International Journal of Research in Engineering, Science and Management
Volume 5, Issue 6, June 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

A Survey on Face Detection and Recognition using CUDA

Shweta S. Kumar[1*], K. Raju[2]
[1]Student, Department of Computer Science and Engineering, NMAM Institute of Technology, Nitte, India
[2]Associate Professor, Department of Computer Science and Engineering, NMAM Institute of Technology, Nitte, India

***Abstract*: Face recognition and Face detection techniques have gained a lot of observation in past few decades. Identifying criminals, tracking attendance, disease diagnosis, and personal identification are few applications of face detection and face recognition. Face recognition requires large computation due to the dimensional structure of the face and computation cost increases due to a large number of the image dataset. CUDA has removed the obstacle by computing image data in parallel rather than sequentially. The focus of this survey is to study the techniques used in face detection and recognition using CUDA. As a result of the research, it was discovered that CUDA allows processing to be done faster for face detection and face recognition operations.**

***Keywords*: Face detection, Face recognition, CUDA.**

1. Introduction

Face recognition systems have become increasingly popular in research and development in recent years, especially with the current advancement of computer technology. Many scholars found that designing a face recognition model to be an interesting area, owing to the numerous real-world applications such as law enforcement, verification, security systems, and surveillance systems. Furthermore, implementing a face recognition model is challenging due to its complicated structure of human face. The approach chosen for face representation, has a significant impact on part of processing and memory required for a face recognition system.

Overall, Face detection and recognition are the two components. The first technique is implemented to spot the face before particularly distinguishing the face identity. The initial step in a face recognition system is face detection. The field of face detection has provided a strong stride in terms of enhanced detection accuracy and speed. The integral image, the AdaBoost algorithm, and the classifier cascade were the three main methodologies and concepts used in face detection [15]. Although these methods obtained a high detection rate on the CPU, with the constant increase in image resolution, the CPU no longer meets its requirements in terms of rapid face detection. Parallel computing is the most efficient method of face detection and Recognition.

Parallel processing is the process of splitting a complex obstacle into smaller jobs, distributing these jobs to several processors, and accomplish them all at the same time. The basic goals of parallelization are high achievement through reducing time, increased performance, and better resource use. Each sub-instructions problem runs in parallel on different processors. Parallel computing is achieved by using GPU.

In data processing applications, graphics processing units (GPUs) are commonly utilized. The computing power of GPUs is several times that of CPUs. GPU computes complex operations, especially when it comes to generating frames for real-time computer games and complicated embedded videos. CPUs have a limited number of cores that are intended for sequential evaluation, but GPUs have hundreds of cores that are targeted to parallel processing. As a result, a huge increase in speed can be realized by doing high-level computational tasks on the GPU while the remaining of the code is computing on the CPU. GPGPU supported CUDA is used to speed up the parallel computing approaches.

2. Face Detection and Face Recognition Techniques

A. *Face Detection Technique*

Face detection methods can be classified into two categories: feature-based detection and image-based detection. The first phase involves generating a collection of weak classifiers depend on Haar-like features and then creating a stage cascade classifier with all of the promising weak classifiers [5]. In the second phase, the algorithm will explore for every size of the input frame using the trained stage cascade classifier in a dynamic search window to check for characteristic of a human face.

1) Haar-like Features: The Viola-Jones cascade classification algorithm is a Haar-like feature-based face detection algorithm. Haar features are used to replicate the different features of the face in the image. The sum of pixel values of the white region gets subtracted from the black region. The output of the feature results in a single value. All haar features have some sort of resemblance to the facial expression. For analyzing features, the Voila Jones method uses 24X24 windows as the base window size. Haar features include all feasible characteristics such as position, length, and type, resulting in a total of 160,000+ features being calculated.

