Report 03
Submitted by
Mahmood Ul Hassan

Make a report of the following contents.
1. Summarize Semi-Global Block Matching method on about two pages.

The followings are optional.
2. Make a program of image matching using SIFT or SURF in MATLAB or OpenCV and explain the program with the source and the result(s).

Semi-Global Block Matching Method

The Semi-global matching (SGM) method was introduced by Heiko Hirschmuller for the estimation of a dense disparity map from a pair of rectified stereo image. Semi-Global Matching successfully integrate the idea of global and local stereo method for accurate, pixel-wise matching at low runtime. According to Heiko Hirschmuller. "The Semi global Matching (SGM) technique is based on the idea of pixelwise matching of mutual information and approximating a global, 2D smoothness constraint by combining many 1D constraints.[1]" The core algorithm considers pairs of images with known intrinsic and extrinsic orientation. The method has been implemented for the rectified and unrectified images. For the unrectified images, epipolar lines are efficiently computed and followed explicitly while matching [2].

The algorithm is described in following distinct processing steps.

1) Pixel wise Matching Cost Calculation
2) Directional Cost Calculation
3) Post Processing

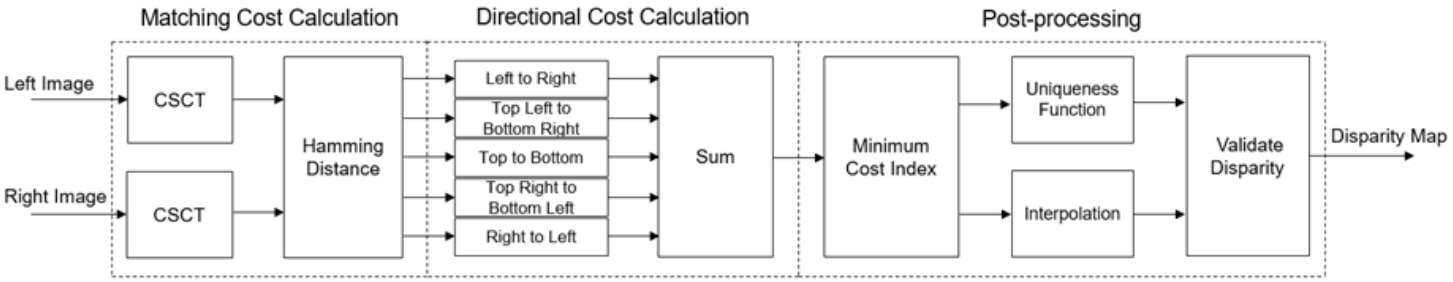

Figure 1 Steps for the Semi-global matching [3]

1) Pixel Wise Matching Cost Calculation

There are many methods in literature for computing the matching cost. Census transform is explained here as given in [4]. This step is sub divided into two sub steps:

a) Center-Symmetric Census Transform (CSCT) of left and right images
b) Hamming Distance computation

2) Directional Cost Calculation

Most of time, because of noise, the matching cost result is ambiguous and sometime wrong match could have lower cost than the correct ones. Hence more constraints are needed to increase the smoothness by penalizing changes of neighboring disparities. This constraint can be obtained by aggregating 1-D minimum cost paths from multiple different directions. The constraints are represented in term of energy E(D) that depends on the disparity image D:

$$E(D) = \sum_{p} \left(C(p, D_p) + \sum_{q \in N_p} P_1 T\left[|D_p - D_q| = 1\right] + \sum_{q \in N_p} P_2 T\left[|D_p - D_q| > 1\right] \right)$$

The first term of above equation represent the summation of all pixel matching costs for the disparities of D. The second term adds a constant penalty P_1 for all pixels q in the neighborhood Np of p. The last term adds up a larger constant penalty P2 for lager disparity changes. The aggregated cost at each pixel position from r different directions is represented as

$$S(p, d) = \sum_{r} L_r(p, d)$$

where $L_r(p, d)$ is current cost of pixel p and disparity d in direction r

3) Post Processing

This step can be divided into following three steps

a) Minimum cost index calculation
b) Interpolation
c) Uniqueness function

Minimum cost index calculation estimate the index corresponding to the minimum cost for a given pixel. The disparities at the sun-pixel level is resolved by applying sub-pixel quadratic interpolation. The reliability and accuracy of the computed minimum disparity is ensured by uniqueness function. If the value of uniqueness threshold is higher, more disparities are unreliable. As a last step, the negative disparity values are invalidated and replaced with -1.

