Dynamic Mode Decomposition

University of Washington
AMATH 582 Homework 5
Winter Quarter 2019

David Long
dmlong888@live.com

Abstract

One of the most powerful methods for modeling the dynamics of systems that evolve over time is the *dynamic mode decomposition*. As it makes no functional assumptions about the variables in the system, it is especially robust in applications to non-linear systems. In this paper, the DMD will be coupled with the technique of singular value decomposition to reduce the rank of video data and to extract background and foreground features from it.

I. Introduction and Overview

Computer enthusiasts are aware of Moore's Law, which states that the number of transistors on a microchip is expected to double every two years while the costs are halved. His observation has roughly held true. Fabricators have reduced the process size to 7nm and packed processors with billions of transistors on multiple cores, enabling computers to perform a volume of calculations that was inconceivable to most people just thirty years ago. The volume of numbers that can be amassed in solving a complex problem such as time-stepping a non-linear partial differential equation is now limited more by the memory of the machine than its speed. It wasn't long ago that 256MB of ram was considered gluttonous but now, 8GB of RAM is commonplace in consumer machines. The sheer volume of data that must be processed by data-mining and classification algorithms can easily saturate 16GB of memory, which usually results in the code halting and the programmer cursing. It has become apparent that mathematical techniques of dimension reduction are a necessary step in algorithms that seek to solve problems involving big data. While there are many model-driven techniques that assume the data conforms to a governing equation, the technique of *dynamic mode decomposition* is purely data-driven and makes no such assumptions. As such, it has been shown to be an effective algorithm for capturing the dynamics of complex non-linear systems. This paper will introduce the DMD algorithm and explore its application to video files with the goal of separating the background and foreground.

II. Theoretical Background

Dynamic mode decomposition accomplishes three tasks.[1] First, it allows for the interpretation of complex dynamical systems through dimension reduction. Characteristic features of the data can be captured in low-rank modes and experts in the field can then interpret how those modes should be interpreted. Secondly, as the DMD is regression based, it easily allows for predictions of future states. Lastly, the DMD can be used to develop control algorithms, where short-term prediction of the state of the system can be used to assess conditions that can be imposed upon it to affect certain goals.

The dynamic mode decomposition combines dimension reduction in space with Fourier transforms in time. Data must first be collected over time. Recent advancements in the theory have made it possible to perform the algorithm with unequal time steps, but here it will be assumed that the data are collected sequentially at equal intervals of Δt. The data are to be arranged as vectors $\vec{x}(t) \in \mathbb{R}^n$ taken at *m* points in time and organized into a matrix

$$X = \left[\vec{x}_1 \;\; \vec{x}_2 \;\; \cdots \;\; \vec{x}_m \right] \qquad \textbf{(Eqn. 1)}$$

where $\vec{x}_k = \vec{x}\left(\left(k-1\right)\Delta t\right)$. The non-linear time evolution is approximated locally by the linear system

$$\frac{d\vec{x}}{dt} = A\vec{x} \qquad \textbf{(Eqn. 2)}$$

Regression algorithms are used to fit the matrix *A* in a least-squares sense, such that $\left\|\vec{x}_{k+1} - A\vec{x}_k\right\|_2$ is minimized over all time points $k \in \{1, 2, \ \ldots \ , m-1\}$.

It is well-known that the time-continuous equivalent of the discrete linear system of differential equations in Eqn. 2 has the solution

$$\vec{x}\left(t\right) = \sum_{k=1}^{n} \phi_k \exp\left(\omega_k t\right) b_k = \Phi \exp\left(\Omega t\right) \vec{b} \qquad \textbf{(Eqn. 3)}$$

where ϕ_k are eigenvectors, ω_k are corresponding eigenvalues, and the b_k are the coordinates of the initial value $\vec{x}\left(0\right)$ in the space spanned by the eigenvectors. The solution of Eqn. 2 can be rank-reduced and expressed as

$$\vec{x}_k = \sum_{j=1}^{r} \phi_j \lambda_j^k b_j = \Phi \Lambda^k \vec{b} \qquad \textbf{(Eqn. 4)}$$

As $\vec{x}_1 = \vec{x}\left(0\right)$, then, by Eqn. 3, $\vec{b} = \Phi^{\dagger}\vec{x}_1$, where $\Phi^{\dagger}$ is the Moore-Penrose pseudo-inverse of Φ.

