Dynamic Mode Decomposition: Foreground & Background Video Separation

Avery Pong – AMATH 482 – 03.13.19

0. Overview

Three traffic videos, from the UW campus and from i5 traffic, are analyzed for their capacity to render foreground-background separation via the Dynamic Mode Decomposition. Each video demonstrates evidence of a low-dimensional attractor that persists throughout the duration of the video—this is the background of the video that stably remains throughout the shot. Each, to varying degrees, had dynamic objects that could be identified in a sparse foreground matrix, upon background subtraction from the video as a whole. The analysis explores the DMD and touches on the features that make some videos more conducive to foreground-background subtraction than others.

I. Introduction: Video Separation

Videos from stationary cameras capture a dynamic foreground—from moving cars, people, to birds—against a largely static background. The applications are endless. Video production teams could benefit from identifying moving objects in their sets that they might want stationary, but are less detectible to the naked eye in the original video. Cell counting schemes when the cells are motile may benefit from subtracting dead background cells, while separating moving ones. Video surveillance of people and cars, like the videos described here, could help to clarify movement within the video frames.

Video separation relies on the fact that the background image is responsible for most of the consistent energy in each frame of the video, and thus correlates well through time. Because of the background's persistence, it should indeed be responsible for most of the features of the video's collective frames. In other words, they define the dominant features, or modes, of the space describing the video.

To facilitate analysis, these collection of frames were organized into a large matrix so that the dominant features across the frames could be assessed by the singular value decomposition (SVD). To this end, it was determined that the video background accounted for about 60% of the video data. This begs the question of how this dominant background feature space might be subtracted from the original, naked video data in order to render the dynamic objects in the videos, which could be rendered in higher-dimensional modes.

II. Theoretical Background: DMD Theory & Algorithm

Dynamic mode decomposition was originally used as an equation-free method to tame non-linear dynamics observed in fluids. At its essence it combines the power of dimensionality reduction within spatially-defined time shots, applying Fourier transforms across the time axis to find the persistence of dominant frequency modes between the time shots. At a high level, the DMD produces an approximate linear dynamic framework for the system.

$$\frac{dy}{dt} = f(y, t, u) \rightarrow \frac{dx}{dt} = Ax$$

Given a known initial condition $x(0)$, a solution of the following form is expected:

$$x(t) = \sum_{k=1}^{n} \phi_k \exp(\omega_k t)\, b_k = \boldsymbol{\Phi} \exp(\boldsymbol{\Omega} t)\, \boldsymbol{b}$$

Here, ϕ_k represents eigenvectors and ω_k represents the eigenvalues of A, while b_k coefficients represents initial conditions $x(0)$ in the eigenvector basis. Continuous dynamics described here can be further extended to a system with discrete time intervals:

$$x_k = \sum_{j=1}^{rank} \phi_j \lambda_j^k b_j = \boldsymbol{\Phi} \boldsymbol{\Lambda}^k \boldsymbol{b}$$

Here, ϕ_j represents the eigenvectors of A, while λ represents its eigenvalues. In this way, the DMD yields a low-rank approximation via an eigenvalue decomposition of A that appropriately fits the trajectory of the dynamics of the data through all of its k frames.

Functionally, A represents the matrix through which the system must, in a sense, move to transform its dynamic behaviors between different points of time. To capture this with a known set of data, data in matrix $\boldsymbol{X^o}$ is sequestered into two smaller matrices: the first matrix $\boldsymbol{X}$ captures the all but the last column of $\boldsymbol{X^o}$, while the second matrix $\boldsymbol{X'}$ captures all but the first column of $\boldsymbol{X^o}$. In this way, we can determine the appropriate transform $\boldsymbol{A}$ that brings the system one time frame forward, from $\boldsymbol{X}$ to $\boldsymbol{X'}$. So, because:

$$\boldsymbol{X'} \approx \boldsymbol{AX}$$

We can then find an appropriate $\boldsymbol{A}$ to describe shifts in the data dynamics through time:

$$\boldsymbol{A} = \boldsymbol{X'X}^{\dagger}$$

Where $\boldsymbol{X}^{\dagger}$ is the pseudo-inverse of the original $\boldsymbol{X}$ matrix. However, these calculations are complicated by the fact that explicitly solving $\boldsymbol{A}$ yields an outer product matrix whose size far exceeds either $\boldsymbol{X}$ matrix, rendering this algorithmic methodology impractical. For this reason, the DMD is typically performed on some low-rank subspace of the data, in order to reduce the dimensional demands that the computation of $\boldsymbol{A}$ requires. The next section develops this DMD algorithm, which was used extensively in this analysis.

