Singular value decomposition calculation comparison in video fragment processing

Sergii Mashtalir[1,†] and Dmytro Lendel[2,†]

[1] *Kharkiv National University of Radio Electronic, 14, Nauky Ave., Kharkiv, Ukraine*
[2] *Uzhhorod National University, 3, Narodna Square, Uzhhorod, Ukraine*

Abstract

This study considers different approaches to calculating the first singular value of the Singular Value Decomposition (SVD) transform. The SVD is closely associated with several common matrix norms and offers an efficient method for their computation. Sum first k singular values called the Ky Fan k-norm. In our approach, the Ky Fan norm is a fragment descriptor. There is no need to do a complete SVD transformation to obtain the norm value. It is enough to obtain a matrix of singular values. In video fragment analysis, the number of fragments and their size significantly affect the calculation speed. The SVD method is robust but does not necessarily scale well to larger matrices. Thus, to use SVD in a practical sense with large datasets, we needed a faster algorithm that finds the same dominant patterns as regular SVD but with only a fraction of the computational cost. We compare the effectiveness of alternative approaches depending on the size of the fragments and their number.

Keywords

Video stream fragmentation; Fragment processing; Ky Fan norm; Singular value decomposition; UTV; ULV; Lanczos SVD; Randomized SVD; Power Iteration; Data Analysis

1. Introduction

There has been significant interest in the Singular Value Decomposition (SVD) algorithm over the last few years because of its wide applicability in multiple fields of science and engineering, both standalone and as part of other computing methods. The singular value decomposition is the most common and valuable decomposition in computer vision [1]. Computer vision aims to reconstruct the three-dimensional world from two-dimensional images. These images often result in square and non-square singular matrices and transformations in real-world scenarios. Reversing transformations from two to three dimensions cannot be entirely accurate, but it can be effectively estimated using singular value decomposition. Singular value decomposition will also allow us to establish a sense of order in objects and is, therefore, useful whenever attempting to compare. Image denoising [2], image re-scaling [3], image compression [4], motion detection [5], and video fragment processing are far from a complete list of SVD applications.

We focused on video fragment processing, and in our approach, we consider fragments to be geometric parts of video frames, represented as matrices with arbitrary dimensions. The research [6] proposes a singular value decomposition of the matrix and the Ky Fan norm for scene change analysis. In the context of motion detection, this approach was expanded [7]. Dividing the frame into 5x5 or 10x10 allowed us to identify the fragments in which motion occurred Figure 1.

In the study [8], increasing the number of fragments to 100x100 allowed us to find the contours of a moving object Figure 2.

The SVD transformation is applied to each fragment, and the first singular value is chosen as the fragment descriptor. If we divide the frame into 5x5, then we need to calculate 25 matrices of a certain size. Increasing the number of fragments will lead to a decrease in the size of the input

[1]*International Workshop on Computational Intelligence, co-located with the IV International Scientific Symposium "Intelligent Solutions" (IntSol-2025), May 01-05, 2025, Kyiv-Uzhhorod, Ukraine*
sergii.mashtalir@nure.ua (S. Mashtalir); dmytro.lendel@uzhnu.edu.ua (D. Lendel)
0000-0002-0917-6622 (S. Mashtalir); 0000-0003-3971-1945 (D. Lendel)

matrices, but the number of SVD applications will increase. Considering the number of transformations, optimization of the calculation process comes to the fore.

