3\cdot x)$ im Intervall $[0,\pi]$. Bestimmen Sie die *Abtastwerte* für alle drei Funktionen an den Stellen $x_k = k\frac{\pi}{4}$ für $k = 0,1,2,3$ erzeugen Sie damit die Matrix mit den Elementen

$$A_{n,j} = \cos(nx_k), \text{ wobei } 0 \leq n,j < 4.$$

Vervollständigen Sie dazu die folgende Matrix:

$$\mathbf{A} = \begin{array}{c} \\ n{=}0 \\ n{=}1 \\ n{=}2 \\ n{=}3 \end{array} \begin{array}{c} \begin{array}{cccc} j{=}0 & j{=}1 & & \end{array} \\ \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ \cos\left(\frac{\pi}{8}\right) & & & \cos\left(\frac{7\pi}{8}\right) \\ & \cos\left(\frac{3\pi}{4}\right) & \cos\left(\frac{5\pi}{4}\right) & \cos\left(\frac{7\pi}{4}\right) \\ & & & \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \end{array}$$

Erkennen Sie diese Matrix? Wenn nicht, rechnen Sie einige dieser Matrixelement mit dem Taschenrechner aus!

Lösung: Man erhält folgende Matrix:

$$\begin{array}{c} \\ n{=}0 \\ n{=}1 \\ n{=}2 \\ n{=}3 \end{array} \begin{array}{c} \begin{array}{cccc} j{=}0 & j{=}1 & j{=}2 & j{=}3 \end{array} \\ \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ \cos\left(\frac{\pi}{8}\right) & \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{5\pi}{8}\right) & \cos\left(\frac{7\pi}{8}\right) \\ \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{4}\right) & \cos\left(\frac{5\pi}{4}\right) & \cos\left(\frac{7\pi}{4}\right) \\ \cos\left(\frac{3\pi}{8}\right) & \cos\left(\frac{9\pi}{8}\right) & \cos\left(\frac{15\pi}{8}\right) & \cos\left(\frac{21\pi}{8}\right) \end{bmatrix} \end{array}$$

Dies ist bis auf c_n die Transformationsmatrix.

2: Mit python (numpy, scipy) zu lösen

Mit Hilfe der DCT (Diskrete Cosinus Transformation) kann ein Vektor $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^T$ [1] wie folgt auf den Vektor $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$ abgebildet werden:

$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

wobei die Matrix $\mathbf{C}$ definiert ist durch

$$C_{n,j} = c_n \cos\left(\frac{\pi n(2j+1)}{2N}\right) \quad 0 \leq n, j < N$$

mit $c_0 = \frac{1}{\sqrt{N}}$, $c_i = \sqrt{\frac{2}{N}}$ für $i = 1, 2, \ldots, N-1$.

Definieren Sie für beliebiges $N \in \mathbb{N}$ die Matrix $\mathbf{C}$ und zeigen Sie, dass die Inverse von $\mathbf{C}$ gleich der Transponierten ist, d.h.

$$\mathbf{C}^{-1} = \mathbf{C}^T$$

Dazu müssen Sie zeigen, dass gilt:

$$\mathbf{C}\mathbf{C}^T = \mathbf{C}^T\mathbf{C} = \mathbf{1}$$

Kontrollieren Sie für $N = 2, 4, 8$ und $N = 16$!

