Chapter 4

Enhanced Viola-Jones Face Detection Algorithm with Particle Swarm Optimization

4.1 Face Detection

Face detection is a computer vision task that involves identifying faces in images or video frames. It is a key component of object detection and has a wide range of applications, including face recognition, facial expression analysis, security, biometrics, entertainment, and more [150]. Achieving high detection accuracy in complex backgrounds is crucial in real-time scenarios, where challenges such as pose variations, changing illumination, occlusions, and scale variations can significantly hinder performance.

Face detection relies on AI algorithms, machine learning, statistical analysis, and image processing techniques to locate human faces within large images. These faces are typically marked with bounding boxes to differentiate them from non-face elements like landscapes, buildings, and other body parts. The process often begins by detecting human eyes, followed by the identification of facial landmarks such as eyebrows, irises, mouth, nose, and nostrils. Once these features are detected, the algorithm labels the region as a face, which is then subjected to an additional confirmation test to ensure accurate results.

On the other hand, to improve prediction accuracy, face detection algorithms are often trained using large datasets consisting of both face and non-face images. These datasets include a variety of images with different facial expressions, poses, lighting conditions, and backgrounds, as well as images containing non-face elements such as landscapes, buildings, or other objects. By training on such diverse datasets, the algorithm learns to distinguish between faces and non-faces with greater precision. This process helps the algorithm identify various facial features, adapt to different environments, and handle challenges such as occlusions or scale variations. As a result, the trained face detection algorithm becomes capable of accurately determining the presence of faces in both still images and video frames, even in complex or dynamic settings.

The remainder of this chapter is organized as follows: Section 4.2 presents the face detection approach and its limitations. Section 4.3 explains appearance-based face detection

approaches such as Viola-Jones and HOG face detection, along with their taxonomy. Section 4.4 describes the performance of the aforementioned face detection approaches. Section 4.5 presents the proposed approach combining Viola-Jones with the PSO approach. Section 4.6 discusses the results of the proposed approach in complex environments and also presents the results in a real-time environment. Finally, Section 4.7 provides a summary of this chapter, including findings and results.

4.2 Face Detection Approach

The face detection algorithm follows different kinds of approaches as shown in Figure 4.1

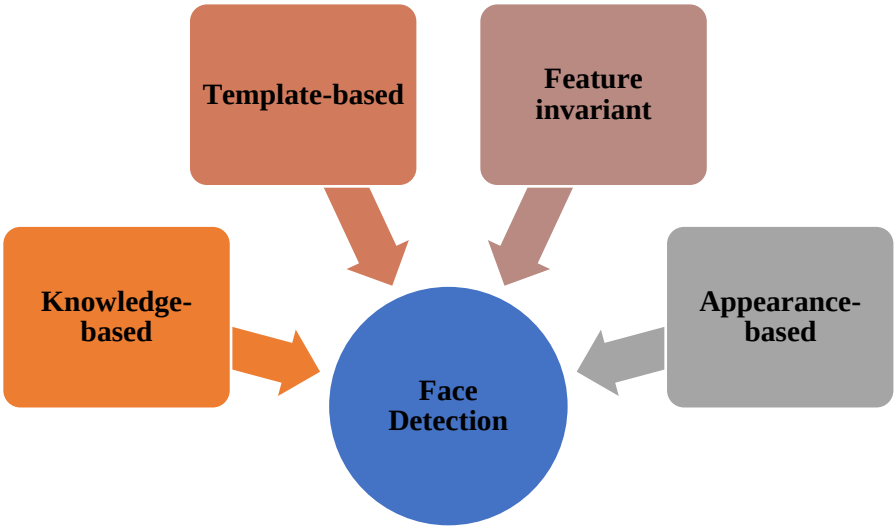


Figure 4.1 Type of Face Detection Approach [149]

4.2.1 Knowledge-based Approach

A knowledge-based approach works based on a set of predefined rules, also designed by human capability and understanding [151]. For instance, the face has features like a nose, mouth, and eyes that are located at certain distances and positions from one another. However, it has some drawbacks, such as developing an effective set of rules to detect faces. In this case, the designed rule was either too vague or too specific, leading to a high false positive rate. Due to this reason, a knowledge-based approach is ineffective and incapable of identifying faces in large face images.

4.2.2 Template-based Approach

The template-based approach uses well-predefined or parameterized face templates to detect or locate faces by comparing them with input samples [152]. For instance, the facial part can be divided into four parts: eyes, nose, lips, and facial contour. Moreover, edge detection is used to construct entire faces with edges. These are simple techniques to implement and detect faces. However, this method is insufficient for face detection when the face is oriented at a different angle.

4.2.3 Feature Invariant Approach

Feature-based techniques use an approach to detect the face using a person's eyes and nose by extracting structural features of the face [153]. It is initially trained as a classifier and then used to differentiate between face and non-face images. However, it has been found to have a high false positive rate when faced with a highly illuminated environment.

4.2.4 Appearance-based Approach

The appearance-based approach uses statistical and machine learning techniques to identify relevant features or characteristics of face images [154]. This approach is considered effective when compared to other methods. Most face-detection approaches have been developed based on this. It employs a set of classifiers trained on a larger dataset, encompassing various complex scenarios such as illumination, partial occlusion, pose variation, and scale orientation.

Among the various face detection approaches, appearance-based methods have been increasingly used in recent times to effectively identify faces in images and video frames.

4.3 Face Detection Algorithms Comparison

The appearance-based face detection approach is considered highly effective in current applications. According to the literature review, two widely applied face detection methods—Viola-Jones [148] and Histogram of Oriented Gradients (HOG) [155]—have been selected for this research study to analyze their performance. This work explains both the algorithms and experiments evaluating their effectiveness on various complex images. The comparative approach used in this study is illustrated in Figure 4.2.

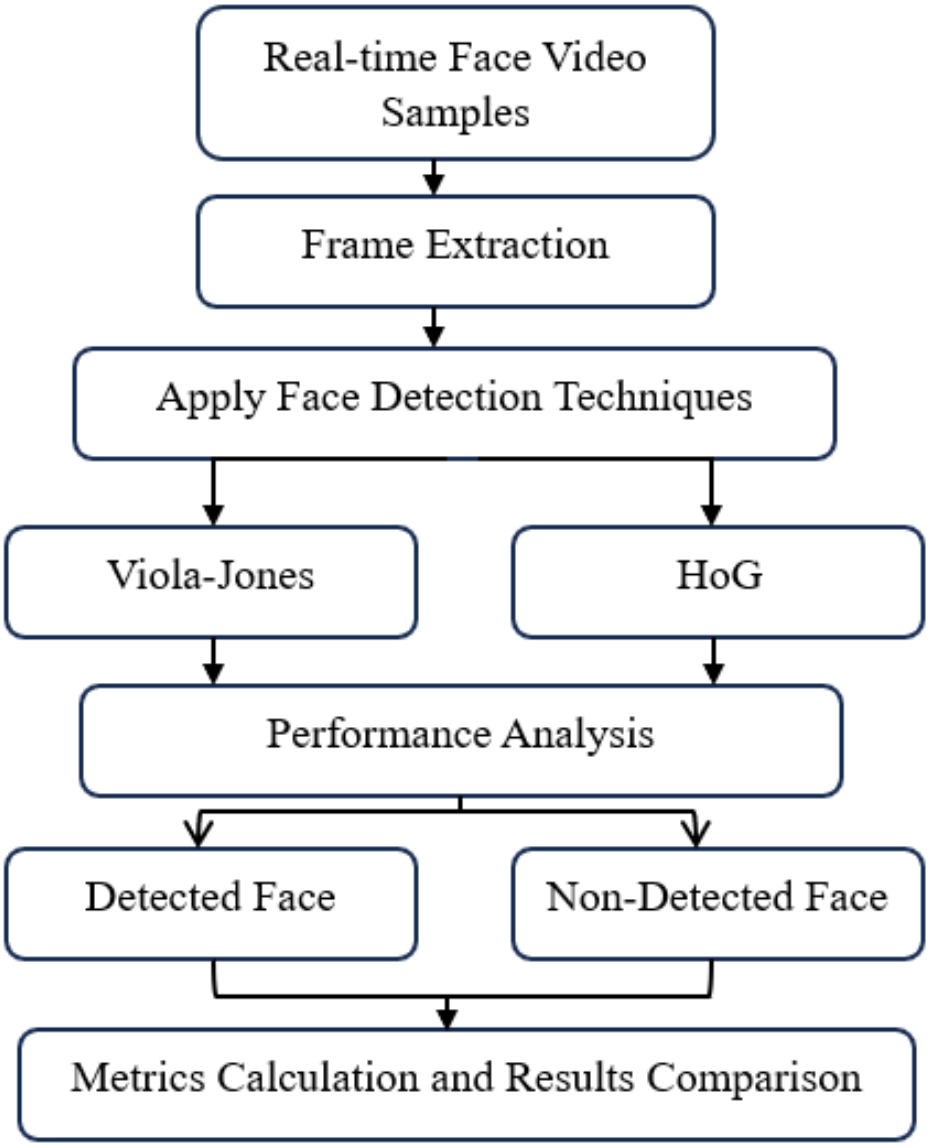


Figure 4.2. Face Detection Comparison workflow

4.3.1 Viola-Jones Algorithm

The Viola-Jones Algorithm is a type of face detection algorithm based on machine learning techniques. It was designed by Paul Viola and Michal Jones in 2001 and was presented at a conference on computer vision and pattern recognition in the paper 'Rapid Object Detection using a Boosted Cascade of Simple Features [148]. Although many algorithms have been developed to tackle face detection challenges, as shown in Figure 4.4, the Viola-Jones algorithm has emerged as a groundbreaking method due to its speed, accuracy, computational efficiency, and cost-effectiveness.

