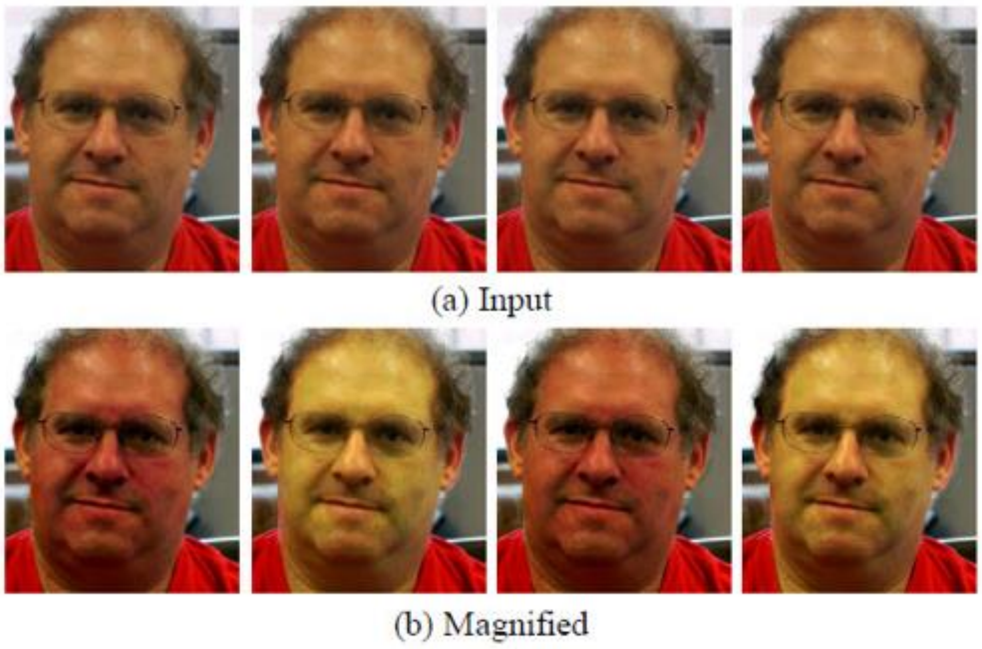
Homework Assignment 2

Due Wednesday, Oct. 3, at 9:00 am



Overview

This project explores Eulerian Video Magnification to reveal temporal image variations that tend to be difficult to visualize. This video briefly describes the technique, and shows a variety of visual effects attainable with it. For example, one can amplify blood flow and low-amplitude motion with Eulerian Magnification without needing to perform image segmentation or computing optical flow.

The primary goal of this assignment is to amplify temporal color variations in two videos (face and baby2), and amplify color or slow-amplitude motion in your own custom short video sequence that you record yourself. For this you will read the paper about Eulerian Video Magnification and implement the approach.

How can you amplify image variations that are hard to see with the naked eye? The insight is that some of these hard-to-see changes occur at particular temporal frequencies that we can augment using simple filters in the frequency domain! For example, to magnify pulse we can look at pixel variations with frequencies between 0.4 and 4Hz, which correspond to 24 to 240 beats per minute. Given the above insight, Eulerian video magnification is done as a sequence of the following steps:

(1) If your video has color, transform it to an appropriate color space.

(2) Create a Laplacian pyramid for each video frame.

(3) Band-pass filter the time series for each pixel, on all levels of the pyramid.

(4) Magnify bands of interest by some scale.

(5) Reverse the Laplacian pyramid and undo the color transform to obtain the final output.

The hard part of the process is to find the right parameters to get the desired magnification effect. For example, one can change the size of the Laplacian pyramid, multiply the time series corresponding to the value of a pixel by different scale factors at different levels of the pyramid, or attenuate the magnification when adding the augmented band-passed signals to the original ones. The choice of the band-pass filter (e.g., the range of frequencies it passes/rejects, its order, etc.) can also influence the obtained results.

