CO 44301: DATA SCIENCE

Session: 2020-2021

Topic: Feature Extraction from Image

(Scale Invariant Feature Transformation (SIFT), reSIFT)

Submitted To:

Prof. Surendra Gupta

Submitted By:

Mrunali Patil

(0801CS171045)

Sarvesh Gupta

(0801CS171068)

Department of Computer Engineering

SHRI G. S. INSTITUTE OF TECHNOLOGY AND SCIENCE

Table of contents

Chapter 1

Introduction

1.1 What is a feature in an image?

Feature is a unit of the image which is unique for the object in that image. Feature can be any shape or distinct color in the image. A good feature is the characteristics of an object which can be used to distinguish objects from one another. In machine learning, the raw data (image) is transformed into 1 D array called feature vector which contains list of features. Feature plays an important role in areas like image processing.

1.2 Feature Extraction from image

Feature extraction is process of extracting the useful features that uniquely defines the objects present in the image. We will take an image and apply feature extraction algorithm on it which will generate a feature vector with the list of features. The main goal of feature extraction is to obtain the most relevant information from the original data(image). When the pre-processing has been done, some feature extraction technique is applied to the data(image) to obtain features, which is followed by application of classification and post processing techniques. There are various feature extraction algorithms such as Speeded up robust features (**SURF**), Scale-Invariant Feature Transform (SIFT), reSIFT, Binary feature extraction (BRIEF, BRISK) etc. In this paper we will be describing the SIFT and reSIFT algorithms for feature extraction from image.

1.3 SIFT

SIFT (Scale Invariant Feature Transformation) technique used for extracting image features. The features extracted using SIFT algorithm are invariant to image scaling, rotation, transformation it means the features should be detectable even in such transformations. Such points are called interest points or key points, they lie on some of the regions of an image. The SIFT algorithm is mainly divided into two modules. The features obtained by SIFT are local and robust. Many features can be generated for even small objects. They are key point detection Module and descriptor generation module.

The SIFT algorithm is mainly divided as four main steps:

1. Scale-space peak selection
2. Keypoint localization
3. Orientation Assignment
4. Keypoint descriptor

1.4 reSIFT

reSIFT (Reliability-Weighted-Scale Invariant Feature Transformation) is image quality estimator based on the SIFT descriptor over reliability features. Resift is used to quantify the perpetual quality of images under noise, compression, blur-based distortion and communication types. resift is the best performing quality estimator in noise, compression and blur in Pearson and Spearman correlation coefficients in case of the LIVE and the LIVE Multiply Distorted image quality databases. Resift is not designed for local, global, and color based distortions because the features extracted by it is all over the image and relies uniquely on lightless channel.

The reSIFT algorithm is mainly divided into four main steps in addition to SIFT implementation.

1. A smoothening operation
2. A color domain transformation
3. A normalization operation
4. A spectral residual calculation

.

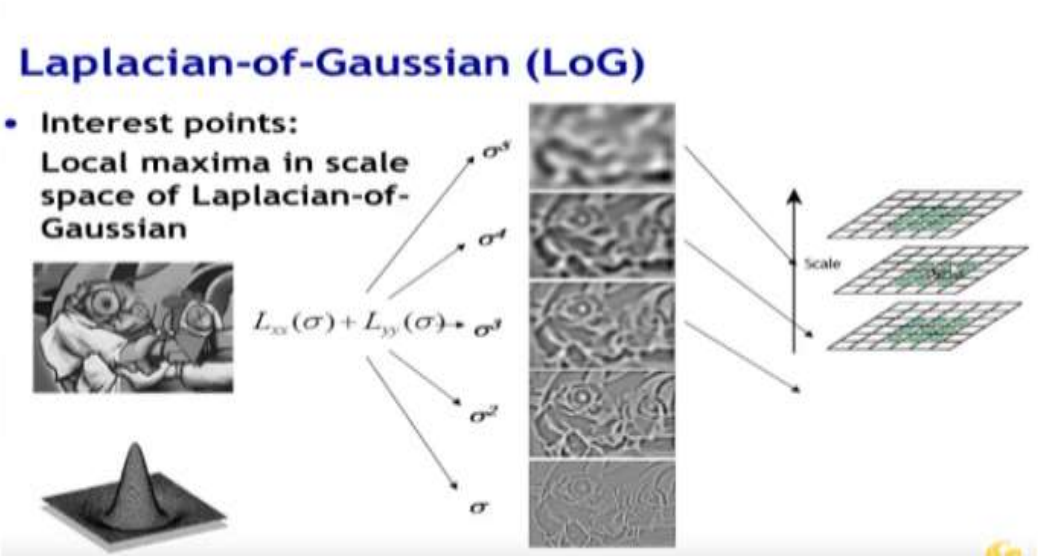
Chapter 2

Mathematical Formulation

2.1 SIFT

The algorithm consists of 4 main steps:

1. **Scale-space peak selection**
   Firstly, identify the key points in an image. For that create a scale space which is a space with features with all possible scales over Gaussian function. Apply Gaussian filter on image using different values of sigma.
   We will use Difference of Gaussian (DoG) in order to obtain the scale space. Take differences to get DoG pyramid (similar to Laplacian of Gaussian)



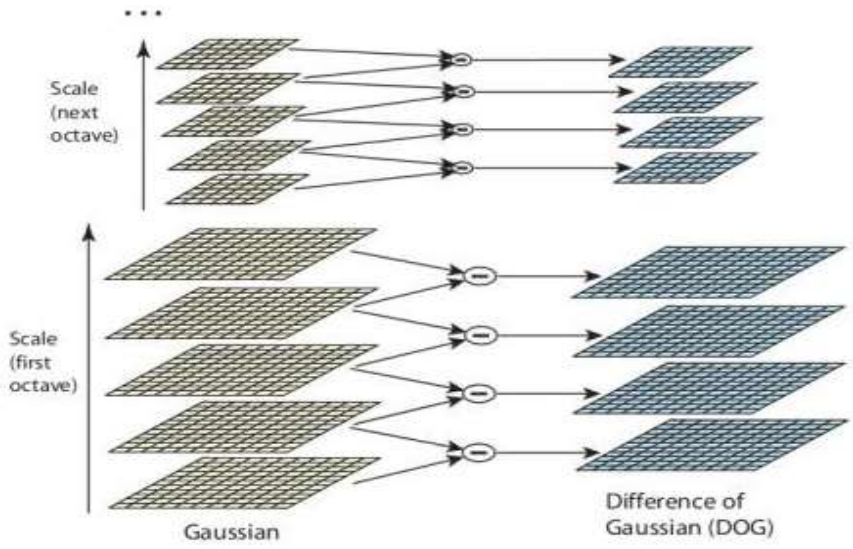
Use DoG in place of LoG:
1) DoG is LoG approximation method.
2) DoG has low computational complexity than LoG.
3) DoG does not need partial derivative computations as LoG.

SIFT uses no. of octaves to calculate DoG. Octave is a set of images where the blur of the last image is **double** the blur of the first image.

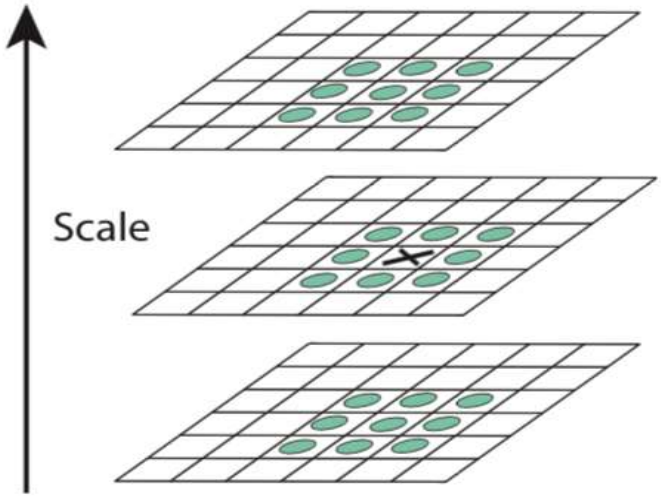
No. of images in an octave: s
Sigma for the Gaussian filter: 2^(1/s)
In each octave take: s+3 images
We can take Difference of Gaussian by first applying a Gaussian filter of varying sigmas (k sigma, k^2 sigma, k^3 sigma) and subtracting one from the other. This parameter is used to scale k so that the blur in each DoG octave goes from sigma = 2*sigma.

