Minimal Solvers for 3D Scan Alignment with Pairs of Intersecting Lines

(SUPPLEMENTARY MATERIALS)

André Mateus[1], Srikumar Ramalingam[2], and Pedro Miraldo[1]
[1]Instituto Superior Técnico, Lisboa [2]Google

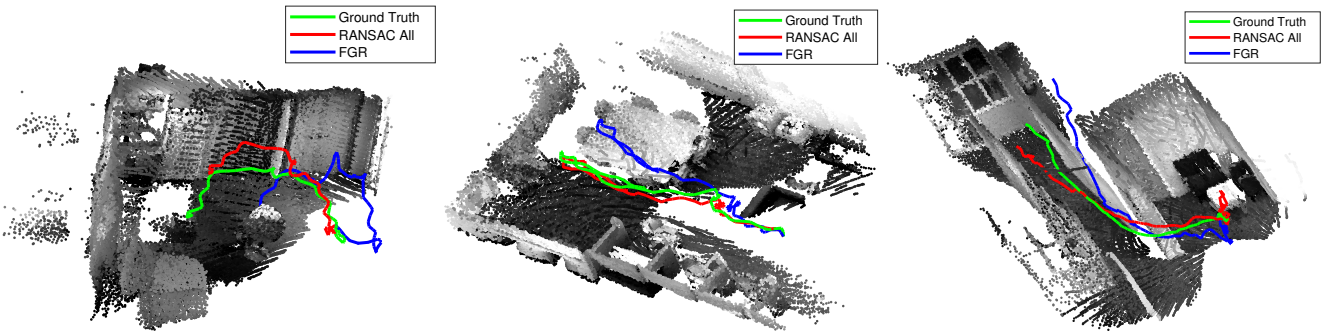


Figure 1: *Visual Odometry results for the MIT/LAB, MIT/76, and BROWN/CS/3 sequences of the SUN3D data-set. For evaluation, we consider the RANSAC ALL, FGR, and Ground-Truth trajectories. Notice we are not running the refinement in these experiments.*

These supplementary materials are organized as follows. We start with showing some new results on nine new sequences from the same two data-sets used in the main document, including some trajectories obtained from our method against the baselines (see Sec. A). **We stress that, to have more challenging scenarios, we increase the baselines of the pairs of 3D scans by considering a gap of 10 scans for each trial**. *Furthermore, these new sequences are from the data-sets used in the paper. We are not using new data-sets. Then, we present the predefined transformations derived for the minimal solvers presented in the paper (Sec. B). To end these supplementary materials, we show some evaluation on the use of different solvers in the proposed RANSAC scheme (results are shown in Sec. C).*

A. Results in New Sequences

This section presents results in additional sequences from each data-set. For both SUN3D [7], and TUM [6], four and five sequences were tested, respectively.

For the SUN3D, Tab. 1 shows that the use of pairs of intersecting lines outperforms ICP [5], GR [1], and FGR [8] in three sequences out of four, for rotation and translation errors. Tab. 2 presents the results for new sequences of TUM data-set. Similar to the SUN3D, the proposed methods outperform the state-of-the-art in the majority of the sequences. In this case in four out of five for the rotation and three out of five for the translation. For the SITTING and 360 sequences, where FGR presents the best results (in one of the cases for the translation and in the other for both the rotation and translation), our methods perform poorly because of the lower amount of lines retrieved by the method in [2, 3], w.r.t. the other sequences from the SUN3D and TUM.

Furthermore, the visual odometry results for three sequences of the SUN3D data-set are presented in Fig. 1. For all three the RANSAC All method presents smaller drift w.r.t the ground truth. We stress that ICP results were omitted, since it presents lower performance than the methods presented in the figure. Finally, the visual odometry results for two of the TUM sequences in Tab. 2 are presented in Fig. 2. The results show that when a sufficient amount of lines that can be retrieved with the method in [2, 3] our method outperforms the FGR method (see the results of the DESK1 sequence). However, when this is not the case, FGR will present smaller drift as shown in the results of the SITTING sequence.

B. Predefined Transformations

This section details the computed predefined transformations used in the derivations for the minimal solvers presented in the main document.