III. Algorithmic Implementation: DMD

Videos of traffic, in different settings and lighting, were taken and read into MatLab. Each was rendered in grayscale to facilitate analysis, and was further resized in order to speed up computations. Furthermore, frames from each video was reshaped into a vector of data, each of which was then put into a larger data matrix representing the video in matrix form.

A singular value decomposition of this data was taken in order to render the most important modes to the data. Upon first visual inspection, the first three modes of could account for approximately 70% of the relative energy in the image space. This step was taken in order to reduce the size of the matrix on which the DMD modes would later be produced.

To this end, 1, 10, and occasionally 50 rank approximation of the SVD on the first matrix were taken to generate an approximate $\boldsymbol{A}$ matrix, $\boldsymbol{A}'$. This, to reiterate, greatly reduces the computation time associated with calculating decompositions involving $\boldsymbol{A}$.

$$\boldsymbol{A}' = \boldsymbol{U}^* \boldsymbol{X}' \boldsymbol{V} \boldsymbol{\Sigma}^{-1}$$

Using only the first r modes ensured that a low-dimensional $U, V,$ and Σ^{-1} could construct a low-dimensional linear model $\boldsymbol{A}'$ of original data matrix $\boldsymbol{A}$.

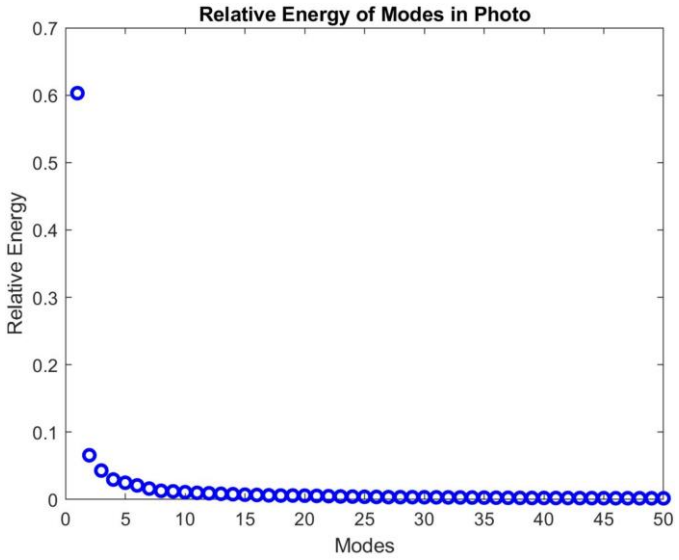


Figure 1. Relative Energy concentrated in first mode.

With $\boldsymbol{A}'$ in hand, an eigen-decomposition was conducted to yield a low rank fit to the trajectory of the data in terms of its eigenvector basis:

$$\boldsymbol{A}' \boldsymbol{W} = \boldsymbol{W} \boldsymbol{\Lambda}$$

This decomposition of the approximate $\boldsymbol{A}'$ matrix allows for the calculation of the eigenvectors of the original matrix:

$$\boldsymbol{\Phi} = \mathbf{X}' \mathbf{V} \boldsymbol{\Sigma}^{-1} \boldsymbol{W}$$

All that was then left to calculate were the discrete-time eigenvalues. For convenience, the logarithm of these values was taken and divided by the framerate (dt) to yield vector Ω. Additionally, the initial conditions from the original data $\boldsymbol{A}$ were put into the eigen-basis. In all, this produced the following calculation of the DMD modes:

$$x(t) = \sum_{k=1}^{rank} \phi_j \exp(\omega t)\, b_k = \boldsymbol{\Phi}\exp(\boldsymbol{\Omega}\mathrm{t})\boldsymbol{b}$$

Having calculated the underlying, highly correlated modes between the frames, a the video background could be rendered by multiplying by reshaping the DMD modes and playing the frames in succession. Yielding the foreground was a simply matter of subtracting this persisting background feature space from the original data, yielding a sparse matrix that contained all of the dynamic features of the original video.