2) Integral Image: The value at pixel (x,y) in an integral image is the addition of the pixels above and to the left of it

*Corresponding author: shwetaskumar24@gmail.com

(x,y). The total number of pixels within any given rectangle can be determined using only four values at the rectangle's corners. The integral image can be calculated by using,

$$IS(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

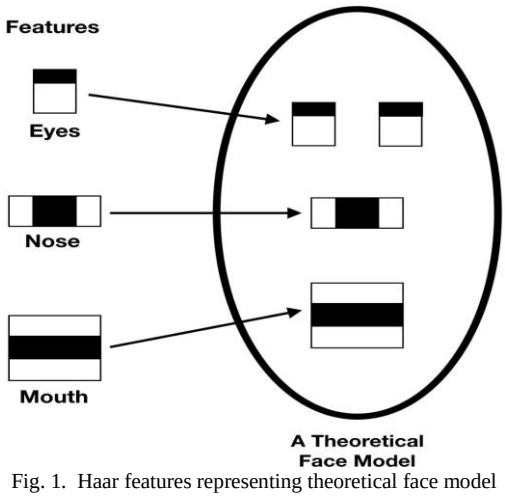

Fig. 1. Haar features representing theoretical face model

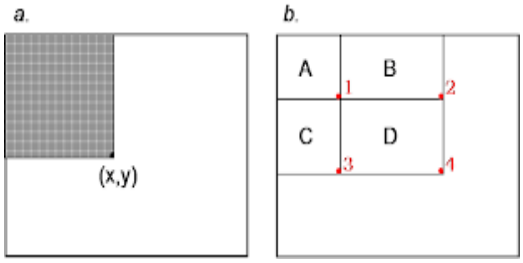

Fig. 2. Addition of pixels in integral image

The value of any rectangular sum in constant time can be computed using integral image computing. The integral entity of rectangle D can be calculated as follows:

Sum(D)=IS(4)+IS(1)-IS(2)-IS(3)

3) Adaboost: Adaboost is a technique for integrating some of the irrelevant features into a relevant feature that may be utilized to model and recognize a face in a scene. It is an efficient technique that evaluates each rectangle feature as if it were a simple weak classifier. The Adaboost algorithm is used to choose a feature rectangle that includes face features, integrate them into a weak classifier, generate a strong classifier from a collection of weak classifiers, and finally concatenate the strong classifiers into a cascade classifier [2]. Adaboost is used to eliminate redundant features. AdaBoost accepts training samples as input.

$$S = (x_1, y_1),.,(x_n, y_n)$$

For each instance of S belonging to a field or instance space X, and each label belonging to the finite label space Y. When identifying the suitable return value of the classification function and threshold, the algorithm can generate a simple and productive classifier, and the classifier may considerably enhance the activity of face detection.

4) Cascade Classifier: The algorithm includes several features in the cascade classification stage, such as eyes and nose, upper cheeks, forehead region, and so on, and it would be impractical to asses all of the features arbitrarily. To avoid this issue, the algorithm uses cascade classification to divide the count of features and quickly remove windows that do not contain a face.

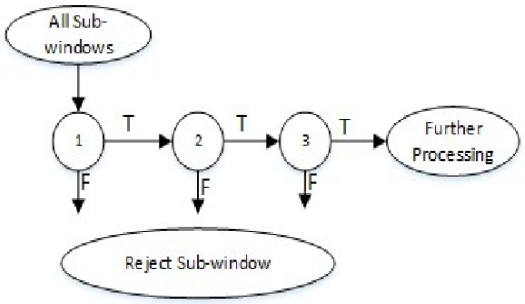

Fig. 3. Cascade-Classifier

Cascade also prevents windows that don't resemble a face from being analyzed repeatedly. When a window fails a certain phase, it is automatically labeled as not a face. In general, initial phases are easier to pass, whereas final phases are more difficult. If the window has all of the features of a specific stage, the stage is considered to be passed, and the window is passed to the neighbouring stage, where it is scanned for features of that stage. If a window passes all of the stages, it is considered a face, and the following window is processed in the same way.

B. Face Recognition Technique

PCA is a statistical process that converts a mathematical model of apparently interrelated variables into a set of exponentially stochastic variable values called principle components using an optimization technique. The number of estimated parameters is underneath or comparable to the number of principal components. PCA can be performed using eigenvalue analysis of a data covariance (or correlation) matrix, usually after normalizing the input vector for each attribute. PCA algorithm is used for features selection and for reducing the dimensional.

PCA computation in Face Recognition

Step 1: The training set will consist of total M images $I_1, I_2....,$ I_M, each of the image is of size NxN. It has N^2 pixels.

Step 2: Each of the faces in the training set is converted into a vector form shown in figure 5. It is represented in τ_i. This is a column Vector.

