Table of Contents


```
clc;
clear;
close all;
```

Initial pre-processing

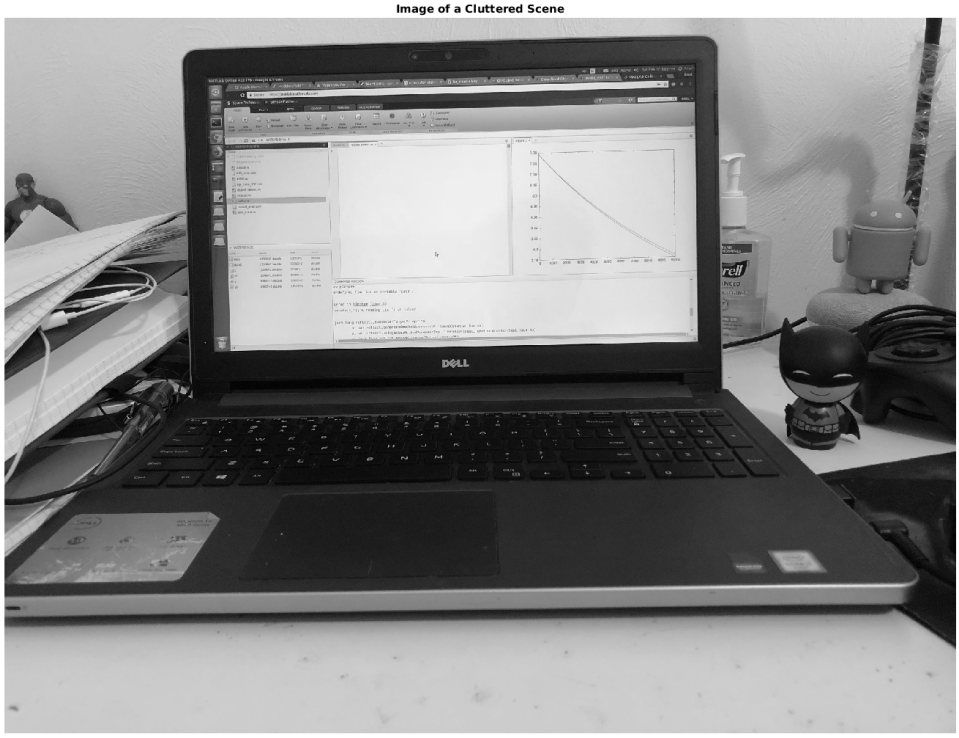
Read the object image

```
boxImage = imread('object.JPG');
boxImage = imrotate(boxImage,180);
boxImage = rgb2gray(boxImage);
figure;
imshow(boxImage);
title('Image of a object');

% Read the scene image
sceneImage = imread('desk.JPG');
sceneImage = rgb2gray(sceneImage);
figure;
imshow(sceneImage);
title('Image of a Cluttered Scene');
```

```
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
```

