Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation

Release 1.0

Ziv Yaniv

October 21, 2010

Imaging Science and Information Systems (ISIS) Center, Dept. of Radiology,
Georgetown University Medical Center, Washington, DC, USA
zivy@isis.georgetown.edu

Abstract

The Random Sample Consensus (RANSAC) algorithm for robust parameter value estimation has been applied to a wide variety of parametric entities (e.g. plane, the fundamental matrix). In many implementations the algorithm is tightly integrated with code pertaining to a specific parametric object. In this paper we introduce a generic RANSAC implementation that is independent of the estimated object. Thus, the user is able to ignore outlying data elements potentially found in their input. To illustrate the use of the algorithm we implement the required components for estimating the parameter values of a hyperplane and hypersphere.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3223]

Contents

Input:
data - Data contaminated with outliers (cardinality(data) = n).
s - Number of data elements required per subset.
N - Number of subsets to draw from the data.
τ - Threshold which defines if a data element, d_i, agrees with model $\mathcal{M}$.
Output:
$\mathcal{M}^*$ - Estimate of model, based on largest consensus set.

0. maximalConsensusSet $\leftarrow \varnothing$
1. Iterate N times:
   (a) consensusSet $\leftarrow \varnothing$
   (b) Randomly draw a subset containing s elements and estimate model $\mathcal{M}$.
   (c) For each data element, d_i:
   if Agree$(d_i, \mathcal{M}, \tau)$, consensusSet $\leftarrow d_i$
   (d) if cardinality(maximalConsensusSet) $<$ cardinality(consensusSet),
   maximalConsensusSet $\leftarrow$ consensusSet
2. Estimate model parameters using maximalConsensusSet.

Table 1: RANSAC, basic algorithm.

The RAndom SAmple Consensus (RANSAC) algorithm, originally introduced in [1], is a widely used robust parameter estimation algorithm. Two reasons contributed to its wide adoption, it is simple and it can potentially deal with outlier contamination rates greater than 50%.

In the interest of making this paper self contained we first present the theory behind the Random Sample Consensus algorithm, and identify the elements required for its use in estimating parametric objects. This presentation is primarily based on [4]. We then describe two such parametric objects, the n dimensional (nD) plane and sphere. Following the theoretical discussion we describe our implementation, highlighting the separation between the RANSAC algorithm and the specific estimation task.

1 Theory

1.1 RANSAC

The RANSAC algorithm is based on the observation that if we can draw a small subset from the data that is sufficiently large to estimate the model parameters and which contains no outliers, then all inliers will agree[1] with this model. This is done by randomly drawing data subsets, estimating corresponding models, and assessing how many data elements agree with each model, its consensus set. The maximal consensus set obtained in this manner is assumed to be outlier free and is used as input for a least squares model estimate. Assuming we have no prior knowledge about the data elements we assign equal probability to all subsets (in the statistics literature this is known as simple random sampling). Optionally, a minimal threshold on the size of the largest consensus set can be used to decrease the probability of accepting an erroneous model. Table 1 summarizes the basic algorithm.

While the standard formulation presented in Table 1 is sequential, it is clear that the RANSAC algorithm is a parallel hypothesize and test algorithm. That is, multiple unique hypotheses can be generated and tested independently with only a minimal amount of required coordination, testing if the local consensus set is the

[1] Most often agreement is in the sense that the distance between the model and data is less than a pre-specified threshold.

largest one.

To complete the algorithm's description we need to answer the following questions: a) How do we set the threshold for inclusion into the consensus set? b) What subset size should we choose? and c) How many subsets do we need to draw in order to ensure that one of them is outlier free?

An informed choice of the threshold, τ, requires knowledge about the probability distribution of the inlier distances. In most cases we do not have this information and the threshold is set empirically. For the derivation of the threshold assuming a zero mean Gaussian distribution see [4].

As our goal is to obtain an outlier free subset, we observe that the probability to choose an outlier free subset of size s given n data elements, out of which o are outliers is:

$$\frac{\binom{n-o}{s}}{\binom{n}{s}} = \frac{(n-s)(n-s-1)\dots(n-s-o+1)}{n(n-1)\dots(n-o+1)}$$

From the equation above it is clear that to increase the probability we should choose the minimal possible value for s, which is the minimal number of elements required for model estimation.

Finally we derive the number of subsets required to ensure that one of them is outlier free. As subsets are drawn independently with replacement, ensuring that one subset is outlier free requires that we check all $\binom{n}{s}$ subsets. In most cases this is computationally prohibitive, although if n is small enough or $s \simeq n$ it may still be feasible [5].