Lösung: Natürlich könnte man dies analytisch und somit für alle $N \in \{2, 3, 4, \ldots\}$ beweisen. Hier geht es aber darum, dass Sie die Matrix $\mathbf{C}$ für ein beliebiges N spezifizieren und dann zeigen, dass gilt:

$$\mathbf{C}\mathbf{C}^T = \mathbf{C}^T\mathbf{C} = \mathbf{1}$$

Also beispielsweise für $N = 1$:

In [1]:
```
import numpy as np
from numpy import linalg as LA
```

In [2]:
```
N = 2
C = np.zeros(shape=(N,N))
# C
```

In [3]:
```
for i in range(N):
    for j in range(N):
        if i == 0:
            c_n = np.sqrt(1/N)
        else:
            c_n = np.sqrt(2/N)
        C[i][j] = c_n * np.cos((2*j+1)*i*np.pi/(2*N))

print(C)
print(LA.inv(C))
```

```
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
```

In [4]:
```
np.max(abs(LA.inv(C)-C.T))
```

Out[4]:
```
2.220446049250313e-16
```

[1] Beachte: wir verwenden *bold-face* um Vektoren zu bezeichnen und nicht, wie auch möglich, das Vektorzeichen über dem x.

3: Mit python (numpy, scipy) zu lösen

Wenn Sie die Aufgabe 1 verstanden haben, können Sie sofort sagen, wie man den Vektor $\mathbf{x}$ aus dem Vektor $\mathbf{y}$ berechnet! Wie sieht die entsprechende Gleichung in möglichst einfacher Form aus (Matrixform!).

Lösung: Man hat nacheinander

$$\mathbf{x} = \mathbf{C}^{-1}\mathbf{y} = \mathbf{C}^{T}\mathbf{y}.$$

4: Mit python (numpy, scipy) zu lösen

(1) Erzeugen Sie ein Graustufenbild mit 8×8 Pixel, in welches Sie einen Kreis mit einem Radius R von 2 Pixel zeichnen. Dabei soll nur die Randlinie des Kreises schwarz sein, der Rest des Bildes weiss. Erzeugen Sie ein Bild der Matrix (mit `matrixplot(..)`).

(2) Führen Sie die Kosinus Transformation durch und komprimieren Sie das Bild, indem Sie nur 90, 70, 50 oder 30 Prozent aller Koeffizienten berücksichtigen.

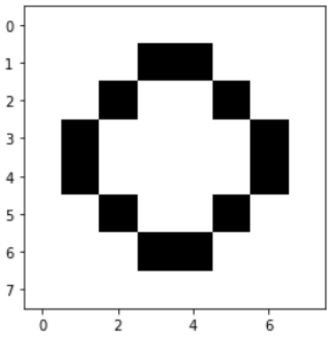
(3) Führen Sie die Rücktransformation durch. Machen Sie die entsprechenden Matrizen mit `matrixplot(..)` sichtbar.

Lösung: (1) Zuerst die entsprechenden Librariers laden:

```
In [1]: import numpy as np
        import cv2
        from matplotlib import pyplot as plt
```

Graustufenbild von erzeugen und anzeigen:

```
In [2]: image = np.matrix([[255, 255, 255, 255, 255, 255, 255, 255],
                           [255, 255, 255,   0,   0, 255, 255, 255],
                           [255, 255,   0, 255, 255,   0, 255, 255],
                           [255,   0, 255, 255, 255, 255,   0, 255],
                           [255,   0, 255, 255, 255, 255,   0, 255],
                           [255, 255,   0, 255, 255,   0, 255, 255],
                           [255, 255, 255,   0,   0, 255, 255, 255],
                           [255, 255, 255, 255, 255, 255, 255, 255]])

        plt.imshow(image, cmap="gray")
Out[2]: <matplotlib.image.AxesImage at 0x12585e048>
```

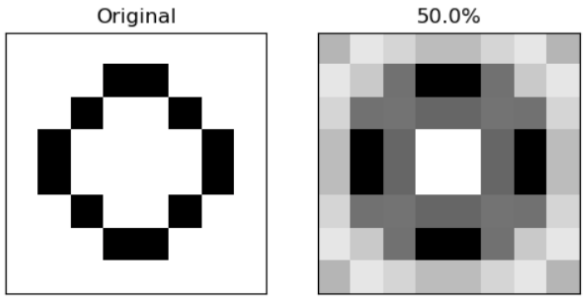


(2) Kosinus Transformation definieren und durchführen. Dann hohe Frequenzen Null setzen!

