A Real-Time Application of Singular Spectrum Analysis to Object Tracking with SIFT

Ali Ozturk
Department of Mechatronics
Vocational School of Technical Sciences
Mus Alparslan University
Mus, Turkey
a.ozturk@alparslan.edu.tr

Ibrahim Cayiroglu
Department of Mechatronics
Faculty of Engineering
Karabuk University
Karabuk, Turkey
icayiroglu@karabuk.edu.tr

Received: 27 April 2022 | Revised: 31 May 2022 | Accepted: 1 June 2022

Abstract- This study combined SIFT and SSA to propose a novel algorithm for real-time object tracking. The proposed algorithm utilizes an intermediate fixed-size buffer and a modified SSA algorithm. Since the complete reconstruction step of the SSA algorithm was unnecessary, it was considerably simplified. In addition, the execution time of a Matlab implementation of the SSA algorithm was compared with a respective C++ implementation. Moreover, the performance of the two different matching algorithms in the detection, the FlannBasedMatcher and Brute-Force matcher algorithms of the OpenCV library, was compared.

Keywords-object tracking; object detection; computer vision; SIFT; SSA

I. INTRODUCTION

Object tracking or visual tracking has been employed, studied, and commercialized in a broad spectrum of applications such as Simultaneous Localization And Mapping (SLAM), traffic surveillance, pedestrian counting, hand gesture recognition, tracking the path of protein stress granules in cells, and video compression [1]. Object tracking can be defined as dynamic object detection [2]. The main problems of object tracking are common with object detection, such as occlusion, clutter in the background, viewpoint changes due to affine transformation such as translation and rotation, scale changes (zoom in and out), and photometric deformation such as distortion, blur, and illumination changes. The most important challenge is still the speed or real-time performance of the online tracker, such as the tracker used in visual servoing in robotics [3]. On the other hand, object tracking has some problems of its own. An optimal algorithm for visual tracking can vary depending on the camera's position, the number of cameras, and the number of moving objects. In the case of a stationary camera, background removal can simplify the task by eliminating the stationary pixels [4]. If the number of moving objects and the target object increase, then the task becomes complicated. Object tracking consists of two important steps: detecting the object of interest and tracking it in the following frames [5]. Object detection can be defined as template matching in which a template or a model is detected within a scene. Typically, a template is the picture of an object of interest. Detection can be defined as enclosing a given template in the scene in a rectangle. In object tracking, the basic task is to track the object as long as it exists in the scene.

Scale Invariant Feature Transform (SIFT) is one of the most robust local descriptors which can be used in object tracking [6-8]. It has been utilized in a broad spectrum of applications, such as face recognition, gender recognition [9], and smoke detection [10]. It is a good candidate to tackle the problems of tracking, such as translation, rotation, occlusion, and scale. A method was proposed in [11] to track objects by combining SIFT with mean shift. In [12], SIFT was utilized with Kanade-Lucas-Tomasi Tracker (KLT) [13] for UAV-based applications. In this approach, the object was detected with SIFT, and its location was computed with KLT. Since the algorithms are not perfectly robust to all image transformations and deformations, the object of interest can be lost or detected with low accuracy during any object tracking task. In such cases, the pose can be recovered or improved by using filters or smoothing. Singular Spectrum Analysis (SSA) is a method of time series analysis based on multivariate statistics. It is also known as the principal component analysis of time series [14]. SSA is considered to be a non-parametric (model-free) filtering technique. However, it can also be used for modeling [15]. It has been mainly used in applications that require post-processing [16, 17]. SSA was combined with the Kalman filter to smooth real-time positioning data obtained from the Global Navigation Satellite System (GNSS) in [18]. However, there is no description of how SSA was implemented in real-time. A rank-one eigenvector update was proposed in a recursive framework in [19] for the identification of structural damage detection in real-time by utilizing First-Order Perturbation (FOP) for the Henkel matrix.

This paper proposes an object-tracking algorithm utilizing SIFT and SSA. To the best of our knowledge, this is the first study that employs SIFT and SSA in object tracking. As SSA always requires a batch of data, its real-time implementation was carried out using a fixed-sized buffer. Since the real-time implementation of SAA does not require a full reconstruction step by diagonal averaging, the reconstruction algorithm was considerably simplified.

