American International University-Bangladesh (AIUB)

Department of Computer Science
Faculty of Science & Technology (FST)

Machine Learning

FEATURE EXTRACTION TECHNIQUES FROM RAW IMAGE

Submitted by

Semester: Summer_23_24		**Section: C** **Group: 05**
SN	Student Name	Student ID
1	Noshin Farzana	21-44647-1
2	Avijit Saha Anto	21-44630-1
3	Shakil Khan	21-45118-2

Submitted To

Prof. Dr. Md. Asraf Ali
American International University-Bangladesh

Submission Date: 25 July 2024

FEATURE EXTRACTION TECHNIQUES FROM RAW IMAGE

Feature Extraction:

The process of turning pixel data into a more condensed and meaningful representation, which can be used for a variety of tasks like object detection, image segmentation, and image classification, is known as feature extraction from raw images. This procedure includes locating and summarizing significant patterns, shapes, textures, or other pertinent details found in an image.

Feature Extraction Techniques:

- **Edge Detection:**

  Edge detection is a technique for identifying the boundaries of an image. It highlights locations with a large variation in intensity, such as object edges.

  How It Works: Common approaches include the Canny Edge Detector. It computes the gradient of image intensity and identify points where the gradient magnitude exceeds a predetermined threshold.

  Application: Object detection, image segmentation, and computer vision applications that require boundary information.

- **HOG (Histogram of Oriented Gradients):**

  HOG is a feature descriptor used in computer vision and image processing for the purpose of object detection. It captures the gradient structure of local patches in an image.

  How it works: HOG involves computing the gradient of the image intensity and creating a histogram of gradient directions within local regions (cells) of the image. These histograms are then normalized over larger spatial regions (blocks) to improve robustness to changes in illumination and contrast.

  Application: Object detection, face recognition.

- **SIFT (Scale-Invariant Feature Transform):**

  SIFT is a feature extraction method used for detecting and describing local features in images.

  How it works: SIFT detects key points in the image that are invariant to scale and rotation. It then computes a descriptor for each key point based on the local gradient information around the key point.

  Application: Image matching, object recognition.

- **ORB (Oriented FAST and Rotated BRIEF):**

  ORB is a fast and efficient alternative to SIFT.

  How it works: ORB combines the FAST key point detector and the BRIEF descriptor, with enhancements to make it invariant to rotation and scale. It is designed to be computationally efficient and suitable for real-time applications.

  Application: Object recognition, image matching, real-time feature detection in embedded systems.

- **LBP (Local Binary Pattern):**

  Local Binary Pattern is a texture descriptor used in feature extraction from images. It’s useful for capturing local texture information by comparing each pixel with its neighbors.

  How it works: LBP assigns a binary value to each pixel based on its neighborhood. For a given pixel, it compares it to its neighbors (typically a 3x3 grid) and assigns a binary code based on whether each neighbor's value is greater than or equal to the center pixel's value. The binary code is created by reading the result of these comparisons in a specific order (usually clockwise), and then converting this binary code to a decimal number.

  Application: Image processing for texture analysis and recognition.

About the Dataset:

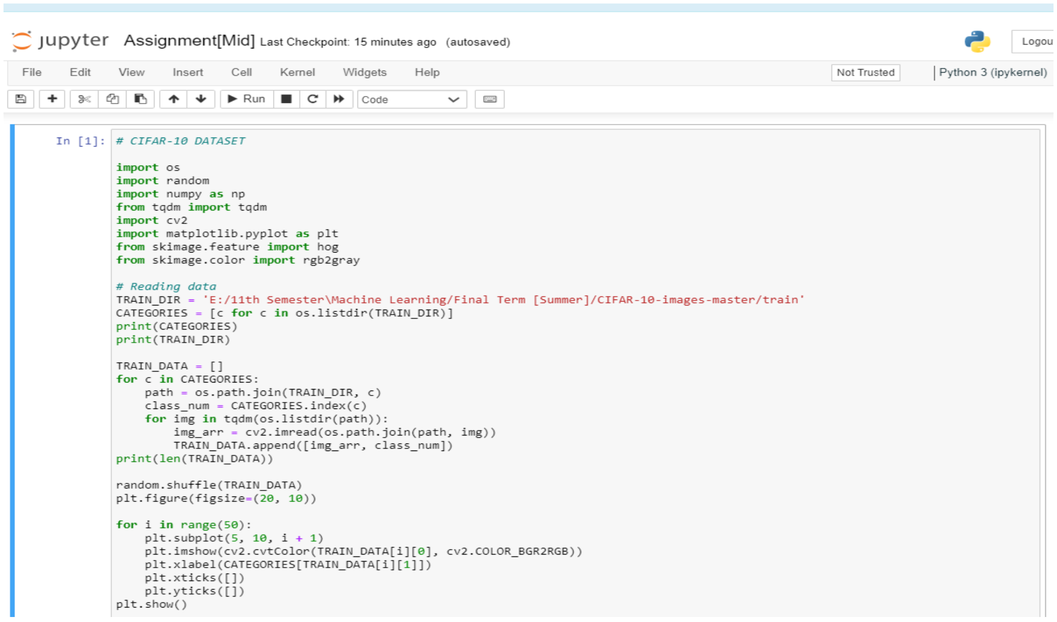
In this assignment, CIFAR-10 dataset was used for feature extraction from raw images, which consists of 60,000 32x32x3 color images in 10 different classes. The CIFAR-10 dataset was downloaded from the following link: https://github.com/YoongiKim/CIFAR-10-images. The top 5 images (truck, automobile, ship, cat, bird) were used for applying feature extraction techniques.

