Background Foreground Speration using Dynamic Mode Decomposition

Jiaqi Zhang
AMATH 482 Winter 2017
University of Washington
zjq95@uw.edu

Abstract

Saving data as video footages is very common these days. These videos are complex data systems. Using data-driven modeling on these complex systems, we usually find out that they base on low-dimensional attractors. In this report, we use the *Dynamic Mode Decomposition* (DMD) technique to separate the foreground and background part of one video footage. Separating these parts can be use on surveillance services, or just to save the storage memory.

I. Introduction

This report will first of all, discuss the theoretical background of the DMD technique mathematically.Followed by that would be the algorithm implementation. We will discuss in detail how to implement this algorithm on a given video. In this report, we use MATLAB to perform the algorithm. Two video clips are used to test the performance of this algorithm. In the last section, we will discuss the computational results and give a summary.

II. Theoretical Background

DMD extracts key low-rank spatio-temporal features of many high dimensional systems[1]. DMD analyzes these systems without necessarily knowing the equations bounding the systems. It processes data collected in snapshots and analyze the current state or predict next state of the dynamic system.

For data with large sizes, its matrix *A is* very hard and time consuming to analyze directly. Therefore we use the DMD algorithm to analyze the rank-reduced eigendecomposition representation of the matrix *A.*

1.SVD

First of all, we take the *Singular Value Decomposition* (SVD) of the data matrix *X*:

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

U and *V* are the left and right singular vevtors of ***X***. The diagonal of Σ are the corresponding singular values. Let *r* be the rank of the reduces SVD approximation to *X*. *r* must be smaller than the total number of modes, which is the number of elements in Σ.

2. Construction of Linear Operator *A*

In pratice, we compute $\tilde{A}$, the *r x r* projection of the full operator onto its low-rank modes:

$$\tilde{A} = U * AU = U * X'V\Sigma^{-1} \tag{2}$$

$\tilde{A}$ defines a low-dimensional linear model of the dynamic system:

$$\tilde{x}_{k+1} = \tilde{A}\tilde{x}_k \tag{3}$$

In order to reconstruct the high-dimensional state of the system, we use:

$$x_k = U\tilde{x}_k \tag{4}$$

3. Eigendecomposition of $\tilde{A}$

First we find the eigenvector and eigenvalues of $\tilde{A}$. Then we use th eigenvector of A to construct Φ:

$$\Phi = X'V\Sigma^{-1}W \tag{5}$$

The columns of Φ are DMD modes, which are the low-rank eigenvectors of A. The eigenvalues of $\tilde{A}$ are the low-rank eigenvalues of A.

4. Reconstruction of engendecomposition of A

Now that we have both the eigenvalue and eigenvector of $\tilde{A}$, we are able to reconstruct the future solution. The approximate solution is given by:

$$x(t) \approx \sum_{k=1}^{r} \Phi_k e^{\omega_k t} b_k = \Phi e^{\Omega t} b \tag{6}$$

b_k represents the initial amplitude of each mode. Φ's columns are the DMD eigenvectors Φ_k. ω_k are the eigenvalues.

In order to compute the initial amplitudes b_k, we consider the initial snapshot *x1* at time *t = 0*. We have

$$x_1 = \Phi b \text{ and therefore } b = \Phi^+ X_1 \tag{7), (8}$$

Φ^+ is known as the Morore-Penrose pseudo inverse of Φ. It is used here because Φ is not a square matrix. This pseudo inverse will give the best-fit solution of b, which means that the least square error $||x_{k+1} - Ax_k||_2$ is minimized.

5. Seperation of Background and Foreground

The eigenvalues ω_k that are close to 0 ($||\omega_m|| \approx 0$) corresponds to the background information of the system. Therefore, we take the smallest eigenvalue of the system and find its index:

$$[\omega_m, index] = min(\omega) \tag{9}$$

Then we find the corresponding eigenvector Φ_{index} based on the index given and separate X_{DMD} by:

$$X_{DMD} = \sum_{j=1}^{k} b_j \varphi_j e^{\omega_j t} = \Phi \Omega^t b = b_m \varphi_m e^{\omega_m t} + \sum_{j \neq m}^{k} b_j \varphi_j e^{\omega_j t} \quad (9)$$

$b_m \varphi_m e^{\omega_m t}$ = Background video

$\sum_{j \neq m}^{k} b_j \varphi_j e^{\omega_j t}$ = Foreground video (10)

III. Algorithm Implementation

The first video used is shot and published by 'The Slomo Guys'. In the video, a glass bottle full of flammable fluid is thrown against the wall so that the wall catches fire. The video is shot in slow motion so that we can see the fire expanding on the wall.
The second video used is published by the BBC. The video starts by a man blowing air into a balloon and eventually the balloon explodes. This video is also shot in slow motion.

1. Video Loading

We load the video to MATLAB, and extract its image dimension - **height** and **width**, number of frames per second - **frameRates**, as well as the number of frames - **numFrames**.

