Applied Signal Processing Laboratory

Assignment 3 - Introduction to audio and image processing

Daniel Gaetano Riviello

A.A. 2024-2025

14/05/2025

Exercise 1 - Pitch estimation and spectrogram

1. Create a function called `my_pitch_estimator` to estimate the pitch of an audio signal as the inverse of the argument of the first maximum (argmax) of the autocorrelation function after lag zero (as illustraded in slide 25 of Lec09). The function has the following specifications:
   - input arguments of the function are an audio signal vector `x`, the sampling frequency f_s, and a boolean variable `pl_on`.
   - the function outputs the estimated pitch and, if `pl_on` is equal to true, shows the plot of the normalized autocorrelation function (use `xcorr` and x-axis in seconds) with a vertical dashed line in correspondence of the inverse of the pitch (use `xline`). The plot will also display a text description next to the dashed line with the estimated pitch in Hz and corresponding lag in seconds (use `text`).
   - Use `findpeaks` to find all maxima of the autocorrelation function and their locations. Check carefully the options of `findpeaks` to avoid possible local maxima in the function.
2. Read the file `A2_guitar.wav` with `audioread`. Work only on the first channel.

Exercise 1

3. Normalize the signal by its maximum absolute value, create an `audioplayer` object and play it with `playblocking`.
4. Plot the signal in time domain. Estimate and plot the power spectral density (y-axis in dB) with the `pwelch` command: use segments with at least 2000 samples, a Hamming window, 50% overlap, and option `'centered'`.
5. Plot the spectrogram with the command `spectrogram` by setting a Hamming window of 4000 samples, 90% overlap, $2^{12} = 4096$ total samples for the FFT and limit the frequency axis from 0 to 8 kHz (check slide 49 of Lec09 as an example).
6. Apply `my_pitch_estimator` to the audio signal `A2_guitar.wav` with `pl_on` set to true. How close to 110 Hz is it? Include the estimated value in the report.
7. Read the file `Bending.wav` with `audioread`. Work only on the first channel.
8. Repeat all steps from 3 to 5 with the new audio signal. Comment on the spectrogram.

Exercise 1

9. Cut and isolate each of the 3 notes of the signal. Estimate the pitch of the first 2 notes with `my_pitch_estimator` and `pl_on=true`. Based on the note frequency chart in NoteFreq, can you tell which notes were played in the WAVE file?
10. The third note is bent, i.e., the pitch of the note is increased by a tone through string bending. For this last note, cut exactly 1.5 seconds of duration after the first 2.1 seconds of the audio signal, divide it into frames of 0.1 seconds and use a sliding window technique with 90% overlap to estimate the pitch of each frame with `my_pitch_estimator` and `pl_on=false`. The total number of frames S can be computed as $S = \left\lfloor \frac{N-L}{M-L} \right\rfloor$, where N is number of samples corresponding to 1.5 seconds, M the number of samples contained in 0.1 seconds, and $L = 0.9\,M$.
11. Plot the estimated pitch for each time frame vs. the time instants related to the central index of each frame. Can you tell which is the starting note (lowest frequency), and the target note (peak of the bending, highest frequency)?

Exercise 1

12. Consider now the total signal `Bending.wav` (first channel only) and filter it with a low-pass FIR filter with cut-off frequency $f_c = 5$ kHz, transition band $B_T = 500$ Hz, choose a window with minimum stopband attenuation $A_s \geq 50$ dB.
13. Under-sample your filtered signal by a factor 4 (just select one every 4 samples of your filtered signal), set the new sampling rate $f_{s,2} = f_s/4$ and show the new spectrogram.
14. Repeat step 3, can you hear any difference from the original?
15. Reconsider the original signal and repeat step 12 with a new $f_c = 2.5$ kHz, repeat step 13 with factor 8, set $f_{s,3} = f_s/8$ and repeat step 3. How different does the new signal sound?

Exercise 1

16. Finally, estimate the pitch of your voice in the following way:
    - Create an `if-else` statement, if the recording of your voice doesn't exist (you can use the function `isfile`), create an `audiorecorder` object and record 3 seconds of your voice pronouncing "car", "cheese", or a stretched vowel, use $f_s = 8$ kHz, 16 bits, single channel, write it to a WAVE file with `audiowrite`.
    - If you have already recorded your voice, read it with `audioread`.
17. Repeat steps 3 and 4 with your voice, adjust `pwelch` parameters.
18. Cut your voice signal by keeping only the vocalized part and estimate your pitch with `my_pitch_estimator` and `pl_on=true`.

Exercise 2 - Digital audio sythesis

You will generate a simple chord progression by using subtractive synthesis to generate the single notes of a bowed string instrument and additive synthesis to generate the chords. Set $f_s = 44100$ Hz.

1. **Note generation**: each note has a duration of $T = 2.5$ seconds and is generated by means of subtractive synthesis. First generate a sawtooth waveform with the function `sawtooth` and argument $2\pi f_0 t$ with t being the vector of time instants and f_0 the fundamental frequency of the note. Now generate a Pulse Width Modulation (PWM) signal by comparing the amplitude of the sawtooth with a low frequency (lfo) sine amplitude. When the sawtooth is less than the sine set the resulting signal equal to 1, otherwise 0. (Check Fig.2 of Wikipedia PWM). Set $lfo = f_0/200$. Filter the PWM signal with a low pass FIR filter with window method (no `fir1` allowed) with transition band $B_T = 15\, f_0$, passband frequency edge $\omega_p = f_0$ and stopband edge $\omega_s = 16\, f_0$, compute the corresponding cut-off frequency as $\frac{\omega_p + \omega_s}{2}$ and number of samples N. Use a Bartlett window. Normalize the filter impulse response by its sum.

Exercise 2

2. **Chord generation**: Use additive synthesis to generate a chord or triad. A triad is a harmonic set (sum) of 3 notes. Given a root note, a major triad contains the root, a major third (ratio $2^{4/12}$ w.r.t. the root) and a perfect fifth (ratio $2^{7/12}$). A minor triad only differs for the minor instead of major third (ratio $2^{3/12}$). To generate any note, you can take as reference A4 at 440 Hz and compute the distance in semitones from A4 (one semitone is a $2^{1/12}$ interval for the equal temperament). For example, if you need to generate a C5 major triad, first set $f_0 = 440 \cdot 2^{3/12}$, then generate the root, the third major with $f_{3M} = f_0 \cdot 2^{4/12}$ and the perfect fifth with $f_5 = f_0 \cdot 2^{7/12}$. Normalize each generated chord signal by its maximum. Make a local function within the script for both the note generation and chord generation.

Exercise 2

3. **ADSR envelope**: generate a simple ADSR envelope by using the function `interp1`. Time instants to interpolate are $[0 \;\; 0.16T \;\; 0.32T \;\; 0.6T \;\; T]$ with amplitudes $[0 \;\; 1 \;\; 0.7 \;\; 0.7 \;\; 0]$. Use the option `'pchip'` for cubic interpolation to smooth the edges, type `help interp1` for more details. Multiply the generated ADSR envelope with each chord signal elementwise.
4. **Chord progression**: you can create a chord progression by appending multiple chord vectors to a single vector. Try to play the following famous progression ('m' indicates a minor chord):

   D, A, Bm, F#m, G, D, G, A

   All roots are meant in the forth octave (as A4, G4, etc.), except for D5, which is one octave higher,'#' indicates a sharp (one semitone higher). Create an `audioplayer` object and play the progression, finally write it to a WAVE file.

