

A parallel algorithm of multiple face detection on multi-core system

Mohammed W. Al-Neama¹, Abeer A. Mohamad Alshiha², Mustafa Ghanem Saeed³

¹Education College for Girls, Mosul University, Mosul, Iraq

²Remote Sensing Center, Mosul University, Mosul, Iraq

³Computer Science Department, Cihan University, Sulaminah, Iraq

Article Info

Article history:

Received Jul 4, 2022

Revised Oct 22, 2022

Accepted Nov 1, 2022

Keywords:

Face detection

Multi-core system

Multiple face detection

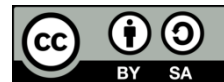
OpenMP

Parallel algorithm

ABSTRACT

This work offers a graphics processing unit (GPU)-based system for real-time face recognition, which can detect and identify faces with high accuracy. This work created and implemented novel parallel strategies for image integral, computation scan window processing, and classifier amplification and correction as part of the face identification phase of the Viola-Jones cascade classifier. Also, the algorithm and parallelized a portion of the testing step during the facial recognition stage were experimented with. The suggested approach significantly improves existing facial recognition methods by enhancing the performance of two crucial components. The experimental findings show that the proposed method, when implemented on an NVidia GTX 570 graphics card, outperforms the typical CPU program by a factor of 19.72 in the detection phase and 1573 in the recognition phase, with only 2000 images trained and 40 images tested. The recognition rate will plateau when the hardware's capabilities are maxed out. This demonstrates that the suggested method works well in real-time.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Mohammed W. Al-Neama

Education College for Girls, Mosul University

Mosul, Iraq

Email: mweama@uomosul.edu.iq

1. INTRODUCTION

Recognizing someone by their face is the most intuitive kind of recognition by a human. Due to the exponential growth of computing power, face recognition algorithms have become more commonplace in scientific inquiry and experimentation in recent years. The study of pattern recognition and image processing has seen a surge in interest in face recognition technologies. The initial phase in face recognition is face detection, which may be used to verify an image's subject and begin cleaning up any imperfections, as shown in Figure 1. In 2001, Viola and Jones, introduced the Viola-Jones cascade classifier [1] as a rapid real-time object identification approach. The algorithm of AdaBoost [2], [3], the integral image, and the cascade of classifier increase the speed of detection in real-time on the central processing unit. Recognizing the faces in the test images comes after the first stage of finding the face area. Principal component analysis (PCA) using Eigen-faces [4], [5] is an established face recognition technique.

Numerous ways are utilized to increase recognition accuracy and minimize the computational complexity [6]-[9]. When it comes to feature extraction from images of faces, there are two main schools of thought: the appearance-based method and the model-based approach [10]. Each of these has its own particular traits. As an illustration, the first strategy is meant to accommodate photographs with poor quality and/or low resolution, whereas in the second method, face variation and other fixed points are assessed before a face

feature model is built. Human interaction is a common component of this strategy. Generally, image face recognition has relied heavily on the first method, which provides a high degree of accuracy but requires a specific face model, such as the expression and position of a human image.

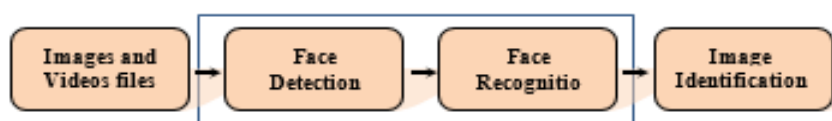


Figure 1. Illustrates the system of face recognition

As a consequence, various suggestions, such as linear discriminant analysis (LDA), independent component analysis (ICA), isometric feature mapping (ISOMAP), fisher analysis (FA), PCA, Kernel PCA (KPCA), and fisher analysis (FA), have embraced the appearance-based technique [6], [11]. PCA and its variations have been employed owing to numerous properties. For instance, principal component analysis (PCA) is a linear technique for reducing a set of high-dimensional vectors to a set of lower-dimensional vectors with a low mean squared error. In PCA, the model parameters can be derived directly from the data, with no additional processing required. This indicates that PCA just needs matrix modification. PCA needs fewer characteristics to retain accuracy quality. These benefits increase identification accuracy, even when working with a limited data set [10]. Even though PCA has numerous advantages, one of its shortcomings is that it is pretty hard to grasp since it includes a lot of massive matrix manipulation operations.