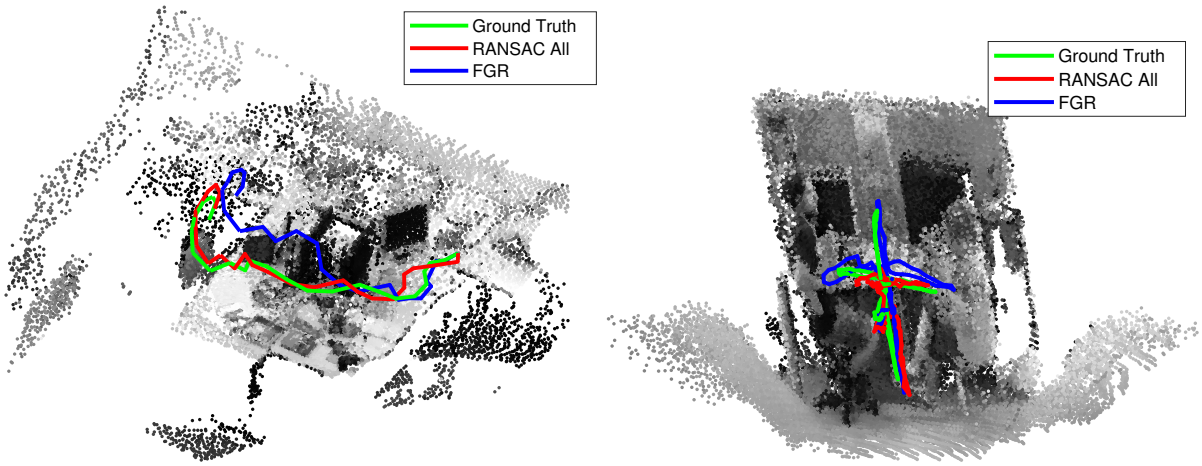


Figure 2: *Visual Odometry results for the DESK1 and SITTING sequences of TUM data-set. We consider the RANSAC ALL and FGR methods, against the ground-truth.*

Rotation Error [Deg]

	SUN3D			
Method	**BROWN CS 3**	**HARVARD C5**	**MIT 76**	**MIT LAB**
ICP [5]	2.73	4.41	2.10	1.20
GR [1]	1.54	**1.14**	1.14	1.45
FGR [8]	1.74	1.64	1.16	1.32
RANSAC 6L	1.49	1.47	**1.11**	**0.99**
RANSAC 3Q	1.67	1.68	1.38	1.09
RANSAC Ours	1.46	1.29	1.14	1.01
RANSAC All	**1.37**	1.26	1.24	**0.99**
Ours + Refinement	1.46	1.25	1.12	**0.99**

Translation Error [cm]

	SUN3D			
Method	**BROWN CS**	**HARVARD C5**	**MIT 76**	**MIT LAB**
ICP [5]	19.61	10.73	11.54	7.63
GR [1]	7.87	6.45	5.93	5.92
FGR [8]	9.75	6.87	**4.78**	4.51
RANSAC 6L	6.89	6.88	4.97	**4.10**
RANSAC 3Q	7.38	8.02	5.10	4.35
RANSAC Ours	6.66	7.59	4.93	4.21
RANSAC All	6.82	6.89	4.81	3.73
Ours + Refinement	**6.47**	**5.95**	**4.78**	4.12

Table 1: *Median rotation and translation errors for SUN3D data-set [7].*

Rotation Error [Deg]

	TUM				
Method	**360**	**DESK1**	**CABINET**	**SITTING**	**PIONEER**
ICP [5]	14.72	5.71	3.40	0.75	7.24
GR [1]	3.35	3.03	3.01	1.14	2.92
FGR [8]	2.96	2.84	2.77	**0.70**	4.31
RANSAC 6L	3.69	2.23	3.29	0.77	2.29
RANSAC 3Q	4.07	2.49	2.91	0.96	2.88
RANSAC Ours	2.97	1.97	2.41	0.82	2.06
RANSAC All	**2.68**	2.06	2.43	0.80	**2.01**
Ours + Refinement	2.97	**1.94**	**2.10**	0.78	2.12

Translation Error [cm]

	TUM				
Method	**360**	**DESK1**	**CABINET**	**SITTING**	**PIONEER**
ICP [5]	28.85	11.52	8.00	1.46	18.63
GR [1]	8.62	5.18	6.84	3.39	13.33
FGR [8]	**6.12**	4.34	5.19	**1.43**	14.62
RANSAC 6L	9.73	3.71	7.22	2.52	8.45
RANSAC 3Q	10.54	4.05	5.72	3.44	10.33
RANSAC Ours	9.66	**3.05**	4.76	2.60	10.38
RANSAC All	8.46	3.12	**3.83**	2.54	**8.54**
Ours + Refinement	9.66	3.27	4.92	2.54	8.90