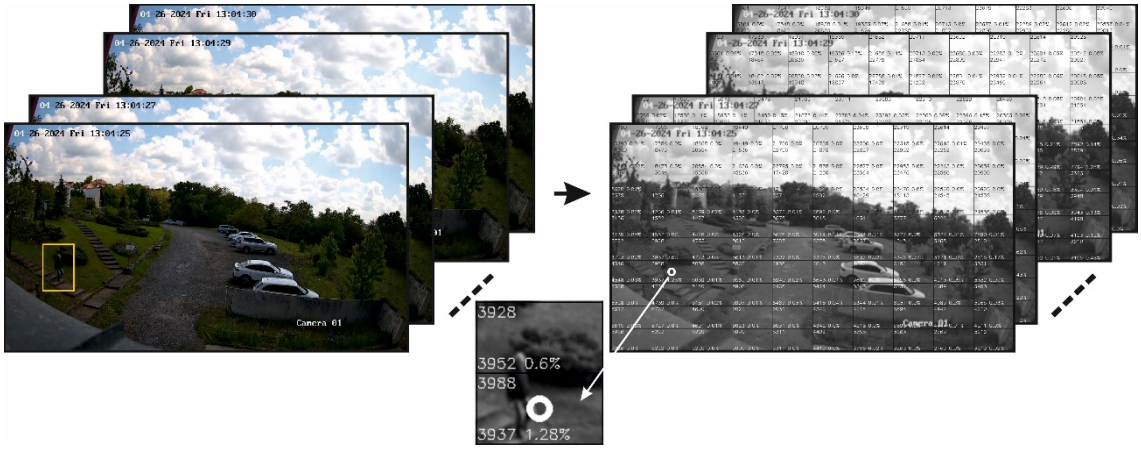

Figure 1: Motion detection. The result of frame-by-frame processing is a new video source in grayscale model with marked blocks with Ky-Fan norm value for each fragment

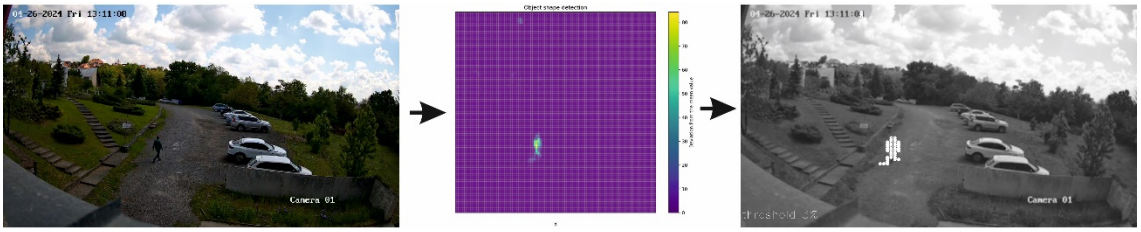

Figure 2: Fragment processing result. Man walking through parking. "Heat map" shows threshold frames. Object-detected frames marked by white spots.

Presently, with the advent of the big data and exascale computing revolutions, the availability of an efficient, scalable SVD implementation turns out to be an issue of crucial concern [9] (even when sometimes the aim is 'simply' to fit a huge amount of data into the distributed memory of the supercomputer, and then apply a parallel SVD). Typically, the SVD acts on a matrix that may be sparse or dense, tall-skinny or fat-shaped, well-conditioned or very ill-conditioned, to mention some frequent scenarios that are commonly linked to the specific fields mentioned above.

Due to the complexity of the SVD, many studies have focused on optimizing the transformation for various tasks. [10,11,12]. Since in our approach we only need to find the first singular value, we consider alternative approaches to the full SVD. In this study, we evaluate the accuracy and speed of alternative approaches depending on the number of frame division fragments. The aim of the study is to evaluate the following approaches in the context of video fragment analysis:

1. SVD
2. Incomplete SVD
3. UTV decomposition
4. ULV decomposition
5. Lanczos SVD
6. Power iteration
7. Randomized SVD

2. Singular value decomposition

2.1. Singular value decomposition step by step

The process of Singular Value Decomposition (SVD) [13] involves breaking down a matrix A into the form:

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

where U is an m×m complex unitary matrix, Σ is an m×n diagonal matrix with non-negative real numbers on the diagonal, and V is an n×n complex unitary matrix. If A is real, U and V can be guaranteed to be also real orthogonal matrices. The singular values (σ_i) describe the "energy" or importance of each corresponding dimension in the matrix. This computation allows us to retain the important singular values that the image requires while also releasing the values that are not as necessary in retaining the quality of the image. The singular values of an m × n matrix A are the square roots of the eigenvalues of the n × n matrix $A^T A$, which are typically organized by magnitude in decreasing order. Before we apply the SVD to image processing, we will first demonstrate the method using a small (2×3) matrix A:

$$A = \begin{bmatrix} 4 & 3 & 7 \\ 2 & 5 & 6 \end{bmatrix} \tag{2}$$

and then follow a step-by-step process to rewrite the matrix A in the separated form $U \Sigma V^t$:

$$A^T A = \begin{bmatrix} 4 & 3 & 7 \\ 2 & 5 & 6 \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 3 & 5 \\ 7 & 6 \end{bmatrix} = \begin{bmatrix} 20 & 22 & 40 \\ 22 & 34 & 51 \\ 40 & 51 & 85 \end{bmatrix}, A A^T = \begin{bmatrix} 74 & 65 \\ 65 & 65 \end{bmatrix} \tag{3}$$