A high quantity of memory is also needed since the memory demand grows with image quality and resolution and the number of training images used for image matching or classification [10], [12]. Many academics have looked at ways to make PCA even more effective, such as symmetric PCA and two-dimensional PCA [10]. Some offered an alternate way to minimize the computing complexity of PCA, i.e., replacing singular value decomposition (SVD) by employing QR decomposition; where Q is an orthogonal matrix (its columns are orthogonal unit vectors meaning $Q^T = Q^{-1}$) and R is an upper triangular matrix (also called right triangular matrix) [12]. However, one viable technique for coping with heavy computational jobs is to minimize the serial constraint by leveraging the parallelism notion. Recent fast improvements in computer chips and integrated circuit technology have made multi-cores [13], [14], and associated parallelization methods accessible and inexpensive, which may give a viable solution to some of the constraints of PCA [15].

As a result of their speed and efficiency, GPUs are increasingly being included in data and computation-intensive programs. Compared to central processing units, the computing capacity of modern GPUs is enormous. Using parallel computing to quicken facial recognition is brilliant because of the inherent data parallelism. NVidia's compute unified device architecture (CUDA) is a framework for GPGPU programming that supports parallel processing and the execution of hundreds of threads simultaneously [16].

We suggested a real-time of the system of facial recognition based on the CUDA platform to speed up the identification process using parallel computing. Extremely rapid facial recognition would have many real-world applications. Kumar [17] identification system obtained a high frontal face detection rate, the study only looked at the detection rate and not the whole system's efficiency. Some work on boosting face recognition performance in parallel was published in papers [15], [18], [19]. Instead of focusing on the number of errors, ours dramatically increased the identification speed like the other systems. This made it possible for real-time facial recognition to become a reality.

2. RECOGNIZING FACES APPROACHES

The camera takes test images, which are subsequently processed by a real-time facial recognition algorithm to extract unique identifiers for individual human faces. This process has two main phases: finding faces and identifying them. Images of faces should be recognized only if they are present. Thus, our suggested method incorporates both facial recognition and detection. After doing a face detection and face ID lookup, we get the exciting face area.

2.1. The technique of face detection

Viola and Jones at Cambridge University developed the cascade classifier system that could recognize faces in real time. The three essential ideas they've brought to the table are the algorithm of AdaBoost, the integral image, and the classifiers' cascade. A face-identification approach that uses Haar-like characteristics is the cascade classifier method. The so-called characteristic value of the feature rectangle is utilized as the foundation for face detection, and it is calculated by subtracting the total pixel values in the white area from

the sum in the black region, as shown in Figure 2. Figure 2(a) displays their work employed six distinct types of Haar-like feature rectangles. While Figure 2(b) illustrates Viola and Jones Haar-like features [1]. The boosting approach classifies an image area as a face or non-face using many weak classifiers with a slightly better classification rate than a random classifier. These weak classifiers add pixels in rectangles and subtract them from others. Viola and Jones introduced the integral image to reduce summing costs. Each integral picture point is computed once. The gray area in Figure 2(b) represents the integral image, at (a, b) , which contains the total of the pixels, and can be computed by (1), and the sum of pixel (P) values across a rectangular area are shown by (2):

