

# JPEG-Kompressionsverfahren

## Praktikum

**Geben Sie alle Ihre Resultate im folgenden Moodle Test ein:**  
**→ Abgabe Praktikum 08 JPEG**

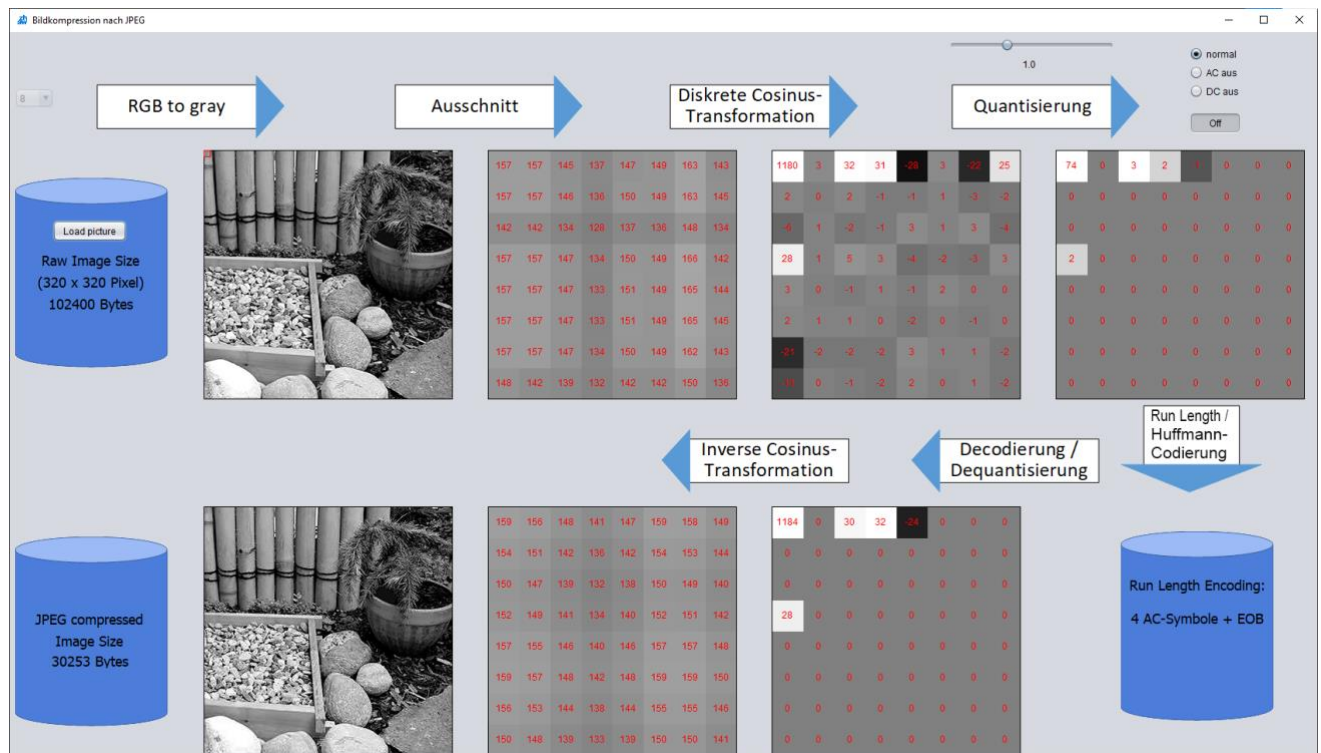
In diesem Praktikum geht es darum, die verlustbehaftete Kompression von JPEG genauer zu betrachten. Dazu muss das Wirkungsprinzip der diskreten Cosinus-Transformation (DCT) zur Irrelevanzreduktion verstanden werden. Ausserdem soll die inverse diskrete Cosinus-Transformation in Java programmiert und getestet werden.

### 1. Verlustbehaftete Kompression von JPEG

Mittels eines Demonstrators sollen verschiedene Fälle betrachtet werden und so ein Verständnis für das Wirkungsprinzip der diskreten Cosinus-Transformation und ihrer Eigenschaften, die eine Kompression erlauben, erlangt werden.

