Exercise on RANSAC

April 24, 2019

1 Harris Corner Detector

```
  imColor=imread('./data/library2.jpg');
figure;imshow(imColor);
if size(imColor,3) == 3
    imGray=rgb2gray(imColor);
else
    imGray = imColor;
end
im = im2double(imGray);

% here starts your code
......
......
......

[row,col] = nonmaxsuppts(C,'radius', 2);

% plot result
figure
img=imshow(imColor),title('my-Harris'),
hold on
plot(col,row, 'ro','MarkerSize',10),
hold off
```

In this exercise, you will work on Harris corner detector. You could do as follows:

1. load image (convert to gray image if the input image is a color image), convert to double array. Useful MATLAB functions: **imread, rgb2gray, im2double**.
2. compute horizontal and vertical first order derivatives: I_x and I_y and show them. Useful MATLAB functions: **edge, imshow**.
3. Compute three intermediate images $I_x^2 = I_x .* I_x$, $I_{xy} = I_x .* I_y$, $I_y^2 = I_y .* I_y$ ($.*$ means per-element product), show them.
4. Smooth them (I_x^2, I_{xy}, I_y^2) with the Gaussian template.Useful MATLAB functions: **fspecial, imfilter**.
5. Recall that $\mathbf{M} = \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix}$ and $\mathbf{C} = \det(\mathbf{M}) - k\text{trace}(\mathbf{M})^2$, now you need to compute the response (score) of the corner detector at each pixel. **Hint**: you do not need to form $\mathbf{M}$ explicitly since the determinant and trace of a 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ are $ab - cd$ and $a + d$, respectively. Thus,

   $$\mathbf{C} = I_x^2 .* I_y^2 - I_{xy}^2 - k .* (I_x^2 + I_y^2).^2$$

   Here, $.*, .^2$ are per-element operations. k is usually set to $0.04 - 0.06$.
6. Threshold $\mathbf{C}$ as

   $$\mathbf{C}(\mathbf{C} < \tau) = 0$$

   Typical value for τ ranges from 0.1 to 0.8 times the maximum value of $\mathbf{C}$.

7. Apply Non-maxima suppression (code is provided), e.g. **[row,col] = nonmaxsuppts(C,'radius', 2)**.

Images can be found in a folder called **data**. You can use the sample code listed above.

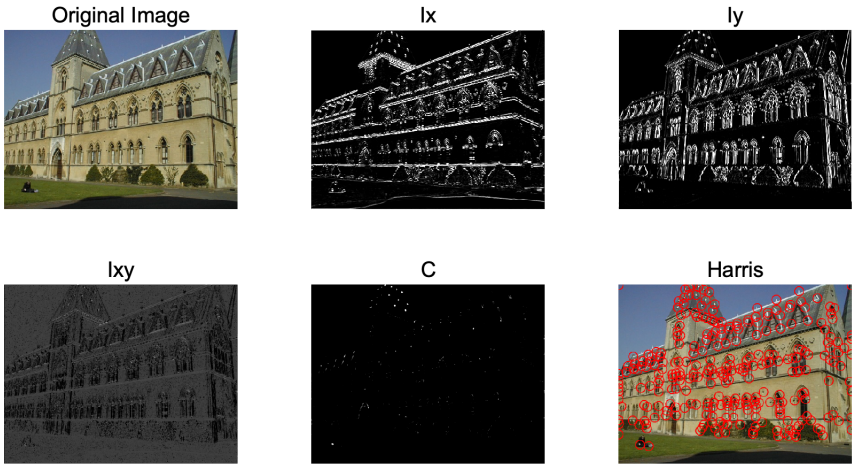

Figure 1: Example of Harris corner detection.

