Using Fourier transforms and singular value decomposition to denoise cytometry biosensor images and track rare cells

Reed Nomura

March 19, 2020

Abstract

In this paper I will be using Fourier transforms to denoise images from a low-cost cytometry biosensor. The denoised Fourier images will then be analyzed using singular value decompositions (SVDs). The algorithms developed in this paper could be used to create machine learning algorithms used to analyze future footage from similar low-cost cytometry biosensors.

1 Introduction and Overview

Research is being conducted by the NIH on low-cost cytometry biosensors that are able to detect rare cells. These sensors are different from traditional cytometry biosensors in that they are able to be produced inexpensively with minimal resources. The sensor used to collect the data I will be analyzing consisted of a webcam, a 450 nm 1 W laser, a flow cell, a computer, and a few lenses and filters. In total, this set up costs less than 1/10th what a traditional cytometry biosensor would cost. Additionally, this low-cost option is significantly better at rare cell detection. Detecting rare cells is highly applicable in a clinical setting. Some of the applications of this sensor include cancer prognosis and detecting metastasis-capable malignancy early. The sensor is able to detect these cells by locating "streaks" in the footage it collects [7].

In this paper I will be analyzing a 762 frame video that this sensor collected. It is my goal to remove noise from each image and highlight any streaks that might appear. I will focus on only one color channel for the sake of processing efficiency. I will then take that data and average it and subtract that average from each frame to remove any constant "background" features. I will then use Fourier transforms and a Gaussian filter to filter out noise from each frame. I will then perform a SVD on the set of frames in the Fourier domain. The SVD will provide information about what the dominate modes are and how the average of the streaked images differ from the dominate modes. This information will be vital in order to develop a machine learning algorithm that will be able to automatically detect and analyze streaks in future sensor footage.

2 Theoretical Background

2.1 Fourier Transforms

Fourier transforms will be used to transform data in the spatial domain into data in the frequency domain in order to apply a Gaussian filter and SVD data in the Fourier Domain. The Fourier contains more valuable information for our purposes in this paper. For more information on Fourier transforms, review chapter 13 Kutz's textbook [1] or the theoretical background in Homework 1 [3].

2.2 Filtering Data

A Gausian filter will be applied to each frame in the Fourier domain in order to reduce noise and focus on key frequencies. In this paper, this will help to bring more clarity to each of the frames that we are analyzing. For more information on Gaussian filters, review chapter 13 in Kutz's textbook [1] or Homework 1 and 2[4] [3].

2.3 Singular Value Decomposition

A SVD will be applied to the collection of frames in the Fourier domain. From the SVD we will be able to reveal what makes images with streaks distinct from the collection of images as a whole. For more information on SVDs, review chapter 15 in Data-Driven Modeling and Scientific Computation[1] or the theoretical background in Homework 3 and 4 [5] [6].

3 Algorithm Implementation and Development

1. Load Footage
2. Isolate Green Channel
3. Subtract Background
4. Apply Fourier Transform
5. Apply Gaussian Filter
6. Perform SVD
7. Analyze Results

Algorithm 1: Remove Background

```
Load 'Frames'
Isolate Green Channel for 'Frames'
Set Number of 'Frames' = k
Create 'AllFrames' as an empty Frame
Create 'AveFrames' as an empty Frame
for i = 1 : k do
   Add Frame i to AllFrames
end for
Set AveFrames = AllFrames / k
for i = 1 : k do
   Store Frame i - AveFrames in the ith frame position of 'Frames'
end for
```

Algorithm 2: Gaussian Filter

```
Load 'Frames'
Set Number of 'Frames' = k
Create Gaussian filter with desired width
for i = 1 : k do
   Perform FFT2 of Frame i
   Perform FFTshift
   Apply Filter
   Perform IFFTshift
   Perform IFFT2
   Store result in the ith frame position of 'Frames'
end for
```

Algorithm 3: Singular Value Decomposition

```
Load 'Frames'
Set [m,n] = dimensions of a single frame
Set Number of 'Frames' = k
Create 'DataMatrix' as an empty [k,m*n] matrix
for i = 1 : k do
   Reshape Frame i into a vector
   Store Vector in Row i of DataMatrix
end for
Set [u,s,v] = svd(DataMatrix)
Plot diagonals of s
Plot dominate modes
```

4 Computational Results

For processing efficiency I chose to analyze all data in only 1 color. Therefore, I had the options of flattening everything to the grayscale or look exclusively at the red, green or blue channel. Although I evaluated each

option extensively in order to make my decision, figure 1 provides a useful illustration as to why I chose to isolate the green channel for observation.

