Winter 12-20-2017

Application of PCA to Cardiac Optical Mapping

Louis Woodhams
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Mechanical Engineering and Materials Science

Thesis Examination Committee:

Guy Genin, Chair

David Peters

Robert Pless

APPLICATION OF PRINCIPAL COMPONENT ANALYSIS TO CARDIAC OPTICAL MAPPING

by

Louis Woodhams

A thesis presented to

The School of Engineering and Applied Science

of Washington University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

December 2017

Saint Louis, Missouri

Table of Contents

List of Figures

Acknowledgments

I would like to thank everyone who has offered their help and support as I wrote this thesis. First, I thank Professor Elson for the opportunity to work in his lab and for the tremendous amount of time and support he has given me. I also thank all the members of the Elson lab over the past few years who have been endlessly patient with all my questions about biochemistry and microscopy, especially Dr. Tony Pryse. I would like to thank Professor Genin for always having time to listen, offer ideas, and provide guidance. I thank Professor Pless for inspiring this current work and for being willing to offer his time even after he has moved on to another institution. I thank Professor Peters and Professor Bayly for their help in navigating academia and figuring out how to make the pieces fit.

Finally, I would like to thank my Mom and Dad, two incredibly smart and accomplished musicians for whom this entire work will seem like a bunch of gibberish.

Louis Woodhams

Washington University in St. Louis

December 2017

ABSTRACT OF THE THESIS

Application of principal component analysis to

cardiac optical mapping

by

Louis Woodhams

Master of Science in Mechanical Engineering

Washington University in St. Louis, 2017

Research Advisor: Professor Guy Genin

Structural remodeling of the heart due to pathologies such as hypertension and myocardial infarction leads to the appearance of myofibroblasts, a phenotype largely absent in physiologic myocardium. While myofibroblasts are responsible for wound healing and structural repair of damaged myocardium, they are thought to have deleterious effects on electrical and mechanical properties of the heart. Understanding these effects is critical to developing effective treatments, and has motivated the development of a series of *in vitro* engineered heart tissues and cardiomyocyte-myofibroblast co-cultures whose mechanical and electrophysiological function can be deduced from video analysis. Electrophysiological properties are evident from changes in intensity of a fluorescent calcium assay, mechanical properties are evident from deformations apparent in the video, and both are used to study excitation-contraction coupling properties. This thesis contributes efficient mathematical tools for denoising and analyzing videos of contracting, vibrating, and flashing structures.

Chapter 1: Introduction

1.1 The Cardiac Action Potential

In a properly functioning human heart, the cardiac action potential (AP) impulse originates in the sinoatrial node (SA node) and propagates through the left and right atrial myocardium as it travels to the atrioventricular node (AV node). The signal is then delayed by around 100ms as it passes through the AV node, to allow for contraction of the atria. After the delay, the action potential propagates along the Purkinje fibers through the bundle of His in the inter-ventricular septum and out to the ventricular myocardium. From here, the signal propagates through the ventricular myocardium, causing the ventricles to contract.

Propagation of the AP through the myocardium is primarily a cardiomyocyte (CM) to cardiomyocyte process involving the spread of ion currents through protein connections called gap junctions [1]. As an individual CM undergoes an AP sequence, positively charged sodium ions (Na^+) rapidly flood into the cell bringing the transmembrane electrical potential in the cell from around -80mV to a brief positive spike. This activates other ion channels in the cell, which cause an efflux of Potassium ions (K^+), neutralizing the transmembrane potential, and an influx of Calcium ions (Ca^{2+}), which further release Ca^{2+} ions sequestered in the sarcoplasmic reticulum to initiate contraction of the cell. The rapid increase in positive ion concentrations within the cytoplasm causes positive ions to diffuse to other CMs through gap junctions. The diffusion of positive ions into adjacent cells causes the membrane potentials within those cells to rise to a threshold voltage which initiates depolarizations in those cells and the cycle continues. After a cell has depolarized, it must repolarize to its resting membrane potential and there is a

refractory period during which the fast Na^+ channels are inactivated, and a cell cannot initiate an AP sequence. [2] [3]

The myocardial AP propagation described above is possible because individual CMs "buffer" the signal by depolarizing and flooding with positive ions as they are activated. Fibroblasts (FBs), which are responsible for maintaining the extracellular matrix (ECM), are the most numerous cell in the myocardium [4] [5]. FBs do not have APs, and it is unclear whether they form gap junctions with CMs under physiological conditions [6]. However, under pathological conditions, *fibroblasts* (FBs) within the myocardium may undergo a phenotypic change to *myofibroblasts* (mFBs), also known as *activated fibroblasts*, which are known to form heterocellular gap junctions with CMs *in vitro*, though whether or not such heterocellular connections are formed *in vivo* is still a matter of active research [6] [4]. MFBs do not have action potentials and therefore cannot buffer the cardiac AP. This creates several potential signal propagation issues.

Firstly, signal propagation through mFBs is delayed as ions must diffuse through a passive cell [1]. This may cause the AP wave front to travel more slowly and meander through regions with high concentrations of mFBs (as may occur with fibrosis), or it may lead to signal blockage in areas where there are clumps of mFBs (such is the aftermath of an infarction). While slowing of the signal is undesirable, heterocellular coupling may allow the impulse to travel through regions of scar tissue where the signal might otherwise be blocked.

Secondly, if mFBs act as passive ion reservoirs, they may contribute to source-sink mismatches and create unidirectional conductance blocks. When CMs depolarize, there is an ion efflux through gap junctions into adjacent cells. If all cells are CMs, and the cell density is sufficient, then the ions entering quiescent cells are enough to depolarize them and the buffering of the

signal means that there are never too few ions leaving the depolarized cells to trigger the next quiescent cells. However, if some of the cells are passive (do not experience APs), or the densities become low, there can exist situations in which the ion current is not great enough to trigger the next group of CMs. When the signal travels from a sparse group of cells surrounded by mFBs (and the accompanying excessive ECM) to a larger group of CMs, the ions exciting the first group may not be sufficient to excite the second group (however, the signal may pass in the opposite direction). This is known as unidirectional conduction block.

Perhaps the greatest danger related to the above two properties of AP propagation in mFBs is their contribution to arrhythmogenesis [5] [2]. To function properly, the heart must contract in the orderly fashion described previously. The impulse typically originates in the SA node because cells in this node have automaticity. Cells in the AV node and Purkinje fibers also have automaticity, but typically do not initiate cardiac impulses due to their slower firing rate (the SA node usually fires first). Problems with AP propagation can lead to ectopic impulse initiation and re-entrant arrhythmia. Re-entrant arrhythmia may be due to complex interactions between CMs and mFBs where a signal that has been delayed by slow mFB conduction re-emerges into post-refractory myocardium and initiates another depolarization sequence (see [2] for a graphical description).

In vitro, immature induced-pluripotent-stem-cell (IPSC) derived CMs demonstrate automaticity, though it a matter of ongoing investigation as to how closely CM-mFB interactions and behaviors *in vitro* model those *in vivo*.

We wish to investigate the effect of varying concentrations of mFBs on AP propagation, and to map on the microscopic scale how the cardiac impulse travels within different configurations of

cells. It has been suggested that there may be more to AP propagation than simply gap-junctional coupling, and we hope through microscopic optical mapping of APs in different cell configurations to gain insight and understanding this phenomenon. [1]

1.2 Past Work

This work studies images of beating cell monolayers and multilayers. The acquisition of images was not a focus of this thesis, and we therefore list methods used to acquire these images in this introductory section. To study AP propagation in CMs and mFBs, we have created monolayer and multilayer CM and mFB co-cultures with varying concentrations of mFBs. These cultures are typically stained with a fluorescent reporter such as Fluo-4 (which indicates intracellular Ca^{2+} concentrations) so that we may optically map AP propagation in CMs.

The cells are typically self-paced, although our group has performed experiments with external pacing in the past and may pursue this again.

Images were acquired using a pco.edge sCMOS 5MP camera capable of recording images at 100FPS at full resolution and up to 1000 FPS using a reduced imaging area. All images were recorded directly to uncompressed .tif format. A custom 7-LED lamp (Figure 1) with 470nm LEDs from luxeonstar.com and a 12° beam optic array were interfaced with the Zeiss Confocor Microscope for fluorescence excitation.

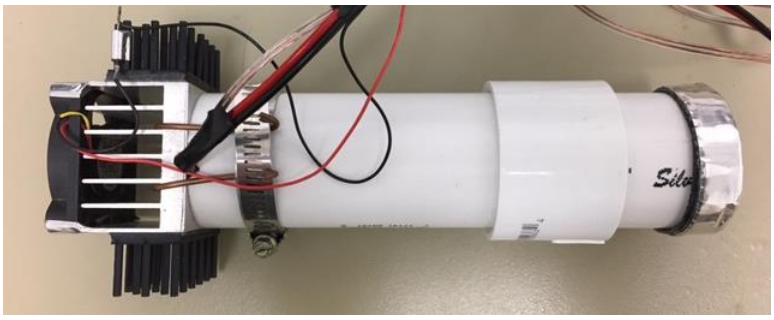

Figure 1 – Custom LED fluorescence excitation lamp for Zeiss Microscope.

Videos were obtained of various concentrations of mFBs in CMs undergoing spontaneous depolarization sequences using fluorescence imaging as described, and using phase contrast microscopy. Fluorescence imaging using Fluo-4 is very effective for mapping action potential propagation as indicated by calcium transients. Phase contrast imaging is useful for identifying cell boundaries and mapping motion and deformation in the cells. These two types of videos could not be taken concurrently with the setup used.

1.3 Image Analysis Methods

The focus of this thesis will be image analysis methods used to enhance and extract information from videos of depolarizing heart cells. Here we will give a brief overview of the image analysis methods that we have used prior to the work that is the focus of this thesis.

The first and simplest image analysis method that we used was the plotting of average intensities of selected spatial regions in an image array (video). This method involves importing a set of images (usually from hundreds or thousands of .tif files, or from a single multi-page .tif), into a 3-dimensional array (height x width x frame number) in MATLAB. The camera we used is monochromatic, so there is not a dimension for RGB color channel. We then use the *imfreehand()* function in MATLAB to select an ROI and create a binary mask (with ones inside

the region and zeros outside). This mask is then elementwise multiplied by the images in the array and the remaining non-zero values are averaged. Using this method, we obtain a plot of average intensities over time in our ROIs. A single such plot is shown in Figure 2 on the left, and multiple ROIs are shown on the right. An image of the ROI selection process is shown in Figure 3.

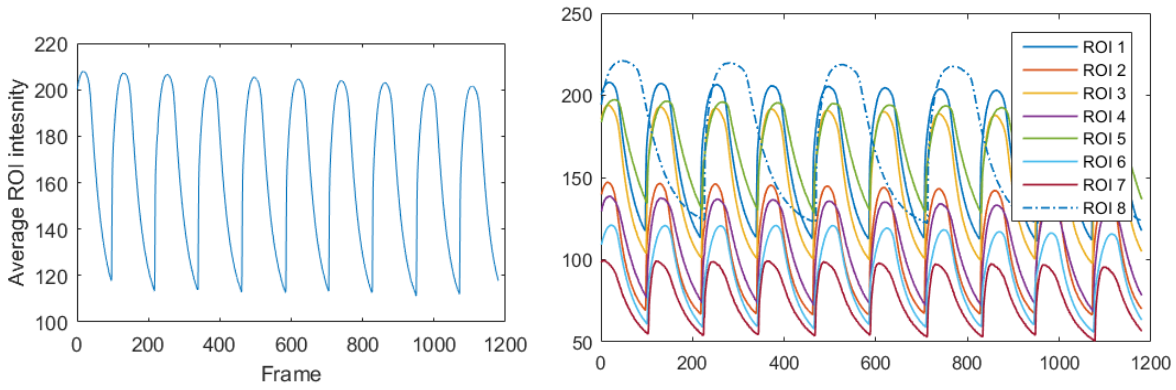

Figure 2 – Left: Single ROI intensity plot. Right: Intensity plot for multiple ROIs.

The ROI plotting method allows us to obtain several types of qualitative and quantitative information. We obtain calcium transient shapes which give us AP durations and periods. By looking at the initial calcium upstroke times, we can get AP delays between regions. We can compare different regions by normalizing the intensity plots and comparing maximum upstroke velocites to see how these are affected by mFB concentrations.

The second kind of analyisis typically performed involved enhancing certain aspects of the video in order to aid visualization of AP propagation. The Ca^{2+} transients shown in the plots above are easy to visualize because we can look at them all at once (especially when the plot is shown larger), and we have reduced noise by averaging pixel values over the entire region. However, when viewing the video, there may be significant image noise and it may be difficult to discern slow monochromatic changes in intensity to track AP propagation.

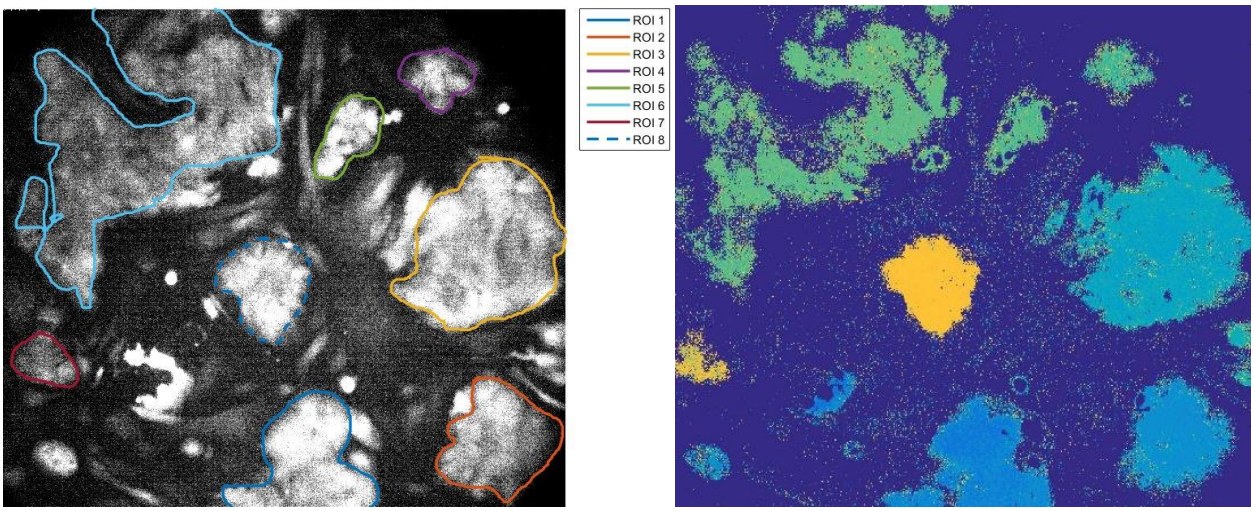

Figure 3 – Left: manual ROI selection. Right: AP isochrone map. Dark blue is earliest, yellow is latest. The background has been thresholded.

One visualization aid is to superimpose scaled time derivatives (technically *finite differences*) of the pixel intensities over over the original video in another color. We commonly use red to show increasing pixel intensity (representing Ca^{2+} transient upstroke velocity) and blue to show a falling pixel intensity. As you can see from the plots in Figure 2, Ca^{2+} upstrokes are brief, so we are able to get good separation of the signal location if we step through the enhanced video one frame at a time. We may obtain an AP isochrone map for a single AP sequence by creating an image in which each pixel is assigned the frame number of the maximum upstroke velocity in the corresponding video pixel (Figure 3).

An issue that we ran into early on with the time derivative based image enhancement was noise. Noise was a problem in the original videos, but when taking derivatives of noisy data, the noise becomes overwhelming to the original data. Initially, we tried combinations of Gaussian blur and median filtering to get the derivatives under control. Heavy application of spatial filtering worked well, and 3D versions of Gaussian and median filters (extremely resource intensive)

were also effective. However spatial filtering methods come at the expense of image resolution and 3D filters reduce temporal resolution as well.

Professor Robert Pless introduced the idea and method of using the Singular Value Decomposition (SVD) for image denoising and potentially for gathering data about APs in our videos. This method works extraordinarily well for getting noise-free visually satisfactory approximations of our original image sets. However, there are limits to the application of this powerful mathematical tool and it may introduce artifacts and omissions which may or may not be immediately visible to the eye.

The focus of this thesis will be an exploration of several software methods and their application to the analysis of scientific images, especially images of heart cells.

Chapter 2: Introduction to the Singular Value Decomposition

This section draws on several sources for general information about singular value decomposition and principal component analysis. See [7] [8] [9] [10] [11] [12].

2.1 Singular Value Decomposition:

The Singular Value Decomposition (SVD) decomposes a square or rectangular matrix into two orthogonal matrices and one diagonal matrix:

$$\boldsymbol{A_{mxn}V_{nxn} = U_{mxm}S_{mxn}}\,, \qquad \boldsymbol{or} \qquad \boldsymbol{A_{mxn} = U_{mxm}S_{mxn}V^T_{nxn}} \tag{1}$$

Here **A** is our original Matrix (*m x n*), **U** is an orthogonal matrix (*m x m*), **S** is a diagonal matrix (*m x n*), and **V** is an orthogonal Matrix (*n x n*). The columns of **U** (the left singular vectors) are the eigenvectors of $\boldsymbol{AA^T}$, and the columns of **V** (the right singular vectors) are the eigenvectors of $\boldsymbol{A^TA}$. The diagonal entries of **S** (the singular values) are the square roots of the eigenvalues of $\boldsymbol{AA^T}$ *and* $\boldsymbol{A^TA}$ (the non-zero eigenvalues are the same) *sorted by decreasing size.* The diagonal entries in **S** are always positive or zero.

Because **U** and **V** are *orthogonal* matrices ($\boldsymbol{U^TU = UU^T = I_{mxm}, V^TV = VV^T = I_{nxn}}$), they have columns of unit length. Magnitudes are contained in **S**.

To see that the columns of **U** are the eigenvectors of $\boldsymbol{AA^T}$, and the diagonal entries of $\boldsymbol{S}$ the square roots of the eigenvalues of $\boldsymbol{AA^T}$, write:

$$\boldsymbol{AA^T = (USV^T)(USV^T)^T = USV^TVS^TU^T = USIS^TU^T = USS^TU^T} \quad (2)$$

$\boldsymbol{V}$ is orthogonal because its columns are the eigenvectors of a symmetric (full-rank if the columns of **A** are independent) matrix, so $\boldsymbol{V^TV = I}$. Because **S** is a diagonal matrix, $\boldsymbol{SS^T}$ is a diagonal matrix whose diagonal elements are the squares of the diagonal elements of $\boldsymbol{S}$. We recognize this as the eigen decomposition of the symmetric matrix $\boldsymbol{AA^T}$. We can do a similar procedure to show that the columns of **V** are the eigenvectors of $\boldsymbol{A^TA}$, and the squares of the diagonal entries of $\boldsymbol{S}$ are its eigenvalues.

There are several ways to think about the SVD. One way to think of it is that the columns of $\boldsymbol{U}$ (the left singular vectors) form an orthogonal basis for the columns of $\boldsymbol{A}$ sorted in decreasing order of importance. If we choose an arbitrary number *p* of consecutive columns starting from the left side of matrix $\boldsymbol{U}$, those will form the best *p*-dimensional basis for reconstructing the columns of $\boldsymbol{A}$. The entries of the diagonal matrix $\boldsymbol{S}$ are the weights of those basis vectors (sorted in descending order). The columns of $\boldsymbol{V^T}$ (or the rows of $\boldsymbol{V}$) are the coefficients used to reconstruct the corresponding columns of $\boldsymbol{A}$ from a linear combination of the singular value weighted columns of $\boldsymbol{U}$.

For instance, the first column of $\boldsymbol{A}$ is a linear combination of the columns of $\boldsymbol{US}$ (the *principal components*), where the first column of $\boldsymbol{V}^T$ gives the coefficients of the linear combination. The second column of $\boldsymbol{V}^T$ gives the coefficients used to reconstruct the second column of $\boldsymbol{A}$. We need one column of $\boldsymbol{V}^T$ for every column of $\boldsymbol{A}$.

Looking at the first form of the SVD equation, $\boldsymbol{AV = US}$, we can also look at $\boldsymbol{V}$ as an orthogonal transformation that takes the correlated columns of $\boldsymbol{A}$ and finds the uncorrelated principal components $\boldsymbol{US}$. The orthogonal principal component vectors (the columns of $\boldsymbol{US}$) are linear combinations of the columns of $\boldsymbol{A}$ (given by the corresponding columns of $\boldsymbol{V}$).

Because the rows of $\boldsymbol{S}$ with row indices greater than *n* will be zeros (assuming *m* > *n*), we can write an "*economy size*" or "*thin*" version of the SVD as follows:

$$\boldsymbol{A_{mxn} = U_{mxn} S_{nxn} V^T_{nxn}} \tag{3}$$

Here the columns of $\boldsymbol{U}$ with column indices greater than *n* have been discarded, and $\boldsymbol{S}$ has been resized to remove the *m* – *n* rows of zeros at the bottom.

2.2 Principal Component Analysis

The terms Principal Component Analysis (PCA) and Singular Value Decomposition are sometimes used interchangeably. Often people will say that they have used the SVD on a mean-centered data matrix to perform PCA. Here we are interested in the contributions of the individual principal components that we have found using the SVD. The contribution of each principal component to the reconstruction of our original matrix may be seen in the rank-1 matrices that are formed by the outer products of the columns of **US** (the PCs) with the corresponding rows of $\boldsymbol{V^T}$. If the columns of **U** and **V** are $\boldsymbol{u_i}$ and $\boldsymbol{v_i}$, respectively, and the diagonal entries of **S** are σ_i, then the contributions of the individual principal components are $\boldsymbol{u_i v_i^T} * \sigma_i$. We can reconstruct **A** exactly by adding the contributions of all the principal components. The contribution matrix of each principal component is rank-1, because each column of the matrix is a scalar multiple of every other column and every row is a scalar

multiple of every other row (because it is formed by an outer product of two vectors). The principal components are ordered in descending order of importance, where $\boldsymbol{u_1 v_1^T} * \sigma_1$ is the most important (contributes the greatest variance to **A**) and $\boldsymbol{u_r v_r^T} * \sigma_r$ is the least important (where r is the rank of **A**). Singular values corresponding to the columns of **U** with indices greater than the rank of the **A** matrix will be zero, meaning the number of non-zero principal components will not exceed the rank of the **A** matrix. In fact, the number of non-zero singular values is a measure of the rank of **A**.

We may choose to reconstruct our original **A** matrix approximately using an incomplete set of principal components. If **A** may be well approximated by a lower rank matrix, this may be desirable. Higher order principal components may capture noise and less important data. All or most of the important data may be captured in the lower order components. We can see this in the plot of the singular values of a 100-frame bright-field video of beating cardiomyocytes (Figure 4). The video may be reconstructed very well with 20 or fewer principal components, leaving mostly noise in the residuals.

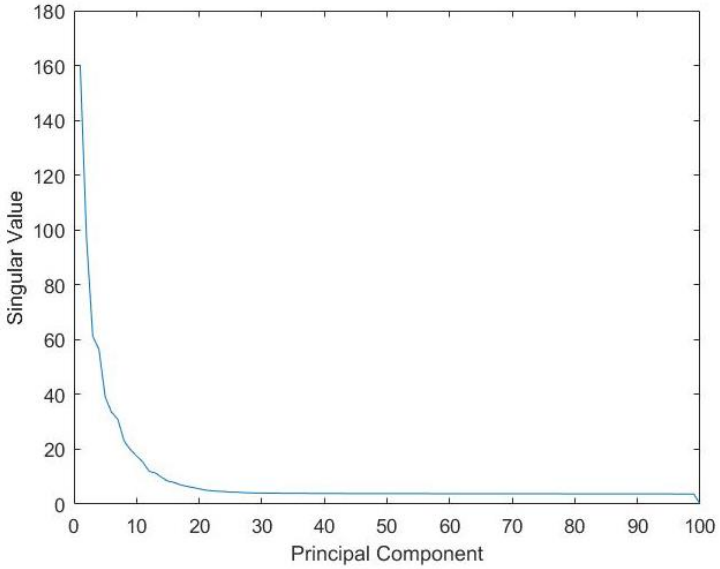

Figure 4 – Typical plot of singular values for a beating heart cell video.

