
EXTERNSHIP PROJECT: VIT AP CAMPUS

TITLE: FACE DETECTION

GROUP: ARAB KALEEMULLA SHAHANSHA (20BCI7039)

MOHAMED HAZIL (20BCI7224)

DILKAS V ANAS (20BCI7047)

K.LAALASA (20BCI7036)

1.INTRODUCTION:

1.1 Overview:

Face Detection is a application software to deal with human face. It has the provisions to collect image from the user so that they can detect the eyes, nose, mouth and whole face of human in the image. There are various advantages of developing an software using face detection and recognition in the field of authentication. Face detection is an easy and simple task for humans, but not so for computers. It has been regarded as the most complex and challenging problem in the field of computer vision due to large intra-class variations caused by the changes in facial appearance, lighting and expression. Face detection is the process of identifying one or more human faces in images or videos. It plays an important part in many biometric, security and surveillance systems, as well as image and video indexing systems. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. The project titled 'Face Detection and Recognition System', is to manage all the front end back end system of finding or detecting particular region in human face. This software helps the people looking for more advanced way of image processing system. Using this software they can easily find or detect faces in image and also recognize the face after saving that. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. A reliable face-detection approach based on the genetic algorithm and the eigen-face technique. Firstly, the possible human eye regions are detected by testing all the valley regions in the gray-level image. Then the genetic algorithm is used to generate all the possible face regions which include the eyebrows, the iris, the nostril and the mouth corners. Each possible

face candidate is normalized to reduce both the lightning effect, which is caused by uneven illumination; and the shirring effect, which is due to head movement. The fitness value of each candidate is measured based on its projection on the eigen-faces. After a number of iterations, all the face candidates with a high fitness value are selected for further verification. At this stage, the face symmetry is measured and the existence of the different facial features is verified for each candidate. Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face.

1.1 Purpose:

Whenever we implement a new system it is developed to remove the shortcomings of the existing system. The computerized mechanism has the more edge than the manual system. The existing system is based on manual system which takes a lot of time to get performance of the work. The proposed system is a web application and maintains a centralized repository of all related information. The system allows one to easily access the software and detect what he wants

2 LITERATURE SURVEY:

2.1 Existing problem:

There are several existing problems or challenges associated with face detection projects. Some of the common issues include:

Face detection algorithms often struggle with variations in lighting conditions, such as low-light environments or extreme backlighting. These variations can affect the visibility and contrast of facial features, making it difficult for the algorithm to accurately detect faces.

When a face is partially or fully occluded by objects like glasses, masks, hands, or other obstructions, it becomes challenging for face detection algorithms to identify and locate the face accurately.

Face detection algorithms may struggle with detecting faces in non-frontal poses. Detecting faces from different angles or orientations, such as side profiles or tilted heads, poses a challenge as the algorithm needs to account for the variations in facial structure.

Faces can appear in various sizes within an image, depending on factors like distance from the camera or image resolution. Detecting faces at different scales accurately is crucial, and algorithms need to be robust to scale variations.

Face detection algorithms may be affected by busy or cluttered backgrounds, where there are numerous objects or patterns that can confuse the algorithm and lead to false positives or false negatives in face detection.

Some face detection algorithms have shown biases towards certain ethnicities or genders, resulting in differential accuracy rates. These biases can lead to unfair or discriminatory outcomes in applications that rely on face detection technology.

Many face detection algorithms require significant computational resources to process images or video streams in real-time. This can pose challenges for resource-constrained devices or systems with strict latency requirements.