Next step is to determine the eigenvalues of $A^T A$. In order to determine the eigenvalues of $A^T A$, we need to compute the determinant of the matrix $A^T A - \lambda I$. In general, we compute the determinant of a 3 × 3 matrix in the following way:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - d \begin{vmatrix} b & c \\ h & i \end{vmatrix} + g \begin{vmatrix} b & c \\ e & f \end{vmatrix} = a(ej - hf) + d(bi - hc) + d(bf - ec) \tag{4}$$

We could extend this computation to an n × n matrix as needed. For our example, we compute the determinant of $A^T A - \lambda I$ which is:

$$\begin{bmatrix} 20 & 22 & 40 \\ 22 & 34 & 51 \\ 40 & 51 & 85 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 20 - \lambda & 22 & 40 \\ 22 & 34 - \lambda & 51 \\ 40 & 51 & 85 - \lambda \end{bmatrix} \tag{5}$$

By setting this determinant equal to zero,

$$det \begin{bmatrix} 20 - \lambda & 22 & 40 \\ 22 & 34 - \lambda & 51 \\ 40 & 51 & 85 - \lambda \end{bmatrix} = 0 \tag{6}$$

we solve the characteristic equation for λ, and here we see that λ = 0, 4.3444, 134.6556. We reorder the eigenvalues in decreasing magnitude, so that: $\lambda_1 = 134.6556, \lambda_2 = 4.3444\ \lambda_3 = 0.0$. The singular values of σ are defined as the square roots of the eigenvalues:

$$\sigma_1 = \sqrt{134.6556} \approx 11.6041, \sigma_2 = \sqrt{4.344} \approx 2.0843, \sigma_3 = \sqrt{0} = 0 \tag{6}$$

To determine the matrix Σ, we list the non-zero singular values, σ_i, in decreasing magnitude down the main diagonal of Σ, where $\sigma_i = \sqrt{\lambda_i}$. Then, we add any additional rows and columns of zeros as needed to retain the original dimension of A in Σ. In our example, we have three singular values: 11.6041, 2.0843 and 0. We only need to retain the non-zero values, and hence, we form the matrix:

$$\Sigma = \begin{bmatrix} 11.6041 & 0 & 0 \\ 0 & 2.0843 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7}$$

Next, find eigenvectors (columns of U). The eigenvectors $u1$, $u2$ are determined from the equation where $\lambda_1 = 134.67$:

$$\left(A A^T - \lambda I\right) u = 0, \begin{bmatrix} 74 - 134.67 & 65 \\ 65 & 65 - 137.67 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{8}$$

We have two singular values in our example, and we use them to form the following vectors:

$$-60.67\, x1 + 65\, y1 = 0, x1 = \frac{65}{60.67}\, y1 \approx 1.0713\, y1 \tag{9}$$

$$y1 = 1, x1 \approx 1.0713, Normalize \sqrt{(1.0713)^2 + 1^2} \approx 1.4656 \tag{10}$$

$$u1 = \left(\frac{1.0713}{1.4656}, \frac{1}{1.4656}\right) = (0.7311, 0.6823) \tag{11}$$

Similarly for $\lambda_2 = 4.33$:

$$u2 = (-0.6823, 0.7311) \tag{12}$$

And the last matrix *U:*

$$U = \begin{bmatrix} 0.7311 & -0.6823 \\ 0.6823 & 0.7311 \end{bmatrix} \tag{13}$$

Next, we need find matrix *V*. The eigenvectors *v1,2,v3* are determined from the equation:

$$\left(A^T A - \lambda I\right) u = 0 \tag{14}$$

Solving the equation for each eigenvalue, we obtain the normalized eigenvectors:

$$V = \begin{bmatrix} 0.3696 & 0.6078 & -0.7029 \\ 0.4830 & -0.7719 & -0.4134 \\ 0.7938 & -0.1867 & 0.5788 \end{bmatrix} \quad (14)$$

Now, we understand the complexity of the SVD calculation. We can explore different calculating approaches since we are only interested in the first singular value. The simplest solution would be to stop the calculation as soon as the first singular value is found. Moreover, we do not need the matrices U and V. This will be an incomplete SVD. Numpy library is the proposed parameter "full_matrices" in linalg.svd function. We can set full_matrices =False and hope to find singular value without full calculation. However, if the dimension of the input matrix is large, this approach may not give the desired result.