Corresponding author: Ali Ozturk

II. Method

This section investigates the SSA algorithm, provides the real-time implementation, describes object tracking with a template matching algorithm, and provides the details of the experiments.

A. Singular Spectrum Analysis

The SSA algorithm can be summarized in four steps: embedding, Singular Value Decomposition (SVD), grouping, and finally, reconstruction. In the embedding step, a Henkel matrix was constructed from the input signal. Each column of the Henkel matrix is a portion of the signal obtained by windowing with a selected window length. This process is also called a caterpillar since it behaves as such. If the length of the raw signal is *n* and the window length is *l*, the size of the Henkel matrix is *l*×*k*, where *l* is the rows, *k* is the columns, and *k*=*n*-*l*+*1*. Figure 1 shows an example of the embedding step. In this example, a Henkel matrix is created from an input signal. The size of the signal is 10, and the window length is 4. Therefore, the size of the resulting matrix is 7×4.

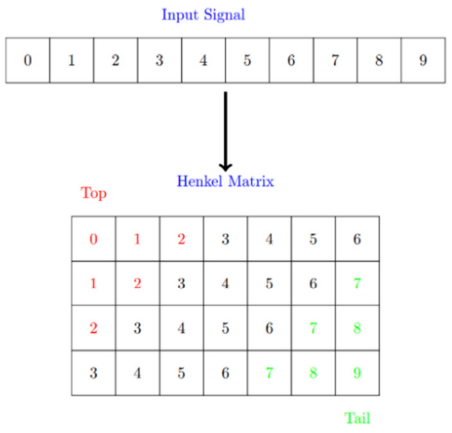

Fig. 1. An example of the embedding step.

In the SVD step, the singular value decomposition of the trajectory matrix is computed. If *X* is the Henkel matrix, then the trajectory matrix is XX^T, where X^T is the transpose of *X*. The Henkel matrix can be written as a sum of different and independent matrices with the aid of SVD. This decomposition can be stated as follows:

$$X = X_1 + X_2 + \cdots + X_m \quad (1)$$

where each X_i is called an elementary matrix. Each elementary matrix X_i is calculated as follows:

$$X_i = \sqrt{\lambda_i} U_i V_i^T, i = 1, \cdots, m \quad (2)$$

where U_i denotes the eigenvectors of SVD, and each vector V_i is calculated as follows:

$$V_i = X^T \frac{U_i}{\sqrt{\lambda_i}}, i = 1, \cdots m \quad (3)$$

Each elementary matrix X_i corresponds to one of the eigenvalues of SVD. It should be noted that:

$$\|X_i\|^2 = \lambda_i, \quad i = 1, \cdots, m \quad (4)$$

The contribution of each elementary matrix to *X* is proportional to its corresponding eigenvalue λ_i. The eigenvalues are sorted in decreasing order. In SSA, elementary matrices corresponding to small eigenvalues are considered to be the noise portion of the signal. This portion is excluded in the grouping step:

$$\hat{X} = \sum_{i=1}^{r} X_i \quad (5)$$

where *r*<*m*. The value of *r* is decided by looking at the ratio of the sum of the first *r* eigenvalues to the sum of all eigenvalues. If the value of this ratio is greater than a certain threshold, the sum of the first *r* elementary matrices (5) is considered to be an optimal approximation of the Henkel matrix *X*. Finally, in reconstruction, the optimal approximation of the Henkel Matrix, $\hat{X}$ is converted to a time series. The reconstruction operation should be considered to be the inverse operation of the construction of the Henkel matrix from the raw signal. This inverse operation is performed by diagonal averaging on the approximated Henkel matrix (Figure 1). The diagonal averaging is realized in three steps: top diagonalization, center, diagonalization, and tail diagonalization on the approximated Henkel matrix as follows:

$$b_t = \begin{cases} \frac{1}{t+1} \sum_{i=1}^{t+1} a_{i,t-i+2}, & if\ 0 \le t < L^* - 1 \\ \frac{1}{L^*} \sum_{i=1}^{L^*} a_{i,t-i+2}, & if\ L^* - 1 \le t < K^* \\ \frac{1}{N-t} \sum_{i=t-K^*+2}^{N-K^*+1} a_{i,t-i+2}, & if\ K^* \le t < N \end{cases} \quad (6)$$

where $a_{i,j}$, $1 \le i \le L$, $1 \le j \le K$ denote entries of the approximated Henkel matrix and, b_t, $0 \le t < N$, are entries of the reconstructed time series.

