Background Subtraction in Video Streams

Yichen Shen

March 14, 2021

Abstract

This paper is going to explore the usefulness of Dynamic Mode Decomposition method in separating the given video stream to both its foreground video and its background, which indicates the moving objects and its fixed background. We will apply the SVD method to decompose the video matrix and essentially we are going yo utilize the DMD method to separate the video accordingly. Finally, we are going to compare our separation results next to each other and see how DMD perform in the background subtraction.

1 Introduction and Overview

1.1 Background and Context

Unlike many other model-based algorithm, Dynamic model decomposition allows the user to process the low dimensional experimental data without using a set of governing equations. Moreover, DMD finds a basis of spatial modes in which time dynamic are exponential functions. Hence, learning to use the DMD method rather than the equation orientated decomposition method is essential to simplify problems into some data driven problem. Furthermore, in video subtraction, using DMD method can extract the spatial movement of the object and decompose it from the background.

1.2 Topic Importance

Data can contain huge dimension of information in its time and space. When dealing with a high dimensional data, it is possible to decompose it into low dimensional data by using the dynamic mode decomposition. In this way, the dimension is being reduce spatial and the time is being transform. Video is a common data that contains high dimension information, hence adapting DMD in decomposing a video is essential to learn in order to understand the mechanisms of DMD.

1.3 Objectives of Research

In this paper, we are given two videos of movement, one is the Monte Carlo racing car clip and the other is a skiting drop motion video. We are going to perform the singular value decomposition first to find the correspond singular value and the energy being captured. Then we will apply the DMD method and we can work with the low dimension data in order to separate the video in two compositions. Lastly, we are going to compare the outcome of separations, and see how effective the DMD is done on the separation.

2 Theoretical Background

2.1 Dynamic Mode Decomposition

2.1.1 Spatial and Time Data

When dealing with a spatio-temporal data, we can first generate a collection of vectors with change of time, then by determine its time interval, we can define a matrix with the increment of time:

$$U(x,t_m) = \begin{bmatrix} U(x_1,t_m) \\ U(x_2,t_m) \\ ... \\ U(x_n,t_m) \end{bmatrix} \tag{1}$$

in which, t_m is the increment in time and x correspond to the spacial information. When we can take these information to form columns of data matrices:

$$X = \begin{bmatrix} U(x,t_m) & U(x,t_m) & ... & U(x,t_M) \end{bmatrix} \tag{2}$$

and

$$X_j^k = \begin{bmatrix} U(x,t_j) & U(x,t_{j+1}) & ... & U(x,t_k) \end{bmatrix} \tag{3}$$

in which, X_k^j is the extract columns j through k in the full matrix X.

2.1.2 Koopman Operation

The dynamic Mode Decomposition approximates the modes of Koopman operation, so here we aer going to take a look at the Koopman operator. A is a linear time-independent operator such that:

$$x_{j+1} = Ax_j, \tag{4}$$

in which, j is the time information and A maps the data from time t_j to t_{j+1}. The vector x is an N dimensional vector that are collected at time j. By apply A to a piece of data can be forward in time by the time increment that allows us to use linear mapping from one time-step to the next one with a nonlinear dynamic of system.

What is worth to notice is that the Koopman operator is global meaning there is no restriction on space or time.

2.1.3 Dynamic Mode Decomposition Operation

With Koopman operator, we a re able to rewrite a matrix

$$X_1^{M-1} = \begin{bmatrix} x_1 & x_2 & x_2 & ... & x_{M-1} \end{bmatrix} \tag{5}$$

into:

$$X_1^{M-1} = \begin{bmatrix} x_1 & Ax_1 & A^2x_1 & ... & A^{M-2}x_1 \end{bmatrix} \tag{6}$$

which are done by applying powers of A to vector x_1. The first element $M-1$ elements can be relate to x_1 by using the Koopman operator:

$$X_2^M = AX_1^{M-1} + re_{M-1}^T; \tag{7}$$

However, the last point x_M was not include in the Krylov basis, so we need to add the residual vector r into the formula. In order to do so, we are using SVD to rewrite X_1^{M-1} into $U\Sigma V^*$, and that

$$X_2^M = AU\Sigma V^* + re_{M-1}^T; \tag{8}$$

Since our goal is to find A, we need to choose A in a way that column of X_2^M can be written as linear combination of column of U. Given $U * r = 0$, and that we isolate U^*AU to get:

$$UX_2^M = U^*AU\Sigma V^*; U^*AU = U^*X_2^M V\Sigma^-1 \tag{9}$$

in which, $U^*X_2^M V\Sigma^-1$ is $\hat{S}$. Since we don't need the full matrix but the first few nonzero singular value on the diagonal, we need to find a relative small value K to lower the dimension of the matrix.