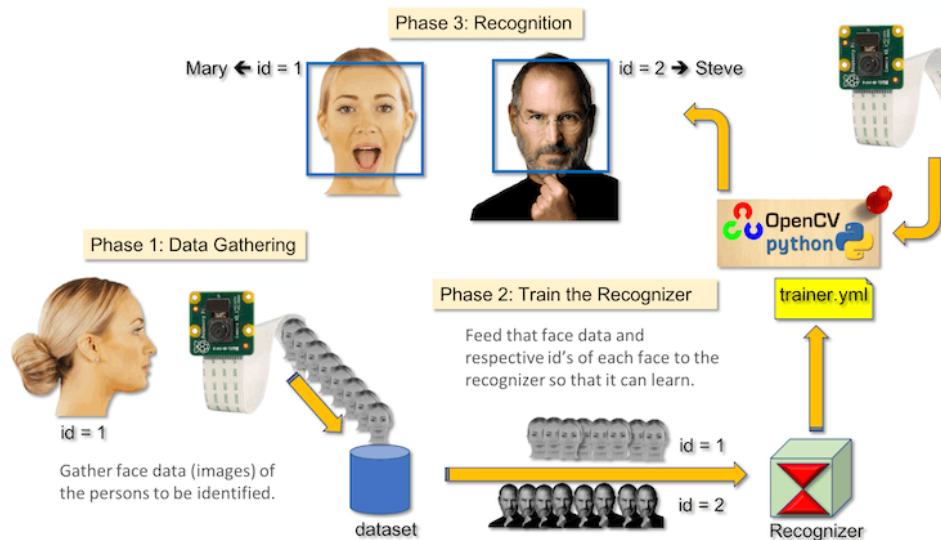
Addressing these challenges requires ongoing research and development in the field of computer vision. Researchers and engineers continuously work on improving algorithms and techniques to enhance the accuracy, robustness, and fairness of face detection systems.

2.2 Proposed solution:

Gather a diverse dataset of images with labeled faces. The dataset should cover various demographics, poses, lighting conditions, and occlusions. Preprocess the images to enhance their quality, normalize lighting conditions, and remove any noise or artifacts that could hinder face detection. Split the dataset into a training set and a validation set. The training set will be used to train the face detection model, while the validation set will help evaluate its performance and tune hyperparameters. Choose an appropriate face detection model based on your project's requirements. Popular choices include Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based models like SSD (Single Shot MultiBox Detector) or Faster R-CNN (Region-based Convolutional Neural Network). Train the selected face detection model using the labeled training set. This typically involves optimizing the model's parameters to minimize the difference between predicted and ground truth face locations. Evaluate the trained model's performance on the validation set. Calculate metrics such as precision, recall, and F1 score to assess its accuracy and robustness. Make necessary adjustments if the results are not satisfactory. Fine-tune the model by adjusting hyperparameters like learning rate, network architecture, and regularization techniques. This process aims to improve the model's performance and generalization ability. Once satisfied with the model's performance, evaluate it on an independent testing dataset. This dataset should not be used during training or validation to ensure an unbiased assessment of the model's effectiveness. Deploy the trained face detection model in your desired application or environment. This could involve integrating it into a software system, mobile app, or any other platform where face detection functionality is required. Continuously monitor the deployed face detection system's performance and collect user feedback. Update the model periodically using new data to account for variations in real-world conditions and improve its accuracy over time.

3 THEORITICAL ANALYSIS:

3.1 Block diagram



3.2.3.2 Hardware / Software designing Hardware and software requirements of the project

Hardware Requirements :

- **Processor:** A fast processor is essential for efficient face detection. A multicore processor, such as an Intel Core i7 or AMD Ryzen processor, would be suitable. For real-time applications or large-scale systems, you might consider more powerful processors like Intel Xeon or AMD EPYC.
- **Memory (RAM):** Sufficient RAM is crucial for storing and processing image data. A minimum of 8 GB of RAM is typically recommended, but for more demanding scenarios or larger datasets, 16 GB or more would be beneficial.
- **Graphics Processing Unit (GPU):** While face detection can be performed on a CPU, utilizing a GPU can significantly accelerate the process, especially when dealing with large amounts of data. NVIDIA GPUs, such as the GeForce GTX or RTX series, are commonly used for accelerating deep learning algorithms.
- **Storage:** Adequate storage is necessary for storing the face detection model, datasets, and any intermediate results. A solid-state drive (SSD) is recommended for fast data access and retrieval.
- **Camera:** A suitable camera is required to capture the images or video for face detection. The choice of camera depends on your specific application requirements, such as resolution, frame rate, and environmental conditions.
- **Optional:** Additional peripherals, such as a microphone or speakers, might be needed if your face detection project involves audio input or output.

