Computer Vision 1: Assignment 3

Summary:

1. Hough transform for circle detection and Canny edge detection (programming).
2. RANSAC edge proposal (programming)
3. RANSAC number of trials (theoretical)

Submission instructions:

- For each programming task, submit a `.py` source code file. Do not need to include any images or data files you used.
- For each pen & paper task, submit a `.pdf` file. Type your solution in LaTeX, Word, or another text editor of your choice and convert it to PDF – **do not submit photographs or scans of handwritten solutions!**
- In all files, include at the top names of all students in the group
- Choose exactly one person in your group that submits your solution via Moodle, it will count for the entire group.

1 Circle detection

Figure 1 shows a selection of the last Finnish pre-euro coins (10, 5, and 1 marks; and 50 and 10 pennis, respectively). We apply the Hough transformation to detect the coins.

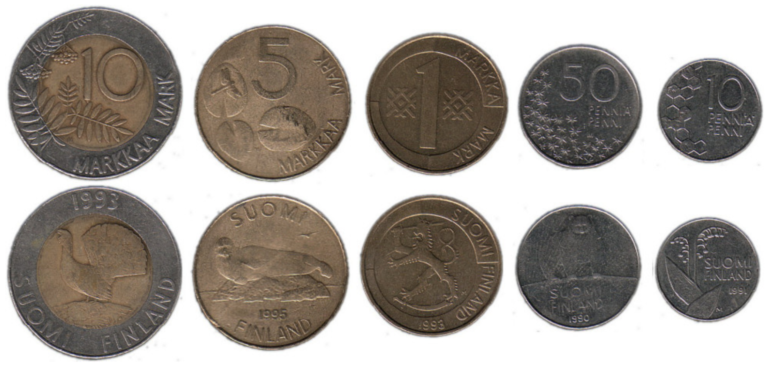

Figure 1: A selection of Finnish coins.

- Download the image `coins.jpg` from Moodle. Read it and convert to grayscale.
- The mint specifies that the diameter of the 5 mark coin is 24.5 millimetres[1]. The resolution of the image is approximately 0.12 mm/pixel. Calculate the radius r of the coin in pixels.
- Apply the Canny edge detector to find edges in the grayscale image. Use the built-in function `skimage.feature.canny`. Visualize the edges and check that the outlines of the coins are detected.

[1]The diameters of the coins shown in Figure 1 from left to right are: 27.25mm, 24.5mm, 22.25mm, 19.7mm, and 16.3mm.

- Use `skimage.transform.hough_circle` to calculate the Hough transform of the edge detection result. Use the radius you calculated above. Draw the result. You should obtain something similar to Figure 2 that peaks strongly around the center of the 5 mark coin.
- Using `skimage.transform.hough_circle_peaks`, select the two highest peaks from the Hough transform. Get all outputs from the function, i.e., your call should look like: `accums, cx, cy, radii = hough_circle_peaks(...)`.
- Apply `matplotlib.patches.Circle` to draw the circles at the coordinates found superimposed on the original image. See `https://matplotlib.org/api/_as_gen/matplotlib.patches.Circle.html` for more help. Use `from matplotlib.patches import Circle` to import the circle tool to your code.

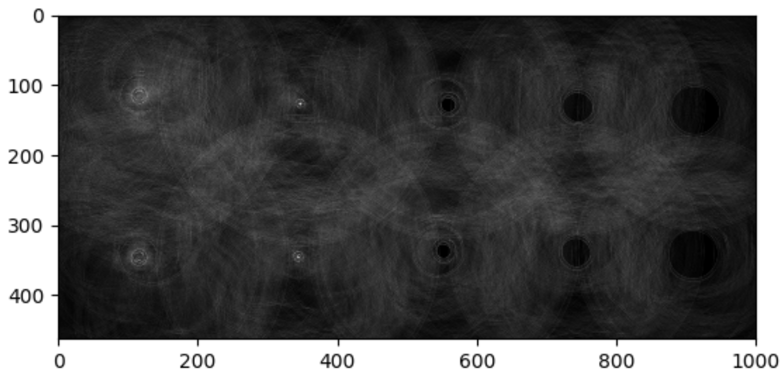

Figure 2: Circle Hough transform result. The strongest peaks highlight the 5 mark coin centers (compare to Figure 1).