Reference

1) Heiko Hirschmuller “Stereo Processing by Semi global Matching and Mutual Information,” IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 30, NO. 2, FEBRUARY 2008
2) Hirschmüller, H. (2005),“Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information”, IEEE Conference on Computer Vision and Pattern Recognition, June 2005, San Diego, CA, USA, Volume 2, pp. 807-814.
3) https://jp.mathworks.com/help/visionhdl/ug/stereoscopic-disparity.html
4) Spangenberg R., Langner T., and Rojas R., “Weighted Semi-Global matching and Center-Symmetric Census Transform for Robust Driver Assistance”, Computer Analysis of Images and Patterns, 2013.

2. Make a program of image matching using SURF in MATLAB.

The code for the image matching consists of three main steps:

I. Detection: Identify the interest points
II. Description: Extract feature vector descriptor surrounding each interest points
III. Matching: Determine the correspondence between descriptors in two views

For image matching, we used SURF in MATLAB. The image used in this code is the famous Mosque in Hangzhou city of China, This image is captured by myself during my visit.

Step (0): Load the Sample Frame and Convert to Gray Scale Image

```
% Load Sample frame
I1_rgb = imread('m111.jpg');
I2_rgb = imread('m222.jpg');
% Convert the sample frame to gray scale
I1 = rgb2gray(I1_rgb);
I2 = rgb2gray(I2_rgb);

% Visualize the original images
figure
subplot(1,2,1)
imshow(I1_rgb)
title('Original Frame 1')

subplot(1,2,2)
imshow(I2_rgb)
title('Original Frame 2')
%set(gcf,'position',[x0,y0,width,height])
set(gcf,'position',[150,150,880,420])

% Visualize the Gray scale images
figure
subplot(1,2,1)
imshow(I1)
title('Grayed Frame 1')

subplot(1,2,2)
imshow(I2)
title('Grayed Frame 2')
set(gcf,'position',[150,150,880,420])
```

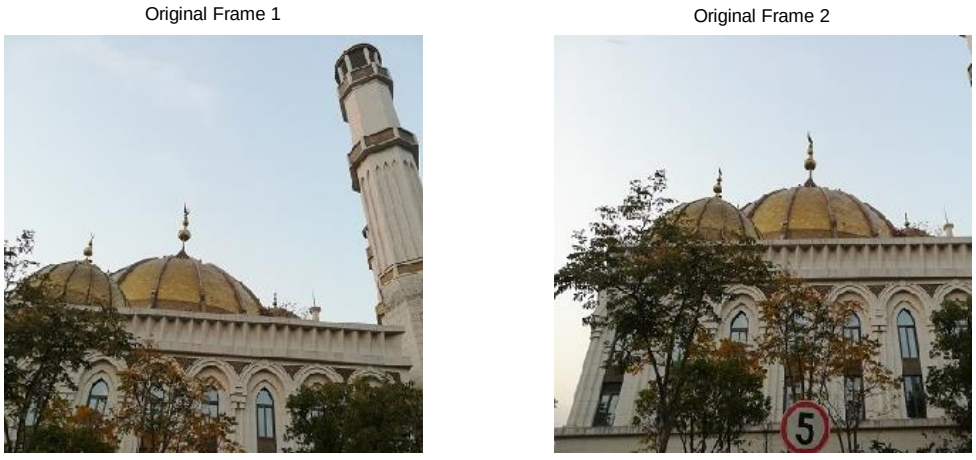

Figure 2 Two different views of same scene named as original frame 01 and 02

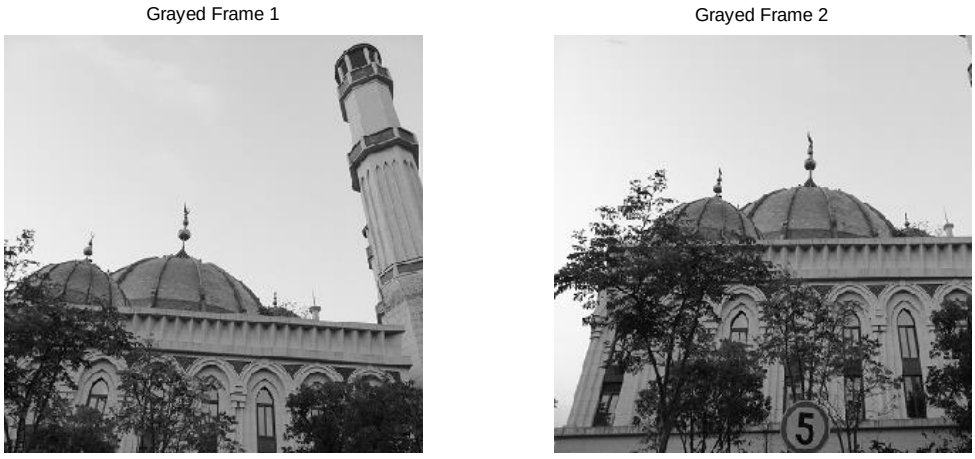

Figure 3 Result of gray scale conversion of original frames

Step (I): Detection (Identify the interest points)

Case 1: Detecting all feature points using Default Values of "`detectSURFFeatures`" function in Matlab

```
%Find the Feature.

points1 = detectSURFFeatures(I1);
points2 = detectSURFFeatures(I2);

% Visualize the Features
figure
subplot(1,2,1)
imshow(I1_rgb)
hold on;
plot(points1)
title('All features Frame 1')
subplot(1,2,2)
imshow(I2_rgb)
hold on;
plot(points2)
title('All features Frame 2')
set(gcf,'position',[150,150,880,420])
```

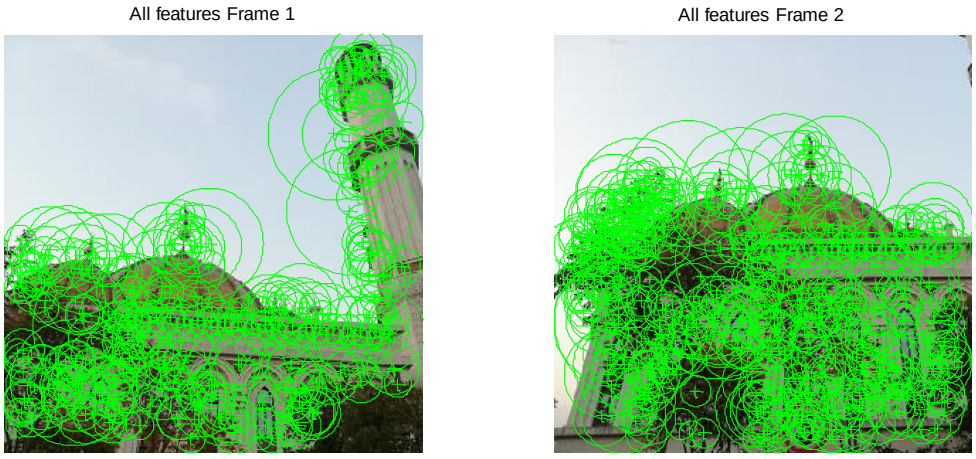

Figure 4 Results of all detected feature points

Case 2: Detecting 200 strongest feature points

```
% Find the Specefic Feature
specfic_feature1 = selectStrongest(points1,200)
specfic_feature2 = selectStrongest(points2,200)
```

