Exploring The World: A Data-Visualization Approach

Kuan H. Chen
Chemical Engineering
Applied & Computational Mathematical Sciences

Abstract

As the human-lead revolution against the machines continues, man seeks methods that can blind computers, disabling them from searching for prey. This report discusses the method of Dynamic Mode Decomposition which machines use to detect moving objects in their frame of view. Whether the revolution is successful depends on the leaders' ability to understand the concepts presented in this document.

Keywords: DMD, PCA

Exploring The World: A Data-Visualization Approach

I. Introduction

Images carry information; videos carry a series of information. This inevitably makes the processing of videos more difficult than processing a single image. However, similar to waves, images have defining characteristics. These characteristics can be exploited and used to simplify data, making processing more efficient. This is accomplished by the Principle Component Analysis (PCA). Further, by performing PCA on a series of images, i.e., videos, the characteristics of images as a function of time is recorded. This is widely applied in the field of computer vision, where sensors on vehicles are required to detect and identify moving objects before it to prevent accidents such as crashing. Without simplifying real-time data, such tasks would be performed sluggishly, jeopardizing pedestrians. This report explores the implementation of the Dynamic Mode Decomposition (DMD), which is used to separate and focus on moving objects from its background in a video.

II. Theoretical Background

1. Dynamic Mode Decomposition:

For a system governed by an unknown equation, measurements, $\vec{x}_k$, can be taken to form a matrix used to approximate it as shown in Equation 1:

$$\mathbf{1}.\boldsymbol{X}'(k,\vec{x}) = \boldsymbol{AX}$$

Where $\boldsymbol{X} = \begin{bmatrix} | & | & ... & | \\ \vec{x}_1 & \vec{x}_2 & \cdots & \vec{x}_{k-1} \\ | & | & ... & | \end{bmatrix}$, $\boldsymbol{X'} = \begin{bmatrix} | & | & ... & | \\ \vec{x}_2 & \vec{x}_3 & \cdots & \vec{x}_k \\ | & | & ... & | \end{bmatrix}$, and k is some point in time. $\boldsymbol{A}$, therefore, is the transformation that approximately relates one state in time to a future state in time. In this context, $\vec{x}_k$ represents an image in a video. Assuming that Equation 1 is a linear equation, the solution can be written as in Equation 2 as follows:

$$\mathbf{2}.\boldsymbol{X} = \sum_{j=1}^{n} b_j \phi_j e^{\lambda_j t}$$

Where λ are the eigenvalues of $\boldsymbol{A}$, ϕ the DMD modes of $\boldsymbol{A}$, and $\boldsymbol{b}$ is a constant defined by initial conditions. $\boldsymbol{A}$, on the other hand, can be found as shown in Equation 3 as follows:

$$\mathbf{3}.\boldsymbol{A} = \boldsymbol{X'X}^*$$

Where $\boldsymbol{X}^*$ is the pseudo-inverse of $\boldsymbol{X}$. Often, the length of $\boldsymbol{X}$, as well as $\boldsymbol{X}^*$ are both very large, which would cause the computation time for $\boldsymbol{A}$ to be unnecessarily long. Thus, the Principle Component Analysis of $\boldsymbol{X}$ is taken first, as explained in the following section.

2. Principle Component Analysis:

The PCA is used to identify the dominant traits in a wave, in this context, images. It allows one to simplify a set of data and only retains its key features. Suppose matrix $\boldsymbol{X}$ contains all images of a given video, and that $\boldsymbol{U}$ contains all unitary vectors that form a basis of $\boldsymbol{X}$ (all vectors u_i are **linearly independent**). $\boldsymbol{U}^*\boldsymbol{X}$ is a low-rank projection of $\boldsymbol{X}$ onto the basis $\boldsymbol{U}$ containing a set of components that describe $\boldsymbol{X}$. Since any matrix of data can be decomposed into a collection of component matrices $\boldsymbol{U\Sigma V^T}$, $\boldsymbol{U}$ can be found for $\boldsymbol{X}$. This decomposition is called the Singular-Value Decomposition (SVD) and is shown in Equation 4,

