Albert Kenneth Okine
Shibam Mukhopadhyay
Jerod Weinman
Fall 2024

Rationale

Our goal is to implement a facial recognition algorithm that not only detects faces but also draws bounding boxes around them and assigns confidence scores to each detection. Initially, we began exploring OpenCV and its pretrained facial detection algorithms. OpenCV uses both face and eye detection classifiers to determine the presence of a face. We aim to build on this idea by implementing Haar Cascade filters, specifically focusing on optimizing the Haar feature coefficients for facial detection.
The Haar Cascade classifier works by combining a set of Haar features—simple rectangular features used for object detection—into a weighted sum, where each Haar feature is associated with a coefficient. The general formula for this is:

$$c_1A_1 + c_2A_2 + \cdots + c_nA_n$$

Where A_i represents the Haar feature at a given location, and c_i represents its corresponding coefficient. Our task will be to optimize the coefficients for detecting faces, extract features from the detected faces, and then compare them (for instance, using Euclidean distance) to match faces.

Background and Planning

To understand the scope of our task, we will conduct extensive research into facial detection and recognition, as well as the use of Haar Cascades for these tasks. The cornerstone of our understanding will come from the paper *"Rapid Object Detection using a Boosted Cascade of Simple Features"* by Viola and Jones (2001), which introduced the Haar Cascade method for face detection. This paper will provide the foundation for understanding how Haar features are used in face detection and how the classifier is trained efficiently.

Additionally, we will consult the book "Computer Vision: Algorithms and Applications" by Richard Szeliski, particularly Chapter 6, which focuses on face recognition. The book discusses techniques for comparing extracted features, such as histogram comparison, Euclidean distance, and other methods to evaluate similarity between faces. These techniques will be crucial when we move towards facial recognition.

The Viola-Jones algorithm for face detection uses a machine learning approach (specifically, AdaBoost) to train a classifier. It differs from Convolutional Neural Networks (CNNs) in that it is designed for efficient real-time detection by focusing on fast feature extraction, thus making it suitable for scenarios where time is a constraint. This aligns well with our goal of achieving real-time facial detection.
From our course, we have learned about feature detection, description, and matching techniques. In this project, we will apply similar principles in detecting faces, starting with Haar feature extraction, followed by classification using AdaBoost. We will also explore image blurring techniques to enhance the detection process by reducing high-frequency noise, which relates directly to our coursework on preprocessing and image smoothing.

Milestones

Overall, we plan to evaluate each of our milestones based on whether we have successfully completed the tasks or not, along with the efficiency and output of our final product. Given that we are implementing the actual "training" part of the Viola-Jones algorithm, our benchmarks break down this process into smaller parts:

Milestone 1: 'D': Implement Preliminary Face Detection with Pretrained Haar Cascade Classifier

For our first milestone, we plan to use OpenCV's pretrained Haar cascade classifier to detect faces. We plan to use this on multiple face images from our dataset described below to consider the viability of this project. This milestone describes that our project is viable since if we are able to implement face detection through this pre-trained classifier, we should be able to create our Haar cascade classifier for face detection. One risk of this step is that relying on a pretrained classifier can cover certain issues we might face later on with our dataset, for example different lighting. However we can mitigate this risk by using a large dataset in our later milestones.

This milestone is achieved by the following tasks:

- Install OpenCV and understand how it works regarding face detection,
- Load the specific pretrained Haar Cascade model,
- Implement a simple program using this model to detect faces in multiple images,
- Draw boxes around the present faces and overlay them on the images.

This milestone is simple since the implementation of the classifier is already completed, and we will just be testing and applying this classifier on faces.

Milestone 2: 'C': Implement Image Integrals and Extract Haar Features

This brings us to our next milestone, where we will calculate the integrals of images and extract Haar Features based on these integrals. In this milestone, we wish to understand and implement integrals for Haar Features which is the foundation for the face detection algorithm we plan to build. However, this milestone also includes some risks. We may have integral image calculation errors because of incorrect implementation, which can affect our next milestones. We plan to mitigate this error by testing the correctness of our integral image function on a sample of images.