Note that Eqn. 2 can be expressed as a Krylov space, where $\vec{x}_{k+1} = A^k \vec{x}_1$, which is the reason for raising the λ_j to the *k*th power. Equating the terms of Eqn. 3 and Eqn. 4 gives

$$e^{\omega t} = \lambda^k \ \Rightarrow \ \omega t = \ln \lambda^k \ \Rightarrow \ \omega = \frac{\ln \lambda^k}{t} = \frac{k \ln \lambda}{k \Delta t} = \frac{\ln \lambda}{\Delta t} \qquad \textbf{(Eqn. 5)}$$

The following computational algorithm for the DMD is modified from [1].

1. Form the following matrices, which are subsets of Eqn. 1:

$$\begin{aligned} X_1^{m-1} &= \left[\vec{x}_1 \ \ \vec{x}_2 \ \ \cdots \ \ \vec{x}_{m-1}\right] \\ X_2^{m} &= \left[\vec{x}_2 \ \ \vec{x}_3 \ \ \cdots \ \ \vec{x}_m\right] \end{aligned}$$

As X_2^m is a time-evolution of X_1^{m-1}, the matrices satisfy Eqn. 2. The objective is to fit *A* to

$$X_2^m = AX_1^{m-1} \qquad \textbf{(Eqn. 6)}$$

such that such that $\left\| \vec{x}_{k+1} - A\vec{x}_k \right\|_2$ is minimized over all time points $k \in \{1, 2, \ \dots \ , m-1\}$, which reduces the problem to a statistical regression.

2. Find the singular value decomposition:

$$X_1^{m-1} = U\Sigma V^* \qquad \textbf{(Eqn. 7)}$$

   where $U \in \mathbb{C}^{n \times r}$ are basis vectors, $\Sigma \in \mathbb{C}^{r \times r}$ describes the proportion of variance captured by each vector of *U*, and $V \in \mathbb{C}^{m \times r}$ gives a basis for the evolution over time. Note that the rank of the data can be reduced from *m* to *r*.

3. Solve Eqn. 6 using Eqn. 7:

$$A = X_2^m V \Sigma^{-1} U^* \qquad \textbf{(Eqn. 8a)}$$

   Project *A* onto *U* to get $\tilde{A} = U^* A U \in \mathbb{C}^{r \times r}$. Substituting Eqn. 8a gives

$$\tilde{A} = U^* X_2^m V \Sigma^{-1} \qquad \textbf{(Eqn. 8b)}$$

4. Calculate the eigen-decomposition of $\tilde{A}$:

$$\tilde{A}W = W\Lambda \qquad \textbf{(Eqn. 9)}$$

   where the columns of *W* are eigenvectors and the diagonal matrix Λ has the corresponding eigenvalues.

5. The DMD modes of the data are the columns of the matrix Φ, whose columns are the eigenvectors of the matrix *A*.

$$\Phi = X_2^m V \Sigma^{-1} W \qquad \textbf{(Eqn. 10)}$$

The original data set *X* (Eqn. 1) can now be approximated by

$$X \approx X_{DMD} = \Phi \exp\left(\Omega t\right)\vec{b} \qquad \textbf{(Eqn. 11)}$$

where the entries of the diagonal matrix Ω are $\omega_k = \ln\left(\lambda_k\right)/\Delta t$, $k \in \left\{1, 2,\ \dots\ , r\right\}$. The columns of Φ are the modes of the DMD and the product in Eqn. 11 evolves them over time. Various subsets of Φ can be constructed and similarly evolved, which is equivalent to extracting feature-sets from the data matrix *X*. In the next section, subsets of this nature will be used to extract the background and foreground modes from the dynamic mode decomposition of several videos.