Also, feature extraction techniques are applied to some raw images (cat, flower, car) downloaded from google.

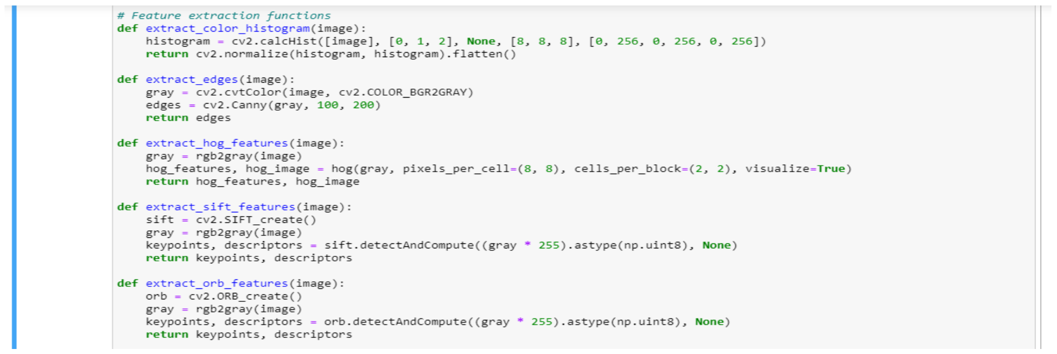
Result:

CIFAR-10 DATASET

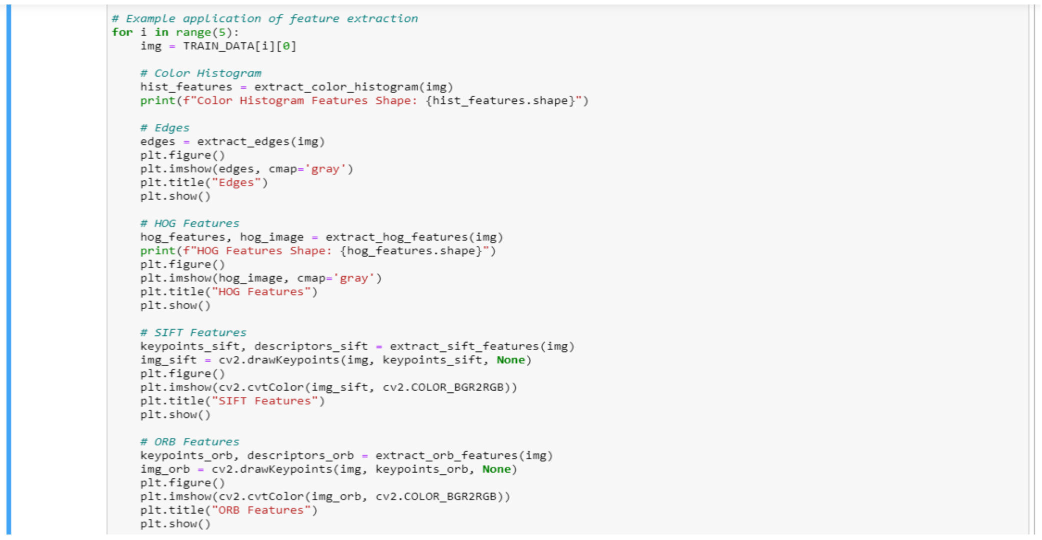
Code:


jupyter Assignment[Mid] Last Checkpoint: 15 minutes ago (autosaved)
Logou
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [1]:

```python
# CIFAR-10 DATASET

import os
import random
import numpy as np
from tqdm import tqdm
import cv2
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage.color import rgb2gray

# Reading data
TRAIN_DIR = 'E:/11th Semester\Machine Learning/Final Term [Summer]/CIFAR-10-images-master/train'
CATEGORIES = [c for c in os.listdir(TRAIN_DIR)]
print(CATEGORIES)
print(TRAIN_DIR)

TRAIN_DATA = []
for c in CATEGORIES:
    path = os.path.join(TRAIN_DIR, c)
    class_num = CATEGORIES.index(c)
    for img in tqdm(os.listdir(path)):
        img_arr = cv2.imread(os.path.join(path, img))
        TRAIN_DATA.append([img_arr, class_num])
print(len(TRAIN_DATA))

random.shuffle(TRAIN_DATA)
plt.figure(figsize=(20, 10))

for i in range(50):
    plt.subplot(5, 10, i + 1)
    plt.imshow(cv2.cvtColor(TRAIN_DATA[i][0], cv2.COLOR_BGR2RGB))
    plt.xlabel(CATEGORIES[TRAIN_DATA[i][1]])
    plt.xticks([])
    plt.yticks([])
plt.show()
```

