Supplementary Material

A. Architecture and Hyperparameters

We train the baseline depth estimation model as described in Section 4 with a ResNet-18 backbone. The networks are implemented in PyTorch and trained on a Tesla V100 GPU with a batch size of 8 and a rehearsal batch size of 8. To control for factors such as dataset size and image size, we fix the training set size to 12,000 images and the test set size to 600 images for each task. The images are center-cropped to maintain the same aspect ratio as in the KITTI dataset and resized to a common width and height of 640 and 192 pixels, respectively. Thus, a total of 48000 samples are used for training with task identities assumed to be known and 2400 samples for testing with task identities assumed to be *un*known. We establish a baseline where the four tasks are trained sequentially for 20 epochs each with an Adam optimizer and an initial learning rate of $1e^{-4}$ for each task. The learning rate is decayed by a factor of 10 after 15 epochs for the respective task. Additionally, we train the four tasks jointly for a total of 20 epochs and an initial learning rate of $1e^{-4}$ which is decayed by a factor of 10 after 15 epochs. Finally, we apply a weight of $\beta = 0.1$ for the spatiotemporal consistency loss with $\rho = 0.85$ (Equation 2), and update the context model at a frequency of 0.05 with $\alpha = 0.999$. The results of MonoDepthCL are shown for the working model.

B. Impact of task order on depth ranges

We pick our task order of VKITTI→ KITTI→ NYUv2→ Cityscapes from the 24 possible combinations for reasons outlined in Section 3 including covering all domain shifts such as indoor-to-outdoor, outdoor-to-indoor, and real-to-sim. However, NYUv2 has a short range depth ($10m$), whereas the remaining tasks, i.e. datasets have longer ($80m$-$100m$) range depths. This raises an interesting question - if NYUv2, i.e. short range depth estimation is the last task performed with no return to a longer depth range task such as Cityscapes, does our method still remember to perform a long range depth estimation?

To test this, we swap the order of appearance of NYUv2 and Cityscapes in our CUDE framework, and train MonoDepthCL on this swapped sequence. Table S1 shows that MonoDepthCL still outperforms NCT on all metrics at different buffer sizes by a large margin on this sequence. Therefore, MonoDepthCL still remembers to perform long range depth estimation right after learning a short range depth estimation task, compared to NCT which undergoes catastrophic forgetting of long range depth estimation.

C. Unsupervised Depth Estimation

Here, we provide some of the details of the unsupervised monocular depth estimation method (see Section 4).

Depth network: The depth network f_D parameterized by θ_D predicts inverse depths at four resolutions as follows:

$$D_1^{-1}, D_2^{-1}, D_3^{-1}, D_4^{-1} = f_D(I_t; \theta_D). \tag{4}$$

Ego-motion network: The ego-motion network f_E parameterized by θ_E predicts the relative rotation $R_{s\leftarrow t}$ and translation $T_{s\leftarrow t}$ between each source-target image pair concatenated along the channel dimension as follows:

$$R^j_{s\leftarrow t}, T^j_{s\leftarrow t} = f_E(I^j_s, I_t; \theta_E), \tag{5}$$

Perspective projection: For each source image I^j_s when warping to the i^{th} upsampled target image using the i^{th} depth prediction;

$$p^j_s \sim K R^j_{s\leftarrow t} D_i[p_{i,t}] K^{-1} p_{i,t} + K T^j_{s\leftarrow t}, \tag{6}$$

where K is the camera intrinsics matrix, and p^j_s and $p_{i,t}$ refer to the pixel locations in j^{th} source and target images, respectively. Then, we use bilinear interpolation to obtain the value of the warped image I^j_s at each location $p_{i,t}$.

Photometric error: The appearance based per-pixel photometric error between the original *target* image and the synthesized target images from n_s *source* images for the i^{th} prediction (Equation 4) is defined as follows:

$$\begin{aligned} \mathcal{P}^j_i &= \frac{\rho}{2}(1 - SSIM(I_t, \hat{I}^j_{i,t})) + (1-\rho)\|I_t - \hat{I}^j_{i,t}\|_1, \\ \mathcal{P}_i &= \min_j \mathcal{P}^j_i, j = 1, 2, ...n_s. \end{aligned} \tag{7}$$

This per-pixel minimum serves to deal with out-of-view pixels and occlusion, such that only the source for which the synthesis is most accurate contributes to the error term. As discussed earlier in Section 4, the loss is masked to counteract the impact of temporally stationary pixels.