III. Algorithm Implementation and Development

Three videos were used to explore the extraction of DMD modes. One was recorded with my cell phone. The other two were taken from libraries of stock video footage. The first was of a Raspberry Pi resting on a white piece of paper. Holding the camera above the field of view produced some shake in the recording. The second video was an overhead view of a steady stream of traffic crossing a bridge.[2] The third video was a sidewalk scene consisting of pedestrian and vehicle traffic.[3]

After importing the videos into MATLAB, I immediately converted them to grayscale and truncated them at 150 frames to reduce computational time. I also reshaped the frames into column vectors as required by Eqn. 1. The video frames are captured at equal intervals in time, which effectively sets $\Delta t = 1$.[4]

Subsets of the data matrix *X* were constructed according to Eqn. 6. The subset X_1^{m-1} was factorized by singular value decomposition as given by Eqn. 7. The SVD offers the opportunity to reduce the rank of the data, but it takes some fine-tuning to balance the reduction with video quality. I eventually settled on a rank of $r = 20$ and subset the matrices $U, \Sigma,$ and V accordingly. Not only did rank-reduction drastically improve performance, but there was a point beyond which including more modes didn't improve the quality of separation.

The matrix $\tilde{A} = U^* X_2^m V \Sigma^{-1}$ of Eqn. 8b was used to formulate the eigen-decomposition equation $\tilde{A}W = W\Lambda$ of Eqn. 9. The eigenvectors were used to calculate the dynamic modes $\Phi = X_2^m V \Sigma^{-1} W$ (Eqn. 10) and the eigenvalues were used to calculate the frequencies $\omega_k = \ln\left(\lambda_k\right)/\Delta t$ of the DMD that are used in Eqn. 11.

Recall from Eqn. 11 that $X_{DMD} = \Phi \exp\left(\Omega t\right)\vec{b}$. As $\vec{x}\left(0\right) \approx \Phi\vec{b}$, it is clear that Ω controls the evolution of the first frame in time. Modes with $\left|\omega_k\right| \approx 0$ will indicate features of the first frame that are stationary or change very little over time.[4] Identification of the frequencies with small absolute values makes it possible to identify the background modes. To facilitate this process, I sorted the output of the eigen-decomposition and plotted the frequencies in **Figure 1** . I experimented with various numbers of background modes, but the best results were obtained with just one or two.

After calculating the vector $\vec{b} = \Phi^{\dagger}\vec{x}_1$, I created a matrix of the time evolution of the DMD modes by calculating $\sum_{k=1}^{n} \exp\left(\omega_k t\right) b_k$. I used the index of the zero frequencies to multiply the corresponding modes from Φ with the corresponding time evolution vectors, which gave a matrix of the evolution of the background mode over time. This matrix X_{low} had the same dimensions as the original data matrix *X*. I calculated the foreground as $X_{sparse} = X - \left|X_{low}\right|$.

To view the results, I set up a loop to process X_{low} and X_{sparse}. At each iteration, I pulled a column vector, reshaped it as a video frame, took its real component, and converted it to uint8. The results of the separations are described in the next section. I spent a lot of time watching the videos and assessing the impact of choosing different numbers of modes from both the SVD and DMD decompositions. Representative frames from each video are shown in **Figures 2-4**.

For a discussion of the impact of modifying X_{low} and X_{sparse} with a residual matrix, see **Appendix A**.