Value of sigma: 1.6
Value of k: 1.414

In DoG octave we get: s+2 images



Extrema dection in scale space.



1) 3X3 neighborhood is taken which surrounds a point.
2) 3X3 neighborhood is taken one scan above and below the point.
3) Amongst all the points excluding the original point. If the original pont is maxima or minima of all the other points then that point is considered as key point.

2. Keypoint localization

As in the previous step we found a lot of key points. So, we need to remove the low contrast points.

a) In many images the stable keypoints cannot be found because of their low resolution. So, SIFT use the second-order Taylor expansion of the DoG octave to further localize each keypoint.

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad \mathbf{x} = (x, y, \sigma)^T$$

The extrema (minima or maxima) of the equation is located in:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

The threshold value: 0.03
Reject all the key points with D(x)< threshold

b) We need to remove the keypoints on edges because of ambiguity. For this we use Hessian matrix of D, which is a matrix formed by getting the second derivative of D in x, y or xy

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$
$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \lambda_1\lambda_2$$

$$r = \frac{\lambda_1}{\lambda_2}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r+1)^2}{r}$$

3. Orientation Assignment

We have scale invariance. The next thing is to assign an orientation to each keypoint to make it rotation invariance.
Here, we will assign magnitude and direction to each keypoint

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Create a weighted direction histogram in each of those neighborhoods (with bins = 36), where weights are gradient magnitudes. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions.

4. Keypoint descriptor

At this point, each keypoint has a location, scale, orientation. Now we will find the feature vectors for every key point with the help of gradient orientation histograms. We can use this vector for different computer vision applications

2.2 reSIFT

1. Reliability-weighted SIFT descriptor extraction.

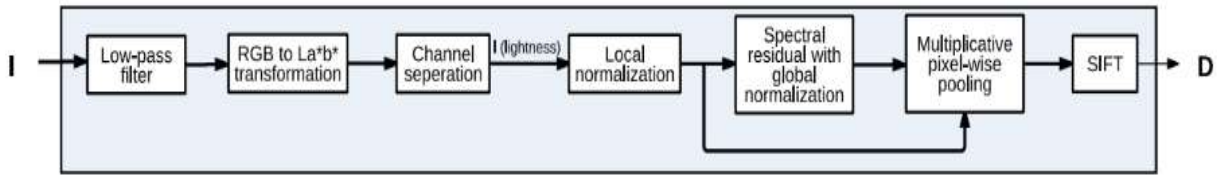


Fig. 1. Reliability-weighted SIFT descriptor extraction

1.1 Low pass filtering:

An input image is smoothened out with help of low pass filter. For this rotationally symmetric Gaussian low pass filter is used.

The cutoff frequency of low pass filter is function of :

i. Filter size (f size)
ii. Standard deviation (fσ)

1.2 Color space transformation

The smoothened image is transformed from RGB to perceptually uniform color space La*b* color domain in order to separate chroma information from lightness.

The transformation parameters:

a. Transformation matrix M
b. CIE standard coefficients κ and E.

Channel separation is done. Only lightness channel is futher taken to the next blocks and the chroma channels are not used due to lack of structural information.

1.3 Local normalization

Local normalization is performed over the lightness channel. performed by a mean subtraction and a divisive normalization operation.

The local mean formulation is given as:

$$\mu[m,n] = \frac{1}{W^2} \sum_{\hat{m}=floor(\frac{m}{W})\cdot W+1}^{(floor(\frac{m}{W})+1)\cdot W} \sum_{\hat{n}=floor(\frac{n}{W})\cdot W+1}^{(floor(\frac{n}{W})+1)\cdot W} l[\hat{m},\hat{n}], \quad (1)$$

where, **m and n** = pixel indices

l = lightness channel

W = window size

floor = an operator that rounds a no. to the next smaller int.

The standard deviation of each window is calculated is:

$$\sigma[m,n] = \sqrt{\sum_{\hat{m}=floor(\frac{m}{W})\cdot W+1}^{(floor(\frac{m}{W})+1)\cdot W} \sum_{\hat{n}=floor(\frac{n}{W})\cdot W+1}^{(floor(\frac{n}{W})+1)\cdot W} \frac{(l[\hat{m},\hat{n}]-\mu[\hat{m},\hat{n}])^2}{W^2}}, \quad (2)$$

The local normalization operation is composed of two steps.

First, a local mean (μrm, ns) is subtracted from each pixel in the lightness map. Second, each mean shifted value is divided by a local standard deviation (σrm, ns).

1.4 Spectral residual with global normalization

The normalized lightness map (lnorm) is transformed from the spatial domain to the frequency domain using the Fourier transform (F).

The magnitude component of transformed map :

$$|L| = |(\mathcal{F}[l_{norm}])|, \quad (3)$$

where, |'| is the magnitude operator.

The phase component is given as :

$$\angle L = \angle(\mathcal{F}[l_{norm}]), \quad (4)$$

where, <' is the phase operator.

The spectrum of a signal is computed by the log of the magnitude (log |L|).

The average spectrum is computed by convolving the spectrum with an averaging filter (g).

The difference between the spectrum and the averaged spectrum results in the spectral residual, which is calculated as:

$$SR(L) = \log|L| - g * \log|L|, \quad (5)$$

where, * = the convolution operator

g = the averaging filter.

The spectral residual is combined with the phase of the normalized lightness map, then an inverse Fourier transform is performed. Reconstructed image is used as a saliency map, which is referred as the unexpected portion of an image.

A reconstruction operation using the spectral residual is calculated as:

$$S = h * \mathcal{F}^{-1}[SR(L)\angle L], \quad (6)$$

where $\mathcal{F}$^-1 = the inverse Fourier transform operator

h = Gaussian low-pass filter used to smooth out a reconstructed map

['<'] = the representation of a signal in terms of its magnitude and its phase.

The reconstructed map is globally normalized so that the pixel values are in between 0.0 and 1.0.

1.5 Multiplicative pixel-wise pooling

The locally normalized lightness map and the spectral residual based reconstructed map are multiplicatively pooled pixel wise to obtain a reliability-weighted lightness map.

1.6 Scale-invariant Feature Transform

It takes an image as an input and output SIFT feature vector. All the steps for this algorithm is same as mentioned in the SIFT algorithm. SIFT descriptors are extracted over the reliability-weighted lightness maps.