IV. Computation Results

The best backgrounds were rendered when only the first mode, yielded by the singular value decomposition of the X matrix, was taken. This is reasonable because the first mode accounted for, in all cases, for at least half of the features gleaned from the videos. Additionally, taking more than the first mode tended to introduce slower objects that, indeed, persisted between frames at a greater frequency than expected.

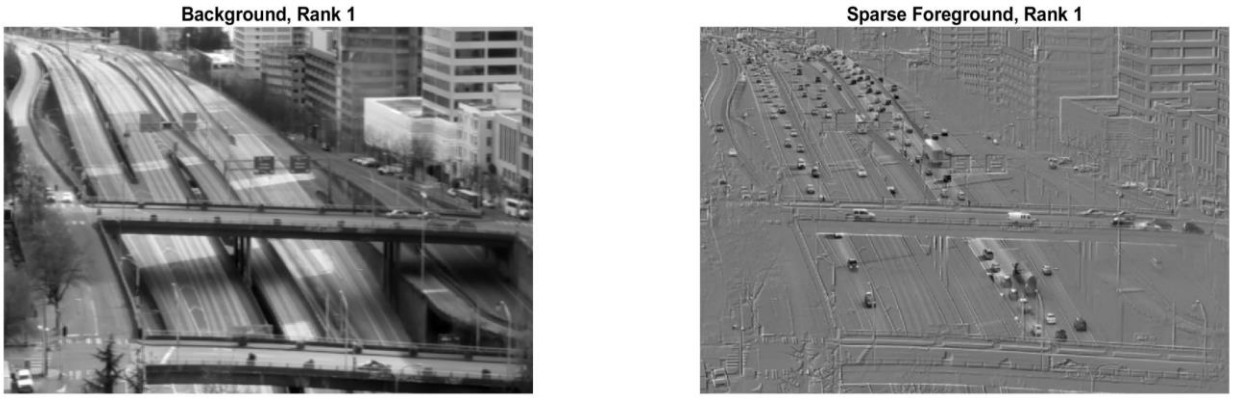


Figure 2. i5, with rank 1 approxiamtion applied

Here, the background shows nearly no moving objects, namely the cars on the freeway. As expected, subtraction of this background from the rest of the video data yielded the cars, separated from their underlying background. Interestingly, a lot of the background remains in the higher-rank space—outlines of the bridges across the freeway, as well as the adjacent buildings, are readily observable. It was hypothesized that because of slight vibrations in the camera, and because of slight variations in the environmental lighting, there may be dynamic qualities to the way in which the stationary bridges and buildings were rendered. Another reasonable hypothesis is that the camera was not perfectly still, and thus the slight variations in the position of the building outlines were captured in the higher feature space. To address these phenomena, a rank 10

approximation was taken to better capture the background information, and thus render better background subtraction for subsequent foreground matrices.

