A Comparative Study of SIFT and SURF Using MATLAB

Summer Internship Report
Submitted in partial fulfillment of the
Requirement for the award of degree of B.Tech in
Information Technology & Computer Science Engineering
By

Shitanshu Shekhar (B.Tech, I.T, 2007722)
Harsh Vardhan (B.Tech, C.S.E, 2007464)

Under the guidance of

Prof.(Dr.) D. R. Gangodkar,
Dean, International Affairs & professor CSE

Department of Computer Science and Engineering
Graphic Era University
(Under Section 3 of UGC Act, 1956)
Dehradun-248002
June-July 2016

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Internship Report titled **"A Compartive study of SIFT and SURF using MATLAB "** in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering / Information Technology of Graphic Era University and submitted in the Department of Computer Science and Engineering of Graphic Era University, Dehradun is a record of our own work carried out during a period from 17 June 2016 to 02 August 2016, under the supervision of **Dr. D.R. Gangodkar, Professor**, Department of Computer Science and engineering of Graphic Era University, Dehradun.

Harsh Vardhan
ENo. - 13207464
Dept. of Computer Science

Shitanshu Shekhar
ENo. - 13207722
Dept. of Information Technology

The internship was carried out under the guidence of (Signature of the Supervisior)

Acknowledgement

We feel ourselves honored to have had suchan opportunity as to work under the capable guidance of **Prof.(Dr.) D. R. Gangodkar**, without which this internship would not have been possible. His exemplary counseling, monitoring, insipiration and continuous encouragment are only of a few ways he played a substantial role in our endeavor and his teaching shall go a long mile to help us in making informed and succesful choices.

We also take this moment to heartly thanks everyone surrounding us for keeping us to the mark and always look to the better future . We are indebted to them for their love and support through this work.

We convey our sincere thanking to the staff of Graphic Era University, especially Mr. Vidit Kumar, for his invaluable assistance. Their Motivation has been a stepping stone in making this undertaking a successs.

Shitanshu Shekhar
Harsh Vardhan

LIST OF FIGURE

Table of Contents

1. Introduction

1.1 Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps:

- Importing the image via image acquisition tools;
- Analysing and manipulating the image;
- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction.

1.2 Digital Image Processing

An Image is defined as:

- Two dimensional function f(x, y) where x and y are spatial coordinates.
- The amplitude of f at any pair of coordinates (x, y) is called the intensity or gray point of that image. When **x,** y and the amplitude values of f are all finite, discrete quantities, we call the image a digital image.
- A digital image is composed of finite elements, each of which has a particular location and values. These elements are referred to as picture elements, image elements and pixels [1].

The term gray level is used often to refer to the intensity of monochrome images. Color images are formed by a combination of individual 2-D images. For example, in the RGB color system, a color image consists of three (red, green, and blue) individual component images. For this reason, many of the techniques developed for monochrome images can be extended to color images by processing the three component images individually.

An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. Converting such an image to digital form requires that the coordinates, as well as the amplitude, be digitized. Digitizing the coordinate values is called sampling; digitizing the amplitude values is called quantization. Thus, when x, y, and the amplitude values off are all finite, discrete quantities, we call the image a digital image [1].

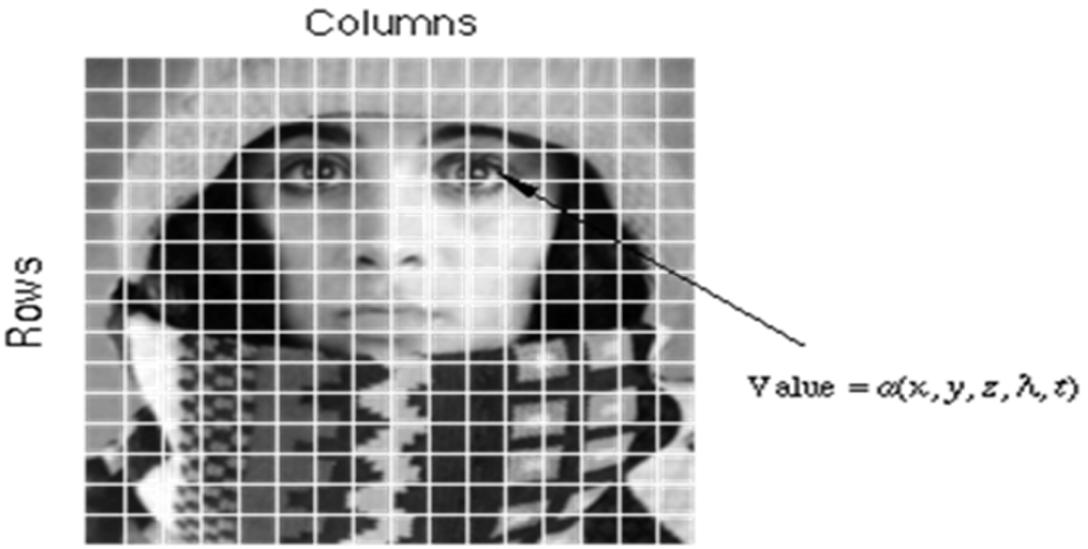

Figure 1.1: Coordinate convention used Image[1]

1.2.1 Images as Matrices[1]

The coordinate system in Figure 1.1 and the preceding discussion lead to the following representation for a digitized image function:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

The right side of this equation is a digital image by definition. Each element of this array is called an *image element, picture element, and pixel.* The terms *image* and *pixel* are used throughout the rest of our discussions to denote a digital image and its elements. A digital image can be represented naturally as a MATLAB matrix:

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}(1,1) & \mathbf{f}(1,2) & \cdots & \mathbf{f}(1,N) \\ \mathbf{f}(2,1) & \mathbf{f}(2,2) & \cdots & \mathbf{f}(2,N) \\ \vdots & \vdots & & \vdots \\ \mathbf{f}(M,1) & \mathbf{f}(M,2) & \cdots & \mathbf{f}(M,N) \end{bmatrix}$$

Where f (1, 1) = f (0,O) (note the use of a monospace font to denote MATLAB quantities). Clearly the two representations are identical, except for the shift in origin. The notation f (p, q) denotes the element located in row p and column q. For example, f (6, 2) is the element in the sixth row and second column of the matrix f. typically we use the letters M and **N,** respectively, to denote the number of rows and columns in a matrix. A 1 x N matrix is called a *row vector,* whereas an *M* x 1 matrix is called a *column vector.* A *1 x* 1 matrix is a *scalar.*

Matrices in MATLAB are stored in variables with names such as **A,** a, RGB, real-array, and so on. Variables must begin with a letter and contain only letters, numerals, and underscores. As noted in the previous paragraph, all MATLAB quantities in this book are written using monospace characters. We use conventional Roman, italic notation, such as f *(x* , y), for mathematical expression.

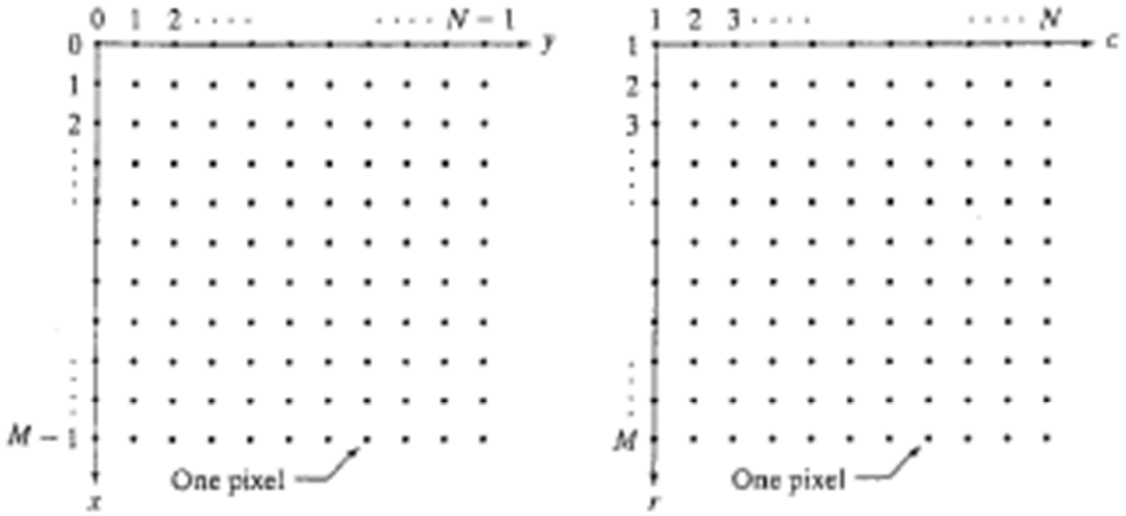

Figure 1.2: Coordinate conventions used in many Image processing and In the Image Processing Toolbox [1].

1.3 Applications of Digital Image Processing

Some of the major fields in which digital image processing is widely used are mentioned below [7]:

- Image sharpening and restoration
- Medical field
- Remote sensing
- Transmission and encoding
- Machine/Robot vision
- Color processing

- Pattern recognition
- Video processing
- Microscopic Imaging
- Others

1.3.1 Image sharpening and restoration

Image sharpening and restoration refers here to process images that have been captured from the modern camera to make them a better image or to manipulate those images in way to achieve desired result. It refers to do what Photoshop usually does [7].