$$y(a, b) = \sum x(a', b'); \forall a' \leq a \text{ and } b' \leq b \quad (1)$$

$$P = y(a_4, b_4) - y(a_3, b_3) - y(a_2, b_2) + y(a_1, b_1) \quad (2)$$

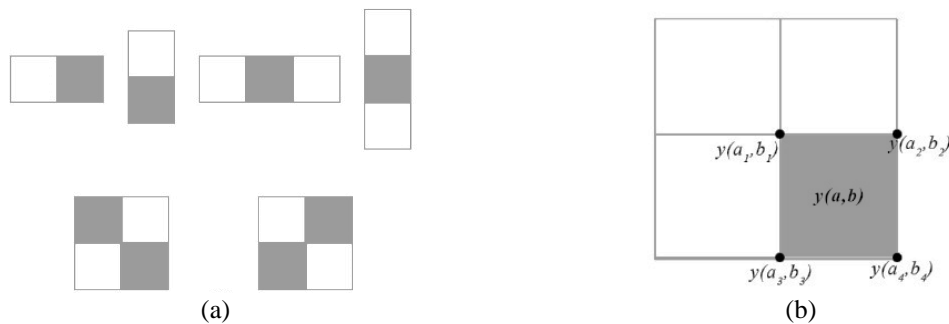


Figure 2. The Haar-like feature rectangles: (a) six distinct types of Haar-like feature rectangles and (b) Viola and Jones' Haar-like features

A classifier function $f = (\alpha, \beta)$, where α is the feature rectangle and β is the threshold value, may be learned using the provided training set and feature rectangle set. The connection between α and β determines the function f 's return value. Rectangles covering face features are picked using the AdaBoost algorithm. This set of feature rectangles may be used to build a weak classifier, and a set of such classifiers can be combined to create a robust one. A cascade classifier, formed by chaining together the strong classifier, can determine whether the scan window includes face information.

2.2. The algorithms for facial recognizability

The goal of principal component analysis (PCA) in face recognition is to identify the characteristics necessary for comparing faces by decreasing the dimensionality of the data. After identifying facial features, the images containing them are scaled (P_w pixels wide by P_l pixels length) and preprocessed. Different training images are used; within each class are many image graphs of the same individual displaying various emotions. A vector $V(P_w P_l' \times 1)$ stores information about each image. Assume that there are $T = \frac{1}{N}$, where N is a training images in the database and that you want to get the average face vector L by using (3). By taking the median away from the V - side, we can calculate K_i . The covariance matrix M is calculated by using 4.

$$L = T \sum_{i=1}^N V_i \quad (3)$$

$$M = T \sum_{j=1}^N K_j K_j^T = SS^T \quad (4)$$

Where $S = [K_1, K_2, \dots, K_N]$.

It can be calculated the eigenvectors v_i of SS^T because the eigenvectors m_i of SS^T are huge and have the same eigenvalues as $S^T S$. Multiplying the v_i by S yields the m_i . In this work, the linear combination of the top E eigenvectors, $N - 1$, represents each face K_i in the training set. There is a particular name for these eigenvectors: eigenfaces. The eigenspace σ is formed by aligning the previously calculated eigenvectors for each image, by using (5).

$$\sigma^T K_i = [s_j^i], \quad i = 1, 2, \dots, N, j = 1, \dots, E \quad (5)$$

After this, the test images are normalized and projected onto the eigenspace σ . We utilize Mahalanobis distance (6) to classify the image and find out where the test face belongs. Projected test images are compared to each projected training image using the Mahalanobis distance. The recognition result is the closest distance to the class if the minimum distance is less than a threshold θ .

$$\omega_i = \sqrt{\sum_{j=1}^e \left(\frac{s_j^i - s_j}{\tau_j} \right)^2} \tag{6}$$

3. CUDA-BASED FACE RECOGNIZER

To begin, a camera captures the test images, which are then preprocessed to a standard size that includes just the facial area. With the help of the recognition algorithm, we can then confidently establish the person's identity. However, the procedure will take a long time since there is so much information. New CUDA-based parallel detection and identification methods were created and optimized.

3.1. Parallel approach

After the classifier and images have been loaded, we compute the image integral using a graphics processing unit (GPU) [20], [21]. To get around the data dependence of image integral computation, a kernel based on the CUDA platform separately calculates the integrals of rows and columns in the order of priority (rows first, columns second). The number of threads is dependent on the image's width. But warp is responsible for scheduling the CUDA threads. The same warp doing the computation of the row integral requires access to data from several columns in the image, but there are only 32 threads in total. Communication costs will increase due to the discontinuity of the column image data. A similar issue also arises with the write-back procedure of row integral calculations. There is a significant drop in system performance.

