Background Subtraction via Dynamic Mode Decomposition

Nick Moore
March 3, 2017

Abstract

Dynamic mode decomposition (DMD) is a data driven analysis technique where the modes of high dimensional, nonlinear data are approximated based on snapshots of the data. DMD was originally developed for fluid dynamics studies, but has applications in a variety of other fields including video analysis, finance, and neuroscience. Herein two videos with moving objects (cars driving or people walking) undergo DMD and are reconstructed into two matrices $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ and $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$, where low-rank contains the background information and sparse contains the foreground information.[1] The DMD method was successful in separating the foreground and background for both videos, shown in figure 1.1.

Sec. I. Introduction & Overview

The goal of this report is to apply dynamic mode decomposition to videos with a moving foreground object to separate the foreground and background into two videos. Two sample videos were recorded, one of cars moving down the street and the other of people walking. Each video was imported and reshaped to form a two-dimensional matrix X, where the column vectors contain the image information of each frame and each column is a new frame. Matrix X underwent DMD to generate matrices $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$and $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ (1) where $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ contains the background information and X_{DMD}^{Sparse} contains the foreground information.[1] The $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ and $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ matrices were reshaped back to original video size and viewed to determine the success of the method. Both videos were separated into background and foreground videos.

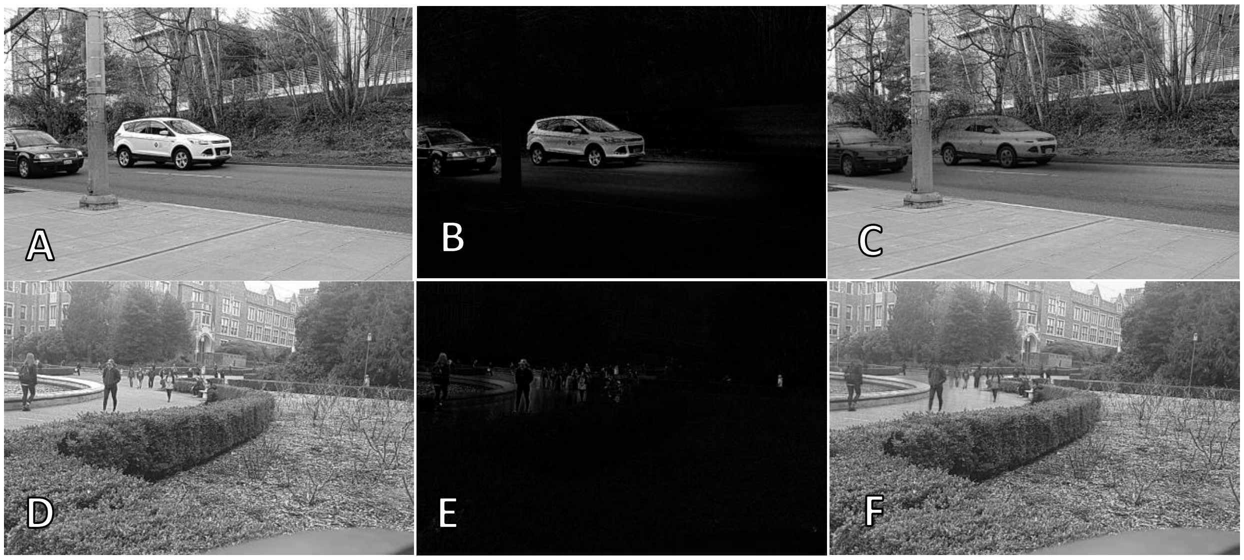

Figure 1.1: Selected frames from DMD results from the car (A-C) and people (D-F) videos. (A) and (D) shows the original frames in grayscale, (B) and (E) shows the $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$, and (C) and (F) shows the $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$. Ranks were reduced to 5 for both videos.

