Power Variable Projection for Initialization-Free Large-Scale Bundle Adjustment

– Supplemental Material –

Simon Weber[1,2] Je Hyeong Hong[3] Daniel Cremers[1,2]

[1] Technical University of Munich
[2] Munich Center for Machine Learning
[3] Department of Electronic Engineering, Hanyang University

This supplemental material is organized as follows:
Appendix A studies robustness of *PoVar* with respect to η and random initialization, as well as the scale of the considered problems.
Appendix B complements the theoretical justifications of *PoVar* and *RiPoBA*.
Appendix C briefly comments a recent follow-up formulation of pOSE error.
Appendix D addresses the metric upgrade stage, necessary to estimate the projective transformation and to get Euclidean reconstruction.
Appendix E gives more details about the 97 BAL problems used in our experiments.

A Robustness

We illustrate the robustness of our solver *PoVar* for solving the first stage with respect to the coefficient η in the pOSE formulation. Figure 1, Figure 2 and Figure 3 represent the performance profile for $\eta = 0.2$, $\eta = 0.3$ and $\eta = 0.4$, respectively. We conclude that expansion methods *PoBA* and *PoVar* are both very competitive for the largest tolerance $\tau = 0.01$ for all coefficients η, in line with

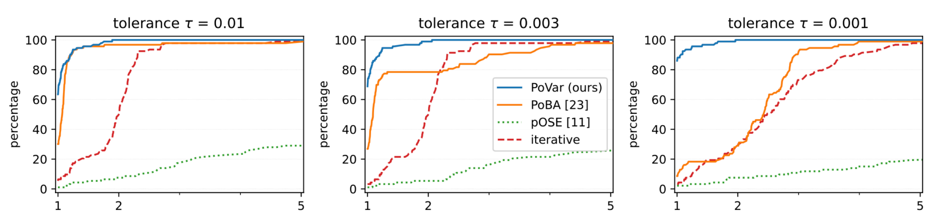

Fig. 1: With $\eta = 0.2$, performance profiles for all real-world BAL problems for solving the first stage (6). Given a tolerance $\tau \in \{0.01, 0.003, 0.001\}$, it represents the percentage of solved problems (y-axis) with relative runtime α (x-axis). Our solver *PoVar* is very competitive, and most notably for the highest accuracy $\tau = 0.001$ and $\tau = 0.003$.

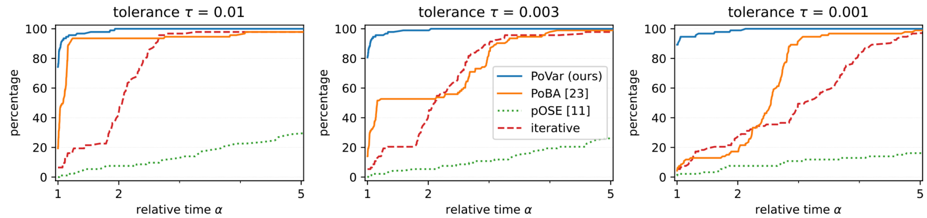

Fig. 2: With $\eta = 0.3$, performance profiles for all real-world BAL problems for solving the first stage (6).

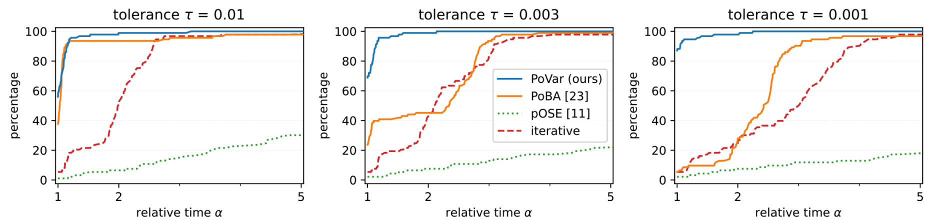

Fig. 3: With $\eta = 0.4$, performance profiles for all real-world BAL problems for solving the first stage (6).

our analysis in the main paper with $\eta = 0.1$. In particular, it outperforms *iterative*, and direct factorization (dashed green curves) shows very poor performance due to its lack of scalability. For highest accuracy $\tau = 0.003$ and $\tau = 0.001$, *PoVar* clearly outperforms all its competitors, in line with the main paper. Note that for each η, we have randomly selected a new set of 97 problems. We can also conclude from our ablation study that our analysis is robust to random initialization.