We used shared memory, as described in CUDA [22], to improve memory access performance. A shared memory of size $N \times N$ has required if each thread in a block has a width of P_w pixels. A block's threads transfer image data into shared memory through row access and perform integral computations. As soon as the calculations are complete, the shared memory containing the results should be written back to the global memory on the GPU. The technique is better suited to the CUDA platform, and memory access time has been decreased.

After that, we use a scan window to locate the offending portion of the image. Since each window may be treated separately, the kernel can speed up this section. The scan window's parallel structure is seen in Figure 3. Although the GPU's execution units are single instruction, multiple data (SIMD), the thread is scheduled via warp. An uneven load occurs when thread1 only goes through the first robust classifier, but thread2, from the same warp, goes through all the robust classifiers.

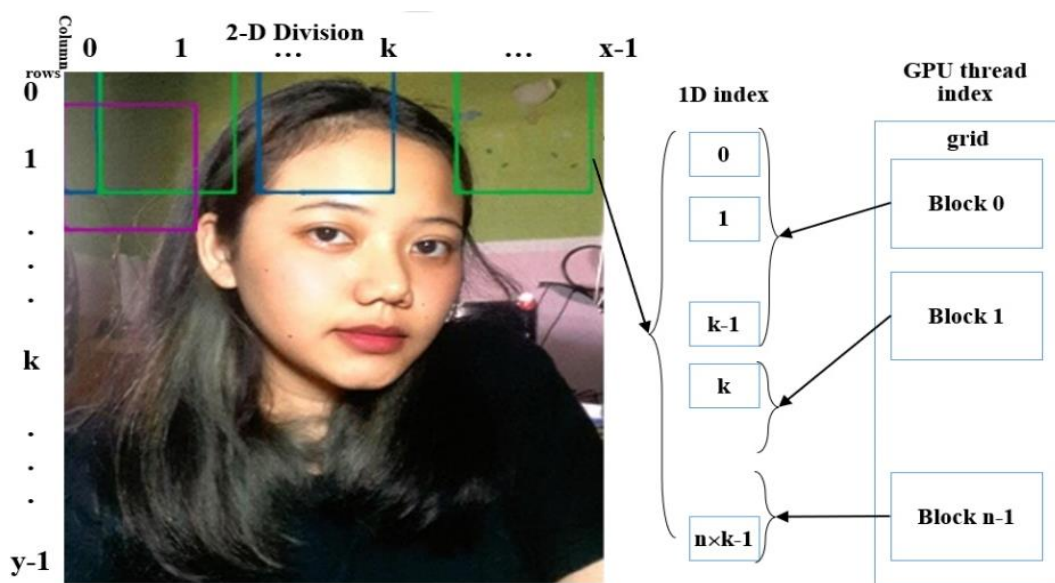


Figure 3. A scan window that is computed in parallel

Reducing the parallel thread granularity to a robust classifier and remapping the thread with the window after one step of the classifier is complete helped us address the unbalanced load issue and ensure that many threads run simultaneously on GPU. Using zero-copy technology [23], the block of page-locked host memory may also be mapped into the address space of the device side, reducing the extra cost of data transfer between the host CPU and the device GPU. Because face size varies, it is necessary to adjust the feature rectangle in the classifier by increasing the scan window size after the first detection. Because the feature rectangle's operation is asynchronous, it only takes one thread to manage one feature rectangle. By running many threads simultaneously, we can reduce the overall processing time, and CPU-to-GPU delays in classification due to a classifier can be eliminated by storing the results in the global memory. When finished, save the merged face window in a fixed-size format. Now that we know where the faces are located, we can go on to the recognition phase of face processing.

3.2. Recognized in a parallel approach

In the recognition phase, training is not required if the database has not been modified. Testing takes the most time; therefore, we focus on how to speed up that part of the process. The flowchart of parallel recognition is shown in Figure 4. Since there is no connection between the testing images, the projection may be computed in parallel after loading the testing images and the training results.