IV. Computational Results

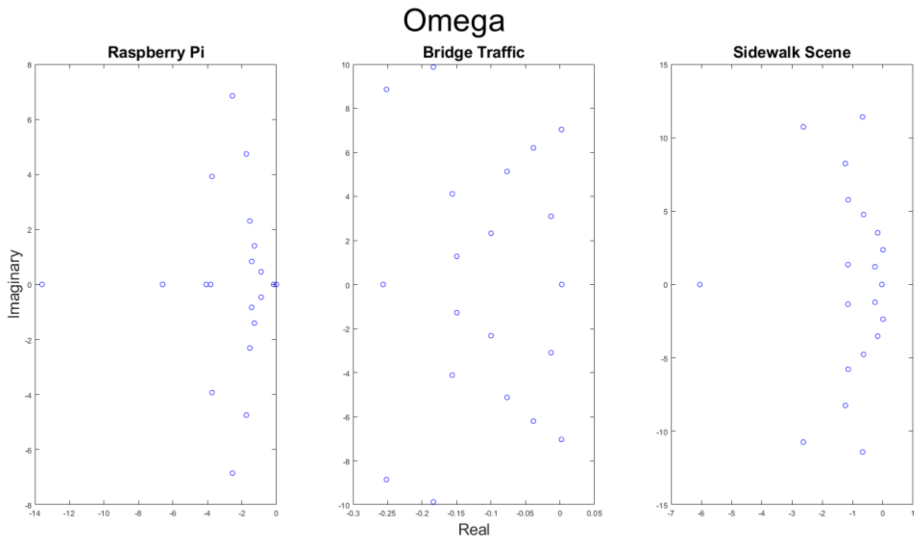

Figure 1: Frequencies near $(0,0)$ indicate DMD modes that have negligible evolutions over time. Two modes were used for the background of the Raspberry Pi video. Just one mode was used for the Bridge Traffic and Sidewalk Scene videos.

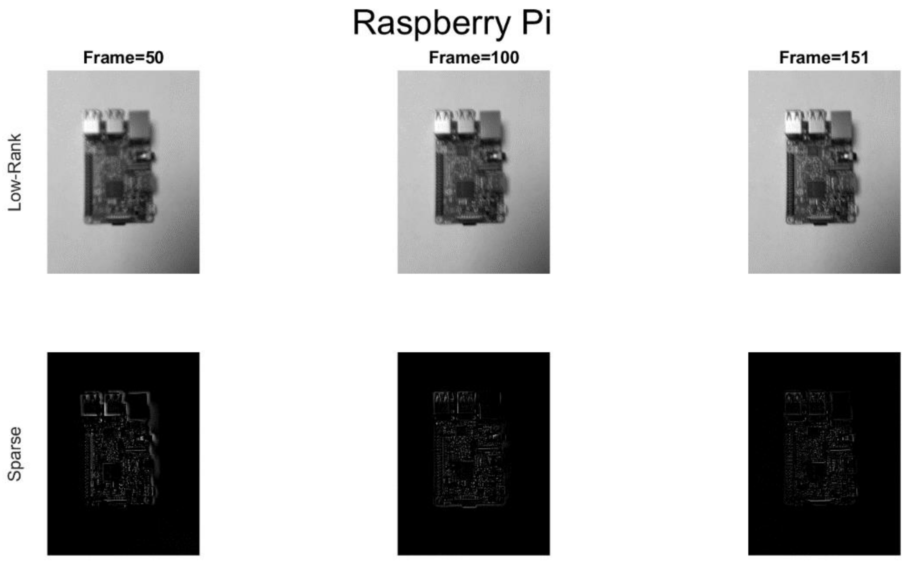

Figure 2: Holding a camera over a stationary object produces the illusion of movement so slight that the object is still captured in the background mode. The foreground extraction is clean, as the remaining modes are all devoted to describing the jitters induced by a hand-held camera. Note the ghosting in the first sparse frame.