This includes Zooming, blurring, sharpening, gray scale to color conversion, detecting edges and vice versa, Image retrieval and Image recognition. The common examples are:

Figure 1.3: The original image [7]

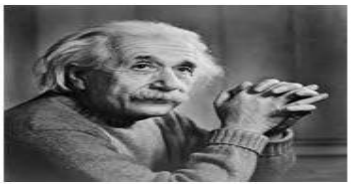

Figure 1.4: The zoomed image [7]

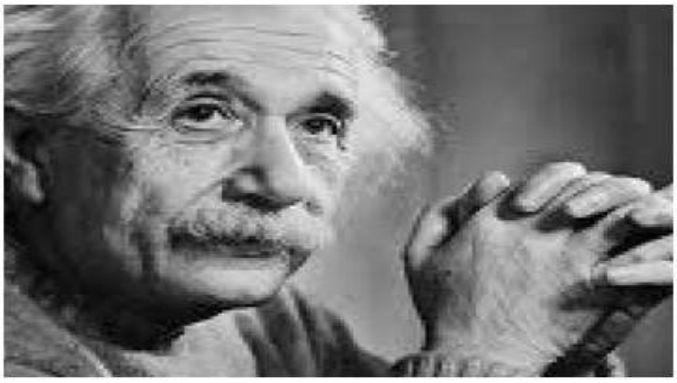

Figure 1.5: Blur image [7]

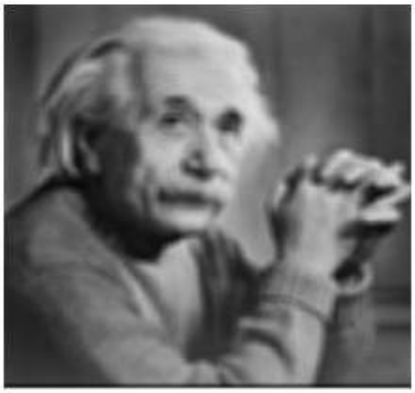

Figure 1.6: Sharp image [7]

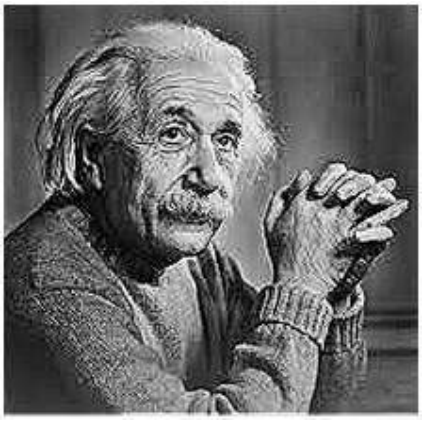

Figure 1.7: Edges [7]

1.3.2 Medical field

The common applications of DIP in the field of medical is

- Gamma ray imaging
- PET scan
- X Ray Imaging
- Medical CT
- UV imaging

1.3.3 UV imaging

In the field of remote sensing, the area of the earth is scanned by a satellite or from a very high ground and then it is analyzed to obtain information about it. One particular application of digital image processing in the field of remote sensing is to detect infrastructure damages caused by an earthquake [7].

As it takes longer time to grasp damage, even if serious damages are focused on. Since the area effected by the earthquake is sometimes so wide, that it not possible to examine it with human eye in order to estimate damages. Even if it is, then it is very hectic and time consuming procedure. So a solution to this is found in digital image processing. An image of the effected area is captured from the above ground and then it is analyzed to detect the various types of damage done by the earthquake.

Figure 1.9: Measuring distance by using UV imaging [7]

The key steps include in the analysis are

- The extraction of edges
- Analysis and enhancement of various types of edges

1.3.4 Transmission and encoding

The very first image that has been transmitted over the wire was from London to New York via a submarine cable. The picture that was sent is shown below.

The picture that was sent took three hours to reach from one place to another [7].

Now just imagine, that today we are able to see live video feed, or live cc TV footage from one continent to another with just a delay of seconds. It means that a lot of work has been done in this field too. This field does not only focus on transmission, but also on encoding. Many different formats have been developed for high or low bandwidth to encode photos and then stream it over the internet or etc.

1.3.5 Machine/Robot vision

Apart from the many challenges that a robot face today, one of the biggest challenge still is to increase the vision of the robot. Make robot able to see things, identify them, identify the hurdles etc. Much work has been contributed by this field and a complete other field of computer vision has been introduced to work on it.

1.3.6 Hurdle detection

Hurdle detection is one of the common task that has been done through image processing, by identifying different type of objects in the image and then calculating the distance between robot and hurdles.

Figure 1.10: Detecting object using certain Algorithm [7]

1.3.7 Line follower robot

Most of the robots today work by following the line and thus are called line follower robots. This help a robot to move on its path and perform some tasks. This has also been achieved through image processing [7].

Figure 1.11: Line follower [7]

1.3.8 Color processing

Color processing includes processing of colored images and different color spaces that are used. For example RGB color model, YCbCr, HSV. It also involves studying transmission, storage, and encoding of these color images.

1.3.9 Pattern recognition

Pattern recognition involves study from image processing and from various other fields that includes machine learning (a branch of artificial intelligence). In pattern recognition, image processing is used for identifying the objects in an images and then machine learning is used to train the system for the change in pattern. Pattern recognition is used in computer aided diagnosis, recognition of handwriting, recognition of images etc.

1.3.10 Video processing

A video is nothing but just the very fast movement of pictures. The quality of the video depends on the number of frames/pictures per minute and the quality of each frame being used. Video processing involves noise reduction, detail enhancement, motion detection, frame rate conversion, aspect ratio conversion, color space conversion etc.

2. Literature Survey

Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill [8] Object Recognition has become one of the most attractive areas of research for most of the scientists over the past few decades. Object recognition has extensive applications in numerous areas of interest. In this paper, the importance of object recognition in different applications has been highlighted. The very famous and impressive technique by David Lowe which is Scale Invariant Feature Transform (SIFT) has been implemented for object recognition and an attempt has been done to compare the results obtained from it with the another very important technique called Speeded-Up Robust Feature Transform (SURF) to conclude with certain interesting results

Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool [9] explores the effectiveness of the Scale Invariant Feature Transform (SIFT) for image matching. There is a popularly used interest point detection method often applied to image matching, the Harris corner detector. The Harris corner detector is non-invariant to scale change. The experiments of image matching based on both the SIFT and the Harris corner detector are performed to show the effectiveness of the scale invariant property of the SIFT method. Furthermore, the image matching method based on SIFT is applied to point tracking, and comparison with Kanade-Lucas-Tomasi (KLT) feature point tracker is also studied.

P M Panchal, S R Panchal, S K Shah [10] describes Image matching task to finding correspondences between two images of the same scene/object is part of many computer vision applications. Image registration, camera calibration and object recognize just few. This paper describes distinctive features from images is divided into two main phases. First, “key points” are extracted from distinctive locations from the images such as edges, blobs, corner etc. Key point detectors should be highly repeatable. Next, neighborhood regions are picked around every key point and distinctive feature descriptors are computed from each region.

Ritu Rani [11] describes a new algorithm called SIFT (Scale Invariant Feature Transform). Using SIFT the reliable matching features between images are extracted to different views of the same object. The extracted features from images are highly distinctive and are invariant to scale and orientation of the image. The feature extraction is done in four steps. In the first step the locations of potential interest points in the image are computed by detecting the maxima and minima of Difference of Gaussian (DoG) filters which is applied all over the image at different scales. Then, by discarding the points having low contrast the detected locations of points are refined. Then based on local image features an orientation is assigned to each key point. Finally, at each key point a local feature descriptor is computed. This descriptor is transformed to the orientation of the key point based on the local image gradient to provide orientation invariance.

David. G. Lowe [12] presents a reliable matching method of extracting distinctive invariant features between images having different views of object or place. The features are invariant to image rotation and scale, distortion, 3D change viewpoint, addition of noise, and illumination change. The features are highly distinctive, which means a single feature of the

images can be matched correctly with high probability of large database of features of many images. The paper also describes the features for object recognition. The recognition is done by matching individual features to a database of features from known objects.

B. Herbert et.al [13] describes a new method Speeded Up Robust Features (SURF) - a fast and good performance interest point detection-description method, which perform the current state-of-the art, in both speed and accuracy. SURF descriptor is based on similar properties, with a compressed complexity. The SURF descriptor is processed in two steps. In the first step, fixing of a reproducible orientation are constructed based on information from a circular region of interest points of features. Then in second step, construction of a square region aligned to the selected orientation is done, and extract the SURF descriptor from it. The SURF descriptor is easily extendable for the description of invariant regions.

Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill [14] describes Accurate, robust and automatic image registration is critical task in many applications. To perform image registration/alignment, required steps are: Feature detection, Feature matching, derivation of transformation function based on corresponding features in images and reconstruction of images based on derived transformation function. Accuracy of registered image depends on accurate feature detection and matching. So these two intermediate steps are

Very important in many image applications: image registration, computer vision, image mosaic etc. This paper presents two different methods for scale and rotation invariant interest point/feature detector and descriptor: Scale Invariant Feature Transform (SIFT) and Speed Up Robust Features (SURF). It also presents a way to extract distinctive invariant features from images that can be used to perform reliable matching between different views of an object/scene.

3. Feature Extraction Algorithms and used Application

3.1 Overview

The basics of feature extraction is the dimensionality reduction, by choosing some dominant or distinct features that can best represents the image with less distortion to the original image. Appropriate algorithms are used to extract the salient features from the relevant patterns. Basic Object recognition systems involve extraction of features and then matching of these features with the features calculated and stored in the database. Features are basically the key points 'which can uniquely define the whole object i.e. the features should be able to give the most of the information about the object/data. Features can be patches, edges, corners. When all images are similar in nature (same scale, orientation, etc.) simple corner detectors can work [11]. But when there are images of different scales and rotations, then there is need to use some very advanced techniques which can help recognize the objects under all these constraints of different scaling, orientations, illuminations and occlusion. Thus, in this internship an introduction to two very effective techniques has been given which is very popular algorithm used in field of computer vision:

- Scale Invariant Feature Transform (SIFT)
- Speeded-Up Robust Feature Transform (SURF)

3.2 Working Environment

MATLAB is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for Matrix Laboratory and the software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has powerful graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. MATLAB also has some tool boxes useful for signal processing, image processing, optimization, etc [1].

