John Curtis

Assignment 5: Background Subtraction in Video Streams

Abstract

This paper explores the dynamic mode decomposition and using it to separate foreground and background elements in videos. This separation was performed on 2 different videos, one with a much clearer distinction between foreground and background and one where it is much harder to distinguish. The DMD method was able to separate the foreground and background for the first video albeit with some imperfections, however it was not able to separate the foreground and background for the second video nearly as clearly.

Introduction and Overview

The dynamic mode decomposition is another method of reducing the dimensionality of data by representing the underlying pattern in the data as a system of linear differential equations. However, by taking a low rank dynamic mode decomposition, we can split data such as the image frames of a video into the background and foreground modes. In this paper we will discuss using the dynamic mode decomposition to separate the foreground and background of 2 different videos, a video of F1 cars driving along the track and a video of a skier coming down a snowy hill.

Theoretical Background

Say that you have data that is comprised of time snapshots of some underlying behaviour, in this case these snapshots will be the frames of the video. The dynamic mode decomposition (DMD) then creates an approximation of the most effective linearly system that governs how to iterate through those snapshots of data. In other words, the DMD allows you to find a set of linear differential equations that well defines the behaviour in the data, even if the underlying behaviour was not linear to begin with. To define the DMD we first need to define the Koopman operator $\boldsymbol{A}$ where:

$$x_{j+1} = \boldsymbol{A}x_j \quad Eq.1$$

where x_j is the jth time snapshot of the initial data X. We can see that the Koopman operator is what defines the iteration of the data from one time snapshot to the next. We also need to define the matrix $\tilde{\boldsymbol{S}}$ as follows:

$$\tilde{\boldsymbol{S}} = \mathrm{U}^* X_2^M V \Sigma^{-1} \quad Eq.2$$

where X_2^M is the matrix X but without the first time snapshot and U, V, Σ are from the singular value decomposition (SVD) of X_1^M which is the matrix X but without the last time snapshot. Then the definition of the DMD is as follows:

$$X_{DMD}(t) = \Psi e^{\omega_k t} b \quad Eq.3$$

where Ψ is a matrix containing the DMD modes which are found using the eigenvectors of $\boldsymbol{A}$, b is a vector are the initial amplitudes of the mode found from the initial conditions, and ω_k is the vector of the eigenvalues of $\tilde{\boldsymbol{S}}$ converted to the continuous time form, which is done by taking the natural log of the eigenvalues and dividing them by the change in time between snapshots.

The DMD itself can also have a low rank approximation if we take only certain DMD modes in the reconstruction of X. If we define ω_p as only the continuous time eigenvalues ω that are close to 0 then we can rewrite the DMD as:

$$X_{DMD}(t) = b_p \varphi_p e^{\omega_p t} + \sum_{j \neq p} b_j \varphi_j e^{\omega_j t} \quad Eq. 4$$

Here we have a low rank DMD approximation, as well as the sparse DMD approximation with the remaining terms.

In our case, using these two DMD approximations we can separate the background and foreground from the videos by taking the low rank DMD approximation as the background of the video and the sparse approximation as the foreground.

Algorithm Implementation and Development

First, I loaded in the video, converting the frames to matrices representing the value of each pixel and then reshaping this matrix so that each column contains the values of one frame and as we go through the columns we move through the different frames of the video. Then I also removed any duplicate frames from the video matrix as they may affect the DMD later on.

I defined my t vector using the framerate of the video and the total number of frames. I defined the X_1^M and X_2^M matrices by truncating the initial data X accordingly.

I then performed the SVD on X_1^M, and used these values to create the $\tilde{\boldsymbol{S}}$ matrix as defined by equation 2. After which I found the eigenvalues and eigenvectors of $\tilde{\boldsymbol{S}}$.

I defined ω by scaling the eigenvalues so they represent continuous time as described earlier. Then I found the values of ω for which the complex magnitude was less than a certain threshold (which was adjusted for the videos) to get the continuous time eigenvalues that are close to 0 as well as their indices. For this a threshold value of 1 was used to determine if it was close enough to 0.

Then I calculated φ by multiplying the matrix U of the SVD with the eigenvectors. And then I used φ to find the initial mode amplitudes b using the pseudoinverse to to solve the linear system $X_1 = \varphi b$.