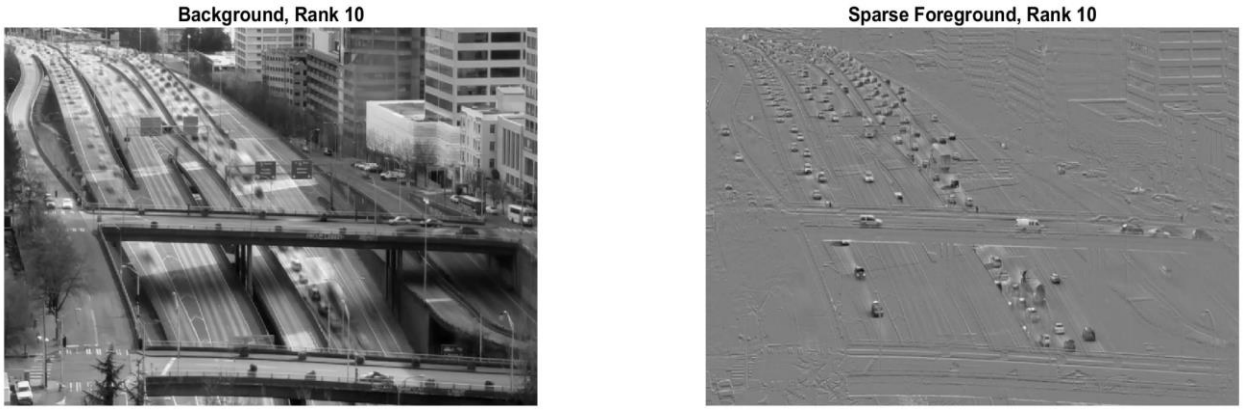


Figure 3. i5 with rank 10 approximation applied

Figure 3 demonstrates that when more SVD modes are captured for the rendering of DMD modes, more of the low-dimensional background feature space can be captured, thus yielding slightly improved background subtraction in the foreground video. However, the rank 10 background itself contains noticeably more dynamic features (cars) than its rank 1 counterpart. This, in many ways, means that the cars present in the background are moving slower. In both shots, it is clear that there are more cars on the right-hand side of the road, indicating a heavier load of traffic. For this reason, cars move slower, and correlate better through each data frame in the video. For this reason, the slow cars in the right lane show up in the background in this higher rank background case, whereas, still, faster cars in the left-hand-side of the road are still not as detectable.

One last interesting factor governing the quality of background subtraction is the distance each car is with respect to the camera. Naturally, cars that are farther move less with respect to the 2-dimensional coordinate plane of the camera. Closer cars, on the other hand, move much more per time. Thus, closer objects tend to be better removed from the background, as evidenced by the rank 10 image from Figure 2. The close cars in the left-hand-lane almost completely disappear, while the cars farther down the road persist.

Figure 4. UW Skybridge, northbound, with rank 1 approximation

Two more videos were analyzed: a north and southbound view from the UW pedestrian sky bridge. Northbound data is discussed here, while southbound video shots are available in Appendix C.

Figure 4. shows that the background, again, can be successfully separated from the foreground dynamics. Interestingly, the background image contains a car (center) and bus (bottom left). These persisted through the video because the bus had traveled and then stopped at the Schmitz Hall bus station, thus becoming part of the background of the video. The car in the center was waiting to turn left into the parking lot underneath Red Square, and was at a standstill for the duration of the video. All other vehicles were moving, and were captured by in the sparse foreground data.

Again, figure 4 demonstrates that large parts of the background imagery remain in the high rank, dynamic foreground space. This was due in part to the fact that the video was taken with a iPhone, without a tripod. Thus, even slight camera shakes meant that the outline of the background was moving slightly with respect to the coordinate system of the camera. To remedy this, a rank 10 approximation was taken like with the i5 case.

