Experiment-1.3

Aim of the Experiment :

To detect and describe the local features in images using SIFT descriptor.

Problem Description :

SIFT (Scale Invariant Feature Transform) Detector is used in the detection of interest points on an input image. It allows the identification of localized features in images which is essential in applications such as:

1. Object Recognition in Images
2. Path detection and obstacle avoidance algorithms
3. Gesture recognition, Mosaic generation, etc.

Unlike other algorithms, which is dependent on properties of the image such as viewpoint, depth, and scale, SIFT can perform feature detection independent of these properties of the image. This is achieved by the transformation of the image data into scale-invariant coordinates. The SIFT Detector has been said to be a close approximation of the system used in the primate visual system.

Steps for Extracting Interest Points

Step 1. Scale Space Peak Selection

The concept of Scale Space deals with the application of a continuous range of Gaussian Filters to the target image such that the chosen Gaussian have differing values of the sigma parameter. The plot thus obtained is called the Scale Space. Scale Space Peak Selection depends on the Spatial Coincidence Assumption. According to this, if an edge is detected at the same location in multiple scales (indicated by zero crossings in the scale space) then we classify it as an actual edge. Here Gaussian function used is DoG.

Difference of Gaussian is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original. DoG creates another set of images, for each octave, by subtracting every image from the previous image in the same scale.

Step 2. Key Point Localization

Key point localization involves the refinement of keypoints selected in the previous stage. Low contrast key-points, unstable key points, and keypoints lying on edges are eliminated. This is achieved by calculating the Laplacian of the keypoints found in the previous stage.

Step 3. Assigning Orientation to Keypoints

To achieve detection which is invariant with respect to the rotation of the image, orientation needs to be calculated for the key-points. This is done by considering the neighborhood of the keypoint and calculating the magnitude and direction of gradients of the neighborhood. Based on the values obtained, a histogram is constructed with 36 bins to represent 360 degrees of orientation(10 degrees per bin). Thus, if the gradient direction of a certain point is, say, 67.8 degrees, a value, proportional to the gradient magnitude of this point, is added to the bin representing 60-70 degrees. Histogram peaks above 80% are converted into a new keypoint are used to decide the orientation of the original keypoint. Formulas used are -

$$\textbf{Magnitude} = \sqrt{[(Gx)2+(Gy)2]} = 16.64$$

$$\Phi = \text{atan}(Gy / Gx) = \text{atan}(1.55) = 57.17$$

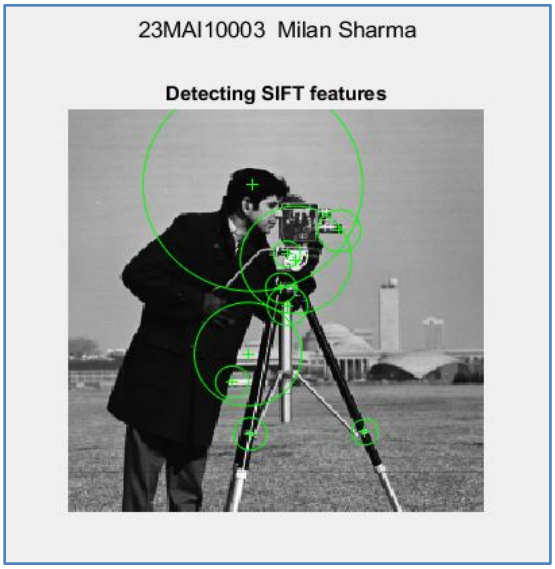
Step 4. Key Point Descriptor

Finally, for each keypoint, a descriptor is created using the keypoints neighborhood. These descriptors are used for matching keypoints across images. A 16×16 neighborhood of the keypoint is used for defining the descriptor of that key-point. This 16×16 neighborhood is divided into sub-block. Each such sub-block is a non-overlapping, contiguous, 4×4 neighborhood. Subsequently, for each sub-block, an 8 bin orientation is created similarly as discussed in Orientation Assignment. These 128 bin values (16 sub-blocks * 8 bins per block) are represented as a vector to generate the keypoint descriptor.