Two parameters affect the result of SSA: the window length *L* and the reconstruction parameter *r*. The last one is determined by the eigenvalues as explained above. In general, the first or at least the sum of two eigenvalues constitutes the largest portion of the sum of all eigenvalues. However, there is no rule on how to select *L*. The choice of *L* affects the result of the filter. If *L* increases, the smoothing effect of the filter increases. This value can be determined by test and trial.

B. Real-Time Implementation of SSA

The real-time implementation of SSA was conducted by using a fixed-size buffer (Figure 2). At first, this buffer is filled with data. The size of the buffer is determined beforehand. So, this method requires a time delay to start filtering. Once the buffer is full, the filtering operation starts. As new frames arrive, the buffer is updated. However, for the real-time implementation of SSA, there is no need for full reconstruction by diagonal averaging. Since the last frame is written on the last element of the buffer, it is sufficient to recover only the last element of the approximated trajectory matrix, as follows:

$$f = \hat{X}(l-1, k-1) \quad (7)$$

where *f* is the data filtered and the size of the approximated Henkel matrix $\hat{X}$ is $l \times k$. It is assumed that the first index starts from zero. Thus, using (7), the reconstruction algorithm is significantly simplified by omitting diagonal averaging, compared to (6). This method is called Real-Time SSA (RT-SSA).

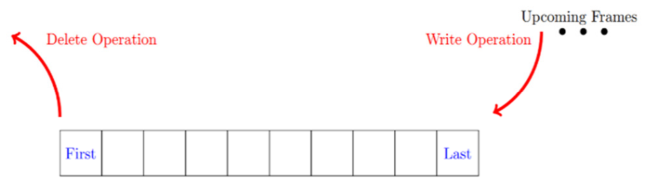

Fig. 2. Fixed-size buffer used for real-time implementation of SSA.

C. Object Tracking with Template Matching

Template matching is the task of detecting a given template within a given scene. Typically, a template is the picture of an object of interest. Detection can be defined as enclosing a given template in the scene in a rectangle. Figures 3 and 4 show the template used in this study and the detection task respectively.

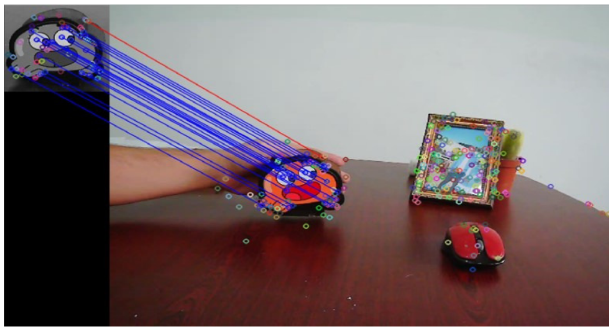

Fig. 3. A matched sample frame from the video.

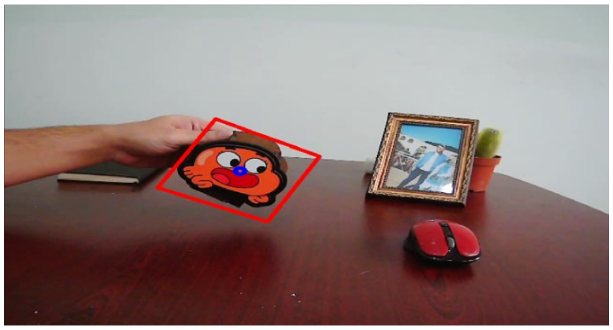

Fig. 4. A detection example from the video.

