Face Recognition

LIVE VIDEO RECOGNITION

The team

Group Leader

Sanskar Sanjay Sarda
65011680

Rachata Raktham
65011500

Thatchai Sangchant
65011592

Demonstrate Concepts

Process Keys of All Techniques Used

- Image loading and preprocessing.
- Training Support Vector Machine (SVM) model.
- Real-time face detection and recognition.

Demonstrated Examples of Application Usage:

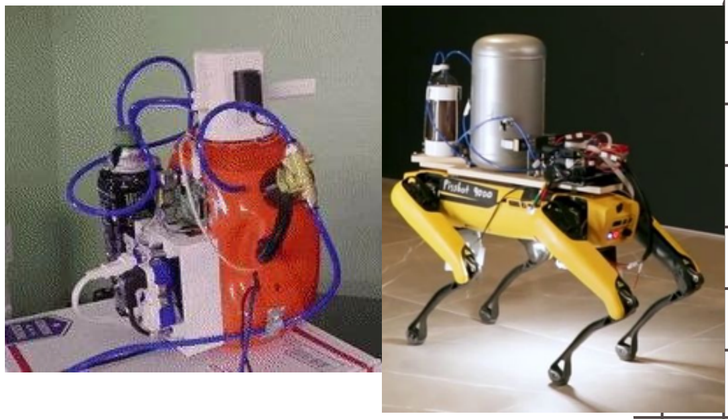
- Security and surveillance.
- Personalization.
- Human-computer interaction.

application of codes (example)

home security

face tracking camera

others



Codes

librarys

CV2

CV2 or OpenCV is library for anything computer vision

OS

library to make our codes work with our computer OS such as accessing camera, microphone and other device both internal and external

