ELEN 4810 Homework 3

Due Wednesday, October 12. Please submit your code for the computational section online via Courseworks. You can submit your answers to the analytical question, as well as your code, via one of three means:

- Hardcopy submission at the beginning of class on Wednesday October 12.
- Electronic submission (in pdf form) on Courseworks.

Thanks.

Analytical Questions

Please complete problems 2.71, 4.27, 4.30 in Oppenheim and Schafer (3rd Edition). Justify your answers!

Computational Questions

1 Introduction

This week we will employ concepts introduced through lecture so far towards practical applications on audio and images. Although the techniques we will be implementing are rather rudimentary, they are illustrative of ideas that underlie solutions for more novel and interesting scenarios (think course project!).

You are again provided with excerpt.wav, as well as two images owl.jpg and owls.jpg. Furthermore, you are provided with three skeleton m-files detectcorr.m, allpass.m and reverb.m.

2 Problems

2.1 Finding Waldo

In the lecture, we briefly introduced the correlation operator. The purpose of this problem is to understand the relationship between correlation and convolution, and to do some very simple image processing. You have been provided with two images waldo.jpg and whereswaldo.jpg. Here we will try to match the template image waldo.jpg - a profile of Waldo - to the original reference image whereswaldo.jpg.
2.1.1 Correlation operations

**Notation.** In describing the correlation operator the following notation is used. For the sake of clarity signals are denoted with bold font.

- The *support* of a 2D (possibly complex) signal $x$ is the set of indices $\text{supp}(x) = \{ m, n : x[m, n] \neq 0 \}$.
- We say $x$ is *supported in* the index set $S$ if $\text{supp}(x) \subseteq S$, and let $x_S$ denote the signal $x_S[i,j] = \begin{cases} x[i,j], & \text{if } [i,j] \in S \\ 0 & \text{elsewhere.} \end{cases}$
- Denote the *inner product* $\langle x, y \rangle = \sum_{i,j} x[i,j] \cdot y[i,j]$ and the $\ell_2$-norm by $\| x \|^2 = \langle x, x \rangle = \sum_{i,j} |x[i,j]|^2$.
- The *shift* of $x$ by $(p,q)$ is denoted by $\left( D_{-1}^{p,q} x \right)[m,n] = x[m+p,n+q]$.
- Finally, for a nonnegative integer $M$ we define an index set $[M]_* = \{0, \ldots, M-1\}$. Note that we are using zero-based indexing which is different from MATLAB.

**Cross-correlation.** Let $a$ denote the template image supported in $S = [M]_* \times [N]_*$, and $b$ denote the larger reference image in which we would like to detect the template. The *cross-correlation* between $a$ and $b$ is a signal given by the inner product between signal shifts $r_{a,b}[p,q] = \langle D_{-1}^{p,q} b, a \rangle = \sum_{i,j} a^*[i,j] \cdot b[i+p,j+q]$. (1) $\sum_{i,j} a^*[i,j] \cdot b[i+p,j+q]$. (2)

Our signals of interest will be real-valued, so $\overline{a} = a$, and since $a[i,j] = 0$ for $(i,j) \notin S$, $r_{a,b}[p,q] = \sum_{i \in [M]_*, j \in [N]_*} a[i,j] \cdot b[i+p,j+q]$. (3)

This can be implemented in MATLAB directly using the function `xcorr2()` or by using the convolution function `conv2()`. Both approaches result in a matrix corresponding to the possible shifts $(p,q)$ such that $r_{a,b}[p,q] \neq 0$. This necessarily a finite window. Why?
The normalized convolution. The correlation can be useful in certain situations, but what we would really like for template matching is a function $f_{a,b}[p,q]$ which is maximized at the locations $(p,q)$ where the reference and template images match or “have the same pattern”.

Unfortunately, the cross-correlation easily fails this criteria. Assuming that the entries of $a$, $b$ – since they represent an image – are restricted to $[0,1]$. We can maximize $r_{a,b}$ trivially at $(p,q)$ by setting

$$D_{p,q}^{-1}b[i,j] = 1, \quad \forall (i,j) \in S,$$

regardless of the choice of $a$. For a more suitable objective function, recall that for two vectors $u$ and $v$, the normalized inner product $\langle \frac{u}{\|u\|_2}, \frac{v}{\|v\|_2} \rangle = \cos \angle (u,v)$. This is maximized iff $u = c \cdot v$ for some $c > 0$, which seems much closer to what we want.

