Feature Matching on Library Catalogue

N V Ganapathi Raju, Department of Information Technology Gokaraju RangaRaju Institute of Engineering and Technology, Hyderabad, Email: nvgraju@griet.ac.in,
Gidde Akhil Reddy, Department of Information Technology Gokaraju RangaRaju Institute of Engineering and Technology, Hyderabad, Email: akhilreddygidde150@gmail.com

Abstract: Feature Matching means two image features are compared which may be different in their perspective, points, lightening, sizes, color, and shape. The main intention of the feature matching algorithm is to figure out the number of features of two images regardless of the transformation. Feature Matching can be categorized into two groups named feature detectors and feature descriptors. Feature detectors are defined as finding the features from an input image. The descriptors describe the distinctive extracted features on the image. This paper proposes Feature Matching on Library Catalogue. This paper is about Feature Matching on Library Catalogue. The purpose behind this paper is to know whether the captured input book image is present in the library catalogue or not and also comparing feature matching algorithms like ORB, SIFT and SURF to suggest the efficient algorithm for image feature matching. The Library Catalogue holds a separate set of terms, and all those terms will be mapped with the given input for appropriate result. If the given input book image is matched with the highest number of features in the catalogue, then the name of the book is displayed. If no image is matched, then it returns no book found in the catalogue.

Index Terms: Feature matching; precision; image; library catalogue ; ORB ; SIFT;SURF;

1.Introduction

In addition to the expeditious advancement in science and technology and Computer perception is widely used extensively in numerous applications, the requirements for image matching with effective performance has brought in demand. The word Feature usually describes or identifies some information or content about the image. Features are like patterns in an image which helps in identifying specific structures such as points. Feature Matching works with two particular types of images usually based on a search distance algorithm. The first image is considered as the input image was used to map the features or derive attributes with the second image is the target image. It could be difficult finding identical feature in at least two real-time images can be challenging in live scenarios. This case is majorly met when there are large images and high -resolution images. The accuracy always depends on certain things like quality of the image, the similarity and complexity of the provided images. In certain conditions one input image feature can match with two or more images but with the one which has more matching attributes is considered as the target image. Many algorithms are developed and introduced in recent times for image analysis, which are used for feature detection and extraction. Few comprehensive algorithms like ORB, BRISK, SIFT, SURF can work on both. Brute Force Matcher is used for comparing features of the input image with the defined target image set using a distance calculation algorithm. When it finds a closet match with the input image feature, it is returned. Due to many large comparisons, it is taking more time, but the results are pretty good. In computer vision, the content of the image is considered as key points. These key points help to locate objects in images. An efficient algorithm should be closed so that it can be performed in the real time and repeatedly. The key points are described by the corresponding descriptors in feature extraction.

Comparisons of three algorithms i.e., Oriented FAST and rotated BRIEF(ORB), Scale Invariant Feature Transform (SIFT) and Speed Up Robust Feature (SURF) are performed here on different images of same resolutions. In this ORB detect more features when compared to SIFT and SURF.SURF and SIFT has fewer features that can be recognized, but they recognize more definite features compared to ORB. Here in this paper Brute force algorithm is used to match features between the input and target image. This algorithm combines with k-Nearest Neighbors. It also helps in removing outliers. The process is usually carried out on two same 2d images in different positions by a camera. The feature detection and extraction of the image is performed by each algorithm one after the other. OpenCV library is used for live computer perception.

The Scale Invariant Feature Transform (SIFT) algorithm usually allows the detection of local feature points which is known as key points in an image. Generally, features are invariant, so that it can rotate and scaled.

