AMATH 482 Homework 5

Erik Huang

March 16, 2021

Abstract

Experiment the application of Dynamic Mode Decomposition on video.

1 Introduction and Overview

We will be processing two short video clips using the Dynamic Mode Decomposition method. These videos both contain a moving foreground object and a steady background. Our task is to separate the foreground apart from the background.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

The concept of Singular Value Decomposition is used in the last two assignments. The SVD is a factorization of a matrix in linear algebra, and one of the most powerful and versatile tool in data analysis. The singular value decomposition of a $m \times n$ matrix A is:

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

$\mathbf{U}$ is a $m \times n$ unitary matrix with orthonormal columns. $\mathbf{\Sigma}$ is a $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal. $\mathbf{V}$ is a $m \times n$ complex unitary matrix. $\mathbf{U}$ and $\mathbf{V}$ contain information on rotation and $\mathbf{\Sigma}$ describes the stretch in each direction. Applying SVD on data set allows us to get the principal components from the data.

2.2 Dynamic Mode Decomposition(DMD)

The Dynamic Mode Decomposition method is a dimensionality reduction algorithm. Most real-world data sets are full of high-dimensional variables but viewing at the right way, we only need to consider a few observable variables to describe the data. DMD can compute a set of modes which are associated with fixed oscillation frequencies. This spectrum of frequencies can be used to subtract background modes. Let w_p, $p \in \{1, 2, \ldots, \ell\}$, satisfy $\|\omega_p\| \approx 0$, we can define the sequence of DMD $\mathbf{X} \in \mathbb{R}^{n\times m}$ as:

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

This expression can be interpreted as separating DMD terms into approximate low-rank and sparse reconstructions. With definitions:

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

and

$$\mathbf{X}_{\text{DMD}}^{S\text{ parse}} = \sum_{j\neq p} b_j\boldsymbol{\varphi}_j e^{\omega_j \mathrm{t}}$$

Although both terms of this DMD reconstruction are complex, they sum to a real-valued matrix. Therefore, we can calculate one of them as a real-valued matrix in the following way:

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \leftarrow \mathbf{R} + \mid \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \mid$$
$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} \leftarrow \mathbf{X}_{\text{DMD}}^{\text{Sparse}} - \mathbf{R}$$

In this expression, The residual negative values from the $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$ are taken out and put into matrix $\mathbf{R} \in \mathbb{R}^{n\times m}$. The operation $\mid \mathbf{X} \mid$ yields the modulus of each element within the matrix $\mathbf{X}$. The result from this expression will not have value below zero. In other words, there will not be negative pixel intensities. We will see the effect of this reconstruction as we plot processed video frames later.

3 Algorithm Implementation and Development

The DMD algorithm utilizes low dimensionality in the data to create a low-rank approximation of the linear mapping. The algorithm for DMD is as follows:

1. Input data should be evenly spaced in time by a fixed time step. This data is used to create matrix $\mathbf{X}$
2. Using matrix $\mathbf{X}$, construct submatrices $\mathbf{X}_1^{M-1}$ and $\mathbf{X}_2^{M}$.
3. Perform SVD on $\mathbf{X}_1^{M-1}$
4. Create matrix $\tilde{\mathbf{S}} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}$ and perform eigenvalue decomposition.
5. Use $\mathbf{X}_1$ and the pseudoinverse of Ψ to find coefficients b_k.
6. Forecasting using the DMD modes along with their projection to the initial conditions.

When applying this algorithm in this assignment, here is the general idea:

1. Process (grayscale, reshape, convert to double) video input frames.
2. Apply the full DMD algorithm above.
3. Once we have the background modes, subtract the original video with it to obtain the foreground object

4 Computational Results

After performing the initial SVD on both videos, I plotted the energy capture of the first 10 modes. In Figure 1 and 2, we can see that the first mode is extremely weighted in both cases. This also gives me a hint that I could use fewer modes when performing the DMD. In both cases, the first mode captures more than 99% of the energy.
Figure 3 and 4 displays the result of DMD on both video. We can see that the separation is well done and the moving object can be easily distinguished. Although the person in the ski drop video is so small, the algorithm still manages to capture the entire movement.