Code (Using detectSIFTFeatures function):

```
% Load an Image
I = imread('cameraman.tif');

% Detect SIFT features in the image.
points = detectSIFTFeatures(I);

% Displaying the results
imshow(I);
title("Detecting SIFT features")
hold on;
plot(points.selectStrongest(10))

sgtitle("23MAI10003  Milan Sharma")
```

Output :



Code (Using SIFT and SIFTKeypointVizualizer functions):

KeyPoint Class –

```
classdef KeyPoint
    %KeyPoint extracted using SIFT.

    properties
        Coordinates = []
        Magnitude
        Direction
        Descriptor
        Octave
        Scale
    end

    methods
        function [x,y] = coordinates(obj)
            % Returns Coordinates
            x = obj.Coordinates(1);
            y = obj.Coordinates(2);
        end
        function co = magnitude(obj)
            % Returns Magnitude
            co = obj.Magnitude;
        end
        function co = direction(obj)
            % Returns direction
            co = obj.Direction;
        end
        function co = descriptor(obj)
```

```
                % Returns descriptor
                co = obj.Descriptor;
            end
            function co = octave(obj)
                % Returns octave
                co = obj.Octave;
            end
            function co = scale(obj)
                % Returns scale
                co = obj.Scale;
            end
        end
    end
```

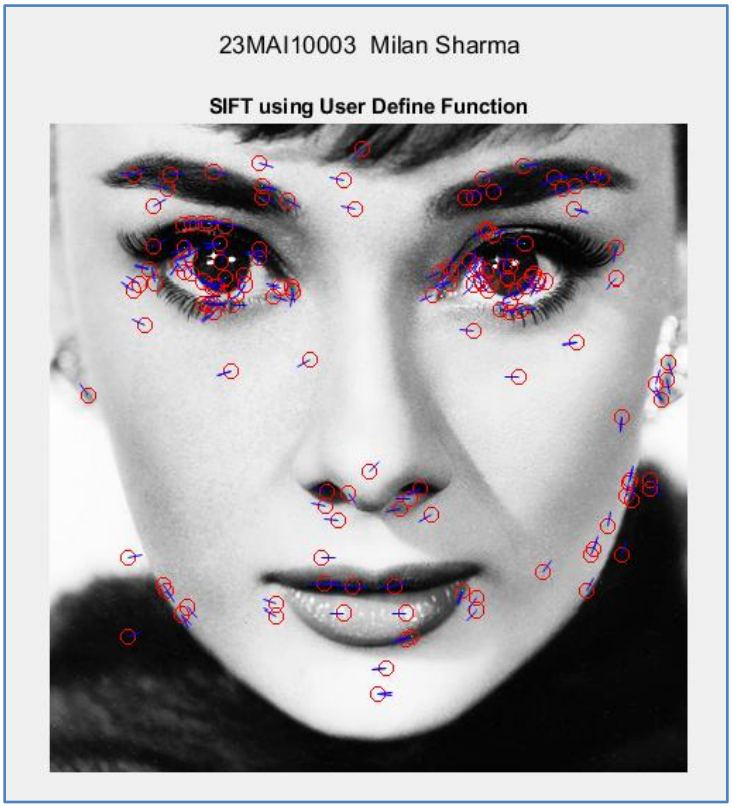
SIFTKeypointVisualizer Function –

```
    function image = SIFTKeypointVisualizer(Image, KeyPoints)
    % This function creates an Image to visualize the SIFT features
    % Input image should be gray scale.

        % Creating color image.
        Image = cat(3, Image, Image, Image);
        for i=1:length(KeyPoints)
            [x, y] = KeyPoints{i}.coordinates();
            circle = [y,x,5];
            % adding circles to key point locations
            image =
    insertShape(image,'circle',circle,'LineWidth',1,'color',[255,0,0],'SmoothEdges',false
    );
        end
        for i=1:length(KeyPoints)
            [x, y] = KeyPoints{i}.coordinates();
            dir = KeyPoints{i}.direction();
            line = [y,x,y+10*sind(dir),x+10*cosd(dir)];
            % adding lines to key point locations
            image = insertShape(image,'line',line,'LineWidth',1,'color',[0,0,255]);
        end
        for i=1:6:length(KeyPoints)
            [x, y] = KeyPoints{i}.coordinates();
            % Distinguishing key point location with green dots
            image(x,y,☺ = [0,255,0];
        end
    end
```

