# Homework 6
# Subband Decomposition and Lifting

- Deadline: Wednesday, 2020-02-05, 23:45
- Submission: Upload your code on Cody Coursework$^{TM}$ Mathworks platform `https://grader.mathworks.com/courses/9867-mdsp-ws-2019-20` 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=50238`. There you can also find the homework rules and more information.
- Result verification: Check your functions as often as you want with Cody Coursework$^{TM}$.
- Lab session for MATLAB questions – see website, room 0943
- Notation: `variable name`, *`file name`*, `MATLAB_function()`

## 1 Subband Decomposition for DCT

A linear block transform, such as the DCT, can be expressed as a subband decomposition and implemented as a two-level filter bank (see lecture). In the following, verify that the DCT can be considered a special case of a discrete wavelet transform. Before you start with the code, sketch the mathematical reasoning for the equivalence of the two representations.

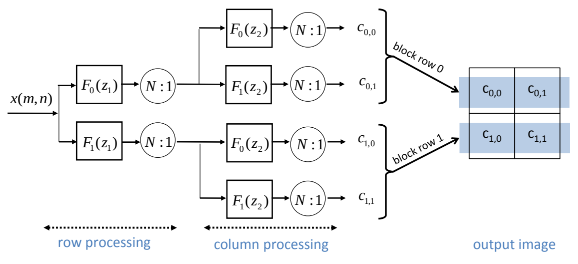

Figure 1: Simple filter bank with 2 channels and separable row/column processing

1. (A1.1) What should be the filters to be used in filter bank to implement DCT as subband decomposition? Compare frequency responses of the filters when doing 8-channel subband decomposition using DCT of size 8x8. How do these differ? Which decimation factor do you have to use to compute subband decomposition with such dimensions? *(6 points)*

2. (C1.1) Build a 2D filter bank with separated row and column processing according to Fig. 1. Pay attention that in the figure subband decomposition for the case $N = 2$ is given, while in our case $N = 8$. Use $N$ channels in each level for row and column processing with critical sampling. The channels correspond to the eight 1D DCT basis functions. Put the output blocks into a single image (☞ `dct_subband`), ordered as in Fig. 1. Hint: if you apply filtering using `imfilter()` function, it is important that you start with the correct offset when decimating in order to include only the image parts that fully overlap with the filter kernel. *(14 points)*

3. (C1.2) In this problem you will use the Matlab symbolic toolbox to verify the condition for perfect reconstruction given analysis filters $F_0(z)$ and $F_1(z)$. These conditions are given in lecture in page

124. They are defined in the form $F_x(z) = a_x + b_x \cdot z^{-1} + c_x \cdot z$. Your function should return true if the conditions are satisfied, and false otherwise. For an example of how symbolic toolbox works, see `mathworks.com/help/symbolic/create-symbolic-numbers-variables-and-expressions.html`. To convert a symbolic value back to numeric, you can use `double().` Prior to coversion of symbolic values back to numeric ones, you need to call `simplify()` to simplify algebraic expressions. *(15 points)*

## 2 Lifting Structure with Motion Compensation

In this problem, a lifting structure is applied to a special application: It is used to compress a video in the time domain, taking advantage of a static background and known motion vectors (see Fig. 2). Using this side information, the update and prediction steps predict a frame from the previous one. Please review also the basic lifting structure in the beginning of lecture chapter 6.

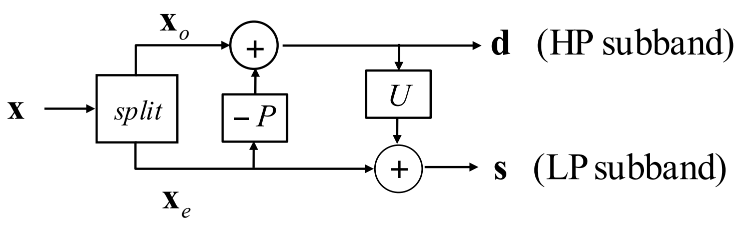

Figure 2: The analysis part of a filter bank with **P**rediction and **U**pdate steps (from lecture slides).

1. A function `make_video_web()` is provided on the website which generates a video sequence in a 3D array `video`, with time (frame number) in the third dimension. Resulting dimensions are $240 \times 320 \times 16$. This artificial video serves as an approximation for a real-world video with a static background and a single moving object. A mask for this object is given in `mask`, and the object positions (top-left element of the bounding box) are provided for each frame in `positions`.
   Run the function, look at the returned variables and write a routine that shows the video. Be careful with copying the `video` array too much, as it requires a lot of memory. *(0 points)*

2. (C2.1) For the analysis part, start by splitting the video into even and odd frames. Note that the first frame ($n = 1$) is an even frame, despite the different indexing in MATLAB. Now implement a **prediction** step that predicts frame $n+1$ using frame $n$, the position/motion vector for the object and the object mask. Therefore, using the mask, cut out the object from frame $n$ and fill the uncovered background with a gray-level of 0.5. Then, paste the object to the new location given by the motion vector and save the *negated* (see the "-" sign in front of prediction result in Fig. 2) output of the predictor for the first frame to ☞ `predict1`. This way, after you fill the uncovered area with gray level in the even frame and then paste the object at a new position, you need to invert the resulting image!
   Implement an **update** operator, which moves the object region in the difference signal back to where it was in the even frame. Save the operator's output for the first odd frame to ☞ `update1`. Uncovered background should be left unchanged.
   A function `paste_mask()` is provided for copying and pasting operations. See Cody Coursework for the details of its usage and the signature. Please note that you have to do prediction and update steps only for the first frame ($n = 1$). *(18 points)*

