COMPUTER VISION

Assignment 2A Report

M22MA003

Question 1 : You are given a scene image containing a logo, and a gallery containing reference logos for 10 business brands. Find out which business brand is present in the scene. Try out three different approaches and compare them.

The three methods used are as follows:

I. SIFT Method

II. ORB Method

III. Using the logo as template and matching over the reference image

(a) Submit your implementation. File name of your python file should be logoMatch.py and it should take scene image, logo image and approach name as input and either show the matched region or say not enough match points found.

Steps Followed :-

1. Import the required libraries.
2. Define method for logo matching and iterate over each logo from the path.
3. Count the number of matches from each logo and check the maximum matching points logo.
4. Follow the steps for both examples and for all the three approaches.
5. Observe the results.

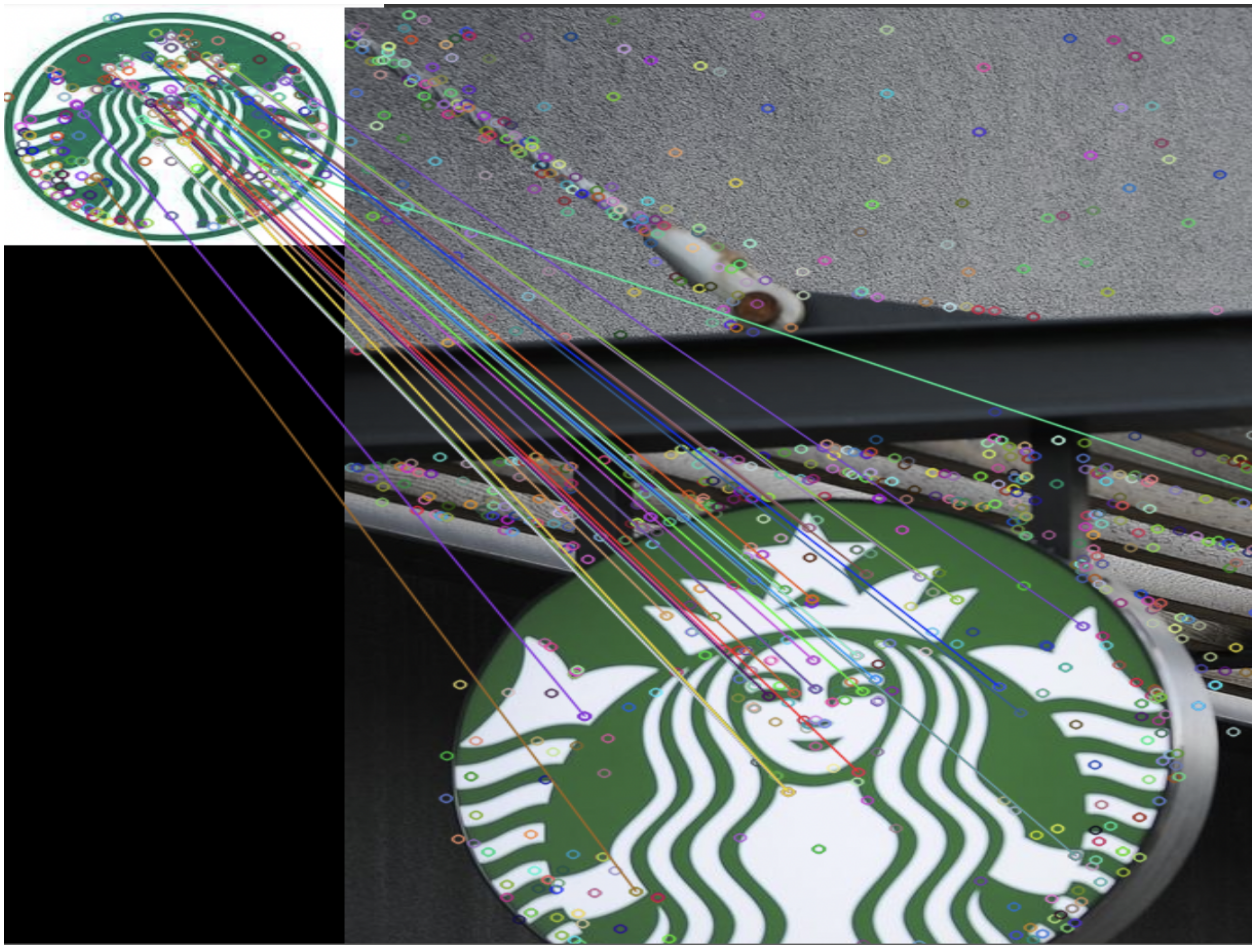
First Approach : Using SIFT Method : Scale Invariant Feature Transformation

Example 1 : Number of Matches found for each logo are as follows:

```
/content/logos/spar.jpg
Number of good matches are 1
/content/logos/tacobell.jpg
Number of good matches are 1
/content/logos/umbro.jpg
Number of good matches are 1
/content/logos/hp.jpg
Number of good matches are 9
/content/logos/kfc.jpg
Number of good matches are 0
/content/logos/tommyhilfiger.jpg
Number of good matches are 0
/content/logos/lg.jpg
Number of good matches are 1
/content/logos/shell.jpg
Number of good matches are 3
/content/logos/levis.jpg
Number of good matches are 19
/content/logos/nescafe.jpg
Number of good matches are 0
Matched logo is
```

Example 2 : **Using SIFT Method**

```
/content/example2/logos/motorola.jpg
/content/example2/logos/honda.jpg
/content/example2/logos/starbucks.jpg
/content/example2/logos/pepsi.jpg
/content/example2/logos/warnerbros.jpg
/content/example2/logos/toyota.jpg
/content/example2/logos/puma.jpg
/content/example2/logos/hp.jpg
/content/example2/logos/rolex.jpg
/content/example2/logos/lg.jpg
Matched logo is
```



Second Approach : Using ORB Method : Oriented FAST and Rotated BRIEF

Example 1 : Number of Matches found for each logo are as follows:

```
/content/logos/hp.jpg
Number of good matches are 3
/content/logos/kfc.jpg
Number of good matches are 34
/content/logos/tommyhilfiger.jpg
Number of good matches are 22
/content/logos/lg.jpg
Number of good matches are 8
/content/logos/shell.jpg
Number of good matches are 33
/content/logos/levis.jpg
Number of good matches are 15
/content/logos/nescafe.jpg
Number of good matches are 36
```

