VIDEO MAGNIFICATION FOR STRUCTURAL ANALYSIS TESTING

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Jean Whitmore

June 2018

COMMITTEE MEMBERSHIP

TITLE: Video Magnification for Structural Analysis Testing

AUTHOR: Jean Whitmore

DATE SUBMITTED: June 2018

COMMITTEE CHAIR: Wayne Pilkington, Ph.D.
Associate Professor of Electrical Engineering

COMMITTEE MEMBER: Jane Zhang, Ph.D.
Associate Department Chair, EE

COMMITTEE MEMBER: Xiao-Hua Yu, Ph.D.
Professor of Electrical Engineering

ABSTRACT

Video Magnification for Structural Analysis Testing

Jean Whitmore

The goal of this thesis is to allow a user to see minute motion of an object at different frequencies, using a computer program, to aid in vibration testing analysis without the use of complex setups of accelerometers or expensive laser vibrometers. MIT's phase-based video motion processing was modified to enable modal determination of structures in the field using a cell phone camera. The algorithm was modified by implementing a stabilization algorithm and permitting the magnification filter to operate on multiple frequency ranges to enable visualization of the natural frequencies of structures in the field. To implement multiple frequency ranges a new function was developed to implement the magnification filter at each relevant frequency range within the original video. The stabilization algorithm would allow for a camera to be hand-held instead of requiring a tripod mount. The following methods for stabilization were tested: fixed point video stabilization and image registration. Neither method removed the global motion from the hand-held video, even after masking was implemented, which resulted in poor results. Specifically, fixed point did not remove much motion or created sharp motions and image registration introduced a pulsing effect. The best results occurred when the object being observed had contrast from the background, was the largest feature in the video frame, and the video was captured from a tripod at an appropriate angle. The final program can amplify the motion in user selected frequency bands and can be used as an aid in structural analysis testing.

ACKNOWLEDGMENTS

I would like to thank my family and friends for all their support throughout this process. I would like to especially thank my father for helping me come up with the idea for the thesis and providing a mechanical engineering perspective to computer vision. Finally, I would like to thank Dr. Pilkington for helping me navigate the scope and reporting of this project.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Chapter 1: Introduction & Background Information

1.1: Non-Destructive Testing

When a structure is in the presence of an external mechanical frequency stimulus or vibration it begins to oscillate at a specific characteristic frequency, which is commonly referred to as the natural frequency response. If the external forces occur at the same frequency as the natural response the structure's motion is amplified, also known as mechanical resonance, and could lead to damage in the structure [1]. The purpose of non-destructive testing is to determine if a structure will be damaged under extremes of its expected environmental conditions, and if so to measure the extent of the damage. Any damage simulated in the testing should not destroy the structure. Such testing is typically used to ensure the structure can survive and continue to function in a given extreme environment, such as a windy hillside, space, or a rocket launch. Originally non-destructive testing results were determined using human observations and visual assessments; but as ensuring structural integrity became more critical, better measuring equipment such as boroscopes, microphones, ultrasonic transducers, and other sensors have been introduced to non-destructive testing to achieve more accurate damage assessments [2]. However, the greater measurement accuracy leads to an increase in cost for the measuring devices and does not provide the tester or structure designer with a visual depiction of the structure's motion which could be very helpful for understanding the source and natural of the structure's response.

1.2: Vibration Testing

One commonly used non-destructive test is vibration testing. Vibration testing, as described in Kenneth McConnel's *Vibration Testing: Theory and Practice*, is used to measure the dynamic response of a system under given environmental conditions [3]. Typically, this means a structure is subjected to a vibrational frequency that would occur in the environment it would be operating in, for example severe vibrations during a rocket launch, and observing the frequency responses from particular areas of the structure. Three of the most common tests that fall under vibration testing are vibration monitoring, vibration survey, and modal analysis [3]. Vibration monitoring is used to assess if the structure can operate correctly in the environment [3]. The vibration survey is used to see how a structure responds to vibrations applied over a certain frequency range, while modal analysis gives details on the structure's dynamic characteristics, specifically how it moves and in which directions [3]. McConnel also describes a general vibration testing configuration (Figure 1.1)

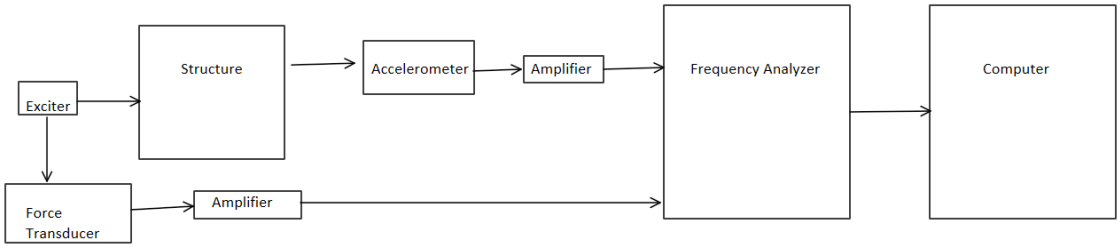


Figure 1.1: General Vibration Testing Method

In this system an excitation is placed on the structure and measured through an input force transducer, while motion of the structure is then measured by an accelerometer. Both signals from the force transducer and accelerometer are then

sent to an amplifier and then into a frequency spectrum analyzer. The data collected from the frequency analyzer is then collected and stored in a computer and displayed and analyzed by a computer program which the engineers use to evaluate the structure's performance. While this test setup provides the desired quantitative motion measurements at different frequencies, it does not provide a way to view the motion of the structure; particularly because most of the structure's movements are extremely small.

To produce a visualization of the structure's dynamic behavior, MIT has developed a MATLAB program that can detect small motions over a set range of vibration frequencies in video recordings of a structure under test [4]. The program uses complex steerable pyramids to magnify motions in the video using the phase difference between each frame. In a separate experiment the researchers at MIT were able to extract vibration data from videos of rotating machinery, a small beam, and an earthquake [2]. With the vibration data from the video, the non-destructive tests could not only gather quantitative vibration data but also produce a video of the structure over that frequency range, which allows an observer to see what parts of the structure move at that frequency and by what relative amount. Having the amplified motion video enables the non-destructive test to be more effective and to provide information on errors in the testing setup and feedback to the modeling methods used to create the structure.

1.3: Scope of Thesis

The goal of this thesis project is to design a MATLAB program to allow a person to see and magnify mechanical vibrations within certain frequency bands

of the primary structure seen in a video recording. Unlike the original MIT system, the program from this project will be designed to operate with the more challenging case of motion video recorded by a handheld device, as well as the simpler case of video from a stable, tripod-mounted camera. The result will provide another method for engineers to verify that modal frequencies occur near the ones predicted by modeling; and it can aid in the detection of unwanted vibration frequencies that occur from the testing apparatus. The program could also provide an inexpensive means of vibration characterization testing without requiring additional expensive equipment or testing time in specialized vibration test facilities; which could provide a considerable cost savings in the development of new structures.

CHAPTER 2: RELEVANT WORK & SYSTEM DESIGN

The focus of this chapter is to explain the related works that contributed to this project and how they are applied in this project. This chapter also covers how the MATLAB program for this project is structured and how it functions, to provide context for where the related work contributed to this project.

2.1: Program Design

The purpose of the MATLAB program is to aid in the visualization of motion of a structure during structural analysis tests, such as vibration testing. To accomplish this the program has three functions: magnification, stabilization, and user input. The top-level functionality of the program is shown in Figure 2.1.1.

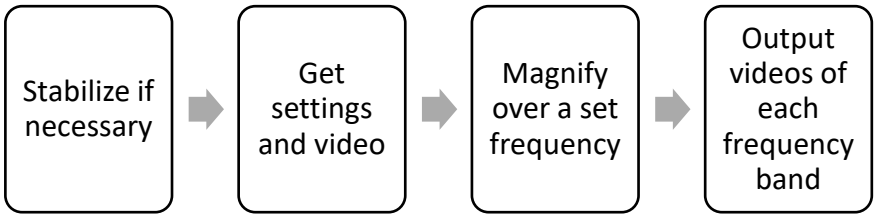


Figure 2.1.1: Top-level functional diagram for MATLAB program

The first block shown in Figure 2.1.1 is the stabilization block. The purpose of this block is to ensure the video does not contain any extraneous motion if it was captured from a handheld device. The second block shown in Figure 2.1.1 gathers the input video, magnification level, and frequency bands from the user. This block is implemented using a function called *freqbandamp* (in Appendix A). The function takes in the user inputs for the frequency bands as an array of low frequencies and an array of high frequencies, the amount of magnification, the output directory, frequency sample rate, and a video to have motions magnified.

Then for each pair of low and high frequencies the function passes the user input to the MIT *phaseamplify* function which magnifies the video at the desired frequency bands (magnification block in Figure 2.1.1). The final block represents the output of the system which are videos with magnified movement for each of the user specified frequency bands.

The program is designed to allow for a person performing a structural analysis test, like vibration testing, to capture a short hand-held video from their cellphone and amplify it over any given frequency range. However, the frequency bands are limited to what the camera can capture accurately because the Nyquist Criterion requires the sampling rate to be at least twice the frequency you desire to reconstruct. For example, the Samsung Galaxy S8 can capture video at 30, 60 or 240 frames per second which would limit the maximum vibration frequencies that can be visualized to 15, 30, and 120 Hz respectively [5].

2.2: MIT Phase Based Amplification Method

The magnification functionality of this project uses the MIT phase-based amplification algorithm MATLAB code. This code was an extension of previous work at MIT that originally used Euclidean models to amplify the amplitude of objects moving in a video. While their Euclidean method was successful; if there was any type of noise in the video, the noise was also amplified, which is undesirable. To address the noise issue, MIT focused on developing a phase-based motion amplification model, which resulted in more amplification and less sensitivity to noise. Since the phase-based method provides more amplification than the previous work, it is an appropriate tool for seeing the smaller motions

present during vibration testing. Another benefit of the phase-based method is that it incorporates a threshold parameter α, which can be used to minimize large motions present in a stable video [4].

The MIT code amplifies the small motions by amplifying the small changes in local phase using a complex steerable pyramid. A pyramid is made up of multiple images in different sizes as shown in Figure 2.2.1.

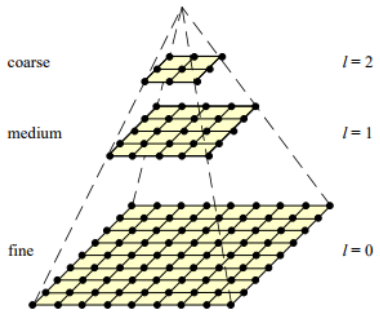


Figure 2.2.1: Basic Image Pyramid diagram from *Computer Vision: Algorithms and Applications* [6]

The reason the higher layers appear smaller in Figure 2.2.1 is because the image pixels in each layer as you move up the pyramid are down sampled by a factor of two. When the image is down sampled it decreases in size and image details become less refined. Multi-resolution pyramids are typically used to accomplish image decomposition for image compression, using orthogonal wavelet, Laplacian, or Discrete Cosine decompositions. Pyramids are also used to detect objects in an image when given a template example of the desired object. Most object detection algorithms that use template matching are sensitive to scale and rotation, which means any difference between the size of the template and the

object in the picture will result in no detection. This is not ideal when trying to detect multiple objects, like faces, in an image. To help find all the objects of the same type in an image despite them having different sizes from each other and compared to the template, multiple levels of image pyramids with different scalings are tested. This allows different sized features in the image to be scaled to better match the size of the template, resulting in a higher likelihood of detection.

The complex steerable pyramid used in the MIT algorithm offers additional benefits beyond a typical image pyramid. A steerable pyramid performs a non-orthogonal polar-separable decomposition in the frequency domain, thus allowing independent representation of scale, orientation, and position. The resulting image representation is both translation-invariant and rotation-invariant. The primary drawback of this method is that the representation is overcomplete by a factor of 4k/3, where k is the number of orientation bands used in the decomposition; thus making this decomposition less data efficient. The overcomplete property, however, means that there is no aliasing between levels in the pyramid, which does occur in other types of pyramid decompositions because information is lost in the downsampling at each of the pyramid levels [4]. Since the complex steerable pyramid does not alias any information, a perfect reconstruction of the video can be recreated from the decomposition.

The other unique element of steerable pyramid decomposition that is used here is the quadrature phase filters, which provide even and odd-phase orientations of the spatial filters [4]. These both output complex values which can

be used to extract the magnitude and phase measurements in a local area in the video. Just as the phase variations of Fourier basis functions (sine waves) are related to translation via the Fourier shift theorem, the phase variations of the complex steerable pyramid correspond to local motions in spatial subbands of an image [4]. The pyramid also stores the position and scale of the video frames, which are useful when you desire to amplify the phase in a video. Also, the complex steerable pyramid uses a Gabor wavelet transform instead of a traditional wavelet transform, which allows for better approximation of large changes in contrast in an image.

The magnification method utilizes the steerable pyramid and bandpass filters to extract and magnify the motion a video. The process is described in Figure 2.2.2, which is the figure used in the MIT phase-based motion paper to describe their magnification method.