Step 3: calculate the average face vector Ψ.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \tau_i$$

Step 4: Normalize the face vector by subtracting the average face vector from the original faces. Normalization is removing all the common features the faces share, they should be re-

moved and left with unique feature.

$$\Phi_i = \tau_i - \Psi$$

Step 5: Calculate the covariance matrix.

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T$$

where A= $\Phi_1, \Phi_2 \ldots, \Phi_M$

Step 6: Compute Eigen vector u_i of the covariance matrix.

$$u_i = Av_i$$

Step 7: Represent each face image a linear combination of all K-Eigen Faces, computing the image back to the original dimension by adding the features removed during the analysis.

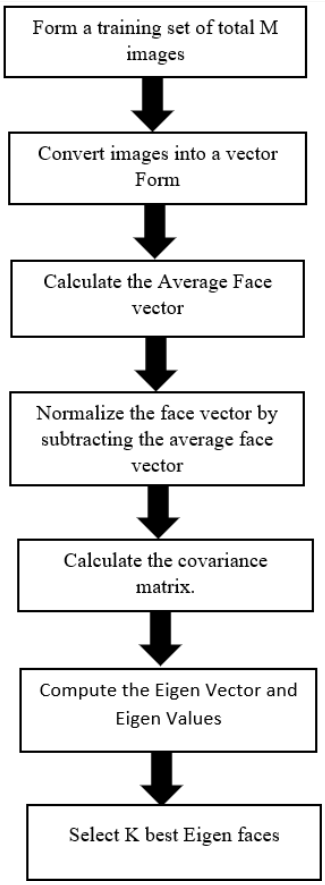

Fig. 4. Computation of PCA algorithm

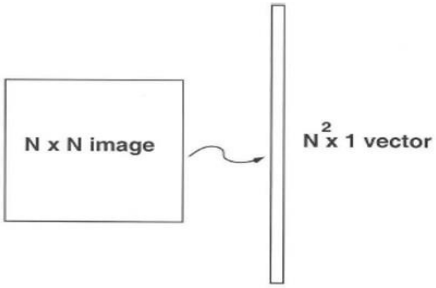

Fig. 5. Conversion of NxN image into N^2x1 vector

3. Survey on Face Detection and Face Recognition

A. Face Detection Survey

The existing system of face detection describes the various methods being used to detect the face. Face Detection methods are categorized into Knowledge-Based Method, Template Based Method, Feature-Based Method, Appearance Based Method, and Viola and Jones Method [17]. Due to the sophisticated structure of the face, it becomes difficult to detect the face, and computation time increases. The researchers have proposed many ways to parallelize the detection method to get an efficient computation time. The feature of the face detection system is implemented in two ways, Implementation on the CPU and Implementation on the CPU and GPU. In the CPU analysis part, a single thread is used to implement all of the features of the face detection system [4]. Some feature was implemented using CPU (host) and the majority of functionality was developed using GPU (device) with data parallelization under the CPU and GPU implementation phase. CUDA technology is used to execute the face detection al- gorithm on the GPU. Every thread in CUDA comprises kernels that are performed n times, and each thread is identified by a unique number. In the Compute Unified Device Architecture (CUDA) platform, the study suggested a rapid face detec- tion acceleration approach based on the Viola-Jones cascade classifier. Li-Chao Sun and Sheng-bing Zhang developed and enhanced unique parallel algorithms for image integral computation, computing of scan windows, and Synthesis of classifiers and correction [2]. The performance of the CUDA version over the CPU version rises exponentially as the image size increase, the CUDA program executing on an Nvidia GTX 570 graphics card could gain 17.04 and 3.22 times faster speeds than the CPU and Open Source Computer Vision (OpenCV) program. This program is still restricted by the complication of too many split at the strong classifier stage. This can be reduced by changing the classifier's structure and enhancing the realization approach, allowing face identification speed to increase.

CUDA environment to take control of the GPU component's power, and the "OpenCV" package to understand the input image and perform the analysis.The block size is a major consideration in computation power [3]. The developer can use CUDA functions to specify the number of kernel threads that execute on the GPU. The serial processes of the Viola- Jones technique can be executed on a correspondent platform using CUDA and OpenCV on the GPU [8]. The algorithm obtained from parallel implementation are used to boost the computing speed, and then compared the results of serial and parallel applications. The shifting of procedure to and from main and GPU memory consumes the majority of the handling time. It performs best in a repressed environment since the parallel performance requires minimum time. The detection rate is higher than the traditional approach, and it is effectively better to analyse massive data using GPU.

