Laboratory of Image Processing

Image Rotation & Object Detection

Pier Luigi Mazzeo

pierluigi.mazzeo@cnr.it

Institute of Applied Sciences and Intelligent Systems

ScienceApp

National Research Council of Italy

Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

Step 1: Read Image

```
original = imread('cameraman.tif');
imshow(original);
text(size(original,2),size(original,1)+15, ...
    'Image courtesy of Massachusetts Institute of
Technology', ...
    'FontSize',7,'HorizontalAlignment','right');
```

Step 2: Resize and Rotate the Image

```
scale = 0.7;
J = imresize(original, scale); % Try varying the scale factor.
theta = 30;
distorted = imrotate(J,theta); % Try varying the angle, theta.
figure, imshow(distorted)
```

Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

Step 3: Find Matching Features Between Images

```
%Detect features in both images.
ptsOriginal  = detectSURFFeatures(original);

ptsDistorted = detectSURFFeatures(distorted);
%Extract features descriptors.
 [featuresOriginal,  validPtsOriginal]  =
                  extractFeatures(original,  ptsOriginal);
 [featuresDistorted, validPtsDistorted] =
                  extractFeatures(distorted, ptsDistorted);

%Match features using their descriptors.

 indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
```

Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

```
 %Retrieve locations of corresponding points of each image.

matchedOriginal  = validPtsOriginal(indexPairs(:,1));

matchedDistorted = validPtsDistorted(indexPairs(:,2));
 %Show putative matched points

figure;

showMatchedFeatures(original,distorted,matchedOriginal,matchedDistorted);

title('Putatively matched points (including outliers)');
```

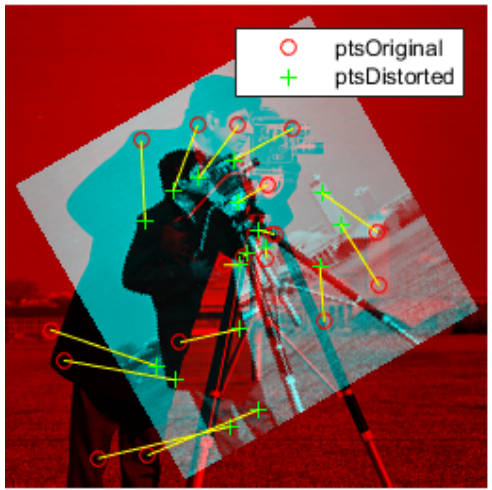
Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

Step 4: Estimate Transformation

Find a transformation corresponding to the matching point pairs using the statistically robust M-estimator SAmple Consensus (MSAC) algorithm, which is a variant of the RANSAC algorithm. It removes outliers while computing the transformation matrix. You may see varying results of the transformation computation because of the random sampling employed by the MSAC algorithm.

```
[tform, inlierDistorted, inlierOriginal] =
estimateGeometricTransform(...
    matchedDistorted, matchedOriginal, 'similarity');
```

```
%Display matching point pairs used in the computation of the
transformation
figure;

showMatchedFeatures(original,distorted,inlierOriginal,inlierDistorted);

title('Matching points (inliers only)');

legend('ptsOriginal','ptsDistorted');
```



Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

Step 5: Solve for Scale and Angle

Use the geometric transform, tform, to recover the scale and angle. Since we computed the transformation from the distorted to the original image, we need to compute its inverse to recover the distortion.

```
Let sc = s*cos(theta)
Let ss = s*sin(theta)
Then, Tinv = [sc -ss  0;
              ss  sc  0;
              tx  ty  1]
where tx and ty are x and y translations, respectively.
```

```
%Compute the inverse transformation matrix
          Tinv  = tform.invert.T;

          ss = Tinv(2,1);
          sc = Tinv(1,1);
          scaleRecovered = sqrt(ss*ss + sc*sc)
          thetaRecovered = atan2(ss,sc)*180/pi
```

The recovered values should match your scale and angle values selected in Step 2: Resize and Rotate the Image.

