DIRSAC: A Directed Sampling And Consensus Approach to Quasi-Degenerate Data Fitting

Chris L Baker
National Robotics Engineering Center
Pittsburgh, PA
clbaker@rec.ri.cmu.edu

William Hoff
Colorado School of Mines
Golden, CO
whoff@mines.edu

Abstract

In this paper we propose a new data fitting method which, similar to RANSAC, fits data to a model using sample and consensus. The application of interest is fitting 3D point clouds to a prior geometric model. Where the RANSAC process uses random samples of points in the fitting trials, we propose a novel method which directs the sampling by ordering the points according to their contribution to the solution's constraints. This is particularly important when the data is quasi-degenerate. In this case, the standard RANSAC algorithm often fails to find the correct solution. Our approach selects points based on a Mutual Information criterion, which allows us to avoid redundant points that result in degenerate sample sets. We demonstrate our approach on simulated and real data and show that in the case of quasi-degenerate data, the proposed algorithm significantly outperforms RANSAC.

1. Introduction

One challenge for real world robotics applications is the estimation of the robot's location and orientation (pose) in a local environment. For any robotic system to perform meaningful tasks, accurate localization information is crucial. Recent availability of relatively low cost 3D range sensors have enabled the use of range information to estimate the platform's local pose. One common approach fits point clouds to a prior model using an Iterative Closest Point (ICP) algorithm [2]. This approach can suffer when the data is contaminated by outliers. Traditional statistical approaches have been used to improve robustness to this noisy data. For example, a RANdom Sampling And Consensus (RANSAC) algorithm has been demonstrated to provide good results in noisy environments. RANSAC has been used in many contexts, but was perhaps first introduced in [6].

It is common for the available data to be in a degenerate, or quasi-degenerate state. By "degenerate" we mean that some of the solution dimensions are not constrained by the data. By "quasi-degenerate" we mean the case where the constraining data is indeed present, but is a small percentage of the full data available, such that it would likely be omitted if a random subset were chosen to compute a fit.

Consider the pathological case of a planar surface. The constraints are along the direction orthogonal to the surface, and the orientation about the axes contained within the planar surface. Unconstrained are the rotation about the plane's normal and the position along the dimensions parallel to the plane. Consider a small non-planar feature in the sensor's field of regard positioned on the planar surface. The current naïve RANSAC algorithm would be overwhelmed by the preponderance of planar data and would likely not consider the relatively small but critical constraints of the non-planar data because this critical data would be considered outlier data. Thus, the algorithm may never select a sample set that includes the critical non-planar points (or, may take a long time to find it). The resulting solution could be a completely erroneous solution even though it may have a low residual error.

The problem is that the naïve RANSAC algorithm is very likely to pick sample sets that are degenerate. Some of the points in such sets are redundant, in the sense that they don't add any information in terms of constraining the solution. While there are methods to detect the degeneracy of such sample sets (as described in Section 2), it would be better to avoid selecting redundant points in the first place. If redundant points could be avoided, then the algorithm would not waste time analyzing degenerate sample sets. Thus, fewer iterations would be needed, resulting in a faster, more robust algorithm.

This paper proposes an algorithm (Section 3) that selects points in such a way as to avoid degenerate sample sets. Instead of a purely random point selection, we direct the selection of points to avoid redundancy with already selected points. The algorithm, called DIRSAC, directs point selection within a sample and consensus framework. The main idea is that we evaluate each point based on its ability to constrain the pose of the solution. We can then detect redundant points by computing the mutual information between them. Experimental results (Section 4) show significant improvements over the naïve RANSAC algorithm.