Image of a object

Image of a Cluttered Scene



Implementation of SURF based feature detection and object detection

Detect the featrure points using SURF feature detection

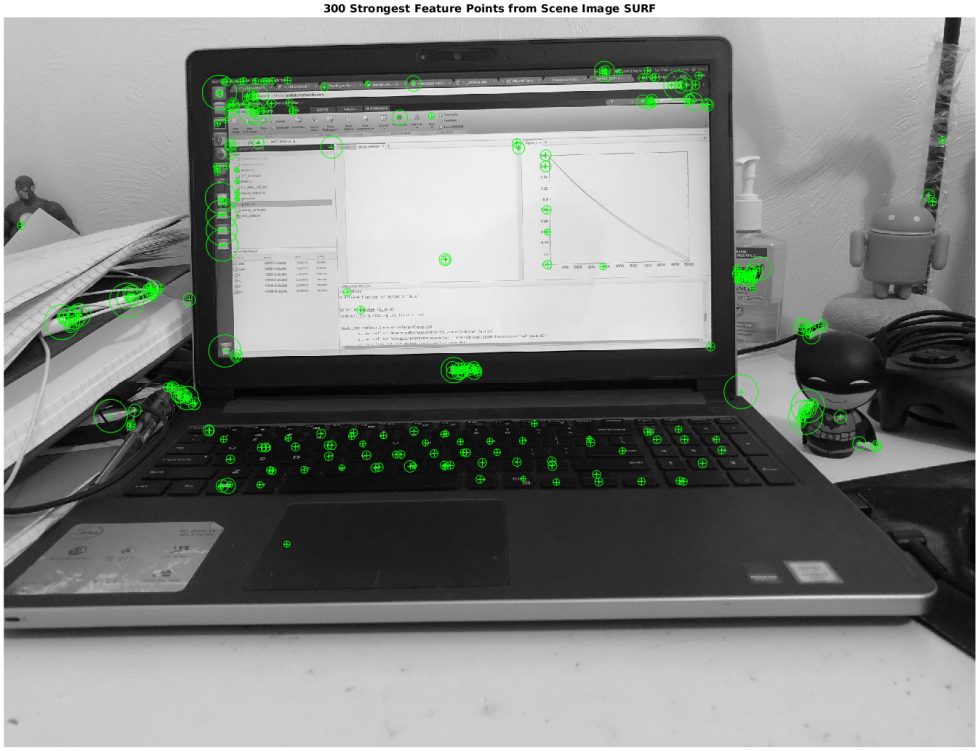
```
boxPoints = detectSURFFeatures(boxImage);
scenePoints = detectSURFFeatures(sceneImage);
figure;
imshow(sceneImage);


% Visualizing the strongest points in the object image
figure;
imshow(boxImage);
title('300 Strongest SURF Feature Points from Box Image');
hold on;
plot(selectStrongest(boxPoints, 300));


% Visualizing the strongest points in the target scene image
figure;
imshow(sceneImage);
title('300 Strongest Feature Points from Scene Image SURF');
hold on;
plot(selectStrongest(scenePoints, 300));

% extracting feature descriptors at interest points in both images
[boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
[sceneFeatures, scenePoints] = extractFeatures(sceneImage,
 scenePoints);

% Matching the features using the descriptors
boxPairs = matchFeatures(boxFeatures, sceneFeatures);

% Display the matched features
matchedBoxPoints = boxPoints(boxPairs(:, 1), :);
matchedScenePoints = scenePoints(boxPairs(:, 2), :);
figure;
showMatchedFeatures(boxImage, sceneImage, matchedBoxPoints, ...
    matchedScenePoints, 'montage');
title('Putatively Matched Points (Including Outliers) SURF');
hold on

% Locating the object in the scene
[tform, inlierBoxPoints, inlierScenePoints] = ...
    estimateGeometricTransform(matchedBoxPoints,
 matchedScenePoints, 'affine');


% Displaying the matched points after removing the outliers
figure;
showMatchedFeatures(boxImage, sceneImage, inlierBoxPoints, ...
```

```
    inlierScenePoints, 'montage');
title('Matched Points (Inliers Only) SURF');


%Drawing bounding box on 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


%Transforming polygon i9n the coordinate system of the transformed
 image
newBoxPolygon = transformPointsForward(tform, boxPolygon);


% Displaying the detectd object
figure;
imshow(sceneImage);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
title('Detected Box using SURF');
```

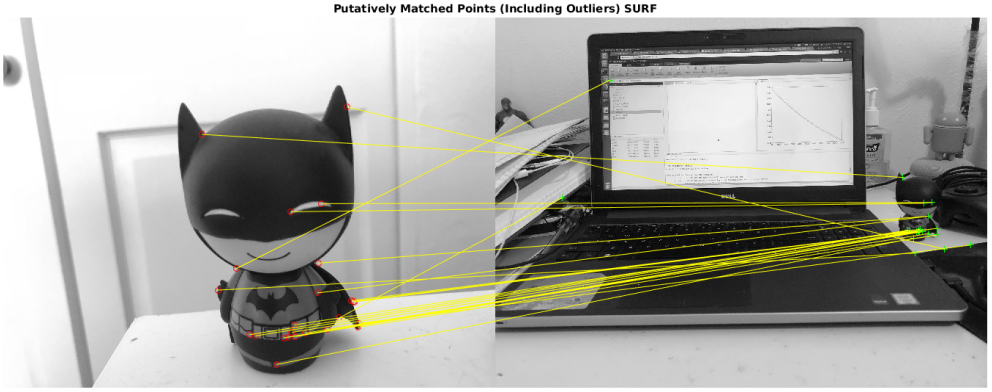
```
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 17%
Warning: Image is too big to fit on screen; displaying at 17%
Warning: Image is too big to fit on screen; displaying at 33%
```

