Task 3

Fourier-Transform and 2-D Wavelet Transform

Christian Riess (christian.riess@fau.de)

1 One Last Remark on Even/Odd Functions: Duality in Time and Frequency Domain

This exercise is somewhat related to our previous theoretical exercises — just to make sure that everyone is on the same page. Let

$$c(t) = \cos(k \cdot t) \ , \tag{1}$$

denote a function where k determines the frequency of the oscillation, and

$$r(t) = \begin{cases} 1, & -\frac{w}{2} \leq t < \frac{w}{2} \\ 0, & \text{otherwise} \end{cases} \ , \tag{2}$$

denote a rectangle function where w determines the width of the rectangle.

1. Plot both functions in matlab in the domain $[-6, 6[$. As a starting point, use the parameters $k = 3$ and $w = 2$.
2. Compute the Fourier transforms of both functions *analytically*.
3. Compute the Fourier transform of the functions in matlab. Use the functions `fft` (Fast Fourier Transform) and `fftshift`.
4. Plot the real and imaginary parts of the Fourier coefficients, respectively.
5. In spatial and frequency domain, what is the relationship between even and odd functions, as well as real and imaginary functions? Validate whether/how these relationships can be confirmed from the analytical solution and the matlab plots.
6. Vary the parameters k and w. Explain your observations. What is the relation between these scaled functions in spatial and frequency domain?
7. Compute the square $r(\omega)$ (i.e., the Fourier transform of $r(t)$). Then, compute its inverse Fourier transform in matlab using `ifft`. Plot the functions in spatial and frequency domain. What do you observe? Explain your observation with the convolution theorem.
8. Have a closer look at the properties of the `fft` function in matlab. What is the difference of that result to the analytic solution? In order to better see that, plot the functions and their Fourier transform in the domain $[-2\pi, 2\pi]$ and compare these findings with the previously created plots. What is different? And what is the reason for that?

2 Discrete Haar Transformation in 2-D

Let us make our wavelet implementation slightly more complex (and useful). Extend the 1-D Haar transform from the previous exercise to two dimensions, such that we can process images. The core idea is to first apply the 1-D wavelet transform to the image rows, then to the image columns. The reason why this works is that the 2-D wavelet transform is separable. You may either use the code template from the webpage or extend your existing code.

2.1 Analysis

Implement the analysis filter bank for 2-D discrete Haar transform in the function `dht2` (in the code template available on the web page). To keep things simple, please assume that we operate on images where the height N and width M are powers of 2, i.e., $N = 2^n$ and $M = 2^m$.

1. Perform a 1-D transformation along the rows of an image. You will obtain approximation and detail coefficients with $N \times M/2$ entries.
2. Further decompose this intermediate result using a 1-D transform along the columns. You will obtain four sets of coefficients with $N/2 \times M/2$ entries each.

2.2 Synthesis

Implement the synthesis filter bank to recover the original image in the function `idht2` (in the code template available on the web page).

1. Calculate first the inverse 1-D Haar transform along the columns. To achieve this, combine the approximation coefficients with the horizontal detail coefficients, and the vertical detail coefficients with the diagonal detail coefficients.
2. Restore the original image by performing an inverse 1-D Haar transform along the rows of the obtained coefficients.

3 2-D Haar Multiresolution and Compression

Extend the implementation by a 2-D multiresolution analysis (MRA).

1. Compute n iterations of `dht2` in the function `mra2` (in the code template available from the web page).
2. Concatenate blocks of coefficients to an image and return it.
3. Compute n iterations of `idht2` in the function `mrs2` (in the code template available from the web page).
4. Compute a complete multi-resolution analysis for the phantom image provided by matlab (see code template). Compute the number of coefficients that are not zero in the wavelet representation and the original image. Assume that only values that are not zero have to be stored or transmitted. What compression rate would you achieve if transmitting the wavelet representation instead of the original image?

5. Add noise to the phantom image, and recompute the compression ratio. What do you observe? And how do you explain your observation?

6. As a second example, examine the image `magic.png` (available from the web page). What do you observe here?

7. Another quick look into lossy compression: introduce a threshold, where all coefficients below that threshold are set to zero. Compute the compression rate and the mean quadratic error. Is there a good compromise between data loss and compression?