$$\mathbf{4}.\boldsymbol{X} = \boldsymbol{U\Sigma V^T} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} u_i \sigma_i v_i^T$$

where σ_i represents the standard deviation in the i^{th} dimension. High σ_i suggests high variance, or ${\sigma_i}^2$, in the i^{th} dimension, which in turn represents a strong presence of component u_i in $\boldsymbol{X}$. The u_i that corresponds to the highest ${\sigma_i}^2$ is denoted as u_1 and is termed the principle component. The set of u_i that corresponds to high σ_i can be used to approximate $\boldsymbol{X}$ and is termed the **low-rank approximation** of $\boldsymbol{X}$. Equation 5 can be used as a measure to determine whether an appropriate low-rank approximation is found.

$$\mathbf{5}.r = \sum_{i}^{m} {\sigma_i}^2 \Big/ \sum_{i}^{n} {\sigma_i}^2 , m < n$$

When the variation in r is little as i increases, the corresponding matrix $\boldsymbol{Y}$ is the low-rank approximation. This simplifies the $\boldsymbol{X}$ so that an appropriate approximation of $\boldsymbol{A}$ can be found by Equation 3.

III. Algorithm Implementation and Development

Figure 1 shows an image from a video that is used for analysis in this report, the data of which can be represented as a 480-by-640 matrix.

Figure 1. Sample image from video

In order to separate the moving object from the rest of the image, a low-rank approximation of all images needs to be found. This approximation will retain only the background of each image in the video, since the background does not change and thus provides the dominant signals in the matrix $\boldsymbol{X}$. The moving object can be found by subtracting each of the frames in $\boldsymbol{X}$ by their corresponding low-rank approximations.

A series of images similar to the one in Figure 1 is first reshaped into a vector and concatenated with each other to form $\boldsymbol{X'}$ and $\boldsymbol{X}$. $\boldsymbol{X'}$ can then be taken the SVD of to find the $\boldsymbol{U\Sigma V^T}$ of the system. Figure 2 shows PCA results for $\boldsymbol{X'}$.