The *Command Window* is where the user types MATLAB commands and expressions at the prompt (>>) and where the outputs of those commands are displayed. MATLAB defines the *workspace* as the set of variables that the user creates in a work session. MATLAB uses a *search path* to find M-files and other MATLAB-related files, which are organized in directories in the computer file system. Any file run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and Math works toolboxes are included in the search path. The easiest way to see which directories are on the search path, or to add or modify a search path, is to select **Set Path** from the **File** menu on the desktop, and then use the **Set Path** dialog box. It is good practice to add any commonly used directories to the search path to avoid repeatedly having the change the current directory.

The *Command History Window* contains a record of the commands a user has entered in the Command Window, including both current and previous MATLAB sessions. Previously entered MATLAB commands can be selected and re-executed from the Command History

Window by right-clicking on a command or sequence of commands. This action launches a menu from which
to select various options in addition to executing the commands. This is a useful feature when experimenting with various commands in a work session.

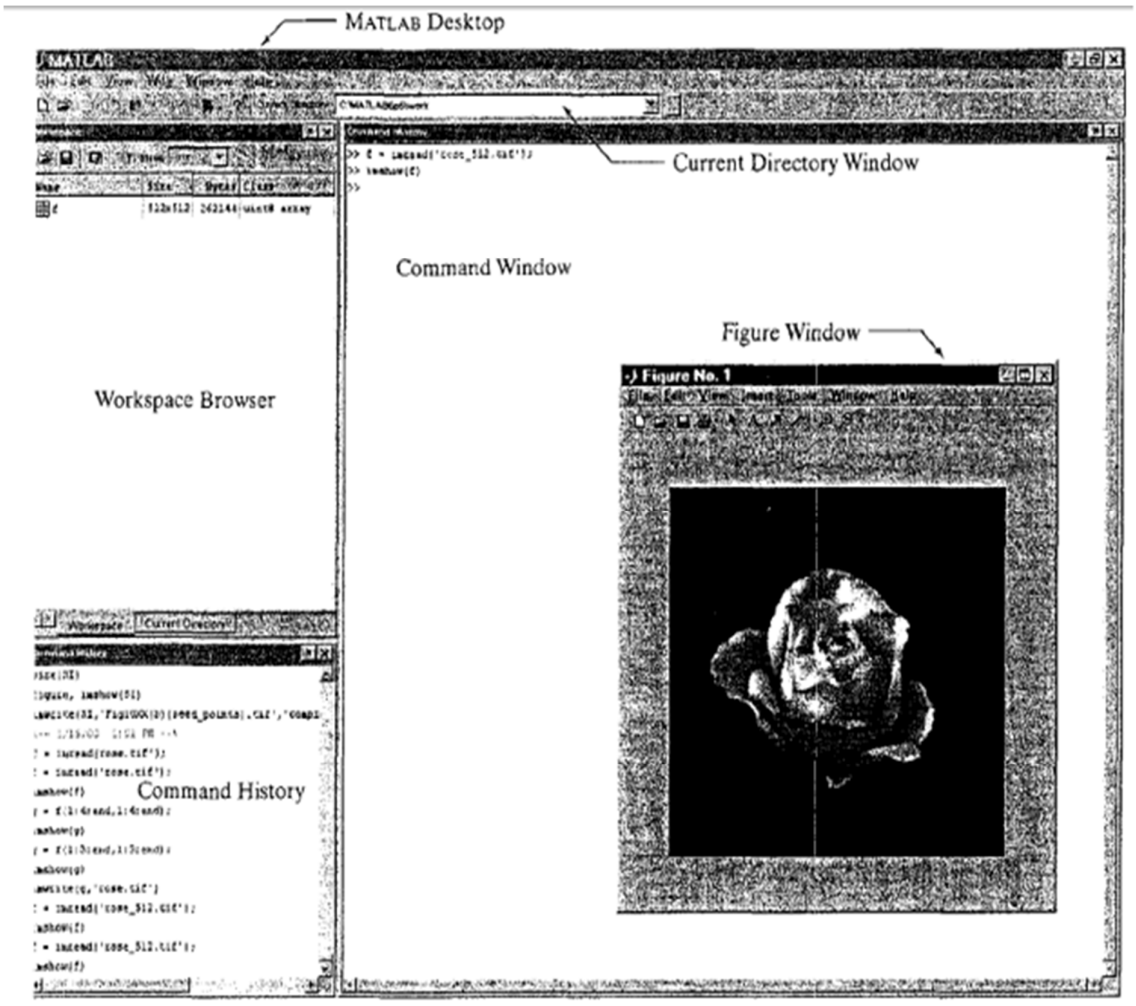

Figure 3.1: The MATLAB desktop and its principal components [1]

3.2.1 Using the MATLAB Editor to Create M-Files

The MATLAB *editor* is both a text editor specialized for creating M-files and a graphical MATLAB debugger. The editor can appear in a window by itself, or it can be a sub window in the desktop. M-files are denoted by the extension **.m,** as in **pixel dup .m**. The MATLAB editor window has numerous pull-down menus for tasks such as saving, viewing, and debugging files. Because it performs some simple checks and also uses color to differentiate between various elements of code, this text editor is recommended as the tool of choice for writing and editing

M functions. To open the editor, type **edit** at the prompt in the Command Window. Similarly, typing **edit filename** at the prompt opens the M-file **filename. m** in an editor window, ready for editing. As noted earlier, the file must be in the current directory, or in a directory in the search path.[1]

3.2.2 Reading Images

Images are read into the MATLAB environment using function imread. Whose syntax is:

>> f = imread (‘filename’)

Here, filename is a string containing the complete name of the image file (including any applicable extension). Reads the JPEG (Figure 8) image chestxray into image array f. Note the use of single quotes (') to delimit the string filename. The semicolon at the end of a command line is used by MATLAB for *suppressing* output. If a semicolon is not included, MATLAB displays the results of the operation(s) specified in that 1ine.The prompt symbol (>>I designates the beginning of a command line, as it appears in the MATLAB Command Window (see Figure 7).

Format Name	Description	Recognized Extensions
TIFF	Tagged Image File Format	`.tif, .tiff`
JPEG	Joint Photographic Experts Group	`.jpg, .jpeg`
GIF	Graphics Interchange Format[1]	`.gif`
BMP	Windows Bitmap	`.bmp`
PNG	Portable Network Graphics	`.png`
XWD	X Window Dump	`.xwd`

[1] GIF is supported by `imread`, but not by `imwrite`.

Figure 3.2: The image graphics formats supported by imread and imwrite starting with MATLAB.

3.2.3 Displaying Images

Images are displayed on the MATLAB desktop using function imshow, which has the basic syntax:

>> imshow (filename)

3.3 Scale Invariant Feature Transform (SIFT)

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images. The algorithm was published by David Lowe in 1999. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving. The algorithm is patented in the US; the owner is the University of British Columbia. [3]

The SIFT algorithm has four main steps:

- Scale Space Extrema Detection
- Key point Localization
- Orientation Assignment and
- Description Generation.

3.3.1 Scale Space Extrema Detection

It is noticeable that, with different scales we can't detect keypoints using the same window. With small corner it is working fine. But larger windows are needed to detect larger corners. For this reason, we used scale-space kernels. In it, Laplacian of Gaussian (LoG) of the fade image is found with different σ values. LoG has been used as a blob detector to find blobs in different scales because of variation in σ. Briefly, σ used as a scaling parameter. For example, for small corners Gaussian kernel that has low σ outputs high value while for larger corners Gaussian kernel that has high σ fits well. So, local maxima can be found across the scale and space which provides us a set of $(x,,)$ values that proves, there is a potential keypoint at (x,y) at σ scale [8]

But this LoG is slightly costly, so SIFT algorithm applies Difference of Gaussians (DoG) on LoG which is an approximation of LoG. DoG is defined as the difference of Gaussian blurring of a face image with two different σ, let it be σ and $k\sigma$. This procedure is applied for various scales (octaves) of the face image that's found in Gaussian Pyramid as shown in Figure 3.3.

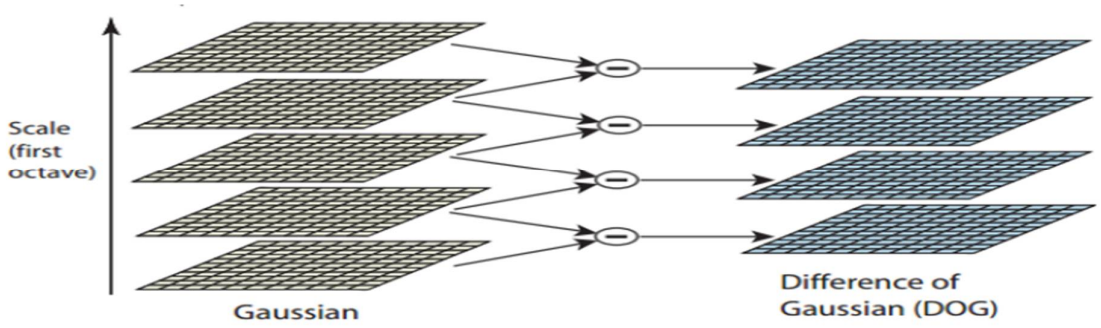

Figure 3.3: Gaussian pyramid [8]

As soon as DoG is found, images are examined to find local minimum and maximum over scale and space. For example, 1 pixel in a face image is compared with 26 pixels which is 8 neighbors, 9

pixels in previous scale and 9 pixels in next scale. It is defined as a possible keypoint, if it's a local minimum or local maximum among pixels. Simply it means that keypoint that's found is the best represented in that scale as shown in Figure 3.4.