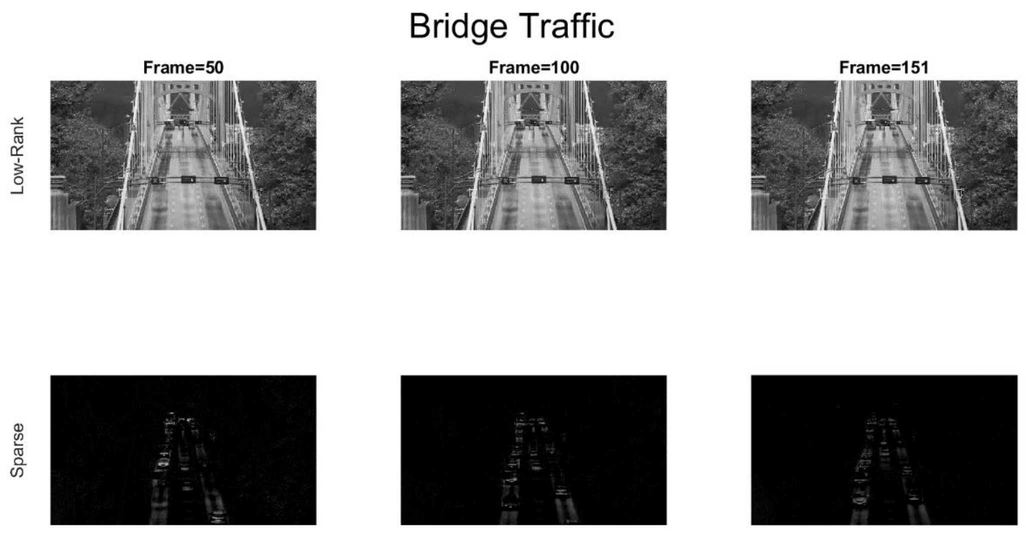

Figure 3: A stationary camera clearly captures the non-moving objects in the low-rank reconstruction. The steady stream of cars is somewhat blurred into three streams. The foreground reconstruction is very good despite the volume of cars, possibly because of the predictability of their locations.

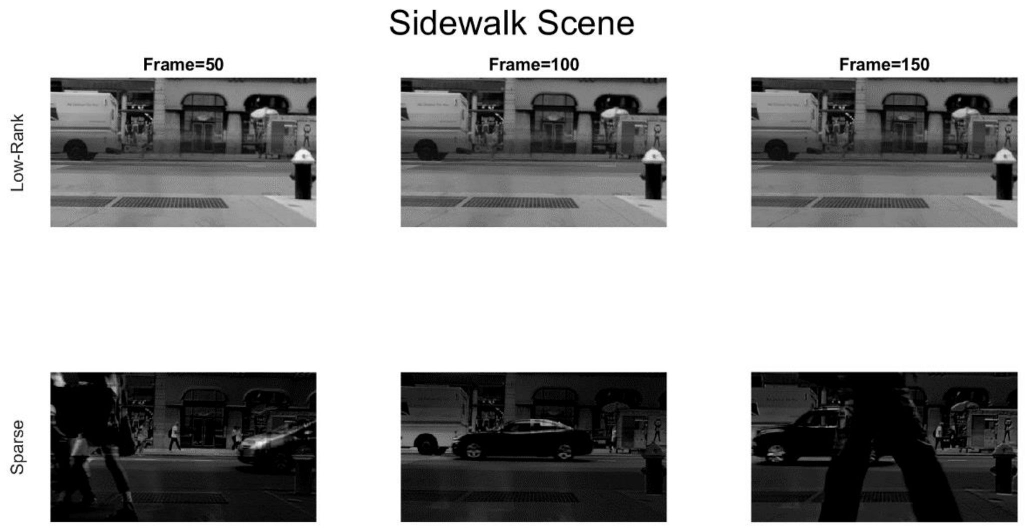

Figure 4: The sidewalk scene had a constant stream of pedestrians and vehicles. The low-rank reconstruction omitted all of them from the background. With so many objects passing in front of what is displayed in the low-rank reconstruction, the remaining modes necessarily had to evolve it over time. Attempts to completely isolate the moving objects in the foreground were unsuccessful, even when bypassing the SVD and/or increasing the number of background modes.

V. Summary and Conclusions

The degree of success achieved in separating the background from the foreground depends on many factors. Factors include the number of frames in the video, which frames are chosen in the sample, the number of modes used from the singular value decomposition, the number of dynamic modes used in the reconstruction of the background, and to a lesser extent, whether or not residual correction was used. The omega values were sensitive to the SVD rank-truncation.

Of the three videos investigated here, the best results were achieved on the Bridge Traffic video. The moving objects were similar in shape and confined to a small and predictable region. In contrast, the Sidewalk Scene was comprised of an assortment of people and vehicles that took up a large part of the frame and that moved quickly in and out of view. The background extraction was good, but the foreground extraction was poor. Finally, the Raspberry Pi video had excellent foreground extraction, but the perceived motion was slight and in reality, the object was stationary. All dynamic modes were served à la Pi.

