Sensors and Control for Mechatronics Systems
Tutorial 5

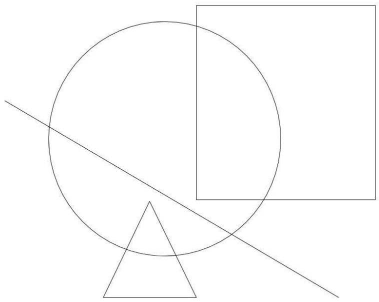
Question 1: Harris Corner Detection

1.1 : Import ***checkerboard.jpg*** image to MATLAB workspace. Convert it into a grayscale image. Use ***detectHarrisFeatures*** function to detect corner points in the grayscale checkerboard image. You can plot the location of the detected corners on top of the image using the following commands. (https://au.mathworks.com/help/vision/ref/detectharrisfeatures.html)

```
I = imread('checkerboard.jpg');
I = rgb2gray(I);
cornerPoints = detectHarrisFeatures(I);
imshow(I)
hold on
plot(cornerPoints)
```

Observe the ***cornerPoints*** object that contains the detected corner points. Note that for each detected corner point, there exists a corresponding value called “Metric”. This value indicates how strong the detected feature is.

1.2 : In the ***harris_corners_example.jpg*** image shown below, how many corners do you expect to be detected?



1.3 : Detect corner features of the ***harris_corners_example.jpg*** in MATLAB following the steps in 1.1. How many features are actually detected? Plot them on top of the image. using your knowledge on the Harris-Stephen corner detection algorithm, explain why this happens.

1.4 : The ***detectHarrisFeatures*** function optionally accepts an argument ‘MinQuality’ which is the fraction of the maximum corner metric value that a pixel should achieve to be detected as a corner. Find a good value for ‘MinQuality’ of corners such that all real corners are detected while the number of outliers is minimal.

Question 2: SURF Feature Extraction and Matching

2.1 : Detect SURF features in the grayscale images of roofs1.jpg and roofs2.jpg and extract them.

```
points1 = detectSURFFeatures(I1gs);
points2 = detectSURFFeatures(I2gs);
[features1, validPoints1] = extractFeatures(I1gs, points1);
[features2, validPoints2] = extractFeatures(I2gs, points2);
```

extractFeatures function returns two arrays. The first array is the set of feature descriptors. The second array has the corresponding location for each feature.

2.2 : Match the features between the two images.

```
indexPairs = matchFeatures(features1, features2);
matchedPoints1 = validPoints1(indexPairs(:,1));
matchedPoints2 = validPoints2(indexPairs(:,2));
```

2.3 : Visualize the matching features using the following command.

```
showMatchedFeatures(I1gs, I2gs, matchedPoints1, matchedPoints2, 'montage')
```

2.4: Repeat exercise with “**detectORBFeatures**” function.

Question 3:. RANSAC Outlier Rejection

3.1 : Find image rotation and scale using automated feature matching. Load images ‘kfc1.jpg’ and ‘‘kfc2.jpg’ to MATLAB workspace.

```
original = rgb2gray(imread('kfc1.jpg'));
distorted = rgb2gray(imread('kfc2.jpg'));
```

Note : The second image is taken after rotating the camera by an unknown angle after capturing the first image. The goal of this exercise is to un-rotate the second image using feature matching.

3.2 : Detect features in both images similar to that in step 3.1

```
ptsOriginal = detectSURFFeatures(original);
ptsDistorted = detectSURFFeatures(distorted);
[featuresOriginal, validPtsOriginal] = extractFeatures(original, ptsOriginal);
[featuresDistorted, validPtsDistorted] = extractFeatures(distorted,
ptsDistorted);
```

3.3 : Match features between the two images

```
indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
matchedOriginal = validPtsOriginal(indexPairs(:,1));
matchedDistorted = validPtsDistorted(indexPairs(:,2));
```

3.4 : Visualize the matching features. You will notice outliers.

```
figure;
showMatchedFeatures(original,distorted,matchedOriginal,matchedDistorted);
```

3.5 : Estimate the transform between the two images

```
[tform, inlierDistorted, inlierOriginal] = estimateGeometricTransform(matchedDistorted,
                                           matchedOriginal, 'similarity');
```

The function estimateGeometricTransform estimates the geometric transformation between the two images using MSAC (a variant of RANSAC). While fitting the detected feature pairs to a transformation matrix, it removes the outliers.

3.6 : View the matching feature points with outliers removed.

```
figure;
showMatchedFeatures(original,distorted,inlierOriginal,inlierDistorted);
title('Matching points (inliers only)');
legend('ptsOriginal','ptsDistorted');
```

3.7 : Recover the scale and rotation of the second image using the inverse of the transformation Matrix.

```
Tinv = tform.invert.T;
ss = Tinv(2,1);
sc = Tinv(1,1);
scaleRecovered = sqrt(ss*ss + sc*sc)
thetaRecovered = atan2(ss,sc)*180/pi
```

3.8 : Recover the second image

```
outputView = imref2d(size(original));
recovered = imwarp(distorted,tform,'OutputView',outputView);
figure, imshowpair(original,recovered,'montage')
```

You will be able to observe the ‘kfc2.jpg’ image as if it was taken from the same camera orientation as ‘kfc1.jpg’ image.