This study used SIFT for object detection. SIFT is a local descriptor, typically invariant to local occlusion [20]. SIFT has two main parts: keypoint detection and feature description. Keypoints (or salient points) are detected on the Gaussian scale-space pyramid [21, 22]. This kind of keypoint detection makes the algorithm scale (zoom in and out) invariant. A dominant orientation of each keypoint is computed using the Histogram of Oriented Gradients (HOG). This computation makes the algorithm orientation invariant. Finally, a feature vector for each keypoint is calculated using HOG [23]. The normalization of the feature vector makes it invariant to changes in illumination. In summary, SIFT is invariant to translation, rotation, scale change (zoom in and out), partial occlusion, and illumination change. The runtime performance of SIFT mainly depends on the size of the image and the number of the extracted keypoints. In this study, SIFT was implemented using OpenCV 4.5.5.

Once the keypoints and features are computed both on the template and scene, the following task is matching. For the matching algorithm, the FlannBasedMatcher of OpenCV can be utilized. The Fast Library for Approximated Nearest Neighbor (FLANN) speeds up the matching task by using the k-nearest neighbor [24]. If the number of the extracted keypoints is low, the Brute-Force matcher of OpenCV can alternatively be utilized. Figure 3 shows the SIFT keypoints on both the template and the scene, as well as the matched keypoints. Once the matching pairs between the template and the scene are detected, the template's corners should be projected on the scene. A homography matrix is needed for this projection. The OpenCV findHomography function was used to obtain the homography matrix using the RANSAC algorithm. Finally, the OpenCV perspetiveTransform function provides four corners of the rectangular polygon that encloses the template in the scene.

D. Experimental Setup

An Everest SC-HD03 webcam was used to capture the video. In the video, the template (a penholder) was moved by hand. It was mobilized, moved away from the camera, zoomed in, rotated, and twisted in different directions. During this process, the pixel coordinates of the center point of the detected four corners of the template in the scene were considered to be the pose of the object. This center point is shown by a blue circle in Figure 4. The video consists of 1439 frames. The size of each frame was 640×480.

A C/C++ code was written for SSA and RT-SSA, using the Eigen library [25], with floating-point arithmetic.The C/C++ code is available in [26]. On the other hand, Matlab utilizes double-point arithmetic by default. The C/C++ code was compiled with GCC 9.4.0 on Ubuntu 20.04 LTS and run on an Intel i9-10850K CPU @3.60GHZ with 16GB RAM. The parameters, notations, and abbreviations used in the results are shown in Table I. As was observed, the first eigenvalue always constituted more than 90% of all. So, the reconstruction parameter *r* of SSA was taken as 1 for all tests.

TABLE I. NOTATIONS AND ABBREVIATIONS USED IN RESULTS

Notation/ Abbreviation	Explanation
Pose	Position of the object (*x* and *y* coordinates)
FLANN	FlannBasedMatcher of OpenCV
Brute-Force	Brute-Force matcher of OpenCV
Filter-30-3	RT-SSA with a buffer of size 30, and window length of 3
Filter-30-9	RT-SSA with a buffer of size 30, and window length of 9

III. RESULTS

This section, initially, presents the results of template matching according to two different matching methodologies, FLANN and Brute-Force. Then, the results of the execution time of SAA are provided. Finally, the results of the application of real-time SAA to object tracking are summarized.

A. Template Matching

Figure 5 shows the results of FLANN and Brute-Force matching. As shown, FLANN lost the pose of the object in frame 562. However, the Brute-Force graph shows no loss at any frame. The average number of keypoints on the scene is 67, which is considered low. Therefore, it is more feasible to use the Brute-Force matcher. FLANN should be preferred when the number of keypoints is high. The average matching time with the Brute-Force matcher was about 3ms, while the average detection time of SIFT was about 33ms.

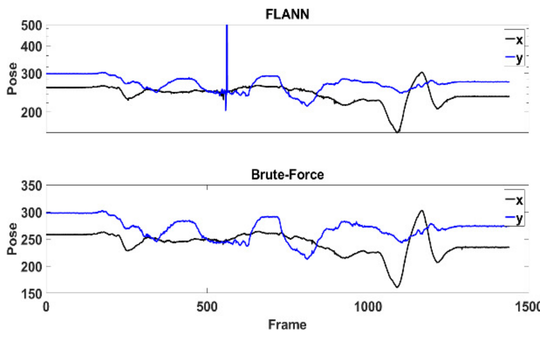

Fig. 5. Results of the FlannBasedMatcher and Brute-Force matching algorithms.

