A Parallel Framework for Object Detection and Recognition for Secure Vehicle Parking

Zahid Mahmood[#], Muhammad Usman Shahid Khan[#], Muhammad Jawad[#], Samee U. Khan[#], and Laurence T. Yang[*]

[#]*Department of Electrical & Computer Engineering, North Dakota State University, USA.*

{zahid.mahmood, ushahid.khan, muhammad.jawad, samee.khan}@ndsu.edu

[*]*Department of Computer Science St. Francis Xavier University Antigonish, NS, B2G 2W5, Canada.*

{ltyang@stfx.ca}

Abstract—**Image analytics, biometrics access control, security, and surveillance applications utilize complex machine learning and computer vision algorithms, such as face detection and recognition. Speedup and accuracy are two important factors that need to be addressed in all such complex applications. Parallel computing breaks down the complex tasks into discrete fragments to be solved concurrently on multiple processors. The parallel computing procedure significantly reduces the execution time with improved speedup. This paper presents a parallel framework for object detection and recognition for a secure vehicle parking. The proposed framework is divided in to three steps: (1) vehicle detection at the parking entry junction, (2) driver's face detection, and (3) identification of driver's face from the huge database of stored facial images. On successful identification of authorized person, vehicle is allowed to enter in the parking-lot. The adaptive boosting algorithm is used for vehicle and face detection, while Eigenfaces based approach is employed for face recognition. Moreover, the scalability comparison of parallely executed driver face recognition algorithm indicates high speedup compared to serial execution. Furthermore, the results of the proposed framework reveal promising performance and encourage outcomes to be deployed in real-time at entrance/exits of the public/private vehicle parking areas.**

***Index Terms*— Eigenfaces, Face Recognition, Haar Classifiers, Parallel Processing**

I. INTRODUCTION

OBJECT detection is a process of locating all regions of interest in an image, such as face, player, or vehicle [1]. Object detection is widely used in a variety of applications, for example security, access control to building, and identification systems [2]. During the past decade, object detection and recognition algorithms have been widely researched and improved in terms of accuracy, performance, and speed [3]. Typically, object detection and recognition algorithms require high performance computations [3]. Increased computation performance has rendered a phenomenal growth in the computational capabilities of microprocessors. However, this increase is hardly able to catch up with the rapid advancements in image and video technology, including camera resolution, and in the computational requirements in real-time applications. Parallelism was introduced by the system designers to implement simultaneous processing concepts. Parallelism improves processing speed by increasing the number of hardware modules that operate concurrently, accompanied by forming multiple processes [4]. Parallel processing and multi-core architectures are the key factors in achieving high performance in future computing systems [5]. In multi-core system standards, the level of parallelization is fixed and the performance is improved with certain architectural features, such as Single Instruction Multiple Data (SIMD) units or exploitation of cache effects [6]. Our contributions to develop object detection and recognition framework are:

- We develop a complete working framework in which a vehicle and driver face is detected followed by a robust driver face recognition algorithm.
- We deploy a robust parallel architecture for efficient driver face recognition from a large pool of stored face database. The parallely executed face recognition algorithm results in a significant boosted speed.
- We simulate a very challenging task, such as when only one gallery (training) face sample is available and four different poses ($+45^0$, $+35^0$, 0^0, and -35^0) as probe (test) are available. These four poses are chosen because a driver's face orientation can vary while sitting on driving seat from frontal (0^0) up to $+45^0$.
- We conduct series of experiments to demonstrate the acceleration of driver face recognition by deploying a parallel scheme. Fig. 1 shows an overview of developed framework.

The rest of the paper is organized as follows. In Section II, we explain the object (vehicle and face) detection and recognition (face) algorithms. The employed parallel architecture is discussed in Section III. Detailed simulation results are presented in Section IV. Finally, conclusions and future research directions are highlighted in Section V.

II. OBJECT DETECTION AND RECOGNITION ALGORITHMS

A. Vehicle and Driver Face Detection

To detect vehicle and drive face, we use the most popular Viola and Jones approach [3] based on Adaptive Boosting (AdaBoost). The approach was initially developed for face detection. We detect the vehicle and driver face using the approach of [3] because of its ability to run in real-time.