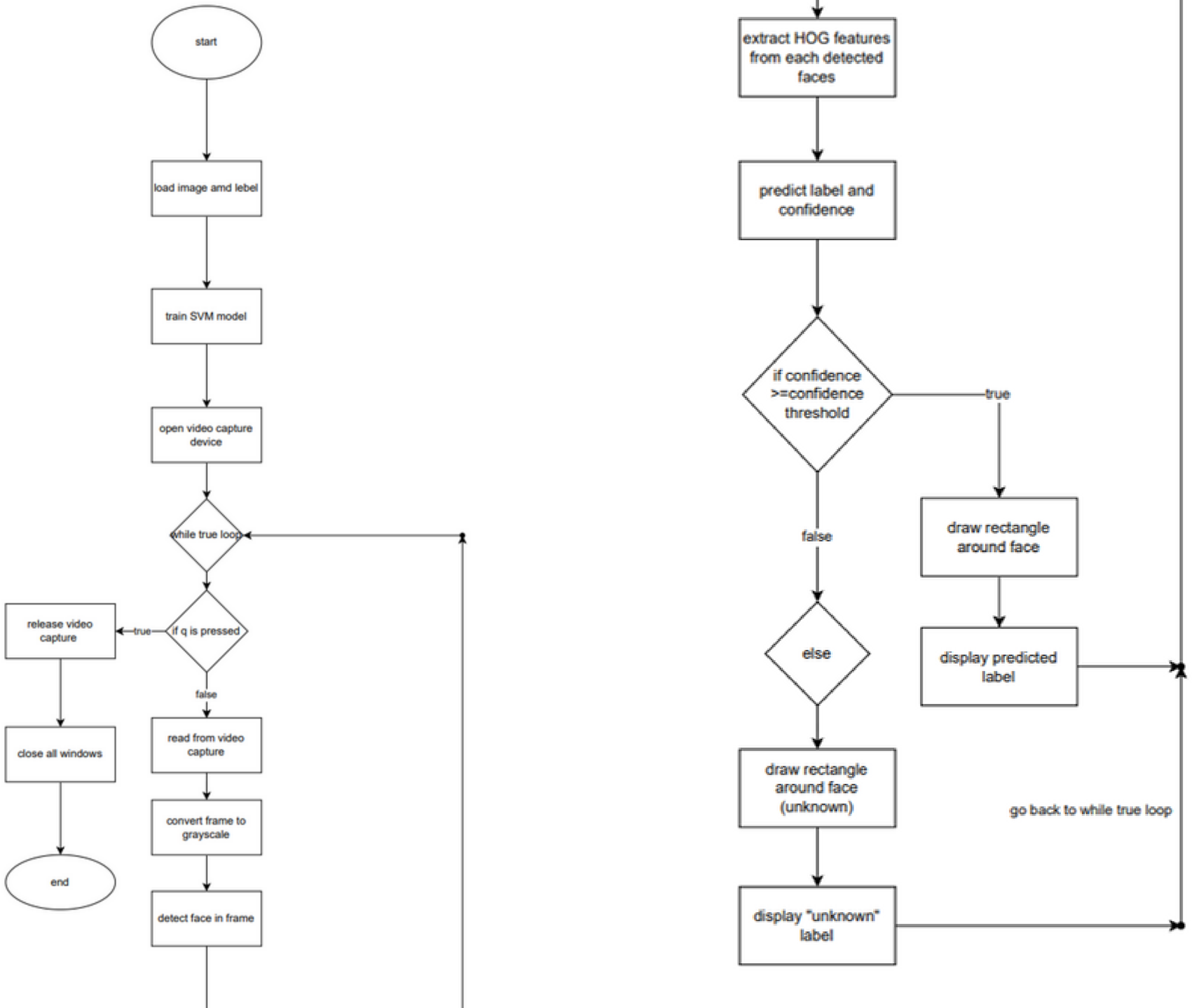
numpy

numpy is library made to calculate math and data with more complexity efficiently while also provide advanced math functions to use

sklearn

sklearn is library for machine learning in python like classification, regression and clustering

Flow chart

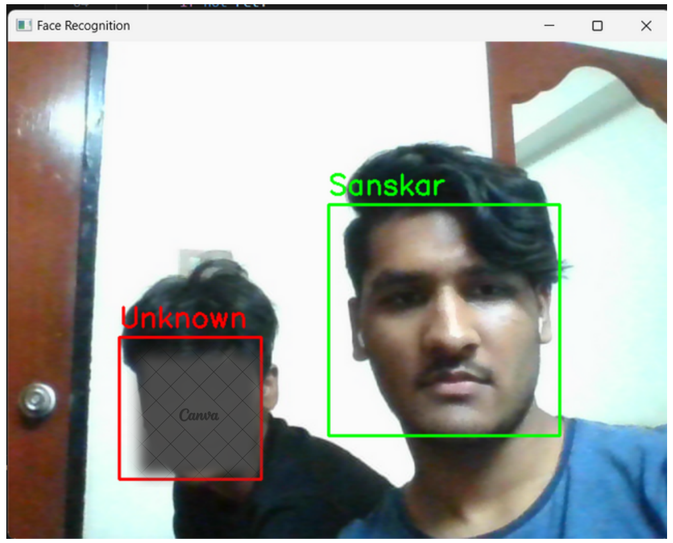

start
load image amd lebel
train SVM model
open video capture device
while true loop
if q is pressed
true
release video capture
close all windows
end
false
read from video capture
convert frame to grayscale
detect face in frame
extract HOG features from each detected faces
predict label and confidence
if confidence >=confidence threshold
true
draw rectangle around face
display predicted label
false
else
draw rectangle around face (unknown)
display "unknown" label
go back to while true loop

