ELEC/COMP 447/546
Assignment 2
Due: Feb 14, 2023, Midnight

## 1.0 Hybrid Images (10 points)

Recall the *hybrid image* of Albert Einstein and Marilyn Monroe introduced in [1] and reproduced below in Fig. 1. Due to the way your brain processes spatial frequencies, you will see the identity of the image change if you squint or move farther/closer to the image. In this problem, you will create your own hybrid image.

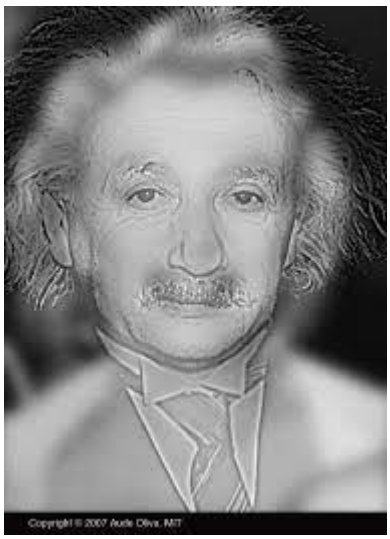


**Fig. 1: The hybrid image "Marilyn Einstein."** This was produced by combining the low-frequency components of an image of Marilyn Monroe, and the high-frequency components of an image of Albert Einstein [1].

### 1.1 Gaussian kernel

Implement function `gaussian2D(sigma, kernel_size)` that returns a 2D gaussian blur kernel, with input argument `sigma` specifying a tuple of x,y scales (standard deviations), and `kernel_size` specifying a tuple of x,y dimensions of the kernel. The kernel should be large enough to include 3 standard deviations per dimension.

### 1.2 Create Hybrid Images

Choose two images (`A` and `B`) of your choice that you will blend with one another and convert them to grayscale (you can use opencv.cvtColor). These images can be of faces, or any other objects. Try to make the objects in the two images occupy roughly the same region in the image (if they don't you can use the `imwarp` function you wrote in Homework 1 to manually align them!).

Construct a hybrid image `C` from `A` (to be seen close-up) and `B` (to be seen far away) as follows: `C = blur(B) + (A-blur(A))`, where `blur` is a function that lowpass filters the image (use the Gaussian kernel you coded in 1.1 for this). Try different values of `sigma` for the Gaussian kernel. How does the amount of blurring affect your perception of the results? In your report, please show your input images labeled clearly as `A` and `B`, and attach the result `C` for a value of `sigma` which you feel demonstrates the illusion the best, at both the original size and a downsampled size. As a sanity check, you should be able to see the identity change when looking at the original and downsampled versions.

**1.3 Fourier Spectra**

For the `sigma` value you chose in 1.2, show images of the Fourier spectra magnitudes of images `A`, `B`, `blur(B)`, `A-blur(A)`, and `C`. You can get the magnitude of the Fourier spectrum coefficients of an image 'x' by running:

```
X = numpy.abs(numpy.fftshift(numpy.fft2(x)))
```

By default, `numpy.fft2` will place the zero frequency (DC component) of the spectrum at the top left of the image, and so `numpy.fftshift` is used here to place the zero frequency at the center of the image. When displaying the Fourier spectrum with `matplotlib.pyplot.imshow`, the image will likely look black. This is because the DC component typically has a much higher magnitude than all other frequencies, such that after rescaling all values to lie in [0,1], most of the image is close to 0. To overcome this, display the *logarithm* of the values instead.

## 2.0 Laplacian Blending (15 points)

The Laplacian pyramid is a useful tool for many computer vision and image processing applications. One such application is blending sections of different images together, as shown in Fig. 2. In this problem, you will write code that constructs a Laplacian pyramid, and use it to blend two images of your choice together.

**Fig. 2: Laplacian Blending Example.** The halves of two images, an orange and an apple (left) are blended together using Laplacian blending (right).

### 2.1 Gaussian Pyramid

Write a function `gausspyr(I, n_levels, sigma)` that returns a Gaussian pyramid for image `I` with number of levels `n_levels` and Gaussian kernel scale `sigma`. The function should return a list of images, with element `i` corresponding to level `i` of the pyramid. Note that level `0` should correspond to the original image `I`, and level `n_levels - 1` should correspond to the coarsest (lowest frequency) image.

### 2.2 Laplacian Pyramid

Write a function `lappyr(I, n_levels, sigma)` that returns a Laplacian pyramid for image `I` with number of levels `n_levels` and Gaussian kernel `sigma`. The function should return a list of images, with element `i` corresponding to level `i` of the pyramid. Note that level `0` corresponds to the *details* of the original image `I`, and level `n_levels - 1` corresponds to the low-frequency residual image.

