MIT CSAIL

6.819/6.869 Advances in Computer Vision

Spring 2022

Problem Set 3

<table>
<tr><td>Posted: Thursday, February 24, 2022</td><td>Due: Tuesday 23:59, March 8, 2022</td></tr>
</table>

6.869 and 6.819 students are expected to finish all problems unless there is an additional instruction.

We provide a python notebook with the code to be completed. You can run it locally or in Colab (upload it to Google Drive and select 'open in colab') to avoid setting up your own environment. Once you finish, run the cells and download the notebook to be submitted.

Submission Instructions: Please submit a .zip file named your ⟨*kerberos*⟩.zip containing 1) report named report.pdf including your answers to all required questions with images/plots and videos showing your results and copy the relevant lines into your PDF writeup. 2) the python notebook provided, with the cells run and the relevant source code. If you include other source code files for a given exercise, please indicate it in the report.

Late Submission Policy: If your pset is submitted within 7 days (rounding up) of the original deadline, you will receive partial credit. Such submissions will be penalized by a multiplicative coefficient that linearly decreases from 1 to 0.5.

Problem 1 *Hybrid images (2 points)*

In this problem you will create hybrid images as described in [1]. You may find the tools in **numpy.fft.*; scipy.signal.convolve2d** useful.

Take two images, A and B, that you'll want to have blend from one to the other. Try to make the objects in the two images occupy more or less the same region. Construct a hybrid image from A (to be seen close-up) and B (to be seen far away) as follows:

$$\text{out} = \texttt{blur}(B) + (A\text{-}\texttt{blur}(A))$$

Where `blur` is a function that low-pass filters the image. You should write your own blur function. You can use a Gaussian filter or try other blur filters, such as the box filter. Which one works best? Try different sigmas for the Gaussian. How does the amount of blurring affect your perception of the results?

In your report, please specify the type of kernel you used and its parameters. In your report, please show your input images labelled clearly as A and B, and attach the result at both the original size and a downsampled size. As a sanity check, you should be able to see the illusion as described in the paper.

Problem 2 *De-hybridizing (3 points)*

Examine the image `einsteinandwho.jpg` included in the notebook. Using the method of your choice, remove the individual represented in the low spatial frequency range to create two images: one of Einstein, and one of the other person. Please intensity scale the images using the provided function **intensityscale** to make them easier to see. Include both images in the report. You may want to try different methods to achieve the best two images. For fun: can you guess who is in the low spatial frequency image?

Problem 3 *Motion Magnification*

In this problem we will investigate motion magnification in videos. Recall that position shifts in image space correspond to phase shifts in the frequency domain of the Fourier transform. This means that for two images, we can compare the Fourier transform of the two images to find the phase shift between the images. Amplifying the phase shift by a fixed factor in the Fourier transform frequency domain will amplify the position shift by the same factor in the image domain after we perform the inverse Fourier transform. We will use this idea to exaggerate the motions in videos.

(a) (1 point) For a purely horizontal offset of an impulse signal, magnifying the phase shift will result in a magnified horizontal offset after the inverse transform. Please fill in lines 6 and 9 in `Magnify Change`. You should find the phase shift between the two input images and magnify it by the specified `magnificationFactor`. When complete, the function `magnifyChange` should return an image showing what image 2 would look like with the magnified offset. Please run `Problem 3.a` and submit the generated plot in the report and include the file named `problem_3a.png` in the zip.

(b) (1 point) If there is motion in more than one direction between two images, we will see that naively magnifying the phase shift of the whole images will not work. In `Problem 3.b`, we have set up a vertical offset of an impulse signal as well as the horizontal one from part a.

(i) Fill in the code to change the `expected` matrix to show what the expected output should be. Please run `Problem 3.b` and attach the generated plot in the report and include the file named `problem_3b.png` in the zip.

(ii) Based on the output and your expected output, what are the key differences? Please explain the cause for the impulse signal in the bottom left of the magnified output.