Figure 1.1 shows frames from the original, the foreground (sparse), and the background (low-rank) videos. Figure 1.1B shows that $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ can capture the foreground objects (cars) while ignoring the background. Matlab 2016b was used for all the work presented. The functions used and the entire code can be found in Appendices A & B respectively.

$$X = X_{DMD}^{Low-Rank} + X_{DMD}^{Sparse} \tag{1.1}$$[1]

Sec. II. Theoretical Background

DMD is a data driven method of decomposition that generates dynamic modes from snapshots of data. The snapshot data is used to approximate the dynamical nonlinear modes of the system given.[1] Given a function (2.1) DMD creates a discrete time (Δt) expression (2.2). An approximation of the discrete time function is expressed as (2.3).[1] This is further used to generate the expression (2.4) where ϕ_k are the eigenvectors and λ_k are the eigenvalues of the system.[1] The matrix A is minimized for all k values of x_k given by (2.5).[1] Once minimized the eigenvectors of A (ϕ_k) are the dynamic modes of the system.[1] The DMD algorithm, described in the next section, takes a different approach that utilizes two snapshot matrices X1 and X2 (3.1) to approximate Ã and in turn used to compute the dynamic modes.[1]

$$dx/_{dt} = F(x, t; \mu) \tag{2.1}$$[1]

$$x_{k+1} = F(x_k) \tag{2.2}$$[1]

$$x_{k+1} = Ax_k \tag{2.3}$$[1]

$$x_k = \sum_{j=1}^{r} \varphi_j \, \lambda_j^k b_j \tag{2.4}$$[1]

$$\|x_{k+1} - Ax_k\|_2 \tag{2.5}$$[1]

Sec. III. Algorithm Implementation & Development

For the development of this algorithm the goal is to solve for the dynamic modes, which are eigenvectors of A from Ax = b. A custom function was written to separate the video streams (lines 1-69). The video files (mp4) were imported to MATLAB to generate three dimensional matrices and the relevant variables from the video file (lines 15-20). The matrices contained three-dimensional data (frame height, frame width, and frame number), they were then reshaped to column vectors and placed in matrix X (lines 24-28). Snapshots of X were arranged into two matrices, X1 and X2 (3.1) where the column vectors x_m contain the data from each video frame.[1] Singular value decomposition was applied to X1 to generate matrices U, Σ, and V (3.2) (line 34). Since the goal is to create matrix $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ the ranks of U, Σ, and V are truncated by a value input by the user (lines 35-37). SVD and rank truncation help to reduce the computation time of the data set and allow for the dynamic modes to be solved for via matrices U, Σ, and V. Matrix $\widetilde{\mathrm{A}}$ was computed from (3.3) (line 41) and underwent eigen decomposition (3.4) to give matrices W and Λ (line 32).[1] Matrix W contains the eigenvectors of $\widetilde{\mathrm{A}}$ as its columns and Λ contains the eigenvalues along its diagonal.[1] With the eigenvector matrix created the dynamic modes can be calculated with (3.5), where Φ contains the dynamic modes (eigenvectors of A) as its columns (line 43).[1] Ax = b can be solved for x_1 to give a value for b (line 47). Up to this point is a general DMD algorithm, but to separate the videos the $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ and $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ need to be solved for.

The first step is to solve for $X_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ based on (3.6) where $b_p = b$ from $Ax = b$ for x_1 (lines 49-56), and $X_{\mathrm{DMD}}^{\mathrm{Sparse}}$ was calculated with (3.7) (line 60).[1] The residual matrix R corrects for the complex values in $X_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ that are a product of (3.6) and any negative pixel intensities in $X_{\mathrm{DMD}}^{\mathrm{Sparse}}$ that arose from (3.7) (lines 70-71).[1] The corrections are shown in (3.8) and (3.9). R is computed by iterating through $X_{\mathrm{DMD}}^{\mathrm{Sparse}}$ and obtaining any values that are less than zero (lines 63-69).[1] The final step of the algorithm reshapes X, $X_{\mathrm{DMD}}^{\mathrm{Sparse}}$, and $X_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ to the original video dimensions for analysis (75-77).

