A Supplementary Material

A.1 Video

The accompanying video shows the evolution (iterations) of the proposed event-only bundle adjustment method on multiple sequences (both synthetic and real).

A.2 Problem unknowns, Operating Point and Perturbation

The unknowns of the problem are the camera trajectory $\mathtt{R}(t)$ and the gradient map of the scene $\boldsymbol{G} \doteq \nabla M$. According to the chosen parameterization (Sec. 3.2), the perturbations of the camera pose at time t (not necessarily a control pose) and the gradient map are:

$$\mathtt{R}(t) = \exp(\delta\boldsymbol{\varphi}^{\wedge})\mathtt{R}_{\text{op}}(t), \tag{12}$$

$$\boldsymbol{G} = \boldsymbol{G}_{\text{op}} + \varDelta\boldsymbol{G}, \tag{13}$$

where we use the exponential map (notation from [4]). The "operating point" (abbreviated "op") consists of the current camera trajectory (parameterized by N_{poses} control poses) and the map (e.g., gradient brightness values):

$$\mathbf{P}_{\text{op}} = \{\mathtt{R}_1^{\text{op}}, \ldots, \mathtt{R}_{N_{\text{poses}}}^{\text{op}}, \boldsymbol{\beta}_1^{\text{op}}, \ldots, \boldsymbol{\beta}_{N_p}^{\text{op}}\}. \tag{14}$$

To linearize the errors for the Gauss-Newton / Levenberg-Marquardt algorithm, we consider pose perturbations in the Lie-group sense (control poses in the Lie group and perturbations in the Lie algebra [4]), and pixel perturbations in gradient brightness space. That is, camera control poses and map pixels are perturbed according to

$$\mathtt{R}_i = \exp(\boldsymbol{\delta\phi}_i^{\wedge})\mathtt{R}_i^{\text{op}}, \tag{15}$$

$$\boldsymbol{\beta}_n = \boldsymbol{\beta}_n^{\text{op}} + \delta\boldsymbol{\beta}_n. \tag{16}$$

A.3 Linearization of Error Terms (Analytical Derivatives)

Perturbing the camera motion and the scene map we aim to arrive at an expression like:

$$\mathbf{e} \approx \mathbf{e}_{\text{op}} + \mathtt{J}_{\text{op},\boldsymbol{\alpha}}\varDelta\mathbf{P}_{\boldsymbol{\alpha}} + \mathtt{J}_{\text{op},\boldsymbol{\beta}}\varDelta\mathbf{P}_{\boldsymbol{\beta}}, \tag{17}$$

where $\mathtt{J}_{\text{op},\boldsymbol{\alpha}} \doteq \frac{\partial\mathbf{e}}{\partial\mathbf{P}_{\boldsymbol{\alpha}}}\Big|_{\text{op}}$ and $\mathtt{J}_{\text{op},\boldsymbol{\beta}} \doteq \frac{\partial\mathbf{e}}{\partial\mathbf{P}_{\boldsymbol{\beta}}}\Big|_{\text{op}}$. Thus, we only consider the first-order terms (i.e., discard higher order ones). Here, $\mathtt{J}_{\text{op},\boldsymbol{\alpha}}$ is an $N_e \times 3N_{\text{poses}}$ matrix, and $\mathtt{J}_{\text{op},\boldsymbol{\beta}}$ is an $N_e \times 2N_p$ matrix, where N_e is the number of events and N_p is the number of valid panorama pixels.

