Revealing and Analyzing Imperceptible Deviations in Images and Videos

by

Neal Wadhwa

Submitted to the Department of Mathematics
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Applied Mathematics
at the
Massachusetts Institute of Technology

February 2016

Signature of Author: Signature redacted

Department of Mathematics
January 29th, 2016

Certified by: Signature redacted

William T. Freeman
Thomas and Gerd Perkins Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Signature redacted

Jonathan A. Kelner
Associate Professor of Mathematics
Chairman, Committee for Graduate Students

Revealing and Analyzing Imperceptible Deviations in Images and Videos

by

Neal Wadhwa

Submitted to the Department of Mathematics
on January 29th, 2016 in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy in
Applied Mathematics

Abstract

The world is filled with objects that appear to follow some perfect model. A sleeping baby might look still and a house's roof should be straight. However, both the baby and the roof can deviate subtly from their ideal models of perfect stillness and perfect straightness. These deviations can reveal important information like whether the baby is breathing normally or whether the house's roof is sagging.

In this dissertation, we make the observation that these subtle deviations produce a visual signal that while invisible to the naked eye can be extracted from ordinary and ubiquitous images and videos. We propose new computational techniques to reveal these subtle deviations by producing new images and videos, in which the tiny deviations have been magnified.

We focus on magnifying deviations from two ideal models: perfect stillness and perfect geometries in space. In the first case, we leverage the complex steerable pyramid, a localized version of the Fourier transform, whose notion of local phase can be used to process and manipulate small motions or changes from stillness in videos. In the second case, we find hidden geometric deformations in images by localizing edges to sub-pixel precision.

In both cases, we experimentally validate that the tiny deviations we magnify are indeed real, comparing them to alternative ways of measuring tiny motions and subtle geometric deformations in the world. We also give a careful analysis of how noise in videos impacts our ability to see tiny motions. Additionally, we show the utility of revealing hidden deviations in a wide variety of fields, such as biology, physics and structural analysis.

Thesis Supervisor: William T. Freeman
Title: Thomas and Gerd Perkins Professor of Electrical Engineering and Computer Science
Thesis Committee: Professor Frédo Durand, Professor Alan Edelman, Professor John W. M. Bush

Acknowledgments

I would like to thank my advisor: Professor William T. Freeman for his support and advice, and Professor Frédo Durand, with whom have I collaborated closely. I would also like to thank my other collaborators: Dr. Michael Rubinstein, Abe Davis, Justin Chen, Dr. Tali Dekel, Donglai Wei, Dr. Gautham Mysore and Tianfan Xue. I also thank Professors John W. M. Bush and Alan Edelman for serving on my thesis examination committee.

I would also like to thank my current and former office mates: Andrew Owens, Joseph Lim, Roger Grosse, and Jiajun Wu, and my fellow Spring 2015 Computational Photography TAs: Katie Bouman, Adrian Dalca and Gaurav Chaurasia. I thank Katie again for giving me helpful feedback on the title and abstract of this dissertation.

I also want to thank Justin Chen again for all the high-speed videos of pipes and cantilevered beams. I thank Jon Blake Sellon and Denny Freeman's group for sending us videos of an in-vitro mammalian tectorial membrane, and Katia Bertoldi's group at Harvard for allowing us to film their metamaterials.

I would like to acknowledge fellowships and grants that have funded my research during my Ph. D: National Defense Science and Engineering Fellowship, National Science Foundation Graduate Fellowship, Quanta Research, Shell, NSF CGV-1111415 (Analyzing Images Through Time), and the MIT Mathematics Department.

Finally, I would like to thank my parents for their life-long support of my decisions.

Table of Contents

List of Figures

Chapter 1

Introduction

A traditional microscope takes a slide that has details that are too small to see with the naked eye and, through the magic of optics, magnifies it hundreds of times to reveal a beautiful world of bacteria, cells, pollen, crystals, and material structures. There is another invisible and fascinating world to be visualized: that of tiny deviations from perfect models. A baby lying in a crib and a construction crane appear to be perfectly stationary and a house's roof appears to be (and should be) a straight line. However, these objects deviate slightly from their ideal models of perfect stillness and perfect straightness. The baby's chest moves because she is breathing, the construction crane sways in the wind and the house's roof is actually sagging (Fig. 1.1). In this dissertation, we develop a *deviation* microscope: a tool that relies not on optics, but on computation to amplify deviations from perfect stillness and perfect geometries to reveal minuscule motions and subtle geometric imperfections in ordinary images and videos.

We focus on two types of perfect models: perfect stationarity in time and ideal geometries in space. In the first case, we consider input videos that look entirely static to the naked eye, and we use algorithms we have developed to output new videos of the same scene, but where tiny invisible motions and changes over time have been magnified so as to become visible. We can take videos where objects move by only a fraction of a pixel (as small as 1/100th of a pixel) and magnify them until the motion is obvious and spans several pixels. In the second case, we look at single images and find objects that

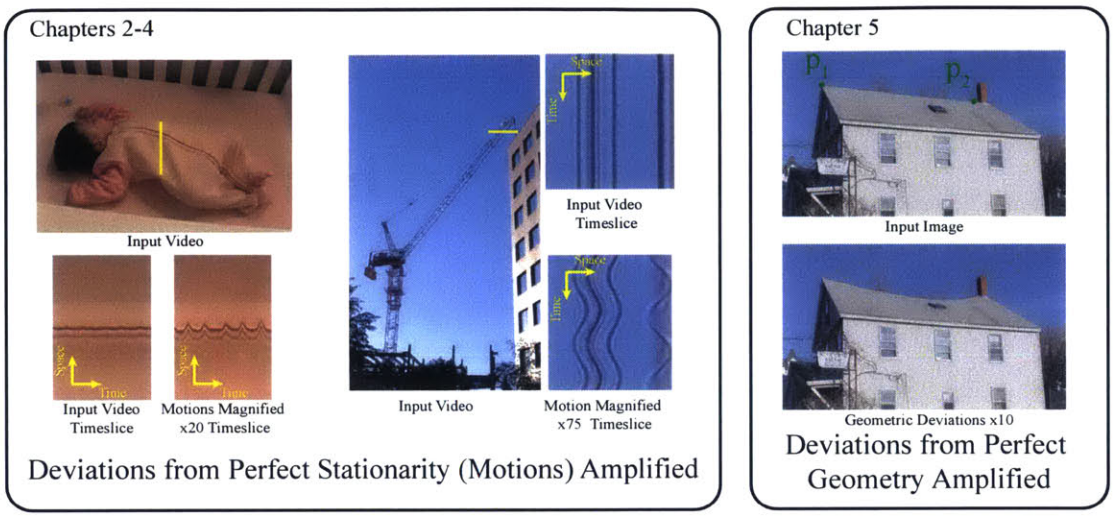


Figure 1.1: Amplifying deviations from models. On the left side, the deviations from perfect stillness in apparently static videos are amplified to reveal the breathing of a baby and the swaying of a crane. On the right side, geometric deviations from a perfect line are amplified to reveal that the house's roof is sagging.

look like ideal geometries, e.g. lines, circles and ellipses. We then produce new images, in which the deviations of those objects from their corresponding geometries have been magnified so as to become visible.

Magnification gives a visualization of tiny motions and subtle geometric deviations that is distinct from quantitatively knowing the exact motions or geometric deviations in a video. These quantitative numbers can be useful for many applications. However, it is often difficult to interpret them, especially when they are computed densely for all pixels or all ideal geometries in an image. Seeing the motions or geometric deviations as if they are larger is easy to understand and can assist in interpreting a dense motion field or a dense set of geometric deviations.

Magnifying Tiny Motions in Apparently Still Videos Subtle changes from perfect stillness are caused by many phenomena and occur in many objects. These tiny motions are surprisingly easy to process. In Chapter 2, we discuss previous methods of amplifying these tiny motions. Then, we describe our new method to amplify them robustly and accurately and show the utility of magnifying motions on examples from a wide variety of fields. In Chapter 3, we discuss a way to make this processing extremely fast and capable of running in real-time on modern laptops.

In Chapter 4, we discuss how magnifying tiny motions can be useful for scientists and engineers. To this end, we add two new capabilities to augment motion magnification: a quantitative readout of the tiny motions, so that they can be further analyzed and a measure of the amount of expected noise in the motion signal. This measure of noise allows us to distinguish when tiny motions are true signal and when they are spurious caused by noise present in every video.

Magnifying Geometric Deviations Many objects in the world follow an ideal geometry, such as a line, circle or ellipse. For example, buildings are usually designed to be straight and in ideal conditions, bubbles tend to be perfect circles due to surface tension. In reality, however, such flawless behavior hardly exists, and even when invisible to the naked eye, objects depart from their idealized models. The building may start to sag or tilt, and in the presence of gravity, the bubble may be slightly oval. In Chapter 5, we discuss techniques to magnify these deviations that can be used to reveal interesting and important information about a variety of objects and phenomena.

This dissertation is largely based on work that has appeared in ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013 and Proceedings of SIGGRAPH Asia 2015) [80, 81] and the 2014 IEEE International Conference on Computational Photography (ICCP) [82].

Chapter 2

Phase-Based Motion Magnification

In this chapter, we focus on magnifying subtle deviations from perfect stillness in almost static videos. Such videos look like still images, but actually contain tiny color changes and motions that if magnified, can reveal numerous and interesting phenomena. We present a new method of amplifying tiny motions in videos that is robust, fast and accurate.

2.1 Introduction

A plethora of phenomena exhibit motions that are too small to be well perceived by the naked eye and require computational amplification to be revealed [53, 85]. There are two methods of revealing these subtle motions, whose difference is best described by a fluid dynamics metaphor. In the *Lagrangian* perspective, the motion of fluid particles is tracked over time, similar to observing a river flow from the moving perspective of a boat. This is the approach taken by Liu et al. [53]; points in the scene are tracked and then pixel colors are advected across the frame according to magnified motion vectors. This method relies on accurate and dense motion estimation in videos, which is challenging and computationally-intensive.

In contrast, the *Eulerian* perspective considers a fixed reference frame and characterizes fluid properties over time at each fixed location, akin to an observer watching the water from a bridge. Wu et al. [85] follow this approach by treating the time series

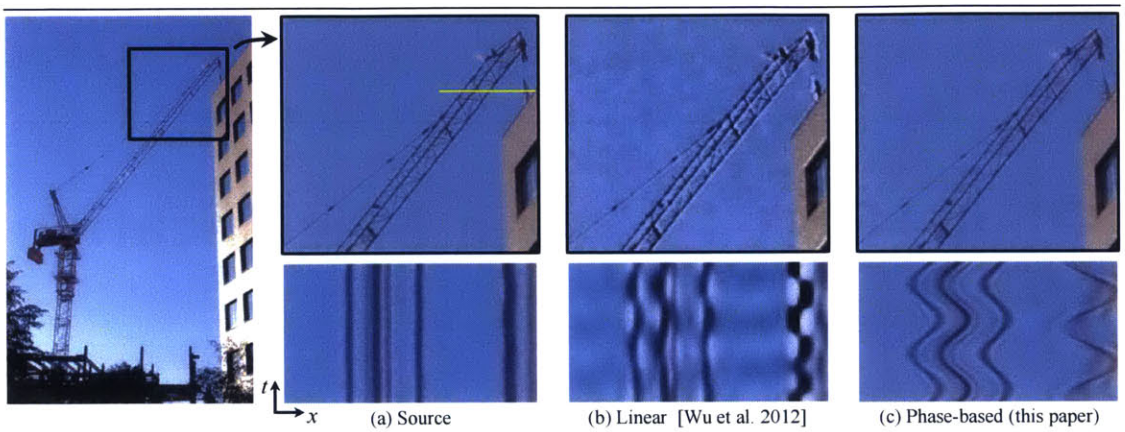


*Figure 2.1: Motion magnification of a crane imperceptibly swaying in the wind. (a) Top: a zoom-in onto a patch in the original sequence (*crane*) shown on the left. Bottom: a spatiotemporal XT slice of the video along the profile marked on the zoomed-in patch. (b-c) Linear [85] and phase-based motion magnification results, respectively, shown for the corresponding patch and spatiotemporal slice as in (a). The previous, linear method visualizes the crane's motion, but amplifies both signal and noise and introduces artifacts for higher spatial frequencies and larger motions, shown by the clipped intensities (bright pixels) in (b). In comparison, our new phase-based method supports larger magnification factors with significantly fewer artifacts and less noise (c). The full sequences are available in the supplemental video.*

of intensity values at each pixel independently and amplifying temporal variations in a frequency band of interest to produce a motion magnified video. Remarkably, this simple method of looking at only color changes at each pixel can be used to manipulate *small* motions because small motions and color changes are linked through a first-order Taylor series expansion. This method is fast and robust eliminating the need for costly flow computation. Unfortunately, because linear Eulerian video magnification [85] relies on a first-order Taylor expansion, it supports only small magnification factors. It can also significantly amplify noise when the magnification factor is increased (Fig. 2.1(b)).

In this chapter, we propose a new Eulerian technique to amplify and manipulate small motions, based on complex-valued steerable pyramids [62, 72], and inspired by phase-based optical flow [29, 34] and motion without movement [32]. Instead of amplifying temporal variations in pixel intensity, we instead amplify local phase variations in a complex steerable pyramid representation of the input video. This representation is similar to a localized version of the Fourier transform. Using local phase to analyze and

manipulate motions is similar to modifying the phase of Fourier basis coefficients—sine waves—to translate them. This change is only in representation, not in algorithm.

Manipulating local phase variations improves on the previous, linear Eulerian magnification method [85] in two important aspects (Fig. 2.1): using local phase (a) achieves larger magnifications, and (b) has substantially better noise performance. Because Wu et al. [85] amplify temporal brightness changes, the amplitude of noise is amplified linearly. In contrast, the presented method modifies phases, not amplitudes, which does not increase the magnitude of spatial noise. We demonstrate that the phase-based method can achieve larger motion magnifications with fewer artifacts, which expands the set of small-scale physical phenomena that can be visualized with motion magnification techniques.

■ 2.2 Linear Eulerian Video Magnification

The motion magnification technique most closely related to the one presented in this chapter is linear Eulerian video magnification [65, 85]. We briefly review this method here. Its core idea is to independently process the time series of color values at each pixel. We do this by applying standard 1D temporal signal processing to each time series to amplify a band of interesting temporal frequencies e.g. around 1Hz (60 beats per minute) for color changes and motions related to heart-rate. The new resulting time series at each pixel yield an output video where tiny changes that were impossible to see in the input, such as the reddening of a persons face with each heart beat or the subtle breathing motion of a baby, get magnified and become clearly visible.

The idea of applying temporal signal processing on pixel values is straightforward, and has been explored in the past for regular videos [33, 61]. However, the results have been limited because such processing cannot handle general spatial phenomena such as large motions that involve complex space-time behavior across pixels. When

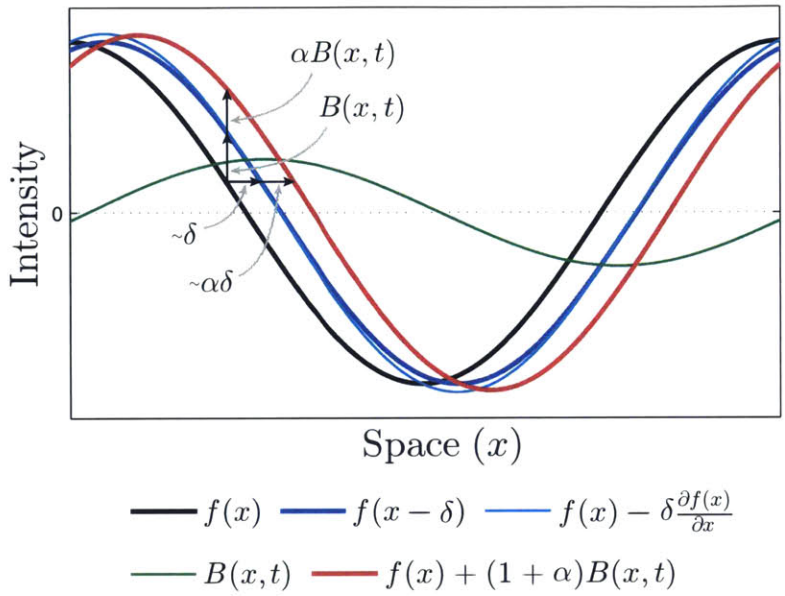


Figure 2.2: Amplifying intensity variations can approximate spatial translation. This effect is demonstrated here on a 1D signal, but equally applies to 2D. This input signal is shown at two time instants: $I(x,0) = f(x)$ at time 0 and $I(x,t) = f(x+\delta)$ at time t. The first-order Taylor series expansion of $I(x,t+1)$ around x approximates the translated signal. The temporal bandpass is amplified and added to the original signal to generate a larger translation. In this example, the amplification factor α is 1, amplifying the motion by 100%. (Reproduced from Wu et al. 2012 [85])

a large motion occurs, color information travels across many pixels and a Lagrangian perspective, in which motion vectors are computed, is required. One critical contribution of Eulerian video magnification is the observation that, in the special case of small motions, Eulerian processing can faithfully approximate their amplification. Because the motions involved are small, we can make first-order Taylor arguments to show that linear, per-pixel amplification of color variations closely approximates a larger version of the motion. We briefly formalize this in the case of 1D translational motion of a diffuse object under constant lighting and then explain the limitations of this technique.

1D translation Consider a translating 1D image with intensity denoted by $I(x,t)$ at position x and time t. Because it is translating, we can express the image's intensities

with a displacement function $\delta(t)$, such that $I(x,t) = f(x - \delta(t))$ and $I(x,0) = f(x)$. Fig. 2.2 shows the image at time 0 in black and at a later time translated to the right in blue. The goal of motion magnification is to synthesize the signal

$$\hat{I}(x,t) = f(x - (1+\alpha)\delta(t)) \tag{2.1}$$

for some amplification factor α.

We are interested in the time series of color changes at each pixel:

$$B(x,t) := I(x,t) - I(x,0). \tag{2.2}$$

Under the assumption that the displacement $\delta(t)$ is small, we can approximate the first term with a first-order Taylor series expansion about x, as

$$I(x,t) \approx f(x) - \delta(t)\frac{\partial f(x)}{\partial x}. \tag{2.3}$$

Because $I(x,0) = f(x)$, the color changes at x are

$$B(x,t) \approx -\delta(t)\frac{\partial f(x)}{\partial x}. \tag{2.4}$$

This is the first order approximation to the well known brightness constancy equation in optical flow [41, 54]: the intensity variation at a pixel x is the negative of the product between the displacement and the spatial gradient. This can be seen as a right triangle in Fig. 2.2, whose legs are the temporal intensity variation (vertical edge marked $B(x,t)$) and the displacement (horizontal edge marked δ) and whose hypotenuse (diagonal edge marked $\frac{\partial f(x)}{\partial x}$) has slope equal to the image's spatial derivative.

In our processing, we do not try to solve for $\delta(t)$. Instead, we amplify the color change signal $B(x,t)$ by α and add it back to $I(x,t)$, resulting in the processed signal

(in red in Fig. 2.2):

$$\tilde{I}(x,t) = I(x,t) + \alpha B(x,t). \tag{2.5}$$

Combining Eqs. 2.3, 2.4, and 2.5, we have

$$\tilde{I}(x,t) \approx f(x) - (1+\alpha)\delta(t)\frac{\partial f(x)}{\partial x}. \tag{2.6}$$

As long as $(1+\alpha)\delta$ is not too large, we can use another first-order Taylor expansion to relate the previous equation to motion magnification (Eq. 2.1). It is simply

$$\tilde{I}(x,t) \approx f(x - (1+\alpha)\delta(t)). \tag{2.7}$$

This shows that this processing magnifies motions. The spatial displacement $\delta(t)$ between frames of the video times 0 and t, has been amplified by a factor of $(1+\alpha)$.

Limitations of the Linear Approach Linear amplification relies on a first-order Taylor-expansion, which breaks down when either the amplification factor is too large or the input motion is too large. For overly large amplification factors, the magnified video overshoots the white level and undershoots the black level of the video causing clipping artifacts near edges where $\frac{\partial^2 f(x)}{\partial x^2}$ is large (Fig. 2.6a). If the input motion is too large, the initial Taylor expansion is inaccurate (Eq. 2.3) and the output contains not magnified motions, but instead ghosting artifacts.

A second limitation is that noise in the video is amplified as well. For example, suppose the image $I(x,t)$ has IID, additive white Gaussian noise $n(x,t)$ of variance σ^2. The difference between the frame at time t and at time 0 will contain a noise term

$$n(x,t) - n(x,0) \tag{2.8}$$

that has noise variance $2\sigma^2$. This noise term will be amplified by a factor α and the

output video will have noise of variance $2\alpha^2\sigma^2$, a much larger amount than in the input video (Fig. 2.7b).

In Wu et al. 2012, this amplification of noise variance was mitigated to some extent by reducing the amplification of high spatial-frequency temporal variations, which are mostly noise rather than signal. They did this by constructing a Laplacian pyramid of the temporal variations and using a lower amplification factor for the higher frequency levels, but this is not necessary and simply lowpassing the temporal variations spatially produces comparable results.

■ 2.3 Phase-Based Motion Processing

The appeal of the Eulerian approach to video magnification is that it independently processes the time series of color values at each pixel and does not need to explicitly compute motions. However, its reliance on first-order approximations limits its scope, and its use of linear amplification increases noise power. In this chapter, we seek to continue using the Eulerian perspective of motion analysis, processing independent time series at fixed reference locations. But, we want to do so in a representation that better handles motions and is less prone to noise.

In the case of videos that are global translations of a frame over time, there is a representation that is exactly what we want: the Fourier series. Its basis functions are complex-valued sinusoids that, by the Fourier shift theorem, can be translated exactly by shifting their phase (Fig. 2.3a,c). However, using the Fourier basis would limit us to only being able to handle the same translation across the entire frame, precluding the amplification of complex spatially-varying motions. To handle such motions, we instead use spatially-local complex sinusoids implemented by a wavelet-like representation called the complex steerable pyramid [71, 72]. This representation decomposes images into a sum of complex wavelets corresponding to different scales,

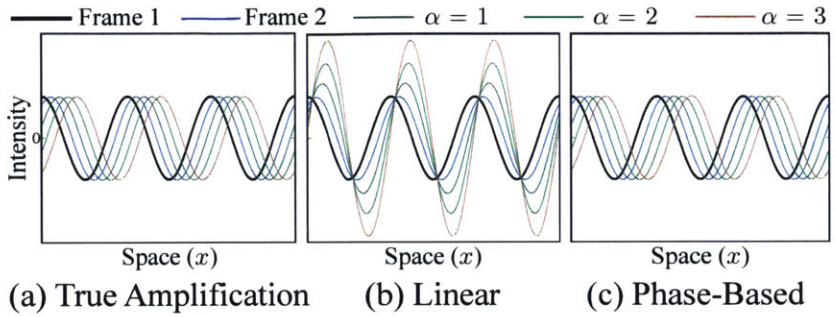


Figure 2.3: Phase-based motion magnification is perfect for Fourier basis functions (sinusoids). In these plots, the initial displacement is $\delta(t) = 1$.

orientations and positions. Each wavelet has a notion of local amplitude and local phase, similar to the amplitude and phase of a complex sinusoid (Fig. 2.4a). The key to our new approach is to perform the same 1D temporal signal processing and amplification described earlier on the local phase of each wavelet, which directly corresponds to local motion as we discuss below.

■ 2.3.1 Simplified global case

To provide intuition for what phase is and how it can be used to magnify motion, we work through a simplified example in which one dimensional translation of an image is magnified using the phase of global Fourier basis coefficients (Fig. 2.3.) This derivation is closely related to the derivation of the Fourier Shift Theorem [59].

Specifically, again let image intensity $I(x,t)$ be given by $f(x-\delta(t))$ where $\delta(0) = 0$. The profile $f(x)$ can be decomposed into a sum of complex coefficients times sinusoids using the Fourier transform

$$f(x) = \sum_{\omega} A_{\omega} e^{i\phi_{\omega}} e^{-i\omega x}. \tag{2.9}$$

Because the frames of I are translations of f, their Fourier transform is given by a phase

shift by $\omega\delta(t)$:

$$I(x,t) = \sum_{\omega} A_\omega e^{i\phi_\omega} e^{-i\omega(x-\delta(t))} = \sum_{\omega} A_\omega e^{i(\phi_\omega+\omega\delta(t))} e^{-i\omega x}. \tag{2.10}$$

where the phase of these coefficients becomes $\phi_\omega + \omega\delta(t)$. If we subtract the phase at time 0 from the phase at time t, we get the phase difference

$$\omega\delta(t), \tag{2.11}$$

which is proportional to the translation. Amplifying this phase difference by a factor α and using it to shift the Fourier coefficients of $I(x,t)$ yields

$$\sum_{\omega} A_\omega e^{i\phi_\omega+(1+\alpha)\omega\delta(t)} e^{-i\omega x} = f(x-(1+\alpha)\delta(t)), \tag{2.12}$$

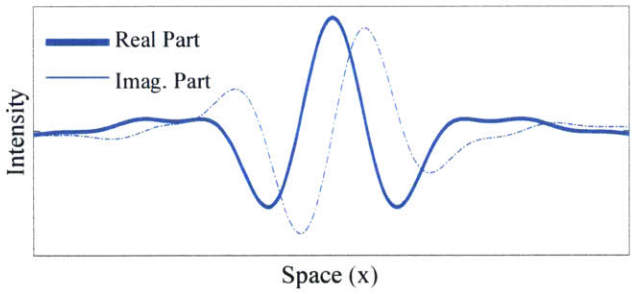
a new image sequence in which the translations have been *exactly* magnified.

Phase-based magnification works perfectly in this case because the motions are global and because the transform breaks the image into a representation consisting of exact sinusoids (formally, the Fourier transform diagonalizes the translation operator.) In most cases, however, the motions are not global, but local. This is why we break the image into local sine waves using the complex steerable pyramid.

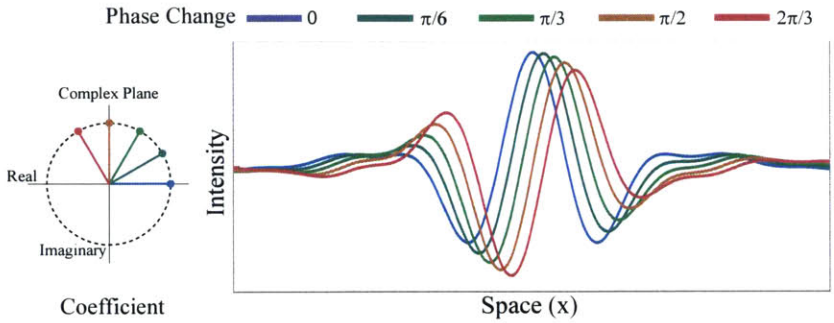
■ 2.3.2 Complex Steerable Pyramid

The complex steerable pyramid [71, 72] is a complex, overcomplete linear transform. It decomposes a single channel image $I(x,y)$ into a set of coefficients that correspond to basis functions that are simultaneously localized in position (x,y), spatial scale (r) and orientation (θ). The image can be reconstructed by multiplying the coefficients by the basis functions and summing.

The transform is best-described by its self-similar basis functions. Each one is a



(a) Single Complex Basis Function



(b) Varying Phase of Coefficient

Figure 2.4: Increasing the phase of complex steerable pyramid coefficients results in approximate local motion of the basis functions. A complex steerable pyramid basis function (a) is multiplied by several complex coefficients of constant amplitude and increasing phase to produce the real part of a new basis function that is approximately translating (b).

translation, dilation or rotation of another. So, it is sufficient to look at just one, a one dimensional version of which is shown in Fig. 2.4a. It resembles an oriented complex sinusoid windowed by a Gaussian envelope. The complex sinusoid provides locality in frequency while the windowing provides locality in space. Each basis function is complex, consisting of a real, even-symmetric part (cosine) and an imaginary, odd-symmetric part (sine). This gives rise to a notion of local amplitude and phase as opposed to the global amplitude and phase of Fourier basis functions.

The basis functions are chosen so that when the pyramid is built and collapsed without modification, the reconstructed image I_R perfectly matches the input image I. This requirement imposes conditions on the basis functions [71], which we derive now. Because the basis functions corresponding to a single scale r and orientation θ are translated copies of one another, the complex steerable pyramid can be implemented by convolving the image $I(x, y)$ with a basis function from each level $\Psi_{r,\theta}$:

$$L_{r,\theta}(x, y) = I(x, y) * \Psi_{r,\theta}. \tag{2.13}$$

These are the coefficients of the complex steerable pyramid representation. To reconstruct the image, we convolve them with the basis functions and then sum over the scales and orientations

$$\sum_{r,\theta} L_{r,\theta} * \Psi_{r,\theta} = I_R(x, y), \tag{2.14}$$

where $I_R(x, y)$ is the reconstructed image. To ensure that $I_R = I$, we need

$$I_R(x, y) = \sum_{r,\theta} L_{r,\theta} * \Psi_{r,\theta} = I(x, y) * \left(\sum_{r,\theta} \Psi_{r,\theta} * \Psi_{r,\theta} \right). \tag{2.15}$$

If we take the Fourier transform of both sides of this equation, we get

$$\hat{I}_R = \hat{I} \sum_{r,\theta} \left(\hat{\Psi}_{r,\theta}\right)^2 \Rightarrow \sum_{r,\theta} \left(\hat{\Psi}_{r,\theta}\right)^2 = 1 \tag{2.16}$$

as the necessary and sufficient condition for perfect reconstruction of the complex steerable pyramid.

■ 2.3.3 Local Phase Shift is Local Translation

The link between local phase shift and local translation has been studied before in papers exploring phase-based optical flow [29, 34]. Here, we demonstrate how local phase shift approximates local translation for a single basis function in a manner similar to the global phase-shift theorem of Fourier basis functions. We model a basis function as a Gaussian window multiplied by a complex sinusoid

$$e^{\frac{-x^2}{(2\sigma^2)}} e^{-i\omega x}, \tag{2.17}$$

where σ is the standard deviation of the Gaussian envelope and ω is the frequency of the complex sinusoid. In the complex steerable pyramid, the ratio between σ and ω is fixed because the basis functions are self-similar. Low frequency wavelets have larger windows.

Changing the *phase* of the basis element by multiplying it by a complex coefficient $e^{i\phi}$ results in

$$e^{\frac{-x^2}{(2\sigma^2)}} e^{-i\omega x} \times e^{i\phi} = e^{\frac{-x^2}{(2\sigma^2)}} e^{-i\omega(x-\phi/\omega)}. \tag{2.18}$$

The complex sinusoid under the window is translated, which is approximately a translation of the whole basis function by $\frac{\phi}{\omega}$ (Fig. 2.4b).

Conversely, the phase difference between two translated basis elements is proportional to translation. Specifically, suppose we have a basis element and its translation

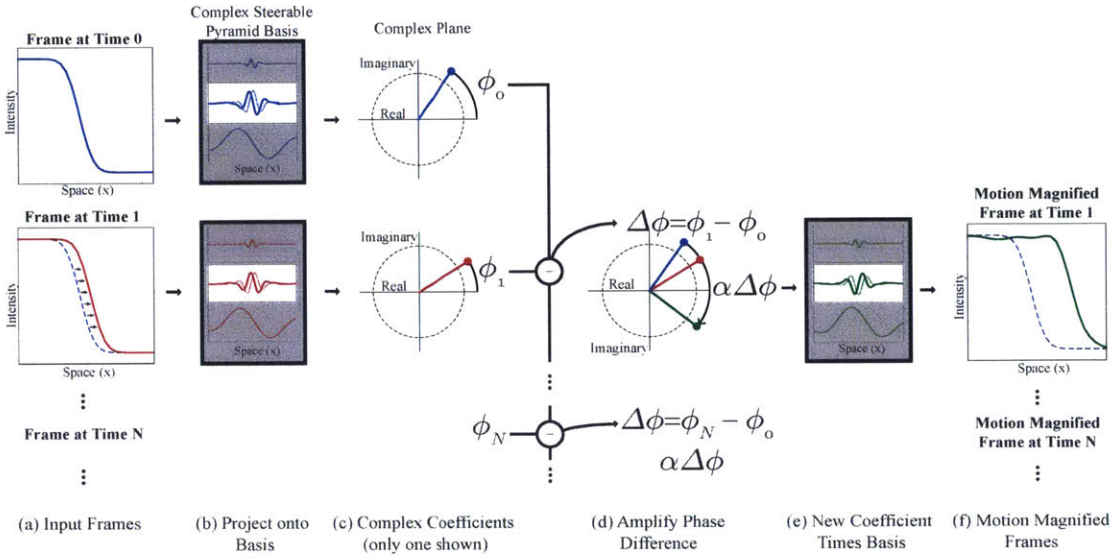


Figure 2.5: Using the local phase of complex steerable pyramid coefficients to amplify the motion of a moving step edge. Two frames from a video of a subtly translating step edge (a) are transformed to the complex steerable pyramid representation by projecting onto basis functions (b). The phase between the resulting complex coefficients (c) is computed and amplified (d). Only the coefficient corresponding to exactly one location and scale is shown; this processing is done to every coefficient. The new coefficients are used to shift the basis functions (e) and a reconstructed video is produced in which the motion between the two step edges is evident.

by δ:

$$e^{\frac{-x^2}{(2\sigma^2)}}e^{-i\omega x}, \; e^{\frac{-(x-\delta)^2}{(2\sigma^2)}}e^{-i\omega(x-\delta)}. \tag{2.19}$$

The local phase of each element only depends on the argument to the complex exponential and is $-\omega x$ in the first case and $-\omega(x-\delta)$ in the second. The phase difference is then $\omega\delta$, which is directly proportional to the translation. Local phase shift can be used both to analyze tiny translations and synthesize larger ones.

■ 2.3.4 Our Method

The observation that local phase differences can be used to manipulate local motions motivates our pipeline. We take an image sequence, project each frame onto the complex steerable pyramid basis and then independently amplify the phase difference between

all corresponding basis elements. This is identical to the linear amplification pipeline except that we have changed the representation from intensities to local spatial phases.

To illustrate the pipeline, consider again an image sequence $I(x,t)$, in which the frame at time 0 is $f(x)$ and the frames at time t are translations $f(x-\delta(t))$ (Fig. 2.5a). In our first step, we project each frame onto the complex steerable pyramid basis (Fig. 2.5b), which results in a complex coefficient for every scale r, orientation θ and spatial location x, y and time t. Because the coefficients are complex, they can be expressed in terms of amplitude $A_{r,\theta}$ and phase $\phi_{r,\theta}$ as

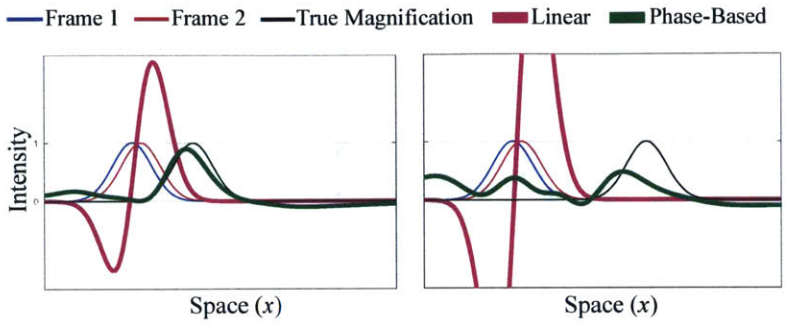
$$A_{r,\theta}(x,y,t)e^{i\phi_{r,\theta}(x,y,t)}. \tag{2.20}$$

In Fig. 2.5c, we show a coefficient at a specific location, scale and orientation in the complex plane at time 0 and at time 1.

Because the the two frames are slight translations of each other, every coefficient has a slight phase difference. This is illustrated in Fig. 2.5c, in which the coefficients have roughly the same amplitude but different phases. The next step in the basic version of our processing is therefore to take the phase difference between the coefficients in the video and that of a reference frame, in this case the frame at time 0:

$$\Delta\phi_{r,\theta}(x,y,t) = \phi_{r,\theta}(x,y,t) - \phi_{r,\theta}(x,y,0). \tag{2.21}$$

This phase difference is then amplified by a factor α as shown in Fig. 2.5d. This amplification yields a new set of coefficients for each frame, in which the amplitudes are the same, but the phase differences are larger. We can reconstruct the new frames using these coefficients by multiplying them by the basis functions (Fig. 2.5e) and then summing the real part to get new frames, in which the translations—and therefore the motions in the video—are revealed.

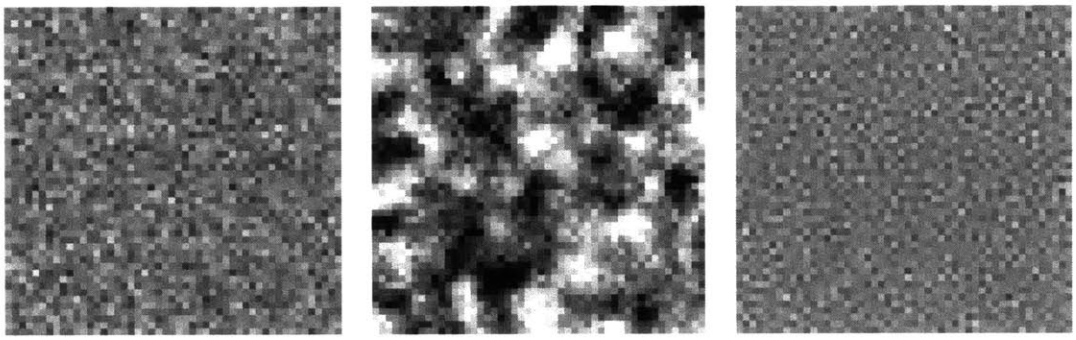


(a) $\alpha = 6$ (b) $\alpha = 14$

Figure 2.6: For general non-periodic structures, the phase-based method can support amplification factors of around four times as high as the linear method and does not suffer from intensity clipping artifacts (a). For large amplification, the different frequency bands break up due to the higher frequency bands having smaller windows (b).

Amplifying phase differences rather than pixel intensity differences has two main advantages: (a) it can support larger amplification factors, and (b) noise amplitude does not get amplified. In Fig. 2.6, we show the two different methods being used to amplify the motions of a Gaussian bump. Amplifying raw pixel differences results in overshoot and undershoot causing the signal to appear as pure white or pure black. In contrast, amplifying phase differences allows us to push the Gaussian bump much farther. At very high amplification levels, the different spatial scales of the bump break apart because the high frequency components cannot be pushed as far as the lower frequency components.

In Fig. 2.7, we show the effect of both methods on a video, which consists of independent and identically distributed (IID) Gaussian noise. Unlike the linear method that increases the power of the noise, the phase based method preserves the noise level preventing objectionable artifacts from making their way into the motion magnified output. For these reasons, we found that amplifying phase differences rather than pixel differences is a better approach for magnifying small motions.



(a) Input (b) Linear (c) Phase-based

Figure 2.7: Comparison between linear and phase-based Eulerian motion magnification in handling noise. (a) A frame in a sequence of IID noise. In both (b) and (c), the motion is amplified by a factor of 50*, where (b) amplifies changes linearly, while (c) uses the phase-based approach.*

■ 2.3.5 Bounds

As we move an image feature by phase-shifting each complex pyramid filter covering that feature, we eventually reach a limit beyond which we can't move the feature because of the limited spatial support of each pyramid filter (Fig. 2.5(b) and Fig. 2.8(1D Wavelets)). We can use the size of the windows to bound the amount by which we can motion magnify an image feature.

As an approximate analytic model of an image feature moved by the localized filters of the steerable pyramid, we consider the case of a single Dirac under uniform translation over time. As in Sec. 2.3.3, we assume it is moved by phase shifting Gabor filters, complex sinusoids modulated by a Gaussian window function. As the Dirac is phase-shifted, it is attenuated by the Gaussian window of the Gabor filter. Therefore, we bound the maximum phase shift such that the Dirac is only attenuated by a small amount.

As described in Eq. 2.17, a Gabor filter is given by

$$e^{-\frac{x^2}{2\sigma^2}} e^{-i\omega x}, \tag{2.22}$$

where σ is the standard deviation of the Gaussian window and ω is the peak frequency the filter corresponds to. Typically, σ depends on the frequency ω (self-similar wavelets). The impulse response of a Dirac function shifted by $\delta(t)$ pixels (not to be confused with the Dirac function) at time t is

$$S_\omega(x,t) = e^{-(x-\delta(t))^2/(2\sigma^2)} e^{2\pi i\omega(x-\delta(t))} \tag{2.23}$$

Note that the spatial Gaussian envelope (the left term on the RHS of Eq. 2.23) does not affect the phase.

We again assume $\delta(0) = 0$. To magnify the motions between the object at time t and time 0, we look at the phase difference between these times, yielding

$$B_\omega(x,t) = 2\pi\omega_0\delta(t). \tag{2.24}$$

Then, the magnified phase difference for magnifying the motion by α is

$$2\pi\omega_0\alpha\delta(t). \tag{2.25}$$

This phase difference corresponds to a shift of the Dirac by an additional $\alpha\delta(t)$ pixels. We need to bound the shift $\alpha\delta(t)$ such that the amplified shift approximates well the true shifted signal. We use one standard deviation of the Gaussian window as our bound. This maintains roughly 61% of the amplitude (Fig. 2.8 (1D Wavelets)), and so we have

$$\alpha\delta(t) < \sigma. \tag{2.26}$$

In the octave-bandwidth steerable pyramid of Portilla and Simoncelli [62] (Fig. 2.8a), there is approximately one period of the sinusoid under the Gaussian envelope. That is, $4\sigma \approx \frac{1}{\omega_0}$, which gives the bound $\alpha\delta(t) < \sigma = \frac{1}{4\omega_0}$. By equating the spatial wavelength

$\lambda = \frac{1}{\omega_0}$, we get

$$\alpha\delta(t) < \frac{\lambda}{4}. \tag{2.28}$$

From Eq. 2.28, we see that the motions of the low spatial frequencies can be magnified more than those of the high spatial frequencies. Indeed, from Eq. 2.24, phase changes between frames will be much greater for the high frequency components than for the low frequency components. While derived for an impulse image feature moved by Gabor filters, we find the bound (and its extension below for sub-octave bandwidth pyramids) to be valid for both synthetic examples (Fig. 2.6) and natural videos (Fig. 2.1, Fig. 2.8).

