Object Detection in a Cluttered Scene Using Point Feature Matching

Effective object detection must be able to handle cluttered scenes: changes to the object size, location, orientation, and other challenges. Computer Vision System Toolbox™ offers a variety of techniques for handling challenges in object detection.

This example shows how to detect a particular object in a cluttered scene, given a reference image of the object. The example presents an algorithm for detecting a specific object based on finding point correspondences between the reference and the target image. It can detect objects despite a scale change or in-plane rotation. It is also robust to small amounts of out-of-plane rotation and occlusion.

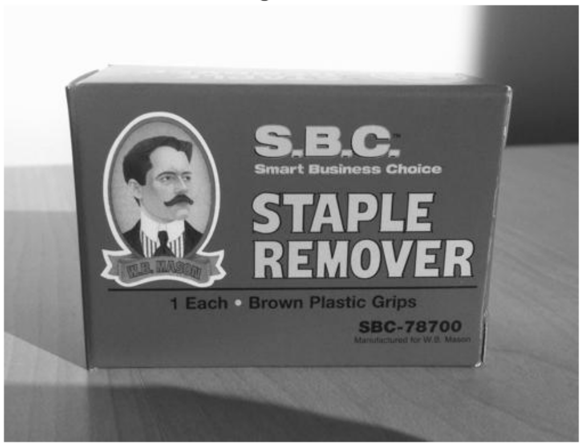
This method of object detection works best for objects that exhibit non-repeating texture patterns, which give rise to unique feature matches. This technique is not likely to work well for uniformly colored objects or for objects containing repeating patterns. Note that this algorithm is designed for detecting a specific object, for example, the elephant in the reference image, rather than any elephant.

Step 1: Reading the Images

Suppose you have an image of a cluttered scene, `clutteredDesk.jpg`, and you want to detect a particular object of which you have a separate image, `stapleRemover.jpg`. You can start by reading the reference image containing the object of interest into MATLAB®.

```
boxImage = imread('stapleRemover.jpg');
figure;
imshow(boxImage);
title('Image of a Box');
```

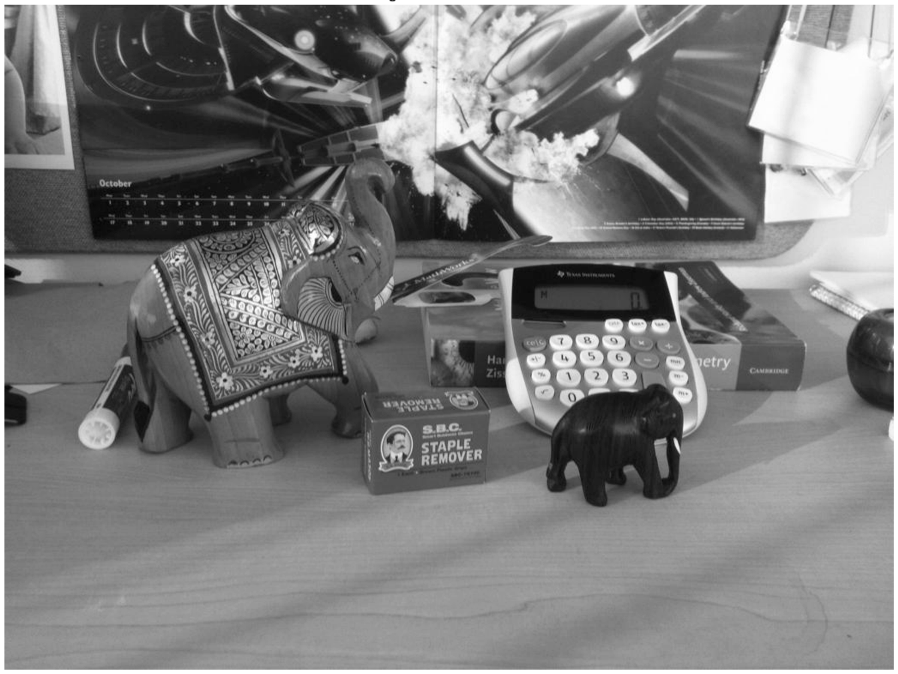
Image of a Box



Next, you can read the target image containing a cluttered scene.

```
sceneImage = imread('clutteredDesk.jpg');
figure;
imshow(sceneImage);
title('Image of a Cluttered Scene');
```

