Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **МК «Машиностроительный»**

КАФЕДРА **МК10 «Высшая математика и физика»**

Домашняя работа №2

«Обработка изображений на основе вейвлетпреобразований»

ДИСЦИПЛИНА: «Вейвлет-преобразование сигналов»

Выполнил: студент гр. ИУК4-11М _____ (Сафронов Н.С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Степанов С.Е.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2024

Задание 1

Рассмотрим матрицу изображения X размера $8x8$. Элемент $x_{ij} = (Gr \cdot i + Num \cdot j)\ mod\ 23$ представляет интенсивность сигнала в точке (i, j).

a) Запишите формулы преобразования двумерного сигнала, используя матричное представление.

b) Примените прямое вейвлет-преобразование Хаара к изображению, используя матрицу анализа W_a.

c) Обнулите все вейвлет коэффициенты, меньшие по абсолютной величине, чем 1.

d) Восстановите сигнал, используя матрицу синтеза W_s.

e) Найдите норму L_1 погрешности между оригинальным и восстановленным изображением.

В отчёт включите матрицы преобразования двумерного сигнала, исходную матрицу X, матрицу вейвлет-коэффициентов Y, матрицу после обнуления части коэффициентов Y_1, восстановленное изображение X_1, норму L_1 погрешности ϵ.

Решение

$$\begin{bmatrix} 0 & 7 & 14 & 21 & 5 & 12 & 19 & 3 \\ 4 & 11 & 18 & 2 & 9 & 16 & 0 & 7 \\ 8 & 15 & 22 & 6 & 13 & 20 & 4 & 11 \\ 12 & 19 & 3 & 10 & 17 & 1 & 8 & 15 \\ 16 & 0 & 7 & 14 & 21 & 5 & 12 & 19 \\ 20 & 4 & 11 & 18 & 2 & 9 & 16 & 0 \\ 1 & 8 & 15 & 22 & 6 & 13 & 20 & 4 \\ 5 & 12 & 19 & 3 & 10 & 17 & 1 & 8 \end{bmatrix}$$

Рисунок 1.1 – Исходная матрица

```python
def haar_matrix(n):
    """
    Строит матрицы анализа (W_a) для вейвлета Хаара.
    """
    n = 2 ** np.ceil(np.log2(n))

    if n > 2:
        h = haar_matrix(n / 2)
    else:
        return np.array([[1, 1], [1, -1]])

    h_n = np.kron(h, [1, 1])
    h_i = np.kron(np.eye(len(h)), [1, -1])
    h = np.vstack((h_n, h_i))
    h /= np.sqrt(np.linalg.norm(h, ord=1, axis=-1, keepdims=True))
    return h
```

Рисунок 1.2 – Функция построения матрицы анализа Хаара

```
W_a = haar_matrix(n)
W_s = np.linalg.inv(W_a)
```

```
sp.Matrix(np.round(W_a, 3))
```

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & -0.25 & -0.25 & -0.25 & -0.25 \\ 0.42 & 0.42 & -0.42 & -0.42 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.42 & 0.42 & -0.42 & -0.42 \\ 0.707 & -0.707 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.707 & -0.707 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.707 & -0.707 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.707 & -0.707 \end{bmatrix}$$

```
sp.Matrix(np.round(W_s, 3))
```

$$\begin{bmatrix} 0.5 & 0.5 & 0.595 & 0.0 & 0.707 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.5 & 0.595 & 0.0 & -0.707 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.5 & -0.595 & 0.0 & 0.0 & 0.707 & 0.0 & 0.0 \\ 0.5 & 0.5 & -0.595 & 0.0 & 0.0 & -0.707 & 0.0 & 0.0 \\ 0.5 & -0.5 & 0.0 & 0.595 & 0.0 & 0.0 & 0.707 & 0.0 \\ 0.5 & -0.5 & 0.0 & 0.595 & 0.0 & 0.0 & -0.707 & 0.0 \\ 0.5 & -0.5 & 0.0 & -0.595 & 0.0 & 0.0 & 0.0 & 0.707 \\ 0.5 & -0.5 & 0.0 & -0.595 & 0.0 & 0.0 & 0.0 & -0.707 \end{bmatrix}$$