Table 2: *Median rotation and translation errors for TUM data-set [6].*

B.1. Predefined Transformations for 3L1P

For the $U_{3\text{L1P}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{3\text{L1P}} \in \mathbf{R}^3$ as shown in Fig. 2(a) of the paper, we start by defining the third column of $U_{3\text{L1P}}$:

$$u_3 = \overline{\pi}_1 / \|\overline{\pi}_1\|. \tag{1}$$

Then, we set two possible guesses for u_1 (this rotation can be defined up to a rotation degree of freedom):

$$u_1^- = [0\ 1\ 0] \times u_3 \ \text{ and } \ u_1^+ = [1\ 0\ 0] \times u_3, \tag{2}$$

and set the first column of $U_{3\text{L1P}}$ as $u_1 = u_1^*/\|u_1^*\|$ where u_1^* is equal to the vector in (2) with the larger norm. Then, we define

$$U_{3\text{L1P}} = \begin{bmatrix} u_1 & u_3 \times u_1 & u_3 \end{bmatrix}, \ \text{ and } \ \mathbf{u}_{3\text{L1P}} = \widetilde{\pi}_1 U_{3\text{L1P}} \overline{\pi}_1. \tag{3}$$

B.2. Predefined Transformations for 1L2P

Consider now the predefined transformations shown in Fig. 2(b) of the main document. For the $U_{1\text{L2P}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{1\text{L2P}} \in \mathbb{R}^3$, we use the method derived in Sec. B.1. For $V_{1\text{L2P}} \in \mathcal{SO}(3)$ and $\mathbf{v}_{1\text{L2P}} \in \mathbb{R}^3$, we first find the 3D line r (in *Plücker* coordinates) that represents the intersection of the two planes π_1 and π_2 [4]:

$$r = \begin{bmatrix} \overline{\pi}_1 \times \overline{\pi}_2 & \widetilde{\pi}_1 \overline{\pi}_2 - \widetilde{\pi}_2 \overline{\pi}_1 \end{bmatrix}. \tag{4}$$

Then, we define

$$V_{\text{1L2P}} = \begin{bmatrix} r_1/\sqrt{r_1^2+r_2^2} & -r_2/\sqrt{r_1^2+r_2^2} & 0 \\ r_2/\sqrt{r_1^2+r_2^2} & r_1/\sqrt{r_1^2+r_2^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

where r_1 and r_2 are the 1st and 2nd elements in vector r. Now, for the translation vector $\mathbf{v}_{\text{1L2P}}$, we first compute the closest 3D point $\mathbf{x} \in \mathbb{R}^3$ to the line r:

$$\mathbf{x} = \hat{r} \times \overline{r}, \quad (6)$$

and $\mathbf{v}_{\text{1L2P}}$ is given by

$$\mathbf{v}_{\text{1L2P}} = V_{\text{1L2P}}\mathbf{x}. \quad (7)$$

B.3. Predefined Transformations for 3L1Q

In this case the transformation consists only in a translation, since we are moving the origin of the 3D data to the point $\mathbf{q}_1$. Thus, the $U_{\text{3L1Q}} \in \mathcal{SO}(3)$ and $\mathbf{u}_{\text{3L1Q}} \in \mathbb{R}^3$ in Fig. 2(c) of the paper, are represented as:

$$U_{\text{3L1Q}} = \mathbf{I}, \quad (8)$$

and

$$\mathbf{u}_{\text{3L1Q}} = -\mathbf{q}_1, \quad (9)$$

where $\mathbf{I}$ is the 3×3 identity matrix.

B.4. Predefined Transformations for 1L2Q

This subsection presents the predefined transformation for the solver in Section B.4, as shown in Fig. 2(d) of the paper. The first transformation consists in a pure translation to the point $\mathbf{q}_1$, which can be written as:

$$U_{\text{1L2Q}} = \mathbf{I}_3, \text{ and } \mathbf{u}_{\text{1L2Q}} = -\mathbf{q}_1, \quad (10)$$

where U_{1L2Q} and $\mathbf{u}_{\text{1L2Q}}$ are a rotation matrix and a translation vector respectively. The second step consists in a pure rotation V_{1L2Q}, which can be obtained in a similar fashion to the matrix U_{1L2P} in Sec. B.2. Both cases can be seen as aligning the $z-$axis with either the plane normal in Sec. B.2 or the direction from $\mathbf{q}_2$ to $\mathbf{q}_1$.

