Final Project Report: Face Detection Using Haar Cascades and LBP

ENCM 509: Fundamentals of Biometric Systems Design
Team members: Tania Rizwan, Muhammad Haris Kashif
Date Submitted: April 14, 2025

Introduction

Face detection is a computer vision technology that recognizes human faces in digital images or videos. It uses algorithms to clearly distinguish faces from surrounding objects and backgrounds and establishes boundaries identifying the location and position of the recognized face [1]. Machine learning and artificial intelligence (AI) algorithms are used to differentiate faces from other elements by analyzing features such as the position of the nose and the pupillary distance [2]. Applications of face detection include enhancing border control at immigration checkpoints, autofocus in digital cameras, and identifying intrusion to sensitive locations through integration into security and body cameras used by security personnel [2]. Face detection aids in criminal identification by tracking individuals in public spaces, is non-invasive and doesn't involve any physical interaction, enhances a user's experience in consumer electronics (identifying faces for smartphone unlocking), and significantly improves the accuracy in facial recognition to verify the identity of detected individuals [2]. While it offers many benefits, significant data storage is required, while continuous data collection from the public raises privacy concerns, and improperly trained algorithms could introduce AI bias to discriminate against people of color [2].

In this project, we explored and compared two algorithms - the Viola-Jones Haar Cascades and the Local Binary Patterns (LBP) algorithm. The Haar Cascades algorithm is a machine learning based object detection algorithm, which uses Haar features to determine the probability of a certain point being in an object, along with a series of cascading classifiers and boosting algorithms to make the final decision [3]. Haar-Cascades use edge and contrast-based features for decision making and are known for their speed, efficiency, and simplicity [4][5]. The LBP algorithm is a very popular texture-based algorithm that compares the intensity of a central pixel to its neighbors to create a unique fingerprint for each face using histograms. It is known for its fast and lightning-robust efficiency, although it may be less sensitive to certain features [6][7][8].

Objectives

The main goal of this project was to evaluate the performance of two classical face detection algorithms, Haar Cascades and LBP, using the AT&T dataset of facial images. These algorithms were chosen because they represent two different approaches: Haar uses edge and contrast features, while LBP focuses on texture-based patterns.

We aimed to compare the effectiveness of each algorithm in detecting frontal faces by adjusting their internal parameters. For Haar, we tuned scaleFactor, minNeighbors, and minSize, while for LBP, we additionally explored the impact of maxSize. These parameters directly affect how the algorithms process image scales and filter detection results, which has a major influence on the accuracy and sensitivity of face detection.

To measure performance, we used three common evaluation metrics:

- Precision: the ratio of correct detections out of all predicted detections
- Recall: the ratio of correctly detected faces out of all actual faces
- F1 Score: a balanced metric that combines precision and recall

We also evaluated performance across three IoU (Intersection over Union) thresholds: 0.25, 0.5, and 0.75. These thresholds let us assess how accurate the bounding boxes were relative to the manually labeled ground truth.

Finally, we tested the algorithms on images of two different subjects from the dataset, allowing us to see how well each approach generalized across individuals. The goal was not only to find the best parameters for each method but also to understand how changes in those parameters influenced detection quality under different evaluation thresholds.

Methods

Dataset

The AT&T dataset was created by the Cambridge University Engineering department between 1992 to 1994 to serve as a large, open-source dataset to serve for face-detection tasks. It consists of ten different images taken per participant, for a total of 40 distinct subjects. All the images were taken against a dark and uniform background with the subjects in an upright and frontal position (with some side movement).

The format of the images is in .pgm and their size is 92x112 pixels each, with 256 grey levels per pixel. There are 40 different directories (one per subject), which have the names of the form 'sX', where X is the subject number (ranging between 1-40). In each subject directory, there are 10 images labelled 'Y.pgm', where Y is the image number (ranging between 1-10) [9]. For the purpose of this project, all 10 images for subject 1 and 2 were used for comparing the two algorithms.

Preprocessing

Preprocessing is applied to enhance and standardize the input images, while also defining bounding boxes as a 'ground truth' to facilitate accuracy evaluation during face detection. This stage involves a variety of steps: performing histogram equalization to improve contrast, normalizing the images to ensure consistency, and manually defining bounding boxes around the face of each subject to serve as a reference for subsequent evaluation of detection performance.

Histogram Equalization

Histogram equalization is a common image processing technique that improves the contrast of an image. It involves spreading the pixels across the histogram of an image to linearize the cumulative distribution function (CDF). While regular histogram equalization is effective, it also exaggerates any noise within the image. Therefore, adaptive equalization techniques are used to improve contrast locally while suppressing noise to provide a higher quality image overall. This technique was used in this project to mitigate any uneven lighting issues while highlighting facial features and edges [10].

A matrix is initialized to store each grayscale image from a particular subject as a column vector. This dataset matrix is then passed into the image_equalizer function which uses the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm using OpenCV to improve contrast in each image. First, an image (represented as a flattened column vector) is reshaped to a 2D-matrix and converted to an 8-bit unsigned integer format (where 256 bits represent 256 gray-levels in each pixel). The algorithm is then applied to the image with a clipLimit of 0.5 and a titleGridSize of 3x3. The clipLimit sets the threshold for contrast clipping to avoid

overamplifying any noise in the image, whereas the titleGridSize divides the image into a 3x3 grid so that equalization is applied locally and contrast is enhanced in small regions [10].

Histogram Normalization

Histogram normalization is a digital image processing technique used to enhance fine detail in an image. It redistributes pixel intensities so that they align more closely with a desired range or distribution, to further improve contrast and clarity. By adjusting the histogram, darker areas become more distinguishable from lighter areas to correct for lighting conditions and reveal subtle features that would otherwise be lost. It is also used to adjust pixel values to ensure that all images follow a similar distribution, while also ensuring numerical stability to speed processing [11] [12].