On the other hand, we link the speed-up of *PoVar* and *RiPoBA*, to the size of the problems. Figure 4 represents the performance profiles for solving the first two stages by considering only the BAL datasets with less than 1000 poses (small to medium-scale problems), and Figure 5 the performance profiles by considering only the BAL datasets with more than 1000 poses (large-scale problems).

B Proof of Lemma 1

The proof in Weber et al. [23] uses the positive-definiteness of U_λ and S – that still holds, to show that $\mu < 1$. It uses the positive-semi-definiteness of $U_\lambda^{-\frac{1}{2}} W V_\lambda^{-1} W^\top U_\lambda^{-\frac{1}{2}}$ to conclude that $\mu \geq 0$. For our VarPro formulation, we consider V_0 instead of V_λ. Note that the generalized Schur complement is written as: $S = U_\lambda - W V_0^\dagger W^\top$, where $V_0^\dagger$ is the (Moore-Penrose) pseudo-inverse of V_0. Nevertheless, it is straightforward that $U_\lambda^{-\frac{1}{2}} W V_0^\dagger W^\top U_\lambda^{-\frac{1}{2}}$ is also symmetric

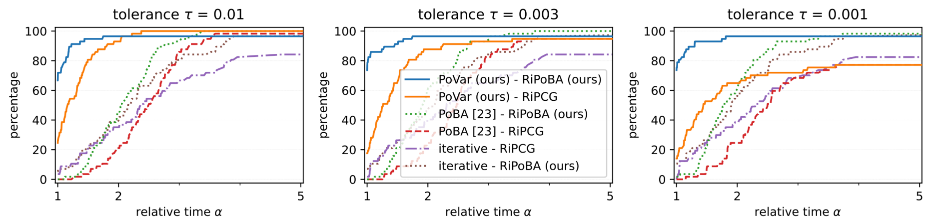

Fig. 4: Performance profiles for real-world BAL problems with less than 1000 poses for solving the first two stages Eq. (6) and Eq. (7).

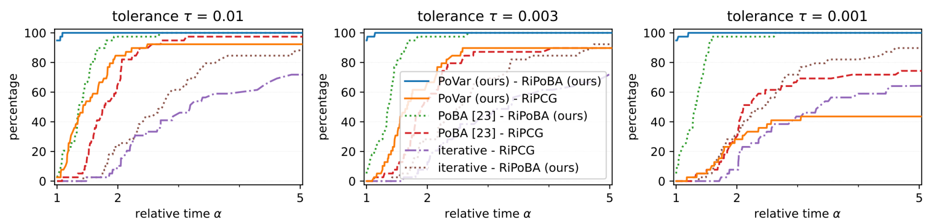

Fig. 5: Performance profiles for real-world BAL problems with more than 1000 poses for solving the first two stages Eq. (6) and Eq. (7).

positive semi-definite, and then the proof stays almost the same. In details, here is the adapted proof:

Proof. On the one hand $U_\lambda^{-\frac{1}{2}} W V_0^\dagger W^\top U_\lambda^{-\frac{1}{2}}$ is symmetric positive semi-definite, as U_λ is symmetric positive definite, and V_0 is symmetric positive semi-definite. Then its eigenvalues are greater than 0. As $U_\lambda^{-\frac{1}{2}} W V_0^\dagger W^\top U_\lambda^{-\frac{1}{2}}$ and $U_\lambda^{-1} W V_0^\dagger W^\top$ are similar,

$$\mu \geq 0\,. \tag{1}$$

On the other hand $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$ is symmetric positive definite as S and U_λ are. It follows that the eigenvalues of $U_\lambda^{-1} S$ are all strictly positive due to its similarity with $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$. As

$$U_\lambda^{-1} W V_0^\dagger W^\top = I - U_\lambda^{-1} S\,, \tag{2}$$

it follows that

$$\mu < 1\,, \tag{3}$$

that concludes the proof.

Concerning Riemannian manifold optimization framework, as the projection $x^\perp$ is full rank, it follows that $\tilde{U}_{\tilde{\lambda}}$ and $\tilde{V}_{\tilde{\lambda}}$ are symmetric positive-definite. Then, the previous proof can be very easily adapted to prove Lemma 2.