Laden Sie alle Dateien dieses Praktikums von Moodle in ein Verzeichnis:

- das Java-Programm JPEG.jar
- die Bilddateien Testbild1.png und Testbild2.png
- und die Programmvorlage DCT\_Aufgabe.java



Starten Sie das Java-Programm JPEG.jar. Klicken Sie auf «Load picture» und wählen Sie Testbild1.png aus. Im angezeigten Fenster wird oben der Prozess der JPEG-Komprimierung und unten derjenige der Dekomprimierung anhand eines 8x8 Blocks angezeigt.

Die aktuelle Position des Blocks wird im Originalbild (oben links) durch einen roten Rahmen angezeigt.

Den Block positionieren Sie durch Klicken in das Bild oder pixelweise mit Hilfe der Pfeil- Tasten.

- a) Wählen Sie im Bild Stellen von unterschiedlicher Helligkeit. Betrachten Sie vom ausgewählten Block die Koeffizienten im Orts- und Frequenz-Bereich.

Achtung: Beachten Sie im Frequenzbereich primär die Zahlenwerte. Die Helligkeit wird dynamisch angepasst und soll primär dazu dienen, die relevanten Koeffizienten schneller zu finden.

Welcher Koeffizient stellt die mittlere Helligkeit des Blocks dar?

- b) Wählen Sie im Bild Stellen von unterschiedlicher Struktur also glatte und strukturierte Flächen. Betrachten Sie wieder die Koeffizienten im Orts- und Frequenz-Bereich.

Wie verändern sich die Koeffizienten im Frequenz-Bereich zwischen einem Block mit glatter und einem mit strukturierter Fläche?

- c) Wählen Sie oben rechts «AC aus», betrachten Sie den Effekt auf das Endresultat (unten links) und erklären Sie den Effekt.

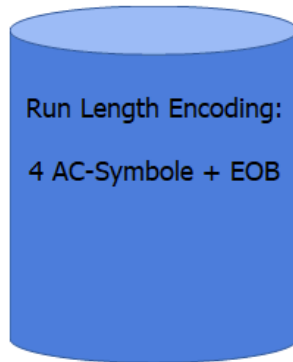
- d) Wählen Sie oben rechts «DC aus», betrachten Sie den Effekt auf das Endresultat und erklären Sie diesen.

- e) Laden Sie Testbild1.png neu, das sollte die Einstellungen zurücksetzen (z.B. den roten Rahmen auf ganz oben links im Bild). Betrachten Sie nun das Feld ganz rechts mit den quantisierten Koeffizienten.

Worin besteht der Unterschied zu den nichtquantisierten Werten?

- f) Im Speichersymbol unten rechts ist angegeben, wie viele AC-Symbole für die Codierung benötigt werden.

In diesem Beispiel:



Q	_0	_1	_2	_3	_4	_5	_6	_7
0_	74	0	3	2	-1	0	0	0
1_	0	0	0	0	0	0	0	0
2_	0	0	0	0	0	0	0	0
3_	2	0	0	0	0	0	0	0
4_	0	0	0	0	0	0	0	0
5_	0	0	0	0	0	0	0	0
6_	0	0	0	0	0	0	0	0
7_	0	0	0	0	0	0	0	0

Welche Null-Koeffizienten  $Q_{vu}$  werden zusammen mit einem der vier AC-Symbolen codiert, welche die Koeffizienten  $Q_{02}$  (4),  $Q_{03}$  (3),  $Q_{04}$  (-1) und  $Q_{30}$  (2) enthalten?

- g) Bewegen Sie den Schieber über dem Quantisierungspfeil. Dies bewirkt, dass die standardisierte Quantisierungsmatrix mit dem angezeigten Wert multipliziert wird. Betrachten Sie den Effekt auf das Endresultat und die Kompression.

