Supplementary Material
RSL-BA: Rolling Shutter Line Bundle Adjustment

Yongcong Zhang[1,3,*], Bangyan Liao[2,*], Yifei Xue[1,3], Chen Lu[4], Peidong Liu[2], and Yizhen Lao[1,†]

[1]Hunan University [2]Westlake University [3]DaHe.AI
[4]Dreame Technology Company

Overview

In this supplementary material, we further discuss the following content:

- The transformation between Plücker coordinates and orthogonal representation(Sec. 1).
- Jacobian derivation for RSL-BA(Sec. 2).
- The impact of the number of feature lines and points(Sec. 3).
  - What is the optimal number of points to measure along a line?(Sec. 3.1).
  - What is the optimal number of lines to employ for RSL-BA?(Sec. 3.2).
- Complete results on the TUM-RSVI and WHU-RSVI dataset(Sec. 4).
  - Synthetic Images.(Sec. 4.1).
  - Real Images.(Sec. 4.2).

1 The Transformation between Plücker Coordinates and Orthogonal Representation

The Plücker coordinate of the line are defined as: $\mathbf{L} = (\mathbf{n}^\top, \mathbf{a}^\top)^\top$ with the orthogonal representation parameters $\boldsymbol{\tau} = [\psi_1, \psi_2, \psi_3, \phi]^\top$. Where $\mathbf{a} \in \mathbf{R}^3$ represents the direction vector of the line, $\mathbf{n} \in \mathbf{R}^3$ represents the normal vector. We have [2,4]:

$$\mathbf{U} = \mathbf{Exp}([\psi_1, \psi_2, \psi_3]^\wedge) = \begin{bmatrix} \frac{\mathbf{n}}{\|\mathbf{n}\|} & \frac{\mathbf{a}}{\|\mathbf{a}\|} & \frac{\mathbf{n}\times\mathbf{a}}{\|\mathbf{n}\times\mathbf{a}\|} \end{bmatrix} \tag{1}$$

The function $\mathbf{Exp}$ maps from $\mathfrak{so}(3)$ to $\mathbf{SO}(3)$, and $\boldsymbol{\psi} = [\psi_1, \psi_2, \psi_3]^\top$ represents the rotation angles from the camera coordinate system to the line coordinate system around the x, y, and z axes, respectively. By utilizing equation Eq. (1), we can obtain the first term of the orthogonal representation.

* Equal contribution
† Corresponding author: (yizhenlao@hnu.edu.cn)
Project page: https://github.com/zhangtaxue/RSL-BA

$$\mathbf{W} = \begin{bmatrix} w_1 & -w_2 \\ w_2 & w_1 \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} = \frac{1}{\sqrt{\|\mathbf{n}\|^2 + \|\mathbf{a}\|^2}} \begin{bmatrix} \|\mathbf{n}\| & -\|\mathbf{a}\| \\ \|\mathbf{a}\| & \|\mathbf{n}\| \end{bmatrix} \quad (2)$$

With Eq. (2), we can obtain the second term of the orthogonal representation.

The transformation from the orthogonal representation to Plücker coordinates can be computed as follows:

$$\mathbf{L}' = [w_1\mathbf{u}_1^\top, w_2\mathbf{u}_2^\top]^\top = \frac{1}{\sqrt{\|\mathbf{n}\|^2 + \|\mathbf{a}\|^2}}\mathbf{L} \quad (3)$$

$\mathbf{L}'$ and $\mathbf{L}$ differ by a scale factor, but represent the same line.

