Dynamic Mode Decomposition Background Separation

Carlos Sama 17, March 2021

Abstract

This report details a method of applying dynamic mode decomposition (DMD) to video files for separating the background of the image. This was performed by finding the low rank components of the derived Koopman operator matrix which corresponded to the static background of each frame.

Introduction

The DMD technique has a vast range of applications, notably for solving complex non-linear systems such as partial differential equations. However, in this report, DMD is leveraged to create essentially a linear model of the video. The various modes that compose this model can be visually inspected to determine which elements correspond to specific functions of the video. This enables the ability to isolate low rank elements and modes that correspond to repetitive, non-dynamic aspects of the input video file as a background. After they are isolated, the video can be recreated without any of the foreground, or conversely, a sparse interpretation of the video modes can be found giving a video without the background.

Theoretical Background

For this algorithm to work over a video, each frame needs to be stored as an entry into a training data array that will serve to create the model. By relating each column of the array to the next the following linear equation can be derived [1].

$$\mathbf{X}_2^M = \mathbf{A}\mathbf{X}_1^{M-1} + \mathbf{r}e_{M-1}^T$$

By applying singular value decomposition of break down X1 and multiplying either side by U, an expression is formed which relates the Koopman Matrix A with the known quantities in X and it's SVD components [1].

$$\mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}.$$

Since this represents a similar group of matrixes, as one is just a change of basis from the other, the eigenvalues of each are the same and the model can be derived easily by finding the initial condition for the Koopman basis. The resulting model's solutions can be interpreted by simple ODE solutions of exponential equations. These solutions make up the modes of the response. Additionally, the eigenvalues can be analyzed for stability margins, and to identify low rank elements within the corresponding solution. The video background can be found from the low rank elements, and the foreground then conversely found by the sparse solution.

Algorithm Implementation

The defined algorithm was implemented in MATLAB using the built in SVD and pseudo-inverse calculation functionalities. Video files were read, and each frame was stored as a vector in a data matrix. This matrix served as the training data for creating the model and SVD was applied to the data matrix, with dimensional padding to enable the following calculations. The Koopman matrix side of the equation was calculated by the returned singular values and transition matrices, with the eigenvalues and vectors of that system easily found through MATLAB's built-in commands. The eigenvalues were transformed into a continuous time value so the exponential solutions could be found. At this point, the user is able to view the modes that

make up the solution. Next, the initial condition was determined using the pseudo-inverse of the derived model and the original video data matrix can be reproduced.

In order to isolate the background however, the eigenvalues closest to zero were determined and their corresponding solutions isolated to reproduce the data with only that information. This data was then subtracted from the initial video file to yield the foreground information, with the negative residuals removed from both ends of the foreground and background data.

Results

The implementation of DMD was able to successfully remove the background from the video, leaving the foreground, or objects of interest left in the view. The video of the Monte Carlo Grand Prix finish used just three modes to identify the background. Snapshots from the same frame of the original video, as well as the recreated background and background-removed videos are shown in Figure 1.

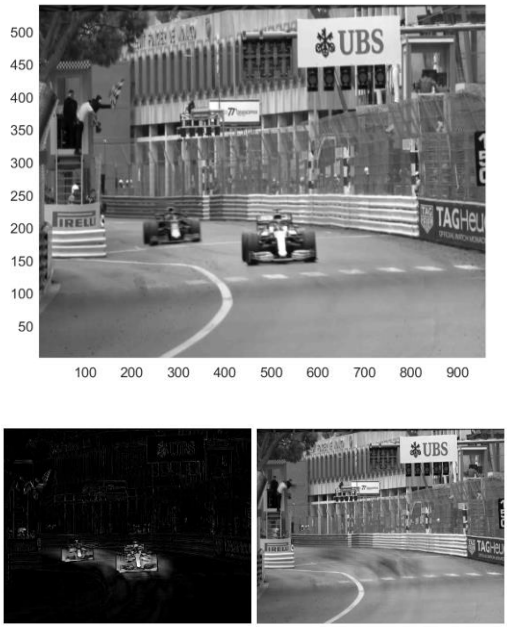

Figure 1 (top) provided video of Monte Carlo GP finish (left) sparse recreation with subtracted background (right) identified background.