I then found the low rank approximation of the DMD using equation 4, doing the matrix multiplications while iterating through time, and only using the modes that correspond to the indices of ω where the magnitude of ω was close to 0.

I then subtracted the magnitude of the low rank DMD from the initial data to get the sparse approximation of the DMD. Rescaling and reshaping these two approximations gave me the video separated into the foreground and background.

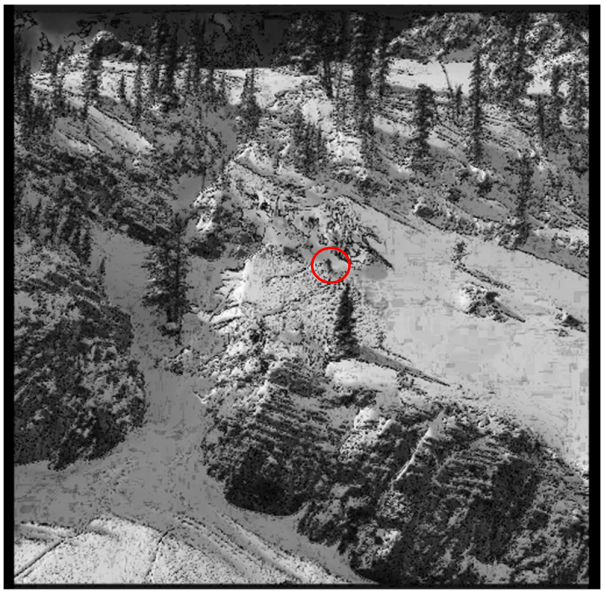
Computational Results

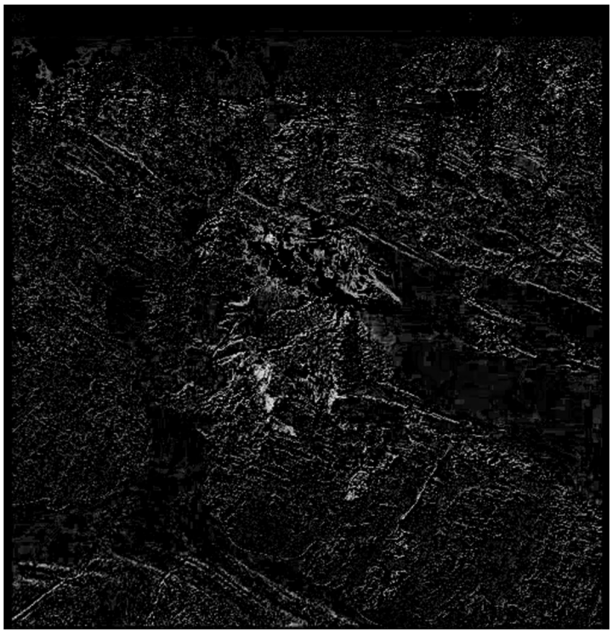
For the first video the foreground and background separation was found as follows:

The first image is a frame taken from the low rank DMD reconstruction and the second image is the frame from the same time take from the sparse DMD reconstruction.

We can see that the low rank frame contains most of the background of the frame which is good, however there are also parts of the moving F1 car which should be the foreground. What is interesting to note is that most of the car seen in the low rank frame is transparent so that part of the car has been removed, but it is around the edges that causes a shadow of the car to remain. In the sparse frame we can see that most of the background has been removed except for some areas near the car and some other parts of the video that were moving such as the flag. We can also see that the part of the car that was removed in the low rank frame is present in the sparse frame.

For the second video the foreground and background separation was found as follows:





Again, the first image is a frame taken from the low rank DMD reconstruction and the second image is the frame from the same time take from the sparse DMD reconstruction.

We can see that for the low rank frame, most of the background is present as expected. However there seems to be a lot of noise which looks to be from the edges of the shapes such as trees in the background. This noise is also present in the sparse frame meaning the separation was not as good as for the first video. Another thing to note is that the skier who is moving and was expected to be part of the foreground is actually part of the low rank frame which contains the background (circled in red), but is much more difficult to see in the sparse frame. This may be because the skier is much smaller and moving behind parts of the background at times so the DMD thought it was part of the background as well.