2. Previous Work

The RANSAC algorithm is a well-established approach in the Computer Vision community. The naïve RANSAC algorithm first randomly selects a minimum subset of points to compute a fit. The Iterative Closest Point (ICP) algorithm is used to compute the point correspondences to the model, and the pose of the model in the sensor's reference frame. Once computed, the remaining data is used to evaluate the fit. In addition to the residual error, a set of inliers is identified as points that fall within some error threshold. This process repeats until a reasonable probability exists that a good solution has been computed. The list of plausible solutions are searched in order of increasing error until a solution with enough inliers has been found.

Significant effort has been made to improve the runtime speed of RANSAC while still guaranteeing a correct solution with some probability. A good review of current approaches to RANSAC improvements is provided in [9]. Some approaches attempt to optimize the model verification step. For example, Matas and Chum [8] have designed a pre-evaluation stage which attempts to quickly filter out bad hypotheses. The concept of early-termination was also extended by Capel [3]. Another strategy is to bias the point selection such that points with a higher inlier probability will be treated preferentially in the random point selection process. For example [4] orders the points based on image feature matching scores, using the assumption that a higher similarity in the feature match will increase the likelihood of the point match being an inlier. Once the points are sorted, they perform RANSAC over a PROgressively larger subset from the top of this sorted set of points (PROSAC).

The number of iterations required for the RANSAC algorithm depends on the frequency of inliers. If λ is the fraction of inliers, then the number of iterations S to find a good solution (*i.e.*, a sample set with no outliers) with probability η must be at least

$$S = \frac{\log(1-\eta)}{\log(1-\lambda^m)} \tag{1}$$

where m is the size of each sample.

In the case of quasi-degenerate data, a sample set must be found that not only contains all inliers, but also is not degenerate. For this, the required number of RANSAC iterations can be much larger as Frahm *et al.* described in detail in Section 3 from [7]. Their approach to solve this problem assumes that the model estimation problem can be expressed as a linear relationship $Ax = 0$, where x is the vector of parameters to be fit, and A is a matrix containing the data from a sample set of matches. They describe an algorithm which can detect degeneracy in A, and are able to estimate the actual number of degrees of freedom in the data. However, they evaluate the sample set after the points have been selected to determine if the selected points are in a degenerate configuration. This approach requires searching through the model constraint space in a non-trivial way which is costly. We would like to avoid degenerate configurations altogether by choosing points in such a way that they will provide us with non-degenerate solutions. Furthermore, the assumption of a linear data fitting relationship is not applicable to our problem of finding a rigid body transformation using ICP.

Although not specifically developed for RANSAC, in [5], Davison develops an algorithm to select points based on the Mutual Information (MI) constraint. He then uses this to determine which point, in a feature tracking context, would provide the most information to the state if used as an observation. As described in the next section, our approach extends this work and applies it to the current context of matching 3D data with a prior model. Using this, we are able to sort the 3D points by their added information and direct our sampling.

3. Method and Approach

We would like to direct the point selection based on the likelihood of a point constraining the fit. In order to do this, we need to compute the effect a point, or a set of points, will have on the uncertainty of the pose. In this development we assume that the only uncertainty is in the range measurement of the sensor. There is additional sensor uncertainty in the direction to the range measurement, but it will be negligible in comparison. We begin by developing the relationship between the sensor's pose and a range measurement.

3.1. Range Covariance from Sensor Covariance

We represent the pose of the sensor in the world as a vector $\mathbf{x}$ of pose elements $(x, y, z, \alpha_r, \alpha_p, \alpha_y)$, where (x, y, z) is the position of the sensor in the world, and $(\alpha_r, \alpha_p, \alpha_y)$ represents the orientation of the sensor in the world ($roll$, $pitch$, yaw). With this, we can predict the range measurement for the i^{th} point using the function $\rho_i = \mathsf{h}_i(\mathbf{x})$. For small perturbation of the pose $\mathbf{x}$, we can write the resulting perturbation of ρ_i. After a Taylor's series expansion and linearization we have the following for the i^{th} point:

$$\Delta\rho_i = \frac{\partial \mathsf{h}_i}{\partial \mathsf{x}} \Delta\mathbf{x} = \mathsf{J}_i \Delta\mathbf{x} \tag{2}$$

