PROJECT

on

FACE DETECTION USING JAVA

Submitted to

GALGOTIAS UNIVERSITY

In partial fulfilment of the requirements for the award of the degree

of

Bachelor of Computer Application

by

MOHD SALMAN (22SCSE1040049)

AARYAN KUMAR SRIVASTAV (22SCSE1040030)

TRIBHUWAN SINGH (22SCSE1040026)

2022 - 2025

Under the guidance of

Dr. AVNEESH KUMAR & Dr. SUDEEPT YADAV

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY

UTTAR PRADESH

REPORT APPROVAL

The project report titled **“FACE DETECTION USING JAVA”** by **Mohd Salman, Aaryan Kumar Srivastav and Tribhuwan Singh** has been approved for the degree of **Bachelor of Computer Application**.

Signature: ………………………………………….

Project Guide

CERTIFICATE

I hereby certify that the work which is being presented in the project entitled **“FACE DETECTION USING JAVA”** in partial fulfilment of the requirement for the degree of **Bachelor of Computer Application** in **“School of Computer Application”** affiliated to **Galgotias University, Uttar Pradesh**, in an authentic record of our own work carried out during the period for **04/2024 to 05/2024** under the guidance of **Dr. Avneesh Kumar & Dr. Sudeept Yadav**, the matter embodied in this project has not been submitted by us for another degree.

MOHD SALMAN (22SCSE1040049)
AARYAN KUMAR SRIVASTAV (22SCSE1040030)
TRIBHUWAN SINGH (22SCSE1040026)

This is to certify the above statement made by the candidate is true to the best of my knowledge.

Signature of Project Supervisor
Dr. Avneesh Kumar & Dr. Sudeept Yadav
School Of Computer Application And Technology
Galgotias University
Uttar Pradesh

Date: 30-05-2024

DECLARATION

We, **Mohd Salman, Aaryan Kumar Srivastav and Tribhuwan Singh**, hereby declare that this major project report entitled **" FACE DETECTION USING JAVA "** is the result of my own research and investigation. All sources of information used in this report have been duly acknowledged and referenced. I affirm that the work presented in this report is original and has not been submitted elsewhere for any other degree or qualification. Any contributions made by others to this work are appropriately cited and acknowledged. Furthermore, I understand the academic honesty policy of **"School of Computer Application"** and affirm that this report is free from any form of plagiarism or academic misconduct. I take full responsibility for the content and conclusions presented in this report.

MOHD SALMAN (22SCSE1040049)

AARYAN KUMAR SRIVASTAV (22SCSE1040030)

TRIBHUWAN SINGH (22SCSE1040026)

TABLE OF CONTENT

Step-by-Step Simplified Guide in VS Code :

Step 1: Download and Install OpenCV

1. **Download OpenCV**:
    - Go to the OpenCV releases page and download the latest version of OpenCV for your operating system.
2. **Extract OpenCV**:
    - Extract the downloaded archive to a directory on your computer (e.g., **C:\opencv** on Windows or **/usr/local/opencv** on Linux/Mac).

Step 2: Configure VS Code Project

1. **Install Java Extensions in VS Code**:
    - Open VS Code.
    - Go to the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window.
    - Search for and install the **Java Extension Pack**.
2. **Create a New Java Project**:
    - Open the command palette (**Ctrl+Shift+P** or **Cmd+Shift+P** on macOS) and select **Java: Create Java Project**.
    - Choose **No Build Tools** for simplicity.
    - Enter the project name (e.g., **OpenCVTest**) and press Enter.
    - Open the newly created project folder in VS Code.
3. **Add OpenCV Library**:
    - Create a **lib** directory in your project folder.
    - Copy the **opencv-<version>.jar** file from the OpenCV extracted directory (e.g., **C:\opencv\build\java\opencv-450.jar**) to the **lib** directory.

