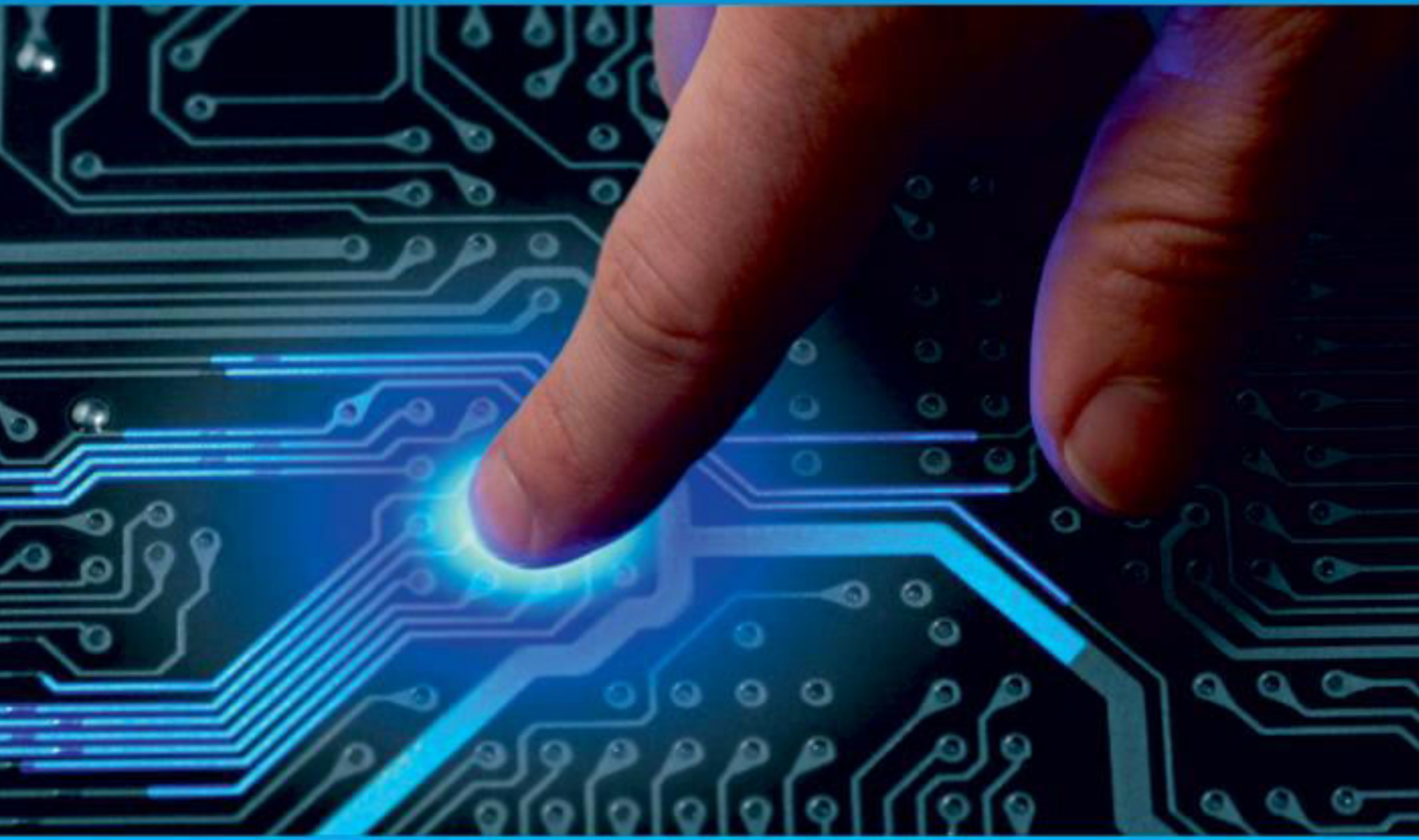
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 3, March 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203011 |

Biometric Matching System Implementation

Sayali Shinde, Vaishnavi Chavan, Trupti Pardeshi

Department of Computer Science, Jayawantrao Sawant Polytechnic, Hadapsar, Pune, India

ABTRACT: This documentation mainly describes about an platform independent Python project which mainly uses the predefined packages that is the ‘opencv’ package & ‘face_regonition’ package. The main function of opencv package is to open the build in camera of the system and face_regonition is used to load the images to be processed and to remember them. Also, some graphics are also used to make the product more attractive like a frame to set a border for the face and frame it self will contain the name and some short information about the person.