In [3]:
```
# Generiere zuerst DCT-Matrix
N = 8
C = np.zeros(shape = (N, N))

for i in range(N):
    for j in range(N):
        C[i][j] = np.cos((2 * j + 1) * i * np.pi / (2 * N))

C[0,:] = C[0,:] * np.sqrt(1 / N)
C[1:,:] = C[1:,:] * np.sqrt(2 / N)

# Generiere DCT-Image (Aus den Slides)
dct_image = np.dot(np.dot(C, image), C.T)

# Komprimiere
ratio = 0.5
size = N * N

compressed_dct_image = dct_image.copy()

# Starte mit 0 setzen rechts unten. So werden zuerst die hohen Frequer
for x in range(N):
    for y in range(N):
        if (size * ratio) < ((x + 1) * (y + 1)):
            compressed_dct_image[x, y] = 0

print(compressed_dct_image)
```

Rücktransformation und Anzeige des Originals und der komprimierten Versiion:

In [4]:
```
# Rekonstruiere Image aus obiger erhaltener komprimierter DCT-Matrix
decoded_image = np.dot(np.dot(C.T, compressed_dct_image), C)

plt.rcParams['figure.dpi']= 100
plt.subplot(1, 2, 1), plt.imshow(image), plt.set_cmap('gray'), plt.tit
plt.subplot(1, 2, 2), plt.imshow(decoded_image), plt.set_cmap('gray'),
plt.show()
```



5: Mit python (numpy, scipy) zu lösen

(1) Erzeugen Sie ein frei gewähltes Bild als Graustufen-Matrix der Grösse 16×16.

(2) Führen Sie die Kosinus Transformation durch, und unterdrücken Sie einige der hohen Frequenzen.

(3) Führen Sie die Rücktransformation durch. Zeichnen Sie die so entstehenden Bilder mit `matrixplot(..)`

Hinweis zu (3): Um einen guten Eindruck zu gewinnen, muss man die Bilder möglicherweise "*aufblasen*", d.h. aus einem Pixel ein 8×8 Quadrat machen. In MATLAB gibt es dazu die unten gezeigte Funktion `pixeldup`.

```
function B = pixeldup(A, m, n)
%PIXELDUP duplicates pixels of an image in both directions.
% B = PIXELDUP(A, M, N) duplicates each pixel of A M times in the
% vertical direction and N times in the horizontal direction.
% Parameters M and N must be integers. If N is not included, it
% defaults to M.

% Check inputs.
```

```
if nargin > 2
  error('At␣least␣two␣inputs␣are␣required.');
end
if nargin == 2
  n = m;
end
% Generate a vector with elements 1:size(A, 1).
u = 1:size(A, 1);

% Duplicate each element of the vector m times.
m = round(m); % Protect against nonintegers.
u = u(ones(1, m), :);
u = u(:);

% Now repeat for the other direction.
v = 1:size(A, 2);
n = round(n);
v = v(ones(1, n), :);
v = v(:);
B = A(u, v);
```

Schreiben Sie diesen Code um in eine **python**-Funktion.

6: Lauflängenkodierung

Kodieren und dekodieren sie die Zeichenkette *00111111100011111110001*. Was ist der Komprimierungsfaktor im schlechtesten Fall?

Lösung: Wenn man abmacht, dass man mit der Anzahl Nullen beginnt, dann ist das Resultat `2,7,3,7,3,1`.