B. *Execution Time of SSA*

Figure 6 shows the speed of the C++ and Matlab SSA implementations. As shown, when the data length is 1K, the C++ implementation of SSA reaches 1000FPS. This speed meets the requirements of many real-time applications.

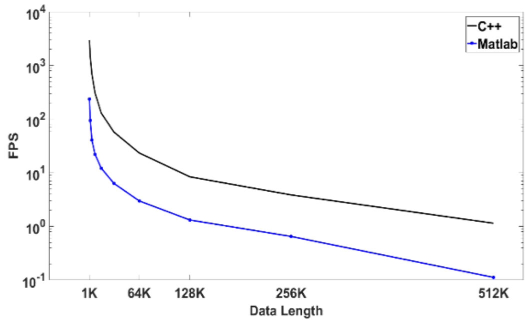

Fig. 6. Speed of C++ and Matlab implementations of SSA algorithm for data length (1K=1024). Execution time is expressed in FPS.

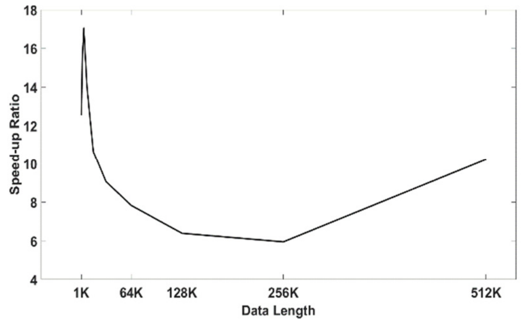

Fig. 7. Speed-up ratio of a C++ over a Matlab implementation of SSA.

Figure 7 shows the speed-up ratio of a C++ over the Matlab implementation of SSA. The C++ implementation of SSA is up to 18 times faster than Matlab. However, Matlab code can still meet the speed requirements of many applications.

C. *Application of SSA to Real-time Object Tracking*

Figures 8 and 9 show the x and y coordinates of the pose of the object when it is filtered with the real-time SSA. Figures 10-12 show the scaled points A, B, and C marked in Figure 8. These Figures show that as the window length increases, the filter smoothing effect increases, and fluctuations smooth out. However, when the object changes its directions (Figure 10), the filter may considerably move away from the actual pose because of over smoothing. Therefore, the extensive parameters of the filter can be avoided.

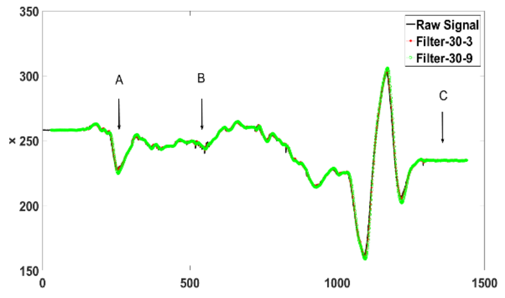

Fig. 8. Application of real-time SSA to object's x coordinate obtained with Brute Force matcher.

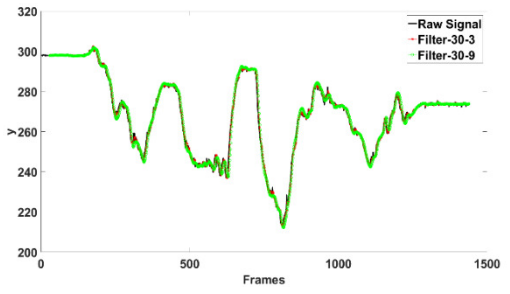

Fig. 9. Application of real-time SSA to object's y coordinate obtained with Brute Force matcher.

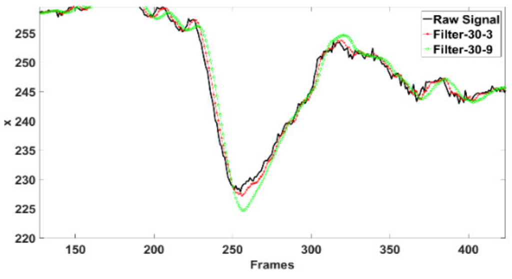

Fig. 10. Zoom in point A of Figure 8.