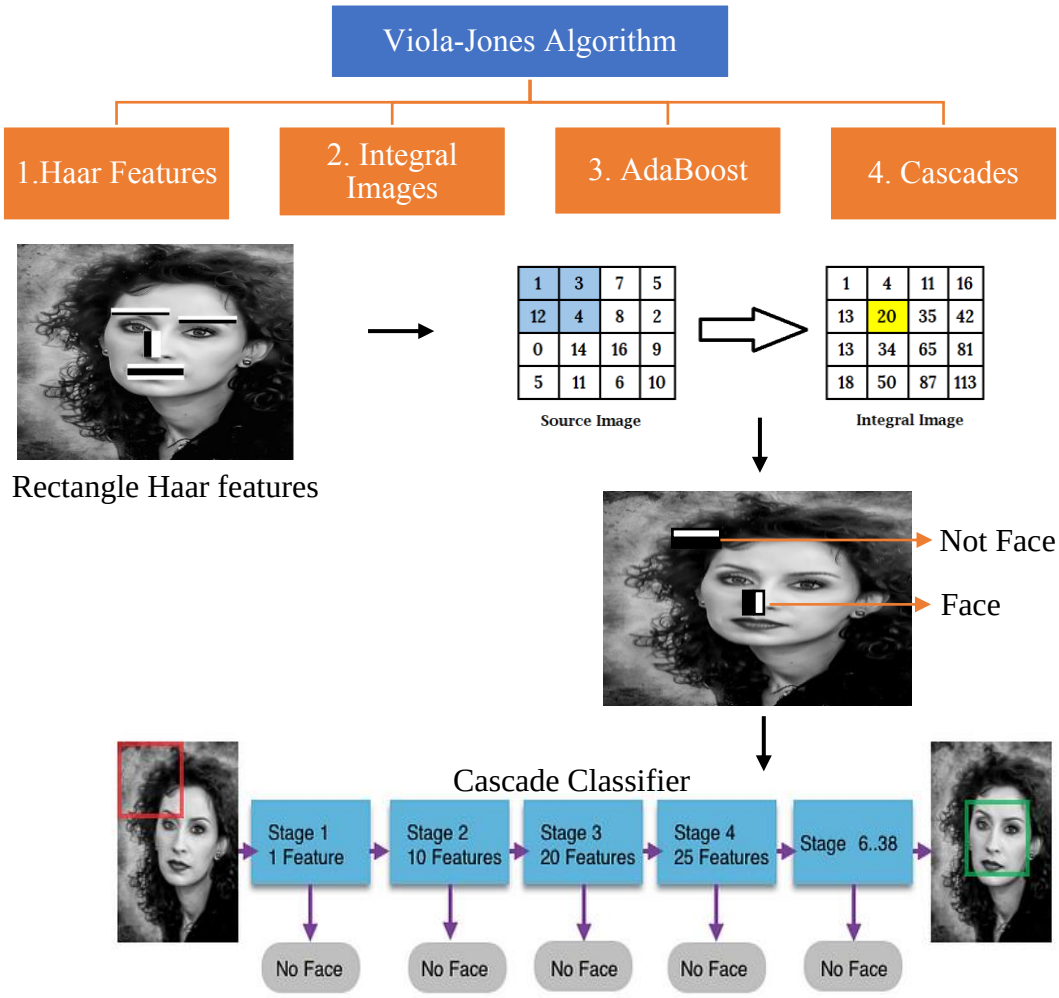


Figure 4.3. Components of Viola-Jones Algorithm

The Viola-Jones algorithm is a widely used method for object detection, particularly for face detection in images. It is specifically designed to work on grayscale images, simplifying the computational complexity by reducing the color dimensions. The algorithm divides an image into numerous smaller subregions, systematically analyzing each region to identify patterns that correspond to facial features, such as eyes, nose, and mouth. This ensures that even small faces or partially visible ones can be detected. To accommodate variations in face sizes and positions, the algorithm employs a scaling approach, examining the image at different resolutions and adjusting its focus to ensure comprehensive coverage. This adaptability makes it robust in handling images with multiple faces of varying dimensions. The algorithm comprises four sections, as shown in Figure 4.3.

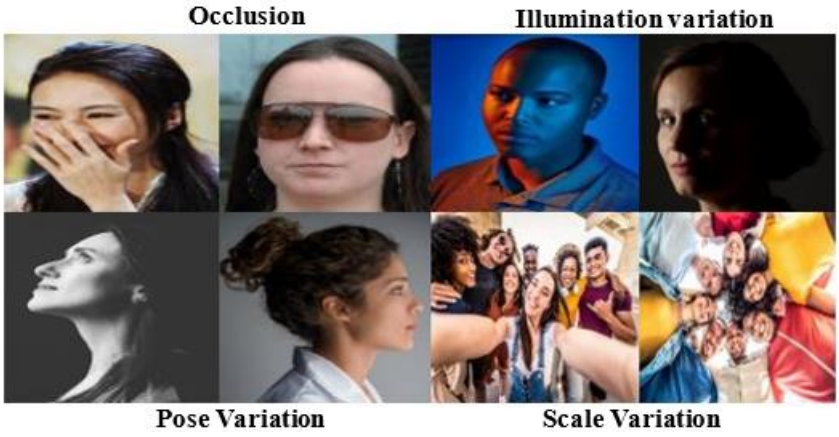


Figure 4.4. Sample face detection challenge images [149]

4.3.1.1 Haar Features

Haar features are a collection of pixels in rectangular shapes and computationally similar to the kernels in a convolutional neural net. Human faces often exhibit distinct characteristics. For example, the area around the eyes tends to be darker than the bridge of the nose, and the cheeks are generally brighter than the eye area. These unique properties are harnessed for face identification in images and videos. This approach involves the simple task of identifying variations in darkness and lightness within an image by summing up pixel values from specific regions, known as Haar Features.

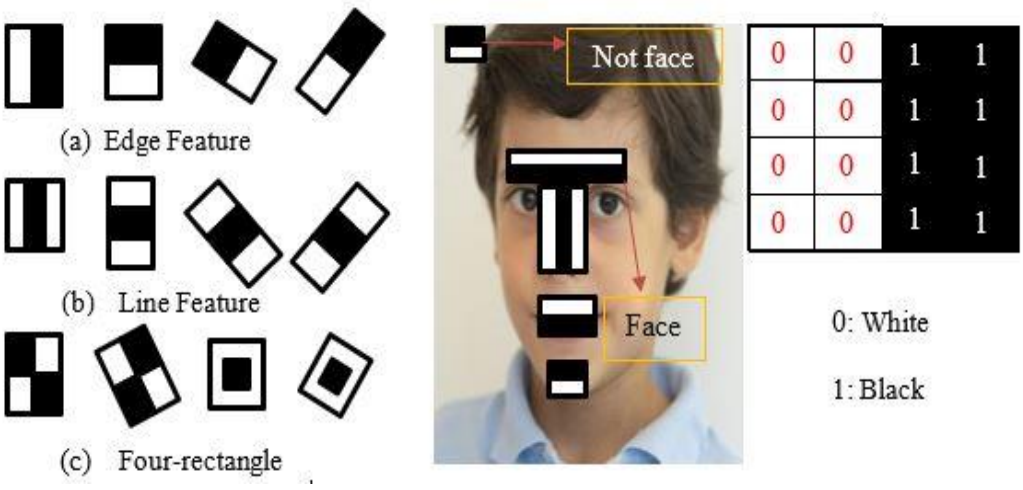


Figure 4.5. Harr-like Features in Viola-Jones Algorithm

There are three main types of Haar-like features widely used: Edge features, Line features, and Four-rectangle features, as depicted in Figure 4.5. In grayscale images, the white bars represent pixels that are closer to white, while the dark bars represent the opposite.