To alleviate the computational burden we relax the requirement for an outlier free subset, changing it to a probabilistic requirement. We now require that we obtain an outlier free subset with probability p. Assuming that the probability for an inlier is w the probability for an outlier is $\varepsilon = 1 - w$. Then we need to perform at least N subset selections, where $(1 - w^s)^N = 1 - p$ which gives us:

$$N = \frac{\log(1-p)}{\log(1-w^s)} = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \tag{1}$$

Finally, it should be emphasized that even when we do draw an outlier free minimal subset we may not be able to estimate the model. For a successful estimate an outlier free minimal subset is necessary but not sufficient. Usually, the subset must satisfy additional problem specific constraints (e.g. three non-collinear points to fit a 3D plane).

Having answered these questions we can reformulate the RANSAC algorithm. We now require as input the desired probability, p, for drawing an outlier free subset. In addition we adaptively update the number of drawn subsets, N, when a new subset results in a larger consensus set (higher percentage of inliers). Table 2 summarizes the adaptive sampling algorithm.

Input:
data - Data contaminated with outliers (cardinality(data) = n).
p - Desired probability for drawing an outlier free subset.
s - Minimal number of data elements required to estimate model $\mathcal{M}$.
τ - Threshold which defines if a data element, d_i, agrees with model $\mathcal{M}$.
Output:
$\mathcal{M}^*$ - Estimate of model, based on largest consensus set.

0. maximalConsensusSet $\leftarrow \varnothing$
1. $N \leftarrow \binom{n}{s}$
2. Iterate N times:
   (a) consensusSet $\leftarrow \varnothing$
   (b) Randomly draw a subset containing s elements and estimate model $\mathcal{M}$.
   (c) For each data element, d_i:
   if Agree$(\mathcal{M}, d_i, \tau)$, consensusSet $\leftarrow d_i$
   (d) if cardinality(maximalConsensusSet) $<$ cardinality(consensusSet),
   maximalConsensusSet $\leftarrow$ consensusSet
   $N = \frac{\log(1-p)}{\log(1-w^s)}$
3. Estimate model parameters using maximalConsensusSet.

Table 2: RANSAC, adaptive sampling.

Thus, to use the generic RANSAC algorithm we require:

1. A method for estimating the parametric entity given the minimal number of data elements.
2. A method for least squares parameter estimation.
3. A method that checks if a given data element agrees with the model.

We end this section with a discussion of methods for simple random sampling, drawing subsets with replacement from a uniform distribution.

The first method draws a subset by drawing data elements *with replacement* using a uniform distribution (in MATLAB: `ceil(rand(1,s)*n)`). That is, s values are independently generated using a uniform distribution on the interval $[1,n]$. As the elements are drawn with replacement a data element can be drawn multiple times for a single subset. The number of possible subsets is thus n^s while the number of valid subsets is smaller, $\binom{n}{s}$. This will yield degenerate configurations when the same data element is drawn multiple times, as we are generating minimal subsets. While this is not optimal, it is simple to implement and the requirement that all valid subsets have the same probability ($\frac{1}{n^s}$) is satisfied.

A slight modification that ensures that only valid subsets are generated is to remove the selected data element and to change the uniform distribution such that the probability for choosing the k'th element is $\frac{1}{n-k}$. Thus only valid subsets are generated and they all have equal probability ($\frac{1}{n(n-1)\cdots(n-s+1)}$).

Another method for simple random sampling is to independently generate a random value from a uniform distribution over the interval $[a,b]$ for each of the data elements, then sort the values and take the s data elements that correspond to the highest ones. It is clear that this approach only generates the $\binom{n}{s}$ valid subsets. We now show that they all have the same probability.

Let $x_{i=1\ldots n}$ be our independently identically distributed (iid) random variables. The probability of incorporating element i into the subset is the probability that x_i is greater than $n-s$ of the random variables. As the

variables are generated independently we have:

$$p(x_i > x_{j,1} \cap x_i > x_{j,2} \ldots \cap x_i > x_{j,n-s}) = \Pi_{l=1}^{n-s} p(x_i > x_{j,l}) \quad (2)$$

If $p(x_i > x_j)$ is a constant, then the probability of incorporating element i into the subset (Equation 2) is constant, which means that the probability for every subset is constant and equal.