300 Strongest SURF Feature Points from Box Image

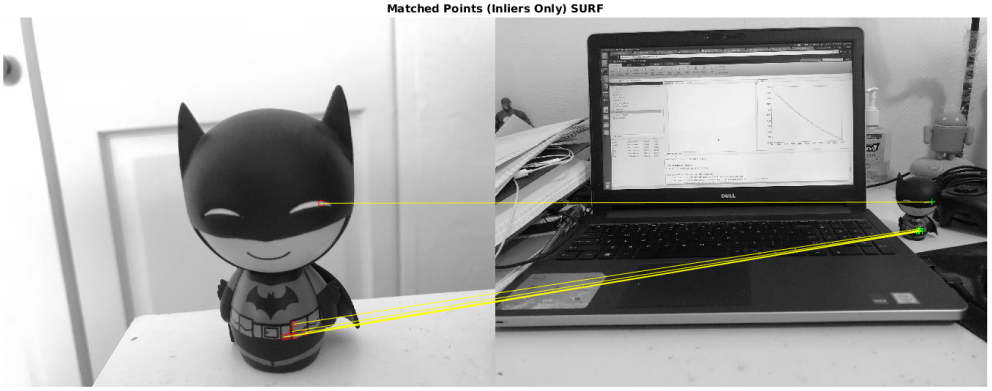
300 Strongest Feature Points from Scene Image SURF