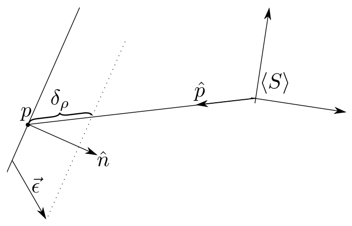

Figure 1. The components of the elements of the Jacobian for translation. The sensor coordinate frame $\langle S\rangle$ has a point p which is in the direction of the unit vector $\hat{p}$. $\hat{n}$ is the normal to a plane representing the local model's surface. Perturbing the plane by $\vec{\epsilon}$ will change the range to the sensor by δ_ρ.

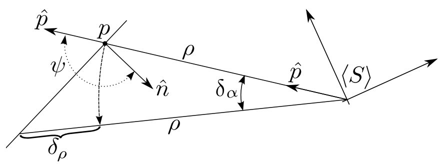

Figure 2. The components for the rotational elements of the Jacobian. The sensor coordinate frame $\langle S\rangle$ has a point p in the direction of the unit vector $\hat{p}$. The normal to the local model surface is $\hat{n}$. The angle between the plane's normal and the point direction $\hat{p}$ is ψ. When the orientation of the sensor is perturbed by some small angle δ_α, the change in the range to the model's surface is given by δ_ρ.

where J_i is the Jacobian which describes how small changes in the elements of the pose will effect the i^{th} range measurement ρ_i, or more specifically,

$$\mathsf{J}_i = \begin{bmatrix} \frac{\partial \mathsf{h}_i}{\partial x} & \frac{\partial \mathsf{h}_i}{\partial y} & \frac{\partial \mathsf{h}_i}{\partial z} & \frac{\partial \mathsf{h}_i}{\partial r} & \frac{\partial \mathsf{h}_i}{\partial p} & \frac{\partial \mathsf{h}_i}{\partial y} \end{bmatrix} \tag{3}$$

3.1.1 Translational Component of Jacobian

We can compute the first three elements of J_i by fixing the sensor frame and perturbing the model in the sensor's frame by some small translational element $\vec{\epsilon}$ (see Figure 1).

$$\vec{\epsilon} \cdot \hat{n} = -\delta_\rho \hat{p} \cdot \hat{n} \tag{4}$$

Where $\cdot$ is the standard vector inner product. By separating ϵ into vectors with components only along x, y, and z, we can compute the positional elements of the Jacobian.

$$\begin{bmatrix} \frac{\partial \mathsf{h}_i}{\partial x} & \frac{\partial \mathsf{h}_i}{\partial y} & \frac{\partial \mathsf{h}_i}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{-n_x}{\hat{p}\bullet\hat{n}} & \frac{-n_y}{\hat{p}\bullet\hat{n}} & \frac{-n_z}{\hat{p}\bullet\hat{n}} \end{bmatrix} \tag{5}$$

3.1.2 Rotational Component of Jacobian

For the rotational elements of the Jacobian, we have (see Figure 2):

$$\rho\cos\psi = \rho\cos\psi\cos\delta_\alpha + \rho\sin\psi\sin\delta_\alpha + \\ \delta_\rho\cos\psi\cos\delta_\alpha + \delta_\rho\sin\psi\sin\delta_\alpha \tag{6}$$

Using a small angle approximation such that $\cos(\delta_\alpha)$ approximately equals 1 and $\sin(\delta_\alpha)$ approximately equals δ_α, we have

$$\delta_\rho = \frac{-\rho\delta_\alpha\sin\psi}{\cos\psi + \delta_\alpha\sin\psi} \tag{7}$$