SIFT need stable points in two images and it needs some features which characterizes a salient point to determine some correspondences between them. When one is trying to match the patches around feature points, it could have some negative result when the position of the object is changed which could lead to wrong matches. SIFT provides features characterizes a salient point which will remain invariant to changes in rotation or scale. Initially, key points are determined (salient feature points). After then refining their scale and location. Determine orientations and descriptors for each key point. The disadvantage of SIFT is very slow.
Speeded Up Robust Features (SURF) was inspired and developed from SIFT descriptor as it slowly and require better and speeded-up version. SURF gives robust features of an image. SURF is much faster than SIFT. Major Applications of SIFT are robot localization, image retrieval, object categorization, object and location recognition. SURF can extract scale and rotation invariant key point descriptor and detector. They have the ability to allocate more than one direction to each key point. It has smaller calculation for feature extraction when compared to SIFT. The Algorithm works in such a way that it detects key points, describe key points and match key points.
Oriented Fast and Rotated Brief (ORB) is a substitute for SIFT and SURF in matching performance and computation cost. ORB is a FAST (Features from Accelerated Segment Test) key point detector whereas the BRIEF descriptor will enhance the performance by making many modifications or adding features. Key points are found by FAST and for finding the main points between them Harris corner measure is used. The only disadvantage is that orientation and descriptors are not computed buy FAST. This is fulfilled by BRIEF. The BRIEF is bad with rotation so ORB rotates the BRIEF so that it works in line with orientation of key points. ORB takes the input image and converts it into grayscale. The ORB detector detects the key points in the input image and computes the descriptors which are similar in the both input image and the target image. It will then match the key points with Brute Force Matcher and displays the result.

2.LITERATURE SURVEY

2.1 Image Feature Matching and Object Detection Using Brute-Force Matchers.

Jackubovic et al[1] has suggested a framework for feature matching and object recognition. They have implemented four algorithms for the feature detection and extraction of an image. The brute Force Matcher is used for feature matching. They have concluded with their experiment results based on number of inline matches, mean error and execution time that SURF is the most effective and precise. It was stated that the outliers do exists which is not avoidable but the number can be decreased and omitted with visualization part.

2.2 A review of feature detection and match algorithms for localization and mapping

Li,Shimiao [2] performed comparative analysis on performance of the algorithms based on their precision,pace,magnitude and rotation transitivity. For detecting and matching features, they have concluded that ORB is the fastest algorithm. And suitable for real time scenarios. Sift performs high accuracy. Harris and FAST has showed poor performance in feature matching where as FAST performs higher speed in detecting features. MSER performs better when compared to Harris and FAST.

2.3 Evaluation of Image Feature Detection and Matching Algorithms

Yiwen el at [3] analysed and performed comparative analysis on SIFT, SURF, ORB, BRISK, KAZE and AKAZE .Oxford dataset has been used to compare the resilience of distinct algorithms under various transformations. The different robustness indicators that have been used for performance comparison are speed, complete running time, detecting time and matching time. With experimental results shows that under blur and lightening transformation SURF is more efficient. In viewpoint transformation BRISK and AKAZE are efficient. They concluded that each algorithm has its benefits and effects under different transformations like few algorithms has higher accuracy rate but takes big time and vice versa few has lower accuracy rate but take less time.

2.4 Research on image feature matching algorithm based on feature optical flow and corner feature

Song el at [4] proposed a quick feature extraction and appropriate algorithm where include feature optical flow with corner features built on ORB algorithm. It uses ORB to extract key points followed displacement vector is calculated. To track feature points Lucas-Kandle feature optical flow is used. knn radius is used to

check for feature key points in the image area. RANSAC algorithm is used to takeaway unwanted features. It can wipe out high number of imbalances matches and get high precision image matching. The resilience of the algorithm is increased in real-time scene and provides feature matching precision.

2.5 Research on feature point matching algorithm improvement using depth prediction

Chen el at [5] They have applied depth prediction with SIFT into feature matching by proposing novel mismatch algorithm to enhance the precision if feature matching. MonoDepth algorithm is used to find the depth value in an image. In their approach the prophecy of depth maps is performed in pixel-wise by red-green-blue images using deep learning algorithm. The tests outcome has demonstrated that they have increased the precision of feature matching by the novel method where the mismatching is lowered by narrowing the matching area. This method cannot rely on surface features for predicting the depth but it has the capability of predicting the depth on low texture surface scenes.

