Singular Value Decomposition

University of Washington
AMATH 582 Homework 3
Winter Quarter 2019

David Long
dmlong888@live.com

Abstract

This paper explores the Singular Value Decomposition, which is one of the most powerful tools available in multivariate analysis. Data streams from three cameras record the motion of a mass attached to a spring, which is subjected to various dynamics of motion. Through the SVD, redundancies and noise in the data are removed and the primary sources of variance in the data are identified.

I. Introduction and Overview

Recent exponential advancements in the field of computer science have drastically reduced the cost of data. Large corporations hoard it with reckless abandon and worry about what to do with it later. The sheer volume of data that can be collected about an experiment is daunting. It is often rife with redundancies and noise and difficult to parse. Which parts of the data are singular and meaningful and which parts are redundant or insignificant? These questions drive the emerging science of big data, as a descriptive and meaningful interpretation of the data can be used to make big money.

Mathematical approaches to parsing large data sets have arisen in applications in the fields of biology, economics, marketing, signal processing, and atmospheric sciences. The processes of *principal component analysis, proper mode decomposition, Hotelling transform, empirical orthogonal functions, reduced order modeling, dimensionality reduction,* and *singular value decomposition* all endeavor to reduce a data set into orthogonal directions that account for as large an amount of the variability in as few directions as possible. The processes are all essentially the same, but have different names as they were arrived at in different disciplines.[1] In this paper, the process of dimension reduction will be referred to as Singular Value Decomposition (SVD).

To demonstrate the process, video data was collected documenting the motion of a paint can that was attached to a spring and set in motion. The data are noisy and redundant, but the SVD will simplify it by projecting it onto a different basis. The process identifies directions of significant variability and orients the data set in those directions. It is usually the case that this process reduces the number of dimensions that are necessary to adequately represent the data.

II. Theoretical Background

To understand the Singular Value Decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$, it is instructive to begin with an interpretation of how a matrix transforms a vector $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle^T \in \mathbb{R}^{n\times 1}$ in a vector space. The product $\vec{y} = A\vec{x} \in \mathbb{R}^{m\times 1}$ is a new vector that is simply a transformation of $\vec{x}$. Depending on the coefficients of *A*, the transformation is a combination of rotation and dilation or compression.

Naturally the same transformation can be affected on a set of vectors $V = \left[\vec{v}_1\ \vec{v}_2 \cdots \vec{v}_n\right]$. The product $A\vec{v}_i = \sigma_i \vec{u}_i$ then represents the transformation of $\vec{v}_i$ onto a unit vector $\vec{u}_i \in \mathbb{R}^{m\times 1}$, where σ_i is a scalar representing the magnitude of the transformed vector. In matrix notation, the simultaneous equations can be expressed as

$$AV = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} \vec{v}_1 \, \vec{v}_2 \cdots \vec{v}_n \end{bmatrix} = \begin{bmatrix} \vec{u}_1 \, \vec{u}_2 \cdots \vec{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} = \hat{U}\hat{\Sigma} \quad \textbf{(Eqn. 1)}$$

where all other elements of the matrix $\hat{\Sigma}$ are zero. Essentially, the space that is spanned by the vector space *V* is rotated and dilated to give a new space that is spanned by $\hat{U}\hat{\Sigma}$.

In the context of SVD, the following assumptions are made[1]:

1. The matrix *A* is unitary, i.e. $A^{-1} = \bar{A}^*$, where $\bar{A}^*$ is the transpose of the complex conjugate of *A* and the columns of *A* are orthogonal.
2. *V* is unitary, i.e. $V^{-1} = \bar{V}^*$, where $\bar{V}^*$ is the transpose of the complex conjugate of *V* and the columns of *V* are orthogonal.
3. $\hat{\Sigma}$ is an $n \times n$ diagonal matrix with positive entries when *A* is of full rank, i.e. it has *n* linearly independent columns.
4. $\hat{U}$ is $m \times n$ with orthonormal columns.

Under these assumptions, Eqn. 1 can be solved for *A* as follows:

$$\begin{aligned} AV &= \hat{U}\hat{\Sigma} \\ AVV^* &= \hat{U}\hat{\Sigma}V^* \\ AI &= \hat{U}\hat{\Sigma}V^* \\ A &= \hat{U}\hat{\Sigma}V^* \end{aligned} \qquad \textbf{(Eqn. 2)}$$

The factorization of *A* in Eqn. 2 is known as the *reduced singular value decomposition* of the matrix *A*. If *A* is rank deficient, an additional $m - n$ orthonormal columns are appended to $\hat{U}$ to form the matrix U and an additional $m - n$ rows of zeros are appended to $\hat{\Sigma}$ to form the matrix Σ. The matrix Σ will now have a number of positive diagonal entries equal to the rank of *A* with the remainder of the entries on the diagonal being equal to zero. The *full singular value decomposition* is now given as

$$A = U\Sigma V^* \qquad \textbf{(Eqn. 3)}$$

Note that $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are unitary and $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal. The entries of Σ are ordered from greatest to least. Geometrically, the interpretation of the decomposition is that V^* affects a unitary transformation of the unit sphere in $\mathbb{R}^{n \times 1}$, Σ stretches it to create a hyper-ellipse, and U applies a rotating unitary transformation.[1] The sequence of transformations

represents the data on a basis that aligns itself in the orthogonal directions that account for the most variance in the data.