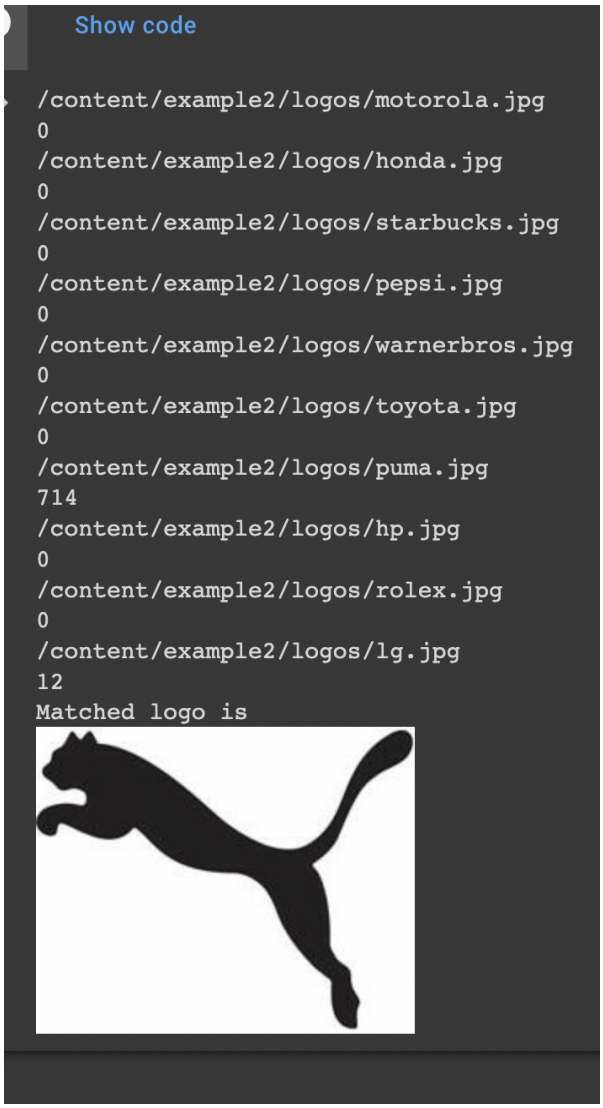
Example 2: **Using ORB Method**

```
/content/example2/logos/motorola.jpg
Number of good matches are 6
/content/example2/logos/honda.jpg
Number of good matches are 3
/content/example2/logos/starbucks.jpg
Number of good matches are 34
/content/example2/logos/pepsi.jpg
Number of good matches are 7
/content/example2/logos/warnerbros.jpg
Number of good matches are 9
/content/example2/logos/toyota.jpg
Number of good matches are 5
/content/example2/logos/puma.jpg
Number of good matches are 3
/content/example2/logos/hp.jpg
Number of good matches are 3
/content/example2/logos/rolex.jpg
Number of good matches are 3
/content/example2/logos/lg.jpg
Number of good matches are 2
Matched logo is
```

Using Template Matching Method:

Example 1:-

Example 2:-



(b) Show your results qualitatively in the report and write down your observations.

I. **SIFT Method - Best Results**
II. **ORB Method - Average Results**
III. **Using the logo as kernel and matching over the reference image. - Not as per expectation**

Continued..

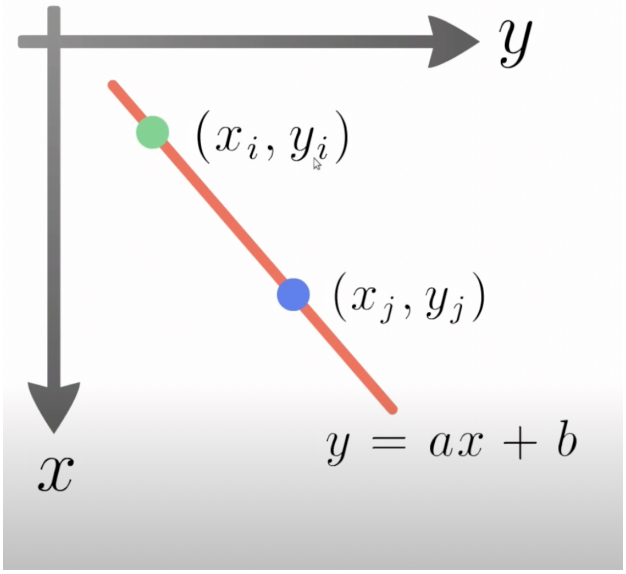
- ★ SIFT is used with FlannBasedMatcher which gives results by utilizing the KNN algorithm and therefore consumes less time and provides more accurate results when compared to ORB Method.
- ★ In this solution, ORB method is used with BruteForceMatcher which takes one descriptor at a time and compares it with all the descriptors in the other image, therefore, resulting in more time taken and hard comparisons leading to less number of matches.

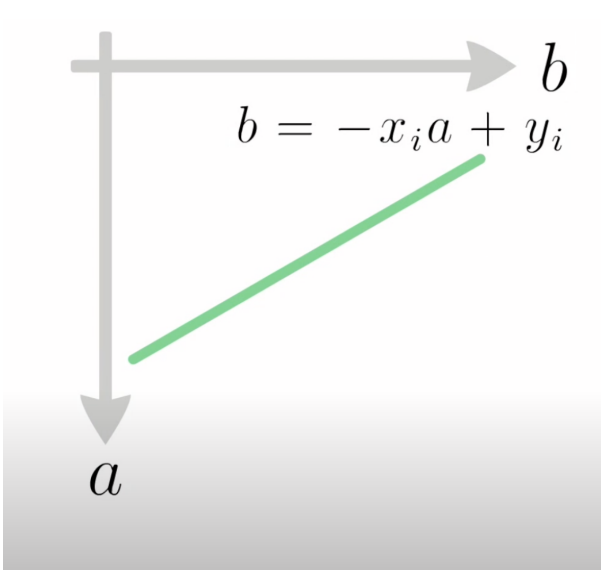
- ★ Coming to Template Matching, it is neither scale invariant nor rotation invariant which gives out the poor results. Using the logo directory as a set of kernels will also bring the same results because these methods follow cross correlation and convolution respectively which inherently follows the same methodology.

Colab Link : [Solution1_2A]

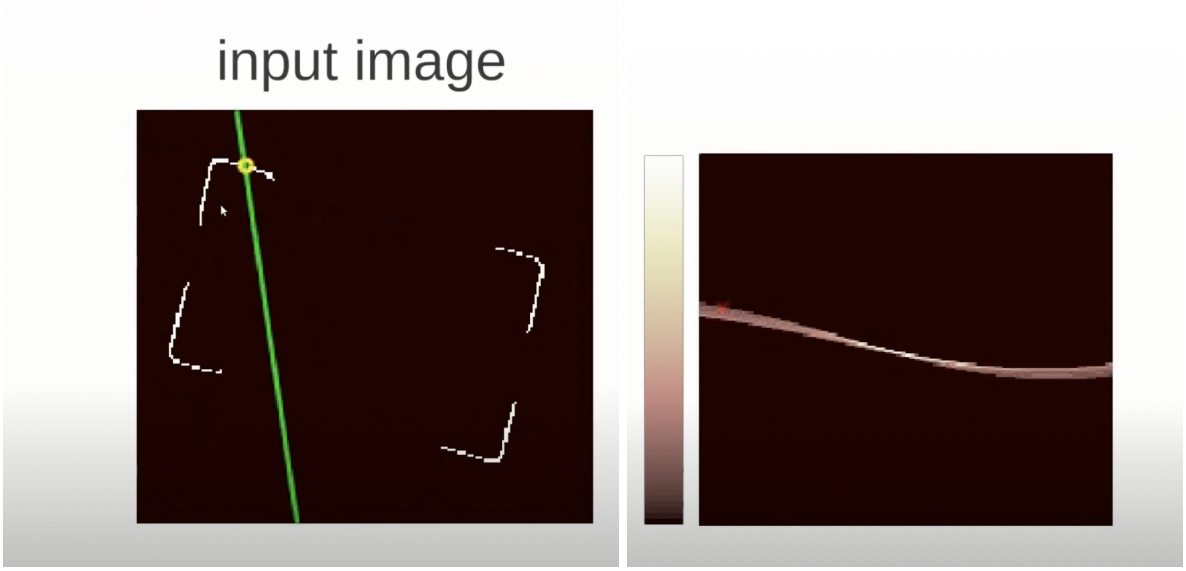
Question 2 : Implement Hough Transform for line detection from scratch. Compare the result of openCV implementation vs your implementation (both speed and performance wise) on a picture of your choice.