Let us write the linearization of each error term in (17). Given the error entry from the problem (5)-(6):

$$(\mathbf{e})_k \doteq \boldsymbol{G}\big(\mathbf{p}(t_k)\big) \cdot \varDelta\mathbf{p}(t_k) - s_k C. \tag{18}$$

After some calculations, we have:

$$
\begin{aligned}
(\mathbf{e})_k \approx\ & (\boldsymbol{G}(\mathbf{p}_{\text{op}}(t_k)) - \nabla \boldsymbol{G}_{\text{op}}(\mathbf{p}_{\text{op}}(t_k))\mathtt{E}_{\text{op}}(t_k)\delta\boldsymbol{\varphi}(t_k) + \varDelta\boldsymbol{G}(\mathbf{p}_{\text{op}}(t_k))) \\
& \cdot (\varDelta\mathbf{p}_{\text{op}} - (\mathtt{E}_{\text{op}}(t_k)\delta\boldsymbol{\varphi}(t_k) - \mathtt{E}_{\text{op}}(t_k - \varDelta t_k)\delta\boldsymbol{\varphi}(t_k - \varDelta t_k))) \\
& - s_k C \qquad (19) \\
\approx\ & \underbrace{\boldsymbol{G}(\mathbf{p}_{\text{op}}(t_k)) \cdot \varDelta\mathbf{p}_{\text{op}} - s_k C}_{\text{this is } (\mathbf{e}_{\text{op}})_k} + \underbrace{\varDelta\mathbf{p}_{\text{op}}^{\top}\varDelta\boldsymbol{G}(\mathbf{p}_{\text{op}}(t_k))}_{\text{linear in } \varDelta\mathbf{P}_{\boldsymbol{\beta}}} \\
& \underbrace{-\varDelta\mathbf{p}_{\text{op}}^{\top}\nabla\boldsymbol{G}_{\text{op}}(\mathbf{p}_{\text{op}}(t_k))\mathtt{E}_{\text{op}}(t_k)\delta\boldsymbol{\varphi}(t_k)}_{\text{linear in } \varDelta\mathbf{P}_{\boldsymbol{\alpha}}} \\
& \underbrace{-\boldsymbol{G}(\mathbf{p}_{\text{op}}(t_k)) \cdot (\mathtt{E}_{\text{op}}(t_k)\delta\boldsymbol{\varphi}(t_k) - \mathtt{E}_{\text{op}}(t_k - \varDelta t_k)\delta\boldsymbol{\varphi}(t_k - \varDelta t_k))}_{\text{linear in } \varDelta\mathbf{P}_{\boldsymbol{\alpha}}}, \qquad (20)
\end{aligned}
$$

where

$$
\begin{aligned}
\varDelta\mathbf{p}_{\text{op}}(t_k) &\doteq \mathbf{p}_{\text{op}}(t_k) - \mathbf{p}_{\text{op}}(t_{k-1}) && (21)\\
\mathtt{E}_{\text{op}}(t) &\doteq \left.\frac{\partial\pi}{\partial\mathbf{z}}\right|_{\mathbf{z}_{\text{op}}} \mathbf{z}_{\text{op}}^{\wedge} && (22)\\
\pi &\ \text{ is the equirectangular projection } \mathbb{R}^3 \to \mathbb{R}^2 && (23)\\
\mathbf{z}(t) &= \mathtt{R}(t)\mathtt{K}^{-1}\mathbf{x}^h && (24)\\
\mathbf{z}_{\text{op}}(t) &\doteq \mathtt{R}^{\text{op}}(t)\mathtt{K}^{-1}\mathbf{x}^h && (25)\\
\mathbf{x}^h &= (x, y, 1)^{\top} \text{are the homogeneous coordinates of point } \mathbf{x} && (26)\\
^{\wedge} &\ \text{ is the hat (skew-symmetric) operator [4]} && (27)\\
\delta\boldsymbol{\varphi} &\ \text{ is the perturbation of } \mathtt{R}(t_k) && (28)\\
\delta\tilde{\boldsymbol{\varphi}} &\ \text{ is the perturbation of } \mathtt{R}(t_k - \varDelta t_k) && (29)\\
\nabla\boldsymbol{G} &\doteq \nabla^2 M_{\text{op}} \ \text{ is the second-order spatial derivative of } M_{\text{op}} && (30)
\end{aligned}
$$

Note that $\delta\tilde{\boldsymbol{\varphi}}$ will use the two control poses closest to time $t_k - \varDelta t_k$, which may not necessarily be the same ones as those of $\delta\boldsymbol{\varphi}$ (at time t_k).

In therms of the problem unknowns, equation (20) states that the predicted (linearized) contrast in (4) depends on: the event camera orientations at two different times $\{t_k,\, t_k - \varDelta t_k\}$ and the first two spatial derivatives of brightness at one pixel location $\mathbf{p}(t_k)$.

A.4 Cumulative Formation of the Normal Equations

A key step of the Levenberg-Marquardt (LM) solver is forming the normal equations. Regarding EMBA, the size of the full Jacobian matrix $\mathtt{J}_{\text{op}}$ in (7) is $N_e \times (3N_{\text{poses}} + 2N_p)$. In general, an event data sequence has millions of events, while N_p is usually in the order of thousands. Hence, the memory needed to compute and store $\mathtt{J}_{\text{op}}$ is unaffordable for normal PCs. To this end, we avoid computing and storing the full $\mathtt{J}_{\text{op}}$. Instead, we directly compute the left-hand side (LHS) matrix $\mathtt{A} \doteq \mathtt{J}_{\text{op}}^{\top}\mathtt{J}_{\text{op}}$ and the right-hand side (RHS) vector $\mathbf{b} \doteq -\mathtt{J}_{\text{op}}^{\top}\mathbf{e}_{\text{op}}$, in a cumulative manner.

LHS Matrix A Let $\mathbf{r}_k^\top$ be the k-th row of $\mathtt{J}_{\text{op}}$, which stores the derivatives of an error term $(\mathbf{e})_k$. With the partitioning in (9), we can further write $\mathbf{r}_k^\top = (\mathbf{r}_{k,\boldsymbol{\alpha}}^\top, \mathbf{r}_{k,\boldsymbol{\beta}}^\top)$, where $\mathbf{r}_{k,\boldsymbol{\alpha}}$ and $\mathbf{r}_{k,\boldsymbol{\beta}}$ are the camera pose part and map part of $\mathbf{r}_k$, respectively. Then we can rewrite the LHS matrix as the sum of the outer product of each row:

$$\mathtt{A} \doteq \mathtt{J}_{\text{op}}^\top \mathtt{J}_{\text{op}} = \sum_{k=1}^{N_e} \mathbf{r}_k \mathbf{r}_k^\top = \sum_{k=1}^{N_e} \begin{pmatrix} \mathbf{r}_{k,\boldsymbol{\alpha}} \mathbf{r}_{k,\boldsymbol{\alpha}}^\top & \mathbf{r}_{k,\boldsymbol{\alpha}} \mathbf{r}_{k,\boldsymbol{\beta}}^\top \\ \mathbf{r}_{k,\boldsymbol{\beta}} \mathbf{r}_{k,\boldsymbol{\alpha}}^\top & \mathbf{r}_{k,\boldsymbol{\beta}} \mathbf{r}_{k,\boldsymbol{\beta}}^\top \end{pmatrix}. \tag{31}$$

Let $\mathtt{A}_{11k} \doteq \mathbf{r}_{k,\boldsymbol{\alpha}} \mathbf{r}_{k,\boldsymbol{\alpha}}^\top$, $\mathtt{A}_{12k} \doteq \mathbf{r}_{k,\boldsymbol{\alpha}} \mathbf{r}_{k,\boldsymbol{\beta}}^\top$, and $\mathtt{A}_{22k} \doteq \mathbf{r}_{k,\boldsymbol{\beta}} \mathbf{r}_{k,\boldsymbol{\beta}}^\top$. They are the contributions of $(\mathbf{e})_k$ to the LHS matrix $\mathtt{A}$. Then (31) becomes:

$$\mathtt{A} = \sum_{k=1}^{N_e} \mathtt{A}_k = \sum_{k=1}^{N_e} \begin{pmatrix} \mathtt{A}_{11k} & \mathtt{A}_{12k} \\ \mathtt{A}_{12k}^\top & \mathtt{A}_{22k} \end{pmatrix}. \tag{32}$$

It shows that the contribution of each event to $\mathtt{A}$ is additive, which offers a cumulative way to form the LHS matrix $\mathtt{A}$. As mentioned at the end of Appendix A.3, an error term depends on map gradients at one map point (nearest neighbor). This leads to a block-diagonal sparsity pattern of $\mathtt{A}_{22k}$, which significantly speeds up solving the normal equations.

