Professor Fearing EECS120/Python MiniLab 3 v1.1 Composer: Ming Jin

Due at 6 pm, Fri. 11/4 on BCourses

Up to 2 people may turn in a single iPython (Jupyter) notebook. Upload .ipynb and answers.

This exercise is worth 3% grade. (Python exercises will be 9% of grade).

Instructions: In this Python exercise, you will magnify tiny motions and changes in videos to reveal hidden yet useful informations – by making use of tools you have learned in EE120, including DFT, convolution, and bandpass filter. To understand the working principles of the video manification, download the iPython notebooks and the accompany packages from the class website.

Note: please install `imageio` (`https://imageio.github.io/`):

```
$ pip install imageio
```

The python package can be downloaded at:
`http://www-inst.eecs.berkeley.edu/~ee120/fa16/homework.html`

PART I: Fundamentals - 1D example

The idea of **motion magnification** is quite simple and elegant: the subtle temporal changes in the video are extracted using a bandpass filter, amplified in magnitude, and added back to the original signal. In this section, you will be working with a simple 1D image "band" (rather than a 2D image), and explore the working principles and conditions.

Let's denote $I(x,t)$ as the gray-scale intensity of the image band, where $x \in [0,2]$ is the *spatial location*, and $t \geq 0$ is the *time variable*. Assume that the band undergoes a subtle (probably unobservable) translational motion, expressed as the observed video intensity function with respect to a displacement function $\delta(t)$, such that $I(x,t) = f(x+\delta(t),t)$. Our goal is to **exaggerate the displacement function and add it back to** $I(x;t)$, i.e., to synthesize the signal $\hat{I}(x,t) = f(x+(1+\alpha)\delta(t),t)$, where $\alpha > 0$ is the amplification factor.

Table 1: Notations used throughout the questions 1 – 4.

	Definition
$\delta(t)$	Oscillation motion signal in time, $0.005\cos(2\pi f_0 t)$, $f_0 = 4Hz$
$I(x,t)$	The **original** signal w/ *unmagnified* motion, $\cos\big(2\pi(x-t+\delta(t))\big)$
$\hat{I}(x,t)$	The signal w/ *magnified* motion, $\cos\big(2\pi(x-t+(1+\alpha)\delta(t))\big)$
$f(x,t)$	The signal w/o the motion of interests, $\cos\big(2\pi(x-t)\big)$
$\tilde{I}(x,t)$	The **estimated** signal w/ *magnified* motion

Using the first-order Taylor series expansion, we can write the image at time t, $f(x+\delta(t),t)$ as: $I(x,t) = f(x+\delta(t),t) \approx f(x,t) + \delta(t)\frac{\partial f(x,t)}{\partial x}$. Let $B(x,t)$ be the result of applying a temporal bandpass filter at every position x that extracts the subtle temporal changes, i.e., $B(x,t) = \delta(t)\frac{\partial f(x,t)}{\partial x}$. Now we can amplify the bandpass signal by α an add it back to $I(x,t)$: $\tilde{I}(x,t) = I(x,t) + \alpha B(x,t) \approx f(x+(1+\alpha)\delta(t),t)$. If the first order Taylor expansion holds for the amplified signal, then we get back the synthesis signal with amplified larger perturbation!

Throughout this section, we will work with the *intensity function*, given by:

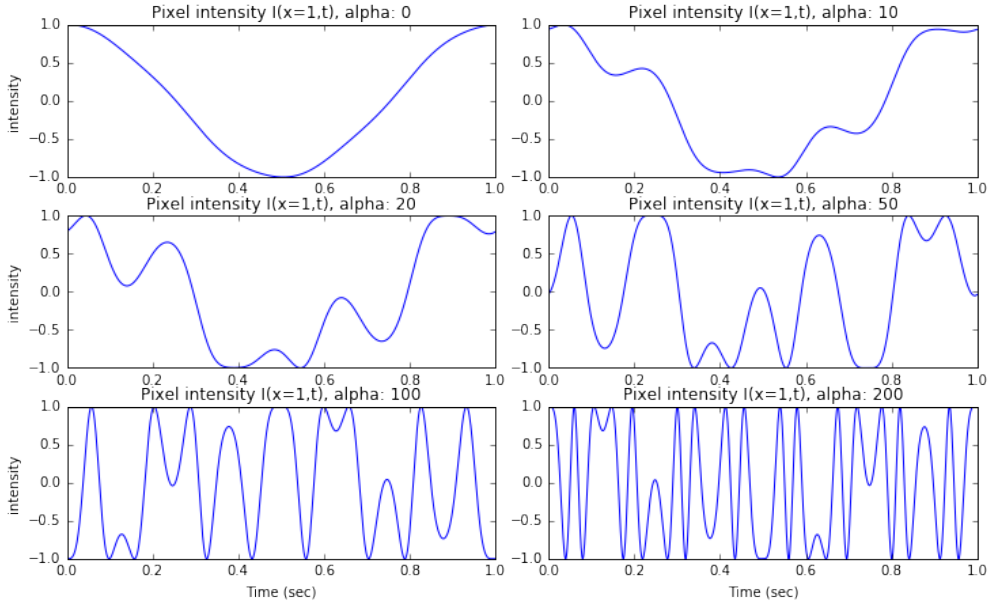
$$\hat{I}(x,t) = \cos\big(2\pi(x-t+(1+\alpha)\delta(t))\big)$$

(updates 10/31, the intensity function is $\hat{I}(x,t)$ not $I(x,t)$ in the definition, since it is the signal with magnified motion.) where $1+\alpha$ (with $\alpha \geq 0$) is the amplitude magnification factor, and $\delta(t) = 0.005\cos(2\pi f_0 t)$

is the "subtle" motion function, with $f_0 = 4Hz$ being the frequency of the oscillation motion.

1. (10 pts) Write out the intensity function $\hat{I}(x,t)$ for pixel at $x = 1$. Attach the plots for amplification factors $\alpha = 0, 10, 20, 50, 100$ (use the iPython notebook to generate these plots). What do you observe?.

The intensity function $\hat{I}(1,t) = \cos\big(2\pi(1-t+(1+\alpha)\delta(t))\big)$ at $x = 1$. The plots show that as we increase the magnification factor α, the signal becomes more erratic and deviates significantly from a cosine wave. The transition happens at around alpha is between 10 to 20. Also we see that the signal without motion magnified looks very like a cosine wave, where the tiny motion is hidden and not easily observable.



2. (15 pts) Write out the Fourier Transform of $\hat{I}(1,t)$ (Updated. NOT $I(1,t)$) from Q1 for the case when α is very small. (Hint: you can use the Taylor series expansion.) What are the frequency components related to the displacement function $\delta(t)$?

Using the first-order Taylor series expansion, and define $f(x,t) = \cos\big(2\pi(x-t)\big)$, we can see that

$$\begin{aligned}\hat{I}(1,t) &= f(1+(1+\alpha)\delta(t),t)\\ &= \cos\big(2\pi(1-t+(1+\alpha)\delta(t))\big)\\ &\approx f(1,t) + (1+\alpha)\delta(t)\frac{\partial f(x,t)}{\partial x}|_{x=1}\end{aligned}$$

Since $\frac{\partial f(x,t)}{\partial x} = -2\pi\sin\big(2\pi(x-t)\big)$, plugging in $x = 1$, we have:

$$\begin{aligned}\hat{I}(1,t) &\approx \cos\big(2\pi(1-t)\big) - 2\pi(1+\alpha)\delta(t)\sin\big(2\pi(1-t)\big)\\ &= \cos\big(2\pi(1-t)\big) - \frac{\pi(1+\alpha)}{100}\cos(2\pi f_0 t)\sin\big(2\pi(1-t)\big)\end{aligned}$$