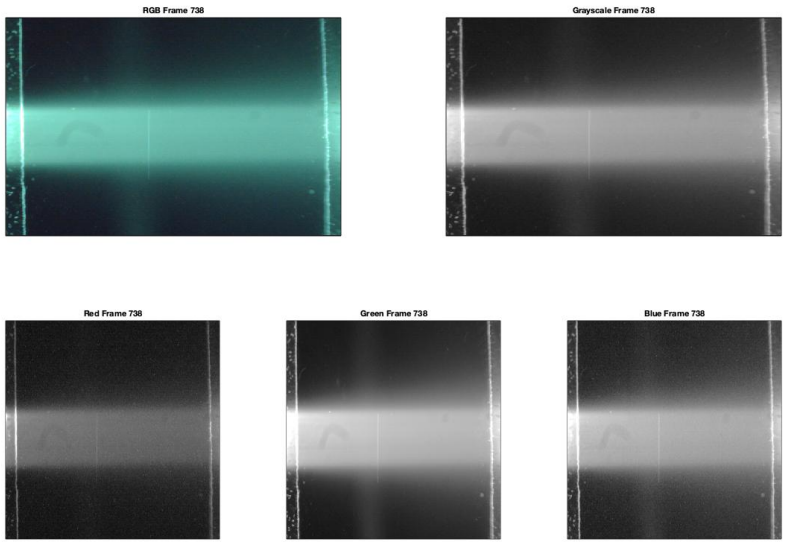

Figure 1: Differences of each color option

In the figure 1 there are 5 photos of frame 738. Multiple frames were observed but this frame was chosen for the illustration because there is a prominent streak visible. The green channel provided the clearest images and best contrast for the streaks. I then averaged together every frame and subtracted that average from each frame to eliminate the consistent background pieces that needed to be removed.

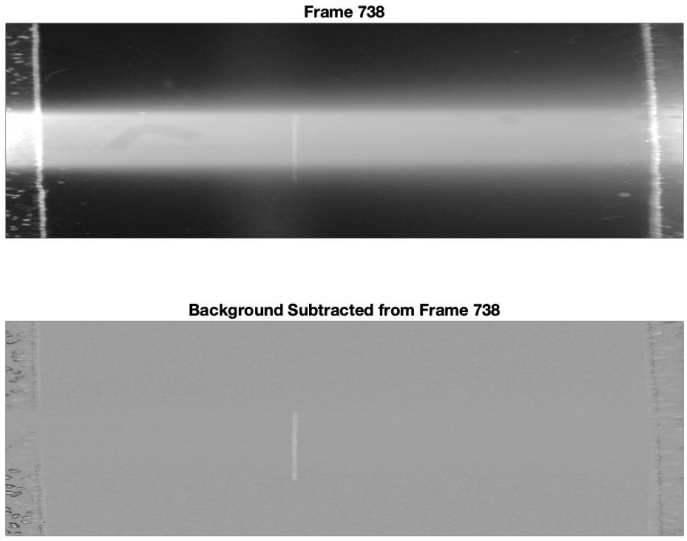

Figure 2: Before and after subtracted average

In the figure 2 you can see the difference between the original green channel of frame 738(on top) versus

the result when the background information is subtracted from the frame. After this step, each frame was converted into the Fourier domain and filtered using a Gaussian filter with a width of 0.00001. Each frame was then converted back into the spatial domain.

