AMATH482: Background Subtraction in Video Streams

Shuying Zhang
email: shuying3@uw.com

Abstract

Dynamic Mode Decomposition (DMD) is a data-driven modeling method originated in fluid dynamics. This project explores the application of DMD in subtracting background of video streams.

1 Introduction and Overview

Dynamic Mode Decomposition is an algorithm defined to analyze high-dimensional fluid dynamics data. DMD builds model completely from data at different times. In this report, we explore the application of DMD in extracting background information of video streams.

The test data are three 10-second video clips. We process each video frames and subtract their background after DMD.

There are five sections in this report. The introduction and overview section gives a concise description of the topics in the report. The theoretical part provides background knowledge for SVD, DMD and how DMD is applied in background subtraction. In algorithm implementation and development, we will introduce how to process videos data in MATLAB and perform DMD. The computational results shows the results of subtracting background with the help of DMD. A summary is concluded in summary and conclusions. All the MATLAB code and related MATLAB commands are in appendix. One can also take a look at the sample video streams in appendix.

2 Theoretical Background

2.1 Singular Value Decomposition (SVD)

SVD is a factorization of matrix into three components and use them in many applications. The full SVD takes the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

where

$$\begin{aligned} &\mathbf{U} \in C^{m\times m} \text{ is unitary} \\ &\mathbf{V} \in C^{n\times n} \text{ is unitary} \\ &\mathbf{\Sigma} \in R^{m\times n} \text{ is diagonal} \end{aligned} \tag{2}$$

The diagonal entries of Σ has the property that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p = \min(m, n)$. We can also do diagonalization via SVD. The matrix Σ contains the information of principle components.

In general, one can apply SVD on every matrix. To compute SVD, in MATLAB, use command `[U,S,V] = svd(A)`.

2.2 Dynamic Modes Decomposition (DMD)

DMD builds the model based on data in the following ways: we collect the data at each measurements and store them as columns in a matrix say $x_1, x_2, ..., x_m$. Then we can construct two matrices as in Figure 1.

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}.$$

Figure 1: DMD Snapshots Data Matrices

Since the columns in X' just has Δt difference in time from X. A linear approximation of these two matrices is:

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}. \tag{3}$$

DMD computes the leading eigendecomposition of the best fit linear operator A in the above equation, and A is given by:

$$\mathbf{A} = \mathbf{X}'\mathbf{X}^{\dagger} \tag{4}$$

DMD modes are defined to be the eigenvectors of A, and they each has a corresponding eigenvalue of A.

2.3 DMD Algorithm

1. Take SVD of X and get U, S, V matrices.

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{5}$$

2. The matrix A can be calculated through a pseudoinverse of X using SVD.

$$\mathbf{A} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^* \tag{6}$$

In practice, we only compute a low-dimensional $\tilde{A}$ with r * r entries. This is the projection of full matrix A onto POD modes.

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1} \tag{7}$$

3. Eigendecompose $\tilde{A}$.

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda} \tag{8}$$

where $\mathbf{W}$ are the eigenvectors, and $\mathbf{\Lambda}$ contains the eigenvalues λ.

4. Reconstruct Eigendecomposition of A.

$$\Phi = \mathbf{X}'\mathbf{V}\Sigma^{-1}\mathbf{W} \tag{9}$$

The columns of Φ are eigenvectors of A (DMD modes).

The low-rank approximate solution of all time is given by:

$$\mathbf{x}(t) \approx \sum_{k=1}^{r} \boldsymbol{\phi}_k \exp\left(\omega_k t\right) b_k = \mathbf{\Phi} \exp(\mathbf{\Omega} t)\mathbf{b} \tag{10}$$

Where $\Omega = diag(\omega) = diag(\frac{log(\lambda)}{\Delta t})$, and b is the initial condition (first column of X).

2.4 Background Substraction

The full DMD of X can be decomposed into two parts:

$$\mathbf{x}_{\text{DMD}} = \underbrace{b_p \varphi_p e^{u_p t}}_{\text{Background Video}} + \underbrace{\sum_{j \neq p} b_j \varphi_j e^{\omega_j t}}_{\text{Foreground Video}} \tag{11}$$

We then can approximate $X_{DMD}^{low-rank}$:

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} = b_p \varphi_p e^{\omega_p \text{t}} \tag{12}$$

This is the background of our videos.