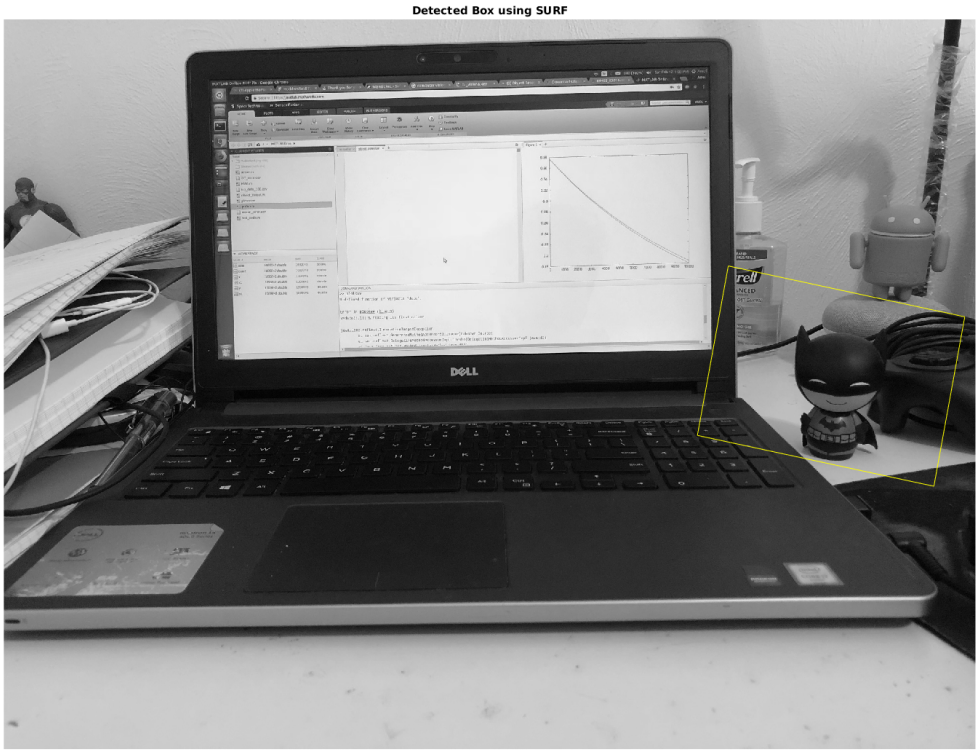


Putatively Matched Points (Including Outliers) SURF



Matched Points (Inliers Only) SURF





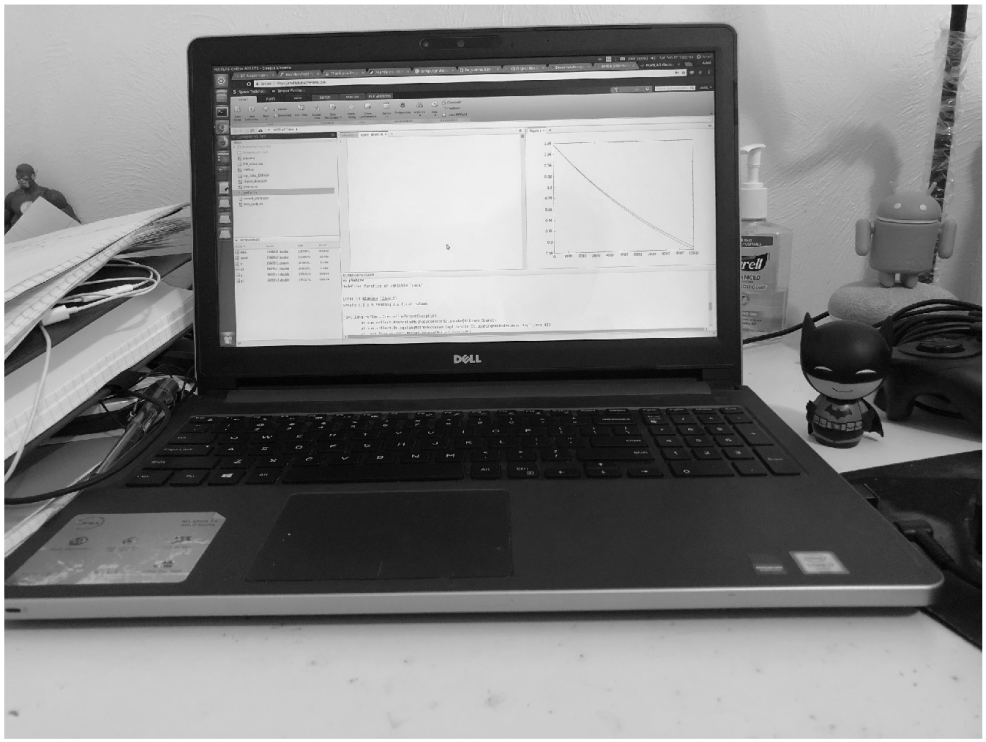
Implementation of BRISK based feature detection and object detection

Detect the featrure points using SURF feature detection

```
boxPointsB = detectBRISKFeatures(boxImage);
scenePointsB = detectBRISKFeatures(sceneImage);
figure;
imshow(sceneImage);


% Visualizing the strongest points in the object image
figure;
imshow(boxImage);
title('300 Strongest BRISK Feature Points from Box Image');
hold on;
plot(selectStrongest(boxPointsB, 300));


% Visualizing the strongest points in the target scene image
figure;
imshow(sceneImage);
title('300 Strongest BRISK Feature Points from Scene Image');
hold on;
plot(selectStrongest(scenePointsB, 300));
```