To implement Hough Transform for line detection, the Hough space method is used where we find the possible values of rho and theta in the plane, other than the original image plane.



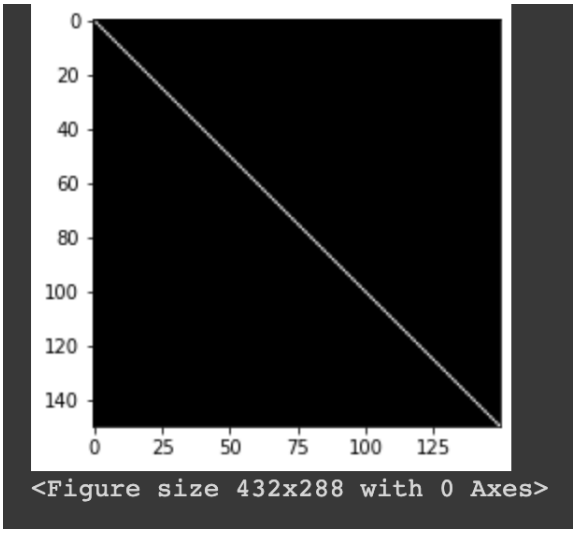


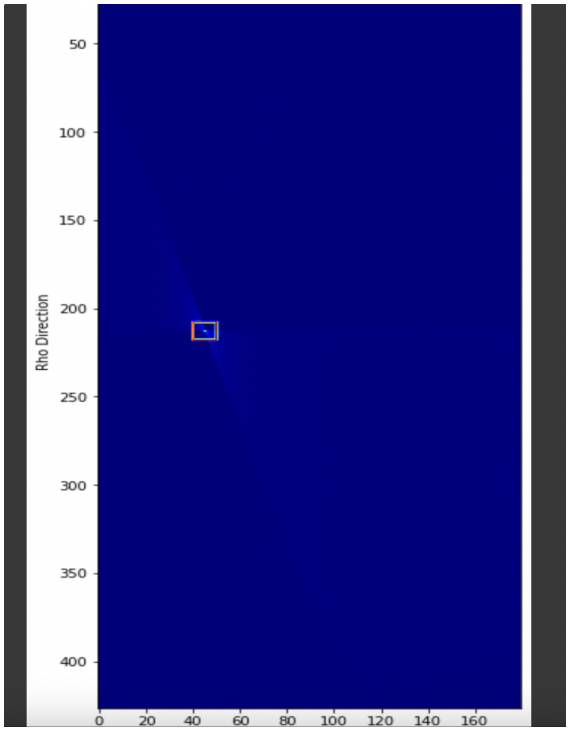
When there are values of x and y which could correspond to similar rho and theta value, then hough space would look like:-



Hough Space Visualisation for straight line in Coordinate plane.

Left Side image is the **line** drawn in the coordinate plane and **Right Side** image is the **hough space transformation** with possible rho and theta values.