Another challenge is to automatically identify an appropriate frequency bandwidth depending on the content of video. It would be useful to observe the histogram of frequency response and determine the meaningful periodic pattern of input video.

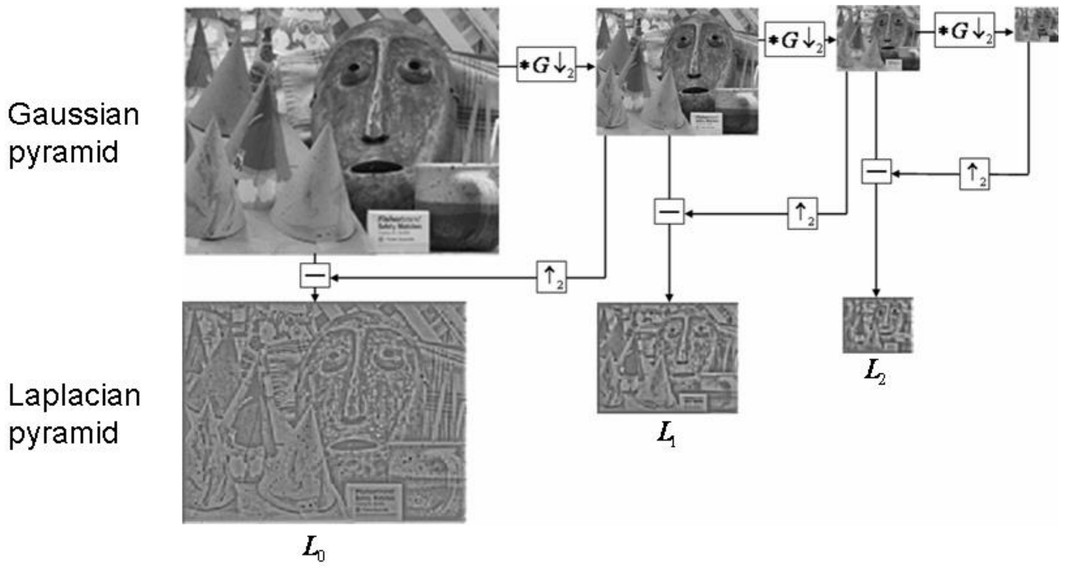
These are an open problem and you need to handle them as a part of the project, so you should start early!

INITIALS AND COLOR TRANSFORMATION (5 PTS)

Load the video file into Matlab, extract its frames, and convert them to double-precision in the range [0, 1]. (See help for VideoReader and GetFrames about reading video files in Matlab.) Then, convert each of the frames to the YIQ color space. The YIQ color space is particularly suggested for Eulerian magnification since it allows to easily amplify intensity and chromaticity independently of each other (we can use `rgb2ntsc` and `ntsc2rgb` to move between RGB and YIQ in MATLAB).

LAPLACIAN PYRAMID (20PTS)

The next step towards motion magnification is to construct a Laplacian pyramid *for every single frame in the video sequence.* The Laplacian pyramid was originally proposed by Burt and Adelson in their 1983 paper The Laplacian pyramid as a compact image code, where they suggested to sample the image with Laplacian operators of many scales. This pyramid is constructed by taking the difference between adjacent levels of a Gaussian pyramid, and approximates the second derivative of the image, highlighting regions of rapid intensity change.



Each level of the Laplacian pyramid will have different spatial frequency information, as shown in the picture above. Notice that we need to upsample one of the images when computing the difference between adjacent levels of a Gaussian pyramid, since one will have a size of $w \times h$, while the other will have *(w/2)x(h/2)* pixels. Since the last image in the Gaussian pyramid does not contain an adjacent image to perform the subtraction, then it just becomes the last level of the Laplacian pyramid.

Notice that by doing the inverse process of constructing a Laplacian pyramid we can reconstruct the original image. In other words, by upsampling and adding levels of the Laplacian pyramid we can generate the full-size picture. This reconstruction is necessary to augment videos using the Eulerian approach.