Now taking the Fourier transform, $\mathcal{F}\{\cos\big(2\pi(1-t)\big)\} = \pi\delta(\omega-2\pi) + \pi\delta(\omega+2\pi)$, and also:

$$\begin{aligned}&\mathcal{F}\{\cos(2\pi f_0 t)\sin\big(2\pi(1-t)\big)\} = \frac{1}{2\pi}\mathcal{F}\{\cos(2\pi f_0 t)\big)\} \star \mathcal{F}\{\sin\big(2\pi(1-t)\big)\}\\ &= \frac{\pi}{2}\big(\delta(\omega+2\pi f_0-2\pi) + \delta(\omega+2\pi f_0+2\pi) + \delta(\omega-2\pi f_0-2\pi) + \delta(\omega-2\pi f_0+2\pi)\big)\end{aligned}$$

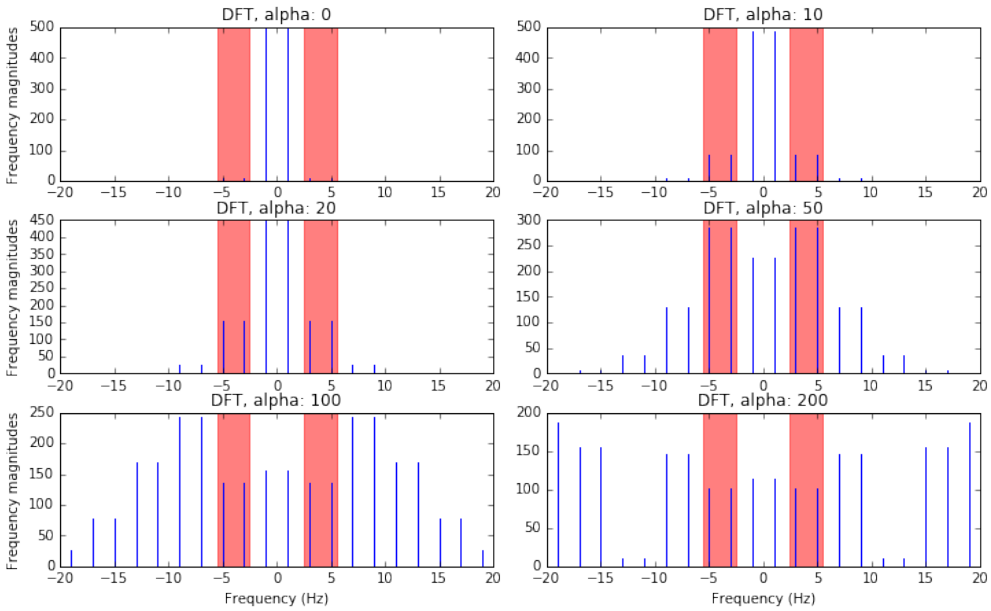
Therefore, the Fourier transform of $\hat{I}(1,t)$ is approximately:

$$\pi\delta(\omega-2\pi)+\pi\delta(\omega+2\pi)-\frac{\pi^2(1+\alpha)}{200}\Big(\delta(\omega+2\pi f_0-2\pi)+\delta(\omega+2\pi f_0+2\pi)+\delta(\omega-2\pi f_0-2\pi)+\delta(\omega-2\pi f_0+2\pi)\Big)$$

. The frequency components related to the displacement function $\delta(t)$ are centered at $2\pi f_0 \pm 2\pi$ and $-2\pi f_0 \pm 2\pi$, with magnitude $\frac{\pi^2(1+\alpha)}{200}$. Note that since $f_0 = 4Hz$, we should expect to find them at $\pm 3, \pm 5Hz$.

3. (15 pts) Does your answer in Q2 apply for large α? Test your argument with the following plots, and attach the results for $\alpha = 0, 10, 20, 50, 100$ (iPython notebook).