2.3 Application to Video:

To perform Matrix operations on our monochrome video array, we must first convert it to a 2D matrix. Our video array starts out 3 dimensional (number of pixels high x number of pixels wide x number of frames). To convert this into a matrix we use linear indexing to stack the columns of each frame into a single column vector and then horizontally concatenate these columns into a matrix. We then have a matrix whose height (m) is the total number of pixels per frame and whose width (n) is the total number of frames.

Lastly, we must center the data by subtracting the data mean. If we did not center our data, the first principal component would simply point from the origin to our data mean, rather than pointing in the direction of greatest data variance from the mean. All lower order principal components would be affected as well. In our case, we can center our data by averaging all the columns (the frames) and subtracting that mean from the individual columns. All data is now represented as a variation from our mean image. We save this mean column for use later. Because the video data all shares a common scale (pixel intensity), we have chosen not to normalize or scale the per-frame intensity values in any way.

We then decompose our video matrix into $\boldsymbol{A = USV^T}$. This can be done using the *svds()* function in MATLAB, which allows the user to specify an arbitrary number of singular values to compute (along with the corresponding columns of **U** and **V**).

2.3.1 Noise Reduction

To reduce noise, we may reduce the number of principal components used to reconstruct **A** by setting higher order singular values to zero. This is called a truncated SVD (TSVD) when a consecutive set of PCs is used starting from PC 1. To reconstruct the original video, we calculate

$\boldsymbol{A = US_kV^T}$, where $\boldsymbol{S_k}$ is our $\boldsymbol{S}$ matrix with singular values of index greater than *k* set to zero, add the mean image back to our reconstructed **A** matrix, and then reshape it into a standard video array format.

To save computer memory usage and processor time, we may choose to not calculate all principal components, which will reduce the number of columns of **U**, the number of rows of $\boldsymbol{V^T}$, and the size of **S** accordingly.

Reconstructing our original bright-field and fluorescent videos using only low order principal components resulted in noise reduction that was both more effective and preserved greater detail than the use of median or Gaussian filtering. This is discussed in greater detail in Chapter 3.

2.3.2 Data Compression

There are several ways that the truncated SVD may be used for image and video compression. For instance, the SVD may be used to find a basis of rows or columns of an image. An image may be broken into smaller regions and the SVD may be used to find a basis set for those regions. Here I will only discuss compression using the SVD to find a full-frame basis for the images of the video.

The pco.edge camera that we have been using for image acquisition is capable of capturing 5MP images at a rate of 100FPS. At this rate, saving .tif files at single-channel 8-bit quantization, ten seconds of video requires $10s * 100frames/s * 5MP/frame * 1\ byte/pixel \approx 5GB$ of storage. If we decide that ten principal components are sufficient to capture the information we are interested in, this reduces the storage needs to a set of 10 basis images and a set of 1000 coefficients for each basis: $5MB * 10 + 1{,}000B * 10 \approx 50MB$. Even if we store the TSVD in double precision format (~400MB) we have reduced storage requirements tremendously.

2.4 Data Completion

One of the difficulties we have had in finding a transfer function between the calcium signal and mechanical motion in cardiomyocytes is getting bright field videos (which are good for analyzing motions) and fluorescent videos (which are good for analyzing calcium signals) that are either taken concurrently, or well synchronized. Because we are not pacing our cells, it is difficult to sync the two types of videos for two reasons: Firstly, we do not know what the delay is between the calcium signal and the motion of the cells. This makes syncing the start of the AP cycle difficult. Secondly, the periods of the APs are not totally consistent. Even if we get the first cycle synced, the sync will be off within a cycle or two.

The idea behind PCA Data Completion of video information is to find two short videos from a given stage location, one fluorescence, one bright-field, and use these to form a basis from which to reconstruct data of one type if we have data of the other type. First, we must sync the two short videos as well as possible either by inspection (we can detect a certain amount of motion from the fluorescence videos) or by using cross correlation between the respective vectors of coefficients of the first principal components. We can then use the SVD to form an optimal basis with common coefficients for the fluorescent and bright-field video space, and a least-squares fit to find our new coefficients (to construct our "missing" video). As we will see, the properties of the SVD make finding a least-squares fit **V** very easy.

We must find a basis for each synchronized representative video with the constraint that the two bases use the same set of coefficients to reconstruct their respective videos. If we call these videos $\boldsymbol{A}_{short} and\ \boldsymbol{B}_{short}$ (where both have been *centered* as described), we can represent this mathematically using the SVD as follows:

$$\begin{aligned} \boldsymbol{A_{short}} &= \boldsymbol{U_1 S V^T} \\ \boldsymbol{B_{short}} &= \boldsymbol{U_2 S V^T} \end{aligned} \tag{4}$$

Now if we have a new video, $\boldsymbol{A}_{long}$**,** we can find a new **V** matrix using a least squares approximation with the principal components of $\boldsymbol{A}_{short}$. Since we have a shared basis for the **A** videos as well as the **B** videos, we can apply the coefficients we obtained from the **A** matrix to reconstruct our unknown **B** matrix.

To find a common basis $\mathbf{U_{AB}}$**,** that shares a common set of coefficients **V**, we vertically concatenate the two short **A** and **B** matrices. Now we have one matrix, where each column vector is the corresponding column vector from matrix **A** stacked above the corresponding column vector from matrix **B**. If we perform the SVD on this vertically concatenated matrix, we find a Matrix of basis vectors, $\mathbf{U_{AB}}$, that forms an optimal basis for reconstructing both videos at once. The coefficients in **V** are for both videos. We can use this basis to reconstruct missing data from one type of video if we have data from the other.

Say we have three videos $\boldsymbol{A}_{short}, \boldsymbol{B}_{short},$ and $\boldsymbol{A}_{long}$. $\boldsymbol{A}_{short}$ and $\boldsymbol{B_{short}}$ are time synchronized and we want to find a long version of **B** using the data in $\boldsymbol{A}_{long}$.

First, we convert our two short synchronized representative videos into matrices (**As** and **Bs**) as described above. Then we vertically concatenate the two matrices:

$$\boldsymbol{AB} = \begin{bmatrix} \boldsymbol{As} \\ \boldsymbol{Bs} \end{bmatrix} \tag{5}$$

Then we center our concatenated matrix:

$$\boldsymbol{ABc} = \boldsymbol{AB} - \boldsymbol{\mu_{AB}} \tag{6}$$

Where column vector $\boldsymbol{\mu_{AB}}$ is the mean of the columns of AB.

Then we perform the SVD:

$$\boldsymbol{ABc} = \boldsymbol{U_{ABc}} \boldsymbol{S_{ABc}} \boldsymbol{V_{ABc}^{T}} \tag{7}$$

To find a new set of coefficients from which to reconstruct a longer version of **B**, we do a least-squares fit using the top half of the above basis to the centered matrix of our long A video.

$$\begin{aligned} \boldsymbol{A_{long}} &= \boldsymbol{U_{ABc}}(\boldsymbol{top}) \boldsymbol{S_{ABc}} \boldsymbol{V_{long}^{T}} \\ \boldsymbol{V_{long}} &= \left((\boldsymbol{U_{ABc}}(\boldsymbol{top}) \boldsymbol{S_{ABc}}) \setminus \boldsymbol{A_{long}} \right)^{\boldsymbol{T}} \\ &= \boldsymbol{A_{long}^{T}} \boldsymbol{U_{ABc}} \boldsymbol{S_{ABc}^{+T}} \end{aligned} \tag{8}$$

In the second line of the equations above, the "\" represents "ldivide" in MATLAB, which will find the best **V** matrix using a least squares algorithm. This will not generally be an exact solution, but rather a projection of the **A** matrix onto our principal component space. We are only using the top half of $\boldsymbol{U_{ABc}}$, the half that corresponds to the "A-type" video. However, this is not the best way to calculate **V**, but it is included to show that what we are doing is indeed equivalent to a least-squares fit. The better way to calculate **V** is to use the third line, where $\boldsymbol{S_{ABc}^{+T}}$ is the transpose of the pseudoinverse of $\boldsymbol{S_{ABc}}$, in this case found by taking the reciprocal of

all non-zero diagonal entries of **S**, and leaving zeros as zeros. This takes advantage of the orthogonality of **U** and the diagonality of **S**. We will often be using a thin or truncated SVD, where **U** and **V** are rectangular and **S** is square, but the equations in (8) will still work.

To calculate the long version of **B**, we use the bottom half of $\boldsymbol{U_{ABc}}$:

$$\boldsymbol{B_{long} - \mu_B = U_{ABc}(bottom) S_{ABc} V_{long}^T} \quad (9)$$

To convert this back into a video, we add the mean values back to the matrix above and convert it to standard video array format.

2.5 Physical Meaning of Principal Components

The first principal component captures the change in the video which contributes the greatest variance to the overall video. In some cases, this may correspond with some physically meaningful aspect of the observed cells, such as an overall change in fluorescence intensity, or a point of greatest displacement or contraction. As such, the first principal component may be used as a proxy for cardiomyocyte action potential state or intracellular calcium concentration in some cases. We have used cross correlation between the coefficients of the first principal components from a bright-field and fluorescence video to synchronize them convincingly.

High-order principal components mostly represent noise (see Chapter 3). This may be seen by looking at the components individually, or by looking at the residuals of reconstructions using only lower order principal components. We can see from the low singular values that their individual contributions are small. However, there may be useful information in higher order PCs, depending on what you are looking for.

Because high order PCs represent information that is uncorrelated with the lower PCs and contributes little to the overall variance of the data, we may find features in higher order PCs that would be difficult to observe otherwise. For instance, buried in the noise of the residuals of a low order reconstruction we may find features such as a cardiomyocyte that fires dimly and irregularly, or shadows from bits of debris floating across the surface of the cell medium. By removing the high-variance regular features, we uncover the faint irregular features.

One motivating idea behind applying PCA to these videos was that the low order PCs might represent different modes of change in the tissue constructs and that perhaps they would allow us to separate contributions from individual cells, or separate changes in fluorescence intensity from changes in cell shape. Disappointingly, beyond the first PC this is generally not the case. For the most part, the low-order PCs are simply mathematically optimal bases that may not have any connection to physical phenomena. The coefficients in time (rows of $\boldsymbol{V^T}$) of the PCs above 1 do not generally to correspond well to any physical phenomena. It may be worth exploring ways to "force" the PCs to represent different phenomena, but that is beyond the scope of the current work.

An additional complication that arises when trying to extract physical significance from the computed PCs is that the components are unique only up to their sense. The entries of **S** must be positive, but any columns of **U** may be multiplied by -1 if the corresponding columns of **V** are also multiplied by -1. If we are using the coefficients of the first principal component as a proxy for intracellular calcium concentration, we must ensure that the values in the vector of coefficients rise as the calcium concentrations rise, rather than doing the opposite.

2.6 Limitations

The use of PCA for noise reduction and data completion is limited in application by several factors.

PCA is useful for our videos of cardiomyocytes because the videos contain static features and repeated events. Because some parts of our image set change very little and others change in repetitive ways, these images are reconstructed quite well using a small set of principal components (often less than ten without appreciable loss of data). The applicability of a reduced set of principal components may be seen in the plot of the singular values (which drop off rapidly in our case). If the data do not contain static or repetitive features, they will not be well represented by a lower rank approximation. This will be seen as a plot of singular values that does not drop off quickly, but instead shows that higher order components contribute significant variance. In this case reconstruction with a reduced set of components will result in loss of significant data.

Data completion as described above is possible only if there are underlying correlations between the data sets. In the case I showed above, this is true because we are looking at the same set of cells undergoing similar action potential sequences, though at different times. The SVD allows us to find the underlying correlation between the two video types, and allows us to infer data from one, given data from the other. However, we cannot use this same basis and coefficient set to recreate data from some other stage location or from a different set of cells. To create a meaningful shared basis between two sets of images, we must start with two sets of images which are synchronized recordings of correlated events.

Lastly, in addition to being limited in applicability, applying the SVD to large matrices is computationally demanding. Processing large data sets is not feasible using standard solvers. However, iterative solvers may allow for finding very good approximations of the SVD with lower resource requirements. See Chapter 4.

Chapter 3: PCA Image Denoising and Using Spectral Analysis for Component Choice

3.1 Motivation

Noise can be a significant problem in recorded data, and this is especially the case in the high-speed videos of fluorescent reporters such as those used in cardiac optical mapping. The emission levels of fluorophores used in optical mapping are much lower than the luminance levels encountered in bright-field microscopy. In addition, excitation source intensity must be kept to a level which will not cause problematic photobleaching, further limiting light levels.

The temporal resolution required to map cardiac impulse propagation (~50cm/s [1]) over microscopic or even macroscopic scales requires the use of high-speed videography. High framerates require short exposures.

The spatial resolution of a high-resolution camera such as the 5MP pco.Edge sCMOS camera we have been using to record cardiac videos means that there is a relatively small light gathering area for each pixel. For this reason, low-resolution photodiode arrays, often used with fiber-optic cable arrays, have been the only practical option for high-speed optical mapping in the past. See [13] and [14] for examples. However, improvements in digital image sensors and continuing reductions in price have led to increased use of scientific CCD and CMOS sensor based cameras and even consumer level digital cameras for cardiac optical mapping applications (e.g. [15]).

The combination of low light levels, short exposure times, and small light-gathering areas leads to problematic noise levels in videos and images. Two possible sources of this noise are *shot*

noise and *Johnson-Nyquist* (thermal) *noise*. Shot noise is caused by the random nature of photon collection, and that the number of photons collected in a given interval will vary according to a Poisson distribution. The fewer the average photons collected in an interval, the lower the signal-to-noise ratio. Johnson-Nyquist noise is temperature dependent electrical noise caused by random fluctuations of electrical currents in circuit resistors. This noise is not signal correlated, but it may be exacerbated by high amplification of a weak signal.

3.2 Noise Reduction Overview

Noisy images and videos may be de-noised in a number of ways. Two common methods include Gaussian smoothing (Gaussian blur) and median filtering. See [16] for an extensive explanation.

Gaussian filtering (*imgaussfilt()* in MATLAB) is a linear filter that may be understood as using an array of weighting values (the kernel of the filter) and moving it around the 2d image, using the values in the kernel as coefficients for a weighted average of all pixel values falling under the current location of the filter. The weighted average value is then assigned to the pixel in the smoothed image corresponding to the center of the kernel location. As the name implies, the weights in the kernel are based on a discretized Gaussian distribution (though the general technique, *convolution,* may be used with other distributions to achieve other kinds of linear filtering). The size and standard deviation of the kernel may be varied to control the characteristics of the smoothing. This is essentially a low-pass filter that reduces high spatial frequency features in the image. This type of filtering may also be performed in the frequency domain using the Fast Fourier Transform (FFT).

Median filtering (*medfilt2()* in MATLAB) is a non-linear filter that involves taking blocks of pixels from an image, finding the median pixel value, and then assigning that value to the pixel

in the filtered image corresponding to the center of the block. Block size may be varied to change the characteristics of the filtering. Median filtering may be effective at removing certain types of noise while better preserving edges and sharp features than Gaussian filtering. Both filters may be effective at reducing noise levels in an image or video, but that reduction in noise comes at the cost of spatial resolution and acuity.

3D versions of both filters exist as well. Applying a 3D filter to a video array amounts to filtering in the temporal dimension of the array as well as the two spatial dimensions (effectively averaging over a volume rather than an area). In this case, temporal resolution will be affected as well.

3.3 Singular Value Decomposition

Singular Value Decomposition (SVD) is appealing as a noise reduction technique because it has the potential to remove noise from certain kinds of videos without loss of spatial or temporal resolution (though it may introduce other artifacts). This is because de-noising using the SVD works in an entirely different way from standard image filters. The SVD may be thought of as decomposing our image set (video) into a set of basis images and a set of coefficients that combine the basis images into the images in the video. The basis images are uncorrelated and sorted by decreasing order of importance. This allows us to keep the lower order number (more important) basis image contributions and discard contributions from higher order number (less important) basis images. This is referred to as the truncated singular value decomposition (TSVD). Because the higher order number components tend to be dominated by noise, we may be able to remove significant noise without losing spatial or temporal resolution. The cost of this

technique is that it is computationally demanding, not always applicable, and may introduce undesirable artifacts.

If we use all the principal components of a matrix **A,** where the columns of **A** represent the stacked columns of the frames of a video, we reconstruct **A** exactly:

$$\boldsymbol{A - \mu = USV^T}, \quad or, \quad \boldsymbol{A - \mu} = \sum_{i=1}^{r} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^T \tag{10}$$

Here we show the standard form of the SVD of our centered (mean image subtracted) **A** matrix on the left, and an equivalent representation on the right showing reconstruction of the centered **A** matrix by a sum of singular matrices formed by the outer product of the columns of **U** ($\boldsymbol{u}_i$) and the columns of **V** ($\boldsymbol{v_i}$), weighted by the corresponding singular values (σ_i, which are the diagonal entries of **S**). Here *r* represents the rank of matrix **A**. We will use the same notation used in [12] and show subtraction of the mean of the columns of **A** from the columns of **A** as $\boldsymbol{A - \mu}$**,** where $\boldsymbol{\mu}$ is a column vector. (For clarity, $\boldsymbol{\mu_i} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{A}_{ij}$, *i* = 1, 2, …, m; where $\boldsymbol{A}$ is an *m x n* matrix.)

A truncated SVD sets all singular values with index above k to zero:

$$\boldsymbol{A_k - \mu = US_kV^T}, \quad or, \quad \boldsymbol{A_k - \mu} = \sum_{i=1}^{k} \boldsymbol{u_i} \sigma_i \boldsymbol{v_i^T} \tag{11}$$

Here $\boldsymbol{A_k}$ is the rank-k projection of $\boldsymbol{A}$ onto principal components 1:*k*, where $k < n$. $\boldsymbol{S_k}$ is the diagonal matrix formed by taking **S** and setting all singular values with index above k to zero. $\boldsymbol{A_k - \mu}$ is the best rank-*k* least squares fit for $\boldsymbol{A - \mu}$.

3.4 Choosing K

An obvious question that arises when attempting to separate the data-dominated lower order components from the noise-dominated higher order components is where to set the cutoff point. There are a number of approaches and criteria, and much has been written about choosing regularization parameters (such as our truncation value *k*) when the SVD is used to solve discrete inverse problems with ill-conditioned coefficient matrices (see [17] [18] [19]). We are using the SVD for de-noising in this case, but we will investigate whether some of the techniques used in the sources listed above can be applied or adapted to choosing a truncation parameter for our purpose.

Start simple - look at the singular values:

Because the contributions of the individual principal components are weighted by their singular values, we can get a sense of the contribution of each component by the size of its singular value. For many of the heart tissue videos that we have looked at, the singular value plot drops rapidly at first and then levels off. We can see this in the singular values of an example noisy fluorescence video in Figure 5.

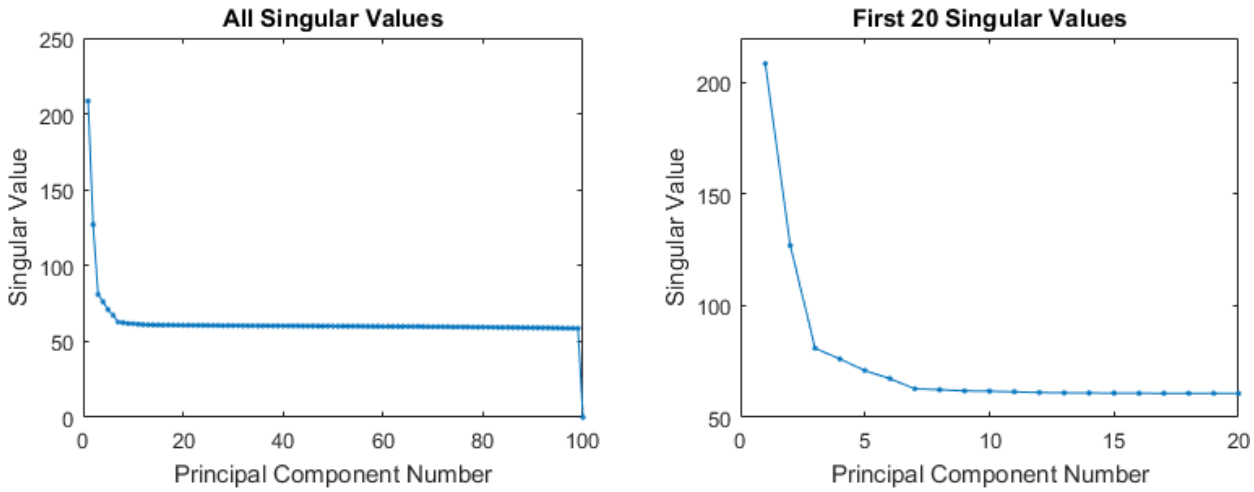

Figure 5 – All 100 singular values from a 100-frame fluorescence video are shown on the left. The first 20 singular values are shown on the right.

One tempting interpretation of a singular value plot such as the one shown in Figure 5 is that the first 6 or so singular values, which are above the plateau line, correspond to data-dominated components and that the components along the plateau line correspond to the noise-dominated components. There are several justifications for this interpretation:

Low frequencies dominate low-order components:

As with other decompositions, such as the Fourier Decomposition, larger coefficients (singular values) in the SVD tend to be associated with lower frequencies and smaller coefficients tend to be associated with higher frequencies (see section 2.5 of [19]). Much of the most important information in our cardiomyocyte videos lies in the lower frequency range. Important spatial features are 10s or 100s of pixels wide and the important temporal changes have periods of around one second (though, as in a Fourier Decomposition, we may need higher frequency components to reconstruct low frequency motions). In our SVD, spatial frequencies are captured in the columns of **U** (our basis images) and temporal frequencies are captured in the corresponding columns of **V** (the associated coefficients in time).

The singular values indicate the variance:

The contribution from each principal component is weighted by its corresponding singular value. This is apparent in the right-hand version of Equation (10). We can also look at the singular values as the square roots of the overall variances contributed by each PC:

$$R_i = \frac{\sigma_i^2}{\sum_{n=1}^{r} \sigma_n^2} \tag{12}$$

Here R_i is the fraction of the overall variance in the original video contributed by PC i, and r is the numerical rank of the original video matrix (total number of non-zero singular values). We can also use this formula to calculate the fraction of the variance that is preserved by a TSVD:

$$R_k = \frac{\sum_{m=1}^{k} \sigma_m^2}{\sum_{n=1}^{r} \sigma_n^2} \tag{13}$$

Here R_k is the fraction of the original variance preserved by the truncated SVD. The reason all of this is useful is that it means that by discarding the higher-order components, we are keeping the ones that have the greatest overall contribution. Large, structured changes (the ones we want to keep) will have large contributions to the overall variance, whereas small, random changes (noise) will individually contribute little to the overall variance. The reason for the plateau in the singular value plot is that there are many basis vectors that contribute similar, relatively small, amounts to the variance and fill out the rank r of the original matrix. It is reasonable to assume that these are primarily bases for random image noise. As noise levels increase, and signal-to-noise levels decrease, it will be increasingly difficult to separate the two, and information will be lost.

Low-order components show clear structure:

It is clear from inspection of the basis images in **U**, the coefficients in **V**, and reconstructions of the original image set by TSVD that the most important information lies in the low-order principal components. Low-order basis images show recognizable features from the image sets, whereas higher-order ones degrade to noise. Low-order columns of **V** (basis coefficients in time) show structure corresponding to video events, whereas higher-order columns degrade to noise. Residual image sets (reconstructed images subtracted from their originals, see Eq. (20)) show mostly noise even with $k = 3,4,5$ components (see Figure 12). This is neither surprising nor new.

What is not clear is exactly where one should set the truncation parameter *k* for the best de-noised reconstruction of our original image set. What if we wish to automate SVD de-noising and do not want to spend the time required to look at multiple reconstructed image sets using different truncation parameters to pick the best one? What is the criterion? Do we truncate at a certain threshold of the singular values? Do we truncate when the negative slope of the singular value plot flattens out to certain value? What if important information is still present in the early principal components that have plateaued singular values?

3.5 Residual Analysis and the Normalized Cumulative Periodogram (NCP)

In the plot of the singular values in the above section, we looked at the *size* of the contribution from each principal component. In this section, we will attempt to look at the *character* of the contribution.

The TSVD is used as a regularization method in solving ill-posed discrete inverse problems (see [17] [18] [19]). Here we will borrow from the methods used in the references above to choose a *regularization parameter k* and apply ideas from those methods to our image de-noising problem.