2 Jacobian Derivation for RSL-BA

Since lines in space only have four degrees of freedom, and Plücker coordinates are over-parameterized, they cannot be directly used for unconstrained optimization. Therefore, we often use Plücker coordinates for initialization and transformation, while employing an orthogonal representation for parameter optimization. Let the representation of the space line $\mathbf{L}$ in the world coordinate system be $\mathbf{L}_w = (\mathbf{n}_w^\top, \mathbf{a}_w^\top)^\top$, where $\mathbf{n}_w$ and $\mathbf{a}_w$ respectively denote the normal vector to the line and the direction of the line from the camera center. The representation of the line $\mathbf{L}$ in the camera coordinate system is $\mathbf{L}_c = (\mathbf{n}_c^\top, \mathbf{a}_c^\top)^\top$. The matrix from the world coordinate system to the camera when the camera exposes the v-th row of pixels is:

$$\mathbf{T}_{cw}^v = \begin{bmatrix} (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} & \mathbf{t}_{cw} + v\mathbf{d} \\ 0 & 1 \end{bmatrix} \quad (4)$$

The transformation of Plücker line coordinates from the world coordinate system to the camera coordinate system is denoted as $\mathbf{N}_{cw}^v$:

$$\mathbf{N}_{cw}^v = \begin{bmatrix} (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} & [\mathbf{t}_{cw} + v\mathbf{d}]_\times(\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \\ \mathbf{0} & (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \end{bmatrix} \quad (5)$$

We have:

$$\mathbf{L}_c^v = \mathbf{N}_{cw}^v\mathbf{L}_w = \begin{bmatrix} (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} & [\mathbf{t}_{cw} + v\mathbf{d}]_\times(\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \\ \mathbf{0} & (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \end{bmatrix} L_w \quad (6)$$

Let the parametric equation of the curve under the aforementioned parameters be:

$$l_1v^3 + l_2uv^2 + (l_3 + l_4)v^2 + l_5uv + (l_6 + l_7)v + l_8u + l_9 = 0 \quad (7)$$

Take a point $q = [u \quad v]^\top$ from the projected curve. Next, we will compute the Jacobian matrices of various error functions. Firstly, let's consider the perpendicular-distance error:

$$\mathbf{e}_{d1} = \frac{|l_1 v^3 + l_2 u v^2 + (l_3 + l_4)v^2 + l_5 uv + (l_6 + l_7)v + l_8 u + l_9|}{\sqrt{(l_8 + v l_5 + v^2 l_2)^2 + (l_6 + v l_3 + v^2 l_1)^2}} \tag{8}$$

According to the chain rule for differentiation, the Jacobian matrix is represented as [2, 4]:

$$\mathbf{J}_{\mathbf{e}_{d1}} = \frac{\partial \mathbf{e}_{d1}}{\partial \mathbf{s_c}} \frac{\partial \mathbf{s_c}}{\mathbf{L}_c^v} [\frac{\partial \mathbf{L}_c^v}{\partial \delta \mathbf{x}} \quad \frac{\partial \mathbf{L}_c^v}{\partial \mathbf{L}_w} \frac{\partial \mathbf{L}_w}{\partial \delta \boldsymbol{\tau}}] \tag{9}$$

The first term represents the partial derivative of the error with respect to the curve parameters, while the second term represents the partial derivative of the curve parameters with respect to the line features in the camera coordinate system. The last term in the matrix contains two parts: one is the derivative of the rotation, translation, angular velocity and linear velocity with respect to the line features in the camera coordinate system, and the other is the derivative of the four parameters increment with respect to the line in its orthogonal representation.

The first term:

$$\frac{\partial \mathbf{e}_{d1}}{\partial \mathbf{s_c}} = [\frac{\partial \mathbf{e}_{d1}}{\partial l_1} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_2} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_3} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_4} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_5} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_6} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_7} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_8} \quad \frac{\partial \mathbf{e}_{d1}}{\partial l_9}]_{1\times 9} \tag{10}$$