After equalization was performed, the updated dataset matrix is passed to the image_normalization method which first determines the mean and standard deviation of the entire dataset as global statistics, as well as the mean and standard deviation of each individual image. The image is then normalized using both sets of statistics to ensure it is standardized to match the overall dataset's intensity distribution, using the following formula [13]:

$$normalized\ image\ =\ (image\ -\ mean\ of\ image)\ *\ global\ std\ /\ std\ of\ image\ +\ global\ mean$$

Ground Truth Bounding Boxes

Bounding boxes are rectangles that define the region required to accurately detect an object. These are typically the outputs of face detection algorithms, however for the evaluation of the quality of an algorithm, bounding boxes must be manually defined to serve as the 'ground truth'. In this project, these were defined in the form [x1, y1, x2, y2] per image and were input into the visualize_ground_truth function so they could be overlaid onto each image, and manually adjusted as needed.

Results of Preprocessing

The series of figures demonstrate the functionality of the preprocessing pipeline. First, images of a subject are loaded (Figure 1), and the histogram of this original dataset is visualized (with each color representing a unique image, Figure 2). After histogram equalization is performed, the contrast of the images is noticeably improved (Figure 3), and the updated histogram represents a larger range in the y-axis to indicate a redistribution of pixel intensities, where more pixels are now concentrated in certain regions to highlight subtle details (Figure 4). Finally, normalization is applied to further enhance the images (Figure 5), and the histogram reflects how the pixel intensities are redistributed to share a more uniform mean and standard deviation, ensuring each image has a similar overall brightness and contrast range (Figure 6).

Figure 1. Original images (Subject 1)

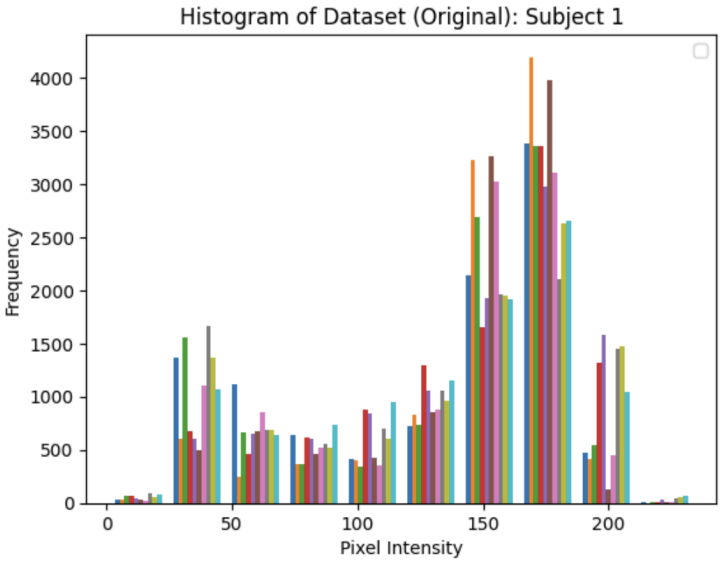


Figure 2. Histogram of original images

Figure 3. Equalized images (Subject 1)

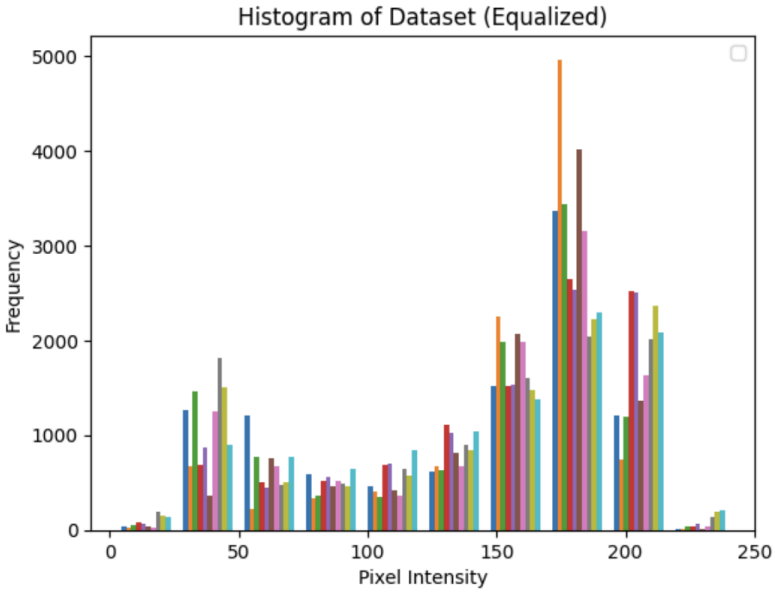


Figure 4. Histogram of equalized images

Figure 5. Normalized (and equalized) images (Subject 1)

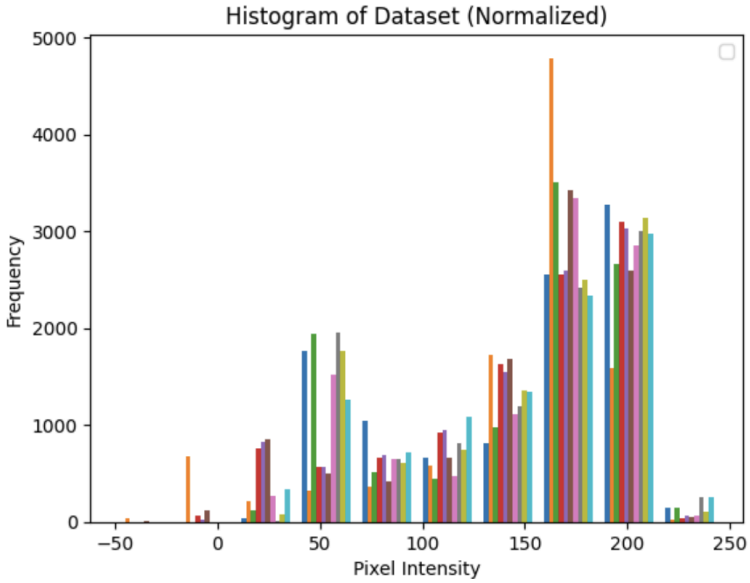


Figure 4. Histogram of normalized (and equalized) images

Algorithms

Haar Cascades

The Haar-Cascades algorithm uses a cascade of classifiers, with each classifier based on Haar-like features. Haar features are determined using a rectangular sliding window across an image, where the area within the window is divided into smaller rectangular areas. The Haar feature is then computed as the difference in the sum of pixel intensities between these divisions. These are used to find edges, lines and center-surrounded features within an image. Viola-Jones hypothesized that the presence of an object will distort pixel intensity variation between regions. For example, backgrounds are typically uniform, whereas objects will follow a significantly different pattern in their distribution. Therefore, this algorithm focuses on comparing the pixel intensities between neighboring rectangular regions to determine an object's presence [14].

