Karlsruher Institut für Technologie

Aufgabenstellung

09. 11. 2022

Digitaltechnik - Challenge 1
Codename Huffman

Institut für Technik der Informationsverarbeitung
Prof. Dr.-Ing. Dr. h.c. Jürgen Becker
M.Sc. Julian Höfer

Wintersemester 22/23

Aufgabe 1

Das ITIV-Logo macht sich klein

In Mikrocontrollern und eingebetteten Systemen müssen Daten häufig auf sehr kleinen Speichern abgelegt oder zwischengespeichert werden. Klein müssen die Speicher deswegen sein, da sie für gewöhnlich teuer sind und auf Chips viel Platz benötigen. Im folgenden Beispiel soll das ITIV-Logo als Schwarz-Weiß Bild verlustfrei in einem eingebetteten Video-RAM (VRAM) einer kleinen Spielekonsole zwischengespeichert werden. Der Speicher ist insgesamt **8 kByte (kB)**, also 64 kBit groß und hat damit die gleich Speicherkapazität wie der VRAM des ersten Nintendo Gameboy aus dem Jahre 1989. Das Logo darf aber nicht mehr als 10% des Speicherplatzes, also **800 Bytes oder 6400 Bit** belegen. Jeder Pixel des Bildes wird üblicherweise mit einem Byte, also 8 Bit, dargestellt und kann damit Werte zwischen 0 und 255 annehmen. 0 bedeutet ein vollständig weißer Pixel, 255 ein schwarzer Pixel. Alles dazwischen sind Grautöne.
Die Aufgabe besteht nun darin, das Bild im Speicher unterzubringen. Dazu können die Pixel nun mit variabler Länge dargestellt werden um eine geringere mittlere Codewortlänge als 8 Bit zu erzielen. Als Hilfestellung sind die Anzahl der auftretenden Grautöne als Histogramm gegeben.
Die Ausgangsgröße des Bildes sind 50x50 Pixel.

Abbildung 1.1 Einfaches ITIV-Logo in Schwarz-Weiß mit einer Größe von 50x50 Pixeln

Um Daten effizient zu komprimieren, hilft ein Blick auf die Häufigkeit der Pixelwerte. In diesem Beispiel ist das sehr einfach. Es gibt genau zwei Werte: 0 für Weiß und 255 für Schwarz. Grautöne kommen noch nicht vor.

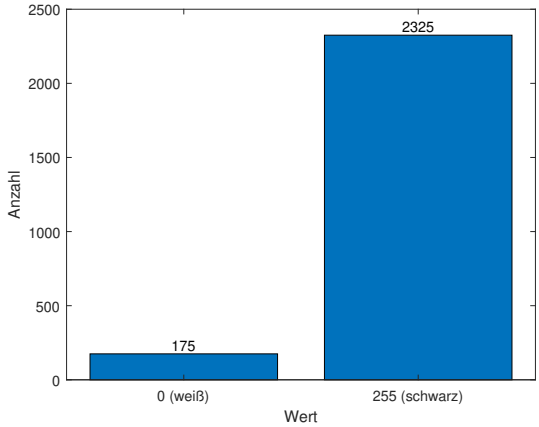


Abbildung 1.2 Auftrittshäufigkeit der Pixelwerte des einfachen ITIV-Logos

a) Wie viel Speicherkapazität nimmt das Bild im unkomprimierten Zustand ein?

b) Wie viele Bits zur einfachsten Codierung werden bei Abbildung 1.1 benötigt, wenn nur zwei verschiedene Grautöne verwendet werden?

c) Welche Speicherkapazität benötigt das codierte Bild dann insgesamt? Passt das Bild in den vorhandenen Speicher?

Mit einem Update soll das ITIV-Logo einen modernen Anstrich erhalten und einen Farbverlauf wie in Abbildung 1.3 aufweisen. Der verfügbare Speicher kann nicht vergrößert werden und das neue Logo darf auch weiterhin nur 10% des gesamten VRAMs einnehmen.

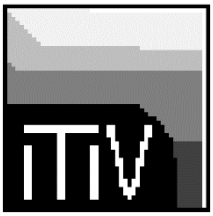


Abbildung 1.3 Neues ITIV-Logo in Grautönen mit einer Größe von 50x50 Pixeln

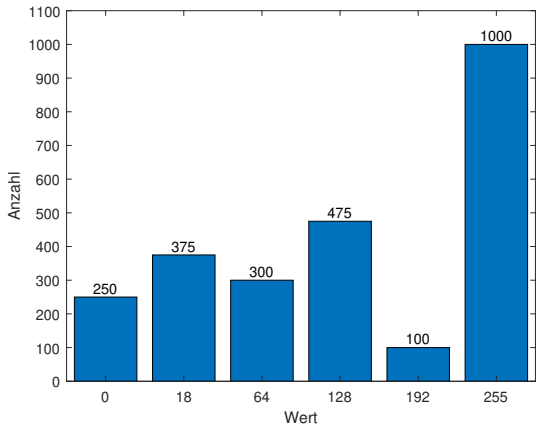


Abbildung 1.4 Auftrittshäufigkeit der Pixelwerte des neuen ITIV-Logos

d) Wie viele Bits werden bei Abbildung 1.3 benötigt, um alle Grautöne darzustellen? Wie lang ist die Codewortlänge in diesem Fall pro Pixel?

e) Welche benötigte Speicherkapazität resultiert daraus bei Abbildung 1.3

f) Wie kann die benötigte Speicherkapazität für Abbildung 1.3 weiter reduziert werden? Zwei in der Vorlesung vorgestellte Verfahren eignen sich hierzu und sollen verglichen werden. Ist es möglich, auch das neue Logo verlustfrei im Speicher abzulegen? Ist ein Verfahren hier besser geeignet als das andere?

g) Theoriefrage zum Abschluss: Ein Bild kann auch als Informationsquelle angesehen werden. Wie würde ein Bild aussehen, welches den maximalen Informationsgehalt (Entropie) enthält.