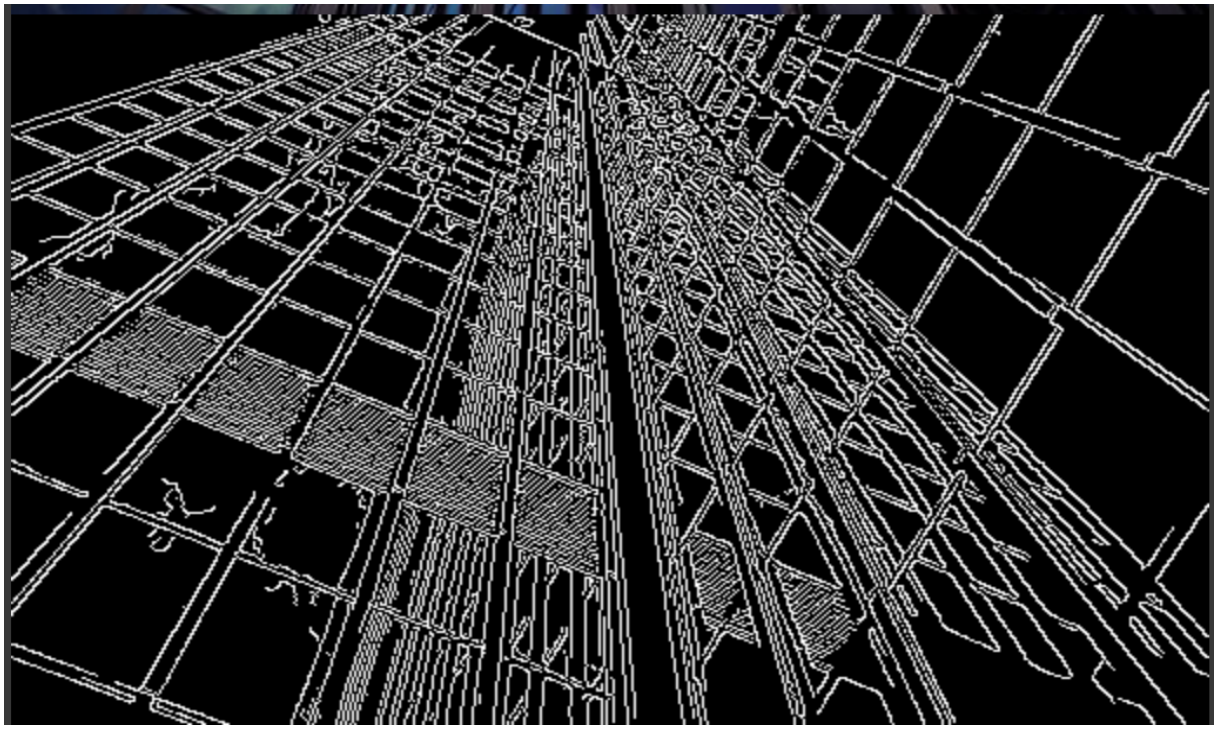




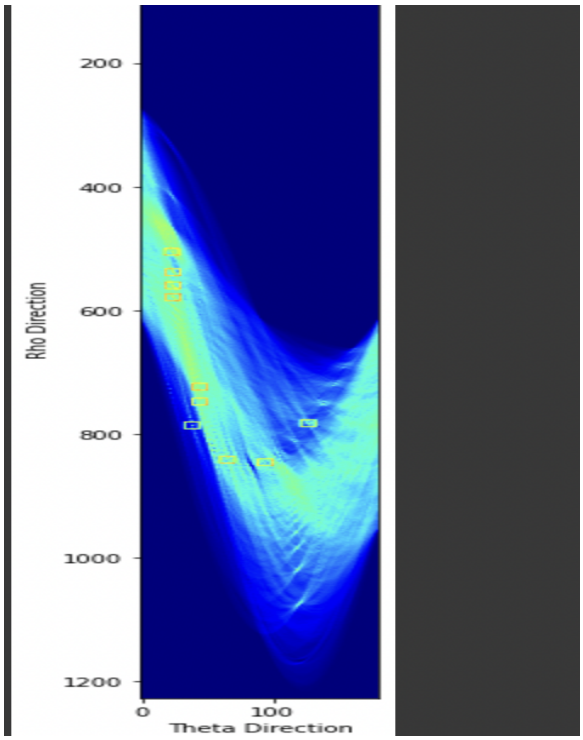
Now using scratch implementation of Hough on a Building image to detect lines.

Input image:-

Canny Edge Image with respect to input image :-



Transformation in Hough Space and Lines Detected using Scratch implementation :-



b. Show your results qualitatively in the report and write down your observations.

Both the implementations took input as Canny_Edge image. It was observed that the time taken by the OpenCV implementation is less compared with the scratch implementation.

Colab Link : Solution2_2A

Question 3 : A manufacturing company in Bangalore, came up with the following problem statement: They have one reference design image for a part of equipment and a probe image of either faulty or perfect. You need to identify faulty images and show the defective region.

Steps followed:-

1. Import the required libraries.
2. Define functions to find ORB and SIFT values.
3. Use ORB method to identify the key descriptors in the reference image and the input image. Once we get the required matches, we will use these points to create the homography matrix and furthermore, use warpPerspective to align and fit the input image according to the reference image.
4. The benchmark to find the Perfect image and Faulty image with the help of reference image is done by implementing two methods - ORB and SIFT. If the ORB and SIFT values are under a certain threshold then mark the image as Faulty else Perfect.
5. If the image is faulty then the two images are compared and the differences could be found between the two using the absolute difference method and allowing some threshold to highlight the differences.
6. The defects or so called differences are then dilated to be able to easily capture.
7. Use contours to mark the difference between the images.
8. Using the dilated image, rectangles are created over defects.

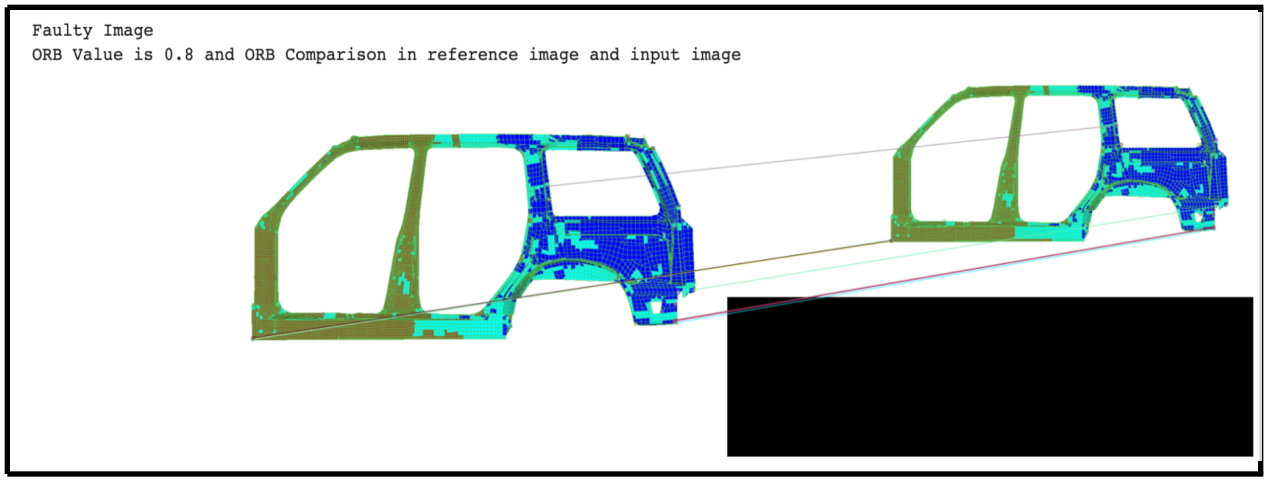
Example 1 Outputs:

Example 1 Input 1:-

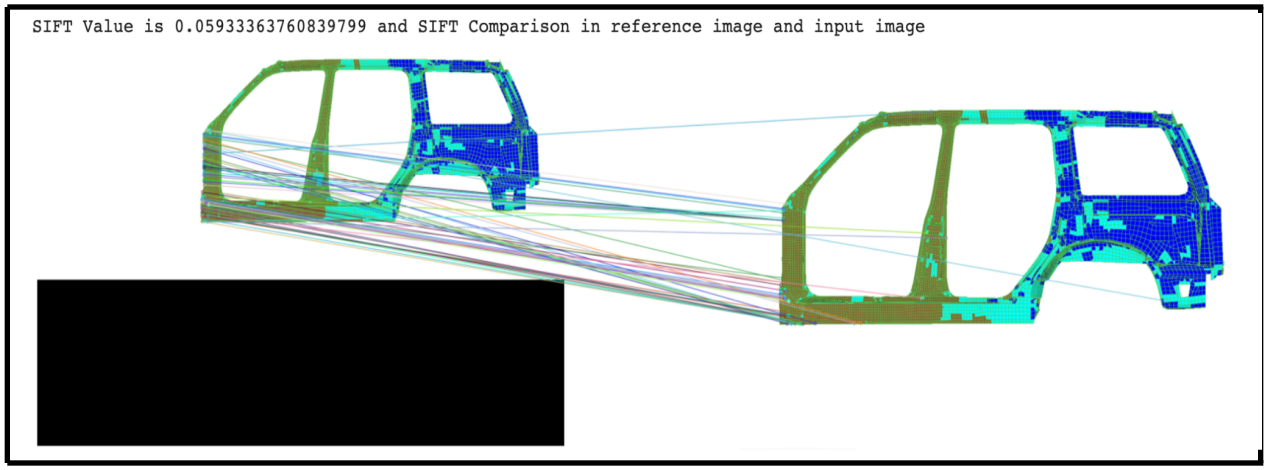
```
input = cv2.imread("Element_Optimised_Colour_ShapeLin.png")
reference_image = cv2.imread("reference.png")
```

Drawing Key Descriptors in the reference image and input image.