As our random variables are iid their joint distribution is symmetric about the diagonal, $p(x_i = x_j)$ which gives us:

$$p(x_i > x_j) = \frac{1 - p(x_i = x_j)}{2}$$

We now show that $p(x_i = x_j)$ is a constant using the fact that the variables are iid from a uniform distribution:

$$\begin{aligned} p(x_i = x_j) &= \int_a^b p(x_i = \lambda \cap x_j = \lambda) d\lambda \\ &= \int_a^b p(x_i = \lambda) p(x_j = \lambda) d\lambda \\ &= \int_a^b \frac{1}{(b-a)^2} d\lambda \\ &= \frac{1}{b-a} \end{aligned}$$

While this approach only generates valid subsets with uniform probability it is computationally expensive, due to the sorting phase which is an $O(n \log n)$ operation.

1.2 nD Plane

A nD plane is defined as the set of points that satisfy the following equation:

$$\mathbf{n}^T (\mathbf{p} - \mathbf{a}) = 0$$

where $\mathbf{a}$ is a point on the hyperplane and $\mathbf{n}$ is the plane's normal.

Exact estimate

Each point, $\mathbf{p_i} \in \mathbb{R}^d$ provides us with one linear equation in $(d+1)$ unknowns:

$$[\mathbf{p_i}, -1]^T [\mathbf{n}, \mathbf{n}^T \mathbf{a}] = 0$$

Given d linearly independent points we obtain a matrix whose one dimensional null space gives us the plane normal, $\mathbf{n}$. The point on the plane, $\mathbf{a}$, is arbitrarily chosen from the point set.

For $d = 2$ and $d = 3$ the line/plane normal can be computed constructively:

- $d = 2$, given two points $\mathbf{p}, \mathbf{q}$ the line normal is:

$$\mathbf{n} = [q_y - p_y, p_x - q_x]$$

- $d = 3$, given three points $\mathbf{p}, \mathbf{q}, \mathbf{w}$ the plane normal is given by the cross product:

$$\mathbf{n} = (\mathbf{p} - \mathbf{w}) \times (\mathbf{q} - \mathbf{w})$$

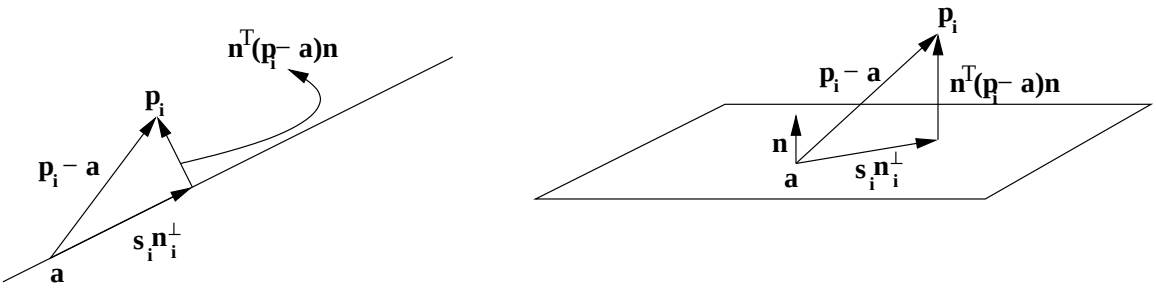

Figure 1: Orthogonal least squares, point-line and point-plane distance.

Geometric Least squares estimate

Given m points in $\mathbb{R}^d$, $m > d$, we want to fit them to a plane such that their orthogonal distance from the plane is minimized.

Given a point $\mathbf{p_i}$ it can be written as follows (Figure 1):

$$\mathbf{p_i} = \mathbf{a} + (\mathbf{n}^T(\mathbf{p_i}-\mathbf{a}))\mathbf{n} + s_i\mathbf{n_i}^\perp$$

where $\mathbf{n_i}^\perp$ is a unit vector perpendicular to $\mathbf{n}$ and s_i is the appropriate scale factor. The signed point-to-plane distance is thus:

$$\delta_i = (\mathbf{p_i}-\mathbf{a})^T\mathbf{n}$$

The optimal plane parameters are computed as:

$$[\mathbf{n}^*, \mathbf{a}^*] = \underset{\mathbf{n},\mathbf{a}}{\operatorname{argmin}} \Sigma_{i=1}^m \delta_i^2$$

and in explicit form:

$$\begin{aligned}\Delta &= \Sigma_{i=1}^m[(\mathbf{p_i}-\mathbf{a})^T\mathbf{n}]^2 \\ &= \Sigma_{i=1}^m((\mathbf{p_i}-\mathbf{a})^T\mathbf{n})(\mathbf{n}^T(\mathbf{p_i}-\mathbf{a})) \\ &= \mathbf{n}^T[\Sigma_{i=1}^m(\mathbf{p_i}-\mathbf{a})(\mathbf{p_i}-\mathbf{a})^T]\mathbf{n}\end{aligned}$$

