Background Subtraction in Video Streams
AMATH 482 HW5

Xuanhui Chen

Abstract

This report aims to separate two given video clips into foreground object and background object using dynamic mode decomposition (DMD). By using the DMD approach, we can reduce the high-dimensional matrices into low-dimensional ones, which can help us to capture different fundamental components of a dynamic system. In this way, movement of objects can be observed and studied.

I. Introduction and Overview

We are given two video clips containing foreground objects and background objects. In the first video, a professional skier is skiing down a mountain of snow. The second video is about a car race in Monte Carlo. Obviously, the skier and the cars are the moving objects, and the background is stationary. DMD is useful in real world for forecasting data and do making predictions because we only need to collect and analyze raw data without building up a model and relying on governing equations. Time dynamics involved in DMD is just oscillations, decay, and growth. Differential equation is a fundamental tool when using the DMD approach.

II. Theoretical Background

The Koopman Operator

The Koopman Operator $\mathbf{A}$ is a linear, time-independent operator that maps the data from t_j to t_{j+1} such that

$$\mathbf{x_{j+1}} = \mathbf{A}\mathbf{x_j}, \quad (1)$$

where j indicates the specific data collection and $\mathbf{x_j}$ is an N-dimensional vector of the data points collect at time j. It is global and thus not restricted to an area of space or time.

Dynamic Mode Decomposition (DMD)

With the Koopman Operator, we can represent the data collected in the following form:

$$\mathbf{X}_1^{M\text{-}1} = [\boldsymbol{x_1}\ \boldsymbol{Ax_1}\ \boldsymbol{A^2x_1}\ \ldots\ \boldsymbol{A^{M-2}x_1}], \quad (2)$$

where we use shorthand $\mathbf{x}_j$ to denote a snapshot of the data at time t_j. Based on the theory of Krylov subpace, we can rewrite (2) in matrix form and get:

$$\mathbf{X}_2^M = \boldsymbol{A}\mathbf{X}_1^{M\text{-}1} + \boldsymbol{r}e_{M-1}^T, \quad (3)$$

where e_{M-1} is the vector with all zeros except a 1 at the $(M-1)$st component and $\boldsymbol{r}$

is the residual vector. Here $\boldsymbol{A}$ is unknown, which we want to find out.

We are going to find $\mathbf{A}$ by finding another matrix with the same eigenvalues:

- By using the SVD to write $\mathbf{X}_1^{M\text{-}1} = \mathbf{U\Sigma V}^*$, we have:

$$\mathbf{X}_2^{\mathrm{M}} = \mathbf{AU\Sigma V}^* + \mathbf{r}\mathrm{e}_{\mathrm{M\text{-}1}}^{\mathrm{T}}. \quad (4)$$

  And after some calculations, we get:

$$\mathbf{U}^*\mathbf{A}^*\mathbf{U} = \mathbf{U}^*\mathbf{X}_2^{\mathrm{M}}\mathbf{V\Sigma}^{\mathbf{-1}}, \quad (5)$$

  where we can have the right hand side being equivalent to $\tilde{\mathbf{S}}$ for convenience, and $\mathbf{U} \in \mathbb{C}^{N\times K}, \mathbf{\Sigma} \in \mathbb{Z}^{K\times K}, \mathbf{V} \in \mathbb{C}^{M-1\times K}$.

If $\mathbf{y}$ is an eigenvector of $\tilde{\mathbf{S}}$, then $\mathbf{U}_y$ is an eigenvector of $\mathbf{A}$, and we can write the eigenvector/eigenvalue pairs of $\tilde{\mathbf{S}}$ as:

$$\tilde{\mathbf{S}}\mathbf{y}_{\mathrm{k}} = \mu_k \mathbf{y}_{\mathrm{k}}, \quad (6)$$

which gives the *DMD modes* as the eigenvectors of **A**:

$$\psi_{\mathrm{k}} = \mathbf{U}\mathbf{y}_{\mathrm{k}}. \quad (7)$$

Now we can expand our eigenbasis to get

$$X_{DMD}(t) = \sum_{k=1}^{K} b_k \psi_k e^{\omega_k t} = \Psi \mathrm{diag}(e^{\omega_k t})\mathbf{b}. \quad (8)$$

When taking t = 0, the above becomes:

$$\mathbf{x_1} = \Psi\mathbf{b} \implies \mathbf{b} = \Psi^{\dagger}\mathbf{x_1}. \quad (9)$$

where $\Psi^{\dagger}$ is the pseudoinverse of the matrix Ψ.