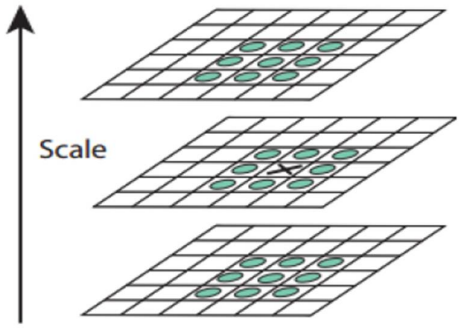

Figure 3.4 Scale space for keypoint [8]

3.3.2 Keypoint Localization

Whenever possible keypoints' positions are found, to get more accurate results from them, they have to be refined. For refining keypoints, Taylor series used as an development of scale space to get more precise position of extrema and if the intensity at this extrema is below the threshold level. Difference of Gaussians has more edge responses, so the edges need to be vanished, for this, a notion is used that is same as Harris corner detector. The principal curvature is computed using a 2x2 Hessian matrix (H) . Harris corner detector states that for any edge, one eigenvalue is bigger than the other eigenvalue. So here they defined a simple function, that is, if the ratio is greater than a threshold value, that keypoint is rejected. This way, any keypoints and edge keypoints with low-contrast were removed, so the remaining is robust and strong keypoints. [8]

3.3.3 Orientation Assignment

Image rotation invariance is achieved by assigning orientation to each keypoint. Depending on the scale, the direction and gradient magnitude of the neighborhood pixels around the keypoint position are taken and calculated. An orientation histogram with 36 bins is formed to cover 360 degrees. It is weight is calculated by Gaussian-weighted circular window and gradient magnitude with σ equal to 1.5 times the scale of keypoint. The calculations of orientation is done by taking the highest peak in the histogram and any other peak above 80% is also considered and taken. It generates keypoints with different directions, but same scale and position. The contribution goes to the stability of matching.

3.3.4 Keypoint Descriptor

A neighborhood of 16x16 scale pixels around the keypoints are taken. It is divided into 16 sub-regions of 4x4 size. 8 bin orientation histogram is created for each sub-block. So a total of 128 bin values are obtained. And, the values are denoted as a vector to create keypoint descriptor as shown in Figure 3.5.

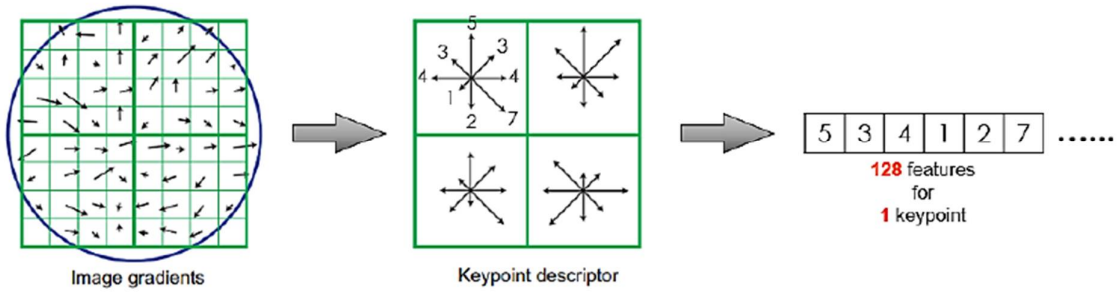

Figure 3.5 : SIFT keypoint descriptor [8]

Along with this, some procedures are applied to gain robustness alongside lighting changes, rotation etc. Figure 3.6 shows distinctive keypoints detected in image.

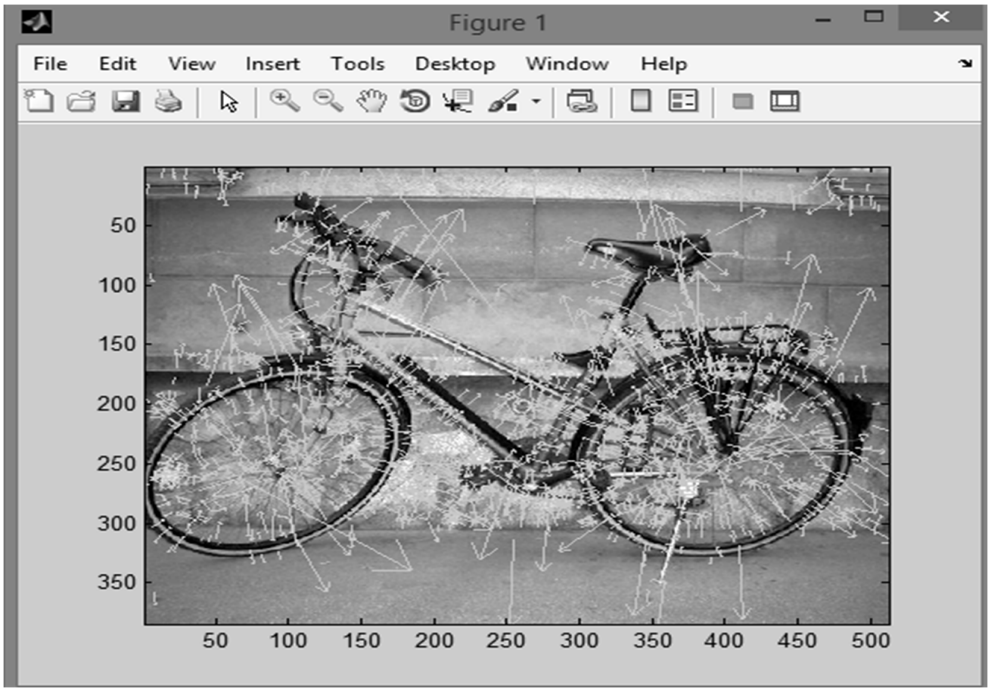

Figure 3.6 : SIFT keypoints detected in a image

3.4 Speeded-Up Robust Features (SURF)

SURF (Speeded-Up Robust Features) was created by Bay, Tuytelaars, and Van Gool in 2006 from ETH Zurich [9]. SURF algorithm is a robust keypoint detector of local features in a face image. It is a developed version of Shift-Invariant Feature Transform (SIFT) and Hessian blob detectors integer approximation to the determinant is calculated with integral images. [10]

In SIFT, Lowe [8] approximated LoG using DoG for scale-space step. SURF goes a slightly more than Laplacian of Gaussian using Box Filter. Figure 2.6 shows approximation demonstration. This approximations biggest advantage is that, it simply uses integral images to calculate the convolution with box filters. Also, for different scales it can be done in parallel. The determinant of Hessian matrix is major component of SURF for both position and scale.

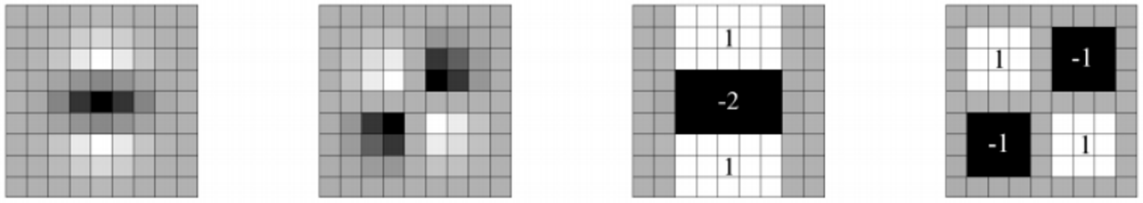

Figure 3.7: The box filters of approximations of Gaussian second order partial derivative [10].

Orientation assignment achieved using wavelet responses in vertical and horizontal direction for a neighborhood of size 6 multiplied by scale in which keypoint is detected. Suitable Gaussian weights are also performed on it. The calculation of the sum of all responses within a sliding orientation window of 60° estimates the main orientation. Exciting part is that, simply integral images can be used to find out wavelet response at any scale. Rotation invariance is not requires in many applications, so finding this orientation is not needed, by this speed of process increases. Figure 2.7 shows distinctive SURF keypoints of a image.

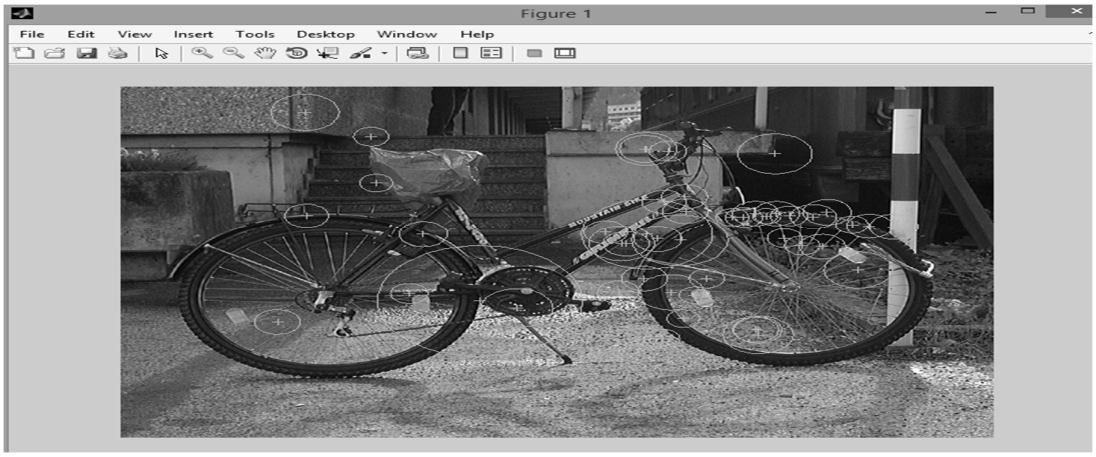

Figure 3.8: SURF keypoints detected in a image.

Around each keypoint a neighborhood of size 20sX20s is taken where s is scale at which keypoint is detected. It is then distributed into 4x4 sub regions as shown in Figure 2.8. Vertical

and horizontal wavelet responses are taken for each sub region and $v=$ (Σdx, Σdy, $\Sigma|dx|$, $\Sigma|dy|$) like, a vector is formed. As it's represented as a vector, it gives a total of 64 dimensions of SURF feature descriptor. So with the SURF keypoint feature vector dimension becomes low, it causes to increase the speed of computation and also increases speed of matching, providing better uniqueness of features. When we use sign of Laplacian (trace of Hessian Matrix), it gives a significant improvement for underlying distinctive keypoints.