TEMPORAL FILTERING (30PTS)

We consider the time series corresponding to the value of a pixel on all spatial levels of the Laplacian pyramid. We convert this time series to the frequency domain using the Fast Fourier Transform (`fft` in MATLAB), and apply a band pass filter to this signal. The choice of the band-pass filter is crucial, and we recommend designing and visualizing the filter with `fdatool` and `fvtool` in MATLAB (see an example by MathWorks).

To make this process easier, in the `./src` directory of the homework ZIP archive, we provide you with a `butterworthBandpassFilter` function to generate a Butterworth band-pass filter of a particular order. This function generates a Butterworth band-pass filter of a particular order. It was generated with `fdatool`, and also uses the function `fdesign.bandpass`, which you can read more about in the Matlab documentation. Use of this filter is optional.

More details on the `fdesign.bandpass` parameters can be found here. Check the Eulerian Video Magnification paper for details on the parameters they used on the face and baby2 videos. You will have to find the right parameters for the other video that you capture yourself, and process.

In order to filter the time series of the pixels fast, we recommend you perform this operation in the frequency domain, since multiplication is faster than convolution. But be careful about `fft`'s output format when doing this! As explained in this tutorial, the DC component of `fftx = fft(x)`, for `x` a 1D signal, is the first element `fftx(1)` of the array. If `x` has an even number of samples, then the magnitude of the FFT will be symmetric, such that the first `(1+nfft/2)` points are unique, and the rest are symmetrically redundant. The element `fftx(1+nfft/2)` is the Nyquist frequency component of `x` in this case. If the number of samples of `x` is odd, however, the Nyquist frequency component is not evaluated, and the number of unique points is `(nfft+1)/2`.

Also, if you decide to use the `butterworthBandpassFilter` function, then you will need to get the frequency components of the filter for fast computation. This can be done by using MATLAB's `freqz` function, by passing the filter and the length of the output that you want (i.e., `fftHd = freqz(Hd,NumSamples)`). Again, be careful about how the frequency components are output by `freqz`.
After extracting the frequency band of interest, we need to amplify it and add the result back to the original signal.

Extracting the Frequency band of interest (30pts)

This is an open research problem. Try to observe a variety of video data in frequency domain and identify the frequency band of interest. You will want to focus on slow motions (e.g. human motion) to eliminate outliers. Analyzing the histogram of frequency responses can be useful to understand the context of your video.

Image reconstruction (20pts, which includes evaluation of your results)

After amplifying the signals, all that is left is to collapse the Laplacian pyramids into a single image per frame. Notice that we can attenuate the amplification to obtain different results, or we can low-pass filter the amplified signal to reduce effects on high frequency components of the images, such as borders. Two different blood flow amplification effects on the face.mp4 image sequence are presented below,

Extra Credit: Capture and motion-magnify your own video(s) (up to 30 points)

Use your own camera (or one that you borrowed) to capture your own video sequences. A phone camera is fine. Try to find scenes where there are interesting motion effects to visualize.

Then, apply Eulerian video magnification to the videos you captured. You will need to search for a whole new set of parameters to use when doing motion magnification on your own videos, in order for the final results to look good.

The total number of points you will get from this bonus question will depend on how visually compelling the motion effect you capture and magnify is.

Deliverables

For this project, and all other projects, you must do a project report in HTML. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Use both words and images to show us what you've done (describe in detail your algorithm parameterization for each of your results). Your presentation should be done in 10 min and:

- Include a brief description of the project, and explain how you constructed the Laplacian image pyramid. (if not different from others, skip!)
- Show your result for amplifying blood flow on the face and baby2 sequences, as well as your results for the additional sequence that you capture and process. Your results should mimic the results obtained by the authors on the face and baby2 sequences (they should amplify similar pulsation rates, though colorizing may be different).
- Explain any difficulties and possible reasons for bad results.
- Include any bells & whistles and explain what parameters you used.

Policy

Do not search the code on the web. Remember, using or even looking at outside code is not allowed!