Homework 4
Transformations: DCT and KLT

- Deadline: Thursday, 2020-06-25, 23:45
- Submission: Upload your code on Cody Coursework$^{\text{TM}}$ Mathworks platform `https://grader.mathworks.com/courses/12790-dsp-ss20` according to the instructions given there for all problems marked with (Cx.y).
- For all problems marked with (Ax.y) refer to the quiz questions at `https://www.moodle.tum.de/course/view.php?id=54118`. There you can also find the homework rules and more information.
- Result verification: Check your functions as often as you want with Cody Coursework$^{\text{TM}}$.
- Lab session for MATLAB questions – see website, room 0943
- Notation: `variable name`, *`file name`*, `MATLAB_function()`

1 Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) is a separable linear transform used in image and video compression. It exhibits very beneficial properties for energy compaction in natural images. Contrary to the DFT, the DCT is a real-valued transformation.

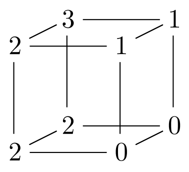
1. (C1.1) Here you will transform images to DCT domain using block processing. For this write a function that takes as input DCT transformation matrix, function handle to block splitter and image for transformation. Its output will be the image transformed into DCT domain. The provided handle to block splitter has the same signature as in previous problems. When displaying the resulting image for visualization purposes, remember to use appropriate scaling. *(5 points)*

2. (A1.1) How do the coefficients in each block in step (1) correspond to the contents of the original image? Calculate the variances and means of the DCT coefficients, using all blocks from the transformed image. Use your previously implemented function `calculate_statistics()`, from **Homework 3**. How does the mean matrix look like? How is it related to the mean of the untransformed image? Create a 3D plot of the variances. What "rule" for the decay of the variances do you observe? *(6 points)*

3. (C1.2) In this task you will perform compression of a DCT-transformed image. For each transformed block of the Lenna image (block size 16×16), keep only the first N coefficients and set the remaining coefficients to zero. Use an appropriate quadratic sub-block of side length $\sqrt{N}$, starting with the DC coefficient. Return the image (in spatial domain) resulting from compression with the given coefficients in ☞ `im_rec_comp`. You are also given as arguments the function handle to split image into blocks as well as DCT transformation matrix. *(7 points)*

4. (A1.2) Calculate the PSNR for each compression level for image in step (3). What effects do you observe for the different compression levels ($N = 9, 16, 36, 64$) in step (3)? What are the critical image regions? Do your observations justify equal compression for all image blocks? In JPEG, coefficients are ordered using the zig-zag scheme. Explain why this is a better way to order them "by importance". *(6 points)*

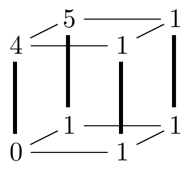
5. (C1.3) Because of its separable property DCT transformation can be efficiently extended to 3 dimensions. With the following example, we provide how 3D DCT can be applied seperately in each dimension of the following 3D block:



Since the M-D version of the DCT is separable, we can successively apply the 1-D transform given by `dct_matrix` along the three dimensions independently.

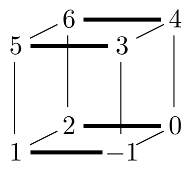
In no particular order, we choose to apply it first on all the vertical columns:

$$\texttt{dct\_matrix}\left[\begin{array}{c|c|c|c} 2 & 1 & 3 & 1 \\ 2 & 0 & 2 & 0 \end{array}\right] = \frac{1}{\sqrt{2}}\left[\begin{array}{c|c|c|c} 4 & 1 & 5 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right] \Longrightarrow \frac{1}{\sqrt{2}}\cdot$$



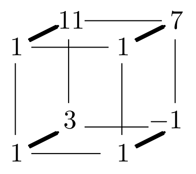
then on all the horizontal rows of the result thereof:

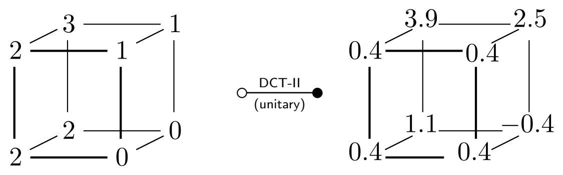
$$\texttt{dct\_matrix}\left[\begin{array}{c|c|c|c} 4 & 0 & 5 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right]\frac{1}{\sqrt{2}} = \frac{1}{2}\left[\begin{array}{c|c|c|c} 5 & 1 & 6 & 2 \\ 3 & -1 & 4 & 0 \end{array}\right] \Longrightarrow \frac{1}{2}\cdot$$



and eventually, we apply `dct_matrix` on the "out-of-plane rows" to obtain the final result:

$$\texttt{dct\_matrix}\left[\begin{array}{c|c|c|c} 6 & 4 & 2 & 0 \\ 5 & 3 & 1 & -1 \end{array}\right]\frac{1}{2} = \frac{1}{\sqrt{8}}\left[\begin{array}{c|c|c|c} 11 & 7 & 3 & -1 \\ 1 & 1 & 1 & 1 \end{array}\right] \Longrightarrow \frac{1}{\sqrt{8}}\cdot$$