Image of a Cluttered Scene



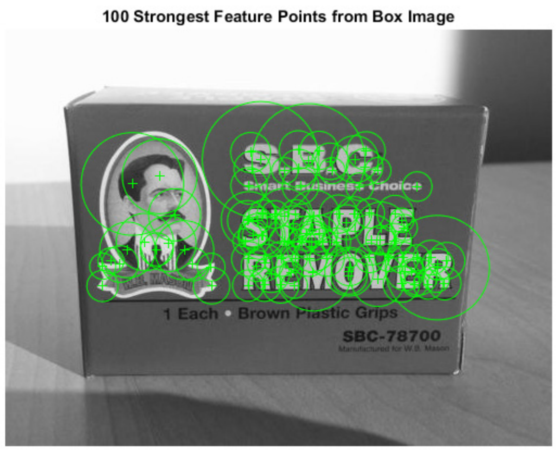
Step 2: Detecting Feature Points

You can detect feature points in both images.

```
boxPoints = detectSURFFeatures(boxImage);
scenePoints = detectSURFFeatures(sceneImage);
```

Then, you can visualize the strongest feature points found in the reference image.

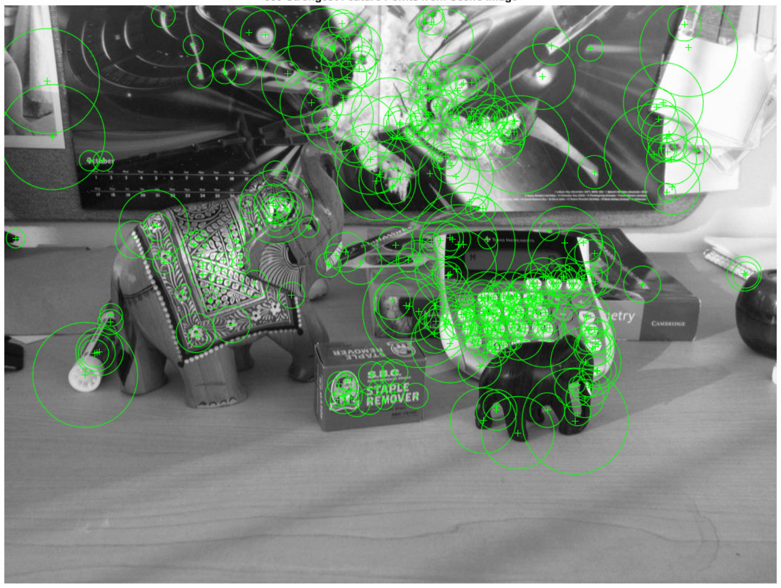
```
figure;
imshow(boxImage);
title('100 Strongest Feature Points from Box Image');
hold on;
plot(selectStrongest(boxPoints, 100));
```


100 Strongest Feature Points from Box Image
S.B.C.
Smart Business Choice
STAPLE
REMOVER
1 Each • Brown Plastic Grips
SBC-78700

For comparison, visualize the strongest feature points found in the target image.

```
figure;
imshow(sceneImage);
title('300 Strongest Feature Points from Scene Image');
hold on;
plot(selectStrongest(scenePoints, 300));
```

300 Strongest Feature Points from Scene Image



Step 3: Extracting Feature Descriptors

You can extract feature descriptors at the interest points in both images.

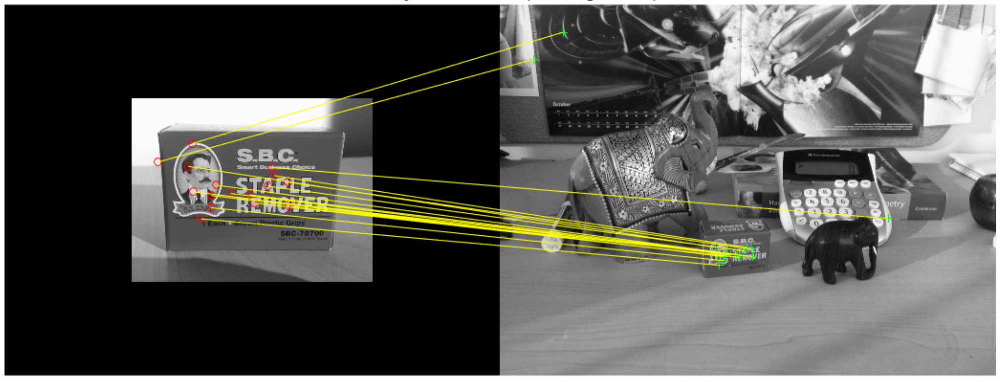
```
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);
```