For small δ_α, this is approximately equivalent to

$$\begin{bmatrix} \frac{\partial \mathsf{h}}{\partial \alpha_x} & \frac{\partial \mathsf{h}}{\partial \alpha_y} & \frac{\partial \mathsf{h}}{\partial \alpha_z} \end{bmatrix} = \\ \begin{bmatrix} \rho\frac{n_z\hat{p_y}-n_y\hat{p_z}}{\hat{n}\bullet\hat{p}} & \rho\frac{n_x\hat{p_z}-n_z\hat{p_x}}{\hat{n}\bullet\hat{p}} & \rho\frac{n_y\hat{p_x}-n_x\hat{p_y}}{\hat{n}\bullet\hat{p}} \end{bmatrix} \tag{8}$$

Thus, using Equation (5) and Equation (8), with Equation (3), we have the full expression for the Jacobian in terms of simple dot and scalar products.

3.1.3 Full Covariance Matrix for $\vec{\Delta\rho}$

Suppose we have N points for which we would like to evaluate the effect they have in constraining the sensor's pose based on their combined information. From the Jacobian in Equation (3), we can write out the full covariance matrix for each point associated with the current pose of the system and stack these for the following vector relationship.

$$\vec{\Delta\rho} = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_N \end{bmatrix} = \begin{bmatrix} \mathsf{J}_1(\hat{n}_1, \hat{p}_1, \rho_1) \\ \mathsf{J}_2(\hat{n}_2, \hat{p}_2, \rho_2) \\ \vdots \\ \mathsf{J}_N(\hat{n}_N, \hat{p}_N, \rho_N) \end{bmatrix} \Delta\mathbf{x} = \mathsf{J}\Delta\mathbf{x} \tag{9}$$

Thus, we know the covariance, $\mathsf{C}_{\vec{\Delta\rho}}$, of the vector $\vec{\Delta\rho}$ is:

$$\begin{aligned} \mathsf{C}_{\vec{\Delta\rho}} &= \mathsf{E}\left[\vec{\Delta\rho}\vec{\Delta\rho}^{\mathsf{T}}\right] \\ &= \mathsf{E}\left[\mathsf{J}\Delta\mathbf{x}(\mathsf{J}\Delta\mathbf{x})^{\mathsf{T}}\right] \\ &= \mathsf{J}\mathsf{C}_{\mathbf{x}}\mathsf{J}^{\mathsf{T}} \end{aligned} \tag{10}$$

where $\mathsf{C}_{\mathbf{x}}$ is the input covariance of the pose $\mathbf{x}$ and $\mathsf{E}[\bullet]$ is the expectation operator. Note, the dimensionality of J is $N \times 6$, and $\mathsf{C}_{\mathbf{x}}$ is 6×6, which implies the covariance of $\mathsf{C}_{\vec{\Delta\rho}}$ is $N \times N$, or explicitly:

$$\mathsf{C}_{\vec{\Delta\rho}} = \begin{bmatrix} \mathsf{C}_{\rho_1\rho_1} & \mathsf{C}_{\rho_1\rho_2} & \dots & \mathsf{C}_{\rho_1\rho_N} \\ \mathsf{C}_{\rho_2\rho_1} & \mathsf{C}_{\rho_2\rho_2} & \dots & \mathsf{C}_{\rho_2\rho_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{C}_{\rho_N\rho_1} & \mathsf{C}_{\rho_N\rho_2} & \dots & \mathsf{C}_{\rho_N\rho_N} \end{bmatrix} \tag{11}$$

where $\mathsf{C}_{\rho_i\rho_j} = \mathsf{J}_i\mathsf{C}_{\mathbf{x}}\mathsf{J}_j^{\mathsf{T}}$.