2. Constructing Data Matrix X

First of all, the video is converted into grayscale and the number formats are set to double. Then we use a for loop to loop through each frame of the matrix. Each frame of the grayscale video will be a matrix of the dimension (**height, width**). We reshape this matrix into a column vector and append each column vector to the matrix X. After the for loop, we will obtain a complete data matrix *X* with dimension (height * width, numFrames).

3. Construct the matrix X1 and X2

Now we will construct the matrices *x1* and *x2*, which will be used in the DMD algorithm. *X1* is composed of the **1** to **numFrames - 1** columns of *X*, and *X2* is composed of the **2** to **numFrames** columns of *X*.

4. SVD on X1

Performing SVD on X1, we will get the eigenvalues and eigenvectors of the matrix.

5. Build A tilde and DMD modes

We construct A tilde as mentioned in equation 2. The we use the eigenvalue of eigenvector of A to construct the DMD modes as mentioned in equation 5.

6. Compute Φ, Φ_bg, **b and b_bg.**

b is computed using equation 8. Note that since Φ is not a square matrix, we use the MATLAB command 'pinv' to find its inverse. b_bg is the low-rank mode which corresponds to the background. b_bg is defined to be the smallest element in b. We locate **b_bg** and its **index** using the command 'min'. Φ_bg is located at Φ[index].

7. Construct background matrix X_bg

Now that we have Φ_bg and b_bg, we are able to construct the background matrix X_bg. We start a for loop that loops through each column of the matrix. Note that each column corresponds to each frame of the video. Using equation 10, we are able to get obtain the dynamics of each frame. After the for loop, we multiply Φ_bg with the time_dynamics matrix to obtain the background matrix X_bg.

8. Construct foreground matrix X_fg

We can see in equation 9 that the background matrix and foreground matrix add up to the original matrix. Therefore in order to obtain X_fg, we subtract X_bg from the original matrix X.

9.Reshaping and Displaying

First of all, we convert the double format matrices back to the 'unit8' format. Then we reshape the each frame of the matrix back into its original (length, width). The result matrix will display the background/foreground video at each frame.

IV. .Computational Results

The algorithm is used to separate the background and foreground of two videos. This section will show the computed results.

Looking at the background (figure 1c) of the first video, we can see the man blowing the balloon, and the shadow of the balloon before it exploded overlapping with the residual after the explosion. Note that the original video was in slow motion, the balloon stayed un-exploded for a long period of time, then followed by a fast period of explosion and then after the explosion there were some leftovers of the balloon. The long period of before/after state of explosion explains the overlapping figure in the background. The bright part of the foreground (figure 1e) shows the balloon and a very dim figure of the person. The figure of the person was in the foreground is due to the fact that the person in the original video was

shaking and moving. When looking at the foreground video, we can actually see the balloon exploding and the dim image of person shows and disappears from time to time.

