On-Device Self-Supervised Learning of Low-Latency Monocular Depth from Only Events

Supplementary Material

https://mavlab.tudelft.nl/depth_from_events

6. Extra qualitative results

DSEC. We present additional qualitative results from the DSEC test set in Fig. 8. While the self-supervised results exhibit less sharp boundaries, they are free from artifacts commonly introduced by supervised learning, such as discontinuities at image borders and around thin objects.

Robot experiments. Figs. 9 to 12 show extended qualitative results for four unseen scenes from flight experiments. Looking at Fig. 9, we see that network's ability to maximize the contrast of the image of warped events increases quickly from "step 0" (only pre-training) to "step 3k" (pre-training + 100 seconds of learning). While this improvement in terms of contrast maximization loss may be mostly due to just learning the correct magnitude of the optical flow (as shown by the change from orange to purple in column 3, and black to red in column 5) through scaling depth and ego-motion, subsequent learning steps improve the depth map in more sophisticated ways, judging from the disappearance of the wrong "depth gap" in the center of the disparity images in the third and fourth column. Similar priorities in learning patterns can be seen in the other scenes.

7. Implementation details

Training. For offline training on datasets, we train for 50 epochs with the Adam optimizer and a learning rate of 1e-4. For contrast maximization, we accumulate 10 bins of events, and warp all events to all bin edges. Furthermore, we set the weight for the geometric consistency loss $\lambda = 0.05$. For on-device learning, we lower the learning rate to 1e-5. Specifics per dataset are mentioned below. In all cases, event streams are undistorted and rectified.

For MVSEC, we train on `outdoor_day2` with input bins of 20 ms of events. We use a batch size of 8, and augment the data with polarity and left-right flips. Training takes around 50 minutes on an RTX 4090.

For DSEC, we train on the daylight sequences in the training set (`interlaken_00_*` and `zurich_city_{04,05,06,07,08,11}_*`). We leave `thun_00_a` for validation. Because of DSEC's high event density and large frames, we lower the batch size to 4, and bin events to 10 ms frames with a cap of 100k events per bin (if there are more events, we end the bin prematurely; we do not discard events). In addition to polarity and left-right flips, we augment by reversing the time dimension, as we saw that this lessened border artifacts with wrong optical flows. Training takes around 11 hours on an RTX 4090.

For UZH-FPV, we train on the forward indoor sequences (`indoor_forward_{3,5,6,7,9}_davis_with_gt` and `indoor_forward_{8,11,12}_davis`). Sequence `indoor_forward_10_davis_with_gt` is left for validation. We use 10 ms bins of events, a batch size of 8 and polarity and left-right flips. Training takes approximately 30 minutes on an RTX 4090.

Network architecture. We make use of a small convolutional recurrent network to predict depth and ego-motion. The encoder and memory backbone are shared between the depth and ego-motion decoders. Tab. 3 lists the details per layer. In addition, we make use of ELU activations as we experienced dying ReLUs. Also, to prevent border artifacts, we use reflect padding for all convolutional layers.

Depth-based control. We slice depth maps into $K = 8$ vertical bins, compute a vector of average inverse depths (disparities) $\boldsymbol{d} \in \mathbb{R}^8$, and use it to set a target yaw rate. Furthermore, $\lambda_{\text{goal}} = 0.2$, $\lambda_{\text{avoid}} = 1.0$, $\alpha = 0.5$, $\sigma = 12.0$.

Drone setup. We built a 5-inch quadrotor for our robot experiments. The drone is equipped with on-board sensors that provide the flight controller with all relevant information to follow high-level control commands. More specifically, the EKF running on the Kakute H7 Mini flight controller fuses IMU measurements with velocity and height measurements coming from an MTF-01 optical flow/range sensor into a stable position and velocity estimate. This allows a neural network (like our depth network) to give high-level commands like velocity setpoints or rotational rates.

All relevant components on the drone, along with their weight and power consumption, can be found in Tab. 4. Communication on the Orin is handled with ROS2 [25], which also allows for logging to rosbags, and can be connected via UART to the internal publish-subscribe messaging API of the PX4 flight controller firmware.