Deriving Δ with respect to $\mathbf{a}$ we get:

$$\frac{\partial\Delta}{\partial\mathbf{a}} = -2[\mathbf{n}^T\mathbf{n}]\Sigma_{i=1}^m(\mathbf{p_i}-\mathbf{a})$$

equating this to zero yields:

$$\begin{gathered}\Sigma_{i=1}^m(\mathbf{p_i}-\mathbf{a}) = 0 \\ \Downarrow \\ \mathbf{a} = \tfrac{1}{m}\Sigma_{i=1}^m\mathbf{p_i}\end{gathered}$$

$\mathbf{a}$ is the mean of the sample points.

Going back to our optimization problem we have:

$$\min_{\mathbf{n}} \mathbf{n}^T\Sigma\mathbf{n}, \;\; s.t. \|\mathbf{n}\| = 1$$

where $\Sigma = \Sigma_{i=1}^{m}(\mathbf{p_i}-\mathbf{a})(\mathbf{p_i}-\mathbf{a})^T$.

The minimum is obtained for the eigenvector[2] corresponding to the minimal eigenvalue of Σ [3].

1.3 nD sphere

A nD sphere is defined as the set of points that satisfy the following equation:

$$(\mathbf{p}-\mathbf{c})^T(\mathbf{p}-\mathbf{c}) = r^2 \tag{3}$$

where $\mathbf{c}$ is the center and r is the radius.

Exact estimate

As all the points, $\mathbf{p_i} \in \mathbb{R}^d$, are on the sphere we have:

$$\forall i,j \quad (\mathbf{p_i}-\mathbf{c})^T(\mathbf{p_i}-\mathbf{c}) = (\mathbf{p_j}-\mathbf{c})^T(\mathbf{p_j}-\mathbf{c}) = r^2$$

Each pair of points provides us with one linear equation in $(d+1)$ unknowns:

$$(\mathbf{p_i}-\mathbf{p_j})^T\mathbf{c} = 0.5(\mathbf{p_i}^T\mathbf{p_i}-\mathbf{p_j}^T\mathbf{p_j})$$

Given $d+1$ linearly independent points we obtain a regular matrix, and solve the equation system to get $\mathbf{c}$. The radius is computed as $\|\mathbf{p_i}-\mathbf{c}\|$, where $\mathbf{p_i}$ is arbitrarily chosen from the point set.

For $d=2$ and $d=3$ we can use Cramer's rule to compute the matrix inverse[3]:

- $d=2$, given three points $\mathbf{p_1},\mathbf{p_2},\mathbf{p_3}$ the equation system is:

$$\begin{bmatrix} (p_{1x}-p_{2x}) & (p_{1y}-p_{2y}) \\ (p_{1x}-p_{3x}) & (p_{1y}-p_{3y}) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} = 0.5 \begin{bmatrix} (p_{1x}^2-p_{2x}^2)+(p_{1y}^2-p_{2y}^2) \\ (p_{1x}^2-p_{3x}^2)+(p_{1y}^2-p_{3y}^2) \end{bmatrix}$$

  and the matrix inverse is given as:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

- $d=3$, given four points $\mathbf{p1},\mathbf{p2},\mathbf{p3},\mathbf{p4}$ the equation system is:

$$\begin{bmatrix} (p_{1x}-p_{2x}) & (p_{1y}-p_{2y}) & (p_{1z}-p_{2z}) \\ (p_{1x}-p_{3x}) & (p_{1y}-p_{3y}) & (p_{1z}-p_{3z}) \\ (p_{1x}-p_{4x}) & (p_{1y}-p_{4y}) & (p_{1z}-p_{4z}) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = 0.5 \begin{bmatrix} (p_{1x}^2-p_{2x}^2)+(p_{1y}^2-p_{2y}^2)+(p_{1z}^2-p_{2z}^2) \\ (p_{1x}^2-p_{3x}^2)+(p_{1y}^2-p_{3y}^2)+(p_{1z}^2-p_{3z}^2) \\ (p_{1x}^2-p_{4x}^2)+(p_{1y}^2-p_{4y}^2)+(p_{1z}^2-p_{4z}^2) \end{bmatrix}$$

  and the matrix inverse is given as:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}, \quad A^{-1} = \frac{1}{a(ei-fh)-b(di-fg)+c(dh-eg)}\begin{bmatrix} ei-fh & ch-bi & bf-ce \\ fg-di & ai-cg & cd-af \\ dh-eg & bg-ah & ae-bd \end{bmatrix}$$

[2] Eigenvector computation is performed either using methods that take advantage of the fact that the matrix is symmetric, or in the case of $d=2$, using an analytic solution.