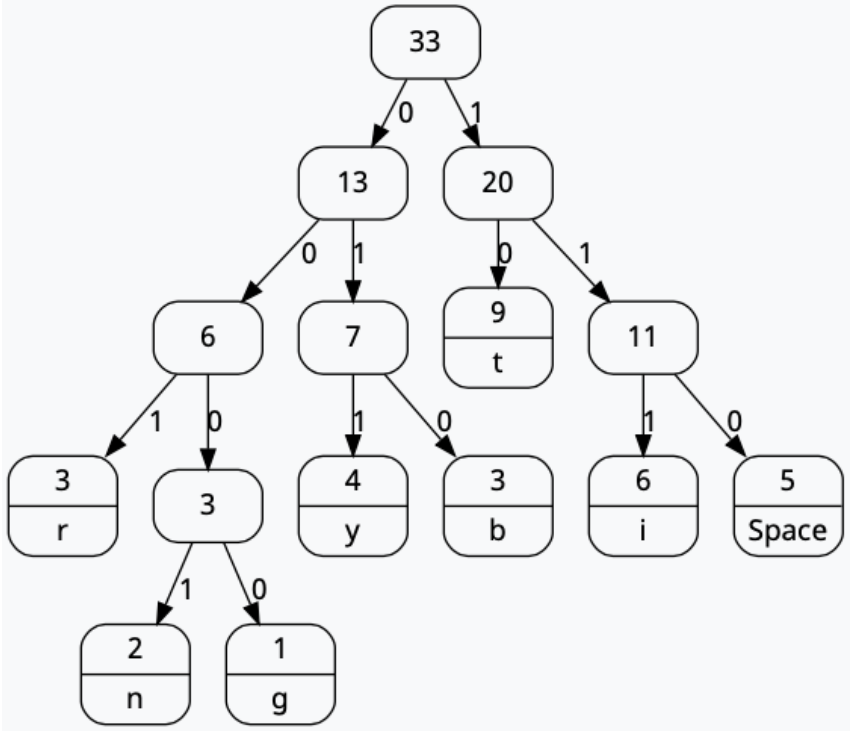
7: Huffmankodierung

Konstruieren Sie zwei verschiedene Huffmann-Kodierungen für folgende Symbole (und zugehörige Häufigkeiten): t (0.2), u (0.3), v (0.2) und w (0.3). Wie viele verschiedene Huffmankodierungen gibt es? Wie lautet eine zweite Möglichkeit?

Bestimmen Sie die Information, die in der Zeichenkette *itty bitty nitty grrritty bit bin* steckt.

Lösung: Zum ersten Teil: hier fasst man als erstes `t` und `v` zusammen zum Superzeichen `tv` mit der Wahrscheinlichkeit 0.4. Da die Reihenfolge nicht wesentlich ist, hat man hier zwei Möglichkeiten. Anschliessend fasst man `u` und `w` zum Superzeichen `uw` mit der Wahrscheinlichkeit 0.6 zusammen. Hier ist die Reihenfolge ebenfalls nicht wesentlich, was zu 2 Möglichkeiten führt. Zusammen hat man bis jetzt 4 Möglichkeiten! Schliesslich fasst man die beiden Superzeichen `tv` und `uw` zusammen zum Superzeichen `tvuw`. Damit erhält man folgenden Code: `t`:00, `v`:01, `u`:10, `w`:11.

Zum zweiten Teil zeigt die folgende Abbildung eine mögliche Huffman-Codierung (Quelle: Huffman Tree Generator)



Man hat folgende Währscheinlichkeiten: `g`:1/33,`n`:2/33,`r`:3/33,`b`:3/33, `y`:4/33,`space`:5/33,`i`:6/33,`t`:9/33.