Summary of the application in discrete ill-posed inverse problems:

Say we wish to solve for $\hat{\boldsymbol{x}}$, an approximate solution for input $\boldsymbol{x}$ in the following equation:

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\eta} \tag{14}$$

Here $\boldsymbol{b}$ is a vector of our measurement data, **A** is some process matrix, and $\boldsymbol{\eta}$ is normally-distributed measurement error. If matrix **A** has a very high condition number, then "…the elements of $\hat{\boldsymbol{x}}$ are pathologically sensitive to small variations in the elements of **b**…" [17]. The naïve least squares solution

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{b} \tag{15}$$

will yield an $\hat{\boldsymbol{x}}$ that bears little resemblance to $\boldsymbol{x}$.

One approach to this problem, if we know that $\boldsymbol{\eta}$ is has an evenly distributed periodogram (as would white noise or normally distributed perturbations of the measurement), is to solve the inverse problem using a TSVD and choose a truncation parameter *k* that gives us a residual vector (Equation (18)) that has the spectral characteristics of white noise. The SVD may be used to compute the least squares solution as in Equation (16), where $\boldsymbol{S}^{+}$is the pseudoinverse of $\boldsymbol{S}$ (In this case, all non-zero diagonal entries of $\boldsymbol{S}$ are inverted and the matrix is transposed. Zeros remain zeros.). A proof may be found on page 65 of [11].

$$\hat{\boldsymbol{x}} = \boldsymbol{V}\boldsymbol{S}^{+}\boldsymbol{U}^{T}\boldsymbol{b} = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i^{T}\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i \tag{16}$$

The above sum may be truncated to remove disproportionately large contributions caused by disproportionately small values of σ_i relative to $\boldsymbol{u}_i^{T}\boldsymbol{b}$ (see *Discrete Picard Condition*):

$$\hat{\boldsymbol{x}}_k = \boldsymbol{V}\boldsymbol{S}_k^{+}\boldsymbol{U}^{T}\boldsymbol{b} = \sum_{i=1}^{k} \frac{\boldsymbol{u}_i^{T}\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i \tag{17}$$

We then analyze the residual vector $\boldsymbol{e}_k$ to find a best truncation parameter k:

$$\boldsymbol{e}_k = \boldsymbol{b} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_k \tag{18}$$

Here $\boldsymbol{A}_k$ is the TSVD reconstruction of our process matrix $\boldsymbol{A}$. If $\boldsymbol{e}_k$ has the spectral characteristics of white noise, then the hope is that $\boldsymbol{e}_k = \boldsymbol{b} - \boldsymbol{A}_k\hat{\boldsymbol{x}}_k \approx \boldsymbol{\eta} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ and $\hat{\boldsymbol{x}}_k \approx \boldsymbol{x}$ (because we know that $\boldsymbol{A}_k$ is the *best* rank-k approximation of $\boldsymbol{A}$). If *k* is too low, we expect low-frequency information to remain in the residuals; if k is too high, we expect that the residuals will contain only high-frequency components of the noise (leaving low-frequency noise components in our solution). We use the word *hope* above, because different regularization methods work best with different problems. There is no single approach which gives consistently optimal results [18].

The periodogram of a signal is found by taking the Fast-Fourier Transform (FFT) of the signal, and then squaring the modulus of the FFT values. This gives an overview of the contributions from individual frequencies to the overall signal power. By creating a vector of the cumulative sum of the periodogram values at each vector element, we create a cumulative periodogram. If we then divide this vector by its largest value (the last element), we have the *normalized*

cumulative periodogram (NCP). If all frequencies contribute equally, this will be a straight line. If lower frequencies dominate, it will rise quickly early then level off. If higher frequencies dominate, it will rise slowly at first and quickly at higher frequencies.

To decide whether or not a signal has the spectral properties of white noise (a flat periodogram), both Rust and Hansen compare the NCP of the signal to a straight line going from 0 to 1 on the ordinate, and 1 to q on the abscissa, where $q = n/2 + 1$, and n is the number of samples. If the ordinates of the NCP fall within the Kolmogorov-Smirnoff limits of the straight line at $\alpha = 5\%$, the signal is considered "white noise like" [18]. Hansen gives this limit as $\pm 1.36q^{-1/2}$.

Application to video de-noising

In a video array, if noise comes from shot noise or Johnson-Nyquist thermal noise, we expect that the noise will have a flat periodogram (we are discarding the noise mean). We expect thermal noise to be normally distributed and independent of the signal. For shot noise, we expect the standard deviation of the noise to vary proportionally to the square root of the signal, so that the signal-to-noise ratio varies inversely to the square root of the signal. *We should note that image and video compression algorithms such as .mpeg compression may drastically change the character of the noise. All videos analyzed here were saved directly into .TIF format.*

This may be written

$$\boldsymbol{A_{ij}} = \boldsymbol{A_{ij}^{*}} + \boldsymbol{N_{ij}^{thermal}} + \boldsymbol{N_{ij}^{shot}}, where\ std(\boldsymbol{N_{ij}^{shot}}) \propto sqrt\left(\boldsymbol{A_{ij}^{*}}\right) \tag{19}$$

Here $\mathbf{A}_{ij}$ is our video array, $\boldsymbol{A_{ij}^{*}}$ is an ideal noise free video array, $\boldsymbol{N_{ij}^{thermal}}$ is thermal noise, and $\boldsymbol{N_{ij}^{shot}}$ is shot noise.

We will define the TSVD video residuals as the differences between the original video file and the TSVD reconstructed video:

$$\boldsymbol{E_k = A - A_k} \tag{20}$$

Notice the residual has the same format as the original video array, which is to say it is a two-dimensional matrix (where each column represents the stacked columns of an image) while we are doing matrix operations in MATLAB, but it can be converted into a three dimensional array (where the first two dimensions are image width and height, and the third is frame number) and viewed in a video player (*implay()* in MATLAB). To analyze the frequency spectrum of the residuals, we will analyze the 2D Discrete Fourier Transform (DFT) of the 2D image residuals using *fft2()* in MATLAB. We avoid analyzing the 1D frequency spectrum of the columns of the matrix format residual, because this introduces artificial periodicity by stacking columns, though this periodicity should become less important as the residual becomes more noise-like.

To analyze the periodogram of the 2D image residuals, we follow the guidelines given in [18]. First, we take the 2D DFT of a residual image using *fft2()* in MATLAB. We then take the square of the modulus of the values in the upper left quadrant to find the 2D periodogram. These values need to be sorted in order of increasing frequency (distance from the upper left corner). To do this we create a matrix of the same size as our quadrant with values $i^2 + j^2$, where i and j are our row and column indices respectively. We can then turn this matrix into a vector and use the MATLAB *sort()* function to find the permutation that orders the values in non-decreasing order. We use this permutation to sort the linearly-indexed version of our periodogram into a 1D function of frequency. Because there is different information in the upper left and lower left

quadrant periodograms, we have averaged the two. Opposite corner quadrants of the 2D periodogram contain the same information.

In MATLAB, we used the following code:

```
% find permutation to arrange periodogram from low to high freq.
x = 1:floor(cols/2)+1;
y = 1:floor(rows/2)+1;
vec_length = x(end)*y(end);
[X,Y] = meshgrid(x,y);
dist2 = X.^2 + Y.^2;
[~, indx] = sort(dist2(:));

% get 2D periodogram and set zero freq. = 0
p_array = abs(fft2(array_in)).^2; p_array(1,1) = 0;
white_line = linspace(0,1,vec_length)';
ncp = zeros(vec_length,frames); %pre-allocate NCP
dev = zeros(frames,1); % pre-allocate deviation
for i = 1:frames
    pdgrm_n = p_array(y,x,i); % get current 2D periodogram
    % get flipped 3rd quadrant to add to 2nd
    pdgrm_flipud = p_array(end-y+1,x);
    %re-order, add left two quadrants
    p_sort = pdgrm_n(indx(:)) + pdgrm_flipud(indx(:));
    % calculate NCP & find deviation from white noise
    ncp(:,i) = cumsum(p_sort);
    ncp(:,i) = ncp(:,i)/ncp(end,i);
    dev(i) = mean(abs(ncp(:,i)-white_line));
end
```

This code was tested with an array of normally distributed white noise (*noisy = randn(300, 300, 25);* in MATLAB) and gave the expected result: all NCPs were nearly straight lines with small deviations (~10^{-3}) and all fell within KS limits.

3.6 Test case 1: Cardiomyocyte Fluorescence Video

For our first test case, we will look at a video of a multi-layered clumps of CardioMyocytes (CMs) derived from Induced Pluripotent Stem Cells (IPSCs) and myoFibroBlasts (mFBs) from dermal tissue. Both are of human origin. The cells have been stained with Calcium-reporting Fluo-4 fluorescent assay, which is responsible for the changes in pixel intensity in CMs

undergoing action potentials (APs) in the video. The cells are not externally paced. Stills from every six frames of the test case video are shown in Figure 6. The video contains deformation motions due to CM contractions, changes in intensity due to Ca^{2+} transients, and significant image noise. Figure 7 shows a crop from the first image to show the noise pattern.

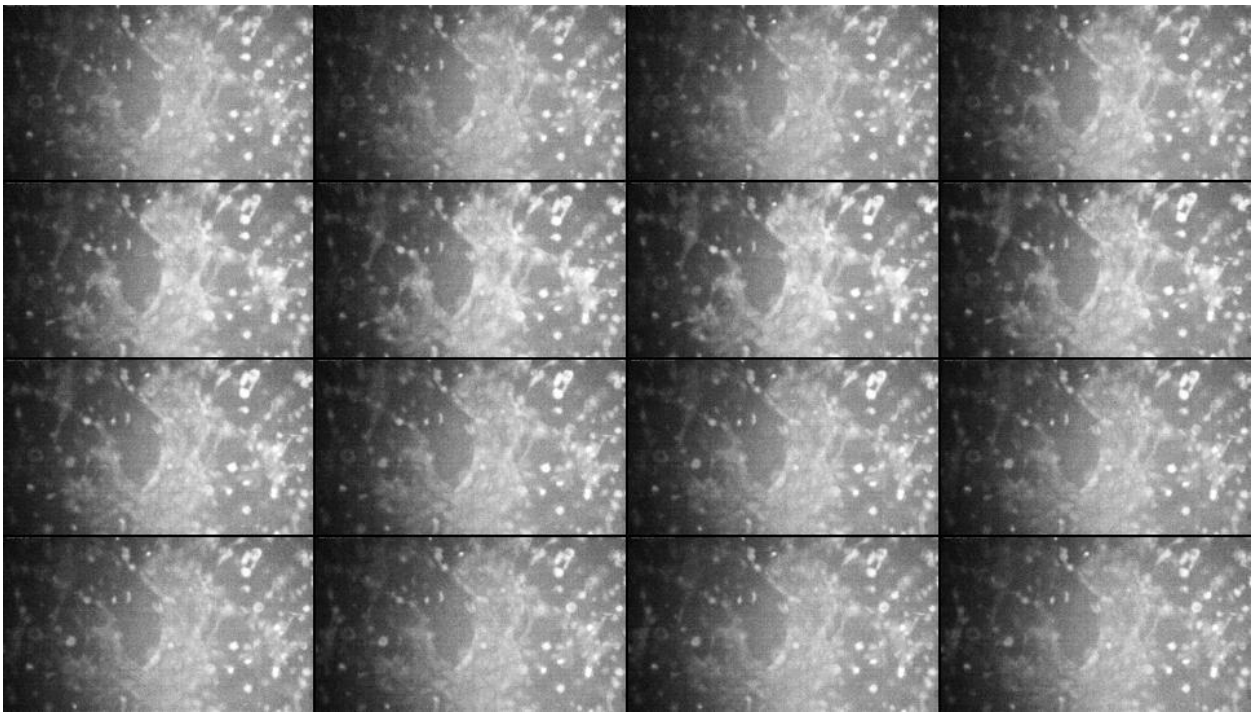

Figure 6 - Still frames from the Fluo-4 stained heart cell video. Every sixth frame is shown reading from left to right starting in the upper left corner.

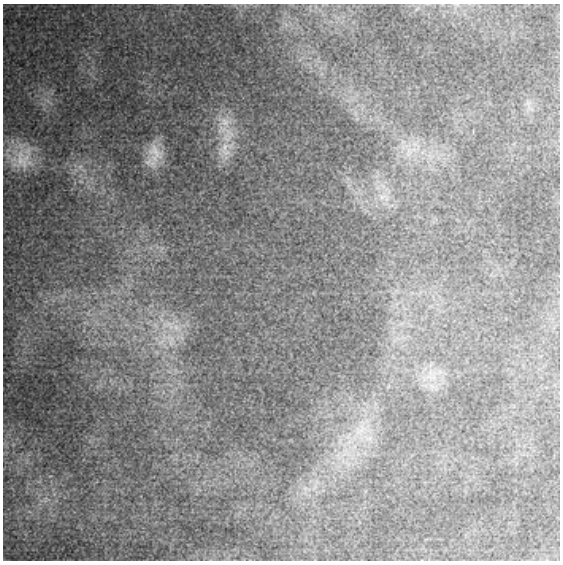

Figure 7 - crop from first frame showing noise pattern

Here we have intentionally chosen a difficult case in which we have only 100 frames and one action potential cycle from which to find our PCs. With more frames, the algorithm will have an easier time separating structured data from random noise.

As a quick way to check that our software is working as it should, we can look at the standard deviations of the residuals at different truncation parameters. We see the expected result in Figure 8, which is that the standard deviation of the residuals decreases monotonically as the truncation parameter increases. We are getting better and better approximations of our noisy video. To avoid an overwhelming amount of data, we have chosen to examine the residuals of only the first 10 images.

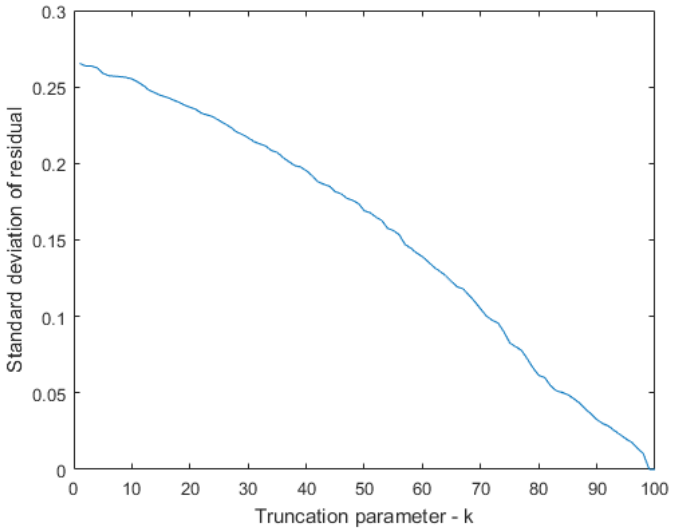

Figure 8 – Standard deviation of the residual array at different truncation parameters k. Using more components decreases the residual error.

Next, we look at the mean of the absolute value (the 1-norms) of deviations of the residual NCPs from a straight line running from zero to one on the ordinate.

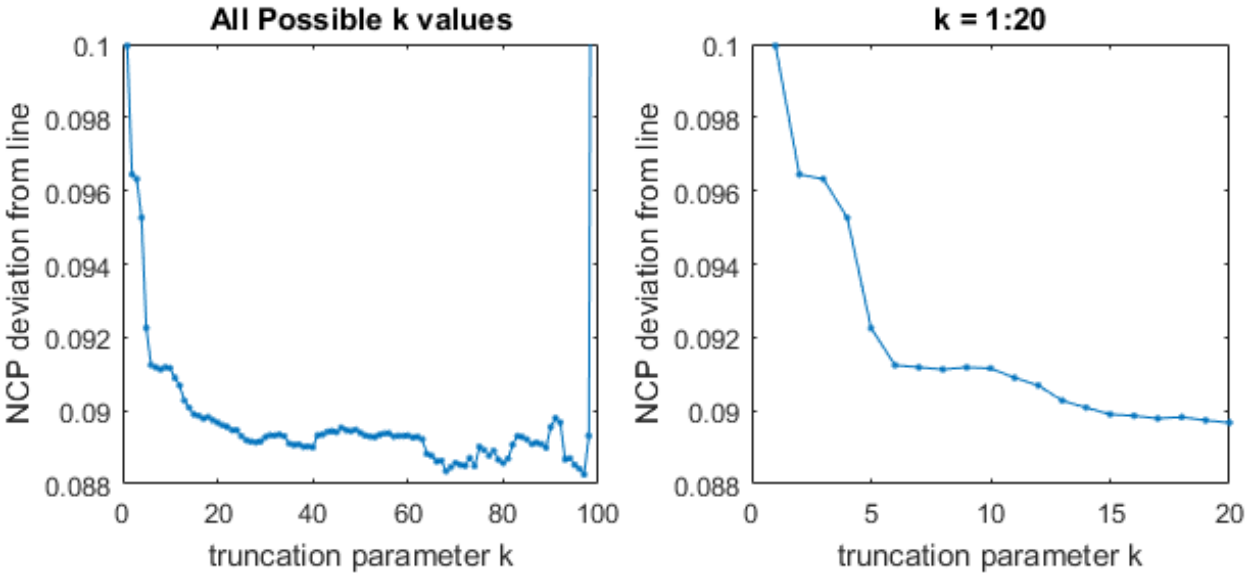

Figure 9 – Mean absolute deviation of the NCP of the residuals of TSVD reconstructions from a straight line. Lower values indicate the residual is more noise-like.

In Figure 9, we see that the mean absolute deviation of the residual NCP has more complex structure than the plot of the singular values. Whereas the singular values decrease monotonically (by definition) and quickly level off, the deviations show that the residuals become more "white noise like" up to k=27 before starting to fluctuate. If this method worked ideally (and we could perfectly separate data from noise), we would expect an absolute minimum where the residuals were most "white noise like" before high frequency dominance of the residuals caused the deviation plot to increase. *Note that the residual with k = 100 is purely rounding error.*

Figure 10 shows that the residuals are nowhere close to white noise according to our NCP of spatial frequencies evaluated by Kolmogorov-Smirnov limits. All of the NCP plots are convex up, indicating dominant lower spatial frequencies. This is the expected result for lower

truncation values, but it does not show the expected shift to high frequency dominance in later values of k. In fact, there is really very little qualitative change in the graphs of any of the residuals.

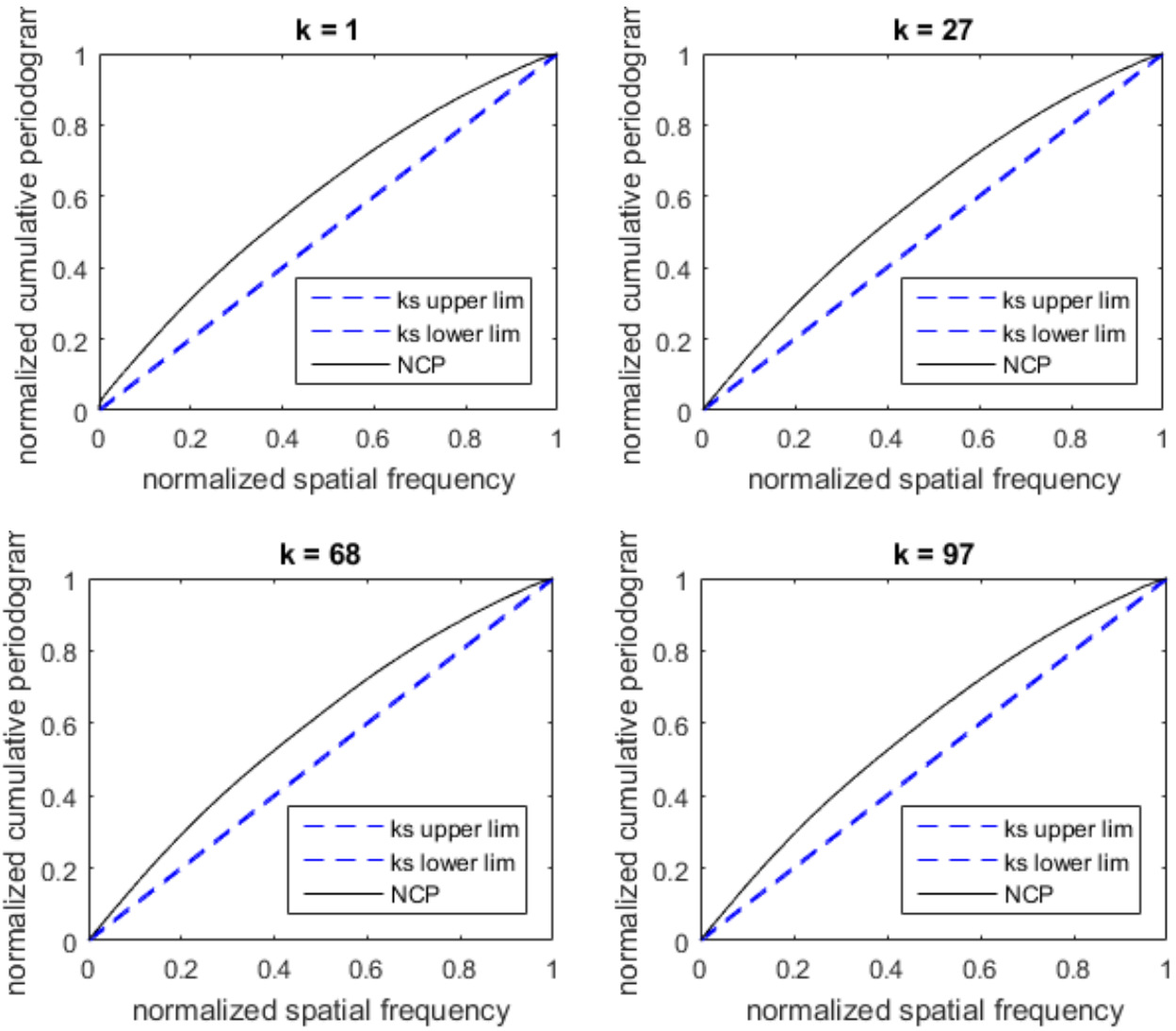

Figure 10 – NCP of the residual $\boldsymbol{A - A_k}$ *at four different values of k. KS limits are so close here they appear as one line. The NCPs for all residuals are well outside of the KS limits. There is little qualitative difference between them.*

One explanation for the results in Figure 10 is that our noise model is innaccurate. Looking at the original video file, we see that there are areas where the video is *saturated* – the values are equal to the maximum uint8 value of 255. We would not expect to see the same noise pattern in

these areas if we see any noise at all. This creates low noise areas in the residuals, adding to low frequency information in the DFT. There may well be other issues with our noise model and the assumption of flat frequency spectrum noise.

Qualitative inspection of the residual video supports the idea that there may be periodic information in the noise. Residuals were viewed as a video by reshaping them into standard video array format, setting the lowest value to zero and the highest value to 1 (double precision format). With truncation parameter *k* set to 1, we see moistly noise with lighter and darker areas in quiescent frames (little to no movement), and we see distinct outlines of moving features in frames with movement (Figure 11). By k=5, we see mostly noise in the residual even in active frames (Figure 12). However, we still see ghosts of our video that show up as slightly lighter and darker areas in the video as well as apparent variations in the noise levels (smoother looking areas). Beyond k=5, this trend continues and the residuals show mostly noise. We do not include more examples as they will appear on the page as noise.

