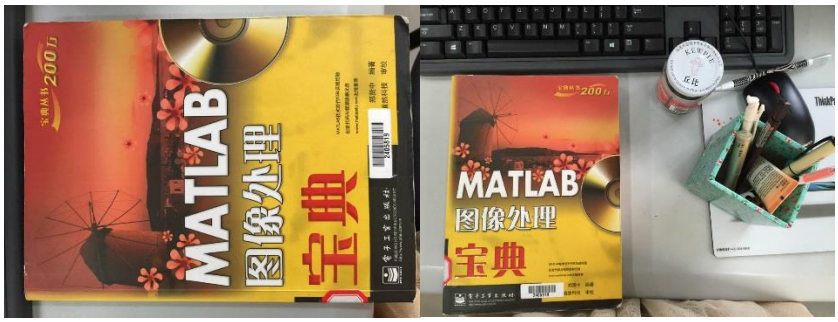
Assignment #2

For this assignment, I implemented a version of the Object Detection technique. Picture1 is the object we want to detect, picture2 is the scene which contains the object we want to detect. What I want to do is detecting and localizing the object.



Step 1&2: Input image and Color space conversion

```
image1_object = imread('object1.png');
image1_object=rgb2gray(image1_object);
figure,imshow(image1_object);
title('Image of a object');

image2_scene = imread('scene1.png');
image2_scene=rgb2gray(image2_scene);
figure,imshow(image2_scene);
title('Image of a Scene');
```

Step 3: Features2D detection

In this step, I use SURF algorithms to detect the features.

Step 4: Calculate descriptors

ExtractFeatures is called descriptors, which is used to work out an M × N matrix from M feature vectors. The length of each vector is N. As can be seen from the pictures, there are two different kinds of keypoints which can be stored in two matrices. Here are the codes.

```
[fea_object, poin_object] = extractFeatures(image1_object, poin_object);
[fea_scene, poin_scene] = extractFeatures(image2_scene, poin_scene);
```

Step 5: Match descriptors

In order to match descriptors, I used “matchFeatures” as a main function. This function can be called from the toolbox directly in Matlab.

```
objectPairs = matchFeatures(fea_object, fea_scene);
```

This piece of code returns a P-by-2 matrix, objectParis, containing indices to the features most likely to correspond between the two input feature matrices.And it worked like that,

Step6： Geometric Transformation

In geometry, an affine transformation is a function between affine spaces

which preserves points, straight lines and planes. Also, sets of parallel lines remain parallel after an affine transformation. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line. So I used “affine” to map the object and the scene.

```
[tform, inlierObjectPoints, inlierScenePoints] = ...
    estimateGeometricTransform(matchedobjectPoints, matchedScenePoints, 'affine');
figure,showMatchedFeatures(image1_object, image2_scene, inlierObjectPoints, ...
```

Step 7: Localize the object

After mapping the two pictures, in the final step, I used a bounding polygon to localize the object in the scene.” TransformPointsForward” is a function that can help us to do the complex job.

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

Here is the result,

We can see the book is well detected.