This milestone includes the following tasks:

- Implement a function which creates Integral Image data from Face Images. Ensure this is correct by testing out the function on multiple images.
- Use integral image data to compute Haar features such as edges and lines features through a sliding window implementation.
- Test Haar feature extraction function on multiple faces and non-faces images to verify that extraction is correct.

This milestone requires a deeper understanding of the Viola-Jones algorithm and Haar features for localization.

Milestone 3: 'B': Train a Face Detection Classifier

Next, we move on to using our Haar features to create a face detection classifier by implementing the AdaBoost algorithm. This proves to be a substantial milestone as it involves training the machine learning model through the features we have implemented. The risks of this implementation include insufficient training data and overfitting. Our training data might turn out to be imbalanced depending on the number of face images we end up finding. Additionally, our trained model may do well at detecting faces for our training dataset however fail to generalize new images leading to false positives and false negatives. We plan to mitigate this by using a sufficiently large and diverse dataset, along with thoroughly testing each component of our implemented AdaBoost algorithm before moving on to the next milestone.

In this milestone we have the following tasks:

- Implement Adaboost algorithm to combine classifiers based on extracted Haar features.
- Train classifier using a dataset of faces and non-faces.
- Test the classifier on a random set of images to see if it is correctly able to detect faces.
- Evaluate the performance of this strong classifier.

The AdaBoost algorithm is a complex algorithm that will showcase our understanding of feature detectors and how we want to combine multiple weak classifiers to create a strong classifier.

Milestone 4: 'A': Implement Real-Time Face Detection & Output Confidence Scores.

Lastly, we will move on to a real-time face detection system which will output confidence scores by using our previous milestones. One risk of this milestone is real-time performance issues. Due to the Haar feature extraction and classification being computationally expensive in a live video feed, it may be challenging to achieve real-time detection.

The tasks for this milestone include:

- Implement real-time face detection on live video output with our previously trained classifier.
- Display confidence scores for detected faces.
- Optimize the real-time performance to ensure that this system handles videos at a reasonable frame rate.
- Test this system with a multitude of conditions, these include lighting, poses, sizes of faces and number of faces.

This milestone requires a significant amount of work as it requires our trained model being integrated into a real-time working system. We plan to mitigate the risk of real-time performance issues by looking at a variety of techniques for optimization including image blurring, downscaling, or looking at a region of interest for faster image processing. We believe that successfully completing this milestones would make our project highly impressive.

Dataset

We will use the BioID Face Database, which includes 1521 grayscale images and marked eye positions for 23 individuals. These images depict frontal views of faces under different lighting and conditions, which will help in training and testing our face detection system. The BioID dataset will serve as the core training and testing data for our facial detection algorithm.

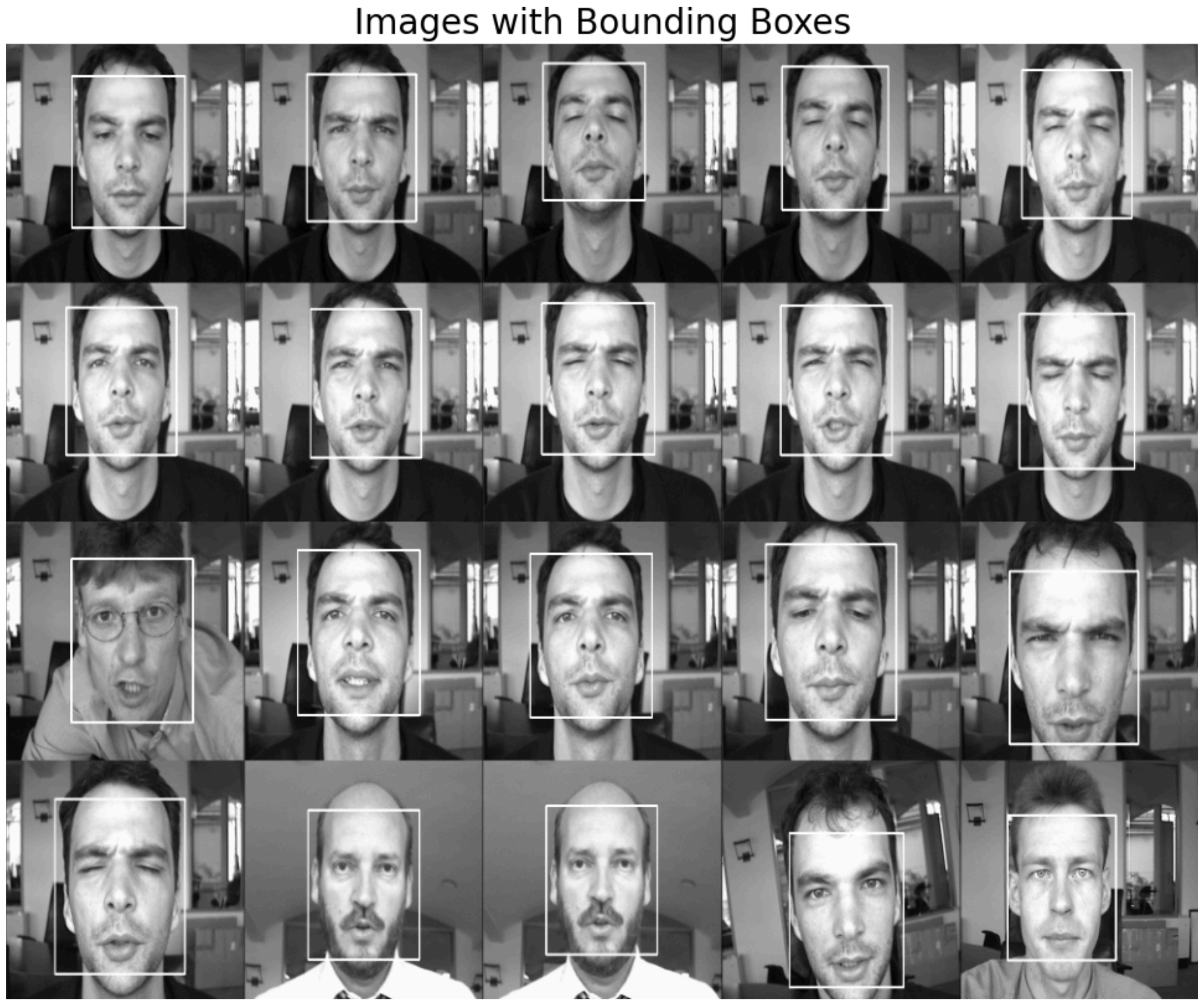
Evaluation

After training our model, we plan to compare the results with the original study by Viola & Jones, which is implemented in OpenCV's pretrained cascade classifier. We will compare the results across various accuracy metrics as well as test the system's real-time performance on video feeds.

