OpenCV project pipeline

Main function breakdown

This is the main function that simply calls the functions chosen by the user, this is done by uncommenting the correct includes as well as uncommenting the corresponding class and function calls.

1. **Include Statements:**
   - The program includes necessary header files such as `<iostream>`, `<opencv2/core.hpp>`, `<opencv2/opencv.hpp>`, and `<opencv2/core/utils/logger.hpp>`.
   - It also includes several header files related to different functionalities (which are commented out in the code).
2. **Global Variables:**
   - `faceCascadePath`: Specifies the file path for the primary face cascade classifier (Stump-based 24x24 Discrete AdaBoost Frontal Face Detector).
   - `logFilePath`: Specifies the file path for the log file.
   - `cascadePaths`: A vector of file paths for multiple cascade classifiers.
3. **Main Function (`main`):**
   - It sets the logging level to silent using OpenCV's utility functions (`utils::logging::setLogLevel`).
   - Instantiates an object of the `FaceDetection` class and calls the `runFaceDetection` method for testing various face detection functionalities.
   - Different functionalities are commented/uncommented based on the desired test case. The active functionality is multiple cascade classifiers on a single image.
4. **Usage:**
   - The program is designed to test various face detection functionalities by using different face cascade classifiers or configurations.
   - The commented/uncommented sections indicate the specific functionality being tested:
     - JPEG debugging
     - JPEG debugging with multiple cascade classifiers
     - Basic face detection
     - Webcam function with log
5. **FaceDetection Class (Assumed Implementation):**
   - The class is instantiated with different configurations based on the commented/uncommented sections.
   - The `runFaceDetection` method of the `FaceDetection` class is called to execute the face detection functionality.

Single cascade classifier on webcam feed pipeline

This C++ code implements a face detection application using the OpenCV library. The application captures video frames from a webcam, detects faces in each frame, draws rectangles around the detected faces, and displays the processed frames in a window. Additionally, the code keeps track of the total number of detected faces and logs information about each frame to a text file.

Here is a detailed breakdown of the code:

1. **Header Inclusions:**
   - `#pragma once`: A preprocessor directive to ensure that the header file is included only once during compilation.
   - `<iostream>`: Provides input and output stream functionalities.
   - `<fstream>`: Enables reading and writing to files.
   - `<opencv2/core.hpp>` and `<opencv2/opencv.hpp>`: Header files from the OpenCV library for core functionalities and computer vision operations.
2. **Namespace:**
   - `using namespace cv;`: Brings the `cv` (OpenCV) namespace into the current scope, allowing the use of OpenCV functions and classes without prefixing them with `cv::`.
3. **Class Definition:**
   - `class FaceDetection { ... };`: Defines a class named `FaceDetection` to encapsulate the face detection functionality.
4. **Public Interface:**
   - `public:`: Specifies the public interface of the class.
   - `FaceDetection(const String& faceCascadePath, const String& logFilePath);`: Constructor that initializes the `FaceDetection` object with the paths to the face cascade classifier XML file and the log file.
   - `int runFaceDetection();`: Method to start the face detection process.
5. **Private Members:**
   - `private:`: Specifies the private members of the class.
   - `CascadeClassifier faceCascade;`: An instance of `CascadeClassifier` for face detection using a Haar cascade classifier.
   -
   - `VideoCapture cap;`: An instance of `VideoCapture` for capturing video frames from the webcam.
   - `String logFilePath;`: The path to the log file where information about each frame is logged.
   - `int totalDetectedFaces;`: Counter to keep track of the total number of detected faces.
   - `std::ofstream logFile;`: An output file stream for writing log information.
6. **Constructor Implementation:**
   - `FaceDetection::FaceDetection(const String& faceCascadePath, const String& logFilePath)`: The constructor initializes the object by loading the face cascade classifier, opening the webcam, and initializing the log file.
7. **Face Detection Loop:**
   - `int FaceDetection::runFaceDetection()`: This method contains the main loop for face detection.
     - It creates a window named "Face Detection" for displaying the video frames.
     - Inside the loop:
       - Retrieves a frame from the webcam.
       - Converts the frame to grayscale.
       - Detects faces in the grayscale frame using the `detectMultiScale` method.

    - Draws rectangles around the detected faces on the original frame.
    - Displays the processed frame in the window.
    - Updates the total number of detected faces.
    - Logs information about each frame, including the frame number and the number of detected faces, to the log file.
    - Prints the number of faces in the current frame and the total number of detected faces to the console.
    - The loop continues until the user presses the 'Esc' key.
  - After the loop, it releases the webcam, closes the window, and closes the log file.