(c) (1 point) One strategy we can use if there are multiple motions between two images is to do a localized Fourier transform by independently magnifying the offsets on small windows of the images and aggregating the results across the windows. When we restrict our window of consideration, it is more likely for everything in the window to be moving the same way. We will use Gaussian filters to mask small windows of the image and perform magnification on each window independently. In `Problem 3.c`, please fill in the Gaussian filter in line 27 and the appropriately windowed input images in line 28. Since we are working with images, we will use the discrete Gaussian filter rather than the continuous one. Run `Problem 3.c` to confirm that the two motions were properly magnified and submit the generated plot in your report and include the file named `problem_3c.png` in the zip.

(d) (2 points) We are now ready to apply motion magnification to videos. We will use the same approach as in part c of magnifying Gaussian windowed regions of the video frames. Rather than directly finding the phase shifts between consecutive video frames, we will keep a moving average of the Fourier transform phases and compare each new frame's DFT phase with the current moving average of phase. The moving average is an IIR low-pass filter, averaging 0.5 times the previous average with 0.5 times the current phase. For simplicity, each of the RGB channels are processed independently and identically. In `Problem 3.d`, you will need to fill in the Gaussian filter in line 28, the DFT phase of the magnified window in line 44, and the DFT of the magnified window in line 47. Please run `Problem 3.d` and submit the generated video. Note that the code may take some time to run - you can temporarily modify `sigma` to decrease the number of windowed regions to process.

As a deliverable, please include `bill_magnified.avi` generated using `sigma` as 13 and `magnification factor` as 10.

Problem 4 [6.869 Only] *Eulerian Motion Magnification*

We will implement the predecessor of the phase based motion magnification, Eulerian motion magnification [2]. This method applies spatial decomposition and temporal filtering to magnify the subtle signals in the input video. The paper showed color magnification which could reveal color change due to the flow of blood as it fills the face, and also subtle motions like pulse on the wrist and moving abdomen of a baby as it breathes.

(a) (1 point) Fill in the function `create_gaussian_pyramid`, which takes in the video frames and outputs the Gaussian pyramid consisting of 4 levels. Please include the figure in your report and also include the file saved as `gaussian_pyramid.png` in your report and the zip.

(b) (1 point) Fill in the function `create_laplacian_pyramid` which takes in the gaussian pyramid of the videos and outputs a list containing the different levels of the laplcian pyramid. Note that you will have only 3 levels for the laplacian pyramid. Please include the output `laplacian_pyramid.png` in your report and the zip file.

(c) (1 point) Now we will perform the temporal filtering of the laplacian pyramids of the video, you can refer to section 3.1 of ref [2] to learn how the filter is applied in the 1D scenario. We will first create a Butterworth bandpass filter [1] to convert a user-specified frequency band to a second-order IIR structure[2]. First, complete the TODOs in the `butter_bandpass_filter` function using the `signal.butter` and `signal.lfilter` functions. This filter is then applied to each of the levels of the Laplacian pyramid and the filtered signal is amplified using the `amplification` factor set to 20. Please use the low and high frequencies as 0.4 and 3.

(d) (1 point) To compute the final amplified signal, loop over all the amplified filtered signals and add the sum of all to the original video. Please submit the generated video named `baby_euler_magnification.avi` in the zip.

---

[1] https://www.geeksforgeeks.org/digital-low-pass-butterworth-filter-in-python/
[2] IIR. https://en.wikipedia.org/wiki/Infinite_impulse_response

References

[1] Aude Oliva, Antonio Torralba, and Philippe G Schyns. Hybrid images. *ACM Transactions on Graphics (TOG)*, 25(3):527–532, 2006.

[2] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William T. Freeman. Eulerian video magnification for revealing subtle changes in the world. *ACM Transactions on Graphics (Proc. SIGGRAPH 2012)*, 31(4), 2012.