Smoothness loss: The per-pixel edge-aware *smoothness loss* for the i^{th} prediction (Equation 4) is defined as follows:

$$\mathcal{S}_i = \left|\partial_x \frac{D_i^{-1}}{\mathbb{E}_{p_t}[D_i^{-1}]}\right| e^{-|\partial_x I_t|} + \left|\partial_y \frac{D_i^{-1}}{\mathbb{E}_{p_t}[D_i^{-1}]}\right| e^{-|\partial_y I_t|}, \tag{8}$$

where the expectation $\mathbb{E}$ of inverse depth predictions are computed across all target pixels [55].

Method	Buffer Size	μ_{final}			μ_{overall}			SPTO		
		abs_rel↓	RMSE↓	a1↑	abs_rel↓	RMSE↓	a1↑	abs_rel↓	RMSE↓	a1↑
NCT	–	0.512	12.930	0.318	0.364	11.008	0.525	0.328	9.542	0.350
MonoDepthCL	50	0.303	8.595	0.543	0.255	8.618	0.637	0.255	8.266	0.604
	200	0.260	7.544	0.633	0.268	8.561	0.655	0.255	8.160	0.664
	500	0.225	6.795	0.672	0.209	7.592	0.704	0.208	7.419	0.704

Table S1. Performance on the CUDE framework for multiple sizes of the memory buffer, when the only short depth range task is learned last.

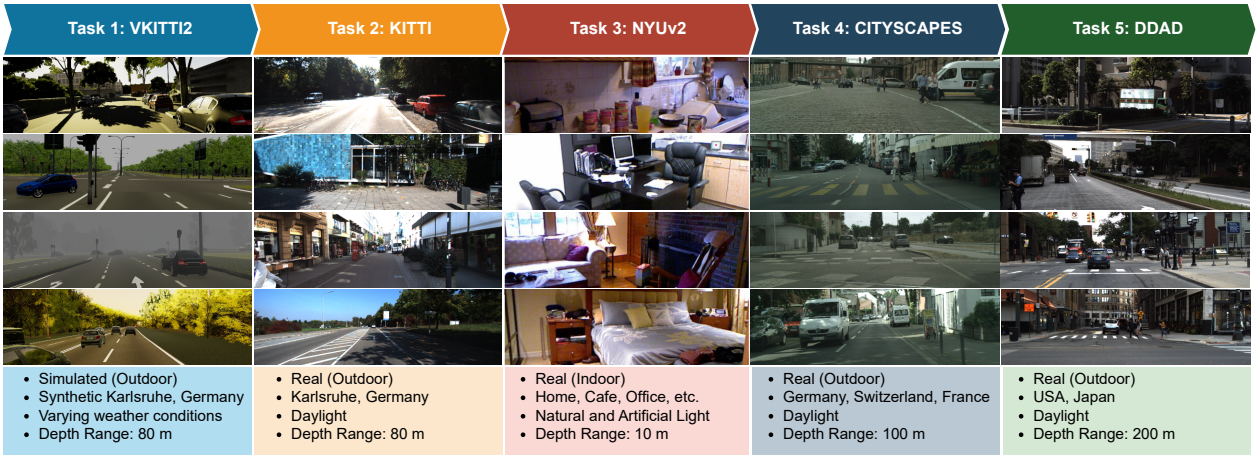

Figure S1. Extended Continual Unsupervised Depth Estimation (CUDE) framework for 5 tasks.

Total task loss : The total combined training loss across all 4 predictions for unsupervised depth estimation is:

$$L_{depth} = \frac{1}{4HW} \sum_{i=1}^{4} \sum_{p_t \in I_t} \mathcal{P}_i[p_t] + \frac{\lambda}{2^{i-1}} \mathcal{S}_i[p_t]. \qquad (9)$$