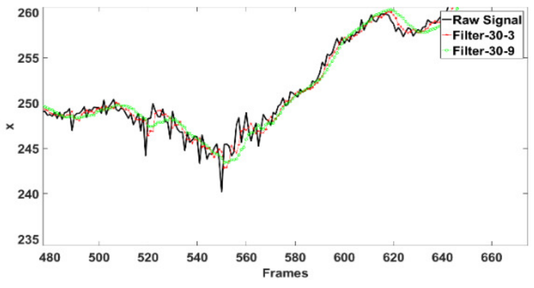

Fig. 11. Zoom in point B of Figure 8.

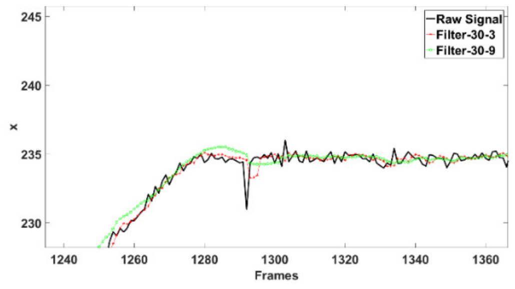

Fig. 12. Zoom in point C of Figure 8.

IV. CONCLUSION

SIFT and SSA are two different methods that have become popular in the last two decades. SIFT is a local object detection algorithm. On the other hand, SSA is a filtering technique that utilizes some methods of multivariate statics, such as PCA. This study combined SIFT with SSA for real-time object tracking. The main contributions of this paper can be summarized as follows:

- The results of two different feature matching methods, FlannBasedMatcher and Brute-Force matcher, were compared.
- SIFT and SSA were used in a real-time object tracking application for the first time.
- A C++ implementation of SSA was coded using the Eigen library.
- The performance (execution time) of C++ and Matlab implementations of SSA were compared.
- A novel method for the real-time implementation of SSA was presented, simplifying the reconstruction step.

Performance comparison showed that the classic SSA algorithm has enough speed for many real-time implementations and that the C/C++ code was faster than Matlab. As the proposed real-time method did not require full reconstruction, it simplifies the SSA reconstruction step. Moreover, instead of updating the raw data buffer, the Henkel matrix can be updated by omitting the embedding step of SSA, which could be future work. On the other hand, it was also observed that the performance of the object detection algorithm heavily depends on the matching algorithm. It was concluded that if the number of keypoints is low, then the Brute-Force matcher should be preferred instead of the FlannBasedMatcher.

REFERENCES

[1] M. Y. Abbass, K.-C. Kwon, N. Kim, S. A. Abdelwahab, F. E. A. El-Samie, and A. A. M. Khalaf, "A survey on online learning for visual tracking," *The Visual Computer*, vol. 37, no. 5, pp. 993–1014, May 2021, https://doi.org/10.1007/s00371-020-01848-y.

[2] M. Haris *et al.*, "Recognition and Tracking of Objects in a Clustered Remote Scene Environment," *Computers, Materials & Continua*, vol. 70, no. 1, Sep. 2021, Art. no. 1699, https://doi.org/10.32604/cmc.2022.019572.

[3] Z. Machkour, D. Ortiz-Arroyo, and P. Durdevic, "Classical and Deep Learning based Visual Servoing Systems: a Survey on State of the Art," *Journal of Intelligent & Robotic Systems*, vol. 104, no. 1, Dec. 2021, Art. no. 11, https://doi.org/10.1007/s10846-021-01540-w.

[4] J. Zuo, Z. Jia, J. Yang, and N. Kasabov, "Moving Target Detection Based on Improved Gaussian Mixture Background Subtraction in Video Images," *IEEE Access*, vol. 7, pp. 152612–152623, 2019, https://doi.org/10.1109/ACCESS.2019.2946230.

[5] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, Sep. 2006, Art. no. 13-es, https://doi.org/10.1145/1177352.1177355.

[6] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Kerkyra, Greece, Sep. 1999, vol. 2, pp. 1150–1157, https://doi.org/10.1109/ICCV.1999.790410.

[7] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[8] I. Rey Otero, "Anatomy of the SIFT Method," *Image Processing On Line*, vol. 4, pp. 370–396, Dec. 2014, https://doi.org/10.5201/ipol.2014.82.