The dynamic mode decomposition provides a powerful tool for capturing the time-evolution of complex data structures. It assumes no underlying equations and is especially powerful in its ability to handle non-linear systems. It can be coupled with singular value decomposition to reduce the rank of the data set with a minimal compromise to the integrity of the data and it is computationally fast. All of these attributes make DMD one of the most versatile model-fitting tools available in the science of data analysis.

A. Residual Correction

Empirical evidence has indicated that the separation of background and foreground can be made more mathematically rigorous by incorporating a residual matrix into the model.[4] The calculation of the background as $X_{sparse} = X - |X_{low}|$ has the potential to introduce negative values into the sparse matrix, which are not interpretable as pixel intensities. The negative residuals can be stored in a matrix R and X_{low} and X_{sparse} adjusted:

$$\begin{aligned} X_{low} &= |X_{low}| + R \\ X_{sparse} &= X_{sparse} - R \end{aligned} \qquad \textbf{(Eqn. A.1)}$$

These modifications ensure that all of the values in X_{low} and X_{sparse} are positive and preserves the full rank DMD reconstruction of the data matrix as $X = X_{low} + X_{sparse}$.

The residual adjustments were explored in this investigation. They failed to improve the foreground extraction and added objects into the background, as shown in **Figure A.1**. All of the output in Section IV was produced with residual correction on X_{sparse} only.

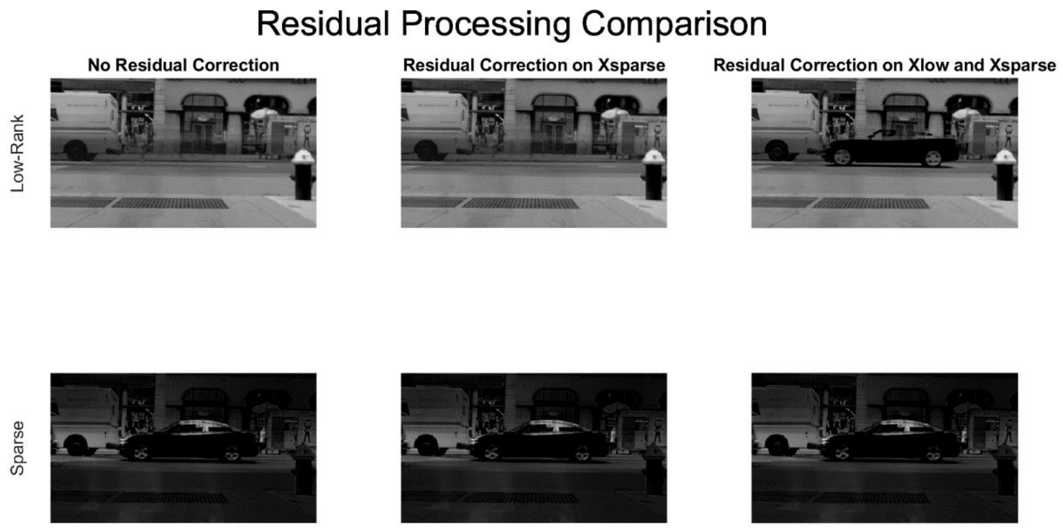
Residual Processing Comparison
No Residual Correction
Residual Correction on Xsparse
Residual Correction on Xlow and Xsparse
Low-Rank
Sparse

Figure A.1: There is no apparent difference between using no residual correction and using correction on just the sparse reconstruction. When using residual correction on both the background and the foreground, foreground objects are added into the background, but there are no obvious gains in reconstructing the sparse matrix.

B. MATLAB Scripts

```
% Homework 5 - Dynamic Mode Decomposition
% David Long
% 3.2.19

clear all; close all; clc

%% Video import

video7reader = VideoReader('hw5video7.mp4');
video7 = read(video7reader);
video7gray = uint8(zeros(size(video7, 1), size(video7, 2), size(video7, 4)));
for j=1:size(video7, 4)
    video7gray(:,:,j) = rgb2gray(video7(:,:,:,j));
end
video7gray = imresize(video7gray, [size(video7, 1)/4 size(video7, 2)/4]);

save('video7gray', 'video7gray')
```

```
% Homework 5 - Dynamic Mode Decomposition
% David Long
% 3.2.19

clear all; close all; clc

numVideos = 3;

for jjj=1:numVideos
    %% Reshape video stream
    switch jjj
        case 1
            load('video1gray.mat')
            subVid = video1gray(:,:,50:200);
            currentTitle = "Raspberry Pi";
        case 2
            load('video3gray.mat')
            subVid = video3gray(:,:,50:200);
            currentTitle = "Bridge Traffic";
        case 3
            load('video7gray.mat')
            subVid = video7gray(:,:,1:150);
            currentTitle = "Sidewalk Scene";
    end

    m = size(subVid, 1);
    n = size(subVid, 2);
    data = zeros(m*n, size(subVid, 3));
    for j=1:size(subVid, 3)
        data(:,j) = reshape(double(subVid(:,:,j)), m*n, 1);
```

```
end

% Periodic time domain
t = 1:size(subVid, 3);
dt = t(2) - t(1);




%% DMD

X1 = data(:, 1:end-1);
X2 = data(:, 2:end);
Xlow = []; Xsparse = []; time = [];

% SVD

[u, s, v] = svd(X1, 'econ');

r = min(20, size(X1, 2));
U = u(:,1:r);
S = s(1:r,1:r);
V = v(:,1:r);




%% Build Atilde and DMD modes

Atilde = U'*X2*V/S;
[W, D] = eig(Atilde);
Phi = X2*V/S*W;

% DMD spectra
lambda = diag(D);
omega = log(lambda)/dt;

figure(1)
subplot(1, numVideos, jjj)
plot(omega, 'bo')
title(currentTitle, 'Fontsize', 20)
if (jjj==1)
    ylabel('Imaginary', 'Fontsize', 20)
end
if (jjj==2)
    xlabel('Real', 'Fontsize', 20)
end
if (jjj==3)
    sgtitle('Omega', 'Fontsize', 40)
end
```

```
    % Initial amplitudes
    b = Phi\X1(:,1);

    % Sorting
    [~, index] = sort(abs(omega));
    omega = omega(index);
    b = b(index);
    Phi = Phi(:,index);

    for iter=1:length(t)
        time(:,iter) = b.*exp(omega*t(iter));
    end

    switch jjj
        case 1
            Xlow = Phi(:,1:2)*time(1:2,:);
            Xsparse = data - abs(Xlow);
        case 2
            Xlow = Phi(:,1)*time(1,:);
            Xsparse = data - abs(Xlow);
        case 3
            Xlow = Phi(:,1)*time(1,:);
            Xsparse = data - abs(Xlow);
    end




    %% Extract negative residuals from Xsparse

    R = Xsparse.*(Xsparse<0);

%     Xlow = R + abs(Xlow);
    Xsparse = Xsparse - R;




    %% Playback

    for j=1:size(Xlow, 2)
        dmdLow{jjj}(:,:,j) = uint8(real(reshape(Xlow(:,j), m, n)));
        dmdSparse{jjj}(:,:,j) = uint8(real(reshape(Xsparse(:,j), m, n)));
    end




   %% Frame display

    figure(jjj+1)

    for j=1:3
```

```
        subplot(2, 3, j)
        imshow(dmdLow{jjj}(:,:,floor((j/3)*size(Xlow, 2))))

        title(strcat('Frame=', num2str(floor((j/3)*size(Xlow, 2)))),
'Fontsize', 20)
        if (j==1)
            ylabel('Low-Rank', 'Fontsize', 20)
        end

        subplot(2, 3, j+3)
        imshow(dmdSparse{jjj}(:,:,floor((j/3)*size(Xlow, 2))))

        if (j==1)
            ylabel('Sparse', 'Fontsize', 20)
        end
    end

    sgtitle(currentTitle, 'Fontsize', 40)

end



%% Residual Processing Comparison
% clear Atilde b D data dmdLow dmdSparse dt index iter j lambda omega Phi r s
