Unique Lives, Shared World: Learning from Single-Life Videos

Appendix

Table of Contents

A. Dataset Details

A.1. Overview of single-life datasets.

Table 1 details the total duration for each of the single life datasets used in our experiments. Our single life training data is sourced from three main datasets: HD-Epic [31], Walking Tours [43], and a private Anonymous Lives Dataset (ALD). We also include a control group of four "non-life" videos. For evaluating model alignment (CAS), we hold out one life from each of the main datasets, (WT9, P03, and Life5) for testing.

Figure 16 shows sample video frames from a few single lives. Due to privacy constraints, frames from private ALD datasets have been stylized for visualization using Gemini 2.5 Flash Image editing model (Nano-Banana). We emphasize that all models were trained exclusively on the original unstylized video frames.

Table 2 summarizes the number of unique image pairs used for pre-training each of our single-life models. For all lives across all datasets, we report the number of pairs sampled using temporal pairing strategy. For the HD-Epic dataset, which contains the necessary ground truth geometry, we have also included the number of pairs sampled via spatial pairing and the union of both strategies.

Table 1. Dataset duration for each life. '*' denotes testing lives.

HD-Epic	Dur.	WalkingTours	Dur.	ALD	Dur.	Other	Dur.
P01	5.09h	WT1: Amsterdam	1.36h	Life1	30.5h	O1	1h
P02	4.58h	WT2: Bangkok	2.92h	Life2	36.0h	O2	1h
P03*	7.15h	WT3: Chiang Mai	1.13h	Life3	30.4h	O3	1h
P04	4.62h	WT4: Kuala Lumpur	1.21h	Life4	37.7h	O4	1h
P05	3.45h	WT5: Singapore	1.61h	Life5*	34.7h		
P06	4.09h	WT6: Stockholm	1.11h				
P07	3.59h	WT7: Venice	1.83h				
P08	4.04h	WT8: Zurich	1.08h				
P09	4.71h	WT9: Istanbul*	1.13h				

Table 2. Number of sampled frame pairs for each life with different pairing strategies.

Life Name	#Temporal	#Spatial	#Union
P01	2,734,960	1,966,080	1,911,509
P02	2,165,619	978,195	953,120
P04	2,484,197	987,465	973,587
P05	1,848,841	718,144	703,284
P06	1,944,373	627,391	619,193
P07	1,933,987	1,261,987	1,228,048
P08	2,172,066	743,630	734,383
P09	2,532,718	407,913	868,444
WT1	736,605	N/A	N/A
WT2	1,574,775	N/A	N/A
WT3	611,025	N/A	N/A
WT4	655,200	N/A	N/A
WT5	869,805	N/A	N/A
WT6	598,200	N/A	N/A
WT7	989,760	N/A	N/A
WT8	584,670	N/A	N/A
Life1	5,183,970	N/A	N/A
Life2	5,494,905	N/A	N/A
Life3	3,897,839	N/A	N/A
Life4	5,076,709	N/A	N/A
O1	537,285	N/A	N/A
O2	549,210	N/A	N/A
O3	539,865	N/A	N/A
O4	544,710	N/A	N/A

A.2. How we obtain dataset properties

In the main paper, Fig. 3 visualizes a few properties for every life to highlight the similarities and differences across individuals' experiences. Here we explain how these properties are computed. Each property is computed using 1000 uniformly sampled frames (for depth, brightness, *etc*.) or consecutive frame pairs (for camera pose and optical flow) from each life. In every case, the statistic is averaged across the sampled frames.

Camera pose. We estimated the relative camera motion between each pair of consecutive frames using DUSt3R [44]. The pair of frames is 0.1 second apart, and the DUSt3R model infers the relative camera transformation between these two frames. From this transformation, we compute the rotation angle.

Depth. To compute depth statistics, we used the Depth Anything V2 model [49], selecting specialized versions for indoor and outdoor scenes. For indoor-dominant datasets (HD-Epic, ALD, O1 and O2), we used the `Depth-Anything-V2-Metric-Indoor-Large-hf` model, which is fine-tuned on the Hypersim dataset.