The power consumption during flight is measured by keeping track of the total mAh consumed by over two flight tests. Together with the measured power consumption of the Jetson using `jtop` and the expected maximum power draw of both cameras, this allows us to calculate the power consumption of the drone (see Tab. 4).

	Layer type	Input shape	Output shape	# Parameters
Encoder	`Conv2D(ksize=7, stride=2)`	$(2, H, W)$	$(16, H/2, W/2)$	1,584
	`ResidualConv2D(stride=2)`	$(16, H/2, W/2)$	$(32, H/4, W/4)$	18,528
	`ResidualConv2D(stride=2)`	$(32, H/4, W/4)$	$(64, H/8, W/8)$	73,920
Memory	`ConvGRU`	$(64 + 64, H/8, W/8)$	$(64, H/8, W/8)$	221,568
Depth	`Conv2D`	$(64, H/8, W/8)$	$(64, H/8, W/8)$	36,928
	`Conv2D(bias=False)` w/ `SoftPlus`	$(64, H/8, W/8)$	$(1, H/8, W/8)$	576
	`Upsample(scale=8, "bilinear")`	$(1, H/8, W/8)$	$(1, H, W)$	
Ego-motion	`Conv2D(stride=2)`	$(64, H/8, W/8)$	$(64, H/16, W/16)$	36,928
	`Conv2D(stride=2)`	$(64, H/16, W/16)$	$(64, H/32, W/32)$	36,928
	`Conv2D(bias=False)` w/ `Identity`	$(64, H/32, W/32)$	$(6, H/32, W/32)$	3,456
	`AdaptiveAvgPool2D`	$(6, H/32, W/32)$	$(6, 1, 1)$	

Table 3. Network layer details. Unless specified otherwise, we use ELU activations, and a kernel size of 3, biases and “reflect” padding for convolutional layers. Total parameter count is 430,416.

Component	**Product**	**Mass [g]**	**$\sim$Power [W]**
Frame	Armattan Marmotte 5 inch		
Motors	Emax Eco II Series 2306		
Propellers	Ethix S5 5 inch		
Flight controller	Holybro Kakute H7 Mini	455	$200^{\dagger}$
Optical flow & range sensor	MicoAir MTF-01		
ESC	Holybro Tekko32 F4 4in1 mini 50A BL32		
Receiver	Radiomaster RP2 V2 ELRS Nano		
Battery	iFlight Fullsend 4S 3000mAh Li-Ion	208	-
On-board compute	NVIDIA Jetson Orin NX 16GB & DAMIAO v1.1 carrier board	62	9^{*}
Event camera	iniVation DVXplorer Micro	22	max $0.7^{\ddagger}$
Stereo camera	Intel RealSense D435i	75	max $3.5^{\ddagger}$
Total	-	822	213.2

Table 4. List of hardware components used during robot experiments. Power consumption estimates are obtained from Jetson’s `jtop`*, battery drain during flight experiments†, or component datasheets‡.