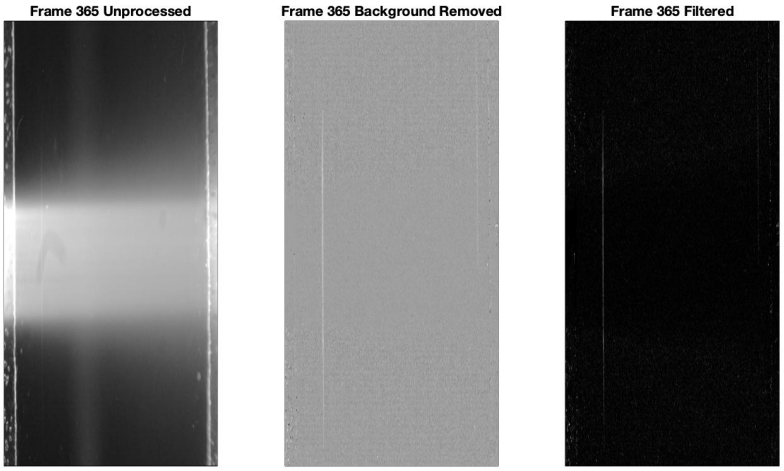

Figure 3: Starting image to filtered image

Figure 3 shows the progression of frame 365 from the unprocessed image (left) to the image with the background removed (center), and lastly, the filtered image. As evidenced, the progressive steps have helped to highlight streaks in the frame. After all of the data had been filtered in the Fourier domain, I performed an svd on all of the frames before they were converted back in to the spatial domain. The diagonals of s were potted on a semi-log plot.

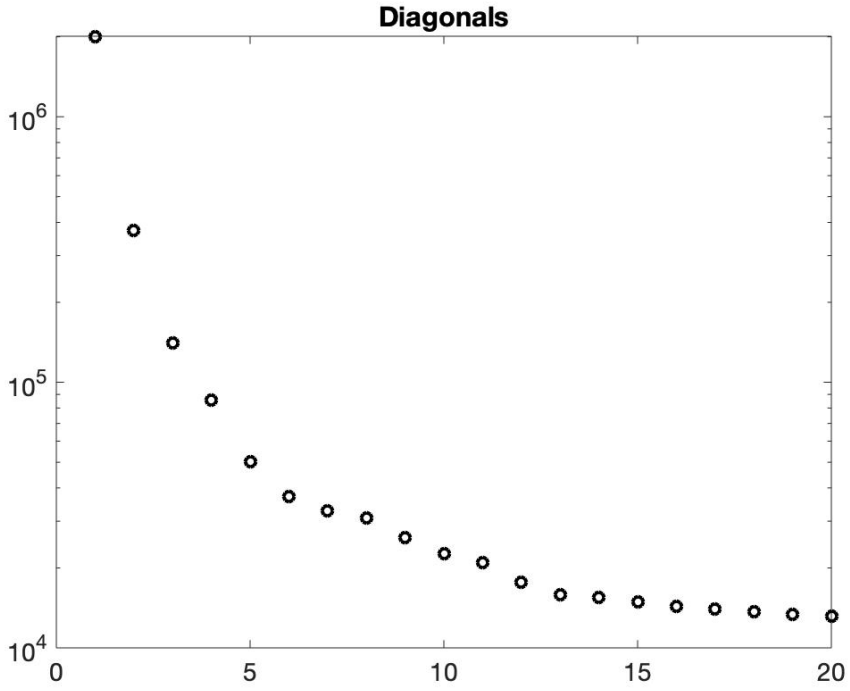

Figure 4: Diagonals of the Fourier SVD

We can see in figure 4 that mode 1 contains a significant amount of information and there is a significant drop off between the next couple of modes.

