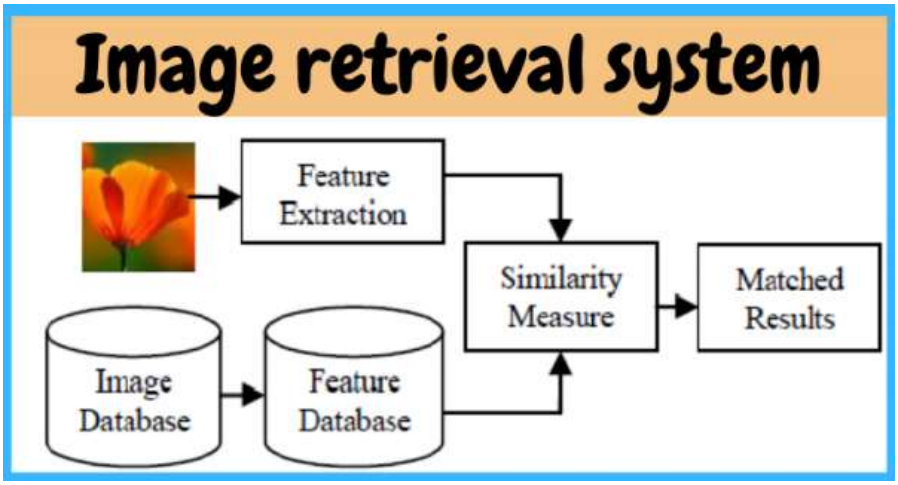
Experiment No.10
To Create Program to perform a retrieving Image and Searching
Date of Performance:09/10/23
Date of Submission: 16/10/23

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 a fundamental and long-standing computer vision task that involves finding images similar to a provided query from a large database. It's often considered as a form of fine-grained, instance-level classification. Not just integral to image recognition alongside classification and detection, it also holds substantial business value by helping users discover images aligning with their interests or requirements, guided by visual similarity or other parameters.



Creating a program for image retrieval and searching using SIFT (Scale-Invariant Feature Transform) descriptors involves several key theoretical concepts and steps.

- SIFT Descriptors: SIFT is a powerful feature extraction technique that identifies key points (interest points) in an image and extracts descriptors around these key points. These descriptors are invariant to scaling, rotation, and partially invariant to changes in illumination and viewpoint. SIFT descriptors are widely used in image retrieval because of their robustness
- Image Database:You need a database of images that you want to search through. This database should contain a collection of images with which you'll compare the query image.
- Feature Extraction for the Query Image:You extract SIFT descriptors from the query image using the SIFT algorithm. These descriptors will serve as the reference for similarity comparison.
- Feature Matching:To find similar images, you compare the SIFT descriptors of the query image with the SIFT descriptors of images in the database. One common approach is to use a nearest-neighbor matching algorithm, such as the brute-force method or a more efficient method like FLANN (Fast Library for Approximate Nearest Neighbors). For each descriptor in the query image, you find the best matching descriptors in the database images.

- Similarity Measure:After finding matching descriptors, you calculate a similarity measure to determine how similar the query image is to each image in the database. Common similarity measures include the number of matches, the ratio of good matches, or a more sophisticated metric like Lowe's ratio test, which is used to filter out low-quality matches.
- Ranking and Sorting:Based on the similarity measure, you rank and sort the images in the database in order of similarity to the query image.
- Retrieval and Display:You retrieve the top-ranked images based on similarity and display them to the user. This can be done in a separate window for each similar image or in a grid format, depending on your application's requirements.
- Scaling and Optimization:If your image database is large, you may need to implement techniques for indexing and efficiently searching through the database, such as creating hierarchical structures like a visual vocabulary (e.g., Bag of Words) and using data structures like KD-trees or product quantization.