Exceeding the bound in Eq. 2.29 manifests as artifacts or blur, as not all image pyramid components are present in their proper ratios to reconstruct the desired translated feature. This is illustrated in Fig. 2.6b, in which a Gaussian function magnified using our approach breaks up.

■ 2.3.6 Sub-octave Bandwidth Pyramids

We see, therefore, that the bound in Eq. 2.28 is directly related to the spatial support of the filters. The smaller the filters in the frequency domain the larger their support is in the spatial domain, which allows us to shift the signals underneath their windows further. In the limit of a having a filter for every frequency band, the representation becomes equivalent to the Fourier transform and motion magnification is achieved via the shift theorem. However, we then lose the ability to measure or synthesize any spatial variation in the amount of motion. We found a good compromise between localization and magnification ability when using pyramid filters about two times as wide (in the sinusoidally varying spatial direction) as those described in Portilla and Simoncelli [62]. They specify their steerable pyramid filters as being self-similar and

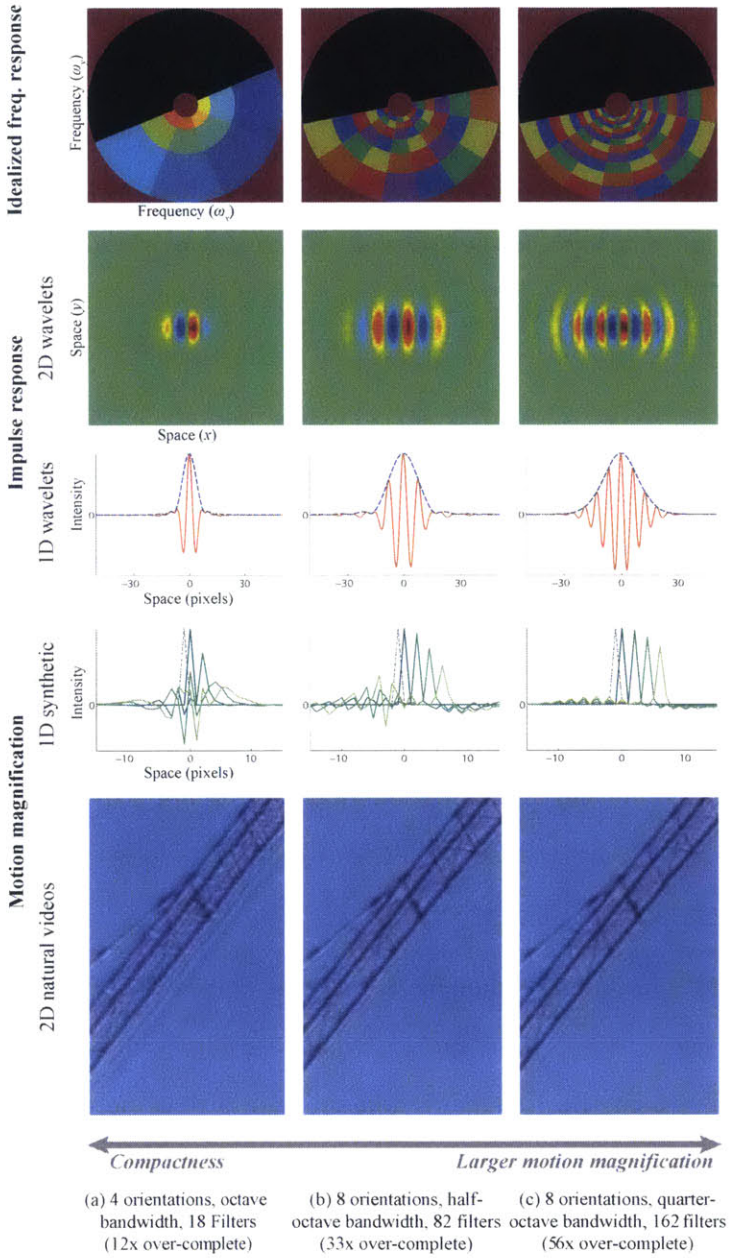


Figure 2.8: A comparison between octave and sub-octave bandwidth pyramids for motion magnification. Each color in the idealized frequency response represents a different filter. (a) The original steerable pyramid of Portilla and Simoncelli [62]. This pyramid has octave bandwidth filters and four orientations. The impulse response of the filters is narrow (rows $2-3$), which reduces the maximum magnification possible (rows $4-5$). (b-c) Pyramid representations with two and four filters per octave, respectively. These representations are more over-complete, but support larger magnification factors.

having octave bandwidth (Fig. 2.8a), and we extend their representation to sub-octave bandwidth pyramids (Fig. 2.8b-c).

A simple way to accomplish this is to scale the filters in log space. This method works well for a half-octave bandwidth pyramid. In this case, there are 2 periods under the Gaussian envelope of the wavelet. Thus, $4\sigma \approx \frac{2}{\omega_0}$, and the bound on the amplification (Eq. 2.28) becomes

$$\alpha\delta(t) < \frac{\lambda}{2}. \tag{2.29}$$

This bound improves over the one derived in Wu et al. [85] using a Taylor series approximation by a factor of 4.

There is a trade-off between the compactness of the representation and the amount of motion-magnification we can achieve. The 4-orientation, octave-bandwidth pyramid of Portilla and Simoncelli (Fig. 2.8a) is over-complete by a factor of 12 (each orientation contributes a real and imaginary part), and can process several frames per second, but limits the amount of motion-magnification that can be applied. On the other hand, an 8-orientation half-octave pyramid (Fig. 2.8b) supports larger amplification, but is over-complete by a factor of 33.

Improved Radial Windowing Function for Sub-octave Bandwidth Pyramids At a larger number of filters per octave (≥ 3 in our experiments), the above scheme produces filters which are very sharp in the frequency domain and have noticeable ringing artifacts (shown in the 1D wavelet plot of Fig. 2.8b).

They define their filters in terms of independent radial and angular windowing functions. For quarter-octave and larger pyramids, we leave the angular windowing function unchanged and propose a different radial windowing function, given by

$$\cos^6(\log_2(r))I_{[-\pi/2,\pi/2]}(\log_2(r)). \tag{2.30}$$

This function has two nice properties: (a) it is smoother, more similar to a Gaussian, and does not introduce ringing in the primal domain, and (b) squared copies scaled by a power of $\frac{\pi}{7}$ sum to a constant factor, so that the transform is invertible and we get perfect reconstruction (Eq. 2.16). An example quarter-octave pyramid generated with this windowing function in shown in Fig. 2.8c.

■ 2.4 Amplifying the Right Signal

Maximizing the signal-to-noise ratio of the local phase changes we amplify is the key to good performance. We improve SNR by temporally and spatially filtering the variations to remove components that correspond to noise and keep those that correspond to signal. The temporal filtering also gives a way to isolate a signal of interest as different motions often occur at different temporal frequencies. A baby's squirming might be at a lower temporal frequency than her breathing.

Narrowband linear filters provide a good way to improve signal-to-noise ratios for motions that occur in a narrow range of frequencies, such as respiration and vibrations. These filters can also be used to isolate motions in an object that correspond to different frequencies. For example, a pipe vibrates at a preferred set of modal frequencies, each of which has a different spatial pattern of vibration. We can use video magnification to reveal these *spatial* patterns by amplifying the motions only corresponding to a range of *temporal* frequencies. A single frame from each motion magnified video is shown in Fig. 2.9, along with the theoretically expected shape [79]. They match closely.

In addition to temporal filtering, spatially smoothing the signal almost always improved signal-to-noise ratios of the motion signal. This is because objects tend to move coherently in local image patches and any deviation from this is likely noise. Because the phase signal is more reliable when the amplitude of the complex steerable pyramid

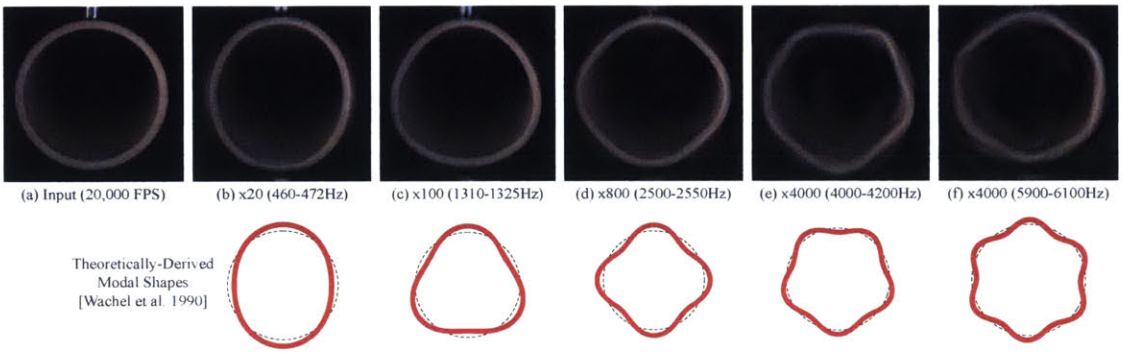


Figure 2.9: Isolating different types of spatial *motions by* temporal *filtering. A pipe was struck with a hammer and a frame from the input high-speed video is shown (a). The motions at several frequencies were magnified to isolate different modal shapes of the pipe. In b-f, a frame is shown from each of the motion magnified videos showing the modal shapes. Below, the the theoretically-derived modal shapes are shown in red overlayed, for comparison, over a perfect circle in dotted black. (Video and idea courtesy of Justin Chen.)*

coefficients is higher (Fig. B.1), we perform an amplitude-weighted Gaussian blur:

$$\frac{((\Delta\phi)A) * K_\rho}{A * K_\rho} \tag{2.31}$$

where K_ρ is a Gaussian kernel given by $\exp(-\frac{x^2+y^2}{2\rho^2})$ and the indices of A and ϕ have been suppressed for readability. We applied this processing to all of our motion magnification videos with ρ equal to 2 pixels in each level.

Because oversmoothing can turn even white noise into a nice-looking signal, it becomes reasonable to ask whether the motions we are amplifying are indeed real. We have done many experiments comparing the visual motion signal with that recorded by an accelerometer or laser vibrometer and the signals always match up, validating that the motions are indeed real [11–13, 81]. These experiments are discussed more in Chapter 4. In addition, there are many videos where the motion is spatially coherent at a scale beyond that imposed by spatial smoothing (e.g. the pipes in Fig. 2.9) providing further evidence for the correctness of the amplified videos.

Finally, we can only recover motions that occur at frequencies less than the Nyquist

frequency of the capturing device. If the motions occurs too quickly, only an aliased version of them gets amplified. In the special case that the motions occur at a single temporal frequency, aliasing can actually be useful. It makes such motions appear slower, which makes it possible to visualize fast vibrations like those in a resonating wine glass in real-time (Fig. 3.10). However, in general we will not be able to recover a meaningful signal if the frames are temporally undersampled.

Pipeline with Filtering To clarify the role of spatial and temporal filtering in our pipeline, we present it again with the example of a real-image (Fig. 2.10). We compute the local phase changes over time at every spatial scale and orientation of a steerable pyramid (Fig. 2.10a). Then, we temporally bandpass these phases to isolate specific temporal frequencies relevant to a given application and remove any temporal DC component (Fig. 2.10). To synthesize magnified motion, we multiply the bandpassed phases by an amplification factor α. We then use these amplified phase differences to magnify (or attenuate) the motion in the sequence by modifying the phases of each coefficient by this amount for each frame (Fig. 2.10d). Finally, the modified complex steerable pyramid is collapsed to produce the motion magnified video (Fig. 2.10e).

■ 2.5 Results

Our algorithm allows users to see small motions without excessive noise or computational cost, as well as remove motions that may distract from an underlying phenomena of interest. We show several applications of our algorithm in this section. Please refer to the original project websites for the full video sequences and results (`http://people.csail.mit.edu/nwadhwa/phase-video/`).

Unless mentioned otherwise, our processing was done using a half-octave, complex steerable pyramid with eight orientations. We found this to be a good tradeoff between speed and quality in our results. We computed the filter responses in the frequency

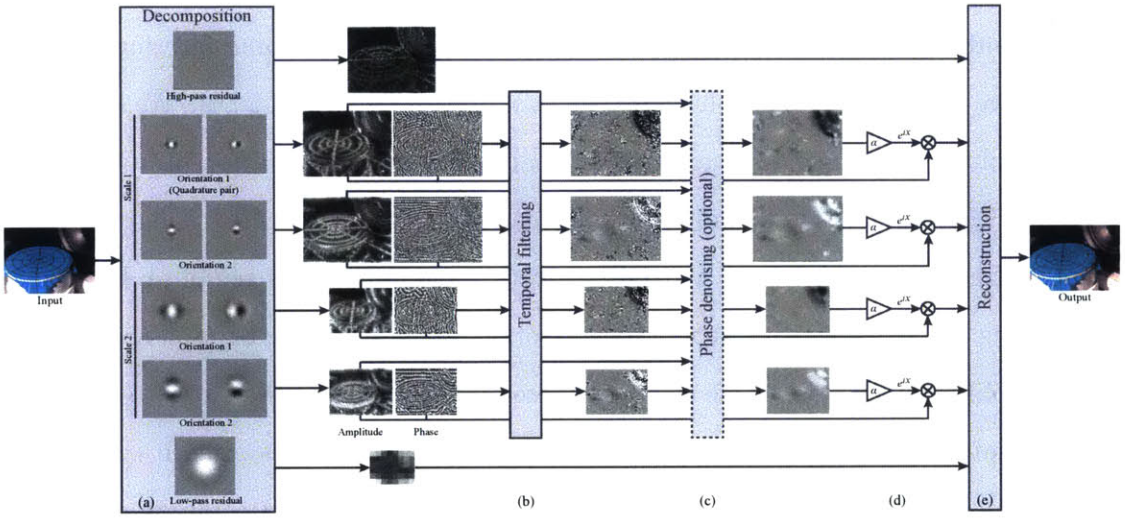


Figure 2.10: Our phase-based approach manipulates motion in videos by analyzing the signals of local phase over time in different spatial scales and orientations. We use complex steerable pyramids to decompose the video and separate the amplitude of the local wavelets from their phase (a). We then temporally filter the phases independently at at each location, orientation and scale (b). Optionally, we apply amplitude-weighted spatial smoothing (c) to increase the phase SNR, which we empirically found to improve the results. We then amplify or attenuate the temporally-bandpassed phases (d), and reconstruct the video (e). This complex steerable pyramid shown has two scales and two orientations (the relative difference in size between the pyramid levels is smaller in this figure for clarity of the visualization).

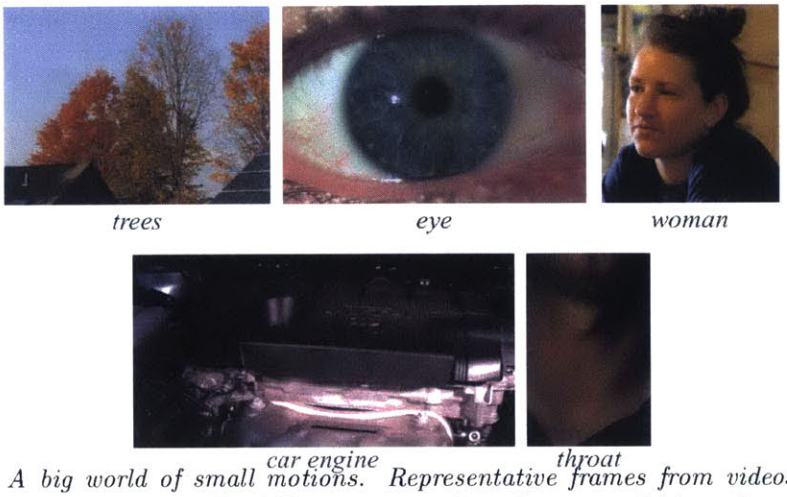


Figure 2.11: A big world of small motions. Representative frames from videos in which we amplify imperceptible motions. The full sequences and results are available on the project website.

domain. The processing was done in YIQ color space and only the luminance (Y) channel was processed.

■ 2.5.1 A Big World of Small Motions

The world is full of subtle and small motions that are invisible to the naked eye. Our phase-based approach allows pushing motion magnification further than before, to reveal imperceptible phenomena, not previously visualized, in clarity and detail.

In *eye* (Fig. 2.11), we were able to magnify subtle, involuntary, low amplitude (10-400 micron) movements in the human eye and head such as *microsaccades* [64]. This video was taken with a high speed camera at 500 Hz. A one second (500 frames) sequence was processed with an ideal bandpass filter with passband between $30-50$ Hz and the motions were amplified 150x. A spatial mask was applied to the phase shifts to emphasize the motion around the iris. Such a detection system may have medical applications, as the frequency content of ocular microtremor was shown to have clinical significance [7].

Structures are design to sway in the wind, but their motion is often invisible. In *crane*, we took a video of a construction crane on a uniform background during a windy

day. In the original video, the superstructure does not appear to move, however when amplifying low-frequency motions in the video within $0.2 - 0.4$ Hz 150x, the swaying of the crane's mask and undulation of its hook become apparent.

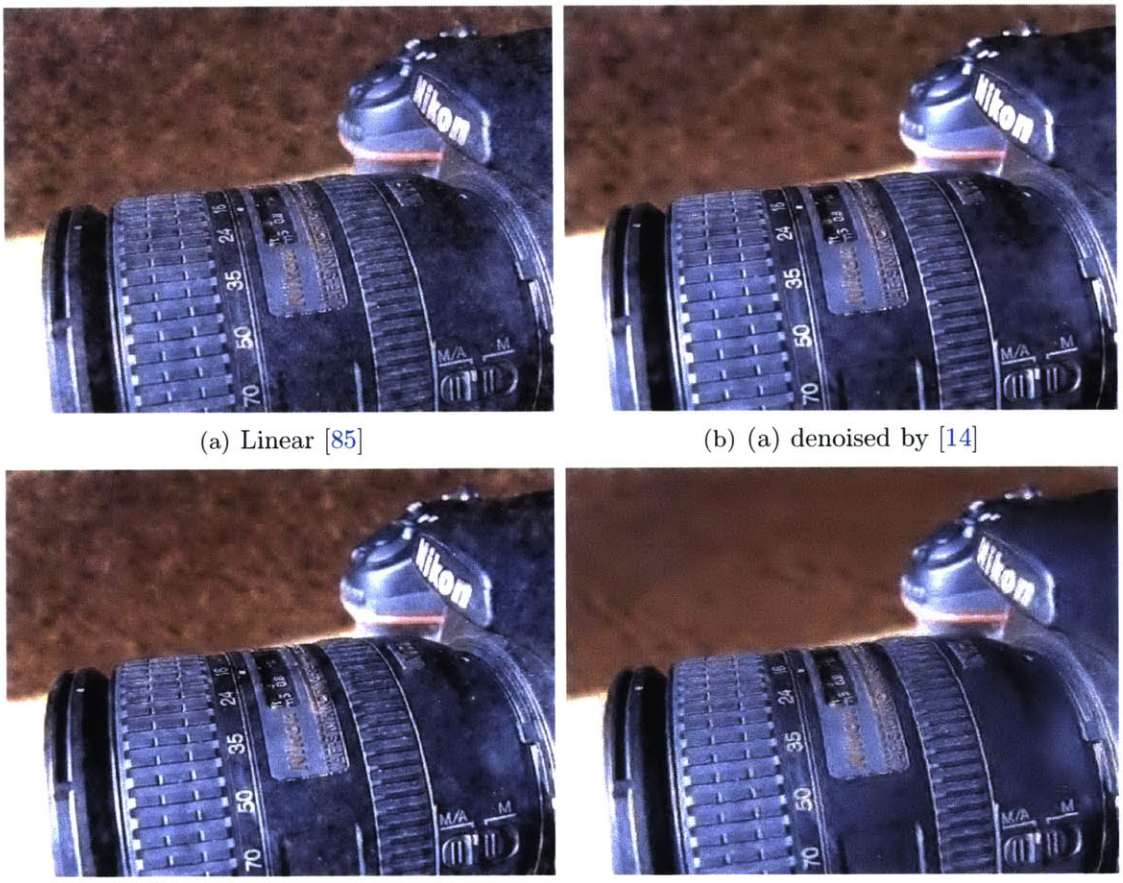
Trees and *woman* (Fig. 2.11) demonstrate ordinary videos also contain changes at different frequencies over time that we cannot normally perceive. In *trees*, motions of lower temporal frequency correspond to larger structures (heavy branches), while motions of higher temporal frequency correspond to smaller structures (leaves). A simple interface allows the user to *sweep* through the frequency domain and examine temporal phenomena in a simple and intuitive manner.

We can also use our technique to reveal information about mechanical parts in running cars. In *car engine*, we filmed a car engine in an idling car. We amplified motions corresponding to twice the engine's RPM as measured by the dashboard tachometer revealing the engine's jumping with every combustion within the engine.

Finally, when a person sings or talks, they use their larynx (voice box) to modulate air within the the throat to produce certain pitches and sounds. In *throat*, we filmed a person humming at around 110Hz. We amplified motions corresponding to this passband and revealed hidden motions on the person's neck.

■ 2.5.2 Comparison with Linear Eulerian Video Magnification

The main differences between the phase-based approach and the linear approach are summarized in Table 2.1. In particular, the new method supports larger amplification factors and gives a fundamentally better way of handling noise for Eulerian motion magnification. To demonstrate that, we compared the results from this work with those from Wu et al. [85]. Several comparisons are available in Fig. 2.1 and the project website. To illustrate that shifting phases is better than directly modifying pixel intensities, we did not spatially-smooth the phase signal in these comparisons.



(a) Linear [85]

(b) (a) denoised by [14]

(c) (a) denoised by [51]

(d) Phase-based

Figure 2.12: Comparison of our result on the camera *sequence (d) with the result of Wu et al. [85] (a), denoised by two state-of-the-art video denoising algorithms: VBM3D [14] (b) and motion-based denoising by Liu and Freeman [51] (c). The denoising algorithms cannot deal with the medium frequency noise, and are computationally intensive. The full videos and similar comparisons on other sequences are available in the supplementary material.*

On *all* the sequences we tested, we found the proposed approach to perform better. In particular, the magnified motions in the phase-based results (e.g. the respiratory motions of the baby and the vibrations of the guitar strings) appear crisper, and contain significantly fewer artifacts and noise.

We also compared the phase-based results with noise removal processing not suggested in the Wu et al. paper: preceding and following the linear magnification processing by video denoising. We tested several denoising algorithms, namely NL-means [8], VBM3D [14], and the recent motion-based denoising algorithm by Liu and Freeman [51]. We tuned the denoising methods so as to produce the best result on each sequence. We achieved the overall best performance with VBM3D applied to the motion-magnified video (comparisons with all the denoising methods in pre- and post-processing are available in the supplementary material). We found that in some cases (e.g. *guitar*) denoising the video before magnification in fact kills the low-amplitude motion signal we are after. For the low-noise *baby* and *guitar* sequences, the denoised results were visually comparable to that of the phase-based method, although achieved at a higher computational cost, 17 times slower. For the higher-noise *camera* and *eye* sequences, the denoised Wu et al. result looks significantly worse than the phase-based results, as the denoising algorithms cannot do much with the medium frequency noise (Fig. 2.12).

	Linear [85]	**Phase-based (This chapter)**
Decomposition	Laplacian pyramid	Complex steerable pyramid
Over-complete	4/3	$2k/(1-2^{-2/n})$
Exact for	Linear ramps	Sinusoids
Bounds	$(1+\alpha)\delta(t) < \lambda/8$	$\alpha\delta(t) < \lambda n/4$
Noise	Magnified	Translated

Table 2.1: The main differences between the linear approximation of Wu et al. [85] and our approach for motion magnification. The representation size is given as a factor of the original frame size, where k represents the number of orientation bands and n represents the number of filters per octave for each orientation.

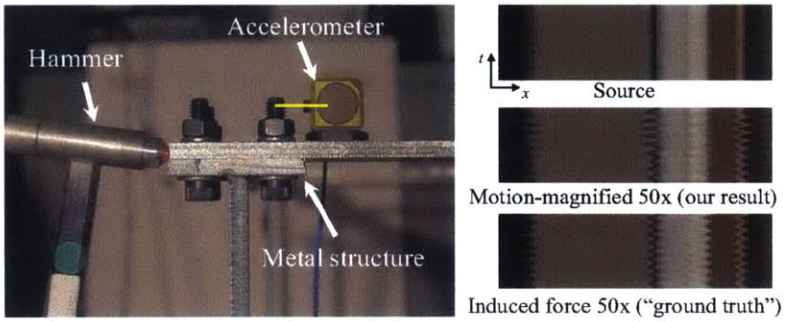
2.5.3 Controlled Experiments

At the miniature scales of motion we are after, one might ask: are the signals we pick out and amplify real (the actual motion signals in the scene)? Would our magnified motions resemble the motions in the scene had they actually been actually larger? To answer these questions, we conducted two controlled experiments.[1] In the first, we recorded ground truth motion data along with a (natural) video (*structure*, Fig. 2.13). We induced small motions in a metal structure, and affixed an accelerometer to it to capture its vibrations. To induce the motion we used an impact hammer with a sensor at its tip allowing us to record the exact amount of force applied. We then recorded the structure using a standard DSLR video camera at 60 frames per second, along with the accelerometer and impact hammer data. We applied our transform to every frame and recorded the phase changes between the Nth frame and the first frame in one level of the pyramid oriented in the direction of the motion for a salient region of pixels near the accelerometer. These phase changes corresponded to displacement. To recover acceleration, we took a second derivative of Gaussian filter. Once scaled and aligned, the resulting signal matched the data from the accelerometer very closely 2.13(c). We also took two different sequences of the structure, one in which the amplitude of the oscillatory motion was 0.1 pixels and another in which it was 5 pixels (50x larger, from a harder hammer hit). We magnified the former 50 times and found the result to be visually comparable to the latter (Fig. 2.13(b)).

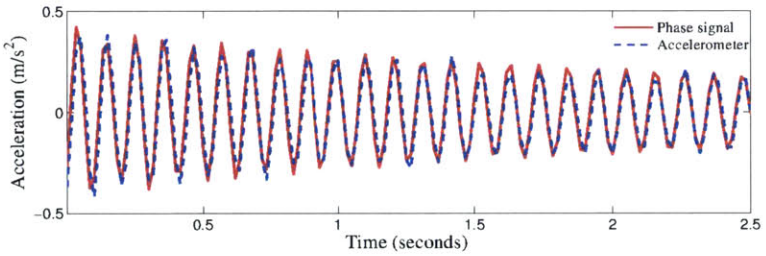
2.5.4 Motion Attenuation

Our phase-based formulation also lends itself naturally to the attenuation of motions in videos, which allows us to remove low-amplitude, short-term motions while larger amplitude motions continue to pass through. Motion attenuation is achieved by setting

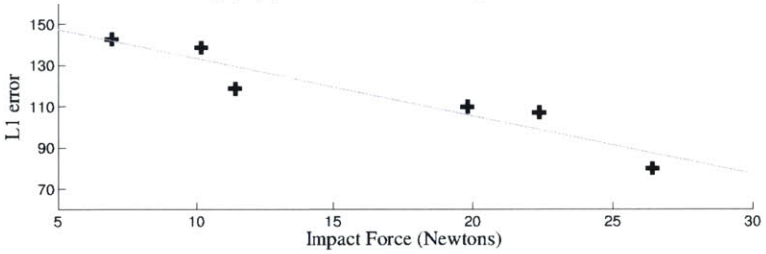
[1] Justin G. Chen helped perform these experiments.



(a) Experimental setup (b) Spatiotemporal slices



(a) (c) Acceleration comparison



(b) (d) Error vs. force

Figure 2.13: A controlled motion magnification experiment to verify our framework. (a) A hammer strikes a metal structures which then moves with a damped oscillatory motion. (b) A sequence with oscillatory motion of amplitude 0.1 *pixels is magnified* 50 *times using our algorithm and compared to a sequence with oscillatory motion of amplitude* 5 *pixels (*50 *times the amplitude). (c) A comparison of acceleration extracted from the video with the accelerometer recording. (d) The error in the motion signal we extract from the video, measured as in (c), as function of the impact force. Our motion signal is more accurate as the motions in the scene get larger.*

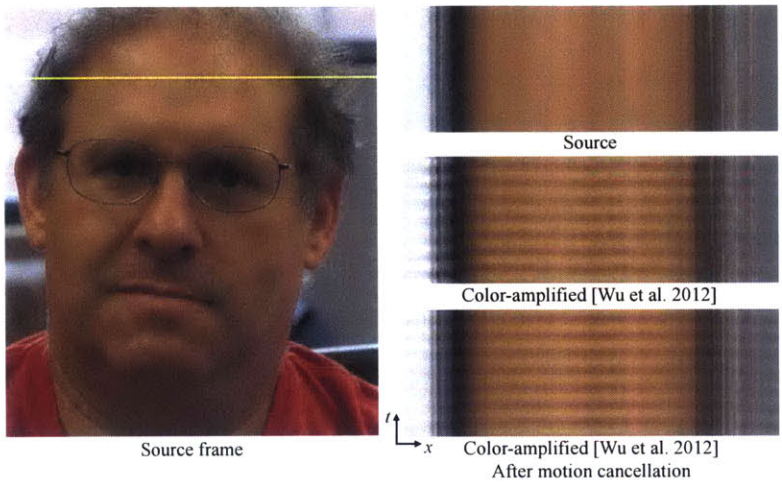


Figure 2.14: Motion attenuation stabilizes unwanted head motions that would otherwise be exaggerated by color amplification. The full sequence is available in the supplemental video.

the amplification factor α to a negative value in the range $[-1, 0)$, where $\alpha = -1$ zeros-out all the phase changes over time within the desired frequency band, effectively canceling out the motions within that band. The result is not the same as a constant frame as the coefficient amplitudes are still evolving over time. This is similar to motion denoising [66] and video de-animation [2], but can be done efficiently in our approach (when the motions in the scene are small enough).

We apply motion attenuation for two applications: turbulence removal and color amplification (Fig. 2.14). Atmospheric turbulence can cause far-away objects to appear blurry and it is desirable to lessen its effect [63]. In a video of the moon moving through the night sky, it manifests as low-mid frequency jitters. To remove it, we pass a temporal window over the video (we used a window of 11 frames), transformed to our representation, and set the phases in each spatial scale and orientation of the center frame to the corresponding median phase of the transformed frames within the temporal window. This effectively *shifts* pixels in order to compensate for the turbulent motions.

Since the magnification method of Wu et al. [85] amplifies color changes and motions *jointly*, small motions of the face become much larger, visible when amplifying the color

changes corresponding to the pulse, which may not be desirable. By canceling the motions as a pre-process to their algorithm, we are able to remove those motions from their results (Fig. 2.14).

■ 2.6 Discussion and Limitations

Lagrangian approaches to motion magnification (e.g. [53]) are complementary to the Eulerian approaches proposed in this chapter. Such methods can amplify the motion in a video arbitrarily, but rely on accurate optical flow estimates, image segmentation, and inpainting. Such processing is difficult to do well and requires long computation times. In addition, Wu et al [85] showed (Section 5 and Appendix A in their paper) that for moderate magnification and noisy inputs, the Eulerian approach performs better than Lagrangian. The phase-based method significantly reduces the sensitivity to noise of Eulerian video magnification over that of Wu et al., as well as increases its supported range of amplification, which further expands the regime where it performs better than Lagrangian approaches. Since the main contribution of this chapter is in an improved Eulerian approach for motion processing, comparisons were done with the state-of-the-art Eulerian method.

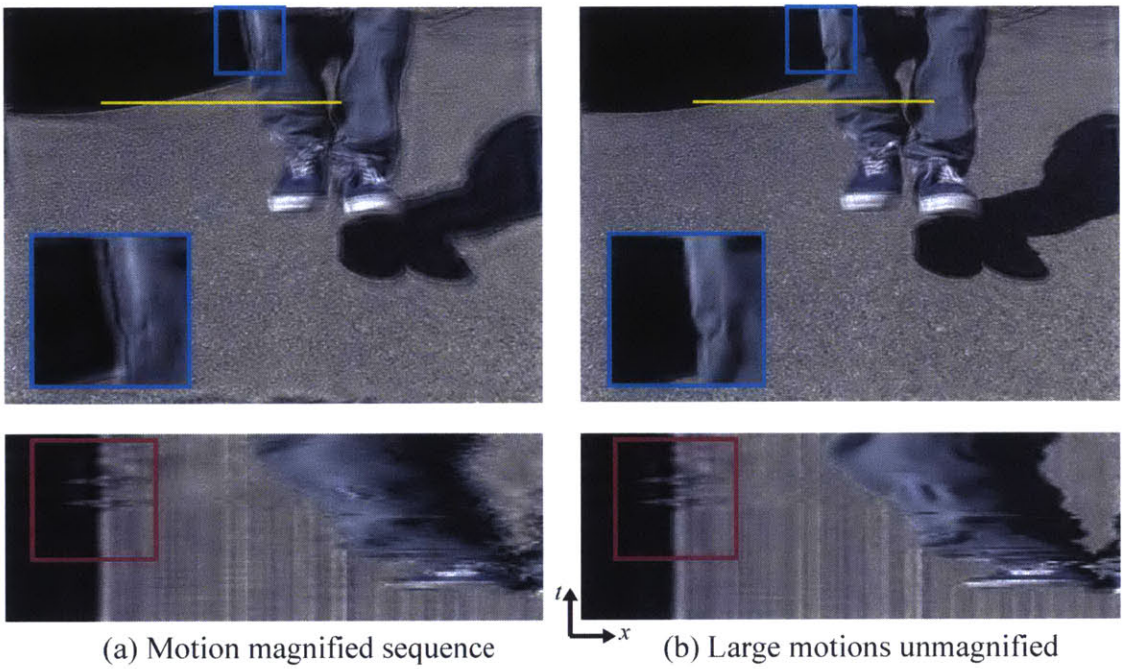
While the analysis of Wu et al. [85] is exact in the case of linear ramps in space, the phase-based approach is exact for sinusoidal waves (Fig. 2.3), since such signals contain only a single spatial frequency. However, both methods rely on spatial pyramids, where each level is band limited. We argue that such spatially bandpassed images are better approximated by sinusoidal waves than linear ramps.

Our half-octave bandwidth pyramid representation, in which the windowing function of the wavelets in the primal domain is larger, extends the magnification capability of Wu et al. [85] by a factor of 4, and pyramids with more filters per octave may improve on it by even larger factors. While this allows us to magnify motions further, the wavelets

are also more likely to span multiple motions as their support get larger, which may corrupt the phase signal and eventually lead to artifacts in the results. Currently, the user can select the desired representation based on the motions in the scene and the available computational resources. The *crane* sequence lends itself to improvements from a quarter-octave pyramid because the background is so smooth and in most of the scene, there is only one motion.

If the input video has large motions, than the bandpassed phase will not reflect the true motion in the scene and the motion magnified video will suffer from artifacts in the regions of large motion (Fig. 2.15(a)). To mitigate this, we can automatically detect regions where phase exceeds our bound (or some user-specified threshold) and set the amplification to be zero in these regions. To increase robustness, we spatiotemporally lowpass the absolute value of the phase and compare the result to a threshold to determine which regions have large motions.

Finally, for sequences in which the phase signal is noisy, parts of the image in the magnified video may appear to move incoherently. Understanding when this is the case and suppressing magnification in such cases is discussed in the Chapter 4.



(a) Motion magnified sequence (b) Large motions unmagnified

Figure 2.15: Motion magnification can cause artifacts (cyan insets and spatiotemporal time-slices) in regions of large motion such as those in this sequence of a boy jumping on a platform (a). We can automatically remove such artifacts by identifying regions where the phase change exceeds our bound or a user-specified threshold (b). When the boy hits the platform, the time slice (purple highlights) shows that the subtle motions due to impact with the platform are magnified in both cases.

Chapter 3

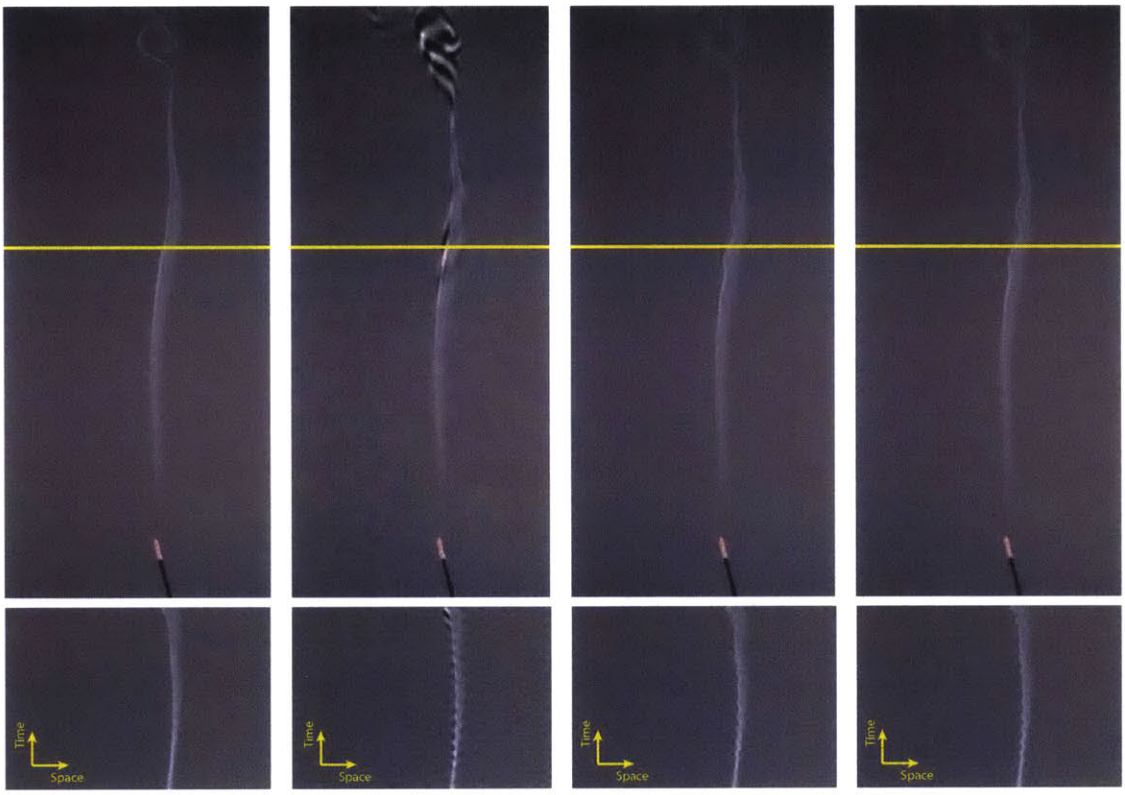
Riesz Pyramids for Fast Phase-Based Motion Magnification

In the previous chapter, we introduced a new method of amplifying tiny motions in videos. It is capable of revealing a wide variety of phenomena, but it is slow to compute because of the overcompleteness of the representation used. In this chapter, we show how a change in representation can make motion magnification capable of running in real-time with only a slight loss in quality.

■ 3.1 Introduction

Manipulating the local phase in coefficients of a complex steerable pyramid decomposition of an image sequence is an effective, robust method of amplifying small motions in video (Chapter 2), but complex steerable pyramids are very overcomplete (21 times) and costly to construct, requiring either a large number of filter taps or a frequency domain construction where care must be taken to avoid spatial wrap-around artifacts [62, 71]. The overcompleteness and high cost of implementing the complex steerable pyramid make current phase-based video magnification slow to compute.

In this chapter, we present a new image pyramid representation, the Riesz Pyramid, that is suitable for Eulerian phase-based video magnification, but is much less overcomplete than the complex steerable pyramid. Our new representation produces



(a) Input | (b) Linear (Wu et al. 2012) 11.7 seconds | (c) CSP (Wadhwa et al. 2013) **92.3 seconds** | (d) Riesz Pyramid (this paper) **25.7 seconds**

Figure 3.1: Motion magnification of sinusoidal instabilities in fluid flow during the transition from laminar flow to turbulent flow. The input (a) is motion-magnified using the linear method of Wu et al. [85] (b) and two phase-based methods, first with an eight orientation octave-bandwidth complex steerable pyramid [81] (c), and second with our new Riesz pyramid (d). The quality of the video produced using our new representation (d) is comparable to that produced using the complex steerable pyramid method (c), but ***is approximately four times faster to compute****. Frames and slices in time along the yellow line from the input and processed sequences are shown. Notice that both (c) and (d) do not have the intensity clipping artifacts and limited amplification of (b). The running time of each method is shown under its caption, based on a MATLAB implementation.*

motion-magnified videos of comparable quality to those produced using a complex steerable pyramid, but the videos can be processed in one quarter of the time, making it more suitable for real-time or online processing (Figure 3.1).