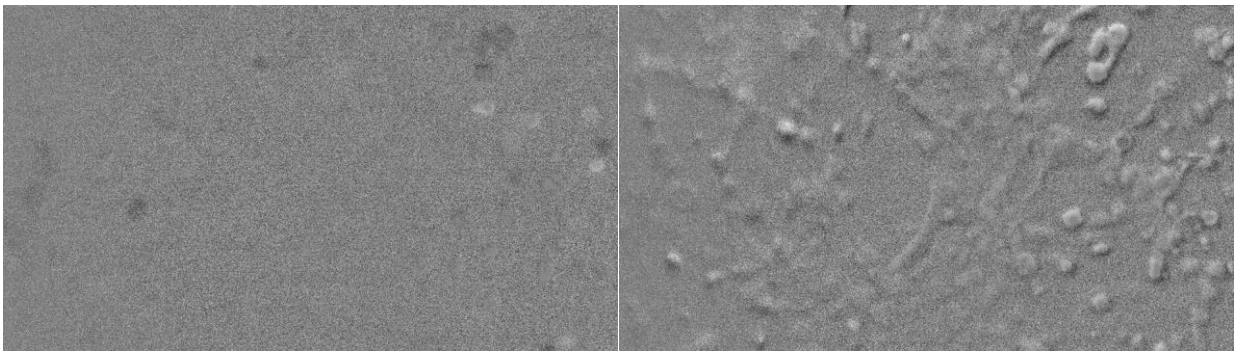

Figure 11 – Two residual frames at truncation parameter k = 1. Left: quiescent residual frame (#1), Right: active residual frame (#46)

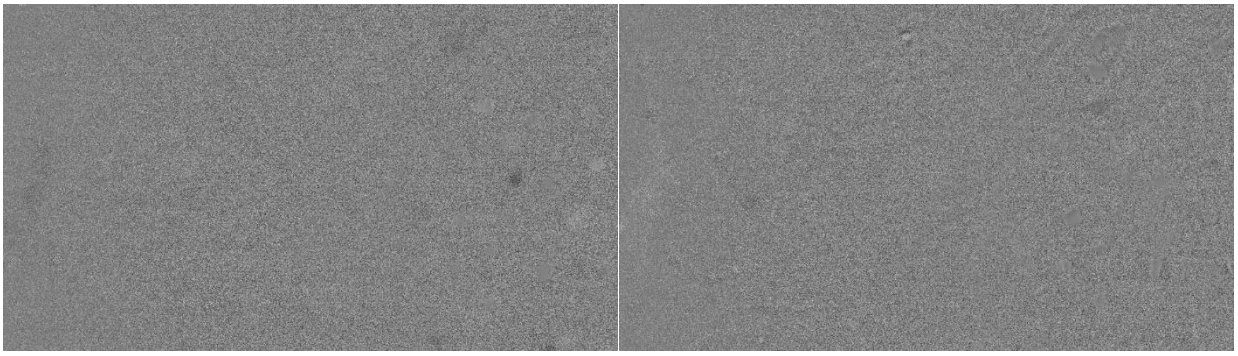

Figure 12 – Two residual frames at truncation parameter k = 5. Left: quiescent residual frame (#1), Right: active residual frame (#46)

3.7 Another Approach: Look at the V matrix

We have seen in the example above that analysis of the NCP of the 2D residuals does not give a clear indication of how noise-like our residual is in this case due to spatial variations in the residual. The next question is, can we determine how noise-like the residual is in time. One way to do this would be to look at the values of the residuals of certain pixel locations over time. These are the same length as the video itself, and there are as many of them as there are pixels per frame. However, perhaps there is an easier approach for what is still only a first best guess at a truncation parameter: look at the columns of the V matrix.

If the higher order principal components do indeed reconstruct the noise of the original video, then we would assume that the coefficients of those components would also be characterized by a noise-like frequency spectrum. We see an arbitrary sampling of coefficient vectors from V in Figure 13. As we would expect, the lower components show clear structure for the reconstruction of low frequency events, whereas the higher frequency components appear to become dominated by noise.

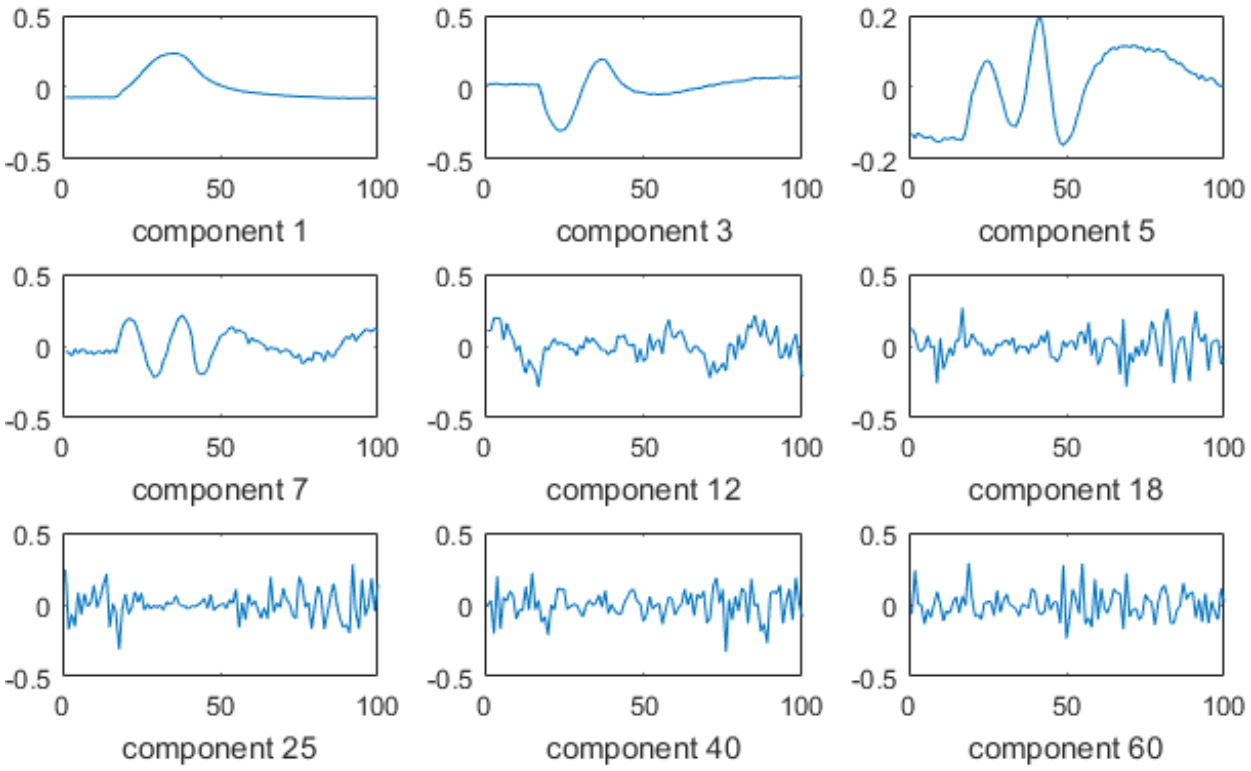

Figure 13 – Plots of nine arbitrarily selected columns of V, showing the trend of lower-number columns showing low-frequency structure, and higher-number components showing higher frequencies and noise.

In Figure 14 we see an overview of the mean absolute deviations of the NCPs of the columns of **V** from the straight-line distribution. Points labeled with an X fall outside of the Kolmogorov-Smirnoff (KS) limits at 95% certainty, and points labeled with an O fall within the limits. We see that the first 13 columns have relatively high deviations and all fall outside of the KS limits, indicating that these columns are not noise-like (Figure 15). From columns 14 through 71, all but one of the columns have deviations within the KS limits (Figure 16). From columns 72 through 100, only about one-third lie within the limits (Figure 17).

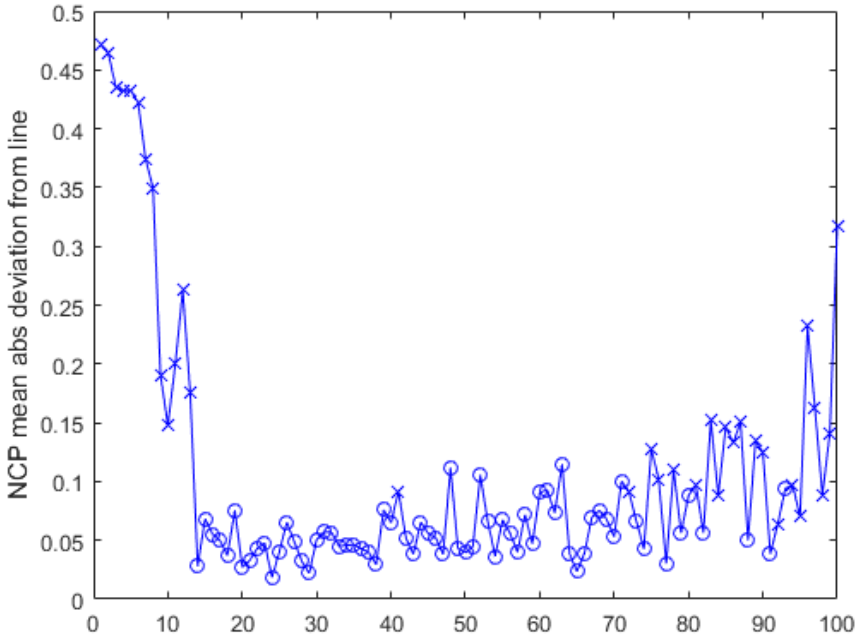

Figure 14 - Mean absolute deviation of the NCP of the columns of V from a straight line. Lower values indicate that the column is more noise-like. The first column within the KS limits (marked by O) is 14.

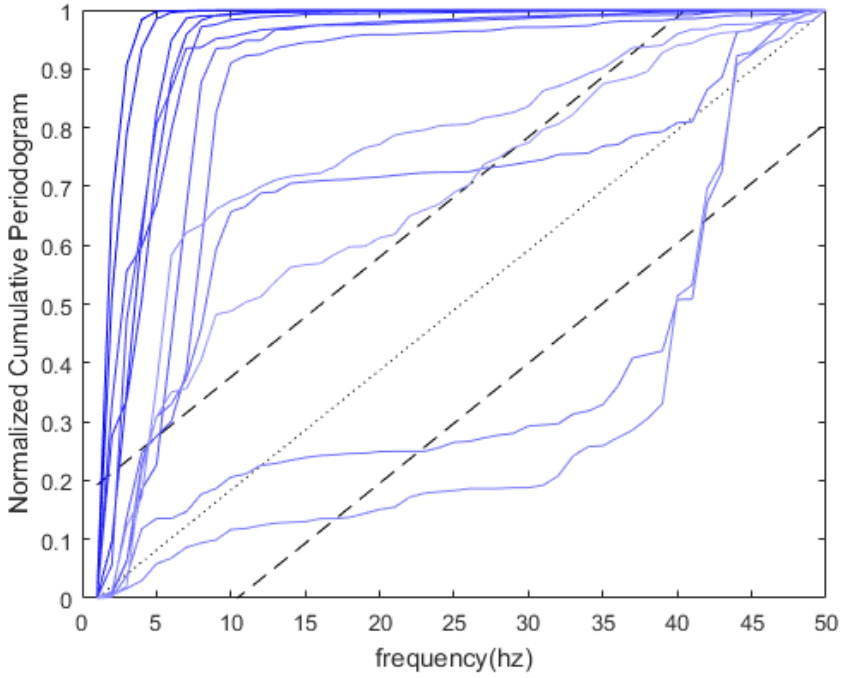

Figure 15 - NCP plots of the columns of V from 1 to 13. All NCPs fall outside of the KS limits (dashed lines), indicating that they are not noise-like. All NCPs are low-frequency dominated except 10 and 11, which are high frequency dominated. Plots go from dark to light blue by number.

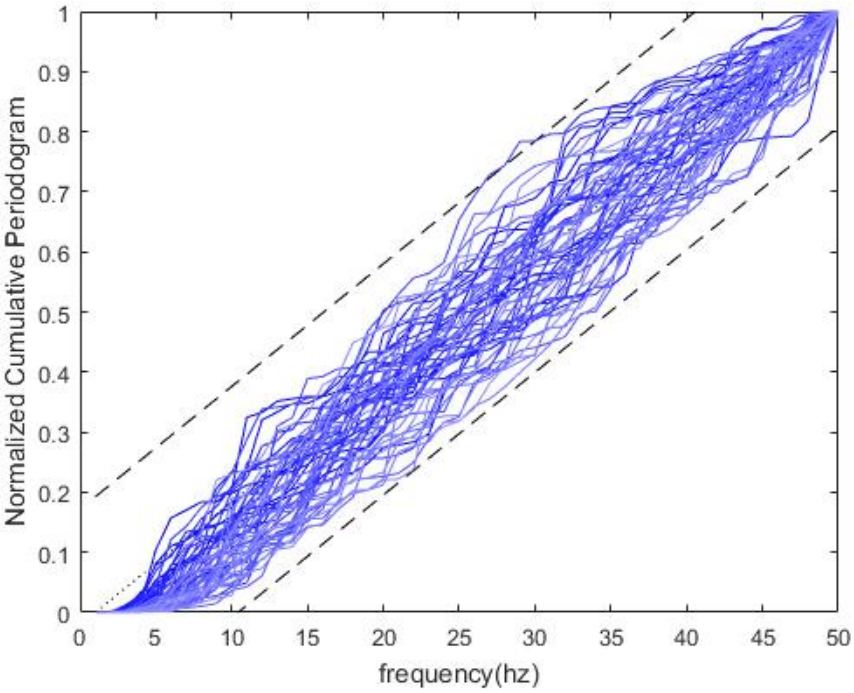

Figure 16 - NCP plots of the columns of V from 14 to 71. All but one NCP falls inside of the KS limits (dashed lines), indicating that they are noise-like. Plots go from dark to light blue by number.

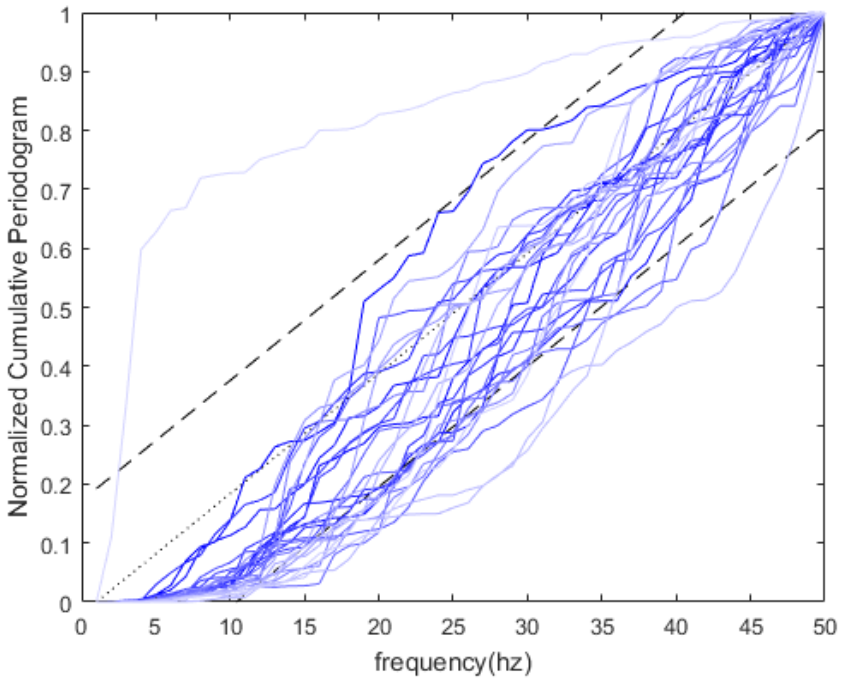

Figure 17 - NCP plots of the columns of V from 72 to 100. 9 of 29 plots lie within the KS limits. Plots go from dark to light blue by number. The low-frequency heavy plot to the upper left is the NCP of column 100.

If we are simply looking for a truncation parameter k, Figure 14 makes a strong argument for k=13. None of the columns 13 or below have a noise-like NCP, and most of the columns above 13 do, including all of the next 27. Interestingly, we do not see any significant change in the singular value going from component 13 to 14 in Figure 5. This may be because the noise levels are such that noisy components contribute as much variance to the signal as some of our data-dominated components do. This makes data-dominated components and noise-dominated components impossible to separate by looking at singular values alone.

3.8 Qualitative Analysis of the Components

We can get a qualitative sense of the differences between components by viewing them as images. Here we take the columns of U, reshape them into the dimensions of the images, and scale the values from 0 to 1 so that they are viewable in a standard MATLAB image viewer (such as *imshow()*). This analysis is admittedly subjective, but may inform the interpretation of quantitative measures.

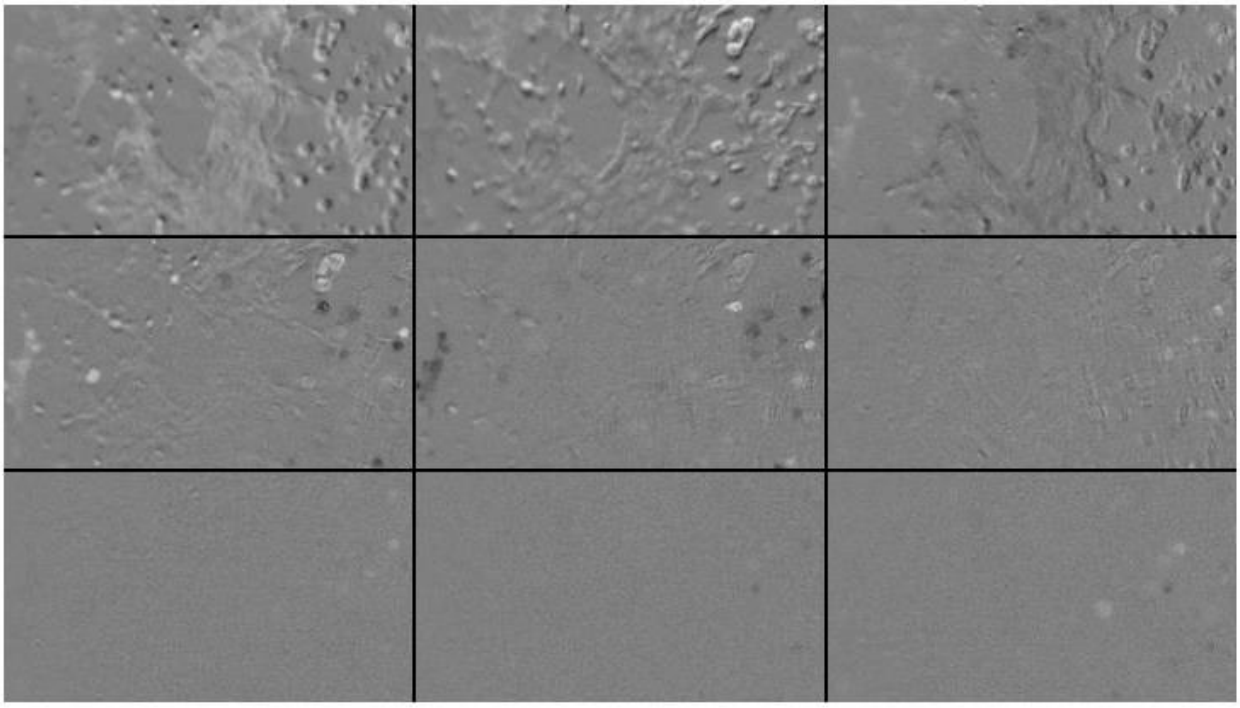

Figure 18 – Top row: PC 1, 2, 3 show clear structure and are the 3 most important components. Second row: PC 5, 6, 7 still show some structure and have dark and light spots corresponding to flashing cardiomyocytes. Bottom row: PC 10,11,12 show no clear edges, but have faint light and dark spots corresponding to flashing cardiomyocytes.

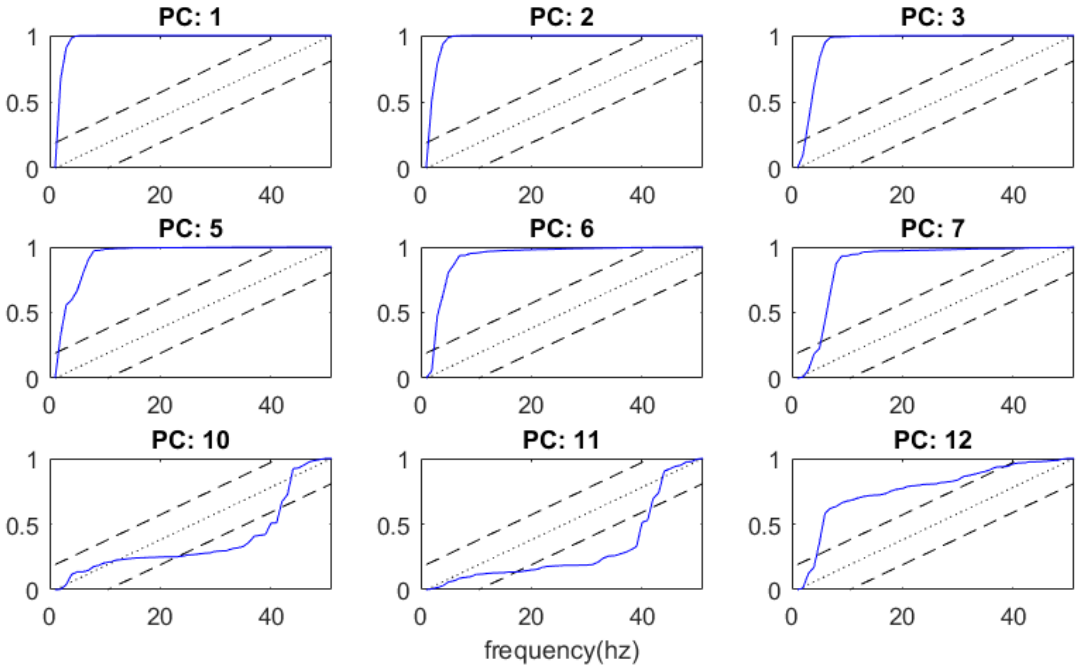

Figure 19 – NCPs of the right singular vectors. PCs 1-7 have strong low-frequency bias. PC 10 and 11 have strong high-frequency bias, PC 12 has strong low-frequency bias.

In Figure 18, the first three components show clear structure of features from the video (clumps of cardiomyocytes). These had the most low frequency biased NCPs and the greatest NCP deviation from a straight line. Components 5 to 7 also show clear structure and dark and light spots corresponding to flashing cardiomyocytes. These had similar NCPs to the first three, but with less low-frequency bias and smaller deviations. Components 10 and 11 appear to have less defined and intense bright spots than component 12, even though they were ranked higher (the singular values are close). Components 10 and 11 had high-frequency biased NCPs whereas component 12 had a low-frequency biased NCP(Figure 19).

In Figure 20 we show PCs 13 through 15, because our NCP criterion suggested a cutoff at k=13. It may be difficult to discern at the scale the images are shown here, but there are more highly pronounced dark and light spots among the noise in PC 13 than there are in PC 14 or 15. By PC 16 and above, we see a sort of textured noise with very little in the way of light or dark spots and no discernable features. There are "smooth spots" where the noise appears to be reduced (presumably due to bright or blown out areas in the video). There are occasional faint variations in the overall brightness at higher components, but they are much subtler than what we see even in PC 13.

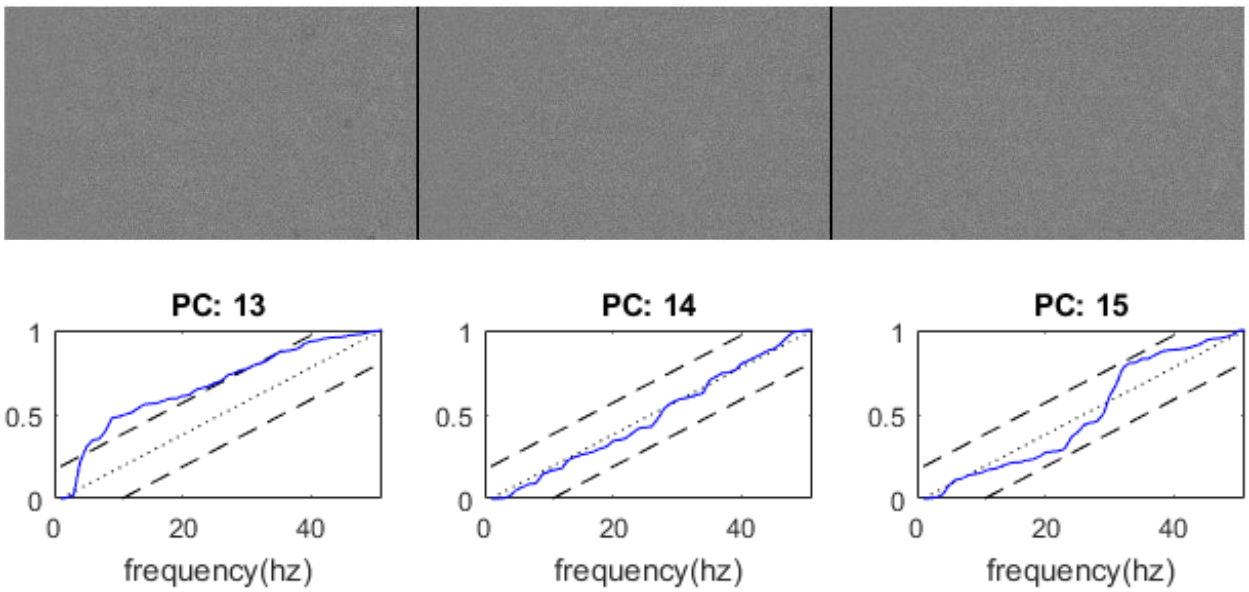

Figure 20 – From left: PC 13, 14,15. Using our NCP of V criterion, the coefficients of PC13 were not noise-like, but those of PC14 and PC15 were.

