FINAL ASSIGNMENT

Seasons of Code

Vision Venture

2024

Motion Magnification

In this final assignment 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) 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 MagnifyChange. 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 (a) and submit the generated plot in the report.

(b) 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 this part, 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 (b) and attach the generated plot in the report.

(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) 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 this, 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 (c) to confirm that the two motions were properly magnified and submit the generated plot in your report.

(d) 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 (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 (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.

Eulerian Motion Magnification

We will implement the predecessor of the phase based motion magnification, Eulerian motion magnification. 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) 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 plot in your report.

(b) 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 laplacian pyramid. Note that you will have only 3 levels for the laplacian pyramid. Please include the plot in your report.

(c) Now we will perform the temporal filtering of the laplacian pyramids of the video. 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) (10 points) 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 .

Note that the output video will likely flicker and show severe magnification artifacts for about 4 seconds before showing a noisy magnified video. The initial artifacts should be ignored.

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, Fr´edo 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.

---

[1] https://www.geeksforgeeks.org/digital-low-pass-butterworth-filter-in-python/

[2] IIR. https://en.wikipedia.org/wiki/Infinite_impulse_response