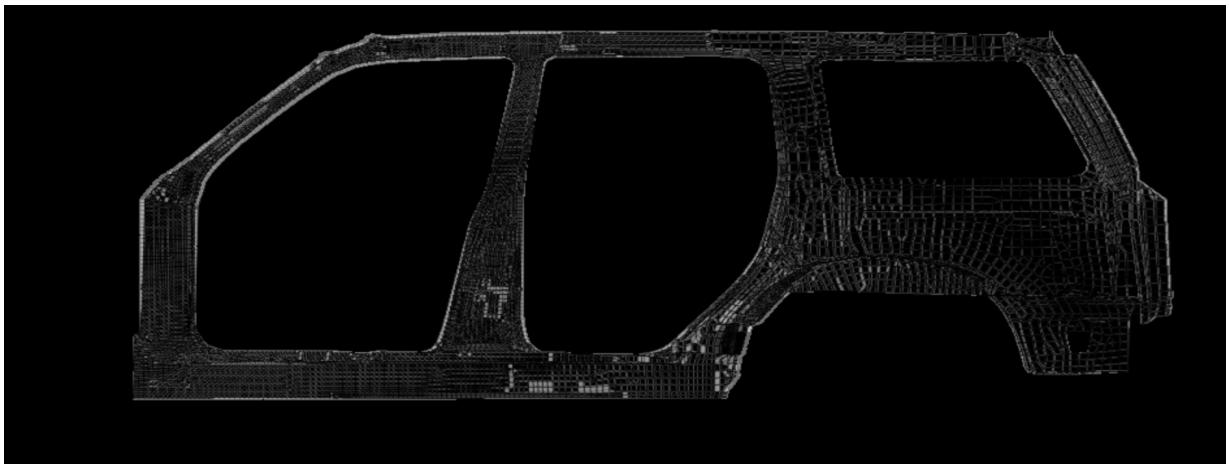
ORB Matching:-



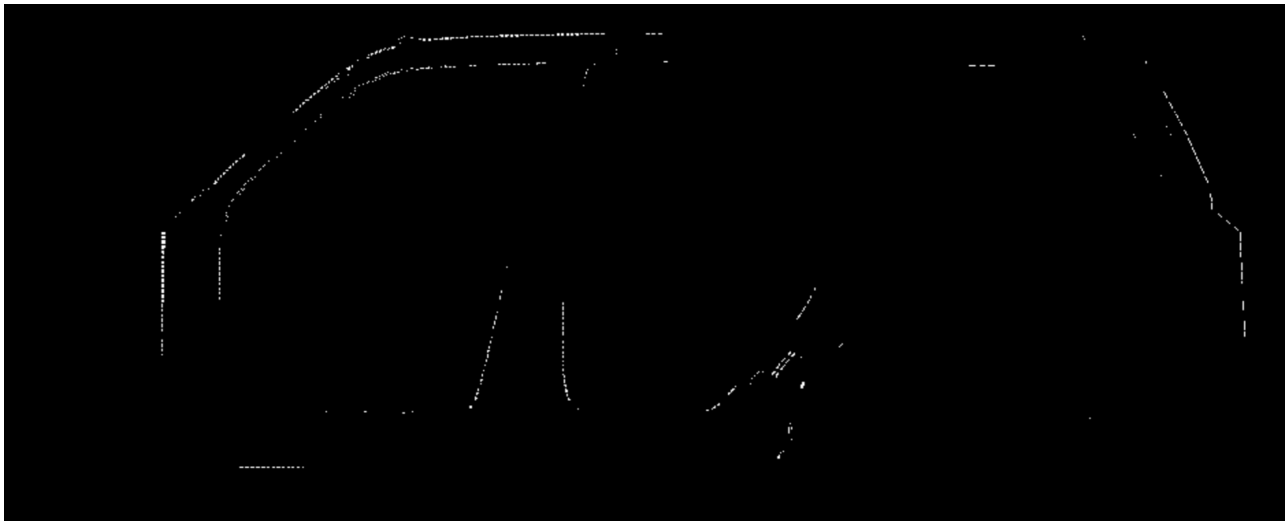
SIFT Matching:-



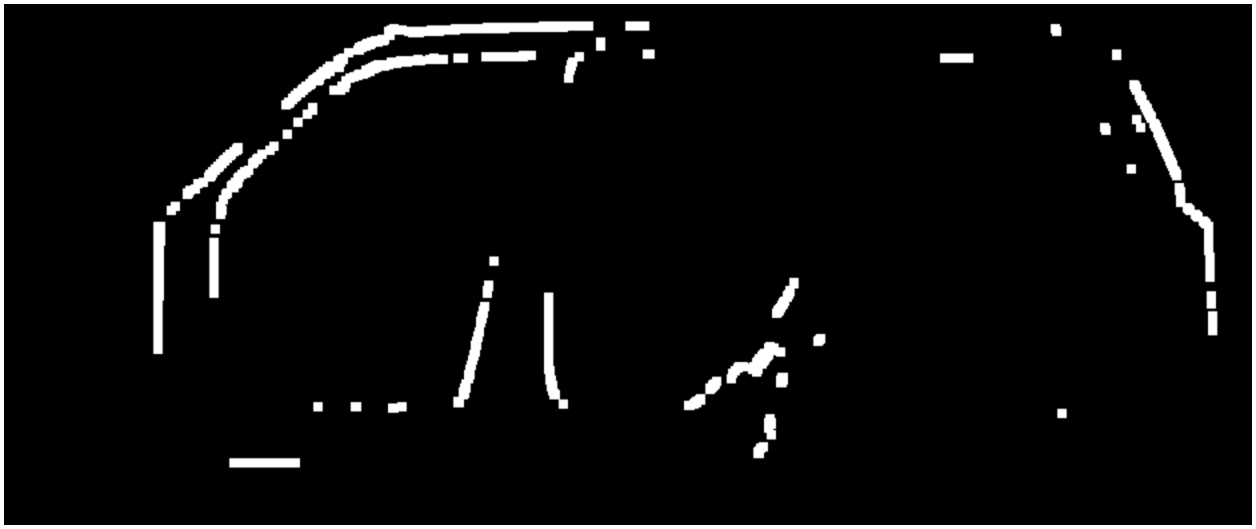
Difference between the two images after warpPerspective