The Riesz pyramid is constructed by first breaking the input image into non-oriented sub-bands using an efficient, invertible replacement for the Laplacian pyramid, and then taking an approximate Riesz transform of each band [9, 25]. This processing is done entirely in the spatial domain, which gives an easy way of avoiding the spatial wrap-around artifacts present in the frequency domain implementation of the eight-orientation complex steerable pyramid used in the previous chapter. Building and collapsing the Riesz pyramid is efficiently implemented because of shared computation between bands, symmetry of the filters, and because the Riesz transform is approximated by two three-tap finite difference filters. Concretely, it uses less than half the number of real multiplies required for the spatial domain implementation of the *two*-orientation real steerable pyramid proposed by Simoncelli and Freeman [71] (this is the smallest possible real steerable pyramid, and computing the imaginary part of the pyramid would require additional processing).

The key insight into why our new representation can be used for motion magnification is that the Riesz transform is a steerable Hilbert transformer and allows us to compute a quadrature pair that is 90 degrees out of phase with respect to the dominant orientation at every pixel. This allows us to phase-shift and translate image features only in the direction of the *dominant orientation* at every pixel rather than a *sampling of orientations* like in the complex steerable pyramid. Felsberg and Sommer [25] introduced the Riesz transform to the signal processing community and Unser et al. extended it to a multiresolution framework [77]. Our representation extends Unser et al. Their framework is not focused on speed and is implemented entirely in the frequency domain, while the Riesz pyramid we propose is implemented in the spatial domain. In

addition, we gain further speedups by approximating the Riesz transform using two three-tap finite difference filters, whereas Unser et al. opt to use the ideal frequency domain version of the Riesz transform, which is slower to compute.

In summary, we present a new representation that can be used for video magnification that is (a) less overcomplete than even a two-orientation octave-bandwidth complex steerable pyramid, (b) is implemented in the spatial domain, which gives an easy way to avoid spatial wrap-around artifacts associated with frequency domain implementations of filters, (c) is implemented with efficient, compact linear filters, and (d) supports real-time phase-based video magnification. We present comparisons with both the linear and phase-based video magnification techniques presented in the previous chapter. We also provide a real-time implementation.

3.2 Background

3.2.1 Local Phase and Quadrature Pairs

Phase-based video magnification relies on the ability to manipulate the local (spatial) phase of image sub-bands. The local phase can be used to edit local motions in a manner analogous to shifting an image using global phase via the Fourier shift theorem.

The local phase of a one-dimensional image sub-band is computed by first computing the sub-band's quadrature pair, a 90 degree phase-shifted version of the sub-band related to it by the Hilbert transform. The sub-band and its quadrature pair form the real and imaginary part of a complex number, whose argument is the local phase. We can manipulate this quantity to shift the sub-band arbitrarily. For example, the quadrature pair of $\cos(x)$ is $\sin(x)$, its local phase is x and

$$\cos(x - \phi) = \text{Real}(e^{-i\phi}(\cos(x) + i\sin(x))) \tag{3.1}$$

is an arbitrary translation of $\cos(x)$.

Two dimensional images can be analyzed in this way using the complex steerable pyramid, an invertible filter bank, which first breaks the image into sub-bands corresponding to different scales and orientations to form the real part of the pyramid. Then, the imaginary part of the pyramid is formed by taking the Hilbert transform of each sub-band along its orientation. The complex steerable pyramid must break the image into at least two orientations because the Hilbert transform is fundamentally a one dimensional transform and in two dimensions is only well-defined with respect to a preferred orientation. The fact that there must be multiple orientations is the reason why the complex steerable pyramid is so overcomplete.

■ 3.2.2 Riesz Transform

The Riesz transform is the natural rotation-invariant, two-dimensional generalization of the one-dimensional Hilbert transform [25]. It can be viewed as a steerable Hilbert transformer that gives a way to compute a quadrature pair of a non-oriented image sub-band that is 90 degrees phase-shifted with respect to the dominant orientation at every point. That is, it allows for phase analysis of non-oriented image sub-bands. The Riesz transform has been applied in the past for image processing applications such as segmentation of ultrasound images [4] and demodulation of fringe patterns in interferometric images [45].

Following Unser et al. [77], in two dimensions, the Riesz transform is a pair of filters with transfer functions

$$-i\frac{\omega_x}{\|\vec{\omega}\|}, -i\frac{\omega_y}{\|\vec{\omega}\|}. \tag{3.2}$$

If they are applied to the image sub-band I in Fig. 3.2(a), the result is the pair of filter responses, (R_1, R_2) in Fig. 3.2(b-c). The input I and Riesz transform (R_1, R_2) together form a triple (the *monogenic signal* [25]) that can be converted to spherical coordinates

to yield the local amplitude A, local orientation θ and local phase ϕ using the equations

$$I = A\cos(\phi), R_1 = A\sin(\phi)\cos(\theta), R_2 = A\sin(\phi)\sin(\theta). \tag{3.3}$$

The Riesz transform can be steered to an arbitrary orientation, θ_0, by multiplication by a rotation matrix

$$\begin{pmatrix} \cos(\theta_0) & \sin(\theta_0) \\ -\sin(\theta_0) & \cos(\theta_0) \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}. \tag{3.4}$$

When the Riesz transform is steered to the local dominant orientation θ (Fig. 3.2(d)), the result is a pair whose first component Q is

$$Q = A\sin(\phi), \tag{3.5}$$

a quadrature pair of the input signal that is 90 degrees phase-shifted with respect to the local dominant orientation (Fig. 3.2(e)). The local phase ϕ (Fig. 3.2(f)) can be viewed as the phase of the complex number

$$Ae^{i\phi} = I + iQ \tag{3.6}$$

whose real and imaginary part are the input sub-band and quadrature pair.

■ 3.2.3 Quaternion Representation of the Riesz Transform

In the formulation described in the previous section, we explained how the Riesz transform is a steerable Hilbert transformer. However, the phase defined in that section suffers from a sign ambiguity (Eq. 3.6). Specifically, both ϕ, θ and $-\phi, \theta + \pi$ are equally valid values for the phase and orientation (both satisfy Eq. 3.4). This can cause problems when the Riesz transform is used in video magnification (Fig. 3.3). To avoid this

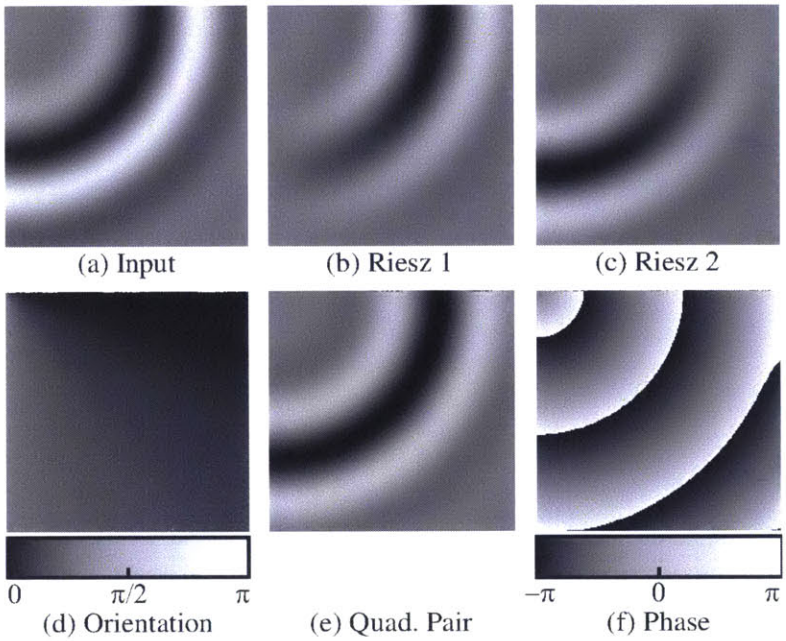


Figure 3.2: The input image sub-band (a), its Riesz transform (b-c) and the orientation (d), quadrature pair (e) and phase (f).

ambiguity, we use a formulation which treats the triple (I, R_1, R_2) as the real, i and j components of a quaternion. Then, instead of filtering and amplifying phase, we filter and amplify the quaternionic generalization of phase,

$$\phi\cos(\theta), \phi\sin(\theta). \tag{3.7}$$

These quantities are invariant to whether the phase and orientation are ϕ, θ or $-\phi, \theta+\pi$.

Quaternions Quaternions are a generalization of the complex numbers, in which there are three imaginary units, denoted i, j and k, so that each quaternion is characterized by four numbers, one real and three imaginary. Quaternion multiplication is associative and distributive with addition and can therefore be fully defined by the following

(a) Input

(b) Phase difference

(c) Amplified with (b)

(d) $\phi\cos(\theta)$ difference

(e) $\phi\sin(\theta)$ difference

(f) Amplified with (d,e)

Figure 3.3: The motion between the input (a) and a copy shifted to the left by one half pixel is magnified without and with the quaternion representation of the Riesz pyramid. First, the phase difference of ϕ (b) is spatially denoised and then used to magnify the second frame (c). In the bottom row, the difference in the quantities $\phi cos()$ and $\phi sin()$ (d-e) are spatially denoised and then used to amplify the second frame (f). In (b,d,e), low amplitude regions are masked in yellow, middle gray corresponds to a difference of zero and only a single sub-band is shown.

property of the imaginary units:

$$-1 = i^2 = j^2 = k^2 = ijk. \tag{3.8}$$

Specifically, multiplication is given by

$$\begin{aligned}(q_1 + iq_2 + jq_3 + kq_4)(r_1 + ir_2 + jr_3 + kr_4) \quad &= \\ (q_1r_1 - q_2r_2 - q_3r_3 - q_4r_4) \quad &+ \\ i(q_1r_2 + q_2r_1 + q_3r_4 - q_4r_3) \quad &+ \\ j(q_1r_3 - q_2r_4 + q_3r_1 + q_4r_2) \quad &+ \\ k(q_1r_4 + q_2r_3 - q_3r_2 + q_4r_1).\end{aligned} \tag{3.9}$$

Note that multiplication is noncommutative.

For a quaternion $\mathbf{q} = q_1 + iq_2 + jq_3 + kq_4$, its conjugate $\mathbf{q}^*$, norm $\|\mathbf{q}\|$ and inverse $\mathbf{q}^{-1}$ are defined as

$$\mathbf{q}^* = q_1 - iq_2 - jq_3 - kq_4, \tag{3.10}$$

$$\|\mathbf{q}\| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}, \tag{3.11}$$

$$\mathbf{q}^{-1} = \mathbf{q}^*/\|\mathbf{q}\|^2, \tag{3.12}$$

where the third definition follows from the first two. The exponential of a quaternion $\mathbf{q} = q_1 + \mathbf{v}$ (where $\mathbf{v} = iq_2 + jq_3 + kq_4$) is defined by its power series,

$$e^{\mathbf{q}} = \sum_{n=0}^{\infty} \frac{\mathbf{q}^n}{n!} = e^{q_1}\left(\cos(\|\mathbf{v}\|) + \frac{\mathbf{v}}{\|\mathbf{v}\|}\sin(\|\mathbf{v}\|)\right). \tag{3.13}$$

The inverse of this function is

$$\log(\mathbf{q}) = \log(\|\mathbf{q}\|) + \frac{\mathbf{v}}{\|\mathbf{v}\|}\operatorname{acos}\left(\frac{q_1}{\|\mathbf{q}\|}\right). \tag{3.14}$$

In the case of a unit quaternion, where $\|\mathbf{q}\| = 1$, this simplifies to

$$\frac{\mathbf{v}}{\|\mathbf{v}\|}\text{acos}(q_1). \tag{3.15}$$

This is the imaginary part of the logarithm of a quaternion and is analogous to the phase of a complex number which is computed in the same way ($Ae^{i\phi}$ has phase ϕ equal to $\text{imag}(\log(Ae^{i\phi}))$. We refer to Eq. 3.15 as the *quaternionic phase* to distinguish it from the local phase ϕ in the non-quaternionic formulation of the Riesz pyramid.

Now that we have defined quaternions, we can represent the triple (I, R_1, R_2) as a quaternion $\mathbf{r}$ with the original subband I being the real part and the two Riesz transform components (R_1, R_2) being the imaginary i and j components of the quaternion

$$\mathbf{r} = I + iR_1 + jR_2 \tag{3.16}$$

or if we use Eq. 3.3, we can write this as

$$\mathbf{r} = A\cos(\phi) + iA\sin(\phi)\cos(\theta) + jA\sin(\phi)\sin(\theta). \tag{3.17}$$

Rather then solving for the local amplitude A, phase ϕ and orientation θ, we instead use the quaternionic phase we defined earlier. That is, we can express the amplitude and quaternionic phase (Fig. 3.7(c)) as

$$\begin{aligned} A &= \|\mathbf{r}\| \\ i\phi\cos(\theta) + j\phi\sin(\theta) &= \log(\mathbf{r}/\|\mathbf{r}\|) \end{aligned} \tag{3.18}$$

The second quantity is computed by applying Eq. 3.15 to the specific case of normalized Riesz pyramid coefficients and is invariant to whether the local phase and orientation are ϕ and θ or the antipode $-\phi$ and $\theta + \pi$. In the remainder of this chapter, we use this

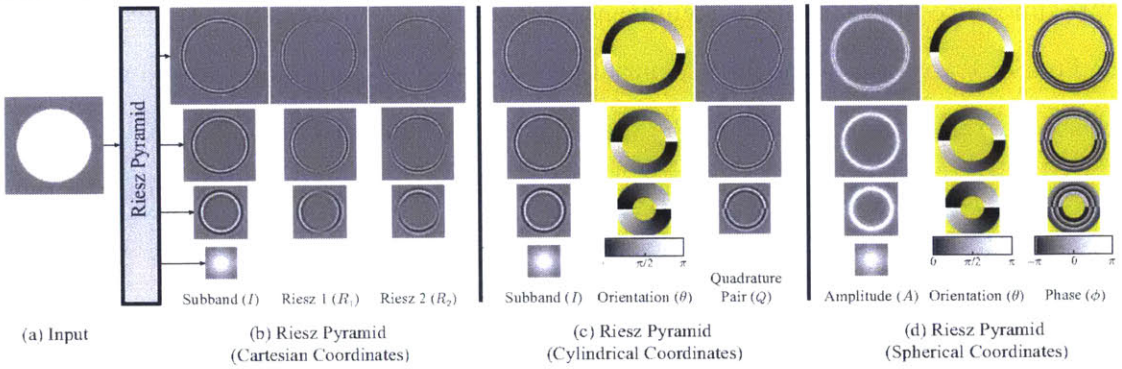


Figure 3.4: Three equivalent representations of the Riesz pyramid. The input is a circle with a sharp edge (a). In (b), the input is decomposed into multiple spatial sub-bands using an invertible transform, and an approximate Riesz transform is taken of each band to form the Riesz pyramid. At each scale, the three channels can be thought of as being components in Cartesian coordinates. In (c), they are expressed in cylindrical coordinates to show the sub-band, its quadrature pair and the local orientation. In (d), they are expressed in spherical coordinates to show the local amplitude, local orientation and local phase. Note the discontinuity in the orientation, quadrature pair and phase, which is due to the fact that orientation wraps around from 0 to π. In all three representations, there is a lowpass residual, of which we do not take the Riesz transform. The orientation and phase are not meaningful in regions of low amplitude (masked out in yellow).

quaternion representation of the Riesz transform to define and use the Riesz pyramid.

■ 3.3 Riesz Pyramids

The Riesz pyramid uses the Riesz transform to do phase analysis on all scales of an input image by first decomposing the image into multiple sub-bands, each of which corresponds to a different spatial scale, and then taking the Riesz transform of each sub-band (Fig. 3.4). An ideal version of the Riesz pyramid can be built in the frequency domain using octave (or sub-octave) filters similar to the ones proposed in the previous chapter and the frequency domain Riesz transform [25]. This can be used to magnify motions in videos faster than a two orientation complex steerable pyramid, but it requires the use of costly Fourier transforms to construct, making it unsuitable for online processing. To remedy this and gain further speedups, we approximate the ideal frequency domain Riesz transform with an approximate Riesz transform given by two

finite difference filters, which is significantly more efficient to compute. To avoid using the Fourier transform in the initial spatial decomposition, we also introduce a new non-oriented pyramid implemented in the spatial domain, similar to the Laplacian pyramid [9] but with wider filters that support a wider range of motion editing. We describe the approximate Riesz transform and the spatial decomposition in the following sections.

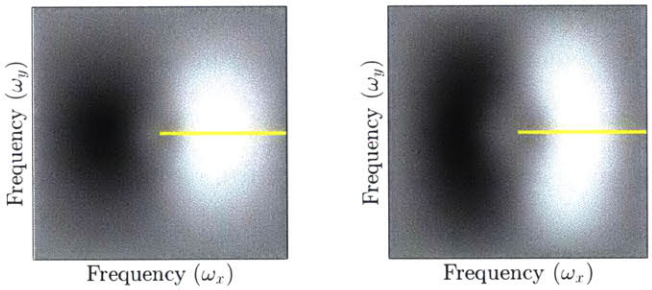
■ 3.3.1 Approximate Riesz Transform

In image pyramids, each sub-band is a critically sampled spatially bandpassed signal with most of the sub-band's energy concentrated in a frequency band around $\|\vec{\omega}\| = \frac{\pi}{2}$ (the Nyquist frequency is at $\omega_x = \omega_y = \pi$). As a result, we can approximate the Riesz transform with the three tap finite difference filters $[0.5, 0, -0.5]$ and $[0.5, 0, -0.5]^T$. These filters have frequency response

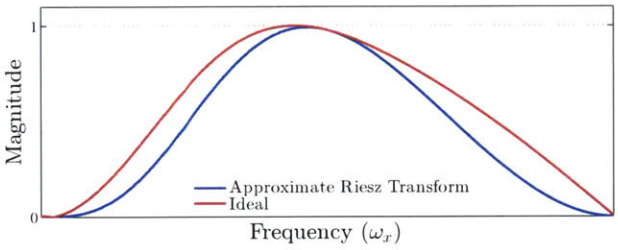
$$-i\sin(\omega_x) \approx -i\frac{\omega_x}{\|\omega_x\|}, -i\sin(\omega_y) \approx -i\frac{\omega_y}{\|\omega_y\|}, \tag{3.19}$$

when $\omega_x, \omega_y \approx \frac{\pi}{2}$. This is similar to the frequency response of the Riesz transform (Fig. 3.5). That is, these filters change the phase of the band by 90 degrees in the x and y directions respectively while not changing the amplitude substantially. For images, rather than image sub-bands, these three-tap filters are a better approximation to the derivative. This is because images have most of their spectral content concentrated at low frequencies. When $\omega \approx 0$, we have $-i\sin(\omega) \approx -i\omega$, which is the frequency response of the derivative operator.

In Appendix A.2, we provide a way to generate higher-tap approximations to the Riesz transform using a technique similar to one Simoncelli proposed to find derivative filter taps [69]. In practice, we found that using two three-tap filters to approximate the Riesz transform gave motion magnification results that were comparable to using higher-tap approximations or the frequency domain implementation of the Riesz transform.



(a) Riesz transform of band (b) Our approximation



(c) Comparison of 1D slices in yellow

Figure 3.5: The first channel of the Riesz transform of a pyramid level's transfer function (a) is compared to the first channel of our approximation of the Riesz transform (b). One dimensional slices along the yellow lines of (a) and (b) are shown in (c). If our approximation was perfect, (a) and (b) would be identical and the lines in (c) would coincide.

3.3.2 Spatial Decomposition

Prior to applying the Riesz transform, we decompose the image into non-oriented subbands using an invertible image pyramid. For the purposes of computational performance, we avoid the Fourier transform, eliminating the choice of a frequency domain construction (Fig. 3.6b). A compact space-domain image pyramid we could use is the Laplacian pyramid [9] (Fig. 3.6(a)). However, this pyramid has a very narrow impulse response, which limits the maximum amplification the pyramid can support (Fig. 3.6d-g). To remedy this problem, we design a self-inverting pyramid similar to the Laplacian pyramid but with wider impulse response (Fig. 3.6c). Simoncelli and Freeman [71] showed that such a pyramid can be constructed from a lowpass and highpass filter pair that satisfy certain properties. Rather than using the symmetric, but nonseparable lowpass and highpass filter taps provided by Simoncelli and Freeman, we design our own pyramid using a similar technique to theirs. Our filters use fewer taps than Simoncelli and Freeman and have additional structure imposed on them, which makes them very efficient to implement when the lowpass and highpass filters are jointly applied to the same input as they are when building the pyramid [50].

As a result, building the proposed pyramid requires a total of 30 multiplies per pixel per scale. Collapsing the pyramid requires applying the symmetric lowpass and highpass filter to separate bands and then summing the results for a total of 42 multiplies per pixel per scale. This results in a total cost of 72 multiplies per pixel per scale or 96 multiplies per pixel to build and collapse the pyramid. The approximate Riesz transform adds 2 multiplies per pixel per scale or 3 multiplies per pixel for a total of 99 multiplies per pixel.

The taps of our filters and more details on the design and implementation techniques can be found in Appendix A.1. A comparison between our new pyramid, a frequency-domain pyramid and the Laplacian pyramid, is given in Fig. 3.6.

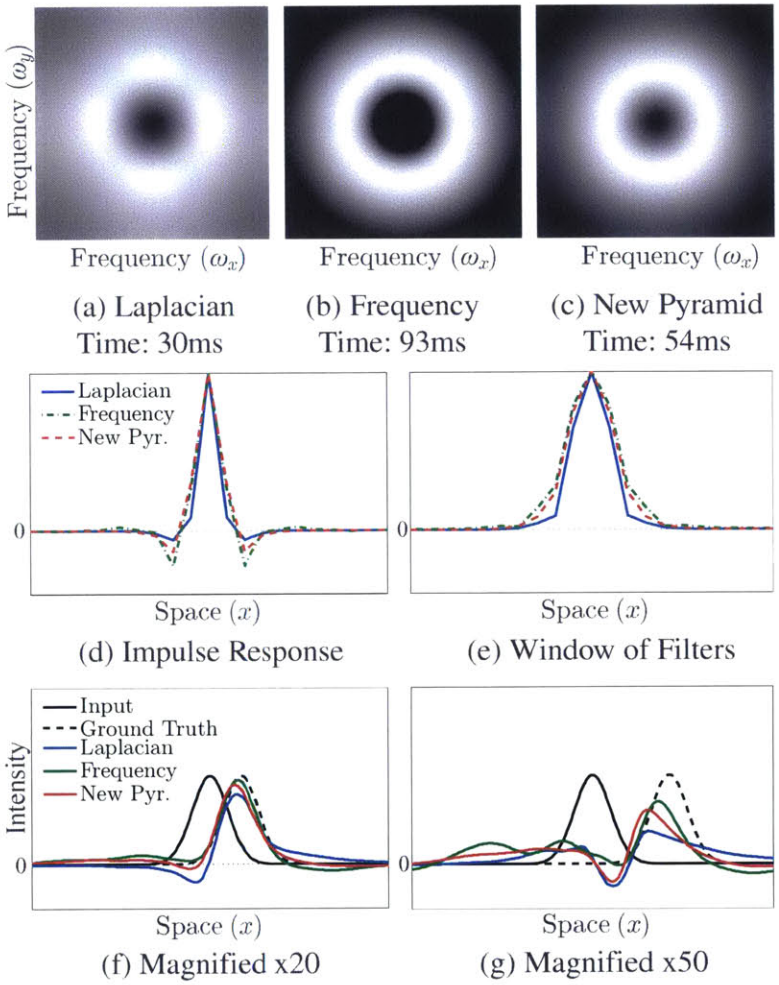


Figure 3.6: Different spatial decompositions for our new algorithm. In the top row, the frequency response of a level of the Laplacian pyramid (a), a frequency domain pyramid (b), and our new spatial domain pyramid (c). In the middle row, a one-dimensional cross section of their impulse responses (d) and windows (e). In the bottom row, a synthetic Gaussian shifted with our technique using a Laplacian pyramid, the frequency domain pyramid and our new pyramid for two amplification factors (f-g). The time in milliseconds to build and collapse a 960×540 *image in MATLAB is shown underneath the frequency response of each pyramid.*

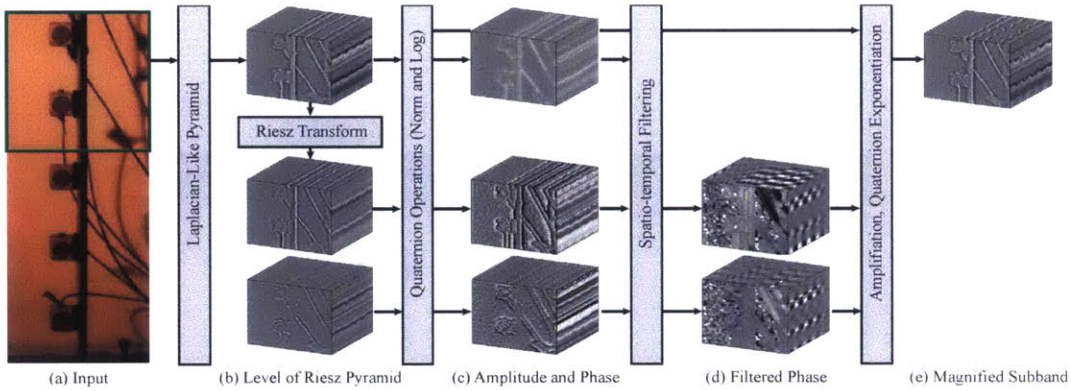


Figure 3.7: A processing pipeline for motion magnification with the Riesz pyramid. A region of interest (highlighted in green) of an input (a) is decomposed using a Laplacian-like pyramid (only one level shown). The Riesz transform of this level is taken to produce the Riesz pyramid (b). The quaternion norm is used to compute the amplitude (c, top row) and the quaternion logarithm is used to produce the quaternionic phase (c, bottom rows). The quaternionic phase is spatio-temporally filtered (d) to isolate motions of interest and then this quantity is used to phase-shift the input Riesz pyramid level to produce a motion magnified subband (e). These subbands can then be collapsed to produce a motion magnified video (not shown).

■ 3.4 Motion Magnification with the Riesz Pyramid

To perform motion magnification with the Riesz pyramid, we break the input image into subbands using our replacement for the Laplacian pyramid and take the approximate Riesz transform of each pyramid level (Fig. 3.7b). Then, we use the processing described in Sec. 3.2.3 to compute the quaternionic phase of each pyramid level as a function of time and position (Fig. 3.7c). We apply temporal and spatial filtering to the phases, magnify them and use the magnified phases to phase shift the Riesz pyramid coefficients (Fig. 3.7d-e). The real part of this pyramid is collapsed to produce the motion magnified video.

Below, we describe the filtering the quaternionic phase and the phase-shifting of Riesz pyramid coefficients in more detail.

■ 3.4.1 Temporal Filtering of Quaternionic Phase

While the quaternionic phase resolves the sign ambiguity in filtering $\phi(x, y, t)$ directly, it is still a wrapped quantity. That is, $i\phi\cos(\theta) + j\phi\sin(\theta)$ and $i(\phi + 2\pi)\cos(\theta) + j(\phi + 2\pi)\sin(\theta)$ correspond to the same value. Therefore, instead of filtering the quaternionic phase directly, we instead use a technique by Lee and Shin [46] to filter a sequence of unit quaternions. This technique is tantamount to phase unwrapping the quaternionic phases in time and then performing LTI filtering. We use it to LTI filter the Riesz pyramid coefficients at each pixel in each scale in time and then in a subsequent step we spatially smooth the pixel values with an amplitude weighted blur to improve SNR. We will also make the assumption that the local orientation at any pixel is roughly constant in time and approximately locally constant in space.

Suppose at a single location (x, y) in a single scale ω_r, the normalized Riesz pyramid coefficients are

$$\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n \tag{3.20}$$

where $\mathbf{r}_m = \cos(\phi_m) + i\sin(\phi_m)\cos(\theta_m) + j\sin(\phi_m)\sin(\theta_m)$, the most general form of a unit quaternion with no k component.

In ordinary complex phase unwrapping, we would take the principle value of the difference between successive terms and then do a cumulative sum to give an unwrapped sequence in which the difference between two successive terms was always in the interval $(-\pi, \pi]$. We do the same thing here. We compute the principle value of the phase difference between successive coefficients by dividing them and then taking the logarithm.

$$\log(\mathbf{r}_1), \log(\mathbf{r}_2\mathbf{r}_1^{-1}), \ldots, \log(\mathbf{r}_n\mathbf{r}_{n-1}^{-1}) \tag{3.21}$$

The terms $\mathbf{r}_m\mathbf{r}_{m-1}^{-1}$ will in general have nonzero k component (Eq. 3.9). However, if we make the assumption that $\theta_m = \theta + \epsilon$, that is that the local orientation is roughly

constant over time at every pixel, the k term will be close to zero. More explicitly,

$$\begin{aligned} \mathbf{r}_m \mathbf{r}_{m-1}^{-1} = & \cos(\phi_m - \phi_{m-1}) \\ + & \; i \sin(\phi_m - \phi_{m-1}) \cos(\theta) \\ + & \; j \sin(\phi_m - \phi_{m-1}) \sin(\theta) + O(\epsilon) \end{aligned} \tag{3.22}$$

which, ignoring the $O(\epsilon)$ term, has logarithm

$$i([\phi_m - \phi_{m-1}]) \cos(\theta) + j([\phi_m - \phi_{m-1}]) \sin(\theta) \tag{3.23}$$

where the bracketed terms are taken modulo 2π.

The second step is perform a cumulative sum of Eq. 3.21

$$\phi_1 \mathbf{u}, (\phi_1 + [\phi_2 - \phi_1])\mathbf{u}, \ldots, \left(\phi_1 + \sum_{l=2}^{n} [\phi_l - \phi_{l-1}]\right) \mathbf{u} \tag{3.24}$$

where $\mathbf{u} = i\cos(\theta) + j\sin(\theta)$. If we let $\phi'_m = \phi_1 + \sum_{l=2}^{m} [\phi_l - \phi_{l-1}]$, we can more compactly write this series as

$$i\phi'_m \cos(\theta) + j\phi'_m \sin(\theta) \tag{3.25}$$

At every pixel, we perform temporal filtering on this quantity to isolate motions of interest.

■ 3.4.2 Spatial Smoothing

We perform a spatial amplitude weighted blur with Gaussian kernel K_ρ with standard deviation ρ on the i and j components of the temporally filtered signal to further increase SNR

$$i\frac{A\phi' \cos(\theta) * K_\rho}{A * K_\rho} + j\frac{A\phi' \sin(\theta) * K_\rho}{A * K_\rho} \tag{3.26}$$

where A is the amplitude of the Riesz pyramid coefficients. If we use our assumption that the orientation does not change substantially in the support of K_ρ, then we can move $\cos(\theta)$ and $\sin(\theta)$ outside of the convolution in Eq. 3.26 to get.

$$i\cos(\theta)\phi'' + j\sin(\theta)\phi'' \tag{3.27}$$

where $\phi'' = \frac{A\phi' * K_\rho}{A * K_\rho}$.

■ 3.4.3 Amplification

We motion amplify a Riesz pyramid coefficient in the same way we would phase-shift a complex number. First, we perform a quaternion exponentiation on the filtered and amplified (by α) quaternionic phase (Eq. 3.27) to produce a unit quaternion

$$\cos(\alpha\phi'') + i\sin(\alpha\phi'')\cos(\theta) + j\sin(\alpha\phi'')\sin(\theta) \tag{3.28}$$

We then multiply this unit quaternion by the original coefficient $I + iR_1 + jR_2$ in the Riesz pyramid. We only need the real part of the result, which by Eq. 3.9 is equal to

$$I\cos(\alpha\phi'') - R_1\sin(\alpha\phi'')\cos(\theta) - R_2\sin(\alpha\phi'')\sin(\theta) \tag{3.29}$$

This gives the coefficients of a real Laplacian-like pyramid for every frame, in which the motions have been magnified, which can then be collapsed to produce a motion magnified video (Fig. 3.7(e)).

■ 3.5 Results

Phase-based video magnification with our new representation allows users to produce high-quality motion-magnified videos in real-time. We show several applications of our

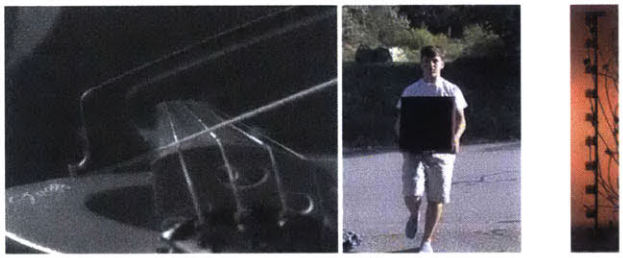
Video Type Domain	Resolution ($h \times w \times t$)	Wu et al. [85] Linear Space	Wadhwa et al.[81] Phase Frequency	2 Orient. CSP Phase Frequency	Riesz (Freq.) Phase Frequency	Riesz (Space) Phase Space
Crane	280× 280× 220	6.0	43.0	15.9	13.6	**10.1**
Guitar	432×192×300	7.9	60.5	23.5	20.4	**14.9**
Baby	960×540×300	35.6	325.9	95.7	101.6	**75.4**
Camera	512×384×1000	46.6	375.7	140.3	122.5	**91.5**
Violin	480×360×300	12.7	115.8	43.1	34.9	**29.3**
Balance	272×384×300	7.7	72.7	30.7	23.6	**18.3**
Smoke	240×600×300	11.7	92.3	32.5	30.6	**25.7**
Column	200×1056×600	41.7	259.7	95.3	90.8	**76.5**

Table 3.1: Running times (in seconds) of comparable MATLAB implementations of phase-based motion magnification, the Riesz pyramid, and several variants of the complex steerable pyramid. All phase-based methods were run with spatial phase denoising of the same value of ρ. Video read and write times were not included. As specified in Wadhwa et al. [81], we use an eight orientation octave bandwidth pyramid (Col. 4). We also present their method using the smallest possible complex steerable pyramid, a two orientation octave bandwidth pyramid (Col. 5). "Domain" (third row) specifies whether the pyramid was constructed in the spatial or frequency domains. For each sequence, the fastest phase-based method is highlighted in bold.

algorithm in this section. For all of our results, we used the approximate Riesz transform (Sec. 3.3.1) with the new spatial domain pyramid (Sec. 3.3.2). We converted the videos to YIQ colorspace and only processed the luma channel.

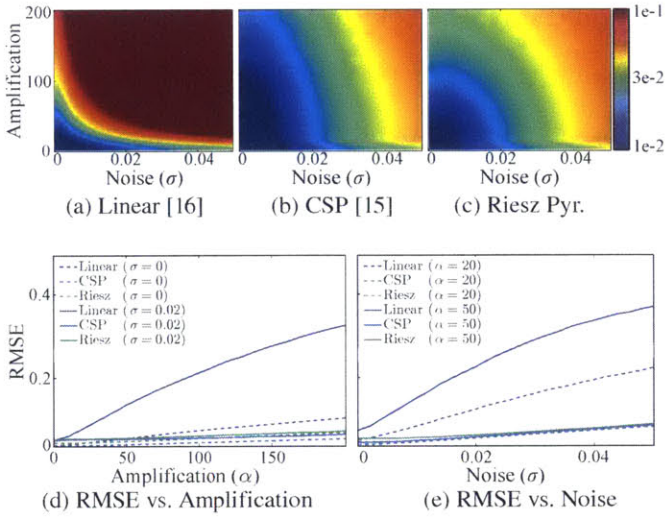
A vibrating string on its own makes only a very quiet sound. As a result, stringed musical instruments are constructed so that the string vibrates a soundboard or a hollow resonating chamber that produces almost all of the audible sound. In *violin*, the G string of a violin is played by a bow and the resulting vibrations were recorded by a high speed camera at 3000 FPS. This high speed video reveals the intricate motions of the string. However, motion amplification with our new representation reveals the invisible vibrations of the soundboard and tailpiece. We suppress amplification near the string in our result.

A man holding a weight struggles to maintain balance, but in a 300 frame per second high speed video, *balance*, this struggle is not clearly apparent. When we amplify the motions ten times in a passband between 1.0-8.0Hz, the man's struggle becomes visible and we see all the work he is doing to hold the weight.



(a) *violin* (b) *balance* (c) *column*

Figure 3.8: Representative frames from videos in which we amplify imperceptible motions. The full sequences and results are available in the supplementary materials.



(a) Linear [16] (b) CSP [15] (c) Riesz Pyr.

(d) RMSE vs. Amplification (e) RMSE vs. Noise

Figure 3.9: A comparison of our new method versus previous Eulerian video magnification methods on a synthetic oscillating Gaussian, in which the ground truth amplified motion is known. The logarithm of the RMSE is shown in color for the linear method (a), for the complex steerable pyramid phase-based method (b) and for our new phase-based method (c). We also show slices of the RMSE vs. amplification (d) and RMSE vs. noise (e) for the three methods.

When laminar flow becomes turbulent, there is a transition region in which sinusoidal instabilities grow before eventually becoming unstable and turbulent [75]. In *smoke*, we reveal these sinusoidal instabilities by applying motion magnification to a column of incense smoke transitioning from laminar to turbulent flow (Fig. 3.1).

Chen et al. [11] used local phase to compute the mode shape of a cantilever beam struck by a hammer from video. We obtained this sequence, *column* (Fig. 3.8(d)), and used motion amplification to visualize the mode shapes by amplifying the motions in the

video along narrow temporal bands. These mode shapes correspond to the theoretically derived ones.

Comparisons with Previous Techniques In Fig. 3.1 and the supplementary materials, we present several comparisons between phase-based motion magnification using the Riesz pyramid and using the complex steerable pyramid [81] on natural videos. The Riesz pyramid yields results that are comparable in quality to those produced with the complex steerable pyramid, but much faster. To verify this quantitatively, we tested phase-based video magnification with our new representation and the eight orientation complex steerable pyramid and linear video magnification on a sequence of a synthetic oscillating Gaussian, in which the ground truth motion magnified sequence is known. We computed the RMSE of these techniques as a function of amplification factor α and spatiotemporal image noise σ (Fig. 3.9). For all amplification factors and noise levels, the RMSE for our new representation is very close to that of phase-based video magnification with the complex steerable pyramid, and substantially better than the linear method [85].

In Table 1, we display the running times of comparable MATLAB implementations of linear video magnification and phase-based video magnification using 8 and 2 orientation complex steerable pyramids, the Riesz pyramid implemented in the frequency domain (Fig. 3.6b) and the Riesz pyramid implemented in the spatial domain (Fig. 3.6c). Using the spatial-domain Riesz pyramid yields the fastest phase-based method, producing results four to five times faster than the 8 orientation complex steerable pyramid used in the previous chapter. It is 20% to 80% faster than even the two orientation complex steerable pyramid. The spatial-domain Riesz pyramid is also faster than the frequency domain implementation, demonstrating the additional speedup that our approximate Riesz transform and spatial-domain decomposition provide.

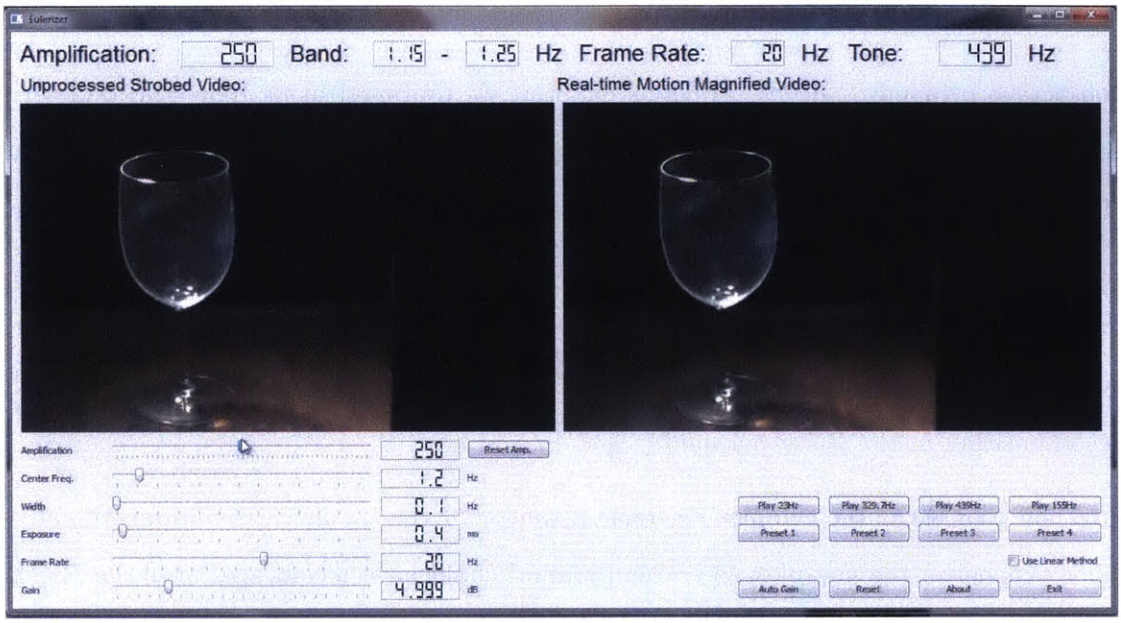


Figure 3.10: A frame from our real-time demo using Riesz Pyramids. A wine glass is filmed in with a video camera while a nearby speaker plays its resonant frequency (449.1Hz) at it. The induced vibrations are too small and too fast to be seen with the naked eye. Filming with very short exposure times (0.4ms) leads to temporal aliasing which makes the fast vibration look like one occurring at 1.2 Hz. When this is motion magnified, the oscillations of the wine glass are visible on the right in the motion magnified video.