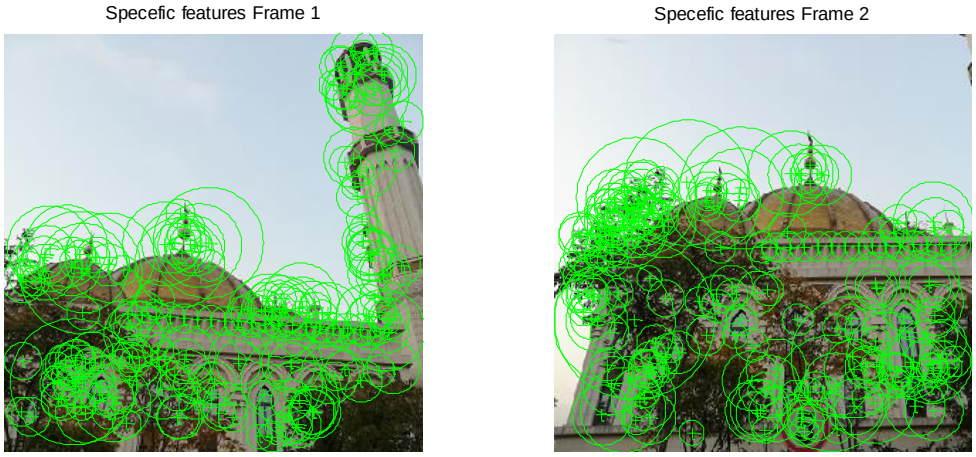

Figure 5 Results of 200 strongest feature points

Case 3: Detecting the feature points by selecting “NumOctaves = 1”

```
% Find the Specefic Feature
specfic_feature1 = detectSURFFeatures(I1, 'NumOctaves',1);
specfic_feature2 = detectSURFFeatures(I2, 'NumOctaves',1);
```

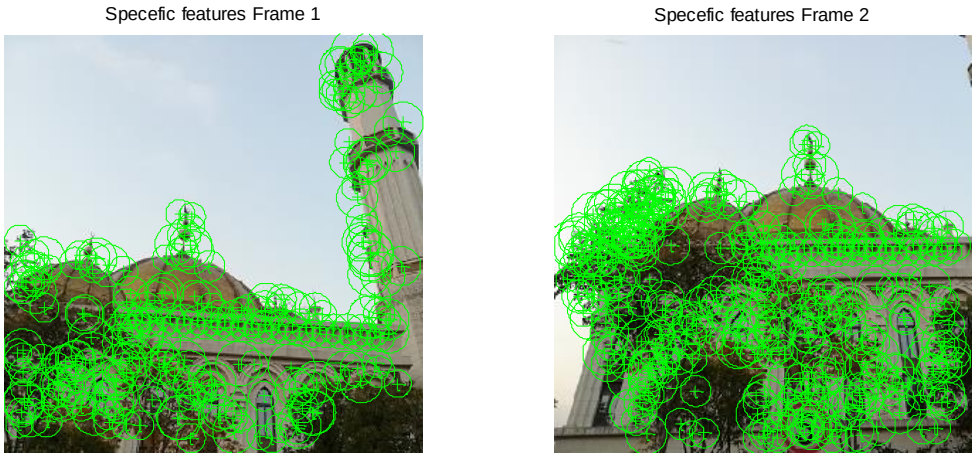

Figure 6 Results of feature points by selecting "NumOctaves = 1"

Case 4: Detecting the feature points by selecting "NumOctaves = 5"

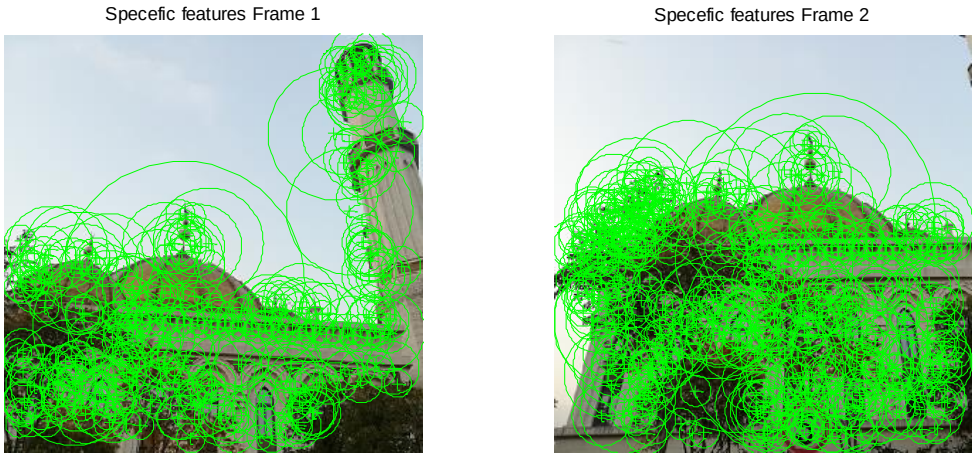

Figure 7 Result of feature points by selecting "NumOctaves = 5"

