Background Separation from Video Streams

Jithin D. George

July 20, 2017

Abstract

Unlike the singular value decomposition, the dynamic mode decomposition makes use of the properties of the fourier transform to incorporate time dynamics into our analysis. As a result, we gain new perspectives and a host of applications like prediction of the future state and here, the ability to obtain the background and foreground from a video.

1 Introduction and Overview

The Dynamic Mode Decomposition allows us to explore different perspectives into data. Here, we go ahead into separating the background and foreground out of a video after seeing the promising results of [2].

2 Theoretical Background

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). DMD has recently been extended to include the effect of control to extract low-order models from externally forced or controlled high-dimensional complex systems.The new method of dynamic mode decomposition with control (DMDc) provides the ability to disambiguate between the underlying dynamics and the effects of actuation, resulting in accurate input-output models like explained in [1]

3 Implementation and Development

Here, we consider two videos. One of the videos is a man walking across the room and the other is a computer generated animation. We use the DMD on them and find the mode with the ω closest. This should correspond to the mode that doesn't change with time and thus the background. We then separate the background out to reveal the foreground.

4 Computational Results

4.1 Movement across a room

The DMD works very well for this case. The original image is shown in 1. The background is obtained to a near perfect state as shown with the foreground in 2. Since we can't scale the foreground accurately, we can only see the white outline of the person moving across. There seems to be a sort of obstruction towards the end of the videoframes which makes the visibility low.

Figure 1: Original Image

Figure 2: Background and Foreground

4.2 Computer generated video

For this case, the DMD works well for the background. The original image is shown in 3. The background is obtained to a near perfect state as shown with the foreground in 4. The background is devoid of the moving black car. The foreground however contains trees because they move slightly. The car is

missing except for a small light at its position. This is because the car is black in color

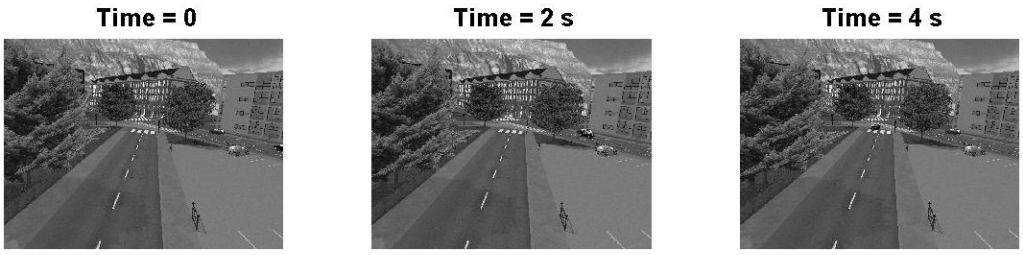

Figure 3: Original Image

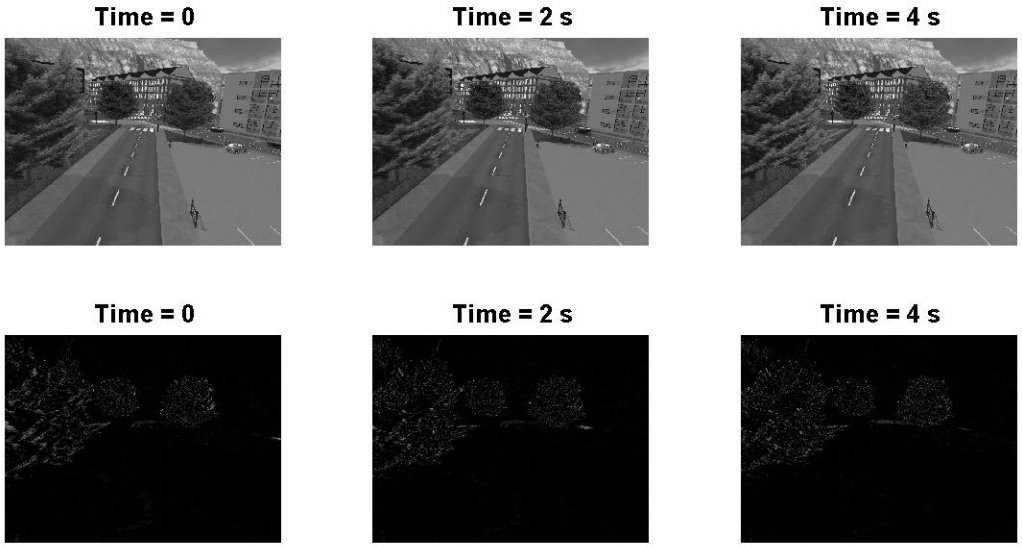

Figure 4: Background and Foreground

4.3 Future State Prediction

We extend the time beyond the length of the videos and show our results in5. The result for the moving man is perfect since he is leaving the video. In the animation, the car takes a turn and changes direction. However, there is hardly a few frames in the new direction. So, the future state seems empty because the car seems to vanish behind the tree. Using a few more frames in the new direction and many more modes, we obtain the correct future state in 6