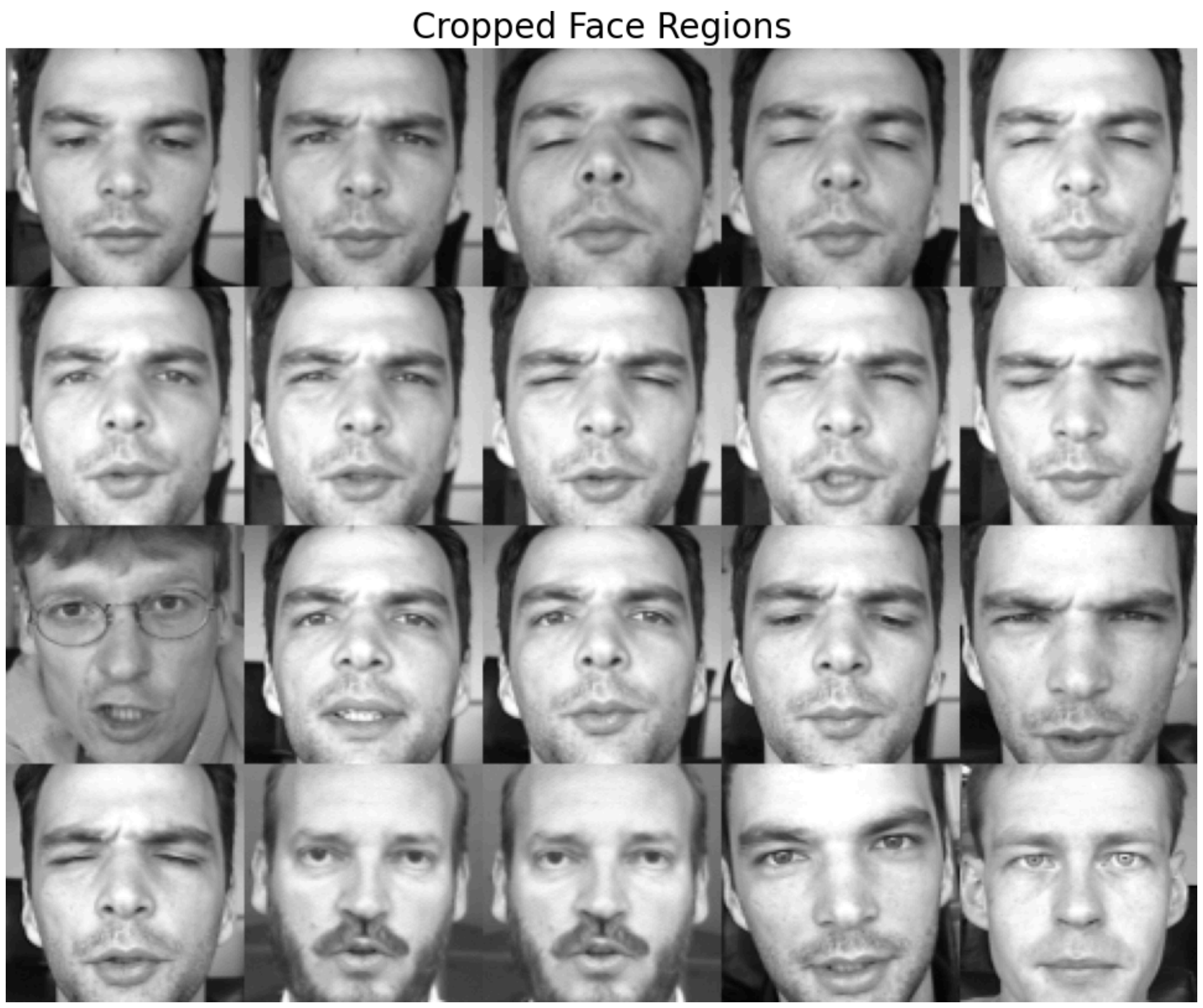
Feasibility & First Milestone

Using the GeeksForGeeks tutorial on OpenCV's face detection, we successfully tested the feasibility of the project and completed our first milestone. Using the pretrained Haar cascade classifier, we achieved face detection in ~40 ms, which translates to approximately 25 FPS. This performance suggests that real-time facial detection is viable for our system, aligning with the goals of the project.
The only limitation noted in this initial test is that the pretrained dataset primarily detects frontal faces. In future milestones, we will address challenges related to variations in lighting, poses, and non-frontal faces by expanding our dataset and optimizing the classifier.

Appendix



Detection of Faces in Images Using Haar Cascades (Bounding Box Montage). This figure showcases a montage of images where faces have been detected and highlighted with bounding boxes using OpenCV's Haar Cascade Classifier. The montage provides an overview of the face detection algorithm's performance across the first 20 images of the dataset. Each bounding box represents the identified face region, demonstrating the algorithm's ability to locate and outline faces in grayscale images effectively.



Extracted Face Regions from Detected Images (Cropped Face Montage). This figure presents a montage of cropped face regions extracted from the first 20 images of the dataset. Each cropped section corresponds to a detected face, resized to a uniform 100x100 pixels for consistent visualization. This montage highlights the ability of the detection process to isolate facial features, which can serve as input for further analysis, such as facial recognition or feature extraction tasks.

Bibliography

BioID Face Database. Retrieved from BioID.

GeeksForGeeks. *OpenCV Python program for Face Detection*. Retrieved from GeeksForGeeks.

Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*, Second Edition, Springer.

Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.