Exercise 2

5. Finally apply a digital reverb to the chord progression:
   - Open the file `impulse_revcathedral.wav` containing the impulse response of a cathedral with `audioread`. Since the sampling frequency $f_{s,cat}$ is a submultiple of f_s, you'll need to interpolate the impulse response with the command `interp(x, r, n, cutoff)`, where `x` is the vector containing the original impulse response, `r` is the interpolation factor, `n=4`, and `cutoff` is the inverse of the interpolation factor.
   - Now apply the convolution reverb in the frequency domain: compute the FFT of both the chord progression and the cathedral interpolated impulse response with the same number of samples M, where M is next higher power of 2 of the sum of the lengths of the 2 sequences. Perform an element-wise multiplication of the 2 FFTs, compute the IFFT of the result and finally normalize it by its maximum absolute value.
   - Lastly, create an `audioplayer` object, play the reverbed progression and write it to a WAVE file.

- **Optional**: feel free to experiment and change the given parameters such lfo, the filter specifications or ADSR envelope to create different sounds.

Exercise 3 - Color space conversion and image filtering

1. Open the file `parrots.jpg` with `imread`. The image is RGB based and has size $M \times N \times 3$. Convert the image to double, normalize it by 255, and display it with the command `imshow(img, [])`.
2. Write a MATLAB function for converting an input image in RGB color model to an image in the YCbCr model. Name your own function `my_rgb2ycbcr_fast`. Use the YCbCr transform at slide 24 of `Lec11`. It should have the syntax: `YCbCrimg=my_rgb2ycbcr_fast(RGBimg)`. The function cannot contain any `for` loop. Measure the execution time with `tic` before the call and `toc` afterwards.
3. Make a second version of your function called `my_rgb2ycbcr_slow` with 2 nested `for` loops on the rows and columns of the image to work on each RGB pixel. Apply the matrix by column vector multiplication of slide 24 for each RGB pixel vector (use `squeeze` to rearrange each $1 \times 1 \times 3$ RGB pixel into a 3×1 vector). Measure again the execution time with `tic` and `toc` commands and report the difference w.r.t. to the first version.

Exercise 3

4. Create a 3×3 subplot in which you display the Y, Cb, and Cr components as grayscale images of the outputs of the functions `my_rgb2ycbcr_fast`, `my_rgb2ycbcr_slow`), and the MATLAB inbuilt function `rgb2ycbcr`.
5. Now work only with the Y component (greyscale) of the image. Create a Gaussian low-pass filter of size $Q \times Q$ to blur the image:

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{D^2(x,y)}{2\sigma^2}}$$

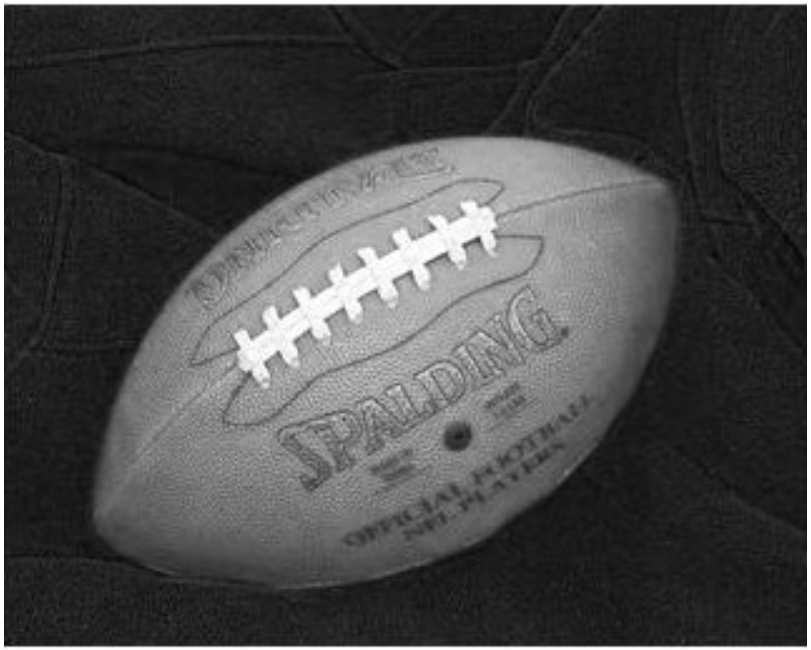
   - $Q = 51$ and $\sigma = 5$.
   - $D(x, y) = \sqrt{x^2 + y^2}$ is the the distance from point (x, y) to the center of the filter. You can compute it for each point by generating the meshgrid matrices $\mathbf{X}$ and $\mathbf{Y}$ ranging from $-Q/2$ to $Q/2$ with step 1.