$$X1 = [x_1 \quad x_2 \cdots \quad x_{m-1}] \qquad X2 = [x_2 \quad x_3 \cdots \quad x_m] \tag{3.1}$$

$$X1 = U\Sigma V \tag{3.2}$$

$$\tilde{A} = U^{*}AU = U^{*}X2V\Sigma^{-1} \tag{3.3}$$

$$\tilde{A}W = W\Lambda \tag{3.4}$$

$$\Phi = X2'V\Sigma^{-1}W \tag{3.5}$$

$$X_{DMD}^{Low-Rank} = b_p \varphi_p e^{\omega_p t} \tag{3.6}$$

$$X_{DMD}^{Sparse} = X - \left| X_{DMD}^{Low-Rank} \right| \tag{3.7}$$

$$X_{DMD}^{Low-Rank} \leftarrow R + \left| X_{DMD}^{Low-Rank} \right| \tag{3.8}$$

$$X_{DMD}^{Sparse} \leftarrow X_{DMD}^{Sparse} - R \tag{3.9}$$

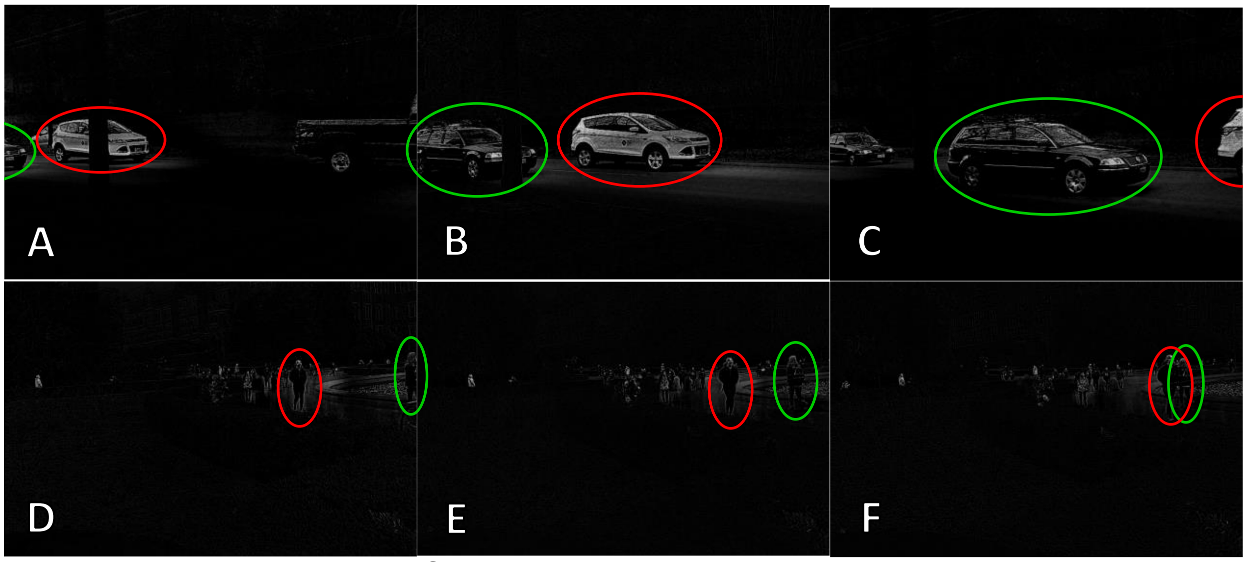

Figure 4.1: Selected frames from $X_{\mathrm{DMD}}^{\mathrm{Sparse}}$ the car (A-C) and people walking (D-F) videos. Frames are 1 second each 1 second apart and show the movement of two foreground objects, circled in red and green. Ranks were reduced to 5 for both videos.

