AMATH 582 Homework 4

Jesse Dumas

September 14, 2018

This is a discussion on the dynamic mode decomposition (DMD) algorithm and its use in separating video streams into individual streams of foreground videos and background videos. Stop motion videos of LEGO minifigures were created to test the algorithm. The method successfully subtracted the background from the test videos, and the shortcomings of DMD are discussed.

Introduction and Overview

As this course has demonstrated, singular value decomposition[1] (SVD) is a useful method for reducing high dimensional data to low rank data by finding the most important dimensions for use as basis vectors. However, the usefulness of SVD is limited to static data, and it fails to capture any dynamic structure in data that evolves with time. Dynamic mode decomposition extends the power of SVD to data from dynamic systems[2]. An excellent testbed for DMD is subtracting the background from videos of moving objects, particularly because the time dynamic (frame rate, in this case) is known ahead of time[3].

This exploration will investigate the subtraction of backgrounds from vidoes of moving objects using the DMD method on four test cases:

- **(test 1)** A video of LEGO minifigures walking past the camera in different directions.
- **(test 2)** A video of a LEGO vehicle moving through the picture.
- **(test 3)** One LEGO minifigure being hit by an oncoming LEGO vehicle.
- **test 4** LEGO minifigures moving in the same direction at different speeds.

Despite successfully separating the background and foreground video streams, the DMD method had two small shortcommings in this experiment. The algorithm struggled with objects in the videos that were moving toward the camera. These objects would be registered as background although they moved in time. Additionally, as objects moved out of the frame, "ghost" objects would appear. Both of these weaknesses are discussed further in the computational results section.

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013

[2] J.N. Kutz. Chapter 1: Dynamic mode decomposition: An introduction

[3] Jacob Grosek and J. Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *CoRR*, abs/1404.7592, 2014

Theoretical Background

SVD

Before time dynamics can be extracted from the data, SVD needs to be performed to understand the low rank structure of the system. If **A** is a data matrix, like a collection of positions evolving in time, SVD will decompose it as

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^* \tag{1}$$

which can also be expressed as

$$[\mathbf{A}] = \begin{bmatrix} & & & \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ & & & \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} . \tag{2}$$

In equations 1 and 2, Σ is a diagonal matrix of eigenvectors comprising the singular values of **A**. The columns in **U** are orthonormal basis vectors for the space and **V** describes how each vector is projected onto the principal components in **U**[4,5]. Knowing the rank of the data

[4] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013

[5] J Dumas. Amath 582 hw1, January 2017

Dynamic Mode Decomposition

In simple terms, DMD takes data from a dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t, \mu) , \tag{3}$$

where **f** represents the dynamics of the system based on the state, $\mathbf{x}(t)$, evolving in time, t, with parameters, μ[6], and constructs an approximate locally linear system

[6] J.N. Kutz. Chapter 1: Dynamic mode decomposition: An introduction

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} . \tag{4}$$

With initial condition $\mathbf{x}(0)$, the solution of Equation 4 is

$$\mathbf{x}(t) = \sum_{k=1}^{n} \phi_k exp(\omega_k t) b_k = \Phi exp(\Omega t) b , \tag{5}$$

where ϕ_k and ω_k are eigenvectors and eigenvalues of **A**. To speed the computation, $\tilde{\mathbf{A}}$, a projection of **A** onto the dominant POD modes, is used. Figure 1 illustrates the DMD procedure for an example fluid dynamics problem.

Algorithm Implementation and Development

Test Videos

The test videos were created using Stop Motion Studio for Android, version 3.0.0.3121 by Cateater. To make background subtraction sim-

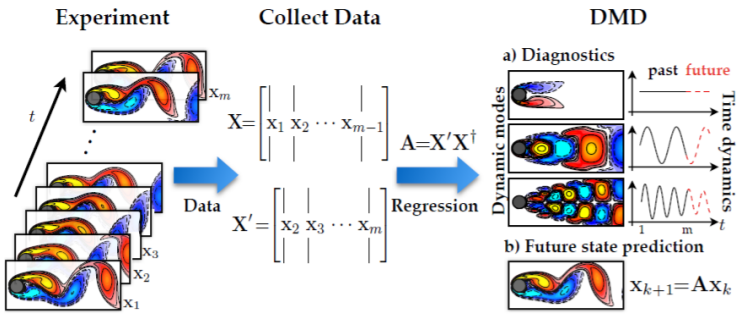

Figure 1: A schematic example of DMD on a fluid problem from Dynamic Mode Decomposition: An Introduction.

pler, extra care was taken to provide ample light in the video, and the camera was not moved during the recording. Poor lighting or a moving camera would have added significant noise to the data.