3.2. Mutual Information to Direct Point Selection

The Mutual Information (MI) of two continuous distributions is the average expected reduction in entropy of one parameter on learning the exact value of the other, given by (from [5]):

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= \sum_{x\in X, y\in Y} p(x,y)\delta x \delta y log_2 \frac{p(x|y)}{p(x)} \end{aligned} \tag{12}$$

where δx and δy are discrete bin sizes where the discrete probability density function is defined. In the limit as $\delta x \to 0$ and $\delta y \to 0$ we have

$$I(X;Y) = \int_{x,y} p(x,y) log_2 \frac{p(x|y)}{p(x)} dxdy \tag{13}$$

For the measurement of our range value, we can assume a Gaussian distribution for $p(x)$. Consider two different ranges ρ_a, and ρ_b. The probability density function of ρ_a is

$$p(\rho_a) = (2\pi)^{-\frac{1}{2}} |\mathsf{C}_{\rho_a\rho_a}|^{-\frac{1}{2}} \mathsf{e}^{-\frac{1}{2}(\rho_a - \hat{\rho_a})^{\mathsf{T}} \mathsf{C}^{-1}_{\rho_a\rho_a}(\rho_a - \hat{\rho_a})} \tag{14}$$

If we know the value that ρ_b will take on, we can write the updated value for ρ_a and the covariance for ρ_a as

$$\hat{\rho}'_a = \hat{\rho}_a + \mathsf{C}_{\rho_a\rho_b}\mathsf{C}^{-1}_{\rho_b\rho_b}(\rho_b - \hat{\rho}_b) \tag{15}$$

$$\mathsf{C}'_{\rho_a\rho_a} = \mathsf{C}_{\rho_a\rho_a} - \mathsf{C}_{\rho_a\rho_b}\mathsf{C}^{-1}_{\rho_b\rho_b}\mathsf{C}_{\rho_b\rho_a} \tag{16}$$

This allows us to write the conditional probability for ρ_a given that we know the range ρ_b as

$$p(\rho_a|\rho_b) = (2\pi)^{-\frac{1}{2}} |\mathsf{C}'_{\rho_a\rho_a}|^{-\frac{1}{2}} \mathsf{e}^{-\frac{1}{2}(\rho_a - \hat{\rho_a}')^{\mathsf{T}} \mathsf{C}'^{-1}_{\rho_a\rho_a}(\rho_a - \hat{\rho_a}')} \tag{17}$$

With this, Equation (13) becomes

$$I(\rho_a;\rho_b) = \frac{1}{2}\log_2 \frac{|\mathsf{C}_{\rho_a\rho_a}|}{|\mathsf{C}_{\rho_a\rho_a} - \mathsf{C}_{\rho_a\rho_b}\mathsf{C}^{-1}_{\rho_b\rho_b}\mathsf{C}_{\rho_b\rho_a}|}. \tag{18}$$

This allows us to compute the full Mutual Information matrix MI in terms of covariances computed in Equation (11):

$$\begin{bmatrix} * & I(\rho_1;\rho_2) & I(\rho_1;\rho_3) & \dots & I(\rho_1;\rho_N) \\ I(\rho_2;\rho_1) & * & I(\rho_2;\rho_3) & \dots & I(\rho_2;\rho_N) \\ I(\rho_3;\rho_1) & I(\rho_3;\rho_2) & * & \dots & I(\rho_3;\rho_N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I(\rho_N;\rho_1) & I(\rho_N;\rho_2) & I(\rho_N;\rho_3) & \dots & * \end{bmatrix} \tag{19}$$

The MI matrix allows us to evaluate the information correlation between points. Each point has an associated row and column. For entries with high scores, the information provided between the row's point and the column's point is likely to be redundant. For example, the redundant information may be that their positions are close, or that their normals are aligned. It is not the precise dimension of the redundancy, but rather the existence of redundancy, as indicated by the MI, that allows us to sort the points for a directed selection. The mutual information between a point and itself is meaningless, thus the $*$'s along the diagonal. We use this to direct the choice of points in DIRSAC for each trial as described in the following section.