8. **Main Program Entry Point:**
   - `void faceDetection(const String& faceCascadePath, const String& logFilePath)` : A function to be called from the main program to initiate the face detection process using the `FaceDetection` class.
9. **Function Implementation:**
   - The `faceDetection` function creates an instance of `FaceDetection` , passing the face cascade and log file paths, and then calls the `runFaceDetection` method.

Overall, the code provides a simple face detection application that is extensible and can be integrated into larger projects for real-time face detection from a webcam feed.

Single cascade classifier on image pipeline

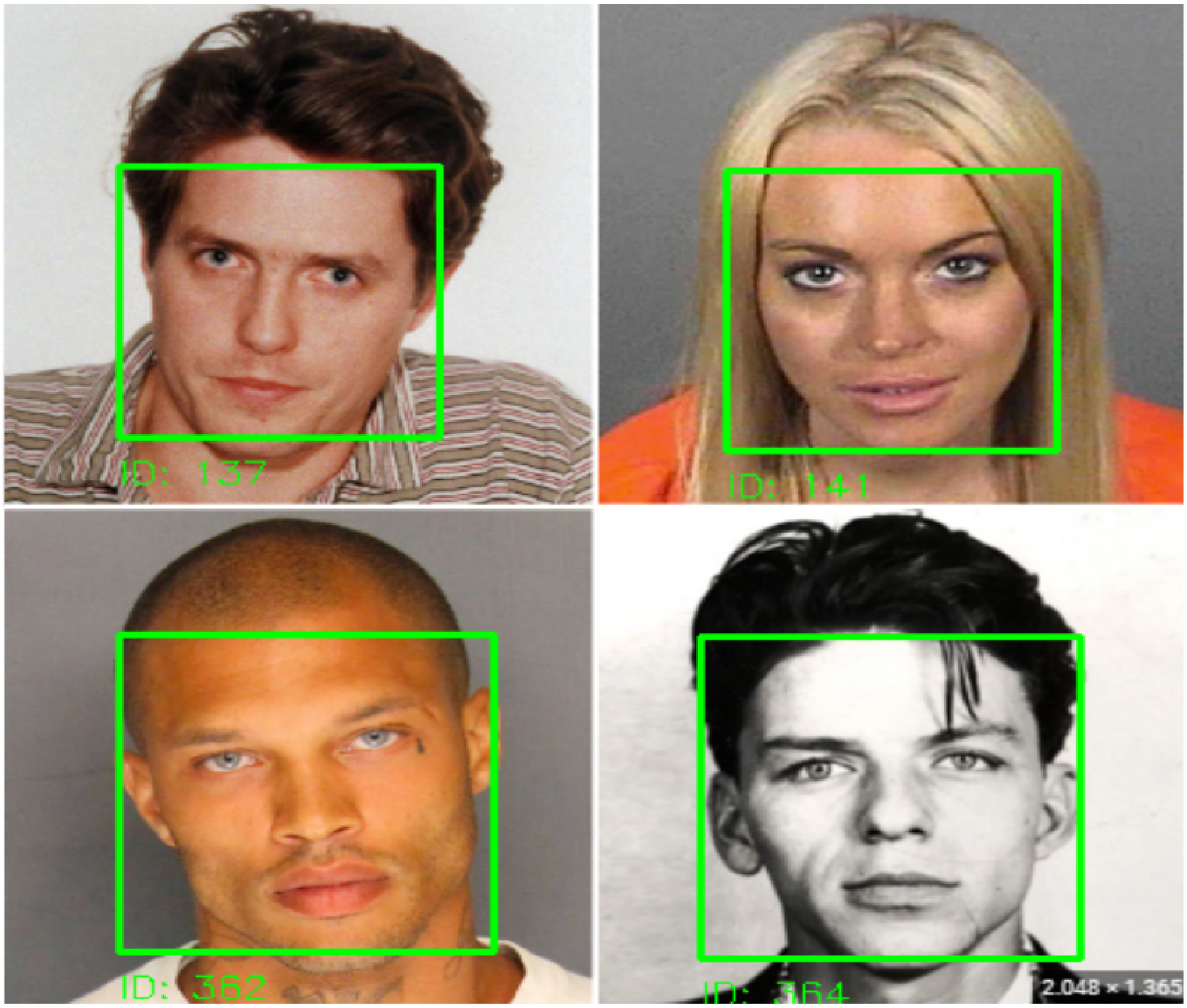
This program is a C++ implementation of a face detection application using the OpenCV library and a single Haar face cascade classifier. Let's break down the key components and functionality:

1. **Header and Include Files:**
   - The program includes necessary header files such as `<iostream>` , `<fstream>` , `<opencv2/core.hpp>` , and `<opencv2/opencv.hpp>` .
2. **Global Variables:**
   - `imagePath` : Specifies the file path of the input image.
   - `screenWidth` and `screenHeight` : Define the dimensions for resizing the displayed image.
3. **FaceDetection Class:**
   - Represents a face detection application.
   - **Constructor (`FaceDetection`):**
     - Takes parameters for the image path (`imagePath`), a single cascade classifier path (`cascadePath`), and the log file path (`logFilePath`).
     - Initializes member variables, including loading a single face cascade classifier from the provided path.
   - **runFaceDetection Method:**
     - Loads an image from the specified file path (`imagePath`).
     - Converts t

       he image to grayscale.