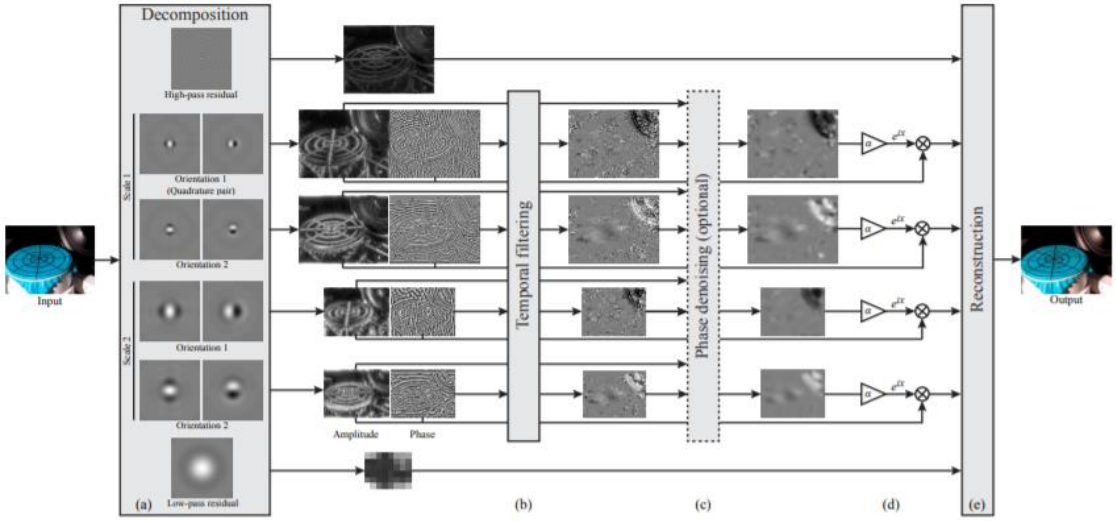
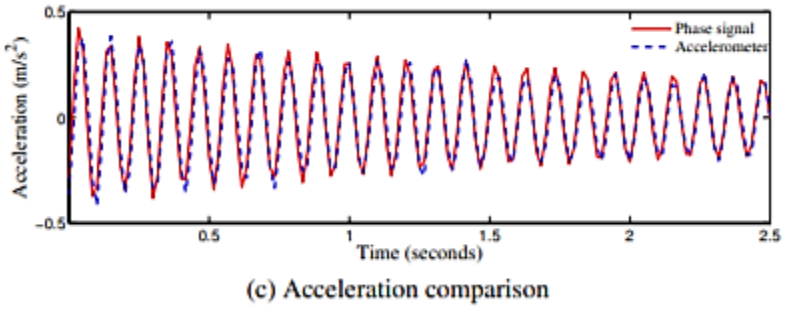


Figure 2.2.2: MIT motion magnification method from “Phase-Based Video Motion Processing” [4]

The first stage of the process is to calculate the local phase over every orientation and scale in the complex steerable pyramid for each single frame of the input video (Figure 2.2.2 a). The number of orientations can be set by the user when deciding what pyramid to use. For example, the octave pyramid has 4 orientations while the half octave has 8. The scales are calculated in the *getFilters* which is part of *phaseAmplify*. The number of scales used is dependent on the size of the video. In the implementation of this project octave pyramids were used, and the size of the video was 720 by 1080, which resulted in 30 scales. Then the localized phase information from time adjacent video frames sent into a temporal bandpass filter (Figure 2.2.2 b) to isolate changes occurring within the user specified range of frequencies. These temporally bandpassed phases correspond to motion in different spatial scales and orientations. Depending on user input, the amplitude and phase can be passed through an amplitude-weighted spatial smoothing filter to increase the phase SNR value (Figure 2.2.2 c). After the optional denoising step, the phase difference is calculated, and the motion is either magnified with the given alpha value or attenuated depending on if the motion is to be magnified or suppressed (Figure 2.2.2 d). The threshold between attenuation and magnification can also be set by the user. The magnification or attenuation is accomplished by modifying the value stored in the complex steerable pyramid coefficient. Finally, the frame is reconstructed by inverting the process with which the image pyramid was created. This process is repeated over every single frame in a given video.

There is a limitation on how much the phase can be shifted until the limits in the spatial support of the pyramid are reached. For a full octave bandwidth the upper bound is $\frac{\lambda}{4}$, where λ is the inverse of the frequency the filter selects; while half-octave bandwidth filters have a bound that is $\frac{\lambda}{2}$ [4]. The increase occurs because the filters in the sub-octave band are narrower in the frequency domain, which means in the spatial domain they are larger and there is more room to move the image frames without causing distortions. In terms of the magnification in a video this mean objects that move at a lower frequency in the video can have their phase shift magnified more than objects that move at a higher frequency. Also, this shows that real-time results, which are typically carried out using an octave pyramid, will have different results than when using a sub-octave pyramid.

Overall the algorithm is successful at magnifying small motions that are not noticeable to the human eye [4]. In the original papers, there was also some experimentation on how effective the algorithm is at accurately representing the actual motion in a scene. To test the algorithm, a metal structure with an accelerometer attached was hit with a hammer to induce vibrations. The test was recorded by a DSLR camera at 60 frames per second and passed through the algorithm to find the phase differences. The structure displacements were calculated from the phase differences; and the acceleration of the structure was found by taking the second derivative of the Gaussian-filtered displacement data, and rescaling and realigning the results to match the scale and position of the accelerometer data. A comparison of the accelerations measured by the accelerometer and the calculated algorithm results are plotted in Figure 2.2.3.

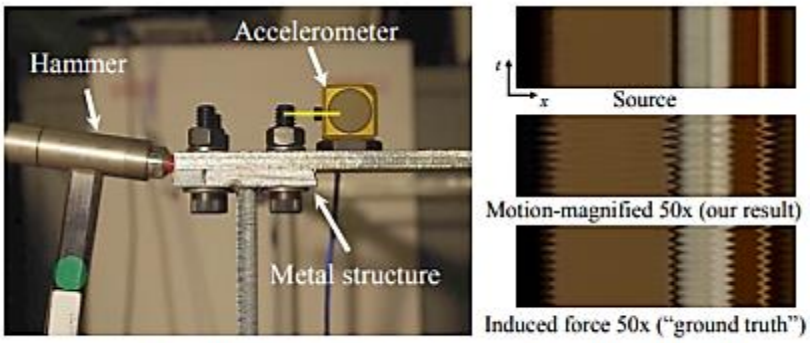


(c) Acceleration comparison

Figure 2.2.3: Acceleration Comparison graph from "Phase-Based Video Motion Processing" [4]

The acceleration generated from the phase signal by the algorithm is shown in red and the measured accelerometer data is shown in blue. The motions between the two are nearly identical with only the peaks not matching up exactly, which means the algorithm does properly capture motions in the video sequence.

Another test performed with the same apparatus was to magnify the original video motion by 50 times and compare that motion in the video to a video that captured the motion with 50 times harder force on the metal structure. The resulting image frames are shown in Figure 2.2.4.



(a) Experimental setup (b) Spatiotemporal slices

Figure 2.2.4: Image comparison from "Phased-Based Video Motion Processing" [4]

The experimental structure is shown in Figure 2.2.4 (a) and the image frames are shown in Figure 2.2.4 (b). The source shown in the top has its motion magnified

50 times by the algorithm, which resulted in the middle picture. The bottom image in Figure 2.2.4 (b) is a frame from when the hammer force was increased by 50 times. The motion magnified image captures the exact same motion, but with the lines extend in the light brown area. These results coupled with the acceleration comparison validate the algorithm's ability to capture real motion and provide strong motivation for use in a vibration testing scenario.

The MIT paper concludes that the phase-based algorithm is less susceptible to noise and has a larger magnification than a previous Eulerian Video Magnification method [4] . However, there are still limitations with the current algorithm. One of the main limitations is that the video must be captured from a camera on a stable platform, like a tripod, and not by a handheld device. The reason behind this is that when a camera is in a person's hand there will be small movements that occur due to minute hand motions. These hand motions fall in the vibration frequency ranges being amplified by the algorithm and will distort the motion of the object or structure being analyzed. Another issue with the phase magnification method is that noise in the video could also cause distortions that would mask the movement. The final limitation is the length of a video that can be processed in a reasonable time. Longer and higher resolution videos take longer to process; although they might yield better results. However, these limitations can be overcome by acquiring shorter videos, using an image stabilization filter, and ensuring that the video does not contain excessive noise.

2.3: Stabilization Approaches

Since the current MIT magnification algorithm is limited to a camera on a tripod, the second component of the project focuses on stabilizing a hand-held video so accurate vibrations can be detected with less expensive and more convenient means. The image stabilization method employed needs to be able to handle the non-linearities that are introduced by the rolling shutter of a CMOS image sensor, which most cellphones use to take videos and pictures [7]. Rolling shutter is when an image is captured row by row instead of all at once. The time difference between each row capture cause distortions in the image when there is movement form the camera [7]. For example, if there is horizontal movement the frame will be bent to one side, while if there is vertical movement the frames will either be shrunk or stretched [7]. Both distortions do not affect every pixel in the frame the same, so using an inverse filter will not work. Typically, inverse filtering is done using a linear method, however since not all the pixels have the same amount of distortion a nonlinear method needs to be used [7]. Also, since hand motion is not the same throughout the video each frame will have to be adjusted one at a time. Two methods were explored as possible solutions: MATLAB's video stabilization algorithm and MATLAB's image registration example.

The MATLAB video stabilization algorithm essentially finds the differences in translation, scaling, and rotation between a point in two frames of a video and applies a transform to make the frames align. This is repeated for all the

frames in a video. The way MATLAB implements the algorithm is described in Figure 2.3.1.

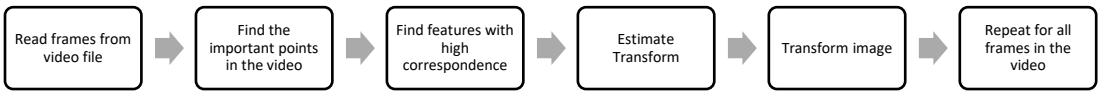


Figure 2.3.1: Top level MATLAB video stabilization method functional diagram

First the stabilization algorithm reads individual frames from the input video. Next the algorithm determines the important points in each frame using the Features from Accelerated Segment Test (FAST) feature detection algorithm, which is one of the fastest corner detection algorithms [8]. FAST works by first detecting features and then generating a feature vector. Specifically the features are detected by using a circle of 16 pixels around a center pixel and if there are 12 pixels in the circle that are either above or below a designated threshold of the center pixel value the pixel is marked as a feature [9]. The 16 points are then saved as a feature vector and can be matched by using the FAST feature matching algorithm or another algorithm [9]. In this implementation, once the corners are detected, they are compared to the other features using the Fast Retina Key (FREAK) point descriptors of each of the features. FREAK descriptors are designed to mimic the human retina, which is accomplished by "comparing the light intensities over a retinal sampling pattern" [10]. The FREAK descriptors produce a binary output and the Hamming distance of features in each of the image frames is then calculated. The points with the smallest Hamming distance between the frames are determined to be the same feature. Next the spatial

transform between images is estimated by using the M-estimator Sample Consensus (MSAC) algorithm, which is a variant of the Random Sample Consensus (RANSAC) algorithm. The RANSAC algorithm takes a random dataset, fits a model to the data set and calculates the number of outliers [8]. The algorithm reiterates until the stop criteria is met. The MSAC algorithm performs in a similar manner to the RANSAC, but it includes an M-estimator to provide a probability component to the algorithm [11]. The probability component reduces the number of iterations the RANSAC algorithm must do to find a model that best fits the data [9]. The process used to estimate the transform is encapsulated in Figure 2.3.2.

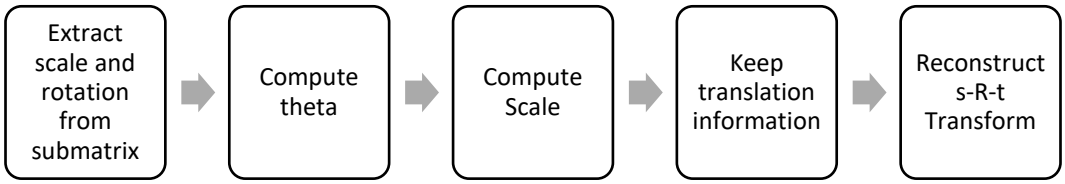


Figure 2.3.2: Process to estimate the transform matrix

The first part of the transform estimation uses the MSAC algorithm and determines the points from the first frame that match closest to points in the second frame. The initial transform from the MSAC algorithm is represented in the form:

$$\begin{bmatrix} a_1 & a_3 & 0 \\ a_2 & a_4 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

where *a* represents the scale, rotation and shearing effects and *t* represents translation. The estimation begins by extracting the scale and rotation parameters

from the initial transform and computes the rotation angle theta and scale through averaging the two possible values for each of the parameters. The translation is kept the same and the matrix is updated to the form:

$$\begin{bmatrix} s * \cos(theta) & s * -\sin(theta) & 0 \\ s * \sin(theta) & s * \cos(theta) & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

which is then used by the algorithm to warp the current image frame to be on the same level as the previous frame; and repeats the process for the rest of the video frames.