3.3. DIRSAC - Algorithm Description

After obtaining a scan of 3D data from the range sensor, we compute the Mutual Information (MI) matrix (Equation (19)) associated with the point cloud. The first point is selected at random which defines a row in MI. This row contains the information correlation with all other points. We choose the next best point which adds the most information to the system. This is the column with the lowest score because it has the least correlated information with our currently selected point. The rows corresponding to the currently selected points in MI are summed together and the next best column is chosen from this sum. This is continued until enough points are selected to compute a fit trial. Each fit trial consists of associating the selected points with their corresponding closest point on the model and computing the rigid transform which best aligns the two sets of points [1, 2]. Because of this point association step we require a reasonably good estimate of the initial sensor pose. Once all the trials are computed, we search through them to find the best solution with the lowest residual error and highest set of inliers in the same manner as the standard RANSAC algorithm.

4. Experiments and Results

We have performed experiments on both simulated and real data. The simulated scanner is configured to mimic the Sick LMS400, mounted in a nodder and positioned such that we have a reasonable view of the surface to be scanned. In simulation we have the real ground truth of the pose of the sensor and the model in the world. For the real data, we used the described method to find the best estimate of the pose of the sensor with respect to the model being scanned as ground truth. For both methods, we add an artificial pose error of around 7cm in position and 2.25° to the sensor's pose. The proposed method is not inherently limited to this. However, the point to model association distance within the ICP algorithm is set to limit the distance to the model to around 10cm. The novelty presented within this paper is in the sorting of the points which is based solely on the point clouds prior to this model association. Thus, we compare to a naïve RANSAC approach which uses the same fitting methods and with the same starting errors. Using the DIRSAC approach increases the probability of getting a good constraining solution so we can reduce the number of iterations required to guarantee a good final solution.

We present here results from three different surfaces. First, a scan of the inside of a box, which represents 3 orthogonal surfaces and is well constrained from both RANSAC and DIRSAC's perspectives. Because of the obvious good constraints of 3 orthogonal planes, we expect both algorithms to perform similarly well. Second, we use a table surface with some small constraining geometry on the surface. This surface is still well constrained, but because of the minimal constraining geometry, we expect DIRSAC to out perform RANSAC. We also present real scanned data of this surface to demonstrate the algorithm's performance. Finally, we scanned a completely flat surface. We know there are three unconstrained dimensions for a flat plane. Because the DIRSAC approach will naturally degrade to a RANSAC approach in the limit, we expect RANSAC and DIRSAC to perform similarly in this unconstrained environment. However, it turns out that DIRSAC does a better job by attempting to find and use the existing constraints even when some dimensions are unconstrained.

The results are presented for each model as a set of four plots. The top row of each figure shows the positional analysis, and the bottom row is the orientation analysis. Each point shows a single trial result of the algorithm. For each trial, six points are selected, and based on the selection of those six points, a trial fit is performed. Depending on the set of points chosen, a different resulting error in the final pose will be computed. The overall distance between the poses after each of the 150 trials is reported in the upper left positional analysis portion of each figure. The orientation error computed as an absolute value of the angle between the computed solution pose and the ground truth pose is shown in the bottom left of each figure. The right column of each figure represents the probability of computing a solution with error less than the error on the x axis. The starting error is displayed for reference.

4.1. "InsideBox" Results

This model consists of three orthogonal surfaces which constrains the fit well. Even in the naïve RANSAC we expect this surface to be easily localized. As Figure 3 shows, Both algorithms perform similarly. However, there are still many cases where RANSAC chooses less than optimally constraining points, resulting in higher trial errors. In contrast, the DIRSAC trial solutions produce the lower errors for nearly every trial because we are actively choosing points to constrain the fit.