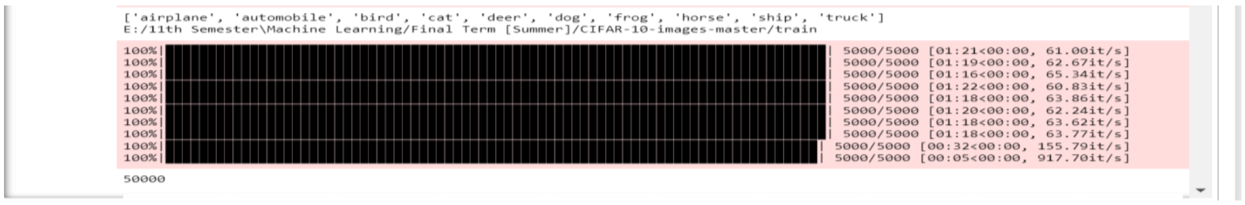


```python
# Feature extraction functions
def extract_color_histogram(image):
    histogram = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])
    return cv2.normalize(histogram, histogram).flatten()

def extract_edges(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 100, 200)
    return edges

def extract_hog_features(image):
    gray = rgb2gray(image)
    hog_features, hog_image = hog(gray, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True)
    return hog_features, hog_image

def extract_sift_features(image):
    sift = cv2.SIFT_create()
    gray = rgb2gray(image)
    keypoints, descriptors = sift.detectAndCompute((gray * 255).astype(np.uint8), None)
    return keypoints, descriptors

def extract_orb_features(image):
    orb = cv2.ORB_create()
    gray = rgb2gray(image)
    keypoints, descriptors = orb.detectAndCompute((gray * 255).astype(np.uint8), None)
    return keypoints, descriptors
```

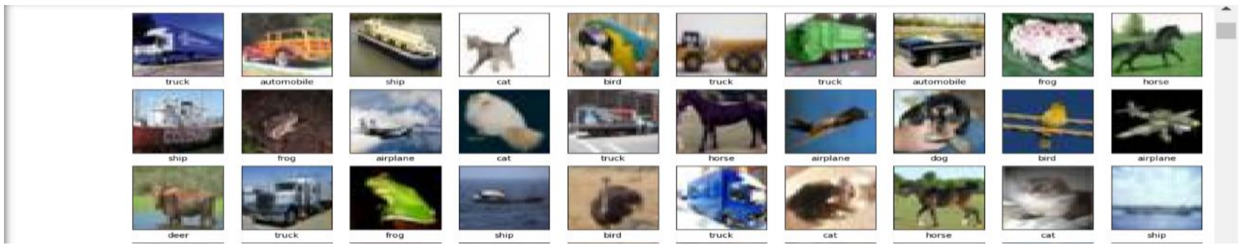


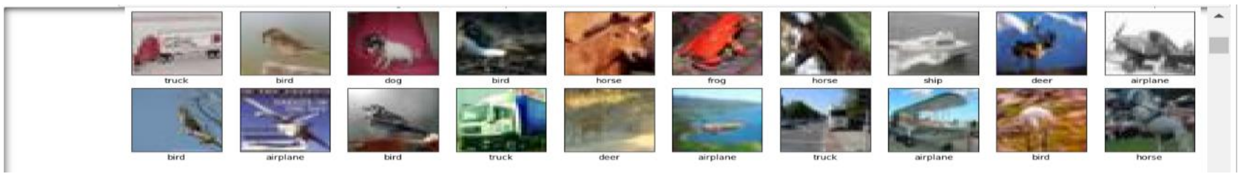
```python
# Example application of feature extraction
for i in range(5):
    img = TRAIN_DATA[i][0]

    # Color Histogram
    hist_features = extract_color_histogram(img)
    print(f"Color Histogram Features Shape: {hist_features.shape}")

    # Edges
    edges = extract_edges(img)
    plt.figure()
    plt.imshow(edges, cmap='gray')
    plt.title("Edges")
    plt.show()

    # HOG Features
    hog_features, hog_image = extract_hog_features(img)
    print(f"HOG Features Shape: {hog_features.shape}")
    plt.figure()
    plt.imshow(hog_image, cmap='gray')
    plt.title("HOG Features")
    plt.show()

    # SIFT Features
    keypoints_sift, descriptors_sift = extract_sift_features(img)
    img_sift = cv2.drawKeypoints(img, keypoints_sift, None)
    plt.figure()
    plt.imshow(cv2.cvtColor(img_sift, cv2.COLOR_BGR2RGB))
    plt.title("SIFT Features")
    plt.show()

    # ORB Features
    keypoints_orb, descriptors_orb = extract_orb_features(img)
    img_orb = cv2.drawKeypoints(img, keypoints_orb, None)
    plt.figure()
    plt.imshow(cv2.cvtColor(img_orb, cv2.COLOR_BGR2RGB))
    plt.title("ORB Features")
    plt.show()
```