The second term:

$$\frac{\partial \mathbf{s_c}}{\partial \mathbf{L}_c^v} = [\frac{\partial \mathbf{s_c}}{\partial \mathbf{n}_c} \quad \frac{\partial \mathbf{s_c}}{\partial \mathbf{a}_c}] = \begin{bmatrix} \frac{\partial l_1}{\partial n_1} & \frac{\partial l_1}{\partial n_2} & \cdots \frac{\partial l_1}{\partial a_3} \\ \frac{\partial l_2}{\partial n_1} & \frac{\partial l_2}{\partial n_2} & \cdots \frac{\partial l_2}{\partial a_3} \\ \cdots\cdots & & \\ \frac{\partial l_9}{\partial n_1} & \frac{\partial l_9}{\partial n_2} & \cdots \frac{\partial l_9}{\partial a_3} \end{bmatrix}_{9\times 6} \tag{11}$$

The first term in the parentheses:

$$\begin{aligned} \mathbf{L}_c^v &= \begin{bmatrix} (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} & [\mathbf{t}_{cw} + v\mathbf{d}]_\times (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \\ \mathbf{0} & (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw} \end{bmatrix} L_w \\ &= \begin{bmatrix} (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{n}_w + [\mathbf{t}_{cw} + v\mathbf{d}]_\times (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w \\ (\mathbf{I} + v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w \end{bmatrix} \end{aligned} \tag{12}$$

We first differentiate with respect to rotation:

$$\frac{\partial \mathbf{L}_c^v}{\partial \delta \boldsymbol{\theta}} = \begin{bmatrix} \frac{(\mathbf{I}+v[\boldsymbol{\omega}]_\times)(\mathbf{I}+[\delta\boldsymbol{\theta}]_\times)\mathbf{R}_{cw}n_w + [\mathbf{t}_{cw}+v\mathbf{d}]_\times (\mathbf{I}+v[\boldsymbol{\omega}]_\times)(\mathbf{I}+[\delta\boldsymbol{\theta}]_\times)\mathbf{R}_{cw}\mathbf{a}_w}{\partial \delta\boldsymbol{\theta}} \\ \frac{(\mathbf{I}+v[\boldsymbol{\omega}]_\times)(\mathbf{I}+[\delta\boldsymbol{\theta}]_\times)\mathbf{R}_{cw}\mathbf{a}_w}{\partial \delta\boldsymbol{\theta}} \end{bmatrix} \tag{13}$$

We observe that all of them have the form $\frac{\partial (A[\delta\boldsymbol{\theta}]_\times b)}{\partial \delta\boldsymbol{\theta}}$, where A is a 3×3 matrix, and b is a 3×1 matrix:

$$\mathbf{A} = \begin{bmatrix} A_1 & A_2 & A_3 \\ A_4 & A_5 & A_6 \\ A_7 & A_8 & A_9 \end{bmatrix}, [\delta\boldsymbol{\theta}]_\times = \begin{bmatrix} 0 & -\delta\theta_3 & \delta\theta_2 \\ \delta\theta_3 & 0 & -\delta\theta_1 \\ -\delta\theta_2 & \delta\theta_1 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b1 \\ b2 \\ b3 \end{bmatrix} \tag{14}$$

Expanding $\frac{\partial(\mathbf{A}[\delta\boldsymbol{\theta}]_\times\mathbf{b})}{\partial\delta\boldsymbol{\theta}}$, we get:

$$\frac{\partial(\mathbf{A}[\delta\boldsymbol{\theta}]_\times\mathbf{b})}{\partial\delta\boldsymbol{\theta}} = \begin{bmatrix}(A_3b_2 - A_2b_3)\ (A_1b_3 - A_3b_1)\ (A_2b_1 - A_1b_2)\\ (A_6b_2 - A_5b_3)\ (A_4b_3 - A_6b_1)\ (A_5b_1 - A_4b_2)\\ (A_9b_2 - A_8b_3)\ (A_7b_3 - A_9b_1)\ (A_8b_1 - A_7b_2)\end{bmatrix} \tag{15}$$