Summary and Conclusion

We were able to use the DMD to separate the background foreground parts of a video relatively well for the first video since the separation between background and foreground was clearer. However, for the second video, since the moving subject of the video was much harder to distinguish, the DMD was not able to separate the foreground and background very well. Furthermore, for both cases we can see that there is some noise in the separation as well as parts of the background and foreground in both low rank and sparse reconstructions meaning the separation was never perfect. Overall, using the DMD to separate the foreground and background of a video can be done and may be useful in some cases, however it is not always reliable especially when there is not such a clear distinction. And most of the time the separation of background and foreground isn't perfect, leaving imprints of other parts of the video.

Appendix A: MatLab Functions Used

- **[U,S,V] = svd(A,'econ')** – Performs a reduced singular value decomposition of matrix A
  - https://au.mathworks.com/help/matlab/ref/double.svd.html
- **[V,D] = eig(A)** – Returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors.
  - https://au.mathworks.com/help/matlab/ref/eig.html
- **Y = abs(X)** – Returns the absolute value of each element in array X.
  - https://au.mathworks.com/help/matlab/ref/abs.html
- **B = reshape(A,sz)** – Reshapes A using the size vector, sz, to define size(B).
  - https://au.mathworks.com/help/matlab/ref/reshape.html
- **C = unique(A)** – returns the same data as in A, but with no repetitions.
  - https://au.mathworks.com/help/matlab/ref/double.unique.html
- **sz = size(A)** – Returns a row vector whose elements are the lengths of the corresponding dimensions of A.
  - https://au.mathworks.com/help/matlab/ref/size.html
- **I2 = im2double(I)** – Converts the image I to double precision.
  - https://au.mathworks.com/help/matlab/ref/im2double.html
- **I = rgb2gray(RGB)** – Converts the truecolor image RGB to the grayscale image I.
  - https://au.mathworks.com/help/matlab/ref/rgb2gray.html
- **x = diag(A)** – Returns a column vector of the main diagonal elements of A.
  - https://au.mathworks.com/help/matlab/ref/diag.html
- **Y = zeros(X)** – Returns an n-by-n matrix of zeros.
  - https://au.mathworks.com/help/matlab/ref/zeros.html
- **v = VideoReader(filename) –** Creates object v to read video data from the file named filename.
  - https://au.mathworks.com/help/matlab/ref/videoreader.html
- **video = read(v)** – Reads all video frames from the file associated with v.
  - https://au.mathworks.com/help/matlab/ref/videoreader.read.html

Appendix A: MatLab Code

```
clear all; close all; clc;

%% Load and Prepare Videos
% obj = VideoReader('monte_carlo_low.mp4');
obj = VideoReader('ski_drop_low.mp4');
frameRate = obj.FrameRate;
duration = obj.Duration;
video = read(obj);

X = zeros(size(video,1),size(video,2),size(video,4));
for i = 1:size(X,3)
    X(:,:,i) = rgb2gray(im2double(video(:,:,:,i)));
end
dim = size(X);
X = reshape(X,[size(X,1)*size(X,2),size(X,3)])';
X = unique(X,'stable','rows');
X = X';
dim(3) = size(X,2);
clear obj; clear video;

%% Perform Low Rank DMD
approx = 1;
t = 0:1/frameRate:(dim(3)-1)/frameRate;
dt = t(2)-t(1);
X1 = X(:,1:end-1);
X2 = X(:,2:end);

%%
[U,Sig,V] = svd(X1,'econ');
STilde = U'*X2*V*diag(1./diag(Sig));
[eVec,eVal] = eig(STilde);
mu = diag(eVal);
clear X2; clear V; clear Sig; clear eVal; clear STilde;
omega = log(mu)/dt;
clear mu;
ind = (abs(omega) < approx);
omega = omega(ind);
phi = U*eVec;
y0 = phi\X1(:,1);
y0 = y0(ind);

u_modes = zeros(length(y0),length(t));
for j = 1:length(t)
   u_modes(:,j) = y0.*exp(omega*t(j));
end
```

```
clear mu; clear omega; clear U; clear V; clear X1; clear
X2; clear eVec;
X_lowrank = phi(:,ind)*u_modes;
clear phi; clear u_modes; clear y0;

X_sparse = X - abs(X_lowrank);
clear X;

R = X_sparse.*(X_sparse < 0);
X_lowrank = R + abs(X_lowrank);
X_sparse = X_sparse - R;
X_lowrank = reshape(X_lowrank,dim);
X_sparse = reshape(X_sparse,dim);

%% View Videos
implay(X_sparse);
```