To subtract the background:

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \mathbf{X} - \left|\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}\right| \tag{13}$$

To avoid negative values in $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$, we can add the negative residuals back to $\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}$ and minus them again from $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$.

3 Algorithm Implementation and Development

3.1 Collect Videos

In this report, the test videos are three 10-second clips from Youtube. I download them using `youtube-dl`. All the videos are for use of this report only.

The process of analyzing three videos are pretty much the same. We only include the general steps here.

3.2 Load Videos

Similarly to what we did in PCA project, all the videos are made of frames. We read them into MATLAB. First, turn each frame into gray scale and reshape each frame as a vector. Then, we stack all the vectors together into a giant matrix where columns are frames. We also need to convert data type from uint8 to double.

3.3 DMD

Decide Reconstruction Rank Since we only perform low-rank reconstruction, the first step is to decide the rank. We do this by identifying the POD modes in the videos with the help of SVD on the video frame data. We eyeball the rank.

Perform Low-Rank Reconstruction To get our model, we need to build two matrices from the frame data.We build the matrices as described in the DMD Algorithm section and reconstruct the low-rank DMD .

We know the reconstructed matrix contains the background information. Subtract the absolute of them from the original image data to get foreground data.

Here, we don't need to worry about the negative values since in MATLAB, they will be scaled to 0.

Visualize use `imshow` to visualize the result. Notice, we convert data from double back to uint8.

4 Computational Results

4.1 Test 1

Figure 2: An Example From the Test Video 1

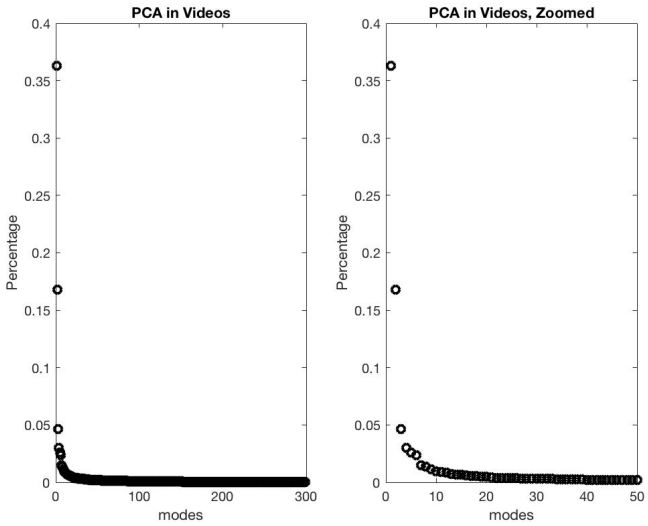

Figure 3: POD Modes in Video 1

One example frame from video 1 is in Figure 2. In this video, the background is the cat and the kitchen as the cat doesn't move. Then the man walks in and caresses the cat. After performing SVD, we decide our rank to be 20 according to the result in Figure 3. Figure 4 shows the result of the example frame. We can see that DMD algorithm successfully subtracts the background in this test.

Figure 4: Result of Video 1

4.2 Test 2

Figure 5: An Example From the Test Video 2

One example frame from video 2 is in Figure 5. After performing SVD, we decide our rank to be 40 according to the result in Figure 6. Figure 7 shows the result of the example frame. In this test, the result is not as good as in test 1. One possible reason is that this video is shoot on a bike, so every object is actually moving in video frames. We will make sure to pick test videos from a stationary camera in the future.