2.2. UTV, ULV

The SVD method is robust but does not necessarily scale well to larger matrices. Thus, to use SVD in a practical sense with large datasets, we needed a faster algorithm that finds the same dominant patterns as regular SVD, but with only a fraction of the computational cost. In search of an alternative approach it would be logical to pay attention on UTV and ULV decomposition. UTV decomposition [14] is an alternative to SVD that factorizes a matrix A:

$$A = UTV^t \quad (15)$$

Where U and V are orthogonal matrices (similar to SVD). T is an upper triangular matrix, unlike the diagonal Σ in SVD. This method is often used as a more efficient alternative to SVD in applications where exact singular values are not required, but a good approximation is sufficient. UTV factorization begins by applying a series of Householder reflections or randomized projections to transform the given matrix A into an upper triangular or upper trapezoidal matrix T while preserving its dominant numerical properties. This transformation ensures that most of the essential information in A is retained while simplifying its structure. The process involves computing orthogonal matrices U and V at capture the column and row spaces of A, respectively, mapping it into the triangular form. Once the factorization is complete, the result is a decomposition A $= UTV^t$, where T serves as a computationally efficient approximation of the singular structure of A, similar to Σ in SVD but with a triangular shape.

ULV factorization [15] transforms the given matrix A into a lower triangular or block lower triangular matrix L while preserving its essential numerical properties. This is achieved through a series of Householder reflections or Givens rotations, which iteratively reduce A into its structured form while maintaining orthogonality. The decomposition also produces orthogonal matrices U and V that encode the column and row transformations, respectively, mapping A into its triangular form. The result is the factorization:

$$A = ULV^t \quad (16)$$

which serves as a computationally efficient alternative to SVD, particularly in cases where structured rank-revealing decompositions are beneficial.

2.3. Lanczos SVD

ULV and UTV are providing full SVD, decomposition and Lanczos SVD [16] is not strictly an optimization of either ULV or UTV, but rather an iterative alternative to SVD that shares similarities with both approaches. UTV factorization transforms a matrix A into an upper triangular matrix T using unitary transformations, maintaining full orthogonality in U and V. Lanczos SVD, on the other hand, iteratively reduces A to a bidiagonal form instead of a strictly triangular one. Both methods use orthogonal transformations, but Lanczos is focused on extracting

dominant singular values efficiently, whereas UTV is more general in preserving an upper-triangular structure. Lanczos SVD, in contrast, uses an iterative Krylov subspace approach to build a bidiagonal matrix rather than a strict lower triangular matrix. Given that full SVD is computationally expensive, we now turn to Power SVD and Randomized SVD, which efficiently approximate dominant singular values without performing a complete decomposition.

2.4. Power iteration

Power iteration [17] starts with b_0, which might be a random vector. At every iteration this vector is updated using following rule:

$$b_{k+1} = \frac{A\,b_k}{\left\| A\,b_k \right\|} \tag{17}$$

First, we multiply b_0 to compute the matrix-vector product A ($A\,b_k$) and divide the result with the norm ($\left\| A\,b_k \right\|$). We will continue until the result has converged, in other words, when the difference between iterations is below a defined threshold. The power method has a few assumptions: b_0 has a non-zero component in the direction of an eigenvector associated with the dominant eigenvalue. Initializing b_0 randomly minimizes the possibility that this assumption is not fulfilled, and matrix A has a dominant eigenvalue that must be greater in magnitude than other eigenvalues. These assumptions guarantee that the algorithm converges to a reasonable result. The smaller the difference between the dominant eigenvalue and the second eigenvalue, the longer it might take to converge.

