Background Subtraction in Video Streams

Chongyi Xu
University of Washington
AMATH 482/582 Winter Quarter 2018
chongyix@uw.edu

March 11, 2018

Abstract

For this project, we will use Dynamic Mode Decomposition method to separate the foreground object from the background object. Three different videos will be imported into Matlab and a proper Dynamic Mode Decomposition reconstruction will produce Low-Rank DMD and Sparse DMD. The Low-Rank DMD will be considered to be the background video and the Sparse DMD will be foreground video. As the results, the reconstruction for the first video goes pretty well.

1 Introduction and Overview

1.1 Introduction

Dynamic mode decomposition (DMD) is a dimensional reduction algorithm developed by Peter Schmid in 2008. Given a time series of data, DMD computes a set of modes each of which is associated with a fixed oscillation frequency and decay/growth rate. For linear systems in particular, these modes and frequencies are analogous to the normal modes of the system, but more generally, they are approximations of the modes and eigenvalues of the composition operator (also called the Koopman operator). Due to the intrinsic temporal behaviors associated with each mode, DMD differs from dimensional reduction methods such as principal component analysis, which computes orthogonal modes that lack predetermined temporal behaviors. Because its modes are not orthogonal, DMD-based representations can be less parsimonious than those generated by PCA. However, they can also be more physically meaningful because each mode is associated with a damped (or driven) sinusoidal behavior in time.

1.2 Overview

There will be three videos to reconstruct in this project. The three videos have been uploaded into Google Drive.

- An umbrella moving vertically and horizontally in the front of a wall
- An umbrella moving vertically and horizontally in the front of lighting screens
- Cars going across a bridge under sunlight.

For each of the video, I will first reconstruct the video using Dynamic Mode Decomposition and find out the low-rank part as well as sparse part. Theoretically, the low-rank part will be background.

2 Theoretical Background

2.1 Dynamic Mode Decomposition

Dynamic Mode Decomposition is used to generate a dynamic model from the measurements data only. Considering we have a data matrix $\mathbb{X}_{data} \in \mathbb{R}^{n*m}$. Our goal is to build a linear fit model for the dynamic system $\frac{d\mathbf{x}}{dt} = A\mathbf{x}$. So we first need to divide the data matrix $\mathbb{X}$ into two matrix.

$$\begin{aligned}\mathbb{X}_{data} &= [x_1\ x_2\ x_3\ \dots x_m]\\ \mathbb{X} &= [x_1\ x_2\ x_3\ \dots x_{m-1}]\\ \mathbb{X}' &= [x_2\ x_3\ x_4\ \dots x_m]\end{aligned}$$

Then we want $\mathbb{X}' = A\mathbb{X}$. So $A = \mathbb{X}'\mathbb{X}^+$, where $\mathbb{X}^+$ is the pseudo inverse of matrix $\mathbb{X}$. Then, take the SVD of $\mathbb{X}_{data}$ and get U, Σ, V^*. However, since the data matrix is always too large such that the computation process becomes expensive, we want to truncate the rank to r for cost-less computation runtime. Therefore, we take the first r ranks to get U_r, Σ_r, V_r^*. Then we have

$$\begin{aligned}\widetilde{A} &= U^*AU\\ &= U^*\mathbb{X}'\mathbb{X}^+U\\ &= U^*\mathbb{X}'V\Sigma^{-1}U^*U\\ &= U^*\mathbb{X}'V\Sigma^{-1}\end{aligned}$$

In order to get the linear fitting model $\frac{d\mathbf{x}}{dt} = A\mathbf{x}$, we hav our guess solution to be $\mathbf{x} = \mathbf{v}e^{\lambda t}$. Back to matrix size we have our solution as

$$\widetilde{A}\Omega = \Lambda\Omega$$

where $\Omega = [eigenvectors]$ and $\Lambda = \begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_r\end{bmatrix}$ After having the low-rank eigenvalues and eigenvectors, we should get back to high ranks to get the solution. The DMD modes are defined to be

$$\Phi = \mathbb{X}'V\Sigma^{-1}\Omega$$

And the solution we are looking for is

$$\begin{aligned}\mathbf{X} &= \Phi e^{\Lambda t}b\\ &= \sum_{k=1}^{r}\phi_k e^{\omega_k t}b_k\end{aligned}$$

And after we get our $\mathbf{X}_{DMD}$ as DMD reconstruction, we are looking for $p \in 1, 2, \dots r$ satisfies $||\omega_p|| \approx 0$ and that any other $k \neq p$ has $||\omega_k||$ bounded away from zeros. Thus we have

$$\mathbf{X}_{DMD} = \sum_{k \neq p}\phi_k e^{\omega_k t}b_k + \phi_p e^{\omega_p t}b_p$$

where the first part is the foreground video and the second part is the background video.