KEYWORDS:- Biometric matching system , Python , OpenCV, Facial recognition, Facial landmark detection , Feature extraction, False acceptance rate, False rejection rate, Preprocessing techniques.

I. INTRDUCTION

Basically this is a simple Python program which is written in Pycharm platform in which the necessary packages are installed and used as per the need.
While in this program we have used the two main packages that is the ‘opencv’ and ‘face_recognition’ packages.
Theses above mentioned packages are externally by using the keyword ‘pip’. Here the pip word is used the install any latest version of the package.

The program’s output is visible on the window itself. The output is on a new window along with the camera open.
In recent years, Python and OpenCV have gained popularity in the field of computer vision and biometric matching systems. In this paper, we will provide an introduction to implementing a biometric matching system using Python and OpenCV.
The first step in implementing a biometric matching system using Python and OpenCV is to collect high-quality biometric data, such as facial images, from the users. This data can be obtained using a webcam or a digital camera.
To preprocess the facial images, we can use OpenCV's built-in functions to perform tasks such as grayscaling, cropping, and normalization. We can also apply techniques such as histogram equalization and Gaussian filtering to enhance the features of the images.
Next, we can extract unique features from the preprocessed facial images using OpenCV's Facial Landmark Detection algorithm. This algorithm detects facial landmarks such as eyes, nose, and mouth, which can be used to generate a unique facial signature for each user.
To compare the facial signatures of two users, we can use a distance metric such as Euclidean distance or L2 norm to calculate the similarity between their signatures. If the distance between the signatures is below a certain threshold, we can consider the users to be a match.

To ensure accuracy and reliability, it is essential to implement robust error handling mechanisms in our biometric matching system. This includes techniques such as liveness detection, which ensures that the input biometric data is not spoofed or fake, and rejection thresholds, which prevent false matches from being accepted.
In conclusion, implementing a biometric matching system using Python and OpenCV requires careful consideration of various factors such as data collection and preprocessing, feature extraction, distance metrics, error handling mechanisms, and threshold values. By following best practices and employing robust techniques, organizations can deploy secure and reliable biometric matching systems that provide enhanced security and convenience for their users using Python and OpenCV.

II. PROBLEM STATEMENT

Providing security for high organizations and also to highly authorized persons like Politicians.
Now a days as security is being one of the most important part of our life, security might be with our personal information/data or with our personal things and many more. On a large scale security is the most important part, like for any organization, departments or even in any public conference security is a high point.
This project simple differentiate between the authorized and unauthorized person. Based on the data saved or encoded in the program this differentiation will work. Basically, it differentiate based on the facial data encoded in it.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203011 |

/*For example, on college level organization, there are different departments/classes, each department has its own identifier like ID cadrds or even some other thing. Now to encode the data of students or teachers, the data needed is their respective names and their department. This data will store in a folder named images, in the same folder sub-folders will be created according to the departments. While recognizing the person it will search the in encoded images if found then it will display the person's name and its department. If data is not found then it will display unknown person.*/

III. PACKAGES USED

In this project we have used two main packages first one is the 'opencv' package and the second is 'face_regonition' package which were externally installed using 'pip' command.