2.5. Randomized SVD

A promising approach for efficient singular value decomposition is Randomized SVD [18], which uses a random projection to approximate the column space of a given matrix, reducing it to a target rank k before applying SVD. This method retains most of the important information while significantly reducing the computational cost. Given an $m \times n$ matrix A, we choose a target rank $k < m$, which determines the dimensionality of the subspace for which we will compute SVD. We first initialize a random matrix P of size $n \times k$ and then transform our original matrix A by computing the matrix product:

$$Z = AP \tag{18}$$

This reduces the column space of A while preserving its dominant features, decreasing the dimensionality from n to k (matching the row dimension of P). However, with high probability, Z will still retain the most significant column space features of A. QR factorization of Z provides an orthonormal basis Q, which captures the dominant column space of A:

$$Q, R \leftarrow QR\,Factorization(Z) \tag{19}$$

Next, we project A onto the subspace defined by Q:

$$Y = Q^t A \tag{20}$$

Now, Y is an k by n matrix and we are computing SVD on a matrix with a column size of k rather than m, which should be much less computational cost if we choose a small k.

$$U_Y \sum_Y V_Y^{\dot{c}} \leftarrow SVD(Y) \tag{21}$$

Finally, we keep Σ and V from our SVD of Y, and then obtain our final U matrix by:

$$U = Q\,U_Y \tag{22}$$

This step extends U_y back to the original column dimension of A.

We are ready to compare the approaches, evaluate their accuracy in calculating the first singular value and speed in the fragment analysis contest.

3. Accuracy of finding the first singular value

In this section, we will consider the results produced by the developed application. Our experiment used a surveillance camera source Figure 3. Codec is h264, frame size is 1280 x 720. To visualize the results of utilizing the Ky Fan norm for video analysis, a Python 3.10.11 application was developed and executed on a system equipped with an Intel Core i5 processor, 16 GB of RAM, and running the Windows operating system. The application relies on two open-source libraries licensed under Apache License: OpenCV version 4.10.0 and NumPy version 2.2.1. Frames are converted from RGB to a grayscale model. Each frame is divided into smaller fragments through a grid-based segmentation technique.

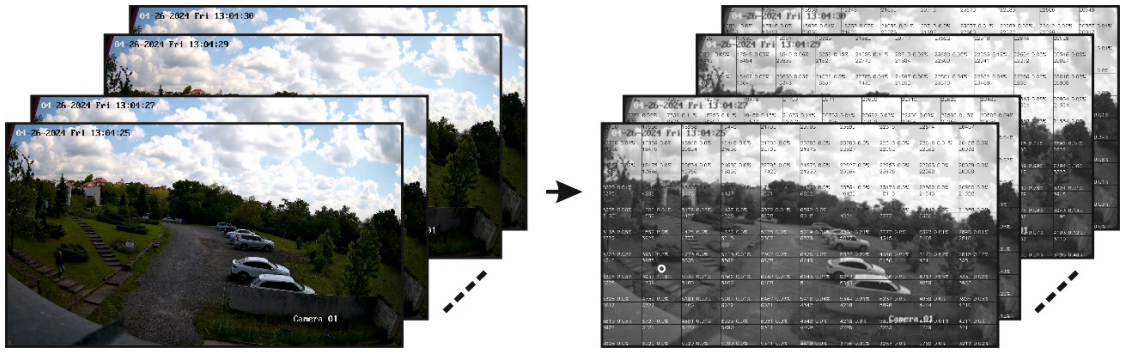

Figure 3: Video source as a sequence of frames. The result of frame-by-frame processing is a new video source in grayscale model with marked blocks with Ky-Fan norm value for each fragment

3.1. Accuracy of finding the first singular value

Before evaluating the speed of the selected approaches, we need to check the accuracy of finding the first singular value for different fragment sizes. The results are presented in Table 1. All approaches: SVD, incomplete SVD, Power iteration, UTL, Lanczos SVD and Randomized SVD show same singular values for different fragment sizes. The ULV approach showed no significant deviation. This deviation may be because ULV does not explicitly compute singular values like SVD-based methods, so minor deviations are expected. If high precision for singular values is required, ULV may not be the best choice—methods like Power SVD, Lanczos SVD, and Randomized SVD are preferable. However, if the goal is structured decomposition or efficient matrix transformation, ULV remains a valuable alternative.

Table 1
First singular value calculation for different fragment sizes

	All approaches exclude ULV	ULV
5x5 (144x256)	31541.411066	31277.636507
10x10 (72x128)	13581.419741	13419.979837
20x20 (36x64)	5857.278687	5767.989675

50x50 (14x25)	575.184362	572.544123

3.2. Comparison of fragments processing time using different methods

We compared the average calculation time for fragments of different sizes. The results are presented on Figure 4. Methods with full decomposition showed the longest time for fragments of high dimension. Methods optimized for finding the first singular value showed the best time.

