Supplementary Material for Deep geometry-aware camera self-calibration from video

Annika Hagemann[1,2], Moritz Knorr[1], Christoph Stiller[2]
[1]Bosch Research, Germany [2]Karlsruhe Institute of Technology
{annika.hagemann, moritzmichael.knorr}@de.bosch.com, stiller@kit.edu

A. Derivation of the intrinsics Jacobian

The optimization in the self-calibrating bundle adjustment layer requires the Jacobian of optimization residuals w.r.t. poses, depth and intrinsics. The Jacobians w.r.t. poses and depth were derived in [11]. In the following, we derive the Jacobian w.r.t. the intrinsics. The optimization residuals are given by

$$\mathbf{r}_{ij} = \mathbf{u}^*_{ij} - \boldsymbol{\pi}(\mathbf{G}_{ij} \circ \boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta}), \boldsymbol{\theta}), \tag{1}$$

where $\mathbf{u}_i$ denotes the image coordinates in image i and $\mathbf{u}^*_{ij}$ denotes the measured correspondences. $\mathbf{G}_{ij} \in SE(3)$ denotes the relative camera pose, z_i denotes the depth, and $\boldsymbol{\theta}$ are the intrinsic camera parameters. The function $\boldsymbol{\pi}$ describes the projection from 3D to the image, and $\boldsymbol{\pi}^{-1}$ the inverse projection.

As (1) contains direct and indirect dependencies of $\boldsymbol{\theta}$, we define $\mathbf{x}_{ij}(\boldsymbol{\theta}) := \mathbf{G}_{ij} \circ \boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta})$ and obtain the Jacobian w.r.t. $\boldsymbol{\theta}$ by using the total derivative:

$$\frac{d\mathbf{r}_{ij}}{d\boldsymbol{\theta}} = -\frac{d}{d\boldsymbol{\theta}}\boldsymbol{\pi}(\mathbf{x}_{ij}(\boldsymbol{\theta}), \boldsymbol{\theta}) \tag{2}$$

$$= -\frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \mathbf{x}_{ij}} \frac{d\mathbf{x}_{ij}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} - \frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{3}$$

To obtain $\frac{d\mathbf{x}_{ij}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}$, we further re-write the pose transformation $\mathbf{G}_{ij}$ in terms of a rotation matrix $\mathbf{R}_{ij}$ and a translation vector $\mathbf{t}_{ij}$, so that $\mathbf{x}_{ij}(\boldsymbol{\theta}) = \mathbf{R}_{ij}\boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta}) + \mathbf{t}_{ij}$. We then get

$$\frac{d\mathbf{x}_{ij}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \mathbf{R}_{ij} \frac{d\boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta})}{d\boldsymbol{\theta}} \tag{4}$$

Substituting this expression in (2) gives

$$\frac{d\mathbf{r}_{ij}}{d\boldsymbol{\theta}} = -\underbrace{\frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \mathbf{x}_{ij}}}_{(A)} \mathbf{R}_{ij} \underbrace{\frac{d\boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta})}{d\boldsymbol{\theta}}}_{(B)} - \underbrace{\frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}}_{(C)}. \tag{5}$$

This general expression holds independent of the choice of the camera model. The terms $(A), (B), (C)$, however, must be determined specifically for the given projection function.

For a pinhole camera model, for instance, projection and inverse projection are given by

$$\boldsymbol{\pi}(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix} \quad \text{and} \quad \boldsymbol{\pi}^{-1}(\mathbf{u}, \boldsymbol{\theta}, z) = z \begin{bmatrix} \frac{p_x - c_x}{f_x} \\ \frac{p_y - c_y}{f_y} \\ 1 \end{bmatrix}. \tag{6}$$

where $\mathbf{x} = (x, y, z)^\mathsf{T}$ is a 3D point, $\mathbf{u} = (p_x, p_y)^\mathsf{T}$ is an image point, (c_x, c_y) is the principal point and f_x, f_y are the focal lengths. By computing the individual derivatives, we obtain the following terms:

$$\frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \mathbf{x}_{ij}} = \begin{bmatrix} \frac{f_x}{z_{ij}} & 0 & -f_x \frac{x_{ij}}{z_{ij}^2} \\ 0 & \frac{f_y}{z_{ij}} & -f_y \frac{y_{ij}}{z_{ij}^2} \end{bmatrix} \tag{7}$$