Рисунок 1.3 – Результат построения матриц анализа и синтеза Хаара

```
Y = W_a @ X @ W_a.T
sp.Matrix(np.round(Y, 3))
```

$$\begin{bmatrix} 41.875 & 1.5 & -6.622 & 3.048 & -1.768 & 2.298 & -1.768 & 2.298 \\ -0.375 & 0.0 & 2.418 & 2.418 & -8.132 & 4.066 & 0.0 & -4.066 \\ -3.784 & 0.0 & -8.132 & 0.0 & 0.0 & 0.0 & -6.838 & 6.838 \\ 1.051 & 0.0 & 4.066 & -4.066 & 13.676 & -6.838 & 6.838 & 0.0 \\ 2.475 & 0.0 & -6.838 & -6.838 & 0.0 & -11.5 & 0.0 & 11.5 \\ 2.475 & 0.0 & -6.838 & 6.838 & 0.0 & 11.5 & -11.5 & 0.0 \\ 2.475 & -8.132 & 0.0 & 0.0 & 0.0 & 0.0 & 11.5 & -11.5 \\ 2.475 & 0.0 & -6.838 & -6.838 & 0.0 & -11.5 & 0.0 & 11.5 \end{bmatrix}$$

Рисунок 1.4 – Построение матрицы вейвлет-коэффициентов

```
threshold = 1
Y_thresholded = np.where(np.abs(Y) < threshold, 0, Y)
sp.Matrix(np.round(Y_thresholded, 3))
```

$$\begin{bmatrix} 41.875 & 1.5 & -6.622 & 3.048 & -1.768 & 2.298 & -1.768 & 2.298 \\ 0.0 & 0.0 & 2.418 & 2.418 & -8.132 & 4.066 & 0.0 & -4.066 \\ -3.784 & 0.0 & -8.132 & 0.0 & 0.0 & 0.0 & -6.838 & 6.838 \\ 1.051 & 0.0 & 4.066 & -4.066 & 13.676 & -6.838 & 6.838 & 0.0 \\ 2.475 & 0.0 & -6.838 & -6.838 & 0.0 & -11.5 & 0.0 & 11.5 \\ 2.475 & 0.0 & -6.838 & 6.838 & 0.0 & 11.5 & -11.5 & 0.0 \\ 2.475 & -8.132 & 0.0 & 0.0 & 0.0 & 0.0 & 11.5 & -11.5 \\ 2.475 & 0.0 & -6.838 & -6.838 & 0.0 & -11.5 & 0.0 & 11.5 \end{bmatrix}$$

Рисунок 1.5 – Обнуление коэффициентов, меньших 1

```
X_rec = W_s @ Y_thresholded @ W_s.T

sp.Matrix(np.round(X_rec, 3))
```

$$\begin{bmatrix} 0.094 & 7.094 & 14.094 & 21.094 & 5.094 & 12.094 & 19.094 & 3.094 \\ 4.094 & 11.094 & 18.094 & 2.094 & 9.094 & 16.094 & 0.094 & 7.094 \\ 8.094 & 15.094 & 22.094 & 6.094 & 13.094 & 20.094 & 4.094 & 11.094 \\ 12.094 & 19.094 & 3.094 & 10.094 & 17.094 & 1.094 & 8.094 & 15.094 \\ 15.906 & -0.094 & 6.906 & 13.906 & 20.906 & 4.906 & 11.906 & 18.906 \\ 19.906 & 3.906 & 10.906 & 17.906 & 1.906 & 8.906 & 15.906 & -0.094 \\ 0.906 & 7.906 & 14.906 & 21.906 & 5.906 & 12.906 & 19.906 & 3.906 \\ 4.906 & 11.906 & 18.906 & 2.906 & 9.906 & 16.906 & 0.906 & 7.906 \end{bmatrix}$$

Рисунок 1.6 – Результат восстановления сигнала

```
L1_error = np.sum(np.abs(X - X_rec))
L1_error
```

```
np.float64(6.0)
```

Рисунок 1.7 – Норма L_1 погрешности

Задание 2

a) Загрузите изображение размера 256×256, 512×512 или 1024×1024, используя шкалу и цветовую карту gray.

b) Проведите одноуровневое разложение, используя вейвлет Добеши 4.

c) Постройте изображения вейвлет-коэффициентов.

d) Проведите сжатие изображения, оставив 20% наибольших по модулю коэффициентов (коэффициент сжатия 0,8), восстановите его.

e) Проведите многоуровневое разложение, используя вейвлет Добеши 4 (число уровней 3-4).

f) Постройте изображение только коэффициентов аппроксимации последнего уровня разложения.

g) Проведите сжатие изображения, оставив 20% наибольших по модулю коэффициентов (коэффициент сжатия 0,8), восстановите его.

h) Повторите пункты a) – g), используя симметричные 4 и биортогональные 4,4 вейвлеты.