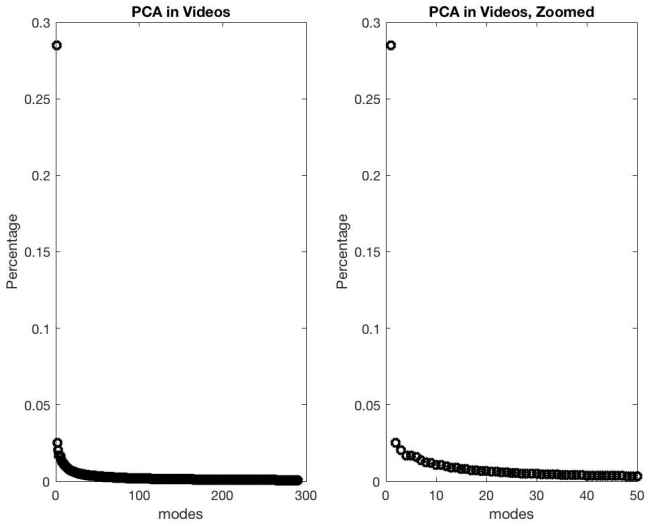

Figure 6: POD Modes in Video 2

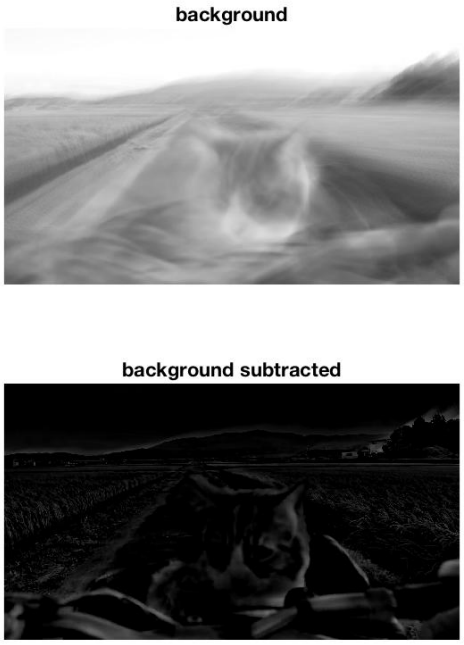

Figure 7: Result of Video 2

4.3 Test 3

One example frame from video 3 is in Figure 8. In this video, the background is some leaves. After performing SVD, we decide our rank to be 20 according to the result in Figure 9. Figure 10 exhibits the result of the example frame. We can see that DMD algorithm

Figure 8: An Example From the Test Video 3

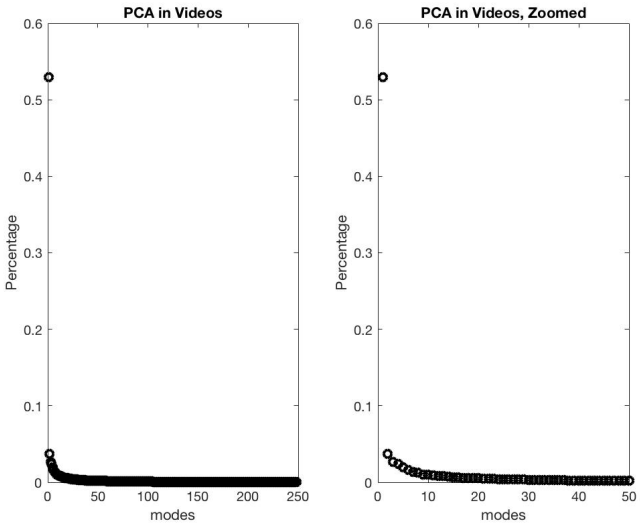

Figure 9: POD Modes in Video 3

successfully subtracts the background in this test. Only the head of the panda is identified as the foreground because the baby panda keeps moving his head while sitting there and eating bamboo.

5 Summary and Conclusions

In this project, we successfully use the DMD algorithm to subtract background in video streams. If the video is shoot by a stationary camera, the result is nice and just as expected. However, if the video data is noisy due to shaking cameras, it will be difficult to get a satisfactory result. Overall, DMD is a powerful method that greatly helps us analyzing high-dimensional data.