Software Requirements :

- **Programming Language:** Choose a programming language that supports computer vision and image processing. Some popular choices are Python, Java, C++, or MATLAB.
- **Integrated Development Environment (IDE):** An IDE will provide you with tools and features for efficient coding. Some widely used IDEs for face detection projects include PyCharm, Visual Studio Code, Eclipse, or MATLAB IDE.
- **OpenCV:** OpenCV (Open Source Computer Vision Library) is a popular open-source library that provides various functions and algorithms for computer vision tasks, including face detection. It supports multiple programming languages and platforms, making it a common choice for face detection projects.
- **Face Detection Library:** Depending on your programming language, you can choose a face detection library that integrates with OpenCV. For Python, you can use libraries like dlib, face_recognition, or OpenCV's built-in Haar cascades. For Java, you can use libraries like JavaCV or OpenCV's Java bindings.
- **Image or Video Input:** Determine how you'll input images or videos for face detection. You may need libraries or tools to handle image or video processing. For example, in Python, you can use libraries like Pillow or OpenCV itself to read and process images or videos.
- **Machine Learning or Deep Learning Frameworks:** If you plan to implement advanced face detection techniques like deep learning-based methods, you'll need a machine learning or deep learning framework. Popular choices include TensorFlow, PyTorch, Keras, or scikit-learn.
- **Additional Libraries:** Depending on your specific requirements, you may need additional libraries for tasks like data manipulation, visualization, or user interface development. Some examples include NumPy, matplotlib, tkinter, or Django.
- **Version Control:** It's recommended to use a version control system like Git to manage your project's source code, track changes, and collaborate with others effectively.
- **Operating System:** Determine the target operating system for your project. Make sure the chosen programming language, libraries, and tools are compatible with the intended platform (e.g., Windows, Linux, macOS).
- **Documentation and Collaboration Tools:** Consider using tools like Jupyter Notebook, Sphinx, or GitHub Wiki for documentation. Collaboration tools like GitHub or Bitbucket can be useful for team collaboration, issue tracking, and code reviews.

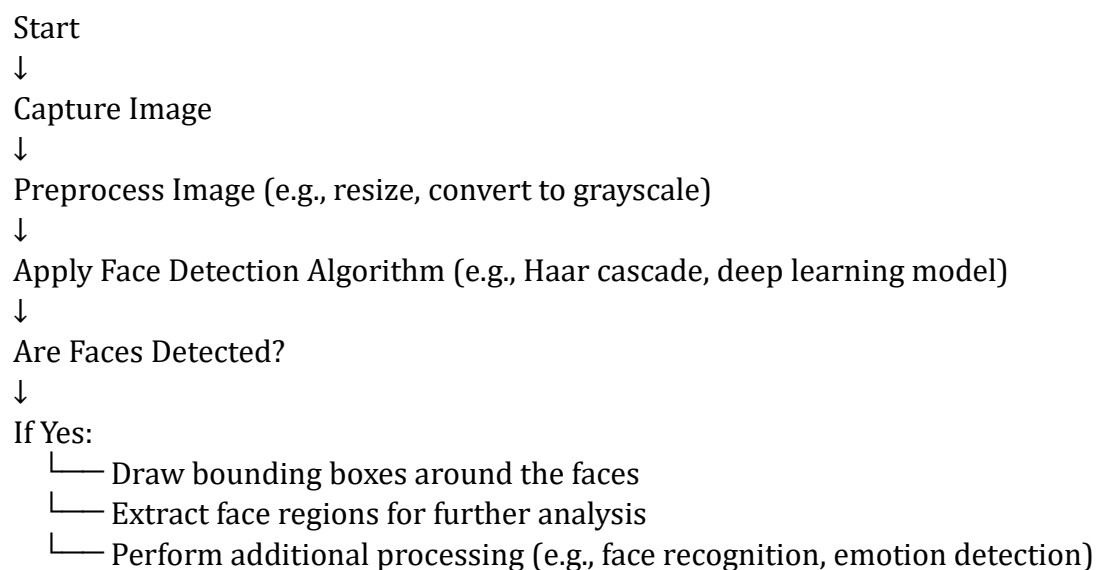
Remember that these requirements may vary depending on your project's specific needs and the technologies you choose to implement.