Case 5: Detecting the feature points by selecting "MetricThreshold = 2000"

```
specfic_feature1 = detectSURFFeatures(I1,'MetricThreshold',2000);
specfic_feature2 = detectSURFFeatures(I2,'MetricThreshold',2000);
```

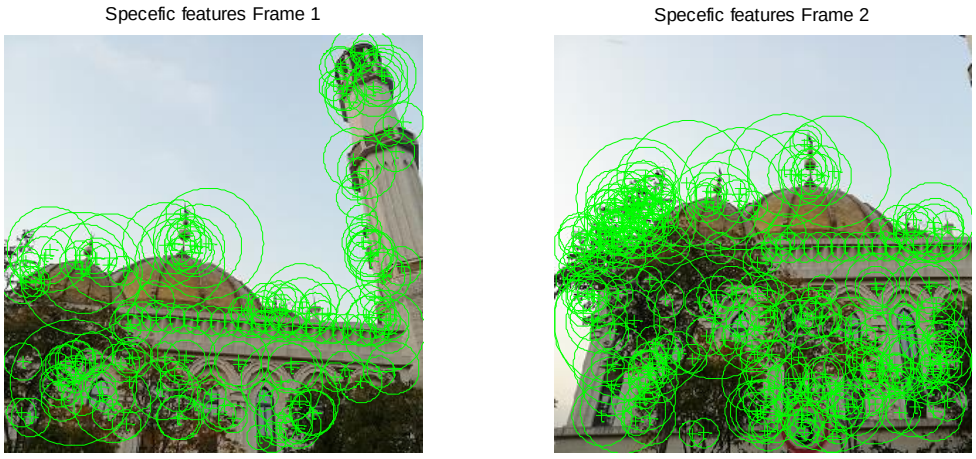

Figure 8 Results of feature points by selecting "MetricThreshold = 2000"

Case 6: Detecting the feature points by selecting "MetricThreshold = 2000" and "NumOctaves = 1".

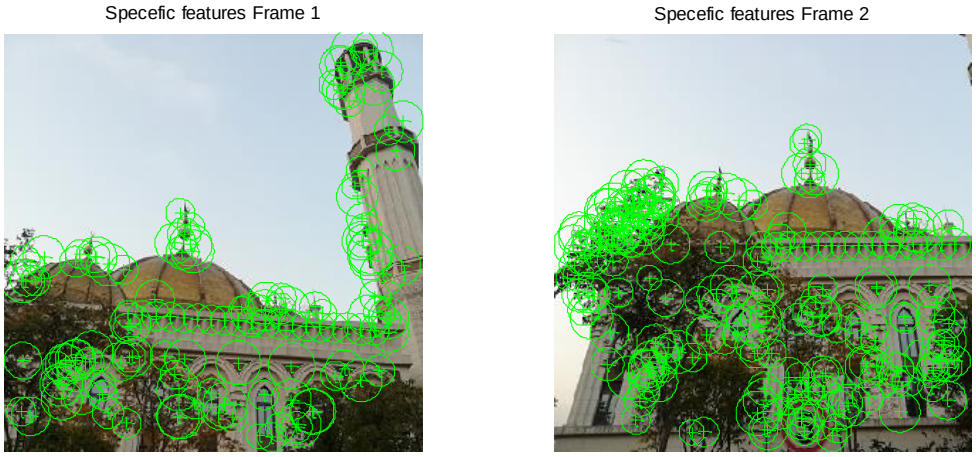

Figure 9 Results of feature points by selecting "MetricThreshold = 2000" and "NumOctaves = 1"