(a) Source image (I_t) (b) Temporal pair ($I_{t+\Delta t}$) (c) Spatial pair (d) Augmented pair (e) Random pair

Figure 10. **Visualization of pairing strategies.** Given a **(a)** source image, we generate pairs using different strategies. **(b)** A **temporal pair** is a future frame from the video, capturing natural motion. **(c)** A **spatial pair** is a non-consecutive frame with high geometric overlap, found using camera poses. **(d)** An **augmented pair** is a simple 2D transformation of the source image itself. **(e)** A **random pair** is randomly selected from the same life data.

For outdoor datasets (WT, O3 and O4), we applied the `Depth-Anything-V2-Metric-Outdoor-Large-hf` model, fine-tuned on Virtual KITTI. Each model processed the sampled frames to produce a high-resolution, per-pixel metric depth map for our analysis.

Optical flow magnitude. We utilized RAFT [39] for optical flow estimation, using the weights trained on the Sintel dataset. Similar to camera pose estimation, the optical flow estimation is performed on pairs of consecutive frames. For each pair, the model infers a dense optical flow field, which is a 2D vector at each pixel location representing its displacement to the subsequent frame. We then compute the average optical flow magnitude by calculating the Euclidean norm of optical flow for each pixel then averaging across all the pixels.

Brightness. We first converted each sampled frame to grayscale. The brightness for a single frame is then calculated as the mean pixel intensity of the resulting grayscale image. This value is subsequently normalized to a range from 0 to 1 by dividing by 255.

Objects. We investigated the presence of objects in every life using two approaches. On HD-Epic where the object bounding box annotations are available with the dataset, we simply count the occurrences of these objects from the annotations. For WT and ALD where there are no object annotations, we prompt Gemini 2.5 Pro to recognize the visible objects in video frames and aggregate the object counts. We collect statistics on 10 objects of interest: phone, table, animal, t-shirt, people, plant, car, food, knife and drawer. For the spider plot on the right of Fig. 3, the object frequencies are normalized to 0-1, with respect to the count of 'plant' which appears to be the most frequent object among the 10 chosen objects.

A.3. Visualization of pairs sampled in each dataset.

Figure 10 provides a visual comparison of the different pairing strategies used in our experiments. Given a source image (a), we create training pairs that provide distinct supervisory signals. A *temporal pair* (b) is a frame captured a short time after the source, containing natural motion and small viewpoint changes. A *spatial pair* (c) is a temporally distant frame with significant viewpoint change but high geometric overlap, found using camera poses. These two strategies aim to capture the rich spatiotemporal structure of a single life. As baselines, we use an *augmented pair* (d), which is a simple 2D transformation of the source image and lacks true 3D viewpoint change, and a *random pair* (e), a completely unrelated frame from the same life that lacks any correspondence.

B. Implementation Details

B.1. Architecture and general pre-training details

We re-implemented the CroCo [45] architecture in JAX [6] using a ViT-Base/16 encoder to encode images of 256×256 with a patch size of 16×16 pixels. For the decoder, we use a series of 12 blocks with 768 dimensions and 12 attention heads. We pre-trained each single-life within HD-Epic, Walking Tours, and ALD for 100, 30, and 10 epochs and a total batch size of 256, 64, and 256 respectively. Overall, each model is trained for roughly 200-250K iterations. We use AdamW optimizer [27] with a cosine learning rate schedule with a base learning rate of 1.5×10^{-4} and a linear warm-up in the first 10% of total number of epochs for each dataset. We train each model using 16 TPUs-v6. We used random crop, resizing, and color augmentation. For temporal pairing we used 95% masking ratio in all our pretraining experiments. We performed a sweep on masking ratio for spatial-based pairing experiments detailed in Section B.4.

B.2. Downstream evaluation details

Depth estimation tasks. We use a consistent set of hyperparameters to evaluate downstream depth estimation on both NYU-Depth-V2 and ScanNet. A lightweight readout module, consisting of a single Transformer block (1024 hidden dimension, 16 heads) followed by two linear layers, is attached to the frozen pre-trained encoder. This module is trained for 40k iterations with a batch size of 32 using the

AdamW optimizer. We use a cosine learning rate schedule with a peak learning rate of 3×10^{-4}. Additional results using Attentive Finetuning and a DPT head are available in Appendix C.4.