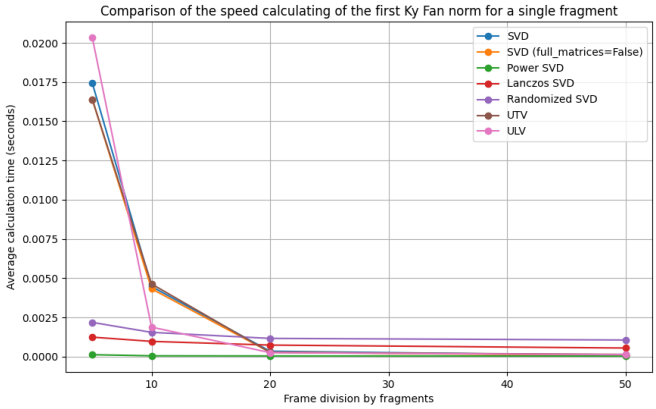

Figure 4: Comparison of the speed calculation of the first Ky Fan norm value (first singular value) for a single fragment

The best optimized approach Power SVD for different fragment sizes. The best result is marked in blue, the worst in red in Table 2.

Table 2
Time (seconds) calculation for different fragment sizes

	SVD (complete decomposition)	SVD (incomplete decomposition)	Power SVD	Lanczos SVD	Randomized SVD	UTV	ULV
5x5 (144x256)	0.017440	0.016400	**0.000120**	0.001240	0.002181	0.016400	**0.020341**
10x10 (72x128)	0.004465	0.004320	**0.000050**	0.000970	0.001545	**0.004630**	0.001880
20x20 (36x64)	0.000342	0.000280	**0.000043**	0.000734	**0.001160**	0.000270	0.000253
50x50 (14x25)	0.000074	0.000066	**0.000036**	0.000546	**0.001058**	0.000137	0.000122

The total processing time of the entire frame using different methods is presented in Figure 5. If the frame is divided into 5x5, then the transformation should be applied to 25 matrices of size 144x256, with a division into 50x50 there will be 250 matrices of size 14x25. The speed is chosen as average, therefore the processing speed of each fragment depends on the sparsity of the matrix. Randomize SVD performed the worst on small matrices. This approach was designed for high-dimensional matrices and is not efficient for low-dimensional matrices. Approaches using full decomposition are quite slow on both small and large matrices.

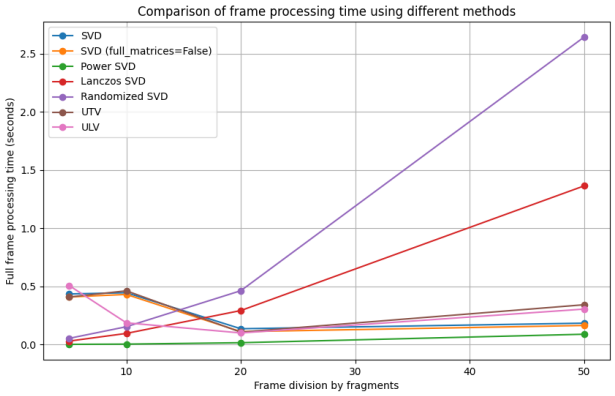

Figure 5: Comparison of the middle-speed calculation of the first Ky Fan norm value for the frame

The best optimized approach Power SVD for different fragment sizes. The best result is marked in blue, the worst in red in Table 3.

Table 3
Comparison of the middle-time (seconds) calculation of the first Ky Fan norm value for the frame

	SVD (complete decomposition)	SVD (incomplete decomposition)	Power SVD	Lanczos SVD	Randomized SVD	UTV	ULV
5x5 (144x256)	0.436000	0.409998	**0.002999**	0.031003	0.054517	0.410002	**0.508518**
10x10 (72x128)	0.446510	0.431996	**0.004992**	0.097003	0.154527	**0.463000**	0.187995
20x20 (36x64)	0.137000	0.112020	**0.017002**	0.293519	**0.464002**	0.107834	0.101002
50x50 (14x25)	0.185009	0.165095	**0.090012**	1.365031	**2.644217**	0.343357	0.306030