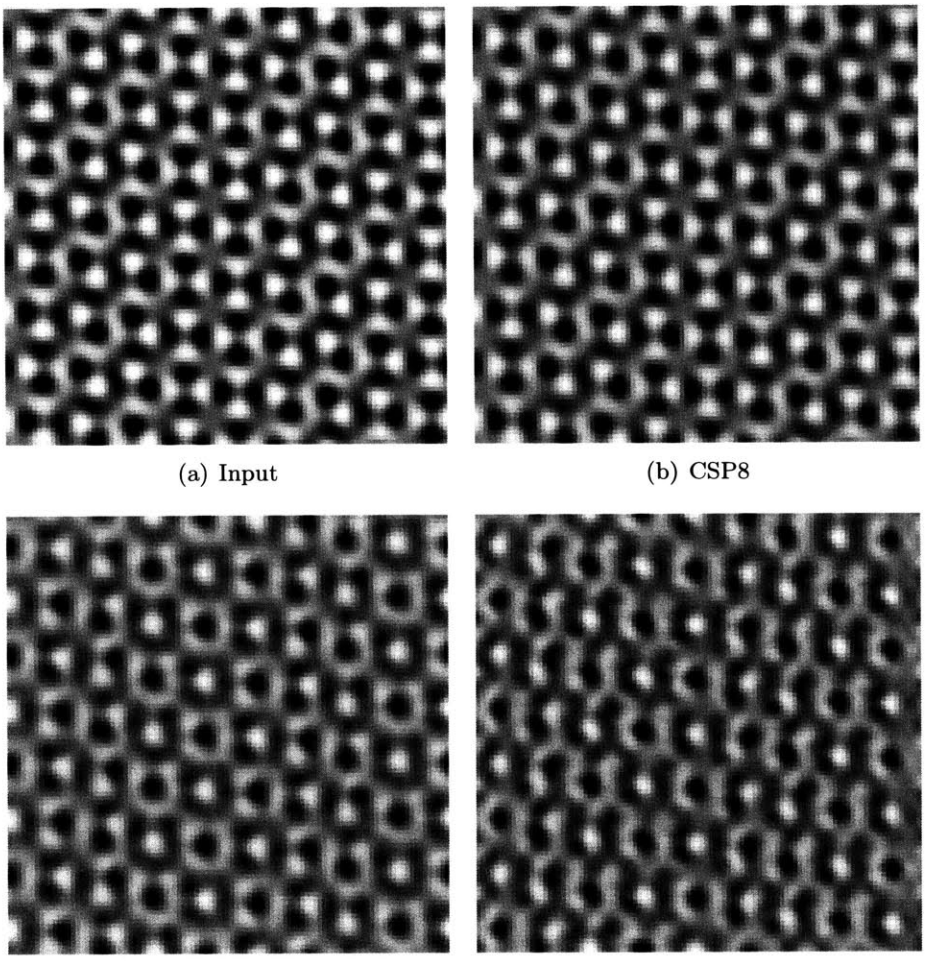
Real Time Implementation We created a C++ implementation of phase-based video magnification with the Riesz pyramid using OpenCV and QT. We can process a live 640×400 pixel video at 40 frames per second on a laptop with four cores and 16GB RAM (the algorithm uses only a single CPU core). Because all of the operations are compact linear filters or element-wise operations, a parallelized or GPU implementation could further increase the speed. In our real time implementation, we use a Laplacian pyramid in which the image is blurred and downsampled with a 5×5 Gaussian kernel (Fig. 3.6a) as the spatial decomposition because it is efficiently implemented in OpenCV. In Fig. 3.10, we show a frame from our real time interface. In it, we amplify the invisible oscillations of a wine glass when its resonant frequency is played at it.

3.6 Discussion and Limitations

Sub-octave pyramids: In the previous chapter, we proposed using half- and quarter-octave bandwidth pyramids to increase the amount by which motions can be shifted. Since our new representation focuses on speed, we concentrated on comparing our technique to an octave-bandwidth complex steerable pyramid since it is the faster among these decompositions. It is possible that our algorithm could be improved further by using non-oriented versions of these sub-octave bandwidth pyramids as the spatial decomposition in the Riesz pyramid.

Pros and cons w.r.t. the complex steerable pyramid: Even though it is computationally more expensive, the complex steerable pyramid could have advantages over the Riesz pyramid in some scenarios. For example, the Riesz pyramid may have trouble at points where there is not a single dominant orientation, as demonstrated in Fig. 3.11. The input image is a sum of four sinusoids of the same wavelength, but of different orientations. Thus, the entire image consists of points that do not have a single dominant orientation. Neither the Riesz pyramid nor the two orientation complex steerable pyramid can properly motion-magnify this image. However, a complex steerable pyramid with eight orientations can better separate this complex texture into one dimensional sinusoids, which can then be motion-magnified more accurately (Fig. 3.11(b)). In general, we would expect the Riesz pyramid to perform similarly to the two orientation complex steerable pyramid as the latter is also not capable of separating two orientations at a single point unless they are exactly horizontal and vertical.

Limitations The approximate Riesz transform does not maintain the power of an input signal like the ideal Riesz transform does, which can cause minor artifacts. That is, a signal like $\cos(x)$ might get mapped to $((1+\epsilon)\sin(x), 0)$ where $\epsilon \neq 0$. As a result, the phase signal may not be exactly x, but rather x plus an order ϵ term that might vary



(a) Input (b) CSP8

(c) CSP2 (d) RP

*Figure 3.11: An example of an advantage of the complex steerable pyramid over the Riesz pyramid on a synthetic sequence. The texture in (a) is the sum of four sinusoids with the same wavelength, but different orientations (*18°, 72°, 108°, 162°*). The texture and a copy shifted to the right by 0.1 pixels are motion-magnified by 30 times using an eight orientation complex steerable pyramid (b), a two orientation complex steerable pyramid (c) and the frequency domain Riesz pyramid (d). Notice how the texture in (b) is more similar to the original (a) in comparison to (c) and (d). The full sequences are available in the supplementary material.*

with location $x + O(\epsilon)f(x)$. This causes different parts of the sinusoid to get magnified slightly differently causing some minor artifacts. The spatial smoothing step (Sec. 3.4.2) can be used to smooth out these spatial inconsistencies and reduce the artifacts. More details are given in Appendix A.2.

Our new representation also still suffers from some limitations of the Eulerian motion magnification framework. For example, in the violin sequence, there are some artifacts near the vibrating string no matter which motion magnification method is used. This is because these motions are relatively large and so are not well-characterized by an Eulerian framework.

Chapter 4

Noise Analysis and Applications in Science and Engineering

In the previous chapters, we described methods to magnify tiny motions in videos revealing numerous subtle phenomena, such as the breathing motions of a baby and the swaying of a construction crane. However, because the motions were so small, we needed to spatially blur and temporally filter them to get adequate signal-to-noise ratios. While this blurring and filtering boosts SNR, it can also shape random noise present in every video into a seemingly realistic and visually sensible signal. In this chapter, we provide tools to determine when a motion magnified signal is caused by noise and therefore spurious.

The increased certainty of what is being amplified makes motion magnification a tool scientists and engineers can use. We also describe a way to convert local phase variations that underpin phase-based motion magnification into optical flow, a 2D estimate of how much pixels in the scene are moving. These numbers make it possible to perform quantitative experiments with tiny motions extracted from video data. We apply our technique to problems in biology and engineering. And we show how both the qualitative and quantitative descriptions of tiny motions may play a role in scientific discovery.

4.1 Introduction

Small motions have scientific and engineering importance. They may correspond to small net forces and can reveal subtle processes. They can indicate dynamics that might otherwise be undetectable, potential precursors to larger displacements. Small motions are often in a linear response regime, and can show the physical response eigenmodes and their temporal eigenvalues. The spatial pattern of small motions can show mechanisms or identify excitation patterns that may occur at a larger spatial scale.

Unfortunately, despite their importance, small motions by their nature are often too small to see. The solution is motion magnification, a technique described in the previous chapters. It is a tool that functions like an ordinary microscope, but instead of making small things bigger, it makes small motions bigger. It takes as input a nearly static video and outputs a new video, in which the motions have been re-rendered to be larger.

To make motion magnification applicable to scientific problems, it must be augmented in two ways. First, there needs to be a quantitative output describing the size and direction of the motions that are occurring in a video. Second, there needs to be a quantification of how noise affects the tiny motion signal in a video. This quantification can be used to disambiguate true tiny motions from spurious ones due to noise present in every video.

Quantitative Readout of Motions As described in chapter 2, phase-based motion magnification works by first decomposing every frame of an image sequence into different spatial scales and orientations using a complex steerable pyramid filter bank. Each band of this representation has a notion of local amplitude and local phase. In phase-based motion magnification, the temporal local phase changes in each band are amplified to magnify the motions (Sec. 2.3).

However, the phase changes can also be used to estimate the motion at every pixel. They provide information about a component of motion approximately perpendicular to the orientation of the corresponding level of the pyramid. Phase changes over multiple orientations can provide enough information to estimate the 2-dimensional motion within the image [29]. However, the reliability of this information depends on the amount of image content the corresponding pyramid level has (e.g. a constant color region of an image will have zero-response in all levels and completely unreliable phases). It can be quantified by the amplitude of the pyramid levels, which we use as the weights in a weighted-least squares problem that combines the information across levels to produce an estimate of the horizontal and vertical motions at every pixel in the video.

This motion estimate combined with motion magnification provides an easy-to-understand and fully quantitative description of phenomena involving tiny motions. We focus on using the complex steerable pyramid rather than the Riesz pyramid because of its superior ability to separate complex textures into accurately analyzable sub-bands (discussed in Sec. 3.6).

Sources of Noise There are many factors that can cause spurious or unwanted tiny motions in a video, some of which are shown in Fig. 4.1. Subtle camera or object motions due to footsteps or earthquakes can cause real, but unwanted tiny motions. Atmospheric turbulence can cause the path of light to bend inducing apparent tiny motions in a video similar to how heated air on a hot day causes visible shimmering [37, 68, 87]. Many lights that draw power from the grid flicker in response to the grid's AC current. This flickering can cause subtle color changes in a video, which subtly affect local phase changes; it may be necessary to eliminate temporal frequencies that correspond to AC power or to use lights powered with DC current to get reasonable results.

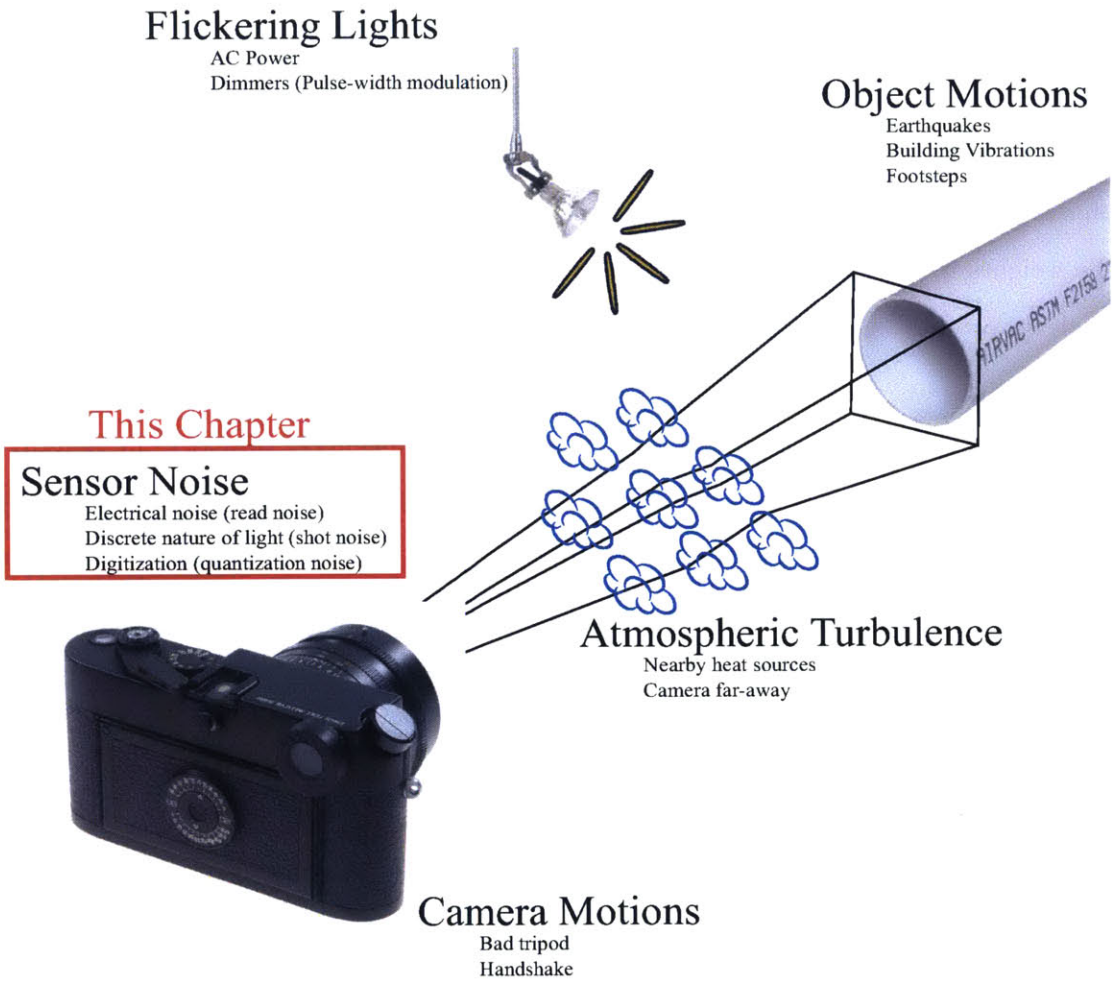
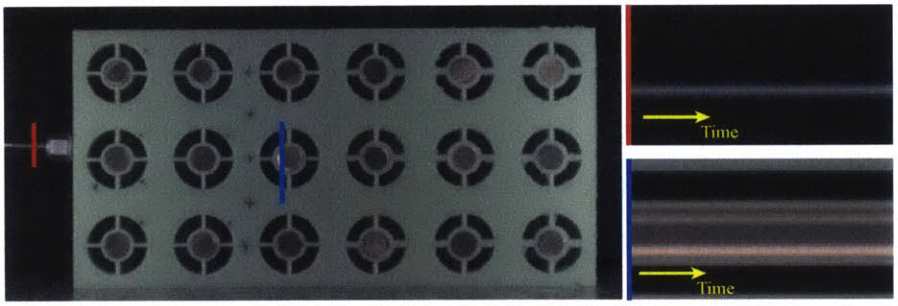
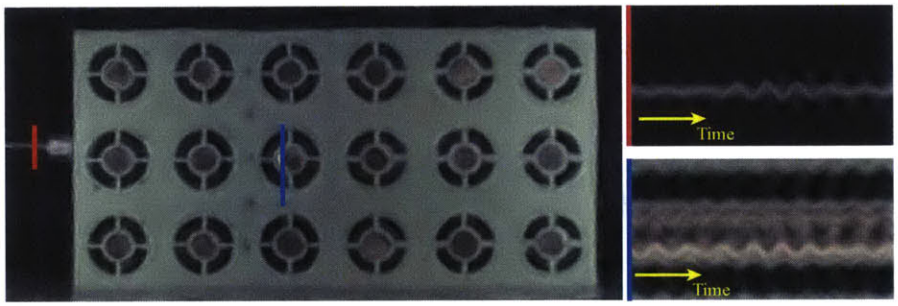


Figure 4.1: Sources of unwanted and spurious tiny motions in video. In this chapter, we focus on spurious tiny motions caused by sensor noise.



(a) Synthetic video with noise and without motions



(b) Motions magnified x600 (40-60Hz)

Figure 4.2: Magnification of spatially smoothed and temporally filtered noise can look like a real signal. A synthetic 300-frame video was created by replicating a single frame 300 times and adding independent noise to each frame (a). The result was motion magnified 600x in a temporal band of 40-60Hz (b). Timeslices from the same parts of each video are shown on the right for comparison. The source frame is of an acoustic metamaterial (filmed in the Bertoldi lab at Harvard University [83] and discussed in Sec. 4.4.5).

The sources of spurious or unwanted motions described in the previous paragraph can be lessened through the use of good experimental methodologies: using sturdy tripods, eliminating heat sources in the optical path and using non-flickering light sources. However, one source of spurious tiny motions that is not possible to completely eliminate and fundamentally limits the size of motions we can unveil is sensor noise. This noise is due to both electrical noise in the camera and the quantum nature of light, in which the number of photons arriving at camera's sensor varies across exposures even if the underlying scene irradiance doesn't change.

Sensor noise is usually zero-mean, roughly independent across space and is well-

modeled as signal-dependent Gaussian noise [39, 52, 56]. However, despite being roughly independent across space, it can cause spurious tiny motions, which when temporally filtered and spatially blurred look like plausible motion signals. This is demonstrated in Fig. 4.2, in which a synthetic video with no motions is produced by replicating a single frame 300 times and then adding realistic, independent noise to each replica. When the synthetic video is motion magnified, several structures appear to be moving (Fig. 4.2b).

We detect when motions are spurious in this and other videos by measuring the expected amount of noise on each motion vector; motions smaller than the noise level are likely spurious. We measure noise by treating each motion vector as a 2D random variable. Its distribution, specifically the covariance matrix of the horizontal and vertical components, gives us a way to quantify the amount of noise in the motion estimate. We estimate the covariance matrices of the motion vectors at each pixel by creating a simulated video with zero motion in which all the changes in pixel color are due to noise. Then, we estimate the motions in this video, which should be zero, and use the sample covariance at every pixel as our measure of noise. We show, via both real and synthetic experiments, that the covariance matrices computed as a result of this simulation, accurately reflect the amount of noise present in the motion signal in textured regions of the video when the motions are non-zero, but small.

Knowing quantitatively both the amount of motion and the amount of noise allows us to compute signal-to-noise ratios of the motions at every pixel. This can be used as both a measure of confidence in the result and a way to suppress the magnification of spurious motions that are due to noise by using a Wiener filter. It also lets us gracefully handle the aperture problem [27]; components of the motion that are not possible to detect due to insufficient image content have high noise variances. Using our noise model, we found that in a typical video taken with a good camera (specifically the Phantom V10), the smallest motion we can reliably detect is on the order of 1/100th

of a pixel.

■ 4.2 Related Work

Motion Estimation Visualizing tiny motions is a useful tool in understanding them. However, in many cases, knowing the quantitative amount by which objects move is also desired for experiments. In 3D scenes, objects or parts of objects will move according to some 3D trajectory $\vec{X}(t)$. When the object is filmed with a video camera, this trajectory will get projected to a 2D path $\vec{x}(t) = (u(t), v(t))$. In discrete time, this 2D path between two frames of a video is known as *optical flow.*

Estimating 2D motions or optical flow is a widely studied problem in computer vision, with many possible solutions [3, 27–29, 34, 41, 54, 73]. Techniques that look at the local phase difference between complex wavelet representations of image sequences are most closely related to phase-based motion magnification [28–30, 34]. Fleet and Jepson demonstrated that contours of constant phase in space-time follow objects as they move [29]. Some authors have also proposed probabilistic models of optical flow [27, 70], which is related to our goal of computing the covariance of an optical flow estimate.

When an image region has insufficient texture, it may not be possible to determine optical flow completely [27]. This is known as the *aperture problem.* For example, when looking at only a local region of an image, it is not possible to tell if an edge is moving in a direction parallel to it. Similarly, it is not possible to tell how a flat, textureless region is moving in any direction.

We use ideas from Fleet and Jepson's work [29] to estimate motions. The aperture problem is implicitly handled in our noise handling. The covariance matrix will be very large in directions in which we cannot estimate the motions.

Noise Models The most widely used noise model in image processing and computer vision is additive white Gaussian noise (AWGN) [8, 15, 88], in which every pixel is assumed to be distributed as a Gaussian random variable with constant standard deviation and mean proportional to the scene irradiance. This model is easy to analyze, but in reality, the number of photons collected at a sensor well is a Poisson-distributed random variable with variance equal to its mean. Noise variance at a pixel therefore depends on a pixel's intensity [39, 52, 56].

To handle this, we use a noise model, in which each pixel has independent Gaussian-distributed noise, but the variance of the noise at a pixel with mean intensity I is a function $f(I)$ of the mean intensity. Liu et al. [52] refer to this function f as a *noise level function* and we do the same.

Noise Estimation Estimating the noise level function from a single image is an underconstrained problem and requires some assumptions be made about the underlying image [52, 88]. However, estimating it from multiple images of the same scene is overconstrained and is a solved problem [40]. If multiple images of the the same scene are available, then the differences in brightness are largely due to noise. Therefore, the sample variance across the multiples images can be used to estimate the noise level.

■ 4.3 Method

Our goal is to use local phase changes in the video to provide a quantitative estimator V of the 2D motion vectors at every pixel and compute the covariance matrix of this estimator Σ_V. This covariance matrix gives us a measure of how sensor noise affects both our quantitative motion estimate and motion magnification. We use it to put error bars on these quantities or compute signal-to-noise ratios. We also use Wiener filtering to suppress motion magnification in regions whose apparent motions are mostly due to noise.

4.3.1 Phase-Based Motion Estimation

In phase-based motion magnification (Sec. 2.3.4), we project every frame of the input video onto the complex steerable pyramid basis. Each frame is transformed into a set of subbands corresponding to different spatial scales r and orientations θ, each of which has a local amplitude $A_{r,\theta}(x,y,t)$ and local phase $\phi_{r,\theta}(x,y,t)$. Then, the local phase variations

$$\Delta\phi_{r,\theta}(x,y,t) := \phi_{r,\theta}(x,y,t) - \phi_{r,\theta}(x,y,0) \tag{4.1}$$

are amplified independently in every subband. In this section, instead of amplifying the local phase variations, we use them to compute quantitative motion vectors. We assume that every pixel is characterized by the motions of a single object, quantitatively described by the 2D vector $V(x,y,t) = (u(x,y,t), v(x,y,t))$ that we seek to compute.

Fleet and Jepson have shown that contours of constant phase in subbands provide a good approximation to the motion field in a video [29]. We make a similar *phase constancy* assumption, in which the following equation relates the phase of the frame at time 0 to the phase of future frames:

$$\phi_{r,\theta}(x,y,0) = \phi_{r,\theta}(x - u(x,y,t), y - v(x,y,t), t). \tag{4.2}$$

This is similar to the brightness constancy assumption from optical flow [41, 54], but with local phase replacing brightness as the quantity that is constant under translation. We use a Taylor series approximation on the right-hand side of this equation around (x,y) to get

$$\Delta\phi_{r,\theta} = \left(\frac{\partial\phi_{r,\theta}}{\partial x}, \frac{\partial\phi_{r,\theta}}{\partial y}\right) \cdot (u,v) + O(u^2, v^2), \tag{4.3}$$

where arguments have been suppressed and $O(u^2, v^2)$ represents higher-order terms in the Taylor expansion. Since we are only interested in small motions, higher order terms

are negligible and the local phase variations are essentially equal to only the linear term:

$$\Delta\phi_{r,\theta} = \left(\frac{\partial\phi_{r,\theta}}{\partial x}, \frac{\partial\phi_{r,\theta}}{\partial y}\right) \cdot (u, v). \tag{4.4}$$

Typically, the spatial gradients of the local phase, $\left(\frac{\partial\phi_{r,\theta}}{\partial x}, \frac{\partial\phi_{r,\theta}}{\partial y}\right)$, will be roughly constant, close to the peak tuning frequency of the corresponding subband's filter [28]. This frequency is a 2D vector oriented orthogonal to the direction the subband selects for, which means that the local phase changes only gives us information about the motions perpendicular to this direction.

In a single oriented subband, Eq. 4.4 is underconstrained. However, if we assume that the motion is the same in every subband, we can combine constraints across a subset of the levels to overconstrain the problem. Were there no noise, all these constraints would be equally valuable. However, subbands with low amplitude $A_{r,\theta}$ have much noisier local phase variations. They are roughly normally distributed with variance approximately proportional to the inverse amplitude squared (Fig. B.1 and Eq. B.13). To take this into account, we estimate the motion using weighted least squares with weights equal to the amplitude squared, so that reliable constraints are weighted more heavily[1]. For every pixel at location (x, y) and time t, the result is an objective function of the form

$$\arg\min_{u,v} \sum_i A^2_{r_i,\theta_i} \left(\left(\frac{\partial\phi_{r_i,\theta_i}}{\partial x}, \frac{\partial\phi_{r_i,\theta_i}}{\partial y}\right) \cdot (u, v) - \Delta\phi_{r_i,\theta_i} \right)^2, \tag{4.5}$$

where arguments have been suppressed for readability. This optimization problem can be solved exactly [44] and its solution V, our motion estimate, is

$$V = (X^T W X)^{-1}(X^T W Y), \tag{4.6}$$

[1]Were the local phase variations uncorrelated and exactly Gaussian, this would give the unbiased, linear estimator with the least variance [44].

where X is a $N \times 2$ matrix consisting of the spatial gradients of the local phases, Y is an $N \times 1$ matrix of the temporal local phase variations and W is a diagonal $N \times N$ weighting matrix of the squared amplitudes. That is,

$$X = \begin{bmatrix} \frac{\partial}{\partial x}\phi_{r_1,\theta_1} & \frac{\partial}{\partial y}\phi_{r_1,\theta_1} \\ \vdots & \vdots \\ \frac{\partial}{\partial x}\phi_{r_N,\theta_N} & \frac{\partial}{\partial y}\phi_{r_N,\theta_N} \end{bmatrix}, Y = \begin{bmatrix} \Delta\phi_{r_1,\theta_1} \\ \vdots \\ \Delta\phi_{r_N,\theta_N} \end{bmatrix} \text{ and } W = \begin{bmatrix} A^2_{r_1,\theta_1} & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & A^2_{r_N,\theta_N} \end{bmatrix}. \tag{4.7}$$

We solve such an optimization problem at every pixel in every frame, yielding a time series of displacement values from a reference frame at time 0 at every location (x, y).

Spatial Smoothing or Temporal Filtering In phase-based motion magnification, we also apply spatial smoothing and temporal filtering to the local phase variations to boost SNR. We add these steps to our processing to ensure that our estimated motion vector field is comparable to the motion magnified video. To replicate spatial smoothing with a filter kernel K, we assume the motion field is constant in a small window around each pixel the size of the support of K. We add additional constraints from neighboring pixels, weighted by both their amplitude squared and the corresponding value in K, to the objective described in Eq. 4.5. To handle temporal filtering, we replace the local phase variations (Eq. 4.1) with temporally filtered local phase variations.

■ 4.3.2 Noise Analysis

In our motion estimation pipeline, we transform frames of a noisy input video into the complex steerable pyramid representation. Because the resulting coefficients are noisy, their local phase is noisy. Since our motion estimate uses these noisy local phases, it too is noisy. Therefore, sensor noise, random fluctuations in pixel value cause random fluctuations in the estimated motions. The amount of noise present in the

motion vectors affects our ability to recover tiny motions and we seek to quantify it by computing its variance and standard deviation. Motions smaller than the standard deviation are likely spurious.

We model the estimated motion as the sum of a true motion (the result in the absence of sensor noise) and a noise term. We quantify the noise level by estimating the covariance matrix of the noise term at every pixel. This requires knowing the distribution of the motion estimate as sensor noise varies, which requires multiple samples of the motion estimate in which the true motions are constant. This is difficult to obtain from a real video because the true motions between pairs of frames are not known and can vary over time. To solve this problem, we create a stochastically simulated noisy video in which the true motions are known to be exactly zero. That is, a single frame of the input video is replicated and realistic, independent noise is added to each replica. We compute the sample covariance over time of the estimated motion vectors in this simulated video. As long as the noise model is realistic, the sample covariance matrix is accurate estimate of noise level in a real motionless video.

We show analytically and empirically that even though the sample covariance matrix was computed for a simulated video with *no* motions, it is accurate for videos with subpixel *small* motions at points where there is some image texture or edges. This means that it is possible to quantify the effect of noise on the motion estimate using just a *single* frame of the input video. This seems counterintuitive because motion implicitly relies on multiple frames of the video. However, in the special case of small motions, the frames are almost identical and it is possible to estimate noise from just one. A big factor in how sensor noise affects the noise of motion vectors is image content, i.e. higher contrast areas will have lower noise levels. For small motions, this is computable from just a single frame.

At points in the image where there is no image content (textureless regions), our

covariance estimate is large ($> 1\text{px}^2$), but only approximately accurate. Since recovering motions at these points is nearly impossible, this accuracy is acceptable. In the case of a simplified noise model, we trace the effect of noise in the motion estimation pipeline and analytically derive what the covariance matrix should be in Appendix B.

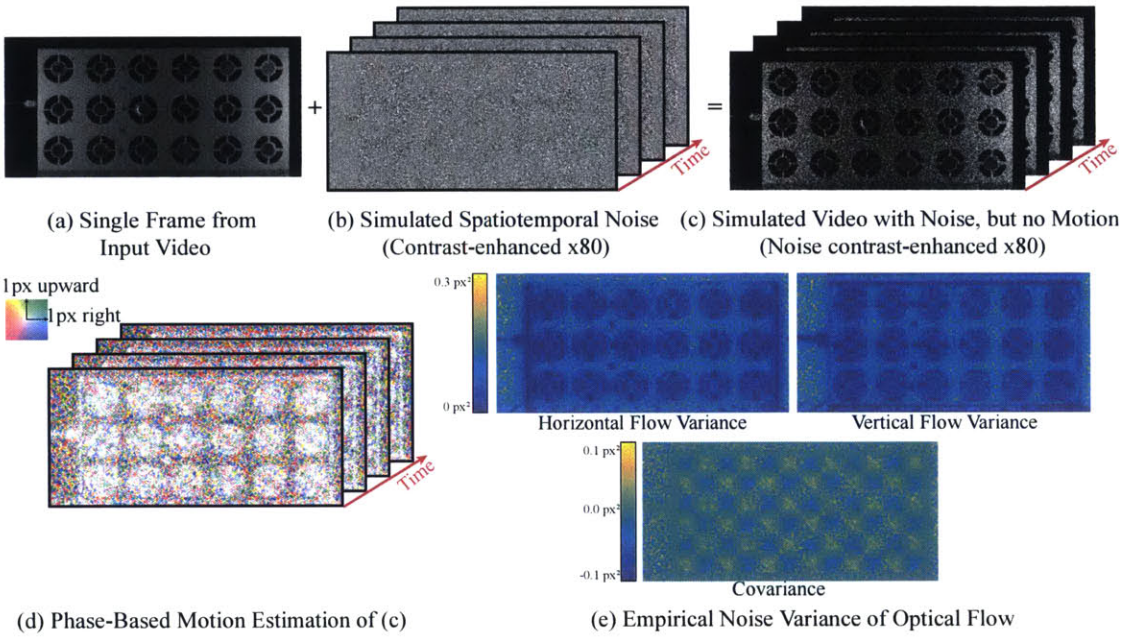
Monte Carlo Simulation We assume that the noise level function f of the video is known. If it is not known, it can be estimated from the input video or a calibration video taken with the same camera settings (Sec. 4.3.2). We perform a simulation to compute the sample covariance of the motion estimate. We create a motionless synthetic video with independent noise added to every frame and compute the sample covariance over time. We take a single frame $I(x, y, 0)$ from the input video (Fig. 4.3a) and replicate it N times, treating the replicas as frames of the synthetic video. For each replica, indexed by time t, we simulate independent Gaussian noise $n_G(x, y, t)$ of unit variance and multiply it by the square root of the noise level function f evaluated at $I(x, y, 0)$ (Fig. 4.3b). We add this to the replica to create the noisy motionless video (Fig. 4.3c)

$$I_S(x, y, t) := I(x, y, 0) + \sqrt{f(I(x, y, 0))} n_G(x, y, t). \tag{4.8}$$

We estimate the motions in I_S (Fig. 4.3d) using our technique with spatial smoothing, but without temporal filtering, which we handle in a later step. This results in a set of 2D motion vectors $V(x, y, t)$. Because the input video has no motions, all variations in V over time are entirely due to noise. We quantify the variations by computing the sample covariance matrix over the time dimension, given by

$$\Sigma_V = \frac{1}{N-1} \sum_t \left(V(x, y, t) - \bar{V}(x, y)\right) \left(V(x, y, t) - \bar{V}(x, y)\right)^T \tag{4.9}$$

where $\bar{V}(x, y)$ is the mean over t of the motion vectors. This results in a 2×2 symmetric matrix at every point, with only three unique components. In Fig. 4.3e, we show these



(a) Single Frame from Input Video

(b) Simulated Spatiotemporal Noise (Contrast-enhanced x80)

(c) Simulated Video with Noise, but no Motion (Noise contrast-enhanced x80)

(d) Phase-Based Motion Estimation of (c)

(e) Empirical Noise Variance of Optical Flow

Figure 4.3: Using a probabilistic simulation to compute the noise covariance of the motion estimate. A single frame from an input video is replicated and simulated, but realistic noise is added (b) to produce a synthetic video with no motions and only noise (c). Optical flow is estimated in this video (d) and the sample variances and sample covariance of the vertical and horizontal components of the motion are computed to give an estimate of how much noise is in the motion estimate (e). The input frame is of an acoustic metamaterial (filmed in the Bertoldi lab at Harvard University [83] and discussed in Sec. 4.4.5).

components, the variances of the horizontal and vertical components of the motion and their covariance.

The temporal filter reduces the elements of the covariance matrix. We take its effect into account by considering its impulse response $T(t)$. Oppenheim and Schafer [59] show that a signal with IID noise of variance σ^2 has variance $\sum_t T(t)^2\sigma^2$ after temporal filtering. Therefore, when a temporal filter is used, we multiply the covariance matrix by $\sum_t T(t)^2$.

We now show analytically and empirically that this estimated noise covariance is valid when the motion estimate is non-zero, but small. In the previous section, the motion estimate V was given by the solution to a weighted least squares problem,

$V = (X^T W X)^{-1} X^T W Y$ (Eq. 4.6). To simplify notation, let $B = (X^T W X)^{-1} X^T W$, the parts of the equation that don't depend on time. Then, the flow estimate is

$$V = BY. \tag{4.10}$$

where the elements of Y are the local phase changes over time.

We first show that if the motions in the scene are small ($u(x, y, t), v(x, y, t) << 1\text{px}$) and there is some image content at the point, it is reasonable to assume that Y is stochastic and that B is fixed. B depends on the local phase gradients and amplitudes of the complex steerable pyramid representation of the video. We show that points with image texture, these quantities are much less noisy than the temporal phase changes in Y.

We first demonstrate that the local phase gradients are much less noisy than the local phase changes. We look at the linearization of phase equation again (Eq. 4.4) with noise terms added to the phase variations (n_t) and phase gradient (n_x, n_y):

$$\Delta\phi_{r,\theta} + n_t = (u, v) \cdot \left(\frac{\partial \phi_{r,\theta}}{\partial x} + n_x, \frac{\partial \phi_{r,\theta}}{\partial y} + n_y \right). \tag{4.11}$$

The total noise term in this equation is $n_t - u n_x - v n_y$. The noise terms n_t, n_x and n_y are of the same order of magnitude. Since u and v are less than 1px, the predominant source of noise is from n_t and the effects of n_x and n_y are negligible. As long as some of the amplitudes of the complex steerable pyramid are large (i.e. there is image texture at the point), the amplitudes and therefore B will be stable as sensor noise varies. Therefore, we model Y, which consists of temporal phase differences, as stochastic and B as fixed.

We split Y into the sum of its mean Y_0 and variance, a multivariate Gaussian random variable, denoted as S, that has zero-mean and variance that depends only on image

noise and local image content (Appendix. B). Then, the flow estimate is

$$V = \underbrace{BY_0}_{\text{True flow}} + \underbrace{BS}_{\text{Noise Term (Covariance Matrix)}} \tag{4.12}$$

The noise term doesn't depend on the value of the true flow BY_0. Therefore, the covariance matrix we compute via our simulation is valid even when the motions are non-zero, but small. In the appendix, we show that the noise term depends only on local image content. Our analysis is slightly inaccurate for components of the variance that are very large. Such components correspond to points where the amplitudes of the pyramid representation are low and noisy. However, for these points, all that matters is that the variance is large ($\sim 1\text{px}^2$) and the motions are not recoverable in that direction. When the variance is smaller, our analysis is accurate.

We also performed a synthetic experiment in Fig. 4.4. We computed the sample covariance matrix of the motions between a frame and translated versions of it. The covariance matrix is approximately constant for motion sizes less than a pixel. There are small variations of around 5% due to the random nature of the simulation. For larger motions of size greater than one pixel, the covariance matrix is within 15% of the covariance matrix computed when the frame doesn't move at all.

Direction of Most Confidence We can rotate the covariance matrix to compute the directions of most and least confidence, denoted by the unit vectors D_M and D_LM respectively [42]. If we project the motion vector V onto these directions, we get components of the motion $V_M := VD_M^T$ and $V_L := VD_L^T$ that we are most confident of (has least variance) and least confident of (has most variance), respectively. This procedure is useful when the motion is computed at a 1D structure, such as an edge in the video. In such cases, we will only be able to reliably recover one dimension of the motion due to the aperture problem. Rotating the covariance matrix allows

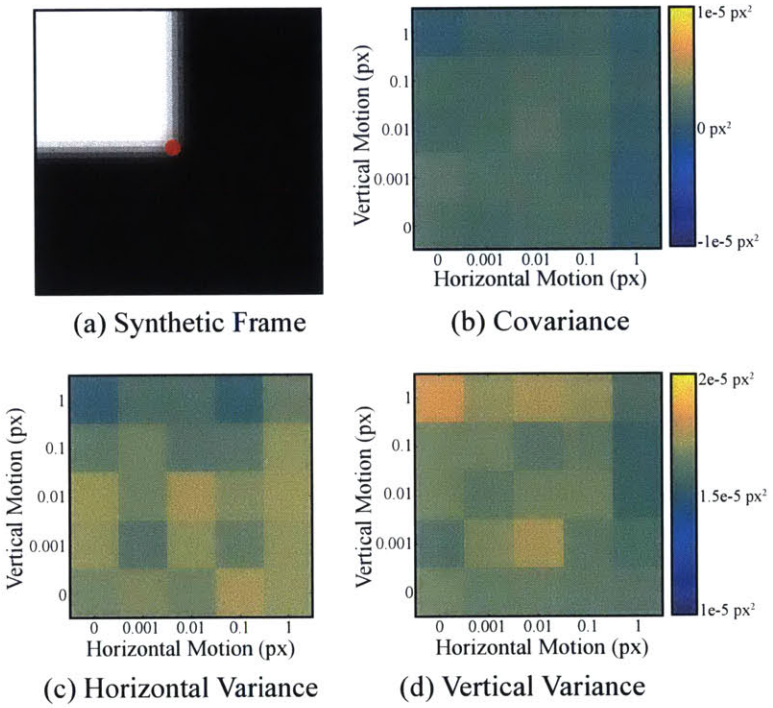
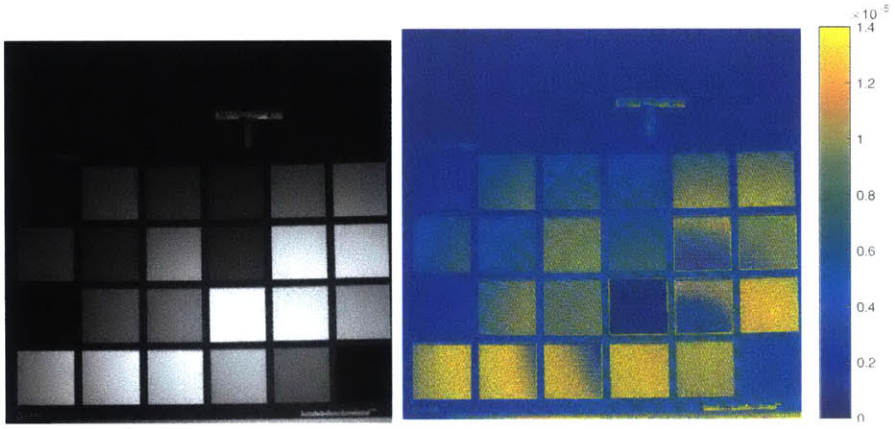
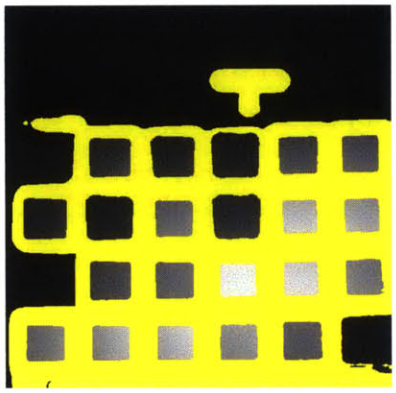


Figure 4.4: Variances and covariance of estimated motion are approximately constant vs. motion size. This means our noise covariance estimation, which assumes that the motions are zero, is also accurate for small non-zero motions. The motion between a synthetic frame (a) and slightly translated versions (not shown) at the marked point in red is computed 4000 times for several different translation amounts. Each time a different, but independent noise is added to the frames. The sample covariance (b) and variances (c-d) are shown as a function of motion size. (c) and (d) are on the same color scale.

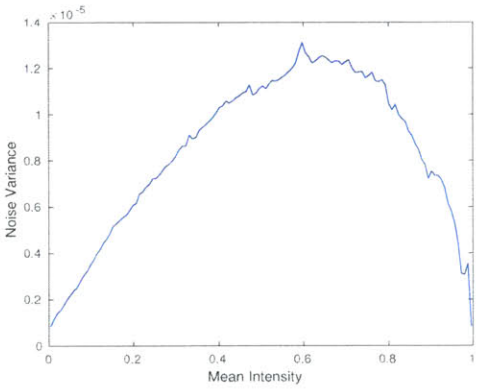


(a) Frame from Input Video

(b) Temporal Variance



(c) Edges marked in yellow removed



(d) Estimate of Noise Level Function

Figure 4.5: Estimating sensor noise from a nearly static video. A frame from a video is shown (a). The variance of pixel intensities over times is then computed (b). Pixels with strong spatial gradients (edges) are computed (c). The remaining pixels are used to compute a noise level function mapping input intensity to noise variance.

us to separate the reliable part of the motion from the unreliable part, whereas the horizontal and vertical components might both be noisy in the case of a diagonal edge. The variance in the direction of most confidence also gives us a noise floor. Its square root roughly tells us the magnitude of the smallest motion we can hope to recover at a pixel.