4 EXPERIMENTAL INVESTIGATIONS:

Choose an appropriate dataset that includes a wide range of face images with various characteristics such as different poses, lighting conditions, occlusions, and backgrounds. Popular face detection datasets include the WIDER Face, FDDB, and Colab datasets.

Perform necessary preprocessing steps on the dataset, such as resizing images to a consistent resolution, normalizing pixel values, and augmenting the data to increase the diversity of training examples. Explore different face detection algorithms and models to find the most suitable one for your project. Popular choices include the Viola-Jones method, Haar cascades, and more advanced deep learning approaches such as Single Shot MultiBox Detector (SSD) or Faster R-CNN. Split your dataset into training and testing subsets. Train your chosen model on the training set, adjusting hyperparameters and optimizing the model architecture if necessary. Evaluate the performance of the model on the testing set, using metrics such as precision, recall, and F1-score to assess the accuracy of face detection. Experiment with techniques to improve the performance of your face detection model. This can include fine-tuning the model, applying transfer learning from pre-trained models, or using data augmentation techniques. Assess the robustness of your face detection system by evaluating its performance under challenging conditions, such as different lighting conditions, occlusions, partial faces, or low-resolution images. This will help identify potential limitations and areas for improvement. Compare the performance of your face detection model with existing state-of-the-art methods to gauge its effectiveness. This can involve benchmarking against established face detection algorithms or participating in face detection challenges like the annual WIDER Face Challenge.

5 FLOWCHART:



- └─ Display or save the processed image
- └─ Continue to next frame or end

If No:

- └─ Display or save the original image without any modifications
- └─ Continue to next frame or end

6 RESULT: VGG16

```
1  import cv2
2  import numpy as np
3  import face_recognition
4  import os
5  from datetime import datetime
6  from flask import Flask, flash, request, redirect, url_for, render_template, Response
7  from werkzeug.utils import secure_filename
8
9  UPLOAD_FOLDER = r'C:\Users\aksha\Downloads\Face-Recognition-Attendance-System-main\Face-Recognition-Attendance-System-main\static\uploads'
10 ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}
11
12
13 usage
14 def allowed_file(filename):
15     return '.' in filename and \
16         filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
17
18 app = Flask(__name__, static_url_path=r'/static')
19 app.secret_key = "secret key"
20 app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
21
22
23 @app.route('/')
24 def upload_file():
25     data = []
26     for i in os.listdir(
27         r'C:\Users\aksha\Downloads\Face-Recognition-Attendance-System-main\Face-Recognition-Attendance-System-main\static\uploads'
28     ):
29         temp = "/IMAGE_FILES/" + i
30         name = i.split(".")[0]
31         data.append([temp, name])
32     print(data)
33     return render_template('upload.html', image_list=data)
34
35 @app.route('/image_upload')
36 def image_upload():
37     return render_template('image-upload.html')
38
39 @app.route('/success', methods=['GET', 'POST'])
40 def success():
41     if 'file' not in request.files:
42         # flash('No file part')
43         return render_template('upload.html')
```

```

44     if file.filename == '':
45         # flash('No image selected for uploading')
46         return render_template('upload.html')
47     if file and allowed_file(file.filename):
48         filename = secure_filename(file.filename)
49         file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
50         # print('upload_image filename: ' + filename)
51         # flash('Image successfully uploaded and displayed below')
52         return render_template('upload.html')
53     else:
54         # flash('Allowed image types are -> png, jpg, jpeg, gif')
55         return render_template('upload.html')
56
57
58 @app.route('/index')
59 def index():
60     """Video streaming home page."""
61
62     return render_template('index.html')
63

