Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('cheeto_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Image of a Box

Read the target image containing a cluttered scene.

```
sceneImage = imread('cheeto_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```

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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

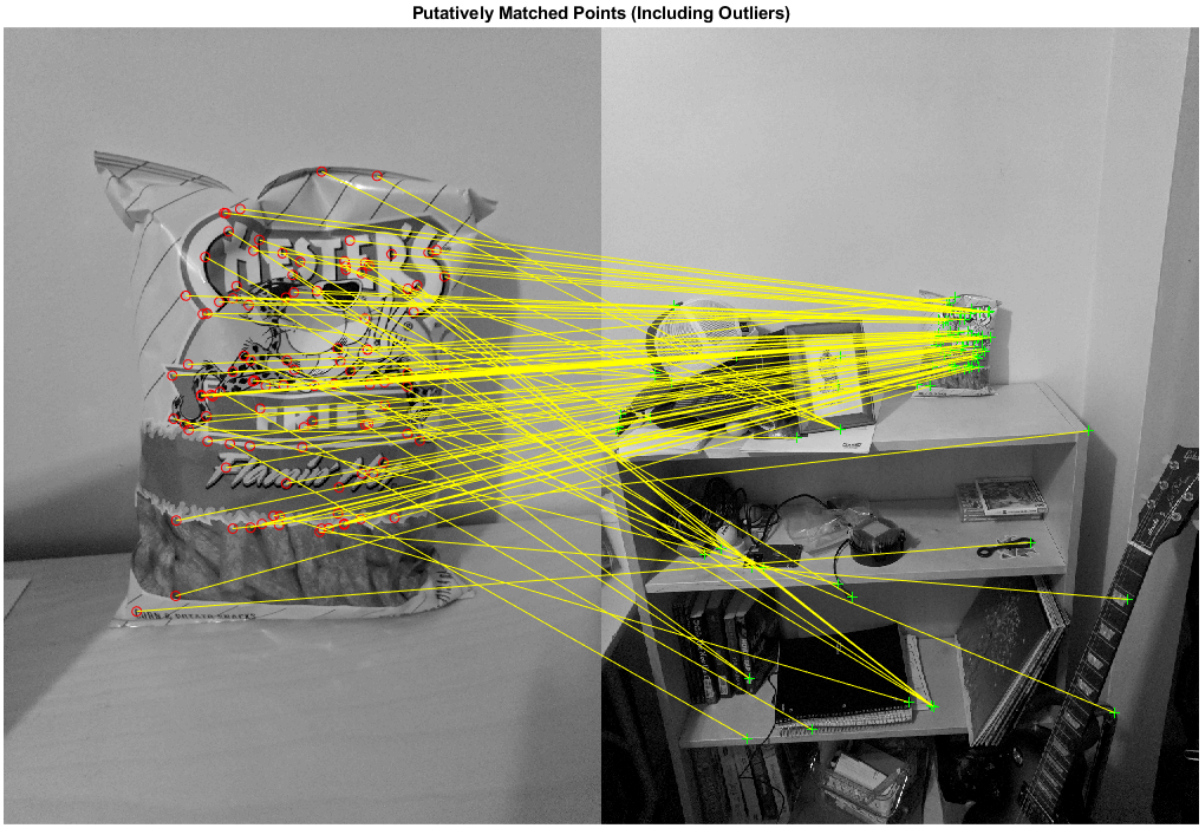
Step 4: Find Putative Point Matches

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)');
```



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

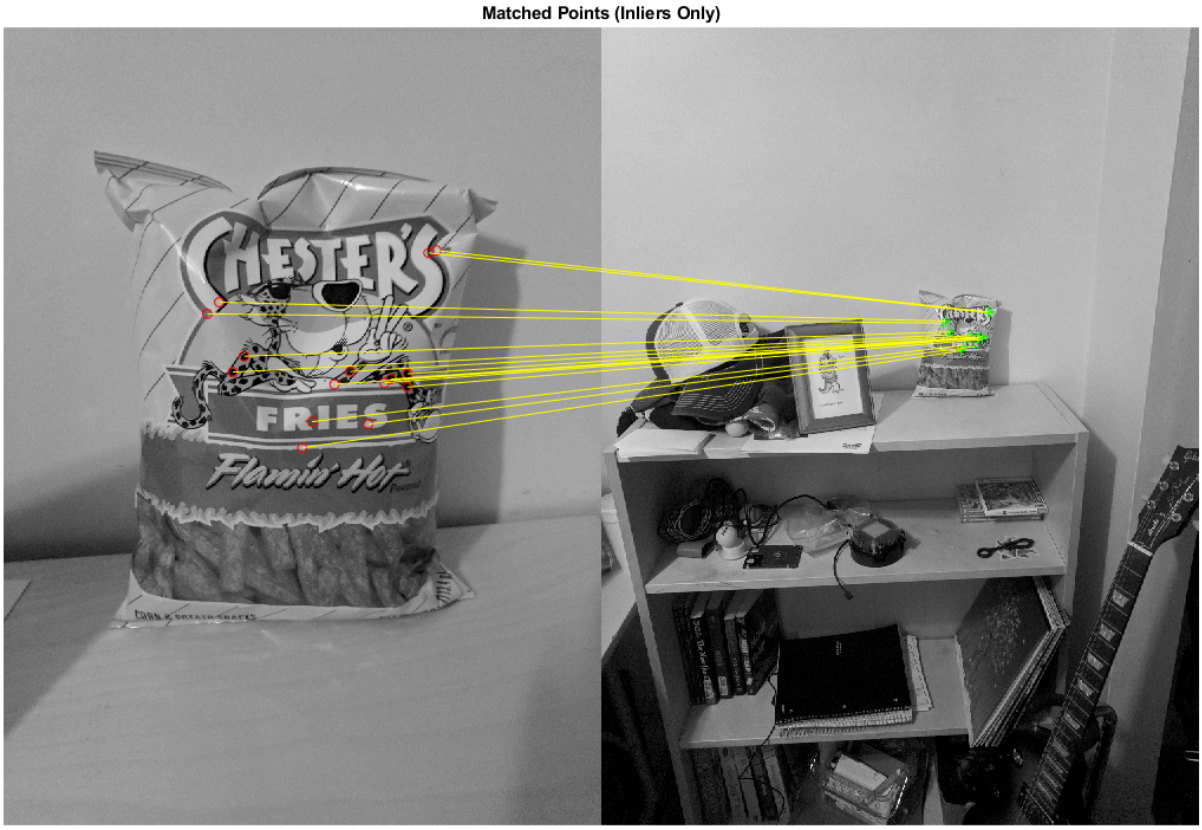
`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

```
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the desired
 confidence.
```

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('coffee_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('coffee_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

Step 2: Detect Feature Points

Detect feature points in both images.

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

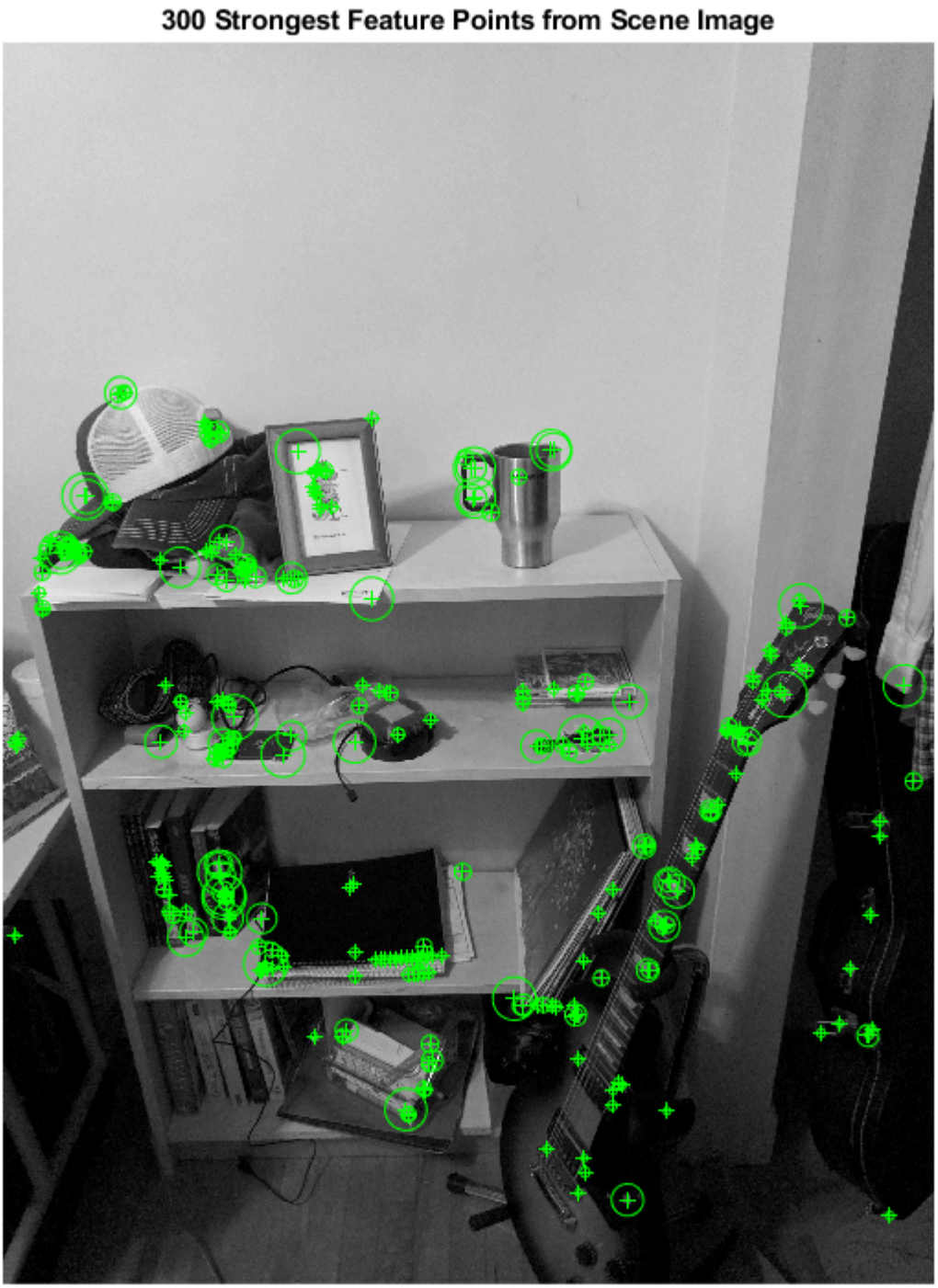
Visualize the strongest feature points found in the reference image.

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

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));
```



Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

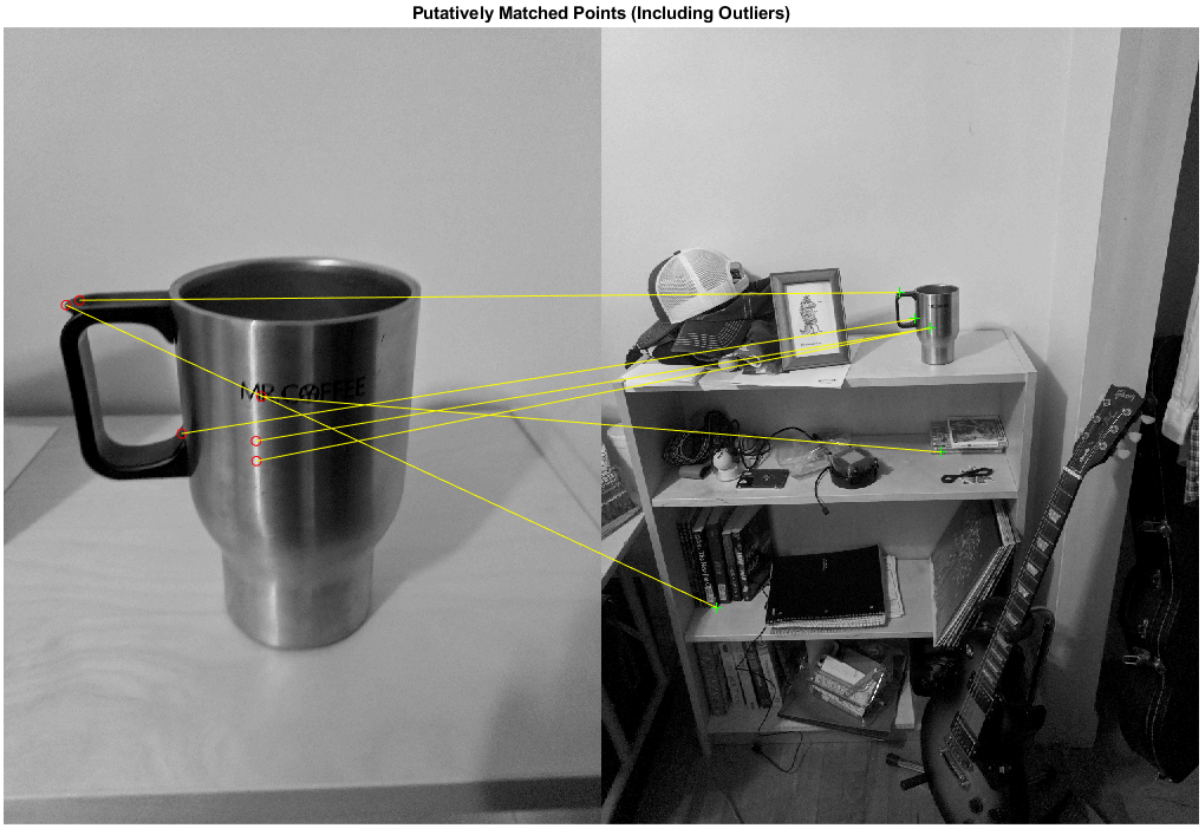
Step 4: Find Putative Point Matches

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)');
```