Sensor Noise Estimation

We estimate the amount of noise present in the input video. As mentioned in the related work (Sec. 4.2), we use a Gaussian signal-dependent noise model, in which the noise at each pixel is independent, but has variance that is a function $f(I)$ of intensity. We estimate this noise level function f.

At each pixel, we compute the temporal sample variance of its intensities (Fig. 4.5b). Were the intensity changes solely due to noise, this sample variance would be an unbiased estimate of the noise variance at that pixel and we could use the sample variances at all the pixels to estimate f. However, even in a nearly static video, subtle motions can cause intensities to vary, causing the sample variance to be an overestimate of the noise variance. We solve this problem by observing that subtle motions do not cause substantial temporal changes in regions of the image with low spatial gradients. To show this, suppose that the video has observed intensities $I(x, y, t)$ that are the sum of its noiseless, true intensities $I_0(x, y, t)$ and a Gaussian noise term $n(x, y, t)$. That is

$$I(x, y, t) = I_0(x, y, t) + n(x, y, t). \tag{4.13}$$

Under the assumption that the subtle motions $(u(x, y, t), v(x, y, t))$ cause brightness values to translate, we have

$$I_0(x, y, t) = I_0(x - u(x, y, t), y - v(x, y, t), 0). \tag{4.14}$$

Because the motions are small, we ignore higher order terms and linearize this around (x, y) to show that the intensity variations are given by

$$I_0(x, y, t) \approx I_0(x, y, 0) - \frac{\partial I_0}{\partial x} u(x, y, t) - \frac{\partial I_0}{\partial y} v(x, y, t). \tag{4.15}$$

This is similar to the linearization of brightness constancy [41, 54] and tells us that in smooth flat regions where the spatial gradients of I_0 are small, the sequence I_0 is roughly constant over time and the sample variance of I will be mostly due to the noise term n.

Therefore, when estimating the noise level function, we ignore pixels where the spatial gradient magnitude, computed using derivative of Gaussian filters, is higher than a threshold (Fig. 4.5c). We then divide the intensity range into bins and for each bin, we compute the mean sample variance of the non-ignored pixels in that bin. This result is our estimate of the noise level function f mapping intensity values to variances (Fig. 4.5d). If low gradient regions span a large enough range of intensities, then it is possible to estimate $f(I)$ from the input video. Otherwise, it may be necessary to take a calibration video of a paper with patches of varying brightness (such as in Fig. 4.5a) with the same camera and camera settings and use that video to estimate the noise level function f.

■ 4.3.3 Suppressing Magnification of Noise

If the motions in a video are of size close to our estimated noise floor, they are probably spurious and we should suppress their magnification. We do this by computing a spatial attenuation map telling us how much to amplify motions. Pixels with low signal-to-noise ratios (SNR) should get amplified less than pixels with high SNR. Formally, we derive a Wiener filter in which we compute the optimal constant p with which to multiply the noisy motion signal to get the mean squared error optimal estimate of the true, noiseless signal [84]. The values of p at each pixel form the desired spatial attenuation map.

At a specific pixel, we look at the observed motion signal $o(t)$ in the direction of most confidence, so that it is 1D. This observation is the sum of a true noiseless motion signal

$s(t)$ and noise $n(t)$, with standard deviation σ_n^2, computable with our noise analysis. That is,

$$o(t) = s(t) + n(t). \tag{4.16}$$

We model the signal $s(t)$ as the product of a univariate Gaussian random variable P that is uncorrelated with the noise and a non-stochastic function $s_0(t)$ that has average unit power, i. e. $\frac{1}{M}\sum_t s_0(t)^2 = 1$, where M is the number of samples in time. The mean squared error function function we seek to minimize is

$$\arg\min_p \mathbf{E}\left[\frac{1}{M}\sum_t (po(t) - s(t))^2\right]. \tag{4.17}$$

This can be solved explicitly:

$$\arg\min_p \mathbf{E}\left[\frac{1}{M}\sum_t (po(t) - s(t))^2\right] = \tag{4.18}$$

$$\arg\min_p \mathbf{E}\left[\frac{1}{M}\sum_t (p-1)^2 s(t)^2 + 2(p-1)ps(t)n(t) + p^2 n(t)^2\right] \tag{4.19}$$

$$\arg\min_p (p-1)^2 P^2 + p^2\sigma_n^2 = \tag{4.20}$$

$$\frac{P^2}{\sigma_n^2 + P^2}. \tag{4.21}$$

The equality between Eq. 4.19 and Eq. 4.20 follows because the signal and noise are uncorrelated by assumption and the noise has zero-mean. The solution to this objective (Eq. 4.21) is the ratio of signal power P^2 to total power of the observation $P^2 + \sigma_n^2$ and is close to one when signal power is much larger than noise and close to zero otherwise. We showed how to estimate the noise power σ_n^2 in the previous section. Since P^2 is not known in advance, we estimate it by taking the observation power and subtracting the

noise power σ_n^2. That is, we estimate P^2 by

$$\max(\frac{1}{M}\sum o(t)^2 - \sigma_n^2, 0). \tag{4.22}$$

We perform this analysis for the time-series of displacements in the direction of most confidence at every pixel to produce an attenuation map. We then modify motion magnification by multiplying this map by the local phase variations in every level of the complex steerable pyramid to produce a motion magnified video in which noisy regions are not motion magnified.

■ 4.4 Results

We validate the accuracy of our motion estimate and our noise analysis on both real and synthetic data and compare it to NCORR, a digital image correlation package [5]. We also show that phase-based motion magnification generally works better than advecting pixel color values according to magnified motion vectors. And finally, we demonstrate the utility of motion magnification and our new capabilities, a quantitative estimate of the motions and an analysis of noise, on applications in biology and mechanical engineering.

To compute motions, we use a four orientation complex steerable pyramid. We use only the two highest frequency scales of the complex steerable pyramid. We found this to be a good trade-off between resolution of the motion field (low frequency levels are lower resolution) and accuracy (using more levels increases the information used). Because we use high frequency scales, the phase differences can exceed π if the motion is greater than 2-3 pixels in size. To prevent artifacts due to phase wrap-around, we unwrap the phases in time. This solution works as long as the frame-to-frame motions are less than 2-3 pixels in size, which is the case for *every* video presented in this dissertation. We also detect regions of local phase that don't follow our phase constancy assumption

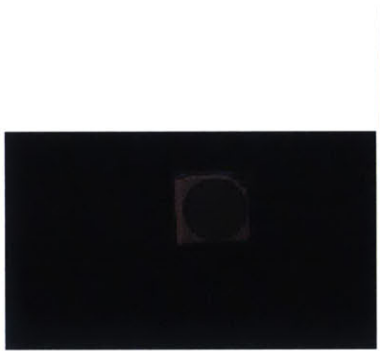
(Eq. 4.2) using a method proposed by Fleet et al. [30]. As suggested by them, we reject regions where the amplitude changes too rapidly or the spatial gradient of the local phase is too far from the peak frequency of the corresponding filter.

■ 4.4.1 Validation of Motion Estimation

We validate the accuracy of our motion estimation with two experiments: one with real data, in which we obtain ground truth motions using a laser vibrometer, and one with synthetic data, in which the ground truth motions are known. In both experiments, we use a Gaussian blur with standard deviation 3px as our spatial smoothing weighting function and we do not employ temporal filtering.

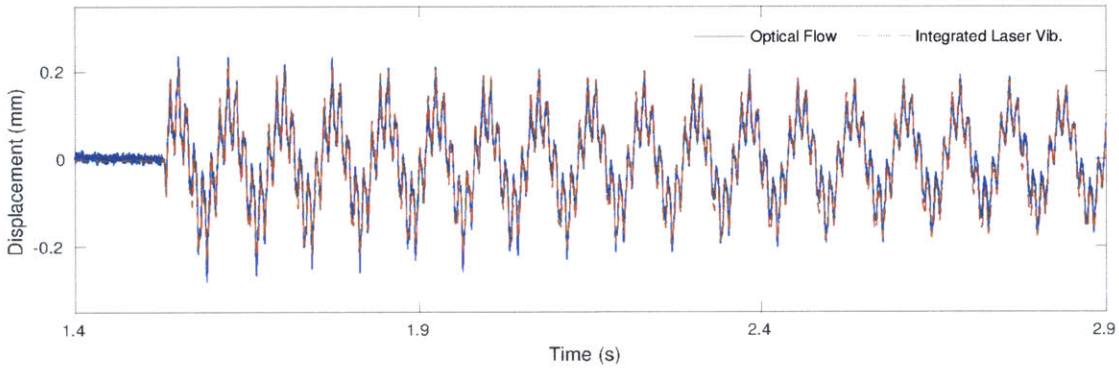
In the first experiment, a cantilevered beam was hit with a hammer to induce subtle vibrations in it. We simultaneously recorded a 5000 FPS video of the beam with a high-speed camera and measured its horizontal velocities with a laser vibrometer (Fig. 4.6b). The beam had an accelerometer mounted on it (Fig. 4.6a), but we did not use it to record motion. We used our motion estimation method to compute the horizontal displacement of a point on the left side of the accelerometer from the video. After manually aligning this signal to the ground truth integrated laser vibrometer data, we were able to compare the two signals (Fig. 4.6c-e). The two methods agree remarkably well with mean absolute difference of only 0.012 mm, 5% of the maximum motion amplitude. Even the higher order modes are well-aligned.

In our second experiment, we created a synthetic dataset of image pairs with known ground truth motion between them. We took natural images from the frames of real videos (Fig. 4.7a) and warped them according to known motion fields using cubic b-spline interpolation [76]. Sample motions fields, shown in Fig. 4.7b, were produced by Gaussian blurring IID Gaussian noise. We used Gaussian blurs with standard deviations (SD), ranging from zero (no filtering) to infinite (a constant motion field). We also varied

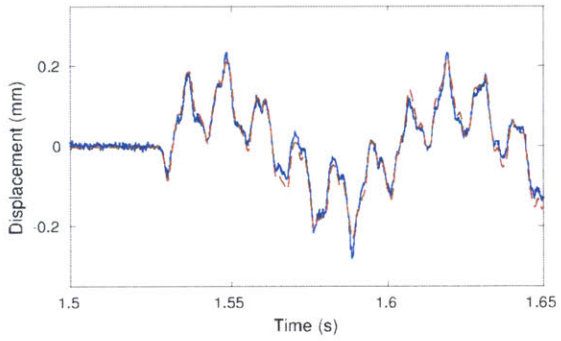


(a) Frame from Input Video

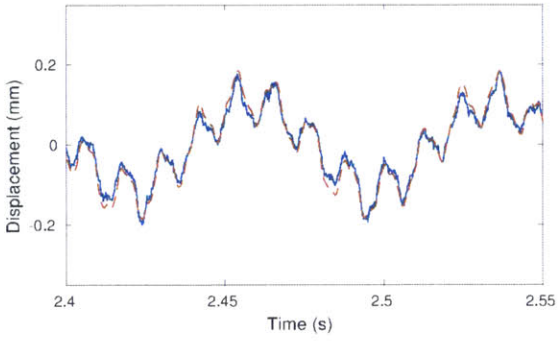
(b) Experimental Setup



(c) Displacement signal on Left Side of Accelerometer



(d) Zoom-in of (c)



(e) Zoom-in of (c)

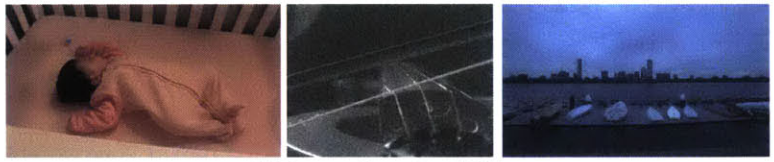
Figure 4.6: Validation of our motion estimation on real data. A frame from the recorded video (a) and the experimental setup (b). Overlayed plots of the displacements of a cantilevered beam, computed using our motion estimation technique and by integrating laser vibrometer velocity measurements (c-e). (Experiment performed with Justin G. Chen, Oral Buyukozturk and colleagues; data taken from [11].)

the root-mean-square (RMS) amplitude of the motion fields from 0.001px to 3px. For each set of motion field parameters, we sampled five different noise patterns to produce a total of 155 motion fields. This gives us a way to test the accuracy of our motion estimate as a function of the motion's spatial coherence and amplitude. We did not add noise to the image pairs, so that we could test the accuracy of our algorithm rather than its sensitivity to noise.

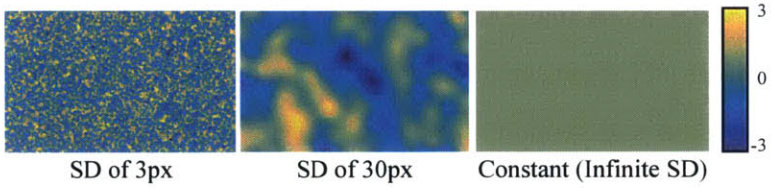
We use this synthetic dataset to compare the accuracy of our motion estimation algorithm to NCORR [5], a digital image correlation software package used by some mechanical engineers [86]. For each image pair, we compute the mean absolute difference between the estimated and ground truth motion fields. Then, for each set of motion field parameters, we average the mean absolute differences across image pairs and divide the result by the RMS motion amplitude to make the errors comparable over motion sizes. The result is the average relative error as a percentage of RMS motion amplitude (Fig. 4.7c).

Both NCORR and our method assume that the motion field is reasonably smooth and perform best when the motions are spatially coherent (filter standard deviations greater than 10 px) with relative errors under 10%. Our method assumes that the motions are small, e.g. when we linearize phase constancy (Eq. 4.4). This is reflected in this experiment; our method performs best for sub-pixel motions (5% relative error). NCORR has twice the relative error (10%) for the same motion fields. NCORR has surprisingly low error when the motions are of constant integer size.

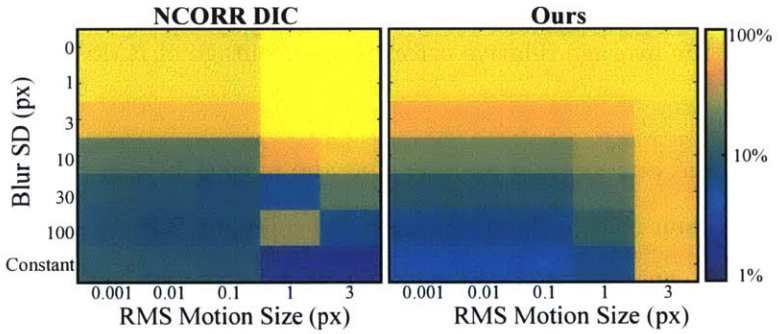
The relative errors reported in Fig. 4.7c are computed over all pixels including those that are in smooth regions where the aperture problem makes it difficult to estimate the motions. If we restrict the error metric to only take into account pixels at edges and corners, the average relative errors for small (< 1px), spatially coherent (filter SD > 10px) motions drops by a factor of 2.5 for both methods.



(a) Frames from Real Videos



(b) Motion Fields are Gaussian Blurred Gaussian Noise



(c) Average Relative Errors

Figure 4.7: An evaluation of our optical flow estimation method and NCORR [5] on a synthetic dataset of images. Frames from real videos (a) were warped using motion fields (b) of various motion size and spatial scale. Our optical flow estimation method and NCORR are used to estimate the motion field and the average relative error is displayed for both methods as a function of motion size and spatial scale. Both methods are only accurate for spatially smooth motion fields. Our method is twice as accurate for spatially smooth, sub-pixel motion fields. NCORR is more accurate for larger motions.

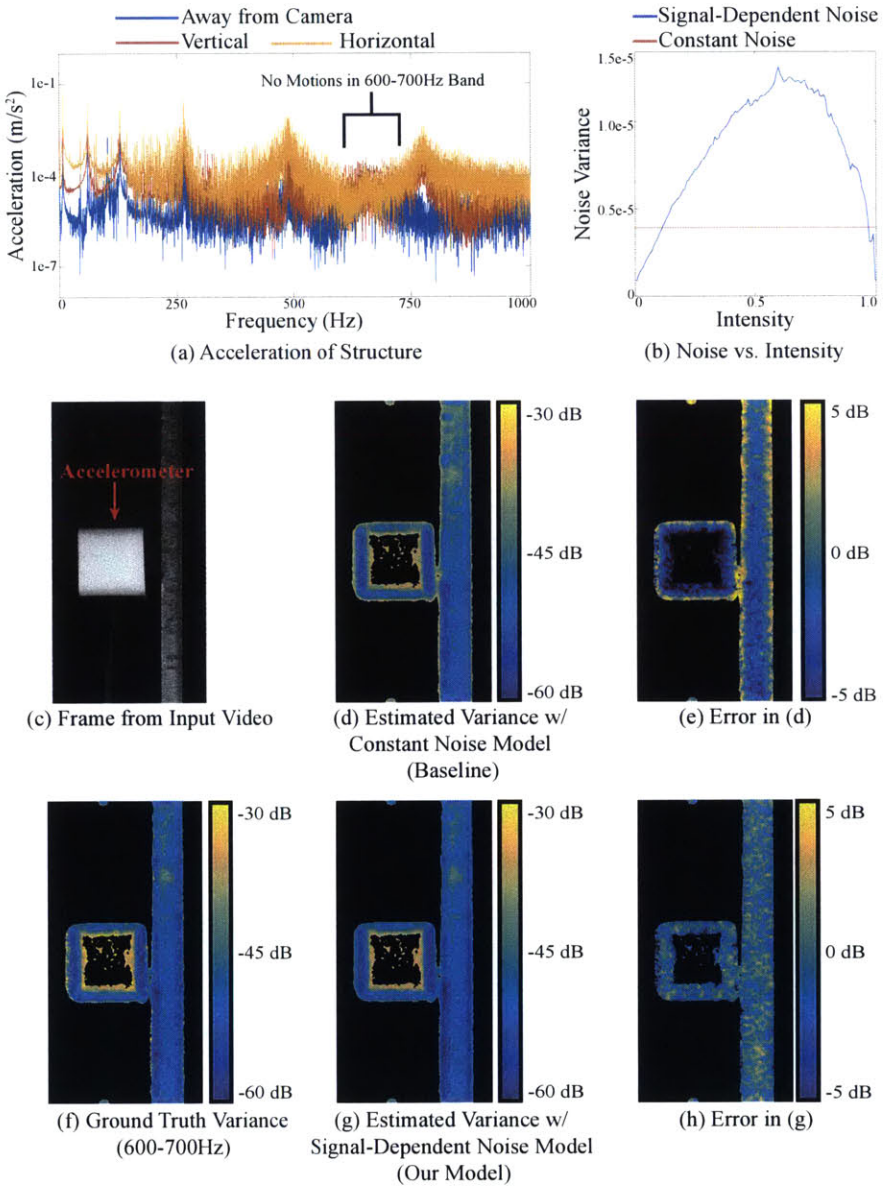


Figure 4.8: Validation of our noise estimation on real data. In a real video (c), there are no motions in the frequency band 600 to 700 Hz (a). The variance of our motion estimate is entirely due to noise and serves as ground truth (f). Estimates of the noise variance using our Monte Carlo simulation using two different noise models (b) are shown in (d) and (g). The difference in decibels between the ground truth variances and the estimated variances are shown in (e) and (h). All variances are of the motions projected to the direction of most confidence. Textureless regions, where the motion estimation is not meaningful, have been masked out in black.

4.4.2 Validation of Noise Analysis

We performed an experiment to validate our noise analysis and to demonstrate the advantages of using a signal-dependent noise model over a constant variance noise model (Sec. 4.2). Specifically, we filmed a cantilevered beam and used a mounted accelerometer to show that it was not moving in the temporal band spanning 600 to 700 Hz (Fig. 4.8a-c). Because there are no motions in this band, any motions we detect are due to noise and we can get a ground truth measurement of the covariance matrix by computing the estimated motions' sample covariance over time in this band. In Fig. 4.8f, we show a component of the covariance matrix, specifically the variance of the motions in the direction of most confidence. Textureless regions, where we cannot measure the motions accurately, have been masked out.

We use our Monte Carlo simulation to estimate the covariance matrix using both a constant noise variance model (not our method) and a signal-dependent variance model (our method). We show the same component (variance of motions in direction of most confidence) of the covariance matrix for each noise model in Fig. 4.8d and Fig. 4.8g, respectively. Both methods produce results that are comparable to the ground truth noise variance. However, if we take the difference between the log of the estimated variances and the log of the ground truth variances (Fig. 4.8e,h), we see that the signal-dependent noise model is more accurate than the constant variance noise model. The constant variance noise model overestimates noise in the black areas of the input image. As a consequence, the predicted covariance is higher than the ground truth covariance in areas near the black background. (Fig. 4.8e). The signal-dependent noise model has errors that are less than 1 dB on average, demonstrating the accuracy of our noise estimation technique.

We also validate our noise analysis on the synthetic example we presented at the beginning of this chapter (Fig. 4.2). In that example, we created a video where all

motions were spurious. We estimated the motions and noise covariance matrix at all pixels. The motion estimate in the direction of most confidence is within three standard deviations of zero at 97.8% of the pixels in the video. This means that almost all of the motions are non-zero by chance and that our noise analysis correctly identifies them as spurious.

■ 4.4.3 Phase-Based Motion Magnification vs. Advecting Color Values

Because we now compute motion vectors, it is natural to ask whether we can produce motion magnified videos by simply advecting pixel color values according to magnified motion vectors. In this section, we show that phase-based motion magnification (Chapter 2) produces higher-quality videos than the just-described algorithm, regardless of whether our own motion estimate or NCORR [5] is used (Fig. 4.9).

To show this, we take an apparently still video of an infant (Fig. 4.9a) and motion magnify it (20x, 0.5-3Hz) in four ways. First, we magnify the baby's breathing motions using phase-based motion magnification, in which local phase variations are computed, amplified and then used to phase-shift complex steerable pyramid coefficients (Fig. 4.9b). In our second method, we instead use our phase-based motion estimation to compute motion vectors, magnify them and then advect pixel color values accordingly (Fig. 4.9c). We performed the advection by creating a triangular mesh with a vertex at the advected location of each pixel, assigning each vertex the pixel's original color and then using barycentric interpolation to color in the triangular faces. This technique also reveals the baby's breathing motion, but it produces very visible artifacts in regions where the motion estimate is noisy and inaccurate, such as the baby's hair or the crib walls. Pixels in these regions are pushed too far.

To confirm that advecting pixel color values causes these artifacts, we replace our phase-based motion estimate algorithm with NCORR [5]. We use NCORR's motion

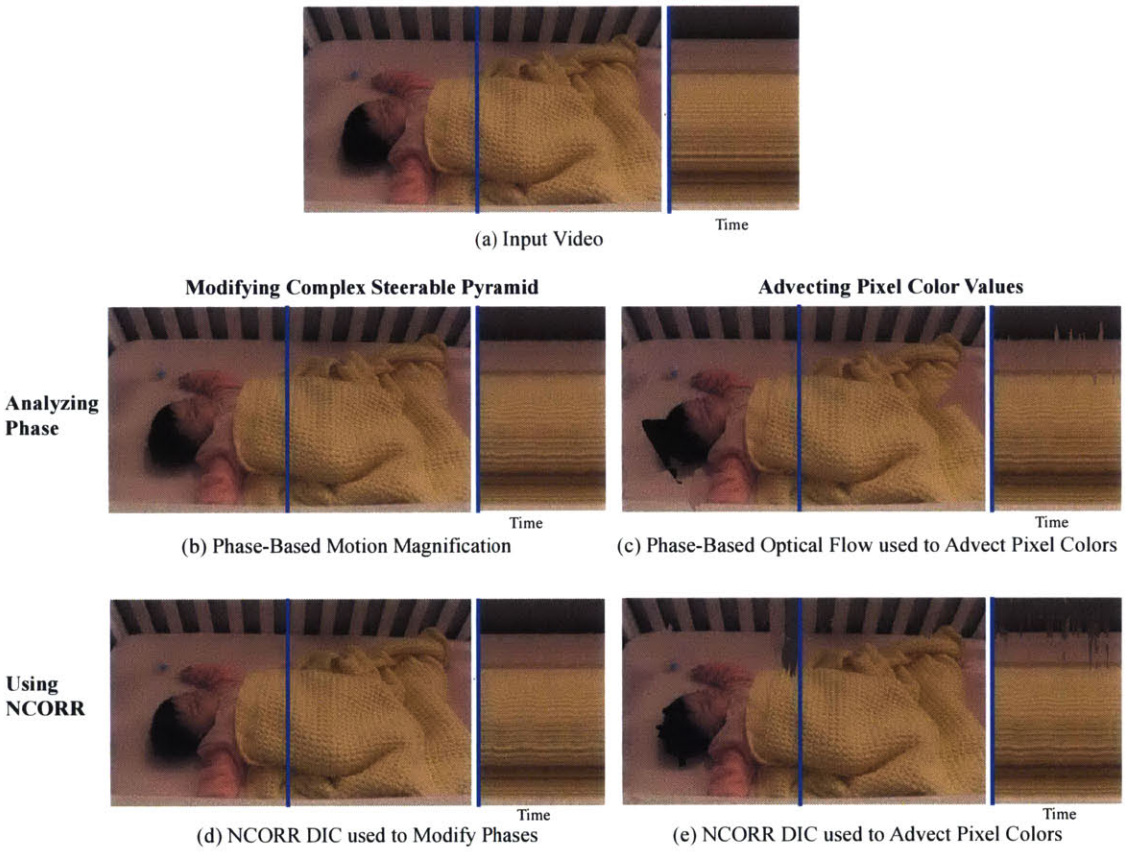


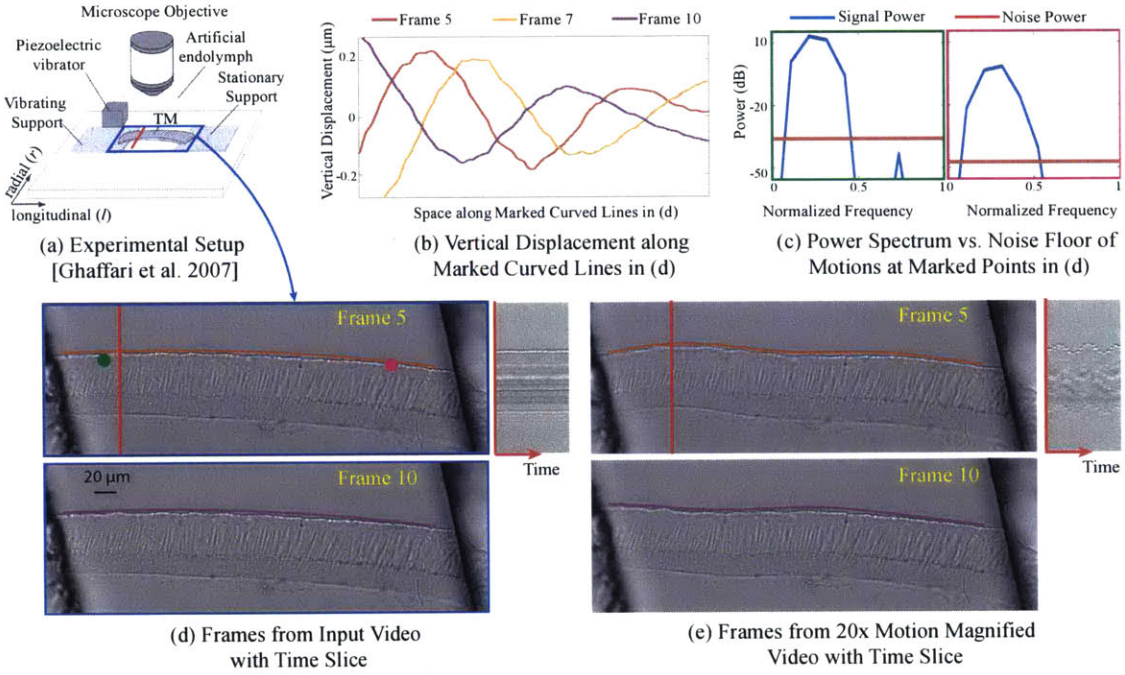
Figure 4.9: Different ways to motion magnify videos. The input video (a) is motion magnified 20x (0.5-3Hz) using phase-based motion magnification (b). Its motions are also computed using our phase-based optical flow. The resulting motion vectors are used to advect pixel colors (c). Similarly, the motions are computed using NCORR [5]. A magnified version of the resulting motion vectors is used to modify the phase in complex steerable pyramid coefficients (d) and also used to advect pixel values (e). Note the artifacts near the baby's head and along the crib wall in (c) and (e).

estimates to magnify the video in two different ways. First, we use the motion vectors to phase-shift coefficients in a complex steerable pyramid representation of each frame. As suggested by the linearization of the phase constancy equation (Eq. 4.4), we phase-shift each coefficient by the dot-product of the magnified motion vector and the spatial gradient of the local phase (Fig. 4.9d). This video is nearly artifact-free and looks almost identical to the one produced using phase-based motion magnification (Fig. 4.9b) suggesting that phase-shifting complex steerable pyramid coefficients produces better results than advecting pixel color values. Second, we use NCORR's motion vectors to advect pixel color values in the same manner as described before (Fig. 4.9e). The same kind of artifacts occur in smooth regions, where the motion estimate cannot be accurately computed due to the aperture problem.

A possible reason for the improved synthesis in the complex steerable pyramid representation is that it has an implicit notion of when components of the motion estimate will be inaccurate. The coefficients in smooth regions and in orientations parallel to edges, where components of the motion are inaccurate, have low-amplitude. Even if they are phase-shifted excessively, they don't substantially affect the motion-magnified result. In addition, the complex steerable pyramid basis functions are bounded and can only shift image features so far (Sec. 2.3.5).

■ 4.4.4 Mammalian Tectorial Membrane

Motion magnification has been used to understand the mechanical properties of the mammalian tectorial membrane (TM), helping lead to the recent discovery of its importance in frequency selectivity during hearing [67]. The mammalian tectorial membrane is a hydrogel that overlies hair cells in the inner ear. When sound waves enter the ear, they mechanically travel through it until they reach the inner ear or cochlea. The final mechanical step in this process is a transmission of energy through the tectorial



(a) Experimental Setup [Ghaffari et al. 2007]

(b) Vertical Displacement along Marked Curved Lines in (d)

(c) Power Spectrum vs. Noise Floor of Motions at Marked Points in (d)

(d) Frames from Input Video with Time Slice

(e) Frames from 20x Motion Magnified Video with Time Slice

Figure 4.10: Exploring the mechanical properties of a mammalian tectorial membrane (TM) with motion magnification and phase-based motion estimation. To simulate the effect of sound, one side of the TM is vibrated while it is stroboscopically filmed under a microscope (a). Frames and a time slice from this video are shown in (d). The vertical displacement along the blue line in (d) is shown for three frames (b). The power spectrum of the motion signal in the direction of most confidence at the marked points in (d) is shown with the computed noise floor (c). All nonconstant motions above the noise floor are magnified 20x in the corresponding frames from the motion magnified video are shown in (c). The blue line on top of the TM in (b) is warped according to magnified motion vectors to produce the orange and purple lines in (c). ((a) from Ghaffari et al. [35] and data from Sellon et al. [67].)

membrane to hair cells that convert the mechanical energy into electrical energy that the brain can interpret.

Sellon et al. [67] performed experiments, in which they took tectorial membranes from mutated mice exhibiting different phenotypes in frequency selectivity. They placed the membranes on two supports and then vibrated the left side to simulate the hearing process in the inner ear (Fig. 4.10a). The membranes were filmed stroboscopically to make their fast motions appear slower. However, the videos were difficult to interpret because the motions were too small to see. Frames and a time slice from one of the

videos they took are shown in Fig. 4.10d. We use our motion estimate to compute motion vectors at every point on the membrane. In Fig. 4.10b, we plot the vertical displacement of the top of the membrane from its mean location. This quantitatively shows the shape of the traveling wave for several frames. We also use our noise analysis to determine which temporal frequencies are above the noise floor. This is shown in plots of the noise floor and power spectrum of the motions at a few points (Fig. 4.10c). We then amplified all nonconstant motions above the noise floor by twenty times. In practice, this was roughly everything above the DC component and less than half the Nyquist frequency. The result is shown in Fig. 4.10e, in which the shape of the traveling wave moving through the membrane is revealed.

In Sellon et al. [67], the authors used optical flow estimation from the videos to show that the tectorial membranes from mice with different mutations have different traveling waves shapes with differing numbers of nodes and decay rates. Motion magnification made it possible for the authors to interpret these numbers and assisted in determining whether the waves were radial or longitudinal. The different shapes of the traveling waves coupled with the different mice phenotypes allowed Sellon et al. to determine that the TM plays an important role in hearing frequency selectivity.

In other experiments, Sellon et al. reported using motion magnification to reject data. Small tears in the membrane caused strange modes of wave propagation. Motion magnification allowed the authors to identify these quickly and discard the torn samples.

■ 4.4.5 Metamaterials

Acoustic metamaterials are artificially structured composites of materials that have the special property that forcing vibrations within a range of temporal frequencies, the *bandgap*, are dramatically attenuated as they travel. Recently, a class of metamaterials was introduced in which the bandgap is tunable at use-time [83] (Fig. 4.11a). This can

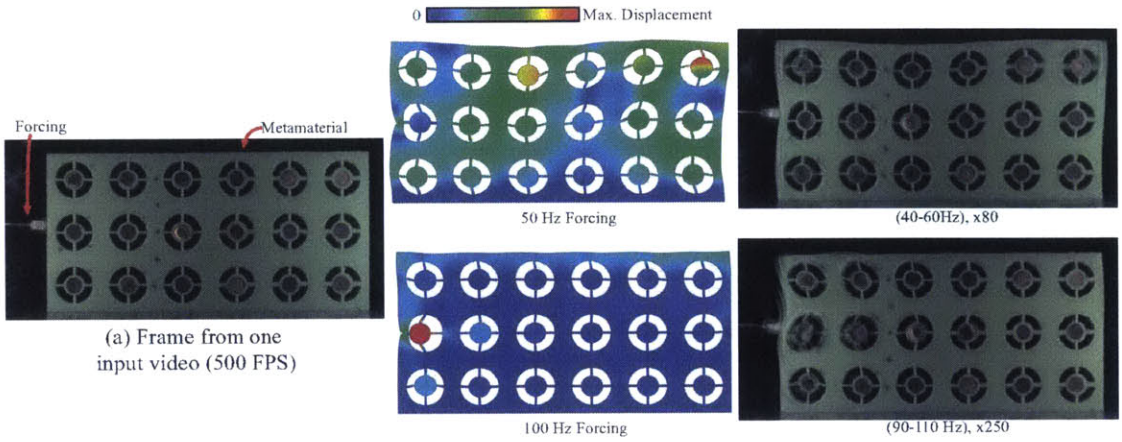
be useful in making things like acoustic on and off switches.

Verifying that the metamaterial is designed properly is difficult. Evidence of the successful attenuation of vibration at a single location can be shown with a mounted accelerometer or a spot laser vibrometer. Mounting a bank of accelerometers onto the metamaterial to densely measure its vibrations is tedious and could affect its dynamics. The laser vibrometer is only able to measure the vibrations at the edges of the metamaterial. Visually verifying the attenuation with the naked eye is not possible because the motions are too small to see.

Motion magnification and the techniques described in this chapter are the natural solution to this problem. To demonstrate this, we conducted an experiment with a metamaterial designed by Wang et al. [83]. We placed a forcing probe next to the metamaterial and vibrated it at 50Hz, a frequency outside of the bandgap, and 100Hz, a frequency at the center of the bandgap. We took two 500 FPS videos, one for each case. A frame from the 50Hz video is shown in Fig. 4.11a.

We motion magnified each video in a 20Hz band of frequencies centered on its corresponding forcing frequency. The motion magnified videos (Fig. 4.11c) show that in the 50Hz case, the metamaterial moves a lot, allowing the vibrations to pass. In the 100Hz case, the motions dramatically attenuate with distance from the forcing probe, showing that the metamaterial is working properly. The motion magnified videos are also well-correlated with finite element analysis simulations of the metamaterial under the forcings (Fig. 4.11b).

We use our motion estimation and noise analysis to further analyze the video of the metamaterial forced at 100Hz. We compute the signal power, the noise power and the signal-to-noise ratio (SNR) for every pixel in the local direction of most confidence (Fig. 4.12a-c). This allows us to confirm quantitatively that the motions attenuate as they propagate through the metamaterial.



(a) Frame from one input video (500 FPS)

(b) Simulated motions of metamaterial

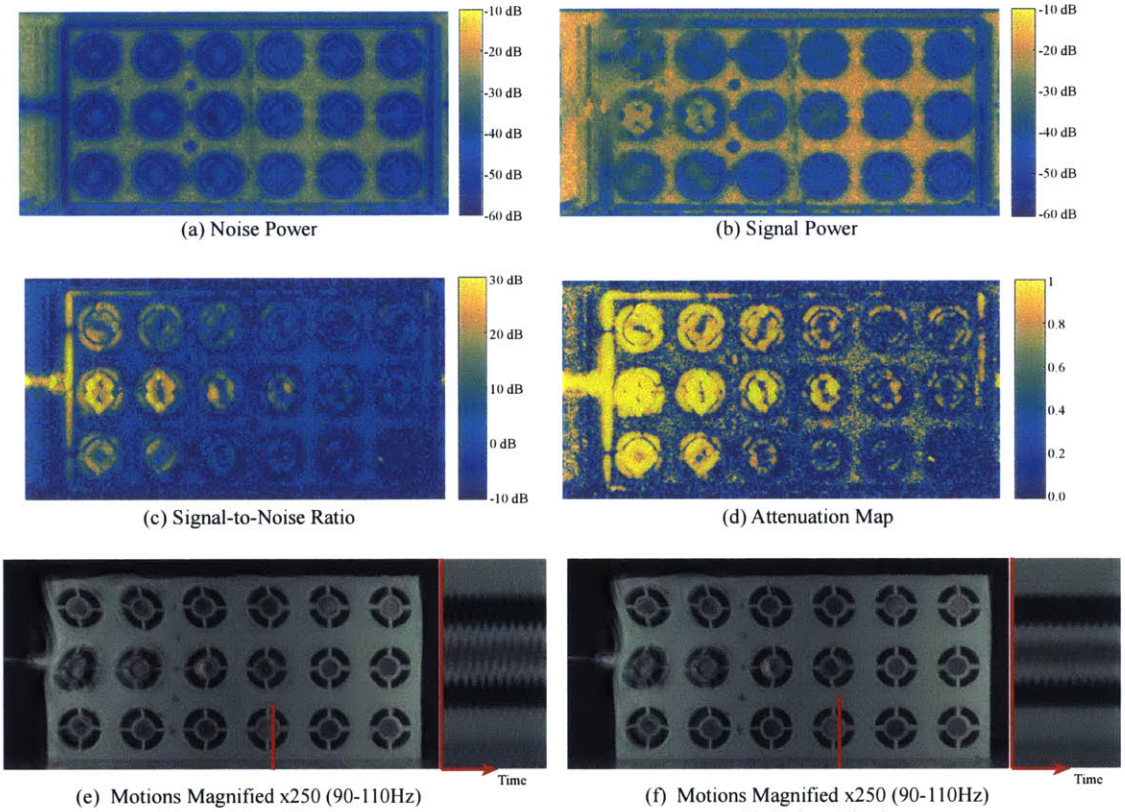
(c) Frames from motion magnified videos

Figure 4.11: Motion magnification applied to a metamaterial with a forced vibration and a comparison to simulation [83]. A probe forces a metameterial to vibrate and the result is filmed. One frame from one of the input videos is shown (a). The simulated displacements with a zero displacement boundary condition on the side of the metameterial touching the table is shown (b) for a 50Hz forcing and a 100Hz forcing. Frames from the motion magnified video are shown in (c). They match the simulation closely. Video filmed with the assistance of Donglai Wei and Pai Wang in the Bertoldi Lab.

Some regions of the motion magnified video look similar to the synthetic example presented earlier in this chapter, in which we motion magnified a noisy, motionless video (Fig. 4.2). This suggests that some of the motions in this video are spurious. To handle this, we use our technique for suppressing the magnification of noise (Sec. 4.3.3). We use the signal and noise powers to compute an attenuation map (Fig. 4.12d), multiply it by the local phase variations in every level of the complex steerable pyramid and amplify. Areas with spurious motions where the SNR is low, such as the structures on the right or bottom of the video, no longer get amplified (Fig. 4.12e,f).

■ 4.5 Discussion and Limitations

In our analysis, we only model sensor noise. While we have demonstrated this is an important source of errors and is worth understanding, other sources of noise could also



(a) Noise Power

(b) Signal Power

(c) Signal-to-Noise Ratio

(d) Attenuation Map

(e) Motions Magnified x250 (90-110Hz) without Weiner Filtering

(f) Motions Magnified x250 (90-110Hz) with Weiner Filtering

Figure 4.12: Using our motion and noise estimates to suppress amplification of noisy regions. The noise power, signal power and their ratio is shown (a-c). The optimal Wiener filter coefficient for each time series is shown at every pixel (d) and the result of doing plain motion magnification vs. attenuating the motion signal by the Wiener filter coefficient is shown in a single frame and timeslices (e-f). Video filmed with the assistance of Donglai Wei and Pai Wang in the Bertoldi Lab.

affect our motion analysis or visualization of tiny motions. Atmospheric turbulence could make it difficult to see motions of objects that are very far from the camera. Other sources of unwanted motions like subtle camera motions could also be problematic, but it might be possible to separate them from interesting subtle motions if they occur in different temporal bands.