In short, to improve the speed of the process SURF adds a lot of features in every step. SURF is good at handling images with blurring and rotation.

In SIFT and SURF keypoints of two images are matched by identifying their nearest neighbors (k-NN) which is shown in Figure 3.9. But sometimes, the second closest-match may be very close to the first. Due to noise or some other reasons. In that case, ratio of closest-distance to second-closest distance is taken into the consideration.

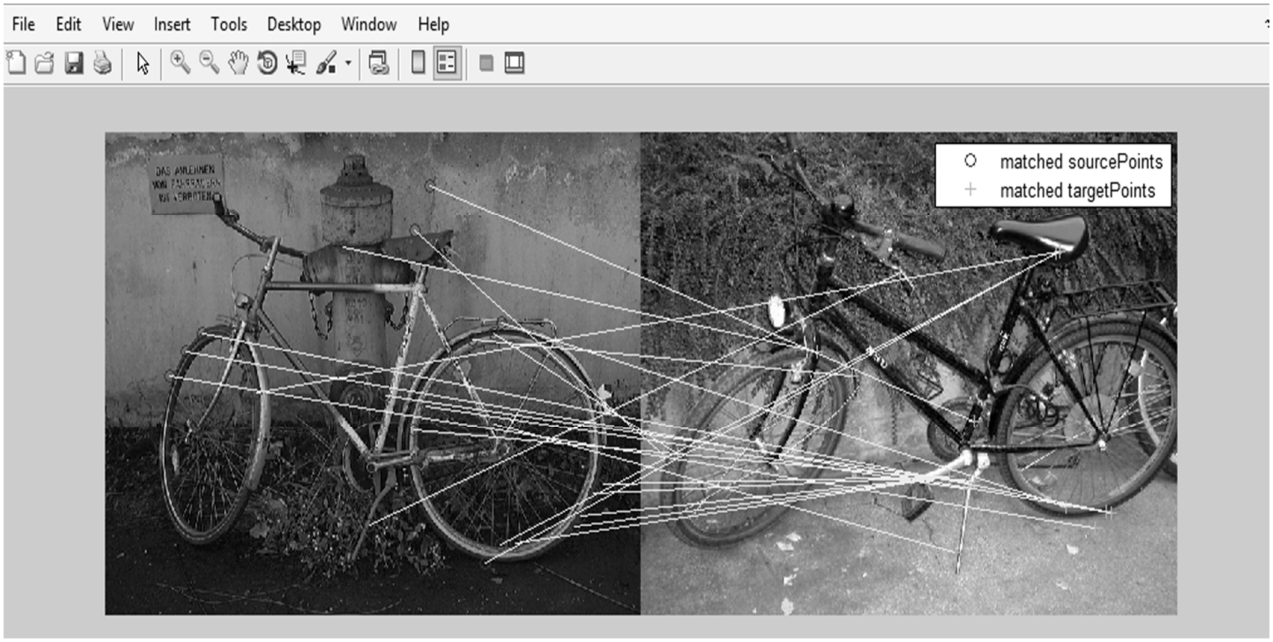

Figure 3.9: Example of point matching result

4. Work Done

4.1 Implementation of SURF

SURF is also popularly known as approximate SIFT. It uses integral images and efficient scale space construction for the efficient generation of keypoints and descriptors. SURF basically involves two stages

- Keypoint detection
- Keypoint description

In the first stage, instead of using Difference of Gaussian like in SIFT, integral images are used which allow the fast computation of approximate Laplacian of Gaussian(LoG) images using a box filter. The computational cost of applying the box filter is independent of the size of the filter because of the integral image representation. Determinants of the Hessian matrix are then used to detect the keypoints. In order to be invariant to rotation, it calculates the Haar-wavelet responses in x and y direction.

4.1.1 Algorithm for SURF

Step 1: Convert image to grayscale
Step 2: Extract interest points
Step 3: Obtain features
Step 4: Match features
Step 5: Get strongest features
Step 6: Match the number of strongest features with each image
Step 7: Take the ratio of numbers of features matched with strongly matched features.

4.1.2 Main part of code for feature matching using SURF

```
source = imread(image_path);  // read the image
target = imread(image_path1);
source = rgb2gray(source);       // convert the colored image to rgb
target = rgb2gray(target);
sourcePoints=detectSURFFeatures(source,'MetricThreshold',100);
// detect strongest 100 features from all feature
selectStrongest(sourcePoints,50);
// select strongest 50 feature from 100 features
targetPoints=detectSURFFeatures(target,'MetricThreshold',100);
selectStrongest(targetPoints,50);
[sourceFeatures,sourcePoints]=extractFeatures(source,sourcePoints);
// extract the feature and store in matrix formate
[targetFeatures,targetPoints]=extractFeatures(target,targetPoints);

boxPairs = matchFeatures(sourceFeatures, targetFeatures);
```

```
matchedSourcePoints = sourcePoints(boxPairs(:, 1), :);
// used to match the feature point between two figure
matchedTargetPoints = targetPoints(boxPairs(:, 2), :);

figure;showMatchedFeatures(source,target,matchedSourcePoints,matchedTargetPoints,'mont
age'); hold on;
// show the matched point
legend('matched sourcePoints','matched targetPoints');
```

4.2 Implementation of SIFT

Scale Invariant Feature Transform (SIFT) wherein he implemented this technique for object recognition. SIFT has now been successfully implemented in number of other applications as well such as fingerprint recognition, face recognition, ear recognition, real-time hand gesture recognition , iris recognition. SIFT provides us with features which are robust to illumination changes, scaling, orientation, occlusion etc.

4.2.1 Algorithm for SIFT

SIFT is quite an involved algorithm and thus it can be broken down into steps as follows:

Step 1: Constructing a scale space
Step 2: LoG Approximation
Step 3: Finding keypoints
Step 4: Get rid of bad key points
Step 5: Assigning an orientation to the keypoints
Step 6: Generate SIFT features

Thus, SIFT is a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different images of the same object or scene.

4.1.2 Main part of code for feature matching using SIFT

```
[image, descrips, locs] = sift(image_path);
showkeys(image, locs);
[image, descrips, locs] = sift(image_path1);
showkeys(image, locs);
match(image_path,image_path1);
```

Sift.m

```
function [image, descriptors, locs] = sift(imageFile)

% Load image
image = imread(imageFile);

% If you have the Image Processing Toolbox, you can uncomment the following
%   lines to allow input of color images, which will be converted to grayscale.
% if isrgb(image)
   image = rgb2gray(image);
 %end

[rows, cols] = size(image);

% Convert into PGM imagefile, readable by "keypoints" executable
f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

% Call keypoints executable
if isunix
    command = '!./sift ';
else
    command = '!siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

% Open tmp.key and check its header
g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end
```

```
% Creates the two output matrices (use known size for efficiency)
locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

% Parse tmp.key
for i = 1:num
   [vector, count] = fscanf(g, '%f %f %f %f', [1 4]); %row col scale ori
   if count ~= 4
      error('Invalid keypoint file format');
   end
   locs(i, :) = vector(1, :);

   [descrip, count] = fscanf(g, '%d', [1 len]);
   if (count ~= 128)
      error('Invalid keypoint file value.');
   end
   % Normalize each input vector to unit length
   descrip = descrip / sqrt(sum(descrip.^2));
   descriptors(i, :) = descrip(1, :);
end
fclose(g);
```

5. Result Analysis

A standard or point of reference against which things may be compared are known as **benchmark.** A benchmark used in my analysis as follows:

- B3DD : Berkley 3D object database household object.
- MIT CBCL bicycle database.
- Fingerprint verification competition (university of Bologna), FVC 2000 and 2002.

5.1 Case (A): For similar image

To verify the effectiveness of the algorithm two images are taken as the experimental data as shown in figure 5.1(a) image1 and (b) image2. The experiments are performed on Intel Core i-3 3210, 2.3 GHz processor and 4 GB RAM with windows 8.1 as an operating system. Features are detected in both images using SIFT and SURF algorithm. Figure 5.1 (c) and (d) shows the detected features using SIFT in image1 and image2 respectively. It is observed that 861 features are detected in image1 and 1156 features are detected in image2.

Figure 5.1 (f) and (g) shows the detected features using SURF algorithm from the original image1 & image2 respectively. It is observed that 482 features are detected in image1 and 373 features in image2.