Differentiate with respect to translation:

$$\begin{aligned}\frac{\partial\mathbf{L}_c^v}{\partial\mathbf{t}_{cw}} &= \begin{bmatrix}\frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{n}_w+[\mathbf{t}_{cw}+v\mathbf{d}]_\times(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{t}_{cw}}\\ \frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{t}_{cw}}\end{bmatrix}\\ &= \begin{bmatrix}\frac{\partial([\mathbf{t}_{cw}]_\times(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{t}_{cw}}\\ \mathbf{0}\end{bmatrix} = -\begin{bmatrix}[(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}a_w]_\times\\ \mathbf{0}\end{bmatrix}_{6\times3}\end{aligned} \tag{16}$$

Differentiate with respect to angular velocity:

$$\frac{\partial\mathbf{L}_c^v}{\partial\boldsymbol{\omega}} = \begin{bmatrix}\frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{n}_w+[\mathbf{t}_{cw}+v\mathbf{d}]_\times(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\boldsymbol{\omega}}\\ \frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\boldsymbol{\omega}}\end{bmatrix} \tag{17}$$

There are two forms: $[\boldsymbol{\omega}]\times\mathbf{b}$ and $\mathbf{A}[\boldsymbol{\omega}]\times\mathbf{b}$. The forms in the differentiation with respect to rotation and translation are the same. Here we will not expand it in detail.

Differentiate with respect to linear velocity:

$$\begin{aligned}\frac{\partial\mathbf{L}_c^v}{\partial\mathbf{d}} &= \begin{bmatrix}\frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{n}_w+[\mathbf{t}_{cw}+v\mathbf{d}]_\times(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{d}}\\ \frac{\partial((\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{d}}\end{bmatrix}\\ &= \begin{bmatrix}\frac{\partial(v[\mathbf{d}]_\times(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w)}{\partial\mathbf{d}}\\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}[v(\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\mathbf{a}_w]_\times\\ \mathbf{0}\end{bmatrix}_{6\times3}\end{aligned} \tag{18}$$

The second term in the parentheses:

$$\mathbf{L}_c^v = \begin{bmatrix}\mathbf{n}_c\\ \mathbf{a}_c\end{bmatrix} = \mathbf{N}_{cw}^v\mathbf{L}_w = \begin{bmatrix}(\mathbf{I}+v[\omega]_\times)\mathbf{R}_{cw} & [\mathbf{t}_{cw}+v\mathbf{d}]_\times(\mathbf{I}+v[\omega]_\times)\mathbf{R}_{cw}\\ 0 & (\mathbf{I}+v[\boldsymbol{\omega}]_\times)\mathbf{R}_{cw}\end{bmatrix}L_w \tag{19}$$

So we have:

$$\frac{\partial\mathbf{L}_c^v}{\partial\mathbf{L}_w} = \mathbf{N}_{cw}^v \tag{20}$$

The last term:

$$\begin{aligned}\frac{\partial\mathbf{L}_w}{\partial\delta\boldsymbol{\tau}} &= \begin{bmatrix}\frac{\partial\mathbf{L}_w}{\partial\psi_1} & \frac{\partial\mathbf{L}_w}{\partial\psi_2} & \frac{\partial\mathbf{L}_w}{\partial\psi_3} & \frac{\partial\mathbf{L}_w}{\partial\phi}\end{bmatrix}\\ &= \begin{bmatrix}0 & -W_1\mathbf{U}_3 & W_1\mathbf{U}_2 & -W_2\mathbf{U}_1\\ W_2\mathbf{U}_3 & 0 & -W_2\mathbf{U}_1 & W_1\mathbf{U}_2\end{bmatrix}_{6\times4}\end{aligned} \tag{21}$$

The derivation of the Jacobian matrices for the remaining error functions is similar to this one, except for the differentiation of the first term Eq. (10) with respect to the curve parameters.