A cascade is then used to combine multiple Haar features to build a classifier in a hierarchical series of stages, and is applied within each window. Instead of using every feature to analyze every part of an image, each stage quickly discards regions that don't show the desired characteristics to ensure that only promising areas are examined further. Since only a few pixels are typically relevant to an object, this approach effectively ignores the majority of the image. The cascade is trained using boosting algorithms (specifically AdaBoost), which combines many weak classifiers into a strong one to reduce computation and speed up detection [15][14].

The parameters used by the algorithm include the scaleFactor, which defines how much the image size is reduced by, minNeighbors which specifies the number of nearby detections required to retain a candidate, and the minSize which filters out small detections [15]. This project explored a variety of combinations of these parameters to determine the best combination. To evaluate the behavior of the algorithm, we tested it at various combinations - using scaleFactors of 1.01, 1.05, and 1.03 (to test against a variety of scales across the typical range of scaleFactors that is 1.0 to 1.3 [14]), minNeighbors of 0, 3, 5 and 10 to span across a large range of neighbors to determine their effect on detection, and minSize values of (5, 5), (10, 10), (30, 30), (40, 40) to evaluate how effective they were in recognizing faces from smaller, distinct features.

The dataset matrix was fed into the haar_cascade_detector method, which reshaped each individual image into a 2D matrix and converted it into an 8-bit unsigned integer. The frontal_haar_cascade classifier was initialized using OpenCV, and was applied to each image at various parameter combinations. This resulted in a list defining the bounding box, which contains the area recognized as a 'face' by the algorithm, as well as a value for the precision, recall and F1-score at each IoU threshold to determine the performance of each parameter set.

Local Binary Patterns (LBP)

The LBP cascade is a fast and efficient face detection algorithm that works by capturing local texture patterns within an image. It does this by comparing each pixel with its surrounding neighbors in a 3×3 grid. If a neighboring pixel's value is greater than or equal to the center pixel, it is assigned a 1. Otherwise, it is assigned a 0. This process generates a binary code for each region, which can then be converted into a decimal number that describes the local texture. These texture-based features make LBP particularly well-suited for detecting faces under consistent lighting and background conditions.

In this project, we used the lbpcascade_frontalface.xml classifier provided by OpenCV. This is a pre-trained model designed to detect frontal faces in grayscale images using a cascade of classifiers. The algorithm applies a sliding window that moves across the image at various scales and locations. At each window, LBP features are extracted and passed through multiple stages of the cascade. Each stage acts as a filter, quickly rejecting non-face regions and only allowing potential face regions to move forward. This design makes the algorithm computationally efficient and suitable for real-time detection.

To evaluate the LBP detector's behavior and tune its performance, we experimented with several key parameters. These were:

- **scaleFactor:** Determines how much the image size is reduced at each image scale. We tested values of 1.01, 1.03, 1.05, 1.1, and 1.2. Smaller values resulted in finer scale changes and allowed the detection of smaller faces, while larger values improved speed but risked skipping faces at in-between scales.

- **minNeighbors:** Specifies how many neighboring rectangles need to agree for a detection to be accepted. We tested 2 to 6, which allowed us to explore both lenient and strict filtering. Lower values increased detection sensitivity but also introduced false positives, while higher values made the detection stricter and more selective.

- **minSize and maxSize:** Define the allowable size range of the detection window. We tested combinations such as (20, 20) to (300, 300), (25, 25) to (160, 160), and (30, 30) to (150, 150). These parameters were useful for constraining the detection to match the approximate face sizes president in the AT&T dataset.

These parameters were combined into multiple configurations and tested systematically on two subjects. For each configuration, we recorded the detected bounding boxes and used them later in the evaluation step to compute precision, recall, and F1 score based on manually labeled ground truth. All images were preprocessed before detection using grayscale conversion, histogram equalization, and normalization to ensure consistent contrast and lighting.

By testing a wide range of parameter values, we were able to analyze how each parameter affected the behavior of the LBP cascade without biasing the results toward any single configuration. This helped us isolate the effects of individual parameters and better understand the overall behavior of the algorithm under different tuning conditions.

Evaluation Metrics

To evaluate how well the face detection algorithms performed, we used three common metrics: Precision, Recall, and the F1 Score. These metrics helped us assess how accurate and complete the detected bounding boxes were compared to the manually labeled ground truth for each image.

Before calculating these metrics, we used a comparison method called IoU. IoU measures how much the predicted face box overlaps with the actual face box. It does this by comparing the area where the two boxes overlap to the total area covered by both boxes. The result is a value between 0 and 1. A higher IoU value means the predicted box closely matches the ground truth.

To determine whether a detection was considered correct, we checked if its IoU was above a certain threshold. If it was, we counted it as a true positive. If a detection was made but didn't match any actual face well enough, it was considered a false positive. If a real face was present but not detected at all, that counted as a false negative.

Using the counts of true positives, false positives, and false negatives, we calculated the three evaluation metrics:

- **Precision:** Tells us how many of the faces detected by the algorithm were actually correct. A high precision means the algorithm made fewer false detections.

- **Recall:** measures how many of the actual faces in the image were detected. A high recall means the algorithm didn't miss many faces.