The following theorem is stated from *AMATH 582: Computational Methods for Data Analysis* (Kutz):[1]

Theorem 1: Every matrix $A \in \mathbb{C}^{m \times n}$ has a singular value decomposition (Eqn. 3). Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined, and, if *A* is square and the σ_j distinct, the singular vectors $\{\vec{u}_j\}$ and $\{\vec{v}_j\}$ are uniquely determined up to complex signs (complex scalar factors of absolute value 1).

Of significant import is the unconditional guarantee of the existence of the SVD. The process of calculating the SVD is derived from the following observation:

$$\begin{aligned} A^T A V &= \left(U \Sigma V^*\right)^T \left(U \Sigma V^*\right) V \\ &= V \Sigma U^* U \Sigma V^* V \\ &= V \Sigma^2 V^* V \\ &= V \Sigma^2 \end{aligned} \qquad \textbf{(Eqn. 4)}$$

Similarly, $AA^T U = U\Sigma^2$ **(Eqn. 5)**. Solving these eigenvalue equations will yield the bases $\{\vec{u}_j\}$ and $\{\vec{v}_j\}$ and the singular values $\{\sigma_j\}$.

Considering that the bases $\{\vec{u}_j\}$ and $\{\vec{v}_j\}$ are obtained in practice through solving eigenvalue systems, it should come as no surprise that the diagonalization of a matrix *A* through fundamental methods of linear algebra has many similarities to the diagonalization of the same matrix through singular value decomposition. A system of eigenvalue equations for the matrix *A* can be expressed as

$$\begin{aligned} A\vec{x}_1 &= \lambda_1 \vec{x}_1 \\ A\vec{x}_2 &= \lambda_2 \vec{x}_2 \\ &\vdots \\ A\vec{x}_n &= \lambda_n \vec{x}_n \end{aligned} \qquad \text{or} \qquad A = S\Lambda S^{-1} \qquad \textbf{(Eqn. 6)}$$

Where *S* is the matrix whose columns are the eigenvectors $\vec{x}_i$ and Λ is a diagonal matrix whose non-zero entries are the eigenvalues λ_i.

Compare this to the diagonalization of *A* through SVD. Given a linear system $A\vec{x} = \vec{b}$, first express the vectors in the bases *U* and *V* as $\vec{x} = V\hat{\vec{x}}$ and $\vec{b} = U\hat{\vec{b}}$. It then follows that

$$\begin{aligned} A\vec{x} &= \vec{b} \\ U^* A\vec{x} &= U^*\vec{b} \\ U^* U\Sigma V^* \vec{x} &= U^*\vec{b} \\ \Sigma\hat{\vec{x}} &= \hat{\vec{b}} \end{aligned} \qquad \textbf{(Eqn. 7)}$$

Thusly, the matrix *A* is reduced to the diagonal matrix of singular values.

The two processes for diagonalization of the matrix *A* do have some notable differences. The first is that the SVD requires two sets of basis vectors whereas the eigenvalue decomposition only requires one. Secondly, the SVD method uses orthonormal bases, which is not necessarily true of the vectors in *S* used in Eqn. 6. Lastly, the SVD is guaranteed to exist, which is not true of the eigenvalue decomposition.

To explore the usefulness of the SVD, data was collected that tracked the motion of a paint can suspended from a spring. Three cameras in arbitrary orientations recorded the oscillating motion. For each camera, the location of the mass in the plane of the camera was observed over time. The ordered pairs of vectors $(\vec{x}_i, \vec{y}_i)$ are not consistent with the coordinate system of the oscillating mass and, as the cameras are all recording the motion from different perspectives, the observations contain redundancies. The singular value decomposition of the data will align it with the coordinate system of the mass and identify the principal motions of the paint can. This identification will be made possible through an analysis of the covariance matrix of the transformed data.

Given two vectors $\vec{a} = [a_1\ a_2 \cdots a_n]$ and $\vec{b} = [b_1\ b_2 \cdots b_n]$ whose means are equal to zero, the variances are defined as $\sigma_a^2 = (n-1)^{-1} \cdot \vec{a} \cdot \vec{a}^T$ and $\sigma_b^2 = (n-1)^{-1} \cdot \vec{b} \cdot \vec{b}^T$. The covariance between the vectors is given by $\sigma_{ab}^2 = (n-1)^{-1} \cdot \vec{a} \cdot \vec{b}^T$. For the data in this experiment, the six data streams will be organized into a matrix $X \in \mathbb{R}^{6\times n}$, where *n* is the number of observations. The covariance matrix is then given by

$$C_X = \frac{1}{n-1} XX^T \qquad \textbf{(Eqn. 8)}$$

Entries along the diagonals correspond to the variances of the $(\vec{x}_i, \vec{y}_i)$. Off-diagonal entries give the covariance between pairs of non-identical data streams. Due to noise and redundancy in the

data, the matrix C_x will likely be uninformative. The SVD will transform the data and remove some of the noise and redundancy in the process.

By Eqn. 3 and Theorem 1, the data can be decomposed into $X = U\Sigma V^*$. Transforming the data into $Y = U^* X$ gives

$$\begin{aligned} C_Y &= \frac{1}{n-1} YY^T \\ &= \frac{1}{n-1}\left(U^* X\right)\left(X^T U\right) \\ &= \frac{1}{n-1} U^*\left(U\Sigma V^*\right)\left(V\Sigma U^*\right)U \\ &= \frac{1}{n-1}\Sigma^2 \end{aligned} \qquad \textbf{(Eqn. 9)}$$

The covariance matrix C_Y of the transformed data, which has been diagonalized, will zero out the covariance between the data streams and rank the dynamical components of the motion of the mass from greatest to least variance.

