VIT-AP UNIVERSITY

COURSE CODE: CSE4047
COURSE NAME : COMPUTER VISION

Name: Syed. Mahammed Sameer
RegNo: 21bce8463
Date: 04-10-2024

Google Drive : Link

PROJECT-2

TITLE-

Extract Human Face Features using SIFT and HOG and calculate similarity between face features of same/different students

The intent is to use this for the Image Attendance System Project. Professor would take a picture of the class for attendance. The system would detect faces, extract features from these faces. These face features would be compared features in Student facial feature database. The system need to works in different conditions

a) Different class rooms have different lighting conditions.
b) Students wear different dresses
c) Students may have moustache, beard, lipstick, mild makeup, hair styles etc.
d) Students Front poses are only considered. 10% angular variation should be supported. (Horizontal / Vertical Tilt)

Steps:

- **Import Required Libraries**:
  - Use OpenCV for image processing, face detection, and SIFT feature extraction.
  - Use `skimage` (scikit-image) for HOG (Histogram of Oriented Gradients) feature extraction.
  - Use NumPy for mathematical operations such as calculating distances between features.
- **Face Detection (Viola-Jones Algorithm)**:
  - The `detect_faces()` function loads the input image and converts it to grayscale.
  - The Haar Cascade classifier detects faces in the grayscale image.
  - Detected faces are marked with bounding boxes on the image.
  - Display the face-detected image using OpenCV, with the window resized to match the image dimensions.
- **SIFT (Scale-Invariant Feature Transform) Feature Extraction**:
  - The `extract_sift_features()` function detects key points and computes SIFT descriptors in a grayscale image.
  - Draw the detected SIFT key points on the image and display them.
  - SIFT descriptors are numerical representations of facial features that can be compared.
- **SIFT Feature Comparison (KNN Matching)**:
  - The `compare_sift_features()` function compares the SIFT descriptors of two images using KNN (k-nearest neighbors) matching.
  - A ratio test is applied to find the best matches, helping to tolerate slight variations in face positions or angles.
- **HOG (Histogram of Oriented Gradients) Feature Extraction**:
  - The `extract_hog_features()` function extracts HOG features from the grayscale image, which represent the face based on the distribution of gradient directions.
  - Display the HOG image using OpenCV, resized to match the original image dimensions.
- **HOG Feature Comparison**:
  - The `compare_hog_features()` function compares two HOG feature sets by calculating the Euclidean distance between them.
  - If the distance is below a certain threshold, the faces are considered similar; otherwise, they are not.
- **Preprocessing (Resize and Normalize)**:
  - Resize both grayscale images to a uniform size (256x256 pixels) using the `resize_image()` function.
  - Normalize the images to ensure pixel values are between 0 and 255, providing consistency for feature extraction.
- **Main Workflow**:
  - Load two input images (e.g., two different images of the same or different individuals).

- Detect faces in both images.
- Resize and normalize the grayscale versions of these images.
- Extract SIFT features from both images and compare them.
- Extract HOG features from both images and compare them.
- Print the number of SIFT matches and the distance between HOG features to determine whether the faces are similar.

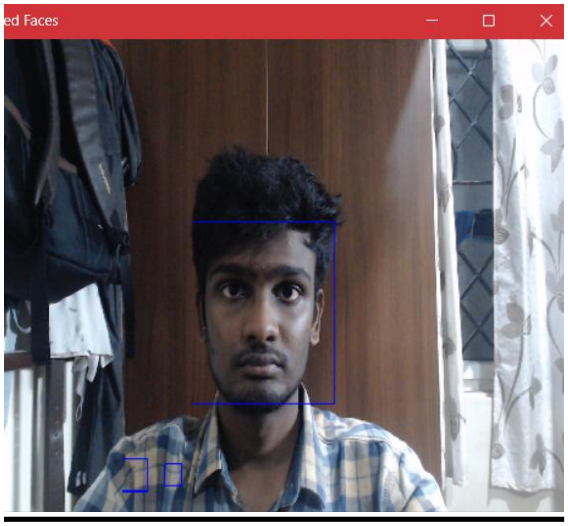
Code:

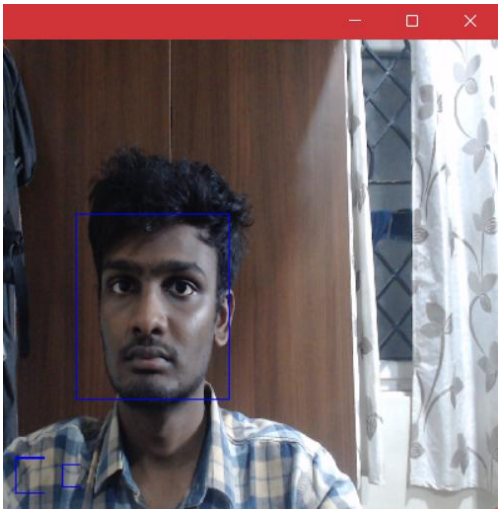
```
import cv2
from skimage.feature import hog
from skimage import exposure
import numpy as np

# Function to detect faces using Viola-Jones algorithm
def detect_faces(image_path):
    img = cv2.imread(image_path)

    if img is None:
        print(f"Error: Could not load image from {image_path}")
        return None, None, None

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Create a named window and resize it to match the gray image size
    cv2.namedWindow('Detected Faces', cv2.WINDOW_NORMAL)
    cv2.resizeWindow('Detected Faces', gray.shape[1], gray.shape[0])
    cv2.imshow('Detected Faces', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    return img, gray, faces

# Function to extract features using SIFT
def extract_sift_features(gray_image):
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(gray_image, None)

    img_sift = cv2.drawKeypoints(gray_image, keypoints, None)

    # Create a named window and resize it to match the gray image size
    cv2.namedWindow('SIFT Keypoints', cv2.WINDOW_NORMAL)
    cv2.resizeWindow('SIFT Keypoints', gray_image.shape[1], gray_image.shape[0])
    cv2.imshow('SIFT Keypoints', img_sift)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    return keypoints, descriptors

# Adjusted SIFT comparison to use KNN matching for better tolerance to slight
variations
def compare_sift_features(descriptors1, descriptors2, ratio_thresh=0.85):
    bf = cv2.BFMatcher(cv2.NORM_L2)
    matches = bf.knnMatch(descriptors1, descriptors2, k=2)
```

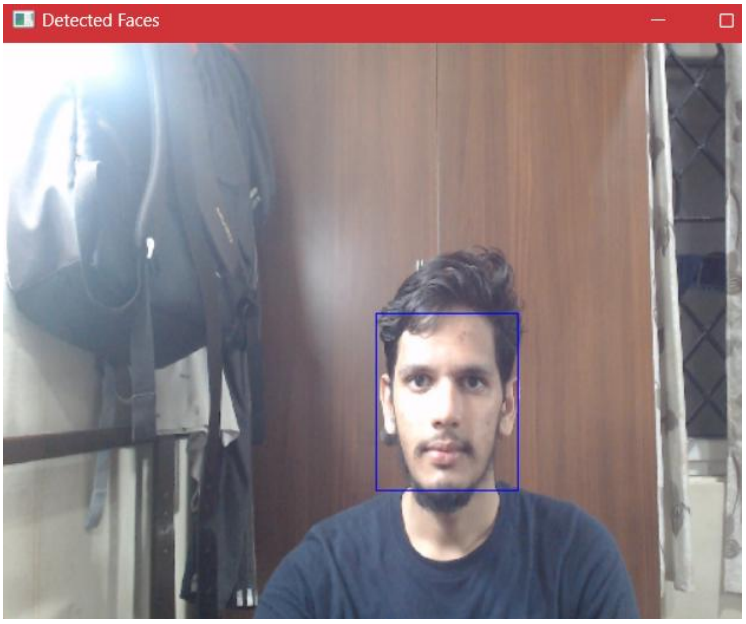
```
    # Apply ratio test
    good_matches = []
    for m, n in matches:
        if m.distance < ratio_thresh * n.distance:
            good_matches.append(m)

    return good_matches

# Function to extract features using HOG
def extract_hog_features(gray_image):
    hog_features, hog_image = hog(gray_image, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualize=True)
    hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

    # Create a named window and resize it to match the gray image size
    cv2.namedWindow('HOG Features', cv2.WINDOW_NORMAL)
    cv2.resizeWindow('HOG Features', gray_image.shape[1], gray_image.shape[0])
    cv2.imshow('HOG Features', hog_image_rescaled)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    return hog_features

# Adjusted HOG feature comparison to allow some tolerance in differences
def compare_hog_features(hog1, hog2, threshold=14):
    if len(hog1) != len(hog2):
        min_length = min(len(hog1), len(hog2))
        hog1, hog2 = hog1[:min_length], hog2[:min_length]

    distance = np.linalg.norm(hog1 - hog2)
    if distance < threshold:
        print("Faces are similar")
    else:
        print("Faces are not similar")

    return distance

# Main code for input image and feature extraction
image_path1 = 'C:/Users/Sameer/Desktop/FallSem 24-25/CV/Lab/images/sameer4.jpg'
image_path2 = 'C:/Users/Sameer/Desktop/FallSem 24-25/CV/Lab/images/koushik.jpg'

def resize_image(gray_image, target_size=(256, 256)):
    return cv2.resize(gray_image, target_size)

def normalize_image(image):
    return cv2.normalize(image, None, 0, 255, cv2.NORM_MINMAX)

img1, gray1, faces1 = detect_faces(image_path1)
if gray1 is None:
    print(f"Error processing the first image: {image_path1}")
    exit()

img2, gray2, faces2 = detect_faces(image_path2)
if gray2 is None:
    print(f"Error processing the second image: {image_path2}")
    exit()

gray1_resized = resize_image(gray1)
gray2_resized = resize_image(gray2)

gray1_normalized = normalize_image(gray1_resized)
gray2_normalized = normalize_image(gray2_resized)

# Extract SIFT features
keypoints1, descriptors1 = extract_sift_features(gray1_normalized)
```

