

Użycie SVD do rozkładów obrazów

Zadanie Twarze Własne

Sprawozdanie z ćwiczeń
Matematyka Konkretna

Data wykonania:
28.06.2025

Autor:
Bartosz Bieniek 058085

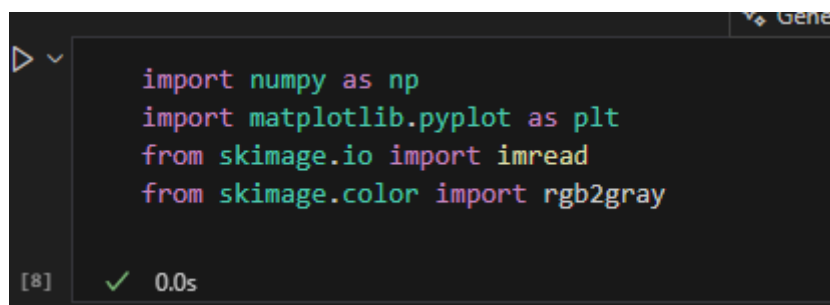
1. Cel ćwiczenia

Zadanie dotyczy liczby r wartości w lasnych (i odpowiednio "twarzy własnych") używając które możemy zachować więcej niż $k\%$ informacji zawartej w zdjęciu (określonym w wariantach do pierwszego zajęcia). Przedstawić wynikowe zdjęcie z użyciem odpowiedniej liczby eigenfaces. Zadanie umieścić na [Github](#).

2. Przebieg ćwiczenia

1. Import bibliotek

Zaimportowano niezbędne biblioteki umożliwiające wczytanie obrazu, jego przekształcenie oraz wykonanie dekompozycji SVD. Zastosowano numpy do obliczeń macierzowych, matplotlib.pyplot do wizualizacji, a skimage do operacji na obrazie i konwersji do skali szarości. Środowisko zostało w ten sposób przygotowane do analizy obrazów twarzy.



```
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread
from skimage.color import rgb2gray
```

[8] ✓ 0.0s

Rys. 1. Import bibliotek

2. Wczytanie i przygotowanie obrazu twarzy


Wczytano obraz twarzy z pliku i przekonwertowano go do skali szarości w celu uproszczenia przetwarzania danych. Uzyskana macierz jasności pikseli posłużyła jako podstawa do dalszych obliczeń. Oryginalny obraz wyświetlono w celu weryfikacji poprawności wczytania i konwersji.

```
# Wczytanie obrazu (np. 1.webp z poprzedniego zadania)
image = imread("1.webp")

# Konwersja do skali szarości
gray = rgb2gray(image)

# Wyświetlenie oryginału
plt.imshow(gray, cmap='gray')
plt.title("Oryginalny obraz twarzy")
plt.axis('off')
plt.show()
```

[9] ✓ 0.1s Python



Rys. 2. Wczytanie i przygotowanie obrazu twarzy

3. Obliczenie SVD obrazu

Na obrazie w skali szarości wykonano dekompozycję SVD, uzyskując trzy macierze: UU , SS oraz VT . Macierz SS zawierała wartości singularne odpowiadające rozkładowi energii informacji wizualnej. Dekompozycja ta umożliwiła późniejsze operowanie tylko na wybranych składnikach obrazu.

```
# Dekompozycja SVD
U, S, VT = np.linalg.svd(gray, full_matrices=False)
```