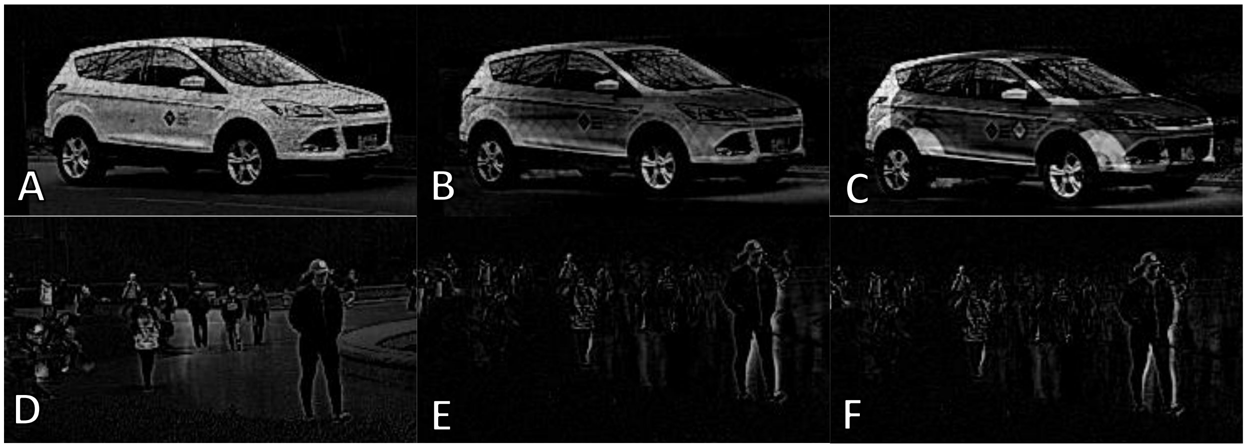

Figure 4.2: Selected, cropped frames from the $X_{\text{DMD}}^{\text{Sparse}}$ car (A-C) and people (D-F) videos with rank truncations (A) 1, (B) 50, (C) 180 (all ranks), (D) 1, (E) 200, (F) 539 (all ranks).

Sec IV. Computational Results

The DMD algorithm was tested on two videos, one of cars driving down the street and the other of people walking. The algorithm and function developed were able distinguish the foreground objects and separate them from the background. Frames from the original video, $X_{\text{DMD}}^{\text{Sparse}}$, and $X_{\text{DMD}}^{\text{Low-Rank}}$ are shown previously in figure 1.1. The $X_{\text{DMD}}^{\text{Sparse}}$ frames from both videos highlight the foreground objects in white, while the static background remains black, so these streams can be used to track moving objects independently of the background. The advantage here is in saving storage space through the background removal. The reconstructions can use as few as a single mode (shown in figure 4.2.A and 4.2.D) for successful separation, so the data that needs to be stored is significantly smaller than the original video file. Figure 4.1 shows three frames that are 1 second (30 frames) apart. For each video, there are two circled objects, one red and one green.