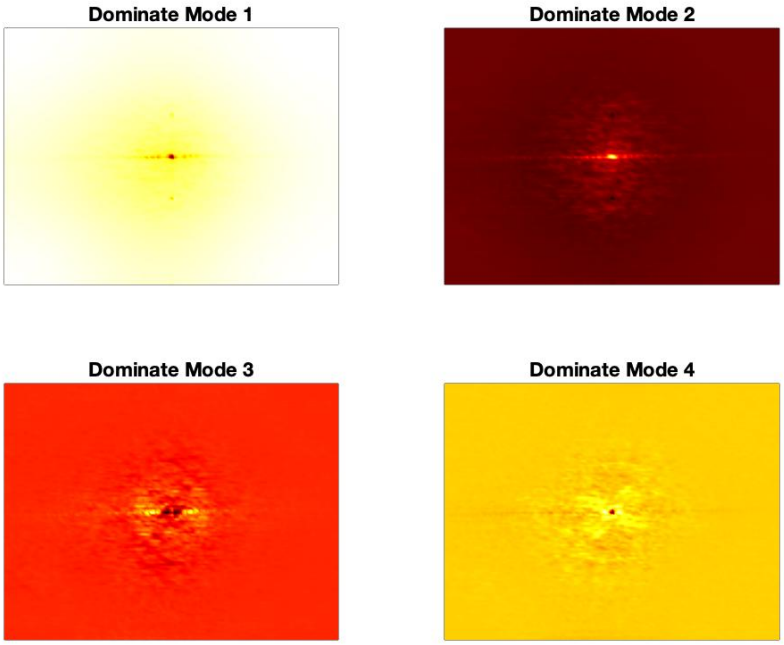

Figure 5: Dominate 4 modes of the Fourier Domain

In figure 5 I have plotted the top 4 dominate modes. After this was completed, I wanted to compare the frames with streaks against the dominate modes. Since this data was collected from existing research, the frames containing streaks were available. I averaged together all frames containing streaks in the Fourier domain and projected it onto the first three modes.

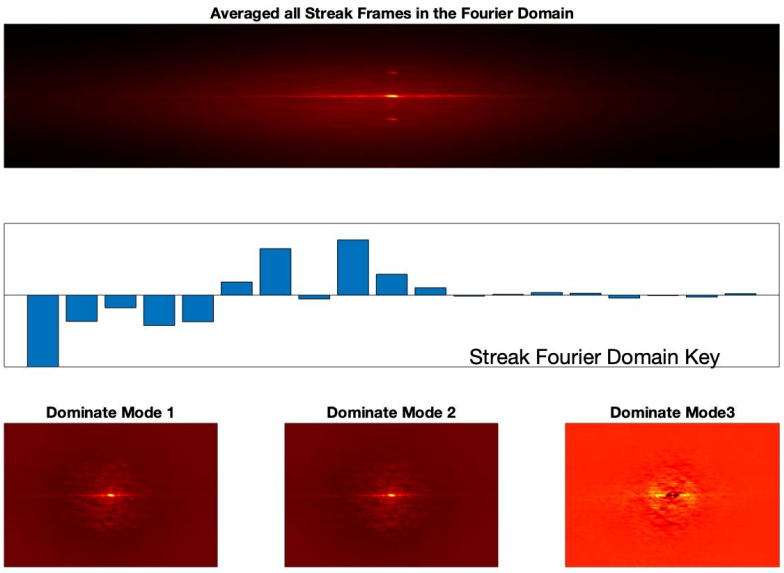

Figure 6: Comparison of the average of the streaked frames against the dominate modes

In figure 6 you can see all of the frames containing a streak averaged (top), the key of how this average compares against the top 20 modes (middle), and the actual visualization of the streaked images projected onto the top three modes (bottom).

5 Summary and Conclusions

There were many strategies applied in this paper to attempt to denoise the data and make it easier to isolate streaked images. The first step was to isolate the green channel. This helped to reduce processing cost and green was chosen because the streaks were more prominent in the green channel. After this, I subtracted the background information, this proved to be one of the most useful steps in highlighting streaks. I then applied a Gaussian filter to the frames to reduce some noise and focus the images. I then performed a SVD on the images in the Fourier domain in order to isolate the distinctive differences between images with streaks and those without. Continuing from here I would recommend applying a supervised machine learning algorithm to identify streak images from future sets of images converted into the Fourier domain.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[2] *MathWorks Website*. URL: `https://www.mathworks.com/help/matlab/index.html`.

[3] Nomura. *Homework 1*. 2020. URL: `https://github.com/ReedNomura/AMATH-582/blob/master/AMATH582_Homework1.pdf`.

[4] Nomura. *Homework 2*. 2020. URL: `https://github.com/ReedNomura/AMATH-582/blob/master/AMATH582_Homework2.pdf`.

[5] Nomura. *Homework 3*. 2020. URL: `https://github.com/ReedNomura/AMATH-582/blob/master/AMATH582_Homework3.pdf`.

