Ritesh Sharma

CS 556 HW#1

21st January, 2016

1.

In order to detect SURF points, first we need to read file using following syntax provided in MatLab:

I = imread(filename);

Since the above command returns integer values, we convert the image into double precision using below Matlab syntax:

I = im2double(I);

Then we change the image into gray scale,

Img = rgb2gray(I);

Once the image is converted into grayscale, we can use detectSURFFeature() to find the interest points. This command returns SURFPoints Object.

SURFPoints=detectSURFFeatures(img);

Then we extract SURF Features for each surfpoints using the below command:

[SURFFeatures,SURFPoints]=extractFeatures(img,SURFPoints);

In order to plot top strongest points, following command can be used:

plot(selectStrongest(SURFPoints,100));

Note: The code for step 1 is provided at the location "/Scene#". My code reads all the files in the folder and the process them together and generate results for each image in one go.

Results:

The results for each scene is provided at the location "Scene#/OutputForProblem1/" . One of the result from scene is shown in Figure 1.:

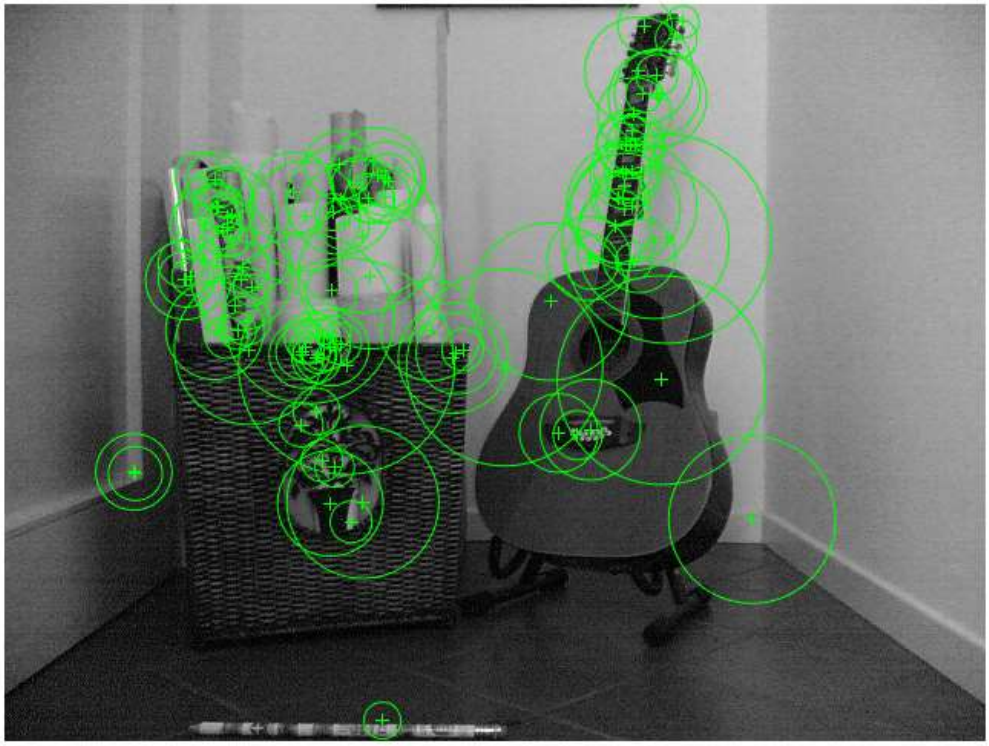

Figure 1. Top 100 SURF Points are shown with green blobs and plus sign at the center

2.

(1)

In order to show matched points, we need to perfrom step 1 for each scene image and the object image and then use the function matchFeatures(… , …) given below:

boxPairs=matchFeatures(boxFeatures,sceneFeatures);

It was found that some of the object image didn't had enough surf points to be matched with the scene image inspite of object being present in the scene. So I had to use MatchThreshold and MaxRatio parameter in the function.

MatchThreshold: The threshold value provided along side MatchThreshold in the function above represents a percent of the distance from a perfect match. Two feature vectors match when the distance between them is less than the threshold set by MatchThreshold. The function rejects a match when the distance between the features is greater than the value of MatchThreshold. Increasing the value return more matches.

MaxRatio: This rejects the ambiguous matches. Increasing the ratio returns more matches.

So, I used the following syntax to get atleast some value so that the object can be matched with the object in the scene.

boxPairs=matchFeatures(boxFeatures,sceneFeatures,'MatchThreshold',2,'MaxRatio',0.9);

To show image next to each other, following function have been used in the code provided with this document:

showMatchedFeatures(boxImage, sceneImage, boxPoints(boxPairs(:, 1),:),scenePoints(boxPairs(:, 2), :), 'montage');

Results:

The results for each scene is provided at the location "Scene#/OutputForProblem2/ Scene_And_ Template" . One of the result from scene is shown in Figure 2:

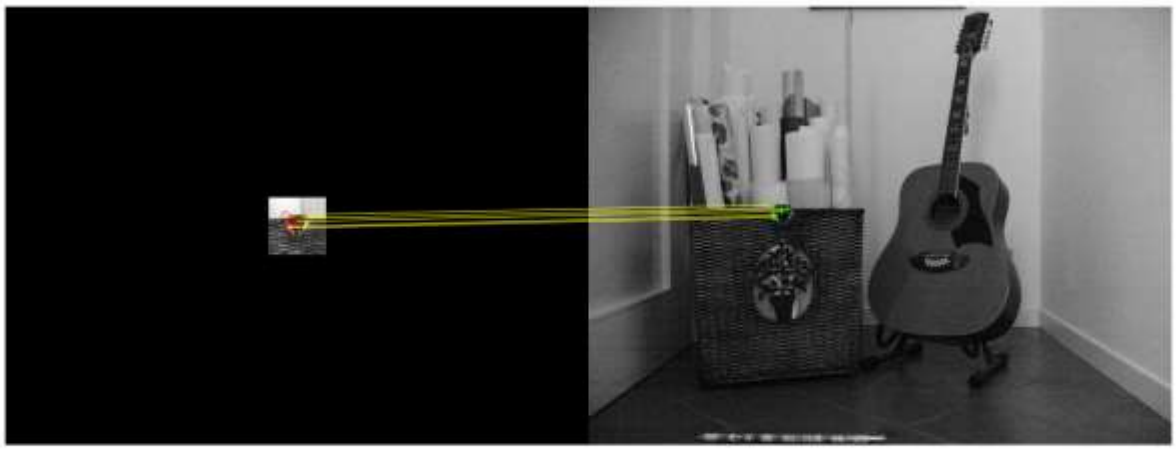

Figure 2

(2)

In order to find the median , I have used median() function provided by matlab. A part of code for finding the median is shown below:

Detectedpoint = scenePoints(boxPairs(:, 2), :).Location;

plot(detectedpoint(:,1),detectedpoint(:,2),'O', 'MarkerSize', 5);

```
if(numel(detectedpoint(:,1))~=1)
    point=median(scenePoints(boxPairs(:, 2), :).Location);
    plot(point(1,1),point(1,2),'r+', 'MarkerSize', 25);
else
    point=detectedpoint(:,:);
    plot(point(1,1),point(1,2),'r+', 'MarkerSize', 25);
end
```

The basic strategy here is to find the median only if the number of detectedpoints in the image is greater than 1.

Results:

The results for each scene is provided at the location "Scene#/OutputForProblem2/ Median_with _SurfPoints" . One of the result from scene 1 is shown in Figure 3:

Figure 3. Median is shown with red colored cross and the detected points are shown with blue blobs

3.

In order to find the ground truth, I used **impixelinfo** functionality provided in Matlab. This helps in viewing the pixel information on the image. With the help of mouse I found the dimension of the image and then compute the diagonal of the image. The center of the object was taken as ground truth. Then I computed the distance between ground truth and the median of the object with different values of thetha. If thetha is larger than the diaogonal of the object, it is considered to be as False Positive (FP) and if its smaller then it is considered as True Positive (TP). Then I computed Recall and Precision using the formula given below:

$$Precision(\theta) = \frac{TP(\theta)}{TP(\theta) + FP(\theta)}$$

$$Recall(\theta) = \frac{TP(\theta)}{Number\ of\ all\ objects}$$

The data for the calculation is provided in the excel sheet attached in the zip folder. ROC curves for the Scene 1, Scene 2 and Scene 3 are shown in Figure 1, Figure 2 and Figure 3 respectively.