Section/Block	Parameter	Value
2.2. Low-pass filtering	f_{size}	4
	f_σ	5
2.3. Color space transformation	κ	903.3 CIE standard
	ϵ	0.008856
	M	Adobe RGB stand. 1998
2.4. Local normalization	W	20
2.5. Spectral residual	g_{size}	3
	h_{size}	10
	h_σ	3.8

Chapter 3

Algorithms

3.1 SIFT Algorithm

1. **Input**: a preprocessed image.
2. **Scale-space peak selection**
   - 2.1 Calculate DoG
     - 2.1.1 No. of images in each octave : s (each octave will contain s+3 images)
     - 2.1.2 Sigma for Gaussian filter : 2^(1/s)
     - 2.1.3 Apply Gaussian filter with varying sigmas
     - 2.1.4 Subtract one from another(get s+2 images in Dog octave)
     - 2.1.5 DoG octave formed makes up DoG pyramid
   - 2.2 Extrema detection
     - 2.2.1 Each point in DoG compared with neighboring point.
     - 2.2.2 Compare with one scale above and one scale below.
     - 2.2.3 Keypoints are points that are maxima or minima(results in keypoints)
3. **Keypoint Localization**
   - 3.1 Remove keypoints of low contrast points.
     - 3.1.1 With help of 2^{nd} order Tylor series expansion of D.
   - 3.2 Remove ambiguous keypoints on the edges.
     - 3.2.1 with the help of Hessian matrix.
4. **Orientation Assignment**
   - 4.1 Calculate magnitude and direction of points.
   - 4.2 Create weighted direction histogram with bins=36
   - 4.3 Highest peak and peak above 80% is considered (keypoints found with different directions)
5. **Local image descriptor**
   - 5.1 Take 16x16 neighborhood around key point.
   - 5.2 Get relative orientation of it wrt keypoint.
   - 5.3 Divide 16x16 neighborhood into 4x4 blocks, total 16 blocks.
   - 5.4 Find weighted direction histogram of each block(creates 16 histograms of bins = 8)
   - 5.5 Concatenate these 16 histograms in one long vector of 128 dims.
   - 5.6 Normalize it to unit vector.
   - 5.7 Remove larger gradients from unit vector.
6. **Output:** feature vector and image depicting the features.

3.2 reSIFT Algorithm

1. **Input**: an image
2. **A smoothening operation**
   2.1 image is smoothened out using a rotationally symmetric Gaussian low pass filter.
3. **A color domain transformation**
   3.1 Transform the RGB into perpetually uniform color space La*b*(separate lightness from chroma information)
   3.2 The lightness channel is transferred further.
4. **A normalization operation**
   4.1 calculate local mean: μ[m,n]
   4.2 calculate standard deviation : σ[m,n]
   4.3 subtract local mean from each pixel in lightness map.
   4.4 Divide each mean shifted value by standard deviation.
5. **A spectral residual calculation**
   5.1 Transform normalized lightness map from special domain to frequency domain.
   5.2 Magnitude component of transformed map is calculated
   5.3 Phase component is calculated.
   5.4 Spectral residual is calculated as difference between the spectrum and the averaged spectrum.
   5.5 Combine the spectral residual with phase of normalized lightness map.
   5.6 Perform fourier transform.
   5.7 A reconstruction operation is done using the spectral residual.
   5.8 The reconstructed map is globally normalized so that the pixel values are in between 0.0 and 1.0.
   5.9 Multiplicatively pool pixel-wise the locally normalized lightness map and the spectral residual based reconstructed map to obtain a reliability-weighted lightness map.
6. **SIFT algorithm**
   Extract the SIFT descriptors over the reliability-weighted lightness maps.
7. **Output**: Features extracted

Chapter 4

Documentation of API

Firstly, the image preprocessing is done. For this, training and testing image is used.

the **libs.py** file is used in which imports all the files/ libraries that are being used in our code**.**

For input image and image preprocessing-

Variables:

- image - an image is taken as an input
- training_image : training_image to RGB
- training_gray : training image to gray scale
- test_image : adding Scale Invariance and Rotational Invariance
- test_gray : test_image to gray scale

function: make_test_image

- parameters:

1. training_image: the training image is passed to this function for adding scale invariance and rotational invariance to it.

- attributes:

1. test_image: the image formed after applying gaussian filter and creating gaussian pyramid. Gaussian pyramid involves applying repeated Gaussian blurring and downsampling an image until some stopping criteria are met.
2. rotation_matrix: the rotation matrix is created.
3. test_gray: test image that is converted to gray scale.

- return attribute:

1. test_image
2. test_gray

Implementing the SIFT algorithm

for this there is **sift.py** file that contains the code for sift algorithm.

Sift.py contains:

function: SIFT_algo

- parameters:

1. training_image
2. training_gray
3. test_image
4. test_gray

- attributes:

1. sift - sift object is created.
2. train_keypoints - Detects keypoints in the training image
3. train_descriptor - Computes the descriptors in the training image
4. test_keypoints - Detects keypoints in the training image
5. test_descriptor - Computes the descriptors in the training image
6. keypoints_without_size - return array copy of training image
7. keypoints_with_size - return array copy of training image

- return attributes:

1. keypoints_with_size
2. keypoints_without_size
3. train_descriptor
4. train_keypoints
5. test_descriptor
6. test_keypoints

Then the keypoints are depicted with the help of plots in the traning image and the testing image with or without the size of features/key points.

Perform the matching between the SIFT descriptors of the training image and the test image.

image_matching.py file contains the code for image matching between training and testing image.

function: image_matching

- parameters:

1.train_descriptor

2.train_keypoints

3.training_image

4.test_descriptor

5.test_keypoints

6.test_gray

- attributes:

1.bf - creating brute force matching object.

2.matches - Perform the matching between the SIFT descriptors of the training image and the test image

3.result - The matches with shorter distance are the ones we want.

- return attributes:

1.matches

2.result

Now, Display the best matching points

And Print total number of matching points between the training and query images

Implementing the reSIFT algorithm

for this there is **resift.py** file that contains the code for resift algorithm.

resift.py contains:

functions:

1.Smoothening_Transformation

- parameters:

1. training_image

- attributes:

1. smoothing- Gaussian smoothing the image using Gaussian Low pass filter
2. lab- moving image in Lab mode

- return attributes:

1. lab

Now, Original image and La*b* image is displayed.

2. Normalise_ResidualCalc

- parameters:

1. lab

- attributes:

1. L, A, B - split La*b* color space
2. float_lumn - Local normalization of luminance channel
3. blur - apply gaussian blur
4. num - float_lumn - blur
5. den
6. luminance - num / den
7. dft - fourier spectrum
8. dft_shift - Shift the zero-frequency component to the center of the spectrum.
9. mag - magnitude
10. ang - phase

11. c - 255 / np.log(1 + np.max(mag))
12. log_image - c * (np.log(mag + 1)), logarithmic image
13. smooth_log_image - smoothened log image
14. residual - residual calculated
15. x, y - polar to cartesian plane coordinates
16. rows, cols - rows and columns
17. crow,ccol
18. mask - mask applied
19. fshift - dft*mask
20. f_ishift
21. img_back - Calculates the inverse Discrete Fourier Transform
22. smooth_img_back - Applying Gaussian LPF and normalize
23. pool_image - Multiplicative pixel wise pooling

- return attributes:

1. pool_image

Output of Normalise_ResidualCalc() function is given instead of training gray as reliability-weighted lightness map decide the descriptor

Image with and without keypoints size is displayed.

the number of keypoints detected in the training image is printed

the number of keypoints detected in the query image is printed

Perform the matching between the reSIFT descriptors of the training image and the test image.

image_matching.py file contains the code for image matching between trating and testing image.

as discussed above.

Now, Display the best matching points

And Print total number of matching points between the training and query images.