A comparison of performance time for extracting features implementation on the GPU and CPU has been analyzed by modifying the magnitude of the image. This model can execute parallel computation in a fraction of the time and with a

considerable speedup over the CPU [3]. The GPU implementation's performance is indicated by its enhanced performance when compared to sequential calculation. This can be further enhanced by designing more advanced im- age transmission algorithms in the GPU by employing the appropriate memory type to minimize memory access and optimizing CUDA programming by making greater use of the GPU's resources to obtain significant performance.

GPU-based implementation compares performance to CPU-based implementation by analyzing images at various scales, sizes, and the number of faces. Robust Face Detection System implementation on the GPU initiated that the GPU-based method performed 5.41 to 19.75 times higher than the CPU version and represented significantly better even at higher resolutions [4]. This can be enhanced by adding some new features to the side pose of an image and extending the same for face recognition. Using CUDA technology, the facial detection method may be optimized for implementation on the GPU. When the Viola-Jones face method is used in parallel, it produces optimum results. For several algorithms, the speed-up factor can be computed, when algorithms involving rigorous computations are divided up into parallel and executed concurrently, they perform better on parallel systems.

The CUDA process begins with memory allotment in the device (GPU) while data is prepared on the host (CPU). The data is then passed from the host to the device. As this is a time-consuming operation, it is vital to reduce the amount of data that must be moved from the host to the device and from the device to the host [5]. It is feasible to run kernels after the data on the device has been compiled. When the estimation is accomplished, the impact are sent back to the host for display, and the allocated memory is freed. The latest GPU technology, such as Compute Unified Device Architecture (CUDA), has demonstrated its ability to speed up parallelization processes and enhance the overall performance of the system. The detection technique was tested for performance on various NVIDIA GPU cards, and a comparison of the face detection application may be measured in terms of frames per second between the CPU and the proposed GPU acceleration. OpenCV and CUDA can use the Viola-Jones algorithm based on Adaptive Boosting to accelerate a frontal face identification system. When compared to the CPU version, the CUDA-based GPU accelerated Viola-Jones face detection has a extreme speed up of twenty two times and is also faster than FPGA implementation of 16 frames per second.

Viola and Jones algorithm can be executed in the simple one-thread CPU version and the initial application is then expanded to include a multi-threaded CPU version. The most essential aspect is that the image size determines the detection time [6]. As an outcome, the single-thread CPU implementation is significantly slower than the multi-thread CPU and GPU implementation. The outcome of CUDA accelerating face detection shows that GPU detection is 35 times faster on average than one thread CPU detection. The results are consistent with the multi-thread CPU model, although the GPU is still faster. It is intended to expand face detection capabilities to allow for the recognition of faces preserved in a database.

Based on the GPU component, the researcher designed a robust face detection system [7]. The code has been enhanced by using a strategy to utilize diverse memory constraints in the GPU and the warp scheduler approach to speed up memory admittance, with greater resource utilization provided by GPU. The entire image is processed during the detection phase by changing the distinguishing window pixel by pixel in either the horizontal or vertical area. The proportion are changed by gradually increasing the magnitude of the detecting window. Images with a small quantity of faces are statistically fast, whereas images with a large quantity of faces take longer to process. When processing a computationally basic image, the GPU operates 1.7 times speeder than when processing a more complicated image. The research outcome show that the approached method exceed the standard method in terms of speed, resulting in a considerable increase in processing time. Furthermore, for a single image of varied sizes, the parallel approach yielded better results. To achieve competitive results, the Viola-Jones face identification algorithm can run on numerous GPUs.

B. Face Recognition Survey

Template-based matching and feature-based matching are the two main types of facial recognition algorithms. To determine key individual components, template based matching projects for a chain of images from a database into a new subspace. Feature-based matching concentrates on the face's overall attributes [16]. The NVIDIA GeForce GTX 770 GPU has a computational capability of 3.0, a strategy is proposed for processing images for face detection and recognition in parallel. Face detection is performed by Viola and Jones, and face identification is performed using the PCA Eigenfaces algorithm. An important feature is that the algorithms are executed in parallel, with one detecting a face and forwarding it to the face-processing system [1]. Face processing on the CPU takes substantially longer than on the GPU because it must wait for all faces to be detected in the frame, whereas with a parallel system, any face detected at any time is recognized instantly, which distinguishes between CPU and GPU execution in a real-time basis. Viola and Jones show results of analyzing the faces at 15 frames per second and the parallelization method shows 109 frames per second in CUDA.

