AMATH HW 5

Zoheb Siddiqui

March 15, 2019

Abstract

In this assignment we use dynamic mode decomposition to separate the foreground of a video from the background.

Introduction

We have a number of test videos in which the foreground consists of moving objects and the background is static. We must use DMD to create a low rank reconstruction of the original video which represents the background. Then we much subtract this low rank reconstruction from the original video to obtain the moving foreground.

Theoretical Background

In DMD we analyze a dynamic system $\frac{dx}{dt} = f(x, t, \mu)$ s.t $x(t) \in R^n$ represents the state of the system at time t and μ represents the parameters of the system.
The goal is to find a matrix A s.t $\frac{dx}{dt} = A$ s.t $||x_{k+1} - Ax_k||_2$ is minimized. This will also result in $x_{k+1} \approx Ax_k$.
We break the data matrix X down in two components X_1 and X_2 s.t the ith column in X_1 is the i-1th column in X_2.
Thus, we must find an A s.t $X_2 \approx AX_1$. A is now given by $X_2X_1^+$ where X_1^+ represents the pseudo-inverse of X_1.
Since the vectors in X can be high dimensional the matrix A will be too. However we can solve for a low rank approximation of A. Let this be $\tilde{A}$.
Now, the low rank approximation to X will be given as $x(t) = \Phi e^{\Omega t} b$ where ϕ_k and ω_k are the eigenvectors and values of $\tilde{A}$ and the coefficients b_k are the coordinates of x(0) in the eigenvector basis.

Now, let's talk about DMD in terms of background separation.
We assume that the background consists of all components that remain stationary throughout the video. Thus, we can say that the Fourier mode associated with the background is ω_p s.t $||\omega_p|| \approx 0$.
Thus, background $= b_p\phi_p e^{\omega_p t}$ and foreground $= \sum\limits_{k \neq p} b_k\phi_k e^{\omega_k t}$.

Now let $X_{DMD}^{low\,rank} = b_p\phi_p e^{\omega_p t}$ and $X_{DMD}^{sparse} = \sum_{k\neq p} b_k\phi_k e^{\omega_k t}$. Since X = background + foreground = $X_{DMD}^{low\,rank} + X_{DMD}^{sparse}$ thus, $X_{DMD}^{sparse} = X - |X_{DMD}^{low\,rank}|$. The absolute value is necessary as we must only consider the real value components of the low rank matrix.
This can possibly result in the sparse matrix having negative values thus, we must find this residual negative value matrix R and alter the low rank matrix and the sparse matrix as follows:

$X_{DMD}^{low\,rank} = |X_{DMD}^{low\,rank}| + R$ and $X_{DMD}^{sparse} = X_{DMD}^{sparse} - R$.

In practice we will not add R to the low rank matrix or subtract it from the sparse matrix. This is because when R is added to the low rank matrix we are adding a part of the foreground into the background and the separation is not good.
We will instead say that $X_{DMD}^{sparse} = |X_{DMD}^{sparse}|$. Note that in doing this we have lost the integrity of the DMD and we can no longer get the X matrix back by adding the low rank and sparse matrices together.

Algorithm

```
Read the .mp4 file using the VideoReader() function. Name it v.
min_frames = v.NumberOFFrames;
frames = read(v).

Declare empty X matrix to store flattened frames.
For each frame in video:
    convert frame to gray scale using rgb2gray().
    flatten frame to 1d vector.
    X = [X;new_frame].

dt = 1/frame_rate.
t = 0:dt:min_frames.
X1 = X(:,1:end-1).
X2 = X(:,2:end).

[U,S,V] = svd(X1, 'econ').
plot(diag(S)).
r = 10. % performing rank reduction
Ur = U(:,1:r), Sr = S(1:r,1:r), Vr = V(:,1:r).

Now we find Atilde and the matrix containing the DMD modes, $\phi$.
Atilde = Ur'*X2*Vr/Sr. % low rank approximation of A
[W,D] = eig(Atilde)
$\phi$ = X2*Vr/Sr*W.

Now we find the discreet and continuous time eigenvalues respectively.
$\lambda = diag(D)$, $\omega = \frac{\log(\lambda)}{dt}$.
```

Now we compute b, the DMD mode amplitude.
b = $\phi\backslash$X(:,1).

Since the background is static thus, in the reconstruction of the low rank matrix we shall use the smallest value of ω as it accounts for the least variance.
ω_{min} = min(abs(ω)).

Now we find the low rank matrix X_low_rank.
X_low_rank = abs($\phi * (b. * e^{(\omega_{min} * t(1,1:min\_frames))})$).
X_sparse = X - X_low_rank.

Now we need to find the Residual matrix.
Residual = X_sparse .* (X_sparse < 0).

X_low_rank = X_low_rank + Residual, X_sparse = X_sparse - Residual.
Note: This step is important to get back to the original X matrix however we do not perform it in the actual implementation to get good separation.

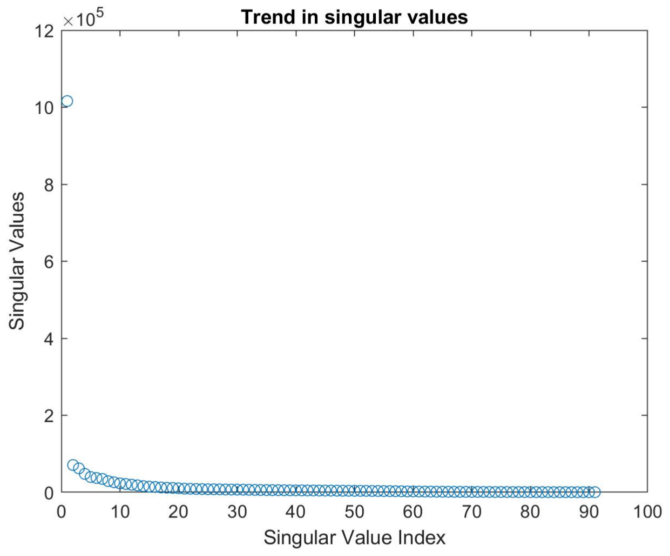
X_low_rank is the background and X_sparse is the foreground.

```
For each frame:
    plot frame in X.
    plot frame in X_low_rank.
    plot frame in X_sparse.
```

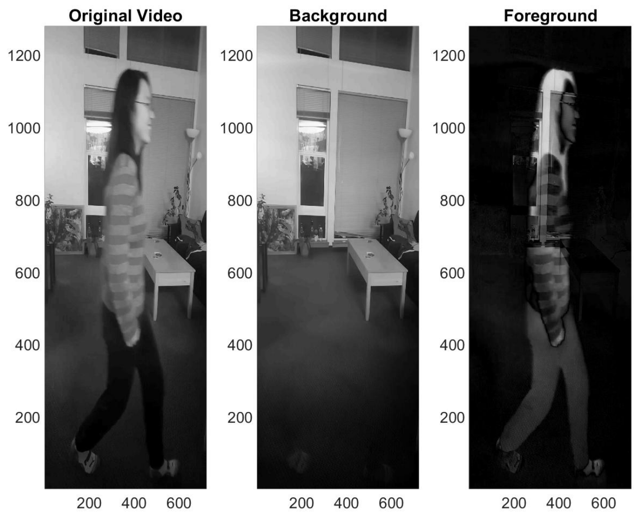
Computational Results

Test Case 1: This video is of a girl walking in a room.

The trend in the singular values is:

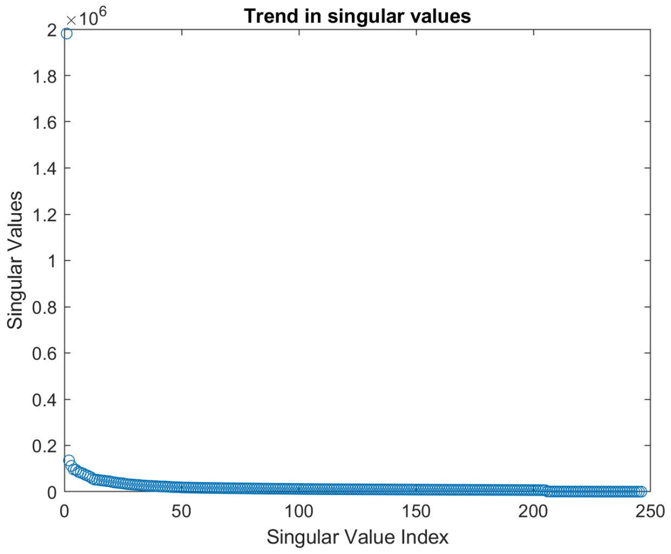


The results of DMD:

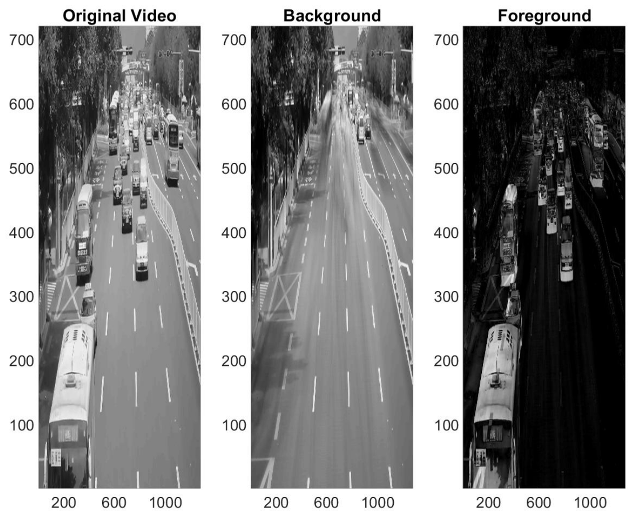


Test Case 2: This video is of a traffic camera.

The trend in the singular values is:

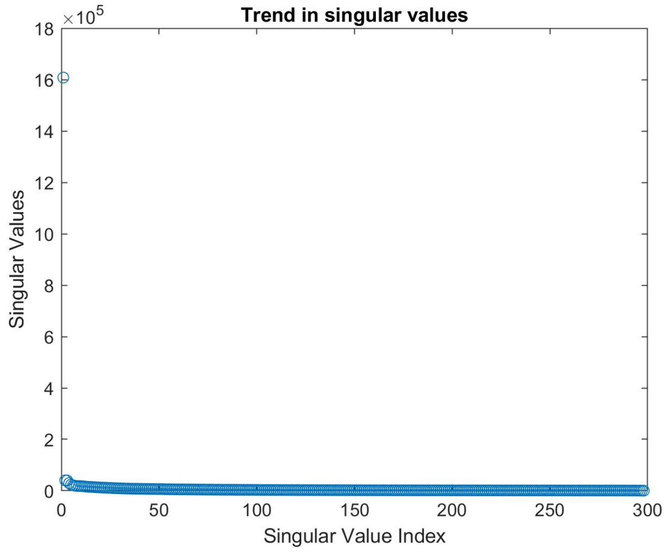


The results of DMD:

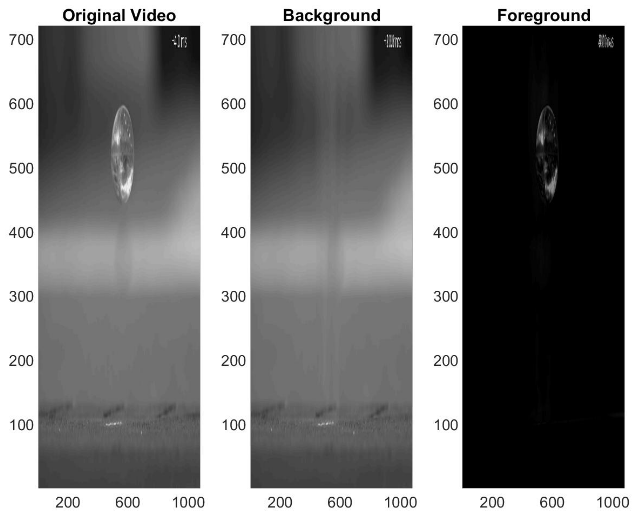


Test Case 3: This video is of a ball bouncing.

The trend in the singular values is:



The results of DMD:



Conclusion

When we look at the singular values of X1 we notice that the first 10 singular vlaues account for nearly 65% of the information stored in X1. Thus, we can perform dimentionality reduction on the output of SVD(X1) and reduce the rank to 10.

Notice that in the first video the intensity of light is greater in the background than in the foreground. While the algorithm extracts the outline of the foreground correctly, the information stored is largely overshadowed by the background and thus we can see some of the background through the moving object.

Another key observation to make is that this technique only works when the background is stationary and there is no camera shake as any bit of movement in the background or presence of noise greatly affects the output.

In my computational results I do not utilize the Residual as it leads to improper separation. This is because when we add the residual to the low rank matrix we are adding a part of the foreground back to the low rank matrix. While, this is necessary if we want to get out original X matrix back, it is not great for our task.

Appendices

Citation

Kutz, Nathan. Section 4.3: Background Separation. Chapter 4: Video Processing Dynamic Mode Decomposition — Society for Industrial and Applied Mathematics.

epubs.siam.org/doi/pdf/10.1137/1.9781611974508.ch4.

Matlab functions

VideoReader(): Reads in a video and extracts information like number of frames, frame rate and the data matrix.
rgb2gray(): Converts a colored image to gray scale.
svd(): Performs SVD on a matrix to get [U,S,V] matrices.
eig(): performs eigenvalue decomposition on a square matrix to get eigenvectors and eigenvalues of the matrix.

Code

```
clear all; close all; clc
%% Loading video from reader
v = VideoReader('Data/test_3.mp4');
min_frames = v.NumberOFFrames;
v = VideoReader('Data/test_3.mp4');
frames_1 = read(v);
%% Converting to gray scale
frames_rgb = zeros(size(frames_1, 1)*...
    size(frames_1, 2),min_frames);
for i = 1:min_frames
    frames_rgb(:,i) = reshape(rgb2gray(frames_1(:,:,:,i)), ...
        size(frames_1, 1)*size(frames_1, 2), 1);
end
%% Creating DMD matrices
X = frames_rgb;
dt = 1;
t = 0:dt:min_frames;

X1 = X(:,1:end-1);
X2 = X(:,2:end);
%% SVD
[U,S,V] = svd(X1, 'econ');
%% Finding a good r to use
figure()
plot(diag(S), 'o');
title('Trend in singular values')
xlabel('Singular Value Index')
ylabel('Singular Values')
r = 10;
%% Doing rank reduction
```

```
Ur = U(:,1:r);
Sr = S(1:r,1:r);
Vr = V(:,1:r);
%% Finding A_tilde and DMD modes
Atilde = Ur'*X2*Vr/Sr;
[W,D] = eig(Atilde);
Phi = X2*Vr/Sr*W;
%% DMD spectra
lambda = diag(D);
omega = log(lambda)/dt;

omega_to_use = min(abs(omega));
%% %% find DMD solution
b = Phi\X(:,1);
time_dynamics = zeros(r,min_frames);
for iter = 1:min_frames
    time_dynamics(:,iter) = (b.*exp(omega_to_use*t(iter)));
end
X_dmd = Phi*time_dynamics;
%% Finding sparse with residual
X_sparse = X - abs(X_dmd);
%% Finding residual
Residual = X_sparse .* (X_sparse < 0);

%% removing residual
X_dmd = abs(X_dmd);% + Residual;
%X_sparse = X_sparse;% - Residual;
%% plot X_dmd
figure()
for i = 40:41
    subplot(1,3,1)
    to_show = X(:,i);
    to_show = reshape(to_show,[size(frames_1, 1), size(frames_1, 2)]);
    pcolor(flipud(abs(to_show))), shading interp, colormap gray;
    title('Original Video')

    subplot(1,3,2)
    to_show = X_dmd(:,i);
    to_show = reshape(to_show,[size(frames_1, 1), size(frames_1, 2)]);
    pcolor(flipud(abs(to_show))), shading interp, colormap gray;
    title('Background')

    subplot(1,3,3)
    to_show = X_sparse(:,i);
    to_show = reshape(to_show,[size(frames_1, 1), size(frames_1, 2)]);
    pcolor(flipud(abs(to_show))), shading interp, colormap gray;
    title('Foreground')
    drawnow
end
```