$$\frac{d\boldsymbol{\pi}^{-1}(\mathbf{u}_i, z_i, \boldsymbol{\theta})}{d\boldsymbol{\theta}} = \begin{bmatrix} -z_i \frac{p_x - c_x}{f_x^2} & 0 & -\frac{z_i}{f_x} & 0 \\ 0 & -z_i \frac{p_y - c_y}{f_y^2} & 0 & -\frac{z_i}{f_y} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

$$\frac{\partial \boldsymbol{\pi}(\mathbf{x}_{ij}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{x_{ij}}{z_{ij}} & 0 & 1 & 0 \\ 0 & \frac{y_{ij}}{z_{ij}} & 0 & 1 \end{bmatrix} \tag{9}$$

These matrices can be substituted in Eq. (5) to obtain the overall Jacobian of the residuals w.r.t. the pinhole intrinsics.

B. Application on YouTube videos

As self-calibration enables performing 3D perception tasks without any knowledge about the camera, we applied DroidCalib on exemplary YouTube videos. From the YouTube8M dataset [1], we manually selected videos taken by a moving camera and extracted video snippets that did not contain any cuts. As all videos contained lens distortion, we used DroidCalib with UCM, initialized with naive intrinsics $\boldsymbol{\theta}_0$. The resulting reconstructions are shown in Fig. S1. Although a quantitative evaluation is not possible due to lack of ground-truth, the reconstructions are visually consistent (e.g. right angles, straight lines).

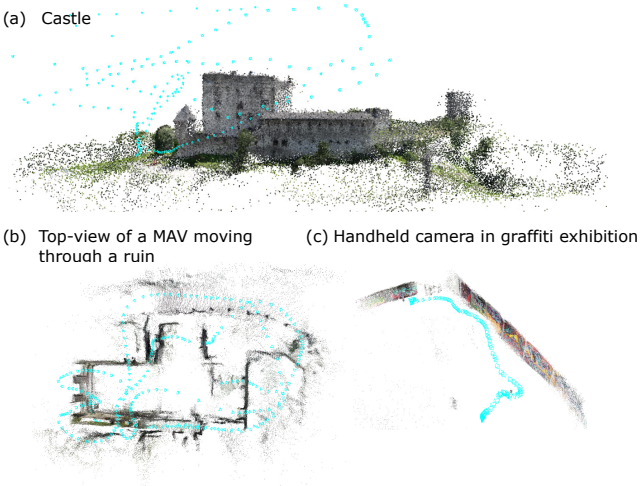

Figure S1. Qualitative results on YouTube videos. Cyan pyramids show the reconstructed camera poses. YouTube8M IDs are *mHQ5*, *RB1V*, *TnVx*, to obtain the associated video links, use YouTube8M.

C. Effect of different training setups

To investigate the effect of the specific training setup on the calibration accuracy, we compare the calibration accuracy achieved with four different setups (Fig. S2):

1. **Weights A** are the weights obtained through training with SC-BA layer, exposing the model to random initial intrinsics errors $\boldsymbol{\Delta\theta}$ during training, and using a dedicated intrinsics loss L_θ. These are the weights used for all main analyses.

2. **Weights B** are obtained through training with SC-BA layer, including random initial intrinsics errors $\boldsymbol{\Delta\theta}$, but without explicit intrinsics loss L_θ.

3. **Weights C** are obtained through training with SC-BA layer, without explicit intrinsics loss L_θ, and without exposing the model to random initial intrinsics errors $\boldsymbol{\Delta\theta}$ during training.

4. **Weights D** are the pre-trained weights from DROID-SLAM [11], obtained through training with the original BA layer, and without exposing the model to random initial intrinsics errors $\boldsymbol{\Delta\theta}$ during training. The SC-BA layer is thus only integrated at inference time.