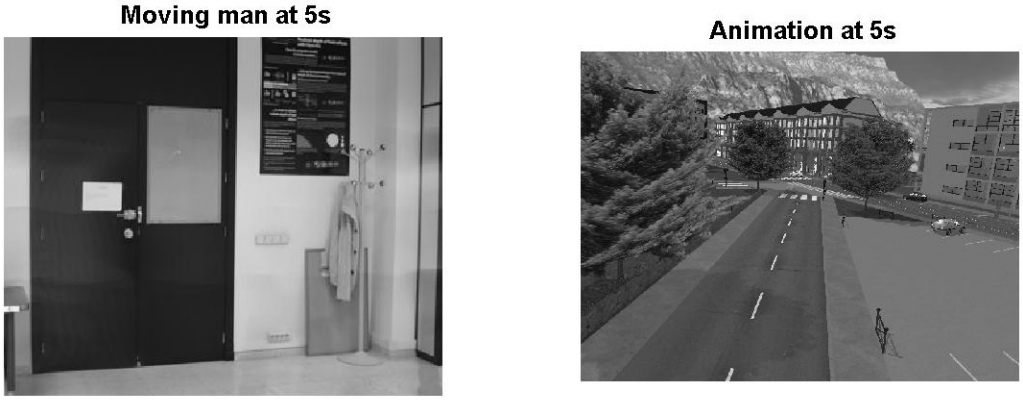

Figure 5: Background and Foreground

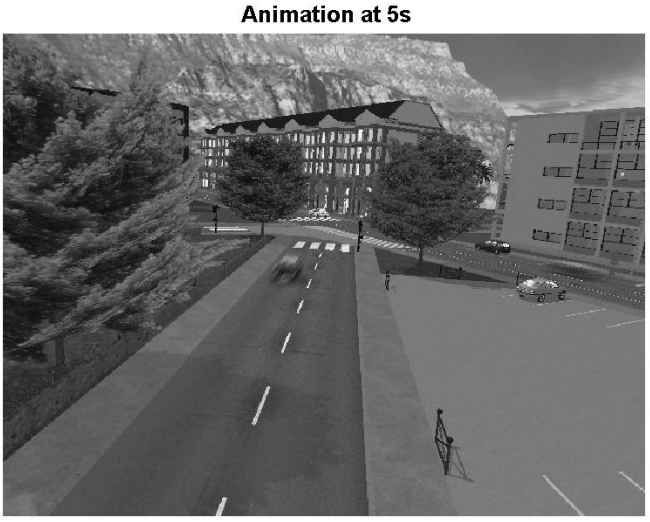

Figure 6: Refined future state

5 Summary and Conclusions

We obtain the following conclusions

- The background can be obtained with relative ease in both cases.
- The DMD shows an incredible ability to predict the future state of the video.
- Black moving objects are harder to detect.
- Slightly moving objects like the trees can be detected and affect the imaging.
- Direction change of the foreground and lack of data in the new direction leads to erroneous future state prediction.
- Using more modes leads to better clarity and better prediction.

References

[1] J. Proctor, S. Brunton and J. N. Kutz, Dynamic mode decomposition with control, arXiv:1409.6358.

[2] J. Grosek and J. N. Kutz, Dynamic mode decomposition for real-time background/foreground separation in video, arXiv:1404.7592.

A MATLAB Functions used

- **[U,S,V]=svd(A):**
  This function performs the singular value decomposition of A and returns U,S and V.
- **mat2gray(A):**
  This function converts a matrix to greyscale.