RHS Vector b Similarly, let $\mathbf{c}_n$ be the n-th column of $\mathtt{J}_{\text{op}}$. With the partitioning in (9), we can rewrite $\mathtt{J}_{\text{op}}$ as

$$\mathtt{J}_{\text{op}} = \left(\mathbf{c}_{1,\boldsymbol{\alpha}},\ \ldots,\ \mathbf{c}_{3N_{\text{poses}},\boldsymbol{\alpha}},\ \mathbf{c}_{1,\boldsymbol{\beta}},\ \ldots,\ \mathbf{c}_{2N_p,\boldsymbol{\beta}}\right), \tag{33}$$

where $\mathbf{c}_{i,\boldsymbol{\alpha}} = \left.\frac{\partial \mathbf{e}}{\partial \mathbf{P}_{i,\boldsymbol{\alpha}}}\right|_{\text{op}}$ and $\mathbf{c}_{j,\boldsymbol{\beta}} = \left.\frac{\partial \mathbf{e}}{\partial \mathbf{P}_{j,\boldsymbol{\beta}}}\right|_{\text{op}}$ store the derivatives of the whole error vector $\mathbf{e}$ with respect to each component of the pose/map state. Substituting (33) into the RHS of (8), we obtain the cumulative formula of each entry of $\mathbf{b}$:

$$\begin{aligned} \mathbf{b}_{1i} &= -\mathbf{c}_{i,\boldsymbol{\alpha}}^\top \mathbf{e}_{\text{op}} = -\sum_{k=1}^{N_e} \left.\frac{\partial (\mathbf{e})_k}{\partial \mathbf{P}_{i,\boldsymbol{\alpha}}}\right|_{\text{op}} (\mathbf{e}_{\text{op}})_k \\ \mathbf{b}_{2j} &= -\mathbf{c}_{j,\boldsymbol{\beta}}^\top \mathbf{e}_{\text{op}} = -\sum_{k=1}^{N_e} \left.\frac{\partial (\mathbf{e})_k}{\partial \mathbf{P}_{j,\boldsymbol{\beta}}}\right|_{\text{op}} (\mathbf{e}_{\text{op}})_k. \end{aligned} \tag{34}$$

where $\frac{\partial (\mathbf{e})_k}{\partial \mathbf{P}_{\boldsymbol{\alpha} i}}$ and $\frac{\partial (\mathbf{e})_k}{\partial \mathbf{P}_{\boldsymbol{\beta} j}}$ are the derivatives of the error term $(\mathbf{e})_k$ with respect to the i/j-th component of the pose/map states.

Equations (32) and (34) allow us to accumulate the contribution of each event to the normal equations (8), so that we can omit forming $\mathtt{J}_{\text{op}}$. The size of $\mathtt{A}$ only depends on the dimension of state parameters, i.e., $(3N_{\text{poses}} + 2N_p)^2$, which is significantly smaller than that of $\mathtt{J}_{\text{op}}$, i.e., $N_e \times (3N_{\text{poses}} + 2N_p)$.

A.5 Sensitivity and Ablation Analyses

We characterize the sensitivity of EMBA with respect to some of its parameters and also show the effect of a robust loss function. In the following, the map size is 1024×512 px, the initial rotations come from CMax-$\boldsymbol{\omega}$, and the sequence used is *bicycle*.