C pOSE Formulation

We extensively use the pOSE formulation [11] for testing our solvers. Recently, the follow-up expOSE formulation [14] has been proposed to override some limitations of pOSE. However, such formulation raises some issues for the scalability analysis. In addition to the fact that the authors wrongly claim that they use VarPro, expOSE requires an experimental preprocessing step over each image measurements. Without this first step, the exponential function is equal to 0 and the algorithm does not update. Nevertheless, such preprocessing is not feasible, in terms of runtime, when the considered dataset is large enough – which is the topic of our paper, where the number of observations goes up to several tens of millions. Although interesting, expOSE is so far limited to small-scale problems, in line with the problems used by the authors – between 19 and 30 poses in their core paper. Extending this pseudo object space error to large-scale formulation is an interesting research direction, orthogonal to our work.

That being said, note that our proposed solver *PoVar* can be used for solving generic nonlinear problems, and is not restricted to the pOSE formulation. In particular, a recent formulation RpOSE [29] extends pOSE to take into account unknown intrinsics and can be easily adapted to *PoVar*.

D Metric Upgrade

The third stage of pOSE [11] is the *autocalibration* step (see e.g. [30]), aiming to find an ambiguity matrix $H \in \mathbb{R}^{4\times 4}$ that forces the camera matrices to satisfy the $SE(3)$ constraints, that is to find H such that, for all poses i:

$$x_p^i H = x_p^i \begin{pmatrix} A & 0 \\ c^\top & 1 \end{pmatrix} \approx K_i[R_i t_i] \,, \tag{4}$$

where $\begin{pmatrix} c^\top & 1 \end{pmatrix}$ represents the plane at infinity. By denoting $\tilde{H}$ the three left-most columns of H, the $SE(3)$ constraint leads to

$$(K_i^{-1} x_p^i)\tilde{H}\tilde{H}^\top (K_i^{-1} x_p^i)^\top \approx I. \tag{5}$$

We find c and the camera scales α_i by solving:

$$\min_{c,\{\alpha_i\}} \sum_{i=1}^{n_p} \|\alpha_i (K_i^{-1} x_p^i)\tilde{H}(c)\tilde{H}(c)^\top (K_i^{-1} x_p^i)^\top - I\|_F^2 \,, \tag{6}$$

with the VarPro algorithm.

In particular, we use the chain rule and the following thorem [31]:

Theorem 1. *The derivative of $\tilde{H}\tilde{H}^\top$ with respect to $\tilde{H}$ is equal to:*

$$\frac{d\tilde{H}\tilde{H}^\top}{d\tilde{H}} = (I \otimes \tilde{H}^\top) + (\tilde{H}^\top \otimes I)T \,, \tag{7}$$

where T *is the matrix that transforms* $vec(\tilde{H})$ *in* $vec(\tilde{H}^\top)$*:*

$$Tvec(\tilde{H}) = vec(\tilde{H}^\top)\,, \tag{8}$$

and $vec(H)$ *is the operator that creates vector by stringing together the columns of* H*.*

E Dataset

	cameras	landmarks	observations
ladybug-49	49	7,766	31,812
ladybug-73	73	11,022	46,091
ladybug-138	138	19,867	85,184
ladybug-318	318	41,616	179,883
ladybug-372	372	47,410	204,434
ladybug-412	412	52,202	224,205
ladybug-460	460	56,799	241,842
ladybug-539	539	65,208	277,238
ladybug-598	598	69,193	304,108
ladybug-646	646	73,541	327,199
ladybug-707	707	78,410	349,753
ladybug-783	783	84,384	376,835
ladybug-810	810	88,754	393,557
ladybug-856	856	93,284	415,551
ladybug-885	885	97,410	434,681
ladybug-931	931	102,633	457,231
ladybug-969	969	105,759	474,396
ladybug-1064	1,064	113,589	509,982
ladybug-1118	1,118	118,316	528,693
ladybug-1152	1,152	122,200	545,584
ladybug-1197	1,197	126,257	563,496
ladybug-1235	1,235	129,562	576,045
ladybug-1266	1,266	132,521	587,701
ladybug-1340	1,340	137,003	612,344
ladybug-1469	1,469	145,116	641,383
ladybug-1514	1,514	147,235	651,217
ladybug-1587	1,587	150,760	663,019
ladybug-1642	1,642	153,735	670,999
ladybug-1695	1,695	155,621	676,317
ladybug-1723	1,723	156,410	678,421
	cameras	landmarks	observations
trafalgar-21	21	11,315	36,455

