```
import pywt
import numpy as np
import cv2
from collections import Counter
import os

def haar_wavelet_transform(image):
    """Выполняет вейвлет-преобразование Хаара."""
    coeffs = pywt.dwt2(image, 'haar')
    LL, (LH, HL, HH) = coeffs
    return LL, LH, HL, HH

def quantize(coeffs, levels=4):
    """Квантует матрицу, разделяя ее на заданное количество
уровней."""
    min_val = np.min(coeffs)
    max_val = np.max(coeffs)
    if max_val == min_val:
        return np.zeros_like(coeffs, dtype=np.int32), min_val, 0
    step = (max_val - min_val) / levels
    quantized = np.round((coeffs - min_val) / step).astype(np.int32)
    return quantized, min_val, step


def run_length_encode(matrix):
    """Сжимает матрицу с помощью кодирования частот."""
    encoded = []
    for value, count in Counter(matrix.flatten()).items():
        encoded.append((value, count))
    return encoded

def save_transformed_image(ll, lh, hl, hh, lh_min, lh_step, hl_min,
hl_step, hh_min, hh_step, filename, mode="text"):
    """Сохраняет преобразованные и сжатые данные в файл."""
    with open(filename, 'w' if mode == 'text' else 'wb') as f:
        if mode == "text":
            np.savetxt(f, ll, fmt='%.3f')
            f.write("\n")
            for value, count in lh:
                f.write(f"{value} {count}\n")
            f.write(f"min: {lh_min} step: {lh_step}\n")
            f.write("\n")
            for value, count in hl:
                f.write(f"{value} {count}\n")
            f.write(f"min: {hl_min} step: {hl_step}\n")
            f.write("\n")
            for value, count in hh:
                f.write(f"{value} {count}\n")
            f.write(f"min: {hh_min} step: {hh_step}\n")
        elif mode == "binary":
```

```
            ll.tofile(f)
            np.array(lh,dtype=np.int32).tofile(f)
            np.array([lh_min,lh_step],dtype=np.float32).tofile(f)
            np.array(hl,dtype=np.int32).tofile(f)
            np.array([hl_min,hl_step],dtype=np.float32).tofile(f)
            np.array(hh,dtype=np.int32).tofile(f)
            np.array([hh_min,hh_step],dtype=np.float32).tofile(f)


if __name__ == "__main__":
    img_path = 'sar_1.jpg'
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)

    # Выполняем преобразование Хаара
    ll, lh, hl, hh = haar_wavelet_transform(img)

    # Квантование
    n_quants = 4
    quantized_lh, lh_min, lh_step = quantize(lh, n_quants)
    quantized_hl, hl_min, hl_step = quantize(hl, n_quants)
    quantized_hh, hh_min, hh_step = quantize(hh, n_quants)

    # Кодирование длин серий
    encoded_lh = run_length_encode(quantized_lh)
    encoded_hl = run_length_encode(quantized_hl)
    encoded_hh = run_length_encode(quantized_hh)

    # Сохраняем
    save_transformed_image(ll, encoded_lh, encoded_hl, encoded_hh,
lh_min, lh_step, hl_min, hl_step, hh_min, hh_step,
"transformed_image_new.txt", mode="text")
    save_transformed_image(ll, encoded_lh, encoded_hl, encoded_hh,
lh_min, lh_step, hl_min, hl_step, hh_min,
hh_step,"transformed_image_new.bin", mode="binary")


    # Сравнение размеров
    original_size = img.nbytes
    text_size = 0
    with open("transformed_image_new.txt", 'r') as f:
        text_size = len(f.read().encode('utf-8'))

    binary_size = os.path.getsize("transformed_image_new.bin")

    print(f"Размер оригинального изображения: {original_size} байт")
    print(f"Размер сжатого текстового файла: {text_size} байт")
    print(f"Размер сжатого бинарного файла: {binary_size} байт")
    print(f"Степень сжатия (текстового): {original_size /
text_size:.2f}")
```

```
    print(f"Степень сжатия (бинарного): {original_size /
binary_size:.2f}")

Размер оригинального изображения: 262144 байт
Размер сжатого текстового файла: 85452 байт
Размер сжатого бинарного файла: 78235 байт
Степень сжатия (текстового): 3.07
Степень сжатия (бинарного): 3.35
```