2 Fitting a line using RANSAC

Suppose you have some prior feature points extracted using some black box algorithm and they are stored in "noisyedgepoints.npy" that you can download from Moodle. Those feature points represent a noisy version of an edge. Some of them are outliers and are not supposed to be part of the edge. Use the scikit library to fit the features such that you generate a best fitting line using RANSAC (`https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RANSACRegressor.html`). Alternatively, you may also implement your own version of RANSAC from scratch. The data file can be read as:

```
with open('noisyedgepoints.npy', 'rb') as f:
    X = np.load(f)
    y = np.load(f)
```

After fitting the line, plot the outliers that got filtered by the RANSAC algorithm and the inliers respectively. The result should be similar to Figure 3.

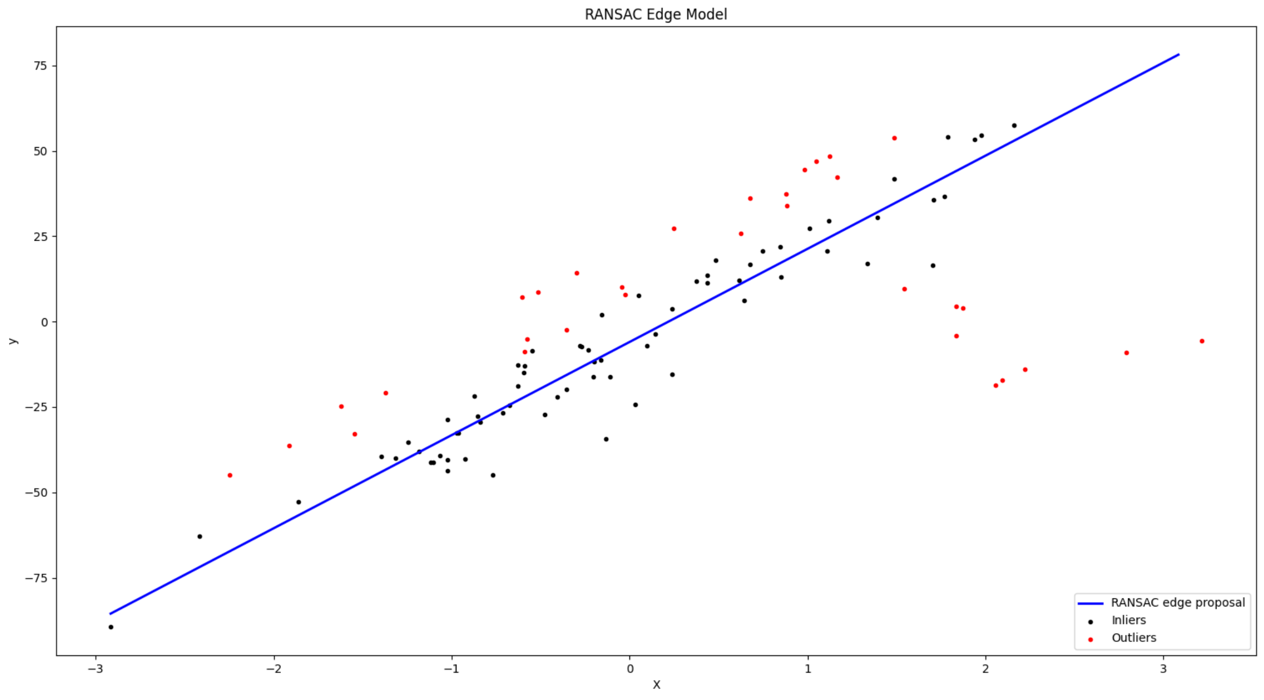

Figure 3: Plot of results for task 2

3 Pen and paper task

This task expands on Question 2 of the "Hough Transform and RANSAC" quiz:

- Suppose the fraction of outliers $\epsilon = \frac{number of outliers}{total number of datapoints}$.
- The minimum number of samples to fit your model is m.
- k is the number of running trials so that with probability p, at least one set of samples is free from outliers.

Prove that the number of trials is $k = \frac{log(1-p)}{log(1-(1-\epsilon)^m)}$.