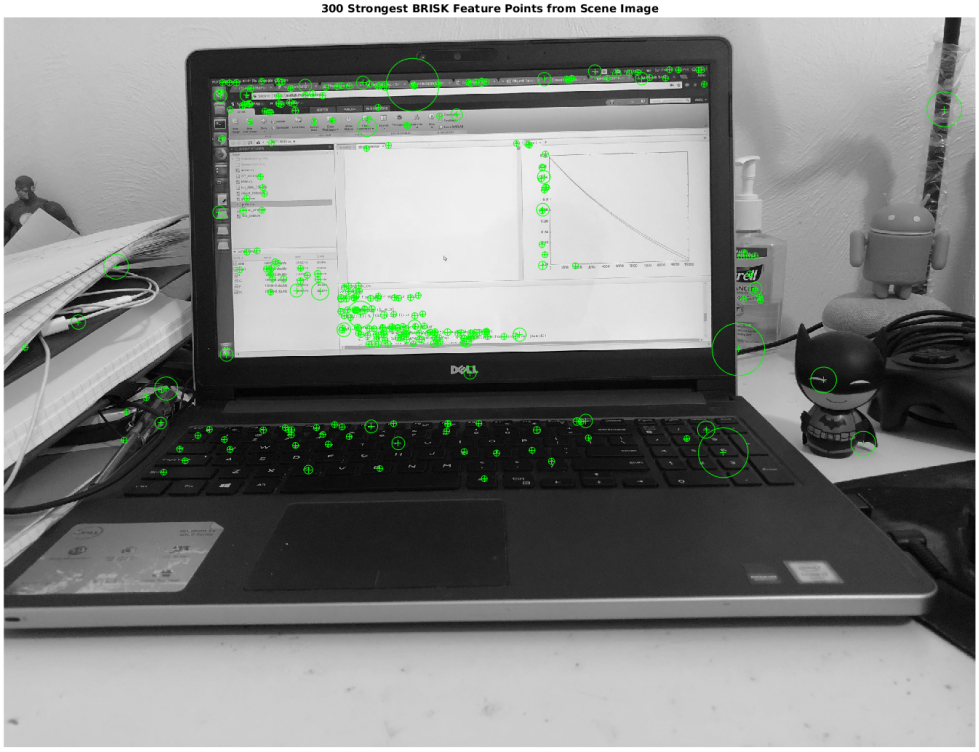
```
% extracting feature descriptors at interest points in both images
boxFeaturesB= extractLBPFeatures(boxImage);
sceneFeaturesB = extractLBPFeatures(sceneImage);

% Matching the features using the descriptors
boxPairsB = matchFeatures(boxFeaturesB, sceneFeaturesB);

% Display the matched features
matchedBoxPointsB = boxPointsB(boxPairsB(:, 1), :);
matchedScenePointsB = scenePointsB(boxPairsB(:, 2), :);
figure;
showMatchedFeatures(boxImage, sceneImage, matchedBoxPointsB, ...
    matchedScenePointsB, 'montage');
title('Putatively Matched Points (Including Outliers) using BRISK');


disp('BRISK Detected very few matchpoints for this data set to
 calculate the estimated geometry transform ');
```

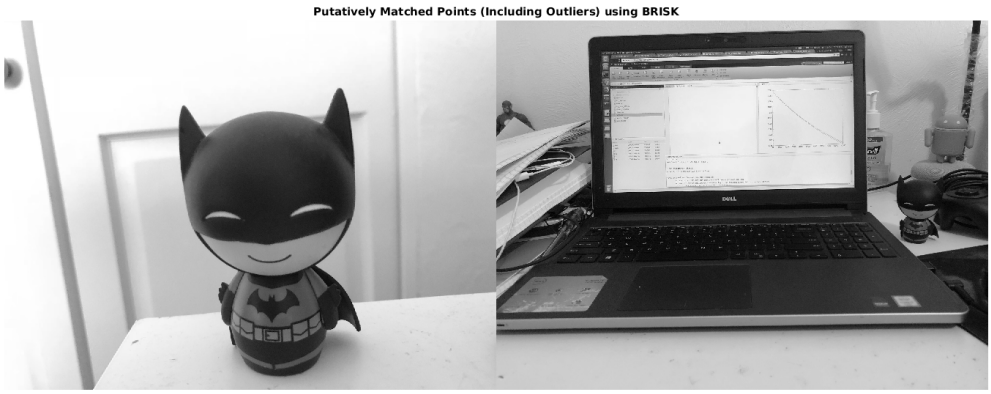
```
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 17%
BRISK Detected very few matchpoints for this data set to calculate the
 estimated geometry transform
```