trafalgar-39	39	18,060	63,551
trafalgar-50	50	20,431	73,967
trafalgar-126	126	40,037	148,117
trafalgar-138	138	44,033	165,688
trafalgar-161	161	48,126	181,861
trafalgar-170	170	49,267	185,604
trafalgar-174	174	50,489	188,598
trafalgar-193	193	53,101	196,315
trafalgar-201	201	54,427	199,727
trafalgar-206	206	54,562	200,504
trafalgar-215	215	55,910	203,991
trafalgar-225	225	57,665	208,411
trafalgar-257	257	65,131	225,698

	cameras	landmarks	observations
dubrovnik-16	16	22,106	83,718
dubrovnik-88	88	64,298	383,937
dubrovnik-135	135	90,642	552,949
dubrovnik-142	142	93,602	565,223
dubrovnik-150	150	95,821	567,738
dubrovnik-161	161	103,832	591,343
dubrovnik-173	173	111,908	633,894
dubrovnik-182	182	116,770	668,030
dubrovnik-202	202	132,796	750,977
dubrovnik-237	237	154,414	857,656
dubrovnik-253	253	163,691	898,485
dubrovnik-262	262	169,354	919,020
dubrovnik-273	273	176,305	942,302
dubrovnik-287	287	182,023	970,624
dubrovnik-308	308	195,089	1,044,529
dubrovnik-356	356	226,729	1,254,598

	cameras	landmarks	observations
venice-52	52	64,053	347,173
venice-89	89	110,973	562,976
venice-245	245	197,919	1,087,436
venice-427	427	309,567	1,695,237
venice-744	744	542,742	3,054,949
venice-951	951	707,453	3,744,975
venice-1102	1,102	779,640	4,048,424
venice-1158	1,158	802,093	4,126,104
venice-1184	1,184	815,761	4,174,654
venice-1238	1,238	842,712	4,286,111
venice-1288	1,288	865,630	4,378,614
venice-1350	1,350	893,894	4,512,735

venice-1408	1,408	911,407	4,630,139
venice-1425	1,425	916,072	4,652,920
venice-1473	1,473	929,522	4,701,478
venice-1490	1,490	934,449	4,717,420
venice-1521	1,521	938,727	4,734,634
venice-1544	1,544	941,585	4,745,797
venice-1638	1,638	975,980	4,952,422
venice-1666	1,666	983,088	4,982,752
venice-1672	1,672	986,140	4,995,719
venice-1681	1,681	982,593	4,962,448
venice-1682	1,682	982,446	4,960,627
venice-1684	1,684	982,447	4,961,337
venice-1695	1,695	983,867	4,966,552
venice-1696	1,696	983,994	4,966,505
venice-1706	1,706	984,707	4,970,241
venice-1776	1,776	993,087	4,997,468
venice-1778	1,778	993,101	4,997,555
	cameras	landmarks	observations
final-93	93	61,203	287,451
final-394	394	100,368	534,408
final-871	871	527,480	2,785,016
final-961	961	187,103	1,692,975
final-1936	1,936	649,672	5,213,731
final-3068	3,068	310,846	1,653,045
final-4585	4,585	1,324,548	9,124,880
final-13682	13,682	4,455,575	28,973,703

Table 1: List of all 97 BAL problems [3] including number of cameras, landmarks and observations.

References

29. Iglesias, J.P., Olsson, C.: Radial distortion invariant factorization for structure from motion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5906–5915 (2021)
30. Pollefeys, M., Koch, R., Gool, L.V.: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. International Journal of Computer Vision **32**(1), 7–25 (1999)
31. Wang, X., Yang, W., Sun, B.: Derivatives of kronecker products themselves based on kronecker product and matrix calculus. Journal of Theoretical and Applied Information Technology **48**(1) (2013)