Multiple threads carry out the task simultaneously, as seen in Figure 5. Each component of the vector is independently computed in parallel. The length of a feature vector is equal to K times the number of threads in a block. Our system is now faster and more efficient, and we were able to accomplish it by creating a parallel between images and inside images.

Ultimately, we do facial recognition by finding the most negligible value between the calculations outlined in (6). However, this process might take a long time if either the training faces number M or the testing faces number N is massive. To address the real-time latency issue, we have implemented this procedure in parallel. Each thread in a given block is responsible for calculating the distance between a testing image and a specific training image. The GPU's global memory is where the final calculations are saved. Then, a reduction method is used to get the smallest possible value. If the minimum distance is less than a threshold, the smallest integer is the identifier for the testing face.

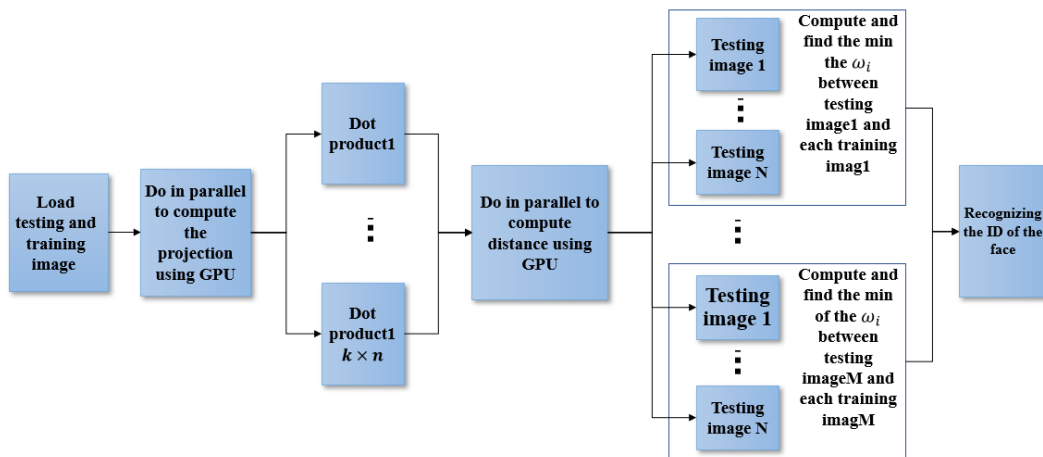


Figure 3. Shows the flow chart of the parallel recognition

3.3. Database

Specifically, we implemented our database of faces (ORL) [24], [25], a widely used database for facial recognition. The ORL Database has 40 separate topics, with 10 images for each. Images of certain subjects were captured at different times of day, with variable lighting conditions and subject attitudes (closed/open eyes, not smiling/smiling, no glasses/glasses). The models were image-graphed in a frontal, standing pose against a uniformly black backdrop (with tolerance for some side movement). The portable gray map (PGM) formatted database files include images that are 92×112 in size and have 256 levels of grayscale. Using the power of the GPU and the adaptability of the software, we may design our own database to meet the standards set by the ORL database. This allowed us to further gauge GPU performance and determine the optimal recognition rate.

4. THE EXPERIMENTS AND RESULTS

The study employs a Core i7 (9300) quad-core processor as the CPU and a (GTX570) GeForce with computational capability 2.0 as the GPU to evaluate CPU and GPU performance. The OS is Windows 10 64-bit. The database of ORL comprised 400 images and was chosen as the training set. To illustrate the full power of a GPU, we added images in the same BMP format to the ORL database, bringing the number of faces from 800 to 1600. The image size has a role in the parallel particle and speedup during the detection phase. The speedup of the recognition phase depends on many factors, such as the number of training and testing images. The results of our testing of the detection phase with images of varying sizes and durations as shown in Table 1. The face detection is the first step in identifying a person's face in real-time, and CUDA is better at this task.

We begin by identifying 10 image graphs in the database with varying magnification degrees. The executing time of GPUs and CPUs (excluding training time), as well as speed-up, are recorded in Table 2. It is clear that as the database grows, the CPU time rises sharply while the GPU time rises more slowly but steadily. It demonstrates that a huge database is ideal for using CUDA for face recognition.

Table 1. Speed-up and the time it takes to detect images of varying sizes

Pixels	CUDA (sec.)	CPU (sec.)	Speed-up
640×480	0.20	1.2	6
800×600	0.21	1.7	8
1024×768	0.20	2.2	11
1280×720	0.19	2.3	12
1920×1080	0.29	4.9	17
3680×2070	0.90	19.7	22

Table 2. The elapsed time for recognition depending on the total number of training images

No.	CUDA (sec.)	CPU (sec.)	Speed-up
50	0.02	0.01	0.5
100	0.05	0.01	0.2
200	0.03	0.02	0.66
300	0.03	1.19	39.66
400	0.04	2.78	69.5
500	0.05	5.38	107.6
600	0.05	9.3	186
700	0.06	14.79	246.5

Training images are represented by threads, and the number of threads in a given block equals the number of training images. Since there are just 10 test images, we can utilize only 10 blocks, and each block will include as many threads as training images. The NVIDIA GeForce (GTX) 570-based experimental platform has 15 multiprocessors, each capable of running 1536 threads. However, not all these threads are currently being used, and some multiprocessors are sitting idle. So, the acceleration is growing horribly with the number of image graphs in the database.

We evaluated 40 video stills with varying database sizes to see how different sizes might affect the performance of the real-time face recognition system. A visualization of CUDA's performance boost is shown in Figure 5. Each sample face serves as a building component in the face recognition process. The maximum speedup cannot be determined since there are not enough images in the database. Based on the information gathered during the trial, the identification rate is close to 99%.

However, the correct of detection, recognition and system recognition rates of CUDA and CPU are (93.77%), (93.53%), (98.72%), (98.15%), (91.21%) and (90.75%) respectively. It is clear that the correct recognition rate of the system is not very high, even though the CPU program and are the CUDA program practically the same. It will be due to the low proportion of accurate detections. When we can't determine where a face is, we can't identify it. It is necessary to initially increase the system's accurate detection rate before increasing the correct recognition rate.

When the training images number surpasses 1024, the block in the GTX 570 can no longer accommodate all of them. Thus, we must rearrange the threads. The grid's square should be shown in two dimensions. Both the x index of the block and the thread index are used to locate the training image. The y-index of the block specifies the test image. The experiment results show that the CUDA-based face recognition system outperforms the CPU-based system in speed and accuracy. This proves that our facial recognition algorithm works in real-time.

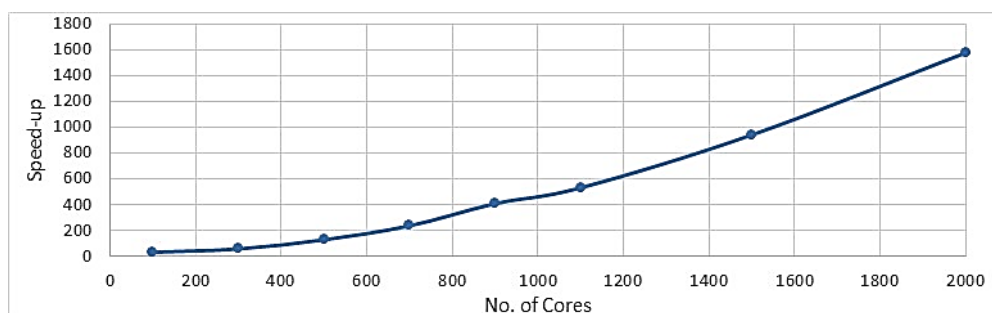


Figure 5. Shows the speedup of visualization of CUDA's performance

5. CONCLUSION

To finish the face detection and identification tasks efficiently, this study suggested a real-time face recognition system based on CUDA. Because we additionally optimized the recognition portion of our face recognition system in tandem with the detection portion, our approach has a high acceleration performance. Our software runs far more quickly than the CPU equivalent. With this speed boost, facial recognition can be used in real-time, which is especially helpful when these apps are needed.

We expect that by using the latest tools, such as a graphics card of NVidia's Kepler architecture, we will be able to better address the data migration problems that arise during the recognition phase. In addition, we want to develop a new classifier that contributes more to the success rate of detection. Optimizing the system will result in significant improvements

ACKNOWLEDGMENTS

The authors would like to thank Mosul University for granting access to running their computations on its platform.




REFERENCES

- [1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988, doi: 10.1109/ICCV.2017.324.
- [2] B. Han *et al.*, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," *Adv Neural Inf Process Syst*, vol. 31, 2018.
- [3] M. A. Ramdhani, D. S. Maylawati, and T. Mantoro, "Indonesian news classification using convolutional neural network," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 2, pp. 1000–1009, 2020, doi: 10.11591/ijeecs.v19.i2.pp1000-1009.
- [4] K. D. Ismael and S. Irina, "Face recognition using Viola-Jones depending on Python," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 3, pp. 1513–1521, 2020, doi: 10.11591/ijeecs.v20.i3.pp1513-1521.
- [5] G. M. Zafaruddin and H. S. Fadewar, "Face recognition using eigenfaces," in *Computing, communication and signal processing*, Springer, 2019, pp. 855–864, doi: 10.1007/978-981-13-1513-8_87.
- [6] D. Shi and H. Tang, "Face recognition algorithm based on self-adaptive blocking local binary pattern," *Multimed Tools Appl*, vol. 80, no. 16, pp. 23899–23921, 2021, doi: 10.1007/s11042-021-10825-z.
- [7] Y. Kortli, M. Jridi, A. al Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, no. 2, p. 342, 2020, doi: 10.3390/s20020342.
- [8] S. L. Oh *et al.*, "A novel automated autism spectrum disorder detection system," *Complex & Intelligent Systems*, vol. 7, no. 5, pp. 2399–2413, 2021, doi: 10.1007/s40747-021-00408-8.
- [9] A. A. M. Al-Shiha, W. L. Woo, and S. S. Dlay, "Multi-linear neighborhood preserving projection for face recognition," *Pattern Recognit*, vol. 47, no. 2, pp. 544–555, 2014, doi: 10.1016/j.patcog.2013.08.005.
- [10] A. L. Machidon, O. M. Machidon, and P. L. Ogrutan, "Face recognition using Eigenfaces, geometrical PCA approximation and neural networks," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019, pp. 80–83, doi: 10.1109/TSP.2019.8768864.
- [11] M. Lal, K. Kumar, R. H. Arain, A. Maitlo, S. A. Ruk, and H. Shaikh, "Study of face recognition techniques: A survey," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 6, 2018, doi: 10.14569/IJACSA.2018.090606.
- [12] C.-S. Hsu and S.-F. Tu, "Enhancing the robustness of image watermarking against cropping attacks with dual watermarks," *Multimed Tools Appl*, vol. 79, no. 17, pp. 11297–11323, 2020, doi: 10.1007/s11042-019-08367-6.
- [13] M. W. Al-Neama, N. M. Reda, and F. F. M. Ghaleb, "Fast vectorized distance matrix computation for multiple sequence alignment on multi-cores," *International Journal of Biomathematics*, vol. 8, no. 06, p. 1550084, 2015, doi: 10.1142/S1793524515500849.
- [14] F. M. Saeed, S. M. Ali, and M. W. Al-Neama, "A parallel time series algorithm for searching similar sub-sequences," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 15, no. 3, 2020.
- [15] Y. Ren, X. Xu, G. Feng, and X. Zhang, "Non-Interactive and secure outsourcing of PCA-Based face recognition," *Comput Secur*, vol. 110, p. 102416, 2021, doi: 10.1016/j.cose.2021.102416.
- [16] S. R. Nayak, S. Sivakumar, A. K. Bhoi, G.-S. Chae, and P. K. Mallick, "Mixed-mode database miner classifier: Parallel computation of graphical processing unit mining," *The International Journal of Electrical Engineering & Education*, p. 0020720920988494, 2021, doi: 10.1177/0020720920988494.