In this question you will apply DCT matrix along each of the 3 dimensions subsequently to get the 3D transformed DCT image. Make sure that you get the same result as in the above example. *(9 points)*

6. (C1.4) Here you will apply the previously computed 3D DCT for analysis of the 3D sequence, in this case video sequence. This way, the third dimension extends along the time dimension of the sequence and allows to encode temporal correlation efficiently. You are given the handle to the 3D DCT transformation function (signature as in problem before). Return the DC and high-frequency components of the resulting DCT transformed sequence. You can ignore padding of the sequence for this problem. *(7 points)*

7. (A1.3) Which DCT component of the 3D DCT transformed sequence contains the most energy? What is the best strategy for video compression using 3D DCT? *(6 points)*

(1) high dynamic content
(2) high frequency spatial, static content over time

2 Karhunen-Loève-Transform

The Karhunen-Loève-Transform (KLT) performs a complete decorrelation of correlated data and thus allows for the best energy concentration using a linear transform. It must be trained from a specific dataset (e.g. a set of images or a set of image blocks), but shows certain general properties, which depend on the statistics of the dataset. Work with formulas given in the lecture using the autocorrelation matrix C_{xx} and do not use built-in MATLAB functions for correlation blindly.

1. (C2.1) *Preparatory step:* As explained in the lecture, 2D image blocks are reshaped into 1D column vectors for the KLT, which allows us to use matrix-vector notation. The block is read out column by column, starting from the top-left (column-major order!). Like that, an 32×32 (base) image is reshaped into a 1024×1 vector. Implement a block processor which reshapes the block into a vector, adds 0.1 to the vector elements `[529:544,272:32:784]` and reshapes the result back to a block. Transform the image with block size 32 and save the result to ☞ `im_reshape`. Make sure you understand how reshaping is performed and how the resulting image is generated. *(5 points)*

2. (C2.2) Calculate the KLT basis functions from the *lenna_gray.jpg* image for 8×8 blocks. The following steps are needed to be performed:

   - First, restructure image into blocks of 8×8 size and return the result in ☞ `blocks`.
   - Calculate the autocorrelation matrix for blocks of 8×8 size and save the result to ☞ `C_im`. `C_im` should have the size of 64×64.
   - Perform the eigenvalue decomposition of the autocorrelation matrix `C_im`.
   - Sort the resulting eigenvalues with descending order and correspondingly the eigenvectors.
   - The resulting eigenvector constitutes the KLT basis images. Reshape each vector to a block of 8×8 and pack them all into a 64×64 image in column-major order.

   Your function should return the restructured image in ☞ `blocks`, autocorrelation matrix ☞ `C_im`, and the KLT basis functions in ☞ `klt_base`. *(10 points)*

3. (A2.1) Compare KLT basis images when using different images for training. *(5 points)*

4. (C2.3) Write a function to project the image to the (same) KLT basis which was obtained above. Return the resulting images in ☞ `im_projected`. Coefficients should be ordered the same way as the basis images. Also compute the back-projected image and return in ☞ `im_backprojected`. Compare the back-transformed image with the original ones and return the calculated PSNR in ☞ `klt_psnr`. *(8 points)*

5. (C2.4) Write a function to calculate the autocorrelation matrix of all blocks from KLT transformation and return in ☞ `autocorr`. Here you can reuse your approach for block grouping and autocorrelation computation from Problem 2.2. *(5 points)*

6. (A2.2) How should the autocorrelation matrix of KLT blocks look like? When evaluating function from step (5), compare results for image *lenna_gray.jpg* and *mandrill_gray.png*. When transforming *mandrill_gray.png* use the KLT basis images trained from *lenna_gray.jpg*. Explain the result for the *mandrill_gray.png* image considering the properties of the KLT. Is KLT suitable as a generic compression method? *(8 points)*

7. (A2.3) Calculate the statistics, ☞ `klt_mean` and ☞ `klt_var`, of the KLT transformed images (using the KLT lenna basis images) *lenna_gray.jpg* and *mandrill.png*. You can use your previously implemented function `calculate_statistics()`, from **Homework 3**. Then, plot the resulting variances using `surf(log(klt_var))`. How do you explain the differences between the decay in variance (or energy) for the two images? *(8 points)*

8. (A2.4) Repeat the previous step for DCT-transformed versions of the *lenna_gray.jpg* image, using this time `semilogy()` to visualize the variance of the transform coefficients in log scale. How does the performance of the DCT and KLT transformations differ? *(5 points)*.

`[V,D] = eig(A)` returns diagonal matrix `D` of eigenvalues and matrix `V` whose columns are the corresponding right eigenvectors, so that `A*V = V*D`.