Bildkompression mit Linearer Algebra

Bildkompression ist eine grundlegende Aufgabe in der digitalen Bildverarbeitung, die darauf abzielt, die Speichergröße von Bildern zu reduzieren, während ihre visuelle Qualität erhalten bleibt. Eine leistungsstarke Methode, um dies zu erreichen, ist die Verwendung der linearen Algebra, insbesondere der Singulärwertzerlegung (SVD). Diese Technik ermöglicht es uns, eine Bildmatrix in einer kompakteren Form darzustellen, indem weniger wichtige Informationen effektiv verworfen werden, während die wesentlichen Merkmale beibehalten werden.

Der folgende Python-Code zeigt, wie man ein Bild mit SVD komprimieren kann. Der Prozess umfasst das Zerlegen des Bildes in seine Bestandteile, das Komprimieren dieser Komponenten durch Beibehalten nur einer Teilmenge der wichtigsten Merkmale und dann das Rekonstruieren des komprimierten Bildes. Dieser Ansatz kann sowohl auf Graustufen- als auch auf Farbbilder angewendet werden und bietet eine flexible und mathematisch fundierte Methode zur Reduzierung der Bildgröße.

```
import numpy as np
from PIL import Image
import argparse
import os

def compress_image(image_path, compression_factor=0.1):
    # Öffne das Bild und konvertiere es in ein numpy-Array
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    # Überprüfe, ob das Bild Graustufen- oder Farbbild ist
    if len(img_array.shape) == 2:  # Graustufenbild
        # Führe SVD an dem Bild-Array durch
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

        # Komprimiere das Bild, indem nur die wichtigsten Singulärwerte beibehalten werden
        k = int(compression_factor * min(img_array.shape))
        S_compressed = np.diag(S[:k])
        U_compressed = U[:, :k]
        Vt_compressed = Vt[:k, :]

        # Rekonstruiere das komprimierte Bild
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else:  # Farbbild
        # Führe SVD an jedem Kanal getrennt durch
        img_compressed = np.zeros_like(img_array)
```

```
        for i in range(img_array.shape[2]):  # Iteriere über jeden Kanal
            channel = img_array[:, :, i]
            U, S, Vt = np.linalg.svd(channel, full_matrices=False)

            # Komprimiere den Kanal, indem nur die wichtigsten Singulärwerte beibehalten werden
            k = int(compression_factor * min(channel.shape))
            S_compressed = np.diag(S[:k])
            U_compressed = U[:, :k]
            Vt_compressed = Vt[:k, :]

            # Rekonstruiere den komprimierten Kanal
            img_compressed[:, :, i] = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))

    # Beschneide die Werte, damit sie zwischen 0 und 255 liegen, und konvertiere zurück zu uint8
    img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

    # Generiere den Ausgabepfad, indem '_compressed' zum ursprünglichen Dateinamen hinzugefügt wird
    file_name, file_extension = os.path.splitext(image_path)
    output_path = f"{file_name}_compressed{file_extension}"

    # Speichere das komprimierte Bild
    compressed_img = Image.fromarray(img_compressed)
    compressed_img.save(output_path)

    return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Komprimiere ein Bild mit SVD.")
    parser.add_argument("input_file", help="Pfad zur Eingabebilddatei")
    parser.add_argument("--compression_factor", type=float, default=0.1, help="Kompressionsfaktor (Standard: (
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f"Komprimiertes Bild gespeichert als: {output_file}")
```