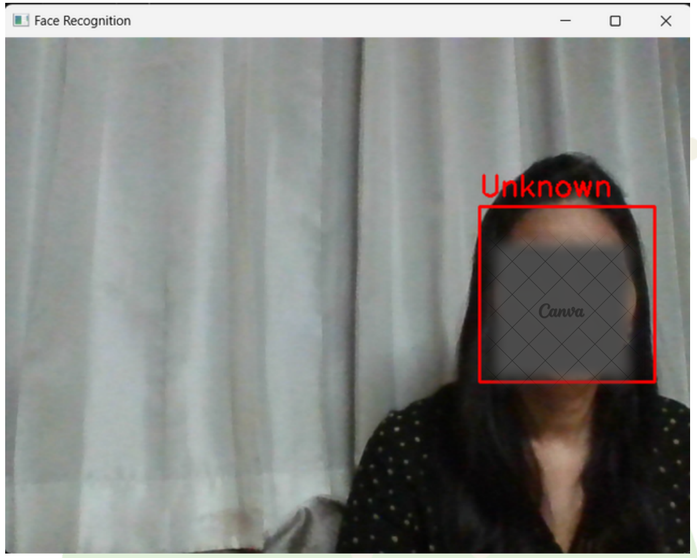
Codes

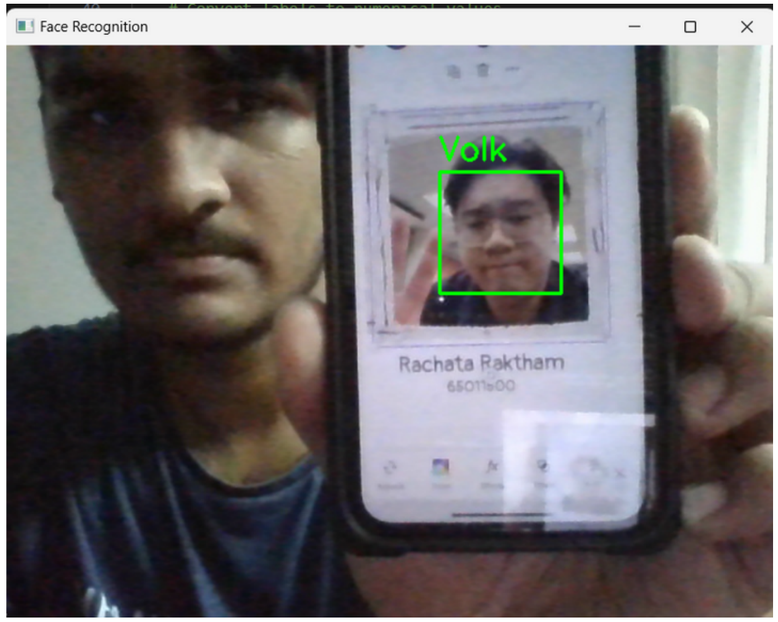
```
import cv2
import os
import numpy as np
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Function to load images and labels from a directory
def load_images_from_folder(folder)
    images = []
    labels = []
    for person_folder in os.listdir(folder)
        person_path = os.path.join(folder, person_folder)
        if os.path.isdir(person_path)
            for filename in os.listdir(person_path)
                img_path = os.path.join(person_path, filename)
                img = cv2.imread(img_path)
                if img is not None
                    images.append(img)
                    labels.append(person_folder)
    return images, labels

# Function to extract HOG features from an image
def extract_hog_features(image)
    # Resize the image to a fixed size (e.g., 64x128)
    resized_image = cv2.resize(image, (64, 128))
    hog = cv2.HOGDescriptor()
    return hog.compute(resized_image).flatten()

# Function to train the SVM model
def train_svm(images, labels)
    X = []
    for img in images
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        hog_features = extract_hog_features(gray)
        X.append(hog_features)
    X = np.array(X)

    # Convert labels to numerical values
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(labels)

    # Split data into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train SVM model
    svm = SVC(kernel='linear', probability=True)
    svm.fit(X_train, y_train)

    # Predict on the test set
    y_pred = svm.predict(X_test)
```

