PRACTICAL FILE

Student Name	Sahil Kaundal
UID	21BCS8197
Section & Group	20BCS-09 & Group-A
Department	Computer Science & Engineering
Session	Aug-Dec 2023
Course Name	Computer Vision Lab
Course Code	20CSP-422
Semester	7th

INDEX

Ex. No	List of Experiments	Conduct (MM: 12)	Viva (MM: 10)	Record (MM: 8)	Total (MM: 30)	Remarks/Signature
1	Write a program to implement various feature extraction techniques for image classification.					
2	Write a program to assess various feature matching algorithms for object recognition.					
3	Write a program to analyze the impact of refining feature detection for image segmentation.					
4	Write a program to evaluate the efficacy of human-guided control point selection for image alignment.					
5	Write a program to compare the performance of different classification models in image recognition.					
6	Write a program to interpret the effectiveness of Bag of Features in enhancing image classification performance.					
7	Write a program to analyze various object detection algorithms with machine learning.					
8	Write a program to determine the effectiveness of incorporating optical flow analysis into object tracking algorithms.					
9	Write a program to examine the performance of various pretrained deep learning models for real-time object tracking tasks.					
10	Write a program to interpret the effectiveness of template matching techniques for video stabilization tasks.					

Experiment 1.1

Aim: Write a program to implement various feature extraction techniques for image classification.

Software Required: Matlab, Google Colab

Description: Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

There are several feature extraction techniques commonly used in image classification tasks. These techniques aim to capture relevant information from images and transform them into meaningful representations that can be used by machine learning algorithms for classification. Some popular feature extraction techniques are:

Scale-Invariant Feature Transform (SIFT): SIFT is a widely used technique that identifies key points and extracts local invariant descriptors from images. It is robust to changes in scale, rotation, and illumination.

Oriented FAST and Rotated BRIEF(ORB):orb is a feature detection and description algorithm designed for efficiency, often used in real-time applications. It identifies key points using the FAST algorithm, computes rotation-invariant descriptors with binary patterns, and is well-suited for tasks like object tracking and robotics. While ORB is faster, its descriptors may be less distinctive compared to methods like SIFT or SURF.

Histogram of Oriented Gradients (HOG): HOG computes the distribution of gradient orientations in an image. It captures the shape and edge information and has been particularly successful in object detection and pedestrian recognition tasks.

Local Binary Patterns (LBP): LBP encodes the texture information by comparing each pixel's intensity value with its neighboring pixels. It is commonly used in texture analysis tasks and has shown good performance in various image classification applications.

Code:

```
# Step 1: Import necessary libraries
import cv2
from skimage.feature import hog
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np
# Load an example image
image_path = '/content/CompCars-1024x567.jpg'
image = io.imread(image_path)
# Display the original image
plt.figure(figsize=(6, 6))
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
# Convert the image to grayscale
gray_image = color.rgb2gray(image)
# Calculate HOG features
hog_features, hog_image = hog(gray_image, visualize=True, block_norm='L2-Hys')
# Display HOG features
plt.figure(figsize=(6, 6))
plt.imshow(hog_image, cmap=plt.cm.gray)
plt.title('HOG Features')
plt.axis('off')
plt.show()
# Calculate color histogram features
color_hist_features = []
for channel in range(3):
    hist, _ = np.histogram(image[:, :, channel], bins=256, range=(0, 256))
    color_hist_features.extend(hist)
# Display color histogram features
plt.figure(figsize=(10, 6))
plt.bar(range(len(color_hist_features)), color_hist_features)
plt.title('Color Histogram Features')
plt.xlabel('Bin')
plt.ylabel('Frequency')
plt.show()
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
# Load an image using OpenCV
image_path = '/content/CompCars-1024x567.jpg'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
# Create a SIFT object
sift = cv2.SIFT_create()
# Detect and compute SIFT keypoints and descriptors
keypoints, descriptors = sift.detectAndCompute(image, None)
```

```
# Visualize keypoints on the image
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)
# Display the image with keypoints
cv2_imshow(image_with_keypoints)
import cv2
# Load an image using OpenCV
image_path = '/content/CompCars-1024x567.jpg'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
# Create an ORB object
orb = cv2.ORB_create()
# Detect and compute ORB keypoints and descriptors
keypoints, descriptors = orb.detectAndCompute(image, None)
# Visualize keypoints on the image
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
# Display the image with keypoints using cv2_imshow for Google Colab
# For other environments, you can use cv2.imshow
try:
    from google.colab.patches import cv2_imshow
    cv2_imshow(image_with_keypoints)
except ImportError:
    cv2.imshow('Image with Keypoints', image_with_keypoints)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
import cv2
import numpy as np
from skimage.feature import local_binary_pattern
from skimage import io
import matplotlib.pyplot as plt
# Load an example image
image_path = '/content/CompCars-1024x567.jpg'
image = io.imread(image_path, as_gray=True)
# Define LBP parameters
radius = 1
n_points = 8 * radius
# Compute LBP image
lbp_image = local_binary_pattern(image, n_points, radius, method='uniform')
# Calculate a histogram of the LBP image
hist, _ = np.histogram(lbp_image.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
hist = hist.astype("float")
hist /= (hist.sum() + 1e-8)
# Display original image and LBP image
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap="gray")
plt.title("Original Image")
plt.subplot(1, 2, 2)
plt.imshow(lbp_image, cmap="gray")
plt.title("LBP Image")
plt.tight_layout()
plt.show()
```

Implementation:

CV Feature Extration 1.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

Extract features from the images using HOG(Histogram of Oriented Gradients) technique:-

```
# Step 1: Import necessary libraries

import cv2
from skimage.feature import hog
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np
```

```
# Load an example image
image_path = '/content/CompCars-1024x567.jpg'
image = io.imread(image_path)

# Display the original image
plt.figure(figsize=(6, 6))
plt.imshow(image)
plt.title('Original Image')
plt.axis('off')
plt.show()
```

Original Image

```
# Convert the image to grayscale
gray_image = color.rgb2gray(image)
```

```
# Calculate HOG features
hog_features, hog_image = hog(gray_image, visualize=True, block_norm='L2-Hys')
```

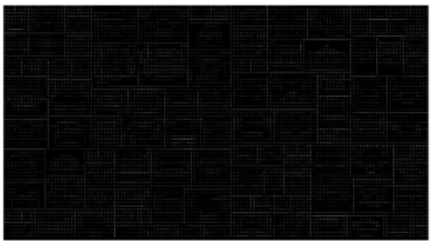
CV Feature Extration 1.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
# Display HOG features
plt.figure(figsize=(6, 6))
plt.imshow(hog_image, cmap=plt.cm.gray)
plt.title('HOG Features')
plt.axis('off')
plt.show()
```

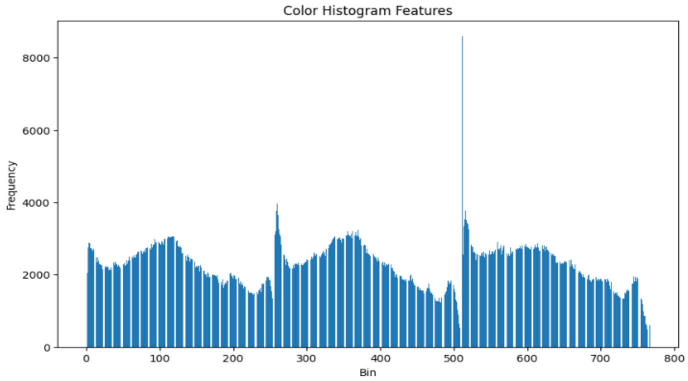
HOG Features



```
# Calculate color histogram features
color_hist_features = []
for channel in range(3):
    hist, _ = np.histogram(image[:, :, channel], bins=256, range=(0, 256))
    color_hist_features.extend(hist)
```

```
# Display color histogram features
plt.figure(figsize=(10, 6))
plt.bar(range(len(color_hist_features)), color_hist_features)
plt.title('Color Histogram Features')
plt.xlabel('Bin')
plt.ylabel('Frequency')
plt.show()
```

CV Feature Extration 1.ipynb



Extract features from the images using Scale-Invariant Feature Transform (SIFT) technique:-

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Load an image using OpenCV
image_path = '/content/CompCars-1024x567.jpg'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
```

```
# Create a SIFT object
sift = cv2.SIFT_create()

# Detect and compute SIFT keypoints and descriptors
```

CV Feature Extration 1.ipynb

```
# Create a SIFT object
sift = cv2.SIFT_create()

# Detect and compute SIFT keypoints and descriptors
keypoints, descriptors = sift.detectAndCompute(image, None)

# Visualize keypoints on the image
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None)

# Display the image with keypoints
cv2_imshow(image_with_keypoints)
```

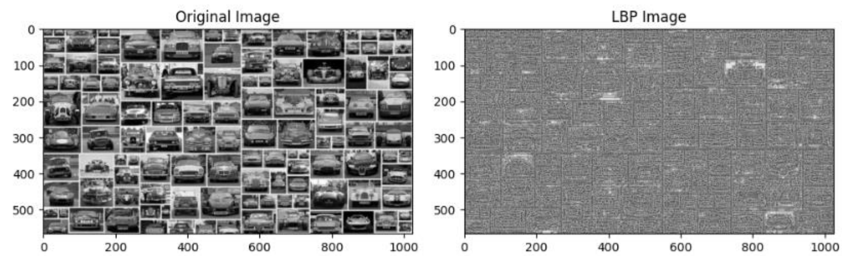
Extract features from the images using Speeded-Up Robust Features (SURF) technique:-

CV Feature Extration 1.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive

```
# Detect and compute ORB keypoints and descriptors
keypoints, descriptors = orb.detectAndCompute(image, None)


# Visualize keypoints on the image
image_with_keypoints = cv2.drawKeypoints(image, keypoints, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)


# Display the image with keypoints using cv2_imshow for Google Colab
# For other environments, you can use cv2.imshow
try:
    from google.colab.patches import cv2_imshow
    cv2_imshow(image_with_keypoints)
except ImportError:
    cv2.imshow('Image with Keypoints', image_with_keypoints)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

CV Feature Extration 1.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive

```
# Define LBP parameters
radius = 1
n_points = 8 * radius


# Compute LBP image
lbp_image = local_binary_pattern(image, n_points, radius, method='uniform')


# Calculate a histogram of the LBP image
hist, _ = np.histogram(lbp_image.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
hist = hist.astype("float")
hist /= (hist.sum() + 1e-8)


# Display original image and LBP image
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap="gray")
plt.title("Original Image")

plt.subplot(1, 2, 2)
plt.imshow(lbp_image, cmap="gray")
plt.title("LBP Image")

plt.tight_layout()
plt.show()
```



Extract features from the images using Convolutional Neural Networks (CNN) technique:-