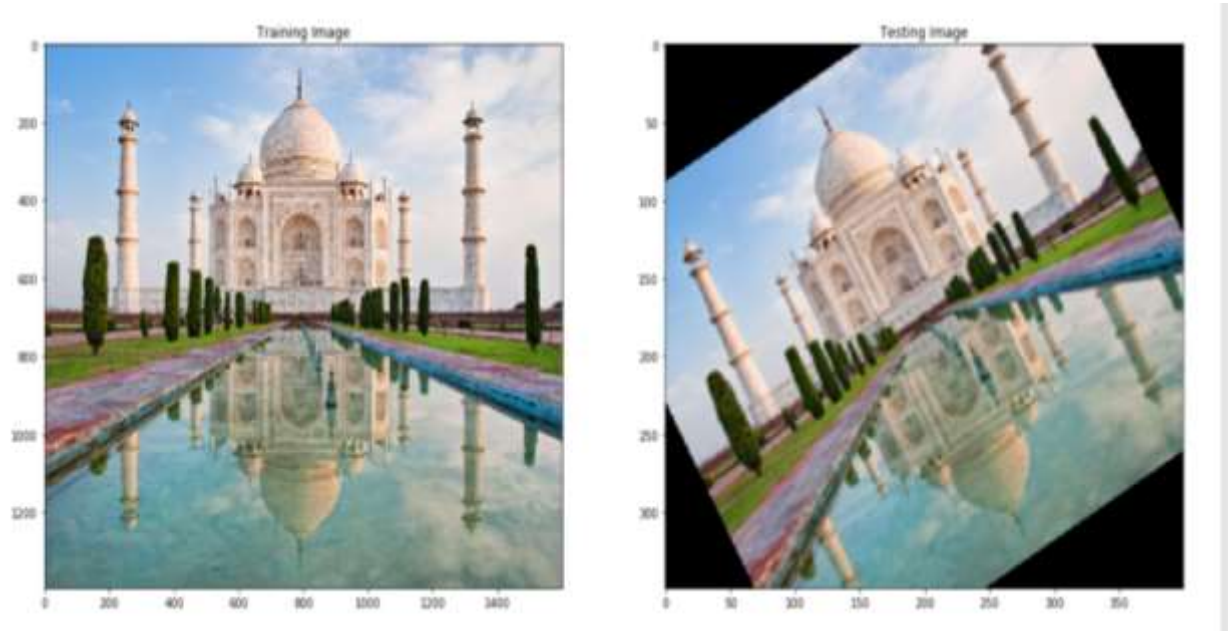
Chapter 5

Examples

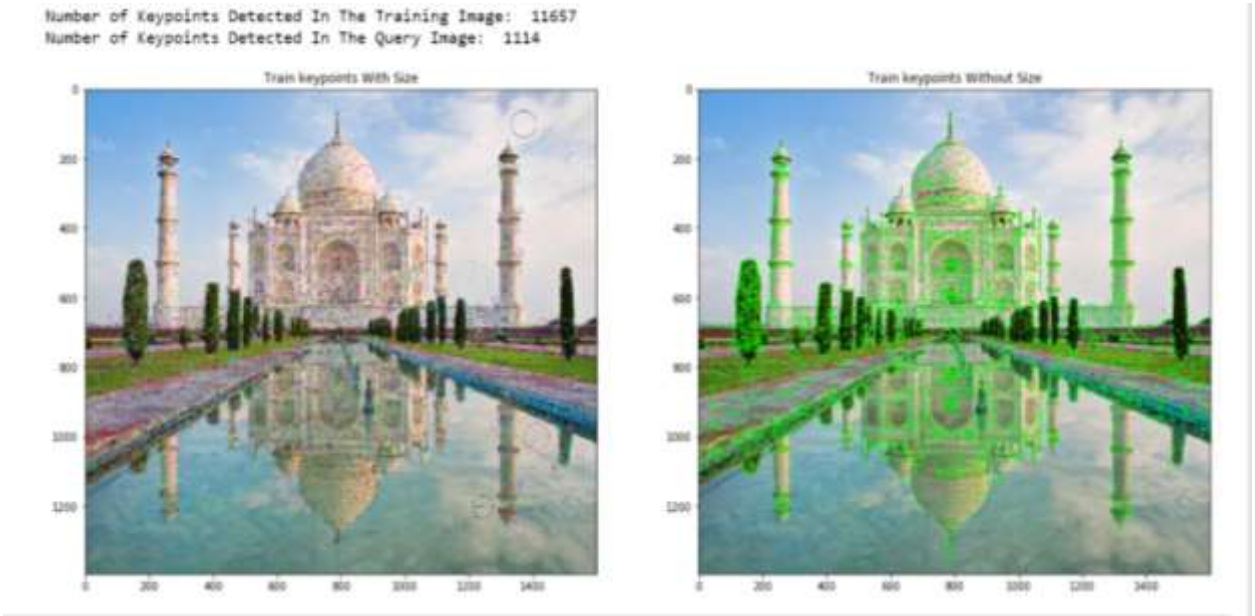
5.1 Example 1

Sift:

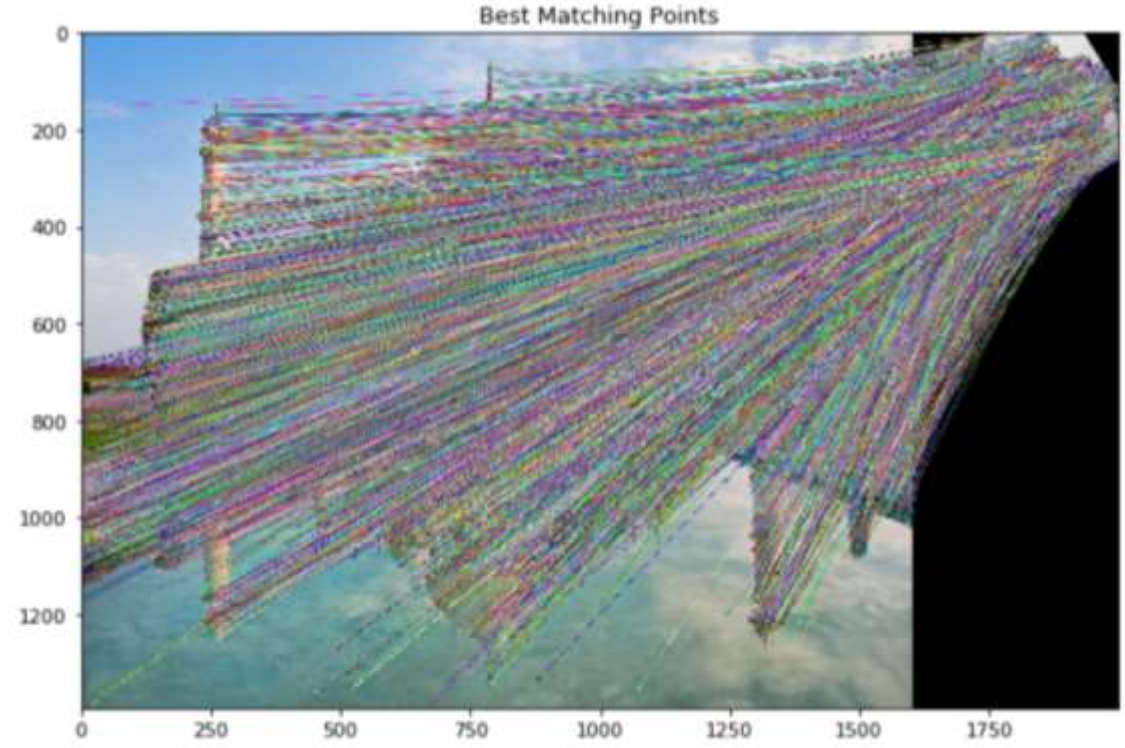
1.Training image (scale and rotation invariance):-