Evaluation is performed on validation sequences from the TartanAir dataset[1], using naive initial intrinsics $\boldsymbol{\theta}_0$, and using random sequence snippets of different length (Fig. S2). The results suggest that for sufficiently long sequences ($N_\mathcal{I} = 500$), all training setups result in high calibration accuracy. Even the weights obtained through training for pose estimation only (**weights D**) yield accurate intrinsics when combined with the SC-BA layer at inference time. For short sequences ($N_\mathcal{I} = 50$), however, the results obtained with the different training setups differ. The model weights obtained through training with SC-BA layer (**weights A-C**) result in higher calibration accuracy than the weights obtained through training without SC-BA layer (i.e. the pre-trained **weights D**). Furthermore, Fig. S2 indicates that the intrinsics loss L_θ, and the exposure to intrinsics errors $\boldsymbol{\Delta\theta}$ during training are beneficial for the calibration accuracy on short sequences. Overall, this suggests that for long sequences, calibration is little sensitive to modifications in the training setup, whereas for short sequences, a dedicated intrinsics training improves the convergence to accurate intrinsics.

[1]Sequences neighborhood/Easy/P021, abandonedfactory/Hard/P011, office/Hard/P007, westerndesert/Easy/P013, gascola/Hard/P009.

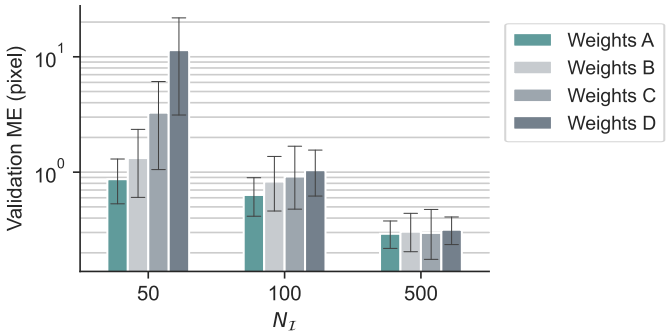

Figure S2. Comparison of different training setups depending on the length $N_\mathcal{I}$ of the input sequence. Evaluation is performed on random sequence snippets from the TartanAir validation sequences. Barplot shows average mapping error and 95% bootstrap confidence intervals across sequence snippets.

D. Performance depending on sequence length

We analyzed calibration accuracy, runtime and memory consumption depending on the length of the input sequence. To this end, we applied DroidCalib on random sequence snippets of different length (Fig. S3). For TartanAir and EuRoC, a mapping error below 1 pixel is achieved even with short sequences, containing less than 200 images. Then, the median peak memory consumption is between 7 GB and 10 GB and the computation time is below two minutes. For the more difficult TUM sequences, the mapping error remains at a higher level, even for sequences containing 900 images. This indicates that it is difficult to compensate the quality of a sequence (amounts of artifacts, amount and diversity of camera motion, structure in the scene) with a larger quantity of images.

E. Additional details on the baseline evaluation

DroidCalib and DROID-SLAM DroidCalib builds upon the DROID-SLAM [11] open source implementation (https://github.com/princeton-vl/DROID-SLAM, commit

Method	ME (pixel)		Runtime	Failures
COLMAP sequential	9.69	[2.42, 24.95]	2 min [1 min, 6 min]	4 / 9
COLMAP exhaustive	4.26	[2.46, 6.89]	2 h 19 min [46 min, 2 days 18 h 15 min]	0 / 9
COLMAP+NetVLAD	6.54	[2.52, 52.1]	4 min [2 min, 36 min]	0 / 9
COLMAP+NetVLAD+Superpoint+Superglue	4.10	[1.66, 8.09]	13 min [5 min, 51 min]	2 / 9

Table S1. Results of the COLMAP-based approaches on the TUM dataset, using different hyperparameters. We report median [min, max] over all converged sequences of the respective method. Failed sequences are only excluded from the method with the failure; not from all sequences. In general, there is a trade-off between runtime and accuracy. Although exhaustive matching leads to most accurate results, the runtime takes up to several days. In the main paper, we report COLMAP+NetVLAD and COLMAP+NetVLAD+Superpoint+Superglue, as they provide a balance between accuracy and runtime. For this supplementary analysis, we only used the TUM dataset, as it contains the shortest sequences and therefore enabled the evaluation of exhaustive matching.