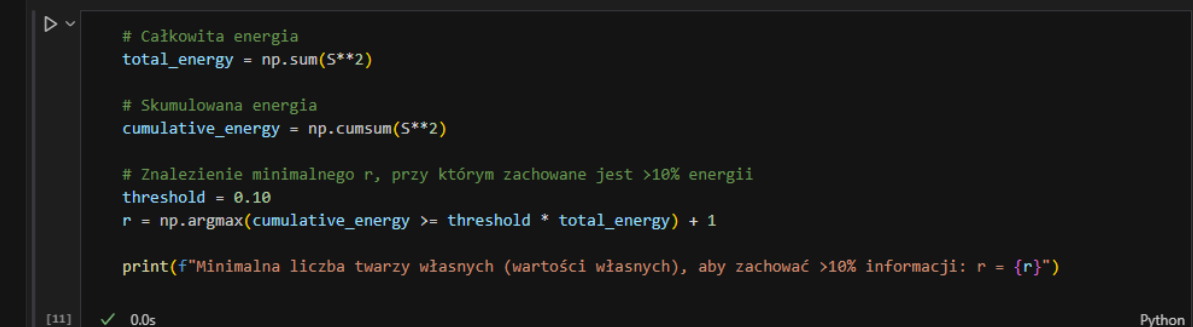
[10] ✓ 0.5s

Rys. 3. Obliczenie SVD obrazu

4. Obliczenie skumulowanej energii i minimalnej liczby twarzy własnych (r)

Obliczono całkowitą energię obrazu jako sumę kwadratów wartości singularnych. Następnie wyznaczono liczbę rr , dla której skumulowana energia przekroczyła 10% całkowitej wartości. Operacja ta pozwoliła określić

minimalną liczbę „twarzy własnych” potrzebnych do zachowania założonego poziomu informacji.



```
# Całkowita energia
total_energy = np.sum(S**2)

# Skumulowana energia
cumulative_energy = np.cumsum(S**2)

# Znalezienie minimalnego r, przy którym zachowane jest >10% energii
threshold = 0.10
r = np.argmax(cumulative_energy >= threshold * total_energy) + 1

print(f"Minimalna liczba twarzy własnych (wartości własnych), aby zachować >10% informacji: r = {r}")
```

[11] ✓ 0.0s Python

Rys. 4. Obliczenie skumulowanej energii i minimalnej liczby twarzy własnych (r)

5. Odtworzenie obrazu z r twarzy własnych (eigenfaces)

Na podstawie pierwszych r wartości własnych odtworzono przybliżony obraz twarzy. Wykorzystano zredukowane macierze $U_r U_r^T$, $S_r S_r$, $V_r V_r^T$, tworząc ich iloczyn jako przybliżenie oryginału. Obraz zrekonstruowany z ograniczonej liczby składników został wyświetlony w celu oceny wizualnej jakości kompresji.

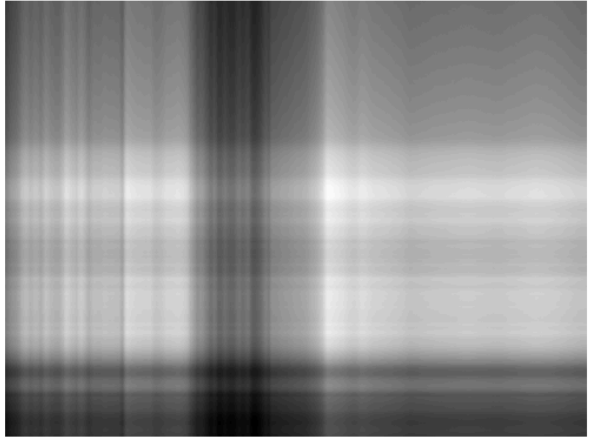
```
# Przycięcie do r składników
U_r = U[:, :r]
S_r = np.diag(S[:r])
VT_r = VT[:, :r]

# Rekonstrukcja
reconstructed = U_r @ S_r @ VT_r

# Wyświetlenie obrazu
plt.imshow(reconstructed, cmap='gray')
plt.title(f"Obraz zrekonstruowany z r={r} twarzy własnych (>10% informacji)")
plt.axis('off')
plt.show()
```

[12] ✓ 0.1s Python

Obraz zrekonstruowany z r=1 twarzy własnych (>10% informacji)



Rys. 5. Odtworzenie obrazu z r twarzy własnych (eigenfaces)

3. Wnioski

Na podstawie obliczeń ustalono, że do zachowania ponad 10% informacji zawartej w obrazie wystarczyło wykorzystać r wartości własnych (twarzy własnych).

Zrekonstruowany obraz zachował rozpoznawalne cechy strukturalne twarzy, pomimo znacznej redukcji danych.

Zastosowanie metody eigenfaces pozwala na efektywną kompresję obrazów twarzy z zachowaniem istotnych informacji wizualnych.