III. Algorithm Implementation and Development

The mass was set into motion under four different initial conditions. Three different cameras recorded the motion of each test. I started by importing the videos and watching the sets for each test. Observing the dynamics of motion gave an indication of how many significant singular values I should expect.

The most challenging and tedious part of this endeavor was extracting the coordinates of the mass from the video frames. I began by trying to isolate the light that was attached to the top of the paint can. The light should have pixels that are close to [255 255 255] on the RGB spectrum, so I set a threshold filter at values such as [253 253 253] to zero out the rest of the image. This usually yielded several pixels that were obviously the light source, but also gave some random pixels around the room. Further complicating the process was the fact that in some of the tests, the can was rotating and the light was not visible. After reading some forums online, I came across the MATLAB function `impixel`, which pauses execution and allows the user to select a pixel with the mouse. It returns the *x* and *y* coordinates of the pixel, as well as its RGB value. I wrote a script that imported one of the video files and processed it frame-by-frame. Each iteration of a loop loaded the next frame, called `impixel`, appended the coordinates that I selected to vectors, and bashed me on the head with a rubber mallet. At the end of the video, the vectors were exported as .mat files. Rinse, wash, repeat until all twelve videos were processed.

The next step was to process the data from each test. The data import yielded six data streams representing the *x* and *y* coordinates of the paint can from the perspective of each of the three cameras. The videos were out of sync, so much so that a false significant singular value was calculated in Test 1. To adjust, I found the minimum value in the *z*-direction of the paint can for the first oscillation of each stream, then took a subset of each stream that ran from the corresponding frame to the end of the stream. I determined the minimum length *n* of the modified streams and truncated the other streams to be the same length. The data were stacked to form a $6 \times n$ matrix.

Before calculating the covariance matrix, it was necessary to center the data. The mean of each stream was calculated and `repmat` was used to repeat a column of the means *n* times. The repeated matrix was subtracted from the data set to center it.

Unsurprisingly, the covariance matrix of the raw data showed several dependencies. The singular value decomposition was accomplished with a single command: `[u, s, v] = svd(X);` The transformed data were calculated as $Y = U^* X$ and the covariance was calculated. I verified the same matrix was calculated with both the first and last lines of Eqn. 9. The diagonal entries of the covariance matrix were the squares of the singular values and the off-diagonal entries were zero. I calculated the proportion of the total variance for each singular value and plotted both the proportions and the cumulative proportions.

The cumulative proportion distributions were the most informative. They reveal a plateau, a point beyond which the singular values no longer capture a significant portion of the total variance. The values ramping up to the plateau represent the dynamics of motion that are present in the system, the rest can largely be attributed to statistical variability and the constraints imposed by the variance formula. There really was nothing left to do at this point except to match the significant singular values to the observed dynamics in the system.

IV. Computational Results

A preliminary discussion of depth perception will be used to justify the interpretations of the singular values obtained through these data. It is known that monocular vision does a poor job of perceiving depth. It is the brain's comparison of images from two sources that gives the impression of depth.[2] An object can swing from a pendulum or rotate around an axis, but what we ultimately perceive is just the position of an object in a three-dimensional rectangular coordinate system. One camera recording the motion of a paint can will not be able to perceive depth, but two or three will. Regardless of the type of motion that the paint can is subjected to, it can ultimately be summarized by a three-dimensional rectangular coordinate system. As the position of the paint can was measured with rectangular coordinates, it seems appropriate to interpret the dimensions yielded by the SVD as rectangular axes that have been resolved from multiple perspectives.

For the sake of thoroughness, the singular values from the experiment are summarized in **Table 1** below. They're only useful in so far as how they are related to the covariance matrix of the new basis (Eqn. 9). The table also gives the proportion of the total variance that is attributable to each of the orthogonal directions in the new basis. Per the discussion in the preceding paragraph, the six data streams from the cameras have simply been reduced to $\mathbb{R}^3$ by the SVD. The interpretation that follows will concern the proportion of total variance, not the singular values. For a prototypical interpretation of the transformed variables in the context of this experiment, see Appendix A.

Table 1: Summary of singular values and variance by test.

Test 1		**Test 2**		**Test 3**		**Test 4**	
Singular Values	**Proportion of Variance**	**Singular Values**	**Proportion of Variance**	**Singular Values**	**Proportion of Variance**	**Singular Values**	**Proportion of Variance**
1190.2	0.9672	1122.5	0.6164	779.9	0.7015	1192.1	0.7553
184.2	0.0232	672.8	0.2215	411.3	0.1951	553.2	0.1626
81.1	0.0045	438.1	0.0939	266.0	0.0816	334.8	0.0596
63.2	0.0027	240.3	0.0283	111.2	0.0143	173.6	0.0160
51.0	0.0018	221.4	0.0240	73.9	0.0063	93.5	0.0046
29.7	0.0006	180.7	0.0160	33.2	0.0013	59.2	0.0019

In the first test case, the paint can was subjected to a simple harmonic oscillation along the *z*-axis. Under ideal conditions, this should yield just one dominant source of variance. After several attempts at recording the location of the paint can, I kept obtaining two significant proportions near 0.51 and 0.46. This is the point at which I realized the videos were out of sync and that I would have to modify the data sets to account for it. The modified data behaved as expected, yielding a primary singular value that accounted for 96.7% of the variance **(Figure 1)**.