2. Keypoints detected:



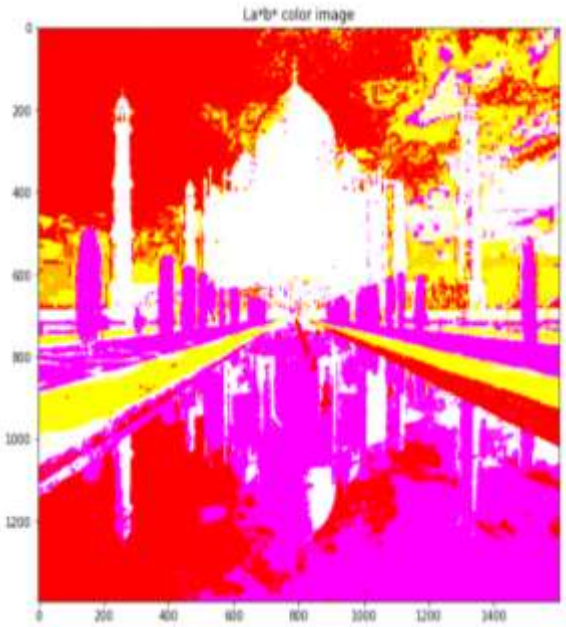
3. Feature matching performed:




```
Number of Matching Keypoints Between The Training and Query Images:  11657
```

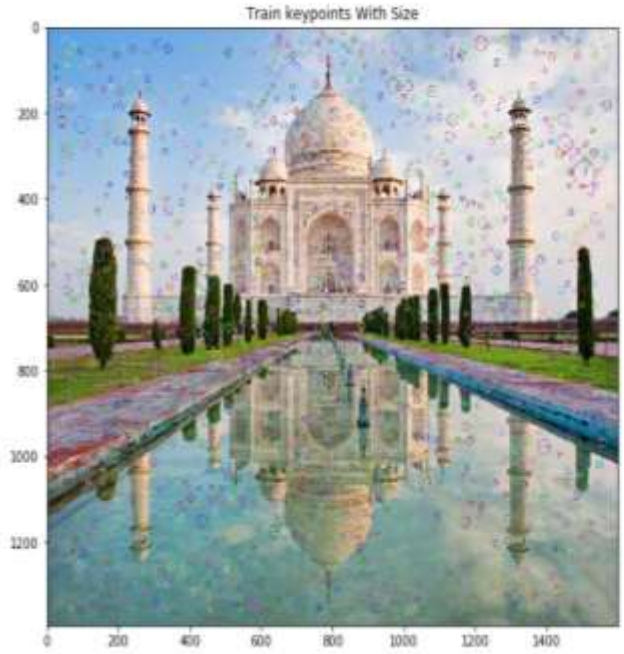
Resift:

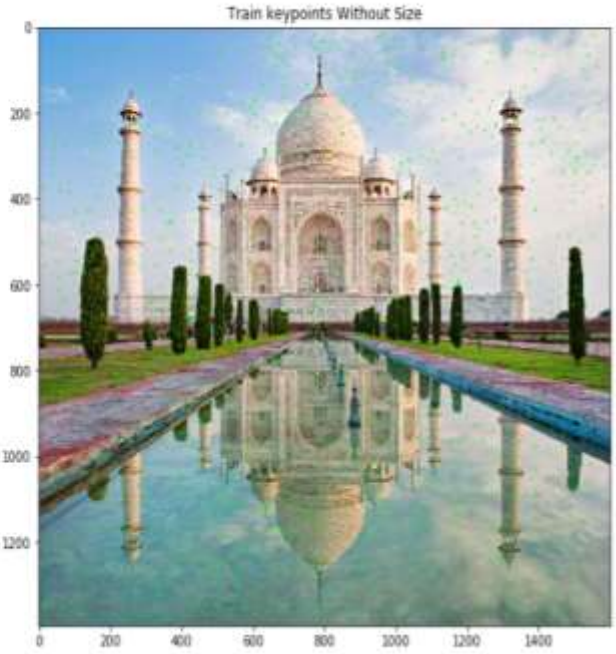
1. Smoothening image:



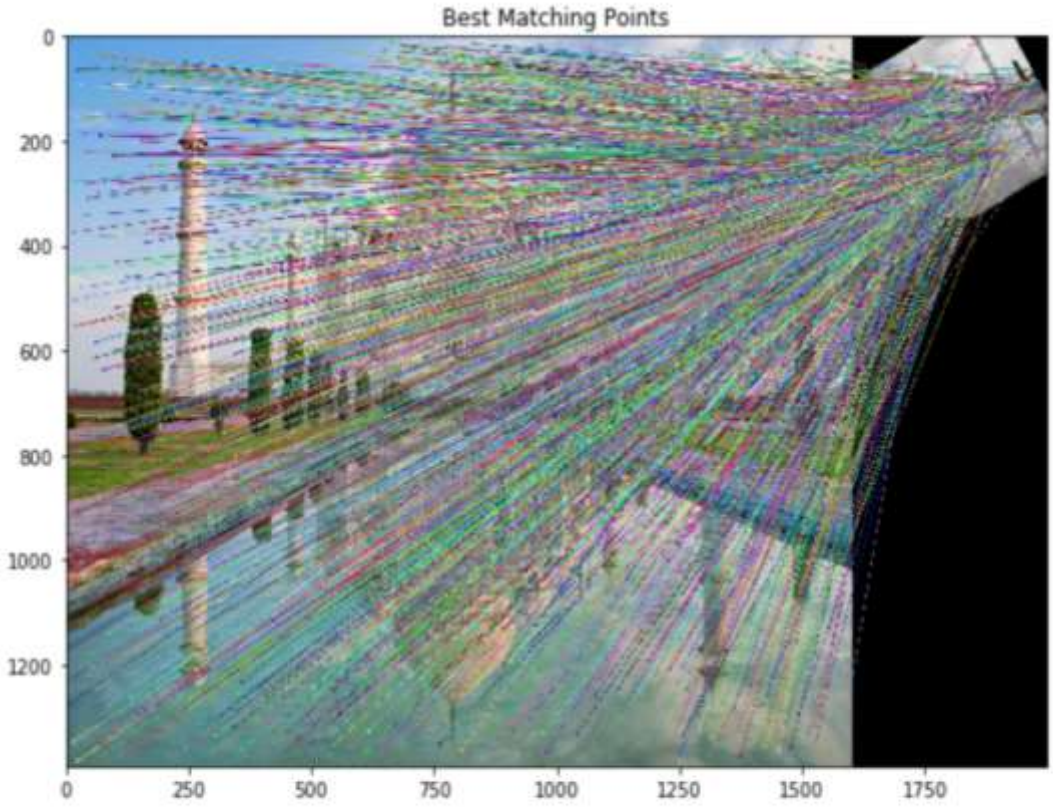
2. Keypoints detected:

```
Number of Keypoints Detected In The Training Image:  1852
Number of Keypoints Detected In The Query Image:  1114
```





3.Feature matching performed:




```
Number of Matching Keypoints Between The Training and Query Images:  1852
```

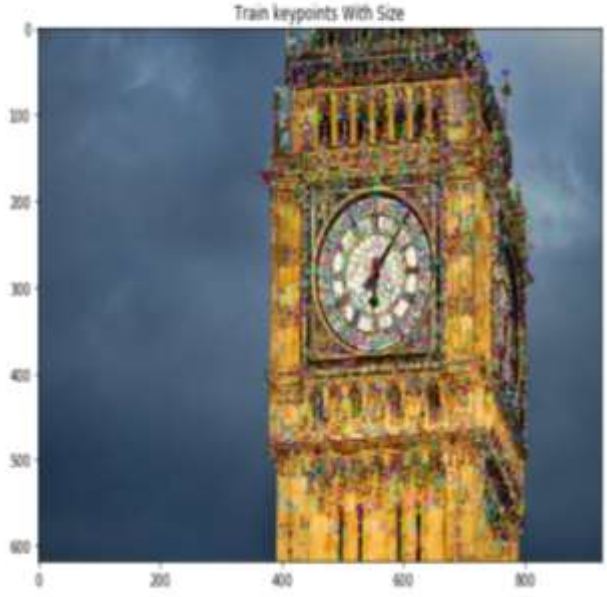
5.2 Example 2

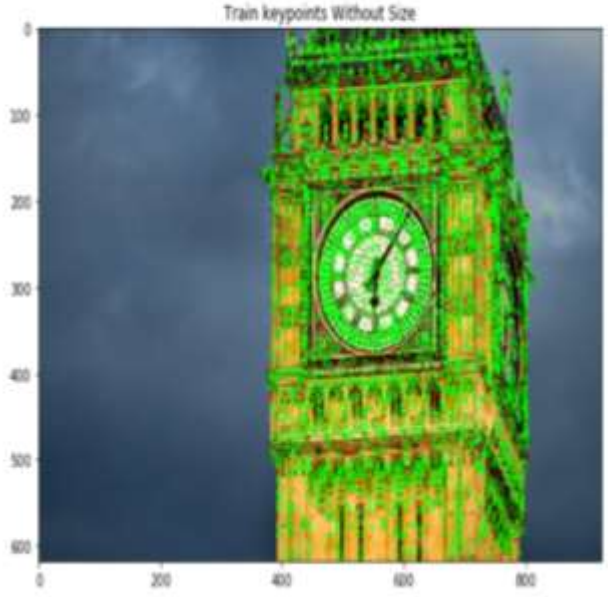
Sift:

1.Training image (scale and rotation invariance):

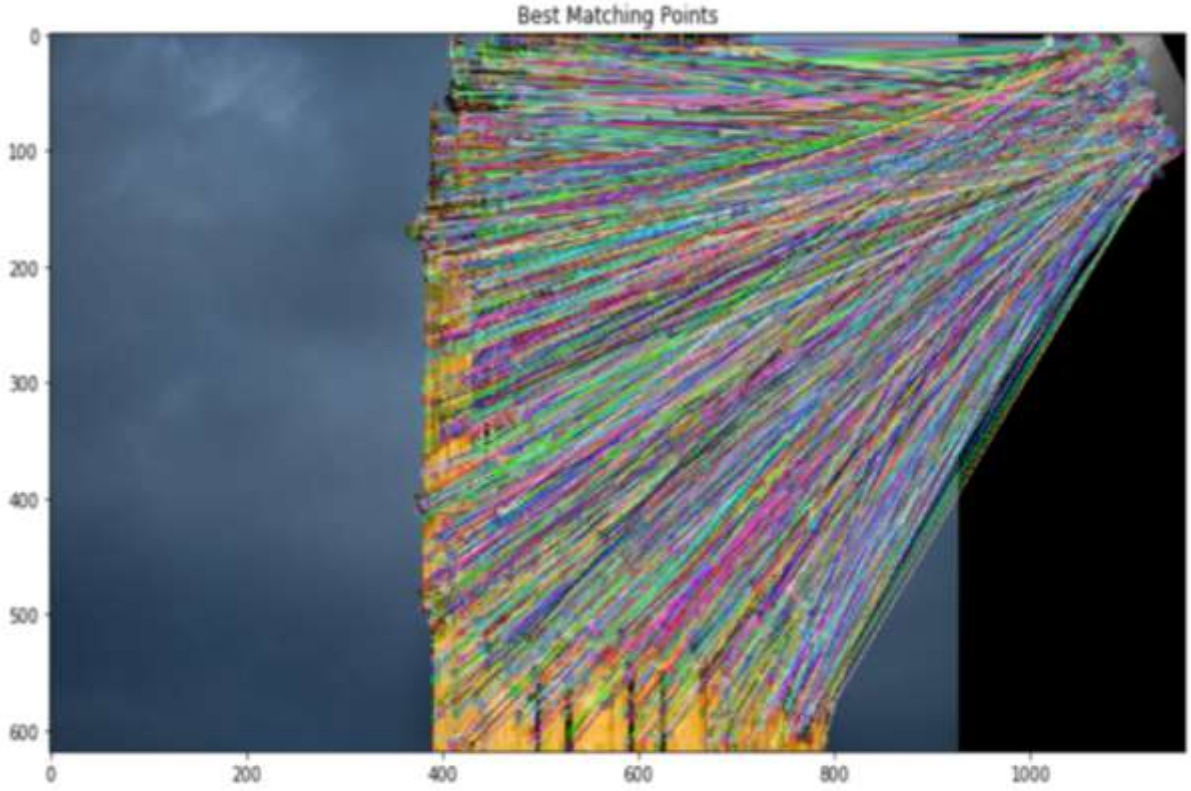
2.Keypoints detected:

```
Number of Keypoints Detected In The Training Image:  5000
Number of Keypoints Detected In The Query Image:  416
```





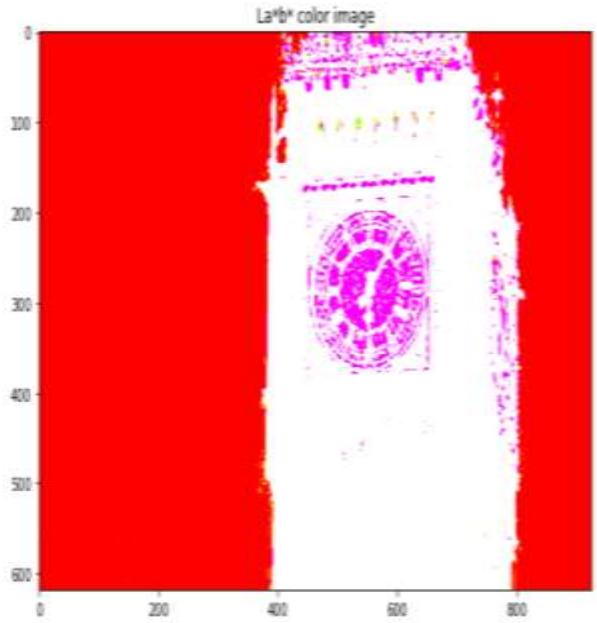
3.Feature matching performed:




```
Number of Matching Keypoints Between The Training and Query Images:  5000
```

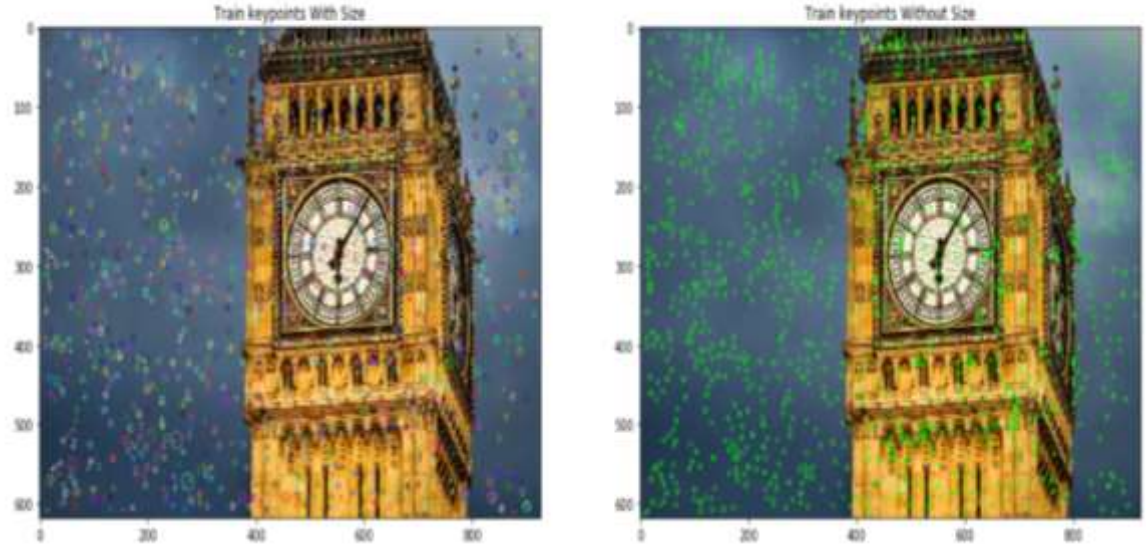
Resift:

1.Smoothening image:

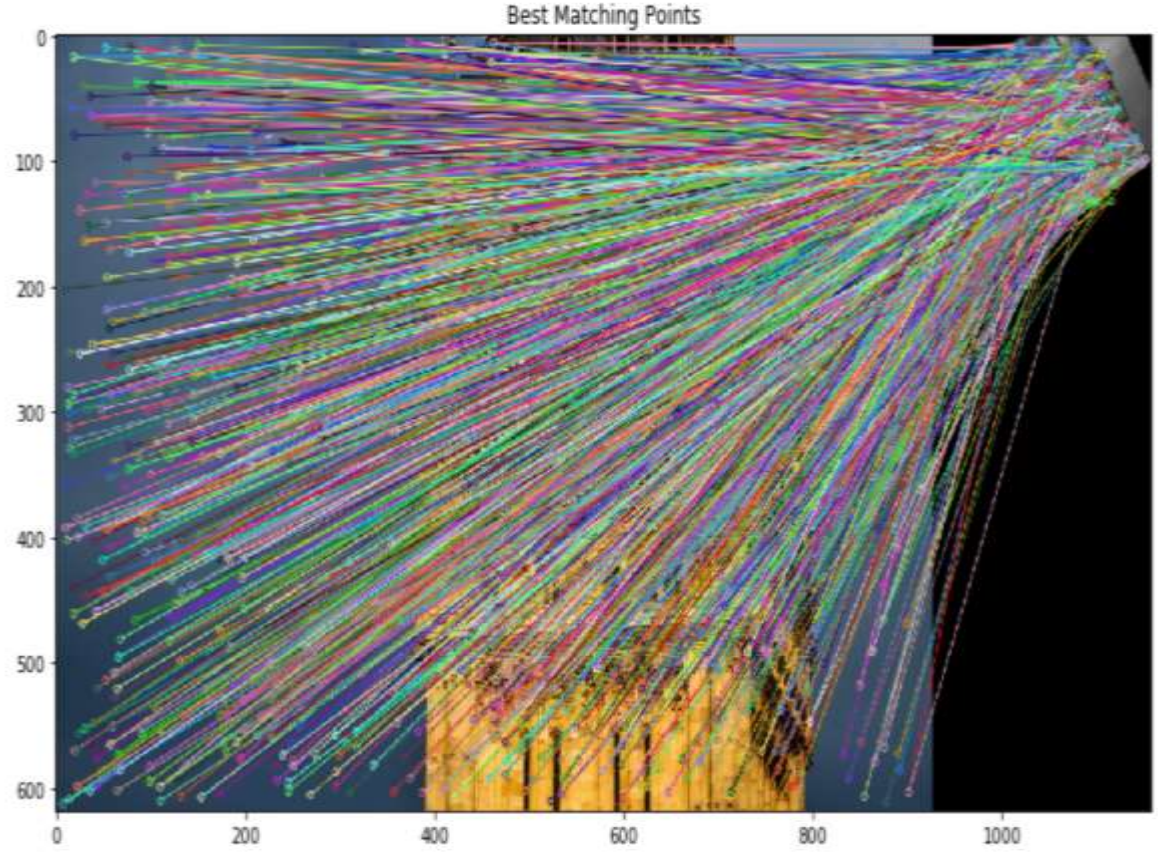


2.Keypoints detected:

```
Number of Keypoints Detected In The Training Image:  1718
Number of Keypoints Detected In The Query Image:  416
```



3.Feature matching performed:




```
Number of Matching Keypoints Between The Training and Query Images:  1718
```

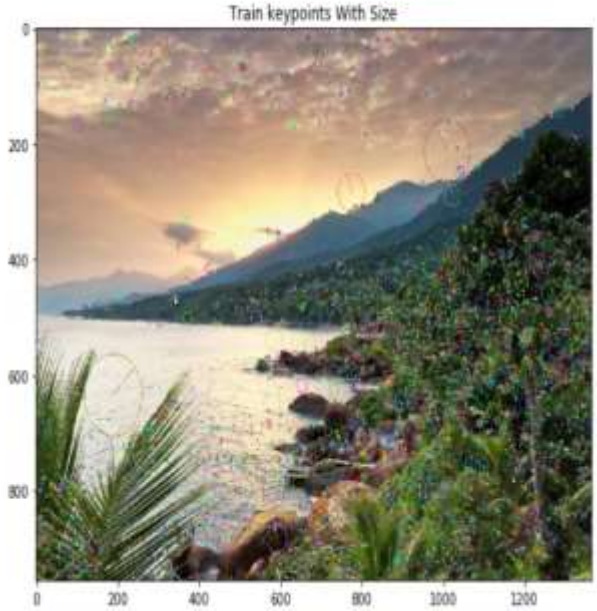
5.3 Example 3

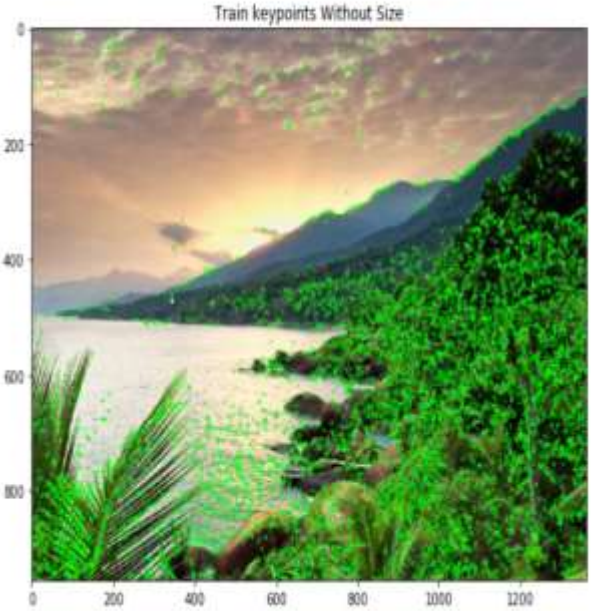
Sift:

1.Training image (scale and rotation invariance):-

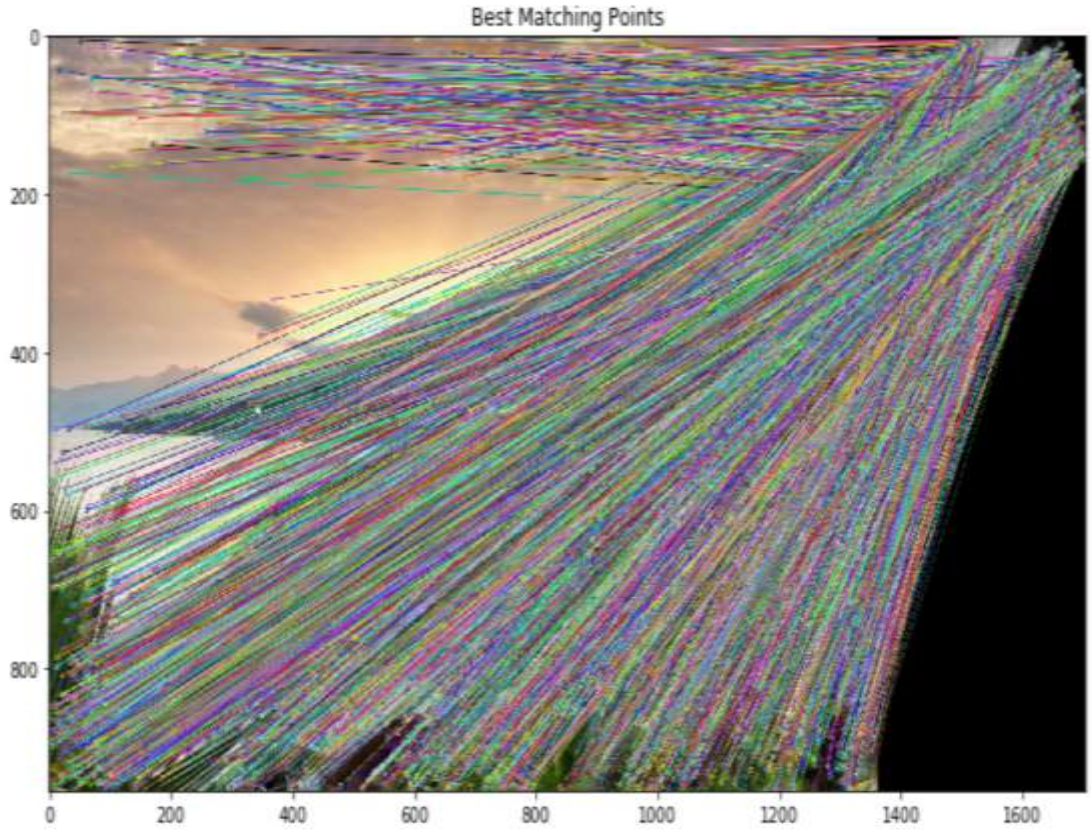
2. Keypoints detected:

```
Number of Keypoints Detected In The Training Image:  9244
Number of Keypoints Detected In The Query Image:  615
```