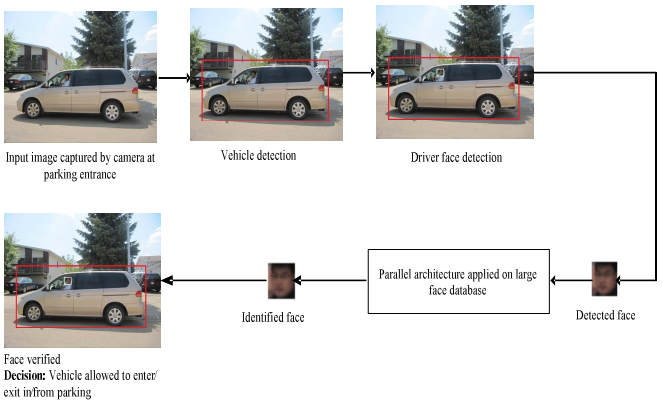


Fig. 1: Proposed framework (image captured at parking entrance of bison sports arena, NDSU, Fargo, USA)

The approach is innovative due to three major ideas: the integral image, boosting, and the attentional cascade.

Integral image: The integral image or summed area table is an algorithm for quickly and efficiently computing the sum of values in a rectangle subset of a grid. We calculate the integral image for rapid computation of Haar-like features using Eq. (1).

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y') \qquad (1)$$

where $ii(x,y)$ is the integral image at pixel location (x,y) and $i(x',y')$ is the original image. Using the integral image to compute the sum of any rectangular area is extremely efficient. The sum of pixels in rectangle region *ABCD* in Fig. 2 is computed as:

$$\sum_{(x,y)\epsilon ABCD} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C) \qquad (2)$$

The integral image is used to compute Haar-like features as shown in Fig. 2.

Boosting: Boosting is a methodology to find accurate hypothesis by combining many *weak* hypotheses, each with reasonable accuracy. We develop a robust vehicle classifier by supervised AdaBoost learning. Algorithm 1 shows the general pseudocode of the AdaBoost learning based approach. A set of positive samples (4800 vehicles and 5000 faces) and 11000 negative samples (non-vehicles/faces) are used to train the classifier. The training data is selected from our own database of vehicle images. To speed up the computation time, we convert all coloured images into gray-scale and normalize the image size to 18x18 pixels. Examples of positive and negative data are shown in Fig. 3. An image captured in any parking area may contain other objects, such as warning boards. Therefore, the only purpose of vehicle detection is to restrict the application processing area by eliminating the redundant information. Vehicle detection provides the exact position of car arriving at the parking entrance. We train the AdaBoost classifier in 28 stages having 1100 weak classifiers. The final classifier $h(x)$ is linear combination of weak classifiers as shown in Eq. (3).

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases} \qquad (3)$$

Algorithm 1: AdaBoost algorithm to obtain vehicle and driver face classifier

1. Give example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative & positive examples respectively.
2. Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$, where m and l are the number of negatives & positives respectively.
3. **for** $t = 1; t < T; t++$ **do**
4. Normalize the weights,
$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
to make w_t as a probability distribution.
5. For each feature j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to
$$w_t, \epsilon_j = \sum_i w_i |h_i(x_i) - y_i|.$$
6. Choose the classifier, h_t, with the lowest error ϵ_t.
7. Update the weights:
$$w_{t+1,i} = w_{t,i}\, \beta_t^{1-ei}$$
where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
8. **end**
9. The final storage classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \ge \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

where x is the input image, $h_t(x)$ are the number of weak classifiers, and $\alpha_t = log\frac{1}{\beta_t}$.

Attentional cascade: The cascade is a critical module. The key intuition is that smaller and more effective, boosted classifiers are built that reject majority of the negative sub-windows while keeping almost all the positive examples. As a result, majority of the sub-windows are rejected in early stages of the detector, making the detection process extremely efficient.