A Haar feature is calculated by subtracting the sum of the white area from the sum of the black area. A single 24x24 window can extract over 180,000 features, which consumes a significant amount of computational time for feature selection. However, not all of these features are equally valuable for detecting faces in images and videos. The subsequent step involving integral images helps address this issue.

4.3.1.2 Integral Images

The integral image plays a crucial role in Haar-like feature calculations and is often referred to as a summed-area table. Integral images allow for the efficient computation of the sum of pixel weights in constant time with minimal computational overhead (as shown in Figure 4.6(a)). The average time complexity of integral images is $O\,(N^2)$. The original integral image pixel values are obtained by summing the values of pixels to the left and above the point (x, y) using equations (4.1-4.2).

$$s(x,y) = s(x,y-1) + i(x,y) \tag{4.1}$$

where $ii(x,y)$ represents the integral image pixel weights, and $s(x,y)$ represents the actual pixel weights at point (x,y). This feature is commonly computed with the assistance of integral images.

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{4.2}$$

where $s(x,y)$ denotes the cumulative row sum and $s(x,-1) = ii(-1,y) = 0$.

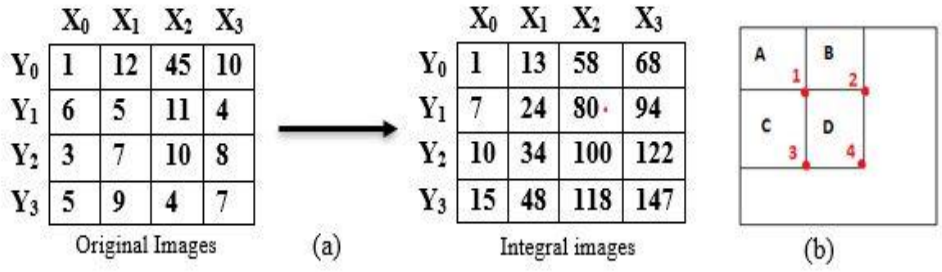


Figure 4.6 (a) Convert the original to integral images (b) Computing sum with given rectangle values

The sum of pixel weights within a four-rectangle area can be calculated more efficiently than summing up all pixel values individually. If A, B, C, and D represent the four corner values of the summed table, the values within this rectangle can be computed. For example, to calculate the sum of area D, there is need to track four connections using the formula: 4 + 1 - (2 + 3) (as shown in Figure 4.6 (b)).

4.3.1.3 AdaBoost

AdaBoost is a machine learning technique that extracts the most valuable features from a pool of potential features gathered from every sub-window, utilizing integral and Haar-like features. It is a widely used boosting algorithm designed to eliminate unnecessary components, selecting only weak classifiers from this feature pool. These weak classifiers are often referred to as the best features. To identify the weak classifiers, the algorithm runs through T iterations, where T represents the number of weak classifiers. During each iteration, the algorithm evaluates the error rates for all features and selects the one with the lowest error rate. This process results in the creation of hundreds or thousands of specialized classifiers for detecting faces in input images (as shown in Figure 4.7). However, managing all of these classifiers for each region in all images can be computationally intensive. To address this efficiency concern within the AdaBoost algorithm, a cascade classifier was proposed [156].

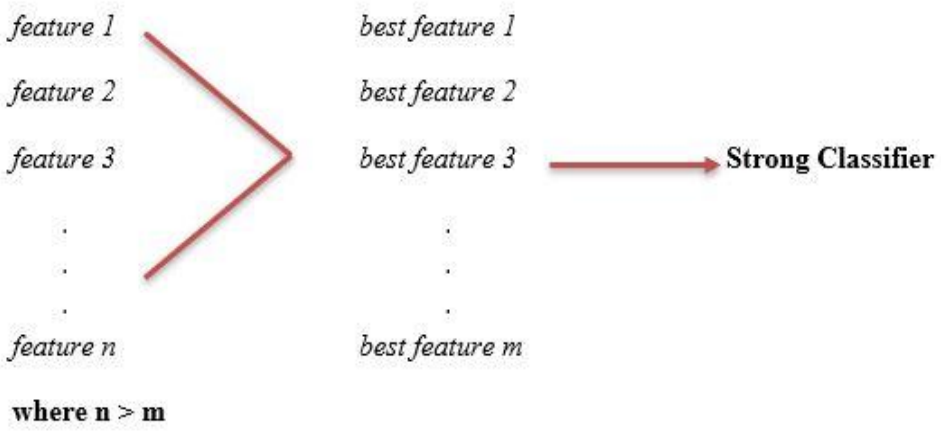


Figure 4.7. AdaBoost algorithm

4.3.1.4 Cascades

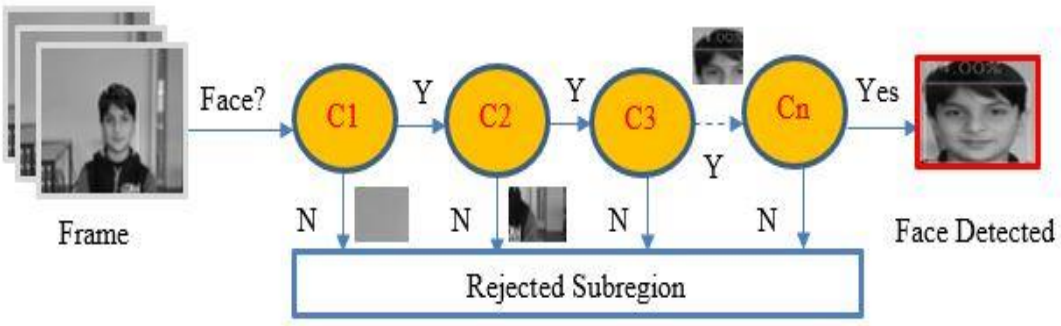


Figure 4.8 Function of Cascading Classifiers

Cascading classifiers are designed to eliminate non-face images during the face detection process, thereby avoiding unnecessary computations. They employ a multistage classifier structure that enables rapid detection, with each stage housing a robust trained classifier generated by the AdaBoost algorithm. If any sub-window fails to meet the classifier's criteria at any stage, it is promptly discarded. This approach significantly accelerates the overall process, allowing the machine to produce results much more quickly. The process of the Cascading classifier is illustrated in Figure 4.8.

4.3.2 Histogram of Oriented Gradients

Histogram of oriented gradients is employed to extract features in the form of a vector, which is then fed into classification algorithms such as SVM to assess whether the face is present in the given image or frames. The vector can be calculated very easily. Indeed, each pivotal aspect is associated with a generated HOG feature. The surrounding region of each key point is partitioned into small blocks known as cells. Within each cell, a local 1-D histogram of gradient directions is constructed for the pixels. As a result, the descriptor comprises the amalgamation of these individual histograms. The gradient vector is determined as outlined below:

$$\mathrm{G_x(x,y)} = \mathrm{H(x+1,y)} - \mathrm{H(x-1,y)} \tag{4.3}$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \tag{4.4}$$

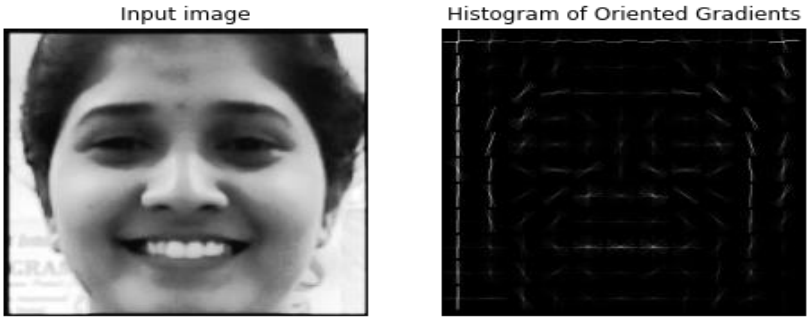
The equation (4.3) represents the horizontal gradient of the image pixel $G_x(x,y)$, while the vertical gradient$G_y(x,y)$, is expressed in equation (4.4). The gradient amplitude and direction for the pixel (x,y) are correspondingly described in equations (4.5) and (4.6).