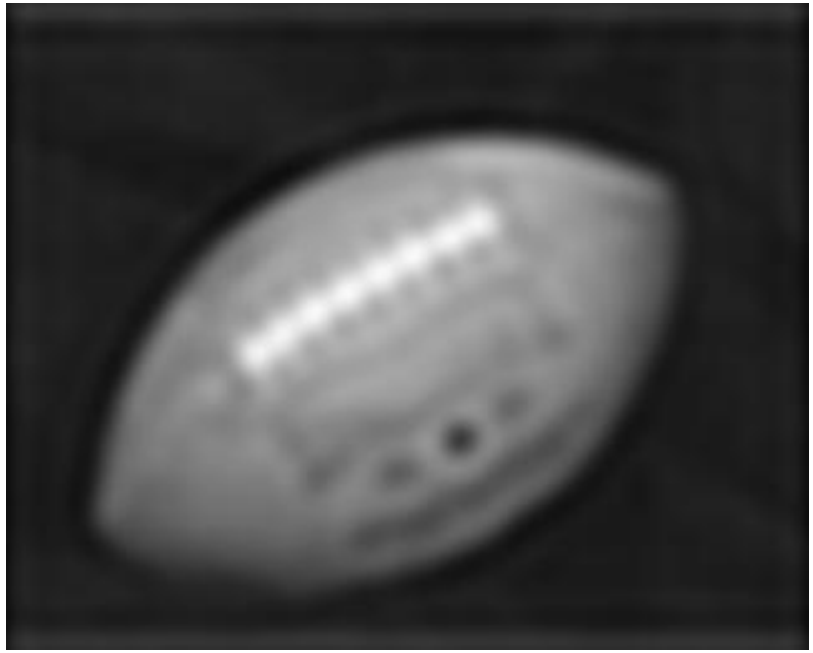
6. Compute the 2-D Fourier transform of the filter with the command `fft2` with 128×128 points. Plot the centered spectrum (`fftshift`) of the filter with `mesh`.

Exercise 3

7. Filter the image in the spatial domain by computing the 2-D convolution between the image and the filter $h(x, y)$ with `conv2` with option `'same'`. Show the blurred image with `imshow`.
8. Check that you obtain the same result by exploiting the separable property of the 2-D Gaussian filter and divide the process into two passes:
   - Use the 1-D Gaussian function $g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$, with x ranging from $-Q/2$ to $Q/2$, to filter the original image along all rows (use `conv` with option `'same'`).
   - Finally, apply the same filter $g(x)$ once more to all columns of the image after the first pass. Show the new blurred image with `imshow`.
9. Compute the centered spectrum of both the original and the filtered image and show the spectrum in logarithmic scale as $\log_{10}(1 + |\mathbf{A}|)$ with `mesh` (try also `imshow`).
10. Write the filtered image `I` with `imwrite` after converting it to `uint8(255*I)`.

Exercise 3 - Example





Exercise 3

11. Read the file `new_york.jpg` and convert it to grayscale by calling your function `my_rgb2ycbcr_fast` and keeping only the Y component. Show the image with `imshow`.
12. Given the image size $M \times N$, determine the 2-D Fourier transform size as $P \times P$, where P is a power of 2 at least as large as $2 \cdot \max(M, N)$.
13. Create a Butterworth high-pass filter in the frequency domain to sharpen the image. Start from the low-pass filter response:

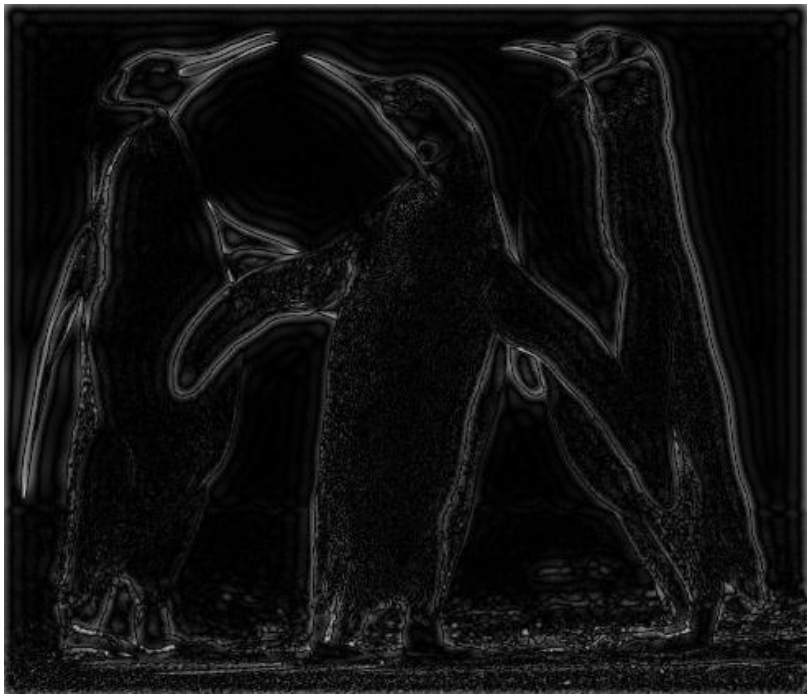
$$H_{lp}(u, v) = \frac{1}{1 + \left(\frac{D(u,v)}{D_0}\right)^{2n}}$$

    - $D_0 = P/16$ is the cut-off frequency between 0 and $P/2$, $n = 6$ is the order of the filter.
    - $D(u, v)$ has the same definition of $D(x, y)$ in point 5 (generate the meshgrid matrices $\mathbf{U}$ and $\mathbf{V}$ ranging from $-P/2$ to $P/2 - 1$).
14. Generate the high-pass filter $H(u, v) = 1 - H_{lp}(u, v)$.
15. Plot the spectrum of the filter with `mesh`.

Exercise 3

16. Compute the Fourier transform of the image with `fft2` and apply `fftshift` to centralize the DC component.
17. Filter the image in the frequency domain.
18. Convert the result in the spatial (or pixel) domain with `ifft2` and take the magnitude of it.
19. Crop the image to the first $M \times N$ pixels and display it with `imshow`.
20. Show the spectrum with `mesh` (try also `imshow`) of both the original and the filtered image in logarithmic scale as $\log_{10}(1 + |\mathbf{A}|)$.
21. Write the final image `I` with `imwrite` after converting it to `uint8(255*I)`.

Exercise 3 - Example



Report and Matlab scripts

- For each exercise include all requested plots and answers.
- Plots must contain labels for all axes, a title and a legend in case of multiple plots in one figure.
- The scripts must run correctly. If the script of an exercise doesn't work, the exercise will be considered failed.
- Deliver a separate Matlab file for each exercise (not a single Matlab file for the entire assignment). All requested functions are supposed to be local within their script.
- Both the report and the Matlab files must be uploaded on the portal in a zip file.
- Naming rule: `Group8_Assignment3_lastname#1_lastname#2.zip`.
- Send an email to daniel.riviello@polito.it when you upload it.

Report delivery and deadlines

- Deliver a single pdf report for Assignment 3 “Introduction to audio and image processing”.
- The report must include all requested plots, comments and answers for all exercises (1 to 3).

Deadlines to get extra points for Assignment 3

- **Tue. 20/05/2025** at 23:59 for 1 point.
- **Tue. 27/05/2025** at 23:59 for 0.5 points.