Figure 21 shows the same cropped region from PC 1,2,3, and 10. We see very low noise in PC1, moderate noise in PC2, and fairly heavy noise in PC3 and above. PC10 is included to show how quickly the components are dominated by noise.

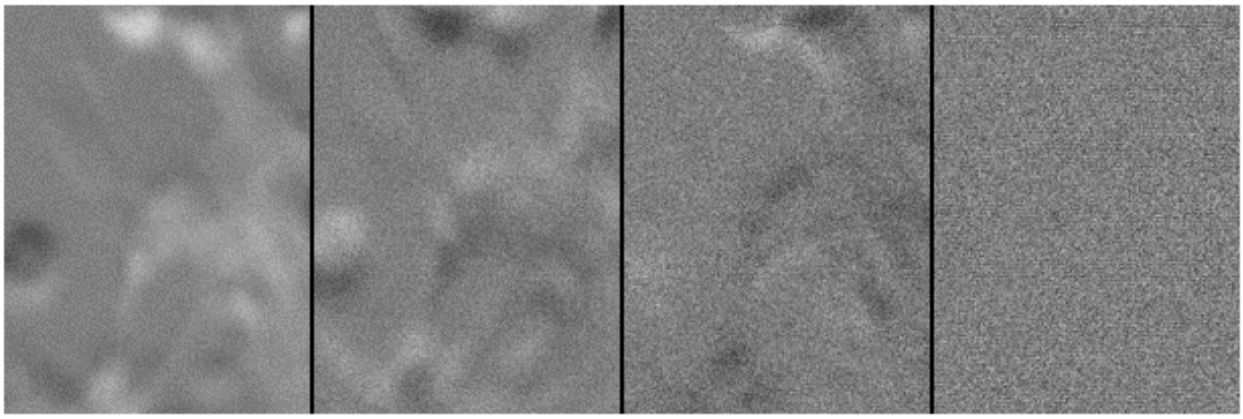

Figure 21 – From left: The same image area cropped from PC 1,2,3,10 to show noise levels. PC1 is smooth, PC2 shows moderate noise, PC3 is noisy but with features, PC10 is dominated by noise (characteristic of later PCs).

3.8.1 Qualitative Analysis of TSVD Reconstructions:

Here again we are aware of the subjectivity of the current analysis. However, because we are interested in reconstruction of video, which is intended to be viewed, we feel it is appropriate to analyze the results visually. Certain aspects of these reconstructions such as flickering regions, apparent noise, edge artifacts (such as dark areas around moving objects), non-smooth transitions during deformations, and apparent sharpness are difficult to quantify. This analysis is intended to inform the interperetation of quantitative methods being evaluated in this work.

Evaluation of reconstructions with several truncation parameters *k*:

k =6: This is the value suggested by the singular values in Figure 5, where the first six singular values were above the plateau line. This reconstruction is visually excellent. There are only very slight artifacts around moving cells that show up as slight dark spots. Noise levels are very low. It is not obvious to the eye that there is information missing from the reconstruction. The residuals show flat spots in the noise where the cells are bright. They also show some edge differences during movement and some blinking of the cells which was not captured in the reconstruction.

k =8: These are the PCs whose associated right singular vectors had NCPs of greatest deviation from noise (Figure 14). They were all low-frequency dominated. This reconstruction is visually excellent as well. There is some slight fluttering that appears around one small bright cell. Residuals show the same smooth spots, some cell blinking, and less edge difference.

k =13: This is the value suggested by the KS limits when looking at the NCP of the columns of V. The first 13 PCs were outside of the KS limits, though some were high-frequency dominated. At this truncation, we are starting to see some reconstruction of the dynamic noise. There is an

overall flickering noise pattern. Some cells with irregular or out of sync blinking patterns are starting to visually flicker. In the residuals, the slow blink of cells not captured in the lower k value reconstructions has turned into a faster irregular flicker.

k =13, minus PC 10,11: Here we will try a selective SVD where we exclude the PCs that showed high frequency dominated NCPs in Figure 19. This reconstruction does show a slight reduction in the overall noise pattern seen in the full k=13 reconstruction. However, cell flickering issues do not seem to be improved.

k =27: This value is tried because of an early local minumum in Figure 9. Some noise has been added back to the original video and it appears somewhat noisy, though still much cleaner than the original. There is an overall flickering quality as only some of the noise components have been added back. Some of the blinking cells are flickering rapidly.

k =40: this value is tried because a local minimum in Figure 9. By this point, much of the noise has been put back into the video, though it is still visually reduced. Flickering issues are not visually problematic. Out of sync cell blinking is well captured. Residuals appear to be mostly noise with some flat spots of low noise. There are occasional flashes in a few CM locations.

3.9 Testing a Synthetic Video with Noise Added

To test the hypotheses posed in the analysis of our cardiomyocyte fluorescence video, we constructed a synthetic, noise-free, video so that we could add, and then try to remove, noise from the video using the SVD. The video contains both deformation and intensity value changes. Intensity value changes were set to lead the deformations by .2 seconds. We then added Poisson and Gaussian distributed noise (in that order) using *imnoise()* in MATLAB. The pixel intensity value (0-255) is used as the mean value for the Poisson distribution. The default

Gaussian variance of .1 gave noise levels for our uint8 format video (the same format we obtain from our tiff videos) that appeared comparable to those in our CM video.

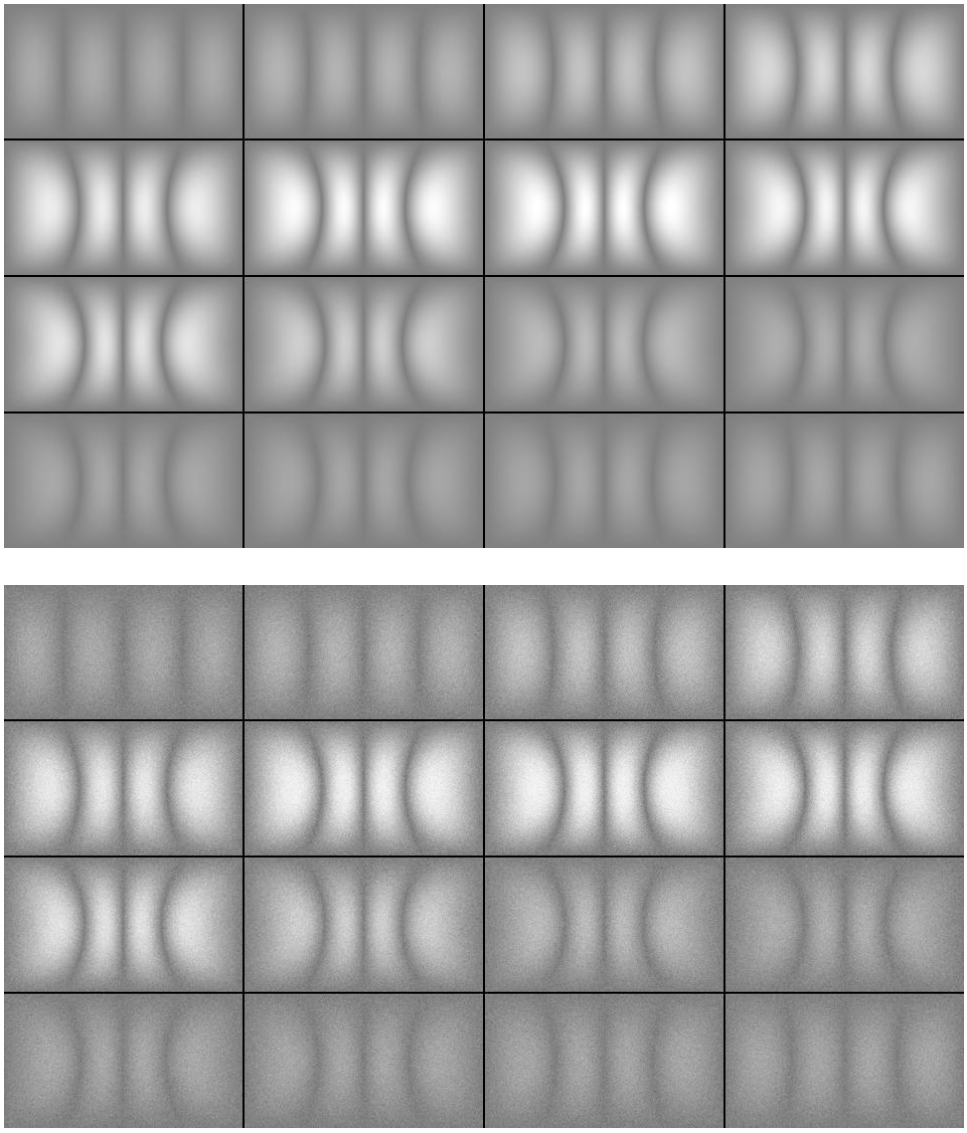

Figure 22 – Top image shows still frames from the synthetic test video. Reading from left to right, every sixth frame is shown starting from the upper left corner. The bottom image shows frames with Poisson and normally distributed noise added. The visual differences are not significant at this scale.

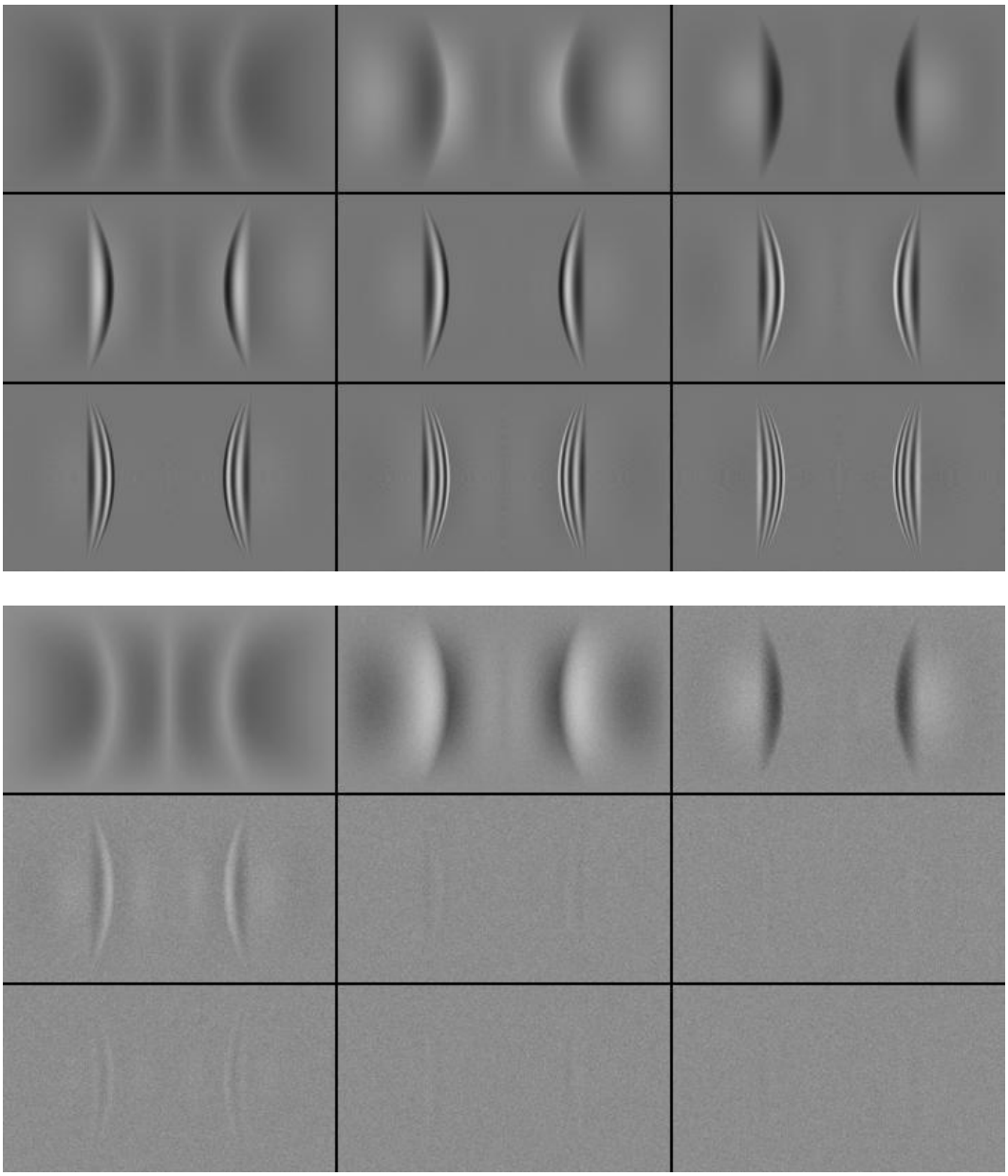

Figure 23 – The first 9 basis images (principal components) for the clean (top) video and for the noise-added (bottom) video. The first four noisy components are visually similar to the first four clean components (the inversion of PC 2 and 4 is arbitrary). Components of the noisy video above PC 4 become dominated by noise, though we see faint elements of structure remaining.

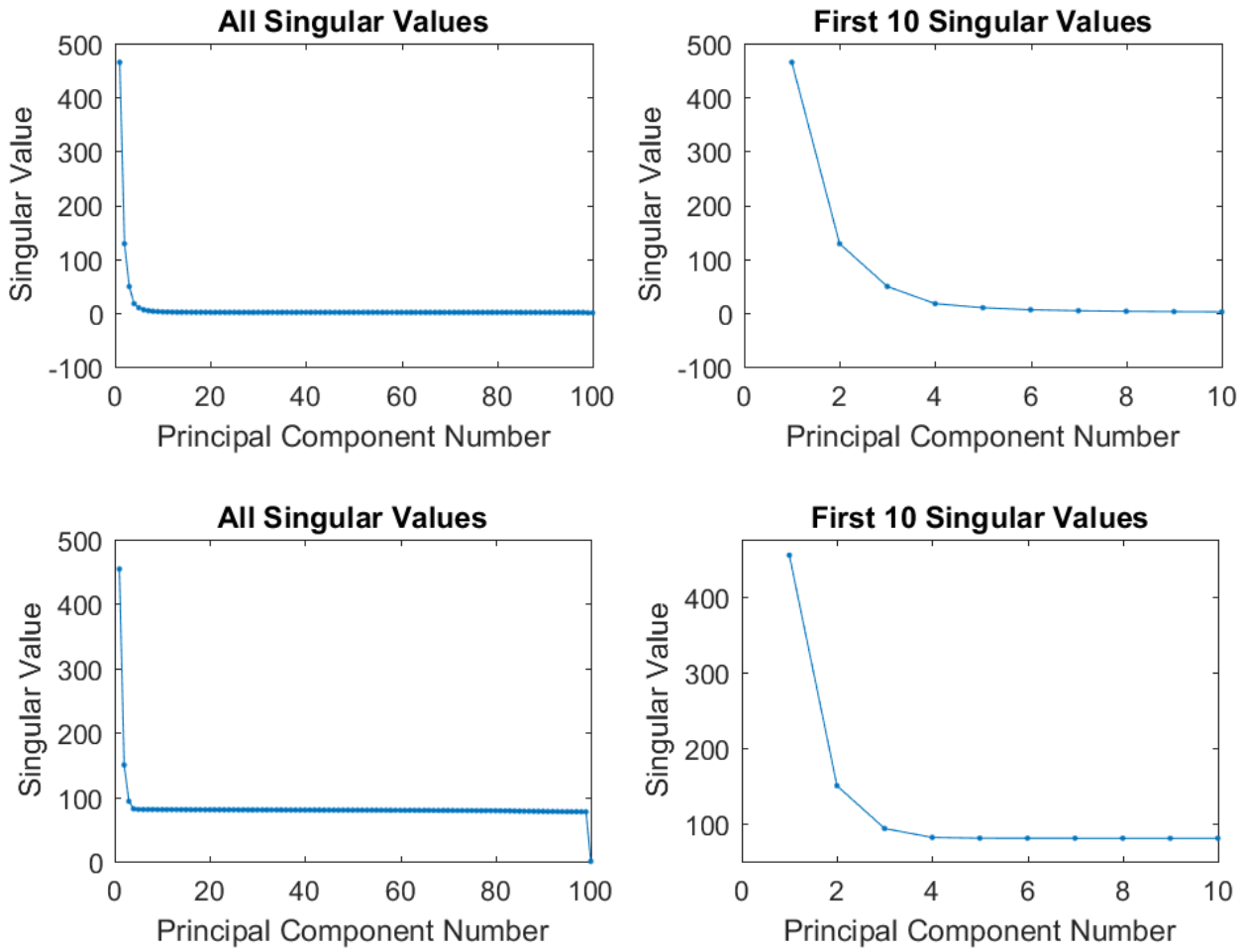

Figure 24 – Singular values for the synthetic video. Top row: Singular values for the noise-free video decay to a value of 1.8 by PC 10, and slowly decay to .6 at PC98 (PC 99,100 are over 12 orders of magnitude smaller). Bottom Row: Singular values for our noisy video have leveled off by PC4 to 81.5 and decay to 77 by PC99 (PC100 is effectively zero).

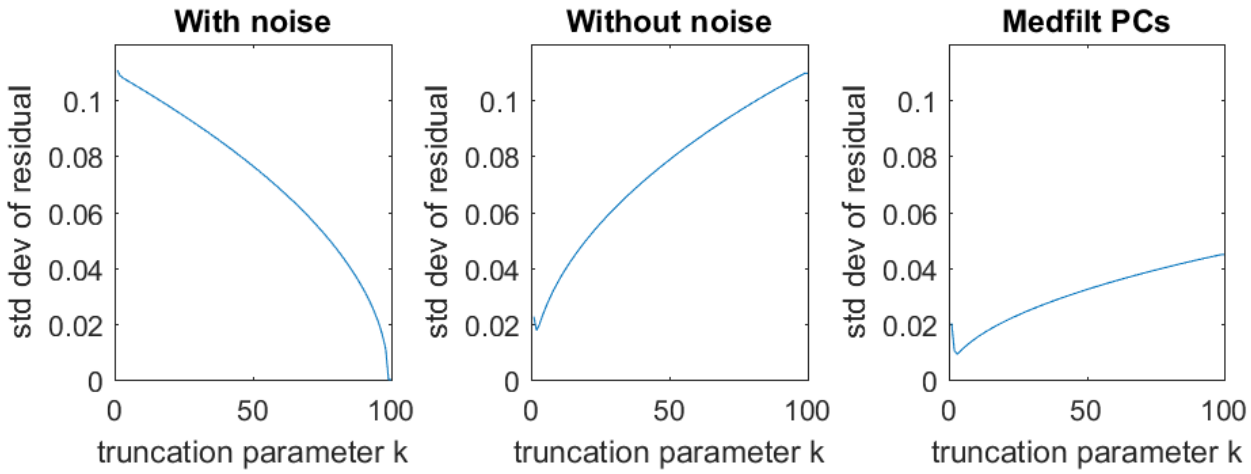

Figure 25 – Standard deviation of the residuals at all truncation parameters k. Left: residuals of all truncated reconstructions of noisy video. Middle: residuals of noise-free video minus truncated reconstructions of noisy video. Low value (.018) is at k = 2. Right: residuals of noise-free video minus reconstruction of noisy video using median-filtered PCs. Low value (.009) is shifted to k = 3.

Figure 25 shows the expected result for the standard deviation of the residuals when the TSVD of the noisy video is used to reconstruct the noisy video. In the middle plot, when finding residuals using the noise-free video and the TSVD of the noisy video, we see a minimum value at k = 2, after which the residual increases monotonically. Adding more components only takes us farther from the original video by the least-squares evaluation. This suggests that the best TSVD reconstruction comes from only the first two components. While this is true in the least-squares sense, there may be crucial information missing or artifacts introduced by the low truncation value. It may be worth balancing the cost of additional noise with the benefit of extra information gained, which is hard to quantify, and may have to be evaluated qualitatively.

The right-most plot of Figure 25 shows the standard deviation of the residuals obtained by subtracting from the noise-free video a TSVD reconstruction of the noisy video where the individual basis images have been median filtered. The overall standard deviations are lower, which is not surprising since we would expect a filtered video to be closer to our noise-free video by least-squares. The interesting result is that the optimal truncation parameter has been shifted

from two to three. The third PC went from adding to the residual to reducing the residual. This raised the question of whether pre-filtering the components before reconstruction had an advantage over post-filtering the reconstructed array. To answer the question, we ran another residual analysis where post-filtered reconstructions were subtracted from the clean video array. The result (not shown) was almost identical to the right-most plot of Figure 25. Both filtering methods shifted the lowest residual from k = 2, to k = 3. This result is useful, because it suggests that if we wish to combine the TSVD with traditional filtering methods (even non-linear median filtering), we may be able to obtain similar results by pre-filtering a few components rather than post-filtering an entire video.

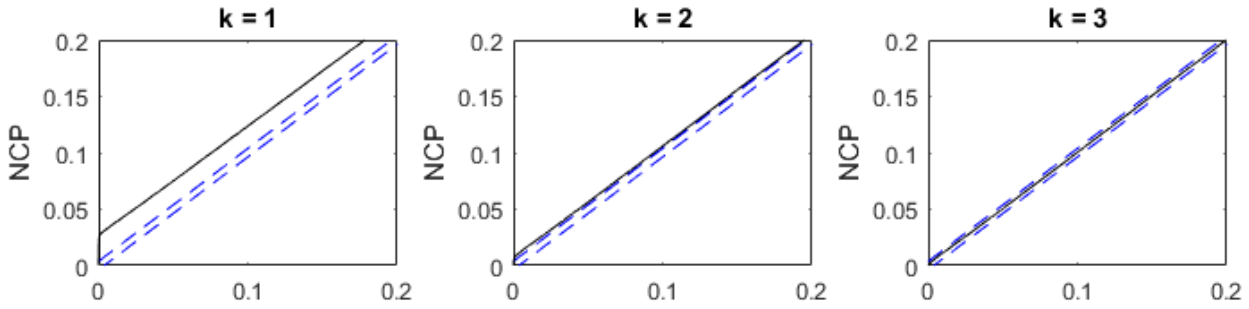

Figure 26 – NCP of the residual of the noisy data minus the noisy reconstruction. Reconstructions with k = 1 and k = 2 fall outside of KS limits, reconstructions for k = 3 through k = 98 fall within KS limits. We show a corner of the plot to improve readability.

For the 2D NCPs of the residuals using the noisy video and the noisy reconstructions, the first two NCPs, corresponding to k = 1 and k = 2 fall outside of the KS limits with low-frequency dominated NCPs (Figure 26). Residuals corresponding to k = 3 through k = 98 fall within the KS limits, indicating that they are noise-like. The residual for k = 99 falls outside of the limits (low-frequency dominated). From this information, we may choose to truncate at k = 3, because this is the *first* noise-like residual. This agrees well with Figure 25, where the best truncation was found to be at k = 2 or k = 3 (with filtering). It is worth noting how quickly the residual NCPs fall within the KS limits here, because they *never* fell within the KS limits for our

engineered heart tissue video. This supports the idea that the residual NCP analysis method is plausible, but will not work for some noise models.

If we want to find the *most* noise-like residual, Figure *27* might suggest k = 4, where we our slope is close to flat (not getting much *more* noise-like), or k = 7 where we have an early local minimum.