[6] Nomura. *Homework 4*. 2020. URL: `https://github.com/ReedNomura/AMATH-582/blob/master/AMATH582_Homework4.pdf`.

[7] Miguel Ossandon et al. "A computational streak mode cytometry biosensor for rare cell analysis". In: *Analyst* 142 (4 2017), pp. 641–648.

Appendix A MATLAB Functions

- `Y = double(X)`: converts the values in X to double precision [2].
- `A = imread(filename)`: reads the image from the file specified by filename, inferring the format of the file from its contents. If filename is a multi-image file, then imread reads the first image in the file [2].
- `s = num2str(A)`: converts a numeric array into a character array that represents the numbers. The output format depends on the magnitudes of the original values. num2str is useful for labeling and titling plots with numeric values[2].
- `pcolor(C)`: creates a pseudocolor plot using the values in matrix C [2].
- `B = flipud(A)`: returns A with its rows flipped in the up-down direction (that is, about a horizontal axis) [2].
- `eval(expression)`: evaluates the MATLAB® code represented by expression. If you use eval within an anonymous function, nested function, or function that contains a nested function, the evaluated expression cannot create a variable. [2].

- `B = reshape(A,sz1,...,szN)`: reshapes A into a sz1-by-...-by-szN array where sz1,...,szN indicates the size of each dimension [2].
- `B = imresize(A,[numrows numcols])`: returns image B that has the number of rows and columns specified by the two-element vector [numrows numcols] [2].
- `size(A)`: returns a row vector whose elements are the lengths of the corresponding dimensions of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4].[2]
- `d = eigs(A,k,sigma)`: returns k eigenvalues based on the value of sigma. For example, eigs(A,k,'smallestabs') returns the k smallest magnitude eigenvalues [2].
- `D = diag(v)`: returns a square diagonal matrix with the elements of vector v on the main diagonal. [2].
- `semilogy(Y)`: creates a plot using a base 10 logarithmic scale for the y-axis and a linear scale for the x-axis [2].
- `bar(y)` creates a bar graph with one bar for each element in y. If y is an m-by-n matrix, then bar creates m groups of n bars [2].
- `w = hamming(L)`: returns an L-point symmetric Hamming window [2].

  s = spectrogram(x,window) uses window to divide the signal into segments and perform windowing.
- `[y,Fs] = audioread(filename)`: reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs[2].
- `[M,I] = max()`: returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes [2].
- `[row,col] = ind2sub(sz,ind)`: returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz. Here sz is a vector with two elements, where sz(1) specifies the number of rows and sz(2) specifies the number of columns [2].
- `B = repmat(A,n)`: returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)*n when A is a matrix [2].
- `sigma = svd(A)`: returns a vector sigma containing the singular values of a symbolic matrix A [2].
- `y = linspace(x1,x2,n)`: generates n points. The spacing between the points is $\frac{x2-x1}{n-1}$ [2]

  [**C6AN02517J**]