- **F1:** Score combines both precision and recall into a single number. It helps us understand how well the algorithm performs overall, especially when there's a trade-off between detecting more faces and avoiding false detections.

To get a more complete picture of how each algorithm performed, we tested them using three IoU thresholds: 0.25, 0.5, and 0.75. Lower thresholds are more forgiving and accept detections that are loosely aligned with the actual face, while higher thresholds require the predicted box to be much more precise. By using multiple thresholds, we were able to see how sensitive each algorithm was to box accuracy and how performance changed indeer stricter matching conditions.

This evaluation process allowed us to fairly compare different parameter settings and understand how tuning affected each algorithm's ability to detect faces accurately and consistently.

Results

Haar-Cascades

Table 1 provides an overview of the average F1 score determined for each parameter set. A variety of parameter sets were chosen to ensure a fair evaluation, and the average F1 score was computed over the F1 score determined at each IoU threshold. A breakdown is provided in the code file, while a summary is provided below.

Table 1. Summary of Haar-Cascades Results

scaleFactor	minNeighbors	minSize	Average F1 Score (S1)	Average F1 Score (S2)
1.01	0	(30, 30)	0.011	0.034
1.01	5	(30, 30)	0.667	0.857
1.01	10	(30, 30)	0.667	0.900
1.05	5	(10, 10)	0.625	0.877
1.05	10	(30, 30)	0.625	0.708
1.05	10	(5, 5)	0.625	0.708
1.05	3	(30, 30)	0.667	0.877
1.3	10	(30, 30)	0.308	0.000
1.3	3	(40, 40)	0.625	0.308

The parameters scaleFactor = 1.01, minNeighbors = 10, and minSize = (30, 30) provided the best performance between both subjects (based on the highest F1 scores and number of resulting bounding boxes). While parameter sets 2 and 7 resulted in the same F1 score as set 3, these were not determined to be the best performing, as set 2 detected two bounding boxes (or faces) in image 5, and set 7 uses a larger scaleFactor, which could miss subtle details on a larger dataset.
Overall, in Subject 10, 7 out of 10 of faces were identified, whereas all 10 faces were identified in Subject 2. This is due to the fact that the classifier used was specific to frontal face detection, and Subject 1 had some images with their face turned. In subject 1, this was reflected by the maximum F1 score being 0.677, whereas it was 0.900 in subject 2 (higher due to all images being front-facing).

The scale factor of 1.01 provided a small reduction between scales, ensuring that the image was very finely stamped. This allowed the cascade to capture subtle variations in the face size and position, ensuring the detection window closely matched the face boundaries. The number of neighbors was large enough to ensure false positives were removed (which would likely occur due to background noise or variability in the images), therefore, a parameter of 10 ensured more overlapping candidate windows were required to confirm an output. Finally, the minSize of (30, 30) ensured that extremely small regions that were unlikely to represent a face were being ignored. This also verified that the detected regions were large enough to discern an object that was likely a face.

The figures below demonstrate how the Haar-Cascade face detection performed on each image for both subjects, using the best parameter settings. Each image shows the detected face outlined with a bounding box, while the accompanying table summarizes the corresponding precision, recall, and F1-score at various IoU thresholds. The precision vs recall curves are shown to evaluate the overall performance as well.

Figure 7. Best parameter setting results for Subject 1

Table 2. Evaluation metric summary for Subject 1 at the best parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	0.700	0.824
0.5	1.000	0.700	0.824
0.75	0.429	0.300	0.353

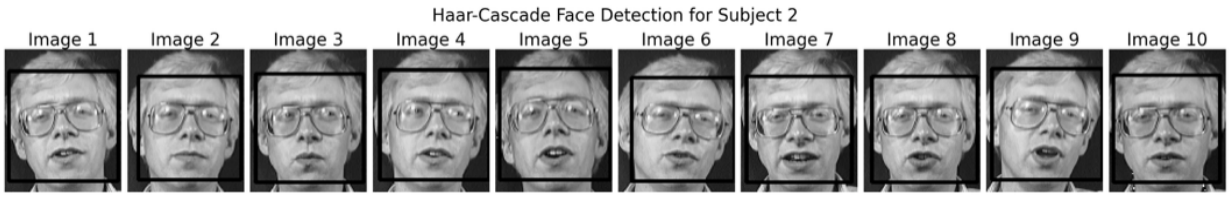
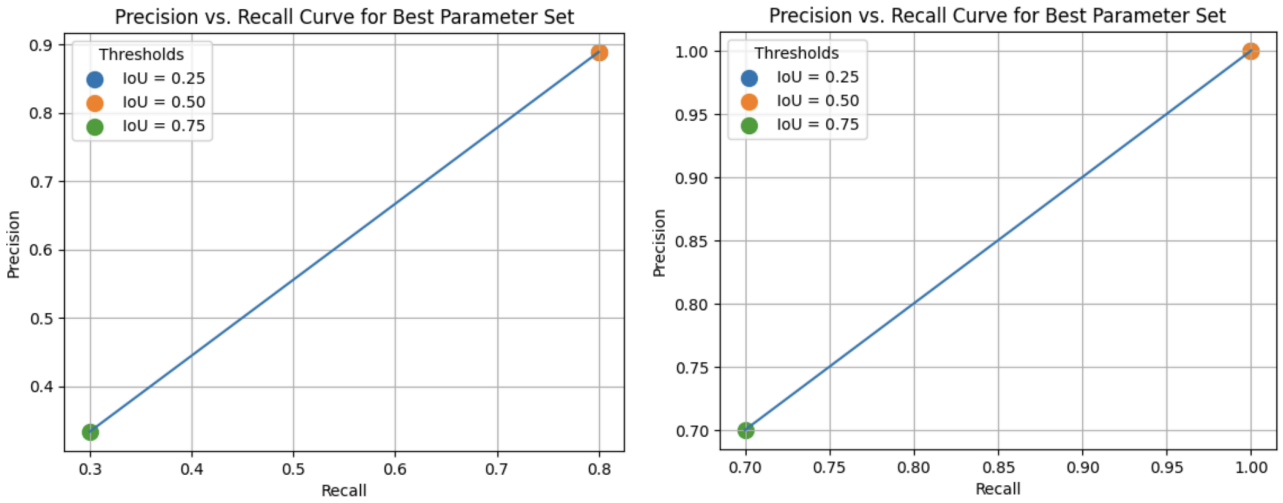