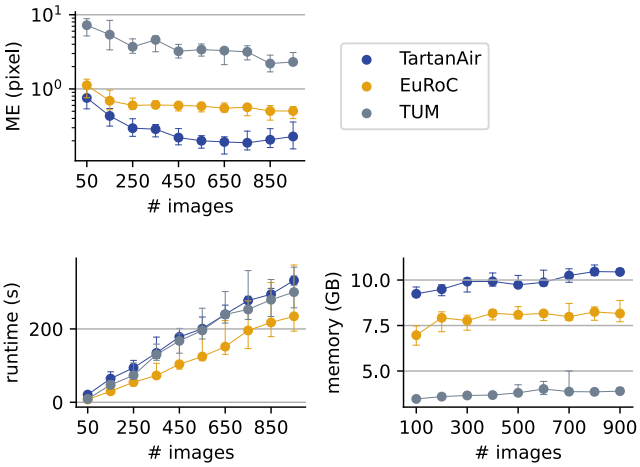

Figure S3. Calibration accuracy, runtime and peak memory consumption of DroidCalib depending on the number of input images. Plots show median values and 95% bootstrap confidence intervals on random sequence snippets of different length, with $\Delta_{\boldsymbol{\theta}} = 25\%$.

8016d2b9b72). A full estimation consists of (i) running the frontend until all images of the sequence have been registered, (ii) running a total of 19 iterations of the backend, and (iii) running a motion-only bundle adjustment to estimate the poses of all images that had not been selected as keyframes. DroidCalib performs the self-calibrating bundle adjustment in both, frontend and backend. We did not tune any hyperparameters of the SLAM system specifically for DroidCalib, but used the values proposed in DROID-SLAM for evaluating both, DROID-SLAM and DroidCalib. For DROID-SLAM, we found that the reproduced ATE values deviate slightly from the published values, though not systematically. We report the reproduced rather than the published values so that differences to DroidCalib can be fully attributed to the SC-BA layer.

Target-based calibration We used the calibration datasets provided in the EuRoC and the TUM datasets. Both datasets contain more than 500 images of a checkerboard target, captured using different relative poses between camera and target. To assess the accuracy of target-based calibration, we randomly sampled calibration datasets of different sizes (20, 50 and 100 images) from the available checkerboard images and performed an undistortion using the reference intrinsics. Thereby, only pinhole intrinsics must be estimated during calibration, giving a fair comparison with the corresponding DroidCalib results. We used a corner detection algorithm [9] to detect the image coordinates of the checkerboard corners. Following Zhang's method [12], we then used the direct linear transform to obtain initial values for the target poses and estimated the intrinsics by minimizing the reprojection error

$$\epsilon_{\text{res}}^2 = \sum_{\nu \in \mathcal{F}} \sum_{c \in \mathcal{C}} ||\mathbf{u}_{c\nu} - \boldsymbol{\pi}(\mathbf{G}_\nu \circ \mathbf{x}_c, \boldsymbol{\theta})||^2, \qquad (10)$$

where $\mathcal{F}$ denotes the set of calibration images, $\mathcal{C}$ denotes the set of visible checkerboard corners, $\mathbf{u}_{c\nu}$ are the detected image coordinates of checkerboard corners, $\mathbf{x}_c$ are the 3D coordinates of the checkerboard corners w.r.t. the board coordinate system, and $\mathbf{G}_\nu$ is the relative pose between board and camera in image ν. To reduce the impact of potential outliers, the error terms are weighted with a Cauchy kernel, and optimization is performed using the Levenberg Marquardt algorithm.

COLMAP-based approaches We used COLMAP 3.8 and hloc version 1.1 [8] and activated the estimation of all intrinsics during reconstruction. We used the same image sizes as for DroidCalib (see main paper) and we did not skip any frames. In the default setup, we used SIFT [6] feature extraction and performed feature matching using the nearest neighbor ratio configuration from hloc [8]. For feature extraction with Superpoint [3], we used the trained model from the InLoc dataset [10], as provided in hloc [8].
The performance of COLMAP-based approaches depends on the image pairs considered during feature matching and there is a trade-off between accuracy and runtime. Tab. S1 shows the the calibration accuracy using different possible configurations, including sequential matching with a window of five images, NetVLAD and exhaustive matching.