The Algorithm

1. Import video clips. Get time duration and number of frames.
2. Convert video to grayscale.
3. Build matrix $\mathbf{X}$ with each column being a frame from the video at a different time t.
4. Take the SVD of $\mathbf{X}$. $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^*$.
5. Compute the matrix $\mathbf{A}$ from the pseudo-inverse of $\mathbf{X}$.
6. Compute the matrix $\tilde{\mathbf{A}}$, the $r \times r$ projection of $\mathbf{A}$ onto POD modes.
7. Compute the eigendecomposition, $\mathbf{W}$, of $\tilde{\mathbf{A}}$.
8. Reconstruct the eigendecomposition of $\mathbf{A}$ from $\mathbf{W}$ and Λ.
9. Subtract the background video from the original video to obtain the foreground video of moving objects.
10. Reshape background and foreground videos and visualize.

The full code for the project is presented in Appendix B.

Computational Results

Background video streams were sucessfully subtracted from the test videos in all cases. However, in situations where the objects in the video were moving toward the camera, the objects were not removed. This happened in case 2. In cases where objects left the frame, "ghost"

objects were created by the algorithm. These were likely the result of the predictive aspect of DMD, meaning that the DMD algorithm was predicting where the objects would be next and created faint objects where it expected them. In all cases, the dynamic data could be projected onto a rank 1 or rank 2 space. Dominant singular values of the system for case 1 are shown in Figure 2. Appendix C contains additional figures to save space in the actual report.

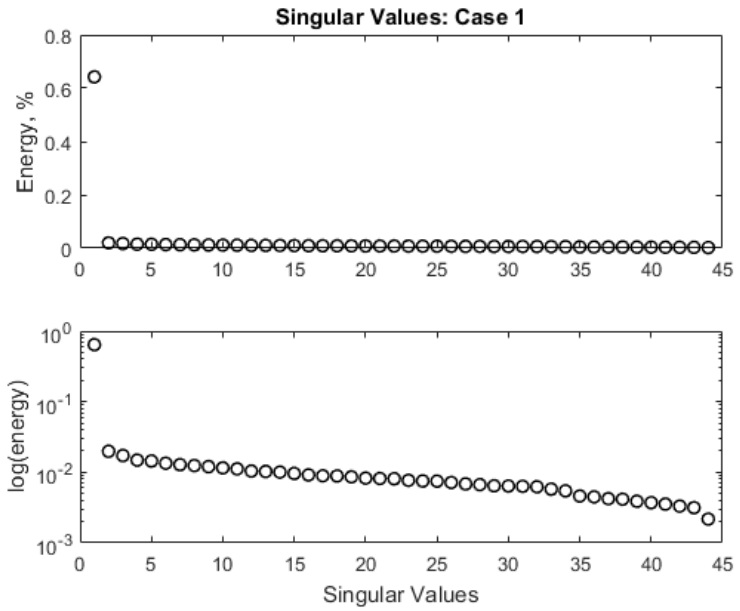

Figure 2: Singular values for Case 1. There is one dominant mode, indicating that the data can be projected to a very low rank space.

Case 1

The video in test case 1 featured LEGO minifigures crossing the frame in different directions, each moving in a different plane. Figures moving perpendicular to the camera were easy to subtract as foreground elements, but one minifigure that was moving toward the camera was mistaken for background in early frames of the video. This is clear in the fourth frame of Figure 3. Once the minifigure started moving in a perpendicular direction, it was no longer mistaken for background. This problem is likely because the minifigure's size is changing as it moves toward the camera, but its position is not, and position is a much more dominant POD mode than size for this system.