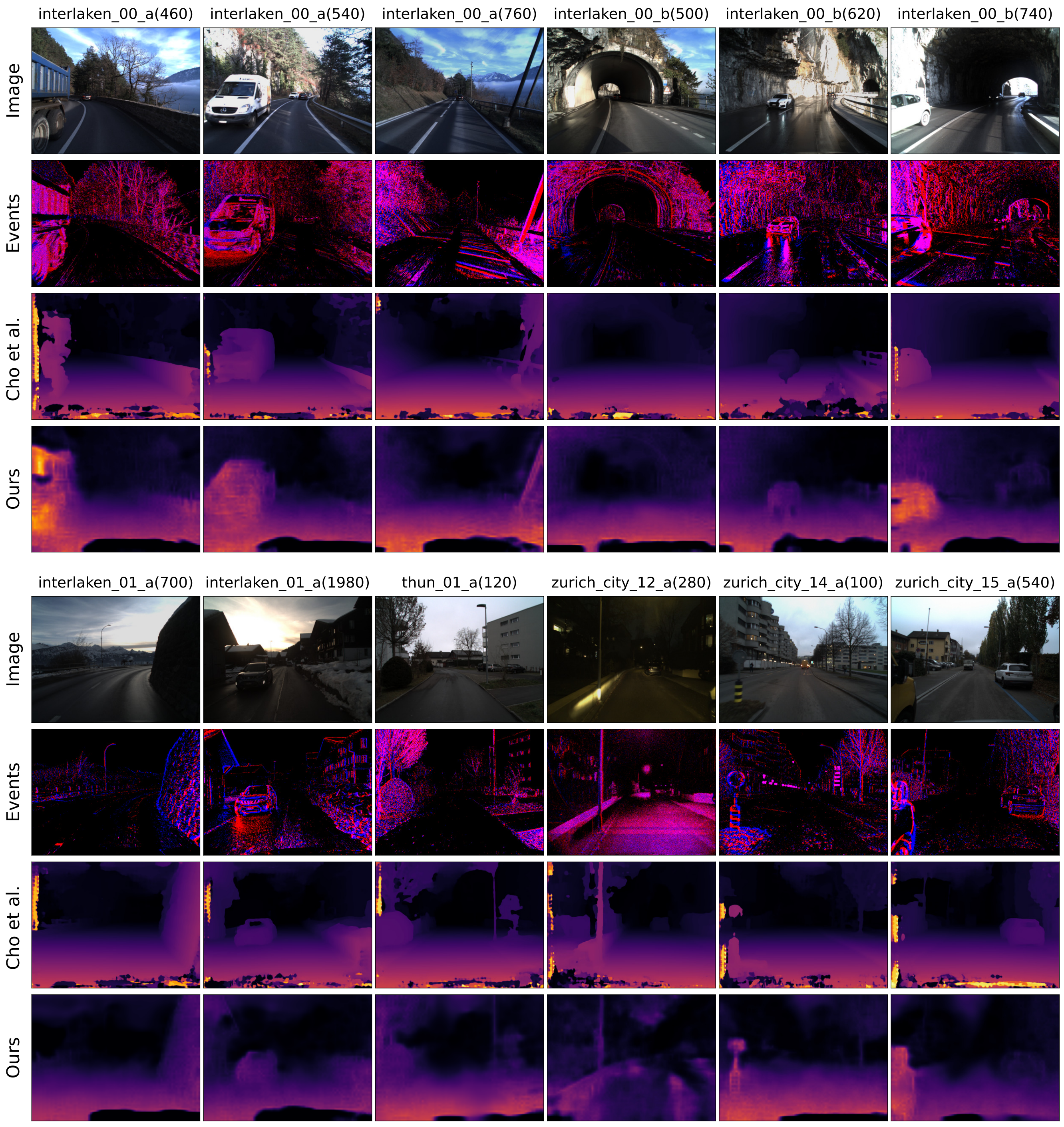

Figure 8. Additional qualitative results of disparity predictions on the DSEC disparity benchmark. Images are for visualization only, as disparity estimation is event-based. The same color map is applied to the disparity values from the stereo- and supervised-learning-based method from Cho *et al.* [8] and ours for easy comparison.