Applying the normalized inner product between $(D_{p,q}^{-1}b)_S$ and $a$ gives rise to the normalized correlation

$$\chi_{a,b}[p,q] = \left\langle \frac{(D_{p,q}^{-1}b)_S}{\| (D_{p,q}^{-1}b)_S \|_2}, \frac{a}{\| a \|_2} \right\rangle \tag{4}$$

$$= \frac{\sum_{i \in [M], j \in [N]} a[i,j] \cdot b[i+p,j+q]}{\sqrt{\sum_{i,j} |a[i,j]|^2} \cdot \sqrt{\sum_{i' \in [M], j' \in [N]} |b[i'+p,j'+q]|^2}}. \tag{5}$$

Notice that the inner product is taken with $b$ restricted to the appropriate window. The normalized correlation be implemented by normalizing the output of $xcorr2()$ or $conv2()$. In particular, notice that the term

$$\sum_{i' \in [M], j' \in [N]} |b[i'+p,j'+q]|^2$$

can also be written as a cross-correlation. Hint: Take $r_{1_S,b}^2$, where $1_S[i,j]$ is equal to 1 for $(i,j) \in S$ and zero elsewhere.

2.1.2 Specficiation (3 pts)

Please implement the function in the skeleton file `detectcorr.m`, which attempts to find Waldo in the input image. Its usage is as follows,

$$[Ccorr, Cconv] = detectcorr(template_name, reference_name),$$

where `template_name` and `reference_image` should be the names of the template and reference images. The function first creates a `double` array representation of both images by applying `imread()` and `double()`, then using `rgb2gray()` to convert the images to grayscale.

Next, the function should compute the normalized correlation $Ccorr$ by normalizing the output of $corr()$ between the two images. Repeat this process to produce $Cconv$ by implementing the normalized correlation via `convolution` using the function $conv2()$. Note $Ccorr$ should take on the same values as $Cconv$.

Finally, the function should find the best match for the template by each method by selecting the pixel locations maximizing $Ccorr$ and $Cconv$ - do this using `max()` and `ind2sub()`. The function should produce a side-by-side subplot of the two detection results, and each subplot should contain a bounding box around the matching template region on the reference image using `rectangle()` – these technical details are written for you in the skeleton code.
Figure 1 shows an example of a correct detection using normalized correlation maximization.

![Template image](image1.png) ![Template detected in reference image](image2.png)

**Figure 1:** Template detection. Left: template image. Right: reference image, with detected template bounding box.

### 2.1.3 Observations

- Enter the following commands:
  
  - `Ccorr = detectcorr('waldo.jpg', 'whereswaldo.jpg');`
  - `imagesc(Ccorr); colorbar;`

  Find the point achieving the maximum. Is it easily distinguishable in the correlation image? What are some possible explanations for this? Potential consequences in practical applications?

- Try adding different levels of random noise to the reference using `sqrt(noisepr)**randn()`.
  Does `detectcorr()` still work?

- What would happen if Waldo is rotated in either the template or reference image?

- How can you make the detector more robust against the aforementioned issues?

### 2.2 System as Difference Equations: All-Pass Filter

In the second problem, we design a digital all-pass filter. Please write a function for the following system in the skeleton file `allpass.m`.

![All-pass filter diagram](image3.png)

**Figure 2:** All-pass filter.
The block diagram in Figure 2 has a number of elements. The arrows determine the signal inputs and outputs, whereas nodes split but do not modify the incoming signal. The triangle blocks indicate a multiplication operation, and the plus block indicates that the output is the sum of all inputs. As an example, the top branch of the block diagram indicates multiplication of $x[n]$ by factor $g$, and the output is added to another signal. Finally, square blocks indicate arbitrary systems, so in this case the $D_k$ block indicates delay of the input signal by $k$ samples.

### 2.2.1 Specification (4pts)

The function `allpass()` as specified in the skeleton file should have the following usage:

\[
y = \text{allpass}(x, k, g, \text{ploton}),
\]

where

- $x$ is a column array of length $N$: assume w.l.o.g that $x$ represents the input sequence $x[n]$ from $n \in [N], \equiv \{0, \ldots, N-1\}$;
- $k$ is the delay parameter in samples, and
- $g$ is the gain operator. Please convince yourself that the system is stable if $|g| < 1$, and unstable if $|g| > 1$.
- If `ploton` = `true`, then the function should create stem plots of the input $x$ and output $y$ in side-by-side subplots.
- $y$ is the output signal corresponding to the system specified by Figure 2: see below.