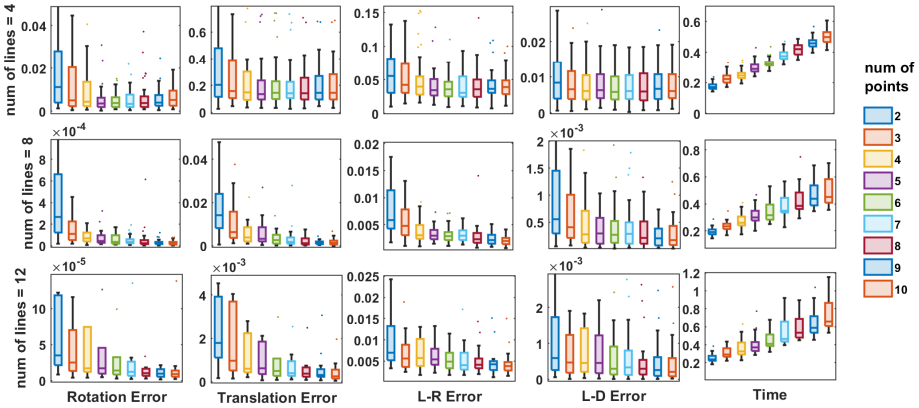

Fig. 1: The impact of the number of points on the curve on the accuracy and time of bundle adjustment.

3 The impact of the number of feature lines and points.

In Sec 3.1, we explore the impact of the number of sampling points on the curve on the optimization results. In Sec 3.2, we verify the effect of the number of sampling lines on the error.

3.1 What is the optimal number of points to measure along a line?

In this section, we investigate the influence of the number of points taken on the curves on the accuracy of our algorithm. Firstly, we establish a predetermined number of lines in space and then conduct experiments by sequentially taking 2 to 10 points on the curve. We iterate this procedure 50 times and draw box plots of the empirical outcomes using varying quantities of points. We perform three sets of tests using constant numbers of lines in space: 4, 8, and 12. The results are displayed in Fig. 1. As the number of points sampled on the lines rises, the precision of the experimental results progressively enhances, albeit at the cost of increased time consumption. However, after the number of points on the lines reaches approximately 5, the enhancement in accuracy becomes less notable while the time consumption steadily increases. The reason for this is that the curve expression has only 9 unknowns, and each point can impose two constraints: slope and distance. Therefore, the line constraints can be maximally effective when there are about 5 points on the line. Hence, we recommend using 4 to 6 points on the curves as constraints.

3.2 What is the optimal number of lines to employ for RSL-BA?

Within this section, we shall examine the impact of the quantity of lines in 3D space on the accuracy of our algorithm. We fix the number of points taken on

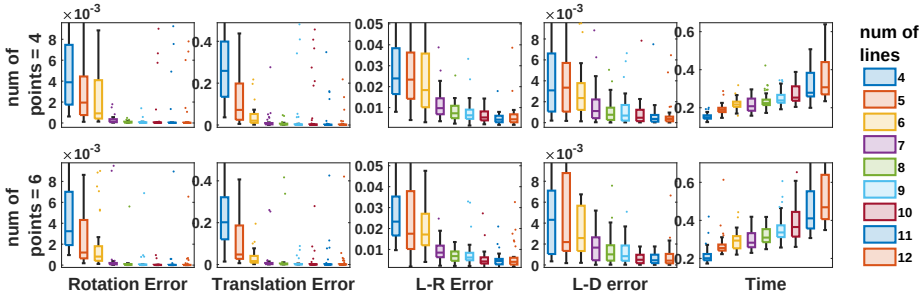

Fig. 2: The impact of the number of 3D lines in space on the accuracy and time of bundle adjustment.

Table 1: The median absolute trajectory error (ATE) of different methods on WHU-RSVI [3] dataset, the best results are highlighted by bold font.