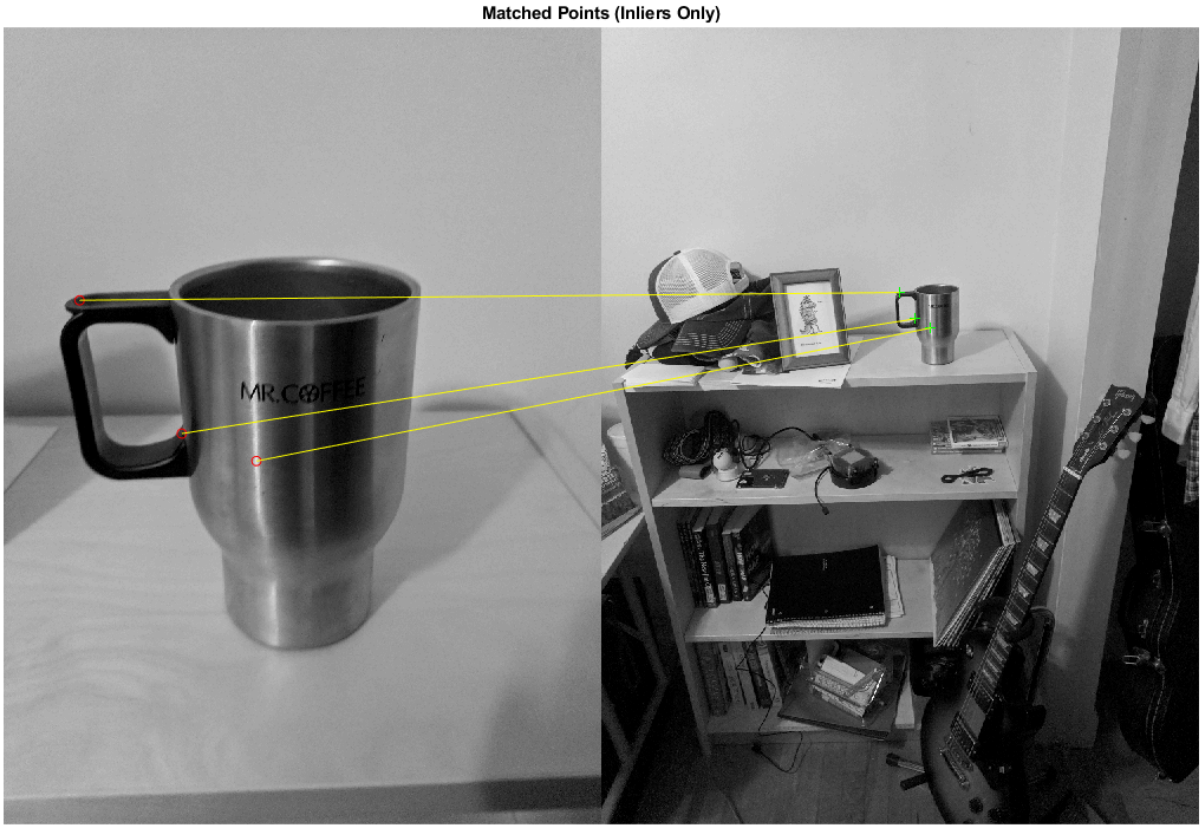
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

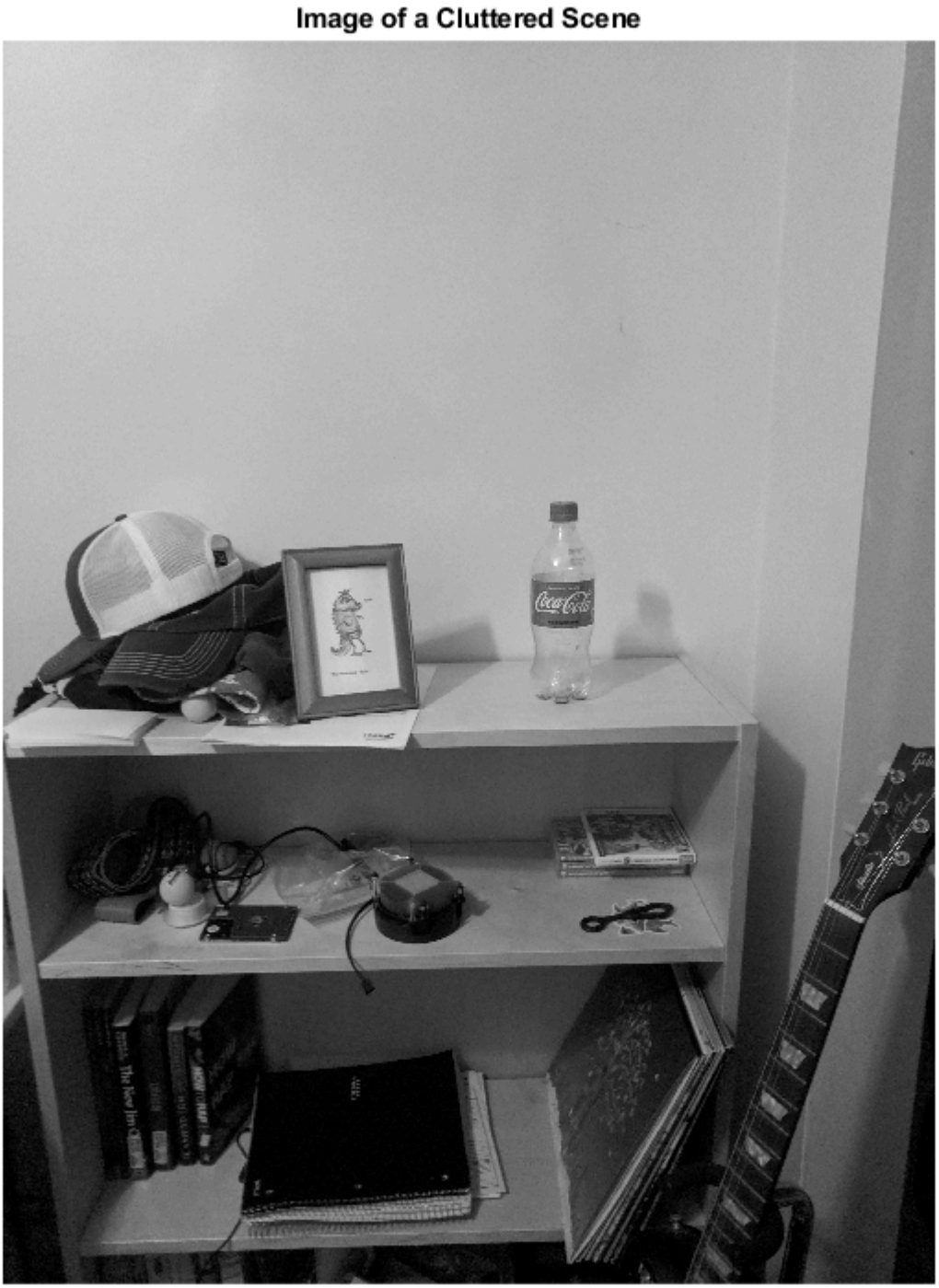
Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('coke_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('coke_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```



Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```

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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