```

```

64
65 ~ def gen():
66     IMAGE_FILES = []
67     filename = []
68     dir_path = r'C:\Users\aksha\Downloads\Face-Recognition-Attendance-System-main\Face-Recognition-
69
70 ~     for imagess in os.listdir(dir_path):
71         img_path = os.path.join(dir_path, imagess)
72         img_path = face_recognition.load_image_file(img_path) # reading image and append to list
73         IMAGE_FILES.append(img_path)
74         filename.append(imagess.split(".", 1)[0])
75
76 ~     def encoding_img(IMAGE_FILES):
77         encodeList = []
78 ~         for img in IMAGE_FILES:
79             img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
80             encode = face_recognition.face_encodings(img)[0]
81             encodeList.append(encode)
82         return encodeList
83
84 ~     def takeAttendance(name):
85 ~         with open('attendance.csv', 'r+') as f:
86             mypeople_list = f.readlines()
87             nameList = []

```



```

88         for line in mypeople_list:
89             entry = line.split(',')
90             nameList.append(entry[0])
91         if name not in nameList:
92             now = datetime.now()
93             datestring = now.strftime('%H:%M:%S')
94             f.writelines(f'\n{name},{datestring}')
95
96     encodeListknown = encoding_img(IMAGE_FILES)
97     # print(len('sucesses'))
98
99     cap = cv2.VideoCapture(0)
100
101     while True:
102         success, img = cap.read()
103         imgc = cv2.resize(img, (0, 0), None, 0.25, 0.25)
104         # converting image to RGB from BGR
105         imgc = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
106
107         fasescurrent = face_recognition.face_locations(imgc)
108         encode_fasescurrent = face_recognition.face_encodings(imgc, fasescurrent)
109
110         # faceloc- one by one it grab one face location from fasescurrent
111         # than encodeFace grab encoding from encode_fasescurrent
112         # we want them all in same loop so we are using zip
113         for encodeFace, faceloc in zip(encode_fasescurrent, fasescurrent):
114             matches_face = face_recognition.compare_faces(encodeListknown, encodeFace)
115             face_distance = face_recognition.face_distance(encodeListknown, encodeFace)
116             # print(face_distance)
117             # finding minimum distance index that will return best match
118             matchindex = np.argmin(face_distance)
119
120             if matches_face[matchindex]:
121                 name = filename[matchindex].upper()
122                 # print(name)
123                 y1, x2, y2, x1 = faceloc
124                 # multiply locations by 4 because we above we reduced our webcam input image by 0.25
125                 # y1,x2,y2,x1 = y1*4,x2*4,y2*4,x1*4
126                 cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
127                 cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (255, 0, 0), 2, cv2.FILLED)
128                 cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255))

```

```

129         takeAttendance(name) # taking name for attendance function above
130
131         # cv2.imshow("campane", img)
132         # cv2.waitKey(0)
133         frame = cv2.imencode('.jpg', img)[1].tobytes()
134         yield (b'--frame\r\n' + b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
135         key = cv2.waitKey(20)
136         if key == 27:
137             break
138
139
140 @app.route('/video_feed')
141 def video_feed():
142     """Video streaming route. Put this in the src attribute of an img tag."""
143     return Response(gen(),
144                     mimetype='multipart/x-mixed-replace; boundary=frame')
145
146
147 if __name__ == "__main__":
148     app.run(debug=True)

```

image-upload.html

```

1
2 <!DOCTYPE html>
3 <html>
4 <head>
5     <title>Attendance Management</title>
6     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
7     <style>
8         .centered {
9             text-align: center;
10        }
11        .rounded-circle {
12            border-radius: 50%;
13        }
14        .card.no-border {
15            border: none;
16        }
17        .card-img-top.smaller-img {
18            width: 150px;
19            height: 150px;
20            margin: auto;
21        }
22    </style>
23    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