B. Driver Face Recognition

Face recognition determines the match-likelihood of each face to a template element from a database. We employ the *Eigenfaces* approach [7], [8] that calculates an orthogonal set of *M* Eigenfaces for a given training set of *N* faces, where *M* << *N*. To recognize a face, first of all face images are centered and decomposed into small sets of featured images, that are actually the principal components (*Eigenfaces*) of initial training

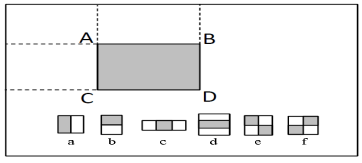


Fig. 2: Computation of integral image and Haar-like features

(a) (b) (c)

Fig. 3: (a)-(b) Positive and **(c)** Negative training samples

Later, all centered images are projected into face space by multiplying in Eigenfaces basis's. Euclidean distance between the projected test image and the projection of all cantered training images is calculated. Test image is supposed to have minimum distance with corresponding image in the training database. In Eigenfaces based algorithm as the image is projected on to the face space, there are three possible options.

1. If input image is near face space and near face class, then the individual is correctly recognized.
2. If input image is near face space, but not near known face class, then it is an unknown person.
3. If input image is distant from face space and near/distant from face class, then it is not a face image.

III. Parallel Architecture

Eigenfaces based approach is one of the most successful feature based algorithm for efficient face recognition. We employ parallel processing on Eigenfaces based face recognition algorithm. In the training phase, algorithm divides training images pool and assign each pool randomly to a group of processors in parpool(N), as described in Algorithm 2. In each parpool(N), we introduce data parallelism methodology in the Eigenface based algorithm for feature extraction of the training image. Finally, we concatenate the feature vector of each training image to form a feature matrix. In the testing phase, each test image is assigned to one parpool for feature extraction, recognition and matching.

IV. Simulation Results

We perform detailed experiments on our internal *Ubuntu Cloud* (UC) setup running on 96 core Supermicro Super-Server SYS-7047GR-TRF machine with on board 128 GB of RAM. To perform the evaluations, we use the driver face dataset that consists of 10, 000 faces. Our proposed scheme comprises three major modules: vehicle detection, driver face detection, and face recognition. In sections below, we analyze the results of each module separately.

A. Vehicle Detection Module

The sole purpose of vehicle detection is to locate the position of the vehicle and confine the processing area for subsequent face detection and recognition steps. Before using our vehicle detector at the parking entrance/exit, we test the feasibility of our developed vehicle detector on real life captured images as shown in Fig. 5(a). Clearly, vehicle detection is achieved under varying illuminations. Vehicle detector successfully detects vehicles under large deviation from frontal, side, and back view.

B. Face Detection Module

Once the vehicles are detected in the input image, in next step, we proceed to locate driver face. The face detection module processes detected vehicles as its input and locates the driver face position. Our developed face detector successfully detects face pose from frontal up to $\pm 45^0$. Fig. 5(b) shows driver face detection result. We can see that each image having different face size, position, and pose are successfully detected. Face detection module does not process extremely blurred and occluded driver face images. Object detection time shown in Table 1 indicates that vehicle and driver face detection is achieved in real-time.

Algorithm 2: Parallel version of the driver face recognition module

Input: Data Set of Training Images (P), Test Image (TI_i)
Output: Maximum Match (minimum Euclidian distance) of Probe Images
Definitions: Probe images in between pose range of: $+45^0$, $+35^0$, 0^0 (frontal), and -35^0, $I(x, y)$ = Training Image with x rows and y columns, T = 2-D matrix containing 1-D image vector with P training images of size ($xy \times P$), m = mean of training images, $Eigenfaces$ = Eigen vector of covariance matrix of training images, A = Matrix of Centered Image Vector. N = total number of assigned CPUs (Ranges from 1 to 96), $D_R \leq N$, Sim_j = Similarity measure of test image with all training images (j) assigned to *parpool k*. (*parpool* is parallel pool).