S ...
%     t time u U v V W X1 X2 Xlow Xsparse;

numTests = 3;
figure(5)

for jjj=1:numTests
    %% Reshape video stream
    switch jjj
        case 1
            data = zeros(m*n, size(subVid, 3));
            for j=1:size(subVid, 3)
                data(:,j) = reshape(double(subVid(:,:,j)), m*n, 1);
            end
            currentTitle = "No Residual Correction";
        case 2
            data = zeros(m*n, size(subVid, 3));
            for j=1:size(subVid, 3)
                data(:,j) = reshape(double(subVid(:,:,j)), m*n, 1);
            end
            currentTitle = "Residual Correction on Xsparse";
        case 3
            data = zeros(m*n, size(subVid, 3));
            for j=1:size(subVid, 3)
                data(:,j) = reshape(double(subVid(:,:,j)), m*n, 1);
            end
            currentTitle = "Residual Correction on Xlow and Xsparse";
    end
```

```
% Periodic time domain
t = 1:size(subVid, 3);
dt = t(2) - t(1);




%% DMD

X1 = data(:, 1:end-1);
X2 = data(:, 2:end);
Xlow = []; Xsparse = []; time = [];

% SVD

[u, s, v] = svd(X1, 'econ');

r = min(20, size(X1, 2));
U = u(:,1:r);
S = s(1:r,1:r);
V = v(:,1:r);




%% Build Atilde and DMD modes

Atilde = U'*X2*V/S;
[W, D] = eig(Atilde);
Phi = X2*V/S*W;

% DMD spectra
lambda = diag(D);
omega = log(lambda)/dt;

% Initial amplitudes
b = Phi\X1(:,1);

% Sorting
[~, index] = sort(abs(omega));
omega = omega(index);
b = b(index);
Phi = Phi(:,index);

for iter=1:length(t)
    time(:,iter) = b.*exp(omega*t(iter));
end

switch jjj
    case 1
        Xlow = Phi(:,1)*time(1,:);
        Xsparse = data - abs(Xlow);
    case 2
```

```
            Xlow = Phi(:,1)*time(1,:);
            Xsparse = data - abs(Xlow);

            % Extract negative residuals from Xsparse
            R = Xsparse.*(Xsparse<0);
            Xsparse = Xsparse - R;
        case 3
            Xlow = Phi(:,1)*time(1,:);
            Xsparse = data - abs(Xlow);

            % Extract negative residuals from Xsparse
            R = Xsparse.*(Xsparse<0);
            Xlow = R + abs(Xlow);
            Xsparse = Xsparse - R;
    end




    %% Playback

    for j=1:size(Xlow, 2)
        dmdLow{jjj+3}(:,:,j) = uint8(real(reshape(Xlow(:,j), m, n)));
        dmdSparse{jjj+3}(:,:,j) = uint8(real(reshape(Xsparse(:,j), m, n)));
    end




   %% Frame display

    subplot(2, 3, jjj)
    imshow(dmdLow{jjj+3}(:,:,floor(0.65*size(Xlow, 2))))

    title(currentTitle, 'Fontsize', 20)
    if (jjj==1)
        ylabel('Low-Rank', 'Fontsize', 20)
    end

    subplot(2, 3, jjj+3)
    imshow(dmdSparse{jjj+3}(:,:,floor(0.65*size(Xlow, 2))))

    if (jjj==1)
        ylabel('Sparse', 'Fontsize', 20)
    end

    sgtitle('Residual Processing Comparison', 'Fontsize', 40)

end
```

C. References

[1] Kutz, J. N. *Dynamic Mode Decomposition: An Introduction*. Department of Applied Mathematics, University of Washington, Seattle, WA 98195.

[2] Bridge, C. (2019). *Free stock video of bridge, cars, daylight*. *Videos.pexels.com*. Retrieved 13 March 2019, from https://videos.pexels.com/videos/cars-passing-on-a-bridge-852183

[3] Sidewalk, P. (2019). *Free stock video of cars, city life, pavement*. *Videos.pexels.com*. Retrieved 13 March 2019, from https://videos.pexels.com/videos/people-walking-by-on-a-sidewalk-854100

[4] Grosek, J., & Kutz, J. (2014). *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video*. *arXiv.org*. Retrieved 13 March 2019, from https://arxiv.org/abs/1404.7592