Homework 5
Autoregressive Models and Pyramids

- Deadline: Thursday, 2020-07-09, 23:45
- Submission: Upload your code on Cody Coursework™ 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™.
- Lab session for MATLAB questions – see website, room 0943
- Notation: `variable name`, *`file name`*, `MATLAB_function()`

1 Autoregressive Models

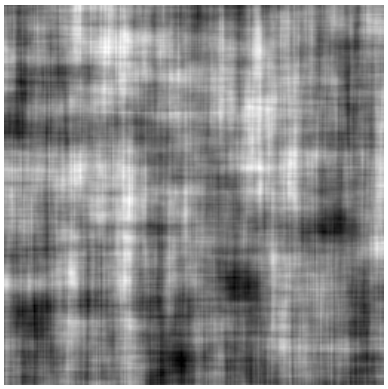
An autoregressive process is a generator for correlated random sequences which show similar local statistics as natural images. The simplest case for a 2D signal is a separable AR(1) process, which exhibits an autocovariance of:

$$C_{xx}(k,l) = \sigma_x^2 \cdot \rho_h^{|k|} \cdot \rho_v^{|l|}; \qquad \sigma_x^2 = \frac{\sigma_z^2}{(1-\rho_h^2)\,(1-\rho_v^2)} \tag{1}$$

The covariance between neighbouring pixels may be obtained by plugging in their distance k, l (in the horizontal and vertical direction respectively). The model has 3 parameters σ_x, ρ_h, ρ_v and can be implemented by a causal recursive filter according to:

$$x_{AR}(x,y) = \rho_h x_{AR}(x-1,y) + \rho_v x_{AR}(x,y-1) - \rho_h\rho_v x_{AR}(x-1,y-1) + \eta_{\sigma_z}(x,y), \tag{2}$$

using a zero-mean Gaussian noise source η_{σ_z} with standard deviation σ_z.



Autocorrelation:

$R_{xx} = E[xx^T]$

Auto covariance

$C_{xx} = E[(x-\mu_x)(x-\mu_x)^T]$

Figure 1: A 256×256 image generated by an AR(1) process

1. (C1.1) Calculate the autocovariance of a natural image I, using specified neighbourhood 11×11 for each pixel as follows: **definition**

$$C_{II}(k,l) = E\left[I(x,y) \cdot I(x+k, y+l)\right] \forall x, y \quad \text{with} \quad -5 \le k, l \le 5$$

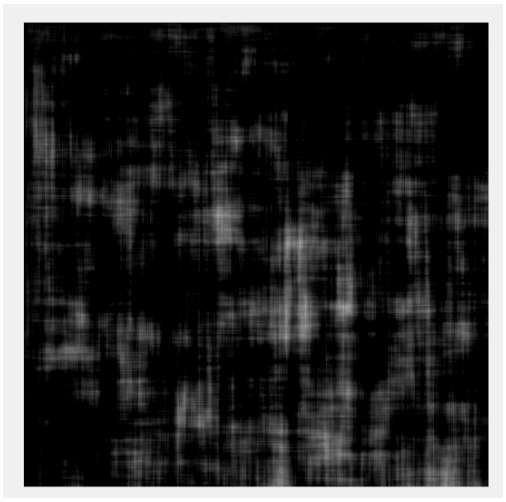
Before processing, do not forget to subtract the mean value of the image. Save the autocovariance matrix to ☞ `acov`. Taking into account a 5 pixel border, you will calculate the covariances using $(512-10) \times (512-10)$ measurements assuming the image of size 512×512. Extract the variance (σ_x^2, ☞ `var`) and the covariances for 1 pixel horizontal (☞ `cov_h`) and vertical displacement (☞ `cov_v`). *(10 points)*

2. (A1.1) Discuss the values for `cov_v, cov_h, acov, var` you obtained in step (1) for the image ***lenna_gray.png*** and explain the structure of the autocovariance matrix. Plot autocovariance matrix as a 3D surface. *(10 points)*

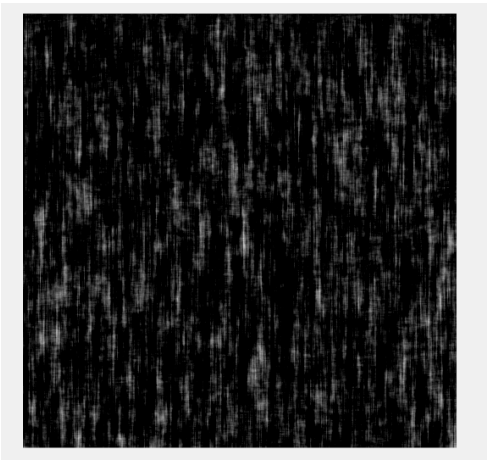
3. (C1.2) Generate a 512×512 image I_{AR} (☞ `ar_gen`) using the separable AR(1) process Eqn. (2). For this, you are given covariance in horizontal `cov_h` and vertical directions `cov_v` as well as variance `var`. Based on this and Eqn. (1), you would need to compute rv, rh and σ_z. The process will need some pixels to "start up", so add a temporary border of 100 pixels filled with zeros to the top and to the left. To verify your results, you can measure the autocovariance of this image in an 11×11 neighbourhood. *(10 points)*

