Experiment 10

Aim: To Create Program to perform a retrieving Image and Searching

Objective: The fundamental need of any image retrieval model is to search and arrange the images that are in a visual semantic relationship with the query given by the user. Most of the search engines on the Internet retrieve the images based on text-based approaches that require captions as input.

Theory:

Image retrieval is the process of searching for and organizing images in a manner that reflects their visual content. Unlike text-based image retrieval, which relies on textual metadata such as captions or keywords, content-based image retrieval (CBIR) operates on the visual features of images themselves. This section will delve into the fundamental concepts and techniques that underlie a CBIR system.

1. Feature Extraction:In CBIR, images are represented using a set of features that capture their visual characteristics. These features can be low-level, like color histograms, texture descriptors, or more advanced, such as deep learning-based feature vectors.Convolutional Neural Networks (CNNs) are commonly used for feature extraction in modern CBIR systems. Pre-trained CNN models, like VGG, ResNet, or Inception, can transform images into high-dimensional feature vectors.Feature extraction aims to create a compact and meaningful representation of the image's visual content. This representation enables efficient comparison and retrieval.

2. Similarity Metrics:

To search for similar images, a similarity metric is employed to compare the feature vectors of the query image and database images.Common similarity metrics include cosine similarity, Euclidean distance, or Jaccard similarity, depending on the nature of the features used.Cosine similarity is often preferred for feature vectors as it measures the cosine of the angle between the vectors, providing a measure of their similarity without being sensitive to vector length.

3. Query Processing:When a user submits a query image, its features are extracted using the same methodology as the database images.These query features are then compared to the features of images in the database using the chosen similarity metric.

4. Ranking and Retrieval:The result of the similarity comparison is a list of database images ranked by their similarity to the query image.The images most similar to the query appear at the top of the list, providing an ordered retrieval result.

5. Challenges:

Image variability: Images can have variations in scale, viewpoint, lighting, and background, making it challenging to establish robust feature representations.

Scalability: Handling large image databases efficiently is a significant challenge. Indexing and retrieval speed become critical in large-scale systems.

Semantic gap: There may be a discrepancy between low-level visual features and high- level semantic content in images, which can affect retrieval accuracy.

6. Improvements:

Fusion of multiple features: Using multiple feature types and combining their results can enhance retrieval performance.

Relevance feedback: Allowing users to provide feedback on retrieved images to refine future searches.

Machine learning techniques: Utilizing machine learning models to learn feature embeddings and improve ranking algorithms.

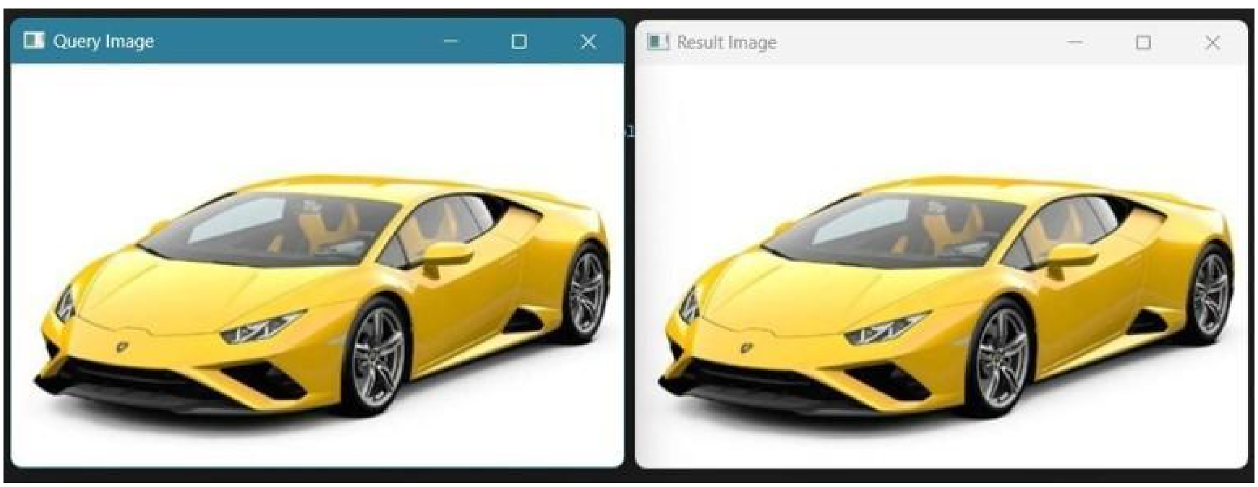
Code:

```
import cv2
import numpy as
np
from sklearn.metrics.pairwise import cosine_similarity
import os
```

```
# Function to extract image features (histograms) defextract_features(image_path):

    image = cv2.imread(image_path)
    hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    return hist
# Function to search for a query image in the images
folder defsearch_image(query_image_path,
images_folder):
    query_features =
    extract_features(query_image_path) image_paths
    = []

    similarities = []
        for root, dirs, files in
            os.walk(images_folder): for file in
            files:
                iffile.endswith(('.jpg', '.jpeg', '.png', '.bmp')):
                    image_path = os.path.join(root, file)
                    image_features =
                    extract_features(image_path)
                similarity = cosine_similarity([query_features], [image_features])
                image_paths.append(image_path)

                similarities.append(similarity)
    # Find the index of the most similar image
    most_similar_idx = np.argmax(similarities)
    most_similar_image_path =
    image_paths[most_similar_idx]
    returnmost_similar_image_path
    if name == ' main ':
        images_folder = 'images'
        query_image_path =
        'image.png'
```

```
result_image_path = search_image(query_image_path, images_folder)

if result_image_path:
   result_image = cv2.imread(result_image_path)
   cv2.imshow('Query Image',
   cv2.imread(query_image_path)) cv2.imshow('Result
   Image', result_image)
   cv2.waitKey(0)
   cv2.destroyAllWindo
   w s()

else:
     print('No matching image found.')
```

Output:



Conclusion:

In this image retrieval experiment, we developed a content-based image retrieval (CBIR) system that allows users to search for images based on visual content rather than textual metadata. The experiment extracts image features, such as histograms, to represent their visual characteristics and employs cosine similarity to compare these features. Users can provide a query image, and the experiment identifies the most visually similar image from a specified folder, offering an ordered retrieval result.