i) Сделайте вывод, какой из вейвлетов лучший при одноуровневом, а какой – при многоуровневом разложении

Решение

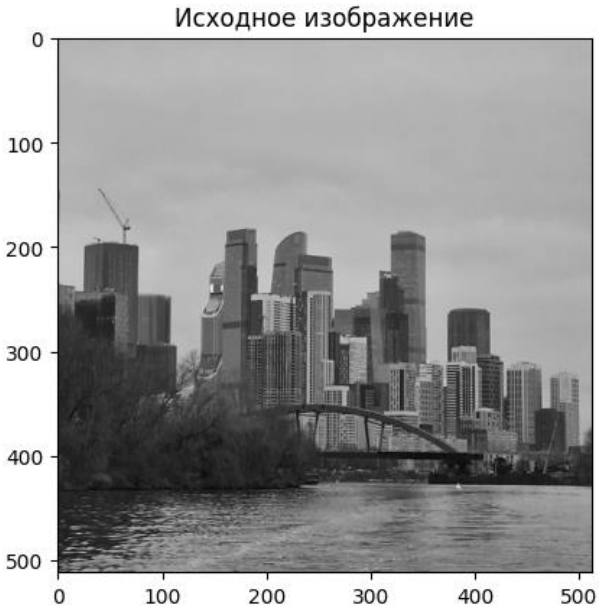


Рисунок 2.1 – Исходное изображение

```python
def single_level_decomposition(image, wavelet):
    coeffs2 = pywt.dwt2(image, wavelet, mode='periodization')
    cA, (cH, cV, cD) = coeffs2
    return cA, cH, cV, cD

def multilevel_decomposition(image, wavelet, level):
    coeffs = pywt.wavedec2(image, wavelet, level=level)
    return coeffs

def compress_single_level_image(cA, cH, cV, cD, wavelet_name):
    flat_coeffs = np.hstack([cA.ravel(), cH.ravel(), cV.ravel(), cD.ravel()])
    threshold = np.percentile(np.abs(flat_coeffs), 80)

    compressed_coeffs = (
        np.where(np.abs(cA) < threshold, 0, cA),
        (
            np.where(np.abs(cH) < threshold, 0, cH),
            np.where(np.abs(cV) < threshold, 0, cV),
            np.where(np.abs(cD) < threshold, 0, cD),
        )
    )
    image_compressed = pywt.idwt2(compressed_coeffs, wavelet_name)
    return image_compressed

def compress_multilevel_image(coeffs, wavelet_name):
    arr, coeff_slices = pywt.coeffs_to_array(coeffs)
    flat_coeffs = np.abs(np.concatenate([c.flatten() for c in arr]))
    threshold = np.percentile(np.abs(flat_coeffs), 80)
    compressed_coeffs = [pywt.threshold(c, threshold, mode='hard') for c in arr]
    compressed_coeffs = pywt.array_to_coeffs(arr, coeff_slices, 'wavedec2')
    image_compressed = pywt.waverec2(compressed_coeffs, wavelet_name)
    return image_compressed
```

```python
def plot_wavelet_coeffs(coeffs, reconstructed_image):
    fig, axs = plt.subplots(3, 2, figsize=(6, 8))
    titles = [
        'cA (аппроксимация)',
        'cH',
        'cV',
        'cD',
        f'Восстановленное изображение',
    ]
    coeff_images = [*coeffs, reconstructed_image]
    for i, ax in enumerate(axs.ravel()):
        if i < len(coeff_images):
            ax.imshow(coeff_images[i], cmap='gray')
        if i < len(titles):
            ax.set_title(titles[i])
        ax.axis('off')
    plt.show()
```