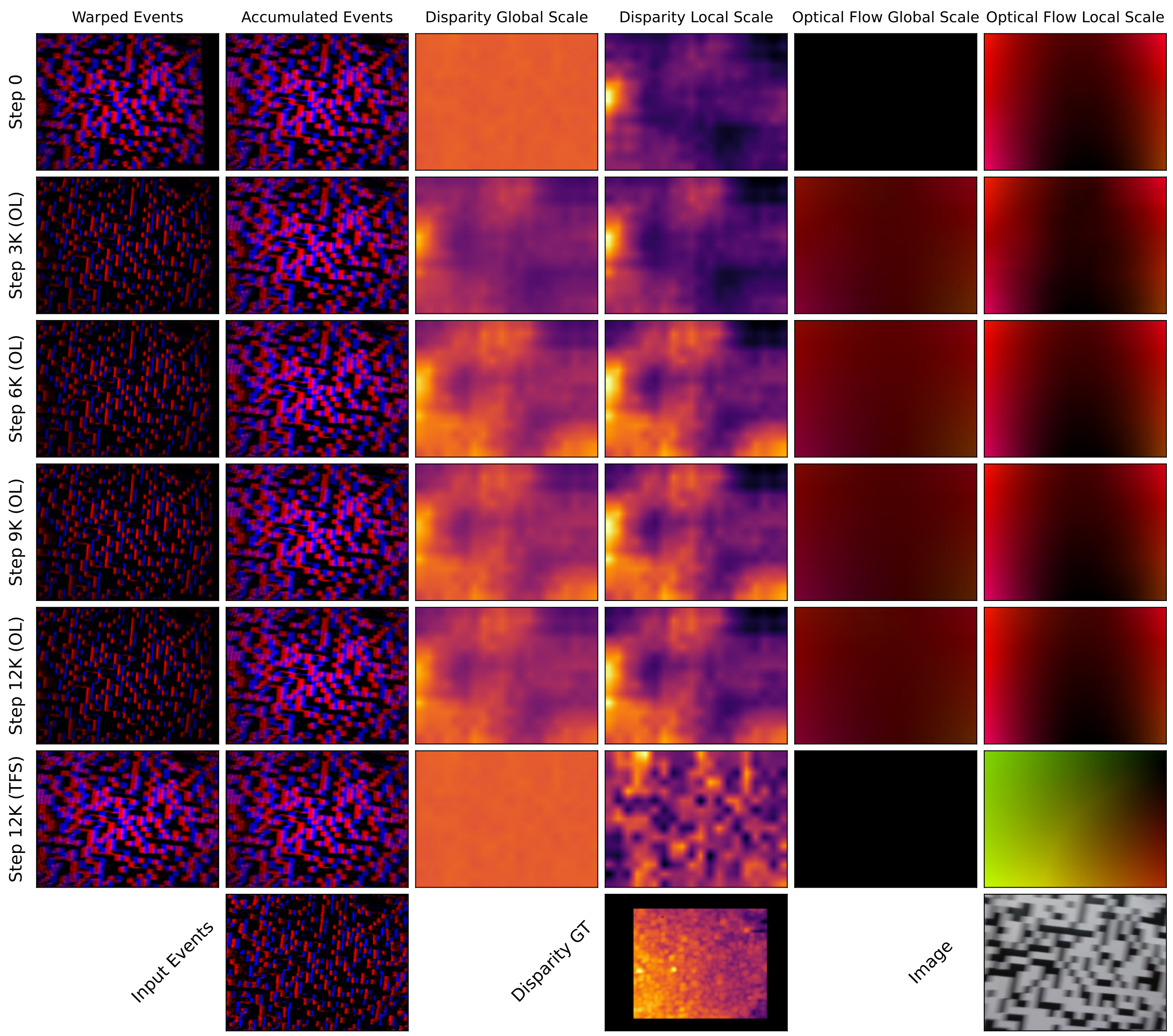

Figure 9. Extended qualitative results on unseen data from a flight test recording. From top to bottom, we evaluate a pre-trained-only network ("step 0"), then four networks after increasing amounts of online learning (OL), and finally a network trained-from-scratch. The bottom row starts with the single 20 ms bin of events currently seen by the network. The rest of the second column shows the accumulated events (multiple bins) in the current contrast maximization loss window. Applying the iterative warp by the optical flow constructed from depth and ego-motion gives the warped and deblurred events in the first column. Columns three and four show disparity at a global (color map shared between rows) and local (color map for only that row) scale. The last two columns show optical flow constructed from depth and ego-motion with global and local color maps, respectively.