1. **Training Module**
2. $P_i \leftarrow$ GetTrainImages ($P_1, P_2, \ldots, P_m$)
3. $parpool(N)$
4. $D_R \leftarrow$RandomDivideTrainImages (P_i)
5. $parpool_i \leftarrow$AssignTrainImages(D_R)
6. $T_i \leftarrow$Convert2-DImage1-DArray(P_i)
7. $(m, A, Eigenfaces) \leftarrow$EigenFacecore($T$)
8. end parpool(N)
9. **Testing Module**
10. $parpool(N)$
11. $(n, a, Ef) \leftarrow$RandomDivision($m, A, Eigenfaces$)
12. $TI_i \leftarrow$GetTestImage()
13. $I_i \leftarrow$Convert2-DImage1-DArray(TI_i)
14. $Sim_j \leftarrow$Recognition(I_i, n, a, Ef)
15. $M \leftarrow$getMatch(Sim_j)
16. $CI \leftarrow$GetknownFace(M_N)
17. *Result*

C. Driver Face Recognition Module

Face recognition aims at finding a person's identity from a database of known subjects. To start with driver face recognition experiments the database is partitioned into two subsets: the gallery (training) set G and probe (test) set P. For face recognition, we simulate a very challenging task such that given only one gallery image, which is frontal mug shot and four different probe images with different pose (Fig. 6). The four different poses are: $+45^0$, $+35^0$, 0^0 (frontal), and -35^0. These four poses are chosen because a driver face angle can vary under these four ranges while sitting on the driving seat. Face recognition algorithm is first trained with G images and the resultant face recognizer is applied to T images to calculate the *Euclidian distances*. To develop a robust and practical framework, we develop the database of 10,000 subjects in our lab.

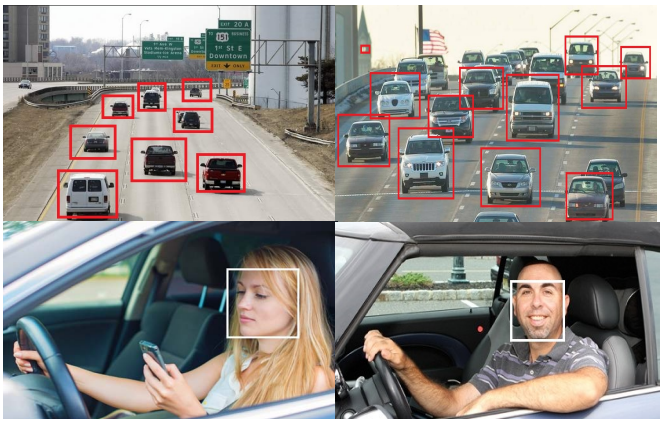


Fig. 5: (a): Vehicle detection **(b)** Driver face detection

Table 1: Vehicle and driver face detection time

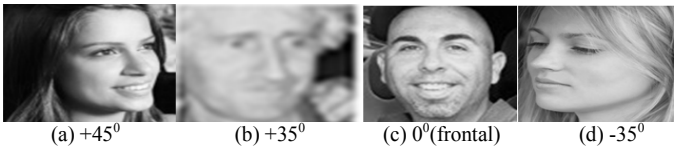
Object Detection Time (seconds)		Complete Object Detection Time (seconds)
Vehicle	Driver face	0.234887
0.121104	0.113783	

D. Parallel Analysis of Face Recognition Module

Our primary aim of parallel implementation of the face recognition module is to achieve a significant reduction in execution time and an improved speed up over serial implementation on large datasets. As the driver displays the face, size of the detected face is found to be 80x92 pixels.

1) Scalability Analysis

For a parallel algorithm to be scalable, with the increase in number of resources, such as the processors and the workload, the performance in terms of execution time and resources' utilization must be consistent and should not significantly degrade. We evaluate the scalability by analyzing the effects of increase in facial images and processors on the time consumption. Fig. 7(a) shows the scalability results by varying facial images and number of processors to observe the recognition time. The results show that by increasing the number of facial images five and ten times results in sudden increase in the processing time. However, substantial decrease in time consumption were observed by increasing the number of processors. On average, by increasing the number of facial images five times increases the time consumption by approximately 35.07% whereas increasing one processor results in an average decrease of 15.27% for the identification task.



(a) $+45^0$ (b) $+35^0$ (c) 0^0(frontal) (d) -35^0

Fig. 6: Sample probe images illustrating four different poses of drive

2) Speedup Analysis

We systematically vary the training images from 1, 000 to 10,000 face images. Fig. 7(b) depicts the speed up comparison of the training phase for parallel face recognition with the serial execution. Clearly, for 1000, 5000, and 10,000 (which we refer 1K, 5K, and 10K) training facial images, the speedup is boosted to 3.55X, 4.75X, and 11.95X, respectively. The speedup comparison for testing phase is shown in Fig. 7(c). As per our framework scenario, we fed four different pose images to the face recognition module. For 1K, 5K, and 10K facial images with four different pose images as probe, the speedup is 3.85X, 5.05X, and 12.25X, respectively. Training and test phase speedups are particularly important for the cases where multiple faces with different pose are detected. Therefore, developed framework can result into significant high performance gain.