Рисунок 2.2 – Вспомогательные функции

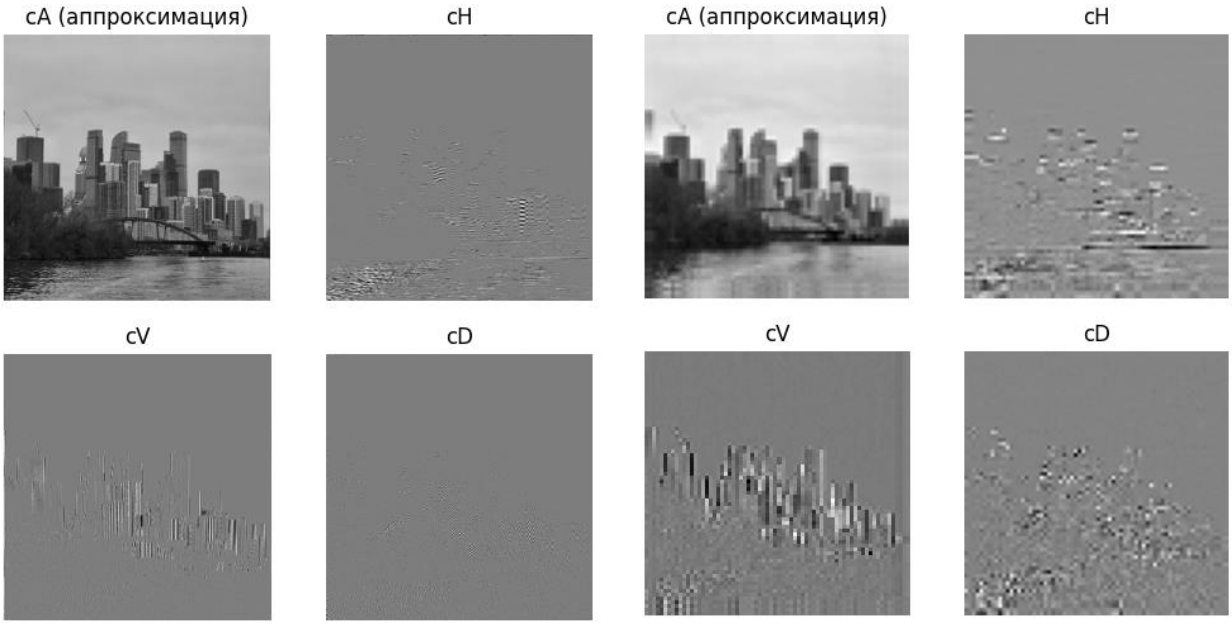
```python
for wavelet_name in ['sym4', 'bior4.4', 'db4']:
    print(f"\nИспользуем вейвлет: {wavelet_name}")

    cA, cH, cV, cD = single_level_decomposition(image, wavelet_name)
    image_compressed = compress_single_level_image(cA, cH, cV, cD, wavelet_name)

    plot_wavelet_coeffs([cA, cH, cV, cD], image_compressed)

    multilevel_coeffs = multilevel_decomposition(image, wavelet_name, 3)
    cA = multilevel_coeffs[0]
    cH, cV, cD = multilevel_coeffs[1]
    image_compressed = compress_multilevel_image(multilevel_coeffs, wavelet_name)
    plot_wavelet_coeffs([cA, cH, cV, cD], image_compressed)
```

