画像圧縮を用いた線形代数

画像圧縮は、デジタル画像処理の基本的なタスクであり、画像の視覚的な品質を維持しながらその保存サイズを減少させることを目指しています。これを実現する強力な方法の一つは、線形代数、特に特異値分解（SVD）の使用です。この技術を使用すると、画像行列をよりコンパクトな形式で表現し、重要でない情報を効果的に捨てながら基本的な特徴を保持することができます。

以下の Python コードは、SVD を使用して画像を圧縮する方法を示しています。このプロセスには、画像をその構成要素に分解し、これらの要素を圧縮するために最も重要な特徴のみを保持し、その後圧縮された画像を再構築することが含まれます。このアプローチは、グレースケール画像とカラー画像の両方に適用でき、画像サイズを減少させるための柔軟で数学的に正しい方法を提供します。

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

def compress_image(image_path, compression_factor=0.1):
    # 画像を開き、NumPy 配列に変換
    img = Image.open(image_path)
    img_array = np.array(img, dtype=float)

    # 画像がグレースケールかカラーかを確認
    if len(img_array.shape) == 2:  # グレースケール画像
        # 画像配列に SVD を実行
        U, S, Vt = np.linalg.svd(img_array, full_matrices=False)

        # 画像を圧縮し、最も重要な特異値のみを保持
        k = int(compression_factor * min(img_array.shape))
        S_compressed = np.diag(S[:k])
        U_compressed = U[:, :k]
        Vt_compressed = Vt[:k, :]

        # 圧縮された画像を再構築
        img_compressed = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))
    else:  # カラー画像
        # 各チャンネルごとに SVD を実行
        img_compressed = np.zeros_like(img_array)
        for i in range(img_array.shape[2]):  # 各チャンネルに対して反復
            channel = img_array[:, :, i]
            U, S, Vt = np.linalg.svd(channel, full_matrices=False)
```

```
            # チャンネルを圧縮し、最も重要な特異値のみを保持
            k = int(compression_factor * min(channel.shape))
            S_compressed = np.diag(S[:k])
            U_compressed = U[:, :k]
            Vt_compressed = Vt[:k, :]

            # 圧縮されたチャンネルを再構築
            img_compressed[:, :, i] = np.dot(U_compressed, np.dot(S_compressed, Vt_compressed))

    # 値を 0 から 255 の間にクリップし、uint8 に戻す
    img_compressed = np.clip(img_compressed, 0, 255).astype(np.uint8)

    # 出力パスを生成し、元のファイル名に'_compressed'を追加
    file_name, file_extension = os.path.splitext(image_path)
    output_path = f"{file_name}_compressed{file_extension}"

    # 圧縮された画像を保存
    compressed_img = Image.fromarray(img_compressed)
    compressed_img.save(output_path)

    return output_path

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="SVD を使用して画像を圧縮します。")
    parser.add_argument("input_file", help="入力画像ファイルへのパス")
    parser.add_argument("--compression_factor", type=float, default=0.1, help="圧縮係数（デフォルト: 0.1）")
    args = parser.parse_args()

    output_file = compress_image(args.input_file, args.compression_factor)
    print(f"圧縮された画像が保存されました: {output_file}")
```