Figure 5.1(a) Original image1

Figure 5.1(b) Original image2

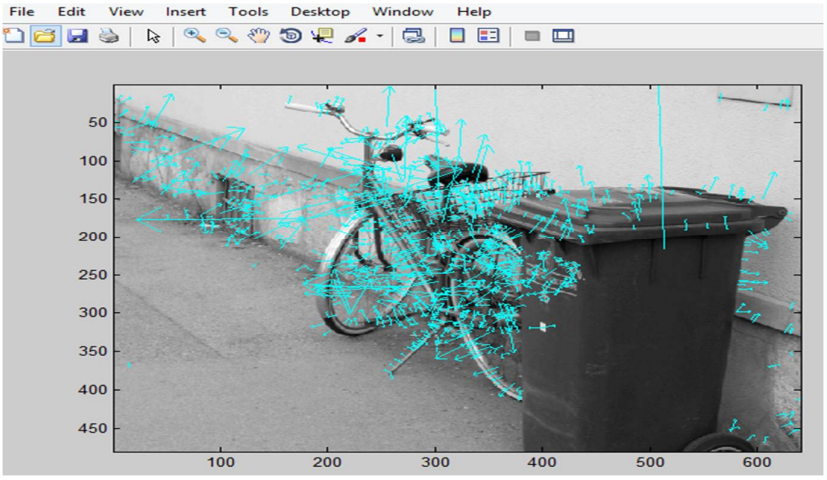

Figure 5.1 (c) Detected features in image1 using SIFT

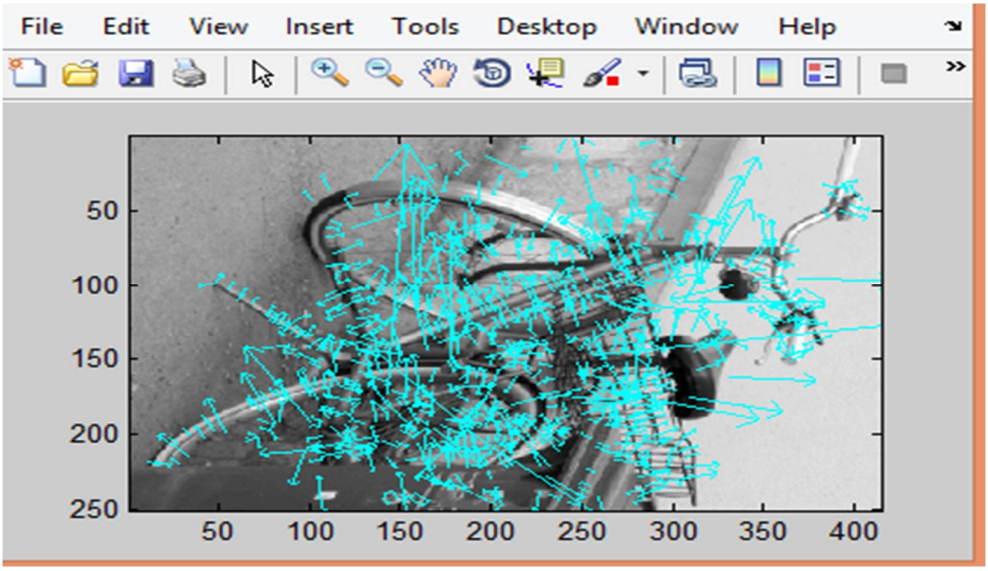

Figure 5.1 (d) Detected features in image1 using SIFT

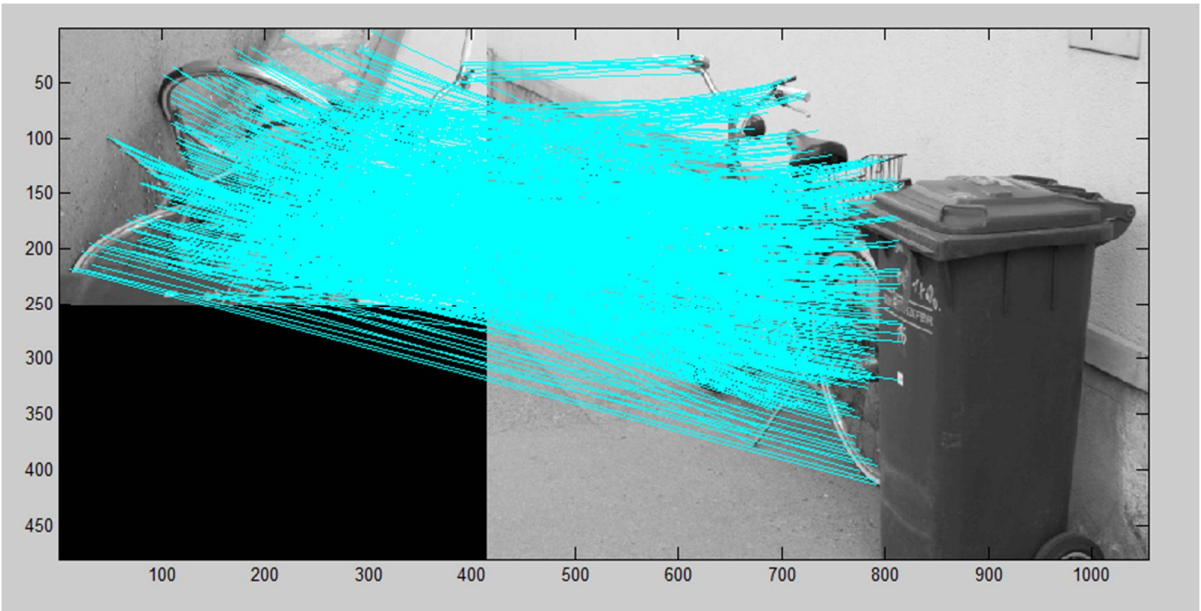

Figure 5.1 (e) Matching pairs identified between the image1 and image2

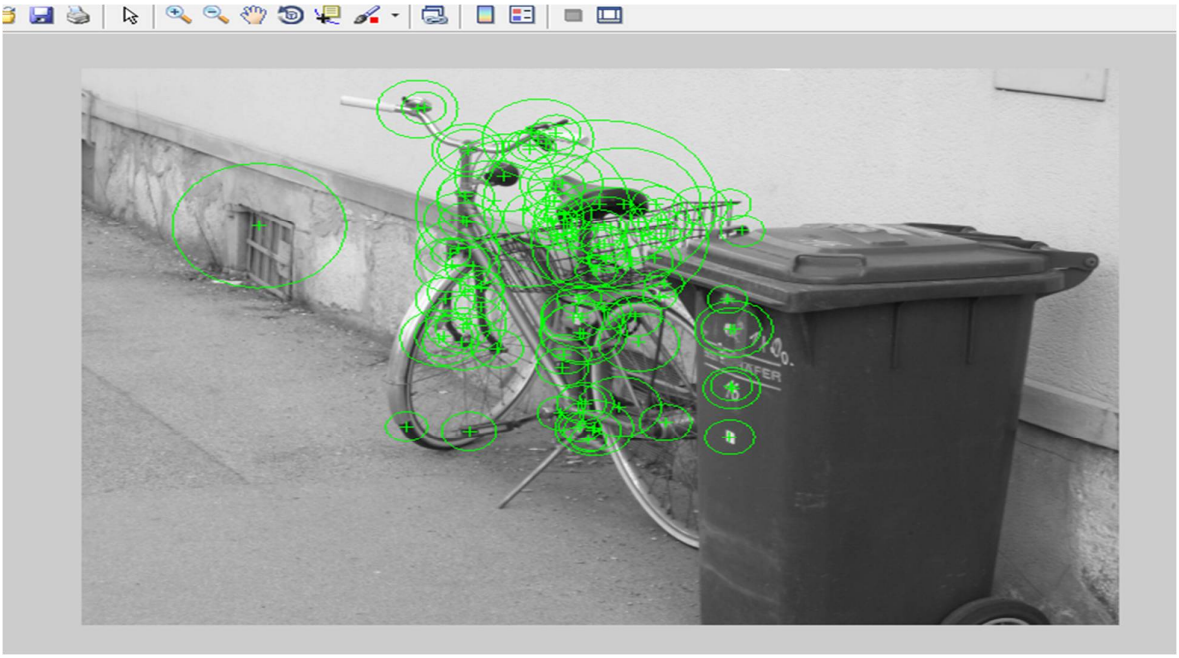

Figure 5.1 (f) Detected features using SURF in image1

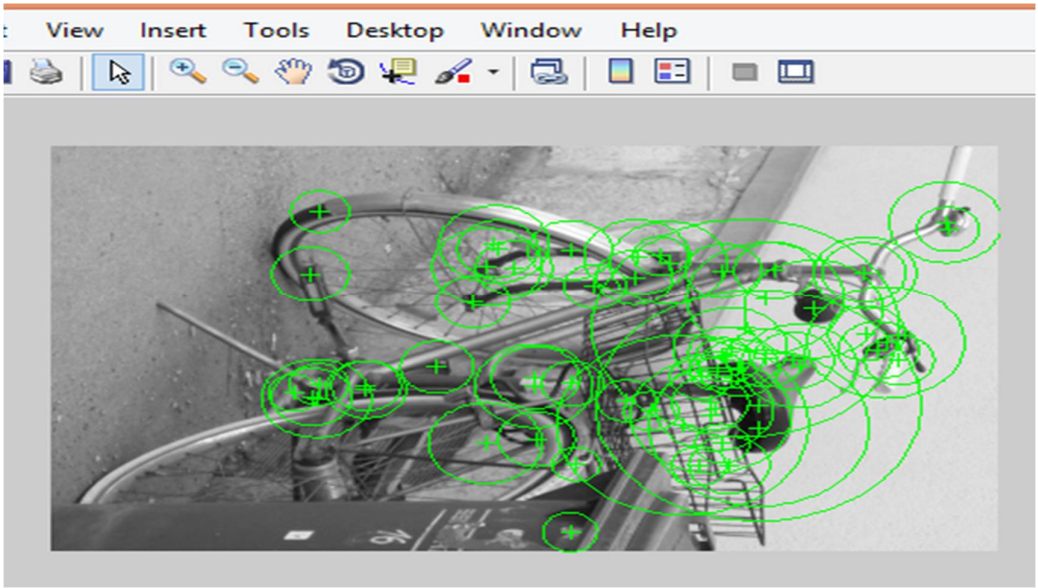

Figure 5.1 (g) Detected features using SURF in image2

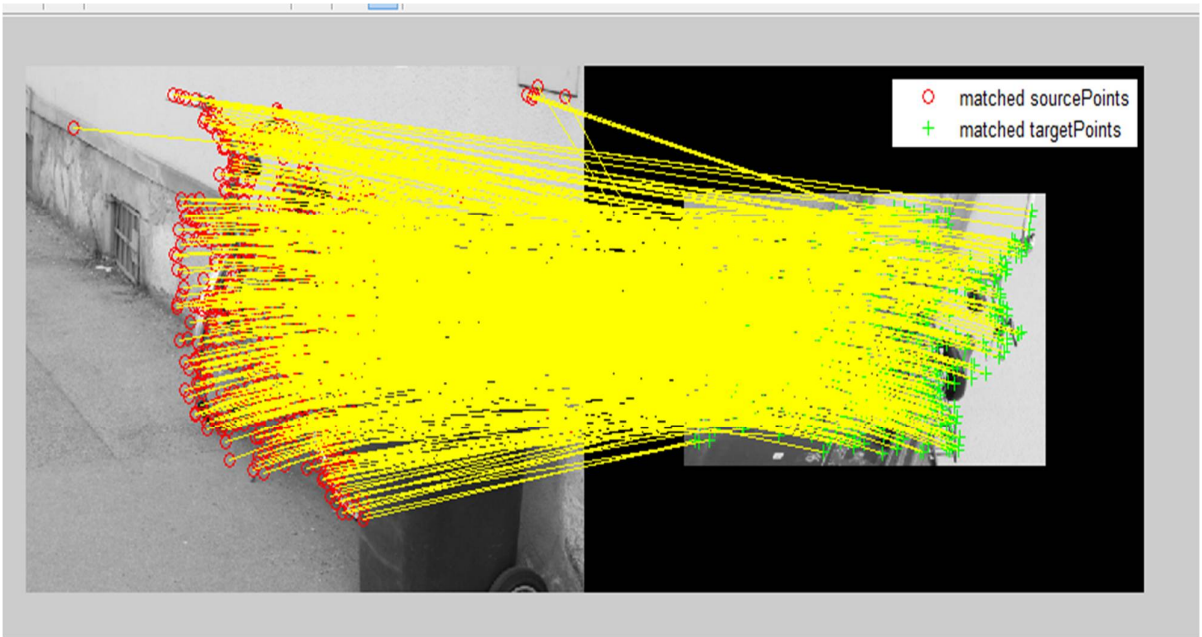

Figure 5.1 (h) Matching pairs identified between the image1 and image2

The features matching is shown in Figure 5.1 (e) are 835 and Figure 5.1 (h) 341 shows matched points. Normalized Cross Correlation technique is used here for feature matching. The experimental results are summarized in Table 1.

5.1 Case (B): For finding identical object in same image

To verify the effectiveness of the algorithm two images are taken as the experimental data as shown in figure 5.2(a) image1 and (b) image2. The experiments are performed on Intel Core i-3 3210, 2.3 GHz processor and 4 GB RAM with windows 8.1 as an operating system. Features are detected in both images using SIFT and SURF algorithm. Figure 5.2 (c) and (d) shows the detected features using SIFT in image1 and image2 respectively. It is observed that 1059 features are detected in image1 and 963 features are detected in image2.

Figure 5.2 (f) and (g) shows the detected features using SURF algorithm from the original image1 & image2 respectively. It is observed that 107 features are detected in image1 and 103 features in image2.