```

```

24 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
25 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
26 </head>
27 <body>
28
29 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
30   <a class="navbar-brand" href="#">Attendance Management</a>
31   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-co
32     <span class="navbar-toggler-icon"></span>
33   </button>
34   <div class="collapse navbar-collapse" id="navbarNav">
35     <ul class="navbar-nav ml-auto">
36       <li class="nav-item active">
37         <a class="nav-link" href="{ { url_for('index') } }">Take Attendance</a>
38       </li>
39       <li class="nav-item">
40         <a class="nav-link" href="{ { url_for('image_upload') } }">Upload person</a>
41       </li>
42     </ul>
43   </div>
44 </nav>
45
46 <br>
47 <h3 class="centered">Upload new Images here.....</h3><br>
48
49   <form action="/success" method="post" enctype="multipart/form-data">
50
51     <div class="input-group mb-3" >
52
53       <input type="file" name="file" class="form-control" id="inputGroupFile02">
54
55       <div class="col-auto">
56         <button type="submit" value="Upload" class="btn btn-primary mb-3">Submit</button>
57       </div>
58     </div>
59   </form>
60
61 </body>
62 </html>
63

```

upload.html

```
62 <!DOCTYPE html>
63 <html>
64 <head>
65   <title>Attendance Management</title>
66   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
67   <style>
68     .centered {
69       text-align: center;
70     }
71     .rounded-circle {
72       border-radius: 50%;
73     }
74     .card.no-border {
75       border: none;
76     }
77     .card-img-top.smaller-img {
78       width: 150px;
79       height: 150px;
80       margin: auto;
81     }
82   </style>
83   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
84   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
85   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
86 </head>
87 <body>
88
89 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
90   <a class="navbar-brand" href="#">Attendance Management</a>
91   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-co
92     <span class="navbar-toggler-icon"></span>
93 </button>
94   <div class="collapse navbar-collapse" id="navbarNav">
95     <ul class="navbar-nav ml-auto">
96       <li class="nav-item active">
97         <a class="nav-link" href="{ url_for('index') }">Take Attendance</a>
98       </li>
99       <li class="nav-item">
100         <a class="nav-link" href="{ url_for('image_upload') }">Upload person</a>
101       </li>
102     </ul>
103   </div>
104 </nav>
```

```

105 <br>
106 <h3 class="centered">Welcome to the Attendance Management System</h3><br>
107 <div class="container mt-4">
108
109   <div class="row mt-4">
110     {% for image in image_list %}
111       <div class="col-md-3"><div class="card no-border">
100   <style>
101     .centered {
102       text-align: center;
103     }
104     .rounded-circle {
105       border-radius: 50%;
106     }
107     .card.no-border {
108       border: none;
109     }
110     .card-img-top.smaller-img {
111       width: 150px;
112       height: 150px;
113       margin: auto;
114     }
115   </style>
116   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
117   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
118   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

```



```

119 </head>
120 <body>
121
122 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
123   <a class="navbar-brand" href="#">Attendance Management</a>
124   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-co
125     <span class="navbar-toggler-icon"></span>
126   </button>
127   <div class="collapse navbar-collapse" id="navbarNav">
128     <ul class="navbar-nav ml-auto">
129       <li class="nav-item active">
130         <a class="nav-link" href="{ url_for('index') }}">Take Attendance</a>
131       </li>
132       <li class="nav-item">
133         <a class="nav-link" href="{ url_for('image_upload') }}">Upload person</a>
134       </li>
135     </ul>
136   </div>
137 </nav>
138 <br>
139 <h3 class="centered">Attendance Management System</h3><br>
140
141
142
143
144 <br>
145 <div class="mx-auto" style="width: 400px;">
146   
147 </div>
148
149
150
151
152 <br><br>
153 <div class="mx-auto" style="width: 150px;">
154   <a href="http://127.0.0.1:5000/">
155     <button type="button" class="btn btn-success">Upload New image</button>
156   </a>
157 </div>
158 </body>
159 </html>

