AMATH 482/582: HOMEWORK 3

JONATHAN MCCORMACK

Applied Mathematics Department, University of Washington, Seattle, WA
jrmack20@uw.edu

Abstract. Video data can be modeled as a combination of a static, low-rank background component and a sparse foreground component containing the moving objects. Principal Component Analysis can be used to exploit this difference by treating each pixel in a video frame as an independent sensor. When a sensor has variation in its data, that sensor is observing meaningful foreground. This has many applications to assist automation techniques built on the ability for computers to sense or "see" the physical world.

1. Introduction and Overview

In Linear Algebra, a key pursuit is to decompose a matrix of data into smaller, simpler marices that can be recombined to approximate the original data. The Singular Value Decomposition (SVD) is incredibly powerful because it orders the original data from most to least prominent. The Principal Component Analysis (PCA) furthers this process by applying statistics to relate the measurements of what may appear to be unrelated sensors. In image and video processing, this allows a program to process key features that may be separated by their static or dynamic nature as well as their color, space, time, etcetera.

In this report, we apply PCA to a video sample in order to identify the static regions of the video frame. We hope to validate this method by presenting the separated background and foreground at a oarticular frame. We also intend to integrate a Randomized SVD algorithm to show how effective SVD is at representing and preserving key features of data without performing excessive computations on the whole data set[3].

2. Theoretical Background

The Singular Value Decomposition (1) separates the matrix A into two sets of orthonormal vectors, traditionally input vectors: v_1 to v_n and output vectors u_1 to u_m[4]. This comes from the way the eigenvector is the input vector applied to A as in $Ax = \lambda x$. The singular vectors U and V^T are derived by finding the eigenvectors of AA^T and A^TA respectively. This comes from the understanding that both U and V^T are unitary matrices and have the quality to multiply with their transpose and produce the Identity matrix, that is $U^TU = I$, as shown in the below proof (2). A key result of SVD is the Eckart-Young Theorem, which says that the closest rank k approximation to the original matrix k is the matrix formed by the first k singular values and their respective singular vectors, $A_k = U_k\Sigma_k V_k^T = \sum_{i=1}^{k} \sigma_i u_i v_i^T$.

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

Date: February 17, 2026.

$$A^T A = (U\Sigma V^T)^T(U\Sigma V^T) = (V\Sigma^T U^T)(U\Sigma V^T) = V\Sigma^T I\Sigma V^T = V(\Sigma^T\Sigma)V^T \tag{2}$$

Principal Component Analysis takes SVD a step further by recognizing the inherent statistics qualities. Imagine that each row of the original data matrix is a separate sensor or measurement and each column is a sample, whether a video frame or a patient in a clinical trial. This would mean that how the data varies inside the row shows the variance between different samples. Similarly, the relationship between different columns, shows the correlation between two different measurements/sensors. When one multiplies the original matrix by its transpose in SVD, the diagonal entries show the variance between the samples and the covariance between the different measurements (this is true for both AA^T and A^TA). The first singular vector U_1 can be seen as the best representation of the most prominant measurement in each row. Geometrically, this is the singular vector with the most variance (distance from the zero vector) because all other singular vectors make small adjustments off of this baseline singular vector [4].

Principal Component Analysis has a couple unique changes to the steps of SVD[1]. First, each row of the original matrix, X is normalized by subtract the row mean, that is $B = X - \bar{X}$. This ensures that if different rows are scaled differently, due to a differences in sensor, it does not over or underweight any of the measurements[2]. Next, the covariance matrix, C is formed by multiplying B^T with B and dividing by $n-1$, where n is the number of rows of X. $C = \frac{1}{n-1}B^TB$. Lastly, the eigenvectors of the covariance matrix are the right singular vectors, V^T, and are referred to as the principal components.

Principal Component Analysis and SVD are both computationally efficient, but with large sets of data, it is important to manage computer memory limitations. The paper by Martinsson and Tropp discus the ability to radically lower the computational cost of SVD by fist sampling the column space of the original data matrix with a Gaussian random projection matrix and computing the QR decomposition of the resulting projection [3][1]. In the next section of this report, we will show the accuracy of their Randomized SVD algorithm and how it extends the effectiveness of PCA on high dimensional data sets.