[3] $A^{-1} = \frac{C^T}{det(A)}$ where C is the cofactor matrix, $C_{ij} = -1^{i+j}det(M_{ij})$.

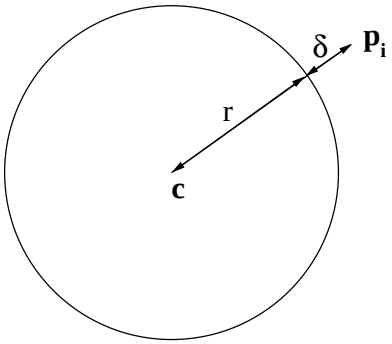

Figure 2: Orthogonal least squares, point-sphere distance.

Algebraic least squares estimate

The derivations described in this subsection are partially due to [2].

Given m points in $\mathbb{R}^d$, $m > (d+1)$, we want to fit them to a sphere such that the sum of the squared algebraic distances is minimized. From the definition of a sphere (Equation 3) we have:

$$\delta_i = \mathbf{p_i}^T \mathbf{p_i} - 2\mathbf{p_i}^T \mathbf{c} + \mathbf{c}^T \mathbf{c} - r^2$$

The optimal sphere parameters are computed as:

$$[\mathbf{c}^*, r^*] = \underset{\mathbf{c},r}{\operatorname{argmin}} \Sigma_{i=1}^m \delta_i^2$$

setting $m = \mathbf{c}^T \mathbf{c} - r^2$ we get the following linear equation system ($Ax = b$):

$$\begin{bmatrix} -2\mathbf{p_1}^T & 1 \\ \vdots & \vdots \\ -2\mathbf{p_m}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ m \end{bmatrix} = \begin{bmatrix} -\mathbf{p_1}^T \mathbf{p_1} \\ \vdots \\ -\mathbf{p_m}^T \mathbf{p_m} \end{bmatrix} \tag{4}$$

The solution of this equation system minimizes $\Sigma_{i=1}^m \delta_i^2 = \|Ax - b\|^2$.

Note that this equation system admits solutions where $m \geq \mathbf{c}^T \mathbf{c}$. That is, we have a solution that does not represent a valid circle, as $r^2 <= 0$. This situation can arise in the presence of outliers.

Geometric least squares estimate

Given m points in $\mathbb{R}^d$, $m > (d+1)$, estimate the parameters of the sphere that minimizes the sum of squared geometric distances to the point set.

The signed geometric distance is (Figure 2):

$$\delta_i = \|\mathbf{p_i} - \mathbf{c}\| - r$$

The optimal sphere parameters are computed as:

$$[\mathbf{c}^*, r^*] = \underset{\mathbf{c},r}{\operatorname{argmin}} \Sigma_{i=1}^m \delta_i^2 = \underset{\mathbf{c},r}{\operatorname{argmin}} \Sigma_{i=1}^m (\sqrt{(\mathbf{p_i} - \mathbf{c})^T (\mathbf{p_i} - \mathbf{c})} - r)^2$$

This nonlinear optimization problem is solved using the Levenberg-Marquardt method which requires the computation of δ_i and its partial derivatives with respect to the $(d+1)$ unknowns:

$$\begin{aligned} \frac{\partial \delta_i}{\partial c_i} &= \frac{c_i - p_i}{\sqrt{(\mathbf{p_i} - \mathbf{c})^T (\mathbf{p_i} - \mathbf{c})}} \\ \frac{\partial \delta_i}{\partial r} &= -1 \end{aligned}$$

RANSAC Caveat

When there is no prior knowledge with regard to the sphere location or its radius the RANSAC algorithm will often yield incorrect estimates. A key assumption of the algorithm is that the maximal consensus set is outlier free. In the case of spheres, the size of the consensus set is related to the sphere radius, causing bias. That is, for a random data set a larger radius will most often lead to a larger consensus set (the surface area of the nD sphere is proportional to r^{n-1}). This can be mitigated by incorporating prior constraints on the parameter values. An implicit approach is to reject hypotheses that invalidate the constraints, similar to the way that we deal with degenerate configurations.