Output:

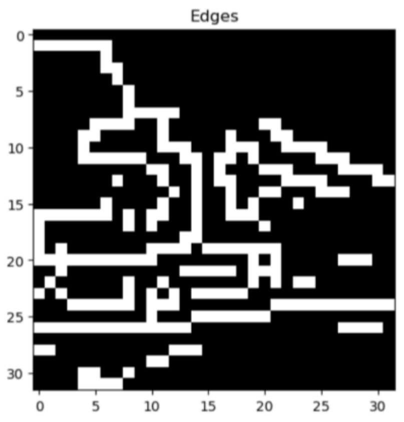


```
['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
E:/11th Semester\Machine Learning/Final Term [Summer]/CIFAR-10-images-master/train
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:21<00:00, 61.00it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:19<00:00, 62.67it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:16<00:00, 65.34it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:22<00:00, 60.83it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:18<00:00, 63.86it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:20<00:00, 62.24it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:18<00:00, 63.62it/s]
100%|██████████████████████████████████████████████████████████| 5000/5000 [01:18<00:00, 63.77it/s]
100%|█████████████████████████████████████████████████████████| 5000/5000 [00:32<00:00, 155.79it/s]
100%|█████████████████████████████████████████████████████████| 5000/5000 [00:05<00:00, 917.70it/s]
50000
```

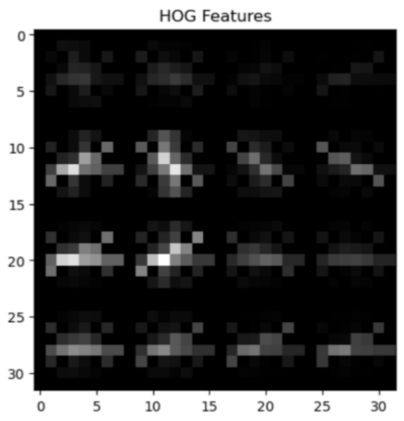


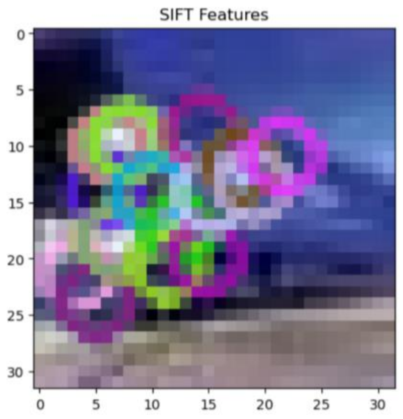


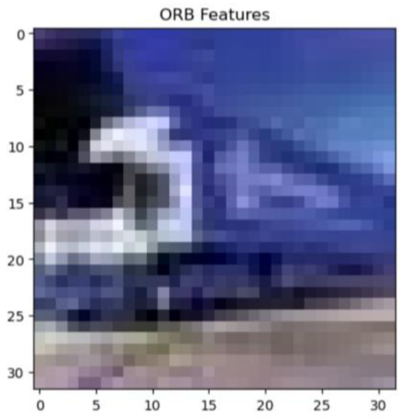
```
Color Histogram Features Shape: (512,)
```



```
HOG Features Shape: (324,)
```

