Machine Vision Homework

Contents

LIST OF FIGURES

1. Homework topic

In this Machine Vision homework, I am going to implement the RANSAC algorithm using MATLAB Script. In order to make sure that the RANSAC algorithm works properly, I am going to detect lines by using the RANSAC algorithm.

2. RANSAC algorithm

2.1 Introduction

RANSAC is an abbreviation for "RANdom SAmple Consensus". It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. The algorithm was first published by Fischler and Bolles at SRI International in 1981. [1]

Unlike many of the common robust estimation techniques suchasM-estimators and least-median-squares thathavebeen adopted by the computer vision community from the statistics literature, RANSAC was developed from within the computer vision community. [2]

A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. In addition to this, the data can be subject to noise. The outliers can come, e.g., from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data. [1] Unlike conventional sampling techniques that use as much of the data as possible to obtain an initial solution and then proceed to prune outliers, RANSAC uses the smallest set possible and proceeds to enlarge this set with consistent data points. [2]

2.2 Main Disciplines

The RANSAC algorithm works by randomly selecting a subset of data points from the given dataset, estimating the model parameters using the selected data points, and checking the model's goodness of fit. This process is repeated a large number of times until the best model is found. [3]

The example is shown below. Suppose you have a set of data points that represent a line.

But, some of these data points are outliers, i.e., they do not lie on the line. The goal is to estimate the parameters of the line that best fit the data points. [3]

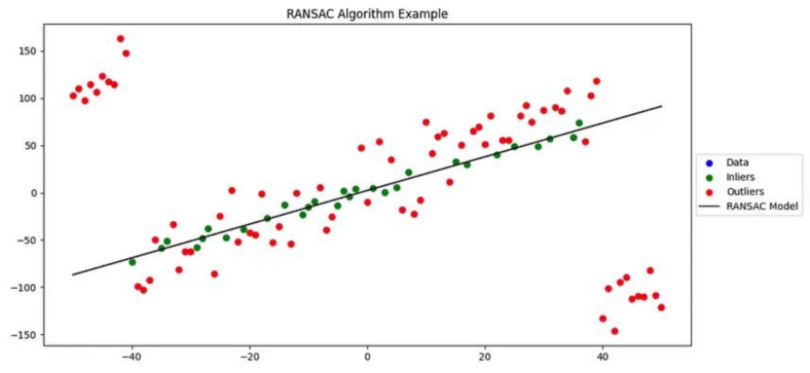


Figure 1: RANSAC Algorithm Example (Source: [3])

The RANSAC algorithm will begin by selecting two random points from the dataset and estimating the parameters of the line passing through these two points. Next, the algorithm will calculate the distance between each point in the dataset and the line. If the distance is below a threshold value, the point is considered an inlier and is used to estimate the parameters of the line. This process is repeated several times until the best line is found. The RANSAC algorithm involves several mathematical calculations. The first step is to estimate the parameters of the model. In our example, we want to estimate the parameters of a line passing through two points. The equation of the line can be represented as:

$$y = mx + b \tag{1}$$

Where m is the slope of the line, b is the y-intercept, and x and y are the coordinates of the data points.

The RANSAC algorithm uses the least-squares method to estimate the parameters of the line. The least-squares method involves minimizing the sum of the squared distances between the data points and the line. This is represented as:

$$\min(\sum(yi - mx - b)^2) \tag{2}$$

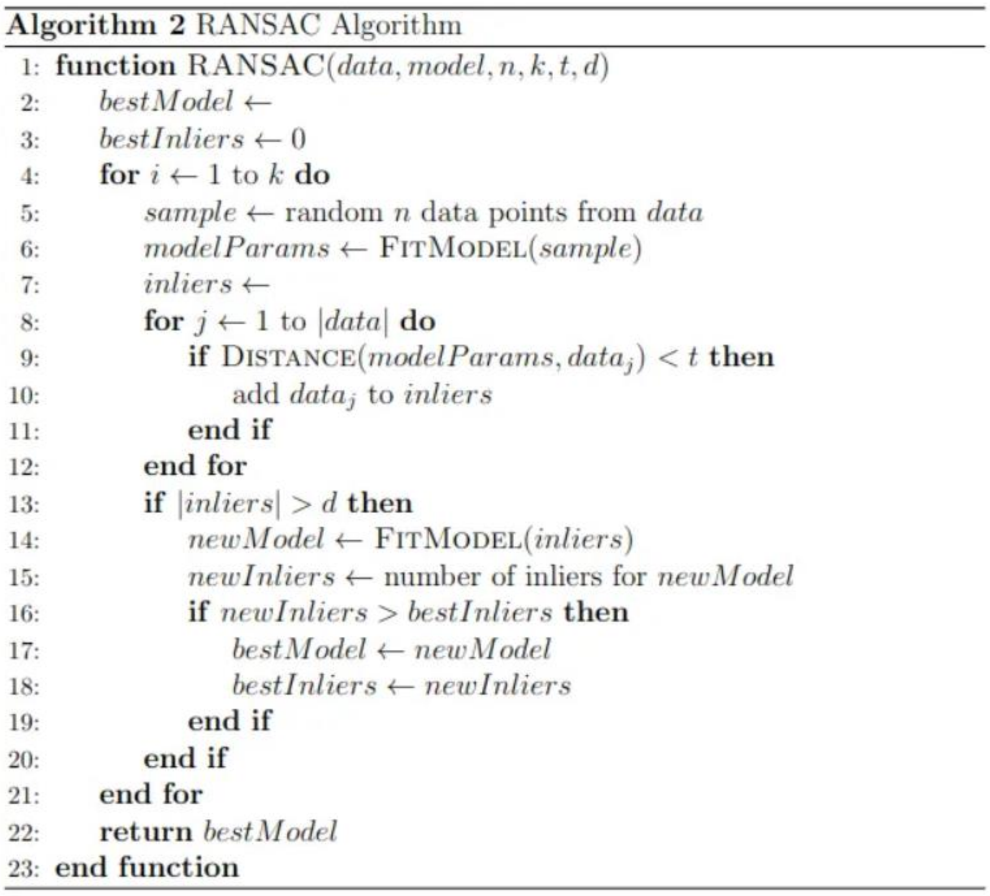
Where yi is the y-coordinate of the data point, xi is the x-coordinate of the data point, and m and b are the parameters of the line.

To calculate the distance between a point and the line, we use the following formula:

$$d = \frac{|y - mx - b|}{\sqrt{(1 + m^2)}} \tag{3}$$

Where d is the distance between the point and the line, and $\sqrt{}(1 + m^2)$ is a normalization factor.

[3]



```
Algorithm 2 RANSAC Algorithm
function RANSAC($data, model, n, k, t, d$)
    $bestModel \leftarrow$
    $bestInliers \leftarrow 0$
    for $i \leftarrow 1$ to $k$ do
        $sample \leftarrow$ random $n$ data points from $data$
        $modelParams \leftarrow$ FitModel($sample$)
        $inliers \leftarrow$
        for $j \leftarrow 1$ to $|data|$ do
            if Distance($modelParams, data_j$) $< t$ then
                add $data_j$ to $inliers$
            end if
        end for
        if $|inliers| > d$ then
            $newModel \leftarrow$ FitModel($inliers$)
            $newInliers \leftarrow$ number of inliers for $newModel$
            if $newInliers > bestInliers$ then
                $bestModel \leftarrow newModel$
                $bestInliers \leftarrow newInliers$
            end if
        end if
    end for
    return $bestModel$
end function
```

Figure 2: RANSAC Algorithm (Source: [3])

2.3 Advantages and Disadvantages

2.3.1. Advantages

The RANSAC algorithm has several advantages over other algorithms, such as robustness, efficiency, and flexibility. [3] RANSAC is a robust algorithm that can handle a large amount of noise and outliers in the data. Also, it can be used with any model that can be estimated from a subset of the data. It is relatively simple to implement and computationally efficient. Moreover, RANSAC can provide a good approximation of the true model even when there are a large number of outliers in the data. [4]

2.3.2. Disadvantages

A disadvantage of RANSAC is that there is no upper bound on the time it takes to compute these parameters (except exhaustion). [1] Also, RANSAC is a heuristic

algorithm, which means that it does not guarantee the optimal solution. [4] The choice of parameters (n, k, t, d) can have a significant impact on the performance of the algorithm. Finding the optimal values for these parameters can be challenging. The algorithm can be sensitive to the initial random sample, which can lead to different results for different runs of the algorithm. [4]
When the number of iterations computed is limited the solution obtained may not be optimal, and it may not even be one that fits the data in a good way. In this way RANSAC offers a trade-off; by computing a greater number of iterations the probability of a reasonable model being produced is increased. Moreover, RANSAC is not always able to find the optimal set even for moderately contaminated sets and it usually performs badly when the number of inliers is less than 50%. Optimal RANSAC was proposed to handle both these problems and is capable of finding the optimal set for heavily contaminated sets, even for an inlier ratio under 5%. Another disadvantage of RANSAC is that it requires the setting of problem-specific thresholds. RANSAC can only estimate one model for a particular data set. As for any one-model approach when two (or more) model instances exist, RANSAC may fail to find either one. [1]

3. Implementation

The RANSAC algorithm is implemented by MATLAB Script code. The codes consist of two MATLAB Scripts, which are called test.m and ransac_linedetection.m. Each code is explained below.

■ test.m

This script calls the ransac_linedetection function from test.m.

```
% test
clc;
clear;
ransac_linedetection('testimage.png', 120, 0.1, 20);
```

ransac_linedetection's arguments are below:
ransac_linedetection(testimage, numIterations, thresholdDistance, minInliers)

・numIterations:

This parameter represents the number of iterations the RANSAC algorithm performs to find the best model. During each iteration, the algorithm randomly samples a minimal subset of points from the input data to fit a model. Increasing the number of iterations generally improves the chances of finding a better model. However, there is a trade-off between computational efficiency and accuracy. A higher number of iterations may increase computational cost but could lead to more robust line detection, especially in the presence of outliers.

・thresholdDistance (Distance Threshold to Consider a Point as an Inlier):

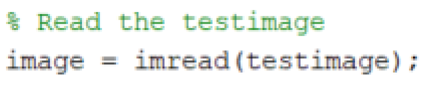
This parameter sets the maximum allowable distance between a data point and the model to consider the point as an inlier. Points within this threshold are considered part of the model; those beyond it are treated as outliers. A smaller threshold makes the algorithm more sensitive to the localization accuracy of data points around the model. It can lead to a more accurate fit but may also increase sensitivity to noise. A larger threshold allows for a more lenient fit, making the algorithm more robust to outliers but potentially sacrificing precision.

・minInliers: Minimum number of inliers to accept a line

This parameter specifies the minimum number of inliers required to accept a detected line as valid. Inliers are data points that fit well with the model. Setting a higher value for minInliers makes the algorithm more conservative, as it demands a stronger consensus among data points for line acceptance. This can help filter out spurious lines caused by noise. However, setting it too high may result in missing valid lines, especially in situations with sparse or fragmented line structures.

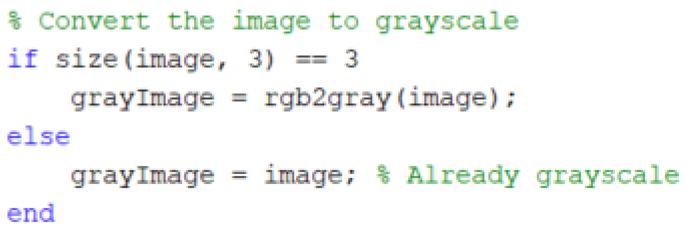
■ ransac_linedetection.m

1: Reading the testimage.
This line reads the input image specified by the file name testimage using the imread function.



```
% Read the testimage
image = imread(testimage);
```

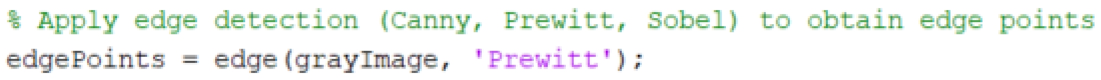
2: Grayscale Conversion:
It checks whether the image is in color (3 channels) and converts it to grayscale using ‘rgb2gray’ if needed. If the image is already grayscale, it assigns the original image to ‘grayImage’.



```
% Convert the image to grayscale
if size(image, 3) == 3
    grayImage = rgb2gray(image);
else
    grayImage = image; % Already grayscale
end
```

3: Edge Detection

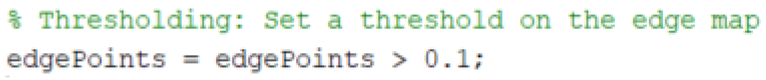
It applies edge detection to the grayscale image using the Prewitt method. The result is a binary image (‘edgePoints’) where edges are marked with 1 and non-edges with 0.



```
% Apply edge detection (Canny, Prewitt, Sobel) to obtain edge points
edgePoints = edge(grayImage, 'Prewitt');
```

4: Thresholding

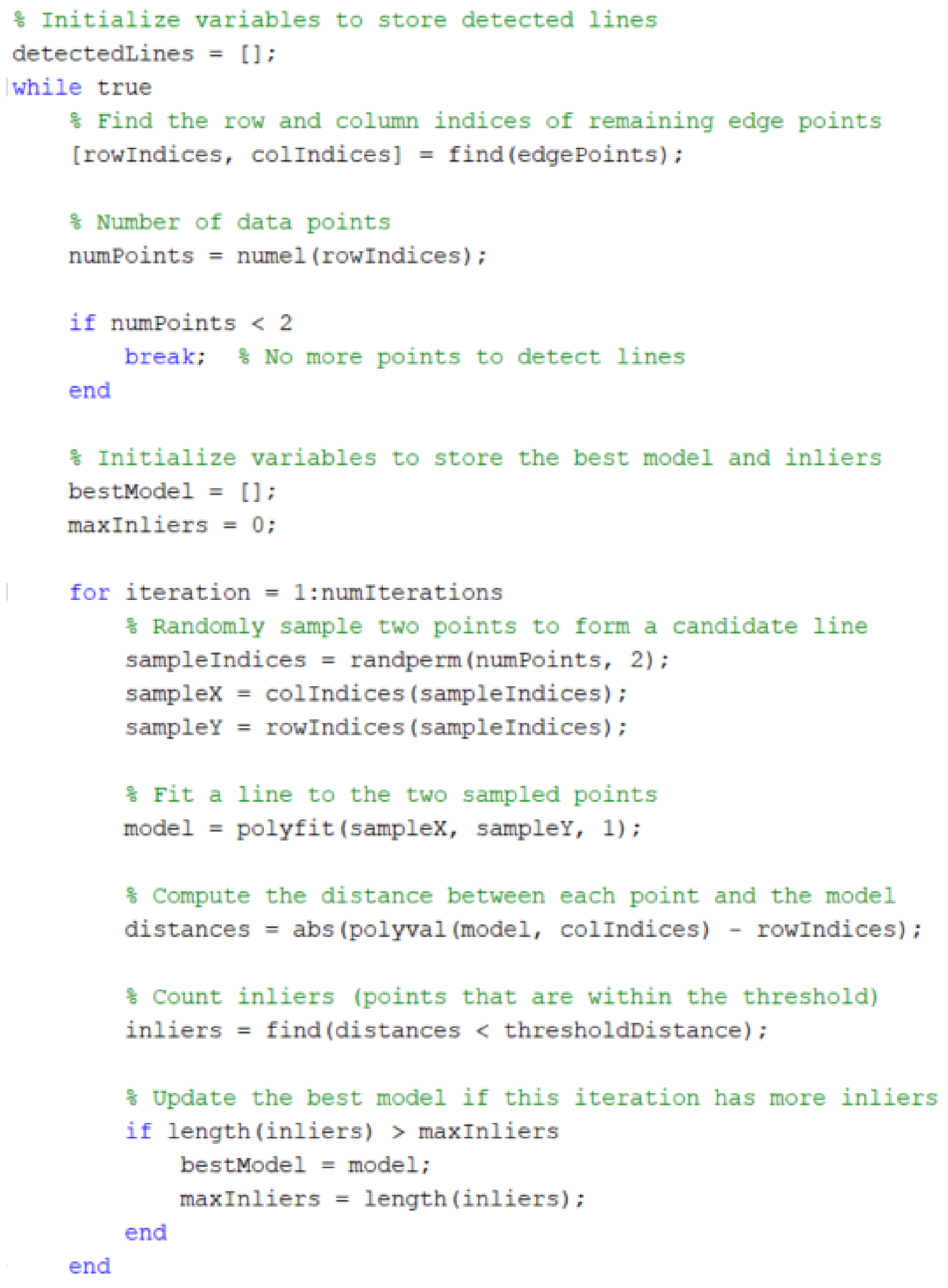
This step thresholds the edge map. It sets pixels with values greater than 0.1 to 1 and the rest to 0. It's a way to clean up the edges and get a more distinct binary map.



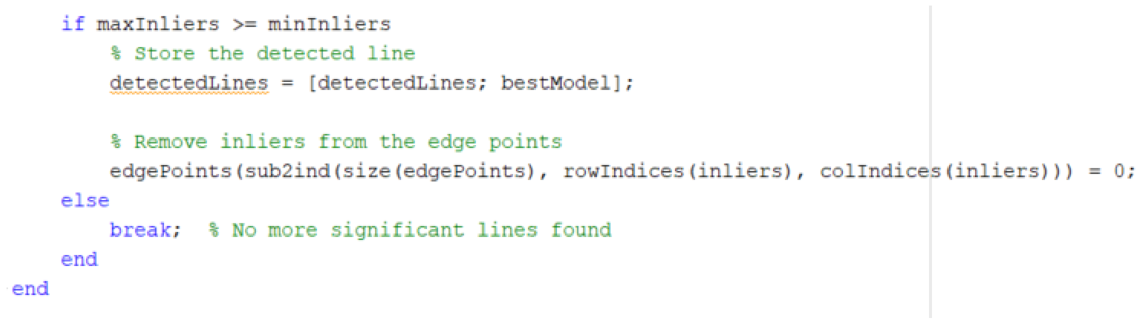
```
% Thresholding: Set a threshold on the edge map
edgePoints = edgePoints > 0.1;
```

5: RANSAC Line Detection

This section implements the RANSAC (Random Sample Consensus) algorithm to detect lines in the edge map. It iteratively fits lines to randomly sampled points, counts inliers (points close to the line), and keeps the best model based on the maximum number of inliers.



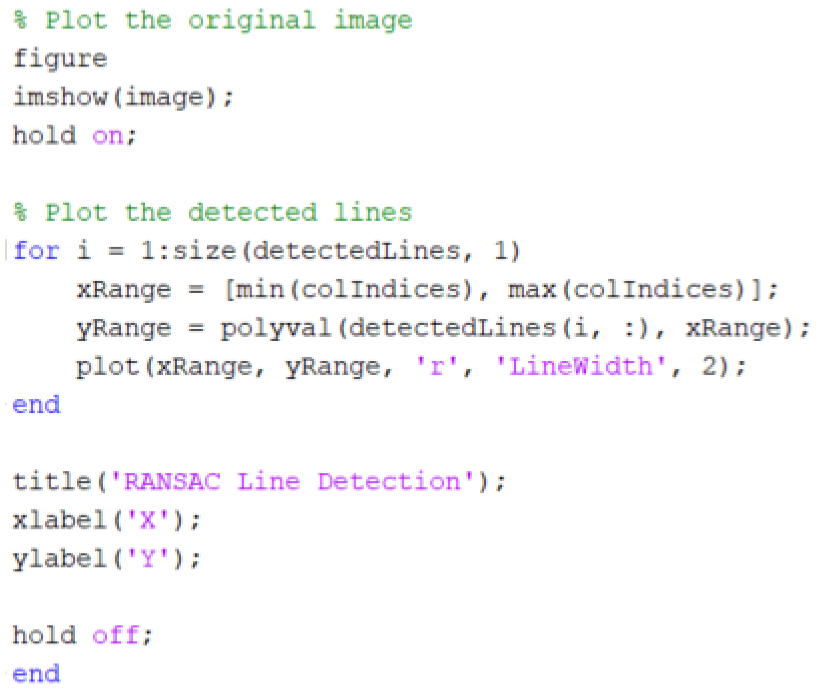
```matlab
% Initialize variables to store detected lines
detectedLines = [];
while true
    % Find the row and column indices of remaining edge points
    [rowIndices, colIndices] = find(edgePoints);

    % Number of data points
    numPoints = numel(rowIndices);

    if numPoints < 2
        break;  % No more points to detect lines
    end

    % Initialize variables to store the best model and inliers
    bestModel = [];
    maxInliers = 0;

    for iteration = 1:numIterations
        % Randomly sample two points to form a candidate line
        sampleIndices = randperm(numPoints, 2);
        sampleX = colIndices(sampleIndices);
        sampleY = rowIndices(sampleIndices);

        % Fit a line to the two sampled points
        model = polyfit(sampleX, sampleY, 1);

        % Compute the distance between each point and the model
        distances = abs(polyval(model, colIndices) - rowIndices);

        % Count inliers (points that are within the threshold)
        inliers = find(distances < thresholdDistance);

        % Update the best model if this iteration has more inliers
        if length(inliers) > maxInliers
            bestModel = model;
            maxInliers = length(inliers);
        end
    end
```



```matlab
    if maxInliers >= minInliers
        % Store the detected line
        detectedLines = [detectedLines; bestModel];

        % Remove inliers from the edge points
        edgePoints(sub2ind(size(edgePoints), rowIndices(inliers), colIndices(inliers))) = 0;
    else
        break;  % No more significant lines found
    end
end
```

6: Plotting the Detected Lines

It creates a new figure, overlays the original image, and plots the detected lines on it. Each detected line is represented by a red line.



```
% Plot the original image
figure
imshow(image);
hold on;

% Plot the detected lines
for i = 1:size(detectedLines, 1)
    xRange = [min(colIndices), max(colIndices)];
    yRange = polyval(detectedLines(i, :), xRange);
    plot(xRange, yRange, 'r', 'LineWidth', 2);
end

title('RANSAC Line Detection');
xlabel('X');
ylabel('Y');

hold off;
end
```

4. Test

Making sure that the code is working appropriately, I used the following the picture including multiple lines, which is called testimage.png.

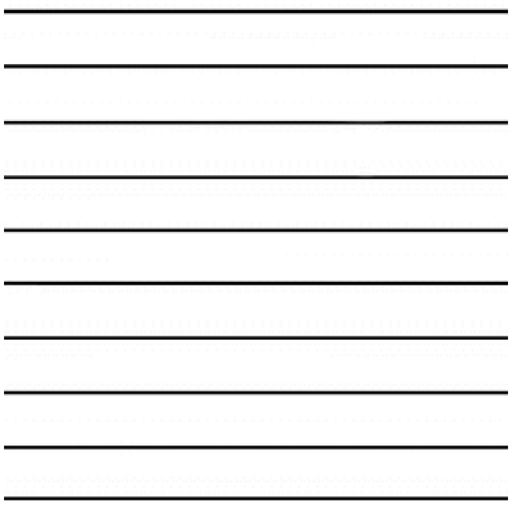


Figure 3: testimage.png

5. Result

The result image of the test is below and the RANSAC parameters are as follows.
For the testimage.png, the parameters are as below.
numIterations = 120
thresholdDistance = 0.1
minInliers:20

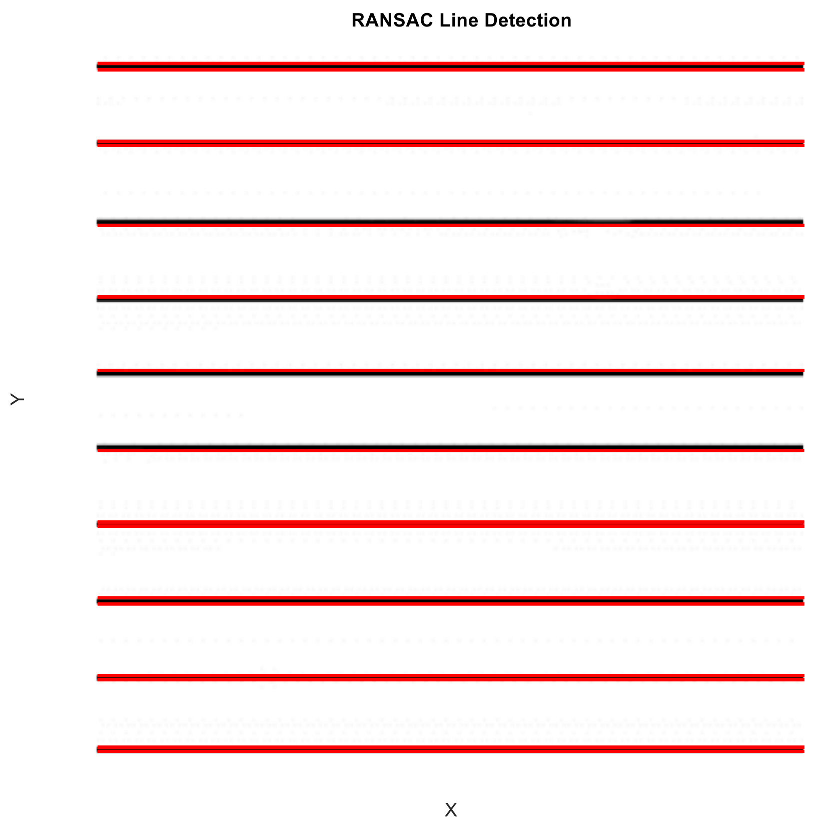


Figure 4: Result of testimage.png

6. Conclusion

The RANSAC algorithm is used for robust line detection. The choice of edge detection method, which is Prewitt in this case, and parameters such as ‘numIterations’, ‘thresholdDistance’ and ‘minInliners’ can be adjusted based on the specific requirements of the application and the characteristics of the images.

7. Bibliography

[1] Wikipedia, "Random sample consensus," Wilipedia, 21 9 2004. [Online]. Available: https://en.wikipedia.org/wiki/Random_sample_consensus. [Accessed 22 11 2023].

[2] K. G. Derpanis, "Overview of the RANSAC Algorithm," York University, 4700 Keele St, Toronto, ON M3J 1P3, Canada, 2010.

[3] V. Sharma, "Robust Outlier Detection with the RANSAC Algorithm in Data Modeling," Medium, 20 4 2023. [Online]. Available: https://levelup.gitconnected.com/robust-outlier-detection-with-the-ransac-algorithm-in-data-modeling-84c5fef92b93. [Accessed 22 11 2023].

[4] C. P. Bathula, "Machine Learning Concept 69: Random Sample Consensus (RANSAC)," Medium, 14 4 2023. [Online]. Available: https://medium.com/@chandu.bathula16/machine-learning-concept-69-random-sample-consensus-ransac-e1ae76e4102a. [Accessed 22 11 2023].