Die Information ergibt sich somit durch

$$
\begin{aligned}
I &= -\sum_i p_i \log_2(p_i) \\
&= -\frac{1}{33}\log_2\left(\frac{1}{33}\right) - \frac{2}{33}\log_2\left(\frac{2}{33}\right) - 2\cdot\frac{3}{33}\log_2\left(\frac{3}{33}\right) - \frac{4}{33}\log_2\left(\frac{4}{33}\right) \\
&\quad - \frac{5}{33}\log_2\left(\frac{5}{33}\right) - \frac{6}{33}\log_2\left(\frac{6}{33}\right) - \frac{9}{33}\log_2\left(\frac{9}{33}\right) \\
&= 2.767\text{Bit/Char}
\end{aligned}
$$

8: Lempel-Ziv Kompression

1977/78 haben Abraham Lempel und Jakob Ziv die verlustfreie LZ77-Komprimierung erfunden, welcher der Deflate-Algorithmus zu Grunde liegt. Eingesetzt wird er beispielsweise in PKZIP und im PNG-Bildformat. Die LZ78-Komprimierung wird vor allem für Bilder benutzt: er ist auch Grundlage des LZW-Algorithmus (studieren Sie das Verfahren anhand geeigneter Videos auf Youtube).

LZ77 erzeugt Pointer zurück auf wiederholte Daten. LZ78 erzeugt ein Verzeichnis von wiederholten Phrasen (Zeichenketten) mit Pointern zu diesen Phrasen.

Komprimieren und dekomprimieren sie die Zeichenkette *itty bitty bit bin* mittels LZ77 (mit einer Fensterlänge von 10 und einer Pufferlänge von 5) und LZ78 (referenzieren Sie die besten von Ihnen gefundenen Videos).

Schreiben sie ein möglichst einfaches Programm, welches genau diese Kompressionen und Dekompressionen durchführt (hardcodierter String!).

Lösung: Verwenden Sie das Applet der ETH Zürich!

9: LZW Kompression

1984 entstand der Lempel-Ziv-Welch Komprimierungsalgorithmus von Terry Welch durch Modifikation der LZ78-Komprimierung. Er eignet sich für allem für Bilder (angewendet in TIFF und GIF).

Komprimieren und dekomprimieren sie die beiden Zeichenketten *itty bitty bit bin* und *itty bitty nitty grrritty bit bin* mittels LZW.

Schreiben sie ein möglichst einfaches Programm, welches genau diese Kompression und Dekompression durchführt (hardcodierter String!).

Lösung: Verwenden Sie das Applet der ETH Zürich!

10: GIF (Zweierarbeit)

Studieren sie das GIF-Format. Beschreiben sie auf ca. 10 Zeilen stichwortartig die Funktion des GIF-Formats. Relevante Kodefragmente sollten (nicht zu sehr ins Detail) erklärt werden. Das Papier soll den anderen Studierenden im Modul IPCV eine Einführung in GIF geben. Sie sollten daraus die Funktionsweise von GIF erfahren können.

11: PNG (Zweierarbeit)

Studieren sie das PNG-Format. Beschreiben sie auf ca. 10 Zeilen stichwortartig die Funktion des PNG-Formats. Relevante Kodefragmente sollten (nicht zu sehr ins Detail) erklärt werden. Das Papier soll den anderen Studierenden im Modul IPCV eine Einführung in PNG geben. Sie sollten daraus die Funktionsweise von PNG erfahren können.

12: JPEG2000 (Zweierarbeit)

Studieren sie das JPEG2000-Format. Beschreiben sie auf ca. 10 Zeilen stichwortartig die Funktion des JPEG2000-Formats. Relevante Kodefragmente sollten (nicht zu sehr ins Detail) erklärt werden. Das Papier soll den anderen Studierenden im Modul IPCV eine Einführung in JPEG2000 geben. Sie sollten daraus die Funktionsweise von JPEG2000 erfahren können.

13: TIFF (Zweierarbeit)

Studieren sie das TIFF-Format. Beschreiben sie auf ca. 10 Zeilen stichwortartig die Funktion des TIFF-Formats. Relevante Kodefragmente sollten (nicht zu sehr ins Detail) erklärt werden. Das Papier soll den anderen Studierenden im Modul IPCV eine Einführung in TIFF geben. Sie sollten daraus die Funktionsweise von TIFF erfahren können.

Ich wünsche Ihnen viel Spass!