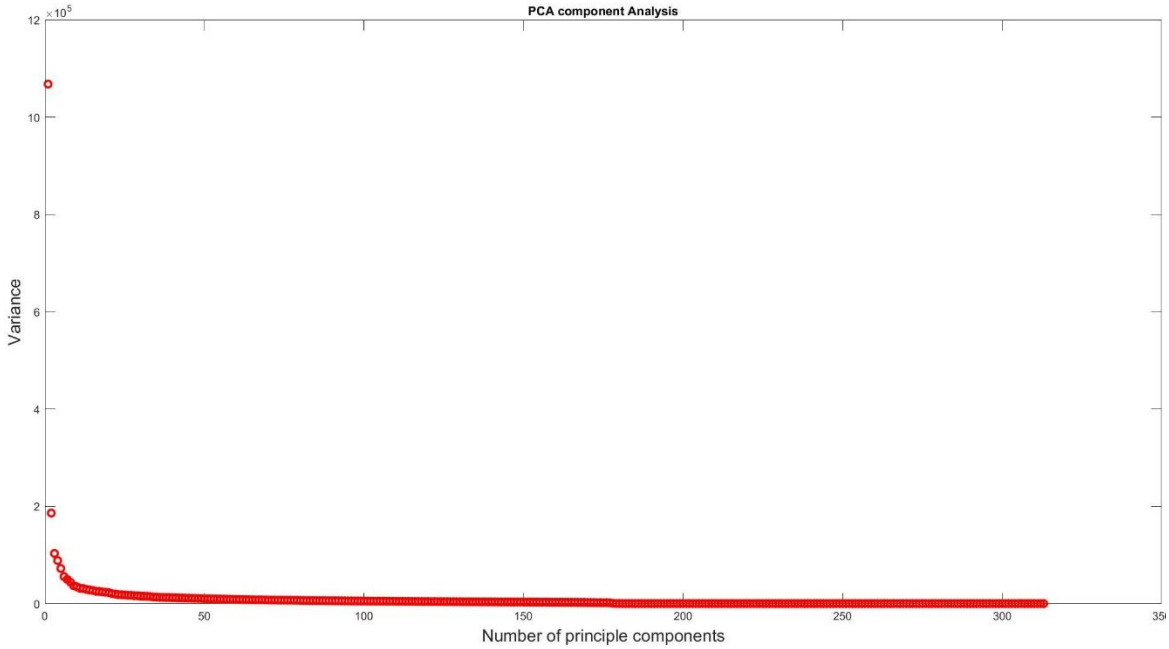

Figure 2. PCA component analysis of collection of images

PCA shows that a rank-2 approximation for these images is sufficient in retaining key elements of the data. The following MATLAB commands are used to construct $\boldsymbol{X'}$ and $\boldsymbol{X}$ from the video and take the SVD of $\boldsymbol{X'}$:

```
[yFrame, xFrame, rgbFrame, tFrame] = size(vidFrames);
numFrames = size(vidFrames,4);
reshapedImages = [];
for i = 1:tFrame
    vidFramesGray = double(rgb2gray((vidFrames(:,:,:,i))));

    framesColumn = reshape(vidFramesGray, [numel(vidFramesGray), 1]);
    reshapedImages = [reshapedImages, framesColumn];
end
X = reshapedImages;
X1 = X(:,1:end-1);
X2 = X(:,2:end);
[U2,Sigma2,V2] = svd(X1, 'econ');
```

The rank-2 approximation of $\boldsymbol{X'}$ and $\boldsymbol{X}$ can then be used to find the rank-2 approximation of $\boldsymbol{A}$, $\boldsymbol{\tilde{A}}$, which is found from the coupling Equations 6 and 7:

$$\boldsymbol{6.\ A = X'X^* = X'V\Sigma^{-1}U^*}$$

$$\boldsymbol{7.\ \tilde{A} = U^*AU = U^*X'V\Sigma^{-1}}$$

To solve for Equation 2, the parameters b, ϕ, and λ need to be found. Equations 8 – 10 define these parameters in relation to $\boldsymbol{\tilde{A}}$:

$$\boldsymbol{8.\ \tilde{A}W = W\lambda}$$

$$\boldsymbol{9.\ \phi = X'V\Sigma^{-1}W}$$

$$\boldsymbol{10.\ X'(0) = \phi b}$$

Where $\boldsymbol{W}$ is the set of eigenvectors of $\boldsymbol{\tilde{A}}$. These steps are summarized in the following MATLAB commands:

```
r = 2;
U=U2(:,1:r);
Sigma=Sigma2(1:r,1:r);
V=V2(:,1:r);
Atilde = U'*X2*V/Sigma;
[W,D] = eig(Atilde);
Phi=X2*V/Sigma*W; % DMD modes
x1 = X(:,1); % initial condition
b = Phi\x1;
```

Equations 6 – 10 thus solve for the rank-2 approximation of $\boldsymbol{X}$ as defined in Equation 2, where each column of $\boldsymbol{X}$ corresponds to the simplified data in each frame. The following section discusses the effectiveness of this method.

IV. Computational Results

Figure 3 shows an image of the moving object being separated from its background.

Figure 3. (a) The background (b) The moving object (foreground)

As can be seen, this separation is very good. Although the silhouette of various objects from the original image is retained, the moving object (paint bucket) is fully retained with much detail of it able to be seen. However, the observant viewer may point out that this separation may only be good because of the apparent displacement of the bucket. High degrees of displacement guarantees that the object does not appear at the same position in every frame, allowing the algorithm to clearly distinguish between the cat and the background. A different video was analyzed to see how the DMD algorithm was able to separate a cat from its background in a video where the cat was lying down, playing around with its limbs. The results are presented in Figure 4.