Appendix B MATLAB Code

This code can be found at: https://github.com/ReedNomura/AMATH-582/blob/master/Homework4.m

```
%% Final Project (Low Resolution Medical Imaging)

%% Initial .mat file conversion
clear all; close all; clc; %Start Fresh
v = VideoReader('RGB_Data.avi');
vid_frames = read(v);  %read all frames
clear v   %release it
save('RGB_Data.mat', 'vid_frames');

%% Load Data
clear all; close all; clc; %Start Fresh
load('RGB_Data.mat');
```

```
vid_frames_double = double(vid_frames); % Make Double Percision
[A1_1,B1_1,C1_1,D1_1] = size(vid_frames); %Initial Size Check
% Isolate Color Channels
Channel_1 = vid_frames_double(:,:,1,:);
Channel_2 = vid_frames_double(:,:,2,:);
Channel_3 = vid_frames_double(:,:,3,:);
%% All Frames Containing Streaks
S = [51, 52, 161, 162, 212, 213, 213, 247, 248, 249, 250, 264, 265, 302, 303, 303, 304, ...
    322, 323, 333, 334, 364, 365, 365, 366, 368, 369, 388, 389, 474, 475, 510, 511, 549, ...
    550, 570, 570, 571, 571, 572, 586, 587, 612, 613, 650, 651, 652, 667, 668, 670, 671, ...
    697, 698, 737, 738, 739, 740];
NS = [1:762]; % All Frames
for kk = 1:57
NS = NS(NS≠S(kk)); %Frams with no Streaks
end
%% Display Channel Differences
Frame = 738; % 1 to 762
a = (['RGB Frame ', num2str(Frame)]);
b = (['Grayscale Frame ', num2str(Frame)]);
c = (['Red Frame ', num2str(Frame)]);
d = (['Green Frame ', num2str(Frame)]);
e = (['Blue Frame ', num2str(Frame)]);
Color = 'gray'; %  parula, jet, hsv, hot, cool, spring, summer, autumn, winter, gray, ...
    bone, copper, pink, lines, colorcube, prism, flag, white
figure()
subplot(2,2,1)
image(vid_frames(:,:,:,Frame))
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(2,2,2)
image(rgb2gray(vid_frames(:,:,:,Frame))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(b)

subplot(2, 3, 4)
pcolor(flipud((vid_frames(:,:,1,Frame)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(c)
hold on

subplot(2, 3, 5)
pcolor(flipud((vid_frames(:,:,2,Frame)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(d)

subplot(2, 3, 6)
pcolor(flipud((vid_frames(:,:,3,Frame)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(e)
%% Isolating Single Color Channel
One_Channel_frames_double = Channel_2; %Pick Color Channel

%% Subtract background
One_Channel_Ave = One_Channel_frames_double(:,:,1);
for jj = 2:762
Frame = jj;
One_Channel_Each = One_Channel_frames_double(:,:,Frame);
One_Channel_Ave = One_Channel_Ave + One_Channel_Each;
end
One_Channel_Ave = One_Channel_Ave/762;

for kk = 1:762
    Frame = kk;
    Sub_One_Channel(:,:,Frame) = One_Channel_frames_double(:,:,Frame) — One_Channel_Ave;
end
%% Compare Subtracted Background
Frame = 738; % 1 to 762
```

```
a = (['Frame ', num2str(Frame)]);
b = (['Background Subtracted from Frame ', num2str(Frame)]);
Color = 'gray'; %  parula, jet, hsv, hot, cool, spring, summer, autumn, winter, gray, ...
    bone, copper, pink, lines, colorcube, prism, flag, white

figure()
subplot(2,1,1)
pcolor(flipud((vid_frames(:,:,2,Frame)))),
shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(2,1,2)
pcolor(flipud((Sub_One_Channel(:,:,Frame)))),
shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(b)


%% Gaussian filter
Width = 0.00001; % Filter Width

kx =1:640;
ky=1:480;
[Kx,Ky] = meshgrid(kx,ky);

G = exp(-Width*(Kx-241).^2-Width*(Ky-321).^2); %Gausian Filter

Filter = G; % Pick Filter

vid_frames_filtered = One_Channel_frames_double;
for j = 1:762
Frame = j ; %Frames 1 to 762
Frame_Text = (['Frame:', num2str(j)]);
A = Sub_One_Channel(:,:,Frame); %Isolate Specific Frame and Channel
AFt = fft2(A); % Two Dimensional Fourier Transform
Ats=fftshift(AFt); % Shift
Atsf = Ats.*Filter; % Apply Filter
Atf = ifftshift(Atsf); % Shift back
Af = ifft2(Atf); % Inverse Fourier Transform