5 Summary and Conclusions

Although the results from DMD are not perfect separations, they still exceeds my expectation. As a data-driven method, this technique of analysis only requires the computational cost of one singular value decomposition. I suppose this method will be one of the superior separation methods at mass-processing of data sets.

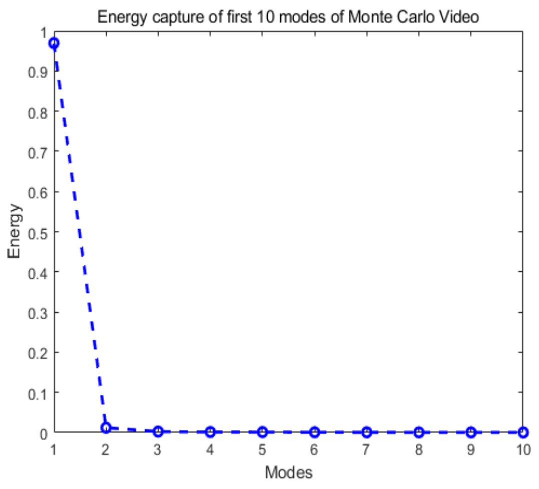

Figure 1: First 10 modes of Monte Carlo video

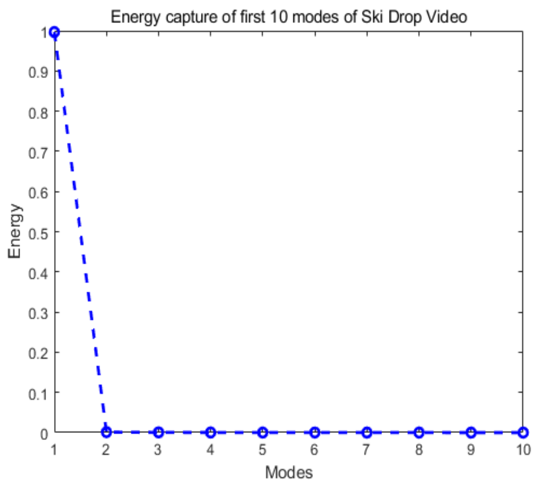

Figure 2: First 10 modes of ski drop video

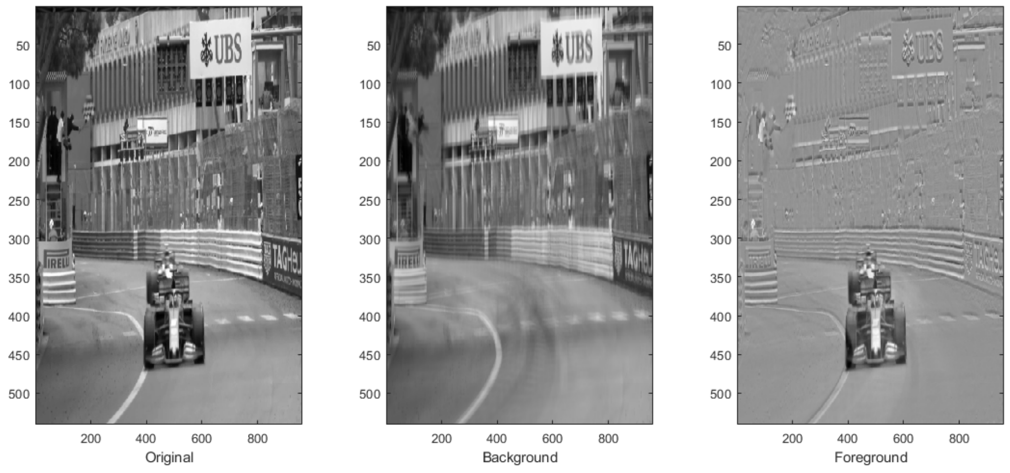

Figure 3: Here is a demo of Background - Foreground deconstruction on Monte Carlo