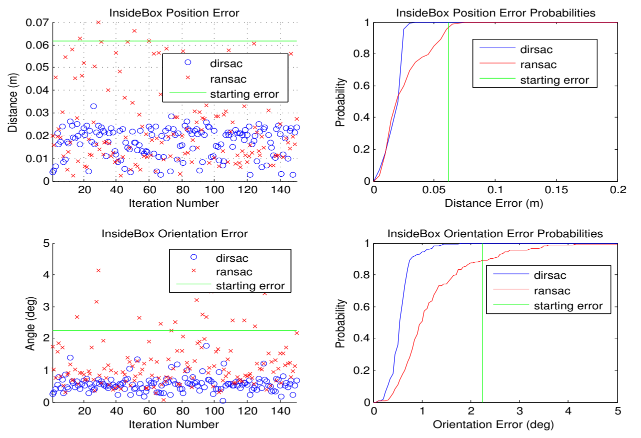

Figure 3. "InsideBox" results. Both RANSAC and DIRSAC find good solutions, but clearly DIRSAC finds lower error solutions more consistently.

4.2. "Table" Results

This model (shown in Figure 4) was used for both the simulated "Table" results (Figure 5) as well as the real data collected in the "TableRealData" results (Figure 6). The model consists of a table surface with four short 2x4's placed in different orientations to provide some minimal constraining geometry. Because of the shape of this model, RANSAC and DIRSAC will both recover the z-position, $roll$, and $pitch$ of the sensor's pose because these dimensions are well constrained by the large model surfaces. However, the x/y-position and yaw of the sensor will be constrained only by the small amount of data collected on the 2x4 sides orthogonal to the table's surface. Thus, for the results shown in Figures 5 and 6 we are displaying the x/y dimensions for the positional results and the yaw dimension for the orientation results.

Notice how in both the simulated and real data cases RANSAC's results rarely compute an improvement in these unconstrained dimensions for any single trial. In contrast, DIRSAC regularly computes solutions which refine these unconstrained dimensions because of the active directed point selection in the DIRSAC algorithm. These results highlight the benefit of using DIRSAC over the naïve RANSAC in cases where small features must not be overlooked to truly constrain the solution.

4.3. "FlatSlab" Results

This model's results shown in Figure 7 demonstrate that DIRSAC still out performs RANSAC. This is because the random point selection does not consider any constraint information. So, the points may not be evenly spread out. In the case where we truly have some dimensions unconstrained, DIRSAC is still able to select points which constrain as much of the solution as possible, such that the

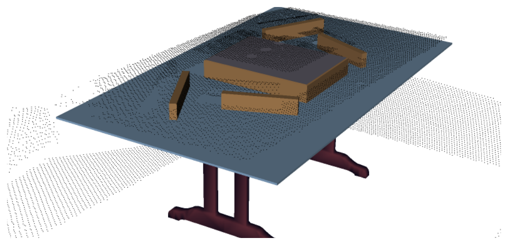

Figure 4. The real data point cloud scan (black dots) of the “Table” surface shown overlaid on the “Table” model.

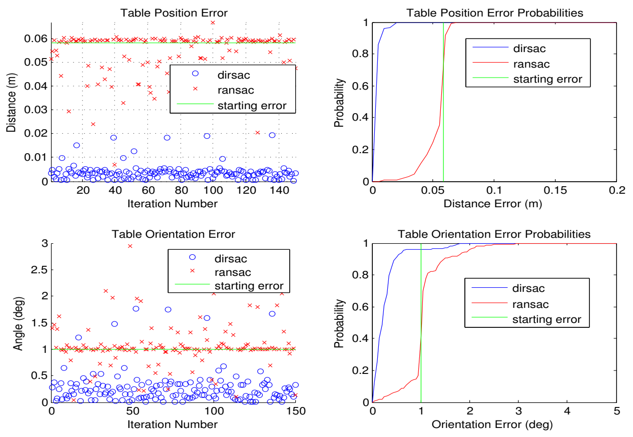

Figure 5. “Table” results from the model displayed in Figure 4. To highlight the performance of DIRSAC *vs.* RANSAC, we are displaying the unconstrained dimensions only. RANSAC rarely computes a good solution which reduces the error in the minimally constrained dimensions. However, DIRSAC reduces the error very consistently along these unconstrained dimensions.

points on a plane would at least be evenly spread out. This is another advantage over the standard random selection even if the solution is not completely constrained. Notably, RANSAC does get solutions with equally low errors when compared to DIRSAC, but not nearly as consistently.