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{4.5}$$

$$a(x,y) = tan^{-1}\left(\frac{G_x(x,y)}{G_y(x,y)}\right) \tag{4.6}$$

The provided frame undergoes division into regions of either rectangular or circular shape, each comprising N*N pixels, referred to as cells. Subsequently, gradient feature vectors are computed for each of these cells (refer to Figure 4.9). The resulting gradient feature vectors are aggregated to form the feature vector for an individual frame. Ultimately, the gradient feature vectors extracted from various images are concatenated to form a unified,

elongated vector, constituting the HOG feature vector. Later, it feeds into SVM for face detection.



(a) Input Images (b) HOG image and vector

Figures 4.9. HOG Feature Extraction

4.4 Comparison of Viola-Jones and HOG Face Detection Algorithm Performance

4.4.1 System Configuration

Python OpenCV in the Google Colaboratory (Colab) platform, using Nvidia K80 Graphics Processing Units (GPUs), was employed for this implementation. The results were compared and analyzed using both CPU and GPU processor times in both real-time and benchmark datasets. Additionally, the research utilized graphics processing units (GPUs) paired with an Intel(R) Core (TM) i5-8400 CPU clocked at 2.80GHz within a Dell desktop environment.

4.4.2 Dataset Description

The children's facial expression dataset used in this comparative study and analysis was collected from LIRIS-CSE [157], with sample video frames depicted in Figure 4.10. This dataset includes 208 movie clips capturing six primary, spontaneous facial expressions exhibited by children aged 6 to 12 years in an unconstrained environment, allowing for natural head and hand movements as well as relaxed, free-sitting postures.

Additionally, real-time datasets were acquired from the Centre for Machine Learning and Intelligence (CMLI) laboratory and involved children aged 8 to 10 years. Before collecting this dataset, ethical clearance was obtained from the Human Ethics Committee of Avinashilingam University under approval number AUW/IHEC/CS-21-22/XMT-03, along with consent forms from the parents of the participating children. This dataset includes a variety of pose variations, and for comparative analysis, scenarios with altered illumination

and partially occluded faces were artificially created using video editing tools. These videos typically contain 150 to 250 frames and run for approximately 5 to 8 seconds.

(a)

(b)

Figure 4.10 Dataset with different challenging conditions: (a) LIRIS-CSE dataset [157], and (b) collected real-time data samples with pose variations, artificially added illumination changes, and occlusions

4.4.3 Evaluation Metrics

Evaluation metrics are used to measure the face detection model's performance. For this comparison study, two metrics are considered: True Positive Rate (TPR), also referred to as sensitivity or recall, is used to estimate the percentage of true positives that are accurately identified. It is measured using equations 4.7 and 4.8 as follows.

$$True\ Positive\ Rate\ (TPR) - Sensitivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad (4.7)$$

False Negative Rate (FNR) also called miss rate is used to measure the actual positive missed by the test. It is measured as follows.

$$False\ Negative\ Rate\ (FNR) - miss\ rate = \frac{False\ Positive}{False\ Positive + True\ Negative} \quad (4.8)$$

4.4.4 Performance Analysis of Face Detection Algorithms

The face detection algorithm has been tested using 1,010 video frames under various conditions, including pose variations, illumination changes, and occlusion. It was also evaluated with a real-time video dataset containing various face-detection challenges like pose variation, illumination, and occlusions. The results obtained for the Viola-Jones and HOG algorithms are presented in Tables 4.1 and 4.2, respectively.

Table 4.1 presents the performance of the Viola-Jones algorithm in various face-detection challenges. It achieved an average accuracy of 90.35%. In the first scenario, with pose variation, it achieves a true positive rate of 94.66% and a false-negative rate of 5.33%. However, when the face angle exceeds 40 degrees, the Viola-Jones algorithm struggles to detect the face. In the second case, when the face is subject to varying illumination, the algorithm achieves a precision of 90.28%. It faces challenges in high lighting conditions because the process of converting the frame from color to grayscale may result in some black pixels turning white. Haar-like features in the Viola-Jones algorithm typically perform this conversion before feature extraction. Finally, the Viola-Jones algorithm was tested with artificially created occlusions in face videos, where it achieved a true positive rate of 86.11% and a false-negative rate of 13.88%. These results highlight the algorithm's moderate error rate when dealing with various face detection challenges.

Table 4.1 Performance of Viola-Jones algorithm in various face detection challenges

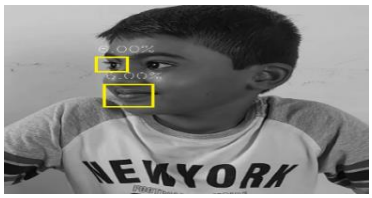
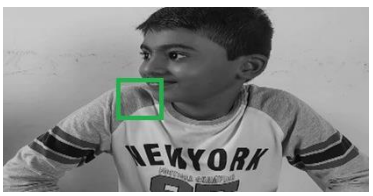
Category	Total Frame	Detected Face	Non-Detected Face	Sensitivity (%)	Miss rate (%)
Pose variation	300	284	16	94.66	5.33
Illumination	350	316	34	90.28	9.71
Occlusions	360	310	50	86.11	13.88
Average				**90.35**	**9.64**
Detection Time (Avg)		10.23 Min			

Table 4.2 illustrates the performance of the HOG algorithm, showcasing a moderate level of accuracy compared to the Viola-Jones algorithm. In the first scenario, involving pose variation, HOG achieves a true positive rate of 66.66%. In the second and third scenarios, addressing illumination and occlusion, it attains rates of 84.28% and 75.00%, respectively. The average accuracy across all conditions is 75.31%. Moreover, in comparison to the HOG algorithm, Viola-Jones exhibits significantly lower error rates. Consequently, Viola-Jones proves to be efficient in detecting faces at different orientations, handling variations in illumination, and managing occluded faces in video frames. However, it demonstrates less effectiveness on some complex images, as indicated in Table 4.3.

Table 4.2 HOG in various face detection challenges

Category	Total Frame	Detected Face	Non-Detected Face	Sensitivity (%)	Miss Rate (%)
Pose variation	300	200	100	66.66	33.33
Illumination	350	295	55	84.28	15.71
Occlusions	360	270	90	75.00	25.00
Average				**75.31**	**24.68**
Detection Time (Avg)		18.54 Min			

Table 4.3 Performance of Viola-Jones and HOG Face Detection Algorithms

Challenges in Face Detection	Viola-Jones	HoG
Right Side Face		
		

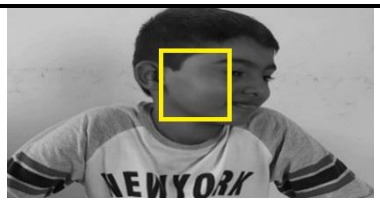
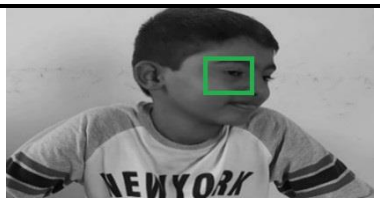
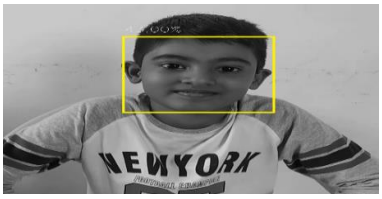
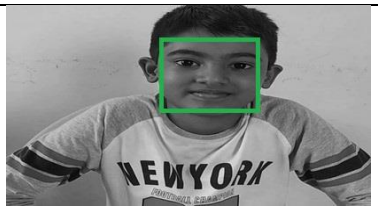
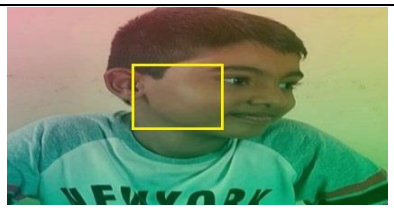
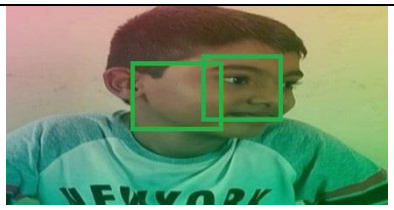
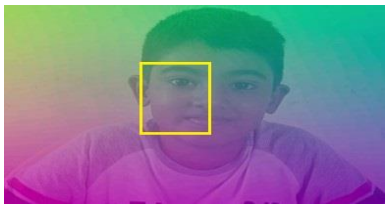
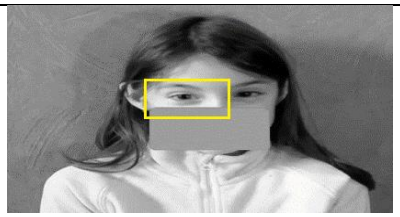
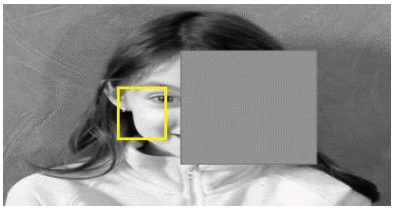
Challenges in Face Detection	Viola-Jones	HoG
Left Side Face		
		
Front Face		
Illumination		
		
Partial Occlusion		
		

Table 4.3 presents a comparative analysis of the Viola-Jones and HOG algorithms. The analysis indicates that the Viola-Jones face detection is significant at various angles, handling different illumination conditions, and coping with partially occluded faces, outperforming the HOG algorithm in these aspects. However, both algorithms perform better in front-face detection. The Viola-Jones algorithm achieves a high detection rate in frame sequences and attains an average accuracy of 90.35%, whereas the HOG algorithm achieves an average accuracy of 75.31% when confronted with various face detection challenges in frames. Despite its computational efficiency and real-time processing capabilities, the Viola-Jones algorithm has some limitations, making it suitable for resource-constrained applications.