	WHU-RSVI1	WHU-RSVI2	WHU-RSVI3	WHU-RSVI4
GSBA	0.080992	0.061310	0.030404	0.023698
GLBA	0.076173	0.065985	0.033554	0.024961
NMRSBA	0.050969	0.041629	0.042317	0.028183
NWRSBA	**0.040640**	0.045313	0.035451	0.022666
RSL-BA(ours)	0.0443502	**0.039314**	**0.023351**	**0.020675**

each curve and subsequently conduct experiments by sequentially arranging 4 to 12 lines in space. We iterate this process 50 times and draw box plots of the experimental outcomes using varying quantities of lines. We conduct two sets of experiments with a fixed number of points on the curves: 4 points and 6 points. The results are displayed in Fig. 2. An increase in the number of lines leads to a noticeable enhancement in the algorithm's accuracy, followed by a period of stabilization. Meanwhile, the time consumption consistently rises.

4 Complete results on the TUM-RSVI [9] and WHU-RSVI [3] dataset

We compare our method with two SOTA GS-based-method: *1)* GSBA [7], *2)* GLBA [10], and two SOTA RS-based-method: *1)* NMRSBA [1], *2)* NWRSBA [5]. The experiments are conducted on a laptop with an Intel i7 CPU and all algorithms are implemented in MATLAB.

4.1 Synthetic Images

In this section, we conduct experiments on input synthetic images. We use the WHU-RSVI [3] dataset, from which we select two sets of data from trajectory1-fast and trajectory2-fast for 3D reconstruction and pose estimation. We first employ [8] to detect RS curves by segmenting curves into multiple short-line segments and performing line fitting for initialization. The GS line-based SfM [6]

is applied to initialize the *RSL-BA* parameters. The comparative methods include GSBA, NMRSBA, NWRSBA, and GSLBA. Table 1 shows the median absolute trajectory error of different methods, it can be observed that the proposed *RSL-BA* method is the most stable one, achieving optimal or near-optimal results in all cases. Qualitative comparison are also provided in Fig. 3. Unlike point-based methods, line-based methods often achieve good results with fewer feature lines. However, the GSL-BA method is not sufficiently stable when RS effects are prominent.

4.2 Real Images

In this section, we conduct experiments on the real image dataset TUM-RSVI [9]. The experimental setup and comparison methods are similar to those in Sec. 4.1. Table 2 shows the median absolute trajectory error of different methods, tracer comparison plots is also provided in Fig 4, it can be observed that the RSL-BA method outperforms other methods but is slightly weaker than NWRSBA. This is because the TUM-RSVI lacks line features and they are not visually prominent, making it less suitable for RSL-BA. Besides, we implement a naive point-line RSBA by jointly optimizing points and lines with NWRSBA and proposed RSL-BA. An important observation is that such a naive NWRSBA+RSL-BA combination significantly outperforms each method individually, which implies that the proposed RSL-BA could be applied solely or combined with the point-based method as point-line BA to the downstream RS vision task such as RSSfM or RSSLAM.

Table 2: The median absolute trajectory error (ATE) of different methods in TUM-RSVI [9] dataset.the best results are highlighted by bold font.

	GSBA	GLBA	NMRSBA	NWRSBA	RSL-BA(ours)	NWRS+RSL-BA
seq1	0.069484	0.086460	0.052366	0.045064	0.037816	**0.034495**
seq2	0.029821	0.030379	0.026028	**0.023227**	0.025132	0.024754
seq3	0.065160	0.063718	0.057918	0.055001	0.048187	**0.045184**
seq4	0.049613	0.052026	0.032214	0.030534	0.031903	**0.027448**
seq5	0.031860	0.035839	0.019407	0.016066	0.017659	**0.015068**
seq6	0.061966	0.061792	0.032434	**0.024658**	0.026448	0.031414
seq7	0.051621	0.056534	0.039154	0.039620	0.039701	**0.033150**
seq8	0.026403	0.028690	0.024807	0.025926	0.024983	**0.024486**
seq9	0.098334	0.098212	0.082481	**0.073580**	0.080523	0.076613
seq10	0.81174	0.81180	0.59390	**0.53296**	0.57477	0.57417