Figure 4. (a) A cute cat playing in bed (b) The background (c) The moving cat (foreground)

As can be seen, although the silhouette of the cat can be seen, as well as the stripes across its body, the separation in this case is not as clear as it was in the former. There are two possible explanations to this – the cat has a dark coat, and in RGB scale, it perhaps has a similar signal amplitude as the red blankets in the background does, complicating separation. another possible explanation is that since the torso of the cat remains at the same position throughout the video, the DMD algorithm might mistake it as part of the background. This explanation seems to hold more ground since the movement of the limbs can be seen throughout the video.

V. Summary and Conclusion

The DMD algorithm is a powerful tool that can separate moving objects from its background. As seen in Figure 3 and 4, regardless of the moving objects' distance from the camera or proximity to their backgrounds, separation is still achieved to a certain extent. Additionally, the algorithm is easy to implement, allowing any video to be analyzed. Most importantly, the algorithm analyzes data quickly; the step that took the longest to complete was the SVD step. However, it should be noted that, the processing speed is highly dependent on the size and length of the video, and that the videos used in this report were all fairly short (less than 30 seconds in length). The performance of this algorithm on longer videos was not explored in this report, but is expected to take significantly more time to process than those used in this report. Looks like the human-lead revolution will have a chance at blinding the machines as long as they find a way to slow down their processors or blend into the background!

Appendix A

- `reshape` – converts a matrix or vector into a matrix of designated width and length.
- `rgb2gray(vidFrames(:,:,:,i))` – removes color dimension from `vidFrames`
- `svd(X1, 'econ')` – Takes SVD of matrix `X1`
- `[W,D] = eig(Atilde)`—Finds the eigenvalues and eigenvectors of `Atilde`

Appendix B

```
%%
clear all; close all; clc
v = VideoReader('short cat video.mp4');
duration = v.Duration;
vidFrames = read(v);


[yFrame, xFrame, rgbFrame, tFrame] = size(vidFrames);
numFrames = size(vidFrames,4);
reshapedImages = [];
for i = 1:tFrame
    vidFramesGray = double(rgb2gray((vidFrames(:,:,:,i))));

    framesColumn = reshape(vidFramesGray, [numel(vidFramesGray), 1]);
    reshapedImages = [reshapedImages,
framesColumn];                                  %#ok<AGROW>
end
X = reshapedImages;
X1 = X(:,1:end-1);
X2 = X(:,2:end);
[U2,Sigma2,V2] = svd(X1, 'econ');

dt = 1;


r = 2;
U=U2(:,1:r);
Sigma=Sigma2(1:r,1:r);
V=V2(:,1:r);
Atilde = U'*X2*V/Sigma;

[W,D] = eig(Atilde);
Phi=X2*V/Sigma*W; % DMD modes
x1 = X(:,1); % initial condition
b = Phi\x1;

mu=diag(D);
omega=log(mu)/dt;


timeDynamics = zeros(r,tFrame);
for iter = 1:tFrame
    timeDynamics(:,iter) = (b.*exp(omega*iter));
end
xDMD = Phi*timeDynamics;
%%
for iter=1:size(xDMD,2)

    dmdFrame = reshape(xDMD(:,iter), [yFrame xFrame]);
    figure(1)
    subplot(2,3,[1,4])
    imshow(vidFrames(:,:,:,iter))
    title('(a) Original Image')
    subplot(2,3,[2,5])
    imshow(abs(dmdFrame)/max(abs(dmdFrame(:))));
```

```
    title('(b) Separated Background')

    foreGround = reshape(X(:,iter),[yFrame, xFrame]) - dmdFrame;
    foreFrame = reshape(foreGround, [yFrame, xFrame]);

    subplot(2,3,[3 6])
    imshow(abs(foreFrame)/max(abs(foreFrame(:))))
    title('(c) Separated Foreground')

%     figure(3)
%     imshow(vidFrames(:,:,:,iter))
end
```