OpenCV Package

OpenCV package is an open-source package in which the developer can easily use them, modify them by using their own logic to develop something which is out of box. It is used for machine learning and image processing. it is also the most important part for any real-time project. It helps to process any Real-time image or video.
First version of OpenCV was 1.0 which was released under BSD(Berkeley Source Distribution) license and it id free for both academic and commercial work. It is written in optimized C/C++ to take advantage of multi core processing. OpenCV is platform independent and can has interfaces in C, C++, Java and Python.
OpenCV package is integrated with various libraries like NumPy, Python, etc. Here Python is capable of processing the OpenCV array structure for analysis.
In OpenCV the word CV stands for 'Computer Vision'.
It preforms two main tasks as: -

1. Object Classification
2. Object identification

- OpenCV Functionality: -
- ✓ Image/video I/O, processing, display (core, imgproc, highgui)
- ✓ Object/feature detection (objdetect, features2d, nonfree)
- ✓ Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- ✓ Computational photography (photo, video, superres)
- ✓ Machine learning & clustering (ml, flann)
- ✓ CUDA acceleration (gpu)
- Applications of OpenCV are:-
- ✓ Face Recognition
- ✓ Number of people-count
- ✓ Interactive art installations
- ✓ Object recognition
- ✓ Movies-3d structure from motion
- ✓ Medical image analysis

IV. FACE RECOGNITION PACKAGE

The Face Recognition package in Python is an open-source library that allows developers to implement face recognition functionality in their applications.
It uses the OpenCV (Open-Source Computer Vision) library as its base and provides a simple and intuitive interface for training and testing face recognition models. Overall, the Face Recognition package in Python is a powerful and accessible tool for developers looking to add face recognition capabilities to their projects.
The Face Recognition package in Python is developed in the Python programming language. It is a Python module that provides a set of functions and classes for implementing facial recognition applications using computer vision techniques. The package is open-source and freely available on GitHub, making it accessible to developers around the world. It can be easily integrated into other Python projects and is compatible with popular Python frameworks like Flask, Django, and PyQt.

- **Face Recognition Functionality: -**

✓ Face Detection
✓ Face Alignment
✓ Face Recognition
✓ Face Verification
✓ Face Identification
✓ Real-Time Face Recognition

- **Application of Face Recognition: -**

✓ Smart Homes
✓ Education
✓ Health Care
✓ Marketing & Advertising
✓ Attendance Tracking
✓ Security & Surveillance

V. METHODOLOGY

Some basic steps are followed to develop this project.

1. Installed Python's latest version that is 3.12.1 or else we can install Pycharm that is also a platform for python development
2. Once install do proper setup
3. Use 'pip' keyword to install the all needed packages like Opencv & face recognition. Commands used are:-' pip install opencv-python' & 'pip install face_recognition'
4. After packages are installed properly, import them in the program. 'import cv2
5. import face_recognition'
6. Make a separate folder named images, which will contain all the images to be encoded in the program.
7. Using Opencv package open the camera to capture real time images
8. First encode the images using 'face_recognition.face_encoding(image floder)'
9. Take webcam stream to make the camera on going
10. Set proper face location and it's recognition. The final output will be decided, what will happen if face is recognized, how the face will be bordered on the screen
11. Testing.

VI. TOOLS AND TECHNOLOGY

1. Python Programming Language :-

Python is a widely-used, high-level, general-purpose programming language that is easy to learn and read due to its simple syntax. It is an open-source language that supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is known for its versatility and can be used for various purposes, such as web development, data analysis, scientific computing, machine learning, and automation. Its vast standard library and third-party modules make it a powerful tool for developers of all levels. Python's popularity is growing rapidly due to its ease of use, flexibility, and the availability of a large community of developers who contribute to its development and provide support.

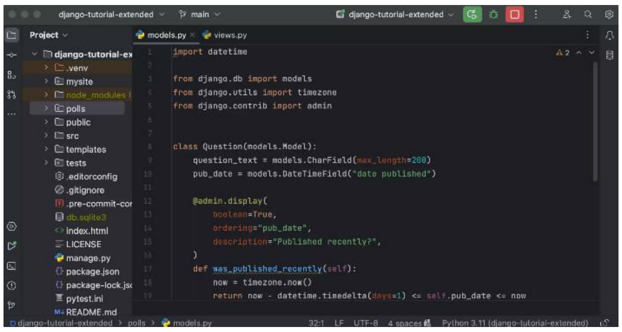
2. PyCharm IDE :-

Pycharm is a powerful Integrated Development Environment (IDE) specifically designed for Python programming. It provides a user-friendly interface with advanced features such as code completion, debugging, and refactoring tools, making it an ideal choice for Python developers of all levels. PyCharm also supports multiple Python versions and frameworks.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203011 |



```python
import datetime

from django.db import models
from django.utils import timezone
from django.contrib import admin


class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField("date published")

    @admin.display(
        boolean=True,
        ordering="pub_date",
        description="Published recently?",
    )
    def was_published_recently(self):
        now = timezone.now()
        return now - datetime.timedelta(days=1) <= self.pub_date <= now
```