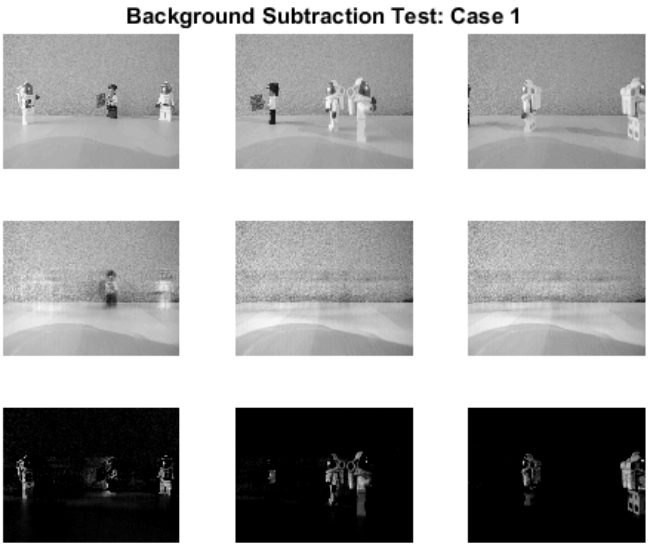

Figure 3: Top: Frames 2, 20, and 30 of the original video stream. Middle: Frames 2, 20, and 30 of the background video stream. Bottom: Frames 2, 20, and 30 of the foreground video stream

Case 2

Test case 2 was a video of a LEGO vehicle crossing the frame, illustrated in Figure 7. Compared to case 1 where only one POD mode was dominant, case 2 exhibits a slightly larger second POD mode, shown in Figure 4. However, this second mode is still much smaller than the first. Despite teh vehicle filling more of the frame than the minifiures, the algorithm was still able to subtract the background without mistaking the large vehicle for background.

Case 3

In case 3, a rightward moving minifigure is hit by a leftward moving vehicle, at which point both objects become leftward moving. The POD modes look similar in trend and magnitude to case 2. See Figure 5. Again, DMD was able to separate the background and foreground video streams. Some "ghost" objects are visible when objects enter an empty frame and when the objects leave the frame.

Case 4

The video in case 4 features four minifigures, all moving rightward but at different speeds. The rightmost miinifigure moved every frame, the second rightmost minifigure moved every other frame, the third rightmost minifigure moved every third frame, and the leftmost minifigure moved every fourth frame. Though the algorithm

was able to separate the background and foreground video streams, substantial "ghosting" occurred as a result of the non-constant motion of three of the minifigures. This phenomenon is illustrated in Figure 9.

Summary and Conclusions

Building a DMD algorithm to separate background and foreground video streams from one source was successful. It is clear that objects with any stop-and-go motion will occasionally be mistaken for background. The issue of "ghost" objects is an area for future exploration. Perhaps it can be mitigated by tuning a parameter related to the predictve aspect of DMD.

References

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013.

[2] J.N. Kutz. Chapter 1: Dynamic mode decomposition: An introduction.

[3] Jacob Grosek and J. Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video. *CoRR*, abs/1404.7592, 2014.

[4] J Dumas. Amath 582 hw1, January 2017.

Appendix A: MATLAB Functions

```
double() -- converts data type to double precision floating point.
load() -- loads saved data.
size() -- returns dimensions of a matrix.
[U,S,V] = svd(A); -- performs SVD.
my_dmd -- returns the background stream from a video via DMD.
vidfunc -- processes a video and performs DMD by calling my_dmd.
```