Figure 5.2(a) Original image1

Figure 5.2(b) Original image2

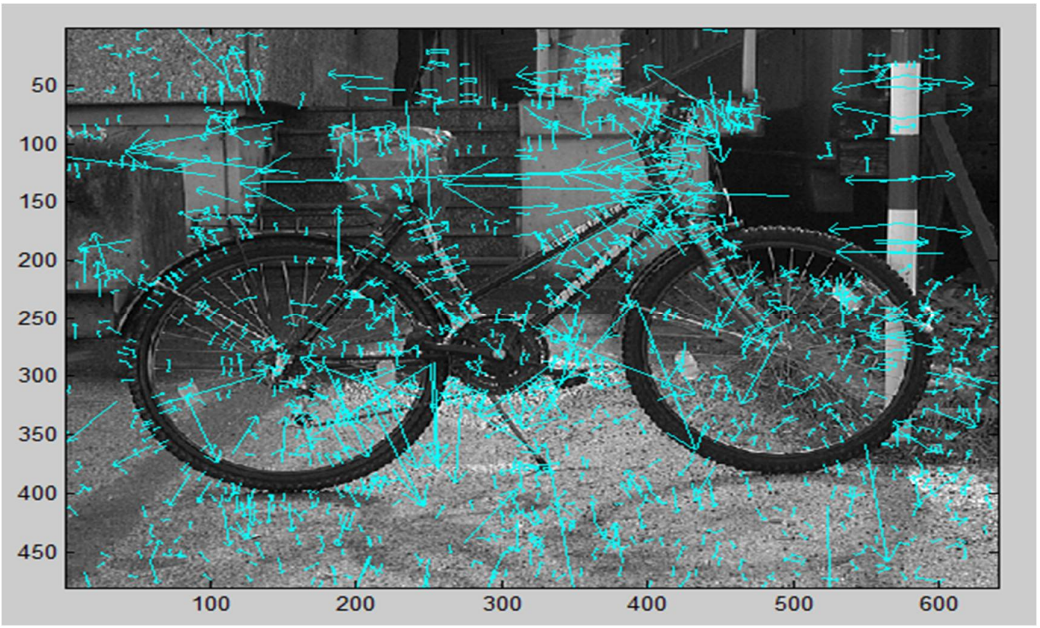

Figure 5.2 (c) Detected features in image1 using SIFT

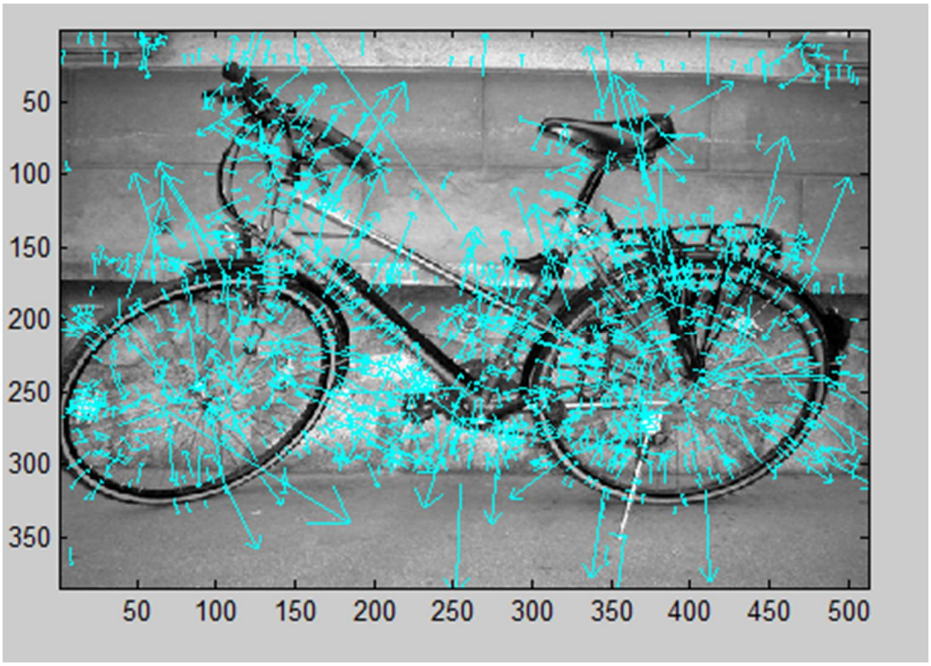

Figure 5.2 (d) Detected features in image2 using SIFT

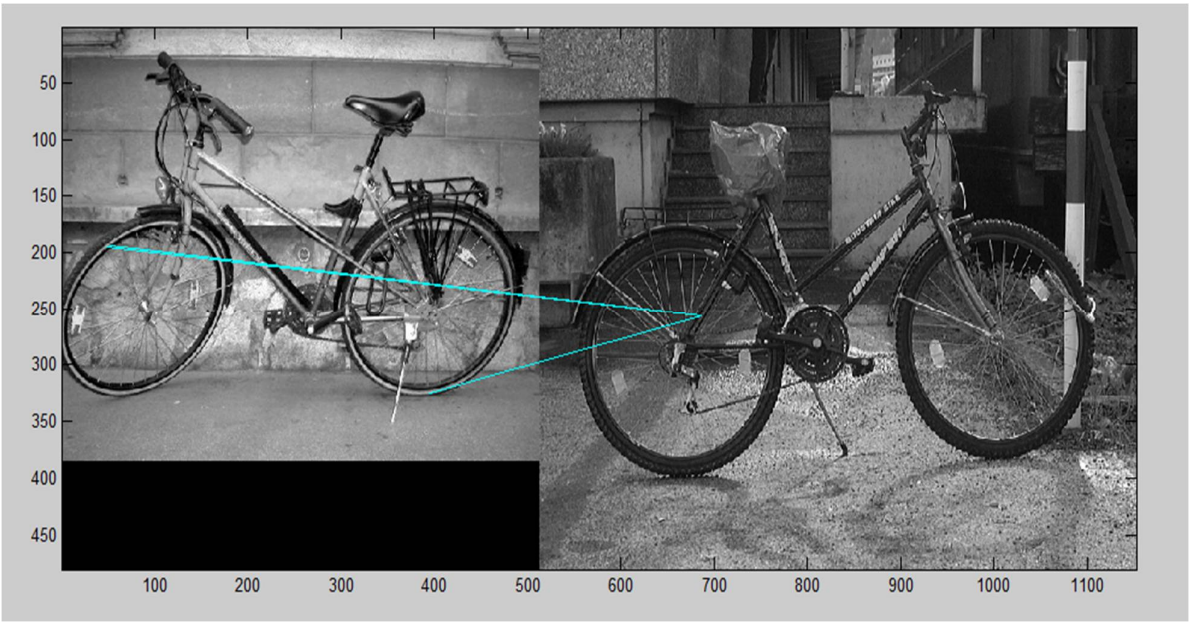

Figure 5.2 (e) Matching pairs identified between the image1 and image2

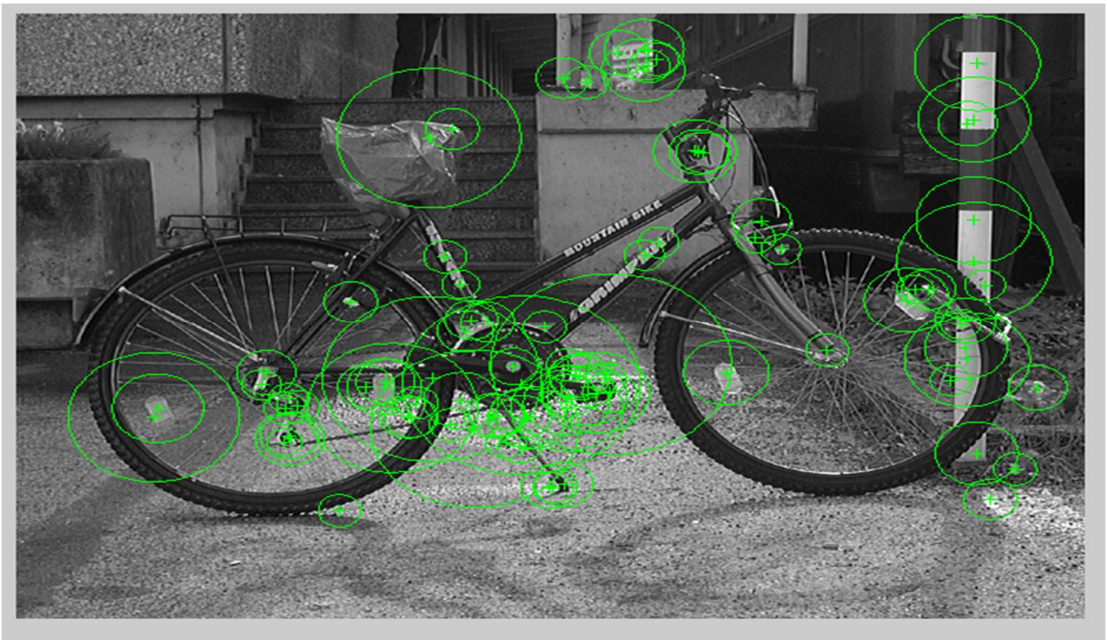

Figure 5.2 (f) Detected features using SURF in image1

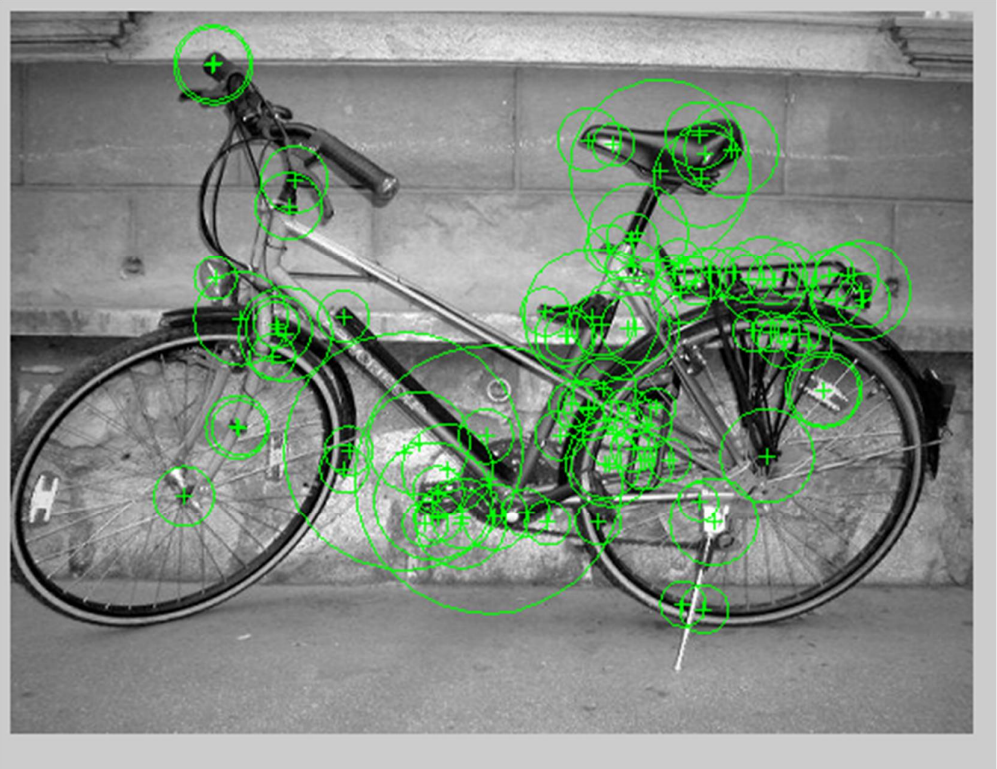

Figure 5.2 (g) Detected features using SURF in image2

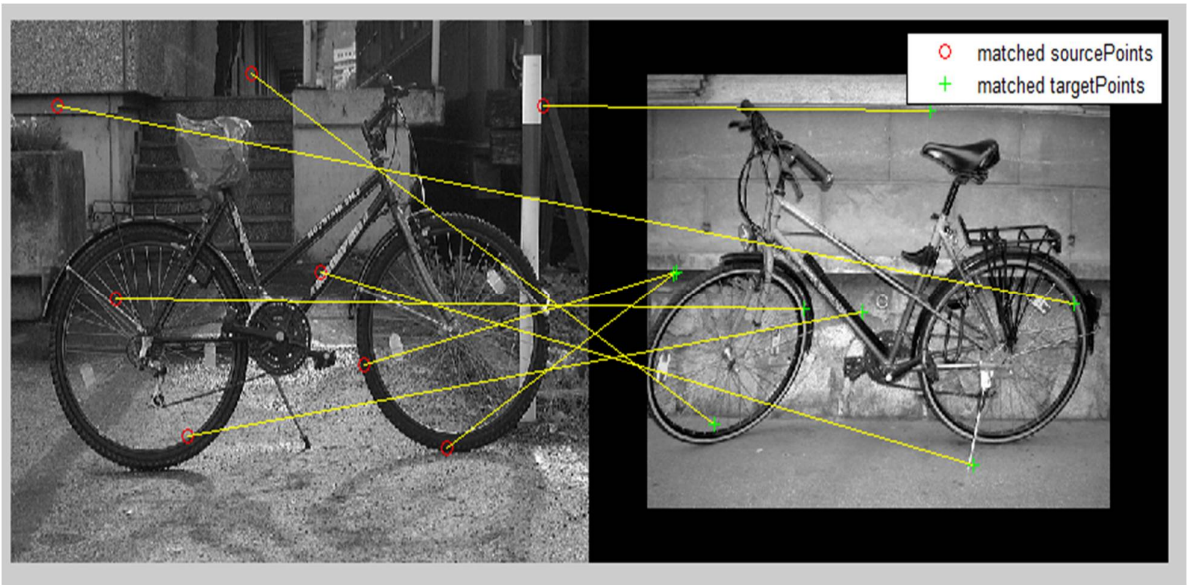

Figure 5.2 (h) Matching pairs identified between the image1 and image2

The features matching is shown in Figure 5.2 (e) are 2 and Figure 5.2 (h) 14 shows matched points. Normalized Cross Correlation technique is used here for feature matching. The experimental results are summarized in Table 2.

For Similar image						
Algorithm	Detected feature Points		Matching feature point	Feature matching Time	Scale Invariant	Rotation invariant
	Image1	Image2				
SIFT(Scale Invariant Feature Transform)	1156	861	835	1.869 sec	Yes	Yes
SURF (Speed Up Robust Feature)	482	373	341	1.159 sec	Yes	Yes

Table 1: Comparisons of results of SIFT and SURF algorithm

For finding identical object in same image						
Algorithm	Detected feature Points		Matching feature point	Feature matching Time	Scale Invariant	Rotation invariant
	Image1	Image2				
SIFT(Scale Invariant Feature Transform)	1059	963	2	2.547 sec	Yes	Yes
SURF (Speed Up Robust Feature)	107	103	14	0.860 sec	Yes	Yes

Table 2: Comparisons of results of SIFT and SURF algorithm

6. Conclusion and Future Work

This report has evaluated two feature detection methods for image. Based on the experimental results, it is found that the SIFT has detected more number of features compared to SURF but it is suffered with speed. The SURF is fast and has good performance as the same as SIFT. For matching the certain object from a same image both the algorithm SURF and SIFT is working competitively same. But in case of identifying the identical object, SIFT algorithm completely fails whereas SURF works perfectly.