Notice that $\hat{S}$ and A are similar and they share the same eigenvalues, therefore, if y is the eigenvector for $\hat{S}$, then U_y is the eigenvector of A, hence:

$$\hat{S}y_k = \mu_k y_k; \tag{10}$$

and the eigenvector if A can be written as:

$$\psi_k = U y_k; \tag{11}$$

Now we expand the eigenbasis and get:

$$x_{DMD}(t) = \Sigma_{k=1}^{K} b_k \psi_k e^{\omega_k t} = \Psi diag(e^{\omega_k t}) b \tag{12}$$

where K is the rank and b_k is the initial amplitude of each mode while Ψ is the matrix of eigenvector.

Lastly, we are able to compute the initial amplitude of each mode using the eigenvector matrix Ψ because we know the initial condition $t = 0$:

$$x_1 = \Psi b \rightarrow b = \Psi^{\dagger} x_1; \tag{13}$$

2.1.4 DMD in Video Subtraction

The spectrum of DMD can be use to subtract the background modes, we assume that ω_p satisfies $||\omega_p|| \approx 0$ and that $||\omega_j|| \forall j \neq p$ is bounded away from zero, and that

$$X_{DMD} = b_p \varphi_p e^{\omega_p t} + \Sigma_{j \neq p} b_j \varphi_j e^{\omega_j t} \tag{14}$$

where the first part represent the background video and the later part represent the foreground video.

Knowing that the matrix can be turned into an approximate low rank and sparse reconstruction we can do the following:

$$X_{Low-RankDMD} = b_p \varphi_p e^{\omega_p t}, X_{SparseDMD} = \Sigma_{j \neq p} b_j \varphi_j e^{\omega_j t} \tag{15}$$

then the sparse reconstruction can be calculate as:

$$X_{SparseDMD} = X - |X_{Low-RankDMD}| \tag{16}$$

the $||$ element result in the modulus of each elements and this can cause the $X_{SparseDMD}$ to have negative entries indicating nonsense pixel intensity. Therefore, we are adding a matrix R which contains the negative entries:

$$X_{Low-RankDMD} \leftarrow R + |X_{Low-RankDMD}|, X_{sparseDMD} \leftarrow X_{sparseDMD} - R \tag{17}$$

After this process we are able to turn the negative pixels into non-negative intensities, and subtract the R from the sparse matrix to find the low rank sparse DMD matrix.

3 Algorithm Implementation and Development

To sample our data, we first need to load our video using `VideoReader` and that we extract the data at N prescribed locations M times and evenly space by time interval δt. Then we go into a loop to go through each frame and turn them into a gray scale in order to better analysis the video. In the loop, we also reshape the frame into a column vector, then we store it into a big matrix m_frames with dimension of pixel and number of frames. Lastly, we change the information into double.

Now that we get the matrix m_frames, we want to split it into two submatrices with X_1 contains the information of frame 1 to the last second frame, while X_2 represent the second frame to the last, so that we are able to find the transformation matrix A. In order to find the low rank value r, we compute SVD of X_1 to plot the singular value and energy capture graph to determine the numbers of modes to capture the enough energy of the system.

3.1 Performace of DMD

From the previous algorithm, we are able to determine the modes value r, then to perform the DMD method, we use the algorithm 1:

Algorithm 1: Perform DMD

Extract the column with the significant singular value (modes value r) found above from SVD
Create the $\hat{S}$ matrix and denote it as `S` using the theoretical computation and find the eigenvalue as well as the eigenvector using `[eV,D] = eig(S)`
Compute Φ,μ, and ω using the eigenvecotr and eigenvalue
Use the initial x_1 and the pseudoinverse of Ψ to find the coefficients b_k

3.1.1 Reconstruction and Sparse

After our operation of DMD, we have the information we need in order to perform DMD on our video analysis. We are going to find the low-rank DMD for our background and the sparse and R in order to compute the foreground. We can do this by using algorithm 2:

Algorithm 2: Time Reconstruction and Sparse

Create an empty matrix and enter a loop
for iter = 1:length(t) **do**
  Create a time dynamic matrix using (bk.*exp(omega*t(iter)))
end for
Compute low rank DMD matrix using the dynamic matrix and the Φ
Compute Sparse matrix and R matrix for the residual negative pixels using `sparse < 0` to filter
Then we are able to find the sparse DMD matrix by operating `sparse - R`

3.1.2 Without R subtraction

We also want to compare the case when there is no R subtraction, therefore, we are adding a single value to all the entries to bring up the intensity of every pixel for the sparse matrix. This value can be set according to the value in the matrix.

3.2 Show the Video After Separation

Lastly, we are going to reshape the column containing time information to its original video frame dimension and to compare the results.

4 Computational Results

4.1 Monte Carlo

Based on the singular value spectrum in Figure 1 that we got, we can see that the first singular value capture an amount 96.94% of energy. This value is greater than 90%, hence we can only use one modes in computing our low-rank DMD matrix.

