Berufsakademie Sachsen
Staatliche Studienakademie Leipzig

Hausarbeit Numerik
Studiengang Informatik
Studienrichtung Informatik

Anwendung des JPEG–Verfahrens

Eingereicht von: Leon Baumgarten
5CS22-1
Matrikelnummer: 5002213

23. Mai 2024

Inhaltsverzeichnis

1 Einleitung

Das JPEG-Verfahren (*Joint Photographic Experts Group*) ist eine weitverbreitete Methode zur Kompression digitaler Bilder, die eine signifikante Reduzierung der Dateigröße bei gleichzeitig geringem Verlust an optischer Qualität ermöglicht. Diese Hausarbeit demonstriert die Anwendung des JPEG-Algorithmus anhand eines gegebenen Graustufenbildes im Byte-Format mit einer Größe von 512×512 Pixeln.

Der Algorithmus beginnt mit der Aufteilung des Bildes in 8×8 Blöcke. Anschließend wird auf jeden Block die diskrete Kosinustransformation (DCT) angewendet. Die resultierenden Koeffizienten werden quantisiert, wobei drei verschiedene Quantisierungsmatrizen (Q20, Q50, Q90) verwendet werden. Nach der Quantisierung erfolgt die Rücktransformation durch Multiplikation mit der jeweiligen Quantisierungsmatrix und inverse DCT, gefolgt von der Rekonstruktion des Bildes.

In der vorliegenden wissenschaftlichen Hausarbeit werden die Schritte des JPEG-Verfahrens nachvollzogen, die Ergebnisse für verschiedene Quantisierungsmatrizen grafisch dargestellt und analysiert, sowie die Kompressionsleistung anhand des prozentualen Anteils der Nullen nach der Quantisierung bewertet.
– Die in der Hausarbeit beschriebene Implementierung zeigt somit die praktische Anwendung numerischer Methoden in der Bildverarbeitung. Durch die Zerlegung des Bildes in 8x8-Blöcke und die Umwandlung der Pixelwerte in das RGB-Format werden die Daten für den Einsatz des JPEG-Algorithmus vorbereitet, was einen wesentlichen Aspekt der modernen digitalen Bildkompression darstellt. Dieses Vorgehen veranschaulicht nicht nur die theoretischen Konzepte hinter der Bildkompression, sondern bietet auch Einblicke in die praktische Umsetzung numerischer Algorithmen in Software.

2 Aufgaben

1. Es ist ein Datensatz *picdat.dat* gegeben. Er liegt für ein Graustufenbild im Byte–Format vor, d.h. die Werte pro Pixel liegen im Bereich $0 \leq px \leq 255$.
Das Bild hat die Dimension 512×512. Lesen Sie den Datensatz ein und stellen Sie das Bild grafisch dar.

2. Der JPEG–Algorithmus wird auf 8×8 Matrizen angewendet. Zerlegen Sie also die 512×512 Datenmatrix in 64 8×8 Untermatrizen. .

3. Berechnen Sie die 8×8 orthogonale Transformationsmatrix der DCT

4.Auf jede Untermatrix wenden Sie den JPEG Algorithmus an:

- Elementweise Subtraktion von 128
- Anwendung der DCT
- Quantisierung mit der Matrix Q
- Rücktransformation (elementweise Multiplikation mit Q, inverse DCT, elementweise Addition von 128)

5. Bilden Sie wieder eine 512×512 Matrix

6. Es sind drei Q–Matrizen (8×8) gegeben: Q20 (Q20.dat), Q50 (Q50.dat) und Q90 (Q90.dat). Der JPEG–Algorithmus soll für alle drei Qi Matrizen ausgeführt werden. Für diese Matrizen geben Sie jeweils an:

- Wie hoch (prozentual) ist der Anteil der Nullen nach der Quantisierung
- Geben sie die resultierenden Bildern grafisch wieder und vergleichen Sie diese mit dem Originalbild

3 Lösungen

3.1 Einlesen & Ausgabe des Algorithmus

Zunächst wird in der main-Methode der Dateipfad festgelegt und die Bilddaten aus der Datei *picdat.dat* gelesen. Diese Daten werden zeilenweise verarbeitet, wobei jede Zeile einen Grauwert darstellt, der in ein Byte-Array umgewandelt wird. Anschließend wird aus diesem Array ein BufferedImage des Typs RGB erstellt, indem jeder Grauwert auf die Rot-, Grün- und Blau-Komponenten verteilt wird. (px, px, px)Schließlich wird das Bild in einem JFrame angezeigt, indem ein JPanel verwendet wird, das das Bild zeichnet.

