CS 101: Motion Magnification

Michelle Zhao[1]
California Institute of Technology[1]
mzhao@caltech.edu

Abstract

In this project, I implemented phase-based motion magnification, an algorithm developed by Wadhwa et al.[1]. The phase-based motion magnification technique uses complex-valued steeerable pyramids to decompose phase variations and amplitude variations over time. By temporally processing the motions, small imperceptible motions can be amplified and the video can be reconstructed with the amplified motion. This processing is lightweight, doesn't require expensive optical flow computations, and is less sensitive to noise than Eulerian video magnificaiton methods. In my implementation, I use an ideal bandpass temporal filter. I experimented with directed control over the areas and patterns of magnification. By using a functions for changing magnification factors and by identifying which frequency subbands of the video corresponded to different strings, I found that we can design controlled patterns of motion in the motion-magnified videos.

1. Project Introduction

Many motions and phenomena pass by unnoticed by the human eye. Cameras are able to capture tiny motions that are generally inperceptible to human eyes, but can be captured through video. Consider the breathing of child, a person shivering, or the rapid flutter of a hummingbird's wings. By magnifying the motion of videos of these phenomena, we can make them visible to the human perception system. This has major implications for security, including developing better visual systems for identifying suspicious activity or better monitoring babies.

Wadhwa et al 2013[1] developed a phase-based approach to motion magnification that amplifies small motions by modifying local phase variations in a complex steerable pyramid representation of the video. First, complex steerable pyramids are used to decompose the video and separate the amplitude of the local wavelets from their phase. The steerable pyramid is built by applying transfer functions to the discrete Fourier transform of an image and decomposing it into different spatial frequency bands. The spatial frequency bands contain motion information. Then, the next step is to temporally filter the phases independently at each location, orientation and scale. Lastly, the algorithm amplifies the temporally-band-passed phases and reconstruct the video.

This process allows us to manipulate the motion by modifying the phase. Then, we can reconstruct the motion magnified video by collapsing the pyramid. To synthesize magnified motion, the approach multiplies the band-passed phases by an amplification factor .

In my project, I wanted to extend their work further by experimenting with two different modifications. First, I wanted to see what happens to the output image when non-constant values of alpha are applied to different band-passed phases. The question was can we make non-constant amplifications or other modifications to the input in the steerable pyramid space such that those changes alter realistically the reconstructed video?

The second modification I wanted to try was seeing if by changing the range of the temporal bandpass filter, I could design the motion magnifier to magnify very particular areas of the video.

2. Related Work

Rudimentary methods of motion magnification required the use of optical flow computations, which remain expensive to compute. Lagrangian approaches, such as [2], require that motion is computed explicitly, and each frame is modified through warping functions.

Eulerian methods of motion magnification removed the need for expensive flow computation, because they process the video separately in space and time. [3] presents linear Eulerian motion magnification, which suffers from being highly sensitive to noise because it magnifies both the amplitude and phase, and thus amplifies noise.

Wadhwa et al 2013[1] developed a phase-based approach to motion magnification that amplifies small motions by modifying local phase variations in a complex steerable pyramid representation of the video. First, complex steerable pyramids are used to decompose the video and separate the amplitude of the local wavelets from their phase. The steerable pyramid is built by applying transfer functions to the discrete Fourier transform of an image and decomposing it into different spatial frequency bands. The spatial frequency bands contain motion information. Then, the next step is to temporally filter the phases independently at each location, orientation and scale. Lastly, the algorithm ampli-

fies the temporally-band-passed phases and reconstruct the video.

First, using a Fourier series decomposition, the displaced image profile is written as a sum of complex sinusoids, in which each band corresponds to a single frequency. The band for a given frequency w is a complex sinusoid. The phase of the sinusoid contains motion information. By modifying the phase, we can modify the motion. To isolate motion in specific temporal frequencies, we filter the phase with a DC balanced filter that removes the DC component. The result is a bandpassed phase. We then multiply the bandpassed phase by a constant alpha and increase the phase of the sinusoidal sub-band by this amount to get the motion of the magnified sub-band. Lastly, the motion-magnified video is reconstructed by collapsing the pyramid.