Рисунок 2.3 – Код выполнения задания



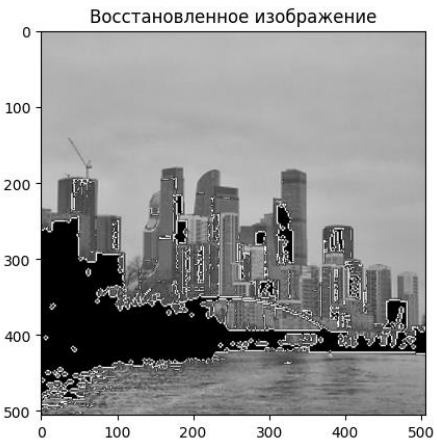


Рисунок 2.4 – Одноуровневое разложение симметричного вейвлета 4

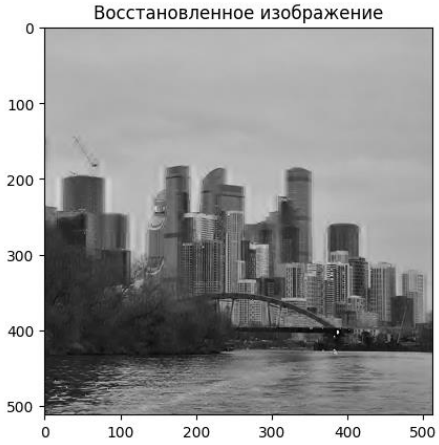
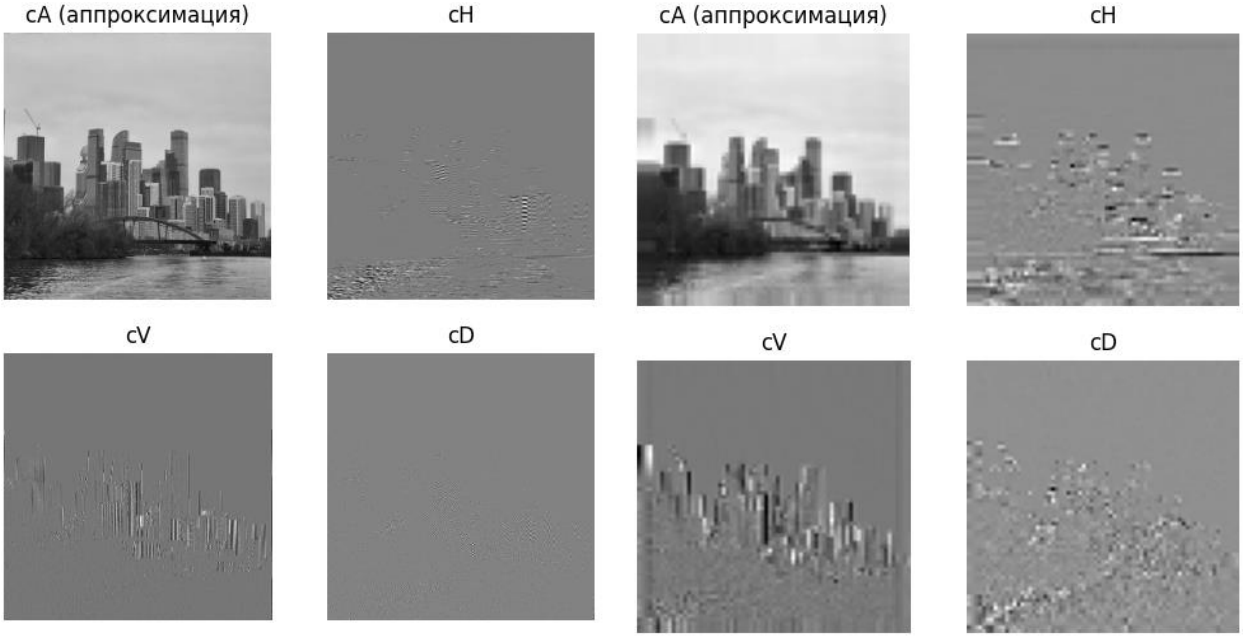