4.4.4.1 Limitation Identified from Viola-Jones Algorithm Through Analysis

- Sensitive to variations in lighting conditions, which can affect the performance of the algorithm.
- Produce false negatives (missing faces that are present).
- Limit its ability to detect faces at different scales efficiently and provide multiple detection.
- Struggle when faced with occlusions, where faces are partially hidden.

4.4.4.2 Identified Reason for this Challenges in the Viola-Jones Algorithm

- AdaBoost needs to search through all possible thresholds for all samples to find the minimum training error.
- Exhaustive search (1,16,000) consumes a significant amount of time to discover the best threshold values and optimize feature selection to build an efficient classifier for face detection.

4.5 Improving Viola-Jones Algorithm Using Particle Swarm Optimization (PSO) Approach

To address the above issues and improve prediction accuracy while reducing the training error of the Viola-Jones algorithm, a nature-inspired algorithm, such as PSO, has been integrated into this algorithm. PSO is applied in two different ways to enhance the prediction accuracy of the Viola-Jones algorithm on complex images.

- Firstly, PSO is employed to dynamically select optimal threshold values for feature selection, thereby improving computational efficiency.

- Secondly, the feature selection process using AdaBoost within the Viola-Jones algorithm, integrating PSO to identify the most discriminative features for constructing a robust classifier.

4.5.1 Particle Swarm Optimization

This algorithm operates using a population-based approach to identify the optimal parameters for a given function. It leverages a naturally inspired optimization technique called Particle Swarm Optimization [158]. PSO is a stochastic optimization method that draws inspiration from the collective behavior of swarms, such as flocks of birds or schools of fish, and was first introduced by James Kennedy in 1995.

In the PSO algorithm, each candidate solution is conceptualized as a "particle" within the multidimensional search space. These particles are characterized by two primary attributes: cost values, which are determined by a predefined cost or objective function, and velocities, which influence their movement and direction within the search space.

The PSO process begins with the initialization of a population of particles, which are randomly distributed throughout the solution space. The velocities of these particles are also initialized randomly, ensuring diverse exploratory movement within the search domain. As the algorithm progresses, the movement of each particle is influenced by two main factors:

1. Global Best Position (gBest): This represents the most promising solution encountered by the entire swarm up to the current iteration.
2. Personal Best Position (pBest): This reflects the best solution discovered by an individual particle during its own search trajectory.

The particles iteratively update their positions by adjusting their velocities based on these two guiding factors. The velocity updates take into account the current velocity, the attraction towards the pBest position, and the attraction towards the gBest position. This dual influence ensures a balance between exploration (searching new areas of the solution space) and exploitation (refining solutions in promising regions).

According to the following equation, the velocity and position of each particle 'i' are updated at iteration 't':

$$V_i(t+1) = V_i(t) + c_1 s_1\big(P_i(t) - Q_i(t)\big) + c_2 s_2\left(P_i^g(t) - Q_i(t)\right) \quad (4.9)$$

$$Q_i(t+1) = Q_i(t) + V_i(t+1) \quad (4.10)$$

where, s1 and s2 represent random values within the range [0,1], while c1 and c2 represent cognitive constants, and P and V denote the position and velocity of the particles, respectively. In each iteration, all particles undergo dynamic modifications based on the aforementioned position and velocity, as defined by the algorithm above. Consequently, in most cases, the velocity quickly attains extremely high values, especially for a population distant from its global optimum.

Through this iterative process, the particles converge toward optimal or near-optimal solutions. The algorithm dynamically tunes the speed and direction of each particle, enabling them to move toward their ideal positions in the search space. As a result, PSO effectively discovers high-quality solutions for complex optimization problems, making it a widely used method in various fields, including engineering, machine learning, and operations research.

4.5.2 Selecting Threshold Values using PSO

In the proposed method, for the selection of threshold values, a weak classifier has significantly improved the computational efficiency of the base algorithm by utilizing a decision tree with two leaves, commonly known as a decision stump instead of exhaustively searching for a multitude of features to construct a weak classifier, we employ PSO to pinpoint the optimal decision stump threshold. When decision stumps are employed as weak classifiers on complex datasets, the algorithm must explore all possible thresholds to minimize training error. Consequently, finding the best threshold values can be time-consuming. In such cases, an evolutionary search strategy PSO is invaluable as a proposed approach, accelerating the training of an AdaBoost classifier.

Furthermore, each iteration of the PSO approach is dedicated to learning a new weak classifier, and through numerous runs, it may uncover the ideal set of values that collectively form a strong classifier.

In the PSO algorithm, the cost function is utilized to optimize each particle within the entire solution space. Particles employ thresholding values to categorize the solution space into two classes: 1 (representing 'face') and 0 (representing 'non-face'). In the initial stage,

sample values greater than the threshold are classified as 1, while values below the threshold are classified as 0. This classification is reversed in the subsequent stage during the training of weak classifiers. The training loss is computed for each subgroup, and the weak classifier output with the lowest error is selected. For instance, let $S = ((x_1, y_1) \ldots . (x_n, y_n))$ represent a training set of weak classifiers, where the labels $y_i \in \{0,1\}$. To calculate each particle, a decision stump requires three parameters: the decision limit (+1 or 0), index characteristics (j), and the optimized threshold value to split the solution space. For input examples x, Equation (4.11) defines the positive cost function, while Equation (4.12) defines the negative stump.

$$h_{j,\theta} + (x) = \begin{cases} +x, & if\ x(j) \geq\ \theta \\ x, & otherwise \end{cases} \tag{4.11}$$

$$h_{j,\theta} - (x) = \begin{cases} -x, & if\ x(j) \geq\ \theta \\ x, & otherwise \end{cases} \tag{4.12}$$

The computational cost of the enhanced Viola-Jones algorithm depends on two factors: the population size (S) and the number of iterations (T). Each step in the boosting procedure optimizes the SxT classifiers. PSO is employed to select the best threshold Haar-like features in the AdaBoost.

4.5.3 Selecting the best features in the AdaBoost algorithm using PSO

Enhancing the speed of the face detector without compromising classifier accuracy is a crucial objective. However, the exhaustive feature selection process in AdaBoost often leads to increased complexity. Furthermore, the limited learning capacity of the simple decision stump classifier reduces the efficiency of the conventional face detection approach.

To address this, PSO has been incorporated in AdaBoost for the selection of the optimal features for face detection and optimizing the computational processing time. Considering these factors, we propose two improvements to our face detector to reduce the computational burden of feature selection and enhance the selection speed. Firstly, we employ PSO to select optimized threshold values, as discussed in the previous section. Lastly, we combine the PSO technique with the AdaBoost algorithm, enabling rapid exploration of the entire feature space and the selection of the most optimal feature sets, thus expediting the training process and

minimize the training error. Algorithm 1 illustrates the proposed Viola-Jones using the PSO approach.

In the AdaBoost classifier, exhaustive searches are conducted each time to select relevant features and minimize classification errors. To address the high complexity associated with this exhaustive search, we introduced the use of PSO within the AdaBoost algorithm. PSO is applied to explore potential feature locations, sizes, orientations, and combinations, resulting in the selection of a discriminative feature set. These selected features are then incorporated into AdaBoost to construct an ensemble classifier. The PSO demonstrates efficient search capabilities compared to exhaustive search techniques.