3. Algorithm Implementation and Development

Pseudocode

```
(1) Vectorize the data, reshaping each n-dimensional array into a column vector
(2) Subtract the mean from each row, centering each row at zero
(3) Apply the Randomized SVD Algorithm[3] with $k = 1$ and $p \geq 5$
(4) Create the rank-1 approximation of the data: $A_1 = U_1\Sigma_1V_1^T$
(5) Find the Background: $A_1$ + mean for each row
(6) Find the Foreground: Original data - Background
```

The entire code took advantage of the numpy, matplotlib, scipy.spatial, and cv2 packages. In order to validate the corrent implementation of the Randomized SVD algorithm and its efficacy as part of PCA, the instructor provided 1000 random functions. These random functions were from $x = 0.0$ to $x = 1.0$ and were created by finding the dot product of a Gaussian random vector with a special Cholesky decomposition of a covariance matrix, this ensured that the random functions had underlying low-rank structure that could be extracted via the SVD. Figure 1 below is a simple plot showing the different functions generated. Although there is underlying structure, it is unrecognizable from the density and variance of data.

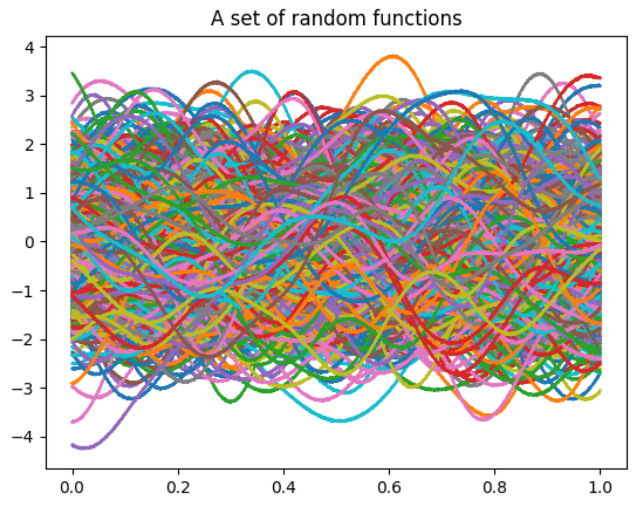

FIGURE 1. Raw test data with underlying, low-rank structure. Each column of the Raw data array is a set of y-values that correspond to the x-values produced by np.linspace(0, 1, 4096)

The test data was run through the above pseudocode without generating background or foreground images. The data was sampled at Rank (k) = 5, 10, 20 and Oversampling parameter (p) = 0, 5, 10. Each Randomized SVD was run 10 times and each time the Frobenius norm of the Full SVD approximation minus the Randomized SVD approximation was stored as error. The below table shows the mean and standard deviations of the Frobenius Error. Clearly the mean error decreases with both higher rank and oversampling behavior. However, the difference between the error at p=5 and p=10 is negligible, making p=5 more than adequate for the size of the data. Additionally, Figure 2 shows how the values of σ^2 differed between the Randomized and the Full SVD. Even without oversampling, increasing the rank of the Randomized SVD quickly increases the approximations accuracy to the original data. The scree plot also has a fairly defined ”elbow” at k=10, where this data set could easily be represented by a rank-10 matrix.

Rank (k)	Oversampling parameter (p)	Mean Frobenius Error	Standard Deviation Frobenius Error
5	0	1104.613	75.391
5	5	808.024	25.158
5	10	757.318	0.480
10	0	365.527	73.303
10	5	160.466	10.316
10	10	144.245	2.566
20	0	48.967	4.437
20	5	34.661	0.978
20	10	30.057	0.896

TABLE 1. Error between the data approximation found by the Full and Randomized SVD algorithm applied to the Raw test data.

