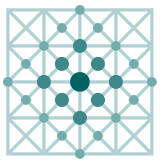
Dr. Andrés Rueda-Ramírez
Prof. Dr. Manuel Torrilhon
Tamme Claus
Dr. Satyvir Singh



Applied and Computational Mathematics

RWTH AACHEN UNIVERSITY

Mathematische Grundlagen IV (CES) | SS 2024
Programmierübung 1 | 01.05.2022
Deadline Anmeldung Testat: 13.05.2022
Testate: 14.05.2023 von 8:00 Uhr — 12:00 Uhr

Hinweise zur Abgabe einer Programmieraufgabe (bitte sorgfältig lesen!):

- Die Programmieraufgaben sind in **Dreier-** oder **Vierergruppen** zu bearbeiten.
- Tragen Sie sich bitte bis zum **13.05.2022** für einen Testattermin in dem entsprechenden Etherpad des Moodle Lernraums ein.
- Stellen Sie sicher, dass alle Programmverifikationen vor Abgabe der Aufgabe erfolgreich sind.
- Die Testate finden am **14.05.2023 von 8:00 Uhr — 12:00 Uhr** in verschiedenen Räumen des Rogowski statt (siehe Moodle). Bringen Sie zu dem Testat bitte einen Laptop und Ihr lauffähiges Programm mit.

Diskrete Fouriertransformation am Beispiel von JPEG

Wir betrachten Graustufen-Bilder, deren Speicherung in einer großen Anzahl von Bildpunkten, den *Pixeln* erfolgt. Jedes Pixel trägt die Information für den Helligkeitswert, auf den wir uns hier beschränken wollen. Die Herausforderung besteht nun darin, den Speicherbedarf möglichst gering und die Quälität möglichst hoch zu halten. Dafür wurden verlustbehaftete Komprimierungsverfahren entwickelt. Ein Beispiel ist das JPEG-Format.

Gegeben sei eine Graustufenmatrix der Dimension $\mathbb{R}^{N\times M}$. Bei der JPEG Kompression bedient man sich der zweidimensionalen *diskreten Kosinustransformation (DCT)*, welche eng verwandt ist mit der Fourier Transformation. Die Implementation der DCT über die hier angegebene Formel hat einen Aufwand von $O(N^2)$. Wie die *diskrete Fouriertransformation*, kann auch die DCT schnell, d.h. mit einem Aufwand von $O(N\log N)$ berechnet werden (schnelle Fourier-Transformation).

Wählt man für $j = 0,\dots,N-1$ und $k = 0,\dots,M-1$ die Stützstellen

$$x_j = \frac{2j+1}{2N}\pi \qquad\qquad y_k = \frac{2k+1}{2M}\pi,$$

so lautet das zugehörige *trigonometrische Polynom* zu einer Funktion $f \in C((0,\pi)^2)$:

$$T_{NM}(f;x,y) := \sum_{j=0}^{N-1}\sum_{k=0}^{M-1} d_{j,k}(f)\cdot c_j \cdot c_k \cdot \cos(jx)\cdot\cos(ky),$$

mit

$$c_j := \begin{cases} 1/2 & j=0 \\ 1 & \text{sonst} \end{cases} \qquad c_k := \begin{cases} 1/2 & k=0 \\ 1 & \text{sonst} \end{cases}$$

und den Koeffizienten

$$d_{j,k}(f) = \frac{4}{NM}\sum_{\tilde{j}=0}^{N-1}\sum_{\tilde{k}=0}^{M-1} f(x_{\tilde{j}}, y_{\tilde{k}})\cos(jx_{\tilde{j}})\cos(ky_{\tilde{k}})$$

wobei $f(x_{\tilde{j}}, y_{\tilde{k}})$ der Pixelwert an der Stelle $(x_{\tilde{j}}, y_{\tilde{k}})$ ist.

JPEG Komprimierung

Jedes zu komprimierende Bild wird zunächst in kleine Blöcke von 8×8 Pixeln, also eine Graustufenmatrix, die insgesamt 64 Einträge mit Werten im Bereich $[0, 256]$ hat, aufgeteilt.

Jeder Block wird nun einzeln bearbeitet.

