CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment:1.2

Aim: Write a program to assess various feature matching algorithms for object recognition.

Software Required: Any Python IDE

Description:

Feature matching algorithms are crucial for object recognition tasks as they establish correspondences between keypoints or descriptors extracted from images. Some commonly used feature matching algorithms for object recognition are:

1. Brute-Force Matching: Brute-force matching involves comparing each feature in one image with every feature in another image using a distance metric. It is a straightforward approach but can be computationally expensive for large feature sets.

2. Fast Library for Approximate Nearest Neighbors (FLANN): FLANN is an approximate nearest neighbor search library that accelerates feature matching by using indexing structures. It performs fast approximate nearest neighbor searches, reducing the computational cost compared to brute-force matching.

3. Nearest Neighbor with Ratio Test: In this approach, the nearest neighbors of each feature are found, and a ratio test is applied to select the best match. The ratio test compares the distances between the best match and the second-best match. If the ratio is below a certain threshold, the match is considered valid. This method helps to reduce false matches caused by ambiguous correspondences.

   a) Import the necessary libraries and modules (e.g., OpenCV).
   b) Load the reference image and the query image.
   c) Extract features from both images using a chosen feature extraction technique (e.g., SIFT, SURF, ORB).
   d) Apply feature matching algorithms to match the extracted features.
   e) Compute the similarity or matching score between the reference image and the query image based on the matched features.
   f) Evaluate and compare the performance of different feature matching algorithms using suitable metrics (e.g., accuracy, precision, recall, F1-score).
   g) Analyze and interpret the results to identify the strengths and weaknesses of each algorithm.

CourseName: Computer Vision Lab **Course Code:** CSP-422

h) Repeat the experiment with different combinations of feature extraction and matching algorithms to gain further insights.
i) Document the findings, observations, and conclusions.

Steps:

1. Import libraries (e.g., OpenCV).
2. Load reference image and query image.
3. Extract features from reference image using a chosen feature extraction technique.
4. Extract features from query image using the same feature extraction technique.
5. Apply feature matching algorithm to match the extracted features.
6. Compute matching score or similarity between reference and query image.
7. Evaluate and compare the performance of the feature matching algorithm using suitable metrics.
8. Repeat steps 3-7 with different feature extraction and matching algorithms.
9. Analyze and interpret the results.
10. Document the findings and conclusions.

Implementation/Output:

In [5]:
```
import cv2
import numpy as np
from skimage.transform import resize
import matplotlib.pyplot as plt
imag1=cv2.imread("C:/Users/Yash Gupta/OneDrive/Pictures/img1.jpg",cv2.IMREAD_GRAYSCALE)
imag2=cv2.imread("C:/Users/Yash Gupta/OneDrive/Pictures/img2.jpg",cv2.IMREAD_GRAYSCALE)
img1=cv2.resize(imag1,(600,600))
img2=cv2.resize(imag2,(600,600))
#plt.imshow(img1)
fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
axis[0].imshow(img1)
axis[1].imshow(img2)
```

In [ ]:
```
orb=cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

brute_force=cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches=brute_force.match(des1, des2)
matches=sorted(matches, key=lambda x: x.distance)

matching_results=cv2.drawMatches(img1, kp1, img2, kp2, matches[:20], None, flags=2)

cv2.imshow("Feature_Matching", matching_results)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

CourseName: Computer Vision Lab **Course Code:** CSP-422

```
Out[5]: <matplotlib.image.AxesImage at 0x24611c09a50>
```