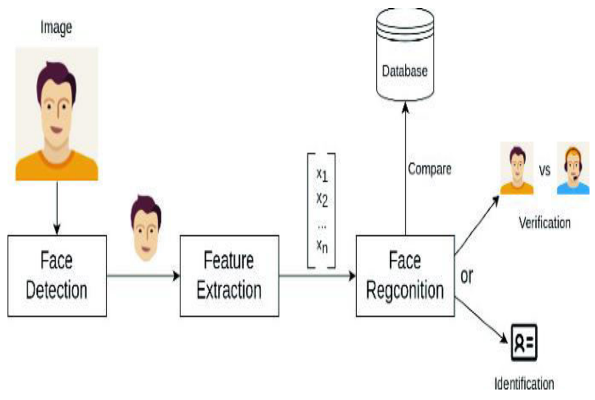
3. OPENCV Library:-

Python is a high-level, general-purpose, interpreted, dynamic programming language with a focus on readability and simplicity. It is open-source and has a large community of developers who contribute to its development and provide support. Python's syntax is easy to learn, making it a popular choice for beginners. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has a vast standard library and third-party modules that make it a versatile tool for various purposes, such as web development, data analysis, scientific computing, machine learning, and automation. Python's interactive interpreter allows developers to test code snippets quickly and iteratively. Its cross-platform compatibility ensures that it can run on different operating systems.

4. Webcam:-

A webcam is an optical sensor that can capture video frames in real-time. It consists of a lens, an image sensor, and a digital signal processor (DSP) that converts the analog signals from the sensor into digital format. Webcams can be connected to a computer via USB or other interfaces and can be accessed and controlled using programming languages like Python. In Python, the OpenCV library provides a simple and efficient way to access and control webcams. OpenCV is an open-source computer vision library that provides a wide range of functions for image and video processing. It supports multiple operating systems, including Windows, Linux, and macOS, and can be used with various programming languages like Python, C++, and Java. To use a webcam with OpenCV in Python, you need to import the necessary modules, initialize the webcam, and then capture frames using the `cv2.VideoCapture()` function. This function returns a `VideoCapture` object that provides methods for controlling the webcam, such as setting the resolution, framerate, and exposure. Once you have initialized the webcam, you can capture frames using the `read()` method of the `VideoCapture` object. This method returns a tuple containing two values: a boolean flag indicating whether the frame was successfully captured (`ret`), and the captured frame itself (`frame`). You can then display the frame using the `imshow()` function of OpenCV or save it to a file using the `imwrite()` function. Overall, using a webcam with Python and OpenCV provides a powerful tool for real-time video processing applications like computer vision, machine learning, and robotics.

VII. SYSTEM ARCHITECTURE



1. Input: The system takes input from various biometric sensors such as fingerprint scanners, iris scanners, or facial recognition cameras. These sensors capture biometric data in the form of images or videos.

2. Preprocessing: The captured biometric data goes through a preprocessing stage where it is cleaned, normalized, and enhanced to improve the quality of the data. This stage includes tasks such as noise reduction, contrast enhancement, and image normalization.

3. Feature Extraction: The preprocessed biometric data is then fed into a feature extraction module where unique features are extracted from the data. This stage includes tasks such as fingerprint minutiae extraction, iris segmentation, and facial landmark detection.

4. Matching: The extracted features are then compared with the features stored in a database using a matching algorithm. The matching algorithm calculates the similarity between the features and returns a match score.

5. Decision: Based on the match score, the system decides whether to accept or reject the user's identity claim. If the match score is above a certain threshold, the user is authenticated; otherwise, they are rejected.

