Appendix

For more video results, please check the local HTML pages.

Appendix Contents

A Additional Dataset Details **14**
- A.1. Pipeline Details . . . 14
- A.2 Dataset statistics . . . 14

B Additional Method Details **14**
- B.1. Text-to-Video Model . . . 14
- B.2. Training Details . . . 15
- B.3. Distillation Procedure Details . . . 15

C Additional Benchmark Details **16**
- C.1. Evaluation Tasks . . . 16
- C.2. EgoEditBench-Human Alignment . . . 17

D Evaluation Details **17**
- D.1. Baseline Details . . . 17
- D.2 Metrics Details . . . 18

E Additional Results **18**
- E.1. Additional In-the-Wild Results . . . 18
- E.2. Additional Comparison to Baselines . . . 18
- E.3. Additional Distillation Ablation Results . . . 18
- E.4. Additional Dataset Ablation Results . . . 19

F. Failed Experiments **19**

A. Additional Dataset Details

A.1. Pipeline Details

Video selection. We select videos from Ego4D [19] and EgoExo4D [20] according to the following criteria. The video must be captured with one of the following camera models: GoPro Hero 4, GoPro Hero Black 7, GoPro Hero Black 8, GoPro Hero Black 9, GoPro Hero Silver 7, or GoPro Max, and it must be monocular rather than binocular. Furthermore, to ensure that the videos are sharp and visually informative, we apply additional filtering based on jitter scores and aesthetic scores.

Hand mask segmentation. We employ WiLoR [47] for hand detection, applying a confidence threshold of 0.75 on a frame-by-frame basis. From the detected frames, we select the three with the highest confidence scores and use their hand masks to generate point prompts for SAM2 [52], enabling the creation of dense and temporally consistent hand masks across frames.

Object names extraction. We employ Qwen2.5-VL-32B [2] to extract object names. For inputting the video, we sub-sample the video with 2 fps into frames and ask the model what is held by the hand in the video.

Object mask segmentation. We use the extracted object names to prompt Grounded-SAM [54] to generate object masks for each frame. A mask confidence threshold of 0.4 and a text threshold of 0.35 are applied. Each object mask is then filtered based on its distance to the hand skeleton and the edge of the hand mask. If multiple objects are detected in a single frame, we select the mask closest to the hand mask. To refine the prompts, we exclude regions corresponding to the hand masks and use the remaining object mask areas from the top three frames with the highest confidence scores to generate point prompts for SAM2, enabling the creation of dense and consistent object masks.

Object Editing. We use the object masks to generate rectangular masks for each frame, expanding them with an additional margin. A Gaussian dilation with a kernel size of 50×50 px is then applied to these rectangular masks. After excluding the regions corresponding to the hand masks, the resulting final masks are used by VACE-14B [24] to produce the edited videos. The resolution of the generated videos is 1920×1104 px.

A.2. Dataset statistics

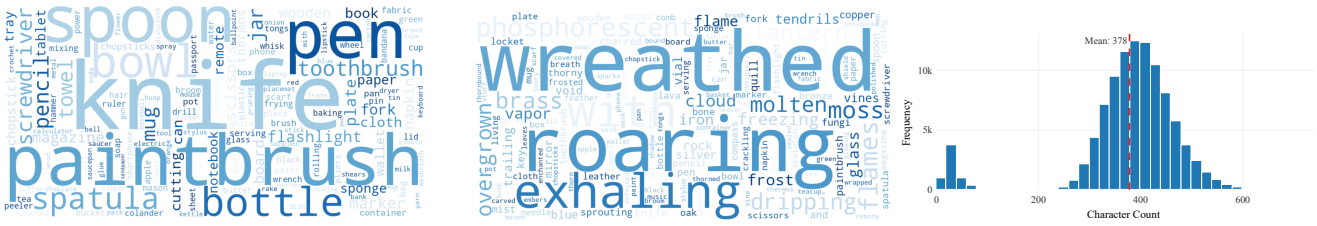
EgoEditData contains approximately 93,600 video editing pairs, each consisting of a source video, a target video, and a corresponding instruction prompt. Our dataset is built upon two egocentric video datasets with 6.4% of the videos originating from EgoExo4D [20], and the remaining 93.6% originating from Ego4D [19]. EgoEditData focuses on object manipulation in egocentric videos.

Figure 7a presents the word cloud of source objects descriptions, while Figure 7b shows the word cloud of target objects descriptions. Overall, thanks to the sampling of target object names through GPT-5 Mini [45] used during the *Object Editing* phase of dataset construction, the dataset contains 13,632 distinct target objects compared to 3,199 unique source objects, enhancing diversity. We also report in Figure 7c the distribution of the instruction prompt lengths. With a mean of 378 characters, most prompts contain detailed instructions specifying target object, scene details and style, while certain tasks not requiring such context such as *Remove/Add Object* have more concise prompts like *"Remove/Add an {object}."*. Finally, Figure 8 shows the ten most frequent scenarios in EgoEditData as categorized by the original egocentric datasets, highlighting a balanced distribution across different types of scenes and scenarios.

B. Additional Method Details

B.1. Text-to-Video Model

State-of-the-art methods [28, 60, 71] in text-to-video generation commonly adopt the latent diffusion [21] paradigm,



(a) Source Word Cloud (b) Target Word Cloud (c) Prompt Length Distribution

Figure 7. Overview of dataset statistics. (a) and (b) illustrate the word clouds for the source and target object descriptions respectively. While source object descriptions focus on everyday objects, target descriptions often contain descriptions of materials and special effects. (c) shows the distribution of prompt lengths. Short prompts in (c) correspond to instructions not requiring context such as *"Remove/Add an {object}"*.

