D2E: Scaling Vision-Action Pretraining on Desktop Data for Transfer to Embodied AI

Suhwan Choi* MAUM.AI
Jaeyoon Jung* MAUM.AI
Haebin Seong* MAUM.AI
Minchan Kim MAUM.AI
Minyeong Kim Stanford University
Yongjun Cho MAUM.AI
Yoonshik Kim MAUM.AI
Park Yu Been MAUM.AI
Youngjae Yu† Yonsei University
Yunsung Lee† MAUM.AI

Abstract

Large language models leverage internet-scale text data, yet embodied AI remains constrained by the prohibitive costs of physical trajectory collection. Desktop environments—particularly gaming—offer a compelling alternative: they provide rich sensorimotor interactions at scale while maintaining the structured observation-action coupling essential for embodied learning. We present **D2E** (Desktop to Embodied AI), a framework that demonstrates desktop interactions can serve as an effective pretraining substrate for robotics embodied AI tasks. Unlike prior work that remained domain-specific (e.g., VPT for Minecraft) or kept data proprietary (e.g., SIMA), D2E establishes a complete pipeline from scalable desktop data collection to verified transfer in embodied domains. Our framework comprises three components: (1) the OWA Toolkit that unifies diverse desktop interactions into a standardized format with 152× compression, (2) the Generalist-IDM that achieves strong zero-shot generalization across unseen games through timestamp-based event prediction, enabling internet-scale pseudo-labeling, and (3) VAPT that transfers desktop-pretrained representations to physical manipulation and navigation. Using 1.3K+ hours of data (259 hours of human demonstrations, and 1K+ hours of pseudo-labeled gameplay), we achieve a total of 96.6% success rate on LIBERO manipulation and 83.3% on CANVAS navigation benchmarks. This validates that sensorimotor primitives in digital interactions exhibit sufficient invariance to transfer meaningfully to physical embodied tasks, establishing desktop pretraining as a practical paradigm for robotics. We will make all our work public, including the OWA toolkit, datasets of human-collected and pseudo-labeled, and VAPT-trained models. (Demo available at *link*)

1 Introduction

Large-scale datasets have driven recent progress in large language models (LLMs) (Kaplan et al., 2020; Hoffmann et al., 2022), where pretraining on internet-scale resources enables strong generalization across diverse downstream tasks. In contrast, embodied AI has yet to experience such a scaling breakthrough. Unlike text, which can be collected from the web with minimum effort, embodied trajectories demand specialized hardware, costly human operation, and complex pipelines for annotation (Mandlekar et al., 2019; Qin et al., 2023; Fu et al., 2024; Cheng et al., 2024; Park et al., 2024). As a result, most existing datasets remain relatively small, domain-specific, and fragmented across incompatible formats (Geng et al., 2025), preventing the emergence of a true "data flywheel" for embodied AI.

Desktop interactions—screen, keyboard, and mouse—offer a compelling alternative for scaling vision-action learning (Baker et al., 2022; Raad et al., 2024). These interfaces are stan-

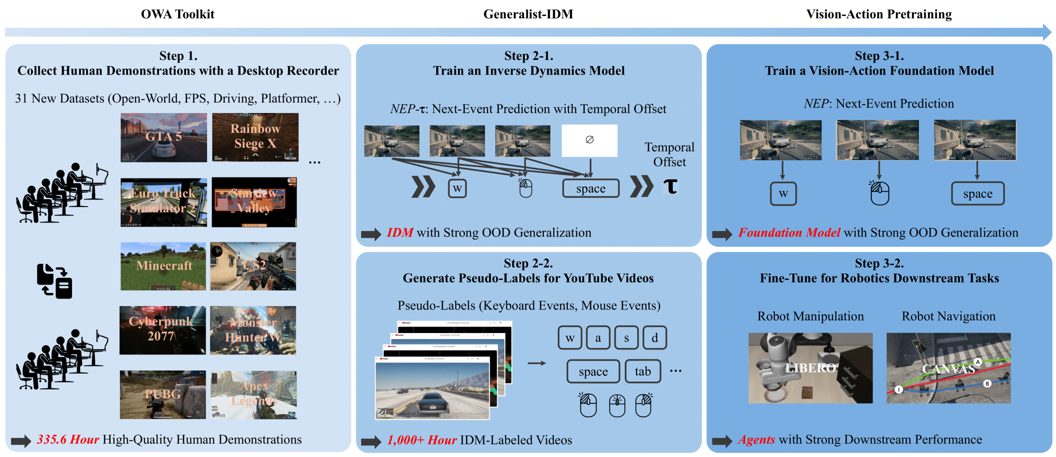

Figure 1: **Overview of D2E framework.** (1) The OWA Toolkit captures 335.6 hours of rich desktop demonstrations across 31 games with 152× compression. (2) The Generalist-IDM uses next-event prediction with temporal offset (NEP-τ) to achieve OOD generalization, enabling pseudo-labeling of 1K+ hours of YouTube gameplay. (3) Vision-Action Pretraining transfers desktop-pretrained representations to embodied AI, achieving 96.6% success on LIBERO manipulation and 83.3% on CANVAS navigation benchmarks which demonstrates desktop-to-robotics transfer.

dardized, human-centric, and naturally abundant: millions of users generate rich interaction trajectories through everyday digital activities. Crucially, desktop environments preserve the tight observation-action coupling essential for embodied learning while abstracting away hardware-specific constraints (Tang et al., 2025; Shridhar et al., 2020; Raad et al., 2024). Gaming interactions, in particular, exhibit complex sensorimotor patterns—navigation, object manipulation, strategic planning—that mirror many embodied AI challenges, yet are freely shared at internet scale through gameplay videos.

We introduce **D2E (Desktop to Embodied AI)**, a framework that systematically transforms desktop interactions into a scalable pretraining substrate for embodied AI. D2E addresses two fundamental challenges: establishing a unified pipeline for high-quality desktop data collection, and extending beyond manual annotations to leverage the vast repository of unlabeled internet videos.

Our first contribution, the **Open-World Agents (OWA) Toolkit**, provides the infrastructure for scalable desktop data capture. Built on Windows APIs and GStreamer (Microsoft Corporation; GStreamer Team), OWA's `ocap` recorder synchronizes multimodal streams—screen, keyboard, and mouse—into time-aligned events, while our OWAMcap format achieves order-of-magnitude compression improvements over existing formats. Through OWA, we collected 335 hours of human demonstrations across 31 diverse games and applications, establishing a foundation for desktop-based pretraining.

Beyond human demonstrations, we introduce the **Generalist Inverse Dynamics Model (Generalist-IDM)** to demonstrate a pathway toward internet-scale data collection. By reformulating action prediction as timestamp-aware next-event prediction (NEP-τ), our model achieves strong zero-shot generalization—substantially outperforming specialist baselines on unseen games with minimal compute requirements. This generalization capability enables automatic pseudo-labeling of YouTube gameplay videos, expanding our dataset by over $1,000$ hours without additional human annotation.

We demonstrate that desktop-pretrained representations transfer meaningfully to physical robotics through **Vision-Action PreTraining (VAPT)**. Models pretrained on our combined desktop corpus show consistent improvements on standardized benchmarks: It achieves a total success rate of 96.6% on *LIBERO* manipulation (Liu et al., 2023) and 83.3% on *CANVAS* navigation (Choi et al., 2024). These results establish, for the first time, that the sensorimotor patterns learned from desktop interactions can directly enhance performance in embodied AI domains, validating desktop data as a practical alternative to costly physical data collection.

Our contributions are threefold:

1. **OWA Toolkit**: A framework that contains `ocap` for synchronized event recording with FHD/QHD 60 Hz support, OWAMcap format for compact storage, and an optimized data pipeline for ML training—achieving up to 152× compression and 16× lower average disk read per image compared to TorchCodec; used to collect 335 hours of human demonstrations.
2. **Generalist-IDM**: An inverse dynamics model that outperforms game-specific Specialist IDMs, exhibiting out-of-domain generalization and in-context adaptation (e.g., calibrating mouse scale). Trained on OWA-collected data with around 192 H100-hours ($\sim$ \$800), the strong generalization of Generalist-IDM allows us to pseudo-label over 1K+ hours of YouTube gameplay.
3. **VAPT foundation model**: A vision-action pretrained model trained on 1.3K hours of desktop data from OWA and Generalist-IDM pseudo-labeling, transferring desktop knowledge to robotics. VAPT achieves 96.6% success on manipulation (*LIBERO*) and 83.3% on navigation (*CANVAS*).