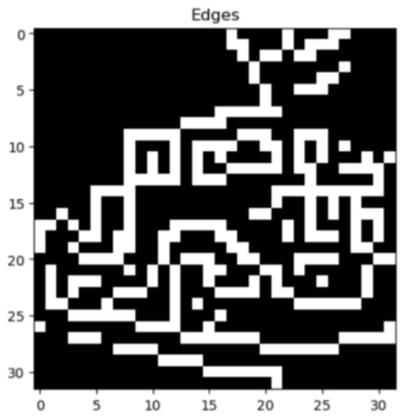




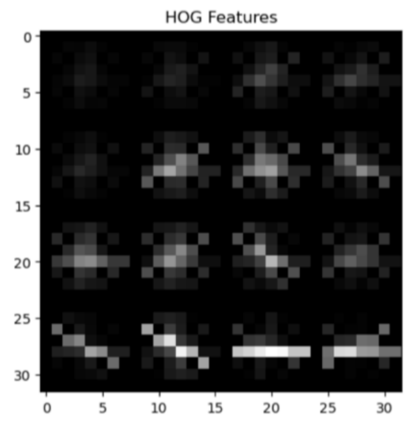


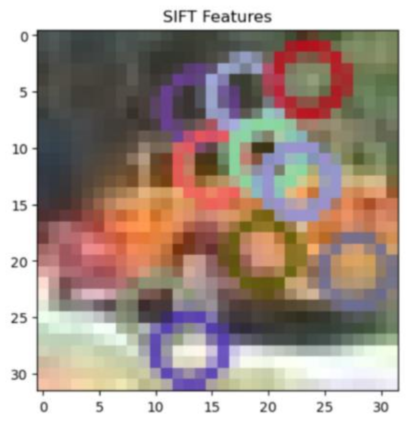
Features of truck can be seen from the output.

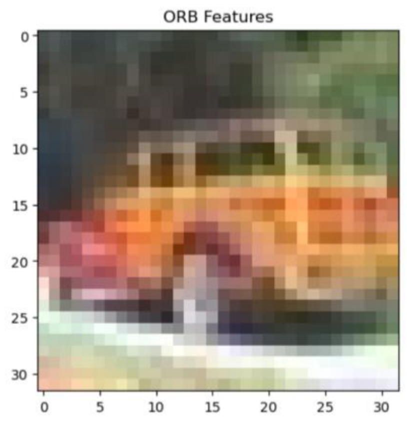
```
Color Histogram Features Shape: (512,)
```



```
HOG Features Shape: (324,)
```

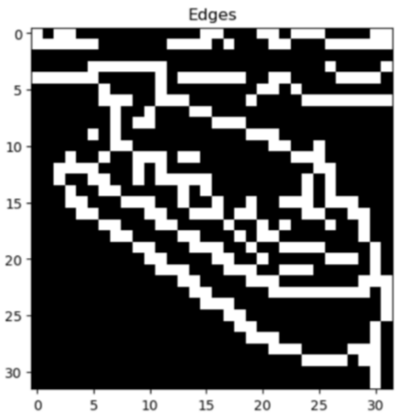




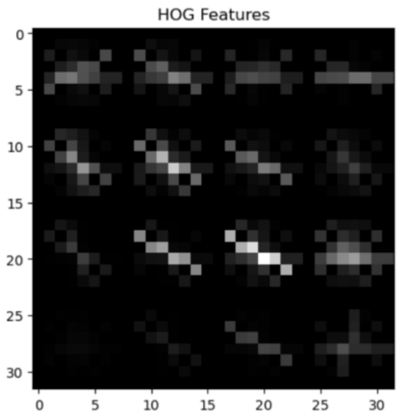


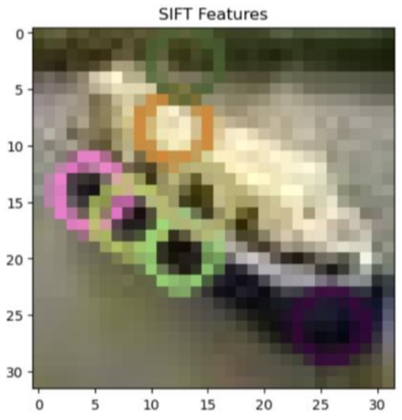
Features of automobile can be seen from the output.

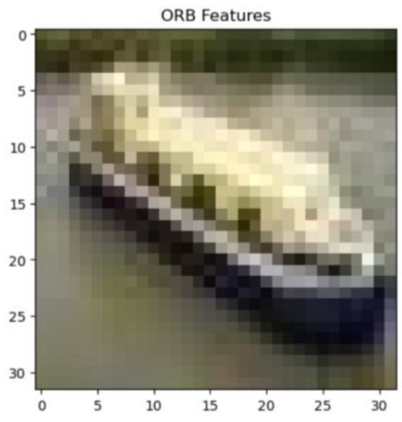
```
Color Histogram Features Shape: (512,)
```



```
HOG Features Shape: (324,)
```