The second method for stabilization focused on applying MATLAB's image registration algorithms. The main difference between MATLAB's implementation of image registration and the previous method is that registration uses only scale and rotation to align images, but not translation; unlike the MATLAB video stabilization method. Since this method does not contain any translation, which is a significant component in hand motion, the performance should be worse than the previous stabilization method described. The overall process for image registration is described in Figure 2.3.3.

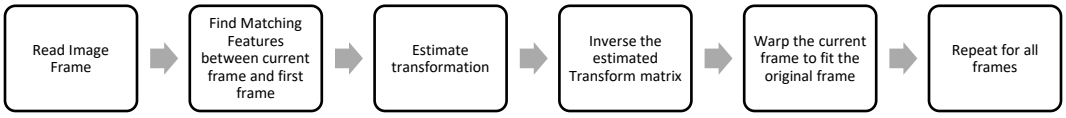


Figure 2.3.3: Top level functionality of the MATLAB image processing example

The method takes in an image frame from the video and collects the features from it using the Speeded Up Robust Features (SURF) algorithm which is a patented feature detector like the FAST algorithm used in the stabilization method [12].

The SURF algorithm is also used to detect features in a reference frame. The SURF algorithm detects features by first using a square hessian matrix detector to simulate a Gaussian smoothing filter [13]. The filter size is upscaled to generate different pyramid levels [13]. The features are detected by suppressing any pixel value that is not the maximum value in its 3x3x3 neighborhood and interpolating the maximum determinant of the Hessian matrix to correspond with the image in the pyramid [13]. However, the FAST algorithm can also be used for feature detection. In this instance FAST feature detection was used because the structures tested were made up of corners, which SURF does not detect as well as FAST [12]. Also keeping the detection schemes the same provides for a better comparison between the two methods. Next the features are matched using the MSAC algorithm, just like the video stabilization method, and a spatial transform between frames estimated. The estimation process is the same as the one described in Figure 2.3.2, but instead of the scale and theta being averaged between the two frames, they are kept the same as the initial values calculated from the original transformation matrix. The inverse of the estimated matrix is used to rotate the current frame back into the reference image frame's orientation. To make the algorithm work over a whole video the example code was formatted into a function that repeats the procedure in Figure 2.3.3 until the end of the video. Also, the reference frame was set to the first frame in the video so a constant reference frame without any distortion would be used throughout the image registration stabilization process.

CHAPTER 3: TESTING & RESULTS

To verify the functionality of the MATLAB program used to implement the vibration detection system, the testing was broken down into verifying three different functions: separating the motions in the video into multiple frequency bands, verifying the ability to amplify and see expected motions, and stabilizing handheld video. The videos resulting from these tests are linked in Appendix B.

3.1: Frequency Separation Testing

To test the frequency separation functionality of the MATLAB program a guitar (guitar.avi) provided by MIT was input into the system. The guitar.avi video captures the motion after a guitar is strummed at 600 fps for 10 seconds at a resolution of 432 by 192 pixels. The first string's (E_2) fundamental musical frequency is 82.4 Hz if properly tuned, while the second (A_2) and third (D_3) are 110 Hz and 146.8 Hz respectively. To test the functionality 5 frequency bands were tested. Three of them are around the fundamental frequencies of the first three strings, one in a band below the first string's fundamental frequency, and the last one in a band between the first and second string's fundamental frequencies. To capture the motion a column capturing the first 4 strings (Figure 3.1.1) was observed over the duration of the video for each of the frequency bands (Figure 3.1.2). Note the bands are wider than the fundamental frequency to account for the possibility the guitar is not tuned.

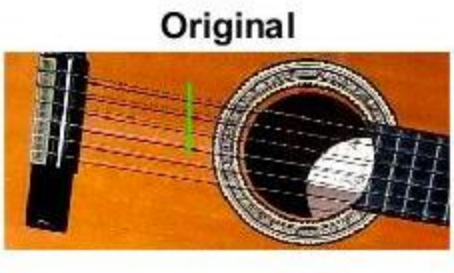


Figure 3.1.1: Location of Column in original video

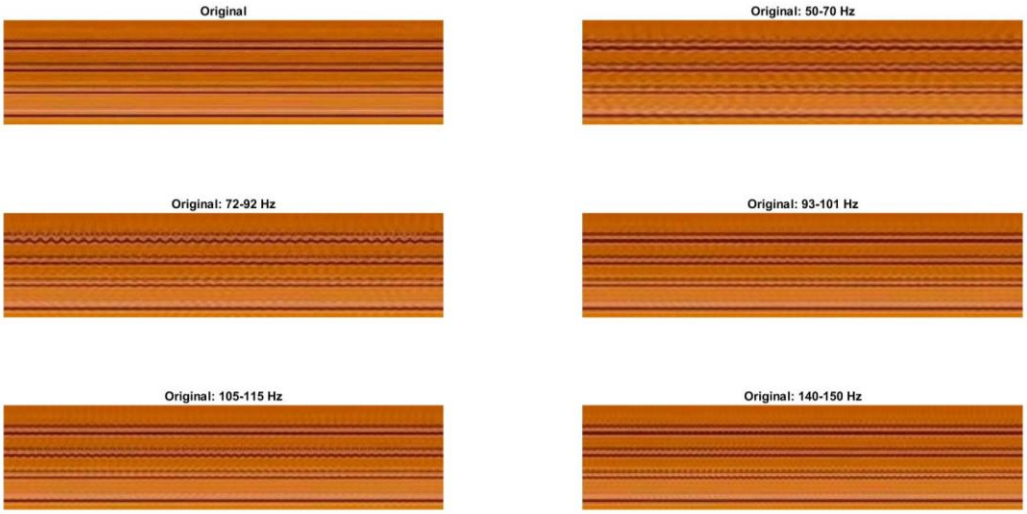


Figure 3.1.2: Motion of guitar.avi (Magnification Factor: 75)

The results from Figure 3.1.2 show that the lowest string (E_2) only appears to vibrate in the 72 to 92 Hz range, as expected. In frequencies below 72 Hz there was no motion in any of the strings other than those caused by distortion, and for frequencies from 92 to 101Hz there were no strings moving. Again, these results are as expected because if there is movement in the higher strings they would occur at their resonant frequencies and since the string that was plucked was the lowest string and only its fundamental frequency and its harmonics would be excited, no vibrations below its fundamental frequency should be observed. The

results also show only A_2 moving in the 105 to 115Hz band and D_3 only moving in the 140 to 150 Hz band, which makes sense since each string only vibrates in their frequency range. All these results collectively illustrate the effectiveness of the frequency separation in the MATLAB program.

Since, the function is effectively separating the frequencies the next aspect tested was the affect resolution has on the results. To test the effect of resolution guitar.avi was down sampled by 2 and by 4, which generated two 9 second videos at 216 by 96 pixels and 108 by 48pixels. Figure 3.1.3 shows the column in each video used for comparing the motion, which are all the same location in each video frame. Figure 3.1.4 to Figure 3.1.6 show the results at each frequency band for each of the resolutions.

Figure 3.1.3: Columns observed in each video

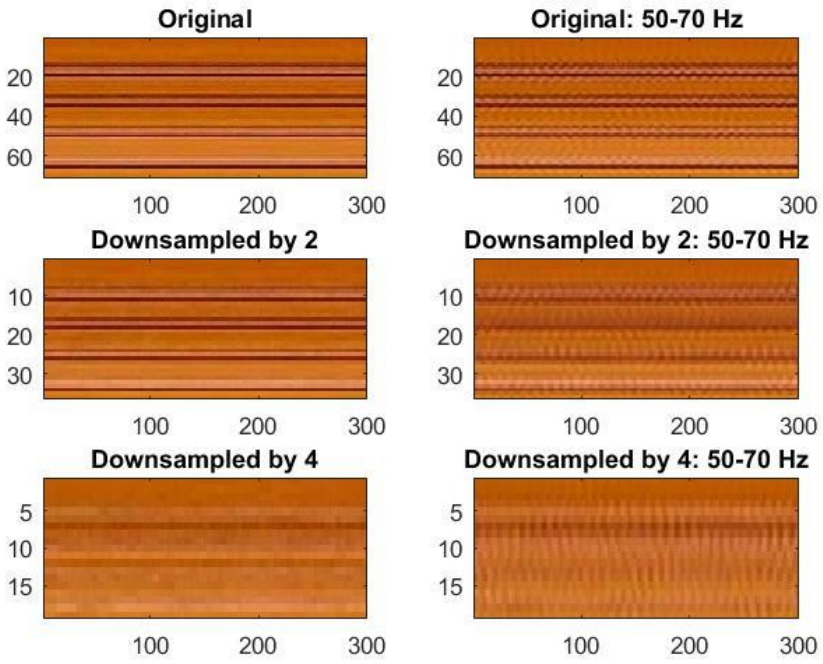


Figure 3.1.4: Original and 50-70 Hz frequency band comparison

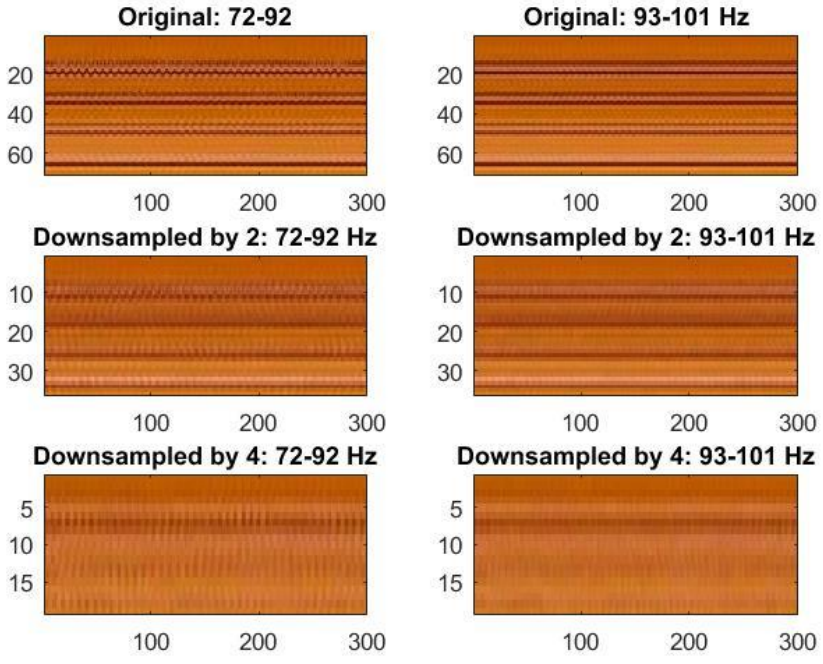


Figure 3.1.5: 72-92 and 93-101 frequency bands comparison

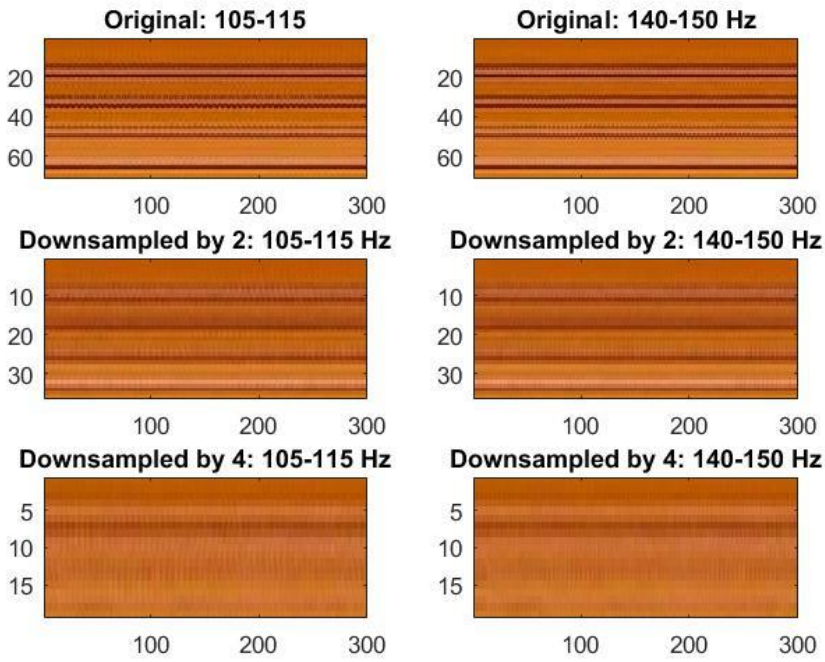


Figure 3.1.6: 105-115 Hz and 140-150 Hz frequency bands comparison

The results show that decreasing the resolution decrease the ability to observe any motion in the video. The 108 by 48-pixel video did not show any motion and the motion in the 216 by 96 -pixel video show some motion, but not at the same clarity at the higher resolution. In fact, the motion in the 216 by 96 video could be distortion from lighting. The resolution test results imply the object under observation should occupy the most pixels possible to get the most accurate motion.