300 Strongest BRISK Feature Points from Box Image

300 Strongest BRISK Feature Points from Scene Image



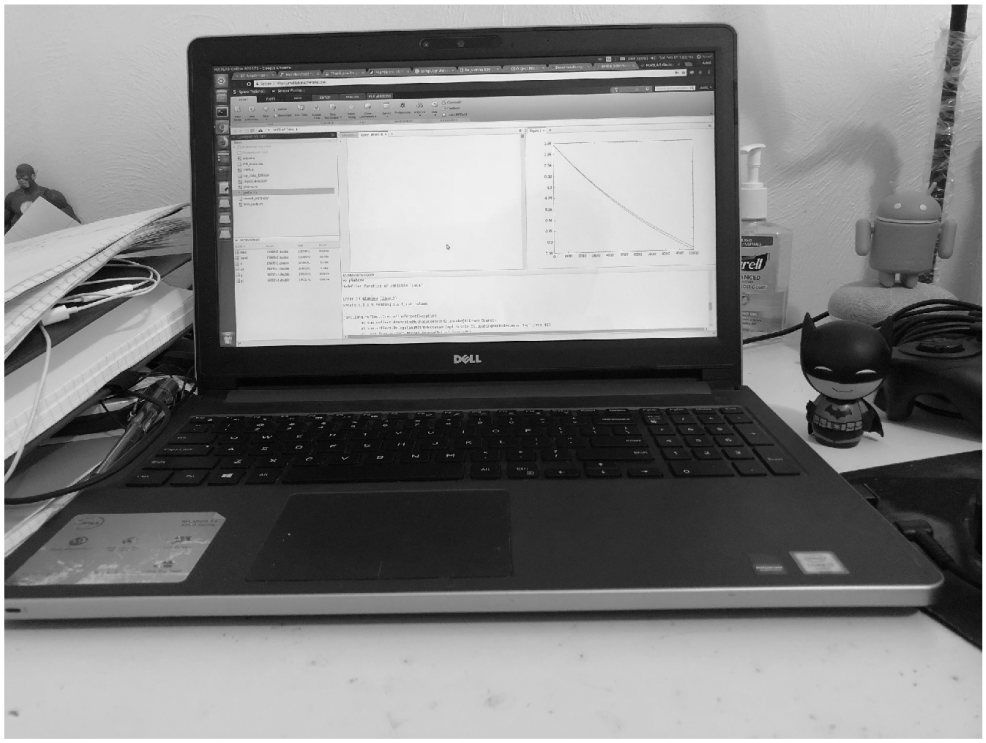


Implementation of MSER based feature detection and object detection

```
% Detect the featrure points using MSER feature detection
boxPointsM = detectMSERFeatures(boxImage);
scenePointsM = detectMSERFeatures(sceneImage);
figure;
imshow(sceneImage);


% Visualizing the strongest points in the object image
figure;
imshow(boxImage);
title('MSER Feature Points from Box Image');
hold on;
plot(boxPointsM);


% Visualizing the strongest points in the target scene image
figure;
imshow(sceneImage);
title('MSER Feature Points from Scene Image SURF');
hold on;
plot(scenePointsM);

% extracting feature descriptors at interest points in both images
[boxFeaturesM, boxPointsM] = extractFeatures(boxImage, boxPointsM);
[sceneFeaturesM, scenePointsM] = extractFeatures(sceneImage,
 scenePointsM);

% Matching the features using the descriptors
boxPairsM = matchFeatures(boxFeaturesM, sceneFeaturesM);

% Display the matched features
matchedBoxPointsM = boxPointsM(boxPairsM(:, 1), :);
matchedScenePointsM = scenePointsM(boxPairsM(:, 2), :);
figure;
```