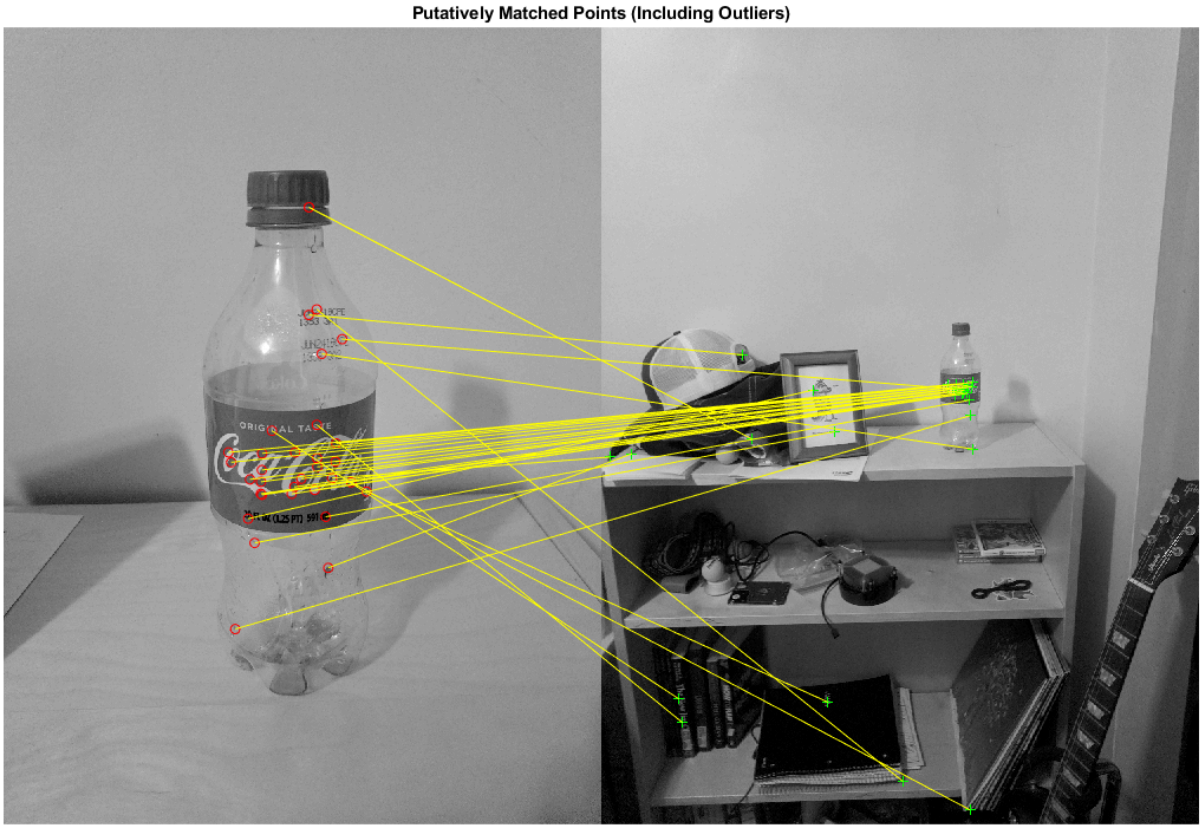
Step 4: Find Putative Point Matches

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)');
```



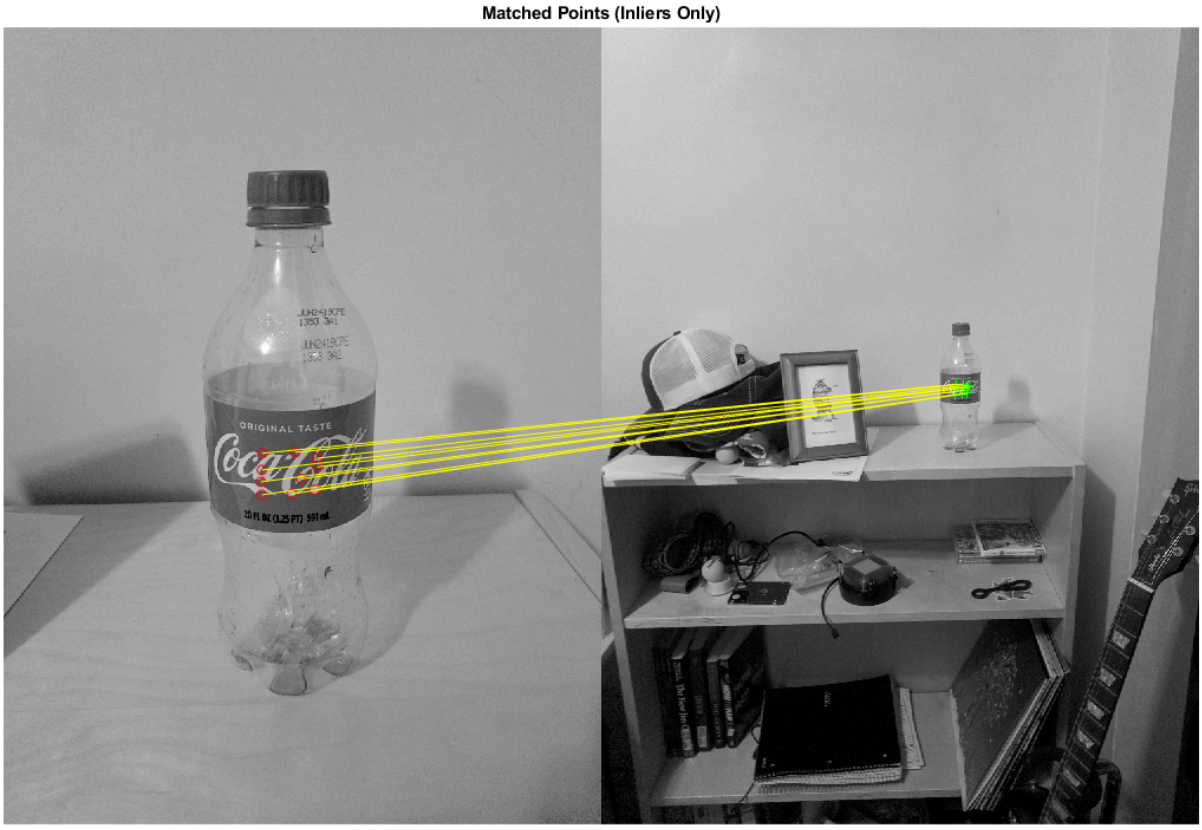
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('falco_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('falco_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

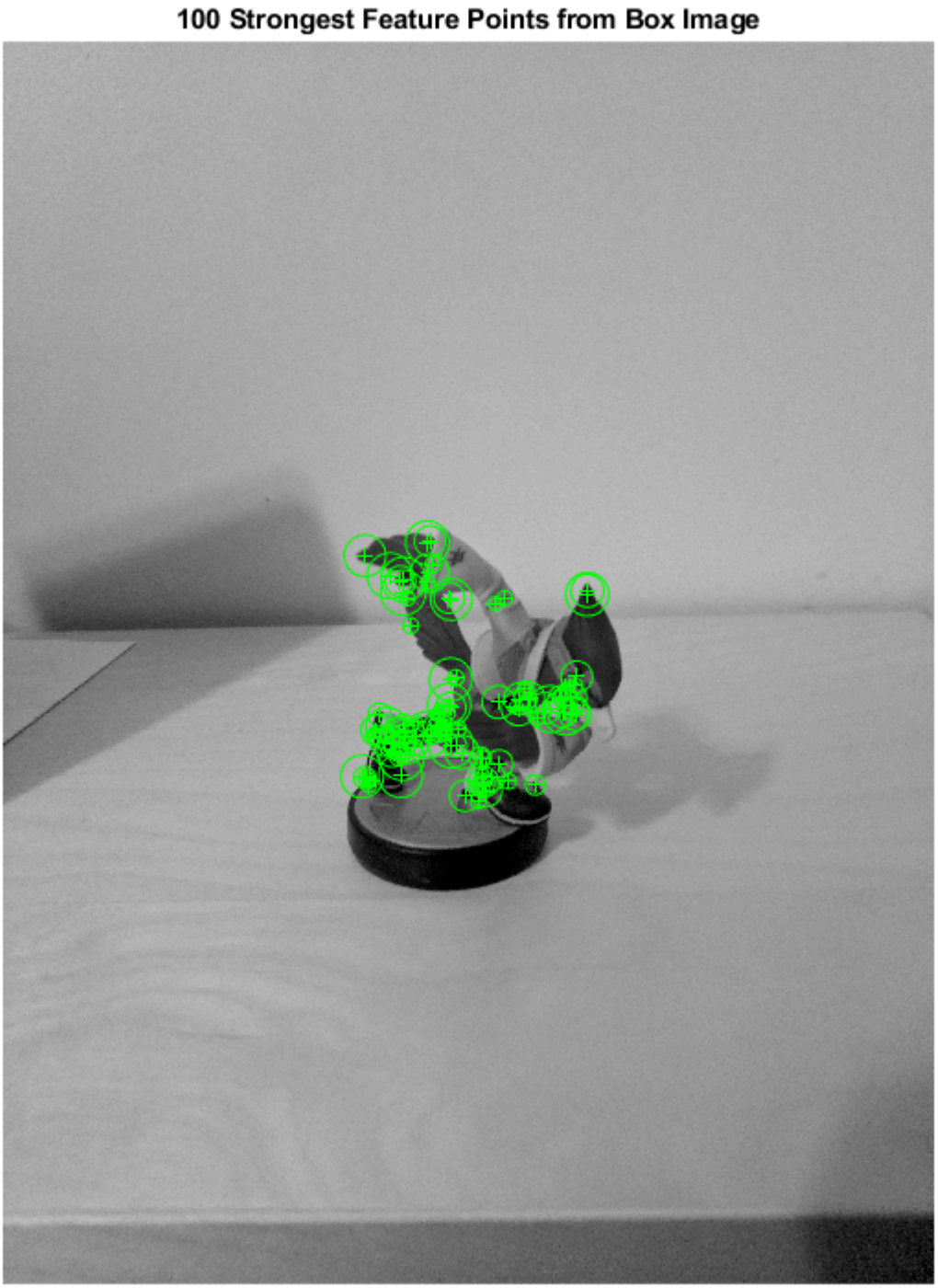
Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```



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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

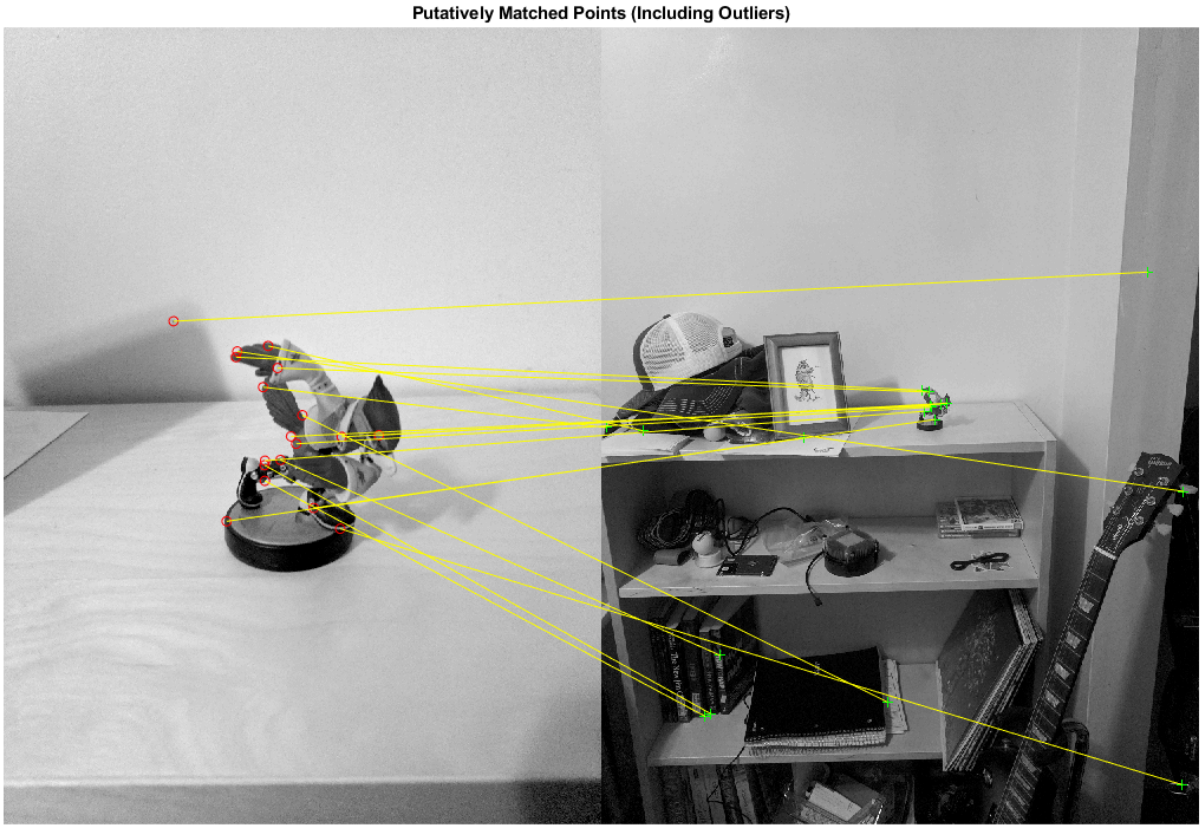
Step 4: Find Putative Point Matches

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)');
```



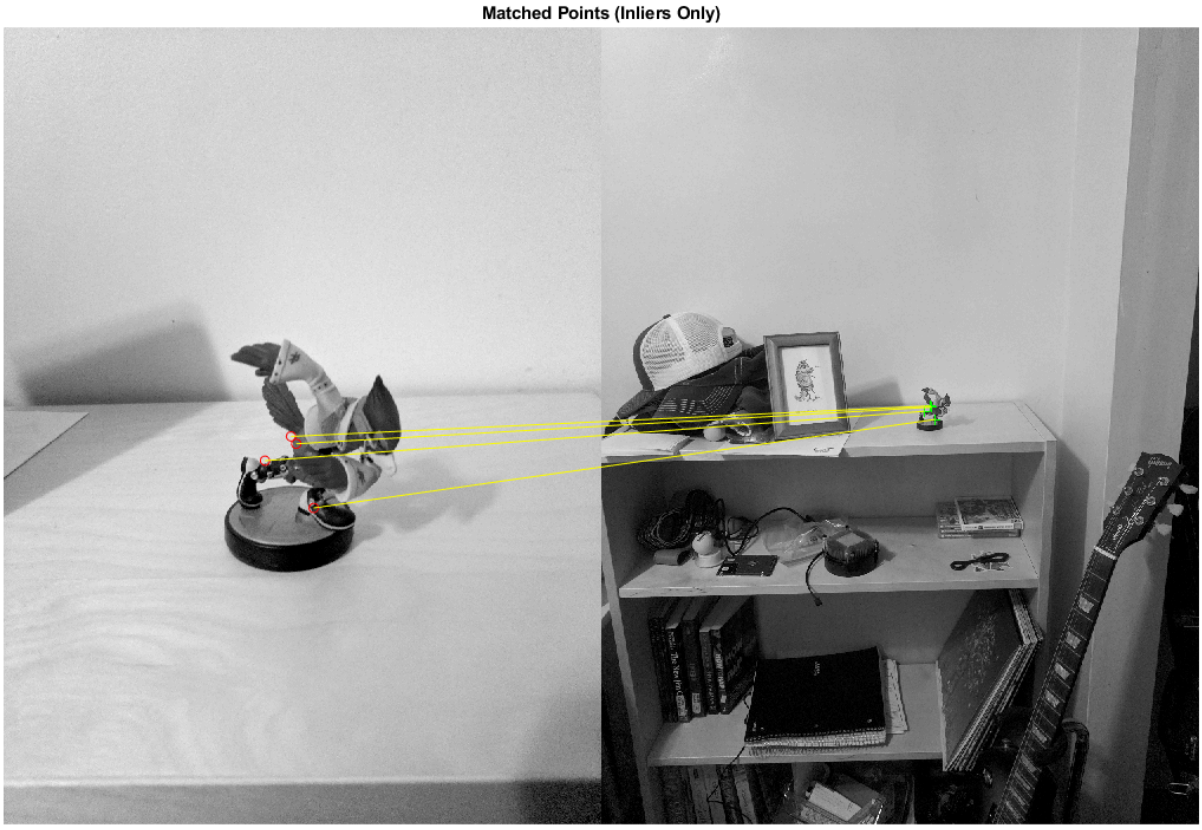
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