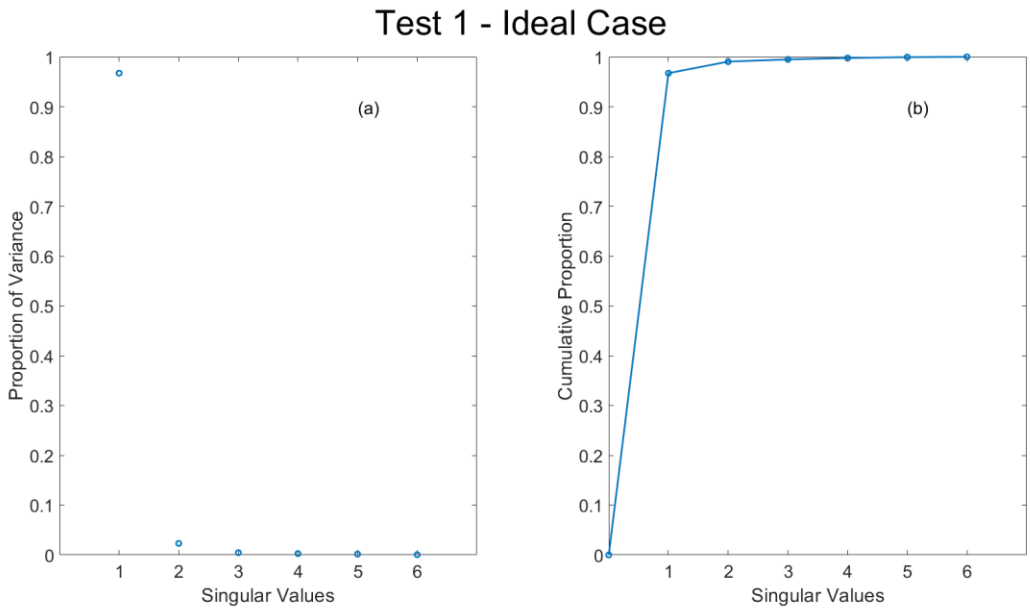

Figure 1: The paint can was initialized with simple harmonic oscillation in the direction of the *z*-axis. The proportion of variance by singular value (a) and the cumulative proportions (b) are shown. One dominant source of variance is revealed.

The second test case had the same initial conditions as the first, but noise was introduced into the recording by shaking the cameras. The noise had the effect of producing a false sense of movement in the *xy*-planes of the cameras. The dominant source of variance is still the simple harmonic oscillation, which accounted for 61.6% of the variance. The next two singular values have significant variances that account for 22.2% and 9.4% of the total variance **(Figure 2)**. These components represent the false impression of movement being resolved into a change in position in the *xy*-plane of the paint can. These three singular values account for 93% of the variance in the data.

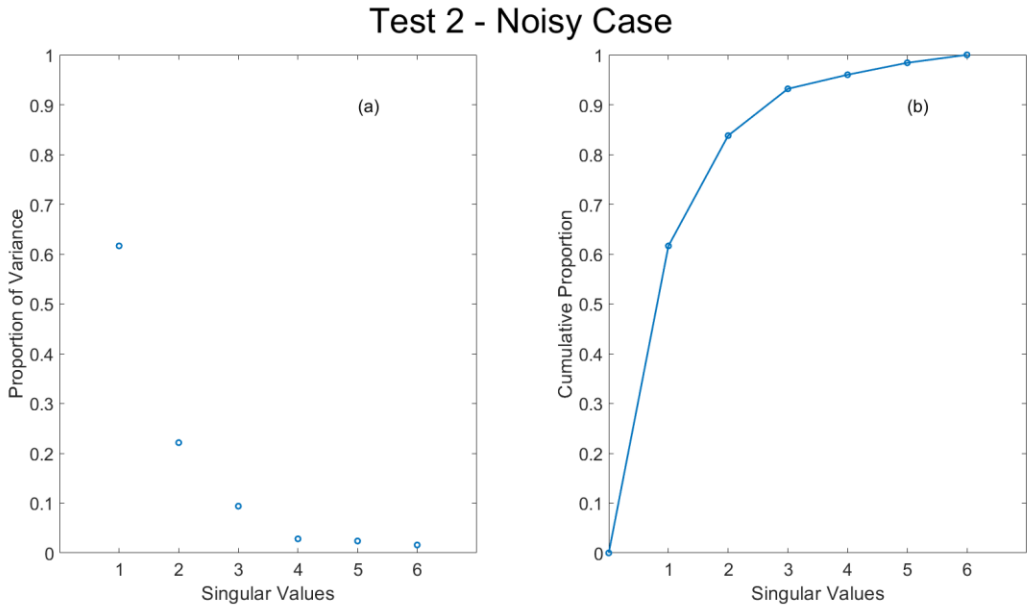

Figure 2: The paint can was initialized with simple harmonic oscillation in the direction of the *z*-axis. The cameras were shaken, not stirred, to introduce noise into the data streams. The proportion of variance by singular value (a) and the cumulative proportions (b) are shown. The significant singular values capture the harmonic oscillations and the shake of the camera, which is falsely resolved into motion in the *xy*-plane of the paint can.

In the third test case, the paint can was released off-center. This introduced a pendulum motion in the *xy*-plane of the paint can. While the swing of the pendulum and the rotation of its plane relative to the *xy*-plane of the mass can be described by angles, recall that the data are rectangular. The net motion of the mass can be described with coordinates in $\mathbb{R}^3$. The most dominant source of variance is due to the simple harmonic oscillations, which accounts for 70.2% of the total variance. The variance in the *xy*-plane of the paint can is accounted for by the next two singular values, which weigh in at 19.5% and 8.2%. The fact that there is significant variance in both the *x*- and *y*-planes indicates that the plane of the pendulum is rotating. The remaining variability is insignificant at 2.1% of the total variance and is an artifact of the SVD process.