Step 4: Finding Putative Point Matches

Then match the features using their descriptors.

```
boxPairs = matchFeatures(boxFeatures, sceneFeatures);
Display putatively matched features.
matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
figure;
showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
     matchedScenePoints, 'montage');
title('Putatively Matched Points (Including Outliers)');
```

Putatively Matched Points (Including Outliers)



Step 5: Locating the Object in the Scene Using Putative Matches

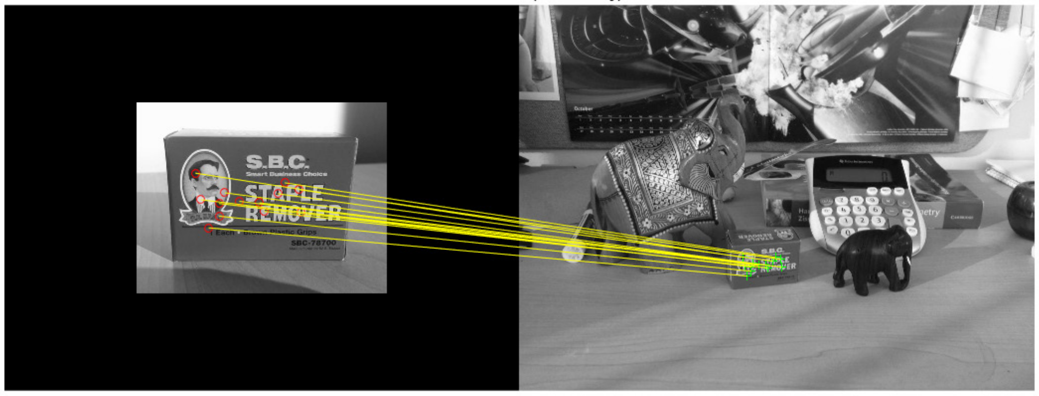
The estimateGeometricTransform function calculates the transformation relating the matched points, while eliminating outliers. This transformation allows you to localize the object in the scene.

```
[tform, inlierBoxPoints, inlierScenePoints] = ...
     estimateGeometricTransform(matchedBoxPoints, matchedScenePoints,...
     'affine');
```

You can then display the matching point pairs with the outliers removed.

```
figure;
showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
     inlierScenePoints, 'montage');
title('Matched Points (Inliers Only)');
```

Matched Points (Inliers Only)



Next, get the bounding polygon of the reference image.

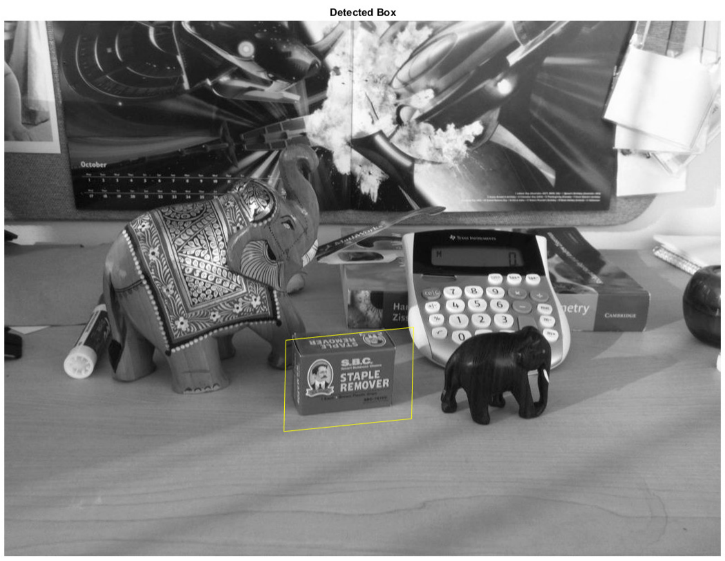
```
boxPolygon = [1, 1;...                              % top-left
         size(boxImage, 2), 1;...                   % top-right
         size(boxImage, 2), size(boxImage, 1);...   % bottom-right
         1, size(boxImage, 1);...                   % bottom-left
         1, 1];                     % top-left again to close the polygon
```