Figure 8. Best parameter setting results for Subject 2

Table 2. Evaluation metric summary for Subject 2 at the best parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	1.000	1.000
0.5	1.000	1.000	1.000
0.75	0.700	0.700	0.700

The precision and recall scores remain high (and are the same) for both subjects at IoU thresholds of 0.25 and 0.5 as a bounding box is still counted as correct even if it is roughly aligned with the ground truth. However, at a higher threshold of 0.75, the detection must overlap very precisely with the ground truth. Even minor misalignments, which were acceptable at lower thresholds, cause the detection to be counted as a false positive, which reduces both precision and recall.



Figures 9-10 (L-R). Precision vs Recall Curves for Subjects 1 and 2, respectively

For both subjects, each point on the graph corresponds to a different IoU threshold (0.25, 0.5, and 0.75). For Subject 1, the curve shows near-perfect detection at the lower thresholds but a noticeable dip at 0.75. For Subject 2, precision and recall remain high at all thresholds, suggesting that the resulting bounding boxes align more accurately with the ground truth across different levels of strictness.

One of the worst parameter combinations for both subjects is scaleFactor = 1.01, minNeighbors = 0, and minSize = (30, 30). This is due to the fact that minNeighbors being set to 0 does not require any overlapping detection windows, therefore, every candidate window is accepted as a face. Even though the scaleFactor finely scans the image, the total lack of neighbor checking produces excessive false positives and severely lowers precision. The (30, 30) minsize ensures

only moderately large windows are considered, but without a proper neighbor filter, the algorithm flags nearly every detected region as a "face," resulting in the worst F1 score overall. The figures below demonstrate the output of the algorithm for both subjects, with their F1 scores summarized.

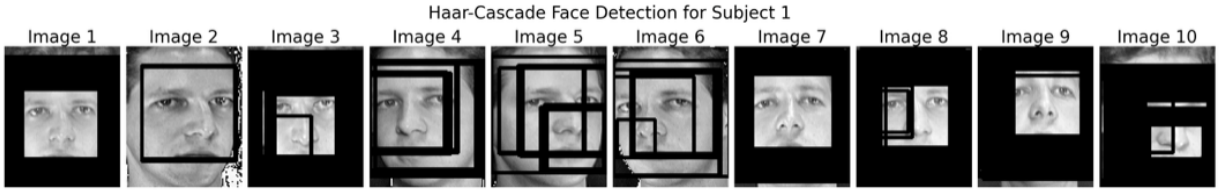


Figure 11. Worst parameter settings for Subject 1

Table 3. Evaluation metric summary for Subject 1 at the worst parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	0.006	1.000	0.012
0.5	0.006	1.000	0.012
0.75	0.004	0.700	0.009

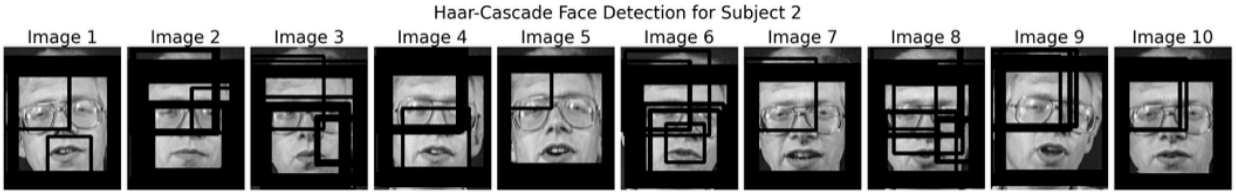


Figure 12. Worst parameter settings for Subject 2

Table 4. Evaluation metric summary for Subject 2 at the worst parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	0.017	1.000	0.034
0.5	0.017	1.000	0.034
0.75	0.017	1.000	0.034

Since the previous configuration was clearly problematic, the next-poorest performer is the configuration involving a scaleFactor = 1.3, minNeighbors = 10, and minSize = (30, 30). This is due to the higher scaleFactor which results in fewer scales in the image, causing the algorithm to miss faces that are not precisely at the expected size. Additionally, the higher neighbor requirement demands a large number of overlapping detections, which is difficult to achieve

when the scale granularity is so coarse. As a result, many genuine faces fail to meet the strict overlap requirement, resulting in many zero detections despite the faces being clear and front-facing in the input images.

Figure 13. Second worst parameter settings for Subject 1

Table 5. Evaluation metric summary for Subject 1 at the second worst parameter settings

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	0.300	0.462
0.5	1.000	0.300	0.462
0.75	0.000	0.000	0.000

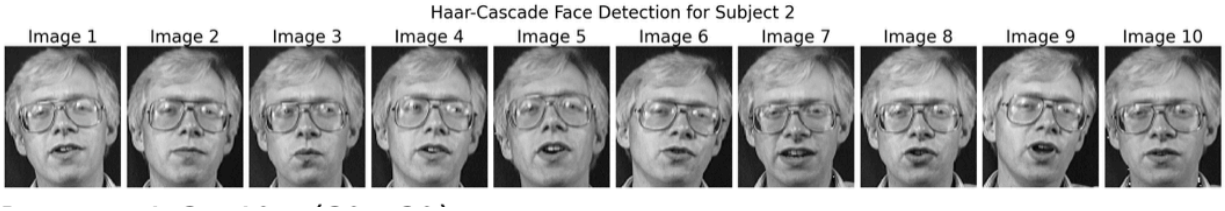


Figure 14. Second worst parameter settings for Subject 2

Table 6. Evaluation metric summary for Subject 2 at the second worst parameter settings

IoU Threshold	Precision	Recall	F1-Score
0.25	0.000	0.000	0.000
0.5	0.000	0.000	0.000
0.75	0.000	0.000	0.000

Local Binary Patterns (LBP)

Table 7 lists the average F1 scores for each LBP parameter configuration tested. Multiple combinations were evaluated to understand the impact of parameter tuning on detection quality. The average F1 score was determined by averaging results at three different IoU thresholds (0.25, 0.5, 0.75). These values helped us identify which parameter sets performed best overall.