The video of the skier required more attention to detail, since as the skier moved along the image, the wake of snow and disturbance left a trail of the interpreted background. However, the algorithm was able so successfully identify the skier, despite only being a handful of pixels. Snapshots from the same frame of the original and edited videos are shown in Figure 2.

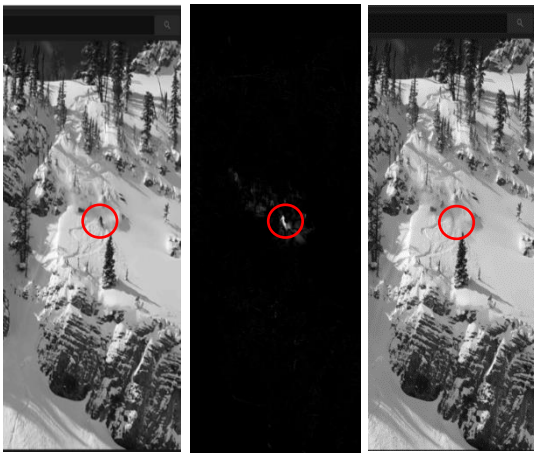

Figure 2 (left) frame from provided video (middle) sparse recreation with background subtracted (right) background frame.

Conclusion

By applying Dynamic Mode Decomposition to video data matrices, the overall background of the data can be identified and subtracted, leaving a sparse data matrix. This resulting matrix contains moving items of interest, who's dynamic behavior is captured and modeled by a Koopman matrix. This algorithm works very well in identifying moving objects of interest when provided stable footage of the tracked objects.

Appendix A

Utilized MATLAB Functions:

`svd:` This function returns U, V, and Sigma matrix of a provided matrix as calculated by singular value decomposition

eig: This function returns the eigenvalues and eigenvectors of a provided matrix

'/': Solves the matrix equation Ax = b by choosing the best method given the provided matrix. In the case of this example, it calculates and applies the pseudo-inverse.

VideoReader: Reads video files to be called frame by frame as uint8 image matrices.

pcolor: This function produces a "checkerboard" color plot of the provided array data tabulated by provided x and y axis breakpoints. Various color patterns can be used to delineate intensity. For this case, the "hot" color pattern was used.

Appendix B

MATLAB CODE

```
clear all; close all;
%% Read in the Data

Vid = VideoReader('monte_carlo_low.mp4');

frames = round(Vid.Duration * Vid.FrameRate);

X = zeros(Vid.width * Vid.height, frames);
for i = 1:frames
    X(:,i) =
reshape((double(rgb2gray(readFrame(Vid)))),
Vid.width * Vid.height, 1);
end
%%
dt = 1/Vid.FrameRate;
t = dt*(1:frames);

%% DMD Calculations

X1 = X(:,1:end-1);
X2 = X(:,2:end);

[U, Sig, V] = svd(X1, 'econ');
S = U'*X2*V*diag(1./diag(Sig));
[eigVec, eigVal] = eig(S);
mu = diag(eigVal);

w = log(mu)/dt;
%%
plot(abs(w));
%%

%Phi = U*eigVec;
clearvars U
Phi = X2 * V / Sig * eigVec;
y_init = Phi \ X1(:,1);

u_modes = zeros(length(y_init),length(t));
for i = 1:length(t)
   u_modes(:,i) = y_init.*exp(w*t(i));
end

%%
u_dmd = Phi*u_modes;
background = y_init.'.*Phi*exp(w.*t);
clearvars V  X1 X2
 %% Plot DMD modes
% close all
figure
%background = b.'.*Phi(:,191)*exp(omega.*t);
for i = 1:378

    pcolor(flipud(reshape(abs(Xdmd(:,i)),
Vid.height, Vid.width))); colormap gray, shading
interp;
    drawnow;

end

%%


 background = zeros(size(Phi));
backgroundIdx = find(abs(w) < 1);
background(:,backgroundIdx) =
u_dmd(:,backgroundIdx);
%%

R = background;
R(R>0) = 0;

background = background - R;

Xsparse = X - abs(background);
Xsparse = Xsparse - R;
%%
close all
for i = 1:length(backgroundIdx)

pcolor(flipud(reshape(abs(Phi(:,backgroundIdx(i))),
Vid.height, Vid.width))); colormap gray, shading
interp;
    drawnow;
end
```

References

[1] Bramburger, Jason “Background Subtraction in Video Streams,” University of Washing, Applied Mathematics, March, 2021