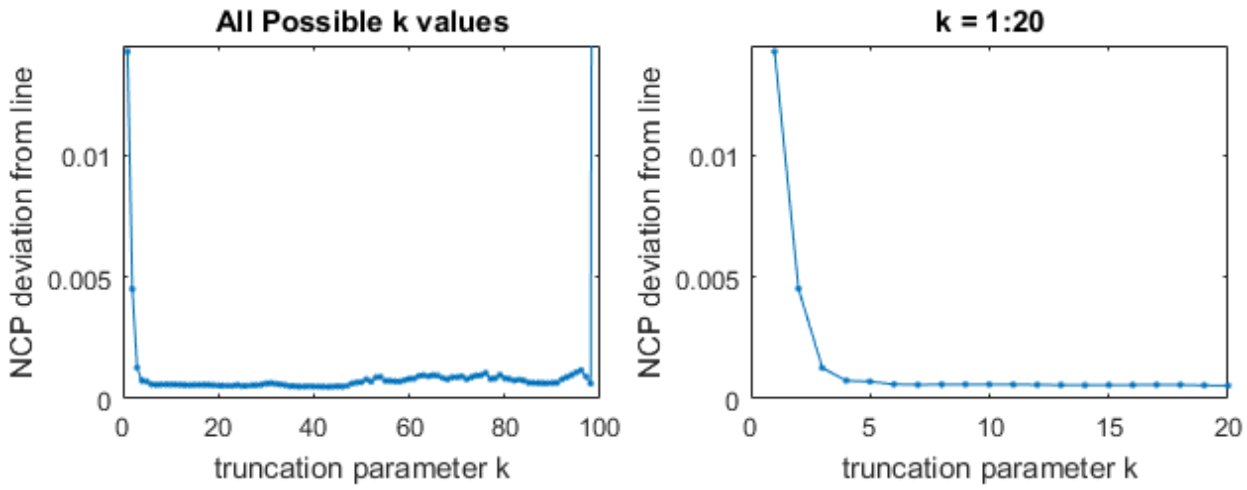

Figure 27 – 1-norm of residual NCP deviations from a straight line. The first two values are significantly higher than the rest and fall outside of the KS limits, indicating non-noise-like residuals. PC 3-98 fall within the KS limits. A local minimum is reached at k=7.

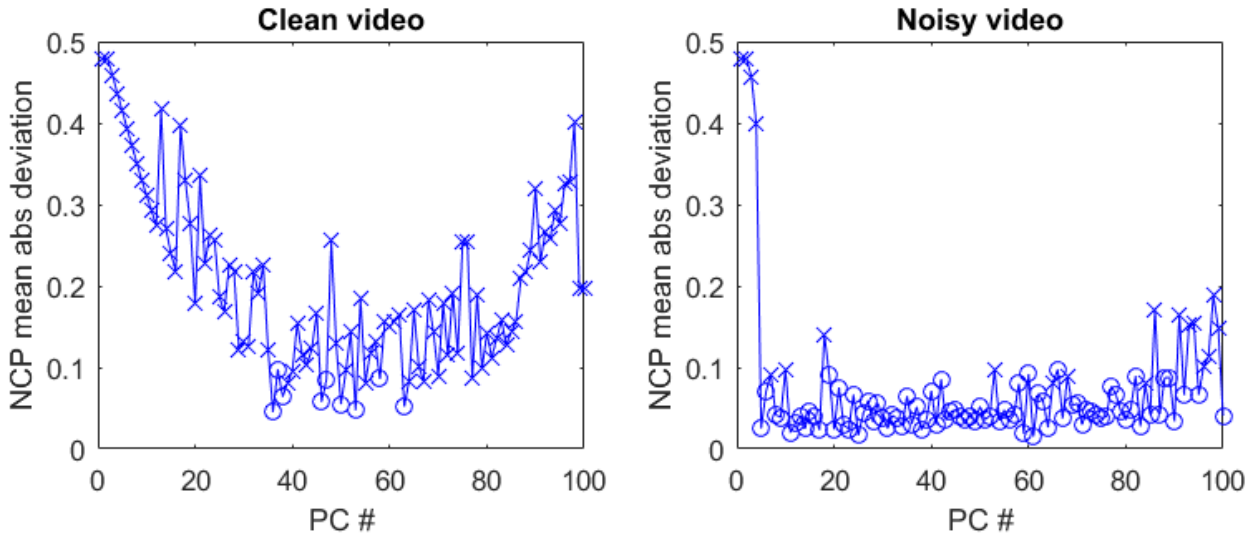

Figure 28 – Deviation of NCPs of the right singular vectors. Points marked by an X fall outside of KS limits (not noise-like), points marked with an O fall within KS limits. Most of the vectors in the clean V matrix fall outside of the KS limits. The first four vectors of the noisy V matrix are not noise-like according to this test.

The right plot of Figure 28 suggests a truncation parameter of k = 4 from looking at the NCPs of the right singular vectors. Here the first four columns of V fall well outside of the KS limits and all further columns are either within or close to within the KS limits. The first four columns are by far the lease noise-like. The left plot – deviations of the clean vector NCPs – shows a scoop shape that agrees with expectations: The earlier PCs are low-frequency dominated and the later PCs are high-frequency dominated. Both of their respective NCPs deviate from the straight line. PCs in the middle have frequency distributions that may be difficult to discern from noise (though only 9 of 100 fall within the KS limits). Overall, this plot is supportive of the idea that clean data components and noisy components will have very different coefficient NCP characteristics.

3.9.1 Qualitative Analysis of Reconstructions

The previous disclaimer about subjectivity applies here. Visually, a reconstruction with $k = 2$, is very clean and smooth and generally looks very good. It captures the changes in intensity very well. However, it fails to do a very good job of capturing the deformation of the borders between the middle two and outer two regions. These borders look crisp in the fully deformed temporal middle of the video, but are fuzzy and crescent shaped otherwise.

$k = 3$: The border shapes are better, but we see some slight artifacts during the middle of the deformation phase. We are starting to see noise now, although it is fairly static.

$k = 4$: The two borders mentioned previously are more accurately produced at several time-points in the video, but now there are bothersome artifacts causing a perception of vibration of the borders during phases of the motion.

k = 5 and above: There are troublesome artifacts in all reconstructions until around k = 50. By *k* = 50, the artifacts that appear as quivering or vibrating features largely disappear, but noise is strong. Interestingly, the noise is not noticeable in the temporal middle of the video, when the deformation is at its greatest. The components of the noise that contribute to that period of the video must be of smaller variance.

The reconstructions of the Synthetic video have visual flickering issues that are stronger than in the heart cell videos. We suspect that this may be due to the geometric nature of the synthetic video.

3.10 Analysis of a Long Fluorescence video

In the first exploration of NCP analysis applied to the SVD of a fluorescence video, we had the less than ideal situation where we had only 100 noisy frames from one action potential cycle from which to find our SVD. In that situation, the SVD was able to separate data from noise well in the first component, somewhat in the second component, and not well at all with higher number components. With a larger image set, and more action potential cycles, we expect the SVD to be able to do a better job of separating noisy components from data dominated ones.

Using 3200 frames, 32 times as many frames as the previous analysis, we see a profound difference in the ability of the SVD to separate noise from data. Even though there are theoretically 3200 PCs for this video (assuming the images are linearly independent), we calculate only the first 100. As before, we start with a plot of the singular values:

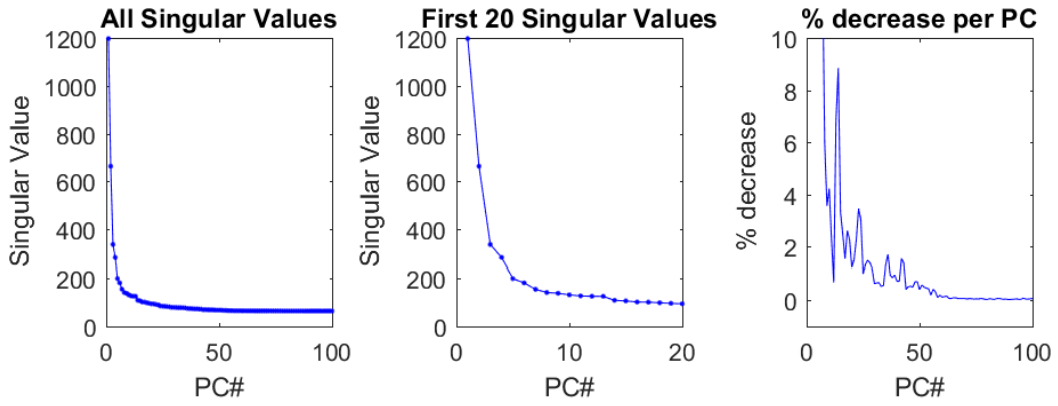

Figure 29 – Singular values for SVD of 3200 frame Fluorescence video. Here the first four are clearly apart from the rest, but singular values decay to a nearly flat line much more slowly. The percent decrease drops below 1 at PC30, and below .1 at PC 58.

The first thing we notice in the plot of singular values (Figure 29) is that they decay more slowly than in the decomposition of the shorter video. More components contribute variance above noise level. The first four clearly stand apart from the rest, but the percent decrease stays above 1% until PC 30. In the short version of the video, the percent decrease drops below 1% by PC 7.

Qualitative analysis of the PCs (Figure 30) shows that there is very little visible noise in the first 10, but some horizontal noise patterns in 11,12, and 13. PC 14,15,16 primarily show dark and light spots corresponding to flashing cardiomyocytes. In the thumbnails of the columns of V (bottom left), we see strong, steady higher-frequencies in PCs 11,12,and 13. These differences are reflected in the NCPs of the columns of V (bottom right).

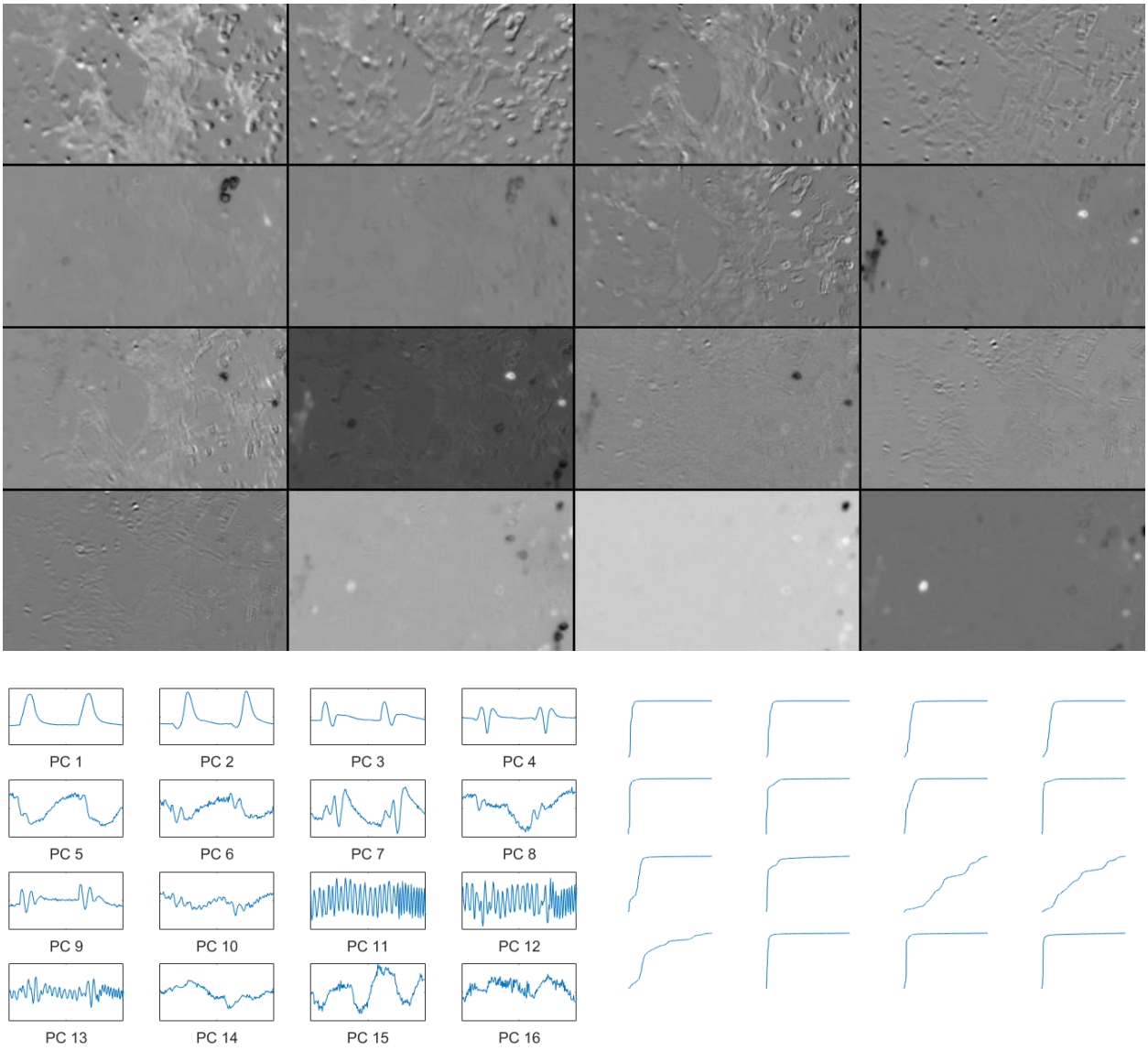

Figure 30 – Principal components(above) and their corresponding right singular vectors (below left). Coefficients from the frame 200 to 400 interval are shown. On the bottom right we show thumbnails of the NCPs of the right singular vectors to show the different spectral character of PC 11,12,13.

Looking at the deviations of the NCPs of the residuals from a straight line, the first two deviations are far above the rest and are not shown. There is a strong global minimum at k=4, suggesting that the residual at k=4 is most noise-like. The next lowest point is at k=20. Interestingly, the shape of the plot changes at k=56. Before k=56 the deviations fluctuate: adding more PCs making the residual more or less noise-like depending on the PC. After k=56, adding more PCs makes the residual less noise-like in a slow smooth progression. It is not immediately clear what happens at $k = 56$, if anything. One possible interpretation of the more predictable increase in the deviation of the residual NCP is that we are adding more uniformly noisy components above PC56 and we are leaving higher frequency filtered noise in the residuals (this is the original idea). However, none of the NCPs here lie within the KS limits. This is the same as with the 100 frame SVD shown previously.

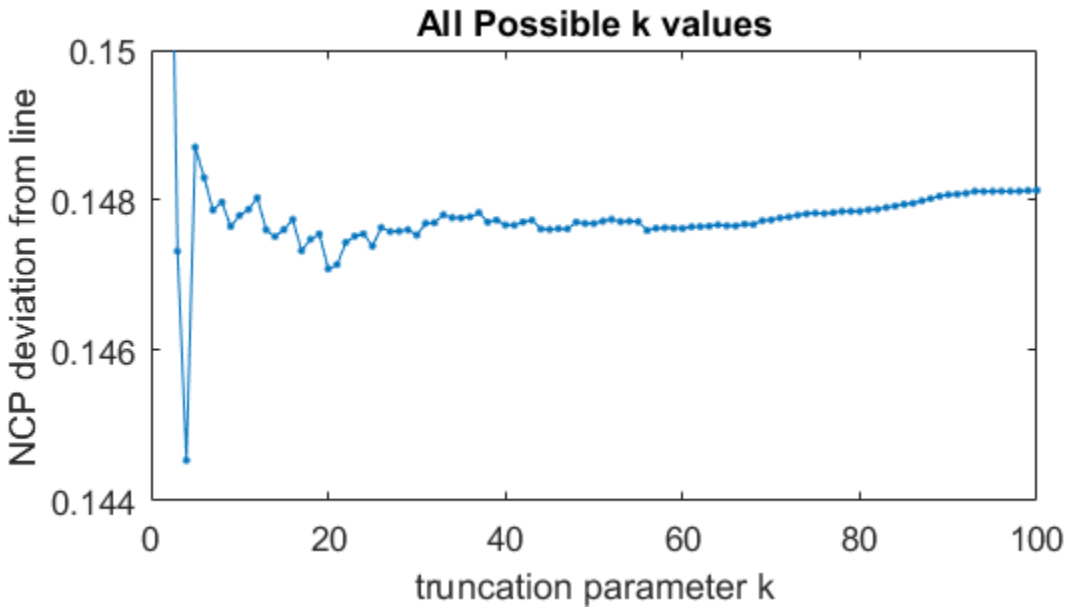

Figure 31 – Deviation of residual NCP from white noise line. Points for k=1 and k=2 are not shown. The lowest point is at k=4. The second lowest point is at k=20. Above k=56, the character of the plot changes and we see a slow, but steady increase in deviation.

In Figure 32 we can see an overall structure to the decay of the deviations as the higher NCPs become more noise-like. We see strong outliers at PC 11, 12, 13 as we would expect from the thumbnails in Figure 30. We also see strong outliers at PC 31, 32, 34, and 36. The deviations drop sharply from PC 56-60, and fall within the KS limits by PC 75. This figure suggests not only truncating the SVD somewhere in the k=50 to k=60 range, but perhaps excluding low deviation PCs such as 11,12,13; and 31,32,34, and 36.

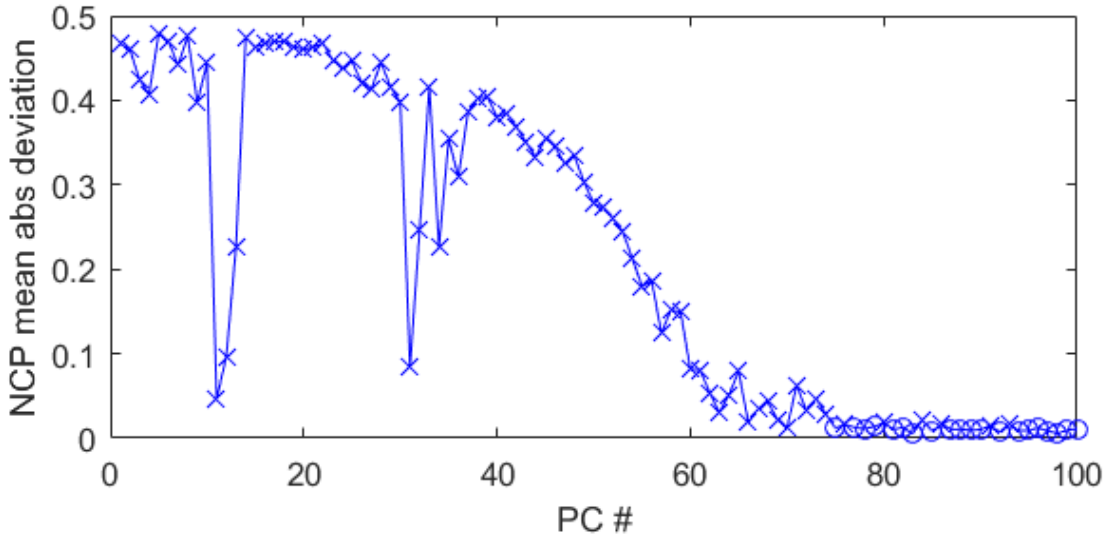

Figure 32 – Deviation of the NCPs of the right singular vectors. We see marked dips at PC 11,12,13, and PC 31,32,and 34. The first vector to fall within the KS limits is PC 75.

3.10.1 Qualitative analysis of reconstructions

Here we will examine several TSVD reconstructions suggested by the analyses above, with the same disclaimer about subjectivity given previously. The first four singular values are much stronger than the rest, and the NCP of the residuals was most noise-like at k=4, so the first reconstruction we will examine is at k=4. We will examine reconstructions by playing a vertically concatenated video of 200 frames of the original video above 200 frames of the reconstructed video. We will then examine the residual (in single precision 0-1 scale), shifting and scaling as necessary to see what features are or are not missing.

At k=4, the reconstructed video looks excellent. There is only one apparent artifact (a small movement artifact at the upper middle of the frame) and no noise. There is a static grid pattern visible from variations in column and row sensitivities of the image sensor. (Perhaps this might be corrected by subtracting a dark frame and dividing by a bright frame, as is commonly done in astronomy imaging). It is not immediately obvious that any information is missing, and the TSVD video actually appears sharper in many areas that the original video, due to the total absence of noise. The residual shows lots of noise (as expected), some edge differences during motion, but also a lot of blinking areas corresponding to out of sync CM APs. These smaller flashing areas are not captured by the k=4 TSVD reconstruction as they are not in phase with the large variance changes captured in the first few PCs.

The next lowest residual NCP deviation was at k=20, so we will examine a reconstruction with k=20 next. At this truncation, we still see excellent clarity, perhaps better than at k=4, but with slightly more noise and a slight decrease in contrast. More complex patterns of CM flashing are apparent, but there are also some flickering artifacts present, where out of sync CMs appear to flicker rapidly (much faster than possible CM AP frequencies due to the refractory period of the cell). From the residuals we see noise, less distinct low-frequency (~1-4 Hz) flashing CMs, and very little in the way of edge differences (visual edge outlines of moving features). We see some high frequency cell flickering, but it is less apparent in the residuals.

The NCPs of the right singular vectors dropped sharply at PC=56, and this is also where the shape of the residual NCP deviations changed, so we examine k=56 next. At k=56, the video quality and lack of noise is excellent, but the flickering artifacts have gotten worse. Strange flickering is apparent in many areas throughout the video and is not subtle. The residuals consist

of noise with smooth noise-free disks where some of the CMs are, and high-frequency flashing (but much less low-frequency blinking).

At k=100, the video is abuzz with flickering cells, but noise levels are still much better than in the original. With enough components, we get back to our original video, but by then we have added all of the noise back as well.

Since the NCPs suggested that there are high-frequency components in early singular values, we will now try a Selective SVD (SSVD) where we omit those components whose right singular vectors have low NCP deviations from a flat line. These include PC 11,12,13,31,32,34,36. This reconstruction exhibits the same flickering issues that were present in the TSVD at k=56. There does not appear to be an improvement.

To see where the flickering artifacts first appear, we will examine TSVDs above k=4. At k=5, we still see the small motion artifact, and we see a small amount of flickering on one cell. At k=6, we start to see some slight flickering in several other cells. From k=7 to k=9, we see similar results, with flickering becoming more constant with more components. By k=10, several cells are flickering constantly, and we see a black motion artifact behind one of the cells.

To see if the information from Figure 30 and Figure 32 is suggesting that PC11-13 are detrimental, we compare the TSVD reconstructions at k=10 and k=13 directly. The k=13 reconstruction does not appear to be worse, though it is hard to visually tell whether it is better. Looking at just a video of the contributions of PC11-13, they do not appear to contribute to flickering or noise.

An SSVD using only the highest NCP of V deviations from the first 10 PCs was also tried. These were PC 1,2,5,6,8. This reconstruction had large motion artifacts and was clearly worse than using a low order TSVD alone.

3.11 Discussion

Residual NCPs and KS limits

With the test cases we have examined here, using the NCP of the residuals does not give clear answer as to which components should be kept in a TSVD reconstruction. With both heart cell videos we examined, the NCPs never fell within KS limits. This may be related to the particular noise characteristics of that image set, because the NCP of our synthetic video residuals did fall within KS limits (and quickly). However, in all cases, the NCP deviations started out very large and dropped quickly. They may not tell us which value of k is best, but they do seem to suggest which early values to avoid. In Figure 9 (short CM video), the first 5 values of k give the clearly least noise-like residual. In Figure 27 (synthetic video), the first two deviations are far higher than the rest. In Figure 31(long CM video), the first two deviations are very high, and we actually have a distinct minimum at k=4. The only problem is that this residual NCP is not within KS limits, and there is clearly valuable video information missing in the k=4 reconstruction.

Beyond the early high deviation values, the behavior of the plots gets more complex. The two heart cell videos both show fluctuations in these values and multiple local minimums. The synthetic video shows less erratic plot behavior, but still gives a large range of values that are low and NCPs that are well within KS limits.

Additionally, the computational resources for this type of analysis are high. For every *k* value tested, we must compute a reconstruction of some portion of the video, subtract it from the original, and then analyze that residual.

NPCs of the columns of V

This method of analysis is appealing due to its computational simplicity. In the test cases above, it seems to give as much information as the NCPs of the residuals do, and in fact it does so in a way that separates the characteristics of different components more cleanly. Additionally, this method does not depend on the (possibly flawed) premise that the previous method depends on: the idea that we may separate the mostly data-dominated components from the mostly noise-dominated components by making a clean break at a certain truncation value. This opens the door to using the automatable selection criteria in this method for picking components in a selective SVD.

We have not yet demonstrated the utility of this method in a conclusive way. However, the method shows promise and may be useful if several obstacles may be overcome. In our short video, the method was able to identify the PCs with the clearest visual contributions by their low frequency dominated coefficient vectors. In the synthetic video, the difference between the NCP deviations of the clean and noisy videos is striking. The right plot of Figure 28 shows a clear identification of the 4 PCs of the noisy video that are data-dominated and resemble the PCs of the clean video.