3.2: Tripod Video Testing and Results

The tripod video tests focused on seeing if motions captured using a cellphone image sensor could be observed in the resulting video. To test the system a video was taken of a washing machine using a tripod and a Galaxy S8 cellphone in the slow-motion setting (240 fps), to ensure the sampled rate would

be above the Nyquist rate for the low frequency range of the washing machine's vibration (stimulated by rotation of agitator and drum). Using a tripod would ensure that any motions detected would only be due to the washing machine and not the sensor. After the video was taken, an application program (cellphone app) called *VibSensor* (by Now Instruments and Software, Inc) was used to determine the power spectral density (PSD) of vibration in the three orthogonal directions x, y, and z. based on data from the cellphone's own internal accelerometers. The frequencies that had the highest peaks in the PSD graph were used to determine the frequency bands that were going to be observed with the motion amplification code. The frequency bandwidths were chosen to be narrow to focus as close to the peak frequencies as possible. Once the frequency boundaries were decided, the video and the frequency bands were passed into the MATLAB function and the motion observed is compared to the expected motion in each frequency band.

In the first set of videos taken of a head-on view of a front-loading dryer during the drying cycle in an indoor garage with only artificial light as the light source (Figure 3.2.1) and the VibSenor was set to have the x, y and z directions relate to left to right, front and back, and up and down motions respectively. The 5 second dryer video was captured at a frame rate of 240 fps with a resolution of 720 by 1280 pixels.

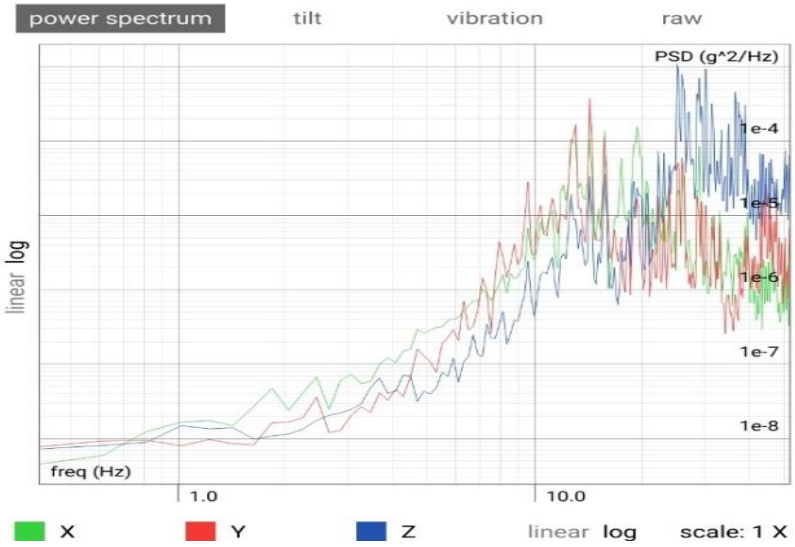
Figure 3.2.1: Dryer Video Frame

The VibSenor data (Figure 3.2.2) reports that there are resonances detected at multiple frequencies in all directions. The x direction had resonances at 19 Hz and 16 Hz, while the frequencies in the y direction occur at 14 and 13 Hz. The resonant frequencies in the z direction occur at 25Hz and 30Hz, which are significantly higher than the other two directions. The power spectrum plot illustrates how the power in the x and y directions decreases at higher frequencies, while the z direction increases.

Start: 30 Mar 2018 (10:20:54.304 pm)
Length: 0 min 29 sec
Points: 3,123 Gaps: none
Data rate: 104.3 Hz Units: g

Peak raw: X (0.10) Y (0.09) Z (1.19)
ISD: X (0.023) Y (0.022) Z (0.056)
Resonances:
X: 19 Hz (0.00016), 16 Hz (0.00014)
Y: 14 Hz (0.00038), 13 Hz (0.00017)
Z: 25 Hz (0.0011), 30 Hz (0.00093)

(a) Resonant Frequencies



(b) Power Spectrum

Figure 3.2.2: VibSensor data for Dryer Test

After looking at the VibSenor data, the frequency bands to test were chosen to be 10 to 19 Hz and 24 to 28 Hz, because these contained the resonant frequencies of all the orthogonal directions.

The motion amplification results however did not show clear movement results, in fact everything appeared to be moving. The movement is most prominent in the 10-19 Hz magnification video, and this could be due to a couple of things. The first being that the tripod was on a laundry basket which might not have been as sturdy and induced a small movement in the video. However,

looking at the original video there is not large-scale motion so this most likely not the cause. In the original video there is a large light flicker at the end which produces motion on the door of the dryer and when the video is slowed down to a quarter of the speed the change in lighting creates feint horizontal flickering lines in the frames. This is most likely the cause of the motion in the vertical direction. There also was a strange dark and light pattern that appeared in the video, which is assumed to be due to the lights since they flicker at 60 Hz, as shown in Figure 3.2.3.

Figure 3.2.3: Frame from magnified video at magnification level of 120

The light distortion overpowered any movement of the dyer, since both the dyer and the wall are white. However, the algorithm was able to enhance the movement of other objects that appeared on top of the dryer. The orange dryer sheet box provided the best source of movement. Figure 3.2.4 compares a frame

from each video taken at the same time stamp to see the effects of the 120-magnification factor in the 24 to 28Hz band.

Original

10 – 19 Hz Band after magnification

After magnification in the 24 to 28 Hz band

Figure 3.2.4: Difference in Dryer Sheet box at a magnification factor of 120

After magnification the floppy part of the lid is significantly more curved than the original and contains more light distortion. The difference shows that the box was moving up and down, which is expected in the 24 to 28 Hz range, thus the magnification filter is doing the job correctly. Since the program is effectively magnifying and producing the movement expected in the higher frequency

ranges, the reason the dryer did not appear to move is an artifact of the way the video was taken.

There are several reasons the motion enhancement did not work as expected in this example. The main issue is that the dryer did not have significant contrast from its surroundings. This is an issue because it would make it more difficult for the eye to see the motion. Another issue is that the dryer does not occupy a majority of the image field. As seen in the guitar example, when the resolution is too low distortions from other parts of the video, such as artificial lighting, prevent the observation of the desired magnified motion. Finally, the angle of the dryer is not ideal to capture motion in all three directions, since the x direction would be coming toward the camera which is hard to detect in the image, and the dryer surfaces are not perpendicular to the image edges.

For the next set of videos, the following changes were made to the video capture to improve the testing conditions:

1) A checker board table cloth was placed on top of a washer to provide high contrast edges to aid the detection of motion;
2) The camera angle was changed to view the top of the washer from an oblique angle so that motion in all three directions could be observed;
3) The camera was moved closer to the washer to fill more of the field of view.
4) More natural light was used to illuminate the objects to reduce the effect of artificial lighting flicker on the results.

The rest of the video capture methods are the same as the first video. The changes ensure there will be enough contrast between the structure we want to observe and the background. Having the large contrast allows the structure's movement to be seen in more clarity. The change in the angle of the camera will also make it easier to capture all the directions the washer can move.

The structure tested in the second testing setup was a washer from my apartment complex, which is different from the first structure. The washer used in the video was in the rinse cycle in an indoor environment with both artificial and natural light sources. The video captured used in testing is 3 seconds in length, has a resolution of 1920 by 1080, and a frame rate of 240 fps. To find the resonant frequencies the VibSenor app was again placed on top of the washer and the data collected is shown in Figure 3.2.5.

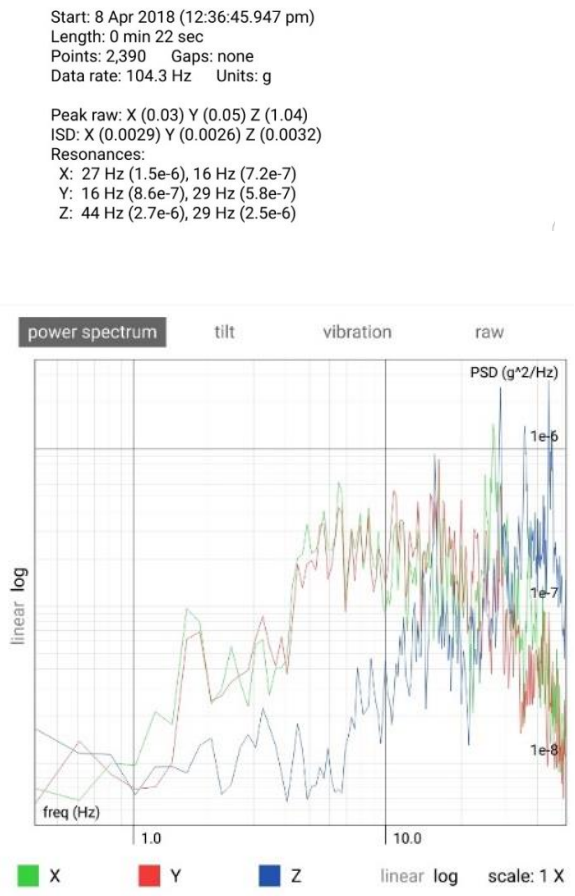


Figure 3.2.5: VibSensor data for washing machine

The VibSenor data reveals the resonant frequencies for the directions occur in similar bands, so the frequency bands generated will contain movement in the x, y, and z-direction. The power spectrum shows a trend like the dryer in the first test, in which the z separates from the x and y movement at higher frequencies. Using the VibSensor data the frequency bands tested were 15 to 17 Hz, 26 to 28Hz, and 43 to 46 Hz. The 15 to 17 Hz band will capture the low resonant frequencies for the x and y directions, while the movement in the z direction will be minimized. The 26 to 28 Hz band captures the high resonant frequency band for the x direction and larger movements in the y and z direction, so this band

should have movement in all directions. The 43 to 46 Hz band will capture the higher resonant frequency in the z direction, while movements in the x and y direction become small. So, in the highest band, movement in the z direction will be the most apparent.

Next the video captured was trimmed and sent into the magnification program. The reason for the trimming was to make the processing time shorter, and the only effect this will have is the output videos will be the same length as the trimmed video. To see the effect of the video in a 2D format, a row or column of pixels was selected and observed over the entire video. The columns and rows only cover a single transition from a black to white square on the checker pattern tablecloth. The row and column that is observed are shown as green and pink respectively in Figure 3.2.6.

Figure 3.2.6: Location of row and column observed over the course of the video

The first set of results focuses on the motion in the y direction (horizontal). To observe the y direction the same pixel row was captured from every frame and placed into a single image, with pixels from the first frame being

on the top of the image and pixels from the last frame on the bottom of the image. The results from the tripod test are shown in Figure 3.2.7.

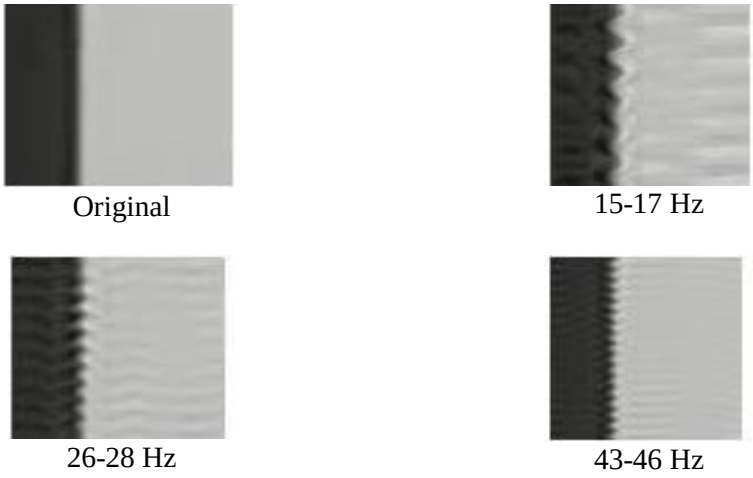


Figure 3.2.7: Behavior of a row of pixels at 100 magnification

The original video shows no oscillation in the row pixels throughout the entire video, however when magnified an oscillation appears. The amplitude of oscillation is the largest (9 pixels) in the 15 to 17 Hz band and the smallest amplitude oscillation (4 pixels) occurs in the 43-46 Hz band, which is expected. Also the frequency of oscillations were calculated using $\frac{(N\ cycles)(frame\ rate)}{Number\ of\ frames\ in\ N\ cycles}$ and yielded 15.8Hz, 27.8Hz, and 45 Hz for 15-17, 26-28, and 43-46 Hz bands respectively as expected. The VibSensor data in Figure 3.2.5 shows that the y axis has the largest magnitudes at lower frequencies and has a large drop in value as the 43-46Hz is reached, so the results fit the data VibSensor collected.

The movement in the x-direction (vertical) was observed next by tracking a single column of pixels throughout each video frame, starting from the far left.

Figure 3.2.8 shows the movement in the x-direction from the original video and the three magnified frequency bands.