2 RANSAC

In this exercise, you will work on line fitting using RANSAC. Use the following code to generate simulated data.

```
clc;close all;clear all;

% generate 500 points
  [X, lineTrue] = gen_line_data(500);

% inhomogeneous to homogeneous
  ph = tohomo(X);

  % your code starts here, do RANSAC line fitting,
  % the result is the parameters of the fitted line, i.e. params(1)*x+params(2)*y+params(3)=0
  params = xxxx(ph);

  % Results Visualization
  figure;
  hold on
  xmin = min(X(1,:));
  xmax = max(X(1,:));
  xx = linspace(xmin,xmax,100);
  yy1 = -(lineTrue(1).*xx+lineTrue(3))./(lineTrue(2)+1e-6);
  yy2 = -(params(1).*xx+params(3))./(params(2)+1e-6);
  plot(xx, yy1, 'k-.','LineWidth',1);
  plot(xx, yy2, 'm--','LineWidth',2);
  xlabel('x')
  ylabel('y')
  title('RANSAC results for 2D line estimation')
  axis equal tight
  set(gca,'FontName','Arial','FontSize',20);
```

Recall the pipeline of RANSAC:

1. Randomly select n samples from observations: for 2D line fitting, we need at least 2 points, so $n = 2$. Useful MATLAB functions: **randperm**.

2. Model estimation: here we need to estimate the line parameters. Recall estimating a line from two points with homogeneous coordinates which you have already done in the first assignment.

3. Compute Consensus: apply a model-specific loss function to each observation and the model obtained, the response could serve as the consensus. So here we can use the point-to-line distance as the model-specific loss function. The distance obtained is the consensus.

4. Classifier inliers and outliers using a predefined threshold. You can try with different threshold. Save the model with the largest number of iniliers.

5. Iterate.

Result expected:

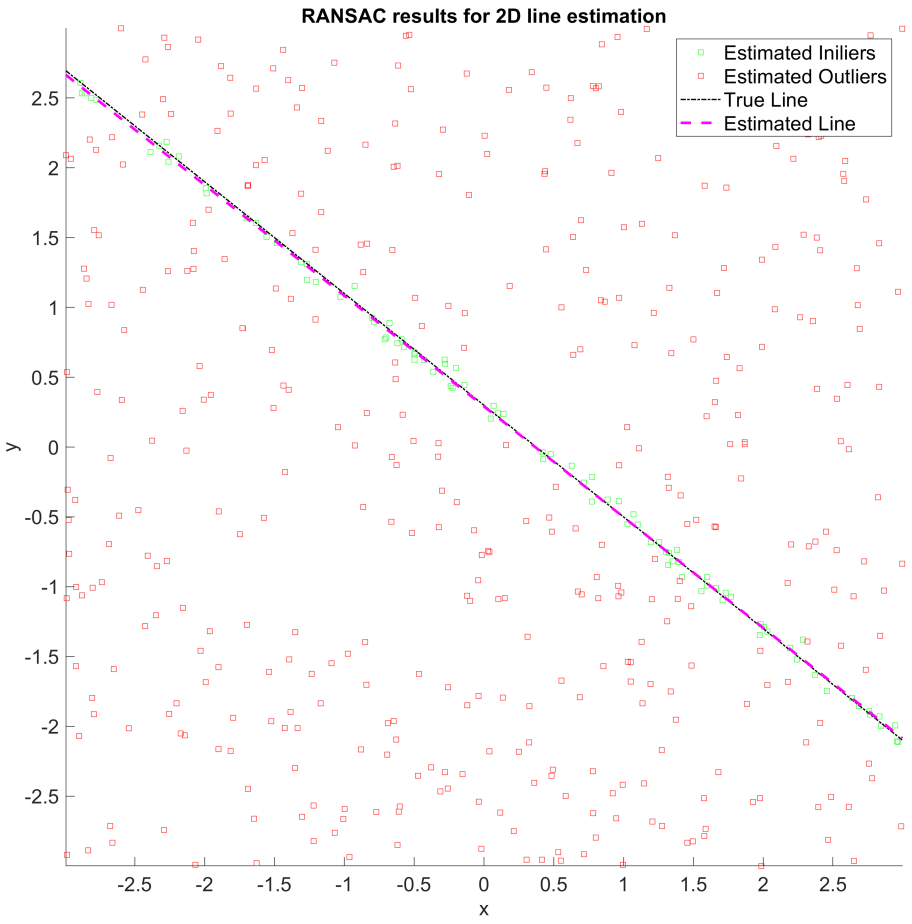

Figure 2: Example of RANSAC line fitting.

References