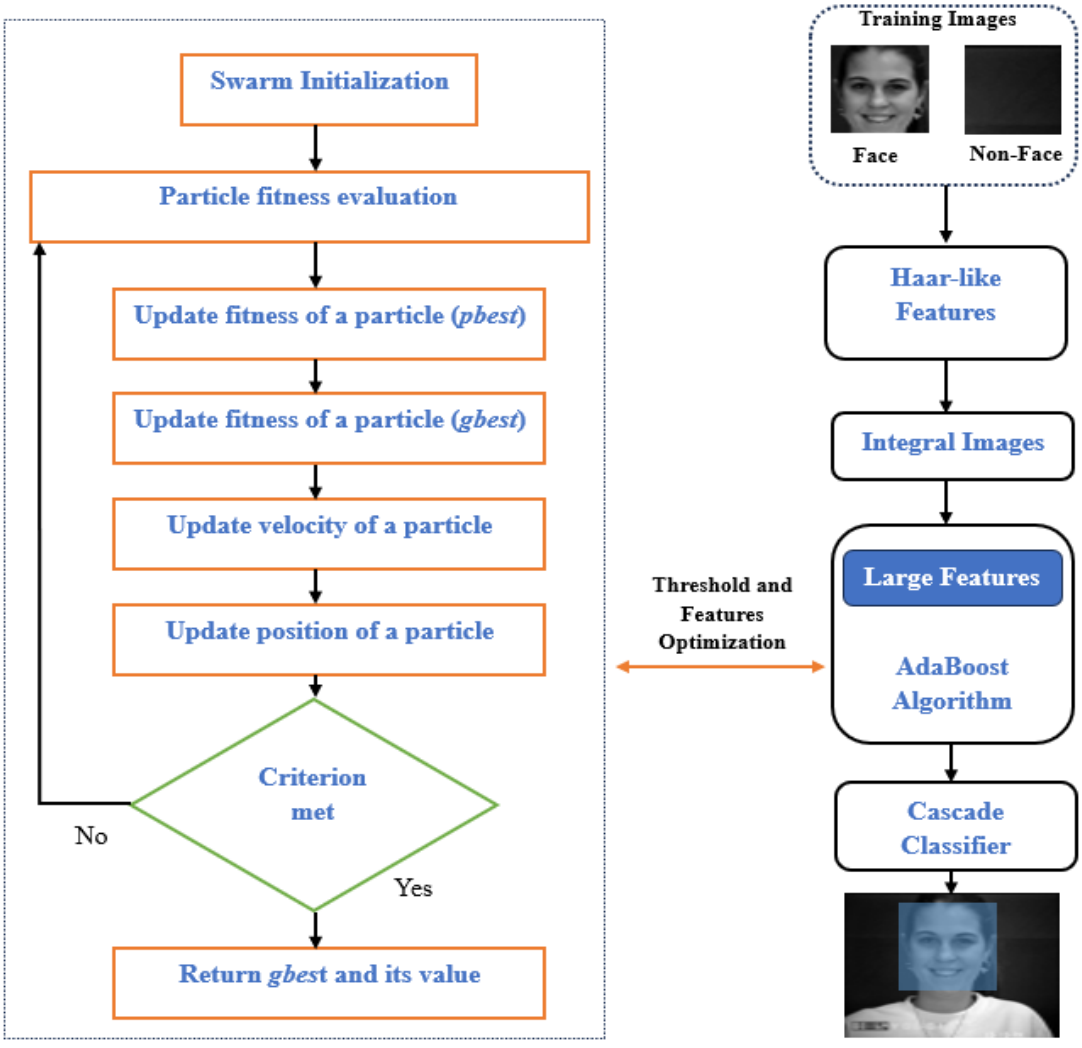


Figure 4.11 Workflow of Enhanced Viola-Jones algorithm with PSO

The proposed algorithm of Viola-Jones using PSO is given below.

Input: Original sample images
Output: Identified face will be shown in the bounding box
for i← 1 to number of different scale images **do**
Load sample image to create $images_i$
Calculate integral image, $images_{i,i}$
Given N-labeled samples $(x_1, y_1), \ldots (x_i, y_i) \ldots, (x_N, y_N)$ where $y_i \in \{0,1\}$ (0,1 class labels)
Initialize $w_{1,i} = 1 / 2m, 1 / 2p$ for $y_i = 0,1$,where m and p represent
no. of negative and positive samples respectively.
for t=1,…..,T **do**
(1) Initialize the weights: $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$ for each feature j, instruct a classifier $h_j()$
(2) **If** t ≤ T/2
Optimize weak classifiers $\{h_j()\}_{j=1}^{J}$ using PSO algorithm:
$$\{h_t, \in_t\} = \textbf{PSO}\left(\{h_j\}_{j=1}^{j}, \{x_n, y_n, w_n\}_{n=1}^{N}\right)$$
Evaluate the fitness function for each particle
Select the classifiers $h_t()$ matching to the particles at the best global position
end if
(3) Assess the classifier weights $\alpha_t = log\frac{1-\epsilon_t}{\epsilon_t}$
(4) Reform weights:$w_{t+1,i} = w_{t,i}\beta_t^{1-b_i}$ where b_i=1 if $h_t(x_i) = y_i$, b_i =--1 otherwise with $\beta_t = \in_t/(1-\in_t)$
end for
Output strong classifier:
$$H(x) = \begin{cases} 1, & if\ sign\ (\sum_{t=1}^{T} \alpha_t b_t(I) \geq \theta \sum_{t=1}^{T} \alpha_t)\ is\ positive \\ -1, & otherwise \end{cases}$$
With $\alpha_t = log(1/\beta_t)$
If sub-window verified all per-stage checks **then**
Select this sub-window as a face
end if
end for
Optimization Function **PSO ( )** for AdaBoost method
Input arguments $\left\{\{h_i()\}_{j=1}^{j}, \{x_n, y_n, w_n\}_{n=1}^{N}\right\}$
Define $C_s, C_g = 2, W_{min} = 0.2, w = w_{max} = 1.5$
Define random parameters: $r_s, r_g \in [0,1]$
Define state vector: $X_t^j \in R^D$ *and* $V_t^l \in R^D$ with random values.
for l=1, … , L
for i=1, …, I
(1) Set a classifier h $(X_t^l; x)$ to the training examples using weights Adw_n
(2) Evaluate $\in_t^j = \frac{\sum_{n=1}^{N} w_n\, X\, |h(X_i^l; x_n) - y^n|}{\sum_{n=1}^{N} W_n}$ (4.13)
(3) Updates the particles:
$$V_i(t+1) = V_i(t) + c_s r_s\left(Q_1(t) - X_i(t)\right) + c_g r_g(Q_2^g(t) - X_i(t))$$
$$X_i(t+1) = X_i(t) + V_i(t+1)$$
(4) Update the personal best point Q_i^s , if necessary
end for
(5) Update the global best point Q^g, if necessary
(6) Update momentum: w ←$w_{max} - \frac{l}{L}(w_{max} - w_{min})$
end for
return$\left\{h_{Q^g}(), \in_{h_{Q^g()}}\right\}$

In PSO, each particle could explore not only its own space but also the spaces of other particles. Consequently, many particles collectively strive to identify the best possible positions. However, this collaborative approach can lead to a decline in the diversity of selected features as we integrate a random feature selection approach. Specifically, we initially employ PSO to identify the most relevant features at an early stage. As the boosting phase unfolds, our proposed approach transitions to random feature selection to uncover additional discriminative features, thus expanding the pool of candidate features. This adjustment strikes a balance between efficient feature selection and the preservation of feature diversity, enabling us to discover a wider range of optimal features during the boosting process.

This diagram illustrates the integration of PSO with a Viola-Jones face detection algorithm. On the left, the PSO process begins with the initialization of a swarm of particles, each representing a potential solution. These particles have their fitness evaluated based on an objective function. During the iterations, each particle updates its personal best position (pBest) and the swarm identifies the global best position (gBest). Particle velocities are adjusted based on these best positions, and their positions are updated accordingly. This iterative process continues until a termination criterion is met, yielding the optimal solution (gBest). On the right, the face detection process starts with training images categorized as faces and non-faces. Haar-like features are extracted from these images, and integral images are used to accelerate feature calculation. Using the AdaBoost algorithm, relevant features are selected, and thresholds are optimized to create a strong classifier. These features are organized into a cascade classifier, which efficiently detects faces by sequentially eliminating non-face regions. PSO plays a crucial role in this system by optimizing the thresholds and feature selection during training, ensuring the cascade classifier is accurate and efficient for face detection.

4.6 Results and Analysis

This section analyzes the performance of the conventional Viola-Jones algorithm and compares it with an improved version that incorporates PSO optimization. In the conventional Viola-Jones algorithm, AdaBoost employs an exhaustive search to build a weak classifier, while in the proposed approach, AdaBoost utilizes an optimized search to select the

best features and threshold values. Furthermore, the proposed approach significantly reduces the false positive rate.