Find Image Rotation and Scale Using Automated Feature Matching and RANSAC

Step 6: Recover the Original Image

```
%Recover the original image by transforming the
Distorted image.

outputView = imref2d(size(original));

recovered  =
imwarp(distorted,tform,'OutputView',outputView);

%Compare recovered to original by looking at
them side-by-side in a montage.

figure,
imshowpair(original,recovered,'montage')
```

The recovered (right) image quality does not match the original (left) image because of the distortion and recovery process. In particular, the image shrinking causes loss of information. The artifacts around the edges are due to the limited accuracy of the transformation. If you were to detect more points in Step 4: Find Matching Features Between Images, the transformation would be more accurate.

Object Detection In A Cluttered Scene Using Point Feature Matching

This 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 amount 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: Read Images

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

%Read the reference image containing the
object of interest.
```

Object Detection In A Cluttered Scene Using Point Feature Matching

Step 1: Read Images

```
%Read the target image containing a cluttered scene.

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

Object Detection In A Cluttered Scene Using Point Feature Matching

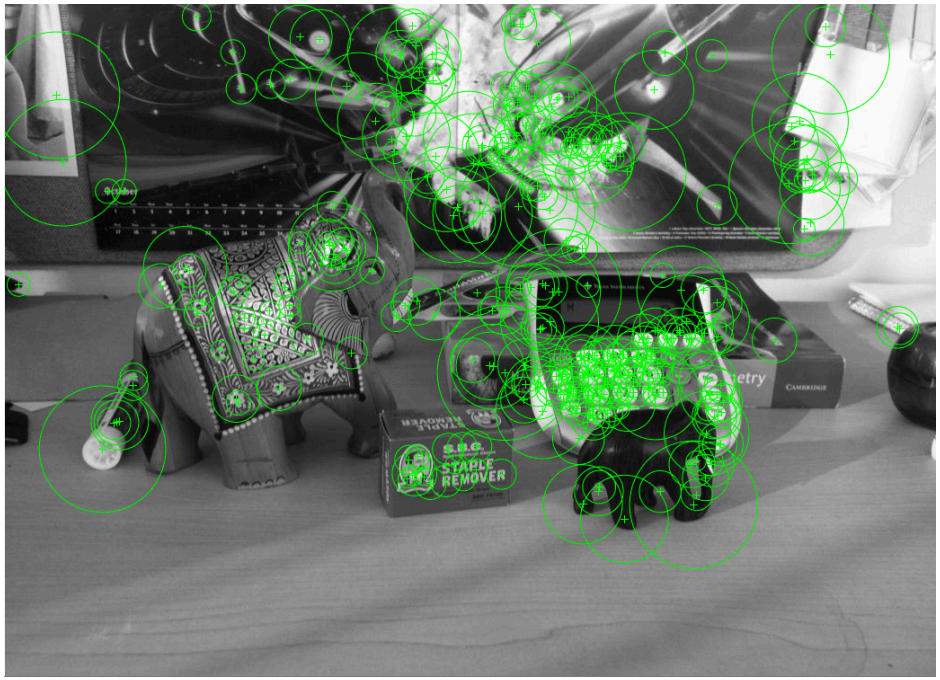
Step 2: Detect Feature Points

```
%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));
```

Object Detection In A Cluttered Scene Using Point Feature Matching

Step 2: Detect Feature Points

```
%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));
```



