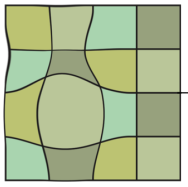

Image Processing & Understanding

Grundlagen der Bildverarbeitung

Praktische Aufgaben - Aufgabe 4

Wintersemester 16/ 17
Bildverarbeitung und Bildverstehen
Prof. Klaus Tönnies,
Tim König, Johannes Steffen

Abgabe: Die Abgabe erfolgt spätestens am 13.12.2016 um 08:00 Uhr. Jedes Team muss insgesamt 5 Dateien einreichen: *p04_execute.m*, *p04_compress.m*, *p04_decompress.m*, *p04_rle.m*, *p04_rld.m*. Allgemeine Details zur Einreichung finden Sie auf der Webseite!

In dieser Aufgabe sollen Sie ein Grauwertbild komprimieren. Dafür sollen Sie eine abgewandelte Form der JPEG-Kompression implementieren, welche im Allgemeinen durch folgende Schritte erfolgt:

- Einteilung des Bildes in $m \times m$ Blöcke ($m = 8$ bzw. 16). Dann für jeden Block:
  - Frequenzraumüberführung mittels Diskreter Kosinustransformation (DCT)
  - Quantisierung der Koeffizienten mit der entsprechenden $m \times m$ Quantisierungsmatrix
  - Zig-Zag-Umsortierung der quantisierten Koeffizienten entlang der Diagonalen (Abbruch, wenn nur noch 0-Elemente folgen)
  - Run-Length-Encoding (RLE) des entstandenen quantisierten Koeffizientenvektors (Achtung: Eigentlich wird bei JPEG die Huffman-Kodierung benutzt!)
- Rückgabe aller RLE-Vektoren der Blöcke des Bildes

Für die Umsetzung dieser Schritte sollen Sie folgende Teilaufgaben erledigen:

1. Schreiben Sie zunächst eine Funktion *vec_enc = p04_rle(vec)*, welche einen Zeilenvektor Run-Length enkodiert. Der resultierende Vektor *vec_enc* soll an den ungeraden Indizes die *Values* und an den geraden Indizes die *Run-Lengths* abbilden.

Beispiel:

$vec = \begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1 & 2 & 3 & 3 & 3\end{pmatrix}$, *p04_rle(vec)* $= \begin{pmatrix}1 & 6 & 2 & 1 & 3 & 3\end{pmatrix}$

2. Schreiben Sie (passend zu Ihrer Kodierungsfunktion) eine Dekodierungsfunktion *vec = p04_rld(vec_enc)*, welche in der Lage ist, aus Ihrem enkodierten Vektor den dekodierten, also den originalen Vektor, wiederherzustellen.

3. Schreiben Sie eine Funktion *img_compressed = p04_compress(image, block_size, quant)*, welche folgende Teilschritte umsetzt:

   a) Zerlegung des Bildes in *block_size* $\times$ *block_size* große Blöcke (also 8×8 bzw. 16×16)

   b) Diskrete Kosinustransformation jedes Blockes mittels der Matlab/Octave-Funktion *dct2*

   c) Quantisierung der DCT-Blöcke mit der passenden *block_size* $\times$ *block_size* Quantisierungsmatrix *quant* (siehe bereitgestellte *p04_quantm.mat*)

   d) Umsortierung der quantisierten Koeffizienten entlang der (Anti-)Diagonalen jedes Blockes mit der bereitgestellten Funktion *p04_zigzag*

   e) Kodierung des zuvor sortierten, quantisierten Koeffizientenvektors mittels Ihrer Funktion *p04_rle* aus Aufgabe 1

   f) Rückgabe der kodierten Vektoren in einem Cell-Array (*img_compressed*) der Größe *Vertikale Blöcke* $\times$ *Horizontale Blöcke*

4. Schreiben Sie (passend zu Ihrer Funktion *p04_compress*) eine Funktion *image = p04_decompress(img_compressed, quant)*, welche Ihre komprimierte Bildspeicherung dekomprimiert und Ihnen als Grauwertbild zurückgibt. Um die Komprimierungsschritte rückgängig zu machen, können Sie folgende Funktionen verwenden: *p04_rld* (Ihre Funktion aus Aufgabe 2), *p04_zagzig* (bereitgestellt), *idct2* (aus Matlab/Octave)

   Hinweis: Achten Sie auf die korrekte Reihenfolge bei der Ausführung der Dekomprimierungsschritte und auf eine Dequantisierung der einzelnen Blöcke.

5. Testen Sie Ihre Funktionen *p04_compress* und *p04_decompress* innerhalb eines Skripts *p04_execute* mit dem bereitgestellten Beispielbild *p04_Bild1.png* für die *block_size* $= 8$ und *block_size* $= 16$ und geben Sie die Kompressionsraten auf der Konsole aus. Geben Sie alle drei Bilder (Original, Block-8-komprimiert, Block-16-komprimiert) aus und vergleichen Sie deren Qualität in den Kommentaren.

Hinweise

- Hilfreiche Funktionen, die Sie verwenden *können* (aber nicht müssen!): *load, mat2cell, cellfun, cell2mat, blockproc, repmat, fprintf, dct2, idct2.*