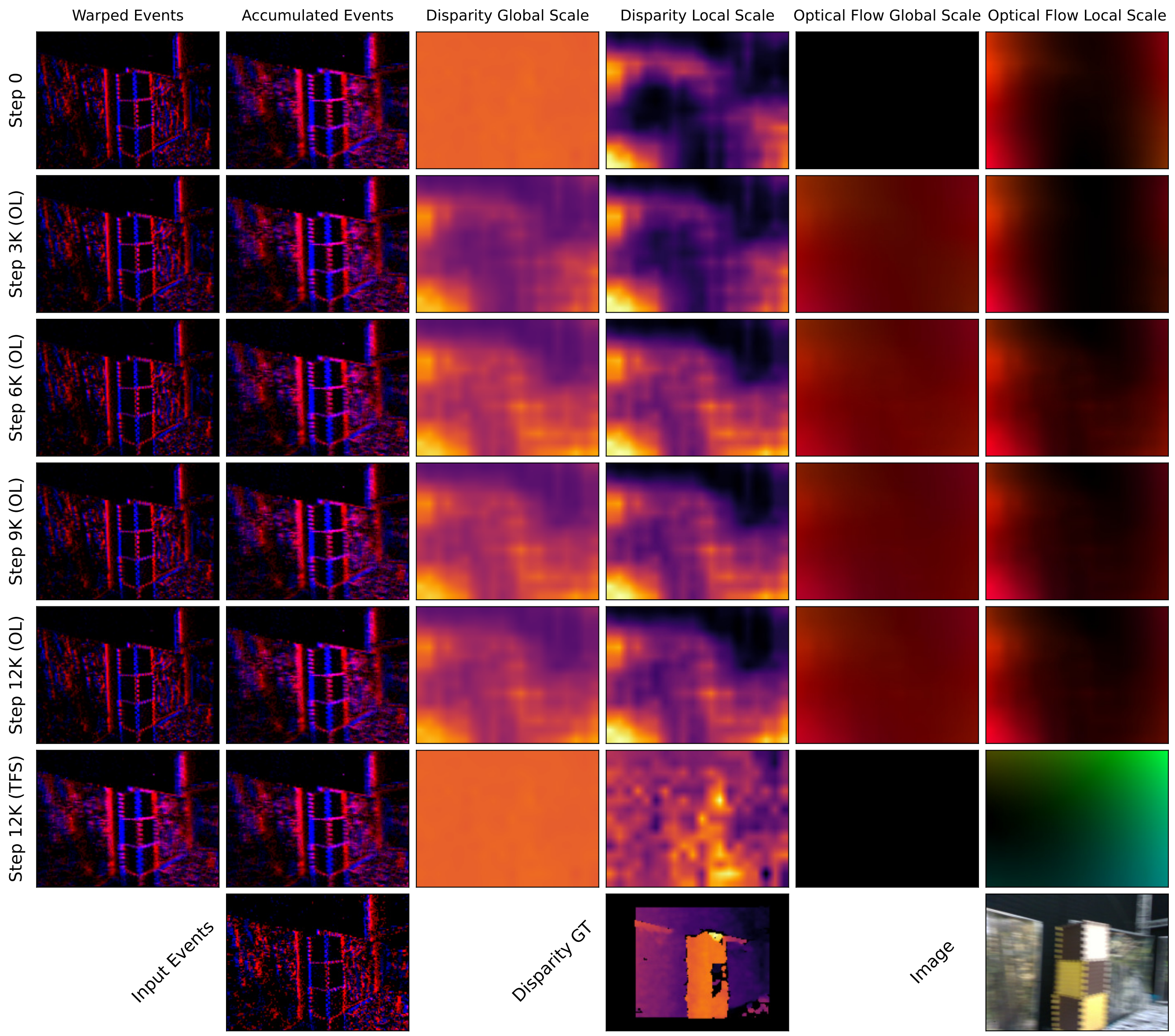

Figure 10. Extended qualitative results on unseen data from a flight test recording. From top to bottom, we evaluate a pre-trained-only network ("step 0"), then four networks after increasing amounts of online learning (OL), and finally a network trained-from-scratch. The bottom row starts with the single 20 ms bin of events currently seen by the network. The rest of the second column shows the accumulated events (multiple bins) in the current contrast maximization loss window. Applying the iterative warp by the optical flow constructed from depth and ego-motion gives the warped and deblurred events in the first column. Columns three and four show disparity at a global (color map shared between rows) and local (color map for only that row) scale. The last two columns show optical flow constructed from depth and ego-motion with global and local color maps, respectively.

