CV2015Spring—Assignment #2

Liu Xiaoyang

Apr 28, 2015

Step1: Input image

I take the object image and scene image for test. They are shown in Figure 1 and Figure 2.

Figure 1. The Object Image

Figure 2. The Scene Image

Step2: Color space conversion

I converse the color space from RGB to GRAY by using the function "rgb2gray". The GRAY image are shown as following:

Figure 3. The GRAY Object Image

Figure 4. The GRAY Scene Image

Step3: Features2D detection

The SURF algorithms is used to detect the features. I use the function "detectSURFFeatures" to complete this step. The function can find SURF

features, and return a SURFPoints object, containing information about SURF features detected in grayscale image. The main code is as following:

```
%step3 features2D detection
objectPoints = detectSURFFeatures(object);
scenePoints = detectSURFFeatures(scene);
```

In this function, the default number of strongest features is 1000, I select 150 points in object image and 300 points in scene image to show.

Figure 5. 100 Strongest Feature Points from Object

Figure 6. 300 Strongest Feature Points from Scene

Step4: Calculate descriptors

In this step, the function “extractFeatures” is used. It can return an $M \times N$ matrix of M feature vectors, also known as descriptors. Each descriptor is of length N. There are two matrices storing descriptors of the keypoints in object and scene. The code is as following:

```
%step4 calculate descriptors
[objectFeatures,objectPoints] = extractFeatures(object, objectPoints);
[sceneFeatures, scenePoints] = extractFeatures(scene, scenePoints);
```

Step5: Match descriptors

In this step, I choose the function “matchFeatures”. It returns a matrix, containing indices to the features most likely to correspond between the two input descriptor matrices. The distance calculation method is “Nearest Neighbor”, a feature vector is matched to its nearest neighbor in the other feature set. The main code is as following:

```
%step5 match descriptors
objectPairs = matchFeatures(objectFeatures, sceneFeatures);
```

The matched result is shown.

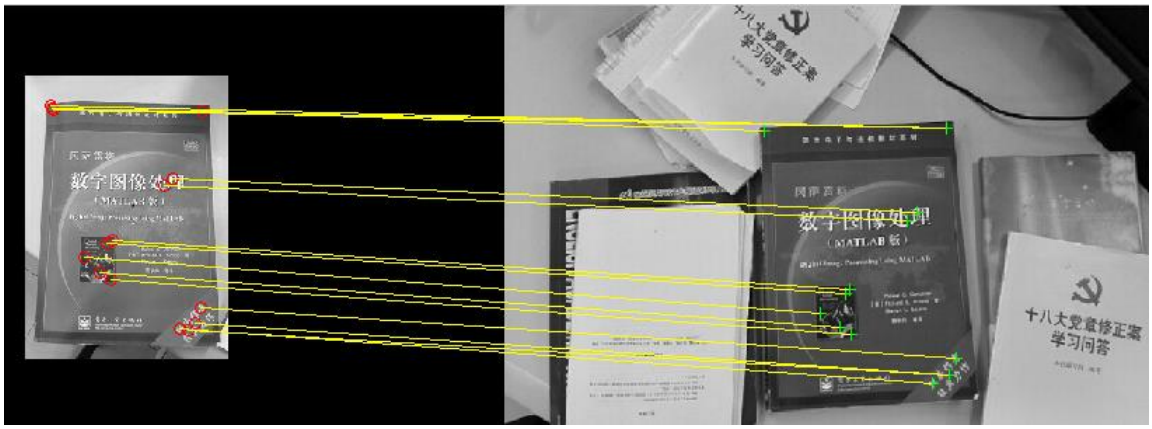

Figure 7. Matched Points

Step6：Geometric Transformation

I choose one of the Geometric transformation algorithms “affine” to map the object and scene for localizing the object. It can calculate the transformation relating the matched points and eliminate outliers. The code is as following:

```
[tform, inlierobjectPoints, inlierScenePoints] = ...
  estimateGeometricTransform(matchedobjectPoints, matchedScenePoints, 'affine');
```

Step7：Localize the object

We should get the bounding polygon of the object image. Then transform the polygon into the coordinate system of the scene image. The transformed polygon indicates the location of the object in the scene. The function “transformPointsForward” can complete the assignment. The main code is as following:

```
newobjectPolygon = transformPointsForward(tform, objectPolygon);
```

Display the detected object in Figure 8.

Figure 8. The Detected Object