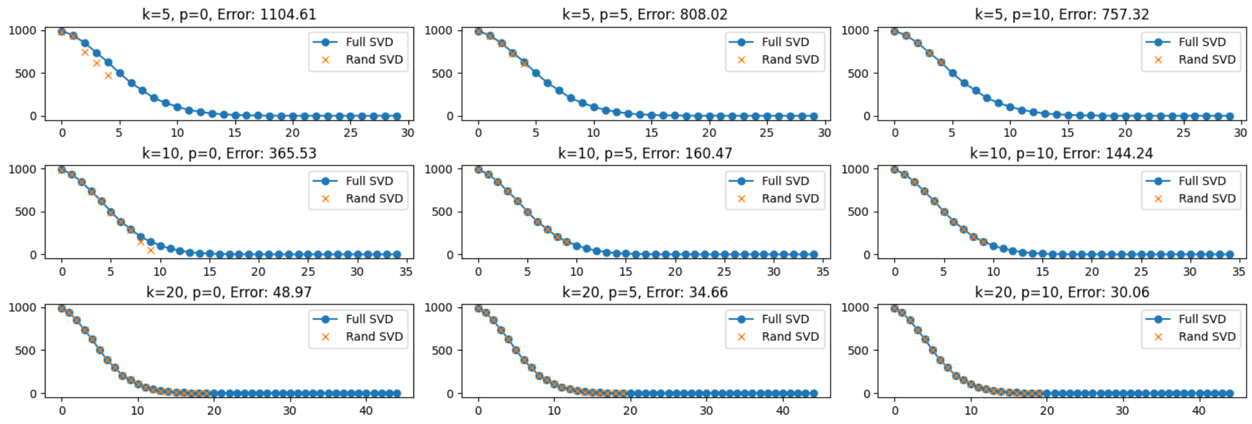

FIGURE 2. Edge detection algorithm applied to noisy images with σ =0.1, 0.25, and 0.5 respectively.

4. Computational Results

When this algorithm is applied to videos for the purpose of extracting the background, the rank of the Randomized SVD must be 1. This is because after the data is centered, the most prominent (or common) value in each row of the original matrix, is the value that was present before and after objects in the foreground interacted with the visual space. This is a flaw in PCA only if an image has a large, dynamic foreground. In this video, the background is the majority of the image and the foreground are individual people, not interacting objects that would affect other areas of the frame.

The video was extracted using the cv.videocapture function and each frame was converted to grayscale and flattened to a column vector to conform to the above algorithm. Each frame is 320 pixels wide and 240 pixels tall. The video includes 794 frames. Although unnecessary, Figure 3 below shows the array of flattened frames as a single image. Clearly some rows are entirely constant, this corresponds to a pixel that is only background. Alternatively, there is few rows that have wide changes in color, meaning PCA will be able to handle the separation of background from foreground.

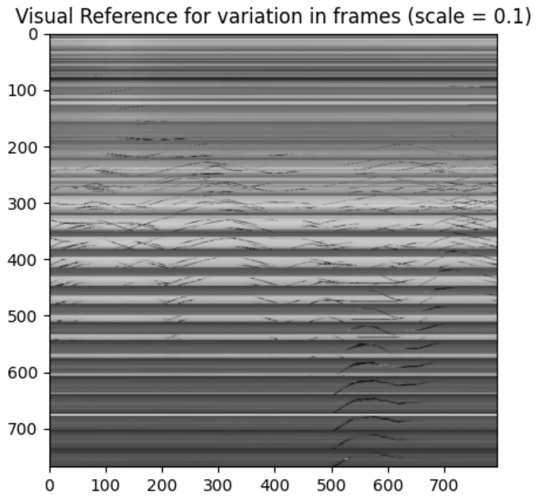

FIGURE 3. Video data array where each column is a frame scaled by 0.1.

When using Principal Component Analysis on the actual video data, the algorithm was run with both Full and Randomized SVD. In the Randomized case, the oversampling parameter was 5 because a higher value provided limited improvement in accuracy. The Full PCA algorithm ran in 4.5 seconds and the Randomized PCA algorithm ran in under 0.6 seconds showing its efficiency. The Full PCA calculated all 795 sigular values and vectors, where the Randomized algorithm only calculated one set. However, the randomized variation had the additional calculations involved in the Gaussian projection and the QR decomposition from the Martinsson and Tropp algorithm[3].