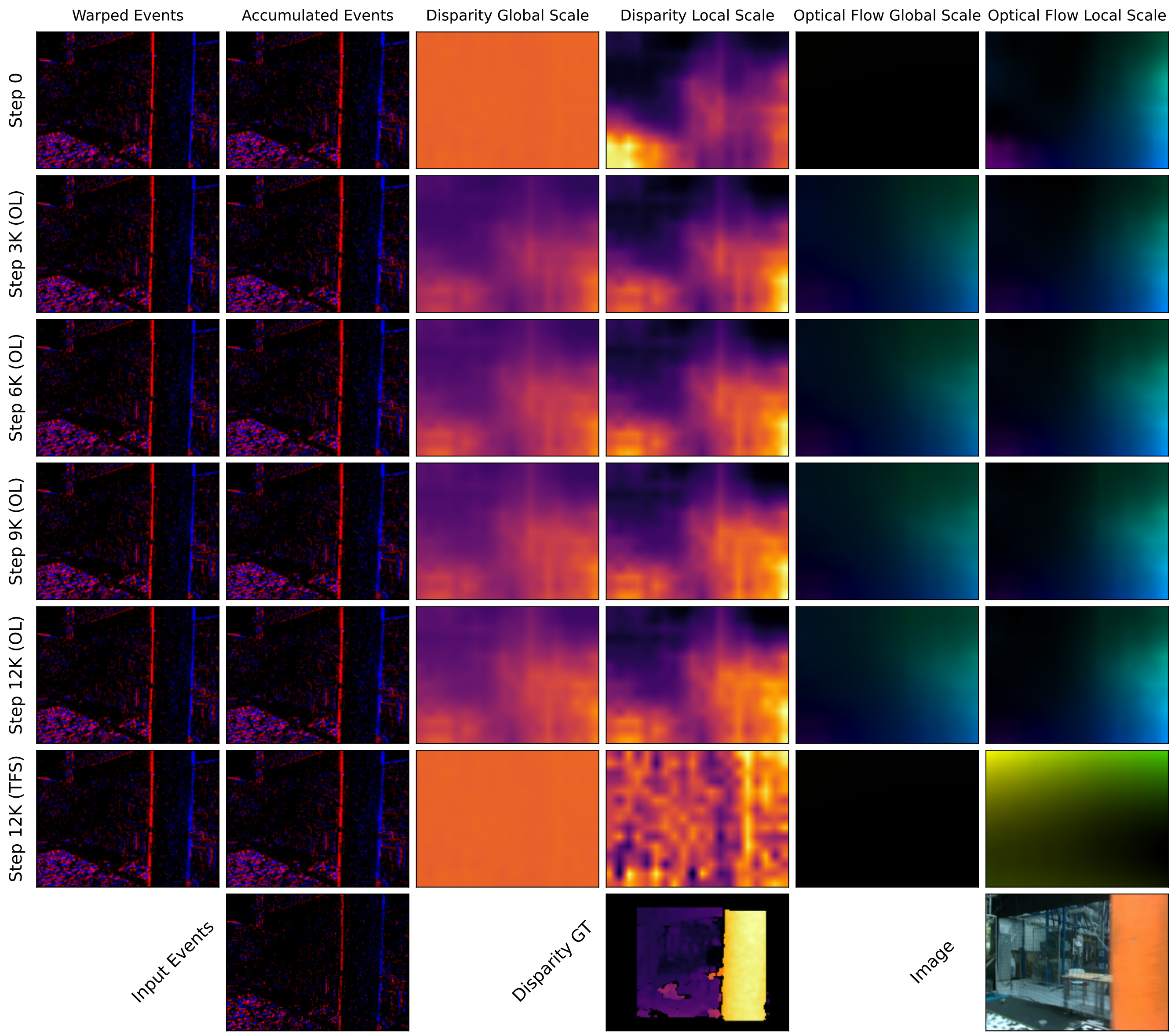

Figure 11. Extended qualitative results on unseen data from a flight test recording. From top to bottom, we evaluate a pre-trained-only network ("step 0"), then four networks after increasing amounts of online learning (OL), and finally a network trained-from-scratch. The bottom row starts with the single 20 ms bin of events currently seen by the network. The rest of the second column shows the accumulated events (multiple bins) in the current contrast maximization loss window. Applying the iterative warp by the optical flow constructed from depth and ego-motion gives the warped and deblurred events in the first column. Columns three and four show disparity at a global (color map shared between rows) and local (color map for only that row) scale. The last two columns show optical flow constructed from depth and ego-motion with global and local color maps, respectively.