Step II: Description (Extract feature vector descriptor surrounding each interest points)

```
%Extract the neighborhood features.
[features1,valid_points1] = extractFeatures(I1,points1);
[features2,valid_points2] = extractFeatures(I2,points2);

Figure, subplot(1,2,1), imshow(I1_rgb),hold on;
plot(valid_points1); title('Extracted features Frame 1')
subplot(1,2,2),imshow(I2_rgb) , hold on;
plot(valid_points2);title('Extracted features Frame 2')
set(gcf,'position',[150,150,880,420])
```

Case 1: Extracting all feature points using Default Values of SURF function in Matlab

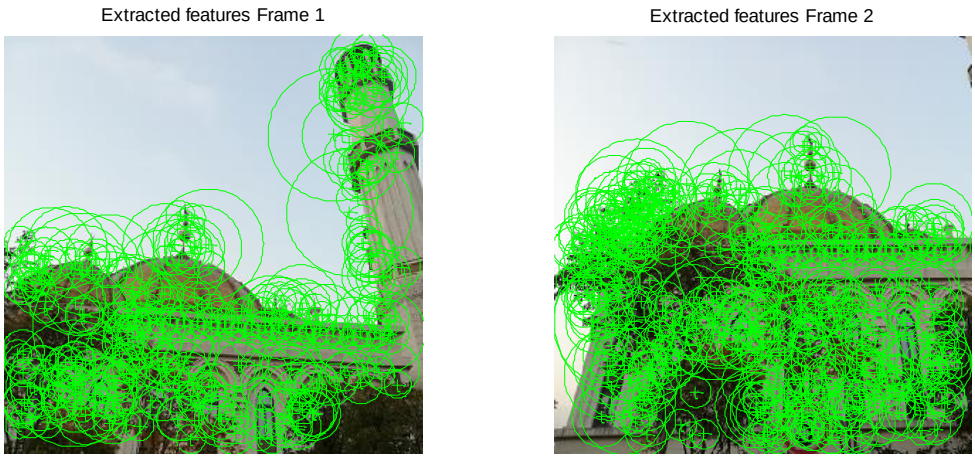

Figure 10 Extracting all feature points

Step III: Matching (Determine the correspondence between descriptors in two views)

Case 1: Image matching by all feature points

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

%Retrieve the locations of the corresponding points for each image.
matchedPoints1 = valid_points1(indexPairs(:,1),:);
matchedPoints2 = valid_points2(indexPairs(:,2),:);

%Visualize the corresponding points.
figure; showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2);
figure;showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2,'montage');
```

Figure 11 Image matching by all feature points

Figure 12 Image matching by all feature points

Case 2: Image matching using 200 strongest feature points

Figure 13 Image matching using 200 strongest feature points

Figure 14 Image matching using 200 strongest feature points

Case 3: Image matching by all feature points by selecting “NumOctaves = 1 of “`detectSURFFeatures`” function in Matlab

Figure 15 Image matching by all feature points by selecting “NumOctaves = 1

Figure 16 Image matching by all feature points by selecting “NumOctaves = 1

Case 4: Image matching by all feature points by selecting “NumOctaves = 5” of “`detectSURFFeatures`” function in Matlab

Figure 17 Image matching by all feature points by selecting “NumOctaves = 5

Figure 18 Image matching by all feature points by selecting “NumOctaves = 5

Case 5: Image matching by all feature points by selecting “MetricThreshold = 2000”, of “`detectSURFFeatures`” function in Matlab

Figure 19 Image matching by all feature points by selecting “MetricThreshold = 2000”

Figure 20 Image matching by all feature points by selecting “MetricThreshold = 2000”

Case 6: Image matching by all feature points by selecting “MetricThreshold = 2000”, and “NumOctaves = 3” of “`detectSURFFeatures`” function in Matlab

Figure 21 Image matching by all feature points by selecting “MetricThreshold = 2000”, and “NumOctaves = 3”

Figure 22 Image matching by all feature points by selecting “MetricThreshold = 2000”, and “NumOctaves = 3”