2 Implementation

We next address several implementation related issues:

1. *What happens if the data is outlier free?*

   In the basic algorithm we continue to iterate even though the algorithm could have terminated immediately with all the data as the consensus set. In the adaptive sampling version we need to explicitly address this situation, as the denominator in Table 2 step 2(d), $(\log(1-w^s))$, is not defined. When encountering this situation we terminate the iterations without attempting to compute the undefined value.

2. *What happens when the selected subset is a degenerate configuration, that is, a unique model cannot be estimated (e.g. fitting a homography to four points where three/four of them are collinear)?*

   We can either accept this subset as a valid draw whose consensus set is empty, or we can require that the sample be drawn again and again till we obtain a non-degenerate configuration. This approach requires a limit on the number of attempts, as we risk the chance of iterating infinitely. This can happen if the data does not contain an instance of the model (fitting a circle to line data). It is also possible that we repeatedly draw the same degenerate configuration as we are drawing with replacement and all subsets have equal probability. In our implementation we consider this a valid subset with an empty consensus set.

3. *What happens if there are multiple consensus sets with the maximal cardinality?*

   One possibility is to simply ignore this scenario and arbitrarily accept the consensus set that was obtained first. This is what we do. Another possible option is to keep all of the sets, estimate the least squares model for each of them and select the one that has the minimal residual error. Again, if multiple models result in the same residual error we can arbitrarily choose one.

4. *Can we improve the algorithm's running time?*

   The most computationally intensive part of the RANSAC algorithm is the test with regard to inclusion into the consensus set, an $O(n)$ operation. We can reduce the constant associated with the test in two ways, early termination and keeping track of the subset selections.

   Testing for inclusion into the consensus set can be terminated if the new consensus set cannot be larger than the current largest consensus set. Stop the iterations if all data elements have been tested or if the number of remaining objects is smaller than the difference between the size of the consensus set and the current largest consensus set.

   As all subsets have equal probability, multiple draws can yield the same subset. If we keep track of the subsets already drawn we can avoid assessing the same subset multiple times.

   Finally we can separate the hypothesis generation and testing into multiple concurrent threads, each thread independently executing step 2 from Table 2.

5. *How do we implement the algorithm in a generic manner?*

   The use of templates enables us to implement the RANSAC algorithm in a generic manner that is oblivious to the type of data elements required by the estimation task at hand.

```
              //Data objects required for estimate are of type T and the parameters, S,
              //are expected to be float or double
template<class T, class S>
class RANSAC : public Object {

  void SetNumberOfThreads( unsigned int numberOfThreads );
  unsigned int GetNumberOfThreads();

    //set the object that actually does the estimation
  void SetParametersEstimator( typename ParametersEstimator<T,S>::Pointer paramEstimator );

  void SetData( std::vector<T> &data );
  double Compute( std::vector<S> &parameters,
               double desiredProbabilityForNoOutliers );
};
```

   The definition of a common interface, abstract superclass, for all estimation tasks completes the required infrastructure. This interface ensures that the required methods identified in section 1.1 are implemented.

```
template<class T, class S>
class ParametersEstimator : public Object
{
        //estimate using minimal number of data objects
  virtual void Estimate( std::vector<T *> &data, std::vector<S> &parameters ) = 0;
  virtual void Estimate( std::vector<T> &data, std::vector<S> &parameters ) = 0;

  virtual void LeastSquaresEstimate( std::vector<T *> &data, std::vector<S> &parameters ) = 0;
  virtual void LeastSquaresEstimate( std::vector<T> &data, std::vector<S> &parameters ) = 0;

  virtual bool Agree( std::vector<S> &parameters, T &data ) = 0;
              //number of data objects required by Estimate()
  void SetMinimalForEstimate( unsigned int minForEstimate );
  unsigned int GetMinimalForEstimate();
};
```

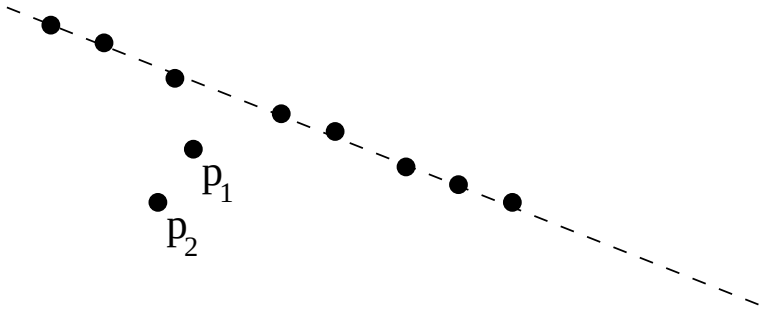

Figure 3: When estimating a line we randomly draw pairs of points. Given the pair $\mathbf{p_1}, \mathbf{p_2}$ the consensus set includes only these two points (for a specific value of τ). The number of trials required to obtain an outlier free subset with a probability of 0.99 is then computed as $N = \frac{\log(0.01)}{\log(0.96)} \simeq 113$, while the number of all possible subsets for this data set is $\binom{10}{2} = 45$.