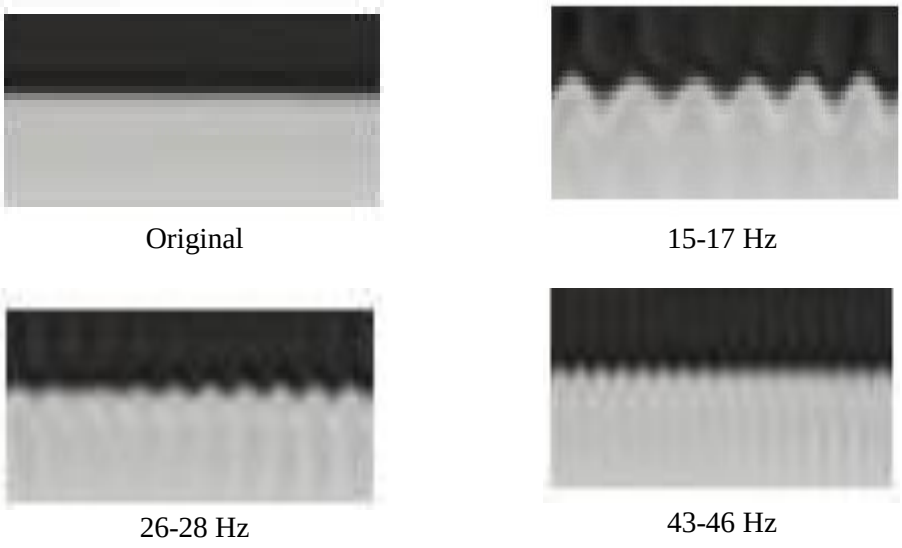


Figure 3.2.8: Motion in the x-direction at 100 magnification

Again, the magnified frequency bands show movement that was not present in the original video. With amplifications for the 15-17, 26-28, and 43-46 Hz bands being 8,4,2 respectively. Again, the magnitude of the oscillation decreases as the frequency increase which is expected from the VibSensor data. The frequencies calculated fell within their designated frequency range with 15.8, 27.3 and 43.6 Hz.

Next the effect of amplification was reviewed. For this test the same input video was used, but instead was only magnified by a factor of 50. The same row and column were observed in both the 100 and the 50-magnification factor for each of the frequency bands and are shown in Figure 3.2.9 and Figure 3.2.10 respectively.

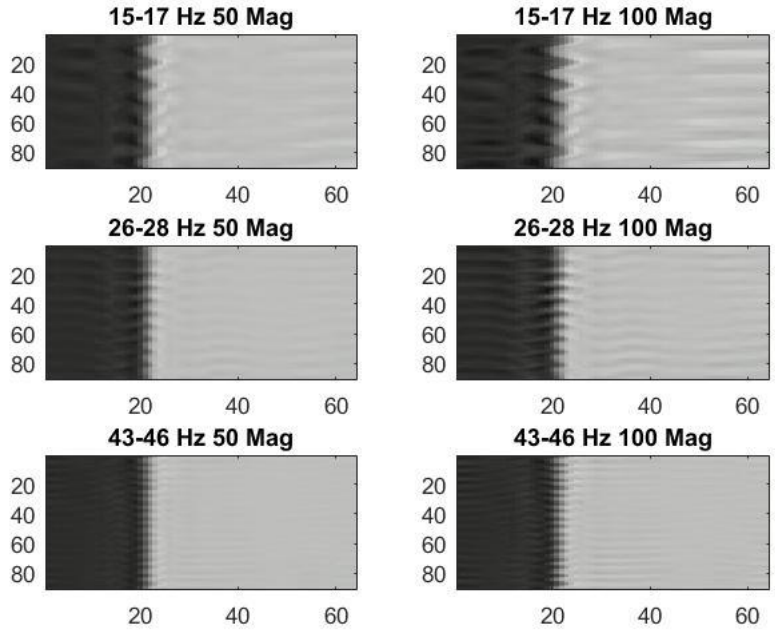


Figure 3.2.9: Comparison of y-direction motion with different magnification factors

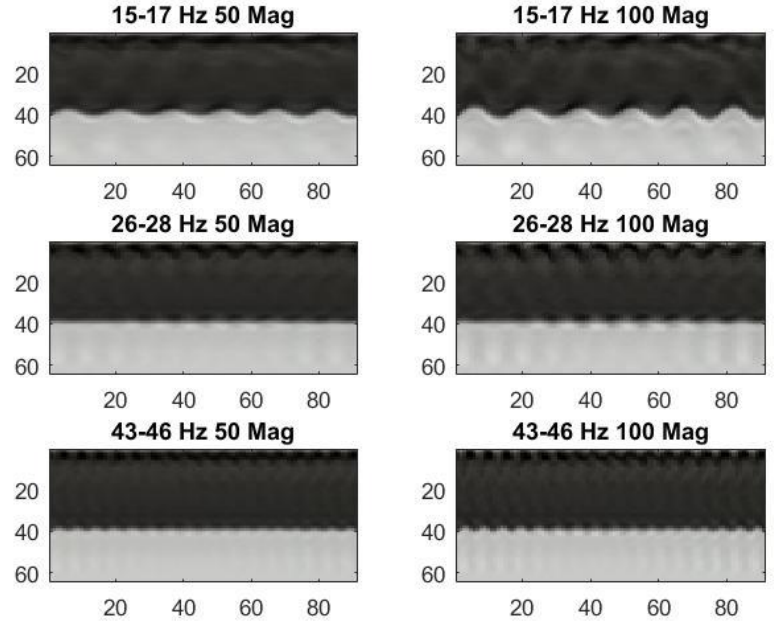


Figure 3.2.10: Comparison of x-direction motion with different magnification factors

The images show there is a difference in motion between the 50 and 100 magnification factors. In the y-direction the amplitudes for the 50 magnification factor starting from the lowest frequency band to the highest are 6, 4, and 3 pixels. These values are only about 50% lower than the 100 magnification factor (9, 6, and 4 pixels). The expected results were for the magnification to be double the size of the 50 magnification factor's amplitudes. The difference could be due to the amount of distortion present in the y-direction, which made it difficult to discern where the peaks occurred on the waveform. The amplitudes in the x direction were 4, 2, and 1 pixel with respect to 15-17, 26-28, and 43-46 Hz bands. These values are exactly half of the amplitudes that appear in the 100 magnification factor images (8, 4, and 2 pixels), which is expected. The peaks in these images were easier to discern than the y-direction because the distortion was not as pronounced on the waveform unlike in the y-direction video. The results from the amplification test show that increasing the magnification factor does increase the observed amplitude, but also distortion will limit the accuracy of the amplitude magnification.

Finally, the videos themselves were observed to see the all the movement captured. The video results show that there is movement which is not seen in the first video. The 15 to 17 Hz video shows the most movement, which occurs in the x and y directions. This behavior is expected because the movement in the x and y direction are significantly higher than the movement in the z direction. In the 26 to 28 Hz video the overall movement seen is less than the 15 to 17 Hz, but this is expected because the magnitude of the accelerations is less than the ones present

in the 15 to 17 Hz band for the x and y directions. In the 26 to 28 Hz video there is some up and down movement that can be seen but it is still not as apparent as the movement in the x and y direction. This is consistent because the z direction is just starting to be larger than the x and y acceleration magnitudes. Finally, in the 43 to 46 Hz frequency band video the movement is the smallest and appears as a vibration in the video itself. Due to the movement of the vibration it is difficult to distinguish the direction the washing machine is moving in, however in the final portion of the video there is an up and down movement, which is expected.

Overall, the second round of testing captured the motion of the structure the best. The checkered tablecloth and the camera angle changes made the motion easier to see. However, there is still the same light distortion, which was seen in the first video testing set. However, since the video in the second round of testing increased the contrast and was taken with the presence of natural light the movement was easier to see. The main problem with the second round of video taken was the dryers in the background were in operation, so their movement was also amplified in the video. The dryer movement in the background draws the eye away from the movement of the washer, which is what is under test; so to prevent this from happening in the future videos should be taken with a background that is stationary. Even with the moving background the MATLAB program was able to show movement that was in the realm of what was expected, so the vibration enhancement program appears to work properly on videos taken from a tripod.

3.3: Stabilization Testing

After verifying the MATLAB program amplifies vibrations in the tripod video in a meaningful way, the image stabilization methods for handheld cameras were then tested. The input to the stabilization method is a handheld cellphone camera video (240 fps frame rate and 1280 by 720 resolution for 21 seconds) of the structure we want to analyze. In this case the washing machine test setup from the second test was filmed again, because it gave the best results for the tripod video and the motion is easier to see. The view of the structure was changed from the original video because there was a wall that made it hard to film a handheld video. The video was captured using the same Samsung Galaxy 8 in the tripod tests with the same settings. Since the washing machine is the same one used in the previous tripod test the same VibSensor data will be used to magnify the videos.

Before the video can be processed through the magnification program, the handheld camera input video must go through an image stabilization process to remove as much of the extraneous overall movement as possible. The stabilization was attempted using each of the two methods previously described.

The first tests used MATLAB's video stabilization method that effectively moves the image frame boundaries in order to make the video appear stable. During the testing, however; the video output of this method was found to still contain a large percentage of the hand movement that was present in the original video without stabilization. Therefore, when magnified the movement of the washing machine was not easily distinguished from the overall movement of the

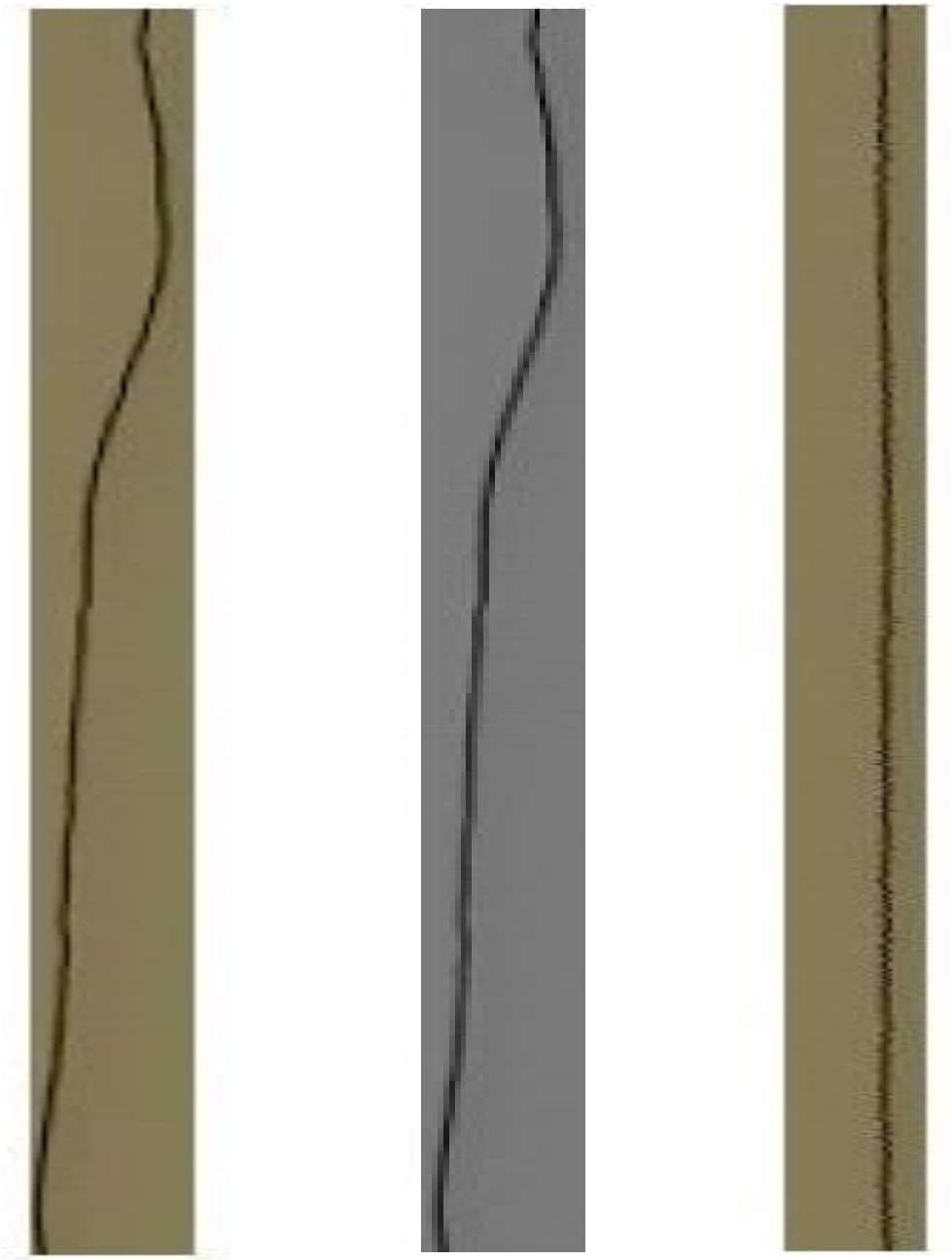
camera. When the motion in the stabilized video is compared to the original input, the amount of motion removed is minimal.