### 2.3 Image Blending

Choose two images A and B depicting different objects and resize them to the same shape. You may want to use your `imwarp` function from Homework 1 to align the scales/orientations of the objects appropriately (as was done in the example in Fig. 2) so that the resulting blend will be most convincing. Create a binary mask image `mask` which will have 1s in its left half, and 0s in its right half (called a 'step' function). Perform blending with the following operations:

1. Build Laplacian pyramids for `A` and `B`.
2. Build a Gaussian pyramid for `mask`.

3. Build a blended Laplacian pyramid for output image `C` using pyramids of `A`, `B`, and `mask`, where each level $l_k^C$ is defined by the equation $l_k^C = l_k^A * m_k + l_k^B * (1 - m_k)$.
4. Invert the combined Laplacian pyramid back into an output image `C`.

Show the following in your report: (1) Images from all levels of the Laplacian pyramids for `A` and `B`, (2) images from all levels of the Gaussian pyramid for `mask`, and (3) your final blended image `C`.

**2.4 (ELEC/COMP 546 Only) Blending two images with a mask other than a step**

Laplacian blending is not restricted to only combining halves of two images using a step mask. You can set the mask to any arbitrary function and merge images, as shown in this example. Demonstrate a Laplacian blend of two new images using a mask other than step.

## 3.0 Pulse Estimation from Video (5 points)

You are convinced that your friend Alice is a robot. You don't have much evidence to prove this because she is quite a convincing human during conversations, except for the fact that she does get very angry if water touches her. One day, you hit upon a plan to figure out this mystery once and for all. You know that a human has a heart which pumps blood, and a robot does not. Furthermore, you read a paper [2] showing that one can estimate heart rate from a video of a human face using very simple computer vision techniques. So the next day, you convince Alice to take this video of herself, linked here. You will now need to implement a simple pulse estimation algorithm and run it on the video. Follow these steps:

**3.1 Read video into notebook and define regions of interest**

Upload the video into your Colab environment. Note that it may take several minutes for the upload to complete due to the size of the file. You can then read the video frames into a numpy array using the `read_video_into_numpy` function provided here.

Using the first video frame, manually define rectangles (row and column boundaries) that capture 1) one of the cheeks and 2) the forehead.

**3.3 Compute signals**

Now compute the average Green value of pixels for all frames for each facial region (cheek, forehead). This gives a 1D signal in time called the Photoplethysmogram (PPG) for each region.

**3.4 Bandpass filter**

It is often useful to filter a signal to a particular band of frequencies of interest ('pass band') if we know that other frequencies don't matter. In this application, we know that a normal resting heart rate for an adult ranges between 60-100 beats per minute (1-1.7 Hz). Apply the `bandpass_filter` function to your signals provided here. You can set low_cutoff = 0.8, high_cutoff = 3, fs = 30, order = 1. Plot the filtered signals.

### 3.4 Plot Fourier spectra

Plot the Fourier magnitudes of these two signals using the DFT, where the x-axis is frequency (in Hertz) and y-axis is amplitude. DFT coefficients are ordered in terms of integer indices, so you will have to convert the indices into Hertz. For each index n = [-N/2, N/2], the corresponding frequency is Fs * n / N, where N is the length of your signal and Fs is the sampling rate of the signal (30 Hz in this case). You can use numpy.fftfreq to do this conversion for you.

### 3.5 Estimate Alice's average pulse rate

A normal resting heart rate for adults ranges between 60-100 beats per minute. What rate does the highest peak in Alice's Fourier spectrum correspond to? Which facial region provides the cleanest spectrum (the one which has the clearest single peak and low energy elsewhere)? Is Alice likely a human or not?

### 3.6 (ELEC/COMP 546 Only) Find your own pulse

Take a 15-20 second video of yourself using a smartphone, webcam, or personal camera. Your face should be as still as possible, and don't change facial expressions. Do a similar analysis above as you did with Alice's video. Show some frames from your video. Was it easier/harder to estimate heart rate compared to the sample video we provided? What was challenging about it?

### References

[1] *Oliva, Aude, Antonio Torralba, and Philippe G. Schyns. "Hybrid images." ACM Transactions on Graphics (TOG) 25.3 (2006): 527-532.*

[2] Poh, Ming-Zher, Daniel J. McDuff, and Rosalind W. Picard. "Non-contact, automated cardiac pulse measurements using video imaging and blind source separation." *Optics express* 18.10 (2010): 10762-10774.

## Submission Instructions

All code must be written using Google Colab (see course website). Every student must submit a zip file for this assignment in Canvas with 2 items:

1. An organized report submitted as a PDF document. The report should contain all image results (intermediate and final), and answer any questions asked in this document. It

should also contain any issues (problems encountered, surprises) you may have found as you solved the problems. The heading of the PDF file should contain:
    a. Your name and Net ID.
    b. Names of anyone you collaborated with on this assignment.
    c. A link to your Colab notebook (remember to change permissions on your notebook to allow viewers).
2. A pdf copy of your Colab notebook.

## Collaboration Policy

I encourage collaboration both inside and outside class. You may talk to other students for general ideas and concepts, but you should write your own code, answer questions independently, and submit your own work.

## Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly. If you have any doubts regarding what is and is not plagiarism, talk to me.