```

Run(prompt)

```

Run  app x
C:\Users\aksha\AppData\Local\Programs\Python\Python311\python.exe C:\Users\aksha\Downloads\Face-Recognition-Attendance-System-main\Face-Recognit
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 348-244-280
[['/IMAGE_FILES/dilkhas.jpg', 'dilkhas'], ['/IMAGE_FILES/hazil.jpg', 'hazil'], ['/IMAGE_FILES/lalasaa.jpg', 'lalasaa'], ['/IMAGE_FILES/shahansha
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static//IMAGE_FILES/dilkhas.jpg HTTP/1.1" 308 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static//IMAGE_FILES/hazil.jpg HTTP/1.1" 308 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static//IMAGE_FILES/lalasaa.jpg HTTP/1.1" 308 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static/IMAGE_FILES/shahansha.jpg HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static/IMAGE_FILES/dilkhas.jpg HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static/IMAGE_FILES/lalasaa.jpg HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:24] "GET /static/IMAGE_FILES/hazil.jpg HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:36] "GET /index HTTP/1.1" 200 -
127.0.0.1 - - [02/Jul/2023 22:50:55] "GET /video_feed HTTP/1.1" 200 -


```



OUTPUT :


Attendance Management


Take Attendance Upload person

Welcome to the Attendance Management System


dilkhas


hazil



laalasa


shahansha

Attendance Management

Take Attendance Upload person

Attendance Management System

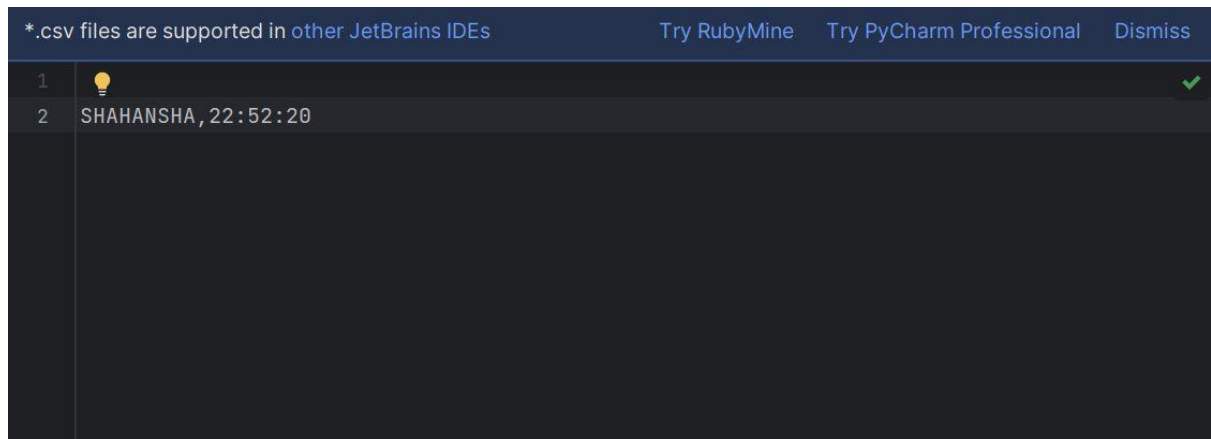


Attendance Management

Take Attendance Upload person

Attendance Management System

Upload New
image



7 ADVANTAGES & DISADVANTAGES:

Advantages

- **Security Through Biometric Authentication:** One of the benefits of facial recognition system centers on its application in biometrics. It can be used as a part of identification and access control systems in organizations, as well as personal devices, such as in the case of smartphones.
- **Automated Image Recognition:** The system can also be used to enable automated image recognition capabilities. Consider Facebook as an example. Through machine learning and Big Data analytics, the social networking site can recognize photos of its users and allow automated linking or tagging to individual user profiles.
- **Deployment in Security Measures:** Similar to biometric application and automated image recognition, another advantage of facial recognition system involves its application in law enforcement and security systems. Automated biometric identity allows less intrusive monitoring and mass identification.
- **Human-Computer Interaction:** The system also supports virtual reality and augmented reality applications. Filters in Snapchat and Instagram use both AR and facial recognition. In both VR and AR applications, the system facilitates further human-computer interaction.
- **Equips Devices with Added Functionalities:** It is also worth noting that equipping devices with facial recognition capabilities means expanding their capabilities. For example, iPhone devices from Apple use Face ID for biometric identification and supporting its AR capabilities.