vid_frames_filtered(:,:,Frame) = abs(Af); % Take the Absolute Value in order to display ...
    without issues
vid_frames_Fourier(:,:,Frame) = abs(Ats);
vid_frames_Fourier_filtered(:,:,Frame)= abs(Ats.*Filter);
end
%% Plot Data in the Fourier Domain
Frame_1 = 738; %frame with Streak
Frame_2 = 670; %frame 2 with Streak
Frame_3 = 530; %frame without streak
Frame_4 = 115; %frame 2 without streak
a = (['Frame ', num2str(Frame_1), ' With Streak']);
b = (['Frame ', num2str(Frame_2), ' With Streak']);
c = (['Frame ', num2str(Frame_3), ' Without Streak']);
d = (['Frame ', num2str(Frame_4), ' Without Streak']);
Color = 'Hot';
figure()
subplot(2,2,1)
pcolor(flipud((vid_frames_Fourier(:,:,Frame_1)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(2,2,2)
pcolor(flipud((vid_frames_Fourier(:,:,Frame_2)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(b)


subplot(2,2,3)
```

```
pcolor(flipud((vid_frames_Fourier(:,:,Frame_3)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(c)

subplot(2,2,4)
pcolor(flipud((vid_frames_Fourier(:,:,Frame_4)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(d)

%% Plot Data in Spatial Domain
Frame_1 = 738; %frame with Streak
Frame_2 = 670;%frame 2 with Streak
Frame_3 = 530; %frame without streak
Frame_4 = 115; %frame 2 without streak
a = (['Frame ', num2str(Frame_1), ' With Streak']);
b = (['Frame ', num2str(Frame_2), ' With Streak']);
c = (['Frame ', num2str(Frame_3), ' Without Streak']);
d = (['Frame ', num2str(Frame_4), ' Without Streak']);
Color = 'gray';
figure()
subplot(2,2,1)
pcolor(flipud((Sub_One_Channel(:,:,Frame_1)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(2,2,2)
pcolor(flipud((Sub_One_Channel(:,:,Frame_2)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(b)


subplot(2,2,3)
pcolor(flipud((Sub_One_Channel(:,:,Frame_3)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(c)

subplot(2,2,4)
pcolor(flipud((Sub_One_Channel(:,:,Frame_4)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(d)

%% Display Filtered vs Unfiltered in Spatial Domain
Frame_1 = 738; %frame with Streak
Frame_2 = 670;%frame 2 with Streak
Frame_3 = 530; %frame without streak
Frame_4 = 115; %frame 2 without streak
a = (['Frame ', num2str(Frame_1), ' Unfiltered With Streak']);
b = (['Frame ', num2str(Frame_1), ' Filtered With Streak']);
c = (['Frame ', num2str(Frame_3), ' Unfiltered Without Streak']);
d = (['Frame ', num2str(Frame_3), ' Filtered Without Streak']);
Color = 'gray';
figure()
subplot(2, 2, 1)
pcolor(flipud((Sub_One_Channel(:,:,Frame_1)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(2, 2, 2)
pcolor(flipud((vid_frames_filtered(:,:,Frame_1)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(b)

subplot(2, 2, 3)
pcolor(flipud((Sub_One_Channel(:,:,Frame_3)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(c)

subplot(2, 2, 4)
```

```
pcolor(flipud((vid_frames_filtered(:,:,Frame_3)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(d)

%% Alt Display Filtered vs Unfiltered in Spatial Domain
Frame_1 = 365; %frame with Streak
Frame_2 = 670;%frame 2 with Streak
Frame_3 = 530; %frame without streak
Frame_4 = 115; %frame 2 without streak
a = (['Frame ', num2str(Frame_1), ' Unprocessed']);
b = (['Frame ', num2str(Frame_1), ' Background Removed']);
c = (['Frame ', num2str(Frame_1), ' Filtered']);
d = (['Frame ', num2str(Frame_1), ' Filtered Without Streak']);
Color = 'gray';
figure()
subplot(1,3,1)
pcolor(flipud((One_Channel_frames_double(:,:,Frame_1)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(a)

subplot(1, 3, 2)
pcolor(flipud((Sub_One_Channel(:,:,Frame_1)))), shading interp,
set(gca, 'Xtick', [], 'Ytick',[])
title(b)

subplot(1, 3, 3)
