PES Institute of Technology and Management, Shivamogga

(Affiliated to Visvesvaraya Technological University "Jnana Sangama", Belgaum)

Department of Artificial Intelligence and Machine Learning

Subject: MACHINE LEARNING
[21AI63]

Academic Year: 2023-24

Semester: VI

Section: 'A'

Batch: 2021-2025

Mini project Title: Capture Surveillance System

Submitted By,

Name : Mubarak M

Sagar Kumar

USN : 4PM21AI023

4PM21AI033

Marks : __20

Real World Problem Identified:

In today's world, security and surveillance have become paramount concerns in various settings, including homes, offices, and public spaces. Traditional CCTV systems offer continuous monitoring but often require significant storage and are not always capable of intelligent detection and recording. The need for a more efficient system that can detect and record specific events, such as human faces, is crucial to optimizing storage and enhancing security measures. This project aims to develop a simple, yet effective, surveillance system using OpenCV that detects and saves images of faces captured by a webcam.

Algorithm used and its details:

The primary algorithm used in this project is the Haar Cascade Classifier for face detection. Haar Cascade is an object detection algorithm used to identify objects in images or video streams. The algorithm involves training a cascade function from positive and negative images and is capable of detecting objects in various sizes and positions within an image.

Haar Cascade Classifier

- **Training**: The Haar Cascade Classifier is trained with a large set of positive images (containing the object to be detected) and negative images (without the object).
- **Feature Selection**: The algorithm uses Haar features, which are simple rectangular features that encode the contrast between adjacent image regions.
- **Cascade of Classifiers**: The Haar features are organized into a cascade of classifiers that are applied sequentially to regions of the image. Each stage in the cascade consists of a set of weak learners (simple classifiers) that collectively form a strong classifier.
- **Detection**: During detection, the image is scanned with windows of different sizes to find the object. If a window passes all stages of the cascade, the object is detected.

Solution/ Program Implementation with screenshot of Result:

```
import cv2
from datetime import datetime
import os

# Initialize webcam
cap = cv2.VideoCapture(0)

# Load Haar Cascade classifiers for face and body detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
body_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_fullbody.xml")

# Create directory for detections if it doesn't exist
if not os.path.exists('detections'):
    os.mkdir('detections')

while True:
    # Capture frame-by-frame
    _, frame = cap.read()
    # Convert frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    bodies = body_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, width, height) in faces:
        # Draw rectangle around the face
        cv2.rectangle(frame, (x, y), (x + width, y + height), (255, 0, 0),
3)
        face_roi = frame[y:y+height, x:x+width]

        # Create directory with current date if it doesn't exist
        date_dir = 'detections/' + datetime.now().strftime('%Y-%m-%d')
        if not os.path.exists(date_dir):
            os.mkdir(date_dir)

        # Save the detected face
        cv2.imwrite(date_dir + '/' + datetime.now().strftime("%H-%M-%S") +
'.jpg', face_roi)

    # Display the resulting frame
    cv2.imshow('frame', frame)
```

```
    # Break the loop on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the capture and close windows
cap.release()
cv2.destroyAllWindows()
```

Output :

Captured images of faces will be saved in detections folder for example as these

Future Work:

- **Enhancements**: Implementing more advanced algorithms such as Deep Learning-based detectors (e.g., DNN, YOLO) for improved accuracy and robustness.
- **Additional Features**: Adding motion detection, alert systems, and remote monitoring capabilities to make the system more comprehensive.
- **Optimization**: Reducing false positives and improving detection speed for real-time applications.

Conclusion:

The project successfully demonstrates a simple and effective method for real-time face detection and saving detected faces using OpenCV. The use of Haar Cascade Classifiers allows for efficient and accurate detection of faces in video streams. This system can be expanded to detect and save other objects or events, making it a versatile tool for various surveillance and security applications.