Disadvantages

- **Issues About Reliability and Efficiency:** A notable disadvantage of facial recognition system is that it is less reliable and efficient than other biometric systems such as fingerprint. Factors such as illumination, expression, and image or video quality, as well as software and hardware capabilities, can affect the performance of the system.
- **Further Reports About It Reliability:** Several reports have pointed out the ineffectiveness of some systems. For example, a report by an advocacy organization noted that the systems used by law enforcement agencies in the U.K. had an accuracy rate of only 2 percent. Applications in London and Tampa, Florida did not result in better law enforcement according to another report.

- **Concerns About Racial Bias:** A study by the American Civil Liberties Union revealed that the Recognition technology developed by Amazon failed nearly 40 percent false matches in tests involving people of colour. In general, the system has been criticized for perpetuating racial bias due to false matches.
- **Issues with Privacy Laws:** Alleged conflict with privacy rights is another disadvantage of facial recognition. In Illinois, for example, its Biometric Information Privacy Act requires affirmative consent for companies to collect biometric data. The fact that the system enables less intrusive mass identification also translates to mass surveillance, which according to groups, is a violation of privacy rights.

8 APPLICATIONS:

- **Gender classification** Gender information can be found from human being image.
- **Document control and access control** Control can be imposed to document access with face identification system.
- **Human computer interaction system** It is design and use of computer technology, focusing particularly on the interfaces between users and computers.
- **Biometric attendance** It is system of taking attendance of people by their finger prints or face etc.
- **Photography** Some recent digital cameras use face detection for autofocus. Face detection is also useful for selecting regions of interest in photo slideshows.
- **Facial feature extraction** Facial features like nose, eyes, mouth, skin-color etc. can be extracted from image.
- **Face recognition** A facial recognition system is a process of identifying or verifying a person from a digital image or a video frame. One of the ways to do this is by comparing selected facial features from the image and a facial database. It is typically used in security systems.
- **Marketing** Face detection is gaining the interest of marketers. A webcam can be integrated into a television and detect any face that walks by. The system then calculates the race, gender, and age range of the face. Once the information is collected, a series of advertisements can be played that is specific towards the detected race/gender/age.

9 CONCLUSION & FUTURE SCOPE:

In recent years face detection has achieved considerable attention from researchers in biometrics, pattern recognition, and computer vision groups. There is countless security, and forensic applications requiring the use of face recognition technologies. As you can see, face detection system is very important in our day to day life. Among the entire sorts of biometric, face detection and recognition system is the most accurate. In this article, we have presented a survey of face detection techniques. It is exciting to see face detection techniques be increasingly used in real-world applications and products. Applications and challenges of face detection also discussed which motivated us to do research in face detection. The most straightforward future direction is to further improve the face detection in presence of some problems like face occlusion and non-uniform illumination. Current research focuses in field of face detection and recognition is the detection of faces in presence of occlusion and non-uniform illumination. A lot of work has been done in face detection, but not in presence of problem of presence of occlusion and non-uniform illumination. If it happens, it will help a lot to face recognition, face expression recognition etc. Currently many companies providing facial biometric in mobile phone for purpose of access. In future it will be used for payments, security, healthcare, advertising, criminal identification etc.

10 BIBILOGRAPHY :

References:

- Big Brother Watch. 2018. *Face Off: The Lawless Growth of Facial Recognition in UK Policing*. London: Big Brother Watch. Available via [PDF](#)
- Krause, M. 2002. "Is Face Recognition Just High-Tech Snake Oil? *Enter Stage Right*. The Independence Institute. ISSN: [1488-1756](#)
- Snow, J. 2018. "Amazon's Face Recognition Falsely Matched 28 Members of Congress With Mugshots." *ACLU*. American Civil Liberties Union. Available [online](#)

APPENDIX A.

SOURCE CODE AND FLASK DEPLOYMENT:

GITHUB LINK:

<https://github.com/AKSHAHANSHA/face-detection>