No it does not apply, since for large α, the first-order Taylor expansion is not accurate, and the approximation does not hold. This is verified in the plots below, where we see that the approximation holds roughly when $\alpha < 20$, however as α increases, there are more higher frequency components outside of our region of interests.

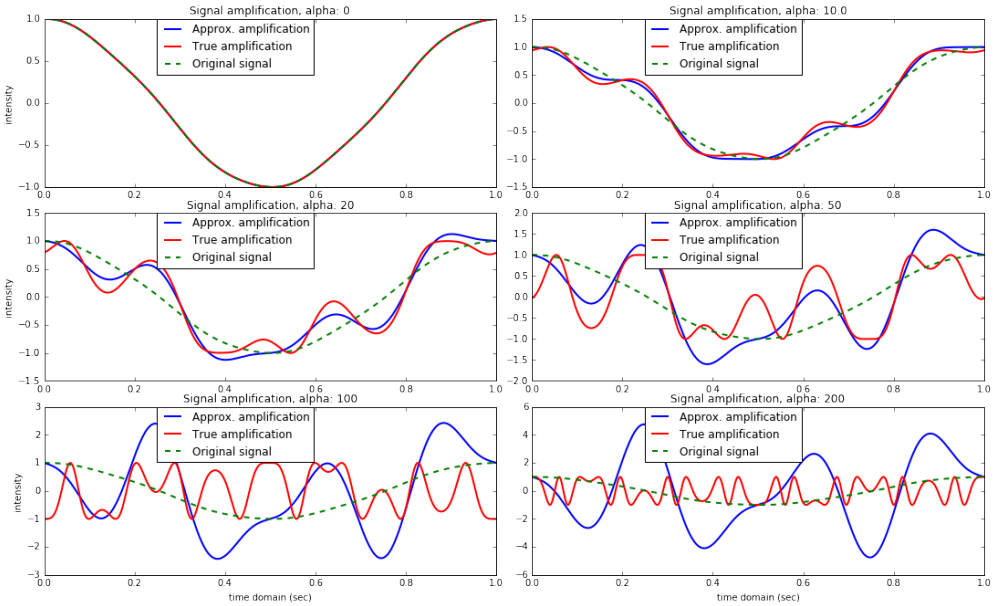


4. (10 pts) What is our amplification signal, $\tilde{I}(x,t)$ (if you use $B(x,t)$, please clearly indicate what it represents)? What do you observe from the above amplification results? Attach the plots for different amplification factors $\alpha = 10, 20, 50, 100$, and comment on the reliability of amplification as α increases.

The amplification signal is given by:

$$\tilde{I}(x,t) = I(x,t) + \alpha B(x,t)$$

where for small α, $B(x,t)$ is approximately $\delta(t)\frac{\partial f(x,t)}{\partial x}$; in general, it is the frequency components of the origianl signal after performing the bandpass filter. As is shown in the plots below, the approximated amplification $\tilde{I}(1,t)$ is only correct when $\alpha < 20$. As α increases, it becomes very different from the true amplified signal. This indicates a region of reliability for the manification. In this case, $\alpha = 20$, i.e., 20 times manification can produce the most manified and yet accurate result.



PART II: Facial Pulse Detection

The above idea can be easily extended to 2D image. A video is a sequence of **frames**, played back at a rate determined by frames per second (fps), typically 30Hz, and each frame is simply an image coded in three color channels, usually Red, Green, and Blue. The motion magnification, when applied to each pixel in its respective color channel, is effectively magnifying **both the motion changes and the color changes**.

Our first step is to spatially process the frames to increase the temporal signal-to-noise ratio, a procedure known as "spatial pooling". Specifically, we will make use of the 2D isotropic (i.e., circularly symmetric) Gaussian filter:

$$G(x,y) = \frac{1}{\sum_{|x'|\leq M, |y'|\leq M} e^{-\frac{x'^2+y'^2}{2\sigma^2}}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where $x, y = 0, \pm 1, \pm 2, \cdots$ represent the centered pixel locations, M is the size of the filter, and the normalization factor is given as $\frac{1}{\sum_{|x'|\leq M, |y'|\leq M} e^{-\frac{x'^2+y'^2}{2\sigma^2}}}$.