B MATLAB Code

B.1 hw4.m

```
t=dir;
numFrames = 63;
dt = 3/numFrames;
Y=zeros(288*352,63);
for j=3:numFrames+2
  p= t(j);
  s=p.name;
  X=imread(s);
  X= rgb2gray(X);
  imshow(X); drawnow
  Y(:,j-2)= reshape(X,[288*352,1]);
end

Y2= Y(:,2:end);
Y1= Y(:,1:numFrames-1);
[Phi,omega,lambda,b,Xdmd] = DMD(Y1,Y2,15,dt);



%% DMD reconstruction
mm1 = size(Y1, 2)+10; % mm1 = m - 1
time_dynamics11 = zeros(1, mm1);
b11=b(11,1);
t = (0:mm1-1)*dt; % time vector
for iter = 1:mm1,
  time_dynamics11(:,iter)=(b11.*exp(omega(11,1)*t(iter)));
end;
Phi11= Phi(:,11);
Xdmd11 = Phi11 * time_dynamics11;
ab= Xdmd11(:,20);
abr = reshape(ab,288,352);

abr = mat2gray(real(abr));

Xsparse = Xdmd-abs(Xdmd11);
R= zeros(size(Xsparse));
R(Xsparse<0)= Xsparse(Xsparse<0);
Xback = R+ abs(Xdmd11);
Xsparse = Xsparse- R;
for j = 1:70
  ab= Xsparse(:,j);
  abr = reshape(ab,288,352);
  abr = mat2gray(abs(abr));
  imshow(abr);drawnow
end
```

B.2 plothw4.m

```
figure;
subplot(1,3,1);
f=20;
A1= reshape(Y(:,1),[dim1,dim2]);
A1g=mat2gray(abs(A1));
imshow(A1g);
title('\bf Time = 0','FontSize', f)
subplot(1,3,2);
A2= reshape(Y(:,40),[dim1,dim2]);
A2g=mat2gray(abs(A2));
imshow(A2g);
title('\bf Time = 2 s','FontSize', f)
subplot(1,3,3);
A3= reshape(Y(:,90),[dim1,dim2]);
A3g=mat2gray(abs(A3));
imshow(A3g);
title('\bf Time = 4 s','FontSize', f)

figure;
subplot(2,3,1);
f=20;
A1= reshape(Xback(:,1),[dim1,dim2]);
A1g=mat2gray(abs(A1));
imshow(A1g);
title('\bf Time = 0','FontSize', f)
subplot(2,3,2);
A2= reshape(Xback(:,40),[dim1,dim2]);
A2g=mat2gray(abs(A2));
imshow(A2g);
title('\bf Time = 2 s','FontSize', f)
subplot(2,3,3);
A3= reshape(Xback(:,90),[dim1,dim2]);
A3g=mat2gray(abs(A3));
imshow(A3g);
title('\bf Time = 4 s','FontSize', f)
%
subplot(2,3,4);
f=20;
A1= reshape(Xsparse(:,1),[dim1,dim2]);
A1g=mat2gray(abs(A1));
imshow(A1g);
title('\bf Time = 0','FontSize', f)
subplot(2,3,5);
A2= reshape(Xsparse(:,40),[dim1,dim2]);
A2g=mat2gray(abs(A2));
imshow(A2g);
title('\bf Time = 2 s','FontSize', f)
subplot(2,3,6);
A3= reshape(Xsparse(:,90),[dim1,dim2]);
A3g=mat2gray(abs(A3));
imshow(A3g);
title('\bf Time = 4 s','FontSize', f)
```

B.3 DMD.m

```
function [Phi,omega,lambda,b,Xdmd] = DMD(X1,X2,r,dt)
% function [Phi,omega,lambda,b,Xdmd] = DMD(X1,X2,r,dt)
% Computes the Dynamic Mode Decomposition of X1, X2
%
% INPUTS:
% X1 = X, data matrix
% X2 = X', shifted data matrix
% Columns of X1 and X2 are state snapshots
```

```
% r = target rank of SVD
% dt = time step advancing X1 to X2 (X to X')
%
% OUTPUTS:
% Phi, the DMD modes
% omega, the continuous-time DMD eigenvalues
% lambda, the discrete-time DMD eigenvalues
% b, a vector of magnitudes of modes Phi
% Xdmd, the data matrix reconstrcted by Phi, omega, b

%% DMD
[U, S, V] = svd(X1, 'econ');
r = min(r, size(U,2));
U_r = U(:, 1:r); % truncate to rank-r
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);
Atilde = U_r' * X2 * V_r / S_r; % low-rank dynamics
[W_r, D] = eig(Atilde);
Phi = X2 * V_r / S_r * W_r; % DMD modes
lambda = diag(D); % discrete-time eigenvalues
omega = log(lambda)/dt; % continuous-time eigenvalues
%% Compute DMD mode amplitudes b
x1 = X1(:, 1);
b = Phi\x1;

%% DMD reconstruction
mm1 = size(X1, 2)+10; % mm1 = m - 1
time_dynamics = zeros(r, mm1);
t = (0:mm1-1)*dt; % time vector
for iter = 1:mm1,
   time_dynamics(:,iter)=(b.*exp(omega*t(iter)));
end;
Xdmd = Phi * time_dynamics;
```