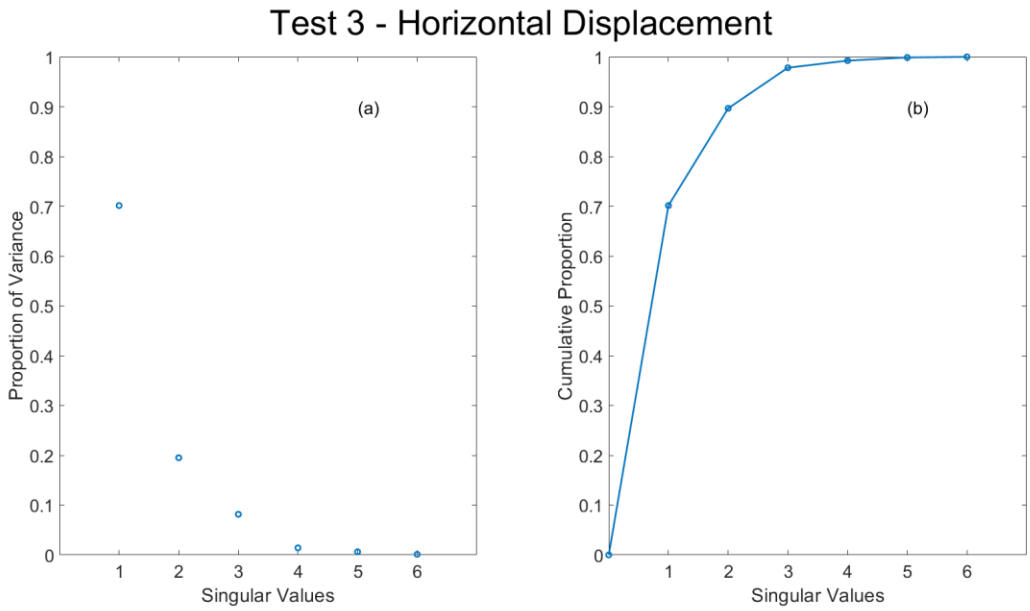

Figure 3: The paint can was initialized with simple harmonic oscillation in the direction of the *z*-axis. A pendulum motion was introduced by releasing the mass off-center, which produces motion in the *xy*-plane of the paint can. The proportion of variance by singular value (a) and the cumulative proportions (b) are shown. The significant singular values represent the overall dynamics as motion in $\mathbb{R}^3$.

In the fourth and final test case, the paint can was released off-center and torqued. The paint can demonstrated simple harmonic oscillations, a pendulum motion, and a rotation. Watching the videos reveals some very erratic dynamics. The pendulum swings are wide and sweeping at first, but they die off quickly and the paint can settles into a stable rotating harmonic oscillation. Ultimately though, all of this can be captured as position in $\mathbb{R}^3$. The proportions of variance captured by the SVD are comparable to those obtained in the third test (0.7553, 0.1626, 0.0596) and should be interpreted in the same fashion.

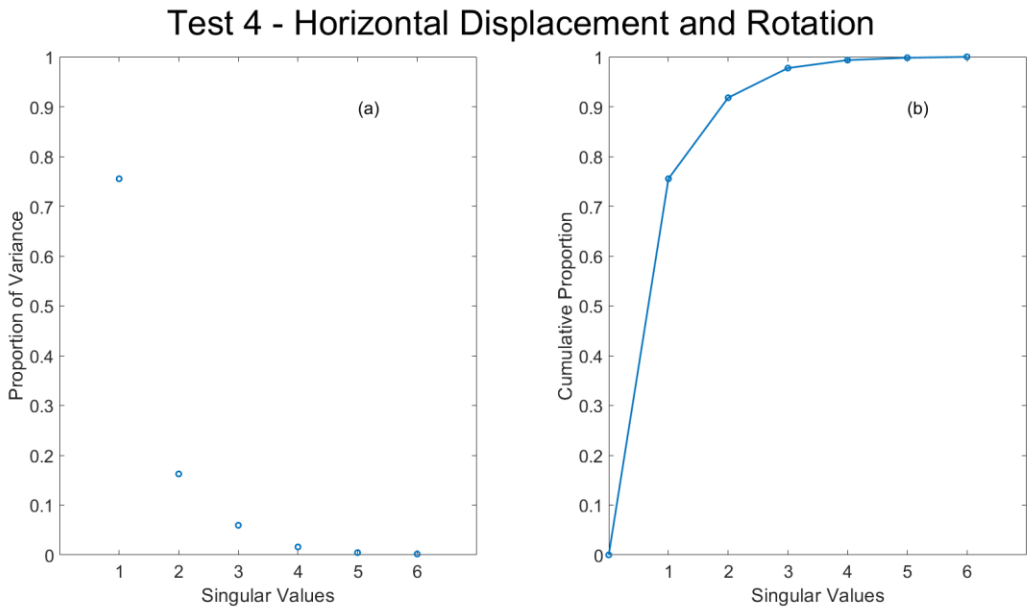

Figure 4: The paint can was initialized with simple harmonic oscillation in the direction of the *z*-axis. A pendulum motion was introduced by releasing the mass off-center, which produces motion in the *xy*-plane of the paint can. The can was also torqued to produce a rotation around the pendulum axis. The proportion of variance by singular value (a) and the cumulative proportions (b) are shown. The significant singular values represent the overall dynamics as motion in $\mathbb{R}^3$.