Table 7. Summary of LBP Results

scaleFactor	minNeighbors	minSize	maxSize	Avg F1 (S1)	Avg F1 (S2)
1.03	3	(30, 30)	(150, 150)	0.630	0.967
1.01	2	(25, 25)	(160, 160)	0.603	0.952
1.03	3	(20, 20)	(300, 300)	0.630	0.967
1.01	2	(20, 20)	(300, 300)	0.603	0.952
1.1	5	(30, 30)	(150, 150)	0.476	0.278
1.2	4	(20, 20)	(200, 200)	0.476	0.182
1.1	5	(20, 20)	(300, 300)	0.476	0.278
1.2	4	(20, 20)	(300, 300)	0.476	0.182
1.05	6	(40, 40)	(180, 180)	0.625	0.571

Among all the sets tested, the combination using a scaleFactor of 1.03, minNeighbors = 3, minSize = (30, 30), and maxSize = (150, 150) produced the strongest results across both subjects. While another parameter set gave the same numerical F1 for Subject 2, this configuration was preferred due to producing tighter and more consistent bounding boxes. For Subject 1, the method correctly identified 7 out of 10 faces, while for Subject 2, it successfully detected all 10 faces. The improvement on Subject 2 is likely because all images were forward-facing, whereas Subject 1 had some variation in pose and lighting.

The detection window scaled gradually due to the low scaleFactor, allowing the algorithm to catch faces of varying sizes. The chosen minNeighbors value of 3 balanced filtering and sensitivity. The minSize and maxSize range provided a practical window size that matched the dimensions of faces in the dataset, avoiding false detections outside the expected region.

Sample outputs for the best parameter set are shown below. Each image demonstrates the predicted bounding box, and each table outlines the corresponding precision, recall, and F1 score

at each IoU threshold. The accompanying graphs visualize how performance changed depending on the strictness of the IoU match.

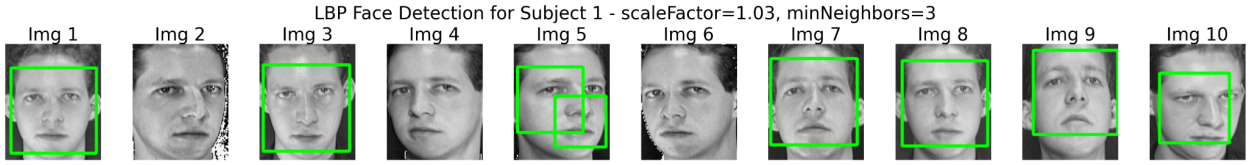


Figure 15. Best parameter setting results for Subject 1

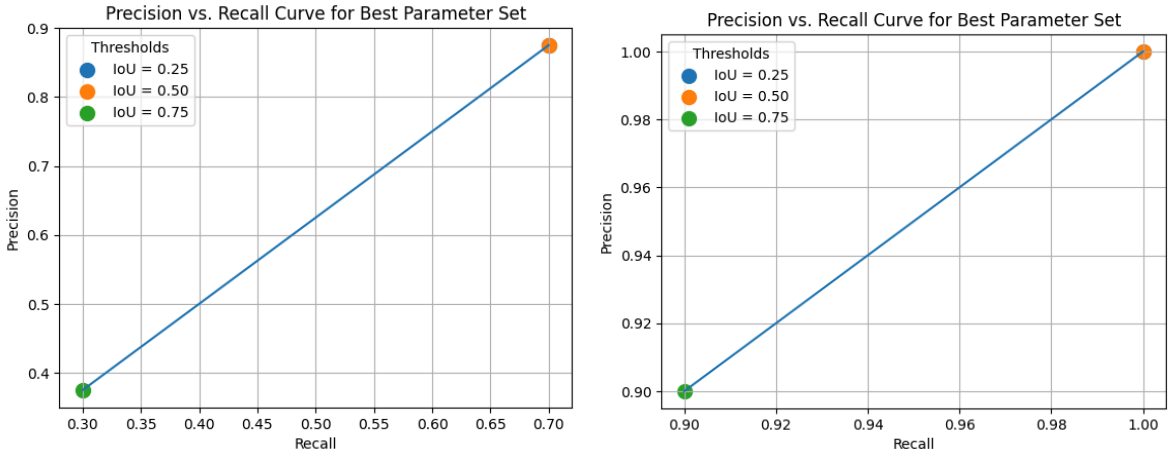
Table 8. Evaluation metric summary for Subject 1 at the best parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	0.875	0.700	0.778
0.5	0.875	0.700	0.778
0.75	0.375	0.300	0.333

Figure 16. Best parameter setting results for Subject 2

Table 9. Evaluation metric summary for Subject 2 at the best parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	1.000	1.000
0.5	1.000	1.000	1.000
0.75	0.900	0.900	0.900



Figures 17-18 (L-R). Precision vs Recall Curves for Subjects 1 and 2, respectively

Each plot shows the precision and recall at the three IoU levels. Subject 1 showed a steep drop in performance at the 0.75 threshold, indicating that while faces were detected, the bounding boxes were not tightly aligned with the ground truth. Subject 2 maintained high values at all thresholds, showing the algorithm's ability to detect and localize frontal faces with high accuracy.

The weakest-performing configuration used a scaleFactor of 1.2, minNeighbors = 4, and a window size ranging from (20, 20) to (300, 300). These settings likely reduced accuracy because the large-scale step skipped important image details, and the broad size range allowed misdetections. In this case, many bounding boxes were drawn inaccurately or failed to appear, especially on Subject 2.