[9] H. Alamri, E. Alshanbari, S. Alotaibi, and M. Alghamdi, "Face Recognition and Gender Detection Using SIFT Feature Extraction, LBPH, and SVM," *Engineering, Technology & Applied Science Research*, vol. 12, no. 2, pp. 8296–8299, Apr. 2022, https://doi.org/10.48084/etasr.4735.

[10] P. Matlani and M. Shrivastava, "An Efficient Algorithm Proposed For Smoke Detection in Video Using Hybrid Feature Selection Techniques," *Engineering, Technology & Applied Science Research*, vol. 9, no. 2, pp. 3939–3944, Apr. 2019, https://doi.org/10.48084/etasr.2571.

[11] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using SIFT features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, Mar. 2009, https://doi.org/10.1016/j.cviu.2008.08.006.

[12] F. Jabar, S. Farokhi, and U. U. Sheikh, "Object tracking using SIFT and KLT tracker for UAV-based applications," in *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, Langkawi, Malaysia, Jul. 2015, pp. 65–68, https://doi.org/10.1109/IRIS.2015.7451588.

[13] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, San Francisco, CA, USA, May 1981, pp. 674–679.

[14] P. More and P. Mishra, "Enhanced-PCA based Dimensionality Reduction and Feature Selection for Real-Time Network Threat Detection," *Engineering, Technology & Applied Science Research*, vol. 10, no. 5, pp. 6270–6275, Oct. 2020, https://doi.org/10.48084/etasr.3801.

[15] N. Golyandina, "Particularities and commonalities of singular spectrum analysis as a method of time series analysis and signal processing," *WIREs Computational Statistics*, vol. 12, no. 4, 2020, Art. no. e1487, https://doi.org/10.1002/wics.1487.

[16] F. J. Alonso, J. M. D. Castillo, and P. Pintado, "Application of singular spectrum analysis to the smoothing of raw kinematic signals," *Journal of Biomechanics*, vol. 38, no. 5, pp. 1085–1092, May 2005, https://doi.org/10.1016/j.jbiomech.2004.05.031.

[17] A. Ozturk, A. Tartar, B. Ersoz Huseyinsinoglu, and A. H. Ertas, "A clinically feasible kinematic assessment method of upper extremity motor function impairment after stroke," *Measurement*, vol. 80, pp. 207–216, Feb. 2016, https://doi.org/10.1016/j.measurement.2015.11.026.

[18] K. Ansari, "Real-Time Positioning Based on Kalman Filter and Implication of Singular Spectrum Analysis," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 1, pp. 58–61, Jan. 2021, https://doi.org/10.1109/LGRS.2020.2964300.

[19] B. Bhowmik, M. Krishnan, B. Hazra, and V. Pakrashi, "Real-time unified single- and multi-channel structural damage detection using recursive singular spectrum analysis," *Structural Health Monitoring*, vol. 18, no. 2, pp. 563–589, Mar. 2019, https://doi.org/10.1177/1475921718760483.

[20] T. Tuytelaars and K. Mikolajczyk, *Local Invariant Feature Detectors: A Survey*. Hanover, MA, USA: Now Publishers Inc, 2008.

[21] T. Lindeberg, "On the Axiomatic Foundations of Linear Scale-Space," in *Gaussian Scale-Space Theory*, J. Sporring, M. Nielsen, L. Florack, and P. Johansen, Eds. Dordrecht: Springer Netherlands, 1997, pp. 75–97.

[22] I. R. Otero and M. Delbracio, "Computing an Exact Gaussian Scale-Space," *Image Processing On Line*, vol. 6, pp. 8–26, Feb. 2016, https://doi.org/10.5201/ipol.2016.117.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, Jun. 2005, vol. 1, pp. 886–893 vol. 1, https://doi.org/10.1109/CVPR.2005.177.

[24] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," presented at the International Conference on Computer Vision Theory and Applications, Feb. 2009, vol. 1, pp. 331–340, https://doi.org/10.5220/0001787803310340.

[25] "Eigen Library," *Eigen*. https://eigen.tuxfamily.org.

[26] A. Ozturk, "ozturk-ali/SSA," May 30, 2022. https://github.com/ozturk-ali/SSA.