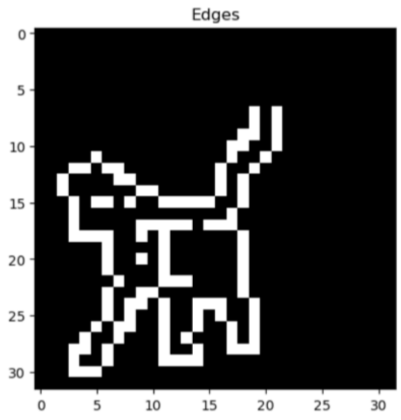




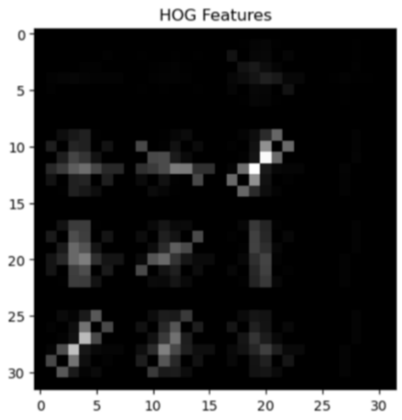


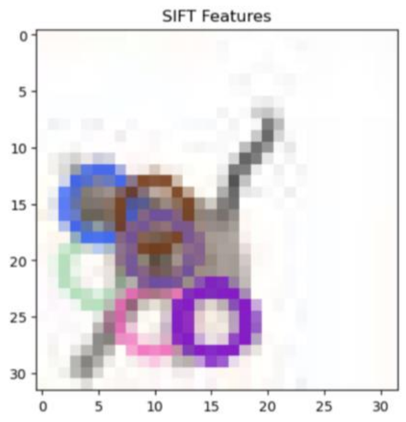
Features of ship can be seen from the output.

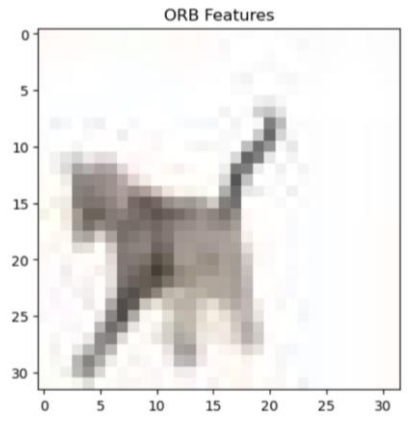
```
Color Histogram Features Shape: (512,)
```



```
HOG Features Shape: (324,)
```

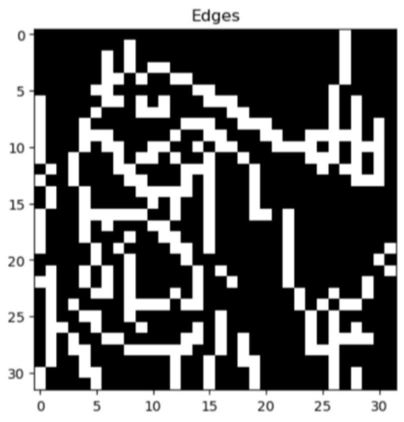




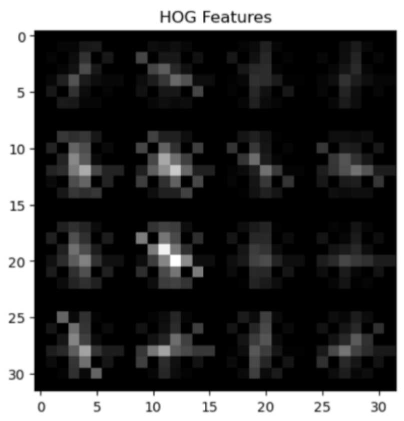


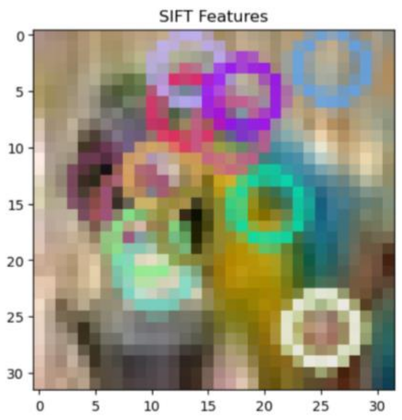
Features of cat can be seen from the output.

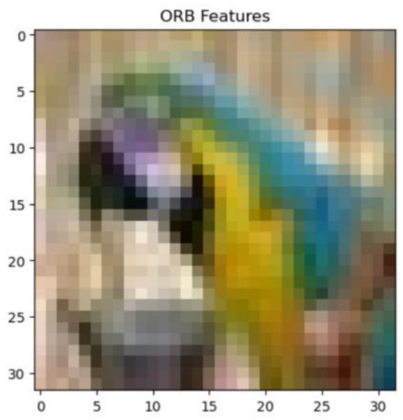
```
Color Histogram Features Shape: (512,)
```



```
HOG Features Shape: (324,)
```