We then generate the four side to side comparison videos (Figure 2) of the video separation results which are: the original video, the separated background, the separated foreground with R subtraction, and the separated foreground without R subtraction. We then capture one frame for comparison. From which we can see that the DMD method did a great job in separating the background in which is the racing track, although it seems blurry, that was because of the shaking camera. When it comes to the foreground, we can see that with R subtraction, we can see the moving racing cars but it is not so clear since the intensity are low

when R is being subtracted. Comparably, the graph without R subtraction and only by adding a constant value to the sparse matrix appear to be clear to identify the motion of the car. This can be because the intensity is bringing up more appropriately physically and make it easier to identify the moving foreground. Although subtracting R is a good way to find the sparse DMD matrix, by adding a constant to force the pixel having non-negative value can be another effective way to do.

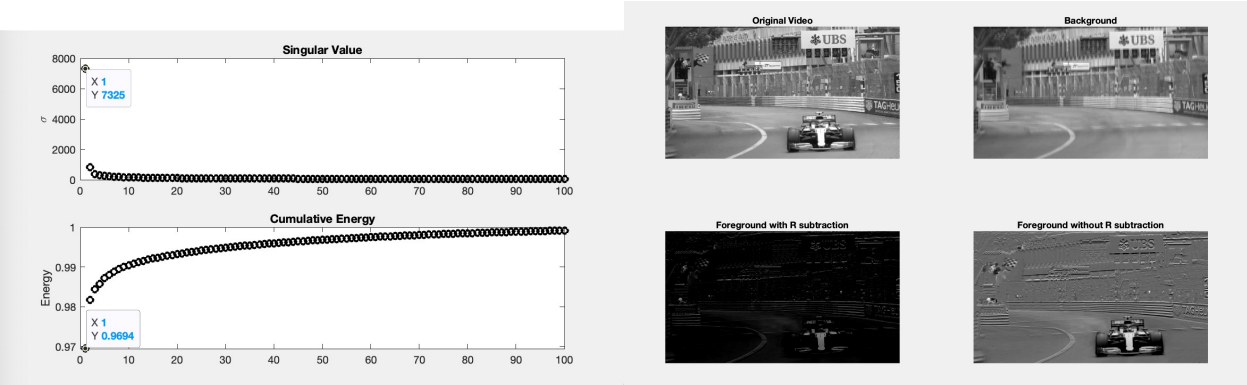

Figure 1: The singular value and energy spectrum

Figure 2: The selected frame of video separation comparison

4.2 Skiing Drop

In the similar algorithms, we are able to find the singular value spectrum and the cumulative energy spectrum for video 2. In Figure 3, we can see that the first singular value captures an amount of 99.68%, therefore, we only use one mode for the low rank DMD matrix computation.

In Figure 4, we generate the similar four comparison video and capture one frame from it. Again, in the background graph, we can see that the DMD separation did a great job in taking out the skiing man out of the frame. However, when it comes to the foreground with R subtraction, since the background is being subtract its intensity is really low to contain any information and the skiing man appear to have very low intensity as well. Hence the moving object in the foreground is being mix with the black surrounding. By directly adding some value to the sparse matrix, we can see that the moving object is distinct from the background since the comparison is more obvious so it can be identify well in this situation.

5 Summary and Conclusions

This research paper studies on the topic of utilizing DMD in separating videos into its background and the foreground. From our observation, we can see that the DMD separation is doing a great job in separating the background but when it comes to the foreground, since we are using R subtraction, it seems like the result is not very satisfying. In contracts, when we adding a constant value to the sparse matrix, we can see that it does a great job in separating the foreground. In gives us the idea that DMD along is a very powerful tool and it can effectively separate the matrix that contains the movement information of the back and fore ground in the change of time. However, the R subtraction method here is not so effective as the intensity in the video is relatively low and that with the subtraction of non-negative R, it brings the comparison of the foreground and the black background really indistinguishable. The significance of this research is the implementation indicates that one can uses DMD to process data with high dimension and turns it into a low dimensional data and process separation. Since data often contains great value of dimension, it is important to understand the background theology and algorithms to prepare for the further implementation on the Dynamic Modes Decomposition.