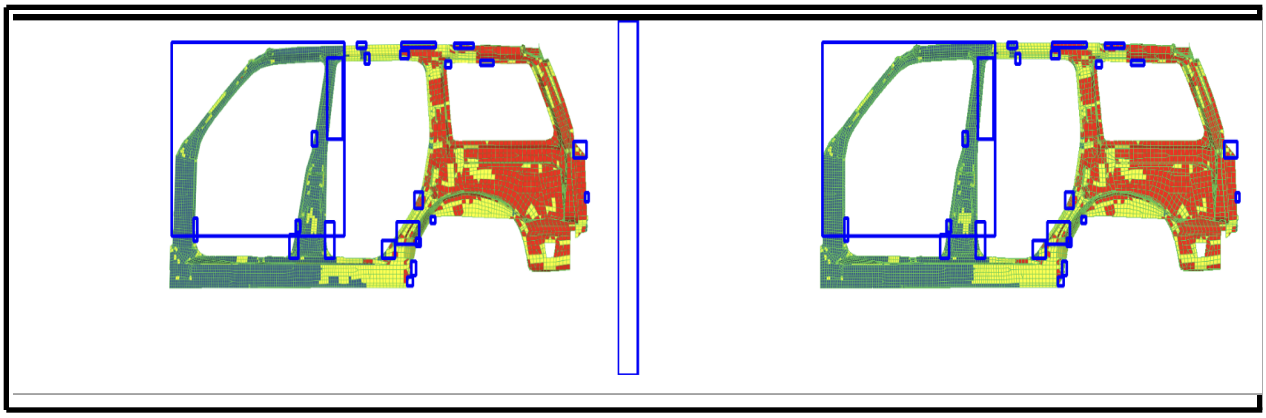
Binary image to reflect the defects:-



Dilation over the differences:-



Marking the defects with rectangles:-

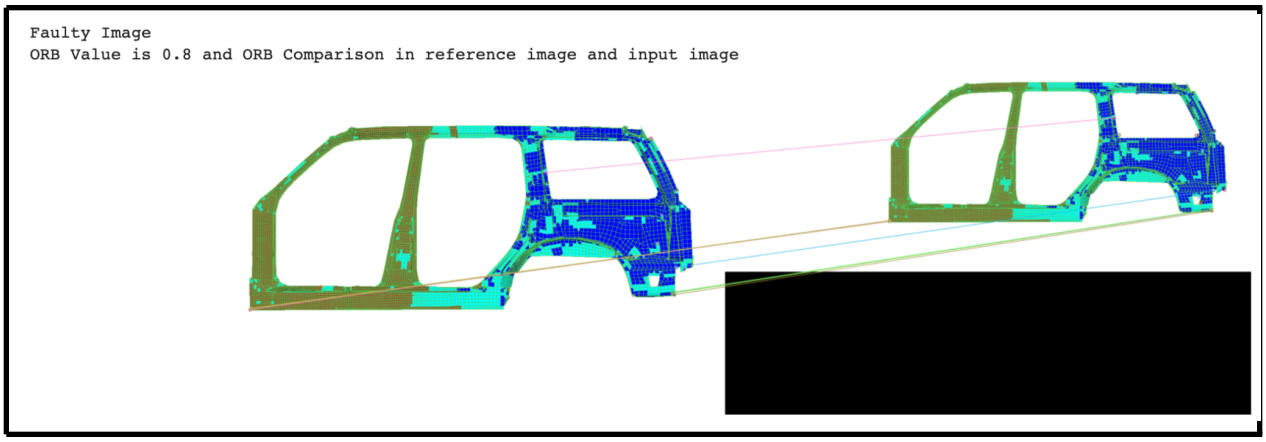


Example 1 Input 2:-

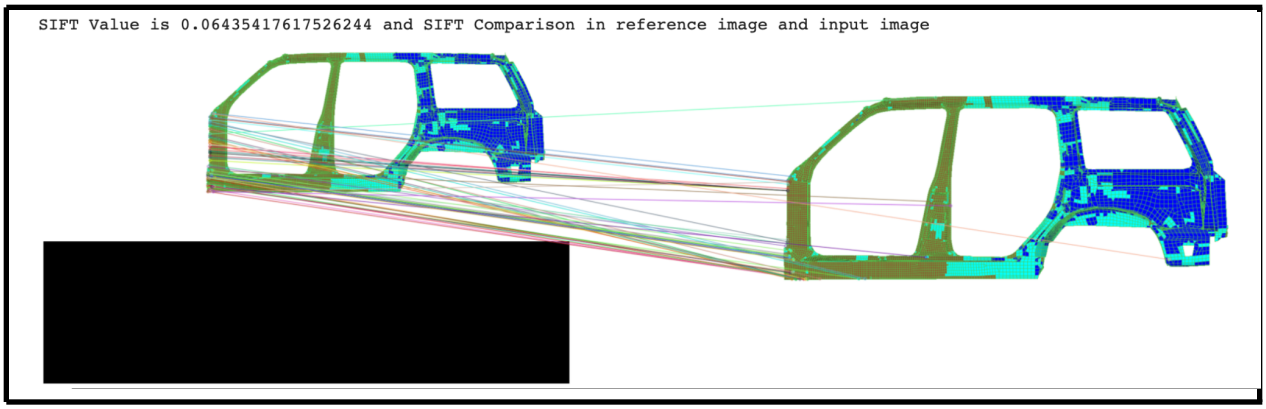
```
input = cv2.imread("MeshFlowLin.png")
reference_image = cv2.imread("reference.png")
```

Drawing Key Descriptors in the reference image and input image.