We don't model video compression in our analysis of sensor noise. When a video is compressed, its noise characteristics change and this could affect the accuracy of our noise estimation. However, cameras that produce uncompressed videos are getting more common. In addition, our processing would still be useful in designing a physical motion microscope consisting of a camera and a computer, in which motion magnification is applied before video compression.

Chapter 5

Geometric Deviation Magnification

In this chapter, we focus on another type of ideal model: elementary geometries, such as lines, circles or ellipses in space. Many objects, such as buildings, planets, bubbles and more, appear to take the shape of these ideal geometries in images. However, the deviations between these objects and their ideal geometries is often very interesting.

At first glance, this problem seems similar to magnifying tiny motions in videos. Can you just look at the color changes (or local phase changes) along the ideal geometry and amplify them? It turns out that this simple processing no longer works because the changes in these quantities along the contour of an object in space are often due to much more than just geometric deviations. They also arise because of things like texture variations and illumination changes. We describe our solution to this problem and our method of amplifying geometric deviations in single images. We also demonstrate the usefulness of this algorithm on a wide array of challenging, natural images.[1]

■ 5.1 Introduction

Many phenomena are characterized by an idealized geometry. For example, in ideal conditions, a soap bubble will appear to be a perfect circle due to surface tension, buildings will be straight and planetary rings will form perfect elliptical orbits. In real-

[1]This chapter is largely reproduced from our SIGGRAPH Asia 2015 paper Deviation Magnification: Revealing Departures from Ideal Geometries with co-authors Tali Dekel, Donglai Wei, Frédo Durand and William T. Freeman.

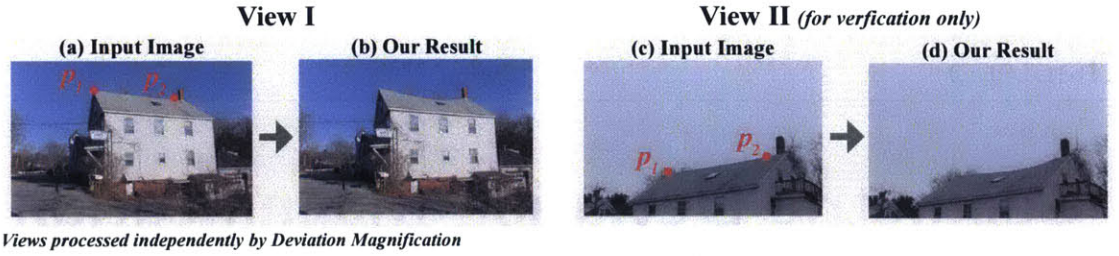


Figure 5.1: Revealing the sagging of a house's roof from a single image. A perfect straight line marked by p_1 and p_2 is automatically fitted to the house's roof in the input image (a). Our algorithm analyzes and amplifies the geometric deviations from straight, revealing the sagging of the roof in (b). View II shows a consistent result of our method (d) using another image of the same house from a different viewpoint (c). Each viewpoint was processed completely **independently**.

ity, however, such flawless behavior hardly exists, and even when invisible to the naked eye, objects depart from their idealized models. In the presence of gravity, the bubble may be slightly oval, the building may start to sag or tilt, and the rings may have slight perturbations due to interactions with nearby moons. We present *Geometric Deviation Magnification*, a tool to estimate and visualize such subtle geometric deviations, given only a single image as input. The output of our algorithm is a new image in which the deviations from ideal are magnified. Our algorithm can be used to reveal interesting and important information about the objects in the scene and their interaction with the environment. Fig. 5.1 shows two independently processed images of the same house, in which our method automatically reveals the sagging of the house's roof, by estimating its departure from a straight line.

Our approach is to first fit ideal geometric models, such as lines, circles and ellipses, to objects in the input image, and then look at the residual from the fit, rather than the fit itself. This residual is then processed and amplified to reveal the physical geometric departure of the object from its idealized shape. The methods described in the previous chapters follow the same paradigm, but magnify small motions over *time*, revealing deviations from perfect stationarity in nearly still video sequence. Here, however, we

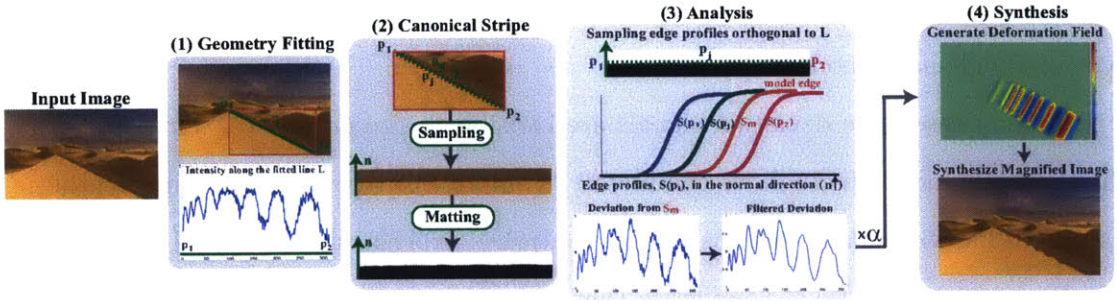


Figure 5.2: Outline of Geometric Deviation Magnification: a parametric shape (e.g., a line segment) is fitted to the input image (either automatically or with user interaction). The region near the contour of the shape is sampled and transformed into a canonical stripe representation. The alpha matte of the stripe is then computed using [48] and then fed into the analysis step. In this step, deviation from the fitted shape is computed: the edge profiles $S(p_j)$ in the vertical direction are sampled for each location p_j along the stripe, and a model edge profile S_m is estimated; the 1D translation between the edge profiles $S(p_j)$ and S_m is estimated to form the deviation signal. The filtered deviation signal is then magnified by a factor of α and used to generate a deformation field. The synthesized image is rendered accordingly and reveals the spatial deviation from the fitted shape. In this case, the periodic ripples of the sand dune ridge are revealed. (Image courtesy of Jon Cornforth.)

are interested in deviations over *space* from canonical geometric forms, using only a single image. Our algorithm serves as a microscope for *form deviations* and is applicable regardless of the time history of the changes. For example, we can exaggerate the sag of a roof line from only a single photo without any prior knowledge on what it looked like when it was built or how it changed over time. The important information is the departure from the canonical shape.

Finding the departures from the fitted model is not trivial. They are often very subtle (less than a pixel in some applications), and can be confused with non-geometric sources of deviations, such as image texture on the object. Our algorithm addresses these issues by combining careful sub-pixel sampling, reasoning about spatial aliasing, and image matting. Matting produces an alpha matte that matches the object's edge to sub-pixel accuracy. Therefore, operating on the alpha matte allows us to preserve the deviation signal while removing texture. The deviation signal is then obtained by estimating small changes in the alpha matte's values, perpendicular to the contour of

the shape. The resulting framework is generic, and is independent of the number or type of fitted shape models.

In many cases, the deviation signals are invisible to the naked eye. Thus, to verify that they are indeed factual, we conducted a comprehensive evaluation using synthetic data with known ground truth as well as controlled experiments using real world data. We demonstrate the use of Geometric Deviation Magnification on a wide range of applications in engineering, geology and astronomy. Examples include revealing invisible tilting of a tower, nearly invisible ripple marks on a sand dune and distortions in the rings of Saturn.

■ 5.2 Related Work

Viewed very generally, some common processing algorithms can be viewed as deviation magnification. In unsharp masking, a blurred version of an image is used as a model and deviations from the model are amplified to produce a sharpened image. Facial caricatures are another example of this kind of processing, in which the deviations of a face image from an idealized model, the mean face, are amplified [6].

As discussed in Chapter 2, motion magnification [24, 53, 81, 82, 85] uses the same paradigm of revealing the deviations from a model. However, there is no need to detect the model as the direction of time is readily given. In addition, motion magnification assumes that objects are nearly static i.e. the appearance over time is assumed to be nearly constant. In contrast, the technique discussed in this chapter amplifies deviations from a general spatial curve detected in a single image. The type and location of this curve depends on the application, and the appearance along it may change dramatically posing new challenges for our new problem.

A recent, related method was presented for revealing and estimating internal *non parametric* variations within an image [18]. This method assumes that the image con-

tains recurring patterns, and reveals their deviation from perfect recurrence. This is done by estimating an "ideal" image that has stronger repetitions and a transformation that brings the input image closer to ideal. In contrast, our method relies on parametric shapes within the image, and thus can be applied for images that do not have recurring structures. Our parametric approach allows our algorithm to accurately reveal very tiny, nearly invisible deviations, which cannot be estimated by Dekel et al. [18].

Our method relies on detecting and localizing edges, a problem for which many techniques have been proposed (e.g. [22, 26, 60]). One class of techniques uses the observation that edges occur at locations of steepest intensity and are therefore well-characterized by the peak of derivative filters of the image (e.g. [10, 31, 57]). More recently, several authors have applied learning techniques to the problem of edge detection to better distinguish texture from edges (e.g. [20, 21, 49]). Because the deviations in the images we seek to process are so small, we obtained good results adopting a flow-based method, similar to Lucas and Kanade [54]. Since texture variations can influence the detected edge location, we used image matting [48] to remove them.

■ 5.3 Method

Our goal is to reveal and magnify small deviations of objects from their idealized elementary shapes given a single input image.

■ 5.3.1 Overview

There are four main steps in our method, illustrated in Fig. 5.2. The first step of our algorithm is to detect elementary shapes, i.e., lines, circles, and ellipses, in the input image. This can be performed completely automatically by applying an off-the-shelf fitting algorithm (e.g., [60]) to detect all the elementary shapes in the image. Alternatively, it can be performed with user interaction as discussed in Section 5.3.5.

The detected shapes serve as the models from which deviations are computed.

With the estimated models in hand, we opt to perform a generic spatial analysis, independent of the number and type of fitted shapes. To this end, the local region around each shape is transformed to a *canonical image stripe* (Fig. 5.2(3)). In this representation, the contour of the shape becomes a horizontal line and the local normal direction is aligned with the vertical axis. To reduce the impact of imperfections that are caused by image texture and noise, a matting algorithm is applied on each of the canonical stripes. This step significantly improves the signal-to-noise ratio, and allows us to deal with real world scenarios, as demonstrated in our experiments.

Each canonical matte is analyzed, and its edge's deviation from a perfectly horizontal edge is estimated. This is done by computing 1D translations between vertical slices in the matted stripe, assuming these slices have the same shape along the stripe. For each canonical matte, this processing yields a *deviation signal* that corresponds to the deviation from the associated model shape in the original image, in the local normal direction. Depending on the application, the deviation signals may be low-passed or bandpassed with user-specified cutoffs, allowing us to isolate the deviation of interest.

Lastly, the computed deviation signals are visually revealed by rendering a new image in which the deviations are magnified. Specifically, a 2D deformation field is generated based on the 1D computed deviation signals, and is then used to warp the input image. We next describe these steps in more detail.

■ 5.3.2 Deviations from a Parametric Shape

Consider a synthetic image $I(x, y)$, shown in Fig. 5.3(a), which has an edge along the x-axis as a model of a matted image stripe. This edge appears to be perfectly horizontal, but actually has a subtle deviation from straight (shown twenty times larger in Fig. 5.3(b)). Our goal is to estimate this deviation signal $f(x)$, at every location x

along the edge.

To do this, we look at vertical slices or *edge profiles* of the image I, e.g. the intensity values along the vertical lines A or B in Fig. 5.3(b). We define the edge profile at location x as

$$S_x(y) := I(x, y). \tag{5.1}$$

We assume that with no deviation (i.e., $\forall x\ f(x) = 0$), the edge profiles would have been constant along the edge, i.e., $S_x(y) = S(y)$. The deviation $f(x)$ causes this common edge profile to translate:

$$S_x(y) = S(y + f(x)). \tag{5.2}$$

Now, the question is how to obtain $f(x)$ given the observations $S_x(y)$. First, the underlying common edge profile, $S(y)$ is computed by aggregating information from all available edge profiles. We observe that since $f(x)$ is small, the mean of the edge profiles can be used to compute $S(y)$. Assuming that the image noise is independent at every pixel, the image I is given by

$$I(x, y) = S(y + f(x)) + n(x, y) \tag{5.3}$$

where $n(x, y)$ is the image noise. A first-order Taylor expansion of $S(y + f(x))$ leads to

$$I(x, y) \approx S(y) + f(x)S'(y) + n(x, y). \tag{5.4}$$

Thus, the mean over x is given by

$$\frac{1}{N_x}\sum_x I(x, y) \approx S(y) + \mu_f S'(y) + \frac{1}{N_x}\sum_x n(x, y) \tag{5.5}$$

where the μ_f is the mean of $f(x)$ over x and N_x is the number of pixels in the x

direction. The new noise term is a function only of y and has less variance than the original noise $n(x, y)$. Because $f(x)$ is small, its mean μ_f is also small, hence using the Taylor expansion again yields:

$$S(y) + \mu_f S'(y) \approx S(y + \mu_f). \tag{5.6}$$

Thus, the average edge profile approximates the common edge profile up to a constant shift of μ_f. This shift is insignificant since it only reflects a constant shift in $f(x)$, i.e., an overall translation of the object of interest. Moreover, for many applications, such global translation is filtered out by band-passing the deviation signal. Therefore, for convenience, we treat this translated edge profile as the original edge profile $S(y)$. In practice, to be more robust to outliers in the edge profiles, we use the median instead of the mean.

With $S(y)$ in hand, the deviation signal $f(x)$ is then obtained by estimating the optimal 1D translation, in terms of least square error, between the $S(y)$ and each of the observed ones. In the discrete domain, this takes the form of:

$$\arg\min \sum_y (I(x, y) - S(y) - f(x)S'(y))^2, \tag{5.7}$$

which leads to:

$$f(x) \approx \frac{\sum_y (I(x, y) - S(y))S'(y)}{\sum_y S'(y)^2} \tag{5.8}$$

As can be seen from the equation, pixels for which $S'(y) = 0$ do not contribute at all to the solution. Our formulation is similar to the seminal method [54] used for image registration.

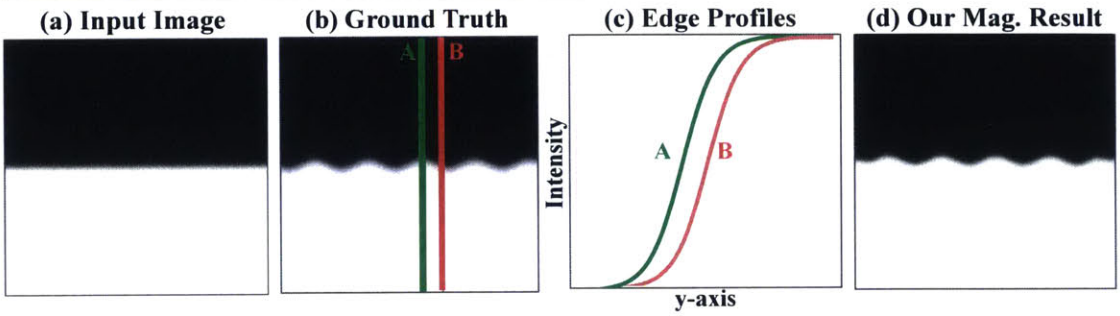


Figure 5.3: Synthetic Example: (a) The input image, a horizontal edge in the middle of the image carries a 0.1 pixel sinusoidal perturbation, $f(x) = 0.1\sin(2\pi\omega x)$. (b) Magnification of the ground truth perturbation by a factor of 20. (c) Two edge profiles obtained by sampling the intensity values in (b) along the green (A) and red (B) vertical lines, respectively. The edge profiles are related by 1D translation. (d), the small perturbation in the input (a) are revealed by our method.

■ 5.3.3 Canonical Stripe Representation

The region in the vicinity of each fitted shape is warped into a canonical stripe. This representation allows us to treat *any* type of fitted shape as a horizontal line. For an arbitrary geometric shape, let $\{\vec{p}_i\}$ be points sampled along it. The shape has a local normal direction at every point, which we denote by $\vec{n}(\vec{p}_i)$. For each point, the image is sampled in the positive and negative normal direction $\pm n(\vec{p}_i)$, using bicubic interpolation to produce the canonical stripe. This sampling is done at a half pixel resolution to prevent spatial aliasing (which may occur for high frequency diagonally oriented textures). To prevent image content far from the shape from affecting the deviation signal, we sample only a few pixels (3-5 pixels) from the shape. In the resulting stripe, the edge is now a horizontal line and the vertical axis is the local normal direction $\vec{n}(p_i)$.

In many cases, the image may be highly textured near the shape's contour, which can invalidate our assumption of a constant edge profile (Eq. 5.2). To address this problem, we apply the matting algorithm of Levin et al. [48] on the sampled image stripe. The output alpha matte has the same sub-pixel edge location as the real image,

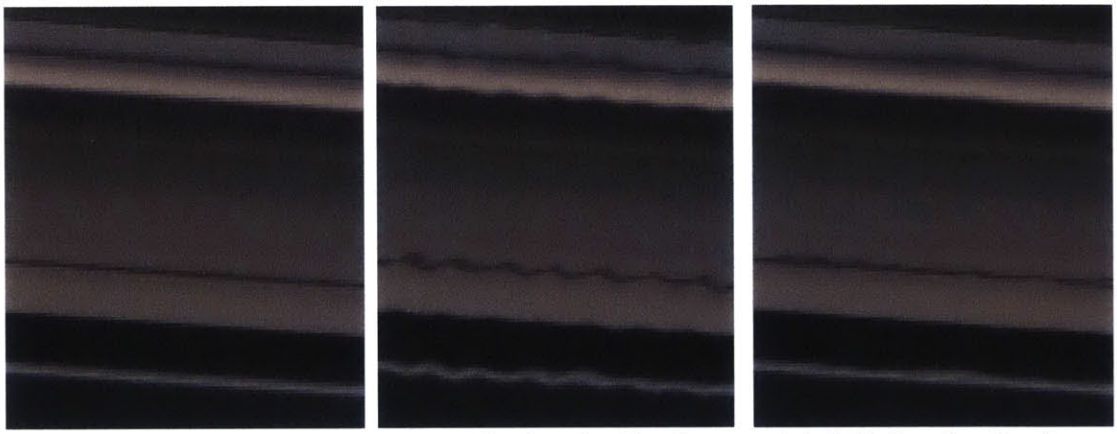
but removes variations due to texture and turns real image stripes into ones that more closely satisfy the constant edge profile assumption. This can be seen in Fig. 5.2(2), where the similarity between the matted stripe and the synthetic image shown in Fig. 5.3 is much stronger than between the synthetic image and the canonical stripe.

The input to the matting algorithm is the canonical stripe and an automatically generated mask, in which pixels on one side of the contour are marked as foreground and pixels on the other side as background. This mask provides the algorithm with a lot of information about where the edge is. This essential step substantially increases the signal to noise ratio and allows us to deal with challenging, real world data as demonstrated in our experimental evaluation (e.g., see Fig. 5.5(d)).

The deviation signal is computed on the estimated alpha matte (as described in Sec. 5.3.2), and therefore represents the amount that the actual shape deviates from the ideal shape in the ideal shape's local normal direction.

Spatial Anti-Aliasing For some images (e.g. astronomy), spatial aliasing in the input image can makes its way into the canonical stripe and then the deviation signal masquerading as a true signal. We address this problem by applying a dedicated spatial anti-aliasing post filter to remove these components. An example is shown in Fig. 5.4, in which we used our method to amplify the deviations from all straight lines in a picture of Saturn's rings. Without the anti-aliasing filter, there is a sinusoidal deviation on all of the processed lines (Fig. 5.4b). Our filter removes this artifactual deviation (Fig. 5.4c).

Theoretically, spatial aliasing can occur at any frequency. However, we can compute it as it depends on the edge's orientation. Once computed, we filter out only that single frequency. The full derivation of our anti-aliasing filter is described in Section 5.6 at the end of this chapter. Note that our anti-aliasing filter may not have a significant impact on all images since reasonable camera prefilters often prevent aliasing. However,



(a) Input Image **(b) W/o Anti-aliasing** **(c) W/ Anti-aliasing**

Figure 5.4: Deviations of Saturn's rings amplified without and with the aliasing post-filter. Without it, there is a sinusoidal perturbation along the rings (b), but our post-filter reveals that it is actually just spatial aliasing in the input image (c). (Image courtesy of NASA.)

we apply it as sanity check to lines in all images.

■ 5.3.4 Synthesis

Now, that we have a filtered deviation signal for every fitted or user-chosen contour in the image, we seek to generate a new image, in which the objects carrying the deviations are warped, but other image content is not. We do this by first computing a 2D warping field, $\vec{V}(x, y) = \{u(x, y), v(x, y)\}$ that is constrained to match the amplified deviation signal at sampled locations along the contours. The flow field at the remaining pixels is determined by minimizing an objective function that aims to propagate the field to nearby pixels of similar color, while setting the field to zero far from the contours.

By construction, the deviation signal is oriented in the normal direction to the contour at each point. At a pixel $\vec{p} := (x, y)$ sampled along the s^{th} contour, we set the warping field to be equal to

$$\vec{V}(\vec{p}) = \alpha f_s(\vec{p})\vec{n}_s(\vec{p}) \tag{5.9}$$

where α is an amplification factor, $f_s(\vec{p})$ is the deviation signal of the sth contour at

location $\vec{p}$ and $n_s(\vec{p})$ is the local normal direction of the sth contour at $\vec{p}$. Every pixel that touches a contour will introduce a hard constraint of this form. If a pixel is on two contours, we average the constraints.

The hard constraints on the warping field imposed by Eq. 5.9 give sparse information that must be propagated to the rest of the image. We follow the colorization method of Levin et al. [47], and define the following objective function for the horizontal component u (the same objective is defined for the vertical component)

$$\arg\min_u \sum_{\vec{p}} \left((u(\vec{p}) - D(\vec{p}) \sum_{\vec{q} \in N(\vec{p})} w_{\vec{p}\vec{q}} u(\vec{q}) \right)^2, \tag{5.10}$$

where $\vec{p}$ and $\vec{q}$ are coordinates in the image, $N(\vec{p})$ is the eight-pixel neighborhood around $\vec{p}$, $w_{\vec{p}\vec{q}} = \exp^{-\|I(\vec{p})-I(\vec{q})\|^2/2\sigma^2}$ is a weighting function measuring the similarity of neighboring pixels and $D(\vec{p})$ is a weighting function that measures the distance from the point $\vec{p}$ to the nearest point on a contour (computed using the distance transform). The inner sum in the objective function is the average warping field of all pixels of similar color to $\vec{p}$ in its neighborhood. The term $D(\vec{p})$ shrinks at pixels far from contours. At pixels far from contours, $D(\vec{p})$ is close to zero and the summand becomes $u(\vec{p})^2$, which encourages the warping field to go to zero. Since the objective function is a least squares problem, it can be minimized by solving a sparse linear system.

Once the warping field is estimated, the rendered image is then given by inverse warping: $\widehat{I}_{\text{dev}} = I(x + u, y + v)$.

■ 5.3.5 User Interaction

While it is possible to perform our processing automatically, we give the user the ability to control which objects or contours are analyzed, what components of the deviation signal should be amplified and what parts of the image should be warped.

A simple GUI is provided for users that want to pick specific objects to amplify. Because it is tedious to specify the exact location of a contour in the image, the user is only required to give a rough scribble of the object. Then, an automatic fitting algorithm is used to find the location of all elementary shapes in the object and we use the one that is closest to what the user scribbled [60]. We show an example of a user selecting a line on top of a bookshelf in the supplementary material.

For a contour specified by points $\{\vec{p}_i\}$, the raw deviation signal $f(\vec{p}_i)$ can contain signals that correspond to several different types of deviations. In addition, the DC component of the signal corresponds to an overall shift of the *entire* object and we may want to adjust or remove it. Noise may also be present in the deviation signal. For these reasons, we apply bandpass filtering to the raw signal $f(x)$. The user can specify the cutoffs of the filter depending on the application. In the sand dune example (Fig. 5.2), we removed the very low and high frequencies to remove noise and the overall curvature of the dune. And in the house example (Fig. 5.1), we only amplified the low frequencies, setting the DC to make the sure the deviation signal was zero at the endpoints, so they did not get warped. The user also specifies an amplification factor indicating the amount by which the deviations should be magnified.

For examples in which the fitted shapes are straight lines, we allow the user to specify a bounding box around the contour of interest (see Fig. 5.5(b)) to ensure that everything within the bounding box gets warped according to the deviation signal. The diagonal of the bounding box is projected onto the fitted line. In the direction parallel to the line, the deviation signal is extrapolated to the ends of the box using quadratic extrapolation of the points close to the end. For all other points in the bounding box, we modify the hard constraints of the above objective function. Specifically, for each point $\vec{p}$ in the bounding box, we find the nearest point on the contour $\vec{q}$ and set the warping field at $\vec{p}$ to be the same hard constraint as $\vec{q}$. This ensures that all objects

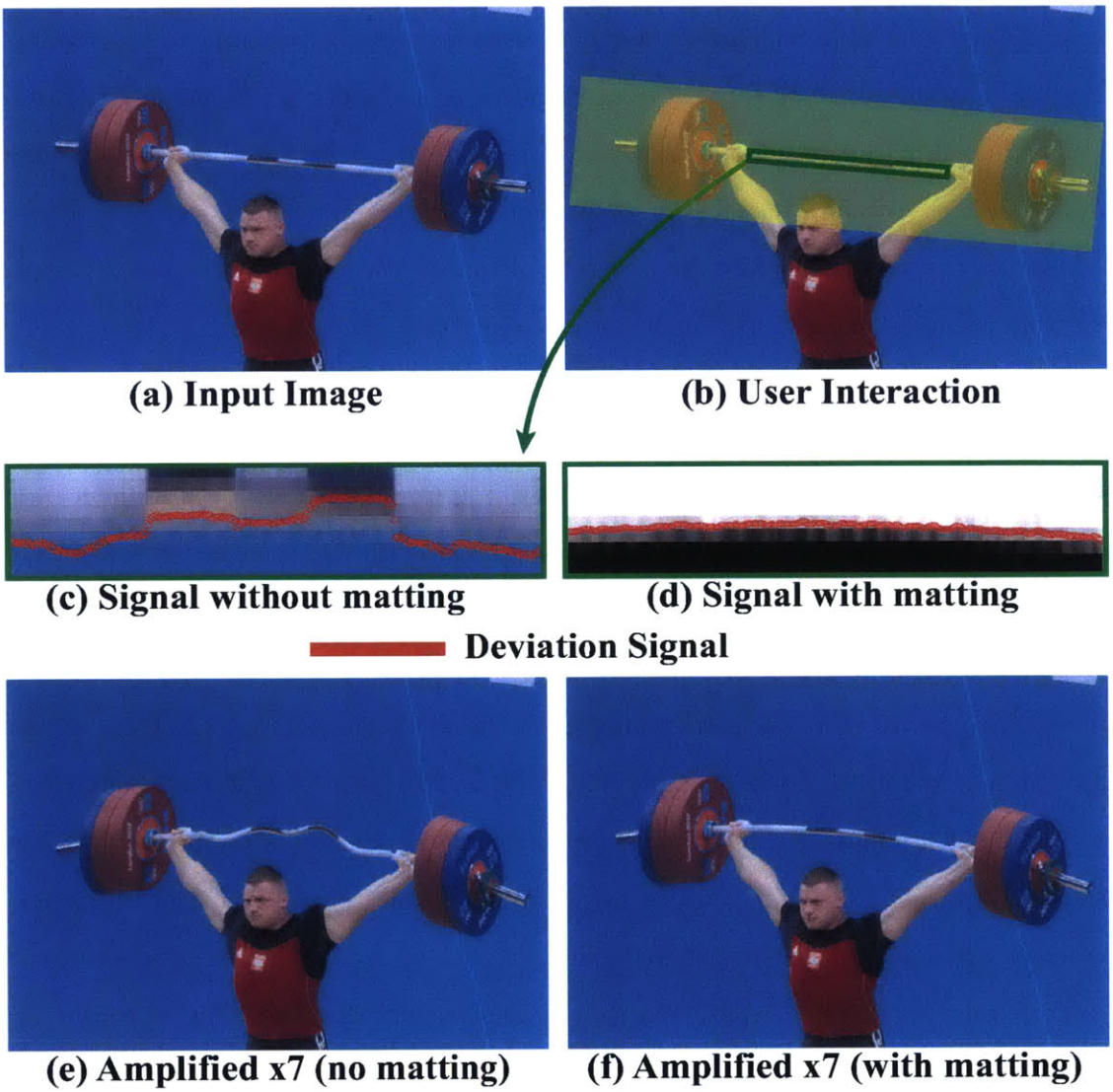


Figure 5.5: Revealing the bending of a weighted steel barbell and a comparison of our method with and without matting. An image stripe is taken from the input image (a) and the deviation signal is overlayed on the image stripe without matting (c) and with it (d). The deviations in the weightlifter's barbell are amplified without matting (e) and with matting (f). (Image courtesy of Jay Smidt.)

within the bounding box get warped in the same way.

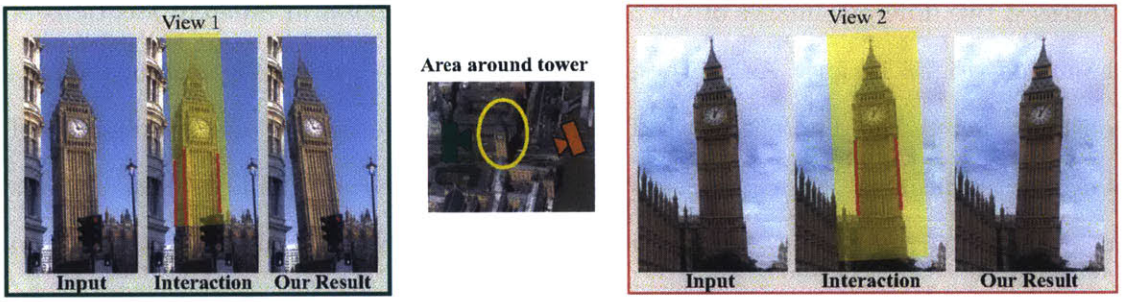


Figure 5.6: Elizabeth Tower (Big Ben) becomes the leaning tower of London. We process the two images of the tower **independently**. *Parallel vertical lines in the input image are used to compute the vanishing point of the input images (only crops of which are shown here). In (b), the user specifies the lines that go through the vanishing point (marked in red) and a region of interest (marked in semi-transparent yellow). Our method computes the deviation from the fitted line, and synthesizes a new image, in which the deviation is exaggerated.*

■ 5.4 Results

The results were generated using non-optimized MATLAB code on a machine with a quad-core processor with 16 GB of RAM. Our processing times depend on the number of shapes processed and the image's resolution. It took twenty seconds to produce our slowest result, in which 180 lines were processed in a 960×540px image.

In some examples, we corrected for lens distortion to prevent it from being interpreted as deviations from straight lines. This is done using the commercial software DxO Optics Pro 10 [23], which automatically infers the lens type from the image's metadata, and then undoes the distortion using this information.

We present our results on real images, and perform a qualitative and quantitative evaluation on both synthetic and real data.

Geometric Deviation Magnification in the World We applied our algorithm on natural images, most of which were taken from the Web. These images are of real world objects, which have highly textured edges making them challenging to process.

In Fig. 5.1, we reveal the sagging of a house's roof by amplifying the deviations from a straight line fitted to the upper part of the roof. To validate our results, we

processed two different images of the house in which the roof is at different locations of the image (Fig. 5.1(a,c)). As can be seen in Fig. 5.1(b,d), the roof's sagging remains consistent across the different views. Revealing this subtle sagging of a building's roof is a useful indication of when it needs to be repaired. Because the house's roof spans such a large part of the image, the effect of lens distortion may not be negligible. To avoid this problem, we used DxO Optics Pro 10 to correct for lens distortion.

In Fig. 5.2, we reveal the periodic ripple pattern along the side of a sand dune by amplifying its deviations from a straight line by ten times. Here, even the intensity variations along the line show the deviations. The raw signal was bandpassed in order to visualize only the ripple marks and not the overall curvature of the dune. Knowing what these imperceptible ripple marks look like may have applications in geology [58].

In Fig. 5.5, we reveal the bending in a steel barbell due to the weights placed on either end by amplifying the low frequencies of the deviation from a straight line. In addition to specifying the line segment to be analyzed, the user also specifies a region of interest, marked in green in Fig. 5.5(b), specifying the part of the image to be warped. In this example, the necessity of matting in our method is demonstrated. Without matting, the color difference between the darker and lighter parts of the barbell causes a shift in the raw deviation signal (Fig. 5.5(c)), which causes the barbell to appear wavy after amplification (Fig. 5.5(e)). With matting, we are able to recover the overall curvature of the barbell and visualize it (Fig. 5.5(d,f)).

Civil engineers have reported that Elizabeth Tower (Big Ben) is leaning at an angle of 0.3 degrees from vertical [36]. Our algorithm reveals this visually in two independently processed images of the tower from different viewpoints (Fig. 5.6). Here too, the consistency across views supports our results. In this example, instead of directly using the edges of the tower as the fitted geometry, we use vertical lines going through the vanishing point. We find them by using a technique from Hartley and Zisserman

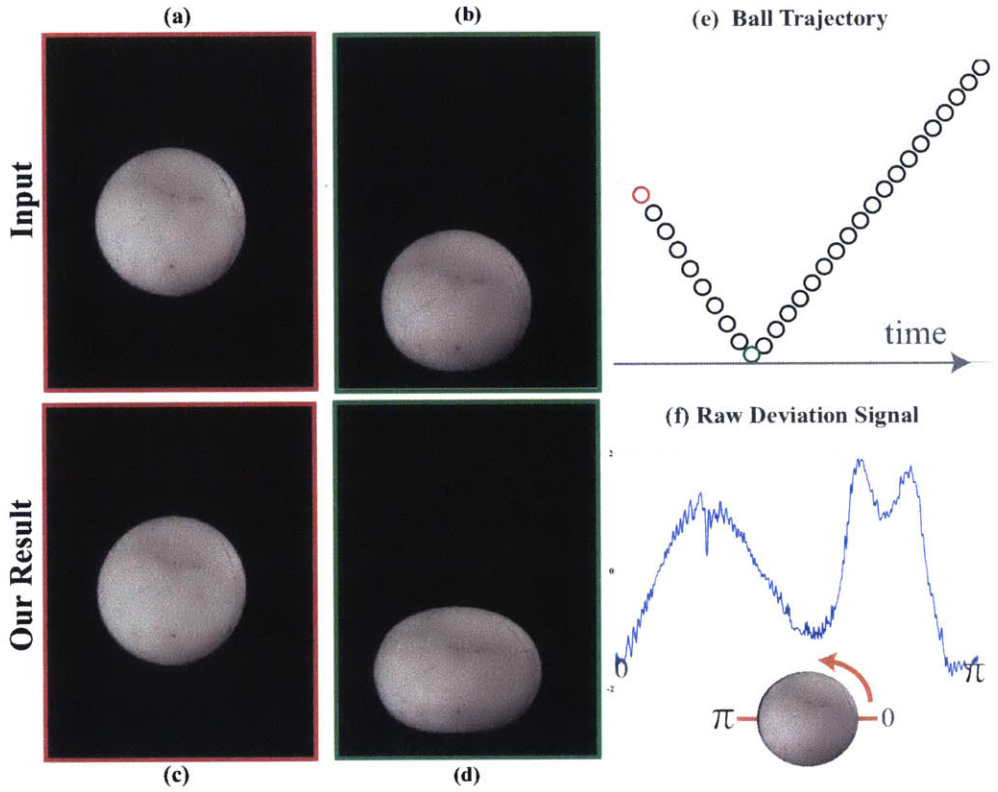


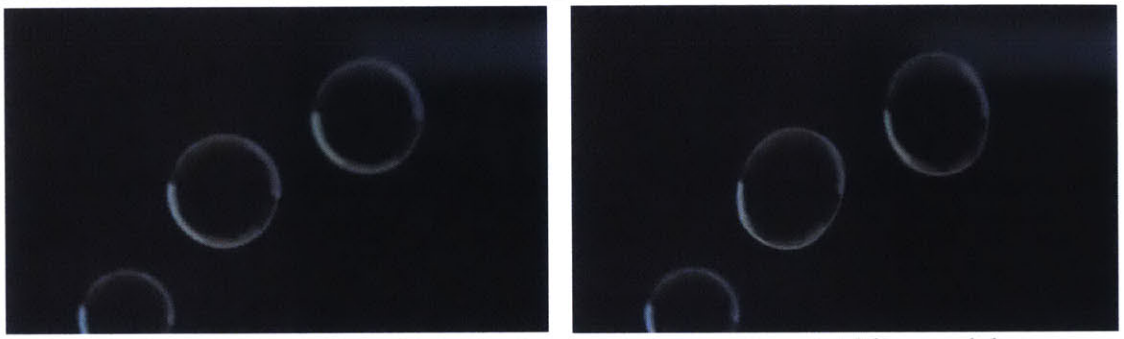
Figure 5.7: Revealing the distortion and vibrations of a ball when it hits a table. (a-b) two frames from the input video that correspond to the red and green locations of the ball in (e), respectively. Our method computes the deviation of the ball from a perfect circle in each of the frames independently. (c-d), the rendering of (a) and (b), respectively, where the deviation is x10 larger. (e), the raw deviation signal, counterclockwise along the ball from 0 to π.

[38]. In order to amplify only the tilting of the tower (which corresponds to low frequencies in the deviation signal), while ignoring deviations due to bricks on the tower (high frequencies), we lowpass the deviation signal. The filtered signal is then extrapolated to the entire user-specified bounding box to warp the entire tower. The size of the deviations for lines on the tower are on average 2-3x larger than the deviations of lines on other buildings indicating that we are actually detecting the tilt of the tower. Visualizing the subtle tilt of buildings may give civil engineers a new tool for structural monitoring. Lens distortion is corrected as a preprocessing step (see the supplementary video for our results w/o lens distortion).

Geometric Deviation Magnification in Videos In the following examples, we tested our method on several video sequences. Specifically, our method was applied to each of the frames **independently**, without using any temporal information. The fitted shapes in each frame were detected automatically. The temporal coherence of the amplified structures, processed independently for each frame, validates our results. For some of the examples, we were also able to compare our results to motion magnification applied to stabilized versions of the sequences [81]. Our results and comparisons on the entire sequences are included in the supplementary material.

The video *ball* is a high speed video (13,000 FPS) of a lacrosse ball hitting a black table in front of a black background. The trajectory of the ball is illustrated in Fig. 5.7, and two of the frames that correspond to the red and green locations of the ball are shown in Fig. 5.7(a,b), respectively. We applied our framework to reveal the distortion in the shape of the ball, i.e., deviation from a perfect circle, when it hits the ground and travels upward from impact.

Fig. 5.7(c,d) shows our rendering of the two input frames where the deviation is ten times larger. Fig. 5.7(f) shows the raw deviation signal for the moment of impact (green location) as a function of the angle. Because the raw deviation signal appears to have most of its signal in a low frequency sinusoid, we apply filtering to isolate it to remove noise. Our results on the entire sequence not only reveals the deformation of the ball at the moment it hits the ground, but also reveals the ball's post-impact vibrations. For comparison, we apply motion magnification with and without stabilizing the input video. Without stabilization, motion magnification fails because the ball's displacement from frame to frame is too large. With stabilization, the results are more reasonable, but the moment of impact is not as pronounced. This is because the motion signal has a temporal discontinuity when the ball hits the surface that is not handled well by motion magnification. In contrast, deviation magnification handles this discontinuity,



(a) Input **(b) Magnified x20**

Figure 5.8: Revealing the vibrations of a bubble from a single frame. An input frame of two bubbles (a) was used to produce our magnification result (b) in which the low frequency deviations of each bubble were amplified. The shapes were automatically detected. No temporal information was used.

as each frame is processed independently.

Bubbles is a high speed video (2,000 FPS) of soap bubbles moving to the right shortly after their generation (Fig. 5.8(a-b)). Surface tension causes the bubbles to take a spherical shape. However, vibrations of the bubble and gravity can cause the bubble's shape to subtly change. In this sequence, we automatically detect the best fit circles for the two largest bubbles and amplify the deviations corresponding to low frequencies independently in each frame. This allows us see both the changing dynamics of the bubble and a consistent change in the bubble's appearance that may be due to gravity.

For comparison, we applied motion magnification to a stabilized version of the sequence. We used the fitted circles to align the bubbles in time and then applied motion magnification (similar to [24]). The magnified bubbles were then embedded back in the input video at their original positions using linear blending at the edges. This careful processing can also reveals the changing shape of the bubbles, but it does not show the deviations from circular that do not change in time, such as the effect of gravity on the bubble.