Dataset	Setting	ME (pixel)		Runtime
TartanAir	50 epochs	18.3	[5.0, 60.1]	28 min
	350 epochs	14.9	[2.95, 64.8]	3 h 08 min
	full dataset	3.52	[–, –]	5 h 36 min
EuRoC	50 epochs	27.6	[14.0, 56.2]	53 min
	350 epochs	24.1	[11.1, 47.4]	6 h 11 min
	full dataset	6.48	[–, –]	8 h 19 min
TUM	50 epochs	29.7	[17.6, 44.5]	25 min
	350 epochs	24.7	[16.2 , 63.6]	3 h 35 min
	full dataset	4.31	[–, –]	2 h 51min
EuRoC raw	50 epochs	10.8	[1.63, 47.9]	53 min
	350 epochs	10.7	[1.81 , 55.9]	6 h 05 min
	full dataset	2.24	[–, –]	8 h 27min

Table S2. Results of SelfSup-Calib [4] using different settings. We report median [min, max] over all sequences in the respective dataset. For the "full dataset" setting we report the single result obtained from training 50 epochs jointly on all sequences.

SelfSup-Calib We used the author's implementation to evaluate the method [4]. We left all hyperparameters untouched and we chose the unified camera model. The parametrization of the unified camera model in [4] is slightly different from the parametrization in [7], however, both parametrizations are mathematically equivalent and can be converted into each other (see [4]). As proposed in [4], the EuRoC images were resized to an input size of 256×384. As the TUM dataset and the TartanAir dataset were not evaluated in the original work, we used the same hyperparameters.
The work originally trained jointly on multiple EuRoC sequences. To obtain per-sequence results, we trained on the individual sequences. We also tried to compensate for the lower number of optimization steps by increasing the number of epochs from 50 to 350 (Tab. S2). This leads to slightly higher accuracy, but comes at the cost of a significant increase in the computation time. We further evaluated the method when training jointly on all sequences in the respective dataset (Tab. S2). This significantly increases the calibration accuracy, indicating that the method benefits from larger datasets and longer training times, as compared to the single-sequence setup assessed in this work.

On the choice of baseline methods Besides [4], the approaches in [5, 2], also use a CNN-based architecture to regress poses and depth from unknown cameras, and train in a self-supervised manner using a photometric loss. However, while [4] *learns* the intrinsics of the training data, [5, 2] *infer* intrinsics on a two-image basis, which is much more challenging, so that accuracy is by design expected to be lower (this is supported by Fig. 9 in [5]). In one experiment, [5] additionally *learns* the intrinsics, which we expect to yield similar accuracy as [4], because it relies on the same principle and only differs in the exact model architecture and the choice of the camera model. Among the two, we decided to use [4] as a representative approach, as it provides complete source code. In general, it must be noted that there are no well-established benchmark datasets and metrics to compare different self-calibration approaches. We thus hope that our work is a step towards establishing such a benchmark.

F. More detailed experimental results

Tabs. S3-S5 contain the per-sequence results of the experiments shown in the main paper.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. 1

[2] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072, 2019. 4

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 3

[4] Jiading Fang, Igor Vasiljevic, Vitor Guizilini, Rares Ambrus, Greg Shakhnarovich, Adrien Gaidon, and Matthew R Walter. Self-supervised camera self-calibration from video. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8468–8475. IEEE, 2022. 4

[5] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 4

[6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 3

[7] Christopher Mei and Patrick Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950. IEEE, 2007. 4

[8] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 3

[9] Tobias Strauss, Julius Ziegler, and Johannes Beck. Calibrating multiple cameras with non-overlapping views using coded checkerboard targets. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 2623–2628. IEEE, 2014. 3

[10] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 3

[11] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34:16558–16569, 2021. 1, 2, 6