This method extends the magnification capability of previous work (Wu et al. 2012), and differs from the previous method by using a complex steerable pyramid instead of a Laplacian pyramid. This new method especially outperformed the previous on noisy videos in generating videos with fewer artifacts and more realistic motions. Both this Eulerian method and Lagrangian methods rely on spatial pyramids, where each level is band limited. [1] argues that such spatially bandpassed images are better approximated by sinusoidal waves than linear ramps.

3. Problem Statement

In this project, I followed the algorithm presented by the original phase-based algorithm in [1]. First, for each frame, I construct a complex steerable pyramid with 4 different orientations, and 3 levels of scaling. Then, I separate the phase variation from the amplitude variation. I next temporally filter the phases independently at each location, scale, and orientation. I did not perform phase denoising, which is an optional step that can achieve spatial smoothing and generate better results. I found that the results I got with the basic algorithm were fairly good, so I did not perform the optional phase denoising. The below diagram is the algorithm I followed, and was created by modifying the algorithm diagram from [1], with phase-denoising removed.

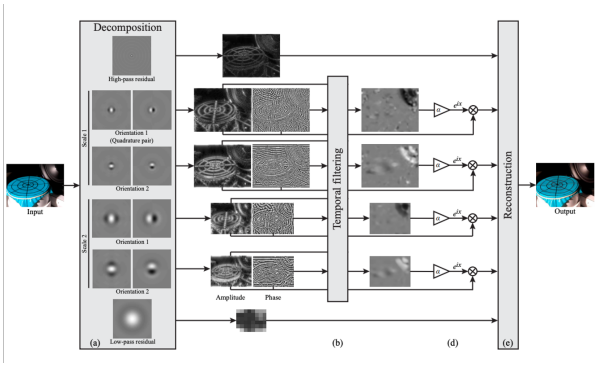

Figure 1. Phase-based motion magnification algorithm.

4. Background

4.1. Complex Steerable Pyramid

The steerable pyramid is linear, multi-scale, and multi-orientation image decomposition that was developed by E. Simoncelli [4]. Given an input image, the algorithm first splits the image into a high frequency and low frequency decomposition. Then, it applies bandpass oriented filters to the low frequency image to get k oriented subbands at each of the orientations at the first scale. Next, we downsample the low pass filtered image, and this becomes input to the next scale. At the second scale, the high-low pass decomposition and oriented bandpass filtering is performed on the downsampled low-pass image. This is recursively performed for all scale levels.

The orientated filters are directional second derivatives oriented at various angles θ. In this algorithm, I used 4 oriented filters at orientation angles $\theta = 0, \frac{\pi}{2}, , \frac{3\pi}{4}, , \frac{\pi}{4}$. The highpass and lowpass filters are necessary for preprocessing the image in preparation for the recursion. The low-pass portion is subsampled, and the recursion is performed by repeated applying the recursive transformation to the low-pass signal. The below algorithm and pyramid example is borrowed from [5].

Algorithm 1: Steerable pyramid decomposition

Input : Number of scales P, number of orientations Q, discrete image u of size $M \times N$ such that M and N are multiples of 2^P.

Output: Steerable pyramid of u: sequence of $PQ + 2$ images of varying size (see figures 1 and 2 for illustration)

1. Compute the high frequency residual $h_0 * u$ and store it in the pyramid.
2. Compute the low frequency band $v \leftarrow l_0 * u$.
3. **for** scale $p = 1$ **to** P **do**
4. Compute the oriented high frequency band $b_q * v$ for each orientation $q = 0, \ldots, Q-1$ and store it in the pyramid (these Q images constitute the p-th scale of the pyramid).
5. $v \leftarrow$ the low frequency image $l * v$.
6. Downsample v by a factor of two.
7. **end**
8. Store the remaining image v (of size $M/2^P \times N/2^P$) as the low frequency residual of the pyramid.

Figure 2. Complex Steerable Pyramid Algorithm.

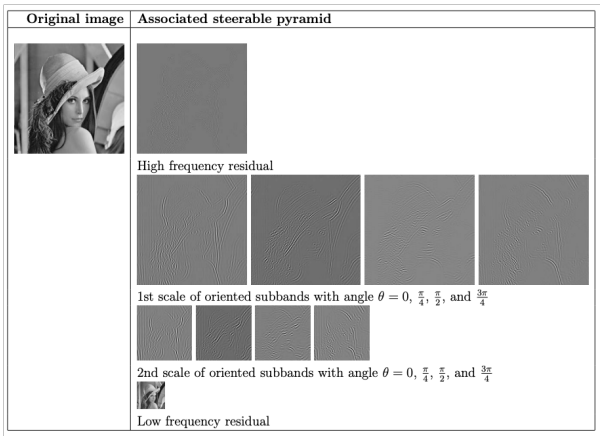

Figure 3. Example of Complex Steerable Pyramid Algorithm.

4.2. Ideal Bandpass Temporal Filter

A bandpass filter passes frequencies within a certin range and zeros, or rejects, frequencies outside of that range. An

ideal bandpass filter has a completely flat passband, meaning there's no gain or loss throughout the filter. All frequencies outside of the passband are zeroed.

The main function of bandpass filters are to limit the bandwidth of an output signal to the band allowed for transmission. In the temporal filter, we limit the window we are looking at to a 30 frame window. We apply a bandpass filter to the 30 frame window to have frequencies within our low and high threshold. The discrete Fourier transform (DFT) converts the finite sequence of 30 equally-spaced frame samples of a video function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency. The band-pass filter is applied to the DFT sampling frequencies of the 30-frame sequence.

5. Algorithm

Algorithm 1: Phase Based Motion Magnification

$factor \leftarrow$ motion magnification factor ;
for *each frame* f **do**
  $P \leftarrow$ Generate steerable pyramid for f ;
  $p \leftarrow$ Flatten steerable pyramid for f ;
  $phase \leftarrow$ Extract phase variation for p ;
  $amp \leftarrow$ Extract amplitude for p ;
  $filterPhase \leftarrow$ Run ideal bandpass temporal filter on $phase$;
  $magnifiedPhase = phase - filterPhase + (filterPhase * factor)$
  $f* \leftarrow$ reconstruct frame from steerable pyramid $Reconstruct(magnifiedPhase, amplitude)$;
end

6. Implementation Details

In constructing the complex steerable pyramid, I used the Perceptual Python package, which has a steerable pyramid implementation. The number of scales for the pyramid was 5, not including the highpass and lowpass images, and there were 3 oriented bandpass scales. At each bandpass scale, there were 4 oriented filters at angles $\theta = 0, \frac{\pi}{2}, , \frac{3\pi}{4}, , \frac{\pi}{4}$. I experimented with temporal window sizes from 10 frames to 40 frames, and found that the 20-30 frame window performed best by visual check. I used a stepsize of 1 for the temporal filter. I experimented with the frequency bandwidth, and tried a range of 70 to 90, 90 to 110, and 110 to 130. The sampling rate used for computing the DFT sample frequencies was 600 frames per second. I experimented with nonuniform magnification factors: magnification sampled from a normal distribution and a sinusoidal normal distribution. The objective of these experiments was to see if nonuniform magnification would produce artifacts in the videos or would create visually realistic videos.

7. Setup

7.1. Data

After combing the internet for videos, one challenge was that I found it was difficult to find short, imperceptible motion videos that served the purpose of motion magnification well. I found many videos with high motion, much too high resolution, and these produced very poor output.
The videos I used where a video of a guitar without perceptible motion, from the source videos of Wadhwa et al. [1]. I also used a video of a tulip in motion, taken from "in-the-wild" online, and despite large motion in the video, the tulips showed decent performance.