2 Related Work

Collecting Data for Vision-Action Pretraining. Large-scale vision-action (or vision-language-action) pretraining depends on multimodal corpora that pair perception with grounded actions across diverse tasks (Kaplan et al., 2020; Hoffmann et al., 2022). Recent embodied agents unify perception and control in a single model across heterogeneous domains (Reed et al., 2022; Firoozi et al., 2024; Wen et al.). In robotics, resources are emerging: RT-1 (Brohan et al., 2022) and RT-2 (Zitkovich et al., 2023) scale vision–language–action to real robots; Open X-Embodiment aggregates heterogeneous datasets to train RT-X models (O'Neill et al., 2024); and LeRobot (Cadene et al., 2024) lowers the barrier to collecting and reusing real-world datasets. Despite this progress, assembling real-robot interaction at meaningful scale remains challenging because of fragmented tooling, hardware overhead, and safety constraints (Xing et al., 2025; Park et al., 2024; Geng et al., 2025). Similarly, desktop interfaces lack open, standardized corpora and toolkits, bottlenecking vision-action pretraining (Tang et al., 2025; Chen et al., 2025). VPT (Baker et al., 2022) offers human-annotated and pseudo-labeled Minecraft trajectories but remains single-domain, while SIMA (Raad et al., 2024) demonstrates cross-game generalization through a unified interface yet keeps data proprietary. PLAICraft (He et al., 2025) advances multimodal Minecraft logging, but these efforts are environment-specific; broad cross-application generalization requires unified schemas that cover diverse desktop applications (McCarthy et al., 2025). Unlike prior single-domain or proprietary efforts, we contribute a open, unified, multi-game desktop-action dataset (31 games; $\tilde{335h}$) and an open-source toolkit, explicitly validated for transfer to embodied tasks.

Inverse Dynamics Models. Agents observe the states up to time $t-1$ and predict the action at time t. In contrast, Inverse Dynamics Models (IDMs) condition on surrounding states—past and future—to infer the action taken at time t. IDMs have been pivotal for scaling imitation learning to Internet-scale datasets, serving as pseudo-labelers for otherwise unlabeled action data (Ye et al., 2024; Bjorck et al., 2025). In robot manipulation, UniPi (Du et al., 2023) explores text-guided video generation to couple language grounding with policy learning, and LAPA (Ye et al., 2024) shows that latent action pretraining from videos can improve scalability and robustness. On the desktop side, VPT (Baker et al., 2022) trained a Specialist IDM on a human-annotated Minecraft trajectories and used it to pseudo-label thousands of hours of Minecraft gameplay on YouTube. We demonstrate the potential of a Generalist-IDM, spanning multi-game, desktop-wide settings (McCarthy et al., 2025). Our design also differs from common tick-based IDMs (Baker et al., 2022; Ye et al., 2024), which fix a prediction window (e.g., 50 ms) and thus must emit a prediction each tick—inefficient in sparse-event regimes and coarse in temporal resolution. Instead, our IDM predicts the event *and* its timestamp, enabling event-driven modeling that avoids "no-op" ticks and makes more efficient use of inference context.

3 Open-World Agents Toolkit

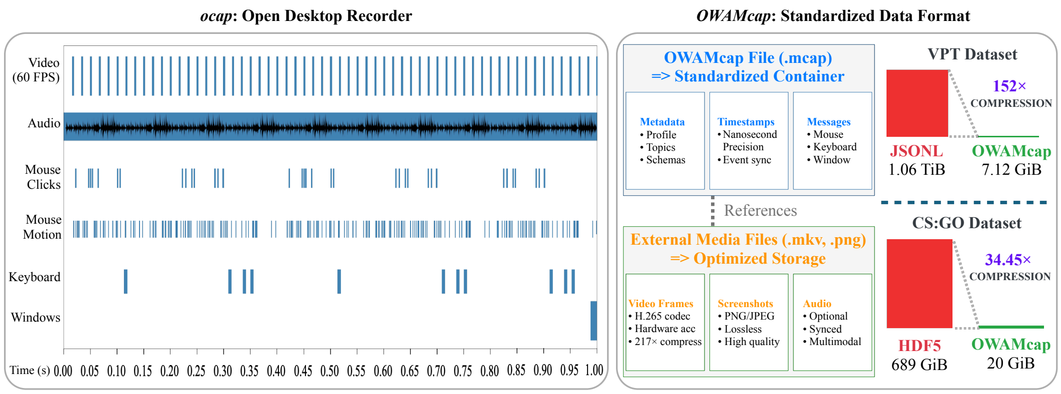

Figure 2: **OWA Toolkit's recording and storage architecture.** (Left) ocap recorder captures perfectly synchronized multimodal streams—video (60 FPS), audio, mouse events, keyboard inputs, and window states—with precise time alignment, enabling accurate reconstruction of desktop interactions. (Right) OWAMcap format revolutionizes desktop data storage through its dual-layer architecture: standardized MCAP container for crash-safe metadata and event logging, paired with external media referencing for optimized video storage using H.265 codec (217× compression). This design achieves dramatic storage reduction—152× for VPT dataset (1.06 TiB → 7.12 GiB) and 34.45× for CS:GO dataset (689 GiB → 20 GiB)—while maintaining event fidelity and enabling efficient random access for training.

We introduce the **Open-World Agents (OWA) Toolkit** alongside large-scale desktop data, establishing both the infrastructure and data foundation for embodied AI research. The toolkit provides a unified interface (Zhang et al., 2024; 2025) for capturing interaction patterns across diverse applications without domain-specific action space definitions, while our data release demonstrates the practical scalability and diversity achievable through this standardized approach.

3.1 `ocap`: Synchronized Desktop Recorder

Existing desktop recording tools lack critical features for desktop data collection. Content creation tools like OBS Studio (OBS Project) focus on streaming quality, while action modeling requires synchronized input event logging to capture the precise keyboard and mouse actions that caused visual changes. The `ocap` (Omnimodal CAPture) tool addresses this gap by capturing desktop signals in a synchronized manner, recording video, audio, keyboard, and mouse interactions with high temporal precision. Figure 2 (Left) illustrates an event timeline where these multimodal streams are well synchronized. By leveraging hardware acceleration using Windows APIs, we achieve real-time FHD/QHD recording at 60 Hz on consumer-grade GPUs with low overhead, ensuring that normal user activities remain unaffected and effectively lowering the hardware barrier for large-scale data collection. Implementation details are in Appendix A.

3.2 OWAMcap: Standardized Data Format

Prior desktop datasets suffer from storage inefficiency and poor random access capabilities. Existing approaches (Baker et al., 2022; Pearce & Zhu, 2022) either store image-encoded frames in monolithic tables unsuitable for real-time recording, or use formats like JSONL that lack proper indexing and crash-safety. To address these limitations, we introduce the OWAMcap, which extends the industry-standard MCAP format (Foxglove, 2022)—widely adopted in robotics for multimodal sensor logging—with minimal additions specific to desktop datasets. Figure 2 (Right) shows the overall structure of OWAMcap. Concretely, it adds three components to standard MCAP: (1) metadata that specifies a profile identifier, desktop-specific message schemas, and encoding conventions for downstream processors, (2) JSON schemas for desktop events (screen, keyboard, mouse), and (3) MediaRef—our external media reference convention that enables efficient video storage through file paths, URIs, or embedded data (e.g., `data:image/png;base64,...`) while maintaining MCAP compatibility. By leveraging MCAP's proven architecture, we inherit efficient indexing, crash-safe writes, and broad ecosystem support. Storage efficiency is critical for foundation model training, where disk I/O often bottlenecks throughput (Zhao & Krähenbühl, 2023; Kim et al., 2024a). OWAMcap's MediaRef enables modern video codecs (H.265) without constraining dataset layout. Table 8 shows that raw 1920×1080 captures require 5.97 MB per frame (358 MB/s at 60 FPS), while H.265 achieves 217× compression with sufficient visual quality for agent training. The MKV container provides reliable audio-video synchronization and crash resilience during recording. More detailed format comparisons are provided in Appendix A.

3.3 Optimized Data Pipeline

Training foundation models on OWAMcap data requires specialized data loading strategies to maximize throughput, as I/O and data pipeline bottlenecks have been identified as critical limitations in large-scale video model training (Zhao & Krähenbühl, 2023; Leclerc et al., 2023). We present a four-stage optimized pipeline: (1) Media transcoding with x264 parameters with fixed keyframe intervals and disabled B-frames for consistent random access; (2) Event dataset conversion to HuggingFace datasets (Lhoest et al., 2021) format for efficient sequential and random access; (3) Fixed Sequence Length (FSL) dataset generation through tokenization and packing to maximize training throughput; (4) On-the-fly media loading with adaptive batch decoding that defers expensive media operations until training time.

Adaptive Batch Decoding Strategy Our adaptive batch decoding algorithm (1) seeks to the target frame; (2) demuxes and decodes until a keyframe is encountered; (3) upon hitting a keyframe, resumes seeking to the target frame. This provides consistent performance across fine-grained, coarse-grained, and mixed access patterns.

Benchmark Setting To quantify the effect of the optimized pipeline and adaptive batch decoding, we measure performance using single-worker random-access iteration over an FSLDataset constructed from 64 five-minute Minecraft videos. We report (i) image throughput (img/s) and (ii) average disk bytes read per image (KB/img; total bytes read during iteration divided by the number of images, capturing seeking and GOP decode overhead). For batch decoding, both TorchCodec v0.6.0 and our implementation are used in a per-sample manner: for each FSLDataset sample, we issue a single batched query that requests all images within the sample at once (no cross-sample batching or parallel workers).