4.6.1 Dataset collection

Face images were collected from the Yearbook Dataset of frontal-facing American high-school seniors [159], while non-face images were obtained from the Stanford Background Dataset [160] and ImageNet [161]. These images are used for both training and testing purposes. These databases contain 4,999 different face images and 6,960 non-face images, all with a pixel resolution of 25x25. The positive and negative images are randomly divided into two folders. The training folder comprises 1,200 positive and 1,000 negative grayscale images (See Fig. 4.12). The test set consists of 750 positive and 658 negative images.

In addition, this experiment was validated using the Wider Face test benchmark dataset, which includes various real-time facial detection challenge images [162]. The dataset comprises 32,203 photographs and labels 393,703 faces, covering a wide range of scales, poses, and occlusions.

Fig. 4. 12. Training and testing sample images (a) Face images (Positive Images) (b) non-face images (Negative Images)

4.6.2 Parameter Setting and Threshold Selection

To analyse the proposed approach reliability, accuracy, and time spent on each sample parameter are used. The proposed approach consists of 200 particles and could run for up to 1,000 iterations for constructing a weak classifier. However, it terminates if there is improvements are observed in the feature selection process within the global solution search space. Initially, the population is randomly defined, with the feature selection parameters (x,

y, w, h) in the range of [0, 250] and the feature type in the range of [0, 4]. The social value parameters are set to c1 and c2, both ranging from 100 to -100. Random values are independently sampled from the range [0, 1], and Q1 and Q2 are both set to 3.05. These experiments were conducted with 1,000 iterations, and the results, including the best, worst, and average, are reported in Table 4.4. These experiments were run on Google Colab with GPU K80.

Table 4.4 Training Error of training Dataset

S (No. of particles)	T (Iterations)	Training Error
5	50	0.9034
5	100	0.8912
10	100	0.8713
10	50	0.9613
20	**100**	**0.8531**
20	50	0.1034
30	100	0.8989

To construct the weak classifier for selecting the best threshold value, we examined the particle's size and maximum iteration using the ImageNet dataset. The results are displayed in Table 4.4 showing the performance of various particle sizes and their iterations. The selected optimal threshold value is then applied in the feature selection section of AdaBoost to optimize the features and computation time. Besides, the best PSO parameters were chosen according to Table 4.4 (Particle size 20 and iteration 100).

4.6.3 Feature Selection Optimization using PSO

The proposed approach is analyzed in terms of classifier accuracy and execution time. The experiments were repeated ten times for each algorithm, and the best, average, and worst outcomes are presented in Table 4.5. Additionally, the analysis indicates the number of features required for the face detection process on complex face images. The performance of face detection is influenced by both the population size and the number of PSO iterations. The results demonstrate the effectiveness of PSO for optimal feature selection in this problem when compared to the conventional Viola-Jones algorithm.

Table 4.5. Number of features needed for detection

	Best	**Average**	**Worst**
Viola-Jones	340	340	340
Proposed	120	134	150

This experiment reveals that the proposed method utilizes as few features as possible compared to the conventional algorithm. Table 4.5 shows the number of features generated for a strong classifier during the training process. Viola-Jones with PSO required only 120 features in the best case, 134 in the average case, and 150 features in the worst case for building the weak classifiers, whereas the conventional Viola-Jones algorithm required 340 features in the best case. Therefore, the proposed method constructs superior classifiers using only 120 features in the best case, which is significantly fewer than the conventional Viola-Jones method.

Table 4.6 provides a comprehensive comparison of the performance metrics between the traditional Viola-Jones (VJ) algorithm and the proposed method when applied to the ImageNet test dataset. The evaluation highlights significant improvements in classification accuracy and false positive rates (FPR), underscoring the superiority of the proposed method. The proposed approach demonstrates a remarkable classification accuracy of 98.73%, significantly outperforming the conventional Viola-Jones algorithm, which achieved an accuracy of 96.63%. Similarly, the proposed method exhibits a much lower false positive rate, with an FPR of 1.27%, compared to the 3.37% FPR recorded by the conventional algorithm. These metrics clearly show that the proposed approach is both more precise and less prone to false detections.

Table 4.6. Face Detection performance

Approach	**TPR (Average)**	**FPR (Average)**	**Time**
Viola-Jones	96.63 %	03%	52.5s
Proposed	98.73 %	01%	30.6s

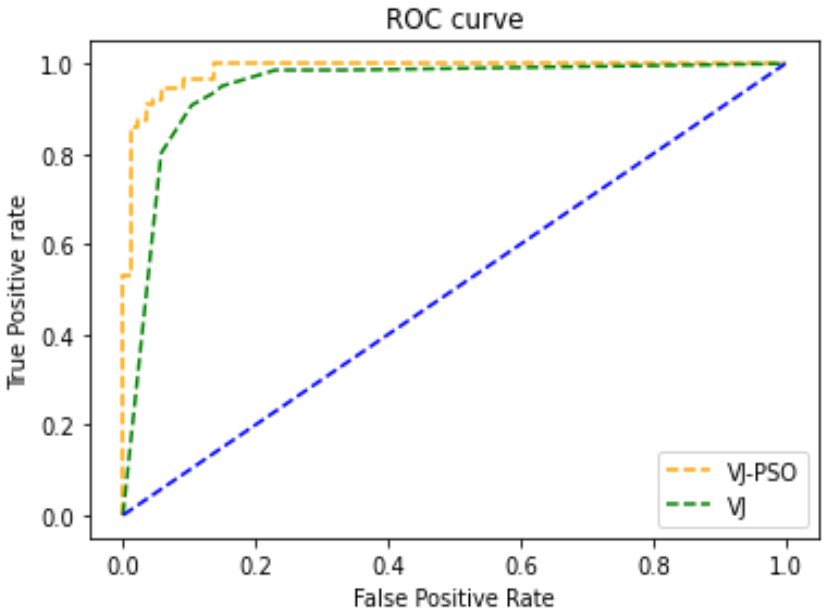
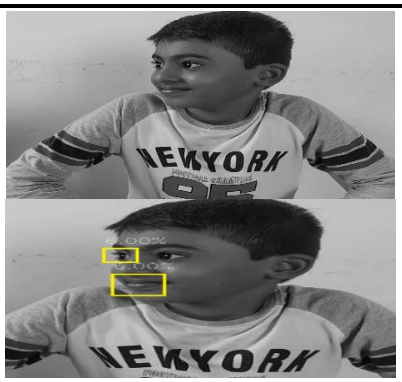
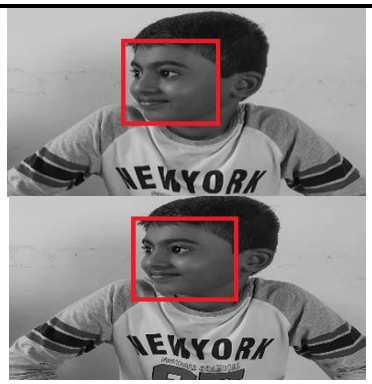
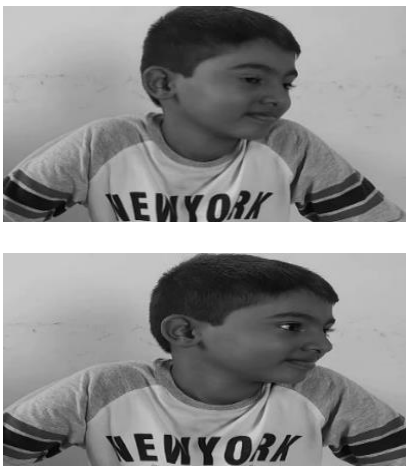
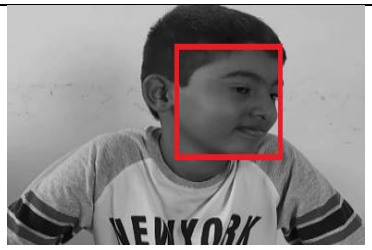
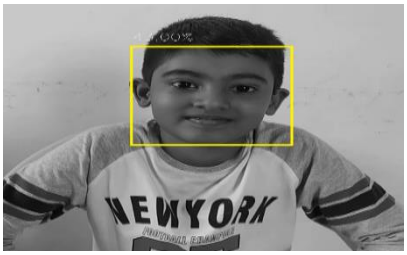
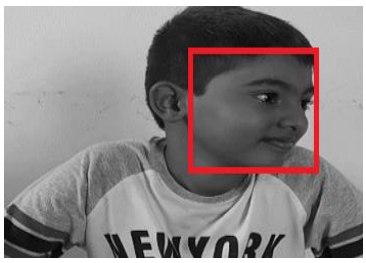
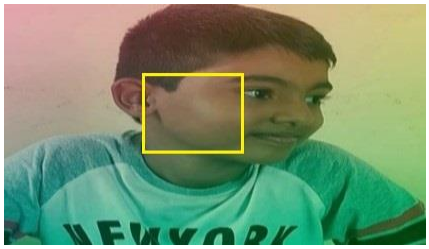


Figure 4.13. ROC curves of Conventional Viola-Jones and Viola-Jones with PSO

The performance of the proposed approach and the conventional Viola-Jones algorithm is depicted using the ROC curve (See Figure 4.13). The performance of the proposed method is represented in orange color, whereas the conventional Viola-Jones algorithm's performance is shown in green. According to the ROC curve, the proposed approach achieved a 98.73% accuracy on the face and non-face image dataset, whereas the conventional algorithm achieved 96.63%. After a successful testing process, the proposed approach was converted as face detection model also saved as. an XML file, allowing it to detect faces in various challenging contexts.