4. (A1.2) Compute parameters of the equivalent AR process such that C_{xx} in Eqn. (1) yields the three values you obtained from the natural image in step (1) at the corresponding positions (k, l). Compare the autocovariance of the image I_{AR} (☞ `ar_gen`) (from step (3)) and compare it to Eqn. (1). Identify the shortcomings of the autocovariance matrix of I_{AR} computed in step (3) compared to the natural autocovariance matrix from step (1). *(10 points)*

5. (A1.3) Repeat step (1) with image ***bell-south.jpg*** and save the resulting autocovariance matrix. Explain why this matrix looks considerably different, even though it is also obtained from a natural image. How would an image generated by an AR(1) process with parameters measured from ***bell-south.jpg*** look like? *(5 points)*



lena



bell (more vertical structure)

2 Pyramid Representations

Pyramid or multiscale representations are very useful for many image processing tasks. For instance, they form the fundamental preprocessing step in modern image feature transforms such as SURF[1], which are used in computer vision applications.

1. (C2.1) A simple 3×3 approximation of a Gaussian lowpass filter is given on lecture page 109. In this problem, assume you used this filter for 2:1 reduction (horizontally and vertically). Now you need to reconstruct the image to the original resolution. For this, write a routine to compute an appropriate 3x3 reconstruction filter for 1:2 upsampling. The filter can be found by inspection. *(8 points)*

2. (C2.2) Generate a Laplacian pyramid of the given image *`lenna_gray.jpg`* using the filters from step 1 with 5 levels. Level 1 holds the finest details at full resolution (512×512), while level 5 corresponds to the lowest level of detail (of resolution 32×32). Like that, the scaling factor between two levels is 2:1 horizontally and vertically. Use `imfilter` with the `replicate` option and return your results from the function for all levels as a cell array ☞ `laplacian_pyr{1,2,3,4,5}`. *(10 points)*

3. (C2.3) In this problem you will work with the 9-7 Cohen-Daubechies-Feauveau wavelet[2]. In particular, you are asked to obtain the 2D filter kernels for analysis and synthesis steps, ☞ `F` and ☞ `G` respectively. You are already given separable 1D lowpass filter coefficients for these wavelets. *(5 points)*

4. (C2.4) Write a function that calculates the variances of each level for your pyramid representation (you will observe results for pyramids from steps 2 & 3 and compare the variances when using Gaussian filters and 9-7 Cohen-Daubechies-Feauveau wavelets). Return the variances in 5×1 vector ☞ `var_pyramid.` To neglect border effects, remove a border of 4 pixels around the images at each level. *(7 points)*

5. (A2.1) Explain the difference you observe for variances when using 9-7 Cohen-Daubechies-Feauveau wavelet and Gaussian filters. *(5 points)*

6. (C2.5) In this problem you will need to reconstruct the image from the pyramid representation. Instead of taking all levels of the pyramid, set the two levels with the finest details to 0. After that reconstruct the images at full resolution and return them in ☞ `im_reconstruct.` Calculate the PSNR of the reconstructed images and return it in ☞ `im_psnr` – again neglecting a border of 4 pixels. *(10 points)*

7. (A2.2) Explain the values you obtain for PSNR for two reconstruction filters in step 6. What kind of artifacts do you observe? Is there a difference between the representations from steps 2 and 3? *(10 points)*

[1] Bay et al., SURF: Speeded Up Robust Features, European Conference on Computer Vision, 2006
[2] http://en.wikipedia.org/wiki/Cohen-Daubechies-Feauveau_wavelet

Desktop

编辑器 - C:\Users\baiyunming\Desktop\aawd.m

videoCodec.m | motionEstimation_halfpel.m | SSD_rec_halfpel.m | enc_huffman_new.m | compare_still_video_halfpel.m | aawd.m

```
f1d = [0.0267487574108098, -0.0168641184428750, -0.0782232665289879, 0.266864118442872, 0.602949018236358, 0.266864118442872, -0.078

% Synthesis filter coefficients
g1d = [-0.0912717631142495, -0.0575435262284996, 0.591271763114247, 1.11508705245699, 0.591271763114247, -0.0575435262284996,

[hf,wf] = freqz(f1d,1);
[hg,wg] = freqz(g1d,1);
plot(wf,abs(hf));
figure;
plot(wg,abs(hg));
```