Under this setting, our adaptive batch decoding achieves 85.08 img/s (7.45× over baseline, 11.42 img/s) while reducing avg. disk read per image to 50.63 KB (6.04× less than baseline, 305.69 KB/img) and 16× less than TorchCodec v0.6.0 batch decoding (824.37 KB/img; TorchCodec (PyTorch Team, 2024)). Table 1 summarizes the results. Table 2 compares our approach against existing end-to-end training frameworks.

Framework Training Throughput For InternVL3-1B training on identical hardware(DGX H100), our codebase achieves 1024 img/s, while OpenVLA-OFT reaches 666 img/s. It's also noteworthy that our codebase loads 448×448 size images but OpenVLA-OFT loads smaller, 256×256 size images.

3.4 Collecting Human Demonstrations at Scale

We collect a desktop dataset that provides high-quality, synchronized multimodal signals for vision-action pretraining. While the OWA Toolkit can capture arbitrary desktop tasks (e.g., web surfing, productivity applications) with multimodal events—including the screen,

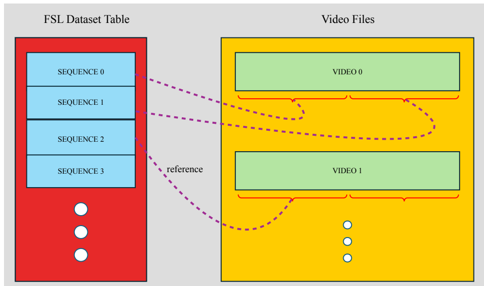

Figure 3: Our FSLDataset design, coupled with a batched decoding API, converts fine-grained random I/O into coarse, coalesced random access, thereby avoiding the limitations of large-scale filesystems that are inefficient for small random reads.

Table 1: Benchmark on FSLDataset (Minecraft, 64×5 min, 640×360 @ 20Hz). "x264 params": fixing keyframe interval and removing B-frames.

Configuration	Throughput (img/s)	Avg. Read (KB/img)
Baseline	11.42	305.69
+ x264 params	23.18	125.45
+ TorchCodec	51.39	824.37
+ **Ours**	**85.08**	**50.63**

Table 2: Framework comparison

Framework	Throughput (img/s)
OWA Toolkit	1024
OpenVLA-OFT	666

mouse, and keyboard—we focus on gameplay interactions. Gameplay data offer behavioral diversity while minimizing privacy concerns, which enables broad community contribution and data sharing. Using the `ocap` desktop recorder for efficient collection, 14 human annotators recorded the dataset. The dataset comprises 335 hours of newly collected human demonstrations across 31 games. It spans diverse genres, including 3D third-person games such as GTA V and Cyberpunk 2077, first-person games like Apex Legends and Minecraft, and 2D top-down games like Brotato and Stardew Valley. This variety captures a wide range of visual environments and interaction styles, making it well-suited for vision-action pretraining. Further details on the dataset and collection process are provided in Appendix B.

4 Generalist Inverse Dynamics Model

Collecting large-scale action data through manual demonstrations is infeasible due to prohibitive costs. The OWA Toolkit (Section 3) closes the instrumentation gap and standardizes over 2.6k hours of synchronized trajectories (Table 10), yet human capture alone remains a bottleneck relative to the ocean of unlabeled gameplay available online. VPT (Baker et al., 2022) addressed this by leveraging Inverse Dynamics Models (IDMs) to pseudo-label YouTube videos, but was limited to *Minecraft*, restricting generalization and dataset diversity. We train a **Generalist-IDM** on our multi-domain corpus collected via the OWA Toolkit, enabling generalization across heterogeneous interaction patterns. Our model can infer actions in out-of-distribution environments never seen during training, as demonstrated in Section 5.1. This capability enables pseudo-labeling of large-scale YouTube gameplay videos across diverse games, laying the foundation for internet-scale dataset collection.

4.1 Timestamp-Based Event Tokenization

We represent desktop interactions as discrete *events*, each serialized into a short token sequence bounded by <EVENT_START> and <EVENT_END>. Observation events capture screen updates (*Screen Events*), while action events represent user inputs: *Keyboard Events* (key presses/releases) and *Mouse Events* (clicks, movements, scrolls). This event-level serialization unifies heterogeneous inputs into a consistent sequential representation for transformer modeling (Vaswani et al., 2017). For example, the tokens emitted for a single event follow the format below:

$$\text{<EVENT\_START>}\{\text{TYPE}\}\{\text{TIMESTAMP}\}\{\text{DETAIL}\}\text{</EVENT\_END>} \tag{1}$$

While most existing IDMs adopt a *tick-based prediction* (Baker et al., 2022; Ye et al., 2024)—predicting actions at fixed intervals—our design employs *timestamp-based prediction*. Unlike tick-based approaches that use a fixed prediction window (e.g., 50 ms), our IDM directly predicts both the event and its timestamp, preserving the asynchronous timing captured by `ocap` and converted corpora. This design provides two key advantages. First, it maintains cross-modal alignment without resampling, allowing screen, keyboard, and mouse streams to stay synchronized even when their natural cadences differ. Second, timestamp-based prediction avoids generating empty ticks when no actions occur. By skipping unnecessary "no-op" tokens, our approach makes more efficient use of the limited inference context, enabling denser packing of relevant information and improving the efficiency of both learning and inference. A detailed specification of the event tokenization process is provided in Appendix C.

4.2 NEP-τ: Next-Event Prediction with Temporal Offset

Once raw desktop interactions are converted to event token sequences, we train the Generalist-IDM with a next-event-prediction objective. Given a trajectory consisting of observed states and actions $(o_1, a_1, o_2, a_2, \ldots, o_T)$, where each action a_t is taken at state o_t and leads to state o_{t+1}, the goal is to predict action a_t based on all preceding observations and actions. This objective enables the model to learn mappings between observed states and actions while preserving temporal dependencies within the trajectory.

$$\mathcal{L}_{\text{NEP}} = -\,\mathbb{E}_{(o_{1:T}, a_{1:T})\sim\mathcal{D}}\left[\sum_{t=1}^{T} \log P_\theta\big(a_t \,\big|\, o_{1:t},\, a_{1:t-1}\big)\right] \tag{2}$$

Inspired by IDM-K (Tot et al., 2025), which conditions on extended future trajectories to improve inverse dynamics, we adopt NEP-τ, a temporal-offset variant of NEP. Unlike IDM-K, which jointly encodes entire past and future trajectories, our method simply rearranges the (observation, action) sequences by shifting the observation window forward by τ steps. This allows the model to incorporate future observations up to τ steps ahead without encoding entire future trajectories, enhancing temporal consistency. Formally, the objective is:

$$\mathcal{L}_{\text{NEP-}\tau} = -\,\mathbb{E}_{(o_{1:T}, a_{1:T})\sim\mathcal{D}}\left[\sum_{t=1}^{T} \log P_\theta\Big(a_t \,\Big|\, o_{1:\,\min(t+\tau,\,T)},\, a_{1:t-1}\Big)\right] \tag{3}$$

4.3 Pseudo-Labeling with YouTube Gameplay Videos

We focus on pseudo-labeling gameplay videos because they are abundant, actively shared, and largely free of personally identifiable content, sidestepping the privacy concerns. YouTube gameplay footage also exhibits consistent HUD layouts and frame rates, which align well with the OWA Toolkit's event schema. Our pipeline first curates long-form gameplay uploads with permissive licenses, retrieves them at 20 Hz, and converts the frames into *Screen* events so they can be fed through the same tokenizer used for human demonstrations. Building on this, we train the Generalist-IDM using the InternVL3-1B (Zhu et al., 2025) architecture with the NEP-τ objective. The Generalist-IDM then produces the corresponding *Keyboard* and *Mouse* events via the NEP-τ objective, after which we apply consistency checks—including removing extended inactive spans as described in Appendix B—before materializing the pseudo-labels. Applying this procedure contributes 1055 hours of additional trajectories across twenty publicly shared titles, as summarized in Table 12, complementing the curated corpus described in Table 10 and Section 3. Importantly, because our model is designed to be *generalist*, we do not require any filtering of domain-specific interfaces such as inventory menus or map screens. Instead, these heterogeneous visual contexts are naturally included as part of the pseudo-labeled demonstrations, broadening the scope of training data without additional heuristics. These pseudo-labeled trajectories form the seed for scaling desktop vision-action pretraining to internet-scale data sources.

5 Results

5.1 Performance of the Generalist-IDM

In-Distribution Performance. We begin by evaluating the Generalist-IDM on six in-distribution video games spanning both 2D and 3D settings, comparing its performance to

Specialist-IDMs trained individually on each game. We employ an autoregressive inference pipeline to generate actions and evaluate model performance across multiple metrics. Further details are provided in Appendix F. As shown in Table 3 and Table 4, our Generalist-IDM achieves strong performance across all environments. Notably, it yields large gains in Pearson correlation (e.g., +39.5 points on Stardew Valley X) and Keyboard accuracy (e.g., +57.6 points on Brotato), demonstrating robust generalization over diverse control dynamics.