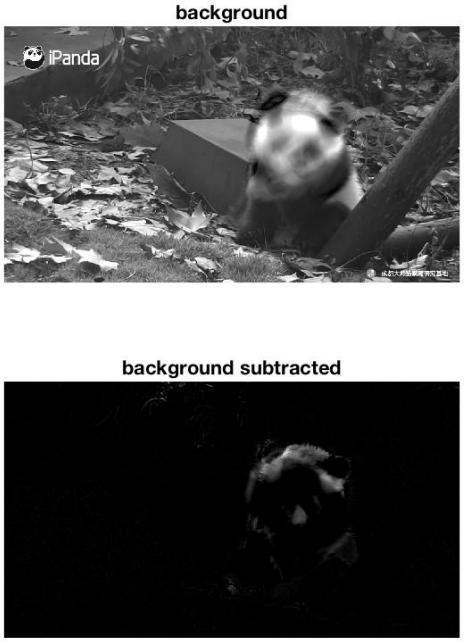

Figure 10: Result of Video 3

References

[1] Part3 : Computation Methods for Data Analysis. Data-Driven Modeling & Scientific Computation: Methods for Complex Systems Big Data, by J. Nathan Kutz, OUP Oxford, 2013, available at `http://faculty.washington.edu/kutz/582.pdf`.

A MATLAB commands

Documentation for those commands can be found in MATLAB using `help command_name`.

A.1 Video Loading

VideoReader(filename) Reads video into MatLab.

hasFrame(video), readFrame(video) iterates over the video frames and reads the data.

rgb2gray() Turns each frame into gray scale.

imshow(frame) Returns a frame in the video.

A.2 Data Modification

double(X) Change the data type in X to double.

uint8(X) Change the data type to image data type uint8.

A.3 Plotting

plot(X,Y) Plots 2-dim data.

subplot(i,j,k)) Draw multiple plots in one figure. i and j specifies the lay out, k is the index of the plots.

A.4 SVD

svd(X), svd(X,'econ') find the SVD of matrix X. Returns u, s, v. One can add 'econ' to perform a reduced SVD rather a full SVD.

B MATLAB code

```
clear all; close all; clc

% load videos
vid = VideoReader('videos/pandaclip.mp4');
% vid information
wid = vid.Height;
ht = vid.Width;

dt = 1/vid.FrameRate;
t = 0: dt : vid.Duration;
%turn into each frame into rgb and reshape to columns
frames = [];
while hasFrame(vid)
    rgb = rgb2gray(readFrame(vid));
    frames = [frames, rgb(:)];
end
numFrames = size(frames,2);
%show example
%%
imshow(reshape(frames(:,100),wid,ht))
title('Example')
%% build two X
X1 = double(frames(:,1: end - 1));
X2 = double(frames(:,2: end));
frames = double(frames);

%view PCA to decide r
[u,s,v] = svd(frames,'econ');
subplot(1,2,1)
plot(diag(s/sum(diag(s))),'ko','Linewidth',[2])
title('PCA in Videos'); xlabel('modes'); ylabel('Percentage')
subplot(1,2,2)
plot(diag(s(1:50,1:50))/sum(diag(s)),'ko','Linewidth',[2])
title('PCA in Videos, Zoomed'); xlabel('modes');ylabel('Percentage');

%r = 20;
%% svd X1
r = 20;
[u2,s2,v2] = svd(X1,'econ');
u = u2(:,1:r);
s = s2(1:r,1:r);
v = v2(:,1:r);

%% DMD low rank
Atilde = u' * X2 * v/s;
[W,D] = eig(Atilde);
phi = X2 * v/s*W; %modes
```

```
lamda = diag(D);
omega = log(lamda)/dt;
b = phi\X1(:,1); %pseuso inverse initial condition

%%
u_modes = zeros(r, length(t));
for i = 1:length(t)
    u_modes(:,i) = b.*exp(omega*t(i));
end
u_dmd = phi*u_modes; %low-rank dmd

%%
bg = abs(u_dmd);
%%
subplot(2,1,1)
imshow(uint8(reshape(bg(:,100),wid,ht)))
%imshow(uint8(reshape(bg(:,100),720,1280)))
title('background')

subplot(2,1,2)
sp = frames - bg(:,1:numFrames);
imshow(uint8(reshape(sp(:,100),wid,ht)))
%imshow(uint8(reshape(sp(:,100),720,1280)))
title('background subtracted')
```

C Videos

The test videos are 10-second clips from the following videos.

Kitchen and Cat https://www.youtube.com/watch?v=Za8qW-n04T8

Cat in Bike https://www.youtube.com/watch?v=5530I$_pYjbo$

Panda https://www.youtube.com/watch?v=Tfe9GP3MyGM