Figure 19. Worst parameter setting results for Subject 1

Table 10. Evaluation metric summary for Subject 1 at the worst parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	0.400	0.571
0.5	1.000	0.400	0.571
0.75	0.500	0.200	0.286

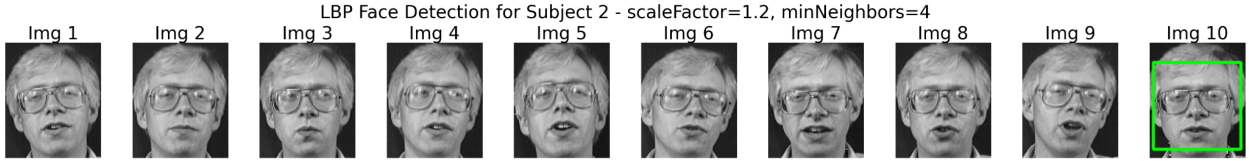


Figure 20. Worst parameter setting results for Subject 2

Table 11. Evaluation metric summary for Subject 2 at the worst parameter setting

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	1.000	0.182
0.5	1.000	1.000	0.182
0.75	1.000	1.000	0.182

Figure 21. Second worst parameter setting results for Subject 1

Another poor configuration included scaleFactor = 1.2, minNeighbors = 4, and a window size ranging from (20, 20) to (200, 200). These values made the model too strict on face size and neighbor agreement, which may have limited detections. Subject 1 experienced the same results as with the previous worst set, and Subject 2's performance remained low across all thresholds, again showing how scale and window range can reduce detection success.

Table 12. Evaluation metric summary for Subject 1 at the second worst parameter settings

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	0.400	0.571
0.5	1.000	0.400	0.571
0.75	0.500	0.200	0.286

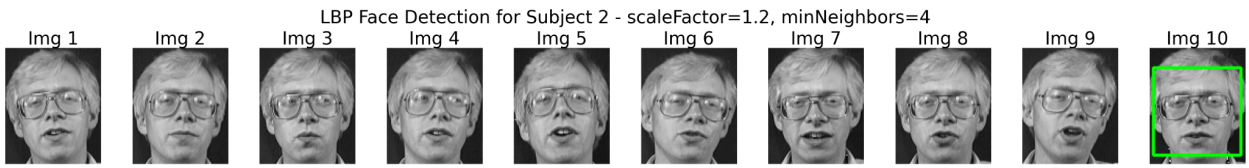


Figure 22. Second worst parameter setting results for Subject 2

Table 13. Evaluation metric summary for Subject 2 at the second worst parameter settings

IoU Threshold	Precision	Recall	F1-Score
0.25	1.000	1.000	0.182
0.5	1.000	1.000	0.182
0.75	1.000	1.000	0.182

Discussion

While both algorithms achieved strong performance overall, the LBP-based detector generally performed better as compared to the Haar-Cascades approach. LBP detects local texture patterns, which makes it more robust and better at generalizing across variations in lighting and angle changes; it was also faster and required fewer neighboring detections to confirm a face. In contrast, Haar-Cascades rely on edge and contrast features, rendering them more sensitive to illumination variations and requiring careful parameter tuning. For example, the worst-case performance metrics for LBP were consistently higher for both Subject 1 and Subject 2 compared to the Haar-Cascades approach, indicating that LBP is more robust even under challenging conditions. In the best-case scenarios, LBP achieved an average F1 score of 0.97 for Subject 2, while Haar-Cascades peaked at around 0.90, although Haar-Cascades slightly outperformed LBP-based detection for Subject 1, achieving an average F1 score of 0.677 compared to the latter that achieved a score of 0.625. These results suggest that LBP's focus on local texture features and its inherent resilience to variations in lighting contributes to more reliable face detection, whereas Haar-Cascades is more sensitive to parameter tuning.

In terms of the strengths and limitations observed, the Haar-Cascades method detects frontal faces reliably when its parameters are tuned appropriately, and even yields high F1 in some faces (for instance, a scaleFactor = 1.01, minNeighbors = 10 and minSize = (30, 30) yielded a score of 0.900 for Subject 2). However, the results showed a substantial dip when the minNeighbors parameter was set to 0 (average F1 = 0.11 for subject 1), as well as when the scaleFactor was set to 1.3 (average F1 = 0.00 for subject 2), indicating that the algorithm is highly sensitive to parameter choices. Moreover, the LBP algorithm demonstrated its robustness to lighting changes, as LBP descriptions capture texture in local regions of an image which makes them less sensitive to illumination. This contributed to the overall better performance of the LBP algorithm, even at its worst parameter settings (for example, yielding an average F1 score of 0.476 for subject 1, and never reaching near-zero F1 scores as compared to Haar-Cascades). At its best performance, the LBP algorithm achieved an average F1 score of 0.97, which

outperformed the best result achieved by the Haar-Cascades approach. However, the choice of parameters can still result in large swings in detection accuracy, and although it is known for its speed, the LBP algorithm is less historically prevalent as compared to the former approach.

Lower IoU thresholds (for example, 0.25) provided more lenient criteria and resulted in higher recall for all parameter configurations across subjects and algorithms since even partially overlapping detects count as correct. This also resulted in a generally higher precision. Moreover, higher IoU thresholds (specifically 0.75) provide a stricter criteria which requires near-perfect overlap between bounding boxes, which resulted in lower recall and precision as fewer detections met the stricter standard, as well as the slight misalignment of detections significantly impacts the results [16].

Parameter tuning is key to achieving a balanced performance in face detection. For instance, when tuning the Haar-Cascades algorithm, setting the minNeighbors parameter to 0 eliminated any requirement for overlapping detection windows and resulted in very high recall (as almost all faces were detected), but very low precision (due to a flood of false positives), which lead to a very low F1 score [17]. This example shows that parameter tuning is critical as it ensures that the algorithm neither misses too many faces nor includes too many spurious detections. Finer adjustments (such as choosing an appropriate minNeighbors value) can significantly improve results by creating a balance to improve the reliability and accuracy of the system.