Game	Model	Pearson		Scale Ratio		Keypress Acc.	
		X	Y	X	Y	Kbd	Mouse
Brotato	IDM	65.92	67.56	1.04	1.04	28.80	97.59
	G-IDM	73.65	82.03	1.37	1.29	86.36	98.50
Stardew Valley	IDM	43.47	63.69	1.19	1.18	69.35	91.90
	G-IDM	82.98	75.57	1.13	1.17	74.35	96.43
Core Keeper	IDM	48.03	62.09	1.15	1.17	69.42	92.33
	G-IDM	77.25	64.55	1.43	1.51	70.00	94.01

Table 3: Evaluation results on 2D games

Game	Model	Pearson		Scale Ratio		Keypress Acc.	
		X	Y	X	Y	Kbd	Mouse
Apex Legends	IDM	65.16	57.84	1.29	1.25	67.47	99.33
	G-IDM	83.90	85.27	1.13	1.23	76.55	99.67
GTA V	IDM	63.64	81.08	1.39	1.23	58.13	94.65
	G-IDM	79.44	83.89	1.09	1.42	69.83	94.11
Minecraft	IDM	59.83	63.83	1.20	1.22	53.54	82.48
	G-IDM	80.29	78.38	1.24	1.27	60.97	91.65

Table 4: Evaluation results on 3D games

Out-of-Distribution Generalization. We evaluate the generalization of our Generalist-IDM on two unseen games: Battlefield 6 (3D) and Ogu and the Secret Forest (2D). In Battlefield 6, it matches or slightly outperforms the Specialist-IDM (achieving 63% keyboard accuracy), indicating solid transfer to an unseen FPS similar to the training set. Moreover, when provided with a few-shot prefix that fills the first 2048 tokens in our streaming inference, the predicted scale ratio improves significantly—indicating that the Generalist-IDM exhibits in-context ability to adapt to mouse sensitivity. In Ogu and the Secret Forest, it more than doubles the Specialist-IDM's performance (from about 12% to nearly 28%), showing that Generalist-IDM delivers substantial gains even under a large domain gap. Taken together, these results demonstrate that Generalist-IDM is capable of adapting across both familiar and substantially different environments.

Model	Pearson		Scale Ratio		Keypress Acc.	
	X	Y	X	Y	Kbd	Mouse
			Battlefield 6			
IDM (FT)	57.28	61.74	1.00	1.00	62.44	94.55
G-IDM (ZS)	57.36	63.17	3.13	3.56	47.75	92.11
G-IDM (FS)	56.79	63.40	1.07	1.05	52.64	93.89
G-IDM (FT)	54.90	62.89	1.06	1.04	58.55	93.41
			Ogu Forest			
IDM (FT)	–	–	–	–	11.73	–
G-IDM (ZS)	–	–	–	–	27.80	–
G-IDM (FS)	–	–	–	–	27.97	–
G-IDM (FT)	–	–	–	–	26.88	–

Table 5: Out-of-distribution performance on unseen 3D and 2D games. Note that *Ogu Forest* uses only keyboard inputs.

Figure 4: Trajectory of Battlefield 6.

5.2 Transferability to Downstream Tasks

To evaluate transferability to downstream tasks, we use the InternVL3-1B model as our backbone, which is also the architecture used in our Generalist-IDM. We train this model under two settings: **VAPT without pseudo-labels**, which uses only the human-collected dataset (259 hours), and **VAPT with pseudo-labels**, which augments the human data with a pseudo-labeled dataset generated from YouTube videos with the Generalist-IDM, resulting in a total of over 1.3K hours of training data. Further details can be found in Appendix E

Robot Manipulation. For robot manipulation, we evaluate on the LIBERO benchmark (Liu et al., 2023). As shown in Table 6, the baseline (InternVL3-1B) performs relatively poorly. In contrast, VAPT without pseudo-labels achieves a substantial improvement, reaching 96.6% on Total and 93.6% on long-horizon tasks. These results are comparable to or even surpass much larger models such as OpenVLA (7B) and SmolVLA (2.25B). Interestingly, and contrary to intuition, incorporating pseudo-labels does not provide additional gains on manipulation tasks. We attribute this to the nature of manipulation tasks, where precise human supervision is more critical than data scale and diversity, making pseudo-labels less effective. Overall, our approach demonstrates that even with only 1B parameters, it can match or outperform significantly larger policies, with particularly strong advantages on long-horizon tasks where careful action sequencing is essential.

Method	**Params**	**VLA Pt**	**Spatial**	**Object**	**Goal**	**10 (long)**	**Total**
Octo (Octo Model Team et al., 2024)	93M	Yes	78.9	85.7	84.6	51.1	75.1
OpenVLA (Kim et al., 2024b)	7B	Yes	84.7	88.4	79.2	53.7	76.5
DiT Policy (Dasari et al., 2025)	115M	No	84.2	96.3	85.4	63.8	82.4
pi0 Black et al. (2024)	3.3B	Yes	90.0	86.0	95.0	73.0	86.0
SmolVLA (Shukor et al., 2025)	2.25B	No	93.0	94.0	91.0	77,0	88.7
PI-KI (Driess et al., 2025)	300M	Yes	98.0	97.8	95.6	85.8	94.3
OpenVLA-OFT (Kim et al., 2025)	7B	Yes	97.6	98.4	97.9	94.5	97.1
Baseline (InternVL3-1B)	1B	No	94.4	97.0	93.6	54.2	84.8
+ VAPT w/o pseudo	1B	No	95.8	98.4	98.6	93.6	96.6
+ VAPT w/ pseudo	1B	No	89.6	98.2	93.8	87.2	92.2

Table 6: Results on Libero tasks (success rates, %).

Robot Navigation. For robot navigation, we evaluate on the CANVAS benchmark Choi et al. (2024), a challenging navigation benchmark that tests robustness to both misleading and precise instructions across diverse simulated environments. Compared to the baseline, our VAPT framework shows clear gains: without pseudo-labels, performance matches the baseline (75.3%), but with pseudo-labeled demonstrations it rises to 83.3%, an 8-point improvement. The effect is strongest under misleading instructions, as in *sim_orchard* (86.7% vs. 53.3%) and *sim_street_sidewalk* (73.3% vs. 40.0%), while precise instructions remain near ceiling. These results show that pseudo-labeling is highly effective for navigation tasks, where success depends more on high-level planning than on the precise low-level control required in manipulation.

Environment	**Instruction**	**Baseline**	**VAPT w/o pseudo**	**VAPT w/ pseudo**
sim_art_museum	misleading	**53.3 (8/15)**	33.3 (5/15)	**53.3 (8/15)**
	precise	**100.0 (15/15)**	93.3 (14/15)	93.3 (14/15)
sim_office	misleading	**100.0 (15/15)**	93.3 (14/15)	**100.0 (15/15)**
	precise	**100.0 (15/15)**	**100.0 (15/15)**	**100.0 (15/15)**
sim_orchard	misleading	53.3 (8/15)	53.3 (8/15)	**86.7 (13/15)**
	precise	40.0 (6/15)	53.3 (8/15)	**60.0 (9/15)**
sim_street_road	misleading	**94.4 (17/18)**	88.9 (16/18)	88.9 (16/18)
	precise	**100.0 (12/12)**	91.7 (11/12)	**100.0 (12/12)**
sim_street_sidewalk	misleading	40.0 (6/15)	53.3 (8/15)	**73.3 (11/15)**
	precise	73.3 (11/15)	**93.3 (14/15)**	80.0 (12/15)
Total	**Overall**	75.3 (113/150)	75.3 (113/150)	**83.3 (125/150)**

Table 7: Results on CANVAS tasks (success rates, %)

6 Conclusion

Embodied AI has long struggled with the prohibitive cost of collecting large-scale physical interaction data, limiting its ability to benefit from internet-scale resources. To address this challenge, we proposed using desktop interactions as an abundant and low-cost substrate for pretraining. Our contributions are threefold: (1) the OWA Toolkit, which standardizes and compresses diverse desktop data into a scalable format; (2) the Generalist-IDM, a timestamp-based inverse dynamics model that generalizes across unseen games and demonstrates a pathway toward internet-scale pseudo-labeling; and (3) VAPT, which explores the transfer of desktop-pretrained representations to robotics tasks. Leveraging 1.3K+ hours of human and pseudo-labeled data, our framework achieves 96.6% success on LIBERO manipulation and 83.3% on CANVAS navigation, demonstrating that digital sensorimotor patterns can directly improve embodied AI benchmarks. We release all our tools, datasets, and models publicly to enable the community to build upon this foundation and further investigate desktop-to-embodied transfer. These results establish desktop data as a practical and scalable resource for advancing embodied intelligence, opening a new path toward general-purpose agents without relying on prohibitively expensive physical data collection.