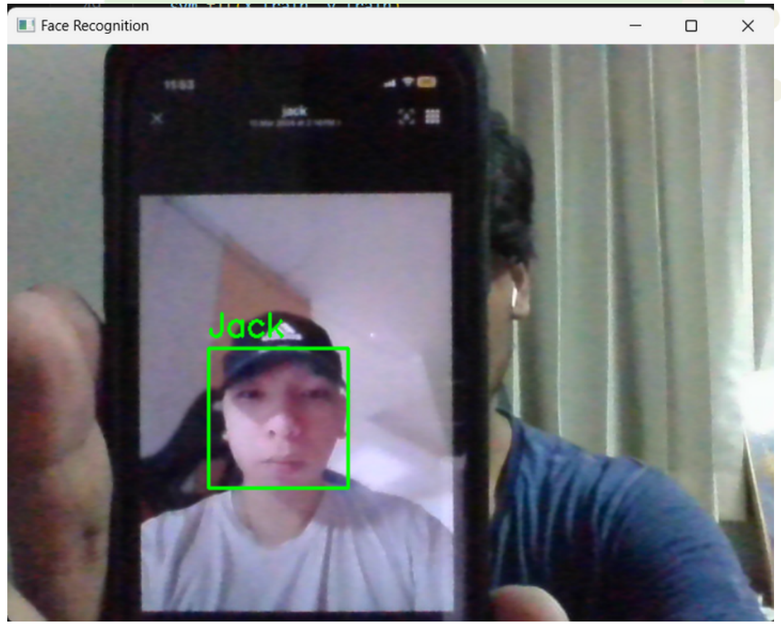
```
    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(Accuracy, accuracy)

    return svm, label_encoder

# Function to detect and recognize faces in live video
def detect_and_recognize_faces(video_capture, svm, label_encoder, confidence_threshold=0.515)
    while True
        ret, frame = video_capture.read()
        if not ret
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Detect faces
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in faces
            face = gray[yy+h, xx+w]
            hog_features = extract_hog_features(face)
            prediction = svm.predict(hog_features.reshape(1, -1))
            confidence = np.max(svm.predict_proba(hog_features.reshape(1, -1)))
            person = label_encoder.inverse_transform(prediction)[0]
            print(Predicted, person, Confidence, confidence)

            # Display the recognized person or Unknown
            if confidence = confidence_threshold
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
                cv2.putText(frame, person, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
            else
                cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
                cv2.putText(frame, Unknown, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

        cv2.imshow('Face Recognition', frame)
        if cv2.waitKey(1) & 0xFF == ord('q')
            break

def main()
    # Step 1 Load images and labels
    folder = r'CUsersSanskarDownloadsMember Photos'  # Path to the folder containing images
    images, labels = load_images_from_folder(folder)

    # Step 2 Train SVM model
    svm, label_encoder = train_svm(images, labels)

    # Step 3 Perform face detection and recognition in live video
    video_capture = cv2.VideoCapture(0)
    detect_and_recognize_faces(video_capture, svm, label_encoder)
    video_capture.release()
    cv2.destroyAllWindows()

if __name__ == __main__
    main()
```

Proof of Concept

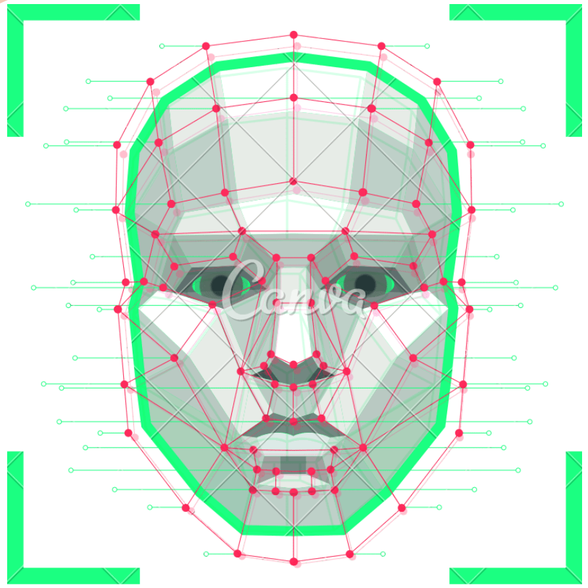








Proposed Improvements



Face Detection

- Integration of Deep Learning Techniques:
  - Exploring the use of Convolutional Neural Networks (CNNs) for feature extraction and classification to potentially enhance recognition accuracy.
- Performance Optimization:
  - Investigating techniques such as parallelization, hardware acceleration, and algorithmic optimizations to improve the overall performance of the face recognition system.
- User Interface Enhancement:
  - Developing a user-friendly graphical interface to improve the interaction between users and the face recognition application, enhancing usability and intuitiveness.
- Data Augmentation and Regularization:
  - Implementing data augmentation and regularization techniques to address overfitting and improve the robustness of the trained model against variations in facial appearance.