        - Uses `detectMultiScale` with a single cascade classifier to identify faces in the grayscale image.
        - Draws rectangles around the detected faces, assigns a unique identifier (`faceId`) based on the vertical position of the detected face.
        - Checks if the face is unique using a set (`uniqueFaceIds`), and if so, logs information about the new face to a file.
        - Resizes the image for display and shows it in a window named "face detection."
        - Pauses the program until a key is pressed (`waitKey(0)`).
        - Closes all OpenCV windows and the log file.
    - **Other Member Variables:**
        - `faceCascade` : A single `CascadeClassifier` object representing a face cascade classifier.
        - `logFilePath` : A string specifying the path for the log file.
        - `uniqueFaceIds` : An unordered set to keep track of unique faces detected.
        - `logFile` : An output file stream for logging face detection information.
        - `lastDetectedFace` : A rectangle storing the coordinates of the last detected face.
        - `totalUniqueFaces` : A counter for the total number of unique faces detected.
4. **faceDetection Function:**
    - A function to be called from the main program.
    - Creates an instance of the `FaceDetection` class and runs the face detection process.
5. **Usage:**
    - The program is designed to load an image, perform face detection using a single cascade classifier, and display the results. It logs information about unique faces to a file.

Note: The program uses the OpenCV library, particularly the `CascadeClassifier` for face detection, and the display functions like `imshow` and `waitKey` for visualizing the results. The unique identifier for faces is based on their vertical position in the image.



Multiple cascade classifiers on image pipeline

This program is a C++ implementation of a face detection application using the OpenCV library and multiple Haar face cascade classifiers. Let's break down the key components and functionality:

1. **Header and Include Files:**
   - The program includes necessary header files such as `<iostream>`, `<fstream>`, `<opencv2/core.hpp>`, and `<opencv2/opencv.hpp>`.
2. **Global Variables:**
   - `imagePath`: Specifies the file path of the input image.
   - `screenWidth` and `screenHeight`: Define the dimensions for resizing the displayed image.
3. **FaceDetection Class:**
   - Represents a face detection application.
   - **Constructor (`FaceDetection`):**
     - Takes parameters for the image path (`imagePath`), a vector of cascade classifier paths (`cascadePaths`), and the log file path (`logFilePath`).

        - Initializes member variables, including loading multiple face cascade classifiers from the provided paths.
    - **runFaceDetection Method:**
        - Loads an image from the specified file path (`imagePath`).
        - Converts the image to grayscale.
        - Iterates through each cascade classifier in the `faceCascades` vector.
        - Uses `detectMultiScale` to identify faces in the grayscale image, drawing rectangles around the detected faces.
        - Assigns a unique identifier (`faceId`) based on the vertical position of the detected face.
        - Checks if the face is unique using a set (`uniqueFaceIds`), and if so, logs information about the new face to a file.
        - Resizes the image for display and shows it in a window named "face detection."
        - Pauses the program until a key is pressed (`waitKey(0)`).
        - Closes all OpenCV windows and the log file.
    - **Other Member Variables:**
        - `faceCascades`: A vector of `CascadeClassifier` objects representing multiple face cascade classifiers.
        - `logFilePath`: A string specifying the path for the log file.
        - `uniqueFaceIds`: An unordered set to keep track of unique faces detected.
        - `logFile`: An output file stream for logging face detection information.
        - `lastDetectedFace`: A rectangle storing the coordinates of the last detected face.
        - `totalUniqueFaces`: A counter for the total number of unique faces detected.
4. **faceDetection Function:**
    - A function to be called from the main program.
    - Creates an instance of the `FaceDetection` class and runs the face detection process.
5. **Usage:**
    - The program is designed to load an image, perform face detection using multiple cascade classifiers, and display the results. It logs information about unique faces to a file.

Note: The program uses the OpenCV library, particularly the `CascadeClassifier` for face detection, and the display functions like `imshow` and `waitKey` for visualizing the results. The unique identifier for faces is based on their vertical position in the image.