Finally, a comparison of the performance of the conventional Viola-Jones face detection algorithm and the proposed technique is presented in Table 4.6. The Viola-Jones algorithm with PSO performed effectively in various face detection complex real-time face images, including scale variation, illumination, pose variation, and occlusion, compared to the conventional method. Table 4.7 shows the results of the proposed approach detection performance and conventional approach.

Table 4.7 Face Detection Performance of Conventional and Proposed Approach

Challenges in Face Detection	Conventional Viola-Jones	Viola-Jones with PSO
Right Side Face		
Left Side Face		
Front Face		
Illumination		

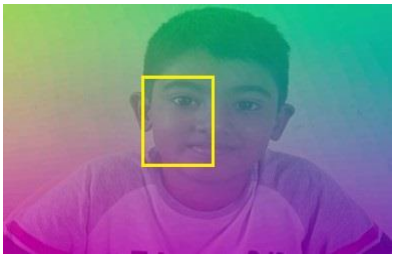
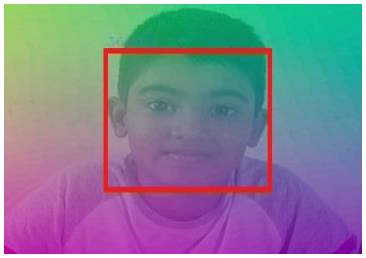
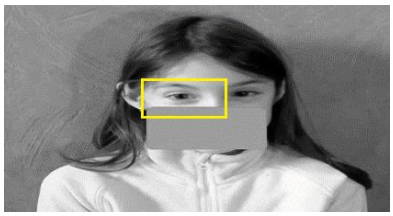
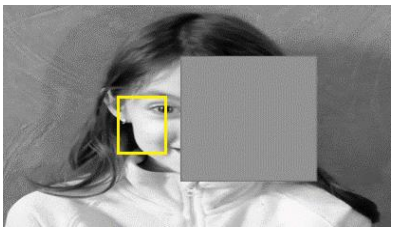
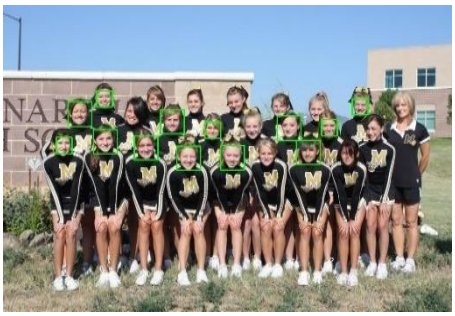
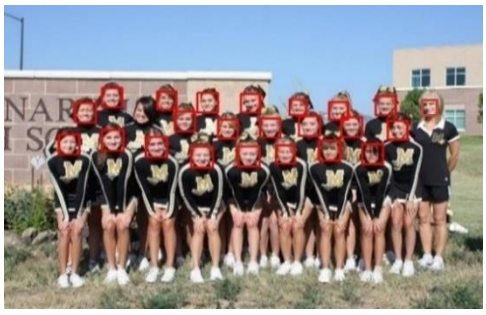
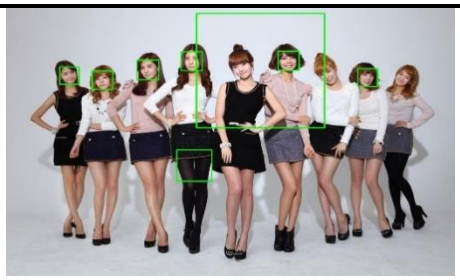
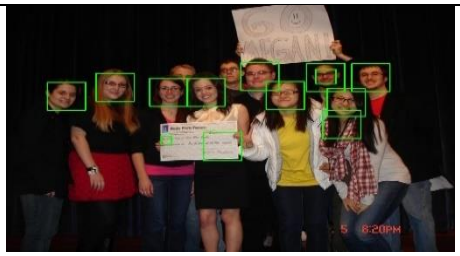
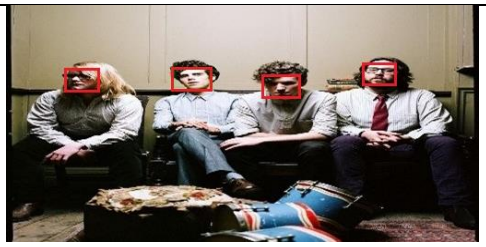
Challenges in Face Detection	Conventional Viola-Jones	Viola-Jones with PSO
		
Partial Occlusion		
		

Table 4.8. Performance comparison of proposed and conventional Viola-Jones algorithm on the Wider benchmark dataset samples to validate the performance

Challenges in Face Detection	Conventional Viola-Jones	Viola-Jones with PSO
Scale Variation		

Challenges in Face Detection	Conventional Viola-Jones	Viola-Jones with PSO
		
Illumination		
		
Pose variation		
		
Occlusion		

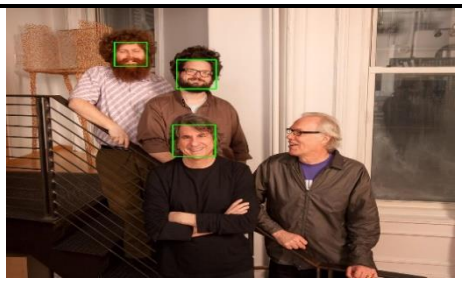
Challenges in Face Detection	Conventional Viola-Jones	Viola-Jones with PSO
		

Table 4.8 presents the performance analysis of the proposed algorithm and the conventional Viola-Jones algorithm on the Wider Face dataset, which contains highly complex facial images. It clearly shows that the proposed Viola-Jones with PSO algorithm significantly performs on various complex images, such as different scale face images, low illumination, pose variation above 40 to 45 degrees, and occlusion.

Table 4.9. Performance comparison of the proposed method with the existing approach

Face Detection Approach	Accuracy
Viola–Jones, Geometric Distribution [162]	95%
Viola–Jones, Condensation Algorithm (CA), NN, SVM [163]	95%
Skin Color Algorithm, Circular Hough Transform [164]	80%
Kalman Filter, Principal Component Analysis (PCA), Local-Binary-Pattern (LBP), SVM [165]	95%
Proposed (Viola-Jone with PSO)	**98.73%**

Table 4.9. presents the prediction accuracy results compared to previous state-of-the-art techniques, showing a 2.73% improvement with the proposed approaches. Furthermore, the computational complexity of the optimized Viola-Jones algorithm is determined by two parameters: S, which represents the number of particles, and T, which represents the number of iterations. In contrast, the computational complexity of the AdaBoost algorithm is determined by the parameter N, denoting the number of samples. The time complexity of the proposed algorithm at each stage of the boosting technique is O(SxT), whereas the time complexity in the base model is O(N^2). The basic AdaBoost technique trains a weak

classifier in polynomial time, while the improved PSO-based Viola-Jones algorithm's time complexity scales linearly with S and T.

4.7 Chapter Summary

This chapter aimed to enhance the Viola-Jones algorithm by optimizing the feature selection process and global threshold determination in AdaBoost using PSO. The use of PSO significantly reduces false-positive rates and computational time. The proposed approach constructs a more efficient weak classifier for face detection in complex face images. Instead of performing an exhaustive feature search, PSO optimizes the selection process, leading to better performance. The proposed method was validated on the Wider Face Detection Benchmark and demonstrated superior results compared to the conventional algorithm. It achieved an impressive average true positive rate of 98.73%, with only a 1.27% false positive rate. Additionally, the proposed approach significantly reduced face detection time on the test samples.