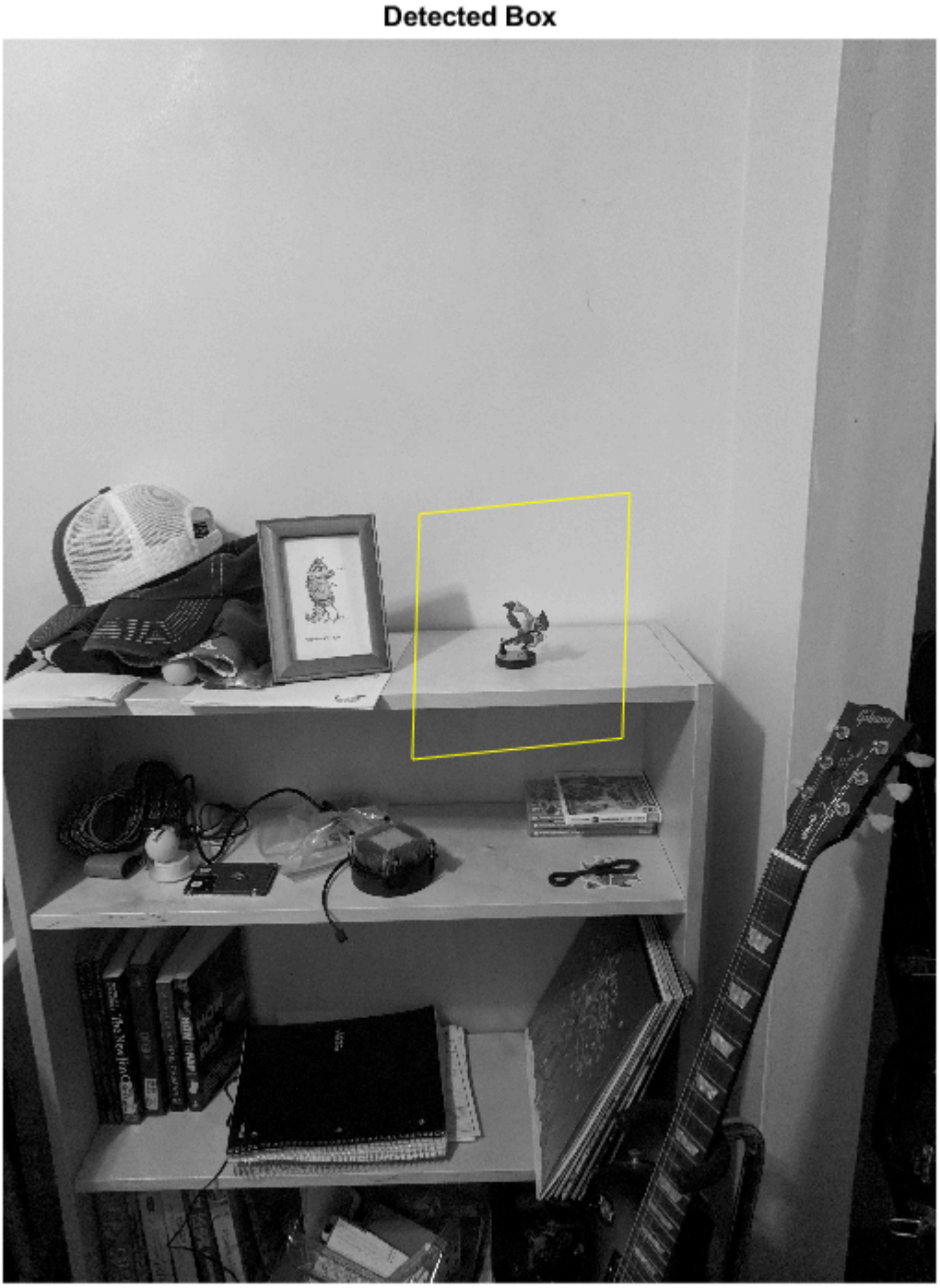
Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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

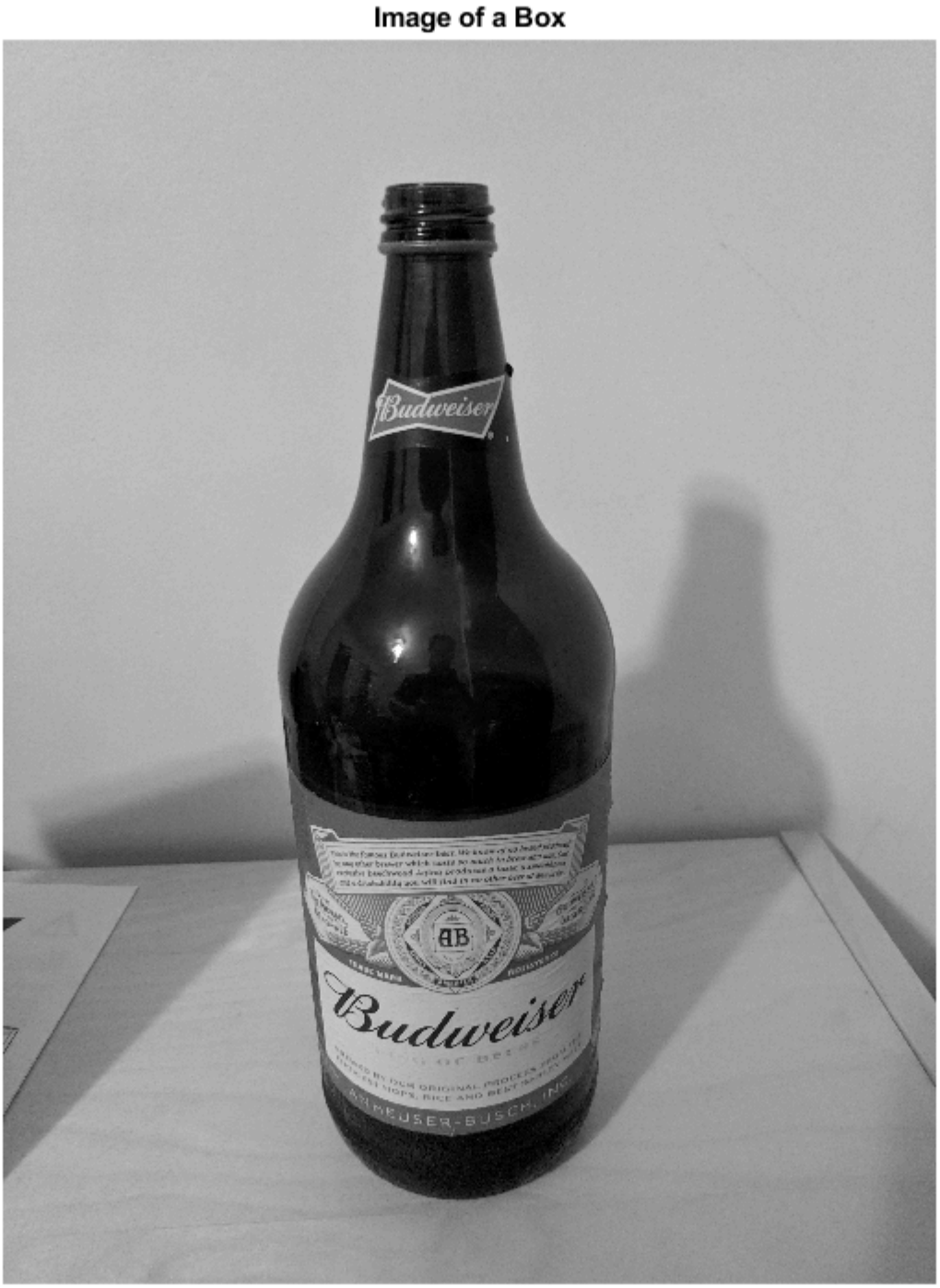


Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('40_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```



Read the target image containing a cluttered scene.

```
sceneImage = imread('40_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```

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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

Step 4: Find Putative Point Matches

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)');
```

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

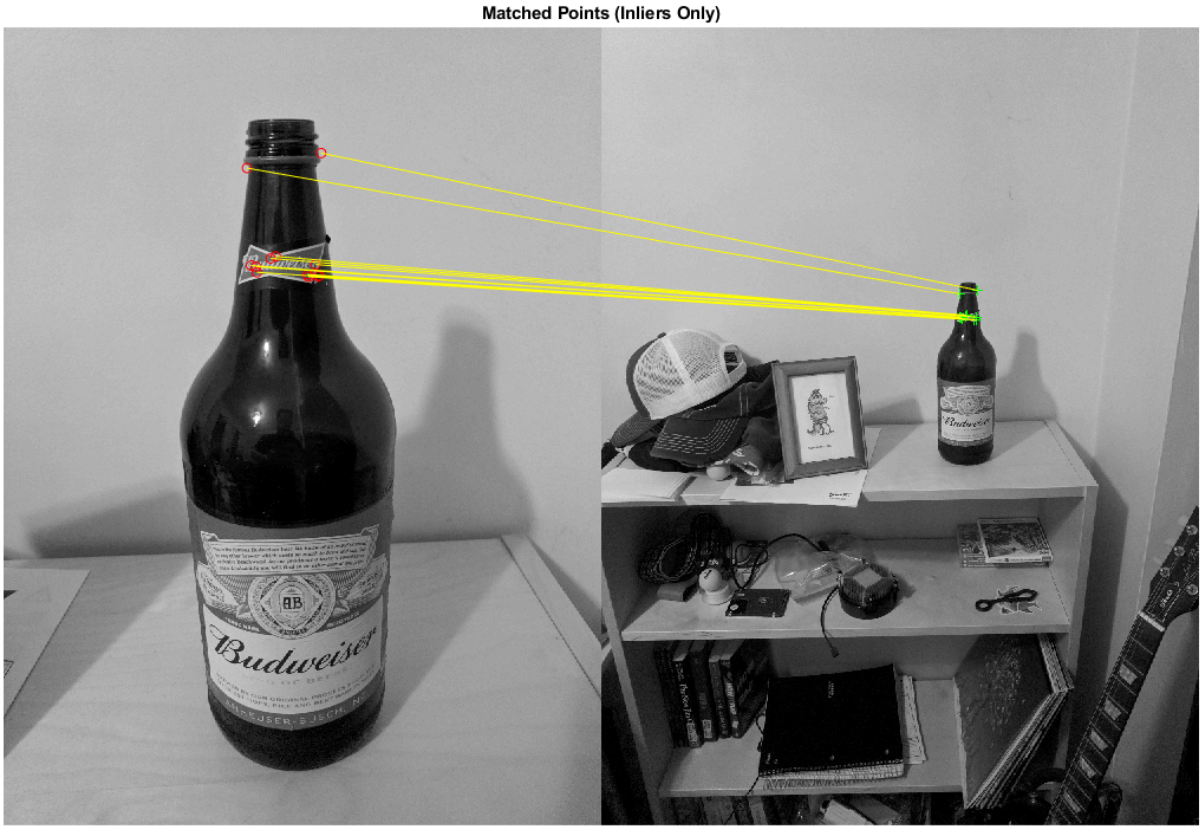
`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

```
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the desired
 confidence.
```

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('jif_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Image of a Box

Read the target image containing a cluttered scene.

```
sceneImage = imread('jif_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```

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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

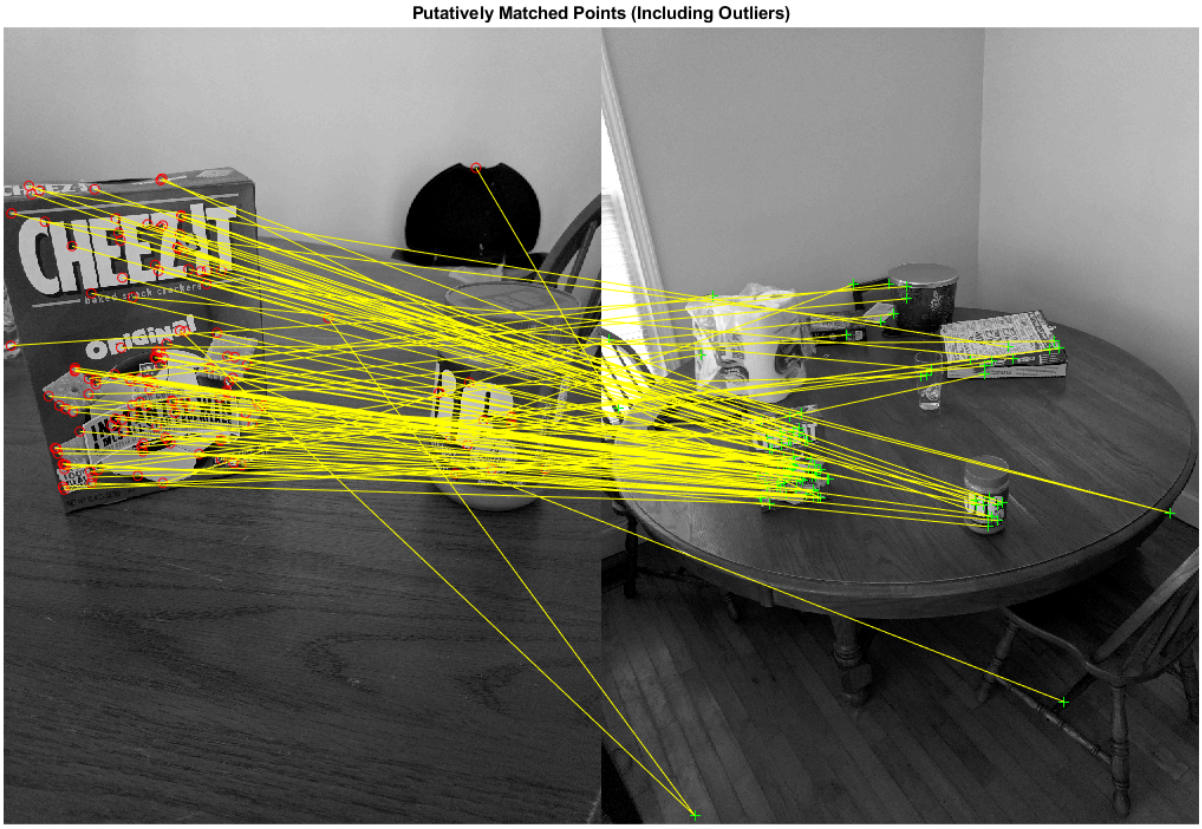
Step 4: Find Putative Point Matches

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)');
```