Below Figure 4 and Figure 5 show the use of the Full and Randomized PCA algorithms on the separation of the foreground from the background at frame 100. Both sets of background retained the curb on the opposite side of the street and the van on the upper right side. This two artifact came from the interaction of the foreground with the background. Both approximations contain ”ghosts”, where foreground images remain as low magnitude components of the background. However, the Randomized PCA background has far more ”ghosts”, which comes from a lack of data causing the foreground to be overemphasized by the PCA. Even with a larger data set, the Randomized PCA will have this issue, where it will dissapear in the Full PCA with sufficient data.

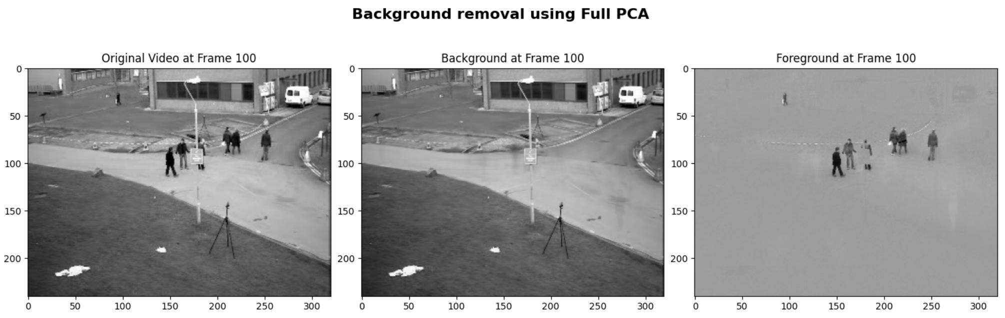
Background removal using Full PCA
Original Video at Frame 100
Background at Frame 100
Foreground at Frame 100

FIGURE 4. The result of using Full PCA to remove the foreground from the background at frame 100.

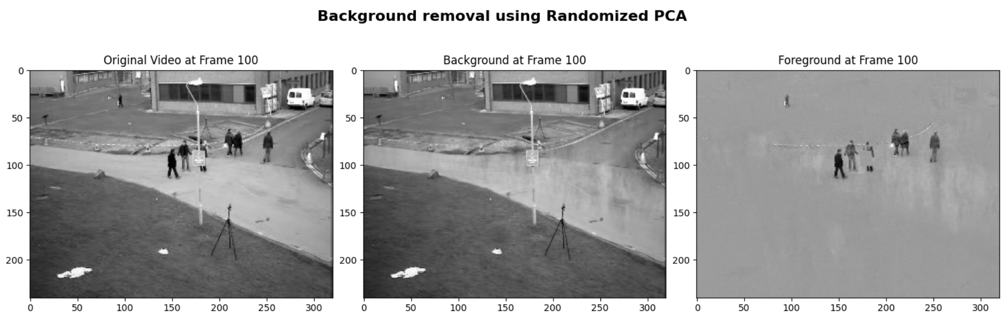
Background removal using Randomized PCA
Original Video at Frame 100
Background at Frame 100
Foreground at Frame 100

FIGURE 5. The result of using Randomized PCA to remove the foreground from the background at frame 100.

5. Summary and Conclusions

Randomized SVD, with a sufficiently high oversampling parameter, gives an accurate low-rank approximation to the original data set. Given a video taken from a static camera with sufficient frames and resolution, Principal Component Analysis using Randomized PCA can separate the static background from the foreground for each frame. Although not addressed above, PCA struggles with foreground that occupies the majority of the data/image and with a dynamic background that is not accounted for. Additionally, PCA picks up high frequency information about both the background and foreground. This means that a reconstructed video has "hallucinated" edges of the foreground and has added white noise in the background.

Future improvements should focus on breaking the PCA into time "chunks" so that the background can be adaptive to each frame. The background should be calculated in color and shifted to grayscale only when saving the output video. Also, the code needs to scale the magnitude of the low-rank approximated data to eliminate the noise produced by the calculations.

Acknowledgements

I would like to thank Professor Hosseini for his lectures, notes, and code examples.

References

[1] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* Cambridge University Press, 2 edition, 2022.

[2] J. N. Kutz. *Data-Driven Modeling Scientific Computation: Methods for Integrating Dynamics of Complex Systems and Big Data.* OUP Oxford, 2013.

[3] P.-G. Martinsson and J. A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

[4] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Philadelphia, PA, 6 edition, 2022.