Object Detection In A Cluttered Scene Using Point Feature Matching

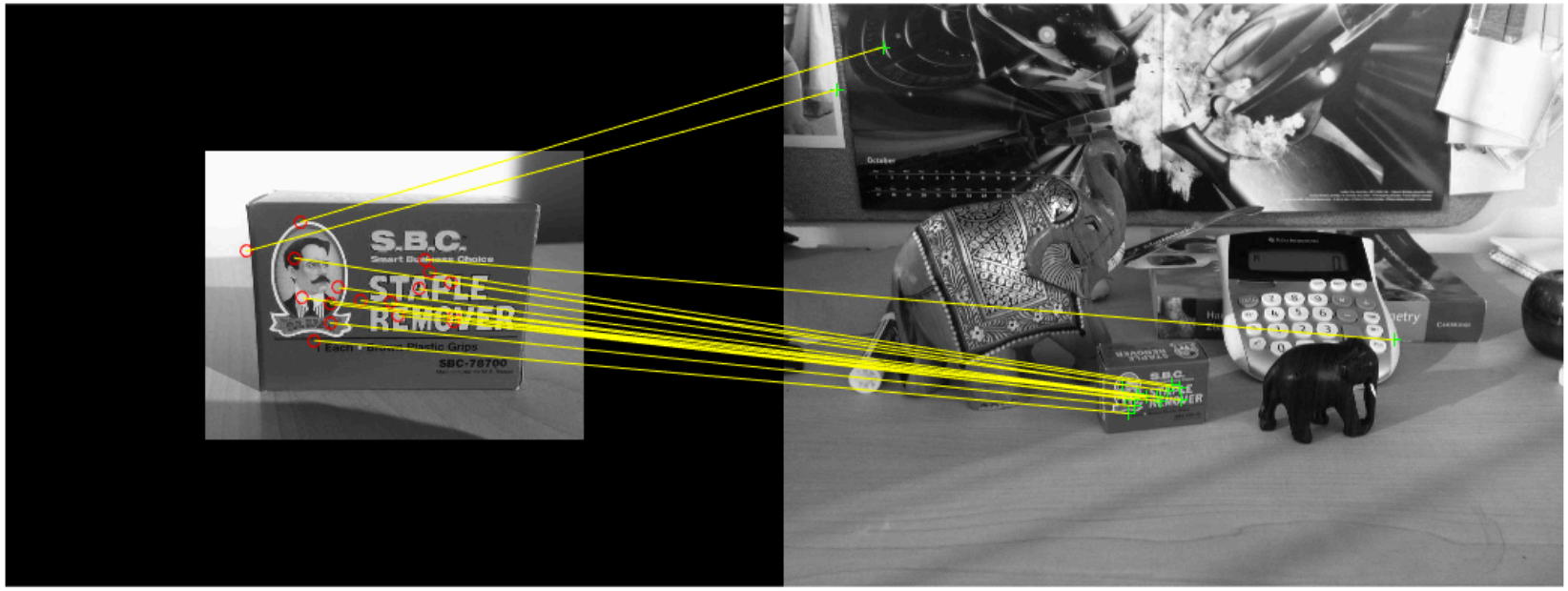
Step 3: Extract Feature Descriptors

```
%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)');
```

Object Detection In A Cluttered Scene Using Point Feature Matching



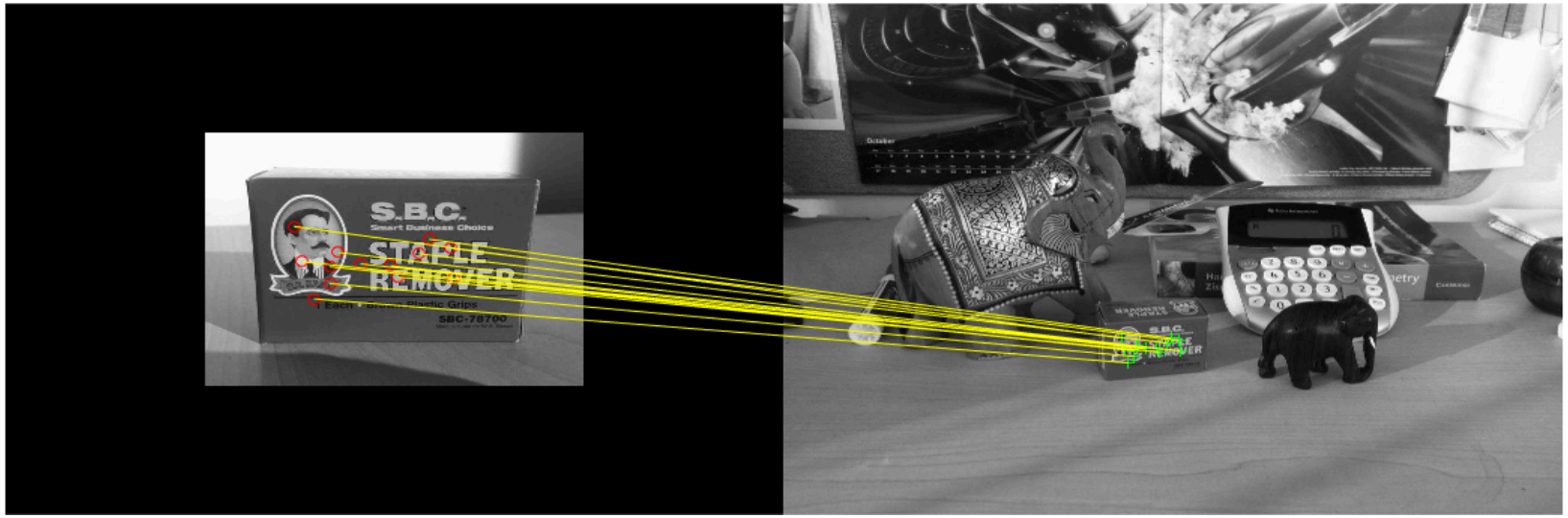
Putatively Matched Points (Including Outliers)