Conclusions

Video fragment analysis involves dividing the frame into geometric regions of different sizes depending on the task. For motion detection, dividing the frame into 5x5 or 10x10 is enough to determine the area of interest. That is, find the part of the frame where the movement occurs. Of course, the object's size must be smaller than the size of the fragment. To determine the contours of the object, we must increase the scale. And as a result, we will get several high-order matrices or many low-order matrices. It should also be noted that on fragments of small sizes, but with a large number of fragments themselves, the worst results were shown by Randomized SVD. At the same time, with the increase in fragment sizes, the performance of the UTV and ULV algorithms has deteriorated, so they are the worst solution for motion detection. For practical real-time video analysis, the Power SVD is best suited. For offline analysis, all algorithms will be pretty effective.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] J. Gallier, J. Quaintance Linear Algebra and Optimization with Applications to Machine Learning - Volume I, World Scientific Publishing Co Pte Ltd, 2020. doi:10.1142/11446

[2] Q. Guo, C. Zhang, Y. Zhang, H. Liu, An Efficient SVD-Based Method for Image Denoising, IEEE Trans. Circuits Syst. Video Technol. 26.5 (2016) 868–880. doi:10.1109/tcsvt.2015.2416631.

[3] M. Motylinski, A. J. Plater, J. E. Higham, Re-scaling images using a SVD-based approach, Signal, Image Video Process. 19.3 (2025). doi:10.1007/s11760-025-03825-1.

[4] H. R. Swathi, S. Sohini, Surbhi, G. Gopichand, Image compression using singular value decomposition, IOP Conf. Ser. 263 (2017) 042082. doi:10.1088/1757-899x/263/4/042082.

[5] S. Mashtalir, D. Lendel, Video pre-motion detection by fragment processing, 12th International Scientific and Practical Conference "Information Control Systems and Technologies", 2024, https://ceur-ws.org/Vol-3790/paper30.pdf

[6] M. Koliada, KY FAN NORM APPLICATION FOR VIDEO SEGMENTATION, Her. Adv. Inf. Technol. 3.1 (2020) 345–351. doi:10.15276/hait01.2020.1.

[7] S. V. Mashtalir, D. P. Lendel, Video fragment processing by Ky Fan norm, Appl. Asp. Inf. Technol. 7.1 (2024) 59–68. doi:10.15276/aait.07.2024.5.

[8] S. V. Mashtalir, D. P. Lendel, Moving object shape detection by fragment processing, Her. Adv. Inf. Technol. 7.4 (2024) 414–423. doi:10.15276/hait.07.2024.30.

[9] M. De Castro-Sánchez, J. A. Moríñigo, F. Terragni, R. Mayo-García, Analysis of the SVD Scaling on Large Sparse Matrices, in: 2024 Winter Simulation Conference (WSC), IEEE, 2024, pp. 2523–2534. doi:10.1109/wsc63780.2024.10838971.

[10] D. Keyes, H. Ltaief, Y. Nakatsukasa, D. Sukkari, High-Performance SVD Partial Spectrum Computation, in: SC '23: International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, New York, NY, USA, 2023. doi:10.1145/3581784.3607109.

[11] X. Feng, W. Yu, Y. Xie, J. Tang, Algorithm xxx: Faster Randomized SVD with Dynamic Shifts, ACM Trans. Math. Softw. (2024). doi:10.1145/3660629.

[12] A. Noorizadegan, C. S. Chen, R. Cavoretto, A. De Rossi, Efficient truncated randomized SVD for mesh-free kernel methods, Comput. & Math. With Appl. 164 (2024) 12–20. doi:10.1016/j.camwa.2024.03.021.

[13] R. Szeliski Computer Vision: Algorithms and Applications, Springer, 2010.

[14] G. A. Watson, D. F. Griffiths, Numerical Analysis 1993, Taylor & Francis Group, 2020. doi: 10.1201/9781003062257

[15] M. Vandecappelle, L. D. Lathauwer, Updating the multilinear UTV decomposition, IEEE Trans. Signal Process. (2022) 1–15. doi:10.1109/tsp.2022.3187814.

[16] T. Chen, The Lanczos algorithm for matrix functions: a handbook for scientists. , 2024, doi:10.48550/arXiv.2410.11090

[17] Y. Nakatsukasa, N. J. Higham, Stable and Efficient Spectral Divide and Conquer Algorithms for the Symmetric Eigenvalue Decomposition and the SVD, SIAM J. Sci. Comput. 35.3 (2013) A1325—A1349. doi:10.1137/120876605.

[18] D. Janekovic, D. Bojanjac, Randomized Algorithms for Singular Value Decomposition: Implementation and Application Perspective, in: 2021 International Symposium ELMAR, IEEE, 2021. doi:10.1109/elmar52657.2021.9550979.