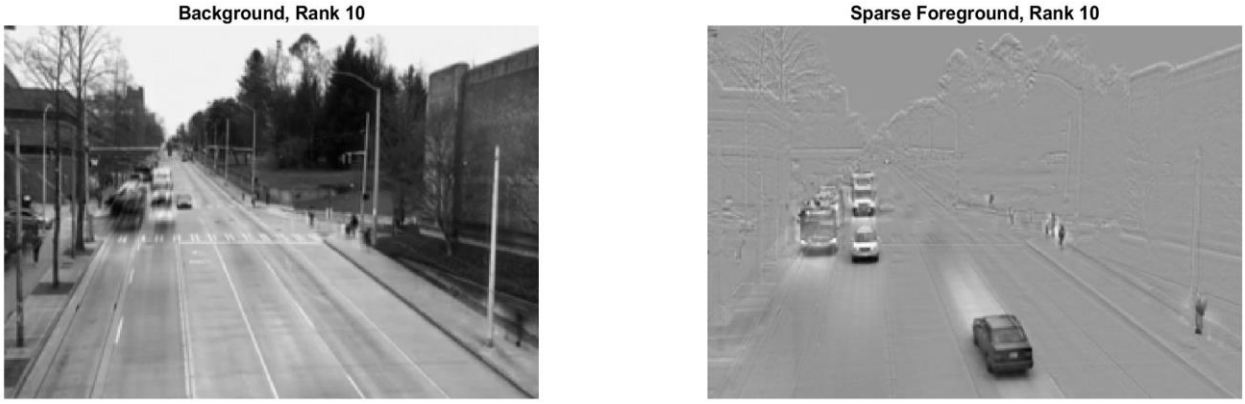


Figure 5. UW Skybridge, northbound, with rank 10 approximation

The rank 10 approximation was able to capture more background information in the DMD modes, and thus, upon subtraction, less of the background imagery is observable in the foreground image. Interestingly, more vehicles are observed in the background shot, as with the i5 case. It was hypothesized that because the vehicles were coming head-on to the camera (as in, stayed in relatively the same camera position throughout the shot), the vehicles would be perceived to correlate well between the shots. For this reason, it would have suited the analysis to use video data where the vehicles moved across, instead of within, the video. This process would have improved the separation, and more importantly shows some shortcomings of using the DMD for video separation.

V. Conclusions

The DMD allows for background and foreground separation in videos where the camera is sufficiently still. Videoframes can be seen as discrete measurements of underlying, non linear dynamics present in the video as a whole, and can help to elucidate them. The weaknesses of this method primarily lay in the fact that, for many videos, the dynamic objects may simply grow in size with respect to the camera's frame of

reference, making it hard to separate these objects from the background. Overall, video processing is an illustrative, visual demonstration of how dimensionality reduction can facilitate improved analysis of non-linear dynamics.

Appendix A: MatLab Functions

`eig(data)` – calculated eigenvalues and vectors for eigenvalue decomposition

`imresize()` – resized larger image/frame matrix to make data more manageable

`readFrame()` – allowed to read frames in movie

`reshape(data)` – reshaped rectangular matrices to vector for analysis

`svd(data, 'econ')` – calculated appropriate U, S, and V matrix for SVD

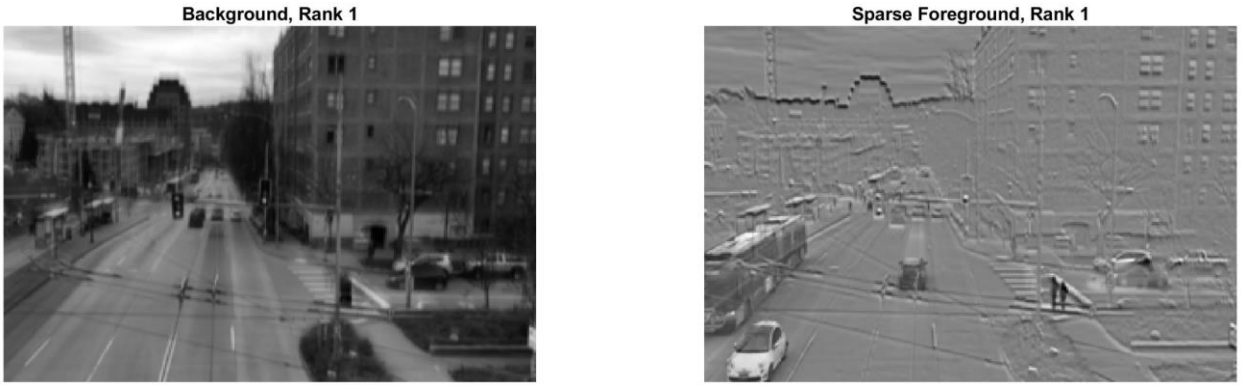
`VideoReader()` – allowed to read .mov and .mp4 files