Appendix B: MATLAB Code

The following script calls $my_dmd.mandvidfunc.mtocompletethehomeworkandbuildplots$.

```
s1_1,4);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Homework 4 for AMATH 582
% Jesse Dumas
```

```
% Background Subtraction via DMD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all; clear all; clc

% video 1
% Separate backround and foreground
[fg, bg, og, fg_reshaped, bg_reshaped, S] = vidfunc('lego1.mp4');

% plot SVD modes
sig = diag(S)/sum(diag(S));

figure(1);
subplot(2,1,1), plot(sig,'ko','Linewidth',[1.1])
title('Singular Values: Case 1')
ylabel('Energy, %')
subplot(2,1,2), semilogy(sig,'ko','Linewidth',[1.1])
ylabel('log(energy)')
xlabel('Singular Values')

% plot original, backgroun, and foreground at frames 2, 20, and 30
figure(2);
subplot(3,3,1), imagesc(rgb2gray(og(:,:,:,2))), colormap(gray), axis off
subplot(3,3,2), imagesc(rgb2gray(og(:,:,:,20))), colormap(gray), axis off
title('Background Subtraction Test: Case 1')
subplot(3,3,3), imagesc(rgb2gray(og(:,:,:,30))), colormap(gray), axis off

subplot(3,3,4), imagesc(bg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,5), imagesc(bg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,6), imagesc(bg_reshaped(:,:,:,30)), colormap(gray), axis off

subplot(3,3,7), imagesc(fg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,8), imagesc(fg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,9), imagesc(fg_reshaped(:,:,:,30)), colormap(gray), axis off

% video 2
% Separate backround and foreground
[fg, bg, og, fg_reshaped, bg_reshaped, S] = vidfunc('lego2.mp4');

% plot SVD modes
sig = diag(S)/sum(diag(S));

figure(3);
subplot(2,1,1), plot(sig,'ko','Linewidth',[1.1])
title('Singular Values: Case 2')
```

```
ylabel('Energy, %')
subplot(2,1,2), semilogy(sig,'ko','Linewidth',[1.1])
ylabel('log(energy)')
xlabel('Singular Values')

% plot original, backgroun, and foreground at frames 2, 20, and 30
figure(4);
subplot(3,3,1), imagesc(rgb2gray(og(:,:,:,2))), colormap(gray), axis off
subplot(3,3,2), imagesc(rgb2gray(og(:,:,:,20))), colormap(gray), axis off
title('Background Subtraction Test: Case 2')
subplot(3,3,3), imagesc(rgb2gray(og(:,:,:,30))), colormap(gray), axis off

subplot(3,3,4), imagesc(bg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,5), imagesc(bg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,6), imagesc(bg_reshaped(:,:,:,30)), colormap(gray), axis off

subplot(3,3,7), imagesc(fg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,8), imagesc(fg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,9), imagesc(fg_reshaped(:,:,:,30)), colormap(gray), axis off

% video 3
% Separate backround and foreground
[fg, bg, og, fg_reshaped, bg_reshaped, S] = vidfunc('lego3.mp4');

% plot SVD modes
sig = diag(S)/sum(diag(S));

figure(5);
subplot(2,1,1), plot(sig,'ko','Linewidth',[1.1])
title('Singular Values: Case 3')
ylabel('Energy, %')
subplot(2,1,2), semilogy(sig,'ko','Linewidth',[1.1])
ylabel('log(energy)')
xlabel('Singular Values')

% plot original, backgroun, and foreground at frames 2, 20, and 30
figure(6);
subplot(3,3,1), imagesc(rgb2gray(og(:,:,:,2))), colormap(gray), axis off
subplot(3,3,2), imagesc(rgb2gray(og(:,:,:,20))), colormap(gray), axis off
title('Background Subtraction Test: Case 3')
subplot(3,3,3), imagesc(rgb2gray(og(:,:,:,30))), colormap(gray), axis off

subplot(3,3,4), imagesc(bg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,5), imagesc(bg_reshaped(:,:,:,20)), colormap(gray), axis off
```

```
subplot(3,3,6), imagesc(bg_reshaped(:,:,:,30)), colormap(gray), axis off

subplot(3,3,7), imagesc(fg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,8), imagesc(fg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,9), imagesc(fg_reshaped(:,:,:,30)), colormap(gray), axis off

% video 4
% Separate backround and foreground
[fg, bg, og, fg_reshaped, bg_reshaped, S] = vidfunc('lego4.mp4');

% plot SVD modes
sig = diag(S)/sum(diag(S));

figure(7);
subplot(2,1,1), plot(sig,'ko','Linewidth',[1.1])
title('Singular Values: Case 4')
ylabel('Energy, %')
subplot(2,1,2), semilogy(sig,'ko','Linewidth',[1.1])
ylabel('log(energy)')
xlabel('Singular Values')

% plot original, background, and foreground at frames 2, 20, and 30
figure(8);
subplot(3,3,1), imagesc(rgb2gray(og(:,:,:,2))), colormap(gray), axis off
subplot(3,3,2), imagesc(rgb2gray(og(:,:,:,20))), colormap(gray), axis off
title('Background Subtraction Test: Case 4')
subplot(3,3,3), imagesc(rgb2gray(og(:,:,:,30))), colormap(gray), axis off

subplot(3,3,4), imagesc(bg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,5), imagesc(bg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,6), imagesc(bg_reshaped(:,:,:,30)), colormap(gray), axis off

subplot(3,3,7), imagesc(fg_reshaped(:,:,:,2)), colormap(gray), axis off
subplot(3,3,8), imagesc(fg_reshaped(:,:,:,20)), colormap(gray), axis off
subplot(3,3,9), imagesc(fg_reshaped(:,:,:,30)), colormap(gray), axis off



This is function that performs DMD. It's from Kutz's DMD
notes.


function [Phi,omega,lambda,b,Xdmd,S] = my_dmd(X1,X2,r,dt)
% Function copied from Kutz DMD Notes
% Computes the DMD of X1, X2
```

```
% DMD
[U, S, V] = svd(X1, 'econ');
r = min(r, size(U,2));

% Truncate to low rank
U_r = U(:, 1:r);
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);
A_tilde = U_r' * X2 * V_r / S_r;
[W_r, D] = eig(A_tilde);
Phi = X2 * V_r / S_r * W_r;

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

% Compute DMD mode amplitudes
x1 =X1(:, 1);
b = Phi\x1;

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

This is the video processing function.