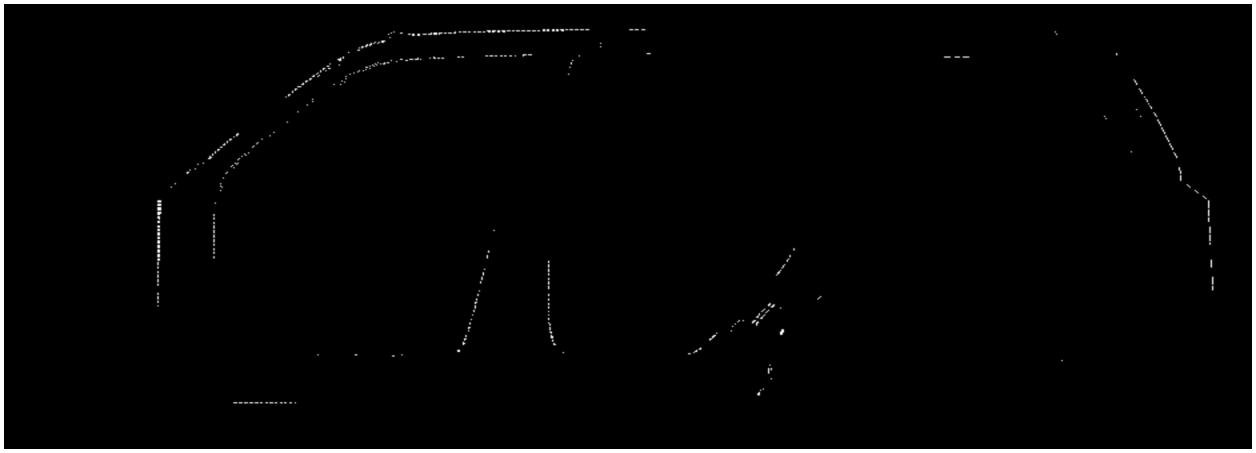
ORB Matching:-



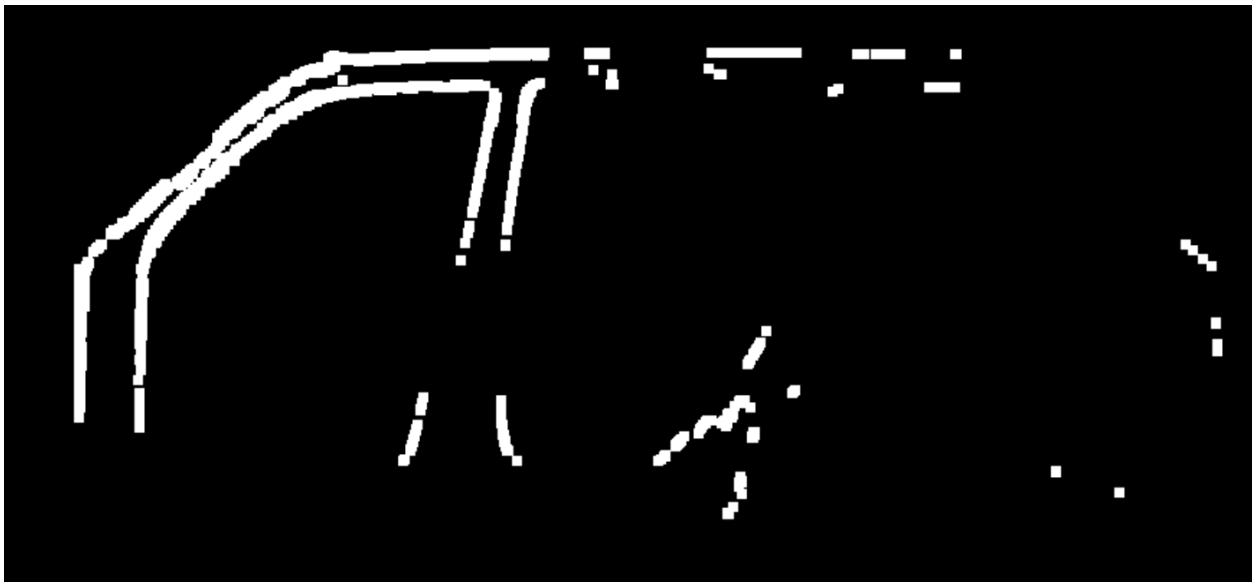
SIFT Matching:-



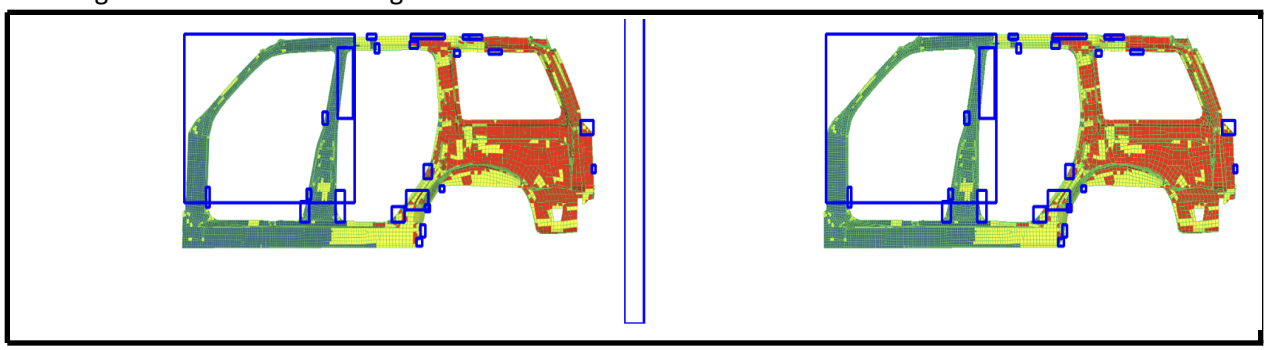
Binary image to reflect the defects:-



Dilation over the differences:-



Marking the defects with rectangles:-



Example 1 Input 3:-

```
input = cv2.imread("penetration_checkLin.png")
reference_image = cv2.imread("reference.png")
```

```
Perfect Image : ORB Value is 0.8 and SIFT Value is 0.06572341396622547
```

Example 1 Input 4:-

```
input = cv2.imread("penetration_checkLinLow.png")
reference_image = cv2.imread("reference.png")
```

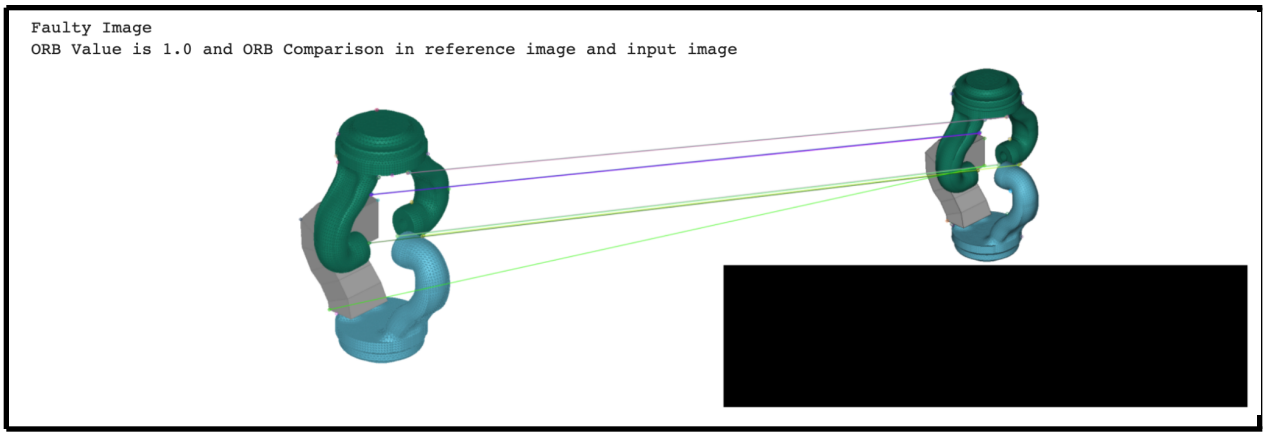
```
Perfect Image : ORB Value is 0.8 and SIFT Value is 0.010953902327704245
```