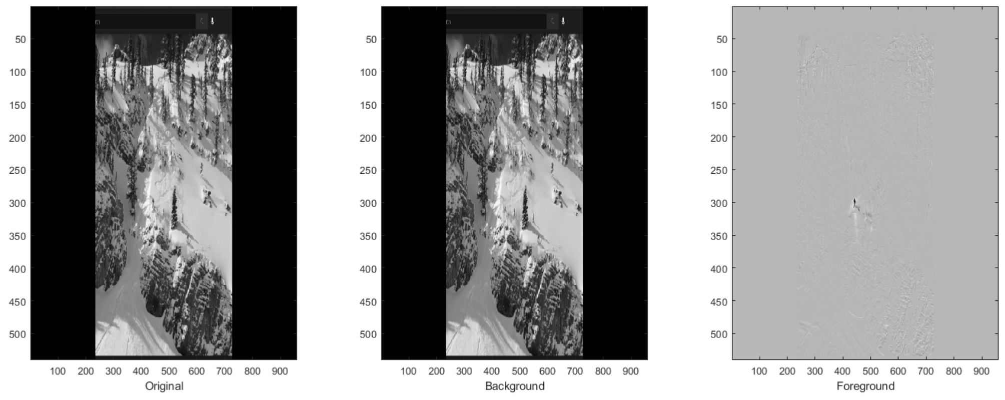

Figure 4: Here is a demo of Background - Foreground deconstruction on Ski Drop

Appendix A MATLAB Functions

unordered list:

- `VideoReader` Create object to read video files.
- `im2double` Convert image to double precision.
- `rgb2gray` Convert RGB image or colormap to grayscale.
- `reshape` Reshape array using the size vector.
- `eig` Eigenvalues and eigenvectors.
- `imagesc` Display image with scaled colors.

Appendix B MATLAB Code

```
clear all; close all; clc;

%% Input data
videoInput = VideoReader('ski_drop_low.mp4');
dt = 1/videoInput.Framerate;
t = 0:dt:videoInput.Duration;
vidFrames = read(videoInput);
numFrames = get(videoInput,'NumFrames');
frame = vidFrames(:,:,:,1);
V = zeros(size(frame,1)*size(frame,2), numFrames);
for iter = 1:numFrames
    frame = vidFrames(:,:,:,iter);
    frame = im2double(frame);
    frame = rgb2gray(frame);
    frame = reshape(frame,[],1);
    V(:,iter) = frame;
end

%% SVD
X1 = V(:,1:end-1);
X2 = V(:,2:end);

[U2, S2, V2] = svd(X1, 'econ');
lambda = diag(S2).^2;
rank = size(U2,2);

%% Plot SVD Results
all_ranks = lambda/sum(lambda);
plot(1:10, all_ranks(1:10), 'bo--', 'Linewidth', 2);
title("Energy capture of first 10 modes of Monte Carlo Video");
xlabel("Modes"); ylabel("Energy");

%% DMD
rank = 10; % Use the first 40 modes only
rang = 1:rank;
S2 = S2(rang,rang);
U2 = U2(:,rang);
V2 = V2(:,rang);
Stilde = U2' * X2 * V2 * diag(1./diag(S2));

[eV, D] = eig(Stilde);
Phi = X2 * V2 / S2 * eV;

y0 = Phi \ V(:,1);
u_modes = zeros(rank, length(X1(1,:)));
omega = log(diag(D));
for iter = 1:length(X1(1,:))
```

```
    u_modes(:,iter) = y0.*exp(omega*iter);
end
u_dmd = Phi*u_modes;
u_dmd = abs(u_dmd);
u_sparse = X1 - u_dmd; % Get foreground

%% Plot DMD Results
h = 540;
w = 960;
f = 200;
colormap(gray);

subplot(1,3,1)
original = reshape(V(:,f), h, w);
imagesc(original);
xlabel('Original');

subplot(1,3,2)
background = reshape(u_dmd(:,f), h, w);
imagesc(background);
xlabel('Background');

subplot(1,3,3)
foreground = reshape(u_sparse(:,f), h, w);
imagesc(foreground);
xlabel('Foreground');
```