To indicate the location of the object in the scene, you can transform the polygon into the coordinate system of the target image.

```
newBoxPolygon = transformPointsForward(tform, boxPolygon);
```

Then, you can display the detected object.

```
figure;
imshow(sceneImage);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
title('Detected Box');
```



Step 6: Detecting Another Object

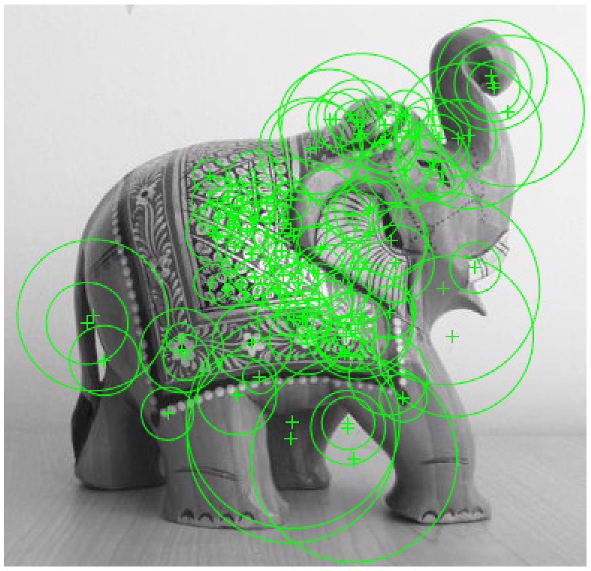
You can detect a second object by using the same steps as before. Start by reading an image containing the second object of interest, `elephant.jpg`.

```
elephantImage = imread('elephant.jpg');
figure;
imshow(elephantImage);
title('Image of an Elephant');
```

Then, detect and visualize feature points.

```
elephantPoints = detectSURFFeatures(elephantImage);
figure;
imshow(elephantImage);
hold on;
plot(selectStrongest(elephantPoints, 100));
title('100 Strongest Feature Points from Elephant Image');
```

100 Strongest Feature Points from Elephant Image



Extract feature descriptors.

```
[elephantFeatures, elephantPoints] = extractFeatures(elephantImage, ...
     elephantPoints);
```

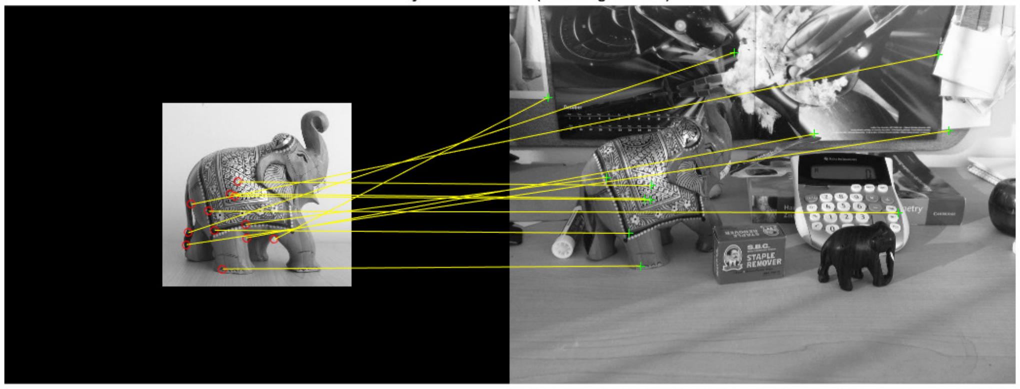
Then match features.

```
elephantPairs = matchFeatures(elephantFeatures, sceneFeatures, ...
     'MaxRatio', 0.9);
```

Display putatively matched features.

```
matchedElephantPoints = elephantPoints(elephantPairs(:, 1), :);

matchedScenePoints = scenePoints(elephantPairs(:, 2), :);

figure;

showMatchedFeatures(elephantImage, sceneImage, matchedElephantPoints, ...

     matchedScenePoints, 'montage');

title('Putatively Matched Points (Including Outliers)');
```