Hinweise:

- Sie können den Schieber mit der Maus bewegen oder mit Hilfe der Pfeiltasten in Einzelschritten.
- Wenn Sie genau sehen wollen, was geschieht, können Sie unter Windows die Lupenfunktion einschalten: «Window»«+» (Ausschalten mit: «Window»«Esc»)
- Bei MacOSX kann der Bildschirm mit «Command»+«Option»+«0» vergrößert werden. Zurück mit «Command»+«Option»+«8»

Erklären Sie welche Artefakte (Bildfehler) bei welchen Quantisierungswerten sichtbar werden.

- h) Auf der linken Seite ist oben die Grösse des Bildes angegeben sowie unten die Anzahl der benötigten Bytes, nachdem dieses JPEG komprimiert wird. Berechnen Sie (für Testbild1.png) die Kompressionsrate, wenn Sie die Quantisierung auf 0 (keine), 1 (normale), 2 (starke) und 3 (sehr starke) stellen. Was sagt die Messreihe aus?

- i) Laden Sie nun die Datei Testbild2.png. Bei welchem Quantisierungswert wird ein Verlust sichtbar? Für welche Art resp. für welches der betrachteten Bilder ist JPEG geeigneter?
- j) Setzen Sie die Quantisierung auf 0. Überprüfen Sie sowohl anhand des Gesamtbilds als auch der Pixelwerte in verschiedenen Bildausschnitten, ob es einen Unterschied gibt zwischen dem komprimierten Bild und Original. Warum ist das so?
- k) Betrachten Sie die Kompressionsrate bei Quantisierung 0.0 ... 0.1. Was ziehen Sie für Schlüsse bezüglich Speicherung von Grafiken im JPEG-Format?
- l) Wählen Sie nacheinander den Bereich mit den feinsten horizontalen und den mit den vertikalen Linien (1 Pixel breit).  
Welche Frequenz-Koeffizienten stehen für die horizontalen und welche für die vertikalen Linien?
- m) Verschieben Sie den Ausschnitt um 1 Pixel (Pfeiltaste) quer zu den Linien. Wie verändern sich der Bildausschnitt und die Frequenz-Koeffizienten?
- n) Verschieben Sie den Ausschnitt über den Bereichen mit den horizontalen und vertikalen Linien unterschiedlicher Dicke und vergleichen Sie die Resultate mit den aus der Theorie bekannten Basisfunktionen.

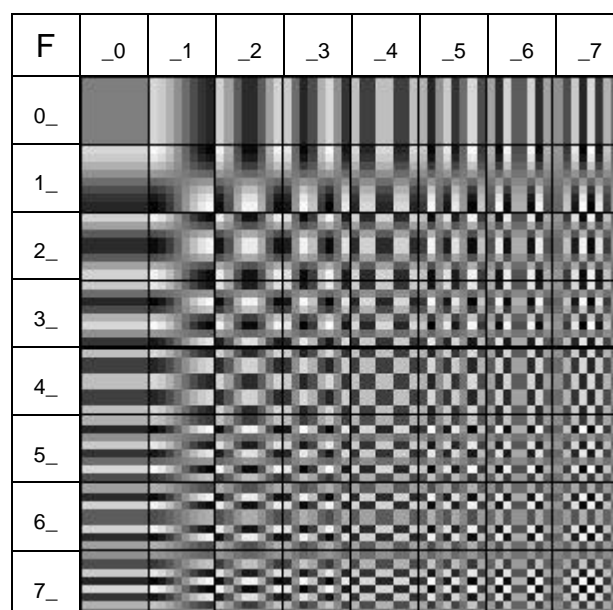
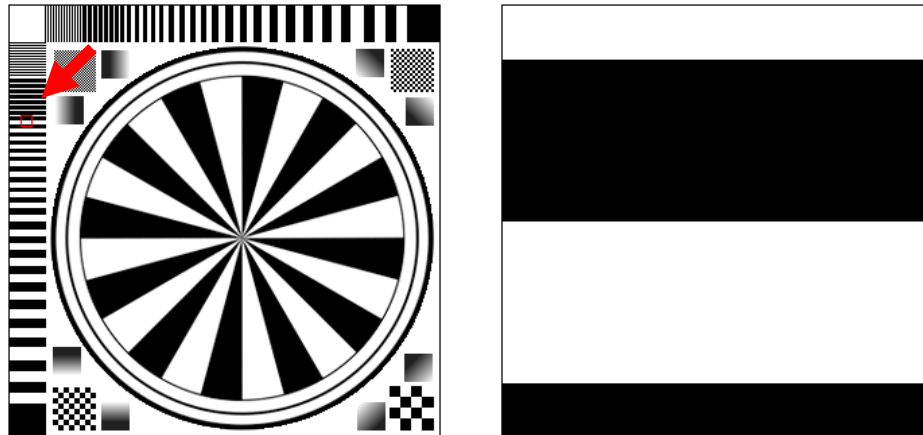