In our long heart cell video, there is more room for interpretation. The overall structure of the plot suggests that the first 60 or so PCs are clearly not noise-like (with several outliers), and the PCs from 75 on are clearly noise-like. The only problem is that reconstructions based on these

guidelines show clear visual issues, as described in the qualitative analyses. This brings us to the final subject of our discussion.

The optimal mathematical basis may not be the best image basis.

This may not be a surprising statement, but it may be the root of several issues that we have had with denoising and video analysis. The SVD finds the orthogonal directions of greatest variance of our image set. These directions form a set of basis images. For our purposes, we would *like* these basis images to correspond to different events in our videos, such as cell contractions and intensity changes, and for the noisy parts to all lie in their own components so that they may be neatly removed. Such is not the case. The early components are the biggest and most predictable. They tend to be well-behaved but incomplete.

To get the whole picture, we need information from higher level components. These combine in unpredictable ways. A given PC may capture, for instance, a noise contribution, part of a blinking cell, and the borders of several cells at a specific deformation point. If we remove that component to remove the noise, we also lose part of the blinking of the cell and part of the edge of our deforming cells. We may have just introduced a flicker in the cell and a discontinuity in the deformation motion.

3.12 Conclusion

Evaluating residuals by their spectral characteristics to determine their resemblance to white noise is computationally expensive and may not be effective. In cases we have examined the residual never becomes noise-like by the KS limit definition. This may be because of the characteristics of the image noise itself. In the synthetic case, where the noise characteristics were known, the residual became noise-like quickly, but it did so at a reconstruction that was deemed somewhat incomplete.

NCP analysis of the V matrix is quick and easy, and gives insight into the PC contributions. This method may help to separate noisy from clean components, especially where the change in PC quality is abrupt. However, there is still quite a bit of subjective interpretation needed when the transition from data-dominance to noise is more gradual. Additionally, if desirable and undesirable contributions are tied up in the same PC, no analysis method will allow us to separate the two. Removing noise dominated components may still create artifacts such as flickering, and discontinuous motion.

3.13 Further Work

It would be worth analyzing more videos with different levels of noise. These could be both synthetic and captured videos of scientific interest. Noise levels and characteristics could be varied in the synthetic videos to see if there are cases in which the methods of PC discrimination are particularly effective.

Another residual analysis scheme used in inverse problem solving [17] that was not used here is to try to match the standard deviation of the residual with the known standard deviation of the

measurement error (noise). We may be able to get a good representation of the noise by finding a set of quiescent images (little to no movement) and subtracting the mean of those images from the set.

Ultimately, the real utility will lie in being able to form sets of basis images that are well suited to our image analysis. If we can find ways to coax our PCs into representing the information that is most relevant to us, it could be of great utility in extracting data from noisy images and obtaining relations between fluorescence reported states such as Ca^{2+} transients or membrane potentials and cell contractions.

Chapter 4: Optimized, Iterative SVD Analysis of Large Datasets

4.1 Motivation

Large video arrays are impractical or impossible to process using standard Singular Value Decomposition (SVD) algorithms in MATLAB and other software packages due to high memory usage. Although a range of schemes exists for estimating principal components from a small subsampling of the video sequence, these are impractical for general application to beating cardiac tissue, in which outlier events such as cardiac alternans or transient arrhythmias need to be captured. This is also impractical for analysis of vibrating structures in a frequency range capable of exciting multiple modes of excitation. Because of this, we have been using an incremental MATLAB SVD solver written by Robert Pless based on a paper by Matthew Brand [20] to estimate principal components from the entire video sequence. This is an iterative solver in which one column of the data matrix may be fed to the solver per iteration and the solver updates an approximation of the SVD each iteration. The major advantage of this approach, for our purposes, is that it allows finding an approximate SVD of matrices which would be far too large to decompose all at once using MATLAB's *svds()* or *svd()* functions, or other standard SVD algorithms.

This comparison and analysis was written to try to understand the differences in output we were seeing as users of both the iterative and standard solver. Some conclusions drawn by this work may not be surprising to those with an expert understanding of the software, but the analysis is

nevertheless included for completeness. A description of two processes used to find a set of right singular vectors without excessive memory usage is included.

4.2 Resource Usage

Using the standard SVD solvers in MATLAB, we are limited in the size of the arrays we can process by RAM. The CM fluorescence video used for several test cases in this document was 540 x 960 pixels and 100 frames long. At double precision, this is a 395MB array. The SVD of an array this size may be processed using a standard SVD solver on a computer with a Core-i7 processor and 16GB of RAM in a matter of a minute or two. However, the original video was 3200 frames long. At double precision, that array is 12.36 GB. This is too large to process in a standard SVD solver "all at once". In the iterative solver we were able to solve for 50 components in 859 seconds, or 100 components in 1,750 seconds. This takes some time, but it is not unreasonable. Incidentally, this array has already had its resolution reduced 50%, the original was 4 times as large!

Using the iterative solver, we may process the SVD one frame at a time. The large video file may be left in uint8 format ($1/8^{th}$ the size of double precision) and have each frame converted to double precision only as it is being fed into the solver. We don't need to convert the entire array to floating point to get a floating point SVD.

4.3 Application to a Trichome Video

Here we will compare the resource requirements of *crunchy_isvd()*, the iterative SVD solver, with the resource requirements of *svds()*, the MATLAB SVD solver, for decomposing a video of a plant trichome experiencing high frequency vibration [21]. Computations were timed using the

tic and *toc* MATLAB commands. Memory usage was monitored using *perfmon.exe*, the built-in performance monitoring application in Windows 10. The object monitored was *private bytes* (MATLAB) under *Process.* The system used is a laptop computer with an Intel Core-i7 quad-core processor running at 2.5GHz base frequency and 16GB of ram.

The baseline MATLAB memory usage recorded by *perfmon.exe* was 1.36GB. The video array was 600x800x500 (height x width x number-of-frames). The format was uint8, which was converted to double to compute the SVD.

Using MATLAB's *svds()* function to compute the truncated SVD using the first ten singular values caused MATLAB's memory usage to exceed the physical ram available almost immediately. Memory usage quickly jumped to 18GB and caused the computer to begin writing data to the swap file for about 25 seconds. Memory usage then dropped to 12GB for the remainder of the SVD calculation. The SVD was complete after 196 seconds. Maximum MATLAB memory usage was over 16.5GB above the baseline memory usage.

Crunchy_isvd() completed a truncated approximate SVD with 10 singular values in 32 seconds. Maximum memory usage was 1.5GB, which is only 140MB above the baseline memory usage.

4.3.1 Comparison of S matrix

In Figure 33, we see that the iterative SVD algorithm produces a first singular value that is within one percent of the singular value calculated by *svds()*. However, higher order singular values begin to diverge by as much as 16%. Singular values computed using the iterative method were consistently lower than those computed using the standard solver.

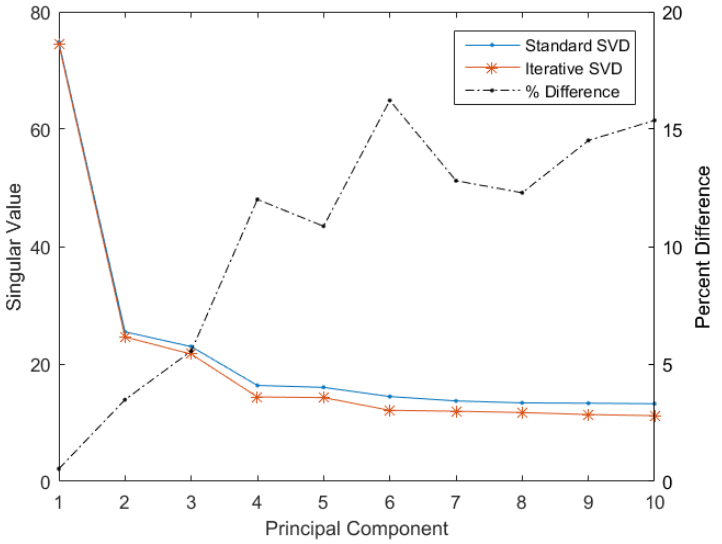

Figure 33 - Comparison of singular values produced by the standard and iterative solvers.

4.3.2 Comparison of U matrices

To visualize the principal components, we have shown the positive pixels of each principal component in red and the negative pixels in blue. Because the columns of **U** (the left singular vectors) are the eigenvectors of the covariance matrix of our original data matrix (and their eigenvalues are non-degenerate), they are unique only up to a sign flip, which is arbitrary. In comparing the principal components generated by the two SVD algorithms, we have flipped the sign of the corresponding columns of **U** where they were clearly inverses of one another (determined by inspection, but this would be easy to implement using positive/negative correlation). In Figure 34, we see the first four principal components computed by MATLAB *svds()* in the top row, and the principal components computed by *crunchy_isvd()* in the bottom row. The values have been rescaled to make the components visible.

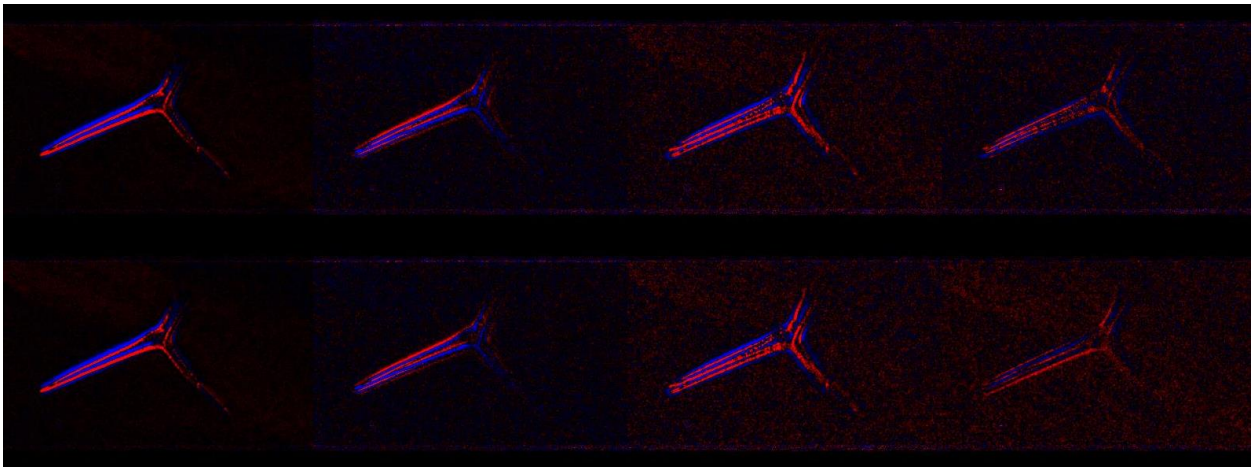

Figure 34 –Principal components 1-4 in a trichome vibration video. PCs obtained by svds() are shown in the top row, PCs obtained using crunchy_isvd() are shown in the bottom row.

In Figure 34, we see that the first 3 PCs are visually nearly identical between the iterative and standard solutions, and that the fourth component begins to diverge. In Figure 35, we see that iterative PC 5-8 diverge somewhat from the standard PCs. Both iterative and standard PCs begin to be dominated by noise starting with PC 7.

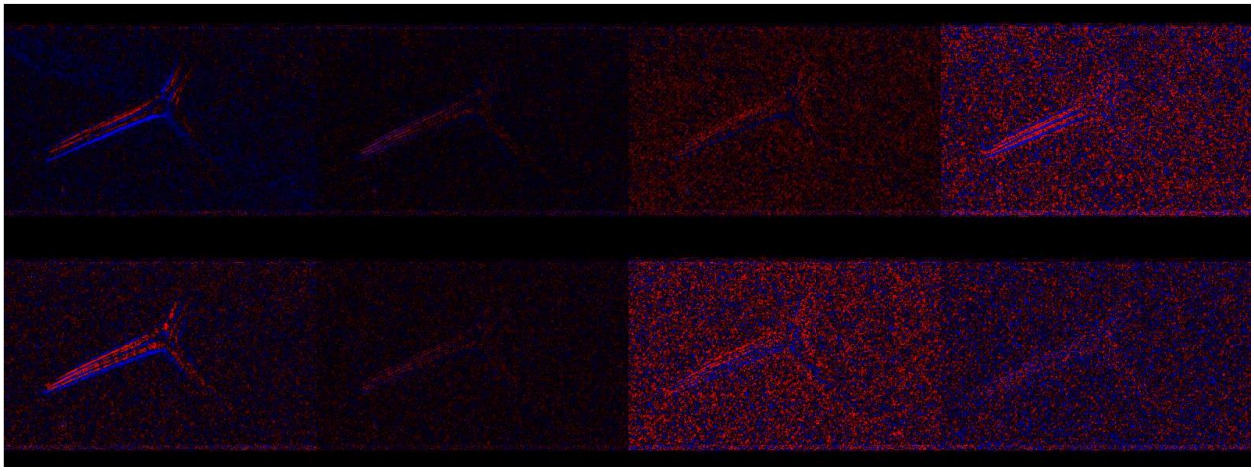

Figure 35 - Principal components 5-8 in the trichome vibration video. PCs obtained by svds() are shown in the top row, PCs obtained using crunchy_isvd() are shown in the bottom row.

4.3.3 Comparison of V matrices

To compare the **V** matrices, we plot the columns of **V**. Each column of **V** shows the contributions of the corresponding principal component to the individual frames of the video.

Because the columns of the **U** matrix may be flipped in sign arbitrarily, we have also flipped the signs of the columns of the **V** matrices from the two SVD algorithms as needed to get them to agree (again, by inspection).

The **V** matrix produced by the iterative solver is computed as the principal components are being refined. As such, all columns of **V** start at zero and converge toward a more accurate solution as more input columns are provided. A better **V** matrix is found by using a least-squares fit of the original input matrix to the **U** and **S** matrices computed by the iterative solver. Computing a new **V** matrix took less than 0.2 seconds when taking advantage of the orthogonality of the **U** matrix and the diagonality of the **S** matrix (if we pre-multiply **US** and solve for **V**, this operation takes over 27 times as long). In MATLAB, this was written, V = A'*U/S; where **A** is our (centered) data matrix. Note that this requires us to convert the original matrix to floating point format which is contrary to our original goal of keeping memory usage low. To keep memory usage low, we may rewrite the operation to complete one row at a time:

```
UdbS = U/S; % calculate this only once
for i = 1:k
     Ai = im2double(A(:,:,i)); % rows of A' (cols of A)
     Ai = (Ai(:) - meanData); % make column and center data
     V(i,:) = Ai'*UdbS; % get a new row of V
end
```

Figure 36 shows plots of the first column of **V**, using three different methods to compute **V**. Here we see that the first column of **V** computed using svds() starts negative and becomes positive around frame 180. The column computed using crunchy_isvd() starts at zero, oscillates, and then begins to converge towards the exact solution by the end of the video, as the first PC is refined. If we ran the incremental SVD for a longer video, we would expect the later values of **V** (with higher row indexes) to be better than the earlier values, but the first values are not very accurate.

We see that column 1 of **V**_lsf (our least squares fit **V** matrix from the incremental **U** and **S**) is so close to column 1 from our *svds()* computed **V** that the two are hard to distinguish in the plot. These are the coefficients for a projection of our centered **A** matrix onto the space spanned by the first PC computed by *crunchy_isvd()*.

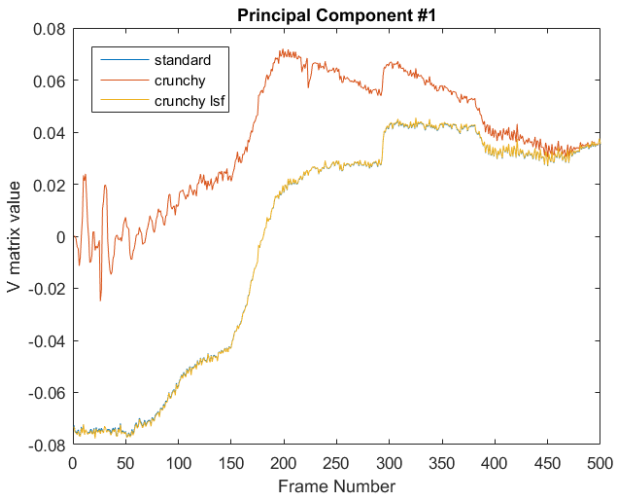

Figure 36 – Right singular vectors for PC1 computed three different ways. The least squares fit to the PC computed iteratively almost completely obscures the standard solution.

We see a similar trend for PCs 2 and 3 (Figure 37). The Column of **V** computed by the iterative approach starts poor, but improves with refinement of the associated PC. Using a least-squares fit after refining **U** and **S** gives us a much better **V** matrix.

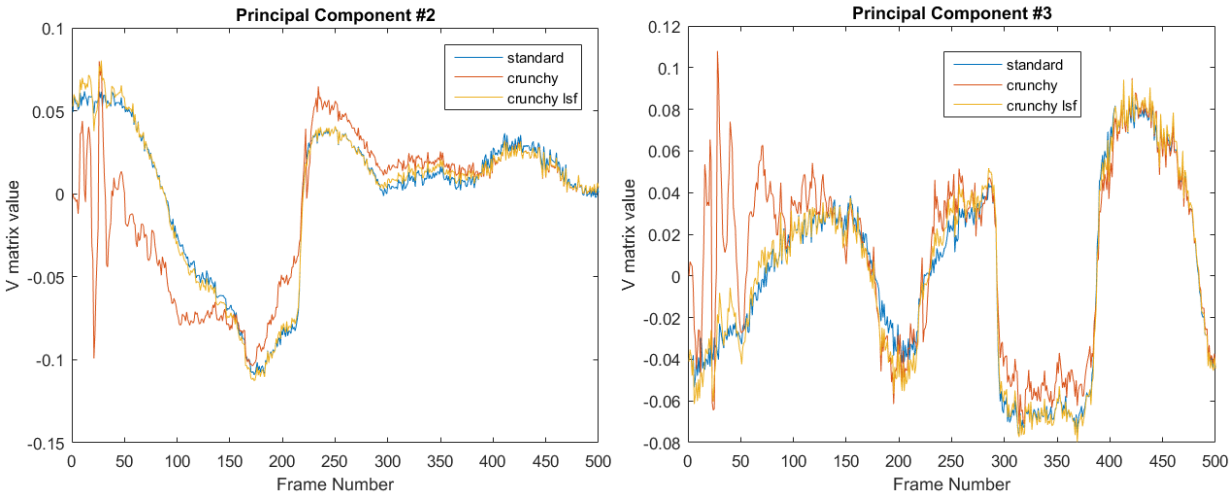

Figure 37 – Second and third right singular vectors computed in three different ways.

However, for PCs 4 and higher (Figure 38), the PCs computed using iterative and standard methods begin to diverge. Because of this, the columns of our iterative **V** and our least-squares **V** also diverge from the columns of our standard **V**. Using PCs above 3, we are looking at a

different SVD. However, it may not be as different as it initially seems. If we look at the iterative "crunchy" plot from PC4 and invert it, it looks very much like the standard plot for PC5. It seems that the two solvers may have switched the two components, which is not all that surprising if we look at how close their singular values are in Figure 33. Comparing basis images in Figure 34 and Figure 35 seems to support this theory.

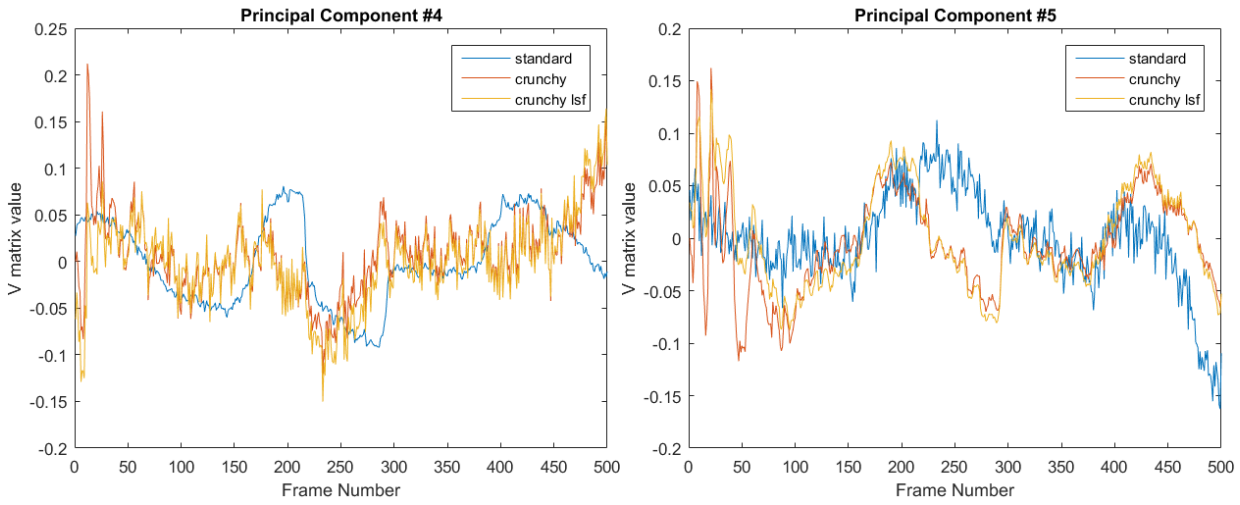

Figure 38 – Fourth and fifth right singular vectors computed in three different ways.

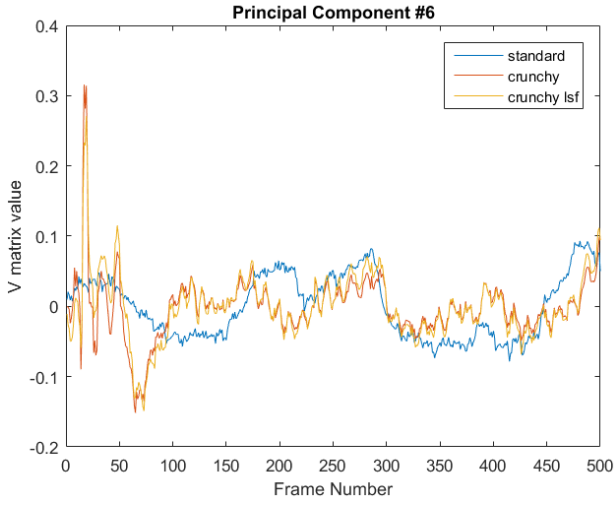

Figure 39 – Sixth right singular vector computed in three different ways.

4.3.4 Multiple Passes

With a longer video of this type (having repeated features and a low effective rank) we expect the values in **V** to converge toward the standard solution as more iterations are available for refinement. To test this, we ran the iterative solver on the same video, simply doubled in length by repetition. We then discarded the upper half of the **V** matrix (leaving only the portion associated with the second repetition of the video). To keep the columns of **V** unit-length, we multiply by the square root of 2. We divided the **S** matrix by the square root of 2 to compensate. The results for the first four PCs are shown in Figure 40.

We see that even in the second pass of our iterative solver, our first column of **V** has not converged to the solution found by the standard method, though it is improved over the single pass solution (). However, our least-squares fit to the iterated PCs fits our standard solution extremely well.

We see that the second pass has improved columns 2 and 3 of our V matrix and brought them much closer to our standard solution. They are still not quite as good as the least squares solution found from the final iterated U and S matrices.