Reproducibility Statement

To ensure full reproducibility of our work, we release comprehensive resources and documentation. All source code for the OWA Toolkit (`ocap` recorder and OWAMcap format implementation), Generalist-IDM training, and downstream task fine-tuning is publicly available at `https://anonymous.4open.science/r/Generalist-IDM-9B13`, including detailed installation instructions and usage examples. The complete 2.6K hour desktop dataset (335 hours newly collected, 2.3K hours converted) and 1K+ hours of pseudo-labeled data are accessible through the same repository with standardized OWAMcap format specifications described in Section 3 and Appendix A. Pre-trained model weights for both Generalist-IDM and VAPT foundation models are provided along with training configurations. Hyperparameters and training schedules are detailed in Appendix E, including batch sizes, learning rates, and hardware requirements (8 H100 GPUs for IDM training). Data preprocessing pipelines, including temporal offset implementation (Section 4) and event tokenization schemes (Appendix C), are fully documented with reference implementations. Evaluation protocols and metrics are specified in Section F with corresponding evaluation scripts in the repository. For compute-constrained researchers, we release smaller dataset subsets and checkpoint models at various training stages to facilitate partial reproduction and ablation studies.

References

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. \π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Huichi Zhou, Qihui Zhang, Zhigang He, Yilin Bai, Chujie Gao, Liuyi Chen, Yiqiang Li, Chenlong Wang, Yue Yu, Tianshuo Zhou, Zhen Li, Yi Gui, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. GUI-world: A video benchmark and dataset for multimodal GUI-oriented understanding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=QarKTT5brZ`.

Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

Suhwan Choi, Yongjun Cho, Minchan Kim, Jaeyoon Jung, Myunchul Joe, Yubeen Park, Minseo Kim, Sungwoong Kim, Sungjae Lee, Hwiseong Park, et al. Canvas: Commonsense-aware navigation system for intuitive human-robot interaction. *arXiv preprint arXiv:2410.01273*, 2024.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15617–15625. IEEE, 2025.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 2024. doi: https://doi.org/10.1177/02783649241281508.

Foxglove. Evaluation of robotics data recording file formats. Technical Report, October 2021. URL `https://mcap.dev/files/evaluation.pdf`. Technical report; accessed 2025-08-23.

Foxglove. Introducing the mcap file format. Foxglove Blog, February 2022. URL `https://foxglove.dev/blog/introducing-the-mcap-file-format`. Blog post; accessed 2025-08-23.

Foxglove Developers. Mcap: serialization-agnostic log container file format, 2024. URL `https://mcap.dev`. Available from https://github.com/foxglove/mcap.

Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Charlie Tianyue Cheng, Haozhe Lou, Peihao Li, Yen-Jen Wang, et al. Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning. *arXiv preprint arXiv:2504.18904*, 2025.

GStreamer Team. Gstreamer: Open source multimedia framework. URL `https://gstreamer.freedesktop.org/`. Official GStreamer Documentation.

Yingchen He, Christian D Weilbach, Martyna E Wojciechowska, Yuxuan Zhang, and Frank Wood. Plaicraft: Large-scale time-aligned vision-speech-action dataset for embodied ai. *arXiv preprint arXiv:2505.12707*, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

ITU-T. High efficiency video coding. Recommendation H.265, International Telecommunication Union, 2024. Also published as ISO/IEC 23008-2.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Hongyeon Kim, Jinhyuk Kwon, Jaehoon Kim, Jaewon Lee, Joo-Young Kim, Jounghoo Lee, Euicheol Seo, and Jaehyuk Lee. Smart-infinity: Fast large language model training using near-storage processing on a real system. *arXiv preprint arXiv:2403.06664*, 2024a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Anikait Srinivas, Jacky Liang, Pieter Abbeel, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024b.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. Ffcv: Accelerating training by removing data bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12011–12020, 2023.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.emnlp-demo.21`.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Ajay Mandlekar, Jonathan Booher, Max Spero, Albert Tung, Anchit Gupta, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Robert McCarthy, Daniel CH Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas G Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey. *Journal of Artificial Intelligence Research*, 83, 2025.

Microsoft Corporation. Dxgi overview. URL `https://docs.microsoft.com/en-us/windows/win32/direct3ddxgi/d3d10-graphics-programming-guide-dxgi`. Microsoft Developer Documentation.

OBS Project. Obs studio. URL `https://obsproject.com/`. Open Broadcaster Software Studio - Free and open source software for video recording and live streaming.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Younghyo Park, Jagdeep Singh Bhatia, Lars Ankile, and Pulkit Agrawal. Dexhub and dart: Towards internet scale robot data collection. *arXiv preprint arXiv:2411.02214*, 2024.

Tim Pearce and Jun Zhu. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, pp. 104–111. IEEE, 2022.

PyTorch Team. Torchcodec: A pytorch library for video decoding. `https://github.com/pytorch/torchcodec`, 2024. PyTorch video decoding library.

Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In *Robotics: Science and Systems*, 2023.

Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, et al. Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*, 2024.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10740–10749, 2020.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, et al. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*, 2025.

Marko Tot, Shu Ishida, Abdelhak Lemkhenter, David Bignell, Pallavi Choudhury, Chris Lovett, Luis França, Matheus Ribeiro Furtado de Mendonça, Tarun Gupta, Darren Gehring, et al. Adapting a world model for trajectory following in a 3d game. *arXiv preprint arXiv:2504.12299*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *Forty-second International Conference on Machine Learning*.

Youguang Xing, Xu Luo, Junlin Xie, Lianli Gao, Hengtao Shen, and Jingkuan Song. Shortcut learning in generalist robot policies: The role of dataset diversity and fragmentation. *arXiv preprint arXiv:2508.06426*, 2025.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, 2024.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. *arXiv preprint arXiv:2410.11758*, 2024.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*, 2024.

Chaoyun Zhang, He Huang, Chiming Ni, Jian Mu, Si Qin, Shilin He, Lu Wang, Fangkai Yang, Pu Zhao, Chao Du, et al. Ufo2: The desktop agentos. *arXiv preprint arXiv:2504.14603*, 2025.

Hao Zhao and Philipp Krähenbühl. Training a large video model on a single machine in a day. *arXiv preprint arXiv:2309.16669*, 2023.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A OWA Toolkit Details

A.1 Format Comparison

Prior desktop datasets commonly adopt one of two storage strategies. The LeRobot dataset (Cadene et al., 2024), CS:GO dataset (Pearce & Zhu, 2022), and the CraftJarvis "minecraft-vla-sft" dataset (He et al., 2025) store image-encoded frames directly in a single, monolithic table. While this layout is sufficient for training, it is ill-suited for recording because long-table stores typically do not support efficient real-time appends. By contrast, the VPT dataset (Baker et al., 2022) packages each sample as an MP4–JSONL pair. However, JSONL lacks the ability to interleave heterogeneous, typed streams with chunking and indexing. In practice, this limitation results in poor or unavailable topic-wise random seeking and reduced crash-safety, as writes are unreliable under unexpected termination. Furthermore, datasets that rely on image encoding are substantially less storage-efficient compared to standard video codecs.

The robotics community has encountered similar multimodal logging challenges. Traditional ROS bags exhibit performance and extensibility limitations (Foxglove, 2021), which motivated the development of the MCAP format (Foxglove, 2022): an open-source container format designed with efficient indexing and compression. MCAP has since become the de facto logging standard for ROS 2 (Foxglove, 2022; Foxglove Developers, 2024), demonstrating the benefits of specialized data formats for embodied AI research. However, no equivalent standard has been established for desktop datasets, motivating our introduction of the OWAMcap format.

A.2 Compression Efficiency

OWAMcap achieves substantial storage savings across multiple datasets, demonstrating its efficiency and scalability. For the CS:GO dataset (Pearce & Zhu, 2022), replacing the original HDF5 storage with OWAMcap (mkv+mcap) reduces the storage requirement from 689 GiB to 20 GiB—a 34.45× reduction. Similarly, converting the VPT dataset (Baker et al., 2022) from JSONL to OWAMcap (mcap format) shrinks disk usage from 1.06 TiB to 7.12 GiB, achieving a 152× reduction. This significant compression arises from two different aspects: (1) from using video encoding instead of saving raw image buffer on the CS:GO dataset's HDF5 and (2) from mcap's efficiency in representing/storing information on the VPT dataset's jsonl.

A.3 Video Compression Performance

OWAMcap's another advantage of MediaRef, flexible system supporting storing media on (1) embedded (2) external media. We support storing media on both external image file and external video file. This flexible design leads opportunity to acquire significant compression efficiency of video encoding, such as H.265/HEVC. To further evaluate the benefits of video encoding, we benchmarked video compression performance for various encoding. Table 8 shows that video encoding provides superior compression rates while maintaining visual quality, enabling large-scale storage without compromising data fidelity. `ocap` is storing all media in H.265 by default and we observed similar compression ratio for recorded files.