Zero-shot correspondence on HPatches implementation details. Following the protocol of ZeroCo [2], we evaluate zero-shot correspondence on the HPatches dataset. For each image pair, we compute a final correspondence map, $\mathcal{A}_i$, by averaging the cross-attention maps from all decoder blocks of our pre-trained model. Performance is measured by computing the Average End-Point Error (AEPE) on the HPatches-240 variant, where correspondences are evaluated at a 240×240 resolution.

B.3. Life-size experiment details

In the main paper, Fig. 4 and 7 compare single-life models trained on different "life sizes". The term "life-size" refers to a fixed duration of videos from an individual's experience. For example, to train a model with a life-size of T, we randomly sample a video segment of duration T from a single-life's video. Specifically, we vary the life size across the following durations: $\{3.6\text{s}, 36\text{s}, 6\text{m}, 12\text{m}, 30\text{m}, 45\text{m}, 1\text{h}, 2\text{h}, 3\text{h}, 30\text{h}\}$ subject to the maximum available duration of each life. Since different lives naturally have different total lengths (e.g., HD-Epic is $\sim$3.5h, Walking Tours is $\sim$1.1h, and ALD is $\sim$30.5h), the curves in Fig. 4 and 7 stop at different points. For our Kinetics baseline, which is composed of a large number of short, 10-second videos, we create subsets of its training videos to match the total duration of each life-size experiment.

For this scaling analysis, all training hyperparameters were held constant: the models shown in Fig. 4 and 7 were each trained for 250k iterations with batch size 64, with other hyperparameters matching those in Section B.1.

B.4. Masking ratio and Jaccard threshold

To determine the optimal hyperparameters for our spatial-based pairing strategies, we conducted a grid search for each life within the HD-Epic dataset. We swept over the masking ratio ($m \in \{0.5, 0.7, 0.9\}$) and the Jaccard co-visibility threshold ($j \in \{0.5, 0.7, 0.9\}$) for both the spatial and union pairing methods. The performance of each configuration was evaluated via attentive probing on the ScanNet and NYU-Depth-v2 depth estimation tasks. The full results of this sweep are presented in Table 3. The best-performing combination of masking ratio and Jaccard threshold is highlighted in bold for each life. For all main experiments involving these pairing strategies, we report results using the optimal hyperparameters selected on a per-life basis from this sweep.

Algorithm 1 CAS Score (NumPy Implementation)

```python
import numpy as np

def get_cas_score(a_i, a_j, k=5):
    """
    Calculate Correspondence Agreement Score.
    a_i: Patch correspondence map of model i. [HW, HW]
    a_j: Patch correspondence map of model j. [HW, HW]
    k: The top-k value
    """
    N = a_i.shape[0]

    # Initialize boolean masks
    pred_mask = np.zeros((N, N), dtype=bool)
    target_mask = np.zeros((N, N), dtype=bool)

    # Create row indices for broadcasting
    row_indices = np.arange(N)[:, None]

    # Populate masks
    pred_mask[row_indices, a_i] = True
    target_mask[row_indices, a_j] = True

    # Calculate intersection over k
    mtopk = (pred_mask & target_mask).sum(axis=1) / k
    cas = mtopk.mean()

    return cas
```

B.5. CAS implementation details

For CAS computation, we use $k = 5$ for Mutual Top-k. A pseudo-code to compute CAS for one pair of images in NumPy style is shown in Algorithm 1.

C. Additional Results

C.1. Varying test set temporal intervals

In Table 4, we evaluated generalization beyond the training distribution ($[1, 16]$ frames) by sampling 1000 pairs per dataset at gaps of $[17, 32]$ and $[33, 64]$ frames. As expected, average CAS on these 3000-pair sets decreases as gaps increase and overlap narrows. However, models maintain consistent relative alignment, validating the robustness of our metric.

C.2. Zero-shot correspondence on HPatches

Figure 11 shows the relative performance gains on this benchmark for all three of our main datasets. The results reinforce our findings from the depth estimation tasks: the Temporal Pairing strategy consistently and significantly outperforms both the Augmented and Random Pairing baselines across all datasets.

C.3. Qualitative results on HPatches

Figure 12 shows two HPatches samples for zero-shot correspondence task. Every single-life model takes both the source and target image as input, and for each pixel in the source frame, the model derives the corresponding pixels in the target frame. For clarity, we query 8×8 pixels in the source frame and visualize their corresponding pixels in the target frame. The models trained on ALD lives perform

Table 3. Hyperparameter sweep for spatial and union pairing strategies on the HD-Epic dataset. We varied the masking ratio (m) and the Jaccard threshold (j). Performance is measured by attentive probing on ScanNet (AbsRel ↓) and NYU-Depth-v2 (δ_1 ↑). The best performing setting for each participant and strategy is highlighted in **bold**.

PID	Pairing	Masking Ratio (m)	Jaccard (j) = 0.5		Jaccard (j) = 0.7		Jaccard (j) = 0.9	
			ScanNet ↓	NYUv2 ↑	ScanNet ↓	NYUv2 ↑	ScanNet ↓	NYUv2 ↑
P01	Spatial	0.5	0.2330	0.5694	**0.22561**	**0.5766**	0.2299	0.5868
		0.7	0.2411	0.5694	0.2304	0.5795	0.2350	0.5773
		0.9	0.2361	0.5772	0.2329	0.5736	0.2417	0.5675
	Union	0.5	0.2326	0.5737	**0.22501**	**0.5818**	0.2301	0.5759
		0.7	0.2392	0.5701	0.2298	0.5791	0.2339	0.5710
		0.9	0.2370	0.5703	0.2369	0.5731	0.2427	0.5684
P02	Spatial	0.5	0.2389	0.5708	**0.24547**	**0.5738**	0.2503	0.5760
		0.7	0.2468	0.5651	0.2408	0.5683	0.2559	0.5685
		0.9	0.2492	0.5607	0.2458	0.5634	0.2705	0.5418
	Union	0.5	0.2292	0.5681	**0.22561**	**0.5773**	0.2319	0.5746
		0.7	0.2310	0.5691	0.2276	0.5787	0.2461	0.5539
		0.9	0.2366	0.5579	0.2426	0.5593	0.2553	0.5391
P04	Spatial	0.5	0.3145	0.5073	0.2531	0.5676	**0.24861**	**0.5677**
		0.7	0.3020	0.5093	0.2663	0.5650	0.2789	0.5607
		0.9	0.2663	0.5495	0.2601	0.5541	0.2577	0.5481
	Union	0.5	0.2674	0.5146	0.2365	0.5617	**0.23137**	**0.5666**
		0.7	0.2660	0.5286	0.2402	0.5580	0.2413	0.5652
		0.9	0.2493	0.5497	0.2414	0.5545	0.2432	0.5559
P05	Spatial	0.5	0.2481	0.5618	0.2566	0.5547	0.2537	0.5537
		0.7	**0.25423**	**0.5733**	0.2564	0.5641	0.2561	0.5635
		0.9	0.2648	0.5619	0.2719	0.5520	0.2760	0.5390
	Union	0.5	0.2357	0.5590	**0.23373**	**0.5692**	0.2431	0.5592
		0.7	0.2409	0.5600	0.2349	0.5658	0.2504	0.5495
		0.9	0.2416	0.5635	0.2453	0.5590	0.2610	0.5400
P06	Spatial	0.5	0.2627	0.5449	0.2677	0.5416	0.2601	0.5572
		0.7	**0.25490**	**0.5711**	0.2555	0.5628	0.2694	0.5585
		0.9	0.2607	0.5651	0.2727	0.5542	0.2785	0.5330
	Union	0.5	0.2521	0.5467	**0.23815**	**0.5700**	0.2413	0.5629
		0.7	0.2506	0.5485	0.2389	0.5616	0.2476	0.5489
		0.9	0.2427	0.5600	0.2466	0.5561	0.2603	0.5351
P07	Spatial	0.5	0.2531	0.5564	0.2555	0.5523	0.2542	0.5580
		0.7	0.2437	0.5675	**0.24997**	**0.5666**	0.2540	0.5590
		0.9	0.2444	0.5646	0.2463	0.5668	0.2657	0.5552
	Union	0.5	0.2377	0.5486	**0.22904**	**0.5730**	0.2332	0.5652
		0.7	0.2423	0.5469	0.2322	0.5685	0.2361	0.5681
		0.9	0.2399	0.5676	0.2391	0.5641	0.2427	0.5603
P08	Spatial	0.5	0.3309	0.4601	0.3280	0.4608	0.2893	0.5055
		0.7	0.2470	0.5559	0.2580	0.5403	0.2574	0.5409
		0.9	**0.24209**	**0.5558**	0.2479	0.5475	0.2507	0.5515
	Union	0.5	0.3334	0.4610	0.2444	0.5489	**0.24133**	**0.5538**
		0.7	0.3240	0.4715	0.2517	0.5415	0.2466	0.5470
		0.9	0.2937	0.4999	0.2549	0.5426	0.2502	0.5501
P09	Spatial	0.5	0.2625	0.5496	0.2551	0.5508	0.2564	0.5586
		0.7	0.2421	0.5707	0.2458	0.5691	0.2523	0.5514
		0.9	**0.25262**	**0.5718**	0.2544	0.5617	0.2651	0.5463
	Union	0.5	0.2378	0.5501	**0.22717**	**0.5793**	0.2337	0.5741
		0.7	0.2405	0.5583	0.2334	0.5657	0.2424	0.5631
		0.9	0.2369	0.5620	0.2428	0.5577	0.2521	0.5451

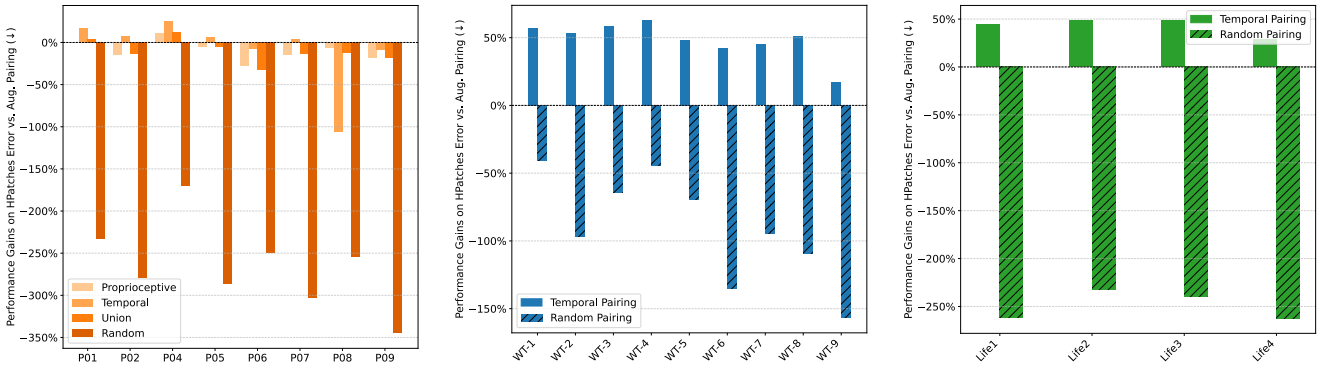
Table 4. Varying test set temporal intervals on all datasets.

Dataset	[1, 16]	[17, 32]	[33, 64]
HD-Epic	0.483 ± 0.021	0.340 ± 0.015	0.277 ± 0.014
WalkingTours	0.452 ± 0.015	0.315 ± 0.011	0.264 ± 0.007
ALD	0.492 ± 0.053	0.353 ± 0.035	0.292 ± 0.026

better on this task, as the corresponding pixels in the target frames are more structured (the + sign). Generally, all single-life models manage to find corresponding pixels to some extent.

C.4. Other downstream evaluation protocols

On downstream depth estimation task, we also experiment with different evaluation protocols other than Attentive probing. For Attentive finetuning, we use the same single transformer block but finetune the whole network end-to-end for depth estimation task. For DPT finetuning, we



(a) HD-Epic (b) Walking Tours (c) Anonymous Lives (ALD)

Figure 11. **Relative performance gains on zero-shot HPatches correspondence.** The plots show the percentage gain in performance (AEPE, lower is better) for different pairing strategies relative to the Augmented Pairing baseline (the 0% line) for our three main datasets: **(a)** HD-Epic, **(b)** Walking Tours, and **(c)** Anonymous Lives.

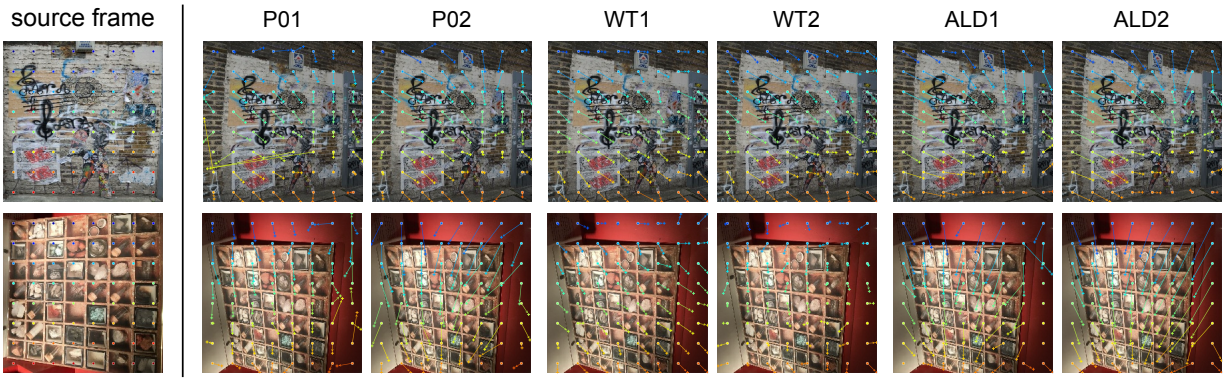


Figure 12. HPatches qualitative results from six single-life models. We query a subset of source pixels (shown as "dot"), and visualize the corresponding pixels in the target frames as "+" sign. The displacement of source pixels are visualized as arrows.

follow the original setup from CroCo [45] and use a DPT head [34] attached on the pretrained encoder, and finetune the whole network end-to-end.

Table 6 presents the comparison between Attentive Probing (Attn Frozen) and full Attentive Finetuning. As the results clearly indicate, allowing the entire network to finetune leads to substantial performance improvements across all datasets and tasks, with ScanNet evaluation on Walking Tours lives being the only case where the performance degrades. For example, on NYU-Depth-v2, finetuning improves δ_1 accuracy by more than 10 percentage points on average, while on ScanNet, it reduces the absolute relative error by nearly 40%. This significant performance boost confirms that the features learned from the single-life paradigm are not brittle; they serve as a strong and highly adaptable foundation for task-specific fine-tuning.

C.5. DINOv2 results and discussion

Our experiments are conducted on the Cross-View Completion (CroCo) architecture. To explore whether our findings extend to alternative self-supervised learning paradigms, we also evaluate on the DINOv2 [30] training adapting it to our proposed 'single-life' learning paradigm. Specifically, we train several DINOv2 models (all using the same ViT-B architecture) *from scratch* on individual lives, using the same single-life data as in the main paper.

There are some notable differences between DINOv2 and CroCo: (1) DINOv2 takes single images as input. We extract video frames and shuffle them, thus creating an im-

Table 5. Different evaluation protocols on NYU-Depth-v2 depth estimation tasks. The δ_1 accuracy is reported (higher the better).

Dataset	Attn Frozen	Attn FT	DPT FT
Habitat [45]	0.767	0.828	0.892
K400	0.590	0.696	0.696
ALD-Life1	0.580	0.694	0.693
ALD-Life2	0.580	0.684	0.705
ALD-Life4	0.570	0.671	0.685

Table 6. Full finetuning results on ScanNet and NYUv2

	ScanNet AbsRel(↓)		**NYUv2** δ_1(↑)	
Dataset	Attn Frozen	Attn Finetune	Attn Frozen	Attn Finetune
HD-EPIC (Temporal)	0.2569 ± 0.0089	0.1558 ± 0.0096	0.5617 ± 0.0109	0.6770 ± 0.0148
HD-EPIC (Spatial)	0.2455 ± 0.0096	0.1545 ± 0.0056	0.5703 ± 0.0063	0.6867 ± 0.0122
WT (Temporal)	0.2783 ± 0.0079	0.3451 ± 0.0261	0.5302 ± 0.0124	0.6410 ± 0.0189
ALD (Temporal)	0.2377 ± 0.0035	0.1574 ± 0.0039	0.5795 ± 0.0072	0.6865 ± 0.0094

age dataset. (2) DINOv2 consists of a single transformer encoder (without decoder). As a result, we adapt our score (CAS) to this scenario, and compute this score from the encoder feature correlation (used in ZeroCo [2]) rather than the decoder cross-attention maps. In other words, given a pair of images, we use patchwise encoder features to compute patch-to-patch similarity matrix A_i associated with each trained model i. We then use these matrices to compute the CAS score using Eq. (1) defined in the main paper. This allows us to compare different trained DINOv2 models in the same way as we did for CroCo.

Table 7 summarizes the single-life DINOv2 results. We find single-life DINOv2 gives similar depth estimation and zero-shot correspondence performance compared with single-life CroCo. Notably, DINOv2 could not be trained stably on redundant lives (O2, O3) or out-of-distribution video (O1) - the training quickly gets infinite or NaN loss, indicating DINOv2 method is better suited for IID data and not robust to redundant input. We trained DINOv2 on a 30 hours long subset of K400 which achieves 63% on NYU-Depth-v2, 0.193 on ScanNet, and 18.0 on HPatches further supporting DINOv2's suitability for IID-like data. We also present the cross-model CAS matrix for single-life DINOv2 models in Fig. 13. Note that CAS is computed on the encoder feature correlation. Comparing with the main paper Fig. 5, we observe that CAS metric results have higher variability for DINOv2: we do not observe a strong CAS intra-datasets compared to inter-datasets, and interestingly on O4 (minecraft) we find high similarity with other single-life models trained on natural lives. Note that O4 (minecraft) serves as a deliberate edge case, pairing first-person 3D consistency with non-realistic, pixelated visuals. We also visualized a 2D MDS plot similar to the main paper Fig. 6 but for single-life DINOv2 models, in Fig. 14.

Our findings highlight that it is possible to train diverse architectures on single-life data. However, the combination of the single-life learning paradigm and a suitable training objective is important for learning representations that are both generalizable and consistently aligned.

Table 7. Single-life DINOv2 results. *DINOv2 training is unstable and does not converge on these datasets, hence we took the last checkpoint before Inf/NaN appears.

Dataset	NYUv2↑	ScanNet↓	HPatches↓	CAS w. CroCo↑
HD-Epic: $\{1, 2, 4\}$	0.589±0.009	0.222±0.005	19.8±1.5	0.536±0.01
WT: $\{2, 3, 7\}$	0.562 ± 0.026	0.256±0.003	21.2 ± 4.16	0.431±0.04
ALD: $\{1, 2, 4\}$	0.598±0.05	0.215±0.033	19.7±9.9	0.516±0.13
K400-30h	0.632	0.193	18.0	0.524
O1*	0.423	0.351	57.3	0.120
O2*	0.453	0.326	51.1	0.210
O3*	0.450	0.325	53.0	0.182
O4	0.596	0.217	20.4	0.514

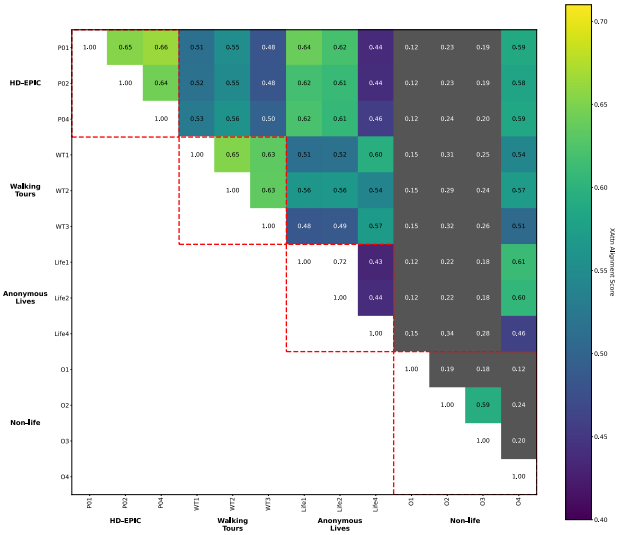


Figure 13. DINOv2 CAS metric across models. We do not observe a similar cross-model alignment within datasets like we observed for CroCo (red blocks).

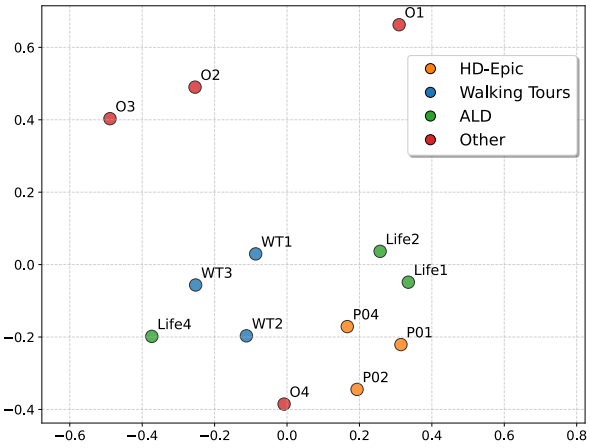


Figure 14. A 2D MDS visualization of the single-life DINOv2 models using the CAS score from Fig. 13 as the similarity metric.

C.6. Results beyond Geometric Tasks

To assess whether our geometry-focused pre-training maintains semantic coherence, we qualitatively analyze performance on a **zero-shot segmentation label propagation task** [23] on **DAVIS 2017** [32]. Specifically, the model is given the ground-truth masks for multiple objects in the first frame and the task is to propagate these masks through the remaining frames of the video. Following common practice, all evaluations are performed on 480p resolution images. To isolate the quality of our learned representation, we employ a simple K-nearest neighbor inference algorithm. The features from our encoder are used to compute a dense similarity map between pixels, allowing labels to be propagated based on the k most similar pixels in the source frame(s). This zero-shot approach directly probes the representation's utility for object-level correspondence without

any task-specific fine-tuning.

As shown in Fig. 15, models trained on three single lives from all three datasets show consistent and reasonable label propagation. These promising results highlight that these models, trained independently on different lives, can generalize to unseen objects – e.g. none of our lives have seen a 'swan' and all our indoor lives have not seen a 'car' during training.

Additionally, we evaluate the models on the Perception Test point tracking task [33], following the protocol in [7]. Using the HD-Epic single-life checkpoints, we observe that performance improves with increased data duration. Scaling the training data from 30 minutes to 1 hour and 3 hours improves the Average Jaccard (AJ) from 51% to 54% and 57%, respectively. This upward trajectory approaches the 64% AJ baseline achieved by a model trained on 850 hours of diverse K400 data, further validating the efficacy of the learning signal inherent in single-life videos.

Figure 15. Video segmentation tracking results on DAVIS 2017 dataset from three single-life models. From top: HD-Epic P01,P02,P04, WT1,2,3, and ALD Life1,2,3. The features produced by the models are used to propagate the ground-truth segmentation labels of the first video frames (left) to the future frames (right) in a zero-shot manner.

(a) HD-Epic

(b) Walking Tours

(c) ALD (stylized for anonymity)

(d) Other non-life videos

Figure 16. A collection of sample frames for each of the datasets.