DMD Spectrum of Frequencies to Subtract Background Modes

Assume that ω_p, where $p \in \{1,2,\dots,l\}$, satisfies $\| \omega_j \| \; \forall \, j \neq p$ is bounded away from 0, we have:

$$\mathbf{X}_{\mathrm{DMD}} = b_p \psi_p e^{\omega_p \mathbf{t}} + \sum_{j\neq p} b_j \psi_j e^{\omega_j \mathbf{t}}, \quad (10)$$

where $b_p \psi_p e^{\omega_p \mathbf{t}}$ represents the background video, and $\sum_{j\neq p} b_j \psi_j e^{\omega_j \mathbf{t}}$ represents the foreground video.

Then we can compute the DMD's approximate low-rank reconstruction according to:

$$\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}} = b_p \psi_p e^{\omega_p \mathbf{t}}. \quad (11)$$

We also have the fact that

$$\mathbf{X} = \mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}} + \mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}, \quad (12)$$

then the DMD's approximate sparse reconstruction,

$$\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}} = \sum_{j\neq p} b_j \psi_j e^{\omega_j \mathbf{t}}, \quad (13)$$

can only be calculated with real valued elements as:

$$\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}} = \mathbf{X} - |\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}|. \quad (14)$$

The residual negative values can be put into a $n \times m$ matrix $\mathbf{R}$ and then be added back into $\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ to avoid negative pixel intensities:

$$\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}} \leftarrow \mathbf{R} + \left|\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}\right| \quad (15)$$

$$\mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}} \leftarrow \mathbf{X}_{\mathrm{DMD}}^{\mathrm{Sparse}} - \mathrm{R} \quad (16)$$

In this way, none of the pixel intensities will be negative, which ensures that the approximate low-rank and sparse DMD reconstructions are real-valued.

III. Algorithm Implementation and Development

We will repeat the following processes for all cases with details changing:

- Load the data using VideoReader:
    - Find the number of frames with the data, which is the total number of snapshots of the video. (See **Appendix B** Part I.)
    - Change each image into grayscale form, reshape the information of each image into a column vector, and gather these column vectors into a matrix by using a for loop. To reduce runtime, we scaled the image down by a factor of 0.2. Here each column vector in the matrix represents an image. (See **Appendix B** Part I.)
    - Find the time information. (See **Appendix B** Part I.)
- Implement the DMD algorithm (See **Appendix B** Part II):
    - Split the matrix we got into two data matrices, with each representing the snapshots of the first frame to the second last frame, and the snapshots of the second frame to the last, respectively.
    - Apply the SVD on X1. In this way we can determine the rank r representing the number of significant modes and draw a graph to see how much energy could be captured in this system.
    - Truncate the matrices we get from the SVD with rank r.
    - Find the approximate low-rank subspace and do an eigen decomposition and look for the spectra modes.
    - Do the calculations to find each component as stated in the theoretical part with the information we get from the last step.
- Draw the graphs for a randomly picked frame (the 200$^{\mathrm{th}}$ frame for the ski_drop video and 60$^{\mathrm{th}}$ for the monte_carlo video) of the original video, its foreground and background. (See **Appendix B** Part III.)

IV. Computational Results

4.1 Ski_drop Video

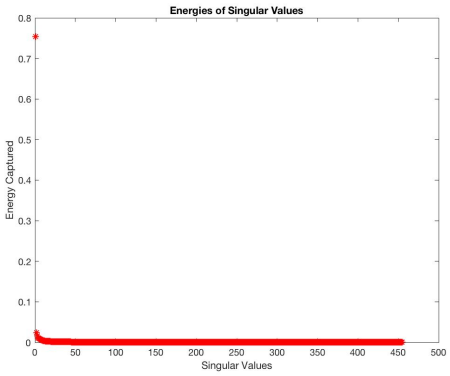
The energy diagram in Figure 1 shows only one significant mode so that the rank is set to be 1, and the value of omega is 6.3748e-04.

Figure 3 represents the separation of the foreground and background of the 200th video frame. From the video we can see the only moving objects are the skier and the cloud, and the stationary background of this video is the snow mountain. Comparing the foreground and background to the original video frame, we can see the moving objects (e.g., the skier and the cloud) are removed for the background while being highlighted in the foreground (e.g., at the center and the upper left corner.) The DMD approach did a good job for this video.

4.2 Monte_carlo Video

The energy diagram in Figure 2 also shows only one significant mode so that the rank is set to be 1, and the value of omega is -6.7886e-04.

Figure 4 represents the separation of the foreground and background of the 60th video frame. From the video we can see the only moving objects are the cars and the person who is waving a flag, and the stationary background of this video is the race track. Comparing the foreground and background to the original video frame, we can see the moving objects (e.g., the cars and the person waving flag) eliminated for the background while being highlighted in the foreground (e.g., at the bottom and the upper left corner.) The separation is perfect for this video.