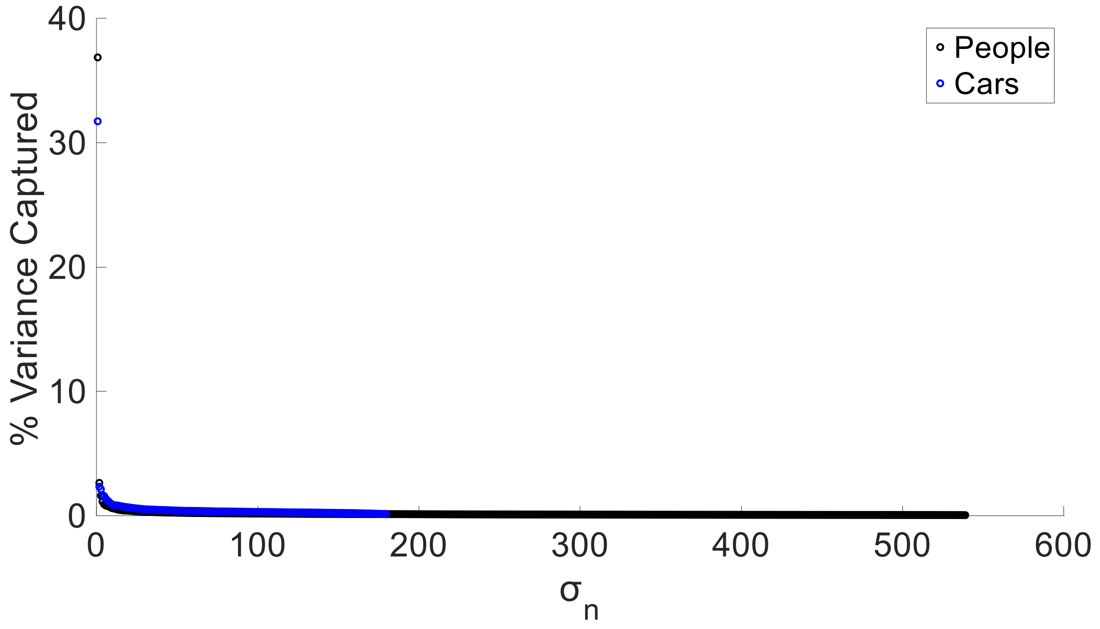

Figure 4.3: Percent variance captreed by each σ_n for the cars (blue) and people (black).

The frames show the movement of the foreground objects through the video with the position of the objects tracked. The DMD algorithm was successfully able to track the movement of different objects in both videos tested. Figure 4.2 shows frames from both videos with different values for the rank reduction. For each set of 3 frames the first (A, C) are reduced to a rank of one, the second (B, E) are reduced to a mid-range rank, and the final (C, F) are not rank reduced. Different rank reductions were chosen because the maximum number of ranks is dependent on the number of frames of the video. These images show that the first rank is enough to track the motion of foreground objects, and in both cases the foreground objects lose intensity versus the black background as the rank increases. This is because the rank equates to the number of dynamic modes, so the higher number dynamic modes that capture significantly less variance are used in the reconstructions. This is supported by the σ_n values for both videos, shown in figure 4.3. The first σ_n is able to capture 36.84% and 31.70% of the variance for the people and cars, respectively. The percent variance captrued drops off immediately, which is why we see little variation in the images in figure 4.2.

Sec. V. Summary & Conclusions

Dynamic mode decomposition was applied to two videos with objects in motion to separate the foreground and background into two videos. The first video used was of cars driving down a street, and the second of people walking near a fountain. The same DMD was applied to both videos and could separate the videos without any tweaking or adjustments for the different videos. The algorithm can be applied generally to any video with ease as the only required inputs are the file name and rank reduction. The $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ videos are display the foreground (moving) objects, while $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Low-Rank}}$ contains the background information. Sparse videos can be used to track the motion of objects independent of the background. This method can be utilized for motion tracking software and video size reduction because only the data for the moving objects are stored in the $\mathrm{X}_{\mathrm{DMD}}^{\mathrm{Sparse}}$ matrix. The DMD algorithm can be modified for other types of input to apply the same methodology to a variety of other data sets.

References

(1) J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, J. L. P. Dynamic Mode Decomposition. 1–26.

Appendix A: Matlab Function

DMD_VideoSeparation(filename, rank) – Takes video file and rank reduction inputs and outputs the original video (in grayscale) and the sparse and low-rank videos.

Eig(A) – Returns the eigenvectors and eigenvalues of the input matrix A.

Videoreader(filename) – Takes a video file and converts it to a Matlab object.

Appendix B: Matlab Code

```
function [mov, mov_sp, mov_low, frame_num] = DMD_VideoSeparation(filename, rank)
% Function takes a video file and outputs a grayscale video, grayscale background video, and grayscale foreground video.