The Gaussian kernel is then convolved with the frame. For any pixel at location x_0, y_0, the new pixel value is given by: $\sum_{|x|\leq M, |y|\leq M} G(x,y) I(x_0 - x, y_0 - y)$, where x, x_0, y, y_0 are integers. Note the similarity to the 1D convolution.

5. (15 pts) Implement the Gaussian filter of any odd size in the notebook. Write out the Gaussian filter for size 5 by 5 and $\sigma = 1$. What does the Gaussian filter do in effect?

The Gaussian filter for size 5 by 5 is gien by:

$$\begin{bmatrix} 0.00296902 & 0.01330621 & 0.02193823 & 0.01330621 & 0.00296902 \\ 0.01330621 & 0.0596343 & 0.09832033 & 0.0596343 & 0.01330621 \\ 0.02193823 & 0.09832033 & 0.16210282 & 0.09832033 & 0.02193823 \\ 0.01330621 & 0.0596343 & 0.09832033 & 0.0596343 & 0.01330621 \\ 0.00296902 & 0.01330621 & 0.02193823 & 0.01330621 & 0.00296902 \end{bmatrix}$$

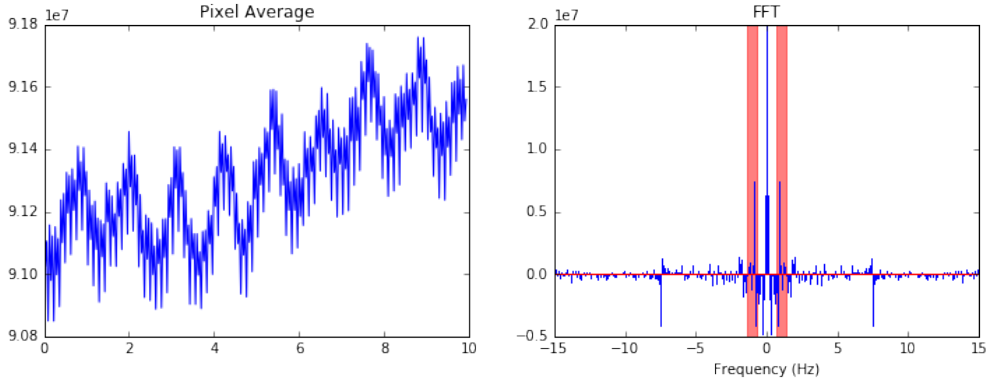
The Gaussian filter is like a low pass filter, which blurs the image. This is useful to reduce noise and improve signal to noise ratio.

6 . (15 pts) Refer to the iPython notebook: what do `gausPyrDown` and `gausPyrUp` do? Why we need this step? (Hint: check the dimensions of the video data, and how many times we need to perform temporal filtering)

`gausPyrDown` is downsampling the image after bluring them. `gausPyrUp` is upsampling the image. By downsampling the image, we can have a faster computation, since we apply the temporal low pass filter at every pixel. The upsampling is able to (approximately) reconstruct the image of the same resolutions (i.e., number of pixels).

7. (10 pts) Specify the cutoff frequencies, `freq_min,freq_max`, of the bandpass filter, and attach the plot given by `show_frequencies(orig_vid, fps,freq_min,freq_max)` in the notebook. Why do you choose this range of frequency?

We shoose the frequency between $\frac{40}{60}$ and $\frac{80}{60}$ because it seems to be the range of heart rate for the normal person. This is also informed by the pixel average plot, where we see a periodic signal of roughly 60 beats per minutes, i.e., 1Hz.



8. (10 pts) Refer to the notebook: print out the plots of both the original and magnified videos at time 4.3, 4.9, 5.5, 6.1 seconds. (Note that it might take several minutes to process, so be patient)