```
function [fg, bg, og, fg_reshaped, bg_reshaped,S] = vidfunc(video)
% vidfunc processes a video file and performs DMD
% on it to subtract the foreground video

% video input
v1 = VideoReader(video);
video1 = read(v1);
numFrames = get(v1,'numberOfFrames');
duration = get(v1,'Duration');

% convert video to a matrix of grayscale
% images where each column is a frame
```

```
for j=1:numFrames
    vid_mat2 = rgb2gray(video1(:,:,:,j));
    vid_reshaped = reshape(vid_mat2, [], 1);
    vid_mat(:,j) = double(vid_reshaped);
end

% DMD
X = vid_mat;
x1 = X(:,1:end-1);
x2 = X(:,2:end);
r=2;
dt=duration/numFrames;
[Phi,omega,lambda,b,Xdmd,S] = my_dmd(x1,x2,r,dt);

% outputs
og = video1;
bg = uint8(Xdmd);
fg = uint8(X(:,1:numFrames-1) - Xdmd);

% reshaped videos
for jj=1:numFrames-1
    fg_reshaped(:,:,:,jj) = reshape(fg(:,jj), 360, 640);
end

for ii=1:numFrames-1
    bg_reshaped(:,:,:,ii) = reshape(bg(:,ii), 360, 640);
end