V. Summary and Conclusions

The experiment discussed in this paper demonstrates how the singular value decomposition can be used to reduce the number of dimensions necessary to describe a data structure. It was observed that getting the data into the computer in an appropriate form for analysis is often the biggest hurdle. The built-in functionality of MATLAB did all of the heavy lifting. In fact, practical applications of the SVD would be impossible without computing technologies.

The SVD is one of the hottest things going in multivariate analysis right now. Its ability to recognize the most efficient basis for a data set is tremendously powerful, especially if that basis has more than three dimensions, as visual models for such data elude us. In the process of removing redundancies and noise, the most significant components of a data structure are identified. The guarantee of its existence is a rare commodity in mathematics. In addition to being useful in its own right, the SVD is also a springboard for classification algorithms, through which technologies such as facial recognition are made possible. Big discoveries loom on the horizon and SVD will be at the heart of many of them.

A. Prototypical Interpretation of Transformed Variables

The variables Y_i are the projections of the data set onto the new basis vectors, which have been determined to be $\mathbb{R}^3$. Recall that the paint can was subjected to various periodic dynamics such as oscillations and rotations. One need only watch the videos to observe that the motion of the paint can exhibits back-and-forth motion in all three planes of $\mathbb{R}^3$. These dynamics generated by Test 4 are observed in **Figure A.1**, which shows clear oscillatory motion in the three directions of significance. Y_1 is a steady and consistent signal representing the harmonic oscillation. It's difficult to determine whether Y_2 captures motion in the *x*- or *y*-plane of the paint can, as their orientation is determined by the SVD.

In **Figure A.2**, the first three components are plotted in $\mathbb{R}^3$. The axes are rotated to give a clear view of the first and last point. The angle gives a clear view of the early influence of the pendulum, which was noted to have died down quickly. The paint can settled into a rotating harmonic oscillation near $(0,0)$ in the *xy*-plane.

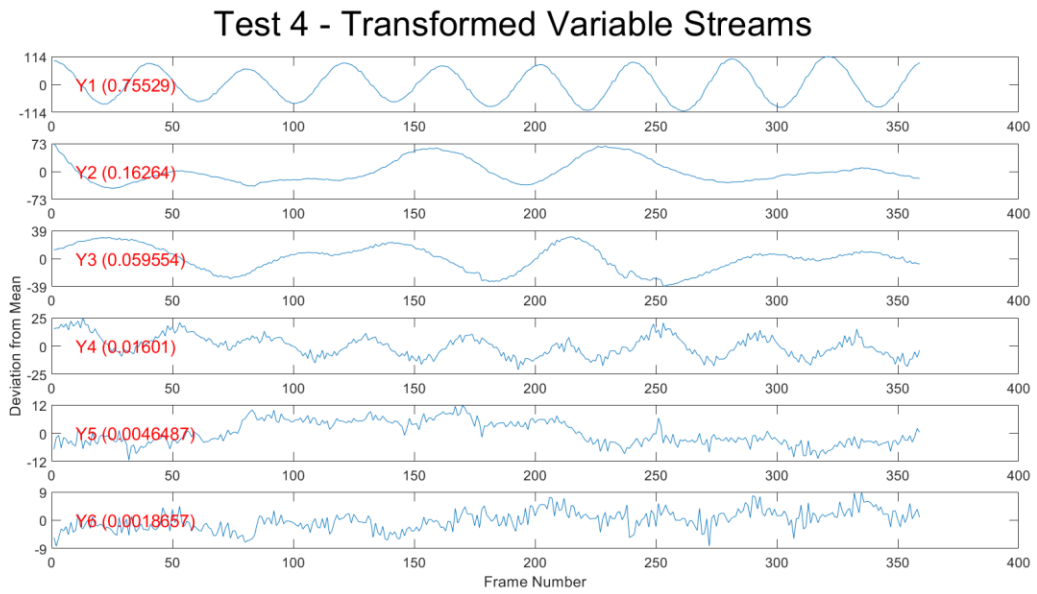

Figure A.1: The six data streams from the cameras were transformed via the SVD. The transformed data streams are ranked according to their contribution to the total variance. The first three are significant sources of variability and exhibit periodic behavior around the mean. By the sixth variable, there's nothing left to account for but trivial amounts of statistical compensation.

