Diagnostic Medical Image Processing (DMIP)
WS 2015/16
Marco Bögel, Room 09.155 marco.boegel@cs.fau.de
Bastian Bier, Room 09.157 bastian.bier@fau.de

LEHRSTUHL FÜR MUSTER-ERKENNUNG

Exercise 4: Random Sample Consensus (RANSAC)

Model estimations on noisy data are usually error-prone. Even a small number of outliers will influence the estimation such that the resulting model can produce large errors. The goal is to identify models that minimize the error and ignore outliers.

1 Theory

RANSAC assumes that a model built with a minimum number of data points does not contain outliers. If we imagine the minimum number of points for a line, the generated line will exactly fit through those two points. To consider every point the model error is evaluated on the whole data set. In a scenario where we expect the majority of the data points to be in a valid range we can use this strategy to find a model fitting the inliers and ignoring the outliers.

RANSAC algorithm

1. Determine the minimum number n_{mdl} of data points required to build the model
2. For n_{it} iterations do
   (a) Choose randomly n_{mdl} points out of your data to estimate the model
   (b) Determine the error of the current model using all data points
3. Choose model with lowest error

We have to consider the probability P_{corr} how likely it is to pick the n_{mdl} right points for an optimal model in n_{it} iterations at least once. Knowing the relative frequency p_o for outliers we can compute the number of required iterations by

$$n_{it} = \left\lceil \frac{log(1 - P_{corr})}{log(1 - (1 - p_o)^{n_{mdl}})} \right\rceil \tag{1}$$

2 Implementation tasks

We will implement a RANSAC algorithm to fit a line through a point plot. Therefore, we assume to obtain sample points on a line where outliers occur due to measurement errors. Complete the gaps in RANSAC.java.

1. Implement the missing parts in the function *fitline(...)*. This function should calculate the parameters for a line through a given the points. This function should return the exact solution if it exists. Otherwise, it should return an approximation.

2. Use this function to calculate a regression line through the given point cloud!

3. Implement the RANSAC algorithm in the function *commonRansac(...)*. It should be implemented in a common way, not only related to line fitting.

4. An error function considering the whole dataset should be implemented. The error should measure, how many points lie within a certain range of the estimated line. Implement this function in *lineError(...)*.

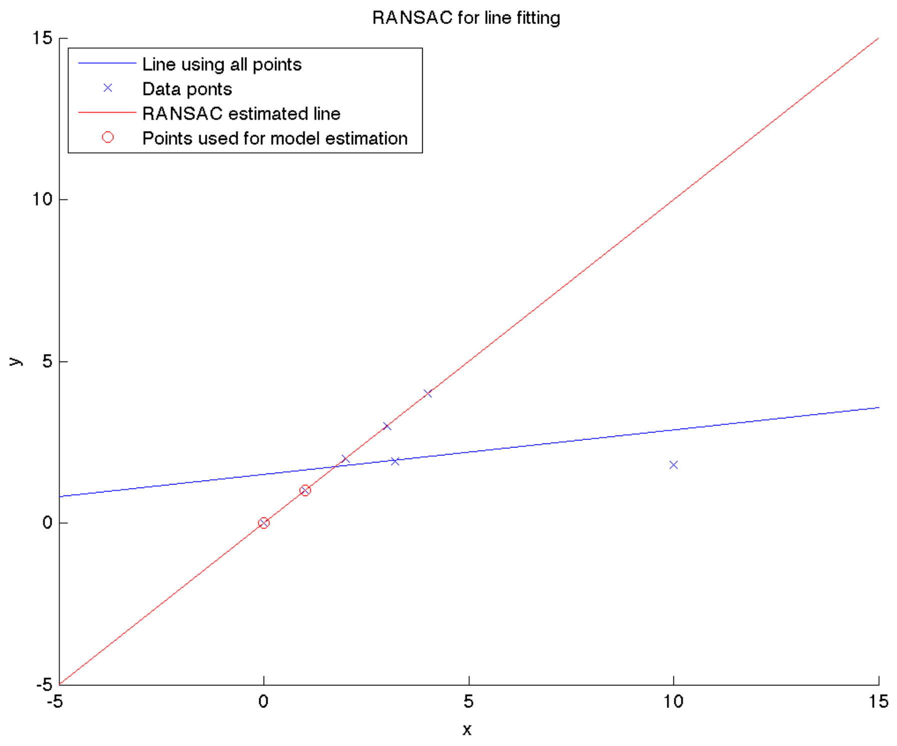

Figure 1: RANSAC for line fitting