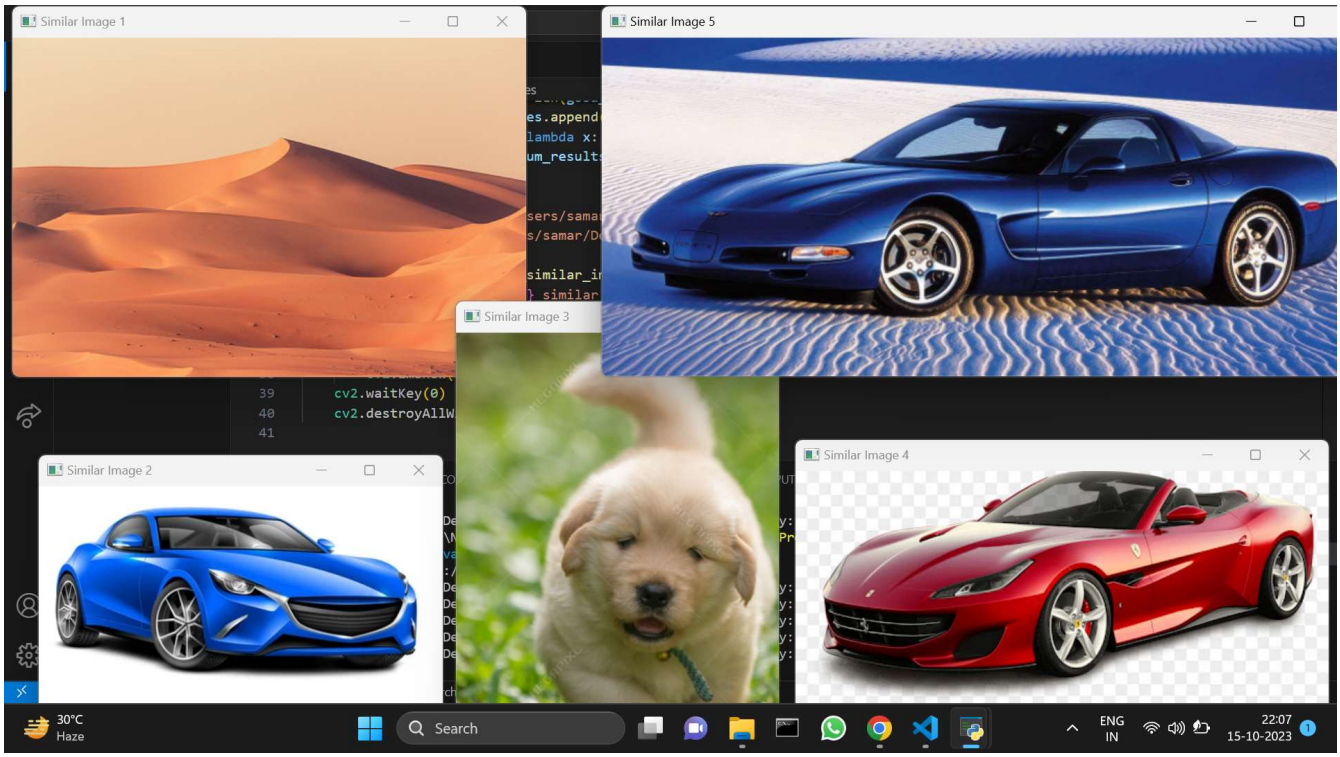
Code:-

```
import cv2
import os

def extract_sift_descriptors(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(image, None)
    return descriptors

def search_similar_images(query_image_path, database_path, num_results=5):
    query_descriptors = extract_sift_descriptors(query_image_path)
    bf = cv2.BFMatcher()
    similar_images = []
    for root, dirs, files in os.walk(database_path):
        for file in files:
            if file.endswith((".jpg", ".jpeg", ".png")):
                image_path = os.path.join(root, file)
                database_descriptors = extract_sift_descriptors(image_path)
                matches = bf.knnMatch(query_descriptors, database_descriptors, k=2)
                good_matches = []
                for m, n in matches:
                    if m.distance < 0.75 * n.distance:
                        good_matches.append(m)
                similarity = len(good_matches)
                similar_images.append((image_path, similarity))
    similar_images.sort(key=lambda x: x[1], reverse=True)
    return similar_images[:num_results]

if __name__ == "__main__":
    query_image_path = "C:/Users/samar/Desktop/New folder/qimg1.png"
    database_path = "C:/Users/samar/Desktop/New folder/imgs"
```

```
num_results = 5
similar_images = search_similar_images(query_image_path, database_path, num_results)
print(f"Top {num_results} similar images to {query_image_path}:")
for i, (image_path, similarity) in enumerate(similar_images, start=1):
    print(f"{i}. Image: {image_path}, Similarity: {similarity}")
    similar_image = cv2.imread(image_path)
    cv2.imshow(f"Similar Image {i}", similar_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:- Query Image: -

Top 5 similar images to C:/Users/samar/Desktop/New folder/qimg1.png:
1. Image: C:/Users/samar/Desktop/New folder/imgs\img41.png, Similarity: 113
2. Image: C:/Users/samar/Desktop/New folder/imgs\img26.png, Similarity: 105
3. Image: C:/Users/samar/Desktop/New folder/imgs\img38.png, Similarity: 98
4. Image: C:/Users/samar/Desktop/New folder/imgs\img14.png, Similarity: 85
5. Image: C:/Users/samar/Desktop/New folder/imgs\img31.png, Similarity: 84



Conclusion:-

The experiment aimed to create an image retrieval program using SIFT descriptors, offering robust feature-based matching for content-based image searches. The process involved extracting SIFT descriptors from images and conducting nearest-neighbor matching, with similarity measures to rank and sort results. Evaluating performance and optimization strategies were emphasized, along with user interaction through a user-friendly interface. This experiment highlights the significance of feature-based image retrieval in various applications. The field continues to evolve with deep learning techniques like CNNs, promising further advancements in content-based image analysis.