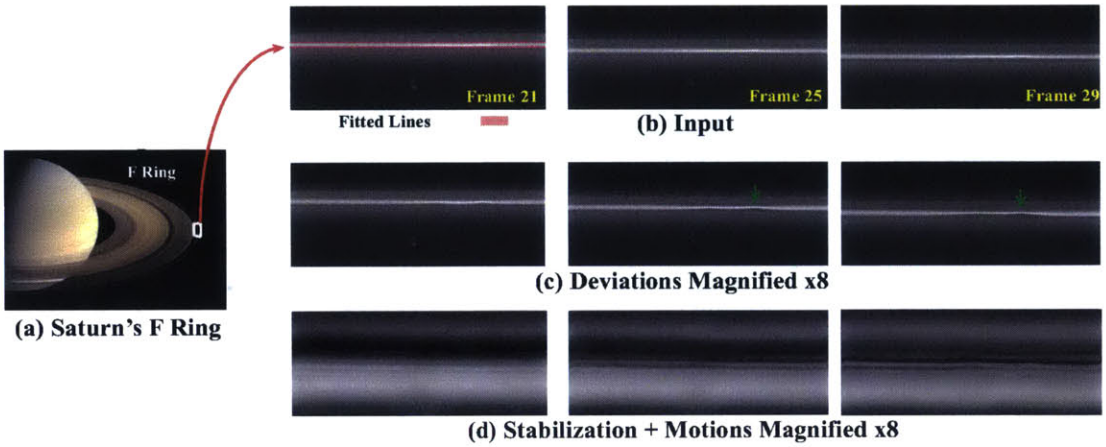


Figure 5.9: Geometric Deviation Magnification independently applied to each frame of a timelapse of Saturn's moon interacting with its ring. Three frames from the timelapse (b) are processed using our new deviation magnification method (c) and using stabilization plus motion magnification (d). The lines used to do the stabilization and the fitting are shown in (b). The green arrows denotes the most salient feature after amplification. The full sequence is in the supplementary material. (Images courtesy of NASA.)

Fig. 5.9(b) presents three frames from a 72 frame timelapse (captured by the Cassini orbiter), showing Saturn's moon *Prometheus* interacting with Saturn's F ring. The frames were aligned by NASA such that the vertical axis is the distance from Saturn, which causes the rings to appear as horizontal lines. For every frame in the video, we amplified the deviations from the best-fit straight lines (marked in red in Fig. 5.9(b)). This reveals a nearly invisible, temporally consistent ripple (Fig. 5.9(c)). These kind of ripples are known to occur when moons of Saturn approach its rings [74]. Applying our technique on such images may be useful for astronomers studying these complex interactions, and even might reveal new undiscovered gravitational influences in the rings.

We also applied motion magnification to a stabilized version of the sequence. As can be seen in Fig. 5.9(d), even with stabilization, magnifying changes over time produces a lot of unwanted artifacts due to temporal changes in the scene unrelated to the main ring. It is the *spatial deviations* from the model shape that are primarily interesting in

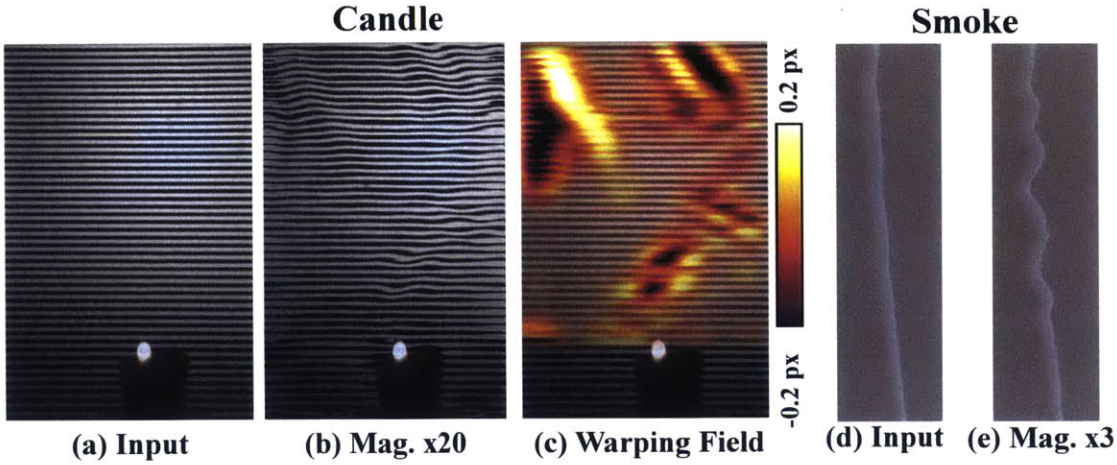


Figure 5.10: Revealing the distortions in a background of straight lines caused by heated air and sinusoidal instabilities of smoke flow in a single image. In candle, the deviation from every straight line is amplified twenty times. Both the amplified result and the overlayed vertical warping field are shown. In smoke, a single line is fitted to the input and the result is magnified three times.

this example rather than the changes in time.

In *candle*, we use our method to reveal heated air generated by a candle flame from a *single* image (Fig. 5.10(b-c)). To do so, we estimate the deviations from every straight line, automatically fitted to the background. As can be seen in Fig. 5.10(b-c), the twenty times amplified image, and the warping field reveal the flow of the hot air. Visualizing such flow has applications in many fields, such as aeronautical engineering and ballistics. Other methods of recovering such flow such as background-oriented Schlieren [37] and refractive wiggles [87], analyze changes over time. While these methods are restricted to a static camera, our method is applied to every frame of the video independently and is able to reveal the heated air even when the camera freely moves. In addition, the bumps in the background are revealed as well. Note that spatially stabilizing such a sequence is prone to errors because the background is one-dimensional, the camera's motions are complex and the candle and background are at different depths.

A similar result is shown in Fig. 5.10(d-e) where a column of rising *smoke* appears

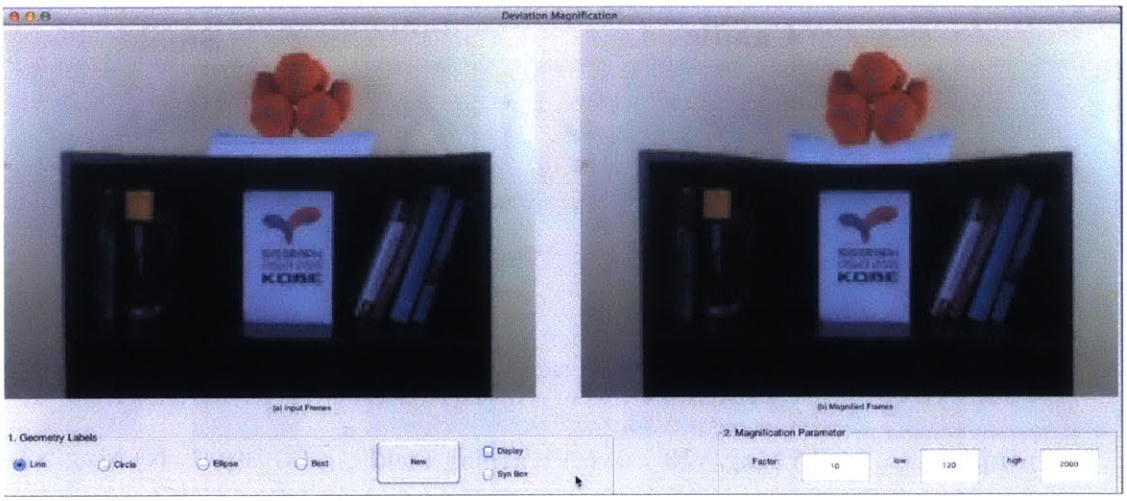


Figure 5.11: A frame from our interactive demo showing a bookshelf buckling under weight when deviations from a straight line are amplified.

to be a straight line. By amplifying the deviations from straight, we reveal sinusoidal instabilities that occur in the smoke's flow as it transitions from laminar to turbulent [75]. Here too, the processing is done on each frame independently.

Interactive Demo We have produced an interactive demo that can process a 200x150 pixel video at 5 frames per second (Fig. 5.11). The user roughly specifies the location of the line at the first frame, which is then automatically snapped to a contour. Our demo is interactive because we only process a single shape. A video of this demo to show the buckling of a bookshelf under weight when the deviations from a straight line are amplified is provided in the supplementary materials.

■ 5.4.1 Synthetic Evaluation

To evaluate the accuracy of our method in estimating the deviation signal, we tested it on a set of 7500 synthetic images. The images had known subtle geometric deviations, so that we could compare our result with the ground truth. The images were 200×200 pixels with a single edge (see Fig. 5.12(a)). We varied the exact deviation, the orientation, the sharpness of the edge, the noise level and the texture on either

side of the edge (Fig. 5.12(a-b)). Specifically, ten different cubic spline functions with a maximum magnitude of 1 pixel were used as the deviation shapes. Ten orientations were sampled uniformly from 0°to 45°with an increment of 5°. The edge profile was set to be a sigmoid function sigmf(δ,x) $= 1/(1 + \exp(-\delta x))$ with $\delta = \{0.5, 2, 5\}$).

We first performed an evaluation on images without texture. We also tested the effect of sensor noise by adding white Gaussian noise with standard deviation $\sigma = \{0.02, 0.05, 0.1\}$. In Fig. 5.12(c), we show the error of our algorithm, which grows only linearly with the noise-level even when it is 25 intensity levels ($\sigma = 0.1$).

In Fig. 5.12(d), we present the mean absolute error between the estimated deviation signal and the ground truth as a function of the line orientation, for the three edge sharpness levels. The average error is very small at 0.03px, 3% of the maximum magnitude of the ground truth deviation signal (1 px). As expected, smoother edge profiles lead to smaller error due to less aliasing.

To quantify the effect of texture and the ability of matting to remove it, we tested our method on textured synthetic images. We used six different textures to perform experiments in which only one side of the edge was textured (Fig. 5.12(e)). We also performed experiments in which both sides were textured using all 15 combinations of the six textures (Fig. 5.12(f)). Fig. 5.12(e-f) show the mean absolute error with and without matting for one-sided, half-textured images and two-sided, fully-textured images respectively. Without matting, the average error of our algorithm is about 0.3px for the half-textured examples and 1.5px for the fully-textured examples. With matting, the average errors shrink by ten times and are only 0.03px and 0.1px respectively. The highest errors are on a synthetic image, in which both sides of the image are of similar color. See the supplementary material for more details.

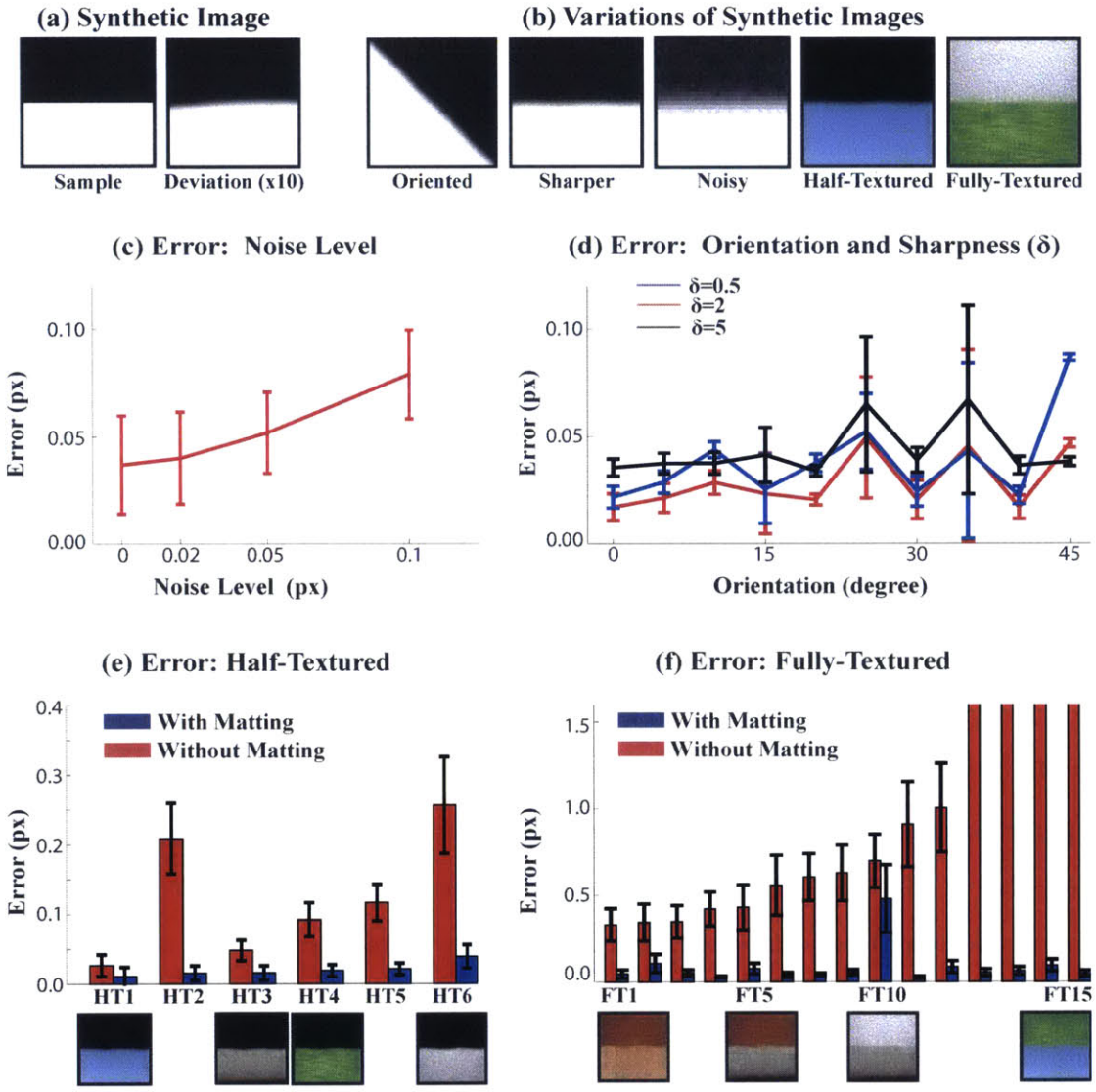


Figure 5.12: Quantitative evaluation on synthetic data: (a), an example untextured image and its amplified version. (b), the data includes images with lines in different orientations, sharpness levels, noise levels and textures. (c), the error as a function of additive Gaussian noise. (d), the mean absolute error in the deviation signal computed by our method, as a function of the orientation, for different sharpness levels. (e), the error w/ and w/o matting for half-textured images (shown below). (f), the error w/ and w/o matting for fully-textured images.

■ 5.4.2 Controlled Experiments

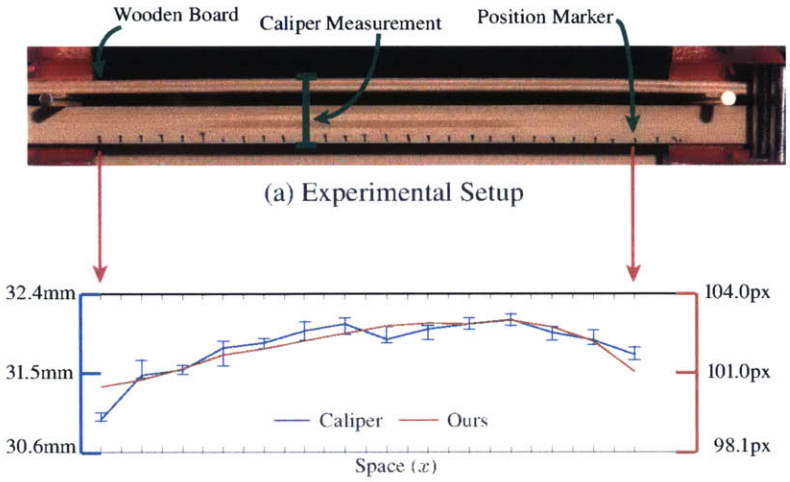
We validated the accuracy of our method on real data by conducting two controlled experiments. In the first experiment, we physically measured the deviations from a straight line of a flexible wooden board. The board was affixed on top of two rods on a table using C-clamps (Fig. 5.13). The base of the table served as the reference straight line. The distance from the bottom of the table to the top of the board was measured across a 29 cm stretch of it, in 2 cm increments using digital calipers (the markers in Fig. 5.13(a)). The deviation signal from a straight line of the image of the wooden board is very similar to the caliper measurements (Fig. 5.13(b)).

In the second experiment, we affixed a stick onto a table and covered it with a sheet, with a pattern of ellipses on it (see Fig. 5.14(a)). The stick caused the sheet to slightly deform, which subtly changed the shape of some of the ellipses. To reveal the deformation, all the ellipses in the input image were automatically detected, and our method was applied to magnify the deviations of each ellipse from its fitted shape. A bandpass filter was applied to the deviation signal to remove overall translation due to slight errors in fitting and to smooth out noise. As can be seen in Fig. 5.14(d)), only ellipses on or near the stick deform significantly, which reveals the stick's unobserved location.

■ 5.5 Discussion and Limitations

We have shown results on lines, circles and ellipses. However, except for the geometry fitting stage, our algorithm can generalize to arbitrary shapes. If a user can specify the location of a contour in an image, our algorithm can be applied to it. For higher-order shapes such as splines, it can be unclear what should be a deviation and what should be part of the fitted model.

While we are able to reveal a wide variety of phenomena with our method, there are



(a) Experimental Setup

(b) Deviation from Straight line

Figure 5.13: Deviation from straight lines, controlled experiment. (a) The experimental setup, which is also the image used as an input to our framework. The measurements from digital calipers between the two wooden boards at each position marker and the deviation signal from our method are shown (b).

circumstances in which our algorithm may not perform well. If the colors on both sides of the shape's contour are similar, we may not be able to compute its sub-pixel location. This is an inherent limitation in matting and edge localization. In some cases, changes in appearance along the contour may look like geometric deviations (e.g. a shadow on the object that is the color of the background). In this case, the deviation signal may have a few outliers in it, but otherwise be reliable.

Our method may also not be able to distinguish artifacts caused by a camera's rolling shutter from a true geometric deviation in the world. If the camera or object of interest is moving, the camera's rolling shutter could cause an artifactual deviation present in the image, but not in the world. Our method would pick this up and "reveal" it. Bad imaging conditions such as low-light or fast-moving objects could cause a noisy image with motion blur, which would be difficult for our system to handle.

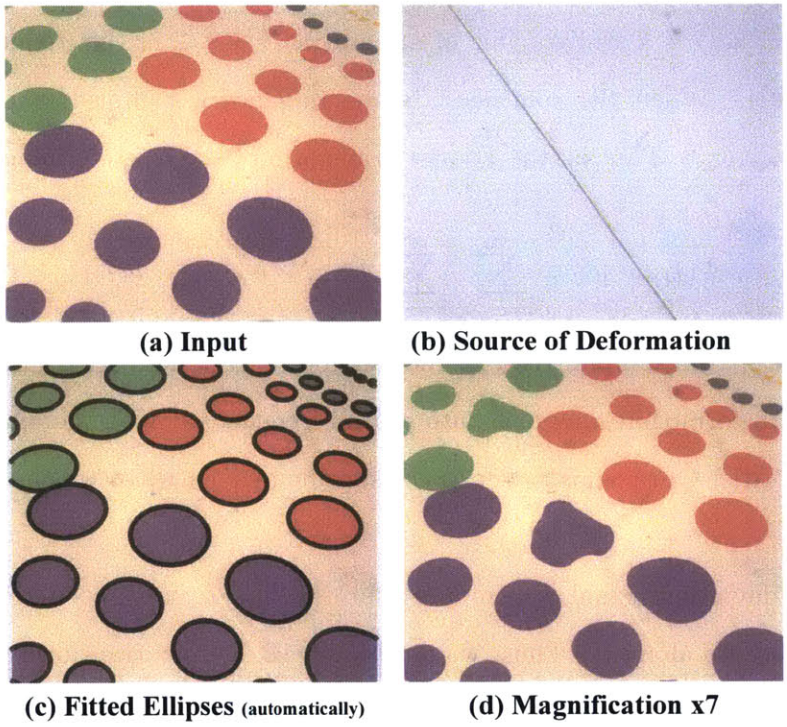


Figure 5.14: Deviation from ellipses, controlled experiment. A sheet with ellipses on it is draped over a table with a stick on it (a-b). The deviation from every ellipse is automatically fitted (c) and then amplified by seven times (d) revealing the unobserved location of the stick.

5.6 Anti-aliasing filter

We describe in more detail our anti-aliasing post-filtering step. To determine which frequencies correspond to aliasing, we perform the following frequency analysis on a continuous image of a straight line.

Let $I(x, y)$ be a continuous image of a step edge of orientation θ (Figure 5.15(a)). If the edge profiles along the formed line L are constant, the 2D Fourier transform of the image $F(\omega_x, \omega_y)$ is a straight line of orientation $\theta + \pi/2$ in the frequency domain (Figure 5.15(b)). When the continuous scene radiance is sampled, a periodicity is induced in $F(\omega_x, \omega_y)$. That is, the Fourier transform of the discrete image $I_D(x, y)$ is equal to

$$\mathcal{F}(I_D(x, y)) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} F(\omega_x - nf_s, \omega_y - mf_s) \tag{5.11}$$

where f_s is the spatial sampling rate of the camera. This periodicity creates replicas in the Fourier transform that may alias into spatial frequencies along the direction of the edge (Fig. 5.15(d)). Our goal is to derive the specific frequencies at which these replicas occur.

Since the deviation signal is computed for the line L, we are only interested in aliasing that occurs along it. Thus, we derive the 1D Fourier transform of the intensities on the discrete line L_D via the sampled image's Fourier transform $\mathcal{F}(I_D(x, y))$. Since $F(\omega_x, \omega_y)$ is non-zero only along the line perpendicular to L, the discrete Fourier transform $\mathcal{F}(I_D(x, y))$ contains replicas of this line centered at $n(f_s, 0) + m(0, f_s)$ for integer n and m (from Eq. 5.11). Using the slice-projection theorem, the 1D Fourier transform of L_D is given by the projection of $\mathcal{F}(I_D(x, y))$, i,e, the image's 2D Fourier transform, onto a line with orientation θ that passes through the origin. This means that the replica's project all of their energy onto a single point on L_D at location

$$nf_s \cos(\theta) + mf_s \sin(\theta), \tag{5.12}$$

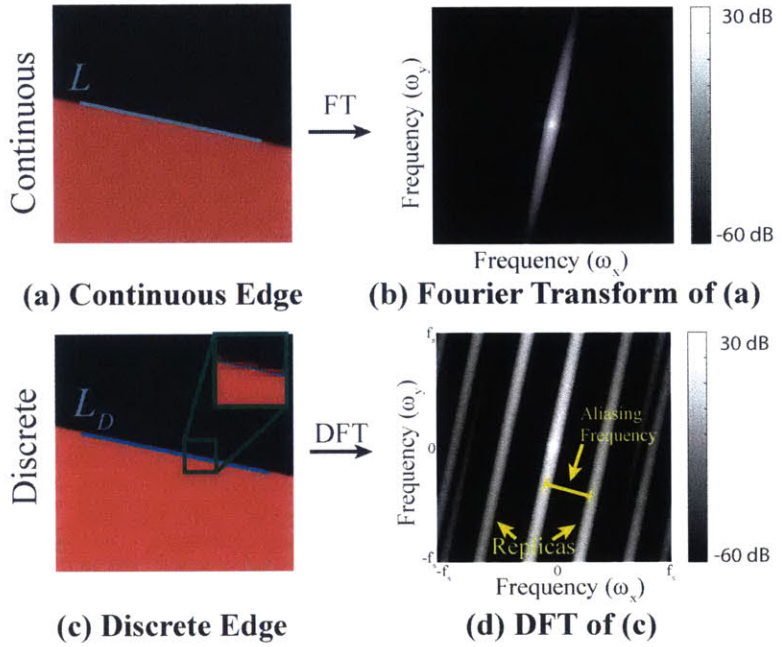


Figure 5.15: Causes of spatial aliasing and how to find the aliasing frequency. (a), a continuous edge and its discretization (c); (b,d), the Fourier transforms of (a,c) respectively. (d), replicas in the Fourier transform cause spatial aliasing along the line L.

which gives us the value of the aliasing frequencies along the image slices. The first and usually most dominant such frequency occurs when exactly one of n or m is equal to one and has value

$$f_s \min(|\cos(\theta)|, |\sin(\theta)|). \tag{5.13}$$

The exact strength and importance of each aliasing frequency depends on the edge profile. Since most real images are taken with cameras with optical anti-aliasing pre-filters, they have softer edges. We found it sufficient to only remove the lowest aliasing frequency (Eq. 5.13) to mitigate the effects of aliasing. To handle small deviations in orientation, we remove a range of frequencies near the aliasing frequency (Eq. 5.13).

Chapter 6

Conclusion

We have proposed novel techniques for analyzing and visualizing imperceptible deviation signals in images and videos. These methods work by first finding a signal that corresponds to the imperceptible deviations and then using that signal to create a new image or video, in which the deviations are larger. This processing is a new type of microscope that makes subtle deviations from a model larger, making them visible to the naked eye. We focused on two models in this dissertation: perfect stillness in videos and perfect geometries in single images.

Deviations from perfect stillness are tiny motions. To reveal these tiny motions, we leveraged the complex steerable pyramid, a localized version of the Fourier transform, with a notion of local phase and local amplitude. Each frame of the video is converted to this representation and then the local phase variations are amplified. We have demonstrated that this phase-based technique improves the state-of-the-art in Eulerian motion processing both in theory and in practice, provides a fundamentally better way of handling noise, and produces high quality photo-realistic videos with amplified or attenuated motions for a variety of applications. We also proposed a new representation for phase-based video magnification, the Riesz Pyramid. It is capable of producing motion magnified videos in real-time on a modern laptop.

In addition to proposing new methods of motion magnification, we augmented it with two new capabilities: a quantitative estimate of how much pixels in the video are

moving and an estimate of the variance of this quantitative estimate. This makes motion magnification a suitable tool for scientists and engineers that may be interested in both visualizing tiny motions and doing experiments with quantitative data related to tiny motions in videos. We also quantified the variance of this motion estimate giving a way to determine when we are amplifying true signal and when we are amplifying shaped noise. We also showed the utility of motion magnification and these new capabilities on two examples from biology and mechanical engineering.

We have also presented a method of exaggerating geometric deviations from ideal shapes in images. This algorithm involves three steps: model-fitting, deviation analysis, and visual exaggeration. Since the deviations from the geometric model may be very small, care is taken to account for pixel sampling and image texture, each of which can otherwise overwhelm the small signal we seek to reveal. This method successfully reveals sagging, bending, stretching and flowing that would otherwise be hidden or barely visible in the input images. We validated the technique using both synthetically generated and ground-truth physical measurement and we believe that this method can be useful in many domains such as construction engineering and astronomy.

Impact One technique described in this dissertation, motion magnification, has received considerable attention in media outlets, such as The New York Times[1], Reuters[2] and The Wall Street Journal[3]. Motion magnification is already making a big impact. Several academic works have also built on the idea of analyzing tiny motions in videos using local phase variations. They have been used to characterize material properties in videos [16], make plausible simulations of perturbed objects [17], remotely analyze the health of structures [12, 13] and reveal the pulsing of arteries and veins in surgical videos [1, 55].

[1] http://bits.blogs.nytimes.com/2013/02/27/scientists-uncover-invisible-motion-in-video/

[2] http://www.reuters.com/video/2015/01/28/amplifying-tiny-movements-to-visualize-t?videoId=363022528

[3] http://www.wsj.com/articles/monitoring-tiny-vibrations-to-avert-big-problems-1431704895

Magnifying imperceptible deviations in images and videos has many applications in many different domains. It has many potential future applications such as monitoring and visualizing the vibrations of buildings and bridges, monitoring vital signs, finding perturbations in planetary rings and assisting scientists and engineers in the lab looking at cells and material structures with tiny motions or hidden geometric deviations.

Appendix A

Filter Taps and Design for Riesz Pyramids and Pseudocode

In this appendix, we provide the filter taps for the approximate Riesz transform and the replacement for the Laplacian pyramid used in the Riesz pyramid in Chapter 3. We also describe our method of computing these taps and the effect of the approximation. We also provide pseudocode for fast phase-based video magnification with the Riesz Pyramid.

■ A.1 Replacement for Laplacian Pyramid

In this section, we describe our method of designing a new pyramid like the Laplacian pyramid, but with a better inverse. Our method is inspired by Simoncelli and Freeman and we review constraints and motivation from their paper [71]. We use their techniques to design a 1D version of our new pyramid. We then show how it can be converted to a 2D pyramid using the McClellan transform [50], that is more efficient to compute than Simoncelli and Freeman's original replacement with non-separable filters.

The Laplacian pyramid decomposes an image into subbands corresponding to different scales [9]. It does this by decomposing an image into the sum of a high frequency component and a low frequency component. The low frequency component is then downsampled and the decomposition is recursively applied to the downsampled image.

The levels of the pyramid form an overcomplete representation of the image, in which each level corresponds to a different set of spatial frequencies. The pyramid can be inverted by upsampling the lowest level, adding it to the second lowest level to form a new lowest level on which the inversion process can then be recursively applied.

While the inversion is exact when the Laplacian pyramid representation of the image is unmodified, it is less than ideal when the pyramid is modified, such as for the purposes of image compression or phase-based video magnification [19]. If we view the modifications as noise and the Laplacian pyramid as a linear transform T, then the mean squared error optimal inverse is the pseudoinverse $(T^T T)^{-1} T^T$. When the downsampling and upsampling filters are separable Gaussian blurs, the inverse we described in the previous paragraph is not the pseudoinverse and is therefore suboptimal. In addition, the pseudoinverse is difficult to compute directly due to the matrix multiplications and inversions. As a result, we seek to design a new pyramid in which $T^T T = I$, so that the pseudoinverse is simply the transpose of the transform. We do this by adopting the construction scheme proposed by [71], in which T is chosen such that $T^T T = I$ and both T and T^T can be evaluated using a series of recursively applied convolutions and subsampling operations.

Specifically, the pyramid we construct is specified by a highpass filter $h_H[n]$ and a lowpass filter $h_L[n]$. The image is highpassed to form the top level of the pyramid. Then, it is lowpassed and downsampled. The decomposition is recursively applied to the downsampled image to build the pyramid (Fig. A.1). The transpose of this operation when viewed as a matrix multiplication is to upsample the downsampled image, lowpass it again and then add it to a highpassed version of the next level up. To ensure that the inverse reconstructs the input image perfectly, we require that the frequency responses

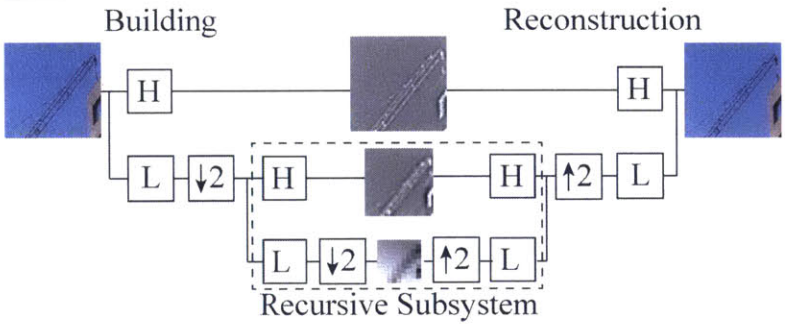


Figure A.1: A signal processing diagram of our pyramid construction showing how a lowpass and highpass filter can be recursively used with subsampling to produce a sequence of critically sampled bandpassed images. The blocks $\downarrow 2$ and $\uparrow 2$ denote downsampling and upsampling by a factor of 2 in both x and y. L and H denote linear shift invariant lowpass and highpass filters respectively.

of the lowpass and highpass filters, $H_L(\omega)$ and $H_H(\omega)$ satisfy

$$|H_L(\omega)|^2 + |H_H(\omega)|^2 = 1 \tag{A.1}$$

In addition, we do not want the downsampled images to be aliased, which imposes the additional requirement that

$$|H_L(\omega)| = 0 \text{ for } |\omega| > \pi/2. \tag{A.2}$$

Our construction is different than Simoncelli and Freeman [71] because we first design a 1D pyramid and then convert it to 2D using the McClellan transform. This will also allow us to evaluate the filters in an efficient manner, described below.

We follow [71] and [43] and find filters that satisfy these constraints by setting up an optimization problem in which the deviation from Eq. A.1 and Eq. A.2 is penalized by the L^1 norm. That is the mean of the deviation is penalized. We also include constraints to ensure that the lowpass filter has energy near the DC component and that the highpass filter has energy near the Nyquist frequency. We minimize this objective

function using Matlab's `fminunc` to give the filters h_L and h_H shown in Table 1(a).

After designing the 1D filters, we convert them to 2D filters using the McClellan transformation [50], which converts a 1D symmetric FIR filter into a 2D FIR filter, which is approximately radially symmetric. We briefly review this transformation now. The frequency response of a one dimensional filter $h_L[k]$ with $2N+1$ taps can be written as a trigonometric polynomial:

$$H_L(\omega) = \sum_{k=-N}^{N} h_L[k]\cos(k\omega) = \sum_{n=0}^{N} b_L[n](\cos(\omega))^n \tag{A.3}$$

where $b_L[n]$ is determined by $h_L[k]$ via Chebyshev polynomials [50].

In the McClellan transformation, the $\cos(\omega)$ is replaced by a 3×3 two dimensional filter $t[x, y]$ with frequency response $T(\omega_x, \omega_y)$. The result is a 2D filter

$$H_L(\omega_x, \omega_y) = \sum_{k=0}^{N} b_L[k](T(\omega_x, \omega_y))^n \tag{A.4}$$

that has contours lines equal to those of $T(\omega_x, \omega_y)$. A good choice for t is the 3×3 filter specified in Table A.1. In this case, $T(\omega_x, \omega_y)$ is approximately circularly symmetric.

Eq. A.4 suggests an efficient way to jointly lowpass and highpass an image. Specifically, the input image $i[x, y]$ is repeatedly convolved with t, N times to yield the quantities:

$$i, t * i, \ldots, \underbrace{t * \ldots * t}_{N \text{ times}} * i \tag{A.5}$$

or in the frequency domain

$$I(\omega_x, \omega_y), T(\omega_x, \omega_y)I(\omega_x, \omega_y), \ldots, T(\omega_x, \omega_y)^N I(\omega_x, \omega_y) \tag{A.6}$$

From this and Eq. 4, it becomes clear that we can take a linear combination of Eq. 5 to

Lowpass:	-0.0209	-0.0219	0.0900	0.2723	0.3611	0.2723	0.0900	-0.0219	-0.0209
Highpass:	0.0099	0.0492	0.1230	0.2020	-0.7633	0.2020	0.1230	0.0492	0.0099

(a) One dimensional filter taps ($h_L[k]$ and $h_H[k]$

0.125	0.250	0.125
0.250	-0.500	0.250
0.125	0.250	0.125

(b) $t[x, y]$ (McClellan Transform)

Lowpass:	0.1393	0.6760	0.6944	-0.1752	-0.3344
Highpass:	-0.9895	0.1088	0.3336	0.3936	0.1584

(c) $b_L[k]$ and $b_H[k]$

-0.0001	-0.0007	-0.0023	-0.0046	-0.0057	-0.0046	-0.0023	-0.0007	-0.0001
-0.0007	-0.0030	-0.0047	-0.0025	-0.0003	-0.0025	-0.0047	-0.0030	-0.0007
-0.0023	-0.0047	0.0054	0.0272	0.0387	0.0272	0.0054	-0.0047	-0.0023
-0.0046	-0.0025	0.0272	0.0706	0.0910	0.0706	0.0272	-0.0025	-0.0046
-0.0057	-0.0003	0.0387	0.0910	0.1138	0.0910	0.0387	-0.0003	-0.0057
-0.0046	-0.0025	0.0272	0.0706	0.0910	0.0706	0.0272	-0.0025	-0.0046
-0.0023	-0.0047	0.0054	0.0272	0.0387	0.0272	0.0054	-0.0047	-0.0023
-0.0007	-0.0030	-0.0047	-0.0025	-0.0003	-0.0025	-0.0047	-0.0030	-0.0007
-0.0001	-0.0007	-0.0023	-0.0046	-0.0057	-0.0046	-0.0023	-0.0007	-0.0001

(d) Taps for direct form of lowpass filter

0.0000	0.0003	0.0011	0.0022	0.0027	0.0022	0.0011	0.0003	0.0000
0.0003	0.0020	0.0059	0.0103	0.0123	0.0103	0.0059	0.0020	0.0003
0.0011	0.0059	0.0151	0.0249	0.0292	0.0249	0.0151	0.0059	0.0011
0.0022	0.0103	0.0249	0.0402	0.0469	0.0402	0.0249	0.0103	0.0022
0.0027	0.0123	0.0292	0.0469	-0.9455	0.0469	0.0292	0.0123	0.0027
0.0022	0.0103	0.0249	0.0402	0.0469	0.0402	0.0249	0.0103	0.0022
0.0011	0.0059	0.0151	0.0249	0.0292	0.0249	0.0151	0.0059	0.0011
0.0003	0.0020	0.0059	0.0103	0.0123	0.0103	0.0059	0.0020	0.0003
0.0000	0.0003	0.0011	0.0022	0.0027	0.0022	0.0011	0.0003	0.0000

(e) Taps for direct form of highpass filter

Table A.1: The filter taps for our pyramid filters specified one dimension (a), in terms of the McClellan transformation (b-c) and direct form (d-e).

get the lowpass and highpass filter responses. The linear combination coefficients are $b_L[k]$ for the lowpass filter and $b_H[k]$ for the highpass filter. b_L, b_H and the full 9×9 filter taps are shown in Table 1(c-e).

In addition to being invertible, our new pyramid has wider filters, which allows for larger amplification factors in phase based motion magnification as described in Fig. 3.6 of Chapter 3.

	−0.03			
−0.50	−0.48	−0.12	−0.37	−0.12
0.00	0.00	0.00	0.00	0.00
0.50	0.48	0.12	0.37	0.12
	0.03			
(a) 3×1	(b) 5×1		(c) 3×3	

Table A.2: Riesz transform taps for a few sizes. Only one of the Riesz transform filters is shown. The other is given by the transpose.

■ A.2 Approximating the Riesz Transform

In this section, we show how we can replace the expensive Fourier domain implementation of the Riesz transform with an approximate Riesz transform that is implemented with simple primal domain filters. We also briefly go over the cost of this approximation and how spatial smoothing of the phase signal alleviates this cost.

The Riesz transform can be computed in the Fourier domain by using the transfer function

$$-i\frac{(\omega_x, \omega_y)}{\sqrt{\omega_x^2 + \omega_y^2}}. \tag{A.7}$$

In Chapter 3, we demonstrated that the finite difference filter $[-0.5, 0, 0.5]$ and $[-0.5, 0, 0.5]^T$ were good approximations to the Riesz transform when the input is a subband. Here, we further motivate this approximation and provide a method to design spatial domain filters that approximate the Riesz transform of image subbands.

We present an optimization procedure, inspired by Simoncelli [69], to find the taps of spatial domain filters for the Riesz pyramid. The method works by finding taps that minimize the weighted mean squared error between the DTFT of the filter and the Riesz transform transfer function $\frac{\omega_y}{\sqrt{\omega_x^2+\omega_y^2}}$. We choose the weights $W(\omega_x, \omega_y)$ to be the transfer function of a subband filter times the expected power spectrum of images $(\frac{1}{\omega_x^2+\omega_y^2})$ [78]. The Riesz transform filters are 90 degrees symmetric, so we only need to design one of the filters. In addition, each filter is anti-symmetric in one direction

and symmetric in the other. This greatly reduces the number of filter taps we need to specify. We will design the filter that is anti-symmetric in y. The objective function then becomes

$$\int\int W(\omega_x, \omega_y) \left(D_a(\omega_x, \omega_y) - \frac{\omega_y}{\sqrt{\omega_x^2 + \omega_y^2}} \right)^2 d\omega_x d\omega_y \tag{A.8}$$

where D_a is the DTFT of the $2N+1 \times 2M+1$ filter a, given by

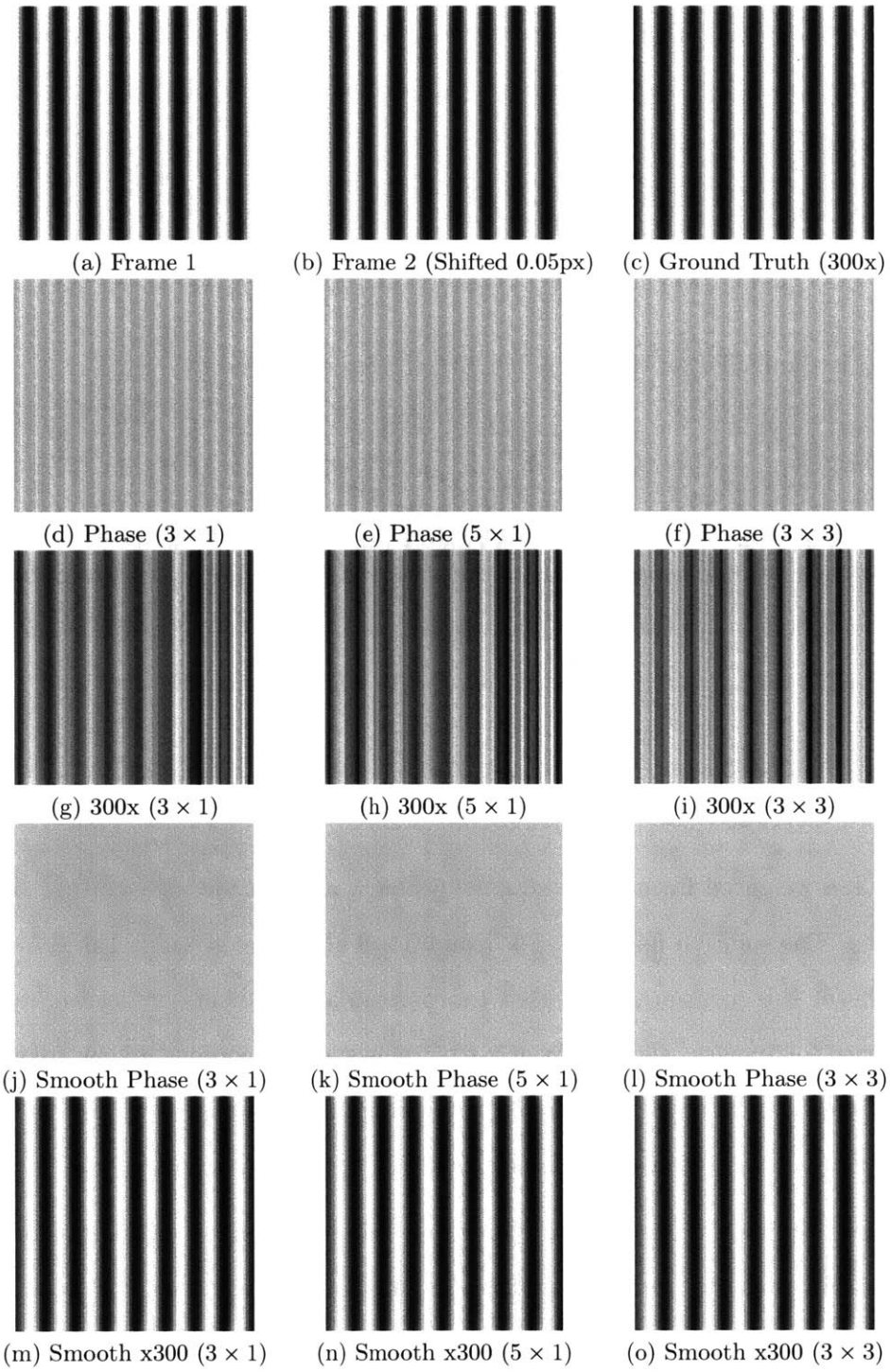
$$\sum_{n=-N}^{N} \sum_{m=-M}^{M} a_{n,m} e^{-i(n\omega_y + m\omega_x)}. \tag{A.9}$$

Note that the first index corresponds to the rows of a (the y-direction), while the second index corresponds to the columns (the x-direction). The symmetries of a imply that $a_{n,m} = -a_{-n,m}$, $a_{n,m} = a_{n,-m}$ and that $a_{0,m} = 0$, which reduces Eq. A.9 to

$$\sum_{n=1}^{N} 2a_{n,0} \sin(n\omega_y) + \sum_{n=1}^{N} \sum_{m=1}^{M} 4a_{n,m} \cos(m\omega_x) \sin(n\omega_y). \tag{A.10}$$

This is a weighted linear least squares problem and can be solved using standard techniques. The solution for 3×1, 5×1 and 3×3 filters are given in Table A.2. Note that filters of this form are often used to approximate gradients or derivatives. This makes sense for images, which have most of their spectral content at low frequencies. Image subbands have much of their spectral content at mid-range frequencies, which is why these filters are better approximations of the Riesz transform.

Limitations Unlike the exact Riesz transform, the approximate Riesz transform does not necessarily preserve the amplitude of a signal. For example, the signal $\cos(\omega x)$ may get mapped to $((1+\epsilon)\sin(\omega x), 0)$. As a result, the phase signal may not be exactly ωx, but $\omega x + O(\epsilon) f(x)$. This means that different parts of the sinusoid get magnified differently. This is illustrated in the second and third rows of Fig. A.2. We use spatial



(a) Frame 1 (b) Frame 2 (Shifted 0.05px) (c) Ground Truth (300x)

(d) Phase (3×1) (e) Phase (5×1) (f) Phase (3×3)

(g) 300x (3×1) (h) 300x (5×1) (i) 300x (3×3)

(j) Smooth Phase (3×1) (k) Smooth Phase (5×1) (l) Smooth Phase (3×3)

(m) Smooth x300 (3×1) (n) Smooth x300 (5×1) (o) Smooth x300 (3×3)

Figure A.2: A comparison of several approximate Riesz transform for phase-based motion magnification with and without spatial smoothing. A sinusoid (a) and a shifted copy (b) are motion magnified 300 times. In the second and third rows, we show the phase signal obtained with three different Riesz transforms and the resulting motion magnified frames. In the fourth and fifth rows, we show the spatially smoothed phase signals and the resulting motion magnified frames.

smoothing to smooth these errors to properly motion magnify the input sequence. As a result, the difference between the three tap Riesz transform filter and the larger filter is negligible (Fig. A.2(n)), which justifies our use of the three tap filter as an approximation.

■ A.3 Pseudocode

Pseudocode to magnify tiny motions using the Riesz pyramid is included below. Please refer to Oppenheim and Schafer for more information on the temporal filters used in this pseudocode [59].

Notation The notation in this pseudocode is based on MATLAB's syntax. All variables are either two dimensional images (possibly of size 1×1) or cell arrays: lists that can contain arbitrary elements. Indexing into an image is denoted by $[\cdot, \cdot]$ and indexing into a cell array is denoted by $\{\cdot\}$. A dot (.) preceding a operator like multiplication ($*$) or exponentiation ($\hat{}$) denotes that the operation is performed element-wise.

In the pseudocode below, we try to use descriptive variable names. However, for variables corresponding to filtered and unfiltered versions of the quaternionic phase

$$\phi\cos(\theta), \phi\sin(\theta), \tag{A.11}$$

this results in overly long variable names. For brevity, we represent $\cos(\theta)$ by only the word `cos` and $\sin(\theta)$ by only the word `sin`. That is, `phase_cos` represents $\phi\cos(\theta)$, not $\cos(\phi)$.

```
1  OnlineRieszVideoMagnification(amplification, low_cutoff, ...
2                                high_cutoff, sampling_rate)
3
4      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Initializes spatial smoothing kernel and temporal filtering
% coefficients.

% Compute an IIR temporal filter coefficients. Butterworth could be replaced
% with any IIR temporal filter. Lower temporal_filter_order is faster
% and uses less memory, but is less accurate. See pages 493-532 of
% Oppenheim and Schafer 3rd ed for more information
nyquist_frequency = sampling_rate/2;
temporal_filter_order = 1;
[B, A] = GetButterworthFilterCoefficients(temporal_filter_order, ...
                                          low_cutoff/nyquist_frequency, ...
                                          high_cutoff/nyquist_frequency);

% Computes convolution kernel for spatial blurring kernel used during
% quaternionic phase denoising step.
gaussian_kernel_sd = 2; % px
gaussian_kernel = GetGaussianKernel(gaussian_kernel_sd);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization of variables before main loop.
% This initialization is equivalent to assuming the motions are zero
% before the video starts.
previous_frame = GetFirstFrameFromVideo();
[previous_laplacian_pyramid, previous_riesz_x, previous_riesz_y] = ...
     ComputeRieszPyramid(previous_frame);
% Do not include lowpass residual
number_of_levels = numel(previous_laplacian_pyramid) - 1;
for k = 1:number_of_levels
   % Initializes current value of quaternionic phase. Each coefficient
   % has a two element quaternionic phase that is defined as
   % phase times (cos(orientation), sin(orientation))
   % It is initialized at zero
   phase_cos{k} = zeros(size(previous_laplacian_pyramid{k}));
   phase_sin{k} = zeros(size(previous_laplacian_pyramid{k}));

```

```
42
43          % Initializes IIR temporal filter values. These values are used during
44          % temporal filtering. See the function IIRTemporalFilter for more
45          % details. The initialization is a zero motion boundary condition
46          % at the beginning of the video.
47          register0_cos{k} = zeros(size(previous_laplacian_pyramid{k}));
48          register1_cos{k} = zeros(size(previous_laplacian_pyramid{k}));
49
50          register0_sin{k} = zeros(size(previous_laplacian_pyramid{k}));
51          register1_sin{k} = zeros(size(previous_laplacian_pyramid{k}));
52      end
53
54
55
56      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57      % Main loop. It is executed on new frames from the video and runs until
58      % stopped.
59      while running
60          current_frame = GetNextFrameFromVideo();
61          [current_laplacian_pyramid, current_riesz_x, current_riesz_y] = ...
62              ComputeRieszPyramid(current_frame);
63
64          % We compute a Laplacian pyramid of the motion magnified frame first
65          % and then collapse it at the end.
66          % The processing in the following loop is processed on each level
67          % of the Riesz pyramid independently
68          for k = 1:number_of_levels
69
70              % Compute quaternionic phase difference between current Riesz pyramid
71              % coefficients and previous Riesz pyramid coefficients.
72              [phase_difference_cos, phase_difference_sin, amplitude] = ...
73                  ComputePhaseDifferenceAndAmplitude(
74                      current_laplacian_pyramid{k}, ...
75                      current_riesz_x{k}, ...
76                      current_riesz_y{k}, ...
77                      previous_laplacian_pyramid{k}, ...
78                      previous_riesz_x{k}, ...
```

```
79                  previous_riesz_y{k});
80
81
82              % Adds the quaternionic phase difference to the current value of
83              % the quaternionic phase.
84              % Computing the current value of the phase in this way is
85              % equivalent to phase unwrapping.
86              phase_cos{k} = phase_cos{k} + phase_difference_cos;
87              phase_sin{k} = phase_sin{k} + phase_difference_sin;
88
89
90              % Temporally filter the quaternionic phase using current value
91              % and stored information
92              [phase_filtered_cos, register0_cos{k}, register1_cos{k}] = ...
93                  IIRTemporalFilter(B, A, phase_cos{k}, register0_cos{k}, ...
94                      register1_cos{k});
95              [phase_filtered_sin, register0_sin{k}, register1_sin{k}] = ...
96                  IIRTemporalFilter(B, A, phase_sin{k}, register0_sin{k}, ...
97                      register1_sin{k});
98
99
100             % Spatial blur the temporally filtered quaternionic phase signals.
101             % This is not an optional step. In addition to denoising,
102             % it smooths out errors made during the various approximations.
103             phase_filtered_cos = ...
104                 AmplitudeWeightedBlur(phase_filtered_cos, amplitude, ...
105                     gaussian_kernel);
106             phase_filtered_sin = ...
107                 AmplitudeWeightedBlur(phase_filtered_sin, amplitude, ...
108                     gaussian_kernel);
109
110
111             % The motion magnified pyramid is computed by phase shifting
112             % the input pyramid by the spatio-temporally filtered quaternionic
113             % phase and taking the real part.
114             phase_magnified_filtered_cos = amplification * phase_filtered_cos;
115             phase_magnified_filtered_sin = amplification * phase_filtered_sin;
```

```
116
117             motion_magnified_laplacian_pyramid{k} = ...
118                 PhaseShiftCoefficientRealPart(current_laplacian_pyramid{k}, ...
119                                               current_riesz_x{k}, ...
120                                               current_riesz_y{k}, ...
121                                               phase_magnified_filtered_cos, ...
122                                               phase_magnified_filtered_sin);
123         end
124
125
126         % Take lowpass residual from current frame's lowpass residual
127         % and collapse pyramid.
128         motion_magnified_laplacian_pyramid{number_of_levels+1} = ...
129             current_laplacian_pyramid{number_of_levels+1};
130         motion_magnified_frame = ...
131             CollapseLaplacianPyramid(motion_magnified_laplacian_pyramid);
132
133
134         % Write or display the motion magnified frame.
135         WriteMagnifiedFrame(motion_magnified_frame);
136         % DisplayMagnifiedFrame(motion_magnified_frame);
137
138
139         % Prepare for next iteration of loop
140         previous_laplacian_pyramid = current_laplacian_pyramid;
141         previous_riesz_x = current_riesz_x;
142         previous_riesz_y = current_riesz_y;
143     end
```

Helper Functions Pseudocode for helper functions is provided below. The helper functions describe how to build a Riesz pyramid, compute quaternionic phase, phase shift Riesz pyramid coefficients, temporally filter phase and spatially blur phase. Pseudocode for functions that compute and collapse Laplacian pyramids, read and write to videos and display images on a screen is *not* included.

```
ComputeRieszPyramid(grayscale_frame)
    % Compute Riesz pyramid of two dimensional frame. This is done by first
    % computing the laplacian pyramid of the frame and then computing the
    % approximate Riesz transform of each level that is not the lowpass
    % residual. The result is stored as an array of grayscale frames.
    % Corresponding locations in the result correspond to the real,
    % i and j components of Riesz pyramid coefficients.
    laplacian_pyramid = ComputeLaplacianPyramid(grayscale_frame);
    number_of_levels = numel(laplacian_pyramid)-1;


    % The approximate Riesz transform of each level that is not the
    % low pass residual is computed. For more details on the approximation,
    % see supplemental material.
    kernel_x = [0.0  0.0  0.0;
                0.5  0.0 -0.5;
                0.0  0.0  0.0];
    kernel_y = [0.0  0.5  0.0;
                0.0  0.0  0.0;
                0.0 -0.5  0.0];
    for k = 1:number_of_levels
        riesz_x{k} = Convolve(laplacian_pyramid{k}, kernel_x);
        riesz_y{k} = Convolve(laplacian_pyramid{k}, kernel_y);
    end
    return {laplacian_pyramid, riesz_x, riesz_y}
```

```
ComputePhaseDifferenceAndAmplitude(current_real, current_x, current_y, ...
                                   previous_real, previous_x, previous_y)
    % Computes quaternionic phase difference between current frame and previous
    % frame. This is done by dividing the coefficients of the current frame
    % and the previous frame and then taking imaginary part of  the quaternionic
    % logarithm. We assume the orientation at a point is roughly constant to
    % simplify the calcuation.

    % q_current = current_real + i * current_x + j * current_y
```

```
% q_previous = previous_real + i * previous_x + j * previous_y
% We want to compute the phase difference, which is the phase of
%    q_current/q_previous
% This is equal to (Eq. 3.12)
%    q_current * conjugate(q_previous)/||q_previous||^2
% Phase is invariant to scalar multiples, so we want the phase of
%    q_current * conjugate(q_previous)
% which we compute now (Eq. 3.9). If the image doesn't change
% much, we can assume orientation is constant and therefore that
% the fourth component of the product is zero.
q_conj_prod_real = current_real.*previous_real + ...
                   current_x.*previous_x + ...
                   current_y.*previous_y;
q_conj_prod_x = -current_real.*previous_x + previous_real.*current_x;
q_conj_prod_y = -current_real.*previous_y + previous_real.*current_y;

% Now we take the quaternion logarithm of this (Eq. 3.15)
% Only the imaginary part corresponds to quaternionic phase.
q_conj_prod_amplitude = sqrt(q_conj_prod_real.^2 + ...
                             q_conj_prod_x.^2 + q_conj_prod_y.^2);
phase_difference = acos(q_conj_prod_real./q_conj_prod_amplitude);
cos_orientation = q_conj_prod_x ./(q_conj_prod_x.^2+q_conj_prod_y.^2);
sin_orientation = q_conj_prod_y ./(q_conj_prod_x.^2+q_conj_prod_y.^2);

% This is the quaternionic phase (Eq. 3.7)
phase_difference_cos = phase_difference * cos_orientation;
phase_difference_sin = phase_difference * sin_orientation;

% Under the assumption that changes are small between frames, we can
% assume that the amplitude of both coefficients is the same. So,
% to compute the amplitude of one coefficient, we just take the square root
% of their conjugate product
amplitude = sqrt(q_conj_prod_amplitude);

return {phase_difference_cos, phase_difference_sin, amplitude)
```

```
IIRTemporalFilter(B, A, phase, register0, register1)
    % Temporally filters phase with IIR filter with coefficients B, A.
    % Given current phase value and value of previously computed registers,
    % comptues current temporally filtered phase value and updates registers.
    % Assumes filter given by B, A is first order IIR filter, so that
    % B and A have 3 coefficients each. Also, assumes A(1) = 1. Computation
    % is Direct Form Type II (See pages 388-390 of Oppenheim and Schafer 3rd Ed.)
    temporally_filtered_phase = B(1) * phase + register0;
    register0 = B(2) * phase + register1 - A(2) * temporally_filtered_phase;
    register1 = B(3) * phase             - A(3) * temporally_filtered_phase;
    return {temporally_filtered_phase, register0, register1}
```

```
AmplitudeWeightedBlur(temporally_filtered_phase, amplitude, blur_kernel)
    % Spatially blurs phase, weighted by amplitude. One half of Eq. 3.26.
    denominator = Convolve(amplitude, blur_kernel);
    numerator   = Convolve(temporally_filtered_phase.*amplitude, blur_kernel);
    spatially_smooth_temporally_filtered_phase = numerator./denominator;
    return spatially_smooth_temporally_filtered_phase;
```

```
PhaseShiftCoefficientRealPart(riesz_real, riesz_x, riesz_y, phase_cos, phase_sin)
    % Phase shifts a Riesz pyramid coefficient and returns the real part of the
    % resulting coefficient. The input coefficient is a three
    % element quaternion. The phase is two element imaginary quaternion.
    % The phase is exponentiated and then the result is mutiplied by the first
    % coefficient. All operations are defined on quaternions.

    % Quaternion Exponentiation
    phase_magnitude = sqrt(phase_cos.^2+phase_sin.^2);  % \|v\| in Eq. 3.13.
    exp_phase_real = cos(phase_magnitude);
    exp_phase_x = phase_cos./phase_magnitude.*sin(phase_magnitude);
    exp_phase_y = phase_sin./phase_magnitude.*sin(phase_magnitude);

    % Quaternion Multiplication (just real part)
```

```
15      result = exp_phase_real.*riesz_real ...
16               - exp_phase_x.*riesz_x ...
17               - exp_phase_y.*riesz_y;
18      return result;
```

Appendix B

Analytic Derivation of Motion Covariance

We analytically derive the covariance matrix of our motion estimate in this section. To simplify the derivation, we assume a model of additive constant variance Gaussian noise rather than signal-dependent noise. That is the observed video $I(x, y, t)$ is contaminated with IID noise $n(x, y, t)$ of variance σ^2:

$$I(x, y, t) = I_0(x, y, t) + n(x, y, t) \tag{B.1}$$

where $I_0(x, y, t)$ is the underlying noiseless video.

To recap, our phase-based motion estimation algorithm works by transforming the image into a complex steerable pyramid representation, computing the local phase of the resulting coefficients and then solving a weighted least squares problem to convert local phase changes into motion vectors at every pixel (Sec. 4.3.1). We investigate the propagation of noise through the stages of this algorithm. First, we compute the noise variance and covariances of the complex steerable pyramid coefficients. Then, we use that to characterize the probability distributions of the local phases. Finally, we describe how to compute the motion's covariance matrix from the variances and covariances of the local phase's probability distributions.

B.1 Noise on Complex Steerable Pyramid Coefficients

Each frame of the video $I(x, y, t)$ is transformed to the complex steerable pyramid representation by being spatially bandpassed by a bank of quadrature pairs of filters G_i and H_i, where i spans the levels of the pyramid across different spatial scales r_i and orientations θ_i. For one such filter pair, the result is

$$G_i * I_0 + G_i * n \text{ and } H_i * I_0 + H_i * n \tag{B.2}$$

where $*$ denote convolution in space. The first term in each expression is the noiseless filter reponse, which we denote $S_{i,0} = G_i * I_0$ for the real part and $T_{i,0} = H_i * I_0$ for the imaginary part. The second term in each expression is filtered noise, which we denote as s_i and t_i. Since n is a zero-mean Gaussian and s_i and t_i are linear functions of n, their probability distribution is characterized entirely by their covariance matrix. We represent this covariance matrix as four functions $C_{i,k,RR}, C_{i,k,RJ}, C_{i,k,JR}, C_{i,k,JJ}$ indexed by pyramid level (i, k) and real (R) and imaginary (J) part. Because n has a shift-invariant probability distribution and convolution is a shift-invariant operator, these functions can be computed via the following auto-correlation and cross-correlation functions

$$C_{i,k,RR}(x_0, y_0, x_1, y_1) := \sigma_n^2 (G_i * G_k)[x_1 - x_0, y_1 - y_0] \tag{B.3}$$

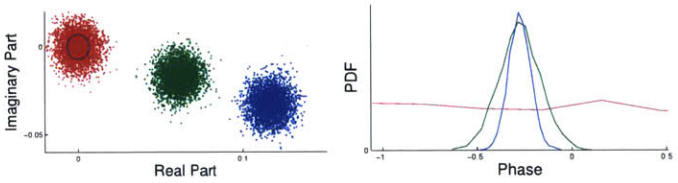
$$C_{i,k,RJ}(x_0, y_0, x_1, y_1) := \sigma_n^2 (G_i * H_k)[x_1 - x_0, y_1 - y_0] \tag{B.4}$$

$$C_{i,k,JR}(x_0, y_0, x_1, y_1) := \sigma_n^2 (H_i * G_k)[x_1 - x_0, y_1 - y_0] \tag{B.5}$$

$$C_{i,k,JJ}(x_0, y_0, x_1, y_1) := \sigma_n^2 (H_i * H_k)[x_1 - x_0, y_1 - y_0] \tag{B.6}$$

where σ_n^2 is the variance of $n(x, y, t)$ [59]. In the steerable pyramid we use [62], each coefficient's real and imaginary parts are uncorrelated and have equal variance, i.e.

(a) Input frame (b) Real Part (c) Imaginary Part



(d) Coefficient Point Cloud (e) PDF of Phases

Figure B.1: Noise model on phases. We take a frame from a synthetic video (a) and build a single level of a complex steerable pyramid (b-c). For three coefficients (red, green and blue dots) with different amplitudes, we show a point cloud of noisy coefficient values (d) and the corresponding histogram of phases (e).

$C_{i,i,RR} = C_{i,i,JJ}$ and $C_{i,i,RJ} = C_{i,i,JR} = 0$. These functions characterize the variances and covariances of the noise term in the complex steerable pyramid coefficients.

■ B.2 Variance and Covariance of Local Phase

Now, we turn to the problem of using these covariance matrices to determine the probability distribution of the local phases. We first demonstrate that if the noiseless amplitude of a coefficient is sufficiently high, its local phase distribution is approximately Gaussian (Fig. B.1). Therefore, as long as we restrict ourselves to points where the amplitude is several times larger than the image noise level, we can characterize the local phase distribution entirely by its second moments (variance and covariance).

Consider a complex steerable pyramid coefficient with noiseless value $F_{i,0} := S_{i,0} + jT_{i,0}$ with noise term $s_i + jt_i$. If its noiseless amplitude $A_{i,0} = \sqrt{S_{i,0}^2 + T_{i,0}^2}$ is close to zero, then its phase is given by

$$\tan^{-1}\left(\frac{t_i}{s_i}\right). \tag{B.7}$$

t_i and s_i are typically uncorrelated (i.e. $C_{i,i,RJ} = 0$) and have equal variance (i.e. $C_{i,i,RR} = C_{i,i,JJ}$), which means that the phase is uniformly random. The phase at such points contains no information and intuitively corresponds to places where there is no image content in a given pyramid level (Fig. B.1, red point).

Now, lets consider coefficients with higher amplitude. The noiseless phase of the coefficient is given by

$$\tan^{-1}(T_{i,0}/S_{i,0}) \tag{B.8}$$

while the noisy phase is given by

$$\tan^{-1}((T_{i,0} + t_i)/(S_{i,0} + s_i)). \tag{B.9}$$

Their difference can be linearized around $(S_{i,0}, T_{i,0})$ to yield

$$\tan^{-1}\left(\frac{T_{i,0} + t_i}{S_{i,0} + s_i}\right) - \tan^{-1}\left(\frac{T_{i,0}}{S_{i,0}}\right) = \frac{sS_{i,0} - tT_{i,0}}{A_{i,0}^2} + O\left(\frac{s^2, st, t^2}{A_{i,0}^4}\right). \tag{B.10}$$

The higher order terms are negligible if the noise terms s and t are small compared to the amplitude $A_{i,0}$. Ignoring these terms, we see that the phase is approximately a linear combination of Gaussian random variables and is therefore Gaussian. This is illustrated empirically by local phase histograms of the green and blue points in Fig. B.1.

Therefore, as long as we ignore coefficients with low amplitude, we can characterize the phase distribution by its covariance matrix $C_{i,k}^{\phi}(x_0, y_0, x_1, y_1)$, which tells us the correlation between the phase at the point (x_0, y_0) in level i and (x_1, y_1) in level k. Let the complex steerable pyramid coefficients at these two points be $F_{i,0} = S_{i,0} + iT_{i,0}$ and $F_{k,1} = S_{k,1} + iT_{k,1}$. $C_{i,j}^{\phi}(x_0, y_0, x_1, y_1)$ is the expectation of the quantity

$$\left(\tan^{-1}\left(\frac{T_{i,0} + t_{i,0}}{S_{i,0} + s_{i,0}}\right) - \tan^{-1}\left(\frac{T_{i,0}}{S_{i,0}}\right)\right)\left(\tan^{-1}\left(\frac{T_{k,1} + t_{k,1}}{S_{k,1} + s_{k,1}}\right) - \tan^{-1}\left(\frac{T_{k,1}}{S_{k,1}}\right)\right). \tag{B.11}$$

We can use the first-order Taylor approximation (Eq. B.10) and the coefficient covariance matrices (Eq. B.6) to simplify this to

$$C^{\phi}_{i,k}(x_0, y_0, x_1, y_1) = \frac{C_{i,k,JJ}S_{i,0}S_{k,1} - C_{i,k,RJ}T_{i,0}S_{k,1} - C_{i,k,JR}S_{i,0}T_{k,1} + C_{i,k,RR}T_{i,0}T_{k,1}}{A^2_{i,0}A^2_{k,1}} \tag{B.12}$$

where the covariance terms on the right side are evaluted at (x_0, y_0, x_1, y_1) and $A_{i,0}$ is the amplitude of $F_{i,0}$ and $A_{k,1}$ is the amplitude of $F_{k,1}$. In the special case of $i = k$, $x_0 = x_1$ and $y_0 = y_1$, we compute the variance of the local phase

$$C^{\phi}_{i,i}(x_0, y_0, x_0, y_0) = \frac{C_{i,i,RR}}{A^2_{i,0}}, \tag{B.13}$$

which follows from the fact that $C_{i,i,RR} = C_{i,i,JJ}$ and $C_{i,i,RJ} = 0$. This gives us the approximate variance and covariance on the phase at points at which the amplitude is of sufficient magnitude. While the coefficient covariance matrix was independent of location and defined fully by auto and cross-correlation function, the phase covariance matrix depends on location because it depends on local image content.

■ B.3 Covariance Matrix of Estimated Motions

In the previous section, we characterized the covariance of the local phases in a single image. Our analysis was independent of the algorithm used to convert local phase changes to motion vectors. To compute the covariance matrix Σ_V of the estimated motions, we must take into account how we compute the motions. In Sec. 4.3.2, we did this by solving a weighted least squares problem, in which the estimated motion $V = [u, v]^T$ is given by a matrix multiply of the nonstochastic matrix $B = (X^T W X)^{-1} X^T W$ by the stochastic vector Y of local phase changes:

$$V = BY. \tag{B.14}$$

We assume that Y is a multivariate Gaussian with mean Y_0 and covariance matrix Σ. The elements of Σ describe the variance and correlations of the local phase changes used to estimate V. They can be filled in using the phase covariance matrix we computed in the previous section (Eq. B.12). The conversion of local phase changes to motion is a linear transform (a matrix multiply by B), which means we can compute Σ_V using the formula

$$\Sigma_V = B\Sigma B^T. \tag{B.15}$$

Computing this covariance matrix analytically can be computationally costly, which is why we opt for a Monte Carlo simulation. In addition, our analysis uses a constant variance noise model. Analytically computing the covariance matrix for a signal-dependent noise model is possible, but doing so would incur an even higher computational cost.

Bibliography

[1] AMIR-KHALILI, A., PEYRAT, J.-M., ABINAHED, J., AL-ALAO, O., AL-ANSARI, A., HAMARNEH, G., AND ABUGHARBIEH, R. Auto localization and segmentation of occluded vessels in robot-assisted partial nephrectomy. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*. Springer, 2014, pp. 407–414.

[2] BAI, J., AGARWALA, A., AGRAWALA, M., AND RAMAMOORTHI, R. Selectively de-animating video. *ACM Transactions on Graphics* (2012).

[3] BAKER, S., SCHARSTEIN, D., LEWIS, J., ROTH, S., BLACK, M. J., AND SZELISKI, R. A database and evaluation methodology for optical flow. *International Journal of Computer Vision 92*, 1 (2011), 1–31.

[4] BELAID, A., BOUKERROUI, D., MAINGOURD, Y., AND LERALLUT, J.-F. Phase-based level set segmentation of ultrasound images. *IEEE Trans. Inf. Technol. Biomed. 15*, 1 (2011), 138–147.

[5] BLABER, J., ADAIR, B., AND ANTONIOU, A. Ncorr: Open-source 2d digital image correlation matlab software. *Experimental Mechanics* (2015), 1–18.

[6] BLANZ, V., AND VETTER, T. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[7] BOJANIC, S., SIMPSON, T., AND BOLGER, C. Ocular microtremor: a tool for measuring depth of anaesthesia? *British Journal of Anaesthesia 86*, 4 (2001), 519–522.

[8] BUADES, A., COLL, B., AND MOREL, J.-M. Nonlocal image and movie denoising. *International Journal of Computer Vision 76* (2008), 123–139.

[9] BURT, P., AND ADELSON, E. The laplacian pyramid as a compact image code. *IEEE Trans. Comm. 31*, 4 (1983), 532–540.

[10] CANNY, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6 (1986), 679–698.

[11] CHEN, J., WADHWA, N., CHA, Y.-J., DURAND, F., FREEMAN, W. T., AND BUYUKOZTURK, O. Structural modal identification through high speed camera video: Motion magnification. *Proceedings of the 32nd International Modal Analysis Conference (to appear)* (2014).

[12] CHEN, J. G., WADHWA, N., CHA, Y.-J., DURAND, F., FREEMAN, W. T., AND BUYUKOZTURK, O. Modal identification of simple structures with high-speed video using motion magnification. *Journal of Sound and Vibration 345* (2015), 58–71.

[13] CHEN, J. G., WADHWA, N., DURAND, F., FREEMAN, W. T., AND BUYUKOZTURK, O. Developments with motion magnification for structural modal identification through camera video. In *Dynamics of Civil Structures, Volume 2*. Springer, 2015, pp. 49–57.

[14] DABOV, K., FOI, A., AND EGIAZARIAN, K. Video denoising by sparse 3d transform-domain collaborative filtering. In *Proc. 15th European Signal Processing Conference* (2007), vol. 1, p. 7.

[15] DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Process. 16*, 8 (aug. 2007), 2080 –2095.

[16] DAVIS, A., BOUMAN, K. L., CHEN, J. G., RUBINSTEIN, M., DURAND, F., AND FREEMAN, W. T. Visual vibrometry: Estimating material properties from small motions in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 5335–5343.

[17] DAVIS, A., CHEN, G, J., AND DURAND, F. Image-space modal bases for plausible manipulation of objects in video. 80.

[18] DEKEL, T., MICHAELI, T., IRANI, M., AND FREEMAN, W. T. Revealing and modifying non local variations in a single image. *ACM Transactions on Graphics (TOG) special issue. Proc. SIGGRAPH Asia* (2015).

[19] DO, M. N., AND VETTERLI, M. Frame reconstruction of the laplacian pyramid. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on* (2001), vol. 6, IEEE, pp. 3641–3644.

[20] DOLLAR, P., TU, Z., AND BELONGIE, S. Supervised learning of edges and object boundaries. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 2, IEEE, pp. 1964–1971.

[21] DOLLÁR, P., AND ZITNICK, C. L. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (2013), IEEE, pp. 1841–1848.

[22] DUDA, R. O., AND HART, P. E. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM 15*, 1 (1972), 11–15.

[23] DxO OpticsPro 10. http://www.dxo.com/us/photography/photo-software/dxo-opticspro.

[24] ELGHARIB, M., HEFEEDA, M., DURAND, F., AND FREEMAN, W. T. Video magnification in presence of large motions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 4119–4127.

[25] FELSBERG, M., AND SOMMER, G. The monogenic signal. *IEEE Trans. Signal Process. 49*, 12 (2001), 3136–3144.

[26] FISCHLER, M. A., AND BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[27] FLEET, D., AND WEISS, Y. Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*. Springer, 2006, pp. 237–257.

[28] FLEET, D. J. *Measurement of image velocity*, vol. 169. Springer Science & Business Media, 2012.

[29] FLEET, D. J., AND JEPSON, A. D. Computation of component image velocity from local phase information. *International Journal of Computer Vision 5*, 1 (1990), 77–104.

[30] FLEET, D. J., AND JEPSON, A. D. Stability of phase information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 15*, 12 (1993), 1253–1268.

[31] FREEMAN, W. T., AND ADELSON, E. H. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence 13*, 9 (1991), 891–906.

[32] FREEMAN, W. T., ADELSON, E. H., AND HEEGER, D. J. Motion without movement. *SIGGRAPH Comput. Graph. 25* (Jul 1991), 27–30.

[33] FUCHS, M., CHEN, T., WANG, O., RASKAR, R., SEIDEL, H.-P., AND LENSCH, H. P. Real-time temporal shaping of high-speed video streams. *Computers & Graphics 34*, 5 (2010), 575–584.

[34] GAUTAMA, T., AND VAN HULLE, M. M. A phase-based approach to the estimation of the optical flow field using spatial filtering. *Neural Networks, IEEE Transactions on 13*, 5 (2002), 1127–1136.

[35] GHAFFARI, R., ARANYOSI, A. J., AND FREEMAN, D. M. Longitudinally propagating traveling waves of the mammalian tectorial membrane. *Proceedings of the National Academy of Sciences 104*, 42 (2007), 16510–16515.

[36] GROSSMAN, W. M. Time shift: Is london's big ben falling down? *Scientific American* (2012).

[37] HARGATHER, M. J., AND SETTLES, G. S. Natural-background-oriented schlieren imaging. *Experiments in fluids 48*, 1 (2010), 59–68.

[38] HARTLEY, R., AND ZISSERMAN, A. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[39] HASINOFF, S. W., DURAND, F., AND FREEMAN, W. T. Noise-optimal capture for high dynamic range photography. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 553–560.

[40] HEALEY, G. E., AND KONDEPUDY, R. Radiometric ccd camera calibration and noise estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 16*, 3 (1994), 267–276.

[41] HORN, B., AND SCHUNCK, B. Determining optical flow. *Artificial intelligence 17*, 1-3 (1981), 185–203.

[42] JOLLIFFE, I. *Principal component analysis.* Wiley Online Library, 2002.

[43] KARASARIDIS, A., AND SIMONCELLI, E. A filter design technique for steerable pyramid image transforms. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on* (1996), vol. 4, IEEE, pp. 2387–2390.

[44] KARIYA, T., AND KURATA, H. *Generalized least squares.* John Wiley & Sons, 2004.

[45] LARKIN, K. G., BONE, D. J., AND OLDFIELD, M. A. Natural demodulation of two-dimensional fringe patterns. i. general background of the spiral phase quadrature transform. *JOSA A 18*, 8 (2001), 1862–1870.

[46] LEE, J., AND SHIN, S. Y. General construction of time-domain filters for orientation data. *Visualization and Computer Graphics, IEEE Transactions on 8*, 2 (2002), 119–128.

[47] LEVIN, A., LISCHINSKI, D., AND WEISS, Y. Colorization using optimization. 689–694.

[48] LEVIN, A., LISCHINSKI, D., AND WEISS, Y. A closed-form solution to natural image matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 2 (2008), 228–242.

[49] LIM, J. J., ZITNICK, C. L., AND DOLLÁR, P. Sketch tokens: A learned mid-level representation for contour and object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on* (2013), IEEE, pp. 3158–3165.

[50] LIM, J. S. *Two-dimensional signal and image processing.* Prentice Hall, Inc., 1990.

[51] LIU, C., AND FREEMAN, W. T. A high-quality video denoising algorithm based on reliable motion estimation. In *Computer Vision ECCV 2010*, vol. 6313. Springer Berlin Heidelberg, 2010, pp. 706–719.

[52] LIU, C., FREEMAN, W. T., SZELISKI, R., AND KANG, S. B. Noise estimation from a single image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 1, IEEE, pp. 901–908.

[53] LIU, C., TORRALBA, A., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. Motion magnification. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 519–526.

[54] LUCAS, B. D., AND KANADE, T. An iterative image registration technique with an application to stereo vision. In *IJCAI* (1981), vol. 81, pp. 674–679.

[55] MCLEOD, A. J., BAXTER, J. S., DE RIBAUPIERRE, S., AND PETERS, T. M. Motion magnification for endoscopic surgery. In *SPIE Medical Imaging* (2014), International Society for Optics and Photonics, pp. 90360C–90360C.

[56] NAKAMURA, J. *Image sensors and signal processing for digital still cameras.* CRC Press, 2005.

[57] NALWA, V. S., AND BINFORD, T. O. On detecting edges. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6 (1986), 699–714.

[58] NICHOLS, G. *Sedimentology and stratigraphy.* John Wiley & Sons, 2009.

[59] OPPENHEIM, A. V., AND SCHAFER, R. W. Discrete-time signal processing. *Prentice Hall, New York* (2010).

[60] PĂTRĂUCEAN, V., GURDJOS, P., AND VON GIOI, R. G. A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In *Computer Vision–ECCV 2012.* Springer, 2012, pp. 572–585.

[61] POH, M.-Z., MCDUFF, D. J., AND PICARD, R. W. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Optics express 18*, 10 (2010), 10762–10774.

[62] PORTILLA, J., AND SIMONCELLI, E. P. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision 40*, 1 (Oct. 2000), 49–70.

[63] ROGGEMANN, M. C., WELSH, B. M., AND HUNT, B. R. *Imaging through turbulence.* CRC press, 1996.

[64] ROLFS, M. Microsaccades: Small steps on a long way. *Vision Research 49*, 20 (2009), 2415 – 2441.

[65] RUBINSTEIN, M. *Analysis and Visualization of Temporal Variations in Video.* PhD thesis, Massachusetts Institute of Technology, Feb 2014.

[66] RUBINSTEIN, M., LIU, C., SAND, P., DURAND, F., AND FREEMAN, W. T. Motion denoising with application to time-lapse photography. *IEEE Computer Vision and Pattern Recognition (CVPR)* (June 2011), 313–320.

[67] SELLON, J. B., FARRAHI, S., GHAFFARI, R., AND FREEMAN, D. M. Longitudinal spread of mechanical excitation through tectorial membrane traveling waves. *Proceedings of the National Academy of Sciences 112*, 42 (2015), 12968–12973.

[68] SETTLES, G. S. *Schlieren and shadowgraph techniques: visualizing phenomena in transparent media.* Springer Science & Business Media, 2012.

[69] SIMONCELLI, E. P. Design of multi-dimensional derivative filters. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference* (1994), vol. 1, IEEE, pp. 790–794.

[70] SIMONCELLI, E. P., ADELSON, E. H., AND HEEGER, D. J. Probability distributions of optical flow. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on* (1991), IEEE, pp. 310–315.

[71] SIMONCELLI, E. P., AND FREEMAN, W. T. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Image Processing, 1995. Proceedings., International Conference on* (1995), vol. 3, IEEE, pp. 444–447.

[72] SIMONCELLI, E. P., FREEMAN, W. T., ADELSON, E. H., AND HEEGER, D. J. Shiftable multi-scale transforms. *IEEE Trans. Info. Theory 2*, 38 (1992), 587–607.

[73] SUN, D., ROTH, S., AND BLACK, M. J. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision 106*, 2 (2014), 115–137.

[74] SUTTON, P. J., AND KUSMARTSEV, F. V. Gravitational vortices and clump formation in saturn's f ring during an encounter with prometheus. *Scientific reports 3* (2013).

[75] TRITTON, D. J. Physical fluid dynamics. *Oxford, Clarendon Press, 1988, 536 p. 1* (1988).

[76] UNSER, M. Splines: A perfect fit for signal and image processing. *Signal Processing Magazine, IEEE 16*, 6 (1999), 22–38.

[77] UNSER, M., SAGE, D., AND VAN DE VILLE, D. Multiresolution monogenic signal analysis using the riesz–laplace wavelet transform. *Image Processing, IEEE Transactions on 18*, 11 (2009), 2402–2418.

[78] VAN DER SCHAAF, V. A., AND VAN HATEREN, J. V. Modelling the power spectra of natural images: statistics and information. *Vision research 36*, 17 (1996), 2759–2770.

[79] WACHEL, J., MORTON, S. J., AND ATKINS, K. E. Piping vibration analysis.

[80] WADHWA, N., DEKEL, T., WEI, D., DURAND, F., AND FREEMAN, W. T. Deviation magnification: revealing departures from ideal geometries. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 226.

[81] WADHWA, N., RUBINSTEIN, M., DURAND, F., AND FREEMAN, W. T. Phase-based video motion processing. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 80.

[82] WADHWA, N., RUBINSTEIN, M., DURAND, F., AND FREEMAN, W. T. Riesz pyramid for fast phase-based video magnification. In *Computational Photography (ICCP), 2014 IEEE International Conference on. IEEE* (2014).

[83] WANG, P., CASADEI, F., SHAN, S., WEAVER, J. C., AND BERTOLDI, K. Harnessing buckling to design tunable locally resonant acoustic metamaterials. *Physical review letters 113*, 1 (2014), 014301.

[84] WIENER, N. *Extrapolation, interpolation, and smoothing of stationary time series*, vol. 2. MIT press Cambridge, MA, 1949.

[85] WU, H.-Y., RUBINSTEIN, M., SHIH, E., GUTTAG, J. V., DURAND, F., AND FREEMAN, W. T. Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph. 31*, 4 (2012), 65.

[86] XU, J., MOUSSAWI, A., GRAS, R., AND LUBINEAU, G. Using image gradients to improve robustness of digital image correlation to non-uniform illumination: Effects of weighting and normalization choices. *Experimental Mechanics 55*, 5 (2015), 963–979.

[87] XUE, T., RUBINSTEIN, M., WADHWA, N., LEVIN, A., DURAND, F., AND FREEMAN, W. T. Refraction wiggles for measuring fluid depth and velocity from video. In *Computer Vision–ECCV 2014*. Springer, 2014, pp. 767–782.

[88] ZORAN, D., AND WEISS, Y. Scale invariance and noise in natural images. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 2209–2216.