Bei Fehlern während des Lesens der Datei oder der Bildgenerierung werden Fehlermeldungen in der Konsole ausgegeben. Der Code illustriert grundlegende Techniken der Datei- und Bildverarbeitung sowie der GUI-Anzeige in Java.

3.2 Aufteilen in 8x8 Matrizen

Das 512x512 Pixel große Bild wird in 8x8 Pixel große Blöcke unterteilt, ein Schritt, der grundlegend für den JPEG-Komprimierungsalgorithmus ist. Dabei wird zuerst die Blockgröße auf 8 Pixel festgelegt. Mit Hilfe einer Methode *splitImageIntoBlocks* wird das Bild in ein zweidimensionales Array von BufferedImage-Objekten zerlegt. Dies geschieht durch das Durchlaufen des Bildes mit zwei verschachtelten Schleifen, die jeweils die Startpunkte der 8x8 Blöcke bestimmen und diese mittels der *getSubimage*-Funktion extrahieren. Das Ergebnis ist ein Array, in dem jedes Element einem 8x8-Block des Originalbildes entspricht

3.3 Berechnung orthogonale Transformationsmatrix der DCT

Die diskrete Kosinustransformation (DCT) hilft dabei, das Bildsignal in Frequenzkomponenten zu zerlegen, wobei niedrige Frequenzen (Bereiche langsamer Änderung im Bild) von hohen Frequenzen (Bereiche schneller Änderung im Bild) getrennt werden. Dies ermöglicht eine effizientere Speicherung, da die visuell weniger wichtigen hohen Frequenzen stärker komprimiert werden können.[5]

Die diskrete Kosinustransformation (DCT) für eine $n \times n$-Matrix wird durch die folgende Transformationsmatrix T repräsentiert:

$$T_{ij}(n) = \begin{cases} \frac{1}{\sqrt{n}} & \text{für } i = 0, \\ \sqrt{\frac{2}{n}} \cos\left(\frac{\pi i(2j+1)}{2n}\right) & \text{sonst.} \end{cases}$$

Der Skalierungsfaktor α wird definiert als:

$$\alpha = \begin{cases} \frac{1}{\sqrt{n}} & \text{wenn } u = 0 \\ \sqrt{\frac{2}{n}} & \text{sonst} \end{cases}$$

Die Matrix T für $n = 8$ kann dann folgendermaßen geschrieben werden:

$$T = \begin{pmatrix} \alpha(0) \cdot \cos\left(\frac{(2\cdot 0+1)0\pi}{16}\right) & \alpha(0) \cdot \cos\left(\frac{(2\cdot 1+1)0\pi}{16}\right) & \cdots & \alpha(0) \cdot \cos\left(\frac{(2\cdot 7+1)0\pi}{16}\right) \\ \alpha(1) \cdot \cos\left(\frac{(2\cdot 0+1)1\pi}{16}\right) & \alpha(1) \cdot \cos\left(\frac{(2\cdot 1+1)1\pi}{16}\right) & \cdots & \alpha(1) \cdot \cos\left(\frac{(2\cdot 7+1)1\pi}{16}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha(7) \cdot \cos\left(\frac{(2\cdot 0+1)7\pi}{16}\right) & \alpha(7) \cdot \cos\left(\frac{(2\cdot 1+1)7\pi}{16}\right) & \cdots & \alpha(7) \cdot \cos\left(\frac{(2\cdot 7+1)7\pi}{16}\right) \end{pmatrix}$$

Daraus ergibt sich folgende Transformationsmatrix:

$$T = \begin{pmatrix} 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 & 0.354 \\ 0.490 & 0.416 & 0.278 & 0.098 & -0.098 & -0.278 & -0.416 & -0.490 \\ 0.462 & 0.191 & -0.191 & -0.462 & -0.462 & -0.191 & 0.191 & 0.462 \\ 0.416 & -0.098 & -0.490 & -0.278 & 0.278 & 0.490 & 0.098 & -0.416 \\ 0.354 & -0.354 & -0.354 & 0.354 & 0.354 & -0.354 & -0.354 & 0.354 \\ 0.278 & -0.490 & 0.098 & 0.416 & -0.416 & -0.098 & 0.490 & -0.278 \\ 0.191 & -0.462 & 0.462 & -0.191 & -0.191 & 0.462 & -0.462 & 0.191 \\ 0.098 & -0.278 & 0.416 & -0.490 & 0.490 & -0.416 & 0.278 & -0.098 \end{pmatrix}$$

3.4 Anwendung JPEG Algorithmus

Elementweise Subtraktion mit 128:
Zu Beginn wird die Methode *subtract128* verwendet, um von jedem Pixel eines Graustufenbildes 128 zu subtrahieren. Diese Zentrierung der Daten um den Nullpunkt optimiert die Daten für die nachfolgende diskrete Kosinustransformation (DCT).

Anwendung der DCT:
Die Methode *applyDCT* führt die DCT auf die zuvor zentrierten Daten aus, die blockweise in 8x8 Segmenten verarbeitet werden. Die dafür benötigte DCT-Matrix wird durch die Methode *createDCTMatriX* generiert, welche die notwendigen Transformationen für die Umwandlung in den Frequenzraum berechnet.

Quantisierung:
Nach der DCT folgt die Quantisierung durch die Methode *quantizeBlock*, bei der die DCT-Koeffizienten durch spezifische Werte einer Quantisierungsmatrix Q geteilt und gerundet werden. Dieser Prozess reduziert weniger wahrnehmbare, hochfrequente Details und erhöht somit die Kompressionseffizienz.

Rücktransformation:
Der letzte Schritt der JPEG-Kompression besteht aus mehreren Teilschritten, die die ursprünglichen Bildinformationen rekonstruieren.

- Dequantisierung:
  Durch die Methode *dequantizeBlock* werden die quantisierten Werte mit den entsprechenden Werten der Quantisierungsmatrix Q multipliziert.
- Anwendung der inversen DCT:
  Die inverse DCT, durchgeführt von der Methode *applyIDCT*, transformiert die Daten zurück aus dem Frequenzraum in den örtlichen Raum.
- Elementweise Addition von 128:
  Mit *add128* wird zu jedem Wert des rekonstruierten Blocks 128 addiert.

3.5 512 x 512 Matrix bilden

Durch die Methode *reassembleImage* wird das finale 512x512 Pixel große Bild aus den 8x8-Blöcken rekonstruiert. Diese Methode ordnet die einzelnen Bildblöcke in ihrer ursprünglichen Reihenfolge in einem neuen BufferedImage an. Es wird die Position jedes Blocks basierend auf dessen Index und Blockgröße berechnet. Anschließend werden die Pixelwerte in das Gesamtbild übertragen und mittels der *displayImage* Methode dargestellt.

3.6 Vergleich Bild durch Q10,Q50 und Q90

Die Verwendung unterschiedlicher Quantisierungsmatrizen Q10, Q50 und Q90 im JPEG-Kompressionsprozess führen zu signifikanten Unterschieden in der Bildqualität und Dateigröße.
Bei der Verwendung von **Q10** erhält man einen prozentualen Anteil an Nullen von 96,11%. Bei der grafischen Wiedergabe des Bildes sieht man eine deutliche Komprimierung. Es ist deutlich unschärfer als das Original. Die Quantisierungsmatrix ermöglicht eine starke Komprimierung, was jedoch auch zur deutlichen Abnahme der Bildqualität beiträgt.
Bei dem Einsatz von Matrix **Q50** lässt sich erkennen, dass das Bild ausgeglichener und schärfer ist als bei Q10 und dem Original mehr ähnelt. Der Anteil an Nullen beträgt hierbei 86,98%.
Bei der Implementierung der Matrix **Q90** lässt sich erkennen, dass die Bildqualität nicht sichtbar schlechter geworden ist. Der Anteil an Nullen liegt bei 83,43%.

Diese unterschiedlichen Raten an Nullen verdeutlichen, wie die Wahl der Quantisierungsmatrix direkt die Balance zwischen Kompressionseffizienz und Bildqualität beeinflusst.

4 Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Hausarbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt.

Leipzig, 23. Mai 2024 _____

Leon Baumgarten

Literatur

[1] Plato, Robert
Numerische Mathematik kompakt. Grundlagenwissen für Studium und Praxis..
Berlin, Germany: Springer Spektrum (Lehrbuch), 2021

[2] Schwarz, Hans Rudolf; Köckler, Norbert
Numerische Mathematik. [mit Online-Service.
akualisierte Aufl. Wiesbaden: Vieweg + Teubner, (2011)

[3] Prof. Dr. Holger Perlt
Skript Vorlesung Numerik.

[4] A.M.Raid, W.M.Khedr, M. A. El-dosuky, Wesam Ahmed *Skript Mansoura University.*
https://www.airccse.org/journal/ijcses/papers/5214ijcses04.pdf (2014)

[5] Saupe, D., Hamzaoui, R. *Bild- und Videokompression.*
https://www.degruyter.com/document/doi/10.1524/itit.45.5.245.22713/html
(2009)