- [17] R. M. S. Kumar, "Robust multi-view videos face recognition based on particle filter with immune genetic algorithm," *IET Image Process*, vol. 13, no. 4, pp. 600–606, 2019, doi: 10.1049/iet-ipr.2018.5268.
- [18] J. K. Mani, "Face recognition with frame size reduction and DCT compression using PCA algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, 2021, doi: 10.11591/ijeecs.v22.i1.pp168-178.
- [19] F. Zou, J. Li, and W. Min, "Distributed face recognition based on load balancing and dynamic prediction," *Applied Sciences*, vol. 9, no. 4, p. 794, 2019, doi: 10.3390/app9040794.
- [20] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters/crc Press, 2019, doi: 10.1201/9781315365459.
- [21] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, pp. 39–64, 2020, doi: 10.1016/j.neucom.2020.01.085.
- [22] A. Esteva *et al.*, "Deep learning-enabled medical computer vision," *NPJ Digit Med*, vol. 4, no. 1, pp. 1–9, 2021, doi: 10.1038/s41746-020-00376-2.
- [23] Z. Pan *et al.*, "Exploring data analytics without decompression on embedded GPU systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1553–1568, 2021, doi: 10.1109/TPDS.2021.3119402.
- [24] M. J. Alam and T. M. S. Ali, "A smart login system using face detection and recognition by ORB algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 20, no. 2, pp. 1078–1087, 2020, doi: 10.11591/ijeecs.v20.i2.pp1078-1087.
- [25] N. Rathika and N. Sathya, "Recognition of face CLAHEM based on using GPP–HM," *J Ambient Intell Humaniz Comput*, vol. 12, no. 6, pp. 6735–6739, 2021, doi: 10.1007/s12652-020-02297-0.

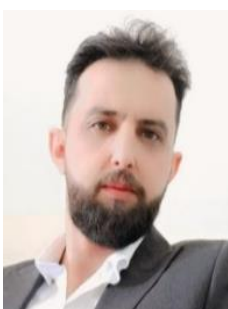
BIOGRAPHY OF AUTHORS






Mohammed W. Al-Neama    was born in Baghdad, Iraq. He got his bachelor's from the University of Mosul, Iraq in 1995. He got his M.Sc. from the University of Mosul in 2004 and got his Ph.D. in High performance computing (Parallel tools and applications) in 2014, Al-Azhar University – Egypt. He is currently a Assistant Professor at the Education College for Girls, University of Mosul, Mosul, Iraq. His main research area is High Performance Computing, Bioinformatics, Parallel Computing. His research interests are in exact string-matching algorithms, parallel and distributed processing. He can be contacted at email: mwneama@uomosul.edu.iq.



Abeer A. Mohamad Alshiha    received the BSc in Computing Science and the MSc degrees (First Class Hons.) in Computing Science/ Remote Sensing from the Mosul University, Iraq, in 1995 and 2002, respectively. Her Ph.D. Degree in Artificial intelligence from the School of Electrical and Electronics Engineering, Newcastle University, Newcastle Upon Tyne, UK, in 2013. She works as a lecturer & researcher in the Remote Sensing Center, Mosul University, Iraq since 1996 yet. And also, works as a lecturer in different colleges such as the College of Computing and Mathematical Science and the College of Petroleum and Mining, Mosul University, Iraq. Her current research interests include biometrics, artificial Intelligence, tensor object processing, and statistical pattern recognition, with applications in human recognition, tracking, and detection. She can be contacted at email: abeer.allaf@uomosul.edu.iq.



Mustafa Ghanem Saeed    was born in Mosul, Iraq. He got his bachelor's from the University of Mosul, Iraq, in 2007. He got his M.Sc. from University of Mosul in 2013 in software engineering. He is currently a Lecturer at Cihan University-Sulaminah, Iraq. His research interests are in deep learning and software engineering quality. He can be contacted at email: mustafa.saeed@sulicihan.edu.krd.