7.2. Pre-processing

I used a maximum of 300 frames of each video for processing. I also converted each video to grayscale in order to enable use of the Perceptual package for constructing the steerable pyramid. This also sped up the rate of computation.

7.3. Evaluation Metrics

The problem of motion magnification is difficult to apply robust evaluation metrics to. Instead of mathematical evaluation metrics, I observed the output videos of each of my experiments and gave a detailed description of the change made and the quality of the output.

8. Results

8.1. Experiment 1: Magnifying First String of Guitar

By applying a bandpass temporal filter to frequencies within a 70 to 90 range, I was able to magnify only the motion of the first string. The first string refers to the topmost string in the frame, since I don't know the string name. This demonstrates that because the phase variation of the first string was operating at a frequency in the range of 70 to 90, we were able to magnify only the first string's motion. Below is a screenshot, and the videos folder in the Github link contains a video of the magnified first-string motion.

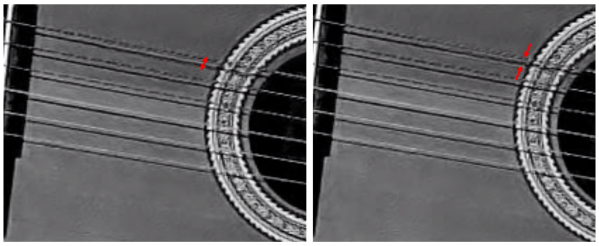

Figure 4. Motion of first string magnified.

8.2. Experiment 2: Magnifying Second String of Guitar

By applying a bandpass temporal filter to frequencies within a 110 to 130 range, I was able to magnify only the

motion of the second string (second from the top). This demonstrates that because the phase variation of the second string was operating at a frequency in the range of 110 to 130, we were able to magnify only the second string's motion. Experiments 1 and 2 show that if we can identify the approximate frequency of a particular subband of motion, we can control exactly what motion we are magnifying. Below is a screenshot of two significant frames of the outputted video, and the videos folder in the Github link contains a video of the magnified first-string motion.

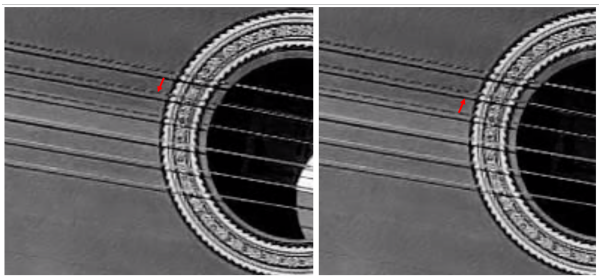

Figure 5. Motion of second string magnified.

8.3. Experiment 3: Non-uniform Magnification: Sinusoidal Magnification Function

A sinusoidal magnification function causes the motion of the string to be periodic. We magnify using the 70-90 frequency band-pass filter, and the motion of the first string is magnified. The sinusoidal magnification causes the first string to oscillate, and then stop, oscillate again, and stop, continuously. The string doesn't actually stop oscillating when it appears to stop, the motion is just reduced to an imperceptible level. The videos are available in the submitted code and on the Github link. This experiment shows that using phase-based motion magnification we can generate patterns of motion as we please. The video looks as if the guitar is strumming on its own, and by changing the period or amplitude of the sinusoidal pattern, we can strum the guitar in such a pattern that we can design.

8.4. Experiment 4: Non-uniform Magnification: Gaussian Sampling

Gaussian sampling gives the video erratic movement. In this experiment, I generated magnification factor from a normal distribution with a mean of 30 and standard deviation of 15. The result was choppy motion of the first string. Although the results are not very smooth due to the stochasticity, it further demonstrates that we can successfully make very specific changes to the motion.

8.5. Experiment 5: Non-uniform Magnification: Decreasing Magnification Generates Focusing Effects

I found that using a linearly decreasing function for motion magnification can generate a focusing effect. At the start of the video, I use an extremely high magnification of 300, which causes a watery effect on the frame. We decrease the magnification down to 30 at the end, and we get a focusing effect, where the watery filter becomes more and more in focus.

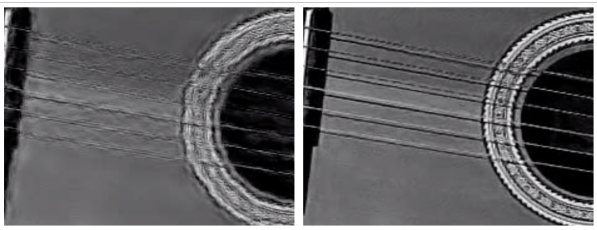

Figure 6. Left: Video not in focus, watery-effect. Right: Video in focus.

8.6. Experiment 6: Minimizing Tulip Motion

My final experiment was minimizing tulip motion. In my tulip video, the tulips are swaying violently in the wind, and I aimed to minimize their motion by using a magnification factor < 1. I did not achieve a successful minimization of the tulip motion. The issue likely is in that the video contained motion throughout all subbands, and there was both background and foreground motion. This experiment draws attention to the limitations of this algorithm on in-the-wild videos that usually contain motion throughout the video. Magnification of very specific parts of the in-the-wild videos requires careful tuning.

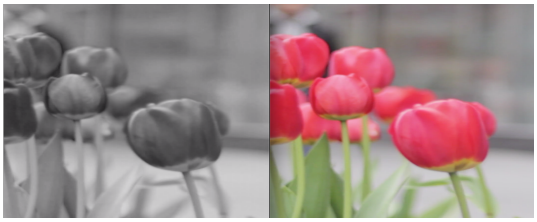

Figure 7. Tulip output screenshot.

9. Implementation

My code and videos can be found at `https://github.com/mzhao98/Motion-Magnification`.

10. Conclusion

In this project, I was able to successfully magnify the motion of low-motion videos using [1] phase-based video magnification algorithm. I experimented with directed control over the areas and patterns of magnification. By using a sinusoidal magnification factor, I found that I could simulate the strumming of a guitar at a particular rhythm or pattern. By identifying which frequency subbands of the video corresponded to different strings, I found that I could select which string I wanted to magnify the motion of, and selectively choose the parts of the video I wanted to magnify. My experiments show that using phase-based motion magnification we can generate patterns of motion as we please. Using

this type of directed control, we can select the strings and pattern with which to use to play the guitar as if strumming on its own.

11. Acknowledgements

I would like to thank Professor Katie Bouman and CS101 teaching assistants Mason McGill and Brendan Hollaway for their guidance in research and teaching throughout the term.

12. Bibliography

1. Wadhwa, N., Rubinstein, M., Durand, F., Freeman, W.T.: Phase-based video motion processing. ACM Trans. Graph. (SIGGRAPH) 32(4), 80 (2013)
2. Liu, C., Freeman, W. 2010. A high-quality video denoising algorithm based on reliable motion estimation. In Computer Vision ECCV 2010, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6313 of Lecture Notes in Computer Science. Springer, Berlin Heidelberg, 706–719.
3. Wu, H.-Y., Rubinsten, M., Shih, E., Guttag, J., Durand, F., Freeman, W. 2012. Eulerian video magnification for revealing subtle changes in the world. ACM Trans. Graph. (Proc.SIGGRAPH) 31 (aug). iE. P. Simoncelli and W. T. Freeman, "The steerable pyramid: a flexible architecture for multi-scale derivative computation," Proceedings., International Conference on Image Processing, Washington, DC, USA, 1995, pp. 444-447 vol.3.
4. Thibaud Briand, Jonathan Vacher, Bruno Galerne, and Julien Rabin, The Heeger Bergen Pyramid Based Texture Synthesis Algorithm, Image Processing On Line, 4 (2014), pp. 276–299.