6. What should the initial number of iterations be? Setting this number to 1 or any magic number, $k < n, \binom{n}{s}$ can potentially cause the algorithm to fail. As we do not know the number of outlying data elements it is possible that the first k subsets all contain outliers. Thus we set the initial number of iterations to $\binom{n}{s}$. The only issue with this choice is the potential for data overflow during the computation which is addressed in our implementation.

Finally, we note an interesting aspect of the probabilistic approach to computing the required number of subsets. It is possible that the number of subsets computed in this manner is greater than all possible subsets. This is a result of using the proportion of inliers/outliers and not their nominal numbers. Figure 3 shows an example of how this can happen.

Algebraic Least Squares Sphere Estimate: The validity of the solution of Equation 4 is checked and rejected if the solution is invalid. That is, solutions in which $r^2 <= 0$ which potentially arise in the presence of outliers.

Geometric Least Squares Sphere Estimate: The use of the iterative nonlinear solver requires an initial estimate, this is obtained using the analytic algebraic least squares method.

2.1 Code Examples

We can use the parameter estimators on their own:

```
                        //create and initialize the sphere parameter estimator
double maximalDistanceFromSphere = 0.5;
SphereEstimatorType::Pointer sphereEstimator = SphereEstimatorType::New();
sphereEstimator->SetDelta( maximalDistanceFromSphere );
                    //least squares is either GEOMETRIC or ALGEBRAIC, default is GEOMETRIC
                    //but in this example we set it explicitly
sphereEstimator->SetLeastSquaresType( itk::SphereParametersEstimator<DIMENSION>::GEOMETRIC );
sphereEstimator->LeastSquaresEstimate( data, sphereParameters );
           //always check if successful
if( sphereParameters.empty() )
  std::cout<<"Least squares estimate failed, degenerate configuration?\n";
else {
  do something useful...
}
```

Or we can use the RANSAC algorithm:

```
                        //create and initialize the parameter estimator
double maximalDistanceFromPlane = 0.5;
PlaneEstimatorType::Pointer planeEstimator = PlaneEstimatorType::New();
planeEstimator->SetDelta( maximalDistanceFromPlane );

                          //create and initialize the RANSAC algorithm
double desiredProbabilityForNoOutliers = 0.999;
double percentageOfDataUsed;
RANSACType::Pointer ransacEstimator = RANSACType::New();
ransacEstimator->SetData( data );
ransacEstimator->SetParametersEstimator( planeEstimator.GetPointer() );
percentageOfDataUsed =
  ransacEstimator->Compute( planeParameters, desiredProbabilityForNoOutliers );

       //always check if we obtained an estimate
if( planeParameters.empty() )
  std::cout<<"RANSAC estimate failed, degenerate configuration?\n";
else
{
 do something useful...
}
```