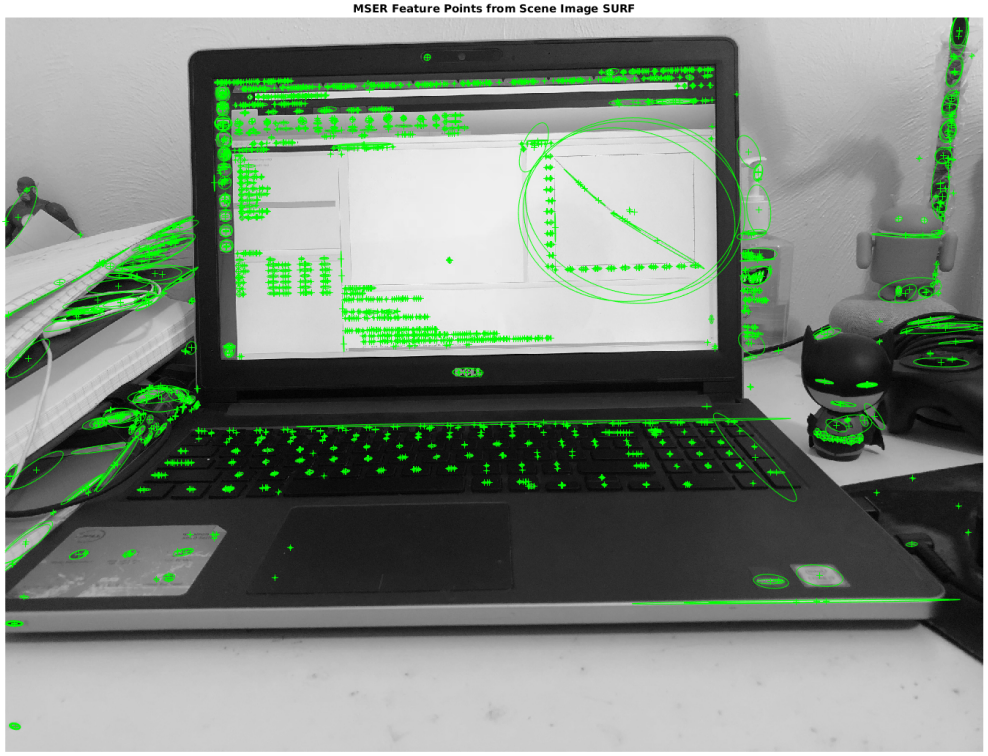
```
showMatchedFeatures(boxImage, sceneImage, matchedBoxPointsM, ...
    matchedScenePointsM, 'montage');
title('Putatively Matched Points (Including Outliers) MSER');
hold on

% Locating the object in the scene
[tformM, inlierBoxPointsM, inlierScenePointsM] = ...
    estimateGeometricTransform(matchedBoxPointsM,
 matchedScenePointsM, 'affine');


% Displaying the matched points after removing the outliers
figure;
showMatchedFeatures(boxImage, sceneImage, inlierBoxPointsM, ...
    inlierScenePointsM, 'montage');
title('Matched Points (Inliers Only) MSER');


%Drawing bounding box on the reference image
boxPolygonM = [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


%Transforming polygon in the coordinate system of the transformed
 image
newBoxPolygonM = transformPointsForward(tformM, boxPolygonM);


% Displaying the detectd object
figure;
imshow(sceneImage);
hold on;
line(newBoxPolygonM(:, 1), newBoxPolygonM(:, 2), 'Color', 'g');
title('Detected Box using MSER')
```

```
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 33%
Warning: Image is too big to fit on screen; displaying at 17%
Warning: Image is too big to fit on screen; displaying at 17%
Warning: Image is too big to fit on screen; displaying at 33%
```