2.6 A Novel Approach of Object Detection Using Point Feature Matching Technique for Colored Images

Sharma el at [6] For object recognition in colored image they have implemented point feature matching technique using SURF approach. They have used a novel SURF algorithm for extracting, describing and then matching the objecting from colored images. The flow of the algorithm is that it detects a specific object by finding the correlation points among the referential image and the objective grey scale image. The SURF has non-repeating patterns which help to detect the objects for unique features matches. The experimental results have proven that it works well with the colored objects and with the regular pattern objects. This has greater advantage when used in real time applications due to its fast computation.

3.METHODOLOGY

In this project, three algorithms have been implemented for feature matching and compared. Initially the input image captures from the webcam and is stored in a temporary location. The image is read from the temporary location and it is resized to 500X500 pixels. After reading the image, adding each and every image path from the catalogue folder to the files list using the list comprehension method. Looping each and every path from the list folder, reading the image and resizing every image by 500X500 pixels.

```
image1 = cv2.imread("photo.jpg")
image1 = cv2.resize(image1,(500,500))

txp = "/content/drive/MyDrive/src_img/"
ext = ['png', 'jpg', 'jpeg']    # Add image formats here

files = []

[files.extend(glob.glob(txp + '*.' + e)) for e in ext]
print(files)
fc = 0
namelist = {}

for file in files:
    img = cv2.imread(file)
    print(type(img))
    img = cv2.resize(img,(500,500))
    try:
       # s,d = siftMatching(img, image1)
        s = orb(img,image1)
        print(len(s))
        namelist[file] =  len(s)
    except Exception as e:
        print(e)
        pass
```

Code Snippet of calling the algorithm functions of ORB,SIFT and SURF.

3.1 Algorithm for SIFT

Step 1 :The input image is captured from the webcam and stored at a temporary location. It is resized to 500X 500 px.

Step 2 : Each image from the dataset is resized to 500 X 500 px are passed to SIFT Matching function as parameters.
Step 3: A Model object for SIFT is created from OpenCV module SIFT_create() . Images are passed to detectAndCompute() for detecting and computing the key points and descriptor points.
Step 4: Flann Based Matcher is used to detect the matches and uses K Nearest Neighbor algorithm to find the nearest match algorithm when more than one match is found. KNN will return the best match out of the matches found. A better suitable algorithm is being opted out of all algorithm, it has in its collection. Two dictionaries are to be passed to the FLANN based matcher First parameter is indexParams where FlANN_INDEX_KDTREE is passed as an algorithm. Second parameter is search_params where checks are passed. Using the KNN match trying to get good matches
whose points distance is less than the threshold value of 0.7.
Step 5:Matching points are used with homography function from pydegensac where a set of points need to be sent from both the input and target image, it will find the transformative perspective of the object. It will then return the inliers. Pydegensac executes LO-RANSAC and DEGENSAC.
Step 6: Now need to extract all the locations of the matched key points and inliers are drawn using cv2.drawmatches function and the image is displayed.
Step 7: The inliers that add to dictionary with file name as the key. Finding max value from the dictionary above the threshold value and accessing it with the key which is the output book name.
The input image has been captured from the webcam to detect the features and match the features with the dataset images.

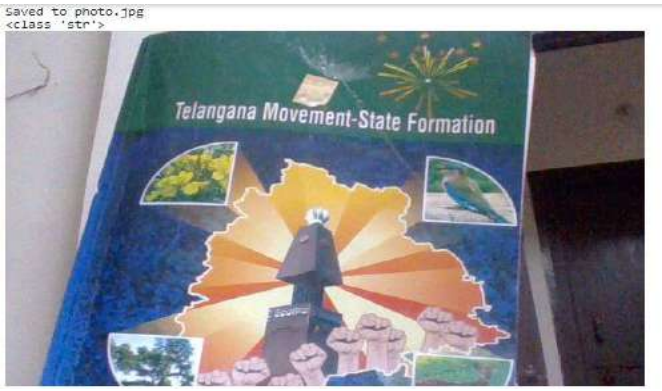


Figure 2:**Captured Input Image from webcam**

Each image inliers has been matched with the input image and the highest number of inliers of an image is taken as result image.