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

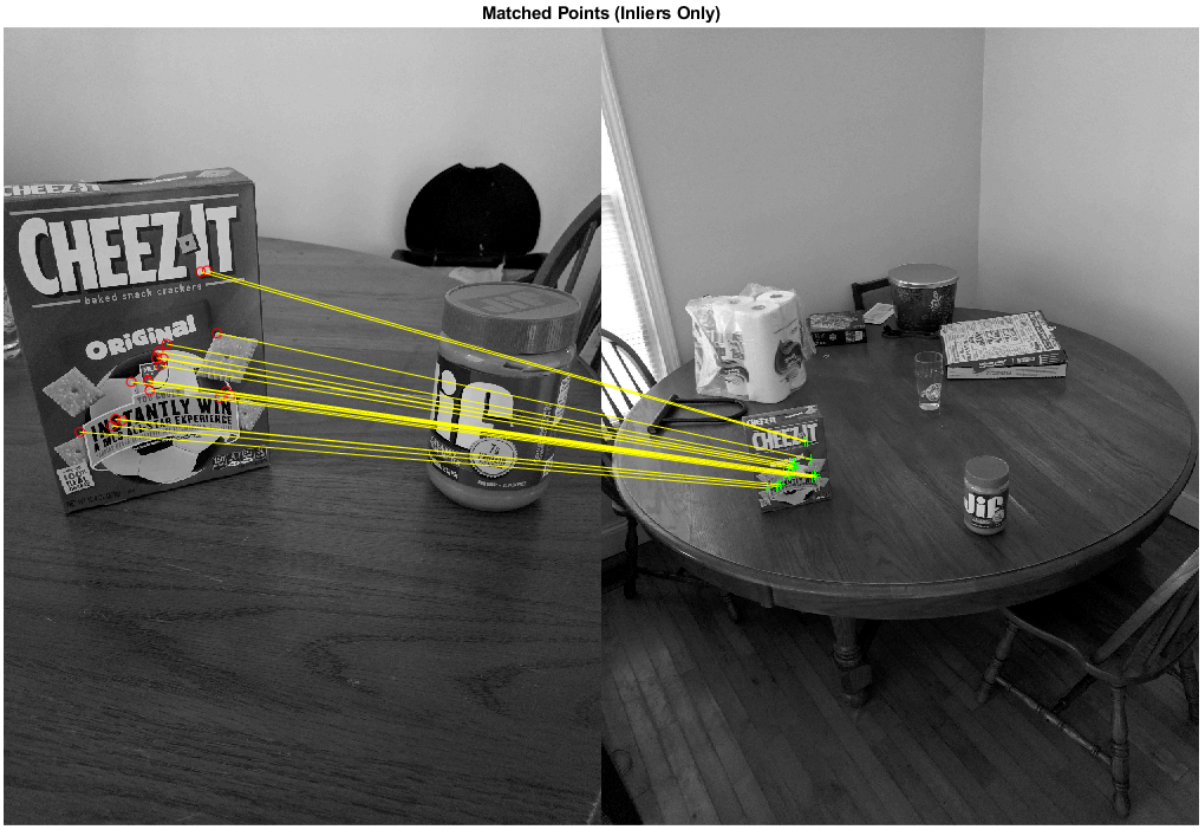
`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

```
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the desired
 confidence.
```

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('kevin_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('kevin_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

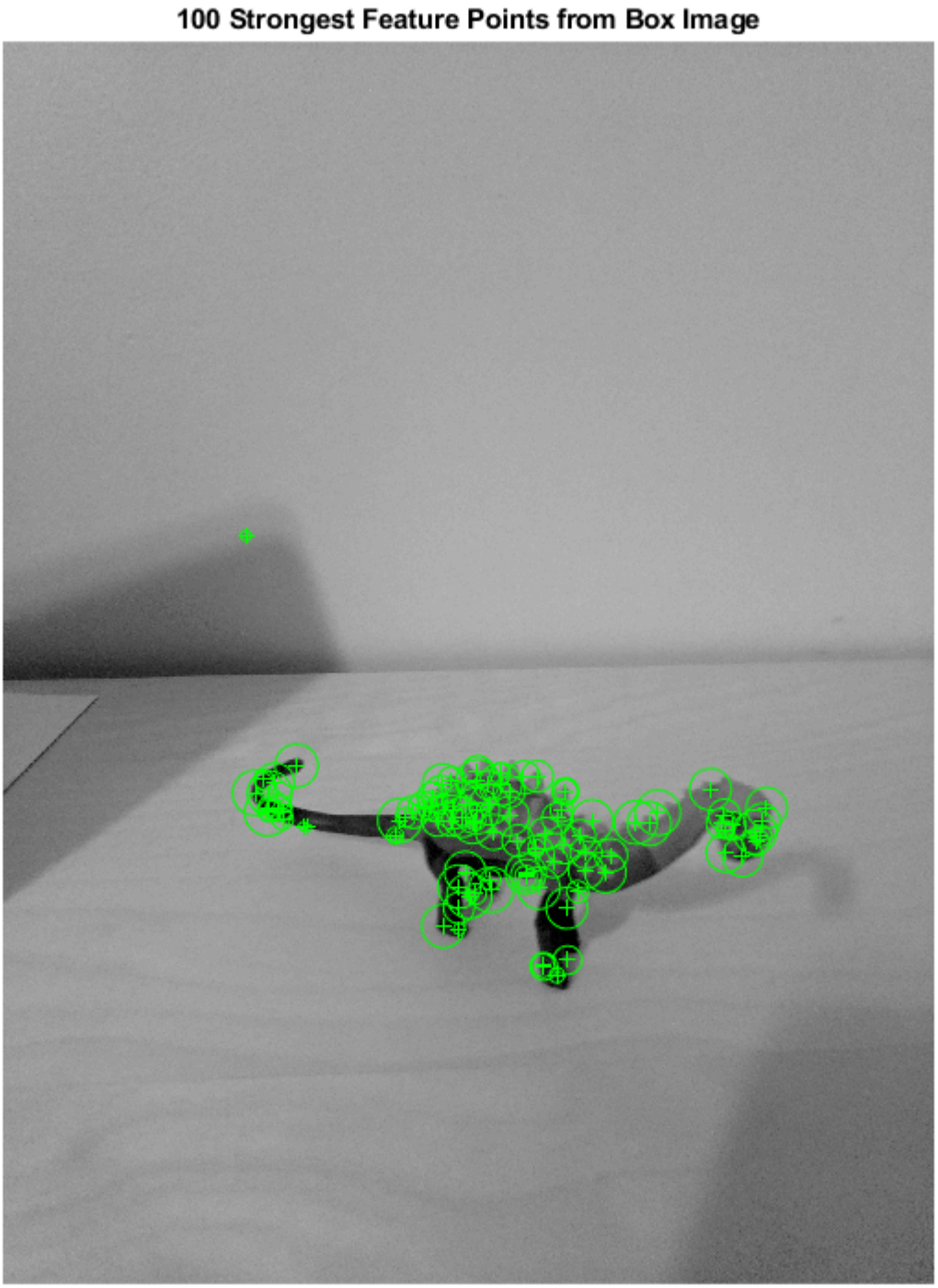
Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```



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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