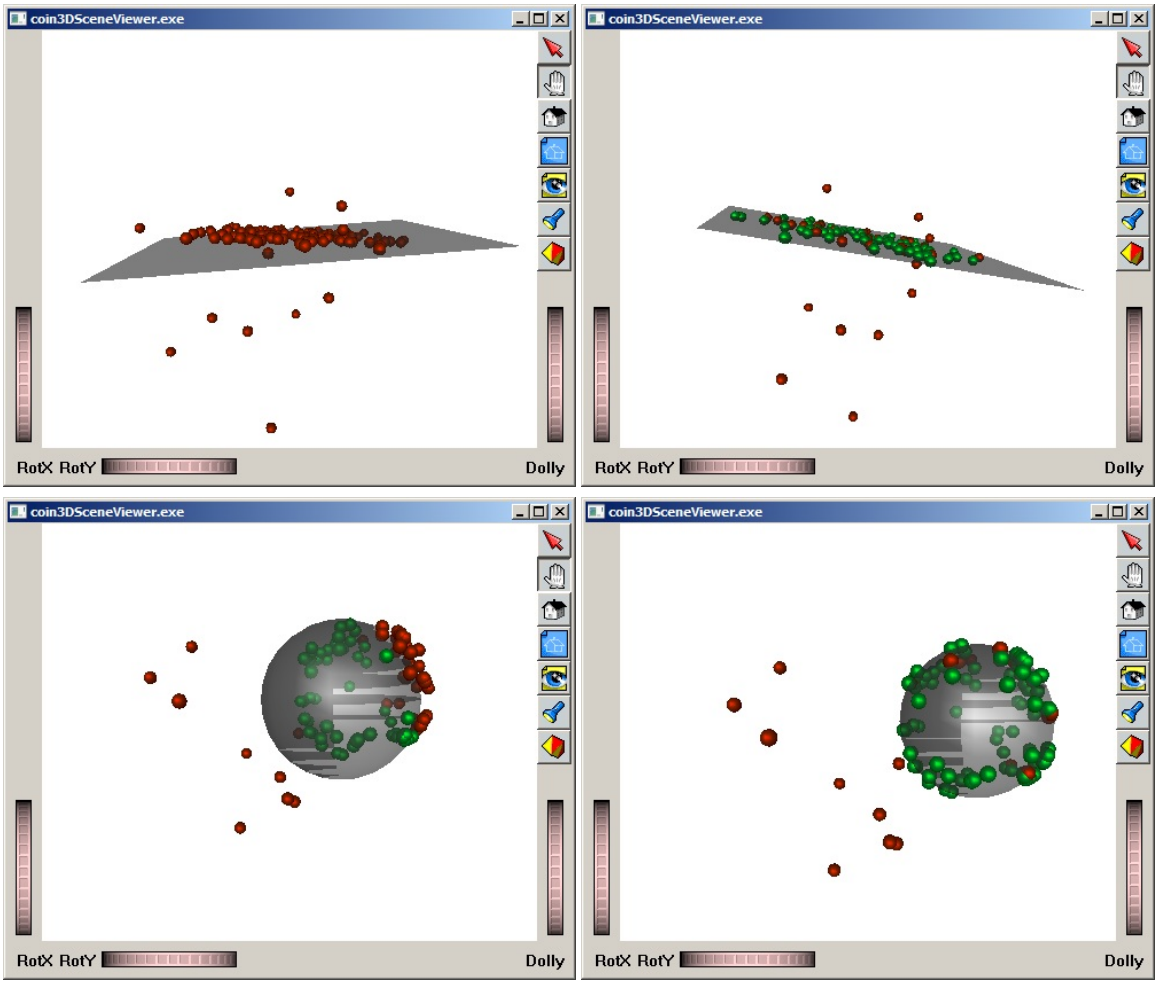

Figure 4: Least squares (left column) and RANSAC (right column) based estimates of a plane and sphere. Green spheres are the data points that agree with the models. Quantitative comparison of these data sets is given in Table 3.

Visual comparison of results obtained using a least squares estimate versus those obtained using our RANSAC implementation are shown in Figure 4, with the corresponding quantitative comparison given in Table 3.

Plane	parameter values	dot with known normal, point known plane distance
Ground Truth	[0.654, 0.672, 0.346, 502.241, 564.592, -207.497]	1, 0
Least Squares	[0.573, 0.757, 0.310, 354.318, 381.459, 274.547]	0.992, 52.961
RANSAC	[0.654, 0.672, 0.346518, 364.896, 425.513, 321.966]	1, 0.005

Sphere	parameter values	distance to known center, radius difference
Ground Truth	[798.387, 497.428, 164.981, 515.132]	0, 0
Least Squares	[628.500, 433.788, 158.136, 609.403]	181.544, 94.272
RANSAC	[798.242, 497.395, 165.015, 515.158]	0.152, 0.026

Table 3: Quantitative comparison between the least squares and RANSAC estimates of the data shown in Figure 4. As a plane does not have a unique, point-normal, representation we compare the dot product of the known and estimated normals (a good estimate should be close to +1 or -1). We also look at the point to plane distance between the estimated point and the known plane. For the sphere we look at the distance between the estimated and known center and the difference between the two radii. All measurements are in metric units (mm,m,...) except the dot product.

References

[1] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. (document)

[2] W. Gander, G. H. Golub, and R. Strebel. Least-squares fitting of circles and ellipses. *BIT*, 34(4):558–578, 1994. 1.3

[3] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996. 1.2

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2001. (document), 1.1

[5] Z. Yaniv and L. Joskowicz. Precise robot-assisted guide positioning for distal locking of intramedullary nails. *IEEE Trans. on Medical Imaging*, 24(5):624–635, 2005. 1.1