[12] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000. 3

Seq	$\Delta_{\boldsymbol{\theta}} = 0.0$		$\Delta_{\boldsymbol{\theta}} = 0.25$	
	DroidSLAM	DroidCalib	DroidSLAM	DroidCalib
TartanAir				
ME000	**0.126**	0.408	1.983	**0.205**
ME001	0.041	**0.039**	1.667	**0.032**
ME002	0.467	**0.403**	11.463	**0.230**
ME003	**0.595**	1.428	14.143	**0.904**
ME004	**0.577**	1.509	7.296	**1.669**
ME005	0.189	**0.071**	3.639	**0.094**
ME006	1.732	**0.902**	2.071	**1.310**
ME007	0.076	**0.064**	4.618	**0.064**
MH000	**0.040**	0.077	33.391	**0.136**
MH001	0.634	**0.033**	0.711	**0.041**
MH002	0.030	**0.018**	2.284	**0.021**
MH003	**0.019**	0.031	0.563	**0.014**
MH004	3.012	**1.601**	**1.757**	2.738
MH005	0.622	**0.432**	8.779	**0.400**
MH006	0.392	**0.104**	2.924	**0.215**
MH007	**0.065**	0.154	6.380	**0.136**
Median	0.291	**0.129**	3.282	**0.170**
EuRoC				
MH_01	**0.011**	**0.011**	0.187	**0.011**
MH_02	0.018	**0.012**	0.182	**0.012**
MH_03	0.022	**0.021**	1.746	**0.021**
MH_04	0.043	**0.039**	0.437	**0.039**
MH_05	**0.040**	0.041	0.398	**0.040**
V1_01	0.036	**0.031**	0.148	**0.031**
V1_02	0.012	**0.011**	0.398	**0.011**
V1_03	0.024	**0.013**	0.269	**0.015**
V2_01	0.016	**0.012**	1.053	**0.013**
V2_02	**0.009**	0.079	0.166	**0.041**
V2_03	0.013	**0.011**	0.909	**0.011**
Median	0.018	**0.013**	0.398	**0.015**
TUM				
360	**0.061**	0.079	**0.160**	0.179
desk	**0.018**	0.019	0.083	**0.019**
desk2	**0.026**	0.069	0.109	**0.055**
floor	0.022	**0.017**	0.234	**0.017**
room	**0.043**	0.889	0.748	**0.548**
xyz	**0.010**	0.011	0.023	**0.011**
rpy	**0.023**	0.026	0.044	**0.026**
plant	**0.017**	0.018	0.170	**0.018**
teddy	0.035	**0.034**	0.137	**0.034**
Median	**0.023**	0.026	0.137	**0.026**

Table S3. Average trajectory error (ATE) in meters of DroidCalib and DROID-SLAM [11] for different initial errors in the intrinsics. Values show medians over three runs per sequence, using $\Delta_{\boldsymbol{\theta}} = 0\%$ and $\Delta_{\boldsymbol{\theta}} = 25\%$. For DROID-SLAM, we report the reproduced results rather than the values originally published in [11], so that differences can be fully attributed to the self-calibrating bundle adjustment layer. The table is associated with Fig. 6 in the main paper.

	$\mathbf{f_x}$	$\mathbf{f_y}$	$\mathbf{c_x}$	$\mathbf{c_y}$	**ME** (pixel)
TartanAir Ground-truth	**320.0**	**320.0**	**320.0**	**240.0**	
ME000	320.2	320.3	320.0	240.1	0.12
ME001	320.2	320.6	320.2	240.4	0.20
ME002	320.6	320.7	320.5	241.0	0.34
ME003	320.9	321.0	320.4	240.3	0.43
ME004	321.8	321.5	320.7	239.9	0.79
ME005	320.5	320.4	320.3	240.0	0.24
ME006	320.8	320.6	320.5	240.5	0.35
ME007	319.4	319.3	320.2	240.3	0.30
MH000	320.2	320.2	320.3	240.0	0.09
MH001	318.7	318.4	320.4	240.4	0.67
MH002	320.2	320.1	320.2	240.2	0.11
MH003	320.3	320.8	319.8	240.1	0.26
MH004	320.5	320.4	320.2	239.4	0.23
MH005	320.0	320.0	320.4	240.2	0.08
MH006	320.1	320.0	320.4	240.7	0.13
MH007	320.4	320.2	320.2	240.1	0.17
					median: 0.23
EuRoC Ground-truth	**458.7**	**457.3**	**367.2**	**248.4**	
MH_01	457.9	457.9	368.0	249.1	0.28
MH_02	457.9	457.1	367.8	248.5	0.25
MH_03	457.9	458.2	368.0	250.2	0.34
MH_04	457.4	457.2	367.7	249.1	0.38
MH_05	458.3	457.8	367.9	248.6	0.16
V1_01	459.1	459.0	368.2	250.0	0.42
V1_02	459.4	459.4	367.9	249.7	0.49
V1_03	459.6	459.5	368.3	249.4	0.55
V2_01	459.2	459.7	368.2	249.7	0.52
V2_02	459.5	459.8	367.8	249.9	0.58
V2_02	459.5	459.9	367.8	250.0	0.59
V2_03	460.2	460.1	368.2	249.5	0.74
					median: 0.42
TUM Ground-truth	**517.3**	**516.5**	**318.6**	**255.3**	
360	530.2	529.4	320.4	261.1	3.71
desk	531.7	527.5	321.0	248.4	3.70
desk2	525.6	553.6	317.4	284.0	6.93
floor	530.5	518.4	323.0	240.2	3.09
room	523.2	526.1	320.9	261.8	2.22
xyz	523.4	504.0	325.3	281.2	3.03
rpy	531.2	532.7	323.7	276.2	4.66
plant	525.4	527.3	319.5	254.3	2.58
teddy	523.1	519.6	324.1	249.3	1.50
					median: 3.09