A.4 `ocap` Architecture

The implementation of `ocap` is designed to maximize recording performance and reliability. `ocap` leverages Windows APIs, including DXGI (Microsoft Corporation) for hardware-accelerated screen capture, WASAPI for low-latency audio recording, and direct input event capture for precise keyboard and mouse logging. The media pipeline is built on GStreamer (GStreamer Team) and employs H.265/HEVC encoding (ITU-T, 2024; Sullivan et al., 2012) to achieve high compression efficiency while maintaining visual quality. The overall architecture, shown in Figure 5, integrates video, audio, and interaction streams within the OWAMcap format while ensuring synchronized, crash-safe recording.

Format	Size per Frame	Total Size	Compression Ratio
Raw BGRA	5.97 MB	4.2 GB	1.0× (baseline)
PNG	1.87 MB	1.31 GB	3.2×
JPEG (Quality 85)	191 KB	135 MB	31.9×
H.265 (keyframe 0.5s)	27.8 KB	19.6 MB	217.8×

Table 8: **Compression performance comparison for various encoding on our recorded Minecraft video.** Desktop screen capture at 1920×1080 resolution, 12 seconds @ 60 Hz. H.265 encoding uses nvd3d11h265enc for hardware acceleration. Video encoding yields significantly higher compression ratios than other formats. `ocap` is storing all media in H.265 by default and we observed similar compression ratio for recorded files. Note that size per frame for H.265 is an average over all frames, as keyframes are larger.

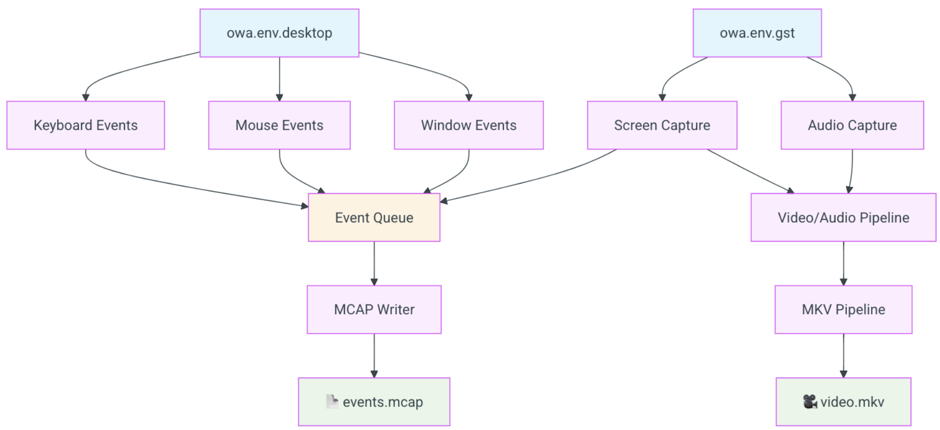

Figure 5: Architecture of `ocap` desktop recorder.

A.5 Screen Capture Performance Benchmarks

`ocap` employs H.265/HEVC encoding for video content and AAC encoding for audio streams, enabling real-time recording with minimal system overhead. Table 9 compares the capture performance of `ocap` against existing alternatives, showing that our implementation consistently achieves higher frame rates and lower CPU utilization while preserving recording fidelity.

Library	Avg. Time per Frame	Relative Speed
`owa.env.gst`	5.7 ms	1.0× (baseline)
`pyscreenshot`	33 ms	5.8× slower
`PIL`	34 ms	6.0× slower
`MSS`	37 ms	6.5× slower
`PyQt5`	137 ms	24× slower

Table 9: **Screen capture performance comparison.** Benchmarked on Intel i5-11400 with GTX 1650. `ocap` achieves 6× faster performance than common alternatives through Windows API and GStreamer integration.

A.6 Comparison with Existing Recorders

To assess feature coverage and efficiency, we compared `ocap` against commonly used desktop recording frameworks. As shown in Figure 6, `ocap` is the only system that provides synchronized multimodal recording, robust crash-safety guarantees, and efficient compression in a single framework. These advantages make `ocap` a uniquely comprehensive solution for large-scale desktop interaction logging.

Feature	**ocap**	OBS	wcap	pillow/mss
Advanced data formats (MCAP/MKV)	✅ Yes	❌ No	❌ No	❌ No
Timestamp aligned logging	✅ Yes	❌ No	❌ No	❌ No
Customizable event definition & Listener	✅ Yes	❌ No	❌ No	❌ No
Single python file	✅ Yes	❌ No	❌ No	❌ No
Audio + Window + Keyboard + Mouse	✅ Yes	⚠️ Partial	❌ No	❌ No
Hardware-accelerated encoder	✅ Yes	✅ Yes	✅ Yes	❌ No
Supports latest Windows APIs	✅ Yes	✅ Yes	✅ Yes	❌ No (legacy APIs only)
Optional mouse cursor capture	✅ Yes	✅ Yes	✅ Yes	❌ No

Figure 6: Comparison of key features between `ocap` and other desktop recording tools.

B Dataset Details

B.1 Collection and Quality Assurance

We collected the dataset using a distributed approach supported by contributions from community volunteers. To ensure participant privacy, we applied automated detection techniques followed by manual review to remove any sensitive information. Quality assurance involved both automated and manual procedures. Automated validation checked for temporal alignment issues and corrupted recordings, while human annotators manually evaluated the realism and fidelity of recorded behaviors. The final dataset captures a wide range of desktop interaction patterns, including navigation behaviors, application switching, text input, menu interactions, and multi-step task execution.

B.2 Annotator Calibration and Protocols

Before recording, contributors completed an `ocap` calibration wizard that verified refresh rate, display resolution, cursor fidelity, and input-device mapping. Annotators—either modestly compensated participants or volunteers—followed standardized game prompts covering navigation, combat, and resource-management scenarios; detailed environment statistics are listed in Table 10. All sessions were screen-captured at FHD or QHD 60 Hz with synchronized mouse and keyboard traces, and `ocap`'s turnkey workflow meant anyone could gather synchronized data with minimal setup; annotators re-ran the calibration sequence whenever their hardware changed.

Game/Application	Category	Genre	External	Hours
Apex Legends	ID	FPS	No	25.8
Euro Truck Simulator 2	ID	Driving	No	19.7
Stardew Valley	ID	Top-Down Sim	No	16.1
Cyberpunk 2077	ID	Open-World, RPG	No	14.6
Rainbow Six Siege	ID	FPS	No	13.8
Grand Theft Auto V	ID	Open-World, Driving	No	11.7
Slime Rancher	ID	Simulation	No	11.1
Medieval Dynasty	ID	Simulation, RPG	No	10.7
Dinkum	ID	Sandbox, Survival	No	10.5
Raft	ID	Survival, Co-op	No	10.3
Satisfactory	ID	Factory-Building	No	10.1
Minecraft (SP 1.21.8)	ID	Open-World, Sandbox	No	10.1
Grounded	ID	Survival, Co-op	No	10.1
Ready Or Not	ID	Tactical FPS	No	10.0
Counter-Strike 2	ID	FPS	No	9.9
Core Keeper	ID	Sandbox, Survival	No	9.4
Barony	ID	Roguelike RPG	No	9.3
Monster Hunter Wilds	ID	Action RPG	No	8.7
Brotato	ID	Top-Down Shooter	No	6.1
PUBG: Battlegrounds	ID	FPS, Battle Royale	No	4.9
Total Used for Train and test				**258.7**
Ogu and the Secret Forest	OOD	Adventure, Puzzle	No	2.3
Battlefield 6 (Open Beta)	OOD	FPS	No	2.3
Eternal Return	Collection	MOBA, Survival	No	17.3
MapleStory Worlds-Southperry (EA)	Collection	Open-World, Sandbox	No	14.1
Overwatch	Collection	FPS, Hero Shooter	No	10.3
Enshrouded	Collection	Survival, RPG	No	10.1
Ogu and the Secret Forest	Collection	Adventure, Puzzle	No	2.3
Vampire Survivors	Collection	Top-Down Platformer	No	2.8
Battlefield 6 (Open Beta)	Collection	FPS	No	2.3
Skul	Collection	Roguelike Platformer	No	2.0
PEAK	Collection	Casual/Arcade	No	1.8
Super Bunny Man	Collection	Platformer, Co-op	No	0.7
VALORANT	Collection	FPS	No	0.3
Total (Collected)				**335.6**

Table 10: **Collected desktop data statistics.** The dataset includes internally collected demonstrations across diverse games and applications.

Game/Application	Category	Genre	External	Hours
Minecraft - VPT (Baker et al., 2022)	Converted	Open-World, Sandbox	Yes	2194
CSGO - CS_DM (Pearce & Zhu, 2022)	Converted	FPS	Yes	100
Total (Converted)				**2294.0**

Table 11: **Converted dataset statistics.** Converted data from existing public benchmarks complement the collected corpus.

B.3 Converted Data

The converted dataset includes Minecraft demonstrations from Baker et al. (Baker et al., 2022) and Counter-Strike 2 data from Pearce et al. (Pearce & Zhu, 2022). These external sources were standardized into the OWAMcap format, ensuring consistency and seamless integration across different datasets.