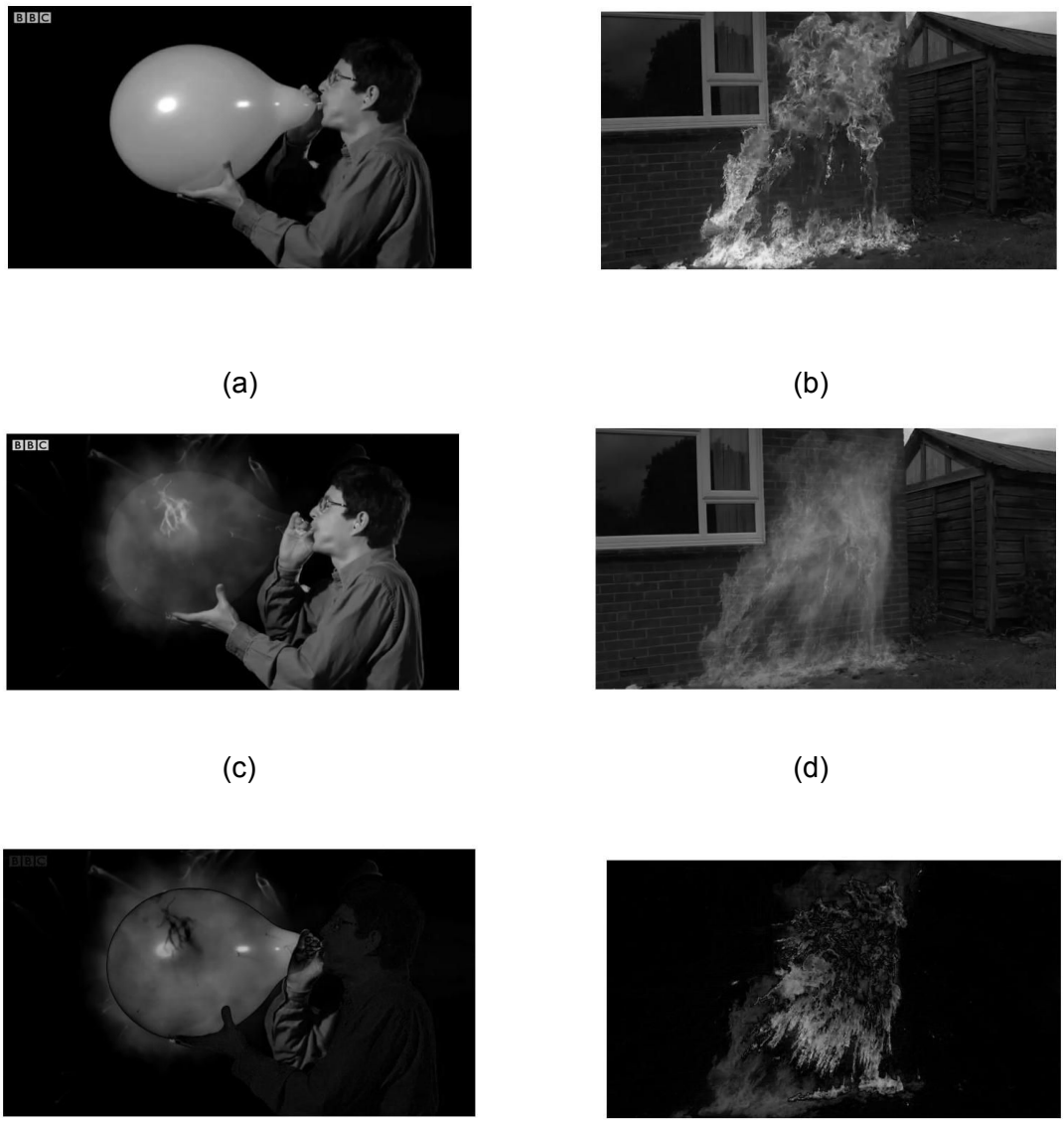

Figure 1. A snapshot of the original, background and foreground of the first and second video.

Looking at the figures for the second video, in which a flammable bottle of liquid is thrown against the wall and started fire on the wall, the snapshots are taken after the wall is on fire. In figure 2d, the background for this video, we can see the wall, the window, as well as part of the flame. In the original footage, the period of 'throwing the bottle' was very short. Then the wall caught on fire pretty fast and stayed on fire for the rest of the footage. Therefore the part of the fire was recognized as the background of the video. In figure 2f, the foreground of the video, we can see that it contains only the flame since it was the only moving part. The rest - window, wall, cabin next to the wall - are all separated from the footage.

V. Summary and Conclusions

In this report, the mathematical implementation and performance of the DMD algorithm on the video background and foreground separation are discussed. This algorithm uses SVD and low rank decomposition to separate the original matrix into the background and foreground part. Two videos are used to test the performance and the resulting separated videos are pretty distinct. It is demonstrated that the implementation is low-cost and the performance of the DMD algorithm is visually good.

Reference

[1] Grosek, J. Kutz, JN. *"Dynamic Mode Decomposition for Real-Time Background/Foreground Speration in Video"* 2014.

Appendix A MATLAB functions used and brief implementation explanation

Imshow - Displays the matrix as a image. Used to read the matrix as a video.

Rgb2gray - Converts colorscale videos to grayscale.

Double - Converts the format of the values into ‘double’.

Reshape - Reshape the given matrix into the specified dimension.

Svd - Perform the singular value decomposition on the matrix.

Size - Returns the size of the matrix as a column.

Pinv - Perfroms the Morore-Penrose pseudo inverse on the matrix.

Abs - Returns the absolute value of the specified element.

Min - Returns the minimum value in the given matrix.

Appendix B MATLAB codes

```
clc; clear all;
obj=VideoReader('video.mp4');
vidFrames = read(obj);
numFrames = get(obj,'numberOfFrames');
dt = get(obj,'FrameRate');
%% Reconstruct the video
X = [];
for k = 1 : numFrames
gray(:, :) = double(rgb2gray(vidFrames(:,:,:,k)));
[width, length] = size(gray);
X = [X, reshape(gray,[width*length, 1])];
end

%%
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);
[U, S, V] = svd(X1, 'econ');
r = min(50, size(U, 2));% rank truncation
Ur = U(:, 1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);
%% Build Atilde and DMD Modes
Atilde = Ur'*X2*Vr/Sr;
[W, D] = eig(Atilde);
Phi = X2*Vr/Sr*W; % DMD Modes
%% DMD Spectra
t = linspace(0, numFrames*dt, numFrames);
%%
lambda = diag(D);
omega = log(lambda)/dt;
b = pinv(Phi)\X(:, 1);

%% Seperation
[omega_bg, min_index] = min(abs(omega));
Phi_bg = Phi(:,min_index);
%% Compute DMD Solution
%% X_bg
b = pinv(Phi_bg)\X(:, 1);

time_dynamics_bg = [];
for iter = 1: numFrames
      time_dynamics_bg= [time_dynamics_bg, b*10*exp(omega_bg*t(iter))];
end
X_bg = Phi_bg * time_dynamics_bg;

%% X_fg
X_bg(X_bg < 0) = 0;
X_fg = X-X_bg;

%% Play Video
```

```
for k = 1: numFrames
      colormap gray;
      imagesc(reshape(abs(X_bg(:, 2)), [width, length]));
end
```