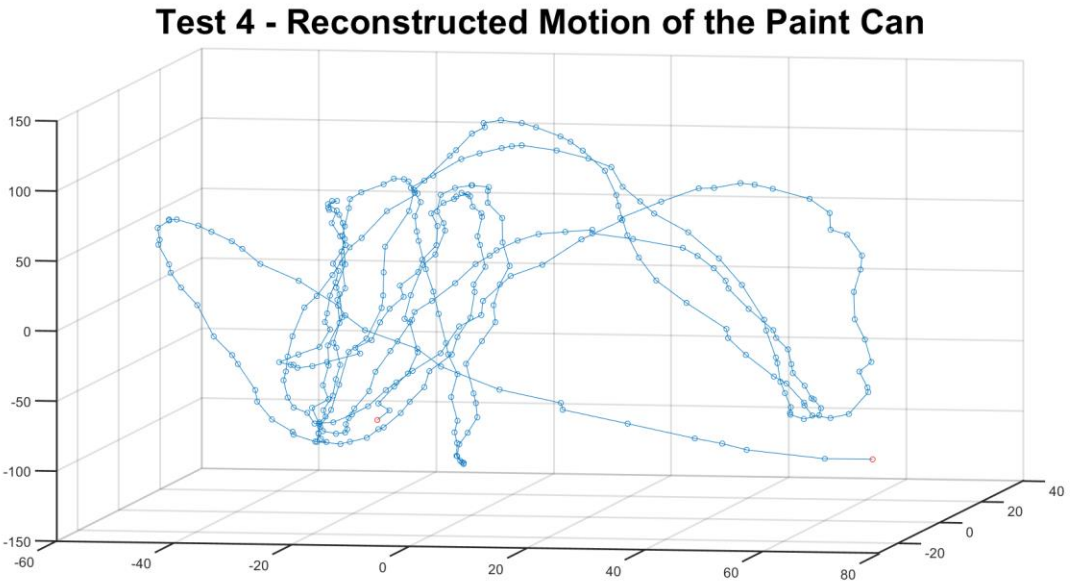

Figure A.2: The principal components of the SVD resolve the multiple two-dimensional data streams from the three cameras into an approximation of the motion of the paint can in $\mathbb{R}^3$.