References

1. Albl, C., Sugimoto, A., Pajdla, T.: Degeneracies in rolling shutter sfm. In: ECCV (2016)
2. Bartoli, A., Sturm, P.: Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. Computer vision and image understanding **100**(3), 416–441 (2005)

3. Cao, L., Ling, J., Xiao, X.: The whu rolling shutter visual-inertial dataset. IEEE Access **8**, 50771–50779 (2020)
4. He, Y., Zhao, J., Guo, Y., He, W., Yuan, K.: Pl-vio: Tightly-coupled monocular visual–inertial odometry using point and line features. Sensors **18**(4), 1159 (2018)
5. Liao, B., Qu, D., Xue, Y., Zhang, H., Lao, Y.: Revisiting rolling shutter bundle adjustment: Toward accurate and fast solution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4863–4871 (2023)
6. Liu, S., Yu, Y., Pautrat, R., Pollefeys, M., Larsson, V.: 3d line mapping revisited. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21445–21455 (2023)
7. Lourakis, M.I., Argyros, A.A.: Sba: A software package for generic sparse bundle adjustment. ACM Transactions on Mathematical Software (TOMS) **36**(1), 1–30 (2009)
8. Purkait, P., Zach, C., Leonardis, A.: Rolling shutter correction in manhattan world. In: ICCV. pp. 882–890 (2017)
9. Schubert, D., Demmel, N., von Stumberg, L., Usenko, V., Cremers, D.: Rolling-shutter modelling for direct visual-inertial odometry. In: IROS (2019)
10. Taylor, C.J., Kriegman, D.J.: Structure and motion from line segments in multiple images. IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(11), 1021–1032 (1995)