Appendix B: MatLab Code

```
% HW 5 Video Separation

clear all; close all; clc

vid = VideoReader('Northbound.MOV');                % Load video data
numFrames = size(vid.duration*vid.FrameRate,3);     % get number of frames

L = [];                                             % initialize data matrix
while hasFrame(vid)
    for i = 1:numFrames
        video = readFrame(vid);                     % reads on per-frame basis
        video = rgb2gray(video);                    % change to grayscale
        image = video(:,:,i);                       % loop over frames
        image = imresize(double(image), [200 300]); % resize to reduce comp time
        L = [L; reshape(image, 1, 200*300)];        % put matrix into vector form, stick in data
    end
end


%% Dyn Mode Decomp

L = L';                             % transpose to get frames as columns
L1 = L(:,1:end-1);                  % take first X
L2 = L(:,2:end);                    % take timestep after X

[U1,S1,V1] = svd(L1, 'econ');       % SVD first matrix
r = 10;                             % Select these dominant modes
U = U1(:,1:r);                      % U unitary matrix
S = S1(1:r,1:r);                    % S diag matrix
V = V1(:,1:r);                      % V uni

Atilde = U'*L2*V/S;                 % finds dim reduced A matrix
[W,D] = eig(Atilde);                % eigenval decomp

Phi=L2*V/S*W;                       % DMD modes

%plot(diag(S)./sum(diag(S)))
```

```
%% Plotting DMD Modes

dt = 1/(vid.FrameRate);              % time in frame rate
t = vid.FrameRate*vid.duration;      % absolute frames

lambda = diag(D);                    % discrete-time eigenvals
omega = log(lambda)/dt;              % continuous time eigenvals


b = Phi\L1(:,1);                     % pseudo-inverse initial conditions
mm1 = size(L, 2);                    % take length
t = (0:mm1-1)*dt;                    % establish t vector
u_modes = zeros(r,mm1);              % DMD reconstruction for every time point
for iter = 1:mm1
    u_modes(:,iter) =(b.*exp(omega*(t(iter))));     % Key dmd mode calculation
end
u_dmd = Phi*u_modes;                 % DMD resconstruction with all modes



%% Play Video

% for x = 1:337
%     image_frame = reshape(sparse(:,x), [200 300]);
%     imshow(image_frame, [])
% end

%% Reconstruct Background

sparse = L - abs(u_dmd);             % Get sparse from L, minus major dmd modes

% Plot Background
subplot(1,2,1)
background = reshape(u_dmd(:,1), [200 300]);
imshow(background, [])
title('Background, Rank 10')

% Plot Foreground
subplot(1,2,2)
select_image = reshape(sparse(:,40), [200 300]);
imshow(select_image, [])
title('Sparse Foreground, Rank 10')
```

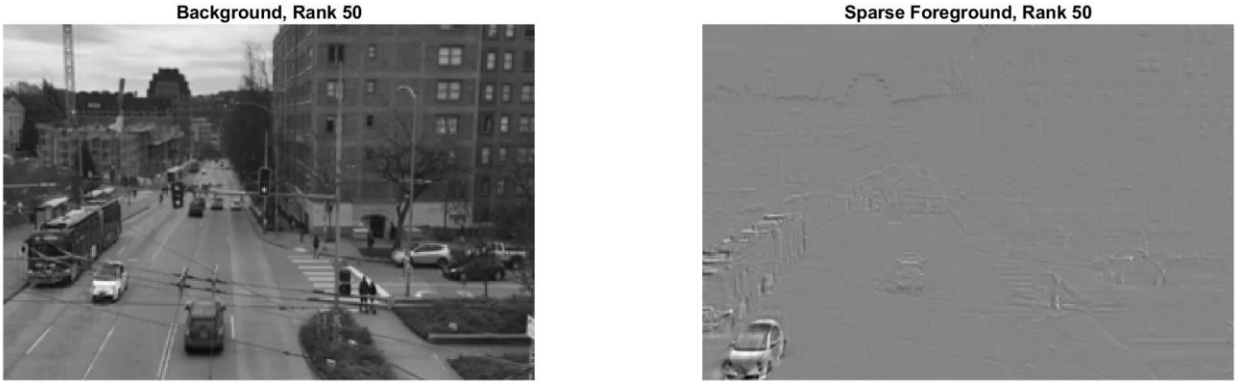
Appendix C: Southbound Skybridge Data



Appendix C Figure 1. *Southbound*, $R = 1$ *approx.*

$Appendix\ C\ Figure\ 2. Southbound\ R = 10\ approx.$

This demonstrates improved background subtraction, because more modes containing background features were captured by the SVD and removed from the foreground.



$Appendix\ C\ Figure\ 2. Southbound\ R = 50\ approx$

Nearly complete background subtraction, only the car remains.