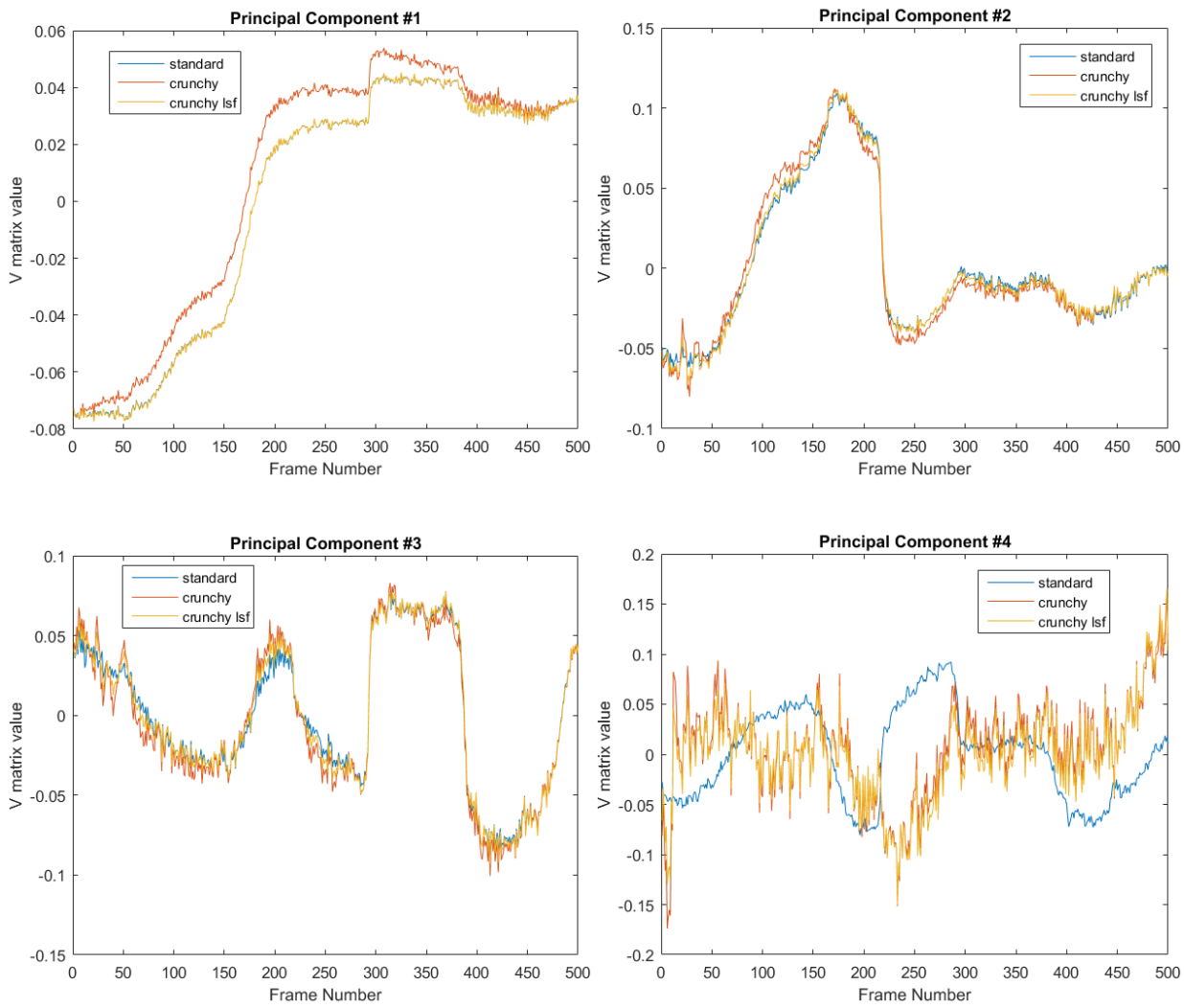

Figure 40 – Using 2-Pass iterative solver

4.3.5 Comparison of Residuals

By its definition, an n-dimensional truncated SVD should give the best least-squares rank-n approximation of the original matrix. By this standard, we can get a measure of how accurate an SVD solution is by comparing the norms of the residuals.

In Figure 41, we compare the 2-norms of the residuals of the trichome video reconstructed from only the first three components. The original video was subtracted from the video reconstructed from the first three PCs found using each of the five methods labeled. The per-pixel differences

were squared, summed, divided by the total number of pixels, and the square root of that value is listed at the top of each column.

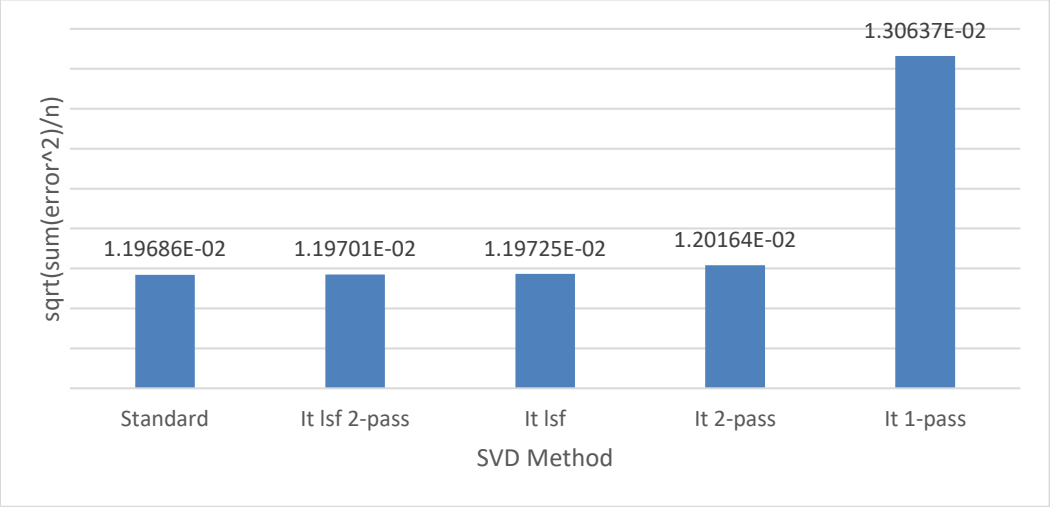

Figure 41 – Residual of 3 component reconstruction using several methods of finding the SVD

We see that the standard SVD (using MATLAB's *svds()* command) gives the lowest error, followed by a least squares fit solution using the iterative solver. Using two passes of the iterative solver before performing the least squares fit improves the accuracy slightly over using one pass (columns 2 and 3). Using the 2-pass iterative solver does not give quite as good accuracy as the first three methods, but it is an improvement over using the iterative solver with only one pass.

4.4 Comparison using Fluorescence video:

The previous analysis was also carried out on a 100-frame video of Fluo-4 stained heart cells. The results of that analysis are shown here with a minimum of commentary both for brevity and because they are consistent with the previous findings.

4.4.1 S matrices

In Figure 42 we see one interesting phenomenon which is that running the iterative solver with three passes brings the low-order singular values closer to those computed by the standard solver, but causes the higher order singular values to deviate further from the standard solution. It is beyond the scope of this work to investigate the reasons for this deviation.

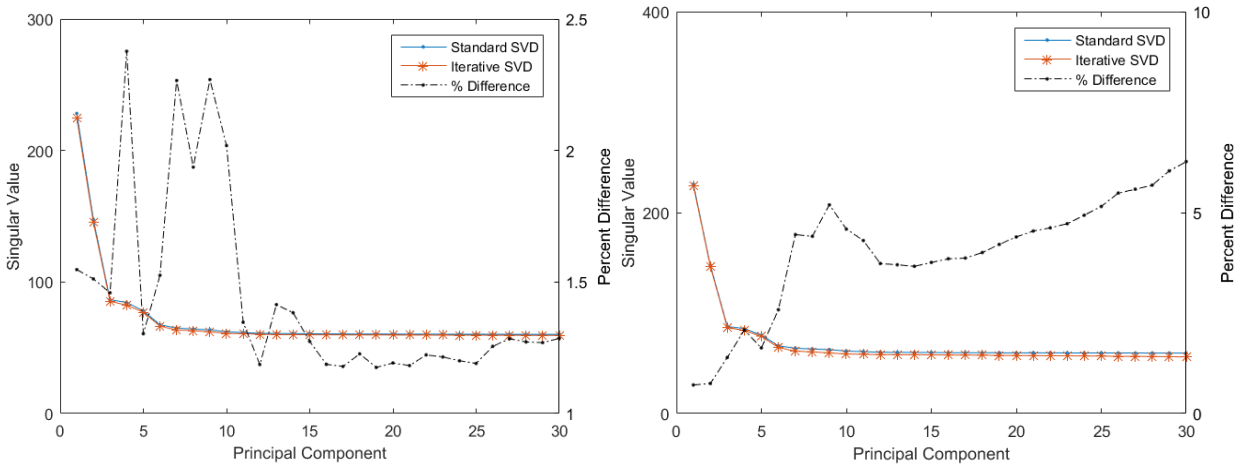

Figure 42 – Singular values computed by the standard and iterative solvers in one pass (Left) and three passes (Right).

Images of the PCs are omitted for brevity. The standard and incremental solvers gave visually similar results.

4.4.2 V matrices

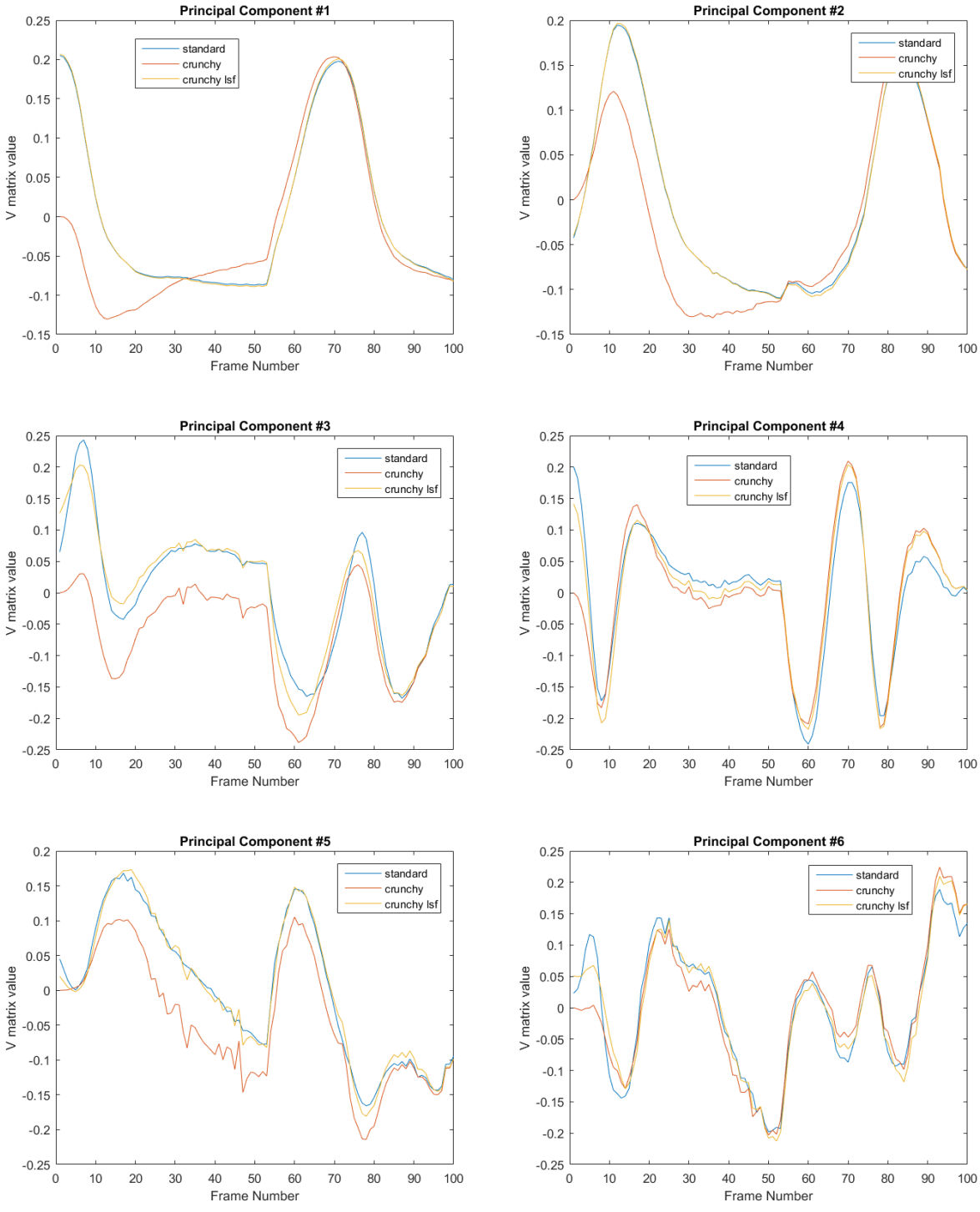

Figure 43 – Right singular vectors for the first 6 PCs. The iterative solver was used with one pass.

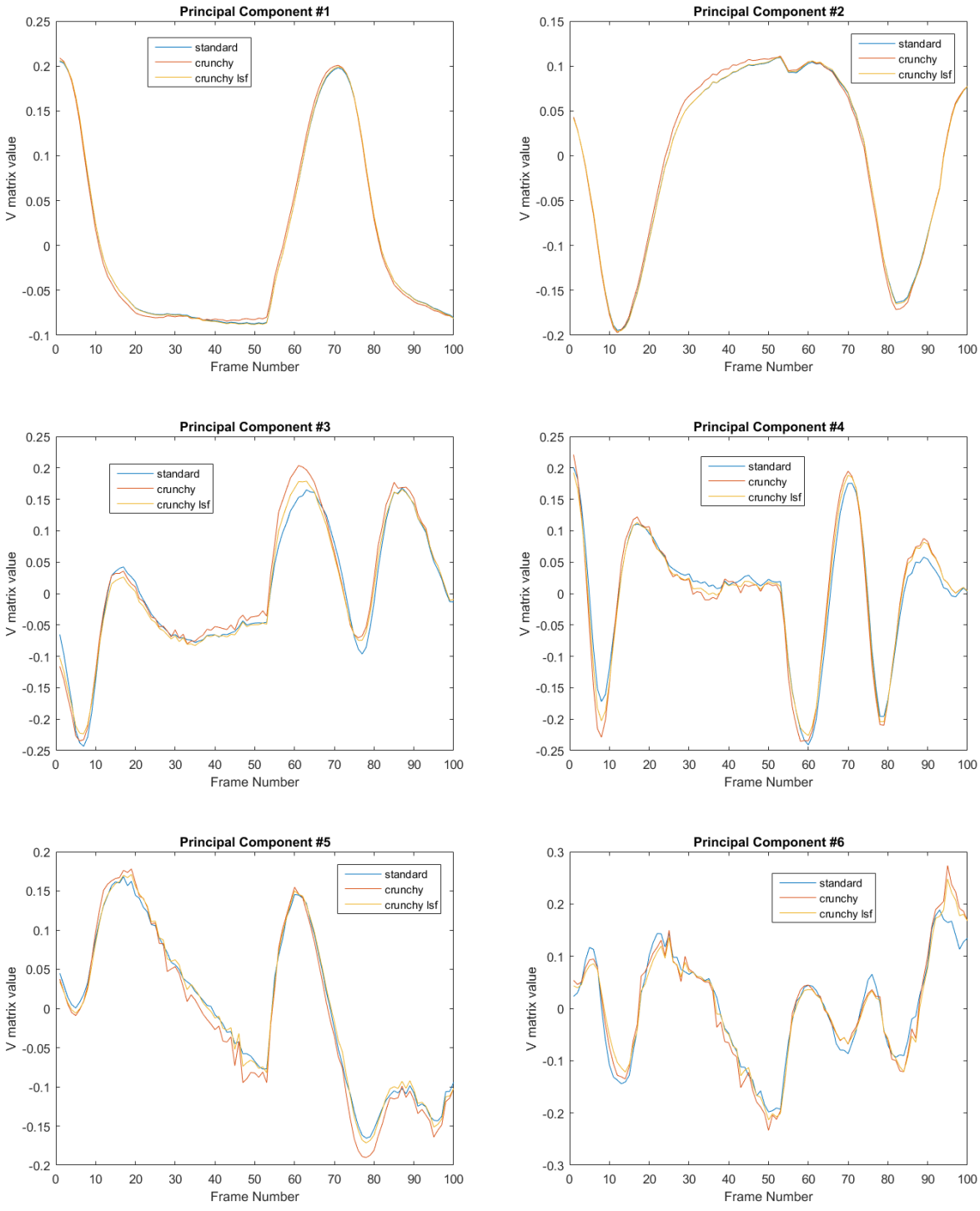

Figure 44 – Right singular vectors for the first 6 PCs. The iterative solver was used with three passes.

4.5 Conclusion

The clear advantage of the iterative SVD solver is that it allows us to process large data matrices that we could not process with standard solvers. It has the added benefit of being faster as well, reducing computation time for the first test case by a factor of six.

With the understanding that the important output of this software is the ideal orthogonal basis **U**, we can compute a good **V** quickly and with little memory usage by breaking the matrix operations down into a loop that feeds in a floating-point column of **A** at a time.

We may get slightly different left singular vectors from the iterative solver than from the standard solver, and differences in the computed singular values may "swap" singular vectors. This is apparent from visual inspection of basis images even without analyzing differences. However, these small differences seem to largely "come out in the wash" when we calculate our **V** matrix. We know that the first few columns of **U** are a very good, if not ideal, basis for **A** because of the negligible error difference between iterative and standard TSVD reconstructions once we find a post-iteration **V**.

Chapter 5: Conclusions

Conclusions

Principal component analysis shows clear utility for certain applications in cardiac optical mapping. PCA or a similar mathematical tool has the promise to be even more useful if we can separate components in a way that is more meaningful to the subject being studied.

The coefficients of the first PC work well as a proxy for the cardiac action potential state in some applications. We have used cross-correlation of the first columns of the **V** matrices to synchronize and find similarly shaped action potential sequences in fluorescence and phase-contrast heart cell videos. This works well as long as we are aware that the synchronization is approximate due to the fact that we are synchronizing calcium transients and deformations in the fluorescence video with deformations only in the phase-contrast video.

PCA data completion may be used to reconstruct missing or nonexistent videos from a related video if we have a sufficient data set to create a shared basis for the two types of video. We have demonstrated recreation of a long phase contrast video from a short fluorescence video, a synchronized short phase contrast video, and a long fluorescence video. This works because the two videos show two representations of the same underlying phenomenon (the cardiac action potential and related contraction).

Denoising with PCA is effective at removing heavy video noise and leaving a clear, but incomplete, approximation of the original video. PCA denoising removes noise much more completely than median filtering or Gaussian blur, without as much loss of detail and resolution. However, smaller, out of sync features and certain aspects of deforming features may be lost in

these approximations. Adding more components adds these details and missing events, but at the cost of adding back noise and potentially creating artifacts such as flickering and discontinuous motion.

Spectral analysis of the TSVD reconstruction residuals is computationally demanding and may not be useful in many cases. Noisy components may not separate cleanly from data-dominated components, so there is no point at which the residual is composed mainly of noise that is neither high-pass nor low-pass filtered. In our real-world fluorescence video, the residuals never resembled noise by our KS limits test. This may be due to structure in the 2D noise due to saturated regions. In the synthetic video case, residuals became noise-like after k=2.

Spectral analysis of the right singular vectors is computationally inexpensive and helps to characterize the contributions of those components, though making use of that information is another matter. In some cases, this may help us separate structured, data-dominated components from noisy components. However, even if we know that a basis image is largely dominated by noise, and that it has coefficients that appear noisy by our NCP test, we cannot conclude that the component is unimportant to the reconstruction and can be thrown out. PCs may be combined in complex ways by the overall system of matrix equations and isolating the components responsible for specific phenomena has proven difficult.

Future Directions

Perhaps the most useful goal we could hope to obtain from Principal Component Analysis would be a way to separate a video into components that represent different events and features, rather than simply relying on the algorithm to pick directions of greatest variance. There are several approaches that may be worth investigating. One is the use of other related statistical tools such

as independent component analysis and exploratory factor analysis. We have not tried these tools, but they may be worth investigating as a way to reduce complicated component interactions, i.e. multiple components combining to create a simple feature, such as a blinking CM.

If we can find a way to control or coax component selection, there are several benefits. For one, we may be able to use effective denoising without giving up important video events such as out of sync blinking cardiomyocytes. We could add in important feature components if they are not tied in to other undesirable features such as noise. A second benefit would be quick access to data from optical mapping recordings. If the individual components represented regions of the video blinking at different delays, for instance, then we would have automatic region of interest selection in the columns of **U**, and fluorescence intensity plots for those regions in the columns of **V**. A third benefit of having this kind of component separation would be the ability to explore the relationships between components related to intensity changes and components related to deformation.

There may be other ways to achieve this kind of separation of regional deformations and intensity changes. We are currently investigating the use of strain mapping software that has been modified to "unwarp" deformed regions in a video [22]. This offers the promise of removing motion from videos of dynamically deforming subjects. We not only get a map of the strains in time, but we stabilize the image for ease of fluorescence intensity mapping. PCA of the stabilized images may offer a simpler case where the components have less complicated events to reconstruct. This may be useful for denoising (we could add deformations back after the stabilized image set has been denoised), automatic region selection, and fluorescence intensity plotting.

References

[1] S. Rohr, "Role of gap Junctions in the propagation of the cardiac action potential," *Cardiovascular Research,* no. 62, pp. 309-322, 2004.

[2] J. H. King, C. L.-H. Huang and J. A. Fraser, "Determinants of myocardial conduction velocity: implications for arrhythmogenesis," *frontiers in Physiology,* vol. 4, 2013.

[3] M. K. Homoud, *Introduction to Electrocardiography,* 2008.

[4] J. Pellman, J. Zhang and F. Sheikh, "Myocyte-fibroblast communication in cardiac fibrosis and arrhythmias: Mechanisms and model systems," *Journa of Molecular and Cellular Cardiology,* no. 94, pp. 22-31, 2016.

[5] S. Rohr, "Myofibroblasts in diseased hearts: New players in cardiac arrhythmias," *Heart Rhythm Society,* 2009.

[6] P. Kohl and R. G. Gourdie, "Fibroblast–myocyte electrotonic coupling: Does it occur in native cardiac tissue?," *Journal of Molecular and Cellular Cardiology,* vol. 70, pp. 37-46, 2014.

[7] G. Strang, Introduction to Linear Algebra, Wellesley, MA: Wellesley-Cambridge Press, 1998.

[8] C. Moler, Numerical Computing with Matlab, SIAM, 2004.

[9] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.

[10] J. Shlens, "A Tutorial on Principal Component Analysis," arXiv.org, 2014.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes in C++, Cambridge, UK: The Press Syndicate of the University of Cambridge, 2002.

[12] A. Abrams, E. Feder and R. Pless, "Exploratory analysis of time-lapse imagery with fast subset PCA," *2011 IEEE Workshop on Applications of Computer Vision,* pp. 336-343, 2011.

[13] S. Rohr and J. P. Kucera, "Optical Recording System Based on a Fiber Optic Image Conduit: Assessment of Microscopic Activation Patterns in Cardiac Tissue," *Biophysical Journal,* vol. 75, no. 2, pp. 1062-1075, 1998.

[14] J. A. Scull, L. C. McSpadden, H. D. Himel, N. Badie and N. Bursac, "Single-Detector Simultaneous Optical Mapping of Vm and [Ca2+]i in Cardiac Monolayers," *Annals of Biomedical Engineering,* vol. 40, no. 5, pp. 1006-1017, 2011.

[15] P. Lee, K. Wang, C. E. Woods, P. Yan, P. Kohl, P. Ewart, L. Loew, D. Terrar and C. Bollensdorff, "Cardiac electrophysiological imaging systems scalable for high-throughput drug testing," *Pflugers Archiv - European Journal of Physiology,* vol. 464, no. 6, pp. 645-656, 2012.

[16] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Third Edition, Pearson Education, 2008.

[17] B. W. Rust, "TRUNCATING THE SINGULAR VALUE DECOMPOSITION FOR ILL-POSED PROBLEMS," NISTIR 6131, Mathematical and Computational Sciences Division, Gaithesberg, MD 20899, 1998.

[18] P. C. Hansen, M. E. Kilmer and R. H. Kjedsen, "Exploiting Residual Information in the Parameter Choice for Discrete Ill-Posed Problems," *BIT Numerical Mathematics,* no. 46, pp. 41-59, 2006.

[19] P. C. Hansen, Discrete Inverse Problems Insight and Algorithms, Philadelphia: Society for Industrial and Applied Mathematics, 2010.

[20] M. Brand, "Incremental Singular Value Decomposition of Uncertain Data with Missing Values," *European Conference on Computer Vision,* pp. 707-720, 2002.

[21] L. H. Zhou, S. B. Liu, P. F. Wang, T. J. Lu, F. Xu, G. M. Genin and B. G. Pickard, "The Arabidopsis trichome is an active mechanosensory switch," *Plant, Cell & Environment,* vol. 40, no. 5, pp. 611-621, 2016.

[22] J. J. Boyle, M. Kume, M. A. Wyczalkowski, L. A. Taber, R. B. Pless, X. Younan, G. M. Genin and S. Thomopoulos, "Simple and accurate methods for quantifying deformation, disruption, and development in biological tissues," *Journal of the Royal Society Interface,* vol. 11, no. 100, 2014.