The second method of stabilization used image registration. The image registration method uses a single frame as a reference and then rotates and scales the subsequent images to try and match the original image. The resulting video using this method had no apparent movement in the horizontal and vertical directions, but variations in the chosen image scaling that is part of this correction creates a pulsing effect. When the image registration video was passed through the magnification algorithm the pulsing was magnified which overshadowed all the movement of the washing machine.

The motion stabilized videos for both stabilization method still show extraneous motion, but to directly compare the two methods, images depicting the x and y directional movement in the frame were created. The technique is the same as the one performed for the video captured using the tripod, and Figure 3.3.1 shows the locations where the row and column pixels are located in the frame.

Figure 3.3.1: Location of row (green) and column(magenta) used to observe stability

The area in blue rectangle was chosen to measure the stability because it is stationary through the entirety of the video, and so any motion seen in these features is undesired motion that we want the stabilization methods to remove. The motion in the y (horizontal) direction can be seen in Figure 3.3.2. Note again the top of each image represents the first frame and the bottom of each image represents the last frame.



Original Video Stabilization Image Registration

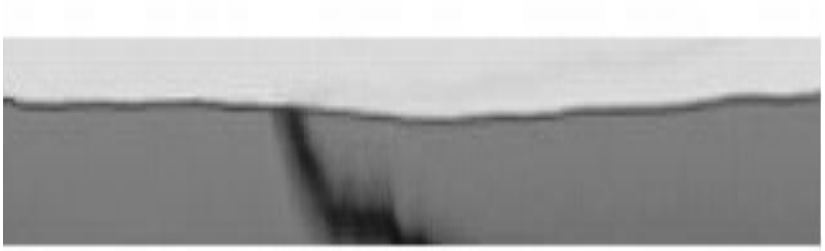
Figure 3.3.2: Movement in the y-direction (horizontal) comparison

The change between the original background motions and the video stabilization movements is very small, as expected. The main differences are the video stabilization image is in black and white and the behavior at the bottom of the motion image (later time in the video) is slightly less drastic. However, the image

registration method eliminated all the slow horizontal variations it found in the original video and only left a slight oscillation due to the scaling pulse behavior observed in the video.

The next step is to observe the movement in the x-direction (vertical) to see how each method behaves. Note in Figure 3.3.3 the left side of each image represents the first frame and the right side represent the last frame.

Original



Video Stabilization

Image Registration

Figure 3.3.3: X direction (vertical) movement comparison

Again, the image registration removes the major vertical movement in the original video but induces small amounts of ripple. The Video stabilization method does not perform as well as the image registration and exacerbates the movement and blurring of the dark curve in the original image.

At this stage image registration out performs the video stabilization method, but the residual motions in each stabilized video are still large enough to affect the vibration magnification method results. The residual movements would have the largest detrimental effect on the vibration magnification's highest frequency band (43-46Hz), because the motion in this band is the smallest and therefore hardest to detect.

To assess the impact of handheld camera motion on the vibration magnified videos, the handheld video was first processed with the magnification program without any image stabilization. Figure 3.3.4 shows the location of the rows and columns observed for motion in this test. Figure 3.3.5 shows the effects of magnification on the original, unstabilized video in the y direction, while Figure 3.3.6 shows the movement in the x direction.

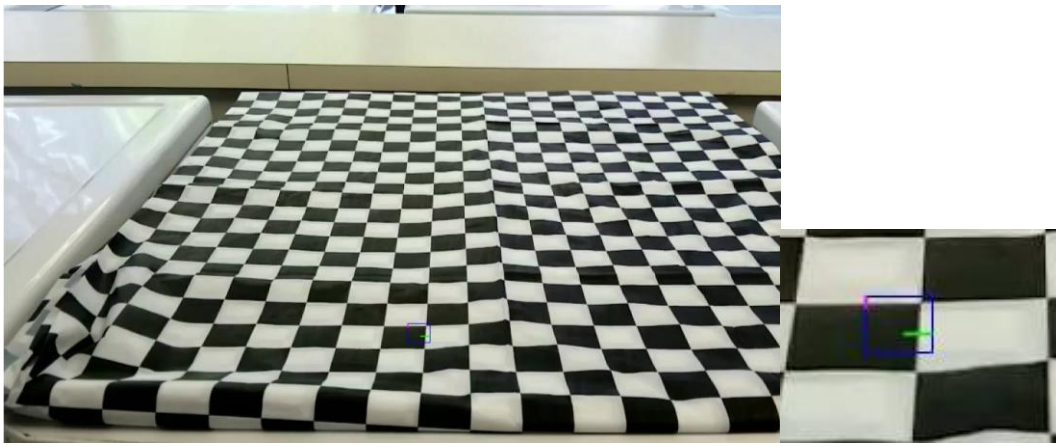


Figure 3.3.4: Row (green) and column (magenta) used to observe movement

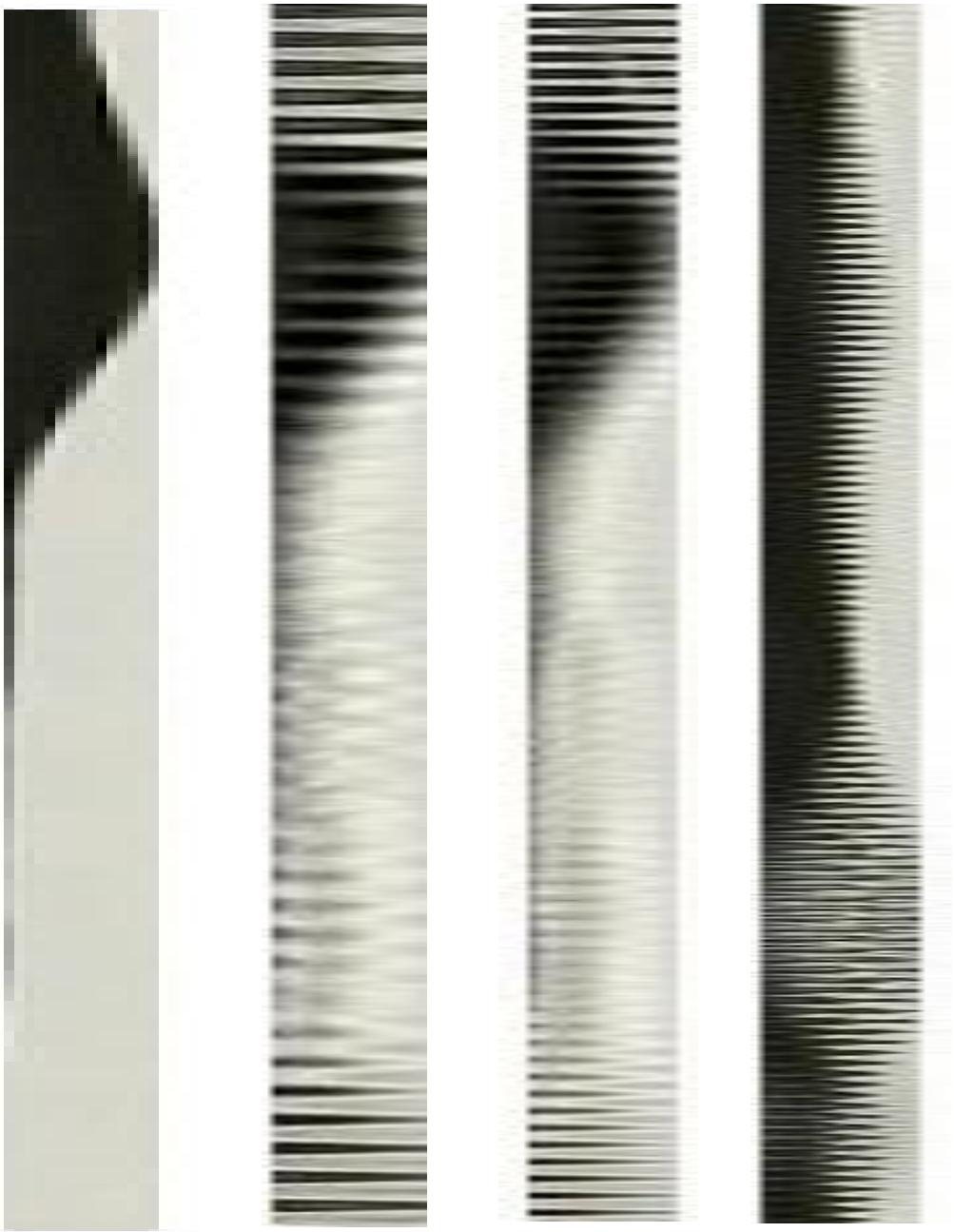


Figure 3.3.5: Effects of extraneous motion on behavior in y-direction

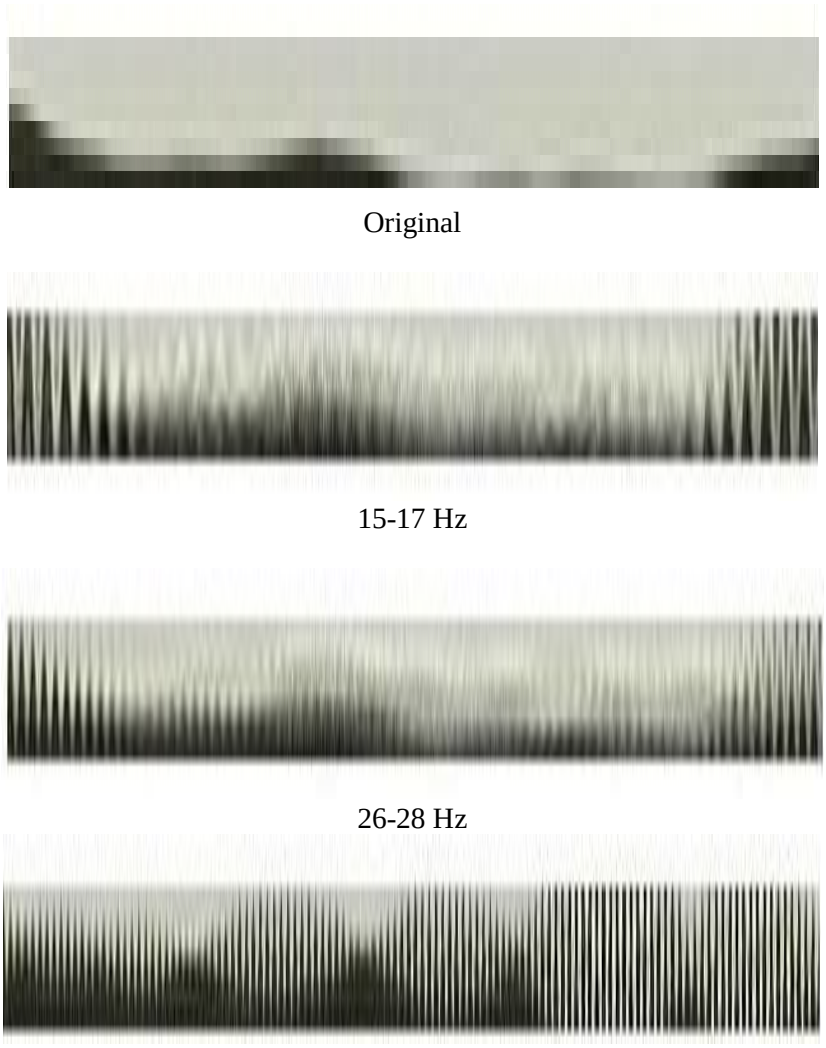


Figure 3.3.6: Effects of extraneous motion on behavior in x-direction

The motion shown in the various frequency bands seems to show motion that changes significantly over time. In both the x and y directions there is a sinusoidal waveform with a sinusoidal envelope. This behavior is not present in tripod video, so the sinusoidal envelope is induced from the hand motion. The envelope causes inaccurate measures of amplitude since the envelope does not necessarily reflect motion in a certain direction or of the washer. So, if the hand motion is not properly removed accurate information cannot be extracted form the data.