3 Algorithm Implementation and Development

3.1 General Procedure

- Load the video files and convert the video from RGB to gray.
- Victories the videos and make the data matrix to have each row containing the value of the pixel and each column containing the time (frame), $\mathbf{X}$.
- Get $\mathbb{X}$ and $\mathbb{X}'$ from the data matrix. And SVD $\mathbb{X}$.
- Perform Dynamic Mode Decomposition on the matrix.
- Find the index of the ω_p that satisfies the conditions introduced in the theoretical background.
- Construct $\mathbf{X}_{DMD}^{Low-Rank}$ and $\mathbf{X}_{DMD}^{Sparse}$.
- Subs tract $\mathbf{X}_{DMD}^{Low-Rank}$ from the data matrix $\mathbf{X}$
- Store the indices with negative values in the rest matrix $\mathbf{X}_{DMD}^{Sparse}$ and add those back to $\mathbf{X}_{DMD}^{Low-Rank}$.
- Get the $\mathbf{X}_{DMD}^{Sparse}$ again by subtracting $\mathbf{X}_{DMD}^{Low-Rank}$ from the data matrix $\mathbf{X}$.

4 Computational Results

4.1 Video 1

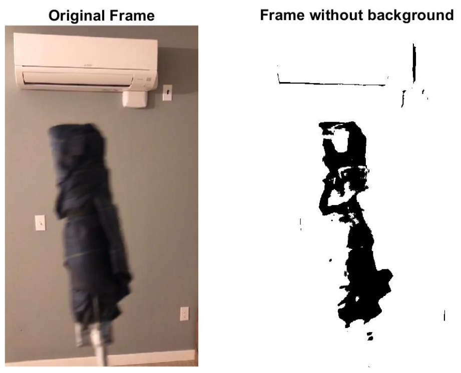

Figure 1: Comparison For Video 1

For the first video, we can see that at this frame, the background has been successfully removed.

4.2 Video 2

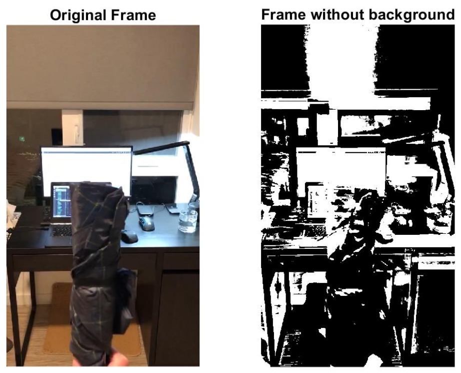

Figure 2: Comparison For Video 2

For the second video, the removal is not as ideal as the first video but still successfully removed most of the background view in the frame.

4.3 Video 3

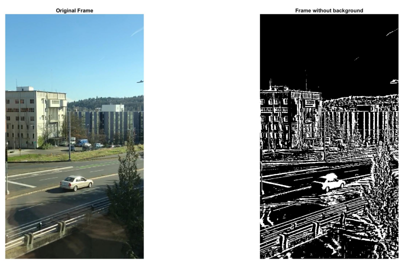

Figure 3: Comparison For Video 2

And for the last video, the removal does not work pretty well, which is reasonable since the video frame is too complicated with too many elements.

5 Summary and Conclusion

Dynamic Mode Decomposition is super powerful. It is possible to build a model to the dynamic system and approximate the future behavior. Using DMD to process the similar application as background substraction to video stream as we did in this project, it might be important to have a "simple" video as possible.

```
clc; clear all; close all;
%%
dir = 'videos/';
v = VideoReader([dir '1.mp4']); % change to see video 1, 2, 3

dt = 1/30; % 30 fps from settings

% store the video frame by frame
count = 0;
while hasFrame(v)
    count = count + 1;
    img = readFrame(v);
    img = double(rgb2gray(img));
    images{count} = img;
end

[~, T] = size(images);
t = 0:dt:T;
%% Vectorize
[col, row] = size(images{1});
for i = 1:size(images,2)
    data(:, i) = reshape(images{i}, col * row, 1);
end

%% DMD Algorithm
X1 = data(:, 1:end-1);
X2 = data(:, 2:end);

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

%% Rank Truncuation if needed
% r = 100;
% U = U(:, 1:r);
% S = S(1:r, 1:r);
% V = V(:, 1:r);
%% Get the model
Atide = U'*X2*V/S;
[W, D] = eig(Atide);
Phi = X2 * V / S * W; % DMD modes
omega = log(diag(D)) / dt;
b = Phi \ X1(:,1);

%% Compute for XLowRank and XSparse
[~,p] = min(abs(omega));

% Construct the low-rank matrix for every time step by using only the mode
% associated with the minimum frequency.
for k = 1:size(data,2)
    XLowRank(:,k) = Phi(:,p)*diag(exp(omega(p).*t(k)))*b(p);
end

XSparse = data - abs(XLowRank);

%% Find negative values
R = XSparse;
R(R>0) = 0;

XLowRank = R + abs(XLowRank);
XSparse = XSparse - R;

%% Reshape Back to 3D
for i = 1:size(XSparse,2)
    newVideo{i} = reshape(XSparse(:, i), col, row);
end
```

```matlab


%% See the new video
% figure(1)
% for i = 1:size(newVideo,2)
%     imshow(newVideo{i});
%     pause(0.03)
% end
%% Comparison
figure(2)
frame = 80;
count = 0;
v = VideoReader([dir '1.mp4']);
while hasFrame(v)
    if count < frame
        count = count + 1;
        imag = readFrame(v);
    elseif count == frame
        imag = readFrame(v);
        break;
    end
end
subplot(1, 2, 1), imshow(imag), title('Original Frame');
subplot(1, 2, 2), imshow(real(newVideo{frame})), title('Frame without background');
```