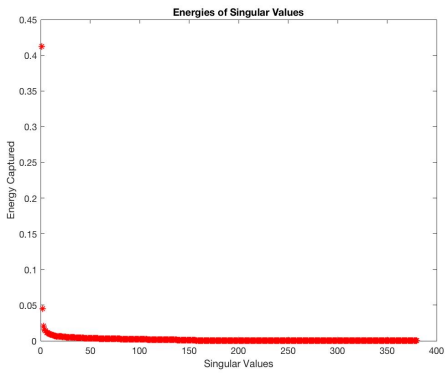


Figure 1 (left): The energy captured by singular values for Video 1.

Figure 2 (right): The energy captured by singular values for Video 2.

Figure 3: Separation of a frame in the original ski_drop video: the background and the foreground.

Figure 4: Separation of a frame in the original monte_carlo video: the background and the foreground.

V. Summary and Conclusion

In conclusion, we have pretty accurate separations overall, which shows that the dynamic mode decomposition (DMD) is helpful for separating the foreground and background of a video involving moving objects. The DMD is completely data-driven without any governing equations. Complicated dataset can be reduced to several components representing different scenes. The process is relatively simple compare to building up mathematical models when the only thing provided is raw data.

Appendix A: MATLAB functions used and brief implementation explanation

- **OBJ = VideoReader(FILENAME)**: constructs a multimedia reader object, OBJ, that can read in video data from a multimedia file.**rgb2gray**(RGB): Convert RGB image or colormap to grayscale.
- **I = rgb2gray(RGB):** converts the truecolor image RGB to the grayscale intensity image I.
- **imresize(A, SCALE):** returns an image that is SCALE times the size of A, which is a grayscale, RGB, or binary image.
- **reshape(X,M,N) or reshape(X,[M,N])**: returns the M-by-N matrix whose elements are taken columnwise from X.
- **[U,S,V] = svd(X)**: produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in order, and unitary matrices U and V so that X = U*S*V'.
- **diag(X):** the main diagonal of X.
- **sum(X)**: the sum of the elements of the vector X.
- **eig(A):** produces a column vector E containing the eigenvalues of a square matrix A.
- **uint8(X):** converts the elements of the array X into unsigned 8-bit integers.
- **imshow(I)**: displays the grayscale image I.

Appendix A: MATLAB Codes


```
clear all; clc;

% Part I: Loading the data
% We can do the same procedures by simply change the movie file loaded in:
v = VideoReader('ski_drop.mov');
%v = VideoReader('monte_carlo.mov');
nf = v.NumberOfFrames;
X = [];

% Part I: Reformat the images
for i=1:nf
    img = double(rgb2gray(imresize(read(v,i), 0.2)));
    [m,n] = size(img);
    img = reshape(img, [m*n,1]);
    X = [X img];
end

t = linspace(0,v.CurrentTime, nf+1);
t = t(1:end-1);
dt = t(2)-t(1);

%% Part II: DMD Algorithm
X1 = X(:,1:end-1);
X2 = X(:,2:end);

[U,S,V] = svd(X1, 'econ'); %SVD

% Plot the engergy captured
figure(1)
plot(diag(S)/sum(diag(S)), 'r*');
title("Energies of Singular Values");
xlabel("Singular Values");
ylabel("Energy Captured");

% Truncate the matrices
r = 1;
U_r = U(:,1:r);
S_r = S(1:r,1:r);
V_r = V(:,1:r);

% Find approximate low-rank subspaces and psi
Atilde = (U_r'*X2*V_r)/S_r;
[W_r, D] = eig(Atilde);
phi = (X2 * V_r) / (S_r*W_r);
lambda = diag(D);
omega = log(lambda)/dt;

x1 = X1(:,1);
b = psi\x1;

t_d = zeros(r,length(t));
for j = 1:length(t)
    t_d(:,j) = (b.*exp(omega*t(j)));
end
X_bg = psi * t_d;

X_sparse = X-X_bg;
R = X_sparse .* (X_sparse<0);
```

```
X_fg = X_sparse - R;
X_low_dmd = R + abs(X_bg);


%% Part III: Draw one frame original video, its foreground and background
cap = 200; % randomly pick the frame

figure(2)
subplot(1,3,1)
img_o = reshape(X, [m, n, nf]);
imshow(uint8(img_o(:,:,cap)))
title("Original Video");

subplot(1,3,2)
img_bg = reshape(X_bg, [m, n, nf]);
imshow(uint8(img_bg(:,:,cap)))
title("Background");


subplot(1,3,3)
img_ob = reshape(X_fg(:,cap), [m, n]);
imshow(img_ob, []);
title("Foreground");
```

Reference

1. Course note.
2. MATLAB help section.
3. Assignment 5 spec.