

Exercise on RANSAC

April 24, 2019

1 Harris Corner Detector

```
1  imColor=imread('./data/library2.jpg');
2  figure;imshow(imColor);
3  if size(imColor,3) == 3
4      imGray=rgb2gray(imColor);
5  else
6      imGray = imColor;
7  end
8  im = im2double(imGray);
9
10 % here starts your code
11 .....
12 .....
13 .....
14
15 [row,col] = nonmaxsuppts(C, 'radius', 2);
16
17 % plot result
18 figure
19 img=imshow(imColor),title('my-Harris'),
20 hold on
21 plot(col,row, 'ro','MarkerSize',10),
22 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 $I_x^2 = I_x \cdot I_x$, $I_{xy} = I_x \cdot I_y$, $I_y^2 = I_y \cdot I_y$ (\cdot 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 are: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is $ab - cd$ and $a + d$, respectively. Thus,

$$\mathbf{C} = I_x^2 \cdot I_y^2 - I_{xy}^2 - k \cdot \text{trace}(I_x^2 + I_y^2)^2$$

Here, \cdot , 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 values for τ range 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, 'N', 1000)`.

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

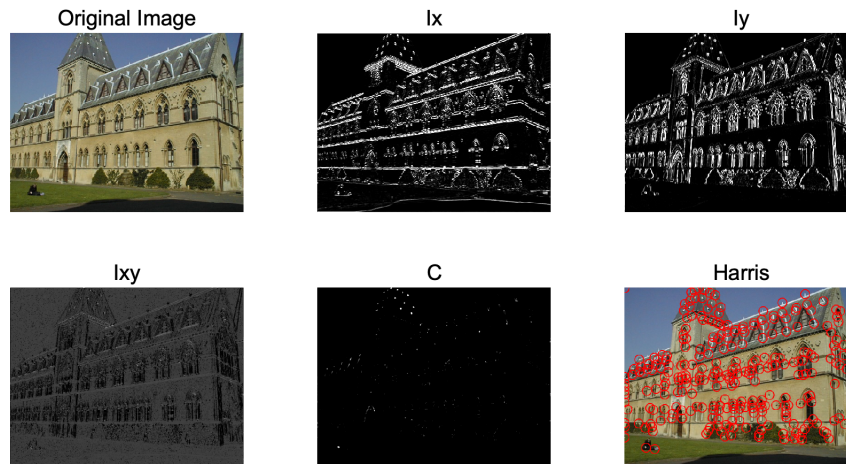


Figure 1: Example of Harris corner detection.

2 RANSAC

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

```

1  clc;close all;clear all;
2
3  [X, lineTrue] = gen_line_data(500);
4
5  ph = tohomo(X);
6
7  % your code starts here, do RANSAC line fitting
8  params = xxxx(ph);
9
10 % Results Visualization
11 figure;
12 hold on
13 xmin = min(X(1,:));
14 xmax = max(X(1,:));
15 xx = linspace(xmin,xmax,100);
16 yy1 = -(lineTrue(1).*xx+lineTrue(3))./(lineTrue(2)+1e-6);
17 yy2 = -(params(1).*xx+params(3))./(params(2)+1e-6);
18 plot(xx, yy1, 'k-.', 'LineWidth',1);
19 plot(xx, yy2, 'm--', 'LineWidth',2);
20 xlabel('x')
21 ylabel('y')
22 title('RANSAC results for 2D line estimation')
23 axis equal tight
24 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 inliers.
5. Iterate.

Result expected:

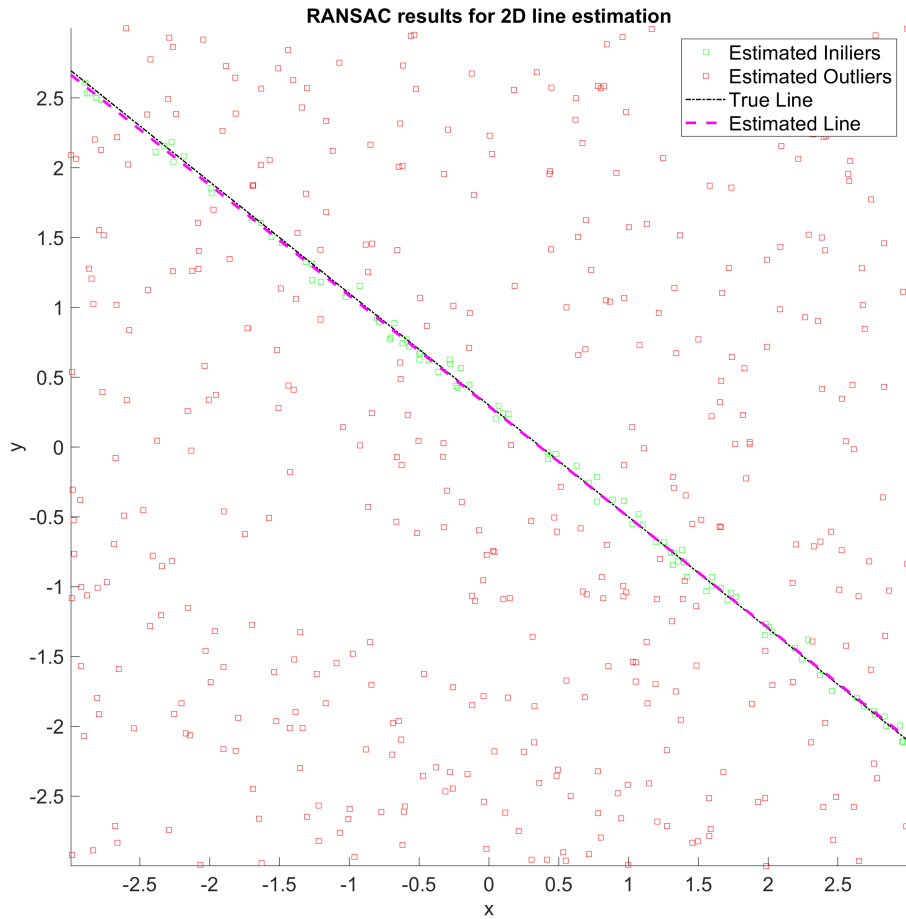


Figure 2: Example of RANSAC line fitting.