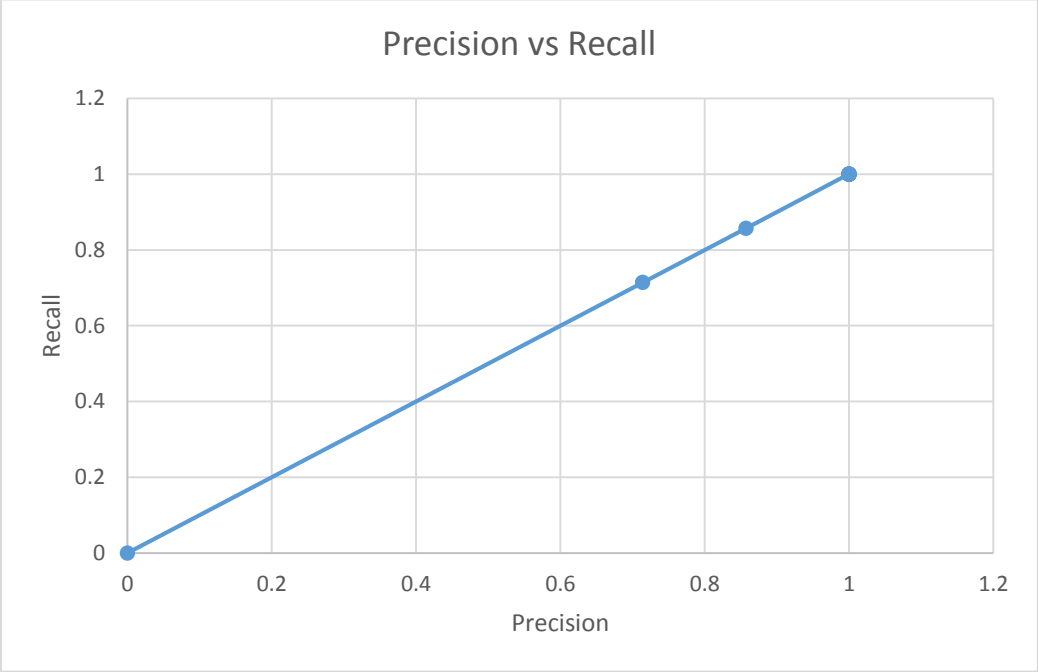

Figure 4. ROC Curve for Scene 1

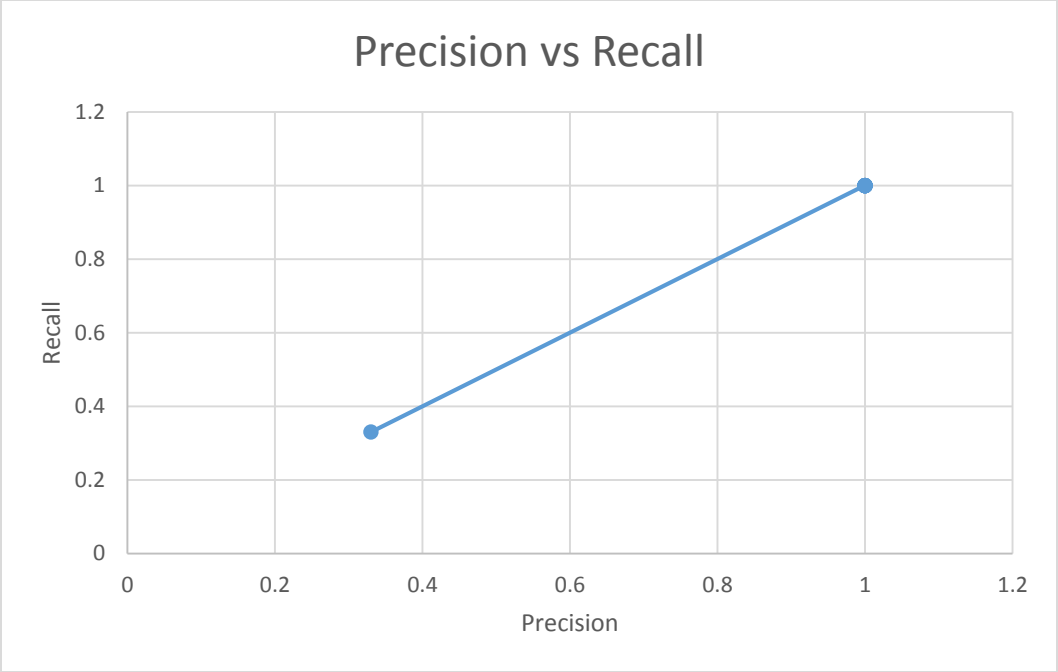

Figure 5. ROC Curve for Scene 2

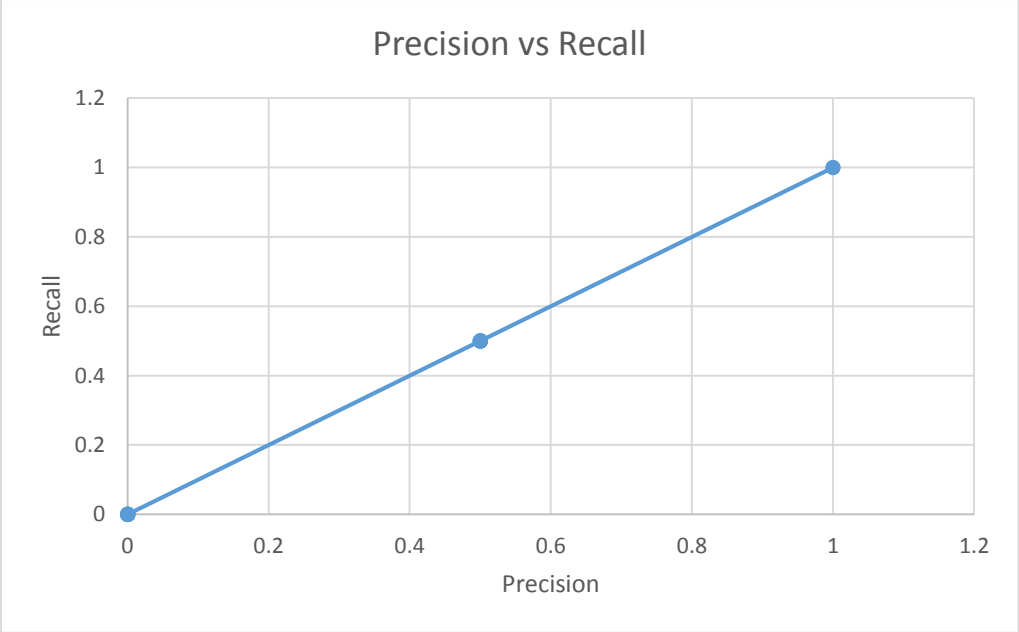

Figure 6. ROC Curve for Scene 3