Main Program to implement SIFT –

```
    clear;
    clc;
    image = imread('audreyhepburn.png');
    image = rgb2gray(image);
    image = double(image);
    keyPoints = SIFT(image,3,5,1.3);
    image = SIFTKeypointVisualizer(image,keyPoints);
    imshow(uint8(image))
```

Output :

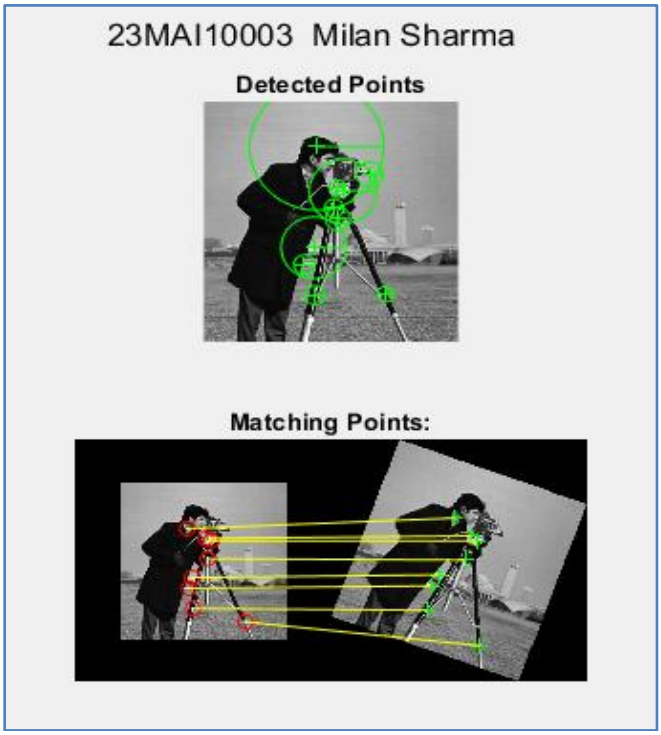


Code (Detecting the Matching Points in two images) –

```
% Read the image
I = imread('cameraman.tif');

% Detect SIFT features in the image
points = detectSIFTFeatures(I);

% Display the image
subplot(2,1,1), imshow(I);
title("Detected Points")
hold on;
% Overlay the strongest 10 SIFT features on the image and show their orientations
plot(points.selectStrongest(10),'showOrientation',true);
hold off;

% Read two images
I1 = imread("cameraman.tif");
I2 = imresize(imrotate(I1,-20),1.2);

% Detect SIFT features
points1 = detectSIFTFeatures(I1);
points2 = detectSIFTFeatures(I2);
```

```
% Extract feature descriptors
[features1, valid_points1] = extractFeatures(I1, points1.selectStrongest(30));
[features2, valid_points2] = extractFeatures(I2, points2.selectStrongest(30));

% Match features
indexPairs = matchFeatures(features1, features2);

% Retrieve matched points
matchedPoints1 = valid_points1(indexPairs(:,1), :);
matchedPoints2 = valid_points2(indexPairs(:,2), :);

subplot(2,1,2)
showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2,'montage');
title('Matching Points:');

sgtitle("23MAI10003   Milan Sharma")
```

Output –



Learning outcomes (What I have learnt):

1. **Learnt about SIFT (Scale Invariant Feature Transform) Detector.**
2. **Learnt about various steps to identify the various Localized feature.**
3. **Learnt about NLoG and DoG used in SIFT.**