CUDA implementation can be efficiently used for Local Binary pattern computation and K-NN classification [11]. The first approach is to show about using a single kernel to compute LBP values from an input image and build weighted regional LBP histograms in GPU. The second approach is to show a GPU implementation of the k-NN algorithm that is designed for dealing with high-dimensional feature vectors. By speeding up both the feature extraction and classification processes of the face recognition algorithm, researchers were able to achieve a 29x gain in recognition speed when compared to CPU implementations. Researchers identified various exaggeration that can fully exploit GPUs to provide problem solving processing of unified face detection and recognition. Optimization strategies for LBP-based integrated face detection and identification algorithms [10], achieved by accelerating

them with OpenCL on ARM Mali GPU and CUDA on Tegra K1 GPU for HD inputs, attaining 22 fps using OpenCL and 38 fps using CUDA. This is equal to speedups of 2.9 and 3.7 times, respectively.

A super-fast Parallel Eigenface for face recognition is implemented, using CUDA on NVIDIA K20 GPU [12]. The goal of the study was to achieve optimum accomplishment by designing highly modified kernels for a comprehensive process and employing the quickest library functions available. Reduce, projection, subtract, transpose, matmul, and euclidian Dist are some of the effective kernels being used in the study. The generation of feature vectors is boosted by 460X, the whole training process is boosted by 73X, and the overall testing procedure of one image is increased by 6X. The output and estimation analysis show that the proposed approach is both accurate and scalable.

Using PCA, the eigenface technique minimizes the number of dimensions that the face classifier must analyze. The eigenvalues generate a feature space, which has far lower-dimensional than the original space [14]. Researchers an- alyze the design space for parallelizing a PCA-based face recognition algorithm and propose a fast face recognizer on GPUs. Three tasks are successfully accelerated for data parallelism, they are Covariance Matrix Computation Task, Eigenface Computation Task, Projecting to Subspace (PS) task [18]. Covariance Matrix Computation achieved about 120-fold speedup considered as best when compared with other tasks. Various possible mapping of PCA-based face recognition onto GPUs are possible by effectively accelerating overall by thirty folds and accelerating the OpenCV APIs for general PCA.

The overall execution time of the training phase, which includes normalization, covariance, Jacobi, eigenface, and weights module, is estimated for serial and parallel implementation. Shifting training image from CPU to GPU takes less time as correlated to computation time [14]. In the training phase, the parallel execution of the eigenface algorithm showed a 5x speedup. As the training database expands, the speedup also increases. The eigenface computation module achieved the largest speedup of 148x during the training phase. With a speedup of 3x to 5x, the Jacobi module has the minimum attainment improvement of all the training phase modules. This can be improved by employing dynamic parallelism to implement the Jacobi module. With dynamic parallelism, the Jacobi module only needs to start one kernel from the host, and no storage copy will be required.

Face detection and recognition tasks were successfully achieved using a face recognition system based on the Compute Unified Device Architecture (CUDA) platform [15]. Researchers developed and optimized unique parallel techniques of image integral, computation scan window processing, and classifier amplification and correction during the face detection phase and explored some of the testing steps parallelized during the face recognition phase. When compared to a classic CPU program, the parallel recognition technique proceeding on a NVidia GTX 570 graphics card could enact 22.42 times speedup in the detection stage and 1668.56 times speedup in the recognition stage when only training two thousand images and testing forty images in comparison to a CPU program.

For the PCA method training and recognition steps, the parallel GPGPU performance was shown to be quicker than the serial multithread performance [16]. The program was performed on two machines with varied CPU speeds and cores, but the GPGPU computation potentiality were similar. The Apollo 3 and Apollo 5 machines were used to examining the GPGPU implementation's performance. By comparing the CUDA and serial performance side by side, the training and recognition codes are analyzed individually. The training step showed maximum improvement, the recognition process also improved significantly. This work can be enhanced by implementing the algorithm on using Singular Value Decomposition and extending other recognition algorithms (LDA and ICA) to the GPGPU platform, as well as evaluating the period of time and exactness of the algorithms in serial and parallel form.