Step 3: Verify OpenCV Setup

1. **Create a New Java Class**:
    - In the **src** directory of your project, create a new Java file named **OpenCVCheck.java** with the following content:

```java
import org.opencv.core.Core;

public class OpenCVCheck {
    public static void main(String[] args) {
        // Load the OpenCV native library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        // Print the OpenCV version
        System.out.println("OpenCV version: " + Core.VERSION);
    }
}
```

2. **Configure Build Tasks**:
    - Create a **tasks.json** file in the **.vscode** directory with the following content:

```json
{
    "version": "2.0.0",
    "tasks": [
        {
            "label": "build",
            "type": "shell",
            "command": "javac",
            "args": [
                "-cp",
                "lib/opencv-490.jar",
                "-d",
                "bin",
                "src/App.java"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "problemMatcher": ["$javac"],
            "detail": "Generated task by Java extension"
        }
    ]
}
```

3. **Add Native Library Path**:
    - Create a **launch.json** file in the **.vscode** directory with the following content:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "Launch App",
            "request": "launch",
            "mainClass": "App",
            "vmArgs": "-Djava.library.path=\"G:/Program 
Files/OpenCV/opencv/build/java/x64\"" // Adjust this path
        }
    ]
}
```

    Adjust the **java.library.path** to point to the directory containing the OpenCV native libraries:

    - For Windows: **C:/opencv/build/x64/vc15/bin** (or similar).
    - For Linux/Mac: navigate to the appropriate directory containing the **.so** or **.dylib** files.

4. **Build and Run the Program**:
    - Open the terminal in VS Code and run the build task: Ctrl+Shift+B
    - After building the project, run the program using the debugger (F5).

Expected Output

If everything is configured correctly, the Terminal window in VS Code should display:

OpenCV version: 4.5.5

Or a similar version number depending on the version of OpenCV you downloaded.

Troubleshooting

- **UnsatisfiedLinkError**: If you get this error, it means that VS Code cannot find the native OpenCV libraries. Double-check that the **java.library.path** in **launch.json** is correctly set to the folder containing the **.dll**, **.so**, or **.dylib** files.
- **ClassNotFoundException**: If you see this error, it indicates that the OpenCV JAR file is not correctly added to your classpath. Ensure that the JAR file path is correctly specified in the **tasks.json**.

Directory Structure