Abbildung 1: Basisfunktionen

- o) Wählen Sie im Programm den folgenden Bildausschnitt aus:



Betrachten Sie die AC-Koeffizienten. Welche Basisfunktionen  $F_{vu}$  hat den höchsten Anteil am obigen Bildausschnitt und welche kommen auch vor (absteigend sortiert nach Betrag)?

- p) Legen Sie nun den Block auf verschiedene Schachbrettmuster und vergleichen Sie die Koeffizienten ebenfalls mit den Basisfunktionen. Wo befinden sich die Basisfunktionen, welche hauptsächlich die Schachbrettmuster bilden?
- q) Finden Sie in einem Schachbrettmuster eine Position, für deren Darstellung nur ein AC-Koeffizient benötigt wird. Hinweis: Suchen Sie zuerst eine Basisfunktionen und dann eine Stelle im Bild, die dieser entspricht.
- r) Legen Sie nun den Block auf einen Gradienten; z.B.  und bewegen Sie den Block bitweise (z.B. horizontal von hell nach dunkel). Was ändert sich an den Koeffizienten?
- s) Suchen Sie eine interessante Position, die nur wenige AC-Koeffizienten benötigt und machen Sie einen Screenshot vom Bildausschnitt und den unquantisierten Koeffizienten. Sie benötigen diese zum Testen der folgenden Aufgabe.

## 2. Programmieren der inversen diskreten Cosinus-Transformation.

a) Öffnen Sie die Datei «DCT\_Aufgabe.java» und programmieren Sie die Funktion

```
public static void dct_inverse(int[][] Freq, int[][] Bild) {

    // Aufgabe: Inverse Cosinus Transformation
    // Input: Freq-Block mit 8x8 Koeffizienten, enthält
    //        den DC-Wert und die AC-Werte
    // Output: Bild mit 8x8 Pixeln

    /* Fügen Sie hier Ihren Code für die iDCT ein */
}
```

gemäss folgender Formel:

$$B_{yx} = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C_u C_v F_{vu} \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

Gehen Sie vor, wie in der Theorie besprochen:

- Wert der Ortsfunktion berechnen
- mit dem Koeffizienten  $F_{vu}$  sowie den Faktoren  $C_u$  und  $C_v$  gewichten
- über alle Ortsfrequenzen aufsummieren
- das obige für alle Pixel des Blocks ausführen

b) Testen Sie die Funktion, indem Sie das Programm übersetzen und mit den vorbereiteten Testwerten laufen lassen.

```
javac DCT_Aufgabe.java
java DCT_Aufgabe
```

Was steht auf dem angezeigten Bildausschnitt?

- c) Füllen Sie die Koeffizienten aus dem obigen Screenshot (1s) in die Frequenzmatrix ein und vergleichen Sie das Bild-Resultat.
- d) Angenommen das Bild besteht aus 320x320 Pixeln, wie viele Cosinus-Operationen werden bei einer inversen diskreten Cosinus-Transformation mit 8x8 Blöcken ausgeführt?
- e) Wie viele sind es mit 8x8 Blöcken bei einem Bild mit 3200x3200 Pixeln?
- f) Was sehen Sie für Optimierungsmöglichkeiten bezüglich Rechenaufwand für ihre Java-Implementation?