pcolor(flipud((vid_frames_filtered(:,:,Frame_1)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(c)

%% Display Filtered vs Unfiltered in Fourier Domain
Frame_1 = 738; %frame with Streak
Frame_2 = 670;%frame 2 with Streak
Frame_3 = 530; %frame without streak
Frame_4 = 115; %frame 2 without streak
a = (['Frame ', num2str(Frame_1), ' Unfiltered With Streak']);
b = (['Frame ', num2str(Frame_1), ' Filtered With Streak']);
c = (['Frame ', num2str(Frame_3), ' Unfiltered Without Streak']);
d = (['Frame ', num2str(Frame_3), ' Filtered Without Streak']);
Color = 'hot';
figure()
% subplot(2, 2, 1)
% pcolor(flipud((vid_frames_Fourier(:,:,Frame_1)))), shading interp,
% set(gca, 'Xtick', [], 'Ytick',[])
% title(a)

subplot(2, 1, 1)
pcolor(flipud(vid_frames_Fourier_filtered(:,:,Frame_3))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(b)

% subplot(2, 2, 3)
% pcolor(flipud((vid_frames_Fourier(:,:,Frame_3)))), shading interp,
% set(gca, 'Xtick', [], 'Ytick',[])
% title(c)

subplot(2,1,2)
pcolor(flipud((vid_frames_Fourier_filtered(:,:,Frame_3)))), shading interp, colormap(Color)
set(gca, 'Xtick', [], 'Ytick',[])
title(d)
%% Create Fourier Data Matrix for all Frames
A = imresize(vid_frames_Fourier_filtered,[120,160]);
 for kk = 1:762
     Frame = kk;
     filtered_fourier_matrix(Frame,:) = reshape(A(:,:,Frame),1, 120*160);
 end
 %% Perform svd on Fourier Domain
[u,s,v] = svd(filtered_fourier_matrix);
```

```
sig = diag(s);
%% Plot Diagonals and Top Modes
figure()
semilogy(sig(1:20), 'ko', 'Linewidth', [2])
set (gca, 'Fontsize' , [14])
title('Diagonals')

figure()
subplot(2,2,1), face1 = reshape(v(:,1),120, 160); pcolor(flipud(face1)), shading interp, ...
    colormap(gray), set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 1')
subplot(2,2,2), face2 = reshape(v(:,2),120, 160); pcolor(flipud(face2)), shading interp, ...
    colormap(gray), set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 2')
subplot(2,2,3), face3 = reshape(v(:,3),120, 160); pcolor(flipud(face3)), shading interp, ...
    colormap(gray), set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 3')
subplot(2,2,4), face4 = reshape(v(:,4),120, 160); pcolor(flipud(face4)), shading interp, ...
    colormap(hot), set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 4')
%% Average Fourier Streak Images
Fourier_Streak_All_filtered = imresize(vid_frames_Fourier_filtered(:,:,51),[120,160]);
 for kk = S
     Frame = kk;
     Fourier_Streak_Each_filtered = ...
         imresize(vid_frames_Fourier_filtered(:,:,Frame),[120,160]);
     Fourier_Streak_All_filtered = Fourier_Streak_All_filtered + Fourier_Streak_Each_filtered;
 end
Fourier_Streak_Ave_filtered = (Fourier_Streak_All_filtered - ...
    imresize(vid_frames_Fourier_filtered(:,:,51),[120,160]))/57;
vec_Fourier_Streak_Ave_filtered = reshape(Fourier_Streak_Ave_filtered,1,120*160);
%%
face1 = reshape(v(:,2),120, 160);
face2 = reshape(v(:,2),120, 160);
face3 = reshape(v(:,3),120, 160);

figure()

subplot(3,1,1)
pcolor(flipud((Fourier_Streak_Ave_filtered))), shading interp, colormap(hot)
set(gca, 'Xtick', [], 'Ytick',[])
title('Averaged all Streak Frames in the Fourier Domain')


subplot(3,1,2)
projB_1 = vec_Fourier_Streak_Ave_filtered*v;
bar(projB_1(2:20)), set(gca, 'Xlim', [0,20], 'Ylim', [-2000, 2000], 'Xtick', [], 'Ytick',[])
text( 12, -1700, 'Streak Fourier Domain Key', 'Fontsize', [15])


subplot(3,3,7)
pcolor(flipud(face1)), shading interp, colormap(hot),
set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 1')

subplot(3,3,8)
pcolor(flipud(face2)), shading interp, colormap(hot),
set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode 2')

subplot(3,3,9)
pcolor(flipud(face3)), shading interp, colormap(hot),
set(gca, 'Xtick',[], 'Ytick',[])
title('Dominate Mode3')
```