B.4 Preprocessed Dataset

Before training, we applied preprocessing to handle temporal offsets. Specifically, after applying a temporal offset τ, only the sequences of action labels were shifted, while the observations remained unchanged. Additionally, we filtered out inactive segments where no actions occurred for extended periods to reduce noise and improve training efficiency.

B.5 Pseudo-labeled Dataset

We collect high-quality YouTube gameplay videos through a combination of targeted search and bulk download. For the search phase, we used the query template *"GAME_NAME no commentary,"* where the term *no commentary* is widely understood to indicate pure gameplay videos without additional overlays, commentary, or editing. After obtaining video links, we downloaded the videos using the open-source tool `yt-dlp`. To ensure consistency, we restricted the maximum resolution to 480p. In addition, frequent cookie renewal and a download rate cap of 62.5 Mb/s were necessary to bypass YouTube's automated bot detection mechanisms. Through this pipeline, we successfully curated over 1,000 hours of high-quality gameplay footage for pseudo-labeling. The total collected video duration per game is summarized in Table 12.

Game	**Duration (h)**
Stardew Valley	69.7
Minecraft	62.8
Monster Hunter Wilds	63.3
Dinkum	60.8
Satisfactory	59.8
Cyberpunk 2077	58.5
Medieval Dynasty	58.4
Raft	58.0
Core Keeper	58.0
Euro Truck Simulator 2	57.3
Grounded	57.2
Rainbow Six	56.3
GTA 5	54.1
Brotato	52.6
PUBG	50.7
Counter-Strike 2	49.8
Apex Legends	48.7
Slime Rancher	33.3
Ready or Not	29.0
Barony	16.7
Total	**1054.8**

Table 12: **Pseudo-labeled Duration by Game (G-IDM).** Total effective hours of successfully processed pseudo-labeled data per game.

C Event Tokenization Details

To train the Generalist IDM effectively, raw desktop interaction logs must be converted into a structured representation that the model can understand. We represent the entire interaction sequence as a stream of discrete *event tokens*. Each event corresponds to either an observation or an action. Observation events capture changes in the visual state of the environment, such as screen updates (*Screen Events*), while action events represent user inputs, including *Keyboard Events* (key presses and releases) and *Mouse Events* (clicks, movements, and scrolls).

By tokenizing data at the event level, we unify heterogeneous inputs into a consistent, sequential representation that can be modeled effectively using a single decoder-only transformer. This representation accommodates both asynchronous observations and actions while preserving fine-grained temporal alignment between them.

C.1 Event Token

We append specialized tokens to the model's vocabulary for desktop interaction modeling. **Event structure tokens** (`<EVENT_START>` and `<EVENT_END>`) delineate the boundaries of interaction sequences, while **event type tokens** (`<KEYBOARD>`, `<MOUSE>`, `<SCREEN>`) semantically categorize the modality of each event.

Numeric encoding tokens (`<0>` to `<9>`) serve multiple purposes:

- Mouse movement deltas are encoded using a configurable base system (default: [2, 10, 10, 10]), allowing efficient representation of signed values within a ± 1999 pixel range.
- Mouse scroll values are similarly quantized using base-10 tokens.
- Timestamps are encoded using temporal bases (default: [10, 10, 10]), covering a 10-second window with 10ms resolution. Timestamps are cyclic, wrapping from `999` back to `000`.

Mouse interaction tokens include:

- Sign tokens (`<SIGN_PLUS>`, `<SIGN_MINUS>`) for indicating the direction of movement deltas,
- Mouse button tokens (`<MB_0>` to `<MB_15>`) for encoding mouse button flags in hexadecimal.

Keyboard interaction tokens consist of:

- Virtual key code tokens (`<VK_0>` to `<VK_255>`) to represent all Windows virtual key inputs,
- Action tokens (`<press>`, `<release>`) to indicate key state transitions.

This factorized token design creates modular, modality-specific token spaces while maintaining a compact vocabulary. Mouse button flag definitions are provided in Table 13, and the full virtual key code mapping is shown in Table 14.

Flag Name	Hex Value	Description
RI_MOUSE_NOP	0x0000	No operation
RI_MOUSE_LEFT_BUTTON_DOWN/UP	0x0001/0x0002	Left button press/release
RI_MOUSE_RIGHT_BUTTON_DOWN/UP	0x0004/0x0008	Right button press/release
RI_MOUSE_MIDDLE_BUTTON_DOWN/UP	0x0010/0x0020	Middle button press/release
RI_MOUSE_BUTTON_4_DOWN/UP	0x0040/0x0080	Side button 4 press/release
RI_MOUSE_BUTTON_5_DOWN/UP	0x0100/0x0200	Side button 5 press/release
RI_MOUSE_WHEEL	0x0400	Vertical scroll wheel
RI_MOUSE_HWHEEL	0x0800	Horizontal scroll wheel

Table 13: Windows Raw Mouse Button Flags

C.2 Event Token Structure

All event tokens follow a consistent structure:

`<EVENT_START>` $<$ event_type $><$ timestamp $><$ event_detail $>$ `<EVENT_END>`

where:

Key Name	VK Code	Description	Key Name	VK Code	Description
LBUTTON	1	Left mouse button	KEY_0–KEY_9	48–57	’0’–’9’ keys
RBUTTON	2	Right mouse button	KEY_A–KEY_Z	65–90	’A’–’Z’ keys
CANCEL	3	Control-break	LWIN	91	Left Windows key
MBUTTON	4	Middle mouse button	RWIN	92	Right Windows key
XBUTTON1	5	X1 mouse button	APPS	93	Applications key
XBUTTON2	6	X2 mouse button	NUMPAD0–9	96–105	Numpad 0–9
BACK	8	Backspace key	MULTIPLY	106	Numpad *
TAB	9	Tab key	ADD	107	Numpad +
CLEAR	12	Clear key	SUBTRACT	109	Numpad -
RETURN	13	Enter key	DECIMAL	110	Numpad .
SHIFT	16	Shift key	DIVIDE	111	Numpad /
CONTROL	17	Ctrl key	F1–F12	112–123	F1–F12 function keys
MENU	18	Alt key	NUMLOCK	144	Num Lock
PAUSE	19	Pause key	SCROLL	145	Scroll Lock
CAPITAL	20	Caps Lock	LSHIFT	160	Left Shift
ESCAPE	27	Esc key	RSHIFT	161	Right Shift
SPACE	32	Spacebar	LCONTROL	162	Left Ctrl
PRIOR	33	Page Up	RCONTROL	163	Right Ctrl
NEXT	34	Page Down	LMENU	164	Left Alt
END	35	End key	RMENU	165	Right Alt
HOME	36	Home key	OEM_1	186	; : key
LEFT	37	Left arrow	OEM_PLUS	187	= + key
UP	38	Up arrow	OEM_COMMA	188	, < key
RIGHT	39	Right arrow	OEM_MINUS	189	- _ key
DOWN	40	Down arrow	OEM_PERIOD	190	. > key
INSERT	45	Insert key	OEM_2	191	/ ? key
DELETE	46	Delete key	OEM_3	192	‘ key

Table 14: Windows Virtual Key Codes

- `<EVENT_START>` and `<EVENT_END>` are special tokens that delimit each event
- `<timestamp>` encodes the precise timing of the event in nanoseconds
- `<event_type>` specifies the type of event (e.g., `<SCREEN>`, `<KEYBOARD>`, `<MOUSE>`)
- `<event_data>` contains event-specific information

C.3 Screen Events

Screen events capture visual observations from the desktop environment. Each screen event contains an image token sequence:

<EVENT_START><SCREEN> < timestamp >< image_tokens > <EVENT_END>

For example:

<EVENT_START><SCREEN><1><8><5><IMG_CONTEXT>^{256}<EVENT_END>

The timestamp `<1><8><5>` represents 185 time units, and the image is encoded using 256 visual tokens following the InternVL3 tokenization scheme.

C.4 Keyboard Events

Keyboard events encode key press and release actions using virtual key code tokens:

<EVENT_START><KEYBOARD> < timestamp >< vk_token >< action > <EVENT_END>

For example:

`<EVENT_START><KEYBOARD><2><0><0><VK_32><release><EVENT_END>`

This represents a key release event at timestamp 200, where `<VK_32>` corresponds to the spacebar. The action can be either `<press>` or `<release>`.

C.5 Mouse Events

Mouse events are the most complex among input modalities, as they encode continuous movement, discrete button states, and scroll actions.

<EVENT_START><MOUSE><timestamp><dx_sign><dx_magnitude><dy_sign><dy_magnitude><button_flags><scroll_data><EVENT_END>

The optional <scroll_data> token is included only when the <button_flags> field indicates the presence of scroll wheel activity.

Mouse Movement Example. Consider the following mouse event:
<EVENT_START><MOUSE><2><4><5><SIGN_PLUS><0><0><0><2><SIGN_MINUS><0><0><1><9><MB_4><MB_8><MB_0><SIGN_PLUS><0><EVENT_END>

This token sequence is decoded as follows:

Timestamp: <2><4><5> represents $2 \times 100 + 4 \times 10 + 5 = 245$ time units.