Object Detection In A Cluttered Scene Using Point Feature Matching

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)');
```

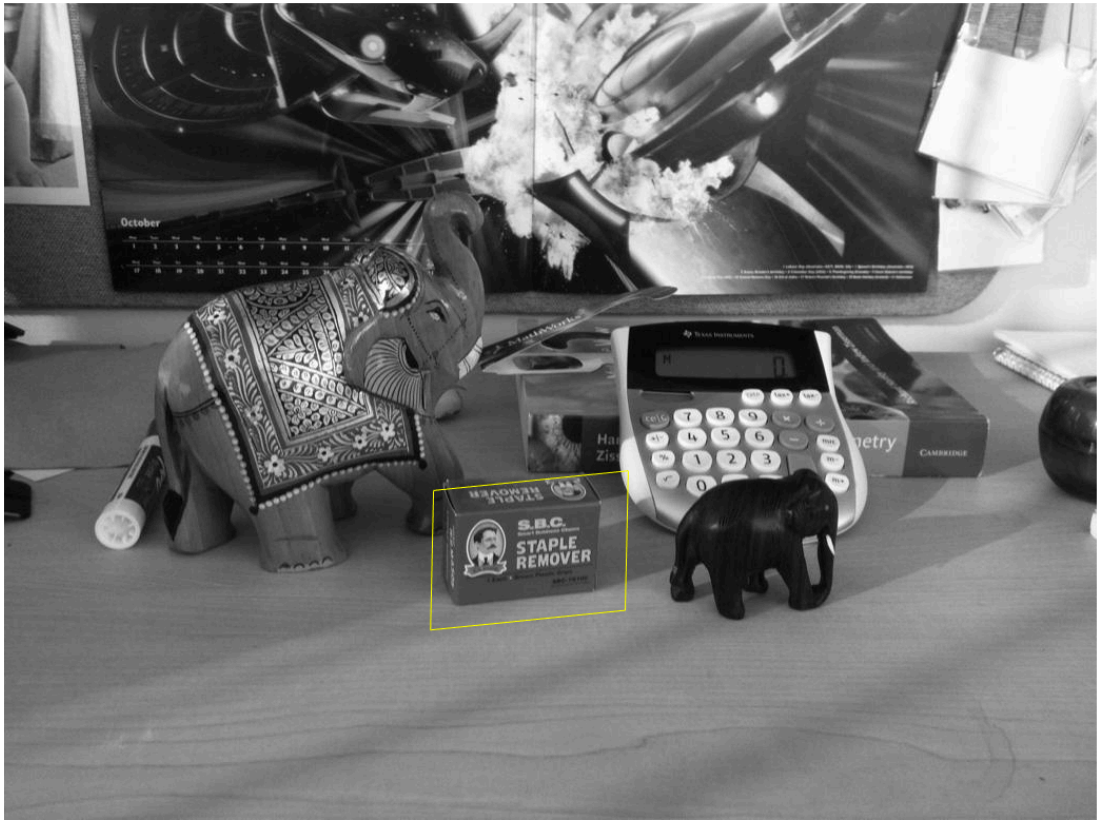


Object Detection In A Cluttered Scene Using Point Feature Matching

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

```
%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



The Detected Box

Object Detection In A Cluttered Scene Using Point Feature Matching

Exercise

Detect a second object by using the same steps as before.

Read an image containing the second object of interest.

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