```

```
    % Inputs
    % filename - Location/name of video file as a string.
    % rank - integer value for rank reduction

    % Outputs
    % mov - video of X
    % mov_sp - video of X_sparse (foreground)
    % mov_low - video of X_low (background)

% Convert video file to matrix and variables

video = VideoReader(filename);
height = video.height;
width = video.width;
dt = 1/video.FrameRate;
video_matrix = read(video);
frame_num = size(video_matrix,4);

% Generate X from matrix

X = zeros(width*height, frame_num);
for iter = 1:frame_num
    video_gray = rgb2gray(im2double(video_matrix(:,:,:,iter)));
    X(:,iter) = video_gray(:);
end
X1 = X(:,1:end-1);
X2 = X(:,2:end);

% SVD

[U,S,V] = svd(X1,'econ');
U_r = U(:,1:rank);
S_r = S(1:rank,1:rank);
V_r = V(:,1:rank);

% DMD

Atilde = U_r' * X2 * V_r*S_r^-1;
[W, D] = eig(Atilde);
Phi = X2 * V_r/S_r * W;
lambda = diag(D);
omega = log(lambda)/(dt);
x1 = X1(:,1);
b = Phi\x1;

% X_low

time = (1:frame_num).*dt;
```

```
time_dynamics = zeros(rank, frame_num);
for k = 1:length(time)
  time_dynamics(:,k) = (b.*exp(omega*time(k)));
end
X_low = Phi*time_dynamics;

% X_sp

X_sp = X - abs(X_low);
R = zeros(length(X_sp),frame_num);

for iter = 1:length(X_sp(:,1))
   for iter2 = 1:length(X_sp(1,:))
      if X_sp(iter,iter2) < 0
         R(iter,iter2) = X_sp(iter,iter2);
      end
   end
end
X_sp = X_sp - R;
X_low = R + abs(X_low);

% Reshape for videos

mov = reshape(X, height, width, frame_num);
mov_sp = reshape(X_sp, height, width, frame_num);
mov_low = reshape(X_low, height, width, frame_num);

end
%% Script for running DMD_VideoSeparation and figure generationh
clear; clc;
[mov_cars, mov_cars_sp, mov_cars_low, S_cars, U_cars, V_cars] =
DMD_VideoSeparation('C:\Users\Nick Moore\Documents\MATLAB\cars.mp4', 5);
[mov_people, mov_people_sp, mov_people_low, S_people, U_people, V_people] =
DMD_VideoSeparation('C:\Users\Nick Moore\Documents\MATLAB\people.mp4', 5);

%%
implay(mov_cars_sp);
implay(mov_cars_low);
implay(mov_cars);

figure(1); imshow(mov_cars_sp(:,:,10));
figure(2); imshow(mov_cars_sp(:,:,40));
figure(3); imshow(mov_cars_sp(:,:,70));
figure(4); imshow(mov_cars_sp(:,:,30));
figure(5); imshow(mov_cars_low(:,:,30));

implay(mov_people_sp);
implay(mov_people_low);
implay(mov_people);

figure(6); imshow(mov_people_sp(:,:,10));
figure(7); imshow(mov_people_sp(:,:,40));
figure(8); imshow(mov_people_sp(:,:,70));
figure(9); imshow(mov_people_sp(:,:,30));
figure(10); imshow(mov_people_low(:,:,30));

%%

ranks = [1 250 539];

for iter = ranks
    [mov_people, mov_people_sp, mov_people_low] = DMD_VideoSeparation('C:\Users\Nick Moore\Documents\MATLAB\people.mp4', iter);
    figure; imshow(mov_cars_sp(:,:,30));
end

ranks = [1 50 180];

for iter = ranks
    [mov_cars, mov_cars_sp, mov_cars_low] = DMD_VideoSeparation('C:\Users\Nick Moore\Documents\MATLAB\cars.mp4', iter);
    figure; imshow(mov_cars_sp(:,:,30));
end

%%

figure(11); hold on;
plot(diag(S_people)./sum(diag(S_people))*100,'ko','Linewidth',[2]);
plot(diag(S_cars)./sum(diag(S_cars))*100,'bo','Linewidth',[2]);
xlabel('?_n','FontSize',35); ylabel('% Variance Captured','FontSize',35);
```