Our future scope is to make these algorithms work for the video processing and make a human identification system which is helpful in the field of security.

7. References

[1] *Rafael C. Gonzalez, Richard Eugene Woods, Steven L. Eddins "Digital image processing using mat lab" 2nd edition, 2004. Chapter: Introduction and fundamental of image processing.*

[2] SIFT: Theory and Practice (/tutorials/sift-scale-invariant-feature-transform-eliminate-low-contrast), http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction

[3] Scale-invariant feature transform - Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Scale-invariant_feature_transform/

[4] computer vision - Using SURF algorithm to match objects on MATLAB . http://stackoverflow.com/questions/26226866/using-surf-algorithm/

[5] Find matching features - MATLAB match Features - Math Works Ind . http://in.mathworks.com/help/vision/ref/matchfeatures.html

[6] Speeded up robust features - Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Speeded_up_robust_features

[7] Image processing tutorial , http://www.tutorialspoint.com/dip .

[8] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill, "SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset", International Conference on Digital Image Computing: Techniques and Applications, pp.501-506, 2011.

[9] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, PP. 346--359, 2008.

[10] P M Panchal, S R Panchal, S K Shah,"A comparison of SIFT and SURF", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 1, Issue 2, April 2013, ISSN : 2320 – 9801.

[11] Ritu Rani, Surender Kumar Grewal, Kuldeep Panwar, "Performance evaluation using SIFT and SURF", International Journal of Computer Applications (0975 – 8887) Volume 75– No.3, August 2013

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, Nov. 2004.

[13] B. Herbert, E. Andreas, T. Tinne, and G. Luc Van, "Speeded-Up Robust Features (SURF*),"* Comput. Vis. Image Understand. vol. 110, no. 3, pp. 346–359, Jun. 2008.

[14] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill, "SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset", International Conference on Digital Image Computing: Techniques and Applications, pp.501-506, 2011.