Aufgabe 2

Fehlerhafte Botschaft

Für die serielle Datenübertragung (seriell = Bits zeitlich nacheinander) wird bis heute in PCs und Mikrocontrollern (Arduino, Raspberry Pi, etc.) ein Universal Asynchronous Receiver Transmitter (kurz UART) eingesetzt. Hierbei werden üblicherweise 5-8 Datenbits pro Datenwort versendet und mit einem Paritätsbit gesichert. Die Asynchronität wird dadurch ermöglicht, dass ein Datenwort immer zwischen einem Startbit und einem Stoppbit eingebettet ist. Dadurch wird ein Taktsignal (Clock) nicht benötigt.

Startbit	5-8 Datenbits	Paritätsbit	Stoppbit

a) Es wird der folgende Datenstrom empfangen. Dieser ist mit gerader Parität gesichert, ohne Start- und Stoppbit und besteht aus ASCII-Zeichen. Dekodiert den Datenstrom. Was fällt euch bei der Dekodierung auf?

1010100 1 1101000 1 1101001 0 1110011 1 0100000 1 1101001 0 1110011 1 0100000 1 1100001 1

0100000 1 1110000 1 1110010 0 1100101 0 1110011 1 1110101 0 1101110 1 1110100 0 0100000 1

1100111 0 1110010 0 1101111 0 1101101 1 0100000 1 1100001 1 0100000 1 1110011 1 1101101 1

1100001 1 1101100 0 1101100 0 0100000 1 1100100 1 1101001 0 1110011 1 1110100 0 1100001 1

1101110 1 1110100 0 0100000 1 1110111 0 1001111 0 1110010 0 1101100 0 1100100 1 0101110 0

b) Mit einem Paritätsbit lassen sich Bitfehler wohl erkennen, nicht aber korrigieren. Wie könnte eine Korrektur beim oberen Beispiel durch das Senden zusätzlicher Codewörter realisiert werden?

Aufgabe 3

Huffman-Rätsel

Dieser Teil der Challenge spielt sich in einer Matlab-Umgebung ab, welche in Ilias im Challenge1-Ordner verlinkt ist. Hier kann das Huffman-Verfahren spielerisch geübt werden.

In dieser Matlab-Umgebung sind 50 RGB-Bilder hinterlegt. Diese sind alle 306x306 Pixel groß und wurden in Schwarz-Weiß-Bilder gewandelt. Um die Speichergröße weiter zu reduzieren wurde die Anzahl der verschiedenen Grautöne stark reduziert (ca. 10 verschiedene Grautöne, je nach Bild). Als letzter Schritt wurde das Huffman-Verfahren angewendet um die durchschnittliche Codewortlänge zu minimieren.

Nun soll das Huffman-Verfahren wieder rückgängig gemacht werden. Leider ist die Codetabelle, welche dem Codewort den Pixelwert zuordnet, verloren gegangen. Bekannt ist aber noch die Häufigkeit der Pixelwerte. Diese wird vom Matlab-Programm ausgegeben, sobald auf ”Run Script” gedrückt wird.

Führt mit der Information der absoluten Pixelhäufigkeiten das Huffman-Verfahren für Bilder eurer Wahl durch und vervollständigt die **Codetabelle *dict*** im Matlab-Code. Das Bild kann durch die Variable *ImageNumber* ausgewählt werden (1-50 möglich). Ist *dict* mit Pixelwerten und Codewörten befüllt, einfach nochmal auf ”Run Script” drücken.

Wenn alles richtig gemacht wurde, wird sowohl das dekomprimierte Schwarz-Weiß-Bild, als auch das originale RGB-Bild angezeigt.

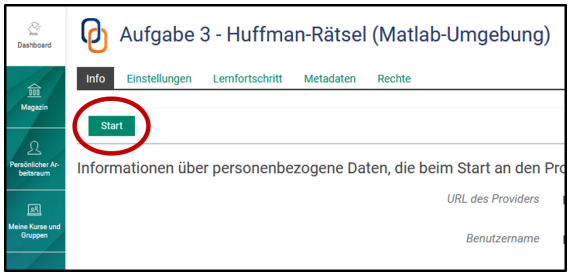
Damit alles funktioniert bitte beachten:

- Wenn zwei Gruppen verschmolzen werden, der Gruppe mit der niedrigeren Auftrittshäufigkeit den linken Ast zuweisen, der Gruppe mit der höheren den rechten Ast
- Der linke Ast erhält das Codierungszeichen 1, der rechte Ast das Codierungszeichen 0
- Die Einträge in der Variable *dict* aufsteigend nach Grauwert anordnen (siehe Beispiel)

Damit viel Erfolg und jetzt kann zur Matlab-Umgebung gewechselt werden!

So öffnet man die Matlab-Umgebung:

1. Klicken im Challenge 1 Ordner auf **Aufgabe 3 - Huffman-Rätsel (Matlab-Umgebung)**
2. Klicken auf **Start**



Abgabe an Tutoren

In Papierform:

- Aufgabe 1
- Aufgabe 2
- Aufgabe 3 - Codierungsbäume für 3 Bilder eurer Wahl