3. (A2.1) What do you expect to happen when you add the predicted image and the odd frame? Verify your arguments! *(5 points)*

4. (C2.2) Finally, apply the complete analysis to the entire video, obtaining two subbands (HP and LP) with $\frac{N}{2}$ frames each ($N$ is the total number of frames). Calculate the variances for a LP-subband frame and save the variance to ☞ `var_0lp`. Repeat the analysis hierarchically for the LP-subband, until there is only one image left in each subband. Return the variances you obtain for each frame of the HP-subband in ☞ `var_hp`. Also return the single retaining LP-image in ☞ `im_lp`. *(18 points)*

5. (A2.2) What do you observe from previously computed variances for each subband `var_0lp`, `var_hp`? What does the computed image show? Does it correspond to your expectation? *(6 points)*

6. (C2.3) Implement the synthesis side, i.e., use the hierarchy of HP-subband images and the last LP-frame to reconstruct the original video ☞ `video`. Make sure the original and the synthesized

signals are equal. Input arguments are: `pyr-` cell array (its elements illustrated in Fig. 3) with 5 elements, where the first 4 cell elements are HP-subbands (from first to the fourth) with dimensions $240 \times 320 \times 8$, $240 \times 320 \times 4$, $240 \times 320 \times 2$, $240 \times 320 \times 1$, respectively. The 5th element is LP-subband of size $240 \times 320 \times 1$. `pyr_vec` is a cell array ($1 \times 5$) that contains motion vectors for each subband for the respective frames, having dimensions $2 \times 16$, $2 \times 8$, $2 \times 4$, $2 \times 2$ and $2 \times 1$, respectively. It thus has the same ordering as `pyr`. `mask` is an object mask (as before). *(18 points)*

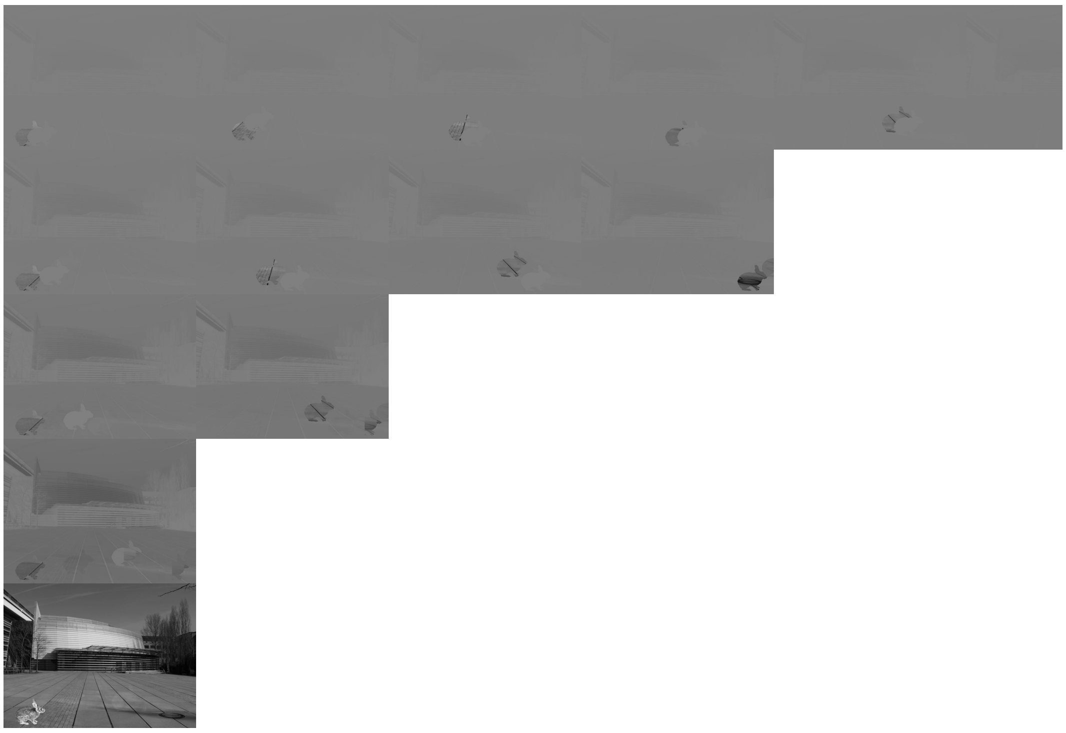

Figure 3: Lifting pyramid is the result of analysis steps for the video sequence and is used as input for synthesis. The bottom subfigure is the LP-subband and the remaining ones are HP-subbands of sizes 1,2,4 and 8, respectively. Please note that the HP-subbands have been grayed out for evaluation reasons (i.e., true HP-subbands will differ from these).