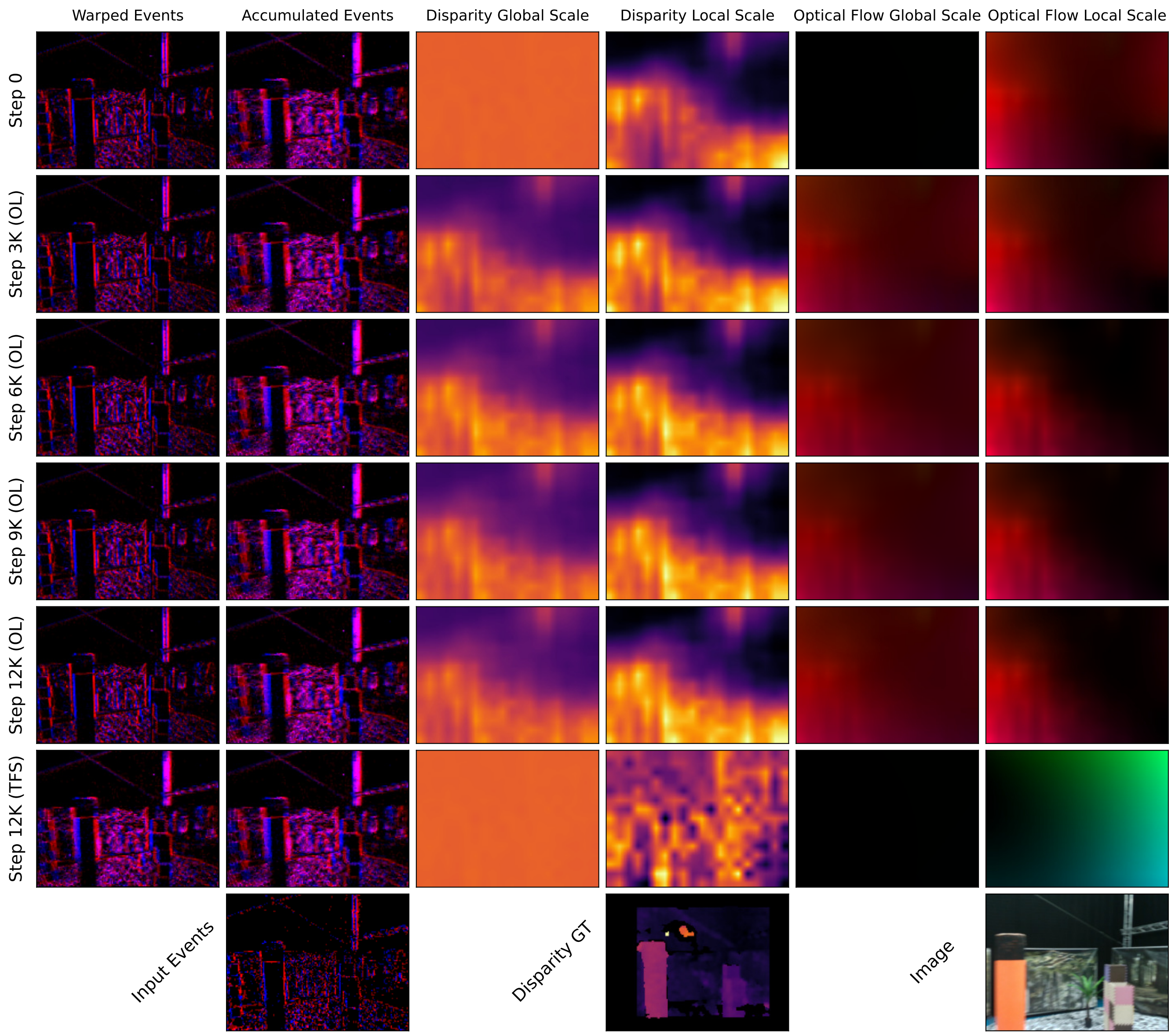

Figure 12. Extended qualitative results on unseen data from a flight test recording. From top to bottom, we evaluate a pre-trained-only network ("step 0"), then four networks after increasing amounts of online learning (OL), and finally a network trained-from-scratch. The bottom row starts with the single 20 ms bin of events currently seen by the network. The rest of the second column shows the accumulated events (multiple bins) in the current contrast maximization loss window. Applying the iterative warp by the optical flow constructed from depth and ego-motion gives the warped and deblurred events in the first column. Columns three and four show disparity at a global (color map shared between rows) and local (color map for only that row) scale. The last two columns show optical flow constructed from depth and ego-motion with global and local color maps, respectively.