Figure 3: **Finding the number of inliers using SIFT**

3.2 Algorithm for SURF
Step 1 :The input image is captured from the webcam and stored at a temporary location. It is resized to 500X 500 px.
Step 2 : Each image from the dataset is resized to 500 X 500 px are passed to SURF Matching function as parameters.
Step 3: A Model object for SIFT is created from OpenCV module SURF_create() . Images are passed to

detectAndCompute() for detecting and computing the key points and descriptor points.

Step 4: Flann Based Matcher is used to detect the matches and uses K Nearest Neighbor algorithm to find the nearest match algorithm when more than one match is found. KNN will return the best match out of the matches found. A better suitable algorithm is being opted out of all algorithm, it has in its collection. Two dictionaries are to be passed to the FLANN based matcher First parameter is indexParams where FlANN_INDEX_KDTREE is passed as an algorithm. Second parameter is search_params where checks are passed. Using the KNN match trying to get good matches whose points distance is less than the threshold value of 0.7.

Step 5:Matching points are used with homography function from pydegensac where a set of points need to be sent from both the input and target image, it will find the transformative perspective of the object. It will then return the inliers. Pydegensac executes LO-RANSAC and DEGENSAC.

Step 6: Now need to extract all the locations of the matched key points and inliers are drawn using cv2.drawmatches function and the image is displayed.

Step 7:The inliers that add to dictionary with file name as the key. Finding max value from the dictionary above the threshold value and accessing it with the key which is the output book name.

Figure 4: **Finding the number of inliers using SURF**

Each image inliers have been matched with the input image and the highest number of inliers of an image is taken as result image.

```
def siftMatching(img1, img2):

    #surf = cv2.xfeatures2d.SURF_create(100)
    surf = cv2.xfeatures2d.SIFT_create()

    kp1, des1 = surf.detectAndCompute(img1, None)
    kp2, des2 = surf.detectAndCompute(img2, None)

    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks = 50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(des1,des2,k=2)
    good = []
    for m, n in matches:
        if m.distance < 0.7*n.distance:
            good.append(m)

    src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1, 2)
    dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1, 2)
    print(src_pts,dst_pts)
    model, inliers = pydegensac.findHomography(src_pts, dst_pts, 10.0, 0.99, 10000)

    n_inliers = np.sum(inliers)
    print('Number of inliers: %d.' % n_inliers)

    inlier_keypoints_left = [cv2.KeyPoint(point[0], point[1], 1) for point in src_pts[inliers]]
    inlier_keypoints_right = [cv2.KeyPoint(point[0], point[1], 1) for point in dst_pts[inliers]]
    placeholder_matches = [cv2.DMatch(idx, idx, 1) for idx in range(n_inliers)]
    image3 = cv2.drawMatches(img1, inlier_keypoints_left, img2, inlier_keypoints_right, placeholder_matches, None)

    cv2_imshow(image3)
    cv2.waitKey(10)

    src_pts = np.float32([ inlier_keypoints_left[m.queryIdx].pt for m in placeholder_matches ]).reshape(-1, 2)
    dst_pts = np.float32([ inlier_keypoints_right[m.trainIdx].pt for m in placeholder_matches ]).reshape(-1, 2)
    return src_pts, dst_pts
```

Code snippet of SIFT and SURF Algorithm.

3.3 Algorithm for ORB

Step 1 :The input image is captured from the webcam and stored at a temporary location. It is resized to 500X 500 px.