Die 64 Helligkeitswerte eines Blocks fasst man nun als Werte einer Funktion f an den Stellen $x_{\tilde{j}}, y_{\tilde{k}}$ im Quadrat $(0,\pi)\times(0,\pi)$ auf. Zur Vorbereitung der Komprimierung werden in einem ersten Schritt die Koeffizienten der *diskreten Kosinustransformation* $d_{j,k}(f)$ berechnet. Die Koeffizienten sind die Amplituden zugehöriger Frequenzen: niedrige (i,j) Paare entprechen niedrigen Frequenzen während große (i,j) Paare hohen Frequenzen entsprechen.

Diese Darstellung alleine liefert noch keine Datenkomprimierung. Der entscheidende Schritt ist nun, 64 Schranken $\sigma_{j,k}$ einzuführen und die Koeffizienten/Amplituden durch $\tilde{d}_{j,k} := \text{round}(d_{j,k}/\sigma_{j,k})$ zur nächsten ganzen Zahl zu runden. σ ist die sog. Quantisierungsmatrix die so gewählt wird, dass gerade die Koeffizienten zu Null werden, die Frequenzen entsprechen, die kaum fürs menschliche Auge zu erkennen sind. Eine typische Wahl für σ ist:

$$\sigma := \begin{bmatrix} 10 & 15 & 25 & 37 & 51 & 66 & 82 & 100 \\ 15 & 19 & 28 & 39 & 52 & 67 & 83 & 101 \\ 25 & 28 & 35 & 45 & 58 & 72 & 88 & 105 \\ 37 & 39 & 45 & 54 & 66 & 79 & 94 & 111 \\ 51 & 52 & 58 & 66 & 76 & 89 & 103 & 119 \\ 66 & 67 & 72 & 79 & 89 & 101 & 114 & 130 \\ 82 & 83 & 88 & 94 & 103 & 114 & 127 & 142 \\ 100 & 101 & 105 & 111 & 119 & 130 & 142 & 156 \end{bmatrix}.$$

Die neue Matrix mit den Werten $\tilde{d}_{j,k}$ hat zumeist die Form einer links-oberen Dreiecksmatrix und kann daher komprimiert als Zickzack-Liste abgespeichert werden. Die Liste bricht dann ab, wenn die restlichen Werte Null sind, siehe `https://de.wikipedia.org/wiki/JPEG#Umsortierung_und_Differenzkodierung_des_Gleichanteils`. Die Datenkomprimierung entsteht dadurch, dass nur ein Teil der Liste abgespeichert wird.

Um ausgehend von $\tilde{d}_{j,k}$ wieder die 64 Helligkeitswerte eines Blocks zu berechnen, müssen beide Komprimierungsschritte umgekehrt werden. Zuerst werden die Koeffizienten der Kosinustransformation berechnet

$$d^*_{j,k} = \tilde{d}_{j,k} * \sigma_{j,k}$$

Dann folgt die Auswertung des trigonometischen Polynoms

$$A_{\tilde{j},\tilde{k}} = \texttt{TDCT}((d^*_{j,k}), x_{\tilde{j}}, y_{\tilde{k}}).$$

Die Blöcke A müssen dann noch zu einem Gesamtbild zusammengefügt werden.

Aufgaben

1. Implementieren Sie die *diskrete Kosinustransformation*. Schreiben Sie hierzu folgende Funktionen:

   $$\texttt{D} = \texttt{DCT(F)}$$
   $$\texttt{A} = \texttt{TDCT(D,x,y)}$$

   Der Input von `DCT(F)` ist die Matrix $(F_{\tilde{j}\tilde{k}}) = (f(x_{\tilde{j}}, y_{\tilde{k}})) \in \mathbb{R}^{N\times M}$ ausgewertet an den Stützstellen $x_{\tilde{j}}$ und $y_{\tilde{k}}$. `DCT(F)` berechnet dann aus F die Koeffizienten der Kosinustransformation und liefert als Output die Matrix $D = (d_{jk}) \in \mathbb{R}^{N\times M}$.