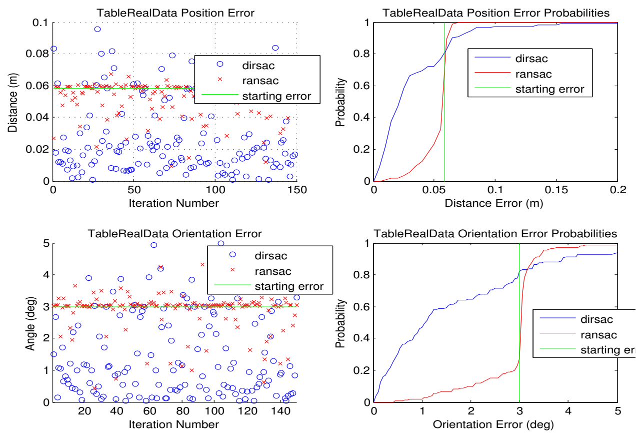

Figure 6. “TableRealData” results from the model displayed in Figure 4. To highlight the performance of DIRSAC *vs.* RANSAC, we are displaying the unconstrained dimensions only. Notice how RANSAC rarely computes a solution which improves the error in the unconstrained dimensions. In contrast, DIRSAC regularly improves the solution along these unconstrained dimensions.

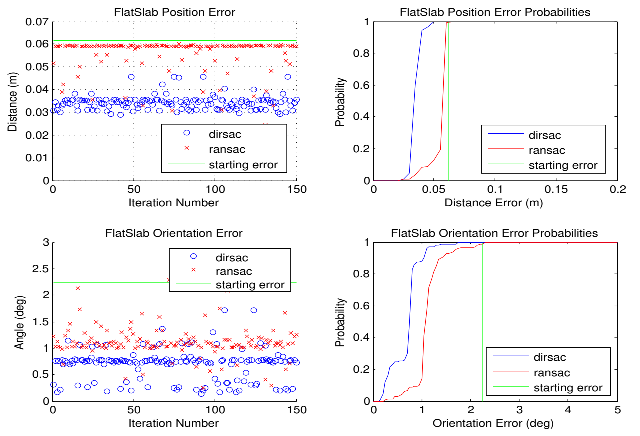

Figure 7. “FlatSlab” results. Both RANSAC and DIRSAC find good solutions, but clearly DIRSAC finds lower error solutions much more consistently.

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 9(5):698–700, May 1987. 4

[2] P. Besl and H. McKay. A method for registration of 3-d shapes. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 14(2):239 –256, Feb 1992. 1, 4

[3] D. P. Capel. An effective bail-out test for ransac consensus scoring. In *BMVC*. British Machine Vision Assn, 2005. 2

[4] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *IEEE Computer Society Conf on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 220–226, June 20-25 2005. 2

[5] A. J. Davison. Active search for real-time vision. In *Tenth IEEE Int’l Conf on Computer Vision (ICCV)*, volume 1, pages 66–73, Dec. 5 2005. 2, 4

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, pages 381–395, 1981. 1

[7] J.-M. Frahm and M. Pollefeys. Ransac for (quasi-)degenerate data (qdegsac). In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, volume 1, pages 453–460, June 17-22 2006. 2

[8] J. Matas and O. Chum. Randomized ransac with td,d test. *Image Vision Computing*, 22(10):837–842, 2004. 2

[9] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *Proceedings of the 10th European Conf on Computer Vision (ECCV): Part II*, ECCV ’08, pages 500–513, Berlin, Heidelberg, 2008. Springer-Verlag. 2