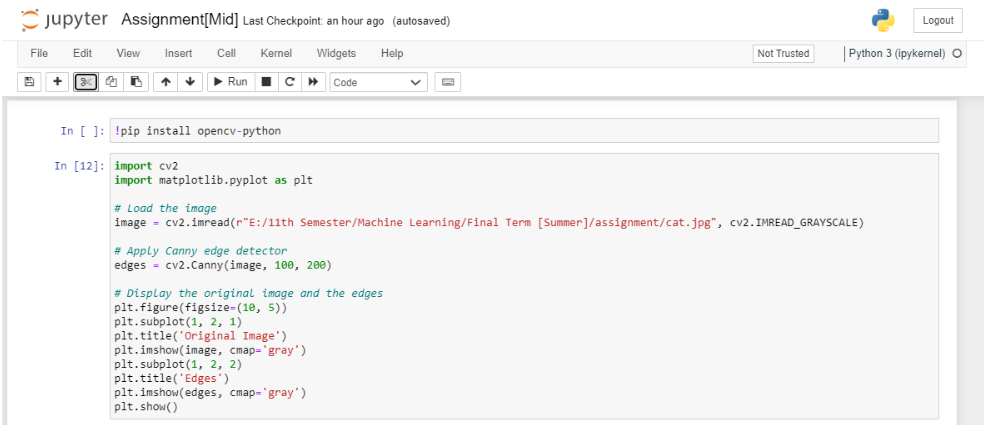




Features of bird can be seen from the output.

RAW IMAGES FROM GOOGLE

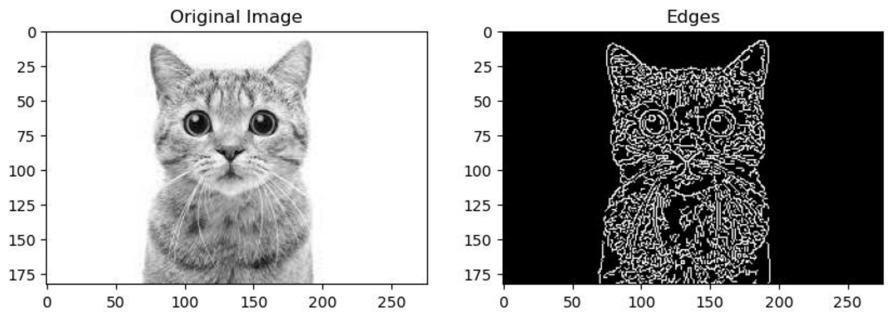
Code (Canny Edge Detector):


Jupyter Assignment[Mid] Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

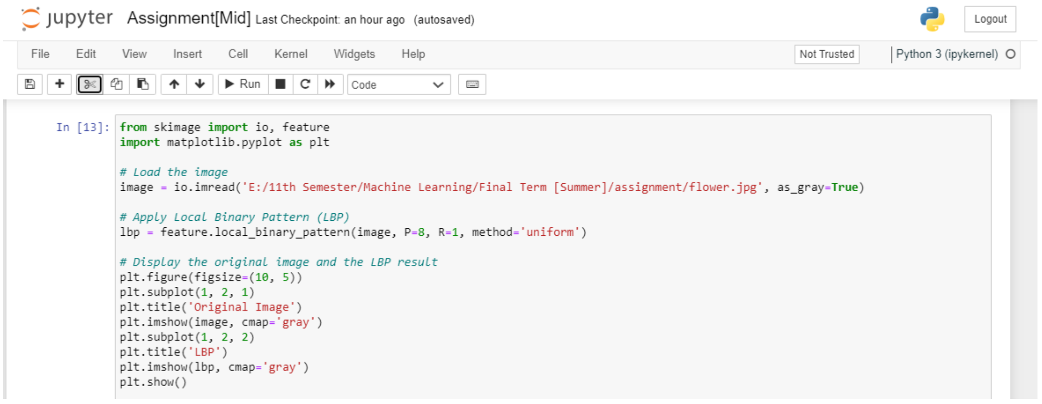
```
In [ ]: !pip install opencv-python
```

```
In [12]: import cv2
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread(r"E:/11th Semester/Machine Learning/Final Term [Summer]/assignment/cat.jpg", cv2.IMREAD_GRAYSCALE)

# Apply Canny edge detector
edges = cv2.Canny(image, 100, 200)

# Display the original image and the edges
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 2, 2)
plt.title('Edges')
plt.imshow(edges, cmap='gray')
plt.show()
```

Output:

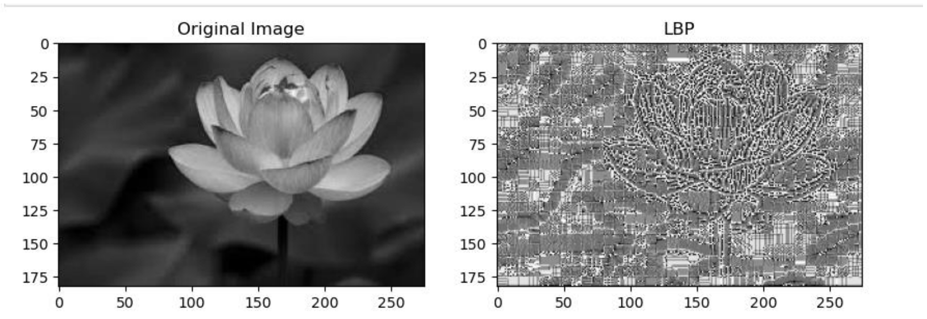


Code (LBP):