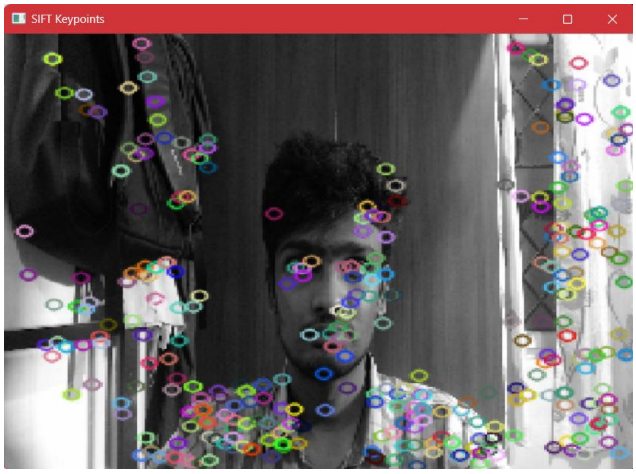
```
keypoints2, descriptors2 = extract_sift_features(gray2_normalized)

# Compare SIFT descriptors using KNN matching
matches = compare_sift_features(descriptors1, descriptors2)
print(f"SIFT Matches found: {len(matches)}")

# Extract HOG features
hog_features1 = extract_hog_features(gray1_normalized)
hog_features2 = extract_hog_features(gray2_normalized)

# Compare HOG features with reduced threshold tolerance
distance = compare_hog_features(hog_features1, hog_features2)
print(f"HOG Feature Distance: {distance}")
```

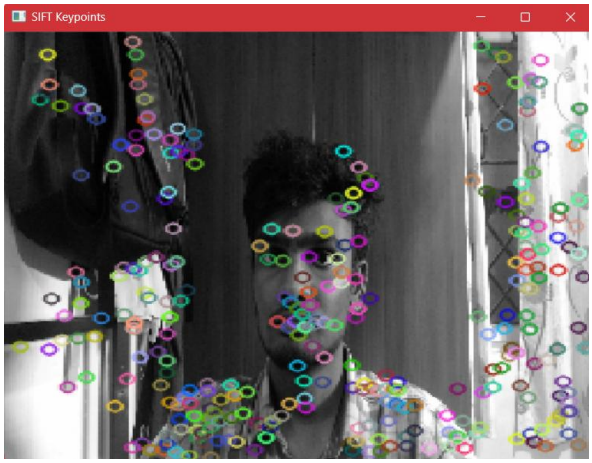
Output:

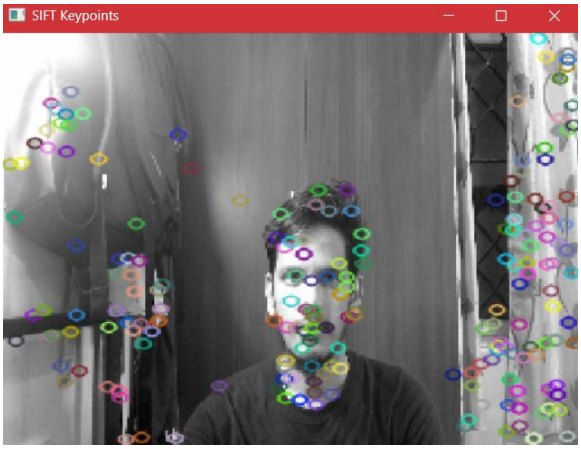


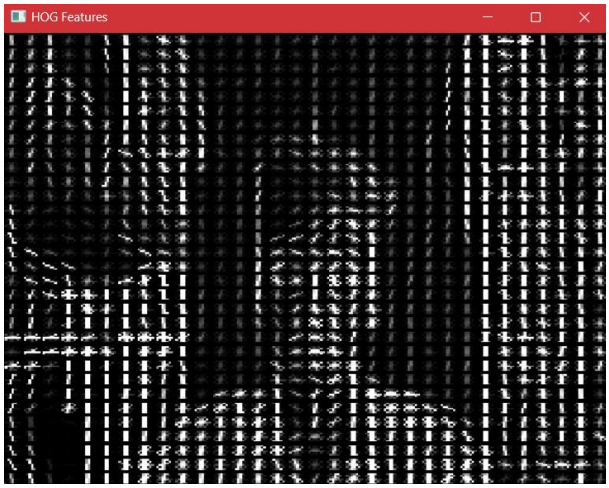


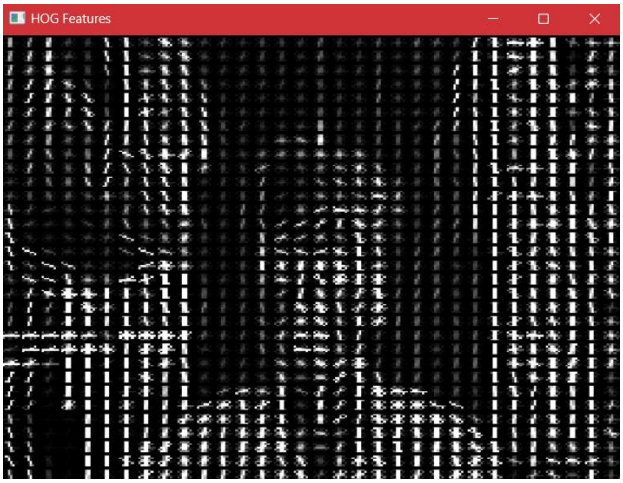


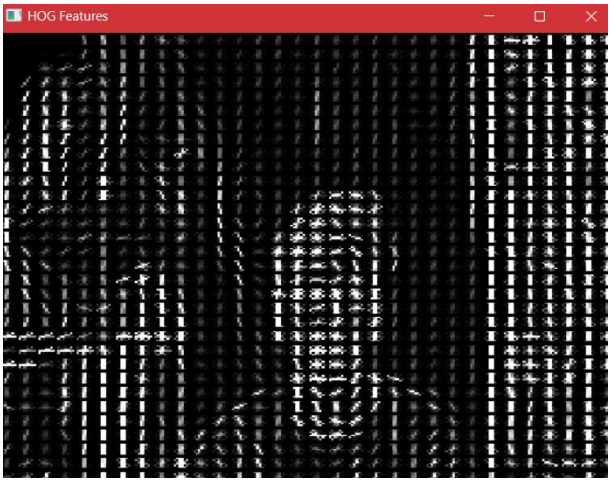

SIFT Keypoints


SIFT Keypoints


SIFT Keypoints


HOG Features


HOG Features


HOG Features

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\Sameer\Desktop\FallSem 24-25\CV\Lab\p2.py
SIFT Matches found: 240
Faces are similar
HOG Feature Distance: 10.470226767003044

========== RESTART: C:\Users\Sameer\Desktop\FallSem 24-25\CV\Lab\p2.py =========
SIFT Matches found: 240
Faces are similar
HOG Feature Distance: 10.470226767003044

========== RESTART: C:\Users\Sameer\Desktop\FallSem 24-25\CV\Lab\p2.py =========
SIFT Matches found: 93
Faces are not similar
HOG Feature Distance: 18.102993141665817
```