4. Conclusion

Face detection and recognition experiments using CUDA were studied in this paper. It has been determined that face detection and recognition tasks performed using CUDA's parallel computing capacity are significantly faster. The majority of researchers have analyzed Viola and Jones algorithm and Principal component analysis for face detection and recognition using CUDA. Faster face detection and identification systems can be utilized in all aspects of life by developing CUDA-based systems with shorter execution times.

References

[1] Shivashankar J. Bhutekar, Arati K. Manjaramkar, "Parallel Face Detection and Recognition on GPU," International Journal of Computer Science and Information Technologies Vol. 5 (2), pp. 2013-2014, 2018.

[2] Li-chao Sun, Sheng-bing Zhang, Xun-tao Cheng, Meng Zhang, "Acceleration Algorithm for CUDA-based Face Detection," International Conference on Signal Processing, Communication and Computing, 2013, pp. 1-5.

[3] Hana Ben Fredj, Mouna Ltaif, Anis Ammar, Chokri Souani, "Parallel implementation of Sobel filter using CUDA", International Conference on Control Automation and Diagnosis (ICCAD), pp. 209–212, 2017.

[4] Vaibhav Jain, Dinesh Patel. "A GPU based implementation of Robust Face detection System" Procedia Computer Science, 87:156-163, 2016.

[5] Adrian Wong Yoong Wai, Shahirina Mohd Tahir and Yoong Choon Chang, "GPU acceleration of real time Viola-Jones face detection," IEEE International Conference on Control System, Computing and Engineering (ICCSCE), pp. 183-188, 2015.

[6] J. Krpec, M. Nemec. "Face detection CUDA accelerating", the Fifth International Conference on Advances in computer Human Interactions, Valencia, Spain; 2012, pp. 155-160.

[7] Hana ben fredj, Souhir Sghair, Chokri Souani. "An Efficient Parallel Implementation of Face Detection System Using CUDA", International Conference on Advanced Technologies for Signal and Image Processing, ATSIP', Sfax, Tunisia, 2020.

[8] Aashna R. Bhatia, Narendra M. Patel, Narendra C. Chauhan, "Parallel implementation of face detection algorithm on GPU". 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), 2016.

[9] Paramjeet Kaur, Nishi, "A Survey on CUDA", International Journal of Computer Science and Information Technologies, Vol. 5 (2), 2014, 2210-2214.

[10] S. Yi, I. Yoon, C. Oh, and Y. Yi, "Real-time Integrated Face Detection and Recognition on Embedded GPGPUs," IEEE 12th Symp. on Embedded Syst. for Real-time Multimedia (ESTIMedia), pp. 98-107, Oct. 2014.

[11] S. C. Tek and M. Gkmen, "CUDA accelerated face recognition using local binary patterns." Proceedings of Winter Seminar on Computer Graphics, 2012.
[12] U. Devani, V. B. Nikam, and B. B. Meshram, "Super-fast parallel Eigenface implementation on GPU for face recognition," in 2014 International Conference on Parallel, Distributed and Grid Computing, 2014, pp. 130–136.
[13] Bhumika Agrawal, Chelsi Gupta, Meghna Mandloi, Divya Dwivedi, Jayesh Surana, "GPU Based Face Recognition System for Authentication". International Journal of Engineering Development and Research, Volume 5, 2017.
[14] Kawale MR, Bhadke Y, Inamdar V (2014), "Parallel implementation of eigenface on CUDA.," in 2014 International conference on advances in engineering technology research (ICAETR - 2014), pp 1–5.
[15] Ren Meng, Zhang Shengbing, Lei Yi and Zhang Meng, "CUDA-based real-time face recognition system", Digital information and communication technology and it's applications, 2014 fourth international conference on, pp. 237–241, IEEE, 2014.
[16] T. Goodall, S. Gibson, and M. C. Smith, "Parallelizing principal component analysis for robust facial recognition using CUDA," Symposium on Application Accelerators in High Performance Computing. pp. 121–124, 2012.
[17] Patel Raksha R, Isha K. Vajani, "Face Detection on a parallel platform using CUDA technology", International Journal of Advance Engineering and Research Development, Volume 2, Issue 5, May 2015.
[18] Woo Y, Yi C, Yi Y (2013), "Fast PCA-based face recognition on GPUs," in Acoustics, speech and signal processing (ICASSP), IEEE International Conference on, pp. 2659–2663, 2013.