```
OpenCVTest/
├── .vscode/
│   ├── launch.json
│   ├── tasks.json
│
├── lib/
│   ├── opencv-450.jar
│
├── src/
│   ├── OpenCVCheck.java
│
├── models/
│
├── images/
│
├── bin/
│
```

Implement Face Landmark Detection

To implement face landmark detection, you can follow these steps:

1. **Load the Pre-trained Models**:
   - Download pre-trained models for face detection and landmark detection.
   - Typically, you can use Haar cascades for face detection and shape predictor models (like Dlib's **shape_predictor_68_face_landmarks.dat**) for landmark detection.
2. **Write Code to Detect Faces and Landmarks**:

Here is an example code snippet that uses Haar cascades for face detection. For simplicity, let's assume you have a face landmark model and the required files in the **models** directory.

Example Code for Face Detection

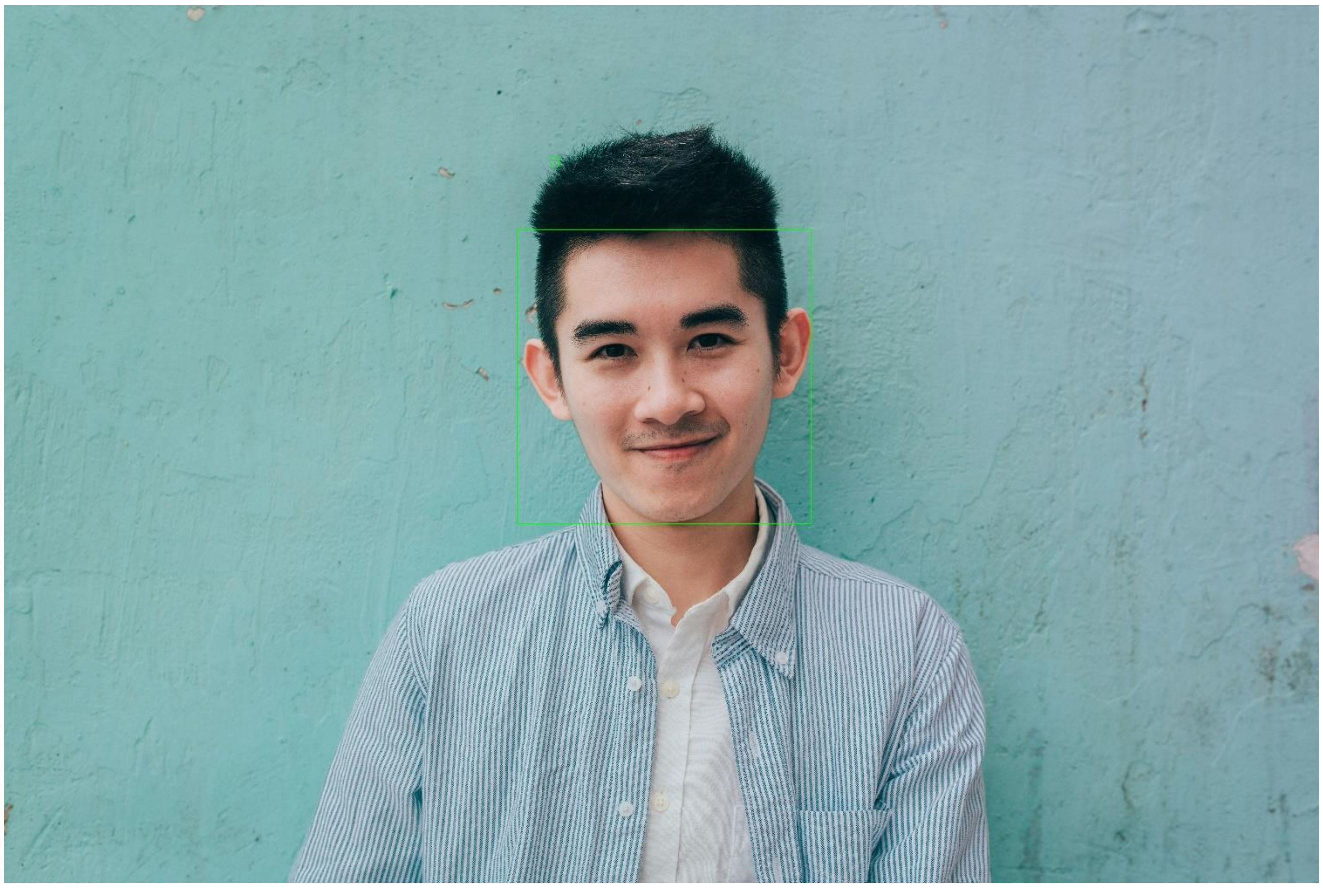
1. **Download Haar Cascades for Face Detection**:
   - You can download it from OpenCV GitHub repository.
2. **Place the haarcascade_frontalface_default.xml file in a model’s directory in your project.**

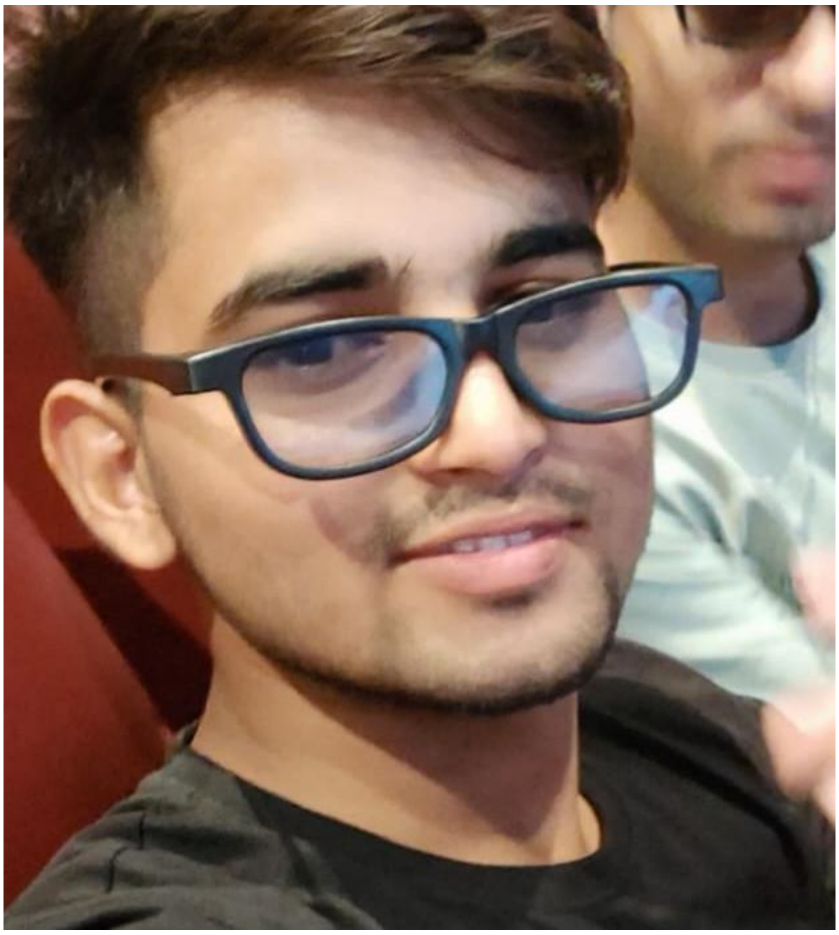
CODE