fg_reshaped = fg_reshaped;
bg_reshaped = bg_reshaped;
```

Appendix C: Additional Figures

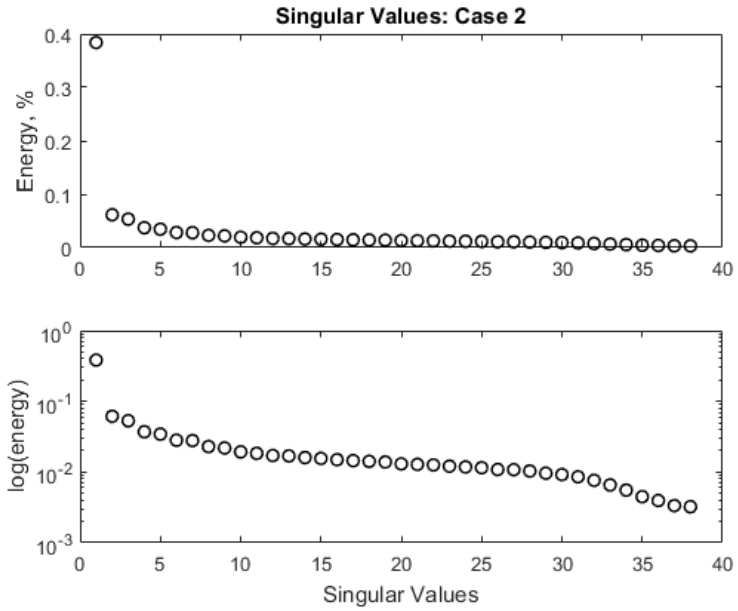

Figure 4: Singular values for Case 2. There is one dominant mode, indicating that the data can be projected to a very low rank space.

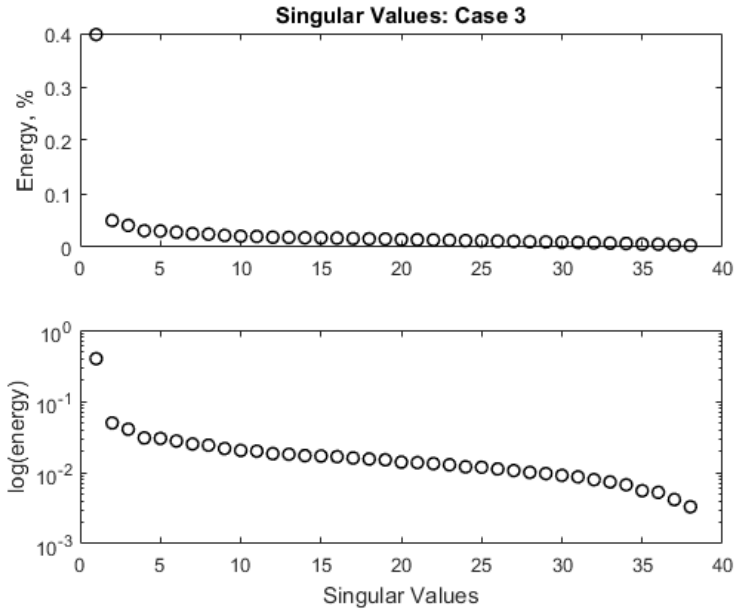

Figure 5: Singular values for Case 3. There is one dominant mode, indicating that the data can be projected to a very low rank space.

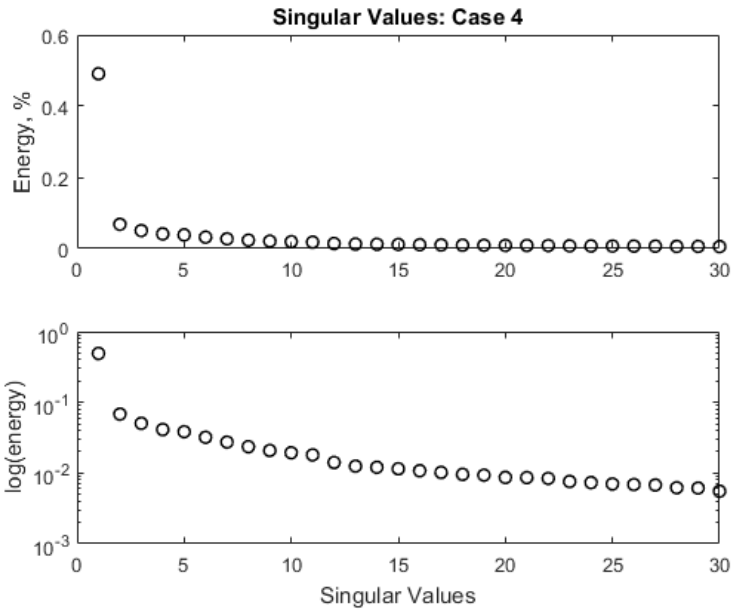

Figure 6: Singular values for Case 4. There is one dominant mode, indicating that the data can be projected to a very low rank space.

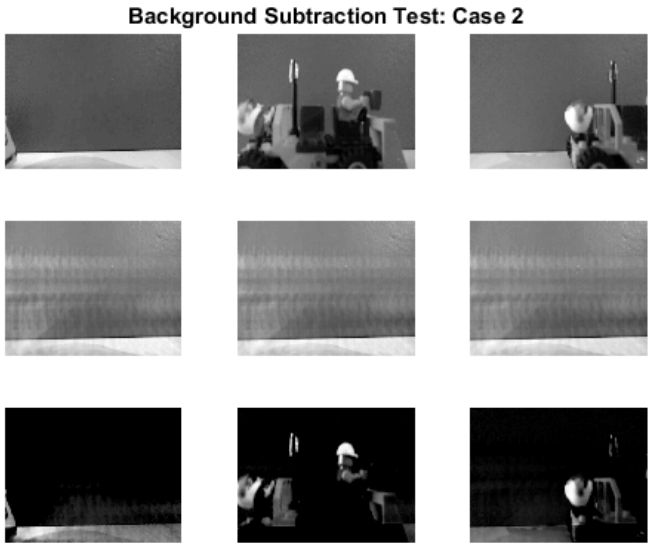