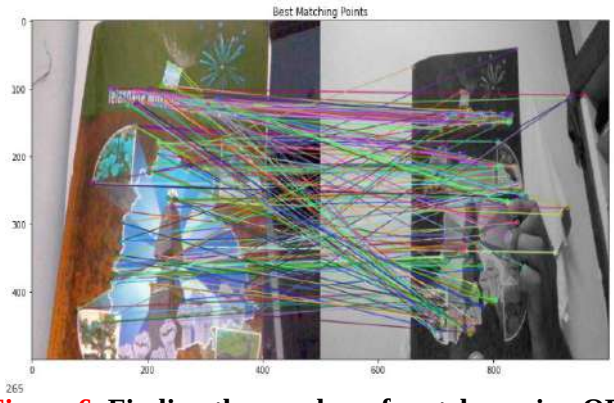
Step 2 : Each image from the dataset is resized to 500 X 500 px are passed to orb function as parameters.

Step 3: ORB_create() is used to create an object. From OpenCV cvtColor is used to convert from one color space to another color space. These images are converted to grayscale.

Step 4: Detectandcompute function will find all the key points and descriptors. These points are drawn on the image.

Step 5: The descriptors of both images are passed to BFMatcher to match the key points.BFMatcher uses cv2.NORM_Hamming function to compute the difference among the two key point descriptors.

Step 6: The matches are sorted and returned besides it are passed to draw function to draw all the matches. The length of matches is considered as inlier count.
Step 7: The inliers that add to dictionary with file name as the key. Finding max value from the dictionary above the threshold value and accessing it with the key which is the output of the book name.
Each image matches have been matched with the input image and the highest number of inliers of an image is taken as result image.



265

Figure 6: **Finding the number of matches using ORB.**

```
def orb(query_img,train_img):
    orb = cv2.ORB_create()
    query_img_bw = cv2.cvtColor(query_img,cv2.COLOR_BGR2GRAY)
    train_img_bw = cv2.cvtColor(train_img, cv2.COLOR_BGR2GRAY)
    train_keypoints, train_descriptor = orb.detectAndCompute(train_img_bw, None)
    test_keypoints, test_descriptor = orb.detectAndCompute(query_img_bw, None)
    keypoints_without_size = np.copy(train_img_bw)
    keypoints_with_size = np.copy(train_img_bw)
    cv2.drawKeypoints(train_img, train_keypoints, keypoints_without_size, color = (0, 255, 0))
    cv2.drawKeypoints(train_img, train_keypoints, keypoints_with_size, flags = cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck = True)
    # Perform the matching between the ORB descriptors of the training image and the test image
    matches = bf.match(train_descriptor, test_descriptor)
    # The matches with shorter distance are the ones we want.
    matches = sorted(matches, key = lambda x : x.distance)
    result = cv2.drawMatches(train_img, train_keypoints, query_img_bw, test_keypoints, matches, query_img_bw, flags = 2)
    # Display the best matching points
    plt.rcParams['figure.figsize'] = [14.0, 7.0]
    plt.title('Best Matching Points')
    plt.imshow(result)
    plt.show()
    return matches
```

Code Snippet of ORB algorithm.

3.4 Results

The dataset images have been collected from the Gokaraju Rangaraju College of Engineering Library.A Test has been performed on 100 book images on each algorithm and out of three algorithms ,SURF is more efficient when compared with SIFT and ORB.ORB is computational faster than SIFT and SURF but not much efficient with respect to results which have been achieved. SURF has higher precision rate but computation is slower than SIFT and ORB. SURF has precision rate of 95 percent whereas SIFT has 91 percent and ORB 85 percent of features matching .

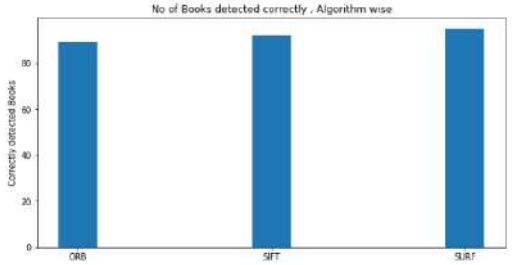


Figure 1: Comparison between Algorithms for Number of books that detected Correctly

```
l=namelist.values()
thresh = 125
res = any(ele > 125 for ele in l)
if res == True:
   BookName = max(zip(namelist.values(), namelist.keys()))[1]
   Bookname = str(BookName).split('/')[-1][:-4]
   print('Bookname======>',Bookname)
else:
  print('No such Book in the catalogue')
```

```
Bookname======> Telangana Movement-State Formation
```

Code Snippet of the end result displaying the book name.