```java
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

public class App {
    public static void main(String[] args) {
        // Load the OpenCV native library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        // Path to the image file
        String imagePath = "Image\\boy5.jpg"; // Replace with your image path

        // Load the image
        Mat image = Imgcodecs.imread(imagePath);

        if (image.empty()) {
            System.out.println("Could not open or find the image!");
            return;
        }

        // Path to the Haar cascade XML file
        String faceCascadePath = "models\\haarcascade_frontalface_default.xml";

        // Load the Haar cascade classifier for face detection
        CascadeClassifier faceCascade = new CascadeClassifier(faceCascadePath);

        if (faceCascade.empty()) {
            System.out.println("Could not load Haar cascade classifier!");
            return;
        }

        // Detect faces in the image
        MatOfRect faces = new MatOfRect();
        faceCascade.detectMultiScale(image, faces, 1.1, 3, 0, new Size(30, 30), new
Size());

        // Draw rectangles around detected faces
        for (Rect face : faces.toArray()) {
            Imgproc.rectangle(image, face.tl(), face.br(), new Scalar(0, 255, 0), 3);
        }

        // Save the result image
        String outputPath = "output_image.jpg";
        Imgcodecs.imwrite(outputPath, image);

        System.out.println("Face detection complete. Result saved at: " + outputPath);
    }
}
```

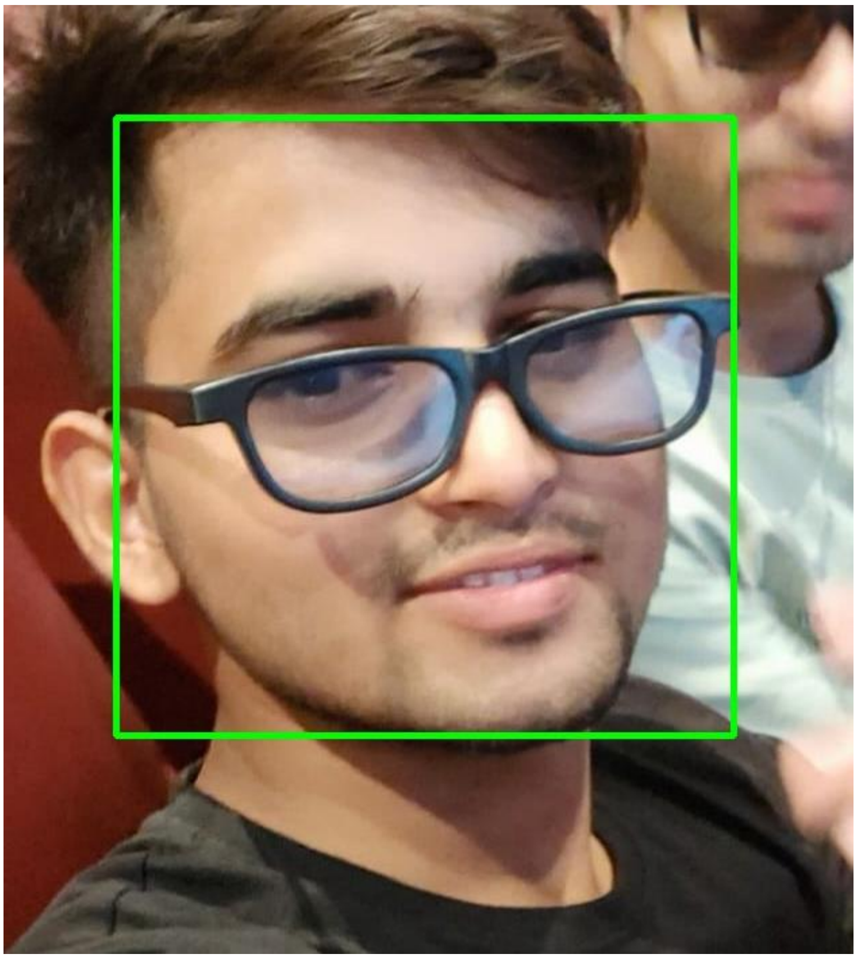
Input Image

Output Image





Output Image



Input Image

Output Image