6. Output: The system outputs the decision in the form of a message or notification to the user or system administrator.

7. Database Management: The system maintains a database of registered users' biometric data and their corresponding identities. This database is updated whenever a new user registers or an existing user's biometric data changes.

Overall, implementing a Biometric Matching System using OpenCV and Python involves various stages such as input capture, preprocessing, feature extraction, matching, decision-making, output generation, and database management. By following this system architecture diagram, you can develop an efficient and reliable Biometric Matching System using OpenCV and Python.

VIII. APPLICATIONS, ADVANTAGES AND DISADVANTAGES OF BIOMETRIC SYSTEM

Applications:-

1. Biometric matching systems can be used to control access to secure areas such as data centers, laboratories, and server rooms.
2. Biometric matching systems can be used by law enforcement agencies to identify suspects, verify identities, and prevent crimes.
3. Biometric matching systems can be used by banks and financial institutions to authenticate customers and prevent fraud.
4. Biometric matching systems can be used in healthcare facilities to verify patient identities and prevent medical errors.
5. Biometric matching systems can be used in educational institutions to monitor attendance and prevent cheating during exams.
6. Biometric matching systems can be used in large industries and organizations for high level security.

Advantages:-

1. Biometric matching systems provide a high level of security as they rely on unique biological characteristics that are difficult to replicate or forge.
2. Biometric matching systems are convenient as they eliminate the need for physical tokens such as keys, cards, or passwords.
3. Biometric matching systems are fast and efficient, as they can quickly match biometric data with a large database of stored information.
4. Biometric matching systems are more accurate, as they can distinguish between similar biometric data with a high degree of confidence.
5. Biometric matching systems are cost-effective in the long run, as they eliminate the need for physical tokens and reduce the cost of lost or stolen items.

DisAdvantages:-

1. Biometric matching systems are subject to technical limitations such as lighting conditions, facial hair, and ageing, which could affect their accuracy and reliability.
2. Biometric matching systems can be expensive to implement.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 3, March 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1203011 |

X. FUTURE SCOPE

1. Multi-modal Biometrics: The use of multiple biometric modalities such as facial recognition, fingerprint recognition, and iris recognition can provide a higher level of accuracy and reliability.
2. Deep Learning: The use of deep learning algorithms can improve the accuracy and speed of biometric matching systems by enabling more sophisticated feature extraction and distance metric calculations.
3. Cloud-based Systems: The use of cloud-based systems can provide a more scalable and flexible solution for biometric matching systems, as it enables the storage and processing of large amounts of biometric data in a centralized location.
4. Mobile Devices: The use of mobile devices for biometric matching systems can provide a more convenient and accessible solution for users, as it enables the use of biometric data on-the-go.

XI. CONCULSION

Biometric matching systems have gained significant attention in recent years due to their high level of security, convenience, and accuracy. The implementation of biometric matching systems using Python and OpenCV offers a cost-effective and efficient solution for various applications such as access control, law enforcement, banking, healthcare, and education. However, the use of biometric matching systems also raises privacy concerns and technical limitations that need to be addressed. To ensure the successful implementation of biometric matching systems, it is essential to consider factors such as accuracy, speed, cost, privacy, and technical limitations.

REFERENCES

1. "Biometric Authentication Using Python" by AnkitChoudhary. (https://www.analyticsvidhya.com/blog/2018/06/biometric-authentication-using-python/) This study provides an overview of biometric authentication systems and their implementation using Python and OpenCV. [1]
2. "Facial Recognition Using OpenCV and Python" by Abhishek Thakur (https://www.datacamp.com/community/tutorials/facial-recognition-opencv-python) This study focuses on the implementation of facial recognition systems using Python and OpenCV. [2]
3. “Biometric Authentication System Using Python" by Abhishek Singh (https://www.geeksforgeeks.org/biometric-authentication-system-using-python/) This study provides a step-by-step guide for implementing a biometric authentication system using Python and OpenCV. [3]
4. "Deep Learning for Biometric Recognition" by Yuxin Wu et al. (https://arxiv.org/abs/1806.06678) This study focuses on the use of deep learning algorithms for improving the accuracy of biometric recognition systems. [4]