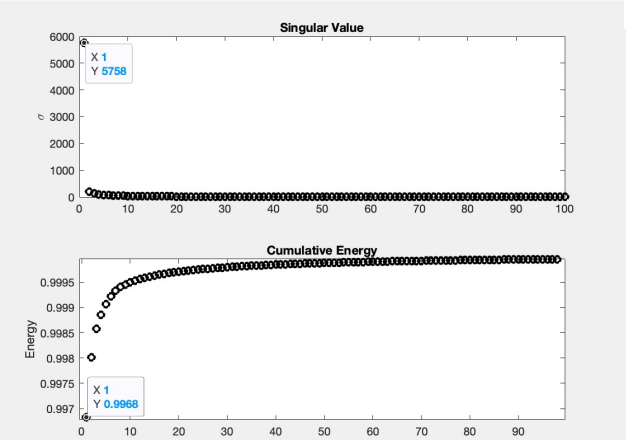

Figure 3: The singular value and energy spectrum

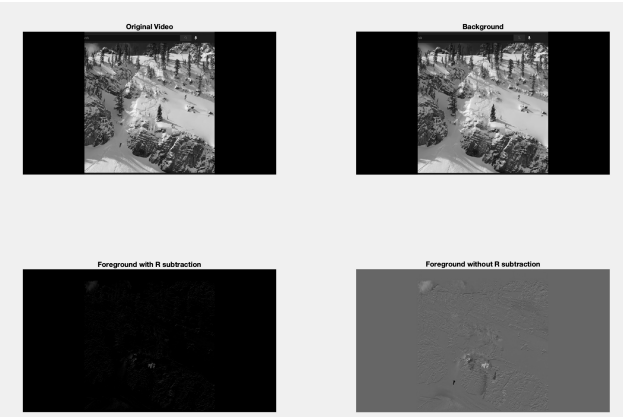

Figure 4: The selected frame of video separation comparison

Appendix A MATLAB Functions

Important MATLAB functions with a brief implementation explanation.

- `VideoReader` create object to read video files
- `.NumberOfFrames/.Duration/.Height/.Width` extract the number frame, duration, height, and width information from the object
- `rgb2gray` convert RGB image to gray scale
- `reshape(A,sz)` reshape array A using size vector sz.
- `im2double` converts the image to a double precision.
- `[U,S,V] = svd(A,'econ')` produces an economy sized singular value decomposition of a m by n matrix A.
- `diag(A)` returns a column vector of the diagonal elements of matrix A.
- `[eV,D] = eig(A)` find the eigenvalue and the eigenvector

Appendix B MATLAB Code

The following code is to perform the algorithm 1 and 2 on both videos. The following code is for the Monte Carlo video and the code for Ski Drop video is almost the same for this code.

```
clear all; clc
%% Load
v = VideoReader('monte_carlo_low.mp4');

numFrame = v.NumberOfFrames;
dura = v.Duration;
height = v.Height;
width = v.Width;

m_frames = [];
for j = 1:numFrame
    frames = rgb2gray(read(v,j));
    frames = reshape(frames,height*width,1);
    m_frames = [m_frames frames];
    %imshow(frames); drawnow
end;
m_frames = im2double(m_frames);

%% Energy
X1 = m_frames(:,1:end-1);
X2 = m_frames(:,2:end);
[U,Sigma,V] = svd(X1,'econ');

figure(1)
sig = diag(Sigma);
energy = zeros(1,length(sig));
for j = 1:length(sig)
    energy(j) = sig(j)^2/sum(sig.^2);
end
subplot(2,1,1)
plot(sig, 'ko', 'Linewidth',2)
xlim([0,100])
ylabel('\sigma')
title('Singular Value')
subplot(2,1,2)
plot(cumsum(energy),'ko', 'Linewidth',2)
xlim([0,100])
ylabel('Energy')
title('Cumulative Energy')

%% DMD
r = 1;
U_r = U(:,1:r);
Sigma_r = Sigma(1:r,1:r);
V_r = V(:,1:r);
S = U_r' * X2 * V_r /Sigma_r;
[eV,D] = eig(S);
mu = diag(D);

dt = dura/(numFrame - 1);
omega = log(mu)/dt;

Phi = X2*V_r/Sigma_r*eV;
bk = Phi\X1(:,1);
```

Listing 1: Code to perform algorithms 1 and 2

```
%% Reconstruction
t = 0:dt:dura;
modes = zeros(r,length(t));
for iter = 1:length(t)
    modes(:,iter) = bk.*exp(omega*t(iter));
end
dmd = Phi*modes;

%% Sparse
sparse = m_frames - abs(dmd);
R = sparse .* (sparse < 0);
fore = sparse - R;
fore_gray = sparse + 0.4;

figure(2)
for j = 1:numFrame
    subplot(2,2,1)
    ori = reshape(m_frames(:,j),height,width);
    imshow(ori);
    title('Original Video');

    subplot(2,2,2)
    bg = reshape(dmd(:,j),height,width);
    imshow(bg);
    title('Background');

    subplot(2,2,3)
    fg = reshape(fore(:,j),height,width);
    imshow(fg);
    title('Foreground with R subtraction');

    subplot(2,2,4)
    fg = reshape(fore_gray(:,j),height,width);
    imshow(fg);
    title('Foreground without R subtraction');

    drawnow
end
```

Listing 2: Code to perform algorithms 1 and 2