B. MATLAB Codes

```
% Homework 3 - Data Import
% David Long
% 1.30.19

clear all; close all; clc

% Load video data

load('cam3_4.mat');




% Create variables

camFrames = cell(1, size(vidFrames3_4, 4));

x3_4 = []; y3_4 = [];




% Select coordinates

for j=1:length(camFrames)
    camFrames{j} = vidFrames3_4(:,:,:,j);
    [xi, yi, P] = impixel(camFrames{j});
    x3_4(j) = xi;
    y3_4(j) = yi;
    disp(j)
end




% Save coordinates

save('x3_4', 'x3_4');
save('y3_4', 'y3_4');




% Plot

plot(x3_4, y3_4)
```

```
% Homework 3 - Test 1 SVD
% David Long
% 2.9.19

clear all; close all; clc

% Import data

for j=1:3

    currentX = strcat('x', num2str(j), '_1.mat');
    currentY = strcat('y', num2str(j), '_1.mat');
    load(currentX);
    load(currentY);

end




% Test 1

% Construct X

[y1max, start1] = max(y1_1(1:40));
[y2max, start2] = max(y2_1(1:40));
[x3max, start3] = max(x3_1(1:40));

n = min([(size(y1_1, 2) - start1), (size(y2_1, 2) - start2), (size(x3_1, 2) -
start3)]);
X = [x1_1(start1:start1 + n - 1); y1_1(start1:start1 + n - 1);
x2_1(start2:start2 + n - 1); y2_1(start2:start2 + n - 1);...
    x3_1(start3:start3 + n - 1); y3_1(start3:start3 + n - 1)];




% Check sync

figure(1)
subplot(3, 1, 1)
plot(X(2,:))
subplot(3, 1, 2)
plot(X(4,:))
subplot(3, 1, 3)
plot(X(5,:))
```

```
% Center data

m = size(X, 1);
mn = mean(X, 2);
X = X - repmat(mn, 1, n);
CX = (1/(n-1))*(X*X');




% SVD

[u, s, v] = svd(X);
Y = u'*X;
CY = (1/(n-1))*(Y*Y');
CY2 = (1/(n-1))*diag(s).^2;
disp(diag(s))
contributions = diag(CY)/trace(CY)
cumContributions = [0; cumsum(contributions)];




% Plots

figure(4)

subplot(1, 2, 1)
plot(contributions, 'o', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(a)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Proportion of Variance')

subplot(1, 2, 2)
plot(0:6, cumContributions, 'o-', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(b)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Cumulative Proportion')

sgtitle('Test 1 - Ideal Case', 'Fontsize', 40)
```

```
% Homework 3 - Test 2 SVD
% David Long
% 2.9.19

clear all; close all; clc

% Import data

for j=1:3

    currentX = strcat('x', num2str(j), '_2.mat');
    currentY = strcat('y', num2str(j), '_2.mat');
    load(currentX);
    load(currentY);

end




% Construct X

[y1max, start1] = max(y1_2(1:40));
[y2max, start2] = max(y2_2(1:40));
[x3max, start3] = max(x3_2(1:40));

n = min([(size(y1_2, 2) - start1), (size(y2_2, 2) - start2), (size(x3_2, 2) -
start3)]);
X = [x1_2(start1:start1 + n - 1); y1_2(start1:start1 + n - 1);
x2_2(start2:start2 + n - 1); y2_2(start2:start2 + n - 1);...
    x3_2(start3:start3 + n - 1); y3_2(start3:start3 + n - 1)];




% Check sync

figure(1)
subplot(3, 1, 1)
plot(X(2,:))
subplot(3, 1, 2)
plot(X(4,:))
subplot(3, 1, 3)
plot(X(5,:))
```

```
% Center data

m = size(X, 1);
mn = mean(X, 2);
X = X - repmat(mn, 1, n);
CX = (1/(n-1))*(X*X');




% SVD

[u, s, v] = svd(X);
Y = u'*X;
CY = (1/(n-1))*(Y*Y');
CY2 = (1/(n-1))*diag(s).^2;
disp(diag(s))
contributions = diag(CY)/trace(CY)
cumContributions = [0; cumsum(contributions)];




% Plots

figure(4)

subplot(1, 2, 1)
plot(contributions, 'o', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(a)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Proportion of Variance')

subplot(1, 2, 2)
plot(0:6, cumContributions, 'o-', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(b)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Cumulative Proportion')

sgtitle('Test 2 - Noisy Case', 'Fontsize', 40)
```

```
% Homework 3 - Test 3 SVD
% David Long
% 2.9.19

clear all; close all; clc

% Import data

for j=1:3

    currentX = strcat('x', num2str(j), '_3.mat');
    currentY = strcat('y', num2str(j), '_3.mat');
    load(currentX);
    load(currentY);

end




% Construct X

[y1max, start1] = max(y1_3(1:40));
[y2max, start2] = max(y2_3(1:40));
[x3max, start3] = max(x3_3(1:40));

n = min([(size(y1_3, 2) - start1), (size(y2_3, 2) - start2), (size(x3_3, 2) -
start3)]);
X = [x1_3(start1:start1 + n - 1); y1_3(start1:start1 + n - 1);
x2_3(start2:start2 + n - 1); y2_3(start2:start2 + n - 1);...
    x3_3(start3:start3 + n - 1); y3_3(start3:start3 + n - 1)];




% Check sync

figure(1)
subplot(3, 1, 1)
plot(X(2,:))
subplot(3, 1, 2)
plot(X(4,:))
subplot(3, 1, 3)
plot(X(5,:))
```

```
% Center data

m = size(X, 1);
mn = mean(X, 2);
X = X - repmat(mn, 1, n);
CX = (1/(n-1))*(X*X');




% SVD

[u, s, v] = svd(X);
Y = u'*X;
CY = (1/(n-1))*(Y*Y');
CY2 = (1/(n-1))*diag(s).^2;
disp(diag(s))
contributions = diag(CY)/trace(CY)
cumContributions = [0; cumsum(contributions)];




% Plots

figure(4)

subplot(1, 2, 1)
plot(contributions, 'o', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(a)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Proportion of Variance')

subplot(1, 2, 2)
plot(0:6, cumContributions, 'o-', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(b)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Cumulative Proportion')

sgtitle('Test 3 - Horizontal Displacement', 'Fontsize', 40)
```

```
% Homework 3 - Test 4 SVD
% David Long
% 2.9.19

clear all; close all; clc

% Import data

for j=1:3

    currentX = strcat('x', num2str(j), '_4.mat');
    currentY = strcat('y', num2str(j), '_4.mat');
    load(currentX);
    load(currentY);

end




% Construct X

[y1max, start1] = max(y1_4(1:40));
[y2max, start2] = max(y2_4(1:40));
[x3max, start3] = max(x3_4(1:40));

n = min([(size(y1_4, 2) - start1), (size(y2_4, 2) - start2), (size(x3_4, 2) -
start3)]);
X = [x1_4(start1:start1 + n - 1); y1_4(start1:start1 + n - 1);
x2_4(start2:start2 + n - 1); y2_4(start2:start2 + n - 1);...
    x3_4(start3:start3 + n - 1); y3_4(start3:start3 + n - 1)];




% Check sync

figure(1)
subplot(3, 1, 1)
plot(X(2,:))
subplot(3, 1, 2)
plot(X(4,:))
subplot(3, 1, 3)
plot(X(5,:))
```

```
% Center data

m = size(X, 1);
mn = mean(X, 2);
X = X - repmat(mn, 1, n);
CX = (1/(n-1))*(X*X');




% SVD

[u, s, v] = svd(X);
Y = u'*X;
CY = (1/(n-1))*(Y*Y');
CY2 = (1/(n-1))*diag(s).^2;
disp(diag(s))
contributions = diag(CY)/trace(CY)
cumContributions = [0; cumsum(contributions)];




% Plots

figure(4)

subplot(1, 2, 1)
plot(contributions, 'o', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(a)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Proportion of Variance')

subplot(1, 2, 2)
plot(0:6, cumContributions, 'o-', 'Linewidth', 2)
axis([0 7 0 1])
text(5, 0.9, '(b)', 'Fontsize', 20)
set(gca, 'Fontsize', 20, 'Xtick', [1 2 3 4 5 6])
xlabel('Singular Values'), ylabel('Cumulative Proportion')

sgtitle('Test 4 - Horizontal Displacement and Rotation', 'Fontsize', 40)
```

```
% Transformed variable plots

figure(5)

for j=1:size(Y,1)
    subplot(6,1,j)
    plot(Y(j,:))
    text(10, 0, strcat('Y', num2str(j), ' (', num2str(contributions(j)),
')'), 'Color', 'r', 'Fontsize', 20)
    set(gca, 'Fontsize', 16, 'Ytick', [-ceil(max(abs(Y(j,:)))), 0,
ceil(max(abs(Y(j,:))))],...
        'Ylim', [-ceil(max(abs(Y(j,:)))), ceil(max(abs(Y(j,:))))])
    if (j == 4)
        ylabel('Deviation from Mean')
    end
end

sgtitle('Test 4 - Transformed Variable Streams', 'Fontsize', 40)
xlabel('Frame Number')




figure(6)
plot3(Y(2,:), -Y(3,:), -Y(1,:), 'o-')
hold on
plot3(Y(2,1), -Y(3,1), -Y(1,1), 'ro')
hold on
plot3(Y(2,end), -Y(3,end), -Y(1,end), 'ro')
grid on
set(gca, 'Fontsize', 16, 'View', [10 10], 'Linewidth', 2)
title('Test 4 - Reconstructed Motion of the Paint Can', 'Fontsize', 40)
```

C. References

[1] Kutz, J. N. February 26, 2010. *AMATH 582: Computational Methods for Data Analysis.* Department of Applied Mathematics, University of Washington, Seattle, WA 98195.

[2] *Advanced Topics in Perception | Boundless Psychology*. (2019). *Courses.lumenlearning.com*. Retrieved 19 February 2019, from https://courses.lumenlearning.com/boundless-psychology/chapter/advanced-topics-in-perception/