   `TDCT(D,x,y)` wertet das trigonometrische Polynom $T_{NM}(f; x, y)$ zu den Koeffizienten $D = (d_{jk}(f)) \in \mathbb{R}^{N\times M}$ an **beliebigen** Stellen (x_p, y_q) aus, wobei $x = (x_1, \ldots, x_P)$ und $y = (y_1, \ldots, y_Q)$ *Vektoren* beliebiger Größen sind. Die Matrix ist dann $(A_{pq}) = (T_{NM}(f; x_p, y_q)) \in \mathbb{R}^{P\times Q}$.

Sie können Ihr Programm testen, indem sie überprüfen, dass das trigonometrische Polynom eine beliebige Funktion an den Stützstellen der `DCT` exakt interpoliert.

$$(f(x_{\tilde{j}}, y_{\tilde{k}}))_{\tilde{j}=1,\dots N,\tilde{k}=1,\dots M} \approx \texttt{TDCT(DCT(F)}, \ (x_{\tilde{j}})_{\tilde{j}=1\dots N}, \ (y_{\tilde{k}})_{\tilde{k}=1\dots M})$$

Für beliebige Stellen (x, y) gilt diese Eigenschaft nicht.

Testbeispiele:

(a) Gegeben sei die Funktion

$$f : (0,\pi) \times (0,\pi) \to \mathbb{R}, \quad x \mapsto \cos(2x) + \cos(3y).$$

Erzeugen Sie für unterschiedliche $N \geq 3$ und $M \geq 4$ die Matrix F. Welches Ergebnis erwarten Sie für `D`? Vergleichen Sie das zu erwartende Ergebnis mit dem Output ihrer Funktion `DCT(F)`.

(b) Betrachten Sie nun die Funktion

$$f : (0,\pi) \times (0,\pi) \to \mathbb{R}, \quad x \mapsto \left(x - \frac{\pi}{2}\right)^2 + \left(y - \frac{\pi}{2}\right)^2.$$

Erzeugen Sie für unterschiedliche $M = N$ die Matrix $F = (f(x_{\tilde{j}}, y_{\tilde{k}})) \in \mathbb{R}^{N\times N}$ mit den Stützstellen der *DCT*. Zeichnen Sie nun den den größten absoluten Fehler in Abhängigkeit der Stützstellenazahl N

$$\max_{p,q=1,\dots,P-1} |f(x_p, y_q) - T_{NN}(f; x_p, y_q)|$$

wobei die Auswertungsstellen (x_p, y_p) auf einen weiteren *äquidistanten* Gitter

$$\Omega_h = \{(x_p, y_q) : x_p = ph, y_q = qh; h = \pi/P; p, q = 1, \dots, P-1\}$$

liegen. Wählen Sie etwa ($P = 100$, $N = 1:40$).

2. Jetzt möchten wir die JPEG-Komprimierung ganz konkret an einem Bild durchführen: Laden Sie dazu ein Bild und konvertieren dieses in Graustufen (Werte zwischen `0` und `256`!). In `julia` ist dies via

```
using Images, TestImages, ImageView
peppers_rgb = testimage("peppers")
peppers_matrix = round.(Float64.(Gray.(peppers_rgb)).*256)
```

möglich. Zur Visualisierung des Bilds in `julia` können sie `mosaicview(Gray.(peppers_matrix ./ 256))` (oder `imshow`) nutzen. Schreiben Sie eine Funktion

$$\texttt{A} = \texttt{JPEG(F)},$$

die für eine gegebene Matrix aus Helligkeitswerten $F \in \mathbb{R}^{8n\times 8m}$ die JPEG-Komprimierung durchführt und anschließend auch wieder umkehrt. Der Output ist dabei die neu entstandene Matrix $A \in \mathbb{R}^{8n\times 8m}$. Zeichnen Sie das ursprüngliche sowie das nach der JPEG-Kompression (und Umkehrung) neu entstandene Bild und vergleichen Sie beide. Berechnen Sie außerdem die Komprimierungsrate r

$$r = \frac{\text{Summe der } \tilde{d}_{j,k} \neq 0 \text{ Einträge aller Blöcke}}{8n * 8m}.$$

3. **Zusatzaufgabe:** Speichern Sie das Bild komprimiert als Zickzack Liste.

20 Points