Contrast Threshold Firstly, we run EMBA with varying values of $C = \{0.05, 0.1, 0.2, 0.5, 1.0\}$ in the loss function, where $C = 0.2$ is the true value for *bicycle*. We set $f = 20$ Hz and $\eta = 5.0$. Note that the value of C affects the value of the PhE. Therefore, for a meaningful comparison, we use the PhE at $C = 0.2$ as reference and calculate the equivalent PhE for the other C values. The results are presented in Tab. 6. The closer C is to 0.2, the smaller the PhE. The trials of $C = \{0.1, 0.2\}$ achieve smaller rotation errors than the others. Nevertheless, the trials of $C = \{0.05, 0.5, 1.0\}$ still show a strong refinement effect, in terms of both ARE and PhE (with respect to 1.69° ARE and $5.5 \cdot 10^5$ PhE, in Tabs. 2 and 3), which implies that EMBA is robust to the choice of C. This is important in applications because the contrast thresholds of real event cameras are difficult to obtain and may vary greatly within the same dataset [41].

Table 6: Sensitivity analysis on the camera's contrast threshold C. Top: absolute rotation error (ARE), in RMSE form. Bottom: equivalent squared photometric error.

C	0.05	0.1	0.2	0.5	1.0
ARE [°]	1.193	0.899	0.923	0.966	1.341
Equivalent PhE [$\cdot 10^5$]	3.024	2.956	2.956	2.968	3.030

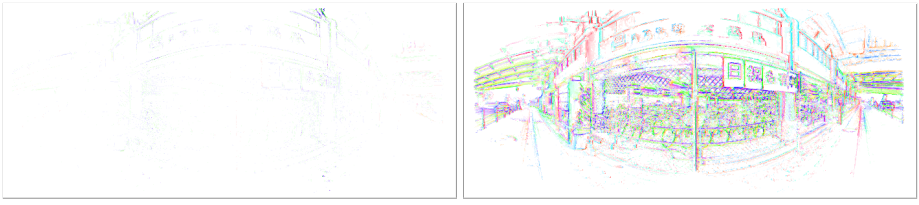
Weight of L^2 Regularization We run EMBA with different values of $\eta = \{0, 0.1, 0.5, 1.0, 5.0, 10.0, 20.0\}$ while setting $C = 0.2$ and $f = 20$ Hz. The results are shown in Tab. 7. When $\eta = 0$, i.e., disabling the L^2 regularization, the resulted gradient map is shown in Fig. 9a, where a few pixels dominate the optimization, thus suppressing the update of other pixels. Meanwhile, it reports the worst ARE and PhE values among all η values (Tab. 7). This reveals that the L^2 regularization is essential, and it effectively encourages a good convergence (like in Fig. 9b). As η increases from 0.1 to 5.0, both ARE and PhE decrease smoothly until they achieve their best values at $\eta = 5.0$; afterwards they increase with η. Empirically, $\eta = 5.0$ is a good choice in most cases.

Robust Loss Function The formula of the Huber loss function is:

$$\rho(u) = \begin{cases} u^2 & \text{for } |u| < \delta, \\ (2|u| - \delta)\,\delta, & \text{otherwise.} \end{cases} \tag{35}$$

Table 7: Sensitivity analysis on the weight of L^2 regularization η.

η	0	0.1	0.5	1.0	5.0	10.0	20.0
ARE [°]	1.527	1.295	1.301	1.222	0.923	1.032	1.086
Equivalent PhE [$\cdot 10^5$]	3.160	3.053	3.049	3.032	2.956	3.015	3.071



(a) $\eta = 0$. **(b)** $\eta = 5$.

Fig. 9: Effect of L^2 regularization on the refined gradient map.

We apply it to each error term, $u = (\mathbf{e})_k$, thus replacing the data-fidelity cost $\sum_k((\mathbf{e})_k)^2$ in (6), (10) by $\sum_k \rho((\mathbf{e})_k)$. In the experiments, we set $C = 0.2$, $f = 20$ Hz, $\eta = 5.0$ and $\delta = 0.1$.

Tables 8 and 9 compare the Quadratic and Huber cost functions in terms of rotation error and PhE on synthetic and real-world data, respectively. For a fair comparison, we present the squared PhE for both Quadratic and Huber loss.