Рисунок 2.5 – Многоуровневое разложение симметричного вейвлета 4



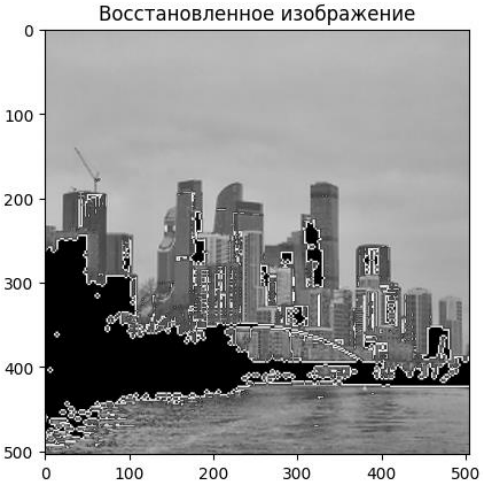


Рисунок 2.6 – Одноуровневое разложение биортогонального вейвлета 4.4

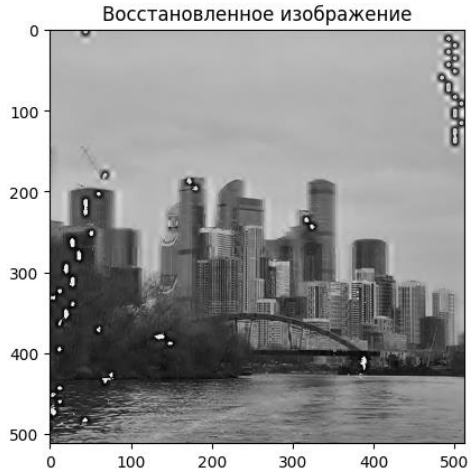
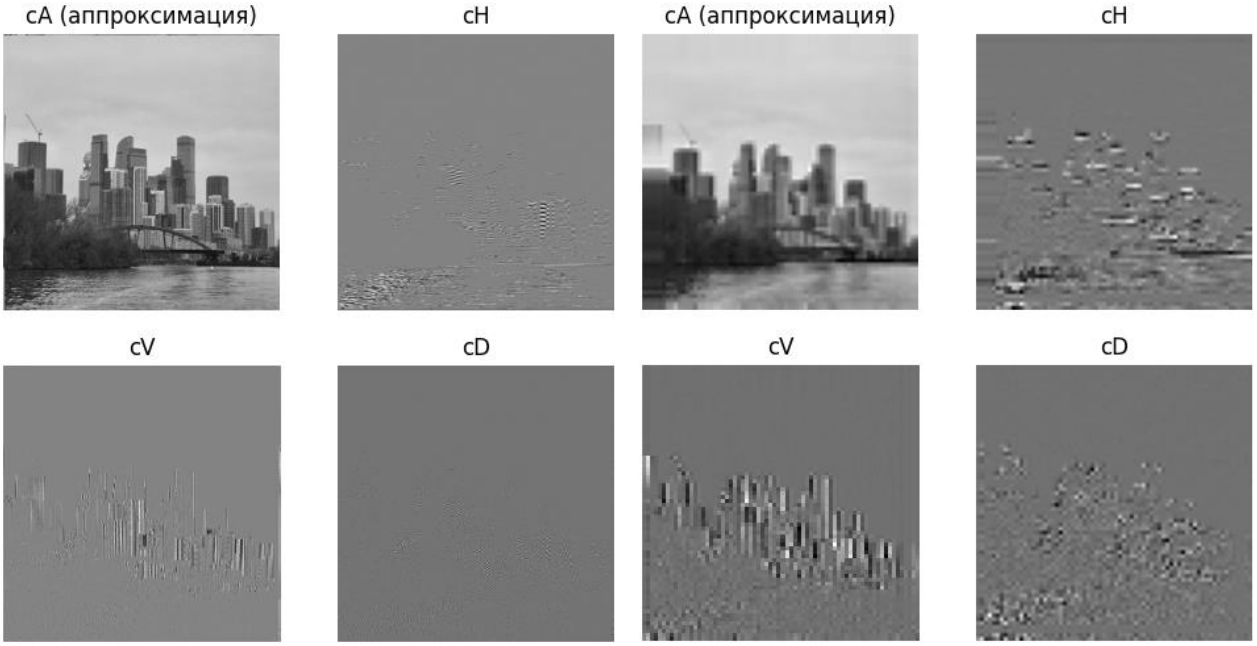


Рисунок 2.7 – Многоуровневое разложение биортогонального вейвлета 4.4



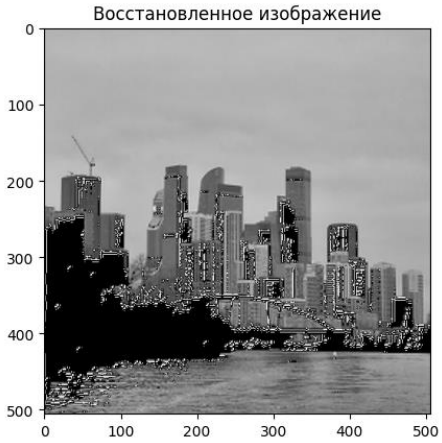


Рисунок 2.8 – Одноуровневое разложение вейвлета Добеши 4

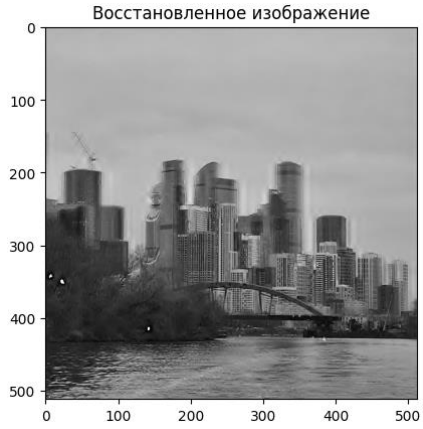


Рисунок 2.9 – Многоуровневое разложение вейвлета Добеши 4

Исходя из полученных результатов, лучшим восстановлением при одноуровневом разложении является вейвлет Добеши 4, а при многоуровневом вейвлет симметричный 4.

Задание 3

a) Загрузите цветное изображение размера 256x256, 512x512 или 1024x1024.

b) Наложите на изображение белый шум со стандартным отклонением $\sigma = 0{,}5$

c) Найдите соотношение сигнал/шум до устранения.

d) Проведите устранение шума, используя вейвлет Хаара, порог – мягкий, метод - байесовский

e) Найдите соотношение сигнал/шум после устранения.

f) Повторите пункты a) – e), используя вейвлеты Добеши 4, симметричные 4 и биортогональные вейвлеты 4.4 (используемые по умолчанию)

g) Составьте таблицу, включив в неё соотношения сигнал/шум, выберите наилучший вейвлет для устранения шума.

h) Постройте рисунок с оригинальным, зашумлённым и синтезированным изображениями.