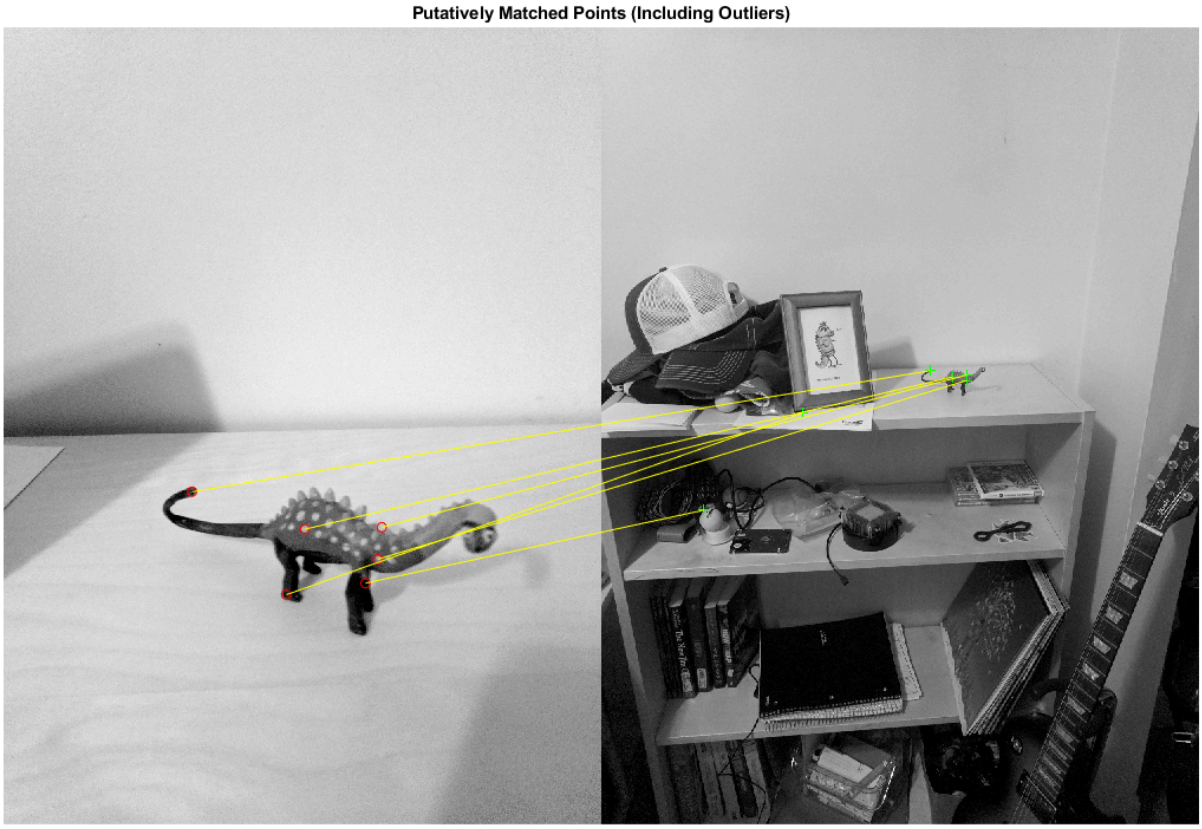
Step 4: Find Putative Point Matches

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)');
```



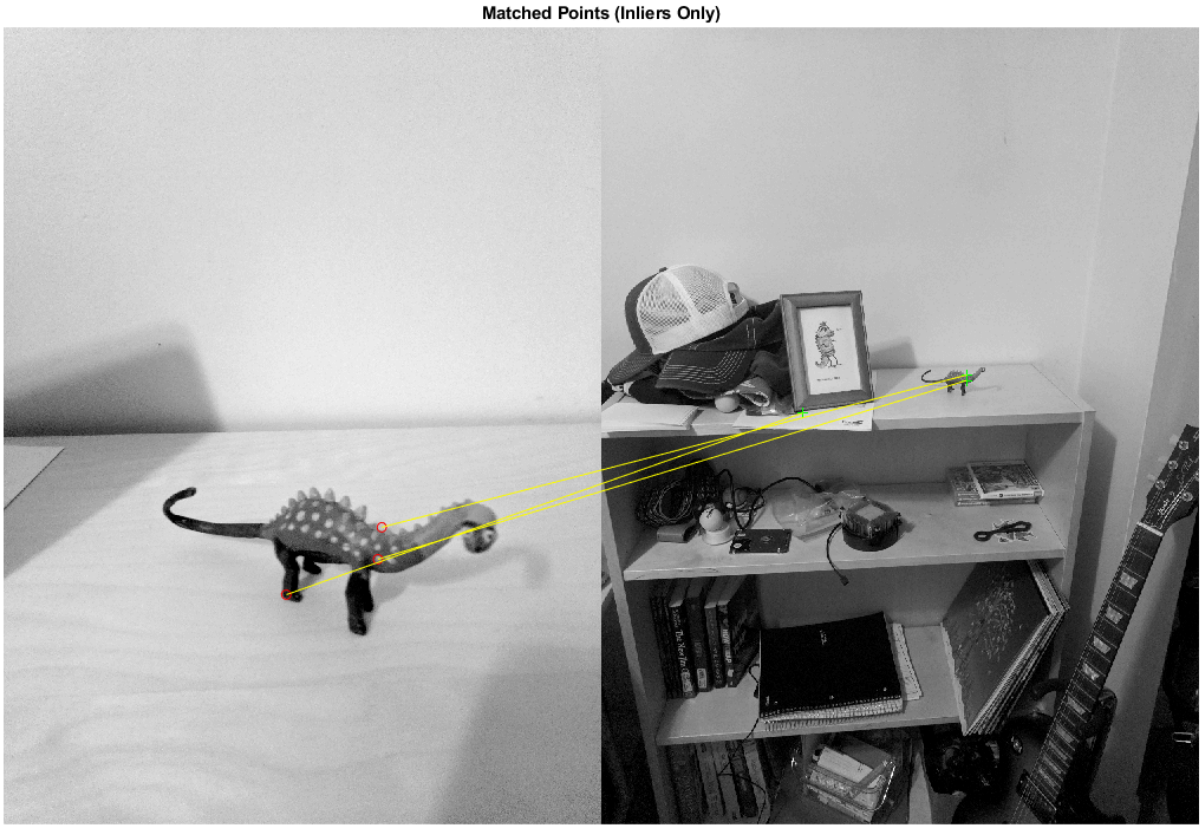
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

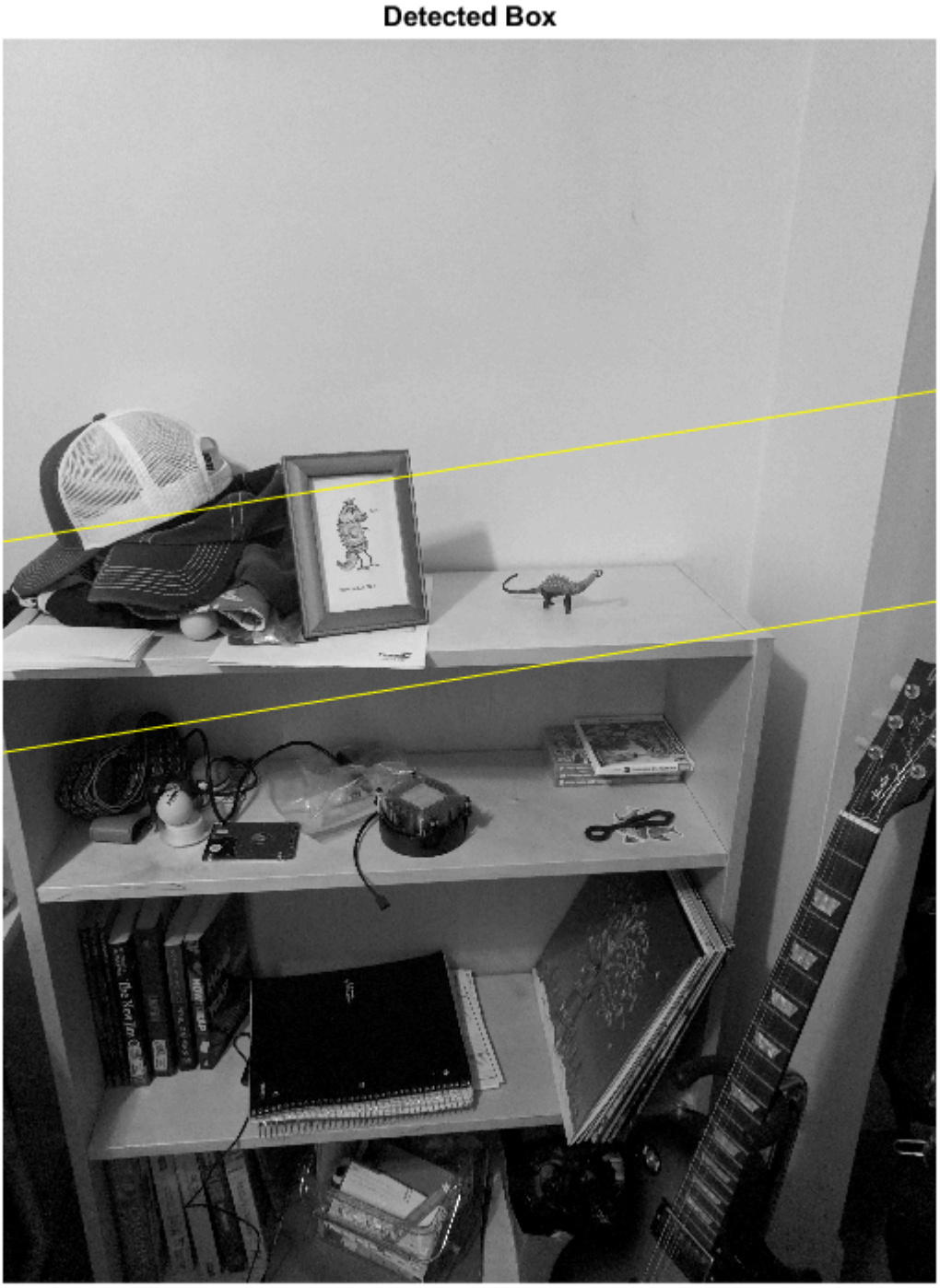
Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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



Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

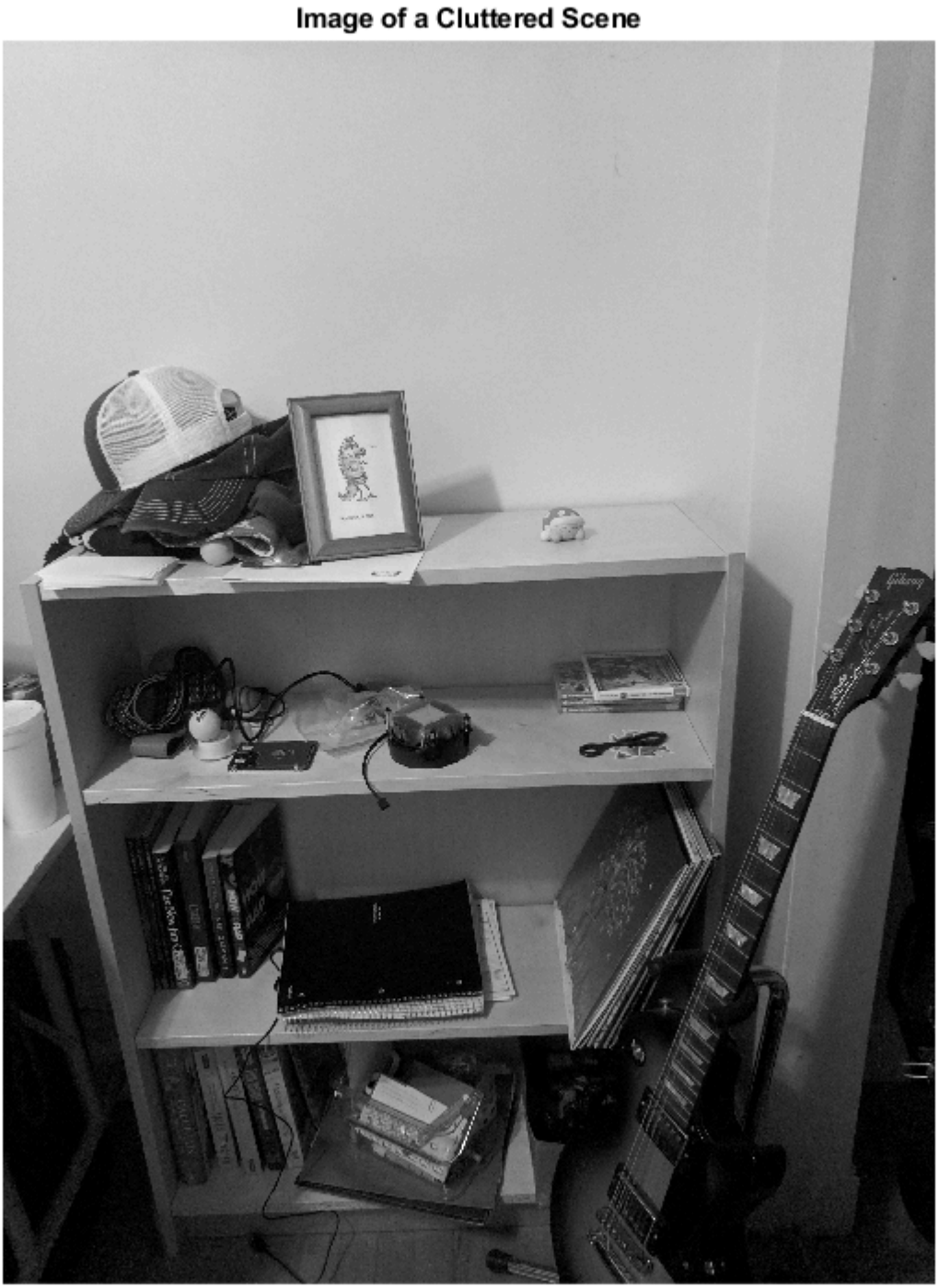
Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('kirby_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('kirby_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```



Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```

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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

Step 4: Find Putative Point Matches

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)');
```

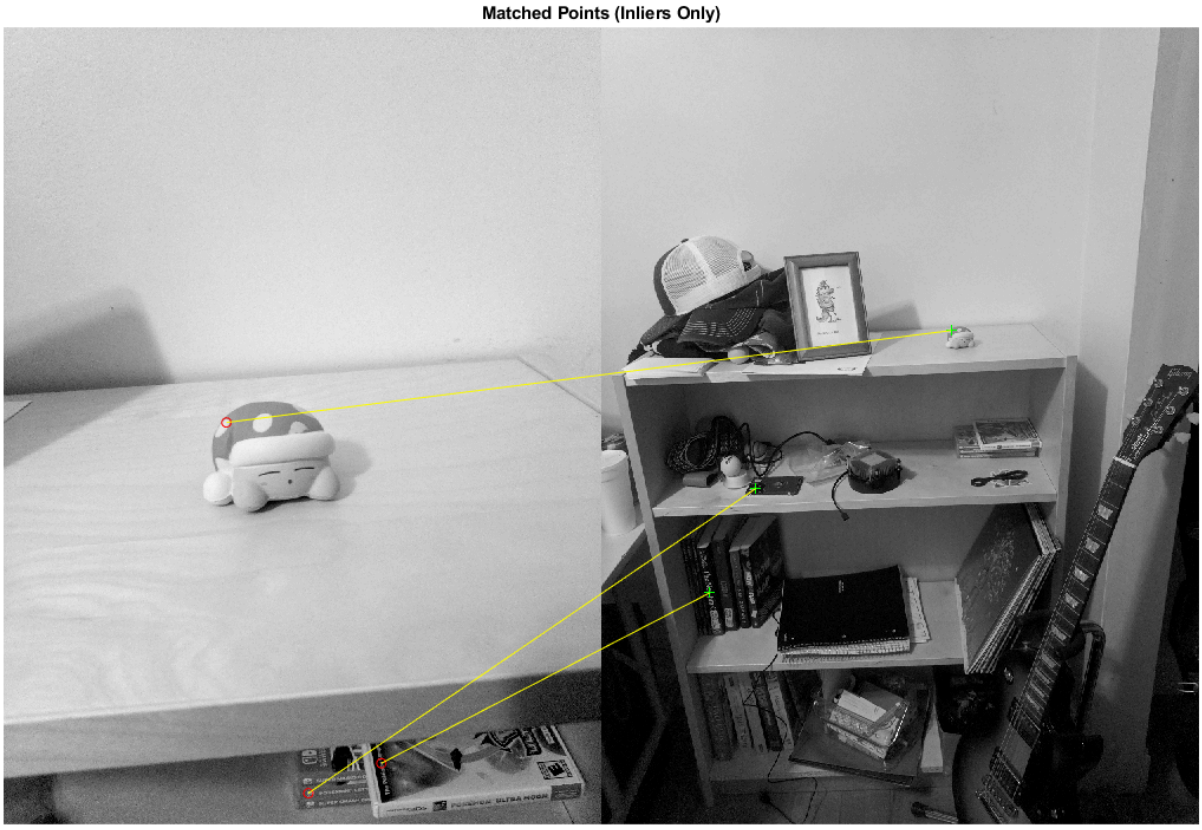
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