ARE: On synthetic data, the Huber loss function results in slightly better rotation error than the Quadratic one in most trials, with only three exceptions. All error differences are less than 0.35 degrees. On real-world data it is hard to analyze the impact of the Huber loss function on rotation accuracy due to the inherent evaluation problems (explained at the beginning of Sec. 4.3).

PhE: On the other hand, the refined PhE of the Huber loss is a little bigger than that of the Quadratic loss on most synthetic and real-world sequences. This is a predictable result, because the objective function of the Huber loss has changed to a new "reweighted" squared PhE, where the weights of the outliers are reduced.

In addition to Tabs. 8 and 9, we show a qualitative result here (more are available in the accompanying video). Figure 10 compares the refined maps produced by the quadratic and Huber loss functions. The Huber panorama is similar and slightly sharper than the quadratic one.

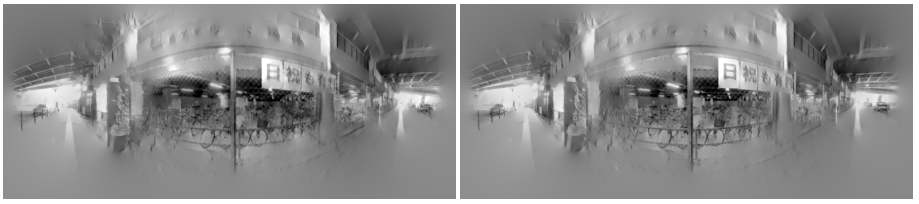
Control Pose Frequency We run EMBA to refine the same initial rotations and maps, but varying the control pose frequency $f = \{10, 20, 50, 100\}$ Hz. $C = 0.2$ is set to its true value and $\eta = 5.0$. The results are reported in Tab. 10. It turns out that EMBA is also robust to the choice of f. As f grows from 10 to 50 Hz, both ARE and PhE decrease slightly and reach a minimum at $f = 50$ Hz. When f is increased to 100 Hz, the errors grow marginally, which implies that a too high f does not lead to a better refinement.

Table 8: Absolute rotation RMSE [deg] (ARE) and squared photometric error [$\times 10^6$] (PhE) on *synthetic* sequences [20] (Schur solver, 1024 × 512 px map).

		EKF-SMT			CMax-GAE			CMax-$\boldsymbol{\omega}$		
	Sequence	before	Quad	Huber	before	Quad	Huber	before	Quad	Huber
ARE	playroom	5.86	6.09	6.15	4.63	4.42	4.32	3.22	2.86	2.79
	bicycle	1.47	1.18	1.01	1.65	1.50	1.41	1.69	0.92	0.97
	city	1.69	1.68	1.39	–	N/A	N/A	1.53	0.97	0.94
	street	3.44	3.46	3.23	–	N/A	N/A	0.97	0.74	0.74
	town	4.32	4.40	4.23	4.66	4.53	4.44	1.91	0.86	1.21
	bay	2.50	2.41	2.30	–	N/A	N/A	1.80	1.41	1.39
PhE	playroom	0.35	0.23	0.26	0.35	0.19	0.21	0.33	0.15	0.18
	bicycle	0.52	0.30	0.32	0.53	0.31	0.34	0.55	0.30	0.33
	city	2.62	2.13	2.19	–	N/A	N/A	2.71	1.98	2.11
	street	1.82	1.52	1.50	–	N/A	N/A	1.90	1.34	1.43
	town	1.88	1.51	1.62	1.90	1.54	1.65	1.92	1.43	1.55
	bay	2.26	1.96	1.95	–	N/A	N/A	2.30	1.83	1.98

Table 9: Absolute rotation RMSE [deg] (ARE) and squared photometric error [$\times 10^6$] (PhE) on *real* sequences [30] (Schur solver, 1024 × 512 px map).