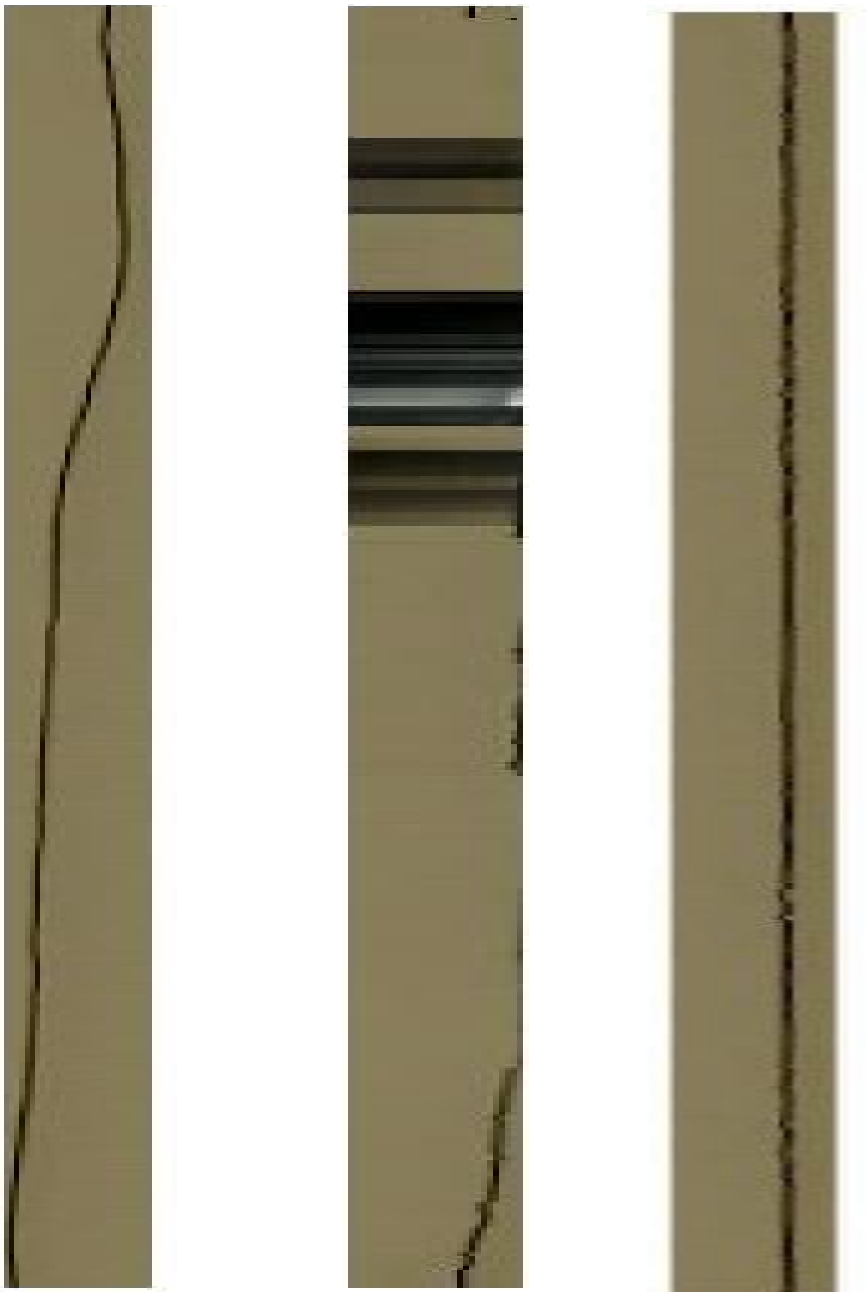
Since neither image stabilization method appeared to be suitable for improving these magnified vibration videos, an attempt to improve the stabilization methods using a mask was implemented. The purpose of using a mask is to only use truly stationary image features in the determination of the stabilization parameters, rather than including the parts of each image frame containing features whose motion we are trying to magnify. In this case the masking focused the image stabilization was on the divider between the two rows of washers since it is stable throughout the video. The only changes to the stabilization code required were to only search for matching image features in a certain area. However, this initially resulted in both the methods breaking down, due to the lack of well-defined feature corners for the stabilization algorithms to operate on. The methods were able to compute stabilizations once more image areas that included corner features were included, which unfortunately had to include part of the top of the vibrating washer. Figure 3.3.7 shows the masks (in blue) used for each stabilization method.

Video Stabilization Mask

Image Registration Mask

Figure 3.3.7: Masks used for each stabilization method (Blue Rectangles)

Once the masked were designed, and the unmagnified hand-held video was passed through each updated stabilization method, and the residual movement in the x and y directions were once again compared (Figure 3.3.8 and 3.3.9).



Original Video Stabilization Image Registration

Figure 3.3.8: Movement in the y direction after masking

Original

Video Stabilization

Image Registration

Figure 3.3.9: Movement in the x direction after masking

When comparing the stabilization results with and without masking, the masking did not improve the image registration method significantly. However, the video stabilization results actually degraded. While the video stabilization code was updated to work in color and not black and white, the output was choppier in the x direction than before. The choppiness causes the video to contain translational motion, which will certainly be amplified by the vibration magnification processing. The y direction also showed large dark bands that are not present in the original video, which suggests large movements and motion blurring occur at these points. When observing the stabilized video results, the image registration video still had scale pulsing like the previous, unmasked implementation; but the video stabilization method contained some segments where the video did not

move. However, the lack of movement did not last, and other exaggerated large movements occurred, which is not desired.

After running the both masked stabilization videos through the magnification filter the performance did not improve with masking for either method. The image registration method results contain the exact same distortions as in the original implementation. The video stabilization performed considerably worse in the vibration magnified video, once the masking was implemented. In the masked results, the 15-17 Hz frequency band is blurry, so no fine movements can be detected, and the large motions seen in the unmagnified results of the masking were amplified even more. Unfortunately, the masking did not improve performance of either method as expected.

Looking at the results of the stabilization methods there is no currently available effective way to handle a hand-held video in a way that will provide meaningful information with vibration magnification. The most promise seems to lie in image registration, but the image scaling would have to be minimized to extract any meaningful information once the hand-held videos are magnified. Until the pulsing problem can be fixed, the vibration magnification system can only handle videos from a stationary tripod that contain high image contrast. However, future work can be done to explore other techniques to minimize the effect of the global motion from a hand-held video.

CHAPTER 4: LIMITATIONS & CONCLUSIONS

4.1: Limitations

According to Nyquist Theorem to be able to recover a signal at a specified frequency the sampling rate needs to be at least twice the frequency. In practice, the signal is oversampled to ensure there is no loss of data. Since cameras have set sampling rates, called frame rates, the frequencies that they can observe are limited. For example, the Samsung Galaxy S8 has frame rates of 30, 60, and 120 fps which means it can observe vibration frequencies only up to 60 Hz [9]. To observe higher frequencies, a highspeed camera would need to be used.

Another limiting factor is the length of the video. To make obtaining the results relatively fast, the video should be at the minimum resolution to needed capture the structure clearly and less than 10 seconds in length. Larger videos can still be processed but will take longer and will require more memory.

The final limitation is that the video needs to be taken from a stable platform such as a tripod. The video from a handheld video currently contains too much motion, which cannot be removed, and masks the motion from the object under test.

4.2: Conclusion

The reason for developing the MATLAB program was to try and make a tool to take in a video from a cell phone and magnify the movements in them. The program successfully implemented the system and allowed for the user to specify the frequency bands and the amount of magnification, which is useful for structural analysis testing, especially vibration testing. The magnification at

specific frequency bands gave expected responses when taken from a stable platform such as a tripod. However, the stabilization methods tested did not remove enough of the image sensor motion from the video captured from the hand-held camera. Since there were still large motion artifacts present when the video was magnified all the small motions of the washing machine were masked by the magnified motions of the camera. Even though the stabilization method does not eliminate enough of the motion, the MATLAB program will still work if the input is from a stable platform such as a tripod.

To make the motion of the structure easier to see in the frequency band videos, the video should have a large amount of contrast and the structure should take up most of the image field. The contrast allows the eye to easily see the motion and if the structure is the focus of the video the eye will naturally be drawn to structure movement. Another consideration would be to make sure all the objects in the background are stationary or are not operating in the frequency band were vibration behavior is being observed. Having a stationary background will also aid in making the structure's movement easier to see and if additional surrounding equipment operates in the frequency band which is being observed, their behavior will also be present in the video. A clear example of this was seen during the second test of the washing machine. The background was comprised of dryers that were operating at the same time. Based off of data taken from the dryer in the first round of testing, dryers would have similar resonant vibration frequencies to the washers. Therefore, when the second-round test videos were processed for motion enhancement, both the motion of the washing machine and

the dryers in the background moved in similar amounts. The background dryer motion distracts from the motion of the top of the washer, making it more difficult to observe the vibration behavior of the washer.

Overall the MATLAB program designed in this thesis is a good starting place for using computer vision in structural analysis testing. The program provides a way to see the motion going on a certain frequency band in a way that is not currently utilized. The program was designed to be as general as possible, so it can be utilized in any type of structural testing and take in data from any type of camera. The limiting factors are the requirement for limited motion in the testing setup and video capture method, computing time, and frame rate of the camera. As time progresses and technology improves this technique might be revisited and incorporated in structural testing or in troubleshooting without using an expensive test setup.

CHAPTER 5: FUTURE WORK

The next logical step for future work in developing a useful system for structural analysis is to input a video from a structural test. One way to do this would be to design a structure with a resonate frequency below 120 Hz and film the structure during a vibration test. The vibration test will test how the current system handles large motions of the structure. Depending on the results of this test other algorithms and programs in the MIT code can be explored. Specifically, there is a code that can attenuate large motions in a video, while magnifying the small ones. This function could prove useful if both the larger motions and the smaller motions want to be seen in the same video.

Another route to be explored is utilizing Riesz pyramids instead of complex steerable pyramids. MIT explored this route in 2014 and just published a pseudo code that describes how to implement this pyramid structure in MATLAB. The reason for the switch to these pyramids would be to decrease the amount of time required to process the video file. The reason for this is because the complex steerable pyramids are more over complete than the Riesz pyramids [14]. Having a more over complete pyramid equates to having more data to process, thus having longer processing time on videos. The results from MIT show that for cases where the motion is confined to one orientation Riesz pyramid motion enhancement can be performed in real time. However, this might not be desired since the behavior of a structural motion is not necessarily constrained to a single orientation, especially at different locations on the structure.

Also, there are new apps being developed that allow the user to increase the frame rate of their cellphone camera. The effects of using an application would allow for a phone camera to increase the range of frequencies that can be observed in the videos captured. However, the effects of these apps have not been observed. To see the effects a test like the one done by MIT to prove the motion captured using the motion amplification method provided similar acceleration values to an accelerometer attached to the metal structure. Another parameter to monitor is if there are any additional distortions such as blur added to the video captured from the camera while using the app.

Finally, new stabilization methods could be implemented to try and remove the hand motion. The image registration showed some promise, so if there was a way to include translation parameters into the transform matrix, that could potentially improve the method's performance. Masking could also be incorporated in different ways or using specific features in a testing environment could be used instead of using the corner detecting algorithm. Also developing a way to better account for the motion blur in a rolling shutter might become necessary if there is a large amount of motion in the structure while it undergoes testing.

WORKS CITED

[1] J. P. Hughes, “MECHANICAL RESONANCE.” [Online]. Available: http://www.physics.rutgers.edu/~jackph/2005s/PS02.pdf. [Accessed: 25-May-2018].

[2] O. Buyukozturk, J. G. Chen, N. Wadhwa, A. Davis, F. Durand, and W. T. Freeman, “Smaller Than the Eye Can See: Vibration Analysis with Video Cameras.”

[3] K. G. McConnell, *Vibration testing : theory and practice*. Wiley, 1995.

[4] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman, “Phase-based video motion processing,” *ACM Trans. Graph.*, vol. 32, no. 4, p. 1, 2013.

[5] “Specifications | Samsung Galaxy S8 and S8+ – The Official Samsung Galaxy Site.” [Online]. Available: http://www.samsung.com/global/galaxy/galaxy-s8/specs/. [Accessed: 25-May-2018].

[6] R. Szeliski, *Computer Vision : Algorithms and Applications*, vol. 5. 2010.

[7] W. Hong, D. Wei, and A. U. Batur, “Video Stabilization and Rolling Shutter Distortion Reduction,” in *2010 IEEE 17th International Conference on Image Processing*, 2010, pp. 3501–3504.

[8] “Video Stabilization Using Point Feature Matching - MATLAB & Simulink.” [Online]. Available: https://www.mathworks.com/help/vision/examples/video-stabilization-using-point-feature-matching.html. [Accessed: 18-May-2018].

[9] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. II, pp. 1508–1515, 2005.

[10] A. Alahi, R. Ortiz, and P. Vandergheynst, “FREAK: Fast Retina Keypoint.”

[11] W. Hoff, “Computer Vision,” 2005. [Online]. Available: http://inside.mines.edu/~whoff/. [Accessed: 25-May-2018].

[12] “Find Image Rotation and Scale Using Automated Feature Matching.” [Online]. Available: https://www.mathworks.com/examples/image/mw/images-ex77496944-find-image-rotation-and-scale-using-automated-feature-matching#9. [Accessed: 18-May-2018].

[13] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006.

[14] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman, “Riesz Pyramids for Fast Phase-Based Video Magnification.”