4.CONCLUSION

In this paper Feature Matching on Library Catalogue has been proposed .In this process three algorithms are implemented for feature matching. The image set holds the library book images. Feature matching is implemented and compared among three Algorithms namely SIFT,SURF and ORB. The input image is captured and features are matched with the features in the image set. The highest number of features matched will return the book name , if features are not matched to the given threshold frequency number, then it returns no book found. From the Experimental results, SURF has highest precision rate in feature matching and found to be the most efficient when compared with SIFT and ORB.ORB has higher speed in computing the features than SIFT and SURF but lacks in precision when compared with the other two Algorithms.

5. **Future Enhancement**

I would like further implement and compare feature matching on library catalogue using the FAST and BRISK Algorithms and improve the efficiency of SIFT,SURF and ORB by proposing a methodology.

6.REFERENCES

[1] Jakubovic, Amila & Velagic, Jasmin. (2018). Image Feature Matching and Object Detection Using Brute-Force Matchers. 83-86. 10.23919/ELMAR.2018.8534641.
[2] Li, Shimiao. (2017). A review of feature detection and match algorithms for localization and mapping. IOP Conference Series: Materials Science and Engineering. 231. 012003. 10.1088/1757-899X/231/1/012003.
[3] Ou, Yiwen & Cai, Zhiming & Lu, Jian & Dong, Jian & Ling, Yufeng. (2020). Evaluation of Image Feature Detection and Matching Algorithms. 220-224. 10.1109/ICCCS49078.2020.9118480.
[4] Song, Tao & Bing, Chen & Zhao, Fu & Huang, Zheng & Huang, Mu. (2020). Research on image feature matching algorithm based on feature optical flow and corner feature. The Journal of Engineering. 2020. 10.1049/joe.2019.1156.
[5] Chen, Yongbin & Wang, Guitang & Wu, Liming. (2019). Research on feature point matching algorithm improvement using depth prediction. The Journal of Engineering. 2019. 10.1049/joe.2018.9142.
[6] Sharma, Manvinder & Singh, Harjinder & Singh, Sohni & Gupta, Anuj & Goyal, Sumeet & Kakkar, Rahul. (2020). A Novel Approach of Object Detection Using Point Feature Matching Technique for Colored Images. 10.1007/978-3-030-29407-6_40.
[7] Lin Chen, Franz Rottensteiner & Christian Heipke (2021) Feature detection and description for image matching: from hand-crafted design to deep learning, Geo-spatial Information Science, 24:1, 58-74, DOI: 10.1080/10095020.2020.1843376
[8] Ting Bai, Le Wang, Dameng Yin, Kaimin Sun, Yepei Chen, Wenzhuo Li, Deren Li. (2022)
[9] Eqbal, Shahid & Verna, Aanchal & Soni, Akanksha & Kumar, Ankit. (2021). Feature Matching of Images. 1-6. 10.1109/I2CT51068.2021.9418133.
[10] Zhao, Liming & Yang, Jingfan & Zhang, Yi & Huang, Junzhi. (2022). Research on Feature Matching of an Improved ORB Algorithm. 765-769. 10.1109/ITOEC53115.2022.9734583.
[11] https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/
[12] https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html
[13] https://analyticsindiamag.com/introduction-to-local-feature-matching-using-loftr/
[14] Serlin Z, Yang G, Sookraj B, Belta C, Tron R. Distributed and consistent multi-image feature matching via QuickMatch. The International Journal of Robotics Research. 2020;39(10-

11):1222-1238. doi:10.1177/0278364920917465
[15] Bellavia, Fabio. (2022). SIFT Matching by Context Exposed. IEEE Transactions on Pattern Analysis and Machine Intelligence. PP. 1-1. 10.1109/TPAMI.2022.3161853.