Решение

```
wavelets = ['haar', 'db4', 'sym4', 'bior4.4']
results = {}

for wavelet in wavelets:
    denoised_image = denoise_image(noisy_image, wavelet)
    denoised_image = resize(denoised_image, original_image.shape)
    snr_after = psnr(original_image, denoised_image)
    results[wavelet] = snr_after

    plt.figure(figsize=(12, 4))
    plt.subplot(1, 3, 1)
    plt.imshow(original_image, cmap='gray')
    plt.title('Оригинальное изображение')
    plt.axis('off')

    plt.subplot(1, 3, 2)
    plt.imshow(noisy_image, cmap='gray')
    plt.title(f'Зашумлённое изображение\nSNR: {snr_before:.2f}')
    plt.axis('off')

    plt.subplot(1, 3, 3)
    plt.imshow(denoised_image, cmap='gray')
    plt.title(f'После устранения шума\nWavelet: {wavelet}\nSNR: {snr_after:.2f}')
    plt.axis('off')

    plt.show()

print("\nРезультаты SNR для разных вейвлетов:")
print("Wavelet\t\tSNR (после устранения шума)")
for wavelet, snr_value in results.items():
    print(f"{wavelet}\t\t{snr_value:.2f}")

best_wavelet = max(results, key=results.get)
print(f"\nЛучший вейвлет для устранения шума: {best_wavelet}")
```

Рисунок 3.1 – Результат выполнения задания

Рисунок 3.2 – Устранение шума вейвлетом Хаара

Рисунок 3.3 – Устранение шума вейвлетом Добеши 4

Рисунок 3.4 – Устранение шума вейвлетом симметричным 4

Рисунок 3.5 – Устранение шума вейвлетом биортогональным 4.4

```
Результаты SNR для разных вейвлетов:
Wavelet         SNR (после устранения шума)
haar            15.10
db4             11.91
sym4            11.28
bior4.4         11.70

Лучший вейвлет для устранения шума: haar
```

Рисунок 3.6 – Соотношение сигнала к шуму для разных вейвлетов

Наилучше убрал шумы вейвлет Хаара, т.к. его значение соотношение сигнала к шуму больше.

Задание 4

Сигнал $f(t)$ задан в 16 точках отрезка $[0, \pi]$ и представляет собой сумму

• тренда $s(t) = t(Num + t)^{1/2}$

• периодической компоненты

$$p(t) = \begin{cases} \cos(2 \cdot (\mathrm{Gr} \cdot \mathrm{t} + \mathrm{Num})), \mathrm{t} \in [0, \pi/2] \\ \cos(\mathrm{Gr} \cdot \mathrm{t} + \mathrm{Num}), \mathrm{t} \in [\pi/2, \pi] \end{cases}$$

• случайной составляющей r(t) со значениями, равномерно распределёнными на отрезке [-0,5; 0,5];

• импульсного шума n(t) со значением -8 в точке 5□/16 и равного 0 в остальных точках.

a) проведите разложение сигнала в вейвлет-пакет, используя вейвлет Добеши 4;

b) постройте дерево разложения. В каждом узле укажите массив коэффициентов;

c) обнулите все листья четвертого уровня разложения, абсолютные значения которых меньше 0,4, восстановите сигнал. Посчитайте, сколько

ненулевых значений использовано. На одном рисунке постройте график исходного сигнала и график восстановленного сигнала;

d) Обнулите во всех массивах третьего уровня разложения коэффициенты, абсолютные значения которых меньше 0,4. Восстановите сигнал, используя только наборы третьего уровня разложения. Посчитайте, сколько ненулевых значений использовано. На одном рисунке постройте график исходного сигнала и график восстановленного сигнала;

e) Обнулите во всех массивах второго уровня разложения коэффициенты, абсолютные значения которых меньше 0,4. Восстановите сигнал, используя только наборы второго уровня разложения. Посчитайте, сколько ненулевых значений использовано. На одном рисунке постройте график исходного сигнала и график восстановленного сигнала;

f) Обнулите во всех массивах первого уровня разложения коэффициенты, абсолютные значения которых меньше 0,4. Восстановите сигнал, используя только наборы первого уровня разложения. Посчитайте, сколько ненулевых значений использовано. На одном рисунке постройте график исходного сигнала и график восстановленного сигнала;

g) выберите лучший (с наименьшим числом ненулевых коэффициентов) уровневый базис разложения;

h) рассчитайте лучший базис вейвлет-пакета для данного сигнала, используя пороговый уровень 0,4. Постройте его. Восстановите сигнал, используя этот базис. Посчитайте, сколько ненулевых значений использовано. На одном рисунке постройте график исходного сигнала и график восстановленного сигнала.

Решение

```
t = np.linspace(0, np.pi, N)
s = t * (Num + t) ** 0.5
p = np.piecewise(
    t,
    [t <= np.pi / 2, t > np.pi / 2],
    [
        lambda t: np.cos(2 * (Gr * t + Num)),
        lambda t: np.cos(Gr * t + Num)
    ],
)
r = np.random.uniform(-0.5, 0.5, N)
n = np.zeros(N)
n[5] = -8
```

```
f = s + p + r + n
```

Рисунок 4.1 – Исходный сигнал

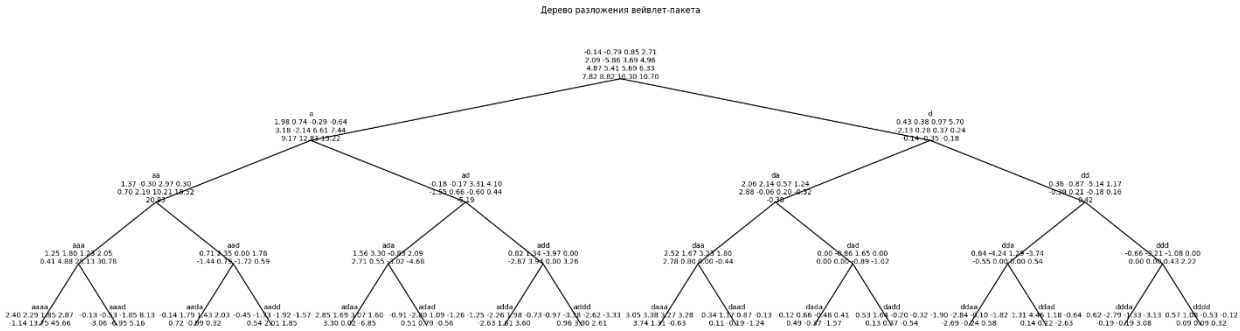


Рисунок 4.2 – Дерево разложения

Рисунок 4.3 – Восстановленный при обнулении сигналов 4 уровня и исходный сигналы

Рисунок 4.4 – Восстановленный при обнулении сигналов 3 уровня и исходный сигналы

Рисунок 4.5 – Восстановленный при обнулении сигналов 2 уровня и исходный сигналы

Рисунок 4.6 – Восстановленный при обнулении сигналов 1 уровня и исходный сигналы

```
print(f"Число ненулевых коэффициентов:")
print(f"1 уровень: {nonzero_1}, 2 уровень: {nonzero_2}, 3 уровень: {nonzero_3}, 4 уровень: {nonzero_4}")

best_level = min([(1, nonzero_1), (2, nonzero_2), (3, nonzero_3), (4, nonzero_4)], key=lambda x: x[1])
print(f"Лучший уровень разложения: {best_level[0]} с {best_level[1]} ненулевыми коэффициентами")
```

```
Число ненулевых коэффициентов:
1 уровень: 13, 2 уровень: 25, 3 уровень: 51, 4 уровень: 85
Лучший уровень разложения: 1 с 13 ненулевыми коэффициентами
```

Рисунок 4.7 – Лучший уровень разложения – 1 уровень

```python
def print_zeros_tree(wp: pywt.WaveletPacket, threshold=0.4, level=0):
    size = 144

    for level in range(wp.maxlevel + 1):
        res = 0
        print(level, end = ' ')
        for node in wp.get_level(level): # type : pywt.Node
            zeros_count = np.count_nonzero(np.where(np.abs(node.data) < threshold, 0, node.data))
            res += zeros_count
            print(f'{node.path + ":" if node.path else ""} {zeros_count}'.center(size), end='')
        size = int(size / 2)
        print(res)
```

```python
print_zeros_tree(wp, threshold=0.4)
```

```
0                                                       15                                                      15
1                                a: 10                                                  d: 3                     13
2              aa: 9                                ad: 6                                  da: 5                    dd: 5                25
3       aaa: 8           aad: 7           ada: 8           add: 7           daa: 6           dad: 4           dda: 7           ddd: 4      51
4  aaaa: 6  aaad: 6  aada: 5  aadd: 5  adaa: 6  adad: 7  adda: 7  addd: 7  daaa: 7  daad: 4  dada: 5  dadd: 3  ddaa: 5  ddad: 4  ddda: 6  dddd: 2 85
```

Рисунок 4.8 – Количество ненулевых элементов в каждом узле

Таким образом, лучший базис совпадает с базисом первого уровня разложения, так как он требует использования наименьшего количества ненулевых коэффициентов, а выбор узлов лежащих ниже уровней лишь увеличивает их количество.