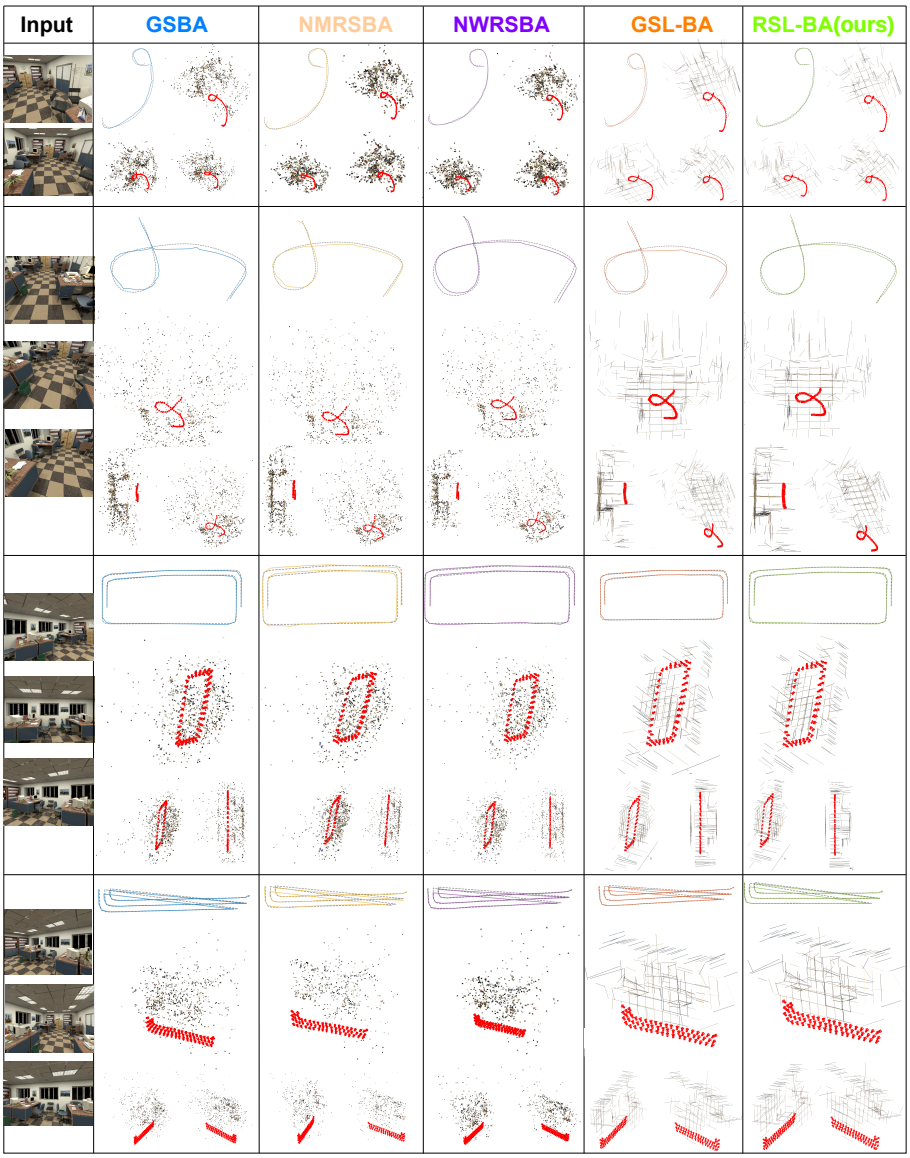

Fig. 3: Trajectories and 3D reconstruction comparison. Each column represents a different bundle adjustment algorithm, and each row represents a different sequence. It can be observed that our algorithm has smaller trajectory errors and better reconstruction results.

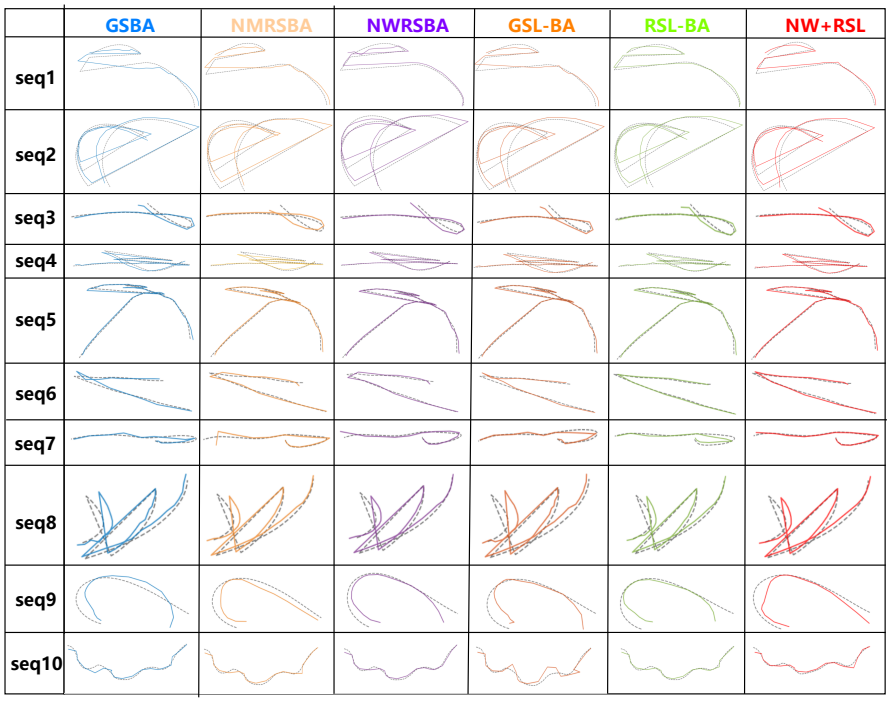

Fig. 4: Comparison of trajectory errors on the TUM-RSVI [9]. Each column represents a different bundle adjustment algorithm, and each row represents a different sequence.