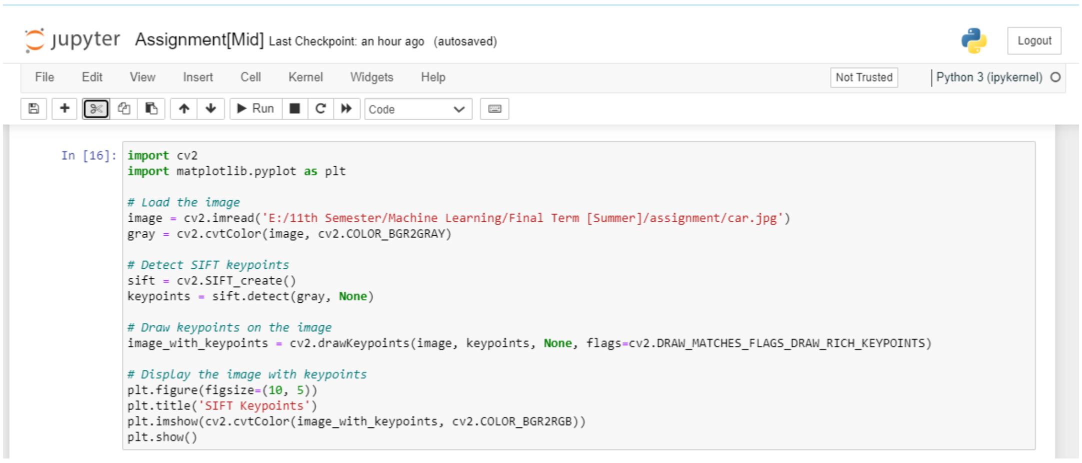

Jupyter Assignment[Mid] Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [13]: from skimage import io, feature
import matplotlib.pyplot as plt

# Load the image
image = io.imread('E:/11th Semester/Machine Learning/Final Term [Summer]/assignment/flower.jpg', as_gray=True)

# Apply Local Binary Pattern (LBP)
lbp = feature.local_binary_pattern(image, P=8, R=1, method='uniform')

# Display the original image and the LBP result
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 2, 2)
plt.title('LBP')
plt.imshow(lbp, cmap='gray')
plt.show()
```

Output:



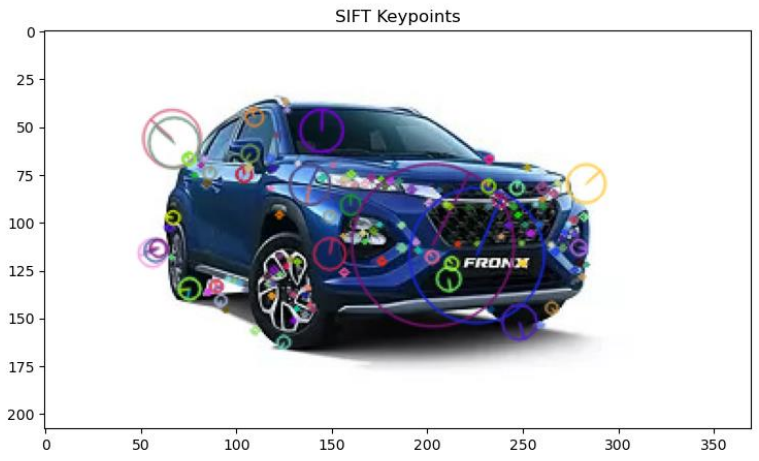
Code (SIFT):



jupyter Assignment[Mid] Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [16]:

```python
import cv2
import matplotlib.pyplot as plt

# Load the image
image = cv2.imread('E:/11th Semester/Machine Learning/Final Term [Summer]/assignment/car.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect SIFT keypoints
sift = cv2.SIFT_create()
keypoints = sift.detect(gray, None)

# Draw 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
plt.figure(figsize=(10, 5))
plt.title('SIFT Keypoints')
plt.imshow(cv2.cvtColor(image_with_keypoints, cv2.COLOR_BGR2RGB))
plt.show()
```

Output:



Discussion:

Advantages: By streamlining the raw image data through feature extraction, it becomes simpler to evaluate and interpret. The dimensionality of the image data is decreased by extracting significant features, which may result in faster processing times and more effective storage. By giving machine learning models concise representations of the image input, feature extraction can dramatically raise the performance of these models. Accurate results can be obtained by employing feature extraction techniques that effectively increase the analysis's robustness to noise and fluctuations in the picture data.

Drawbacks: But the effectiveness of feature extraction algorithms can be greatly influenced by the quality of the input photos. Low-quality or noisy images may lead to less accurate feature extraction. Feature extraction reduces dimensionality, but it may also cause some information to be lost that is crucial for specific applications.

Conclusion:

Image feature extraction is an important step in computer vision and image processing because it allows us to extract useful information from raw image data. We can unlock the potential of visual data and drive advances in a variety of fields by carefully selecting and implementing appropriate techniques.