Example 2 Outputs:-

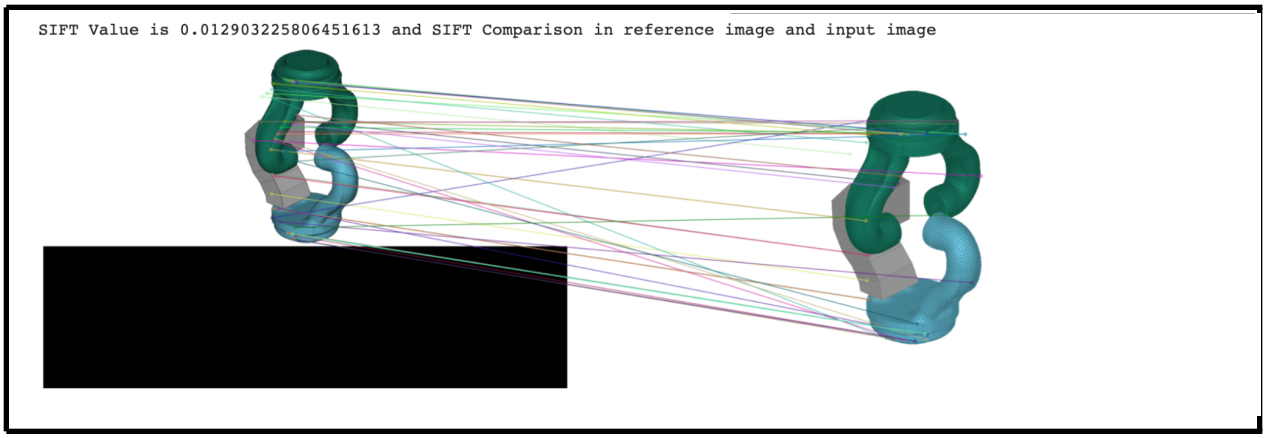
Example 2 Input 1:-

```
input = cv2.imread("2ndOrderElements.png")
reference_image = cv2.imread("Orginal.png")
```

ORB Matching:-



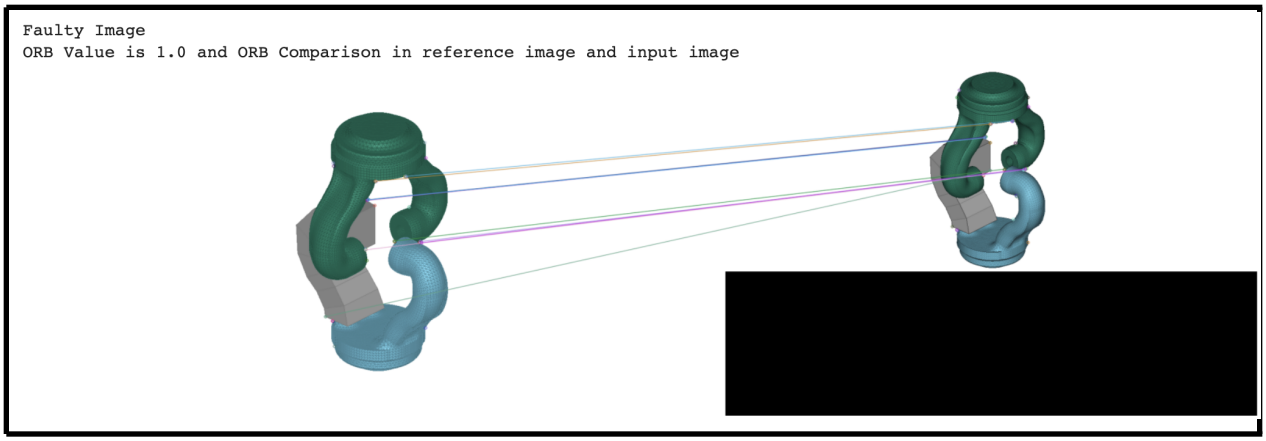
SIFT Matching:-



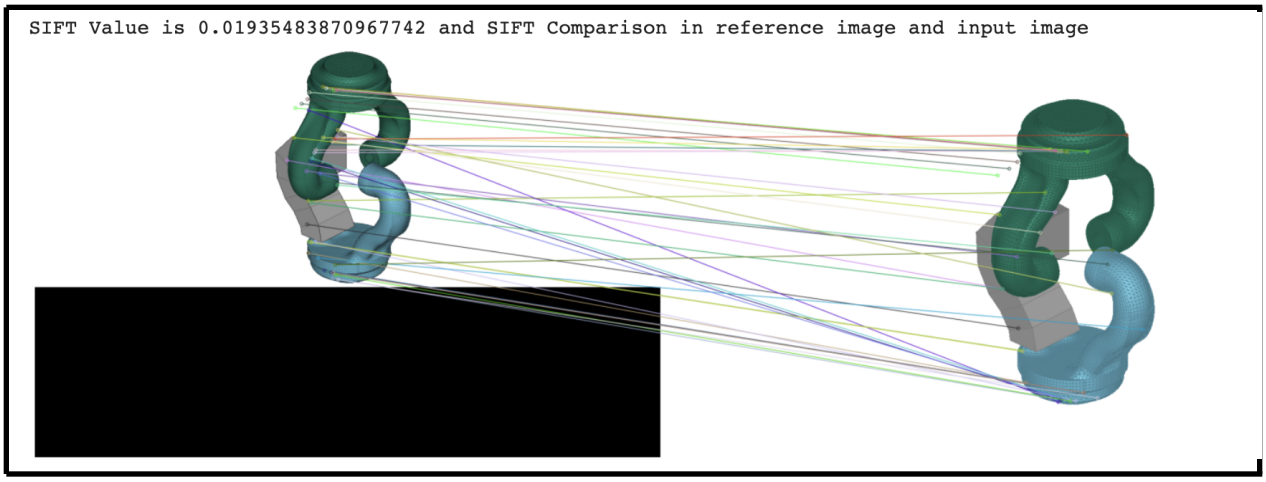
Example 2 Input 2:-

```
input = cv2.imread("Scaled.png")
reference_image = cv2.imread("Orginal.png")
```

ORB Matching:-



SIFT Matching:-



Example 2 Input 3:-

```
input = cv2.imread("Orginal_Lin_Low.png")
reference_image = cv2.imread("Orginal.png")
```

```
Perfect Image : ORB Value is 0.95 and SIFT Value is 0.0064516129032258064
```

Example 2 Input 4:-

```
input = cv2.imread("Orginal_Lin.png")
reference_image = cv2.imread("Orginal.png")
```

```
Perfect Image : ORB Value is 0.9545454545454546 and SIFT Value is 0.006451612903225806
```

Limitations to the approach:-

1. The approach is only useful when the images are sharp and so, warpPerspective can be effectively used to register the image.
2. The threshold value can either be automatically detected or it can be manually set (which could turn out to be a tedious task). When the threshold is automatically set, some of the defects are missed which have lower threshold values but larger surface area.

Colab Link : Solution3_2A

References:-

https://gist.github.com/bygreencn/6a900fd2ff5d0101473acbc3783c4d92
https://www.youtube.com/watch?v=FVBWeEh6Eg0

https://github.com/bnsreenu/python_for_microscopists/blob/master/030-keypoints_homography_for_registration%20in%20opencv.py
https://www.youtube.com/watch?v=Tm_7fGolVGE