APPENDIX A: MATLAB CODE

Main Execution file:

```
%Thesis Main Run
%% Intialization (from MIT setPath)
% Adds directories to MATLAB path

% Paths for the linear method
addpath(fullfile(pwd, 'Linear'));
addpath(fullfile(pwd, 'Util'));
addpath(fullfile(pwd, 'matlabPyrTools'));
addpath(fullfile(pwd, 'matlabPyrTools', 'MEX'));

% Paths for the phase-based method
addpath(fullfile(pwd, 'PhaseBased'));
addpath(fullfile(pwd, 'pyrToolsExt'));
addpath(fullfile(pwd, 'Filters'));

clear;

dataDir = './data';

resultsDir = 'ThesisResults/';
mkdir(resultsDir);
defaultPyrType = 'halfOctave'; % Half octave pyramid is default as discussed in
paper
scaleAndClipLargeVideos = true; % With this enabled, approximately 4GB of
memory is used

% Uncomment to use octave bandwidth pyramid: speeds up processing,
% but will produce slightly different results
defaultPyrType = 'octave';

% Uncomment to process full video sequences (uses about 16GB of memory)
% scaleAndClipLargeVideos = false;
%% Stabilization Filter using video stabilization method
%Use this section if the input video is not on a tripod
inFile = fullfile(dataDir, 'Test2shaky.mp4');
stable = stabl_vid(inFile);

%% STABILIZATION USING IMAGE REGISTRATION
inFile = fullfile(dataDir, 'Test2shaky.mp4');
stable = img_reg(inFile);
%% Guitar Test File
inFile = fullfile(dataDir, 'guitar.avi');
```

```
samplingRate = 600;
loCutoff = 72;
hiCutoff = 92;
alpha = 25; %how much you magnify by
sigma = 2; %amount of bluring
pyrType = defaultPyrType;
fl = [105,140];%[50,72,93,105,140];
fh =  [ 115,150]; %[70,92,101,115,150];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
% phaseAmplify(inFile, alpha, loCutoff, hiCutoff, samplingRate,
resultsDir,'sigma', sigma, 'pyrType', pyrType);
%% Dryer
inFile = fullfile(dataDir, 'still_120fpsTrim.mp4');
samplingRate = 240;
alpha = 120; %how much you magnify by
sigma = 2;
pyrType = defaultPyrType;
fl = [10 24];
fh = [19 28];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
%% Washing Machine
inFile = fullfile(dataDir, 'still_120T2Trim.mp4');
samplingRate = 240;
alpha = 100; %how much you magnify by
sigma = 2;
pyrType = defaultPyrType;
fl = [15 26 43];
fh = [17 28 46];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
%% Handheld Video without Stabilization
inFile = fullfile(dataDir, 'Test2shaky.mp4');
samplingRate = 240;
alpha = 75; %how much you magnify by
sigma = 2; %amount of blurring
pyrType = defaultPyrType;
fl = [15 26 43];
fh = [17 28 46];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
%% Handheld Video after stabilization
%Testing Result
inFile = fullfile(dataDir, 'Test2shaky.mp4_imgreg.avi'); %Put file name of
stabilized video as the inFile
```

```
samplingRate = 240;
alpha = 75; %how much you magnify by
sigma = 2; %ammount of blurring
pyrType = defaultPyrType;
fl = [15 26 43];
fh = [17 28 46];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
%% Masking Stabilization Results
%Video Stabilization Result
inFile = fullfile(dataDir, 'Test2shaky.mp4_Masked_stable.avi'); %Put file name of
stabilized video as the inFile
samplingRate = 240;
alpha = 75; %how much you magnify by
sigma = 2; %ammount of blurring
pyrType = defaultPyrType;
fl = [15 26 43];
fh = [17 28 46];
freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
% %Image Registration Result
% inFile = fullfile(dataDir, 'Test2shaky.mp4_imgreg_updated.avi'); %Put file
name of stabilized video as the inFile
% samplingRate = 240;
% alpha = 75; %how much you magnify by
% sigma = 2; %ammount of blurring
% pyrType = defaultPyrType;
% fl = [15 26 43];
% fh = [17 28 46];
% freqbandamp(inFile, alpha , fl, fh,samplingRate, resultsDir,'sigma', sigma,
'pyrType', pyrType);
```

freqbandamp function:

```
function outName=freqbandamp(vidFile, magPhase , fl, fh,fs, outDir, varargin)
for i = 1:length(fl)
fprintf('PROCESSING FREQUENCY BAND %2d\n',i);
outName = phaseAmplify(vidFile, magPhase , fl(i), fh(i),fs, outDir);
end
fprintf('Operation Complete\n');
end
```

Video Stabilization Code (no color):

```
%created from MATLAB website and modified using research...
function out_vid=stabl_vid(vid_file)
%Load and create video file
hVideoSrc = vision.VideoFileReader(vid_file, 'ImageColorSpace', 'Intensity');
out_vid=VideoWriter(sprintf('%s_stable.avi',vid_file),'Uncompressed AVI');
open(out_vid);

% Reset the video source to the beginning of the file.
reset(hVideoSrc);

hVPlayer = vision.VideoPlayer; % Create video viewer

% Process all frames in the video
movMean = step(hVideoSrc);
imgB = movMean;
imgBp = imgB;
correctedMean = imgBp;
ii = 2;
Hcumulative = eye(3);
while ~isDone(hVideoSrc)
    % Read in new frame
    imgA = imgB; % z^-1
    imgAp = imgBp; % z^-1
    imgB = step(hVideoSrc);
    movMean = movMean + imgB;

    % Estimate transform from frame A to frame B, and fit as an s-R-t
    H = cvexEstStabilizationTform(imgA,imgB);
    HsRt = cvexTformToSRT(H);
    Hcumulative = HsRt * Hcumulative;
    imgBp =
imwarp(imgB,affine2d(Hcumulative),'OutputView',imref2d(size(imgB)));
    writeVideo(out_vid,(imgBp));
    % Display as color composite with last corrected frame
    step(hVPlayer, imfuse(imgAp,imgBp,'ColorChannels','red-cyan'));
    correctedMean = correctedMean + imgBp;

    ii = ii+1;
end
correctedMean = correctedMean/(ii-2);
movMean = movMean/(ii-2);

% Here you call the release method on the objects to close any open files
% and release memory.
```

```
 close(out_vid)
release(hVideoSrc);
release(hVPlayer);

end
```

video stabilization code (color version)

```
%created from MATLAB website and modified using research...
function out_vid=stabl_vid(vid_file)
%Load and create video file
% hVideoSrc = vision.VideoFileReader(vid_file, 'ImageColorSpace', 'Intensity');
v =VideoReader(vid_file);

out_vid=VideoWriter(sprintf('%s_stable_updated.avi',vid_file),'Uncompressed
AVI');
open(out_vid);

% Reset the video source to the beginning of the file.
% reset(hVideoSrc);
%
% hVPlayer = vision.VideoPlayer; % Create video viewer

% Process all frames in the video
movMean = readFrame(v);
imgB = movMean;
imgBp = imgB;
correctedMean = imgBp;
ii = 2;
Hcumulative = eye(3);
 while hasFrame(v) %~isDone(hVideoSrc)
    % Read in new frame
    imgA = imgB; % z^-1
    imgAp = imgBp; % z^-1
    imgB = readFrame(v);
    movMean = movMean + imgB;


    % Estimate transform from frame A to frame B, and fit as an s-R-t
    H = cvexEstStabilizationTform(imgA(:,:,1),imgB(:,:,1));
    HsRt = cvexTformToSRT(H);
    Hcumulative = HsRt * Hcumulative;
    imgBp =
imwarp(imgB,affine2d(Hcumulative),'OutputView',imref2d(size(imgB)));
     writeVideo(out_vid,(imgBp));
    % Display as color composite with last corrected frame
```

```
%     step(hVPlayer, imfuse(imgAp,imgBp,'ColorChannels','red-cyan'));
    correctedMean = correctedMean + imgBp;

    ii = ii+1;
end
correctedMean = correctedMean/(ii-2);
movMean = movMean/(ii-2);

% Here you call the release method on the objects to close any open files
% and release memory.
 close(out_vid)



end
```

Video Stabilization with Mask:

```
%created from MATLAB website and modified using research...
function out_vid=stabl_vid_mask(vid_file)
%Load and create video file
v =VideoReader(vid_file);
out_vid=VideoWriter(sprintf('%s_Masked_stable.avi',vid_file),'Uncompressed
AVI');
open(out_vid);
% Reset the video source to the beginning of the file.
% Process all frames in the video
movMean = readFrame(v);
imgB = movMean;
imgBp = imgB;
correctedMean = imgBp;
ii = 2;
Hcumulative = eye(3);
 while hasFrame(v)
    % Read in new frame
    imgA = imgB; % z^-1
    imgAp = imgBp; % z^-1
    imgB = readFrame(v);
    movMean = movMean + imgB;
    %Mask
    x=zeros(size(imgA(:,:,1)));
    y=x;
    x(1:150, 1:720)=imgA(1:150,1:720,1)/255;
    y(1:150, 1:720)=imgB(1:150,1:720,1)/255;
    % Estimate transform from frame A to frame B, and fit as an s-R-t
    H = cvexEstStabilizationTform(x,y);
    HsRt = cvexTformToSRT(H);
    Hcumulative = HsRt * Hcumulative;
    imgBp =
imwarp(imgB,affine2d(Hcumulative),'OutputView',imref2d(size(imgB)));
     writeVideo(out_vid,(imgBp));
    % Display as color composite with last corrected frame
    correctedMean = correctedMean + imgBp;
    ii = ii+1;
end
correctedMean = correctedMean/(ii-2);
movMean = movMean/(ii-2);

% Here you call the release method on the objects to close any open files
% and release memory.
 close(out_vid)
end
```

Image Registration Code:

```
function stable_vid = img_reg(vid_file,setting)
%Load and create video file
v=VideoReader(vid_file);
v.FrameRate
stable_vid=VideoWriter(sprintf('%s_imgreg_updated',vid_file),'Uncompressed
AVI');

open(stable_vid);

%Stablization Method
%Get First Frame
frameA = readFrame(v);
ind_count=1;
[m n r] = size(frameA);
% frame_color = zeros(m,n,3);

while hasFrame(v)
  fprintf('PROCESSING Frame %2d\n',ind_count);

%Get Second Frame
frameB = readFrame(v);

%Get Color layers
if setting == 1
red = imgreg_layer(frameA,frameB,1);
else
   red = imgreg_layer_mask(frameA,frameB,1,[1 155],[1 1000]);
end

% green = imgreg_layer(frameA,frameB,2);
% blue = imgreg_layer(frameA,frameB,3);
%  %combine
%  frame_color(:,:,1)=red(:,:);
%  frame_color(:,:,2)=green(:,:);
%  frame_color(:,:,3)=blue(:,:);

writeVideo(stable_vid,red);
%   frameA=red;
  ind_count=ind_count+1;
 end

close(stable_vid);
end
```

imgreg_layer:

```
function color_layer = imgreg_layer(frameA, frameB,layer)
%Find Matching Features
ptsframeA  = detectFASTFeatures(frameA(:,:,layer));
[featuresframeA,   validPtsframeA]  = extractFeatures(frameA(:,:,layer),
ptsframeA);
ptsframeB  = detectFASTFeatures(frameB(:,:,layer));
[featuresframeB,   validPtsframeB]  = extractFeatures(frameB(:,:,layer),
ptsframeB);

indexPairs = matchFeatures(featuresframeA, featuresframeB);
matchedframeA  = validPtsframeA(indexPairs(:,1));
matchedframeB = validPtsframeB(indexPairs(:,2));
%Estimate Transformation
[tform, inlierframeB, inlierframeA] = estimateGeometricTransform(...
   matchedframeB, matchedframeA, 'similarity');
%Solve for Scale and Angle
Tinv  = tform.invert.T;

ss = Tinv(2,1);
sc = Tinv(1,1);
scale_recovered = sqrt(ss*ss + sc*sc);
theta_recovered = atan2(ss,sc)*180/pi;
%Recover Image
outputView = imref2d(size(frameA));
color_layer = imwarp(frameB,tform,'OutputView',outputView);
end
```

imreg_layer_mask:

```
function color_layer = imgreg_layer_mask(frameA, frameB,layer,rows,col)
%Find Matching Features given mask
ptsframeA  = detectFASTFeatures(frameA(rows(1):rows(2),col(1):col(2),layer));
[featuresframeA,   validPtsframeA]  =
extractFeatures(frameA(rows(1):rows(2),col(1):col(2),layer),  ptsframeA);
ptsframeB  = detectFASTFeatures(frameB(rows(1):rows(2),col(1):col(2),layer));
[featuresframeB,   validPtsframeB]  =
extractFeatures(frameB(rows(1):rows(2),col(1):col(2),layer),  ptsframeB);

indexPairs = matchFeatures(featuresframeA, featuresframeB);
matchedframeA  = validPtsframeA(indexPairs(:,1));
matchedframeB = validPtsframeB(indexPairs(:,2));
%Estimate Transformation
[tform, inlierframeB, inlierframeA] = estimateGeometricTransform(...
   matchedframeB, matchedframeA, 'similarity');
%Solve for Scale and Angle
Tinv  = tform.invert.T;

ss = Tinv(2,1);
sc = Tinv(1,1);
scale_recovered = sqrt(ss*ss + sc*sc);
theta_recovered = atan2(ss,sc)*180/pi;
%Recover Image
outputView = imref2d(size(frameA));
color_layer = imwarp(frameB,tform,'OutputView',outputView);

end
```

APPENDIX B: External Links

Link to result videos:

https://drive.google.com/open?id=1hwQjJJ3dpnXUW7I15u5hTMoQHyFzV58n

Link to the MIT phase-based magnification code:

http://people.csail.mit.edu/nwadhwa/phase-video/