Mouse Displacement: The displacement uses separate sign and magnitude encoding:

$$\text{dx:} \quad \texttt{<SIGN\_PLUS><0><0><0><2>} = +(0 \times 1000 + 0 \times 100 + 0 \times 10 + 2) + 2 \text{ pixels} \tag{4}$$

$$\text{dy:} \quad \texttt{<SIGN\_MINUS><0><0><1><9>} = -(0 \times 1000 + 0 \times 100 + 1 \times 10 + 9) = -19 \text{ pixels} \tag{5}$$

Button Flags: <MB_4><MB_8><MB_0> encodes button states as hexadecimal digits: $0x480_{16} = 1152_{10}$.

This corresponds to:

- 0x400: Vertical scroll wheel event
- 0x080: Mouse button 4 (side button) released

Scroll Data: <SIGN_PLUS><0> indicates no scroll delta (magnitude 0).

Final Interpretation: Mouse moved $dx = +2$, $dy = -19$ pixels at timestamp 245, with scroll wheel activity and side button release.

D Model Architecture Details

For Generalist-IDM, we adopt the InternVL3-1B model (Zhu et al., 2025), which integrates InternViT as the vision encoder and Qwen2.5 (Yang et al., 2024) as the language backbone. InternVL3 is trained with native multimodal pretraining and demonstrates strong performance on video–text interleaved tasks, making it a suitable foundation for our work.

We expand the model's tokenizer by adding additional event tokens to represent events in our desktop data. Furthermore, we transfer the semantic initialization from corresponding regular language tokens to the newly added event tokens.

E Training Details

The Generalist-IDM was trained on 8 H100 GPUs (80GB) for approximately 24 hours, totaling 192 H100-hours. The entire training process incurred a cost of only $\sim$ \$800 for training on 259 hours of human-collected data, highlighting the efficiency enabled by our OWA Toolkit.

We used the following training schedules:

- **Generalist-IDM**: 5 epochs
- **Specialist-IDM**: 5 epochs
- **Generalist-IDM (fine-tuning)**: 3 epochs

- **VAPT (w/o pseudo)**: 3 epochs on the human-collected vision-action dataset
- **VAPT (w/ pseudo)**: 1 epoch on the pseudo-labeled dataset, followed by 3 epochs on the human-collected dataset

All experiments were conducted using identical hyperparameters: a global batch size of 128, a learning rate of 2×10^{-5}, and the AdamW optimizer.

F Evaluation Details

F.1 Generation Methods

We implemented an efficient autoregressive inference pipeline for predicting keyboard and mouse actions from desktop screen captures or YouTube videos. Starting from MCAP files containing synchronized, timestamped data streams (screen captures and mouse/keyboard events), we resample the events at fixed intervals (50 ms for screen and mouse events, pass-through for keyboard inputs) and tokenize them as described in Appendix C. A dynamic context manager maintains a sliding window of recent events with efficient embedding caching, using a token context length of 2048. To accelerate inference, we apply several optimization techniques, including PyTorch model compilation, FlashAttention, and mixed-precision computation with bfloat16. For multi-GPU inference, we leverage NVIDIA MPS. The generated token sequences are decoded back into structured MCAP events and subsequently evaluated. For pseudo-labeling YouTube videos, we generate MCAP files consisting of two-minute segments of screen events, excluding the first minute and last two minutes to mitigate the influence of introductions and outros.

Throughout this work, we evaluate the Generalist-IDM using fully autoregressive action decoding, both for the experiments in Section 5 and for pseudo-labeling YouTube videos. Teacher forcing was not used.

F.2 Evaluation Metrics

We evaluate the performance of Generalist-IDM using a set of fine-grained metrics that capture the correctness of predicted actions. For mouse movements, we use **Pearson correlation** (X/Y axes) and **Scale ratio** (X/Y axes) to capture the directional and spatial shape of the path and the temporal ordering of points. For discrete actions, such as keyboard presses and mouse button clicks, we report **classification accuracy**. All metrics are calculated over non-overlapping 50ms temporal bins, enabling precise alignment between predicted and ground truth event sequences.

The Scale ratio metrics, including scale_ratio_x and scale_ratio_y, measure relative scaling differences between ground-truth and predicted mouse movements along the x and y axes. They quantify how much predictions are stretched or compressed compared to the source movements.

Formally, for n bins with source vectors $s_i = (s_{i,x}, s_{i,y})$ and predicted vectors $d_i = (d_{i,x}, d_{i,y})$:

$$\text{scale\_ratio\_x} = \frac{\frac{1}{n}\sum_{i=1}^{n} |s_{i,x}|}{\frac{1}{n}\sum_{i=1}^{n} |d_{i,x}|}, \tag{6}$$

$$\text{scale\_ratio\_y} = \frac{\frac{1}{n}\sum_{i=1}^{n} |s_{i,y}|}{\frac{1}{n}\sum_{i=1}^{n} |d_{i,y}|}. \tag{7}$$

To ensure interpretability, ratios < 1 are inverted so that all values are ≥ 1.

Interpretation:

- 1.0: perfect scaling match between source and prediction
- > 1.0: scaling mismatch, where larger values indicate greater discrepancy

G Downstream Details

G.1 Robot Manipulation

We train a manipulation policy identical to *openvla-oft* (Kim et al., 2025), except that the vision–language backbone is replaced with InternVL3-1B (or its OWA variant). The policy retains the L1 regression head for continuous action prediction, employs bidirectional attention in the policy stack, and uses parallel decoding with action chunking (chunk size $K = 8$).

The inputs consist of a third-person image, a wrist-camera image, the robot proprioceptive state, and a language instruction, resulting in two images per step (exocentric and egocentric). Training uses a filtered dataset where unsuccessful demonstrations are removed.

Optimization follows the *openvla-oft* recipe: LoRA rank 32, learning rate 5×10^{-4}, batch size 8, and image augmentation enabled. Linear decay is applied after 10,000 steps, with a total training budget of 15,005 steps. Checkpoints are saved every 1,000 steps, keeping both periodic and latest versions.

Training is conducted on a single node with 8 GPUs via `torchrun`, with the same launch flags as *openvla-oft*, except for swapping the backbone to InternVL3-1B/OWA.

Evaluation is performed on the LIBERO benchmark (Liu et al., 2023), which includes four suites of manipulation tasks: (1) Spatial, varying scene layouts with fixed objects; (2) Object, varying the set of objects in a fixed scene; (3) Goal, testing goal-conditioned behavior; and (4) Long (LIBERO-10), long-horizon compositional tasks involving diverse objects, layouts, and goals. We report the average success rate over 500 episodes for each suite.

G.2 Robot Navigation

We established a baseline following CANVAS (Choi et al., 2024) by training an InternVL3-based model architecture on the COMMAND dataset. The baseline model was initialized with the default InternVL3 weights, whereas the VAPT w/o pseudo and VAPT w pseudo were trained from pretrained weights. All models were trained with full parameter unfreezing.

For optimization, we employed AdamW with separate learning rates: 2×10^{-5} for the LLM, and 5×10^{-5} for both the projector and vision encoder. Training was conducted with a batch size of 32 over 5 epochs, and each model utilized 128 waypoint tokens. In the main experiments, inference of CANVAS models was performed on a single NVIDIA H100 GPU. All evaluations were repeated three times per test dataset with randomized initial orientations.

H Ethics Statement

We acknowledge and adhere to the ICLR Code of Ethics.

Human Data Collection. Our dataset was collected from 14 volunteer annotators who provided informed consent for gameplay recordings. Participants were fully informed about screen capture and input logging procedures and could withdraw at any time. All data underwent automated and manual review to remove any personally identifiable information before research use.

Public Data Usage. We processed only publicly available YouTube videos with permissive licenses for pseudo-labeling. Our focus on gaming content inherently minimizes privacy concerns compared to general desktop recording, as gaming interfaces rarely contain sensitive personal information.

Transparency and Responsible Release. To ensure responsible use, we will publicly release all code, data collection tools, and model weights with comprehensive documentation. We acknowledge that vision-action models could have dual-use potential; however, our focus on standardized gaming environments and transparent methodology helps mitigate misuse risks. Our computational approach (requiring only modest GPU resources) democratizes access while reducing environmental impact compared to large-scale training paradigms.

I Limitations

We evaluate our approach exclusively on simulation benchmarks to establish reproducible baselines, with real robot validation deferred to future work. The differential impact of pseudo-labels (improving navigation but degrading manipulation) suggests task-specific transfer mechanisms that require further investigation. Our dataset focuses primarily on gaming interactions, which may not capture the full spectrum of desktop activities relevant to general-purpose robotics. Despite these constraints, our framework democratizes embodied AI research by reducing storage requirements by 152× and training costs to under $1000, making large-scale vision-action pretraining accessible to resource-limited academic labs.