V. CONCLUSIONS AND FUTURE WORK

We presented a framework of object detection and recognition for a secure vehicle parking. The proposed framework has three major steps: vehicle detection, driver face detection, and driver face recognition through parallely applied architecture. Simulation results show feasibility of developed framework to be deployed in real life. Given an appropriate training face data set, our developed framework can be a best fit for applications, such as real-time surveillance and video analytics. In future, we aim to develop the complete parallel version. Moreover, we are interested in the energy consumption of the developed system. Furthermore, research could be carried out to develop a robust object detector and recognizer from tinted vehicles.

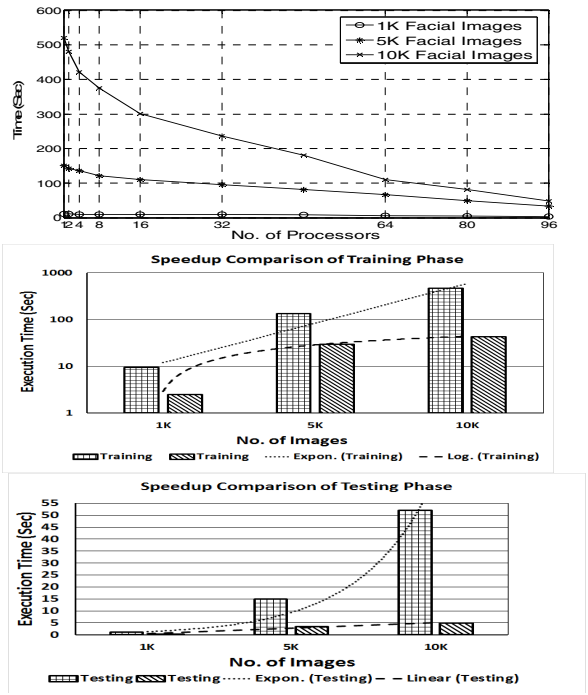


Fig. 7: (a) Scalability analysis of driver face recognition algorithm **(b)-(c)** Speedup comparison of training and testing phase, respectively

REFERENCES

[1] Z. Mahmood, T. Ali, S. Khattak, L. Hassan, and S. U. Khan, "Automatic player detection and identification for sports entertainment applications," *Pattern Analysis and Applications*, 2015, DOI: 10.1007/s10044-014-0416-4.

[2] Z. Mahmood, T. Ali, and S. Khattak, "Automatic player detection and recognition in images using AdaBoost," *9th International Bhurban Conference on Applied Sciences & Technology* (*IBCAST*), Islamabad, 2012, pp. 64-69.

[3] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis,* Vol. 57, 2004, pp. 137–154.

[4] C. A. A. Chandio, K. Bilal, N. Tziritas, Z. Yu, Q. Jiang, S. U. Khan, and C. Z. Xu, "A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems," *Cluster Computing*, Vol. 17, No. 4, 2014, pp. 1349-1367.

[5] L. Wang, D. Chen, R. Ranjan, S. U. Khan, J. Kolodziej, and J. Wang, "Parallel processing of massive EEG data with MapReduce," *18th IEEE International Conference on Parallel and Distributed Systems* (*ICPADS*), Singapore, Dec 2012, pp. 164-171.

[6] D. Chen, X. Li, L. Wang, S. U. Khan, J. Wang, K. Zeng, and C. Cai, "Fast and scalable multi-way analysis of neural data," *IEEE Transactions on Computers*, Vol. 63, No. 3, 2015, pp. 707–719.

[7] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience,* Vol. 3, No. 1, 1991, pp: 71-86.

[8] Z. Mahmood, T. Ali, S. Khattak, and S. U. Khan "A comparative study of baseline algorithms of face recognition," *12th International Conference on Frontier of Information Technology* (*FIT*), Islamabad, 2014.