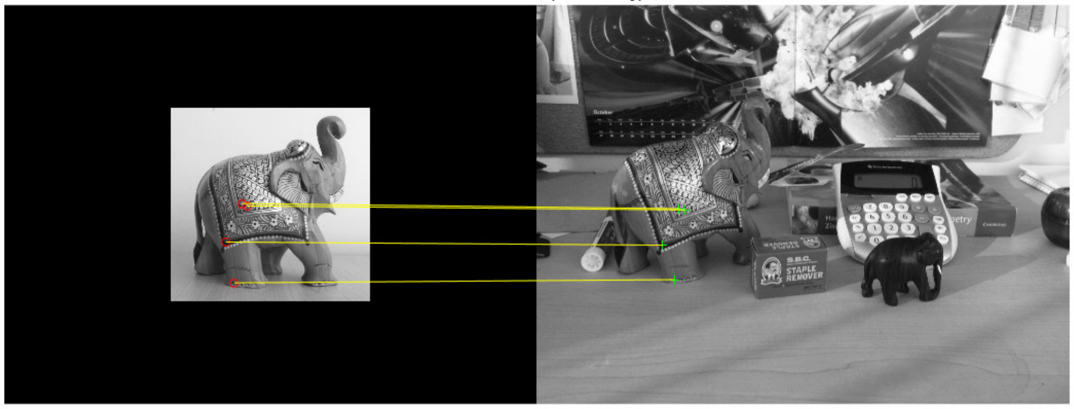
Putatively Matched Points (Including Outliers)



Then estimate the geometric transformation and eliminate outliers.

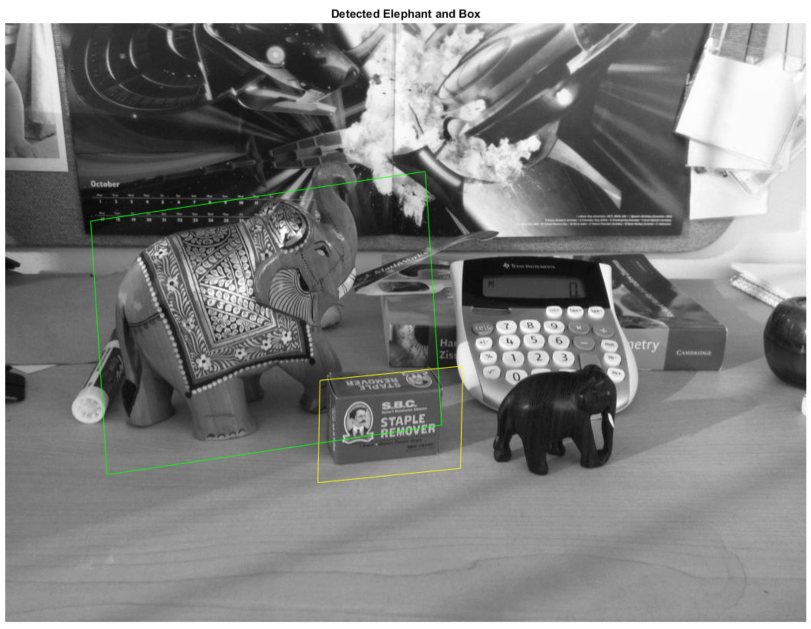
```
[tform, inlierElephantPoints, inlierScenePoints] = ...
     estimateGeometricTransform(matchedElephantPoints, ...
     matchedScenePoints, 'affine');

figure;

showMatchedFeatures(elephantImage, sceneImage, inlierElephantPoints, ...
     inlierScenePoints, 'montage');

title('Matched Points (Inliers Only)');
```

Matched Points (Inliers Only)



Display both objects.

```
elephantPolygon = [1, 1;...                                 % top-left
         size(elephantImage, 2), 1;...                      % top-right
         size(elephantImage, 2), size(elephantImage, 1);... % bottom-right
         1, size(elephantImage, 1);...                      % bottom-left
         1,1];                       % top-left again to close the polygon

newElephantPolygon = transformPointsForward(tform, elephantPolygon);


figure;
imshow(sceneImage);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
line(newElephantPolygon(:, 1), newElephantPolygon(:, 2), 'Color', 'g');
title('Detected Elephant and Box');
```



Learn More About Image Processing

- *Computer Vision for AUVSI Foundation: Object Detection - Part 1* (download)
- *Computer Vision for AUVSI Foundation: Object Detection - Part 2* (download)
- *Computer Vision for AUVSI Foundation: Object Detection Using Blob Analysis* (download)
- *Computer Vision System Toolbox trial* (product trial)