Table S4. Accuracy of intrinsics estimated using DroidCalib. Vaules show estimated focal lengths f_x, f_y and principal point c_x, c_y, and the mapping error ME. This table is associated with Tab. 2 in the main paper and contains all evaluated sequences.

	DroidCalib	COLMAP+N	COLMAP+N +SP+SG	SelfSup-Calib
TartanAir				
ME000	**0.12**	1.46	0.29	7.32
ME001	**0.20**	1.01	1.34	8.72
ME002	0.34	0.28	**0.23**	11.03
ME003	**0.43**	4.19	1.35	14.59
ME004	**0.79**	1.76	10.77	8.86
ME005	**0.24**	—	0.36	22.04
ME006	0.35	245.58	**0.25**	5.00
ME007	0.30	**0.11**	0.20	48.64
MH000	**0.09**	0.86	0.59	8.30
MH001	0.67	1349.43	**0.60**	50.90
MH002	**0.11**	0.24	0.44	43.41
MH003	**0.26**	2.42	0.47	60.09
MH004	**0.23**	2.39	0.51	57.18
MH005	**0.08**	0.45	0.19	13.32
MH006	**0.13**	20.56	0.31	50.86
MH007	**0.17**	0.78	0.64	26.86
EuRoC				
MH_01	**0.28**	0.38	1.15	27.63
MH_02	**0.25**	2.50	1.27	22.60
MH_03	**0.34**	0.45	4.11	39.65
MH_04	**0.38**	8.76	0.71	16.13
MH_05	**0.16**	21.20	0.55	21.14
V1_01	**0.42**	1.02	0.71	14.03
V1_02	0.49	1.13	**0.48**	32.27
V1_03	**0.55**	2.20	1.25	56.15
V2_01	**0.52**	1.77	1.19	24.01
V2_02	**0.58**	9.65	0.70	49.49
V2_03	0.74	1.16	**0.42**	28.94
TUM				
360	**3.71**	52.14	—	17.58
desk	**3.70**	6.54	3.97	22.81
desk2	6.93	**2.53**	5.02	23.04
floor	**3.09**	10.07	8.09	29.73
room	**2.22**	4.01	4.10	37.32
xyz	**3.03**	19.16	—	33.12
rpy	**4.66**	4.28	5.45	18.99
plant	2.58	8.11	**1.66**	32.78
teddy	**1.50**	2.76	2.43	44.53
EuRoC raw				
MH_01	**0.32**	2.92	1.07	6.12
MH_02	**0.31**	3.08	0.66	1.63
MH_03	**0.33**	3.60	3.89	14.93
MH_04	**0.35**	31.10	6.14	9.09
MH_05	**0.36**	11.17	3.11	7.08
V1_01	**0.40**	5.10	3.83	47.88
V1_02	**0.43**	7.58	5.25	29.20
V1_03	**0.56**	3.73	2.05	5.94
V2_01	**0.80**	—	4.66	10.96
V2_02	**0.69**	2.03	3.25	10.78
V2_03	**0.73**	3.15	3.48	24.58

Table S5. Per-sequence results of the mapping error of self-calibration baselines. This table is associated with Tab. 1 in the main paper and shows the mapping error (ME) in units of pixels. Abbreviations are COLMAP+NetVLAD (COLMAP+N), COLMAP+NetVLAD+Superpoint+Superglue (COLMAP+N+SP+SG).