		RTPT			CMax-GAE			CMax-$\boldsymbol{\omega}$		
	Sequence	before	Quad	Huber	before	Quad	Huber	before	Quad	Huber
ARE	shapes	2.19	2.85	2.62	2.51	2.69	2.61	4.11	4.44	4.13
	poster	3.80	3.96	3.99	3.63	4.09	4.16	4.07	4.20	4.13
	boxes	1.74	2.32	2.26	2.02	2.40	2.32	3.22	2.87	2.92
	dynamic	2.00	2.29	2.40	1.70	2.00	1.97	3.13	2.79	2.80
PhE	shapes	0.68	0.37	0.52	0.61	0.38	0.50	0.58	0.36	0.50
	poster	4.69	2.58	2.88	5.03	3.07	3.30	4.37	2.58	2.87
	boxes	4.46	2.30	2.43	4.52	2.93	2.99	3.92	2.25	2.42
	dynamic	3.29	2.24	2.37	3.16	2.39	2.71	3.05	2.13	2.30



(a) Quadratic. **(b)** Huber.

Fig. 10: Effect of robust loss function. Refined maps obtained with (a) Quadratic and (b) Huber loss functions. (*bicycle* sequence, initialized by CMax-$\boldsymbol{\omega}$ trajectory).

Table 10: Sensitivity analysis on the control pose frequency f.

f [Hz]	10	20	50	100
ARE [°]	0.984	0.923	0.890	1.112
PhE [$\cdot 10^5$]	3.120	2.956	2.926	2.929

A.6 Additional Discussion of the Experiments

Front-end failures In the experiments, four different front-end methods are used to initialize EMBA. RTPT fails on all synthetic sequences and EKF-SMT fails on all real-world ones. The explanation is as follows: RTPT loses track due to its limitation on the range of camera rotations that can be tracked. It monitors the tracking quality during operation and stops updating the map when the quality decreases below a threshold, which offen happens if the camera's FOV gets close to the left or right boundaries of the panoramic map. The tracking failure of EKF-SMT happens mostly when the camera changes the rotation direction abruptly. We suspect it is due to the error propagation between the tracking and mapping threads. Small errors in the poses or the map are amplified, corrupting the states and their uncertainty in the respective Bayesian filters.

Camera translation in ECD datasets In Sec. 4, we mentioned that the four sequences from the ECD dataset [30] were recorded by a hand-held event camera, so the camera motion inevitably contains translations, which affects all involved front-end methods as well as our BA approach. Figure 11 displays the translational component of the GT poses provided by the mocap. It shows that the magnitude of the translation grows, as time progresses and the speed of the motion increases. We use the first part of the sequences, where the translational motion is still small (about less than 10 cm) for the desk-sized scenes.

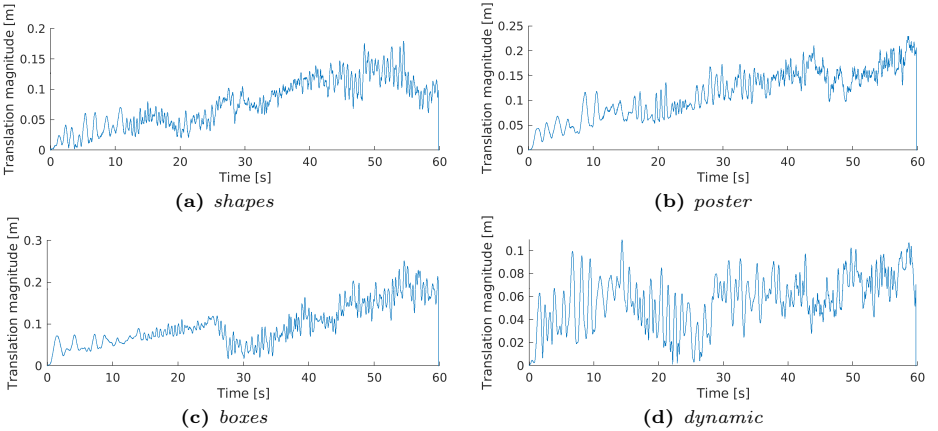


Fig. 11: From the motion capture system: groundtruth camera translation magnitude of the four ECD sequences [30].