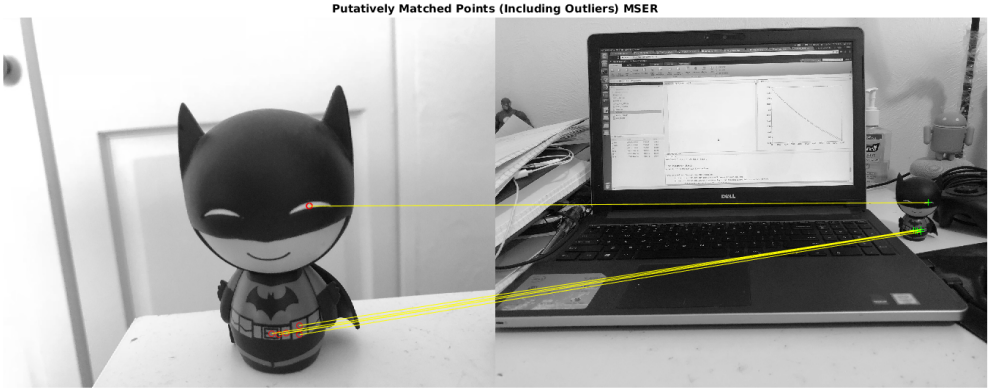


MSER Feature Points from Box Image

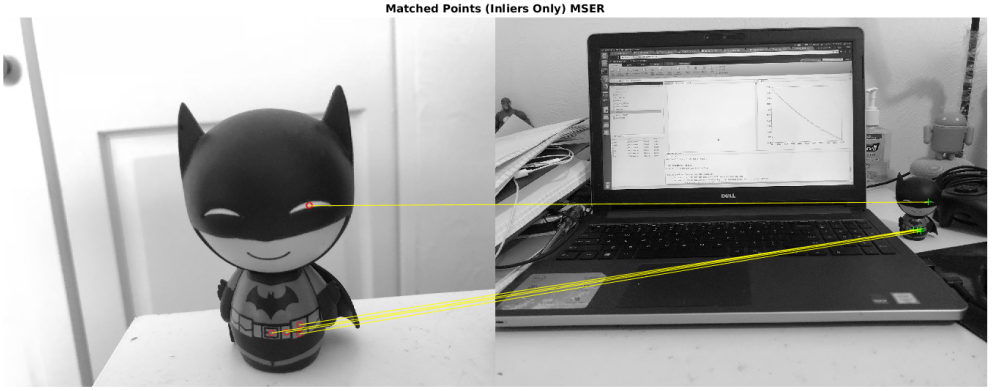
MSER Feature Points from Scene Image SURF

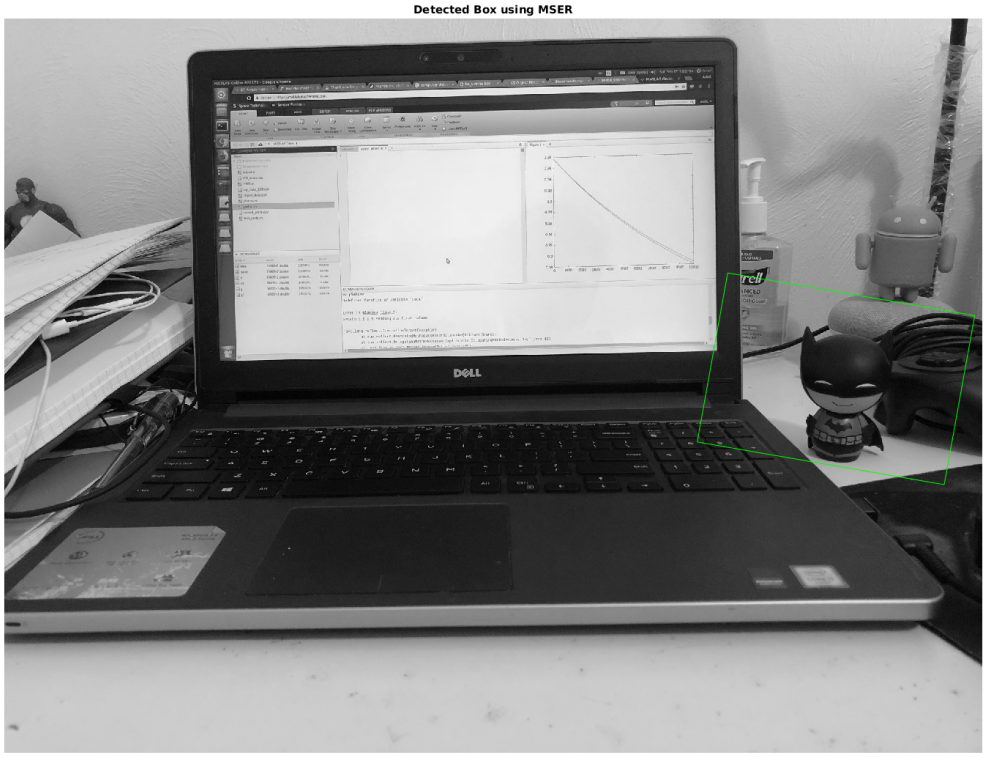


Putatively Matched Points (Including Outliers) MSER



Matched Points (Inliers Only) MSER





Published with MATLAB® R2016a