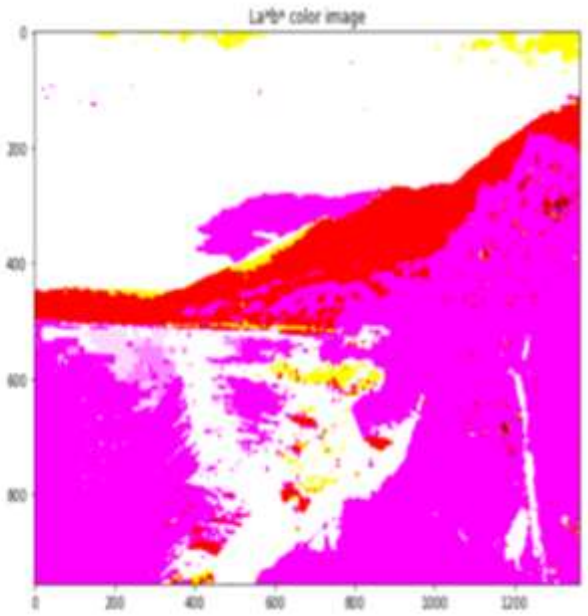
3.Feature matching performed:




```
Number of Matching Keypoints Between The Training and Query Images:  9244
```

Resift:-

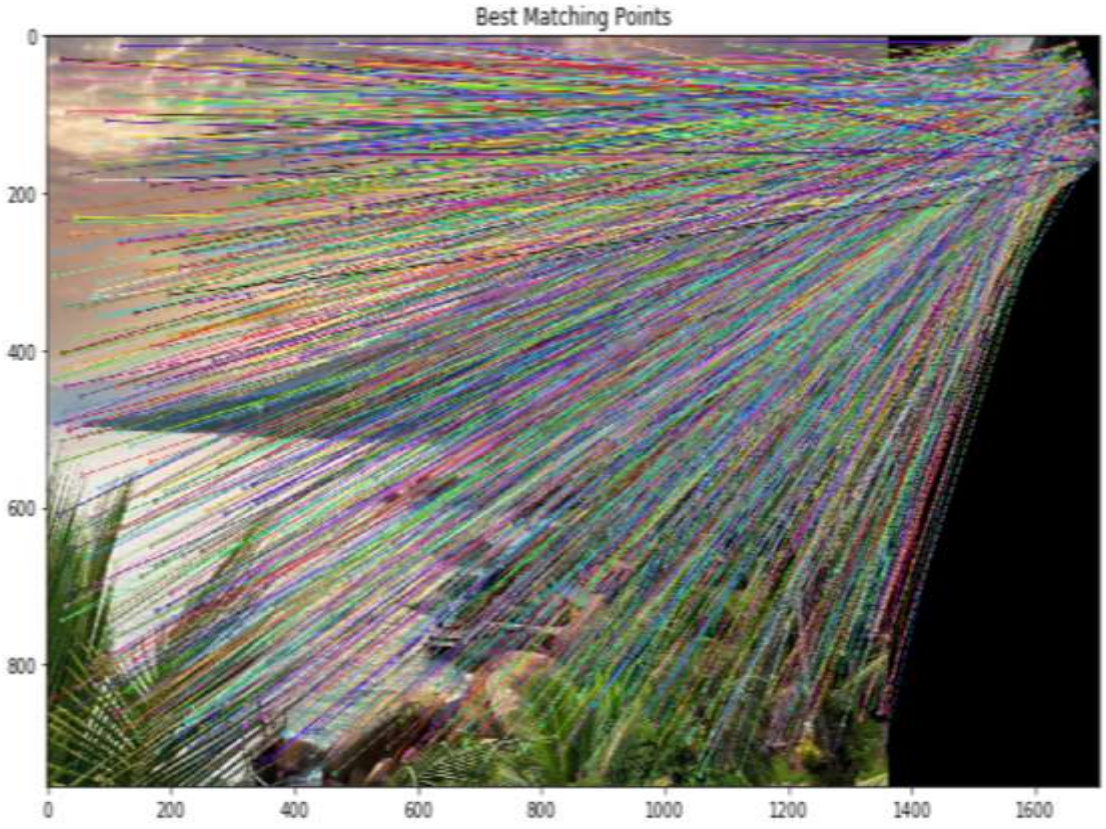
1. Smoothening image:



2. Keypoints detected:

```
Number of Keypoints Detected In The Training Image:  1956
Number of Keypoints Detected In The Query Image:  615
```

3. Feature matching performed:



Chapter 6

Learning Outcomes

In our work, we studied that how feature extraction from image works with the help of sift and resift algorithms. We studied the algorithms in depth. Firstly, we learned why there is a need of feature extraction from image and where is its needed. We then tried understanding the sift algorithm and its mathematical implementation, various intermediate steps it follows and the output that it gives. Then we studied resift algorithm which uses sift algorithm as one of its steps. For example we used various images an input and studied the output as the number of keypoints obtained by each of the algorithms.

We also observed the difference between the output depicted by the sift algorithm and the resift algorithm.

- The number of keypoints obtained by the sift algorithms is large as compared to the number of keypoints observed in case of resift algorithm.
- This is so because in resift algorithm, only the major keypoints are observed and it discards/ ignores the redundant keypoints.
- In sift algorithm the image is taken as input, the sift is applied on the raw image only, resulting in the keypoints which may or may not be good features.
- In resift algorithm the image is first smoothened and normalized, then on that image the sift algorithm is functioned, resulting in more accurate keypoints for features.

So, we can conclude that the resift algorithm obtains more accurate features as compared to sift algorithm. So, the number of keypoints are thus reduced in resift.

Appendix A

References

[1] M. Yuvaraju, K. Sheela Sobana Rani,Feature *Extraction of Real-Time Image Using Sift Algorithm*, International Journal of Research in Electrical & Electronics Engineering Volume 3, Issue 4, October-December, 2015, pp. 01-07.

[2] Dogancan Temel and Ghassan AlRegib, *reSIFT: RELIABILITY-WEIGHTED SIFT-BASED IMAGE QUALITY ASSESSMENT, Center* for Signal and Information Processing (CSIP) School of Electrical and Computer Engineering Georgia Institute of Technology, Atlanta, GA, 30332-0250 USA.

[3] Guo, F., Yang, J., Chen, Y., & Yao, B. (2018). *Research on image detection and matching based on SIFT features.* 2018 3rd International Conference on Control and Robotics Engineering (ICCRE).