**Length of $y$.** For a stable choice of $g$, we would ideally like to obtain from the function a column array $y$ which represents $y[n]$ over the entire range where $y$ may be nonzero. Unfortunately, this is not going to be practical, as the system described by Figure 2 is IIR - please verify this. Nonetheless, if $g$ is chosen so that the filter is stable, we should expect the output to decay geometrically for $n > n_0 = N + k - 1$ (why?).

For this problem, we will set the length of $y$ so that the $y$ gets truncated after its magnitude has decayed to 1% of $|y[n_0]|$. A simple calculation shows that the samples $\hat{n}$ needed for this to happen after $n_0$, for $|g| < 1$, is

\[
|g|^\hat{n} \leq 10^{-2} \quad \Rightarrow \quad \hat{n} \geq -2 \cdot (\log_{10}|g|)^{-1}
\]

\[
\Rightarrow \quad n \geq n_0 - 2 \cdot (\log_{10}|g|)^{-1}
\]

On the other hand, if $|g| \geq 1$, then the system should not be expected to decay. In this case we simply truncate $y$ for $n > n_0 + 10k$. In the skeleton code, we take care of the length of $y$ for you in the same lines as the discussion above.

**Implementation.** When writing the `allpass()` function, note that a difference equation implementation will require using a for-loop. To figure out the update rule, derive the difference equation from Figure 2. We will assume $x[n] = 0$ for $n \notin [N]$, and also assume the initial condition $y[n] = 0, \forall n < 0$. You should convince yourself that the resulting system is causal, and thus the correct output should also be indexed starting from 0, i.e. $y(1) = y[0]$. 
2.2.2 Observations

- Enter the following commands:
  \[ N = 1000; \ n = 0:N-1; \ k = 50; \]

For any \(|g| < 1\), compare the input sequence \(x = \cos(2\pi f n/N)\) with

- \(y = \text{allpass}(x,k,g,1)\);

or a bunch of choices for \(f\), somewhat smaller than \(N\). Why is this system called an all-pass filter?

- How does the output change if you set a different initial condition, say \(y[-1] = 1\)? Is the resulting system LTI? Causal?

2.3 Synthetic Reverberation

In the last question we return to audio signals. We will use an all-pass filter to apply a reverberation operator to music signals.

Functionality (3 pts). Please implement the following system.

![Figure 3: Synthetic reverb system.](image_url)

Here the square blocks are all-pass filters (as implemented in the previous question). Please use the skeleton file `reverb.m`. The usage of the function is as follows,

\[ y = \text{reverb}(\text{fname}, \text{d}, \text{g}, \text{soundon}), \]

where `fname` is the name of the audio file to be read, e.g. `excerpt.wav`; use this to produce an array \(x\) representing the stereo sequence \(x[n]\). The variables \(d\) and \(g\) are arrays in the form \(d = [d_1 \ d_2 \ \cdots \ d_M]\) and \(g = [g_1 \ g_2 \ \cdots \ g_M]\) such that \(M = \text{numel}(d)\), whereas \(d(i)\) and \(g(i)\) are the millisecond delay (as opposed to sample delay \(k_i\), which you will have to compute) and
gain parameters for the $i$-th all-pass filter. The output $y$ is as specified in Figure 2. If soundon = true, then reverb() will also play the processed sequence $y$.

As suggested in the skeleton file reverb.m, you are encouraged to write an iterative implementation of the reverberation system. To do this, update the intermediate signals $y_i$ and $w_i$ for each $i = 1, \ldots, M$ using $y_{i-1}$ and $w_{i-1}$.

Note that your array length will increase at each iteration from extension by the all-pass filter, so you may also want to keep as little intermediate variables as possible to avoid running out of memory! The command clearvars may be helpful here.

Finally, use the settings $d = [15 \ 40 \ 100]$ (in ms) and $g = [0.05 \ 0.008 \ 0.002]$. Try processing the audio signal excerpt.wav with these settings.

Observations.

- Try using different combinations of $d$ and $g$ apply them to different audio tracks!

3 Submission

Submission Instructions. Place the completed functions detectcorr.m, allpass.m and reverb.m into a single .zip file with your UNI and homework number as the filename, e.g. ‘yl3027_hw3.zip’. Please upload and submit to Courseworks before the beginning of class on Wednesday, Oct. 12th.

If you are submitting the analytical questions electronically, please name it in the form of ‘yl3027_hw3_writeup.pdf’ and submit it separately from the .zip file!

Do not submit these: You do not need to submit answers to the observation questions, and please do not submit any audio or image files, or any plots; we are evaluating your code.