# 11B0033 AuV: Meilenstein 04
# Datenkokmpression

Julius Schöning
j.schoening@hs-osnabrueck.de

Abgabedatum: 03. Dezember 2020 23:59:59

In diesem Meilenstein werden Sie Bilddaten komprimieren. Dabei werden Sie Entropie- und Quellenkodierungsmethoden implementieren. Alle Aufgaben dieses Aufgabenblattes können Sie wahlweise mit Matlab oder der Scriptsprache Python bearbeiten.

Matlab steht Ihnen als Total Headcount Lizenz der Hochschule Osnabrück zur Verfügung. Installationsinformationen finden Sie unter `https://wiki.hs-osnabrueck.de/pages/viewpage.action?pageId=8749400`.

## Gruppen

Der Meilenstein *Datenkokmpression* wird in Gruppenarbeit mit den bereits bekannten Mitgliedern umgesetzt. Dieser Meilenstein ist so ausgelegt, dass Sie diesen an Ihrem eigenen Computer bearbeiten können.

Eine Veränderung der Gruppenzusammensetzung ist nicht mehr möglich.

## Abgabe

Die Ergebnisse aller Teilaufgaben sind schriftlich in einem strukturierten Praktikumsbericht im PDF-Format zusammenzufassen. Ergänzen Sie Ihre Ausführungen durch aussagekräftige Screenshots, Abbildungen, Plots und Tabellen. Mögliche Word und LaTeX Vorlagen für Praktikumsberichte finden Sie im Downloadbericht dieses Praktikum.

Den strukturierten Praktikumsbericht und alle Quellcodedateien, die Sie in diesem Meilenstein erzeugen oder auf die Sie sich in Ihrem Praktikumsbericht beziehen müssen in Ihrem Abgabearchiv vorhanden sein.

Für die Abgabe komprimieren Sie alle erstellen Matlab bzw. Python Quellcodedateien sowie Ihren strukturierten Praktikumsberichts als PDF in ein zip-Archive mit dem Namen *04_UserName1_UserName2_UserName3.zip*. Laden Sie das Zip-File bis spätestens zum 03. Dezember 2020 23:59:59 im Abgabebereich AuV Praktikum im OSCA als hoch.

## Testat

Bereiten Sie sich für ein Testat mit ggf. schriftlichen Kurztest von ca. 10 Minuten vor. Inhalt dieses Testats werden die Themen der Meilensteine 03 und 04 sein. Ihre Gruppe erhält einen persönlichen Termin für die KW50 zugeteilt.

# 1 Aufgabe: Huffman-Kodierung

Huffman-Kodierung erzeugt Codebäume, die bei fester Wahrscheinlichkeitsverteilung der Symbole optimale Präfix-Codes darstellen. Bei natürlich Bildern kann so eine Kompressionsrate von 1:3 erreicht werden.

| Symbol | Wahrscheinlichkeit |
|---|---|
| a | 35% |
| b | 25% |
| c | 20% |
| d | 10% |
| e | 5% |
| f | 3% |
| g | 2% |

a) Berechnen Sie für die in der Tabelle gegebenen Zeichen einen Präfix-Code im Huffman-Verfahren. Schreiben Sie entweder ein Script oder gehen Sie manuell vor.

b) Berechnen Sie die mittlere Codewortlänge.

c) Handelt es sich um Entropie-, Quellen-, oder Hybridkodierung? Begründen Sie!

# 2 Aufgabe: Lauflängenkodierung (RLE)

Bei der Lauflängenkodierung (RLE) werden Folgen von gleichen Werten zusammengefasst. Bei geeigneten Werten kann so eine verlustlose Komprimierung der Ausgangsdaten erreicht werden.

Komprimieren Sie die Bilder *moon.bmp* und *space.bmp* mit einer Lauflängenkodierung und beschreiben Sie Ihre Ergebnisse.

moon.bmp

(Gregory H. Revera, Creative Commons Attribution-Share Alike 3.0 Unported Lizenz)

space.bmp

(Chuck Ayoub, Creative Commons Attribution-Share Alike 4.0 International Lizenz)

a) Programmieren Sie ihre eigene Implementierung einer Lauflängenkodierung, die die obenstehenden Bilder im unkomprimierten BMP-Format einliest und komprimiert. Verwenden Sie kei-

ne vorgefertigten Funktionen. Die komprimierten Daten sollen als *moon_compressed.txt* bzw. *space_compressed.txt* im folgendes Format gespeichert werden:

*<Breite>:<Höhe>*

*<Anzahl>:<Zeichen>;<Anzahl>:<Zeichen>;<Anzahl>:<Zeichen>;...*

Dabei sind <Breite> und <Höhe> die Breite und Höhe des Ausgangsbilds. <Anzahl> ist die Anzahl der Wiederholungen der Farbe eines Pixels. <Zeichen> ist die Farbe des Pixels.

Speichern Sie Ihr Script unter *auv_04_Aufgabe_2.m* wenn Sie Matlab oder unter *auv_04_Aufgabe_2.py* wenn Sie Python verwenden.

b) Handelt es sich um Entropie-, Quellen-, oder Hybridkodierung? Begründen Sie!

c) Vergleichen Sie die Dateigrößen der komprimierten Bilder. Warum ist die Komprimierung unterschiedlich effektiv?

## 3 Aufgabe: Quantisierung

Auch durch Quantisierung können Bilddaten komprimiert werden. Implementieren Sie eine Quantisierung und testen Sie ihr Programm mit dem Bild *malojapass.png*.

malojapass.png

(Friedrich Böhringer, Creative Commons Attribution-Share Alike 3.0 Austria Lizenz)

a) Programmieren Sie ein Skript zur Quantisierung des Bildes *malojapass.png*. Benutzen Sie keine vorgefertigten Funktionen. Benutzen Sie den Befehl *im2double(<Bildmatrix>)*, um die Farbwerte des Bildes in double Werte im Intervall $[0, 1]$ zu konvertieren.

Speichern Sie Ihr Script unter *auv_04_Aufgabe_3.m* wenn Sie Matlab oder unter *auv_04_Aufgabe_3.py* wenn Sie Python verwenden.

Hinweis: Es ist nicht nötig mit Schleifen über alle Pixel zu iterieren. Sie können alle Werte in der Matrix des Bildes gleichzeitig multiplizieren, runden und dividieren.

b) Quantisieren sie das Bild *malojapass.png* auf 8, 32 und 128 Farbstufen je Farbkanal und speichern sie die Ergebnisse als *malojapass_quant_8.png*, *malojapass_quant_32.png* und *malojapass_quant_128.png*.

c) Handelt es sich um Entropie-, Quellen-, oder Hybridkodierung? Begründen Sie!

d) Vergleichen Sie die Dateigrößen und Bildqualität der Bilder. Erläutern Sie die Vor- und Nachteile von Quantisierung.