Lab Assignment 6

Problem Statement

Write a program to implement Bag-of-words for a given image using Python.

Description

Bag-of-Words (BoW) is a technique often used in image processing to represent an image as a histogram of "visual words." In this assignment, your task is to implement the Bag-of-Words model for a given image using Python. This involves extracting features from the image, clustering the features into visual words, and then representing the image as a histogram based on these visual words.

Hints to Solve the Problem

Libraries to Use

- Use **OpenCV** for image processing and feature extraction.
- Use **NumPy** for numerical operations.
- Use **Matplotlib** for visualizing the results.

Approach and Pseudo Code

1. **Load the Image:**
    - Read the image file using OpenCV.
    - Convert the image to grayscale to simplify feature extraction.
2. **Feature Extraction:**
    - Use a feature detector such as SIFT or ORB to detect keypoints and compute descriptors.
    - Descriptors are the features of the image that will be used to create the Bag-of-Words representation.
3. **Cluster Descriptors:**
    - Use k-means clustering (e.g., MiniBatchKMeans) to cluster the descriptors into `n_clusters` (visual words).
    - The cluster centroids represent the visual words.
4. **Create a Histogram of Visual Words:**
    - Assign each descriptor to the nearest cluster (visual word).
    - Count the occurrences of each visual word in the image to build a histogram.
    - Normalize the histogram.
5. **Visualize Results:**

- Optionally, plot the histogram using Matplotlib to visualize the Bag-of-Words representation.

Functions to Use and Their Purpose

- **Image Loading and Grayscale Conversion:**
  - Use OpenCV functions to load and convert images to grayscale.
- **Feature Detection and Extraction:**
  - Use SIFT or ORB functions in OpenCV to detect keypoints and extract descriptors.
- **Clustering:**
  - Use MiniBatchKMeans from scikit-learn to perform clustering on descriptors and generate visual words.
- **Histogram Creation and Normalization:**
  - Use NumPy utilities to count occurrences of labels and normalize the histogram.
- **Visualization:**
  - Use Matplotlib to plot the histogram of visual words.

Expected Outcome

By following these steps, you will:

- Extract visual features from an image.
- Cluster these features into visual words.
- Represent the image as a histogram of visual words.
- Optionally visualize the histogram to understand the feature distribution.

In [1]:
```
# Download assignment files
!wget https://github.com/buntyke/vnr_dlcv2024_labs/releases/download/DLCVLab6/harbo
```

```
--2024-12-08 07:15:33--  https://github.com/buntyke/vnr_dlcv2024_labs/releases/dow
nload/DLCVLab6/harbour-bridge.jpg
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e
65be/878811324/fb763cc2-3ea4-4539-aa02-b56783d34e76?X-Amz-Algorithm=AWS4-HMAC-SHA2
56&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2Fs3%2Faws4_requ
est&X-Amz-Date=20241208T071533Z&X-Amz-Expires=300&X-Amz-Signature=16bff65246e9c7ec
b55b0a634ba489bfd3dce464c2c07a9c3409d0ca310d49bf&X-Amz-SignedHeaders=host&response
-content-disposition=attachment%3B%20filename%3Dharbour-bridge.jpg&response-conten
t-type=application%2Foctet-stream [following]
--2024-12-08 07:15:33--  https://objects.githubusercontent.com/github-production-r
elease-asset-2e65be/878811324/fb763cc2-3ea4-4539-aa02-b56783d34e76?X-Amz-Algorithm
=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20241208%2Fus-east-1%2
Fs3%2Faws4_request&X-Amz-Date=20241208T071533Z&X-Amz-Expires=300&X-Amz-Signature=1
6bff65246e9c7ecb55b0a634ba489bfd3dce464c2c07a9c3409d0ca310d49bf&X-Amz-SignedHeader
s=host&response-content-disposition=attachment%3B%20filename%3Dharbour-bridge.jpg&
response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 208114 (203K) [application/octet-stream]
Saving to: ‘harbour-bridge.jpg’

harbour-bridge.jpg  100%[===================>] 203.24K  --.-KB/s    in 0.02s

2024-12-08 07:15:34 (8.72 MB/s) - ‘harbour-bridge.jpg’ saved [208114/208114]
```

In [2]:

```python
### WRITE CODE HERE ###
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
from sklearn.cluster import MiniBatchKMeans

def extract_features(image_path, extractor):
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    keypoints, descriptors = extractor.detectAndCompute(gray, None)
    return descriptors

def generate_codebook(descriptors, n_clusters):
    kmeans = MiniBatchKMeans(n_clusters=n_clusters)
    kmeans.fit(descriptors)
    return kmeans

def image_to_bow(features, kmeans):
    labels = kmeans.predict(features)
    histogram = np.bincount(labels, minlength=kmeans.n_clusters)
    return histogram / histogram.sum() # normalize histogram

# Load image and display
image_path = './harbour-bridge.jpg'
image = cv2.imread(image_path)
cv2_imshow(image)

# Extract features from your image
sift = cv2.SIFT_create()
descriptors = extract_features(image_path, sift)

# Generate codebook
```

```
n_clusters = 100 # Number of visual words
kmeans = generate_codebook(descriptors, n_clusters)

# Represent image as bag-of-words
bow = image_to_bow(descriptors, kmeans)
print(bow)
```

```
[0.00354161 0.00885403 0.00556539 0.00505945 0.00834809 0.00480648
 0.01821401 0.01441943 0.00986592 0.00758917 0.01087781 0.01214268
 0.00581837 0.0073362  0.02352644 0.00961295 0.01366051 0.01037187
 0.0101189  0.00632431 0.01037187 0.01113079 0.00809512 0.01669618
 0.00935998 0.00784215 0.01644321 0.00683026 0.00809512 0.00531242
 0.01441943 0.01037187 0.00986592 0.00809512 0.02175563 0.0045535
 0.00632431 0.02150266 0.00480648 0.02985075 0.01138376 0.00708323
 0.00784215 0.00177081 0.00860106 0.00961295 0.00354161 0.01441943
 0.00758917 0.00885403 0.00860106 0.01239565 0.0118897  0.00632431
 0.0027827  0.01441943 0.0101189  0.00809512 0.01037187 0.00708323
 0.00657728 0.02782697 0.00961295 0.01391348 0.00480648 0.00607134
 0.00758917 0.00328864 0.01087781 0.0045535  0.00404756 0.01391348
 0.00430053 0.00758917 0.00784215 0.00177081 0.01113079 0.01037187
 0.01062484 0.00632431 0.00986592 0.01138376 0.02074374 0.01138376
 0.01340754 0.01138376 0.01138376 0.00531242 0.00834809 0.01037187
 0.01416646 0.01087781 0.00834809 0.00050594 0.00961295 0.02959777
 0.00834809 0.00885403 0.01290159 0.00961295]
```

In [ ]: