RANSAC

From Wikipedia, the free encyclopedia

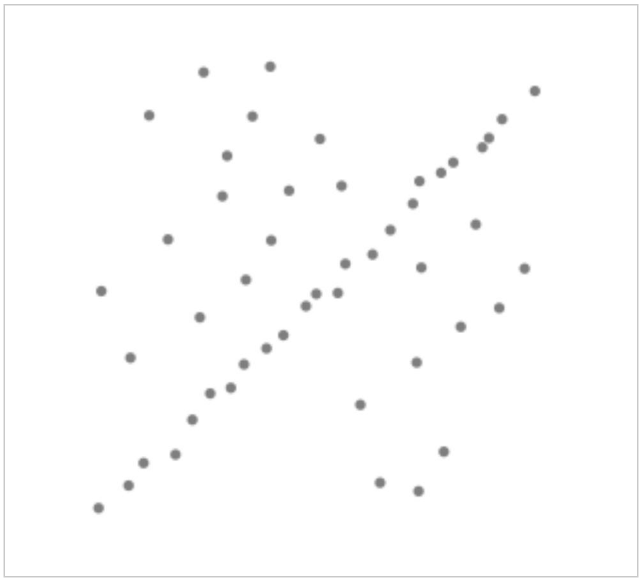
RANSAC is an abbreviation for "RANdom SAmple Consensus". It is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability,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

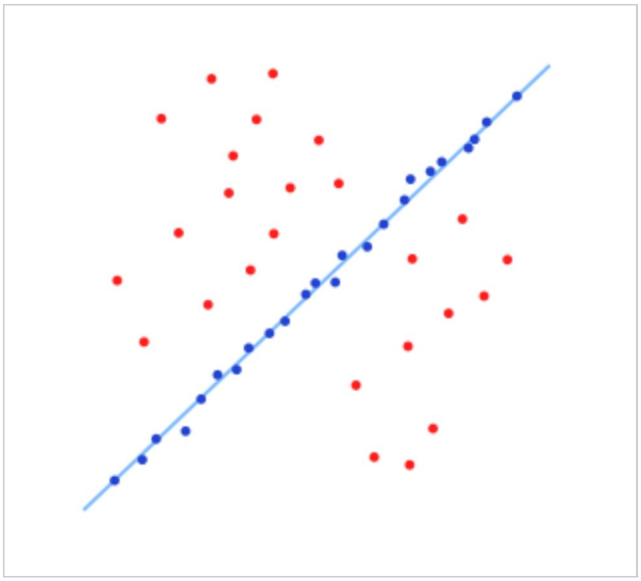
A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. In addition to this, the data can be subject to noise. The outliers can come, e.g., from extreme values of 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.

Contents

Example

A simple example is fitting of a line in two dimensions to a set of observations. Assuming that this set contains both inliers, i.e., points which approximately can be fitted to a line, and outliers, points which cannot be fitted to this line, a simple least squares method for line fitting will in general produce a line with a bad fit to the inliers. The reason is that it is optimally fitted to all points, including the outliers. RANSAC, on the other hand, can produce a model which is only computed from the inliers, provided that the probability of choosing only inliers in the selection of data is sufficiently high. There is no guarantee for this situation, however, and there are a number of algorithm parameters which must be carefully chosen to keep the level of probability reasonably high.


A data set with many outliers for which a line has to be fitted.


Fitted line with RANSAC; outliers have no influence on the result.

Overview

The input to the RANSAC algorithm is a set of observed data values, a parameterized model which can explain or be fitted to the observations, and some confidence parameters.

RANSAC achieves its goal by iteratively selecting a random subset of the original data. These data are *hypothetical inliers* and this hypothesis is then tested as follows:

1. A model is fitted to the hypothetical inliers, i.e. all free parameters of the model are reconstructed from the inliers.
2. All other data are then tested against the fitted model and, if a point fits well to the estimated model, also considered as a hypothetical inlier.
3. The estimated model is reasonably good if sufficiently many points have been classified as hypothetical inliers.
4. The model is reestimated from all hypothetical inliers, because it has only been estimated from the initial set of hypothetical inliers.
5. Finally, the model is evaluated by estimating the error of the inliers relative to the model.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are classified as inliers or a refined model together with a corresponding error measure. In the latter case, we keep the refined model if its error is lower than the last saved model.

The algorithm

The generic RANSAC algorithm, in pseudocode, works as follows:

```
input:
    data - a set of observations
    model - a model that can be fitted to data
    n - the minimum number of data required to fit the model
    k - the number of iterations performed by the algorithm
    t - a threshold value for determining when a datum fits a model
    d - the number of close data values required to assert that a model fits well to data
output:
    best_model - model parameters which best fit the data (or nil if no good model is found)
    best_consensus_set - data points from which this model has been estimated
    best_error - the error of this model relative to the data

iterations := 0
best_model := nil
best_consensus_set := nil
best_error := infinity
while iterations < k
    maybe_inliers := n randomly selected values from data
    maybe_model := model parameters fitted to maybe_inliers
    consensus_set := maybe_inliers

    for every point in data not in maybe_inliers
        if point fits maybe_model with an error smaller than t
            add point to consensus_set

    if the number of elements in consensus_set is > d
        (this implies that we may have found a good model,
        now test how good it is)
        this_model := model parameters fitted to all points in consensus_set
        this_error := a measure of how well this_model fits these points
        if this_error < best_error
            (we have found a model which is better than any of the previous ones,
            keep it until a better one is found)
            best_model := this_model
            best_consensus_set := consensus_set
            best_error := this_error

    increment iterations

return best_model, best_consensus_set, best_error
```

Possible variants of the RANSAC algorithm include

- Break the main loop if a sufficiently good model has been found, that is, one with sufficiently small error. May save some computation time at the expense of an additional parameter.
- Compute `this_error` directly from `maybe_model` without re-estimating a model from the consensus set. May save some time at the expense of comparing errors related to models which are estimated from a small number of points and therefore more sensitive to noise.

The parameters

The values of parameters *t* and *d* have to be determined from specific requirements related to the application and the data set, possibly based on experimental evaluation. The parameter *k* (the number of iterations), however, can be determined from a theoretical result. Let *p* be the probability that the RANSAC algorithm in some iteration selects only inliers from the input data set when it chooses the *n* points from which the model parameters are estimated. When this happens, the resulting model is likely to be useful so *p* gives the probability that the algorithm produces a useful result. Let *w* be the probability of choosing an inlier each time a single point is selected, that is,

w = number of inliers in data / number of points in data

A common case is that w is not well known beforehand, but some rough value can be given. Assuming that the *n* points needed for estimating a model are selected independently, w^n is the probability that all *n* points are inliers and $1 - w^n$ is the probability that at least one of the *n* points is an outlier, a case which implies that a bad model will be estimated from this point set. That probability to the power of *k* is the probability that the algorithm never selects a set of *n* points which all are inliers and this must be the same as $1 - p$. Consequently,

$$1 - p = (1 - w^n)^k$$

which, after taking the logarithm of both sides, leads to

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

This result assumes that the *n* data points are selected independently, that is, a point which has been selected once is replaced and can be selected again in the same iteration. This is often not a reasonable approach and the derived value for *k* should be taken as an upper limit in the case that the points are selected without replacement. For example, in the case of finding a line which fits the data set illustrated in the above figure, the RANSAC algorithm typically chooses 2 points in each iteration and computes `maybe_model` as the line between the points and it is then critical that the two points are distinct.

To gain additional confidence, the standard deviation or multiples thereof can be added to k. The standard deviation of k is defined as

$$SD(k) = \frac{\sqrt{1-w^n}}{w^n}$$

Advantages and disadvantages

An advantage of RANSAC is its ability to do robust estimation of the model parameters, i.e., it can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. A disadvantage of RANSAC is that there is no upper bound on the time it takes to compute these parameters. 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. 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. The Hough transform is an alternative robust estimation technique that may be useful when more than one model instance is present.

Applications

The RANSAC algorithm is often used in computer vision, e.g., to simultaneously solve the correspondence problem and estimate the fundamental matrix related to a pair of stereo cameras.

References

- Martin A. Fischler and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. of the ACM* **24** (6): 381–395. doi:10.1145/358669.358692 (http://dx.doi.org/10.1145%2F358669.358692) .
- David A. Forsyth and Jean Ponce (2003). *Computer Vision, a modern approach*. Prentice Hall. ISBN 0-13-085198-1.
- Richard Hartley and Andrew Zisserman (2003). *Multiple View Geometry in Computer Vision* (2nd ed.). Cambridge University Press.

- P.H.S. Torr and D.W. Murray (1997). "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix". *International Journal of Computer Vision* **24** (3): 271–300. doi:10.1023/A:1007927408552 (http://dx.doi.org/10.1023%2FA%3A1007927408552) .
- Ondrej Chum (2005). "Two-View Geometry Estimation by Random Sample and Consensus" (http://cmp.felk.cvut.cz/~chum/Teze/Chum-PhD.pdf) . *PhD Thesis*. http://cmp.felk.cvut.cz/~chum/Teze/Chum-PhD.pdf
- Sunglok Choi, Taemin Kim, and Wonpil Yu (2009). "Performance Evaluation of RANSAC Family" (http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.pdf) . *In Proceedings of the British Machine Vision Conference (BMVC)*. http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.pdf.

External links

- RANSAC Toolbox for MATLAB (http://vision.ece.ucsb.edu/~zuliani/Code/Code.html) . A research (and didactic) oriented toolbox to explore the RANSAC algorithm in MATLAB. It is highly configurable and contains the routines to solve a few relevant estimation problems.
- Implementation in C++ (http://www.mrpt.org/RANSAC_C++_examples) as a generic template.
- RANSAC for Dummies (http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf) A simple tutorial with many examples that uses the RANSAC Toolbox for MATLAB.
- 25 Years of RANSAC Workshop (http://cmp.felk.cvut.cz/ransac-cvpr2006/)
- Source code for RANSAC in (http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/#robust) MATLAB
- Ransac.js (http://www.visual-experiments.com/demo/ransac.js/) Javascript implementation with visual representation of the iterations (Example of 2D Line fitting).
- ransac.py (http://www.scipy.org/Cookbook/RANSAC) Python implementation for Scipy/Numpy.

Retrieved from "http://en.wikipedia.org/w/index.php?title=RANSAC&oldid=539188534"
Categories: Geometry in computer vision | Statistical algorithms | Statistical outliers | Robust statistics | SRI International