B.5. Predefined Transformations for 1L1Q1P

This subsection presents the predefined transformation for the solver in Section B.5, which are depicted in Fig. 2(e) of the main document. The first transformation consists in a pure translation to the projection of point $\mathbf{q}$ to π. This projection can be obtained by computing the signed distance d_{pq} of the point to the plane, and then subtracting the plane normal vector scaled by the distance to the point. The signed distance can be obtained by inputting the point in the plane equation as:

$$d_{pq} = \overline{\pi}^T\mathbf{q} + \tilde{\pi} \quad (11)$$

The point $\mathbf{q}_\pi$ in the plane corresponding to the projection of $\mathbf{q}$ to π is obtain by:

$$\mathbf{q}_\pi = \mathbf{q} - d_{pq}\overline{\pi}. \quad (12)$$

The first transformation is thus given by:

$$U_{\text{1L1Q1P}} = \mathbf{I}_3, \quad \text{and} \quad \mathbf{u}_{\text{1L1Q1P}} = -\mathbf{q}_\pi, \quad (13)$$

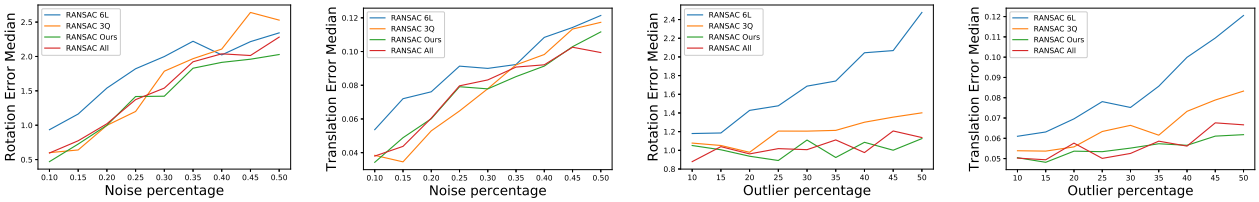
The second step consists in a pure rotation V_{1L1Q1P}, which can be obtained in a similar fashion to the matrix U_{1L2P} in Sec. B.2.

C. Minimal Solvers in RANSAC

This section presents the evaluation of the minimal solvers proposed in this work *vs.* the minimal solvers using only 3D point intersection correspondences or line intersections, in the hybrid RANSAC framework.

The data for this tests was generated by at first sampling a pair of line intersections – represented by two points – in each frame without noise. Then, 15 points are sampled between the two points representing each line. Noise is added to those points, by sampling a point in Gaussian centered at the point, and with standard deviation given by a defined percentage of the size of the cube, where the lines are created from. In this work, the size of the cube is set to 40 units. Finally, a line is fitted to each set of points using least squares. Two line intersection can be created from each pair of the original lines, by pair the first line the one frame to the second and *vice-versa*. A point correspondence is created from each pair of lines by taking the point of intersection of the lines in each frame. Since noise was added to the lines, these do not intersect necessarily, so the mean point of the closest points in each intersecting line was considered. Plane correspondences are created by fitting a plane in each frame to the noisy points using least squares. Outliers are added afterward, by applying a random rigid transformation to a defined percentage of the data.

Two tests were performed. The first consisted of fixing the outlier percentage at 30%, and the percentage of the noise added to the data is varied from 0.1% to 0.5% with increases of 0.05%. For each noise and outlier percentage, 100 runs of RANSAC with different randomly generated data-sets were performed. The results are presented in Fig. 3(a) and Fig. 3(b) for the median rotation and translation errors respectively. Even though RANSAC 3Q presents similar performance to RANSAC Ours and All for the lower noise levels, the latter methods perform better for higher levels of noise. The second consisted of the reverse, i.e., the noise percentage was fixed, and the outlier percentage ranged from 10% to 50% with increments of 5%. The results are presented in Fig. 3(c) and Fig. 3(d) for the median rotation and translation errors respectively. From these results, we conclude that both RANSAC Ours and All are



(a) Median rotation error for different noise levels.

(b) Median translation error for different noise levels.

(c) Median rotation error for different outlier percentages.

(d) Median translation error for different outlier percentages.

Figure 3: Synthetic data evaluation of the hybrid RANSAC framework with different sets of minimal solvers.

more robust to outliers than RANSAC 6L and 3Q, indicating the improvements accomplished by mixing point and plane matches with line intersections.

References

[1] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015.

[2] Kai Li, Jian Yao, and Xiaohu Lu. Robust line matching based on ray-point-ray structure descriptor. In *Asian Conf. Computer Vision (ACCV)*, pages 554–569, 2014.

[3] Kai Li, Jian Yao, Xiaohu Lu, Li Li, and Zhichao Zhang. Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220, 2016.

[4] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer-Verlag Berlin Heidelberg, 1 edition, 2001.

[5] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling (3DIM)*, volume 1, pages 145–152, 2001.

[6] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 573–580, 2012.

[7] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1625–1632, 2013.

[8] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conf. Computer Vision (ECCV)*, pages 766–782, 2016.