Conclusion

This project set out to compare the performance of two widely used face detection algorithms, Haar Cascades and LBP, by analyzing their accuracy across different parameter settings and evaluation thresholds. After applying both techniques to subjects from the AT&T dataset and testing a variety of configurations, we were able to measure how changes in parameters influenced detection quality, particularly in terms of bounding box alignment.

Through our experiments, it was found that the LBP algorithm provided the most consistent and reliable performance overall. In particular, it achieved the highest average F1 score of 0.97 for Subject 2, outperforming Haar's best result of 0.90. The LBP method also showed stronger robustness across multiple parameter configurations, with fewer instances of extremely low F1 scores, which was not the case for Haar Cascades. Even in less optimal scenarios, LBP continued to detect faces at a reasonable accuracy level, whereas Haar's performance dropped sharply under certain combinations like low minNeighbors or high scaleFactor.

A key takeaway from this study is the importance of parameter tuning. Both algorithms are highly dependent on factors like scale factor, neighbor agreement, and window size limits. Small changes in these parameters can dramatically alter performance, especially at higher IoU thresholds. Our analysis confirmed that lower IoU thresholds are more forgiving, while higher

ones demand more precise bounding boxes and, therefore, expose the weaknesses of poorly chosen configurations.

To improve this work further, several enhancements could be made. First, testing on a larger number of subjects from the dataset would help confirm whether the trends observed here hold across more varied faces. Additionally, adding more parameter combinations could help uncover better settings. Exploring different lighting conditions, image orientations, and background complexities would also provide a more rigorous evaluation of each method's robustness. Lastly, integrating more modern face detection techniques as a baseline for comparison could help validate the effectiveness of classical methods like Haar and LBP.

In summary, while both algorithms performed reasonably well under the right conditions, LBP demonstrated stronger generalization, better resilience to parameter changes, and higher accuracy overall, making it a more dependable choice for standard frontal face detection tasks in controlled environments.

References

[1] R. Sheldon, "What is Face Detection and How Does It Work?," SearchEnterpriseAI, Oct. 2024. https://www.techtarget.com/searchenterpriseai/definition/face-detection

[2] Nupura Ughade, "What is Face Detection and How Does it Work?," hyperverge.co, Nov. 15, 2022. https://hyperverge.co/blog/face-detection/

[3] A. Mittal, "Haar cascades, explained - analytics vidhya - medium," *Medium*, Jan. 18, 2024. https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

[4] A. Rosebrock, "OpenCV haar cascades - PyImageSearch," *PyImageSearch*, Apr. 17, 2021. https://pyimagesearch.com/2021/04/12/opencv-haar-cascades/#:~:text=What%20are%20Haar%20cascades

[5] A. Jaiswal, "Guide to Haar Cascade Algorithm with Object Detection Example," *Analytics Vidhya*,Jan.08,2025.https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/#:~:text=Haar%20cascade%20is%20an%20algorithm,%2C%20buildings%2C%20fruits%2C%20etc

[6] A. B. Shetty, N. Bhoomika, N. Deeksha, J. Rebeiro, and N. Ramyashree, "Facial recognition using Haar cascade and LBP classifiers," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 330–335, Aug. 2021, doi: 10.1016/j.gltp.2021.08.044.

[7] A. Ihalapathirana, "Understanding the local Binary Pattern (LBP): a powerful method for texture analysis in computer vision," *Medium*, Aug. 25, 2023. [Online]. Available: https://aihalapathirana.medium.com/understanding-the-local-binary-pattern-lbp-a-powerful-method-for-texture-analysis-in-computer-4fb55b3ed8b8#:~:text=LBP%20works%20by%20comparing%20the,the%20central%20pixel%20(threshold)

[8] F. I. Abdurrahman, M. Nasrun, C. Setianingsih, and School of Electrical Engineering Telkom University Bandung Indonesia, "Face Recognition Using Local Binary Pattern (LBP) and Local Enhancement (LE) Methods At Night Period," *Atlantis Highlights in Engineering*.

[9] "AT&T Database of Faces," Kaggle, Dec. 17, 2019.
https://www.kaggle.com/dataset/kasikrit/att-database-of-faces

[10] A. Rosebrock, "OpenCV Histogram Equalization and Adaptive Histogram Equalization (CLAHE) - PyImageSearch," *PyImageSearch*, Apr. 17, 2021.
https://pyimagesearch.com/2021/02/01/opencv-histogram-equalization-and-adaptive-histogram-equalization-clahe/

[11] "Histogram normalization," *Robot Academy*.
https://robotacademy.net.au/lesson/histogram-normalization/

[12] E. Voronianskii, “Digital Image Processing | Histogram Calculation, Equalization and Normalization,” *Medium*, Dec. 10, 2023. [Online]. Available: https://vrnsky.medium.com/digital-image-processing-histogram-calculation-equalization-and-normalization-ad09b0bba5b1

[13] Dr. S. Yanushkevich, *Fundamentals of Biometric Systems Design*.

[14] J. Brownlee, "Using Haar Cascade for Object Detection," *Machine Learning Mastery*, May 20, 2021. [Online]. Available: https://machinelearningmastery.com/using-haar-cascade-for-object-detection/

[15] A. Mittal, “Haar cascades, explained - analytics vidhya - medium,” *Medium*, Jan. 18, 2024. [Online]. Available: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d

[16] K. Lai and S. Yanushkevich, *Laboratory Project #6: Face Detection*, ENCM 509 – Fundamentals of Biometric Systems Design, Dept. of Electrical and Software Engineering, Schulich School of Engineering, University of Calgary, Jan. 2025.

[17] “Classification: Accuracy, recall, precision, and related metrics,” *Google for Developers*. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

Appendix

Code Running Instructions

Refer to the instructions in the README.md file provided in the GitHub repository that stores the project code. It can be found here:

https://github.com/TaniaRizwan/FaceDetectionRecognition