Figure 7: Case 2. Top: Frames 2, 20, and 30 of the original video stream. Middle: Frames 2, 20, and 30 of the background video stream. Bottom: Frames 2, 20, and 30 of the foreground video stream

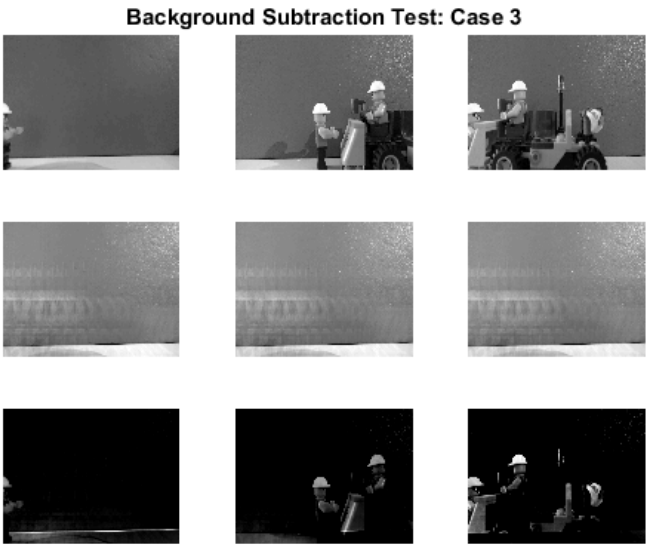

Figure 8: Case 3. Top: Frames 2, 20, and 30 of the original video stream. Middle: Frames 2, 20, and 30 of the background video stream. Bottom: Frames 2, 20, and 30 of the foreground video stream

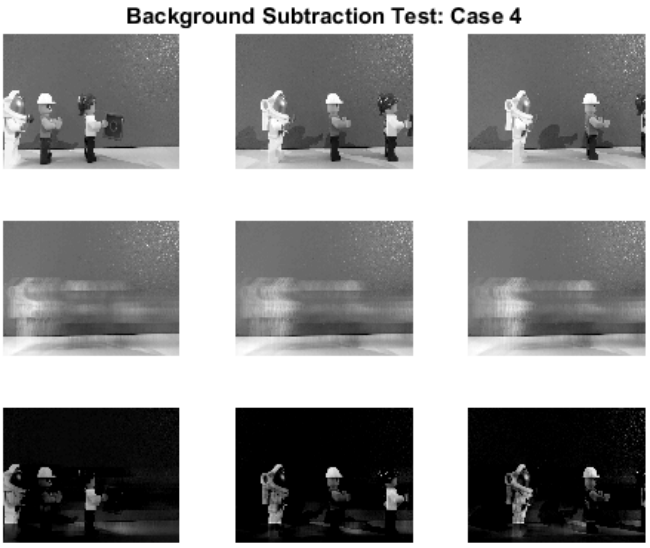

Figure 9: Case 4. Top: Frames 2, 20, and 30 of the original video stream. Middle: Frames 2, 20, and 30 of the background video stream. Bottom: Frames 2, 20, and 30 of the foreground video stream