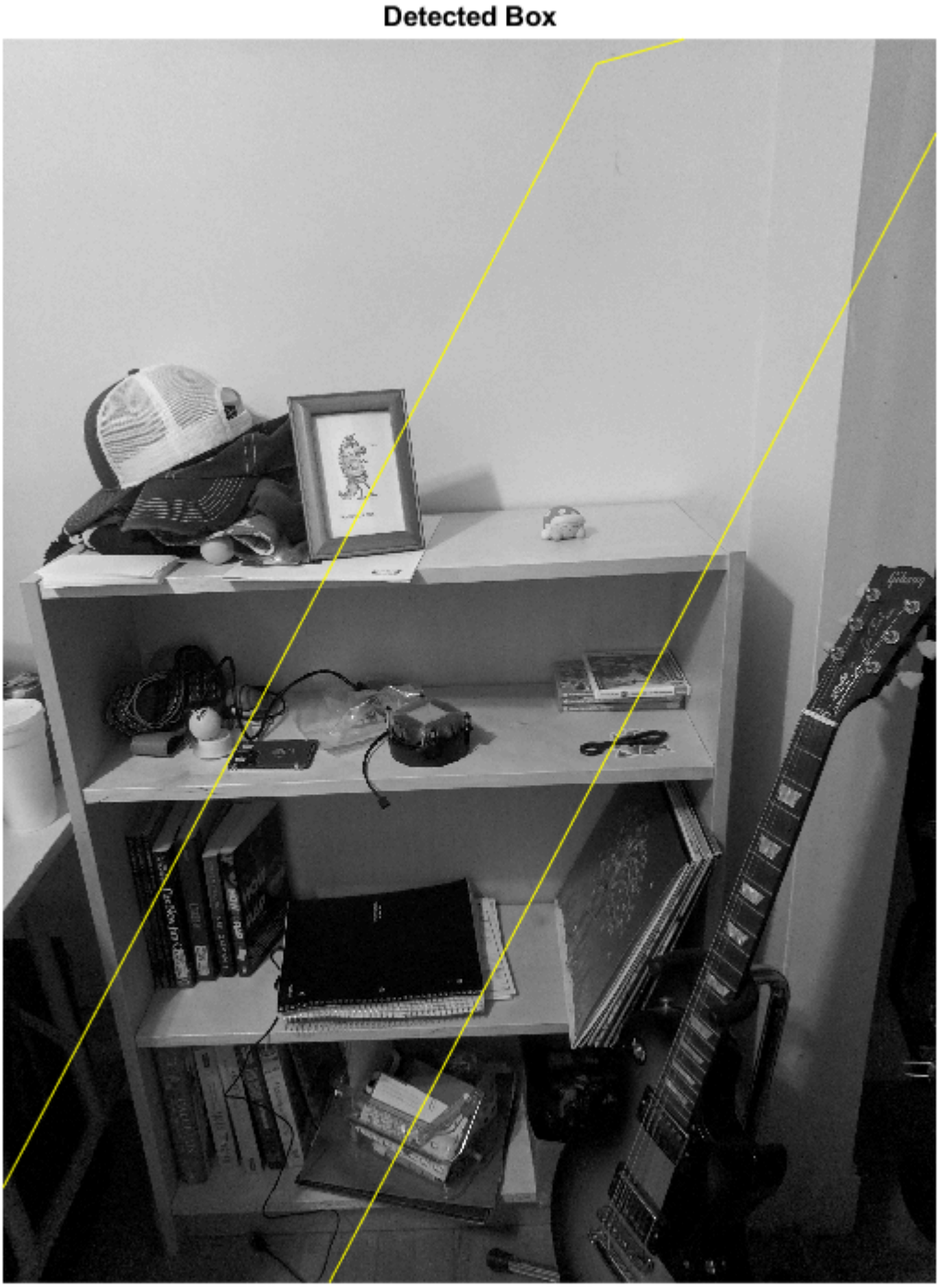
Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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



Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('pickle_1.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('pickle_2.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

Step 2: Detect Feature Points

Detect feature points in both images.

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

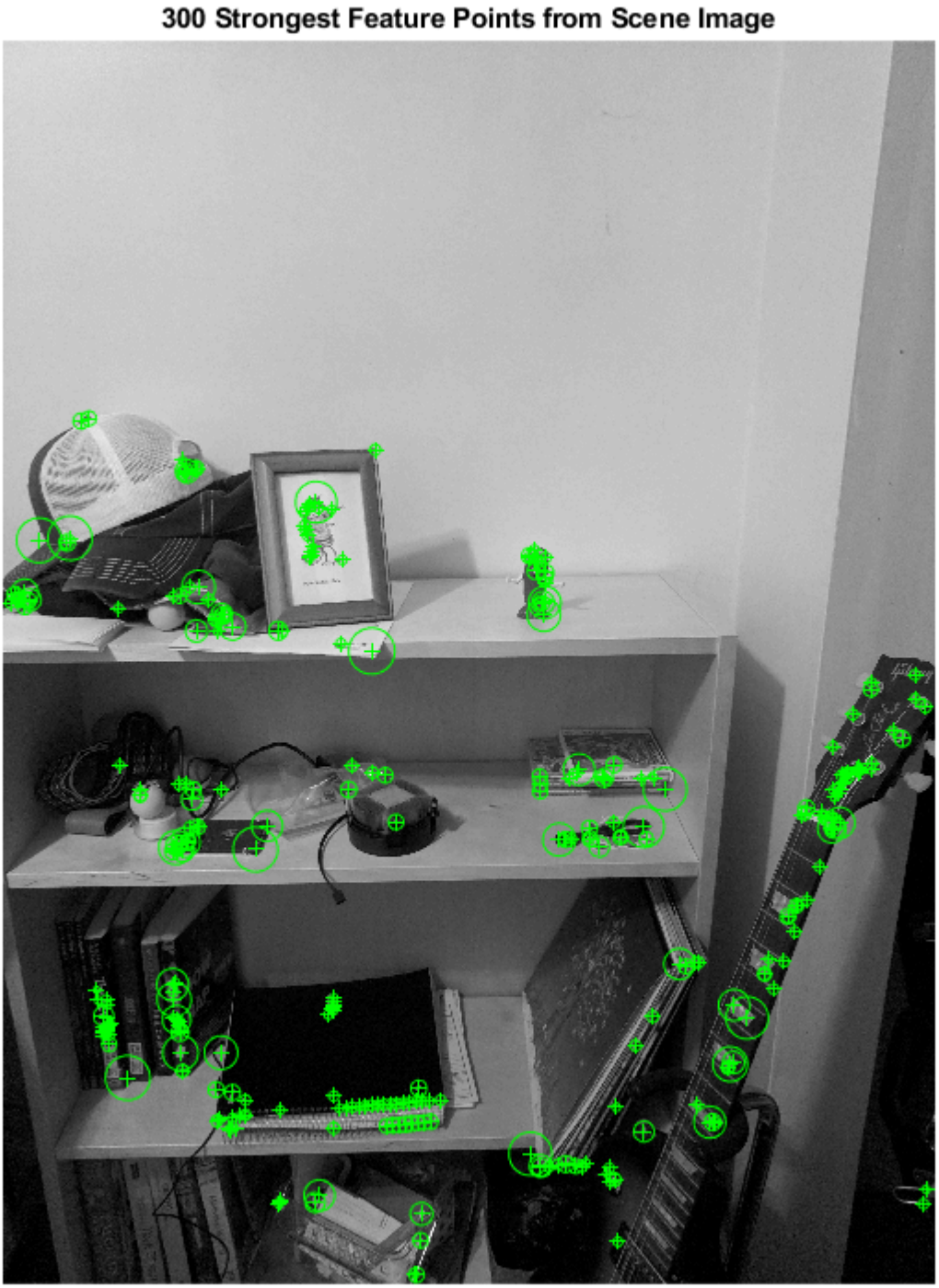
Visualize the strongest feature points found in the reference image.

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

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));
```



Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

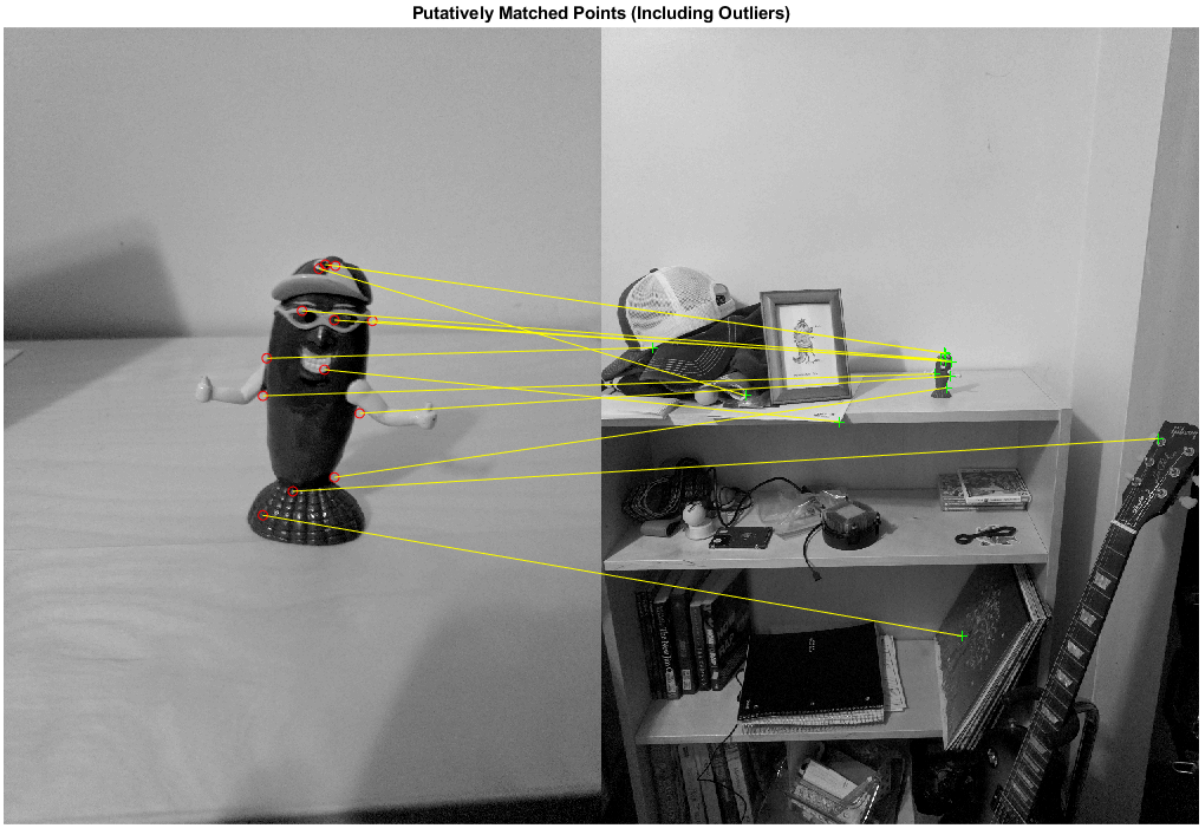
Step 4: Find Putative Point Matches

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)');
```



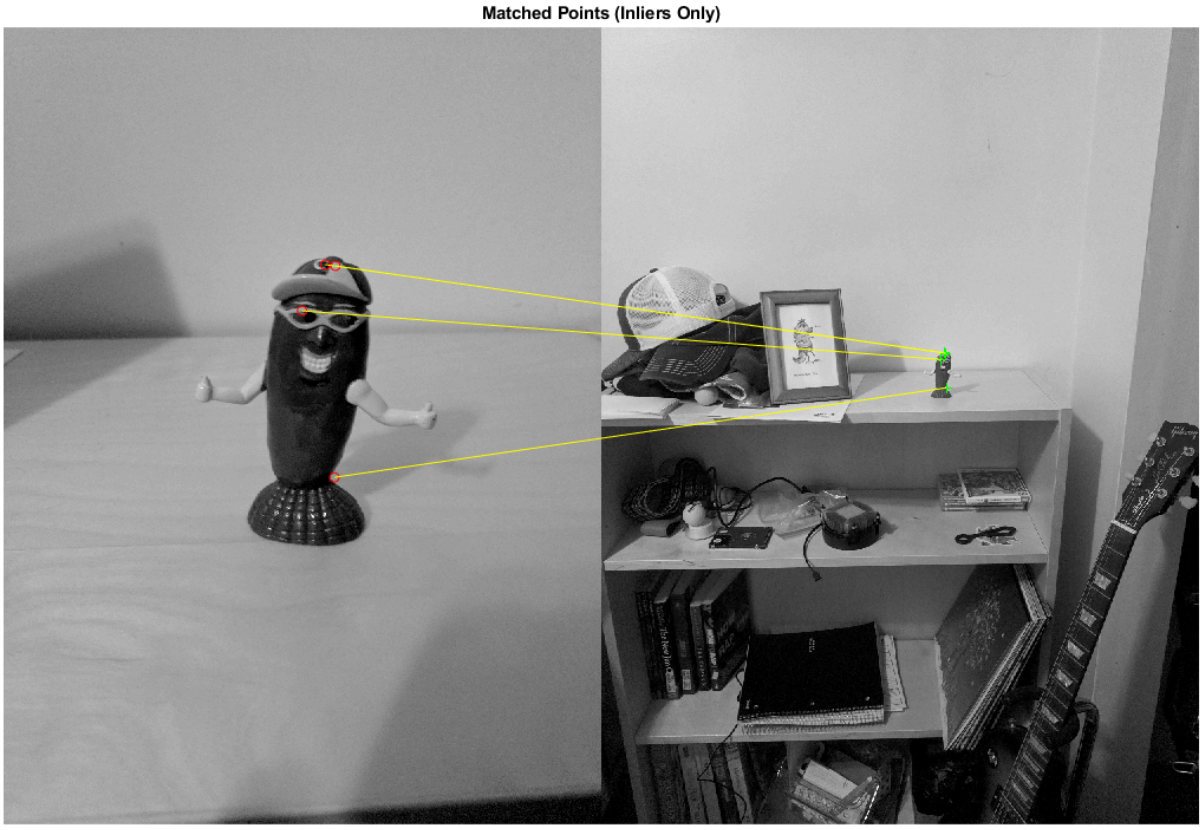
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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

Object Detection in a Cluttered Scene Using Point Feature Matching

Step 1: Read Images

Read the reference image containing the object of interest.

```
addpath 'D:'\homework\'comp vision'\'image pairs'\
boxImage = imread('megaman3.jpg');
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a Box');
```

Read the target image containing a cluttered scene.

```
sceneImage = imread('megaman1.jpg');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
```

```
title('Image of a Cluttered Scene');
```

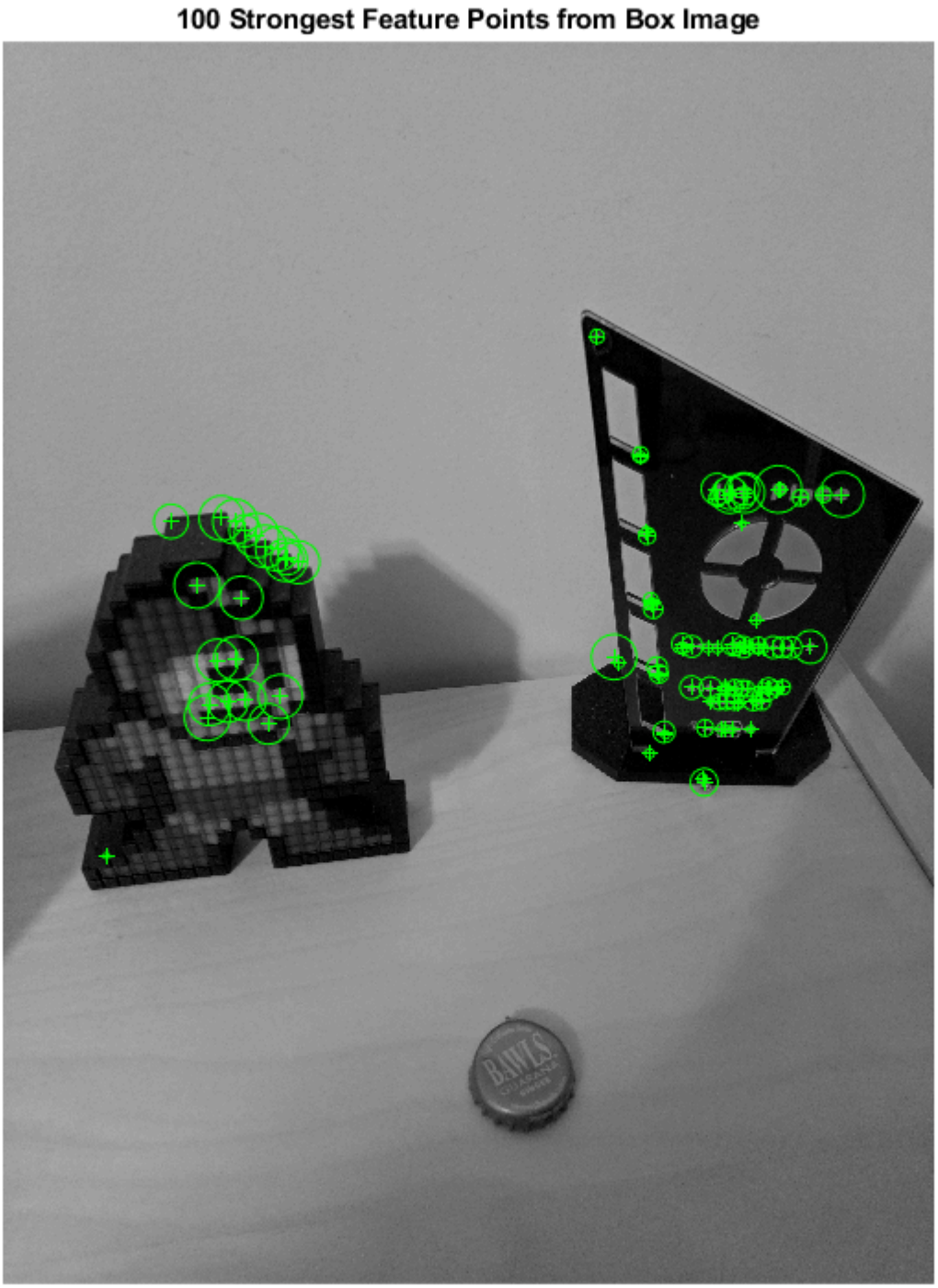
Step 2: Detect Feature Points

Detect feature points in both images.

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

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));
```



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));
```

Step 3: Extract Feature Descriptors

Extract feature descriptors at the interest points in both images.

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

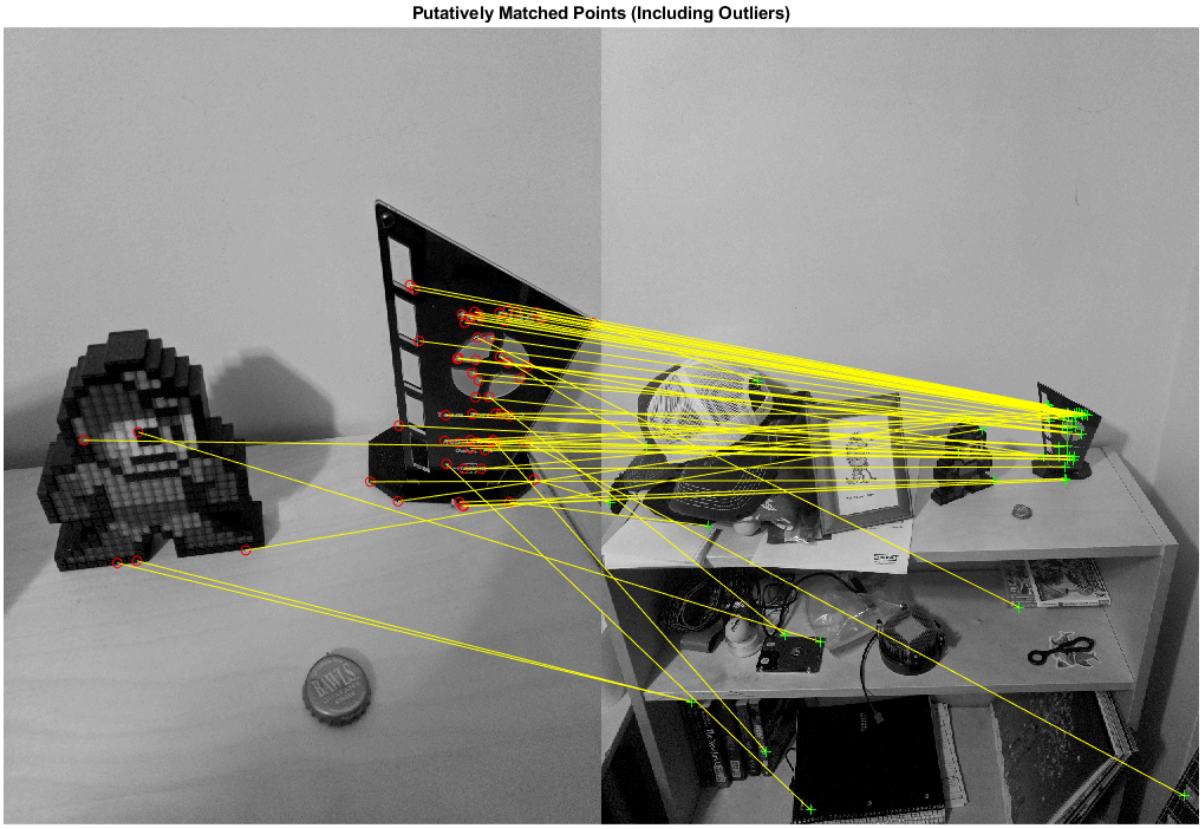
Step 4: Find Putative Point Matches

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)');
```



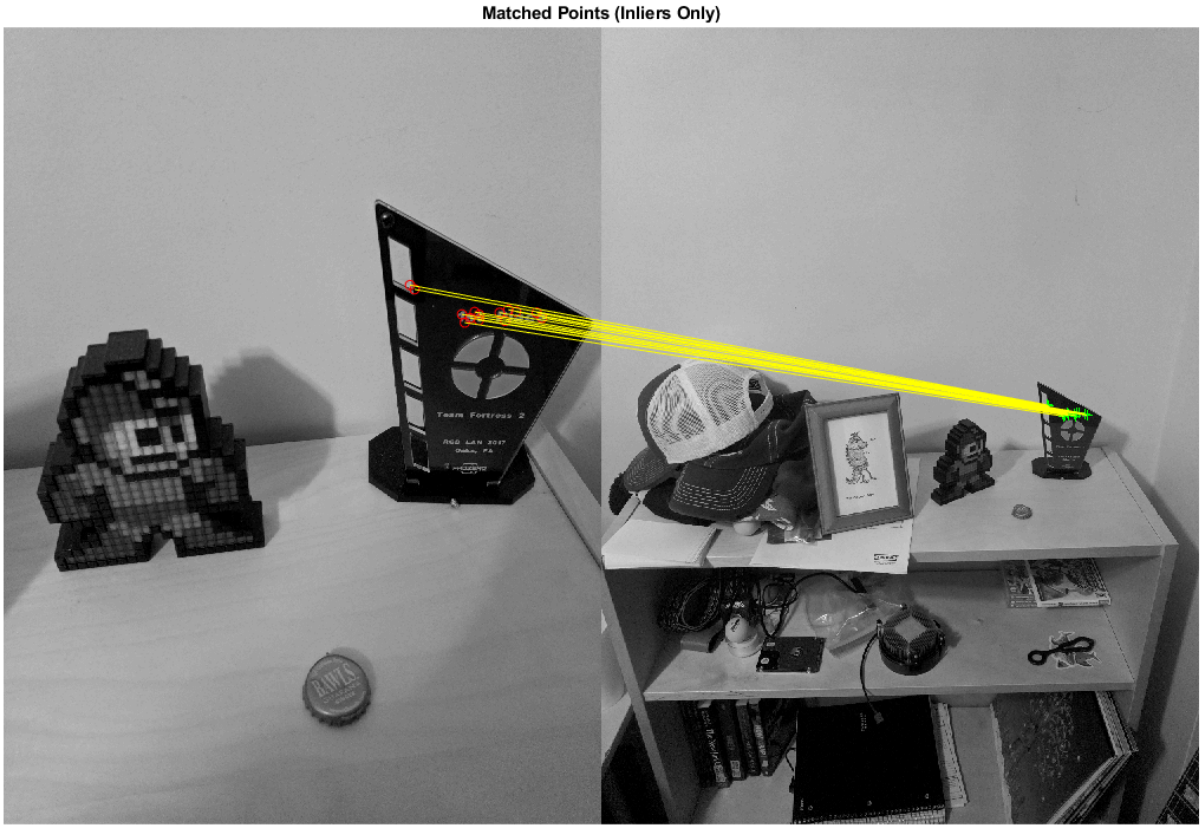
Step 5: Locate the Object in the Scene Using Putative Matches

`estimateGeometricTransform` calculates the transformation relating the matched points, while eliminating outliers. This transformation allows us to localize the object in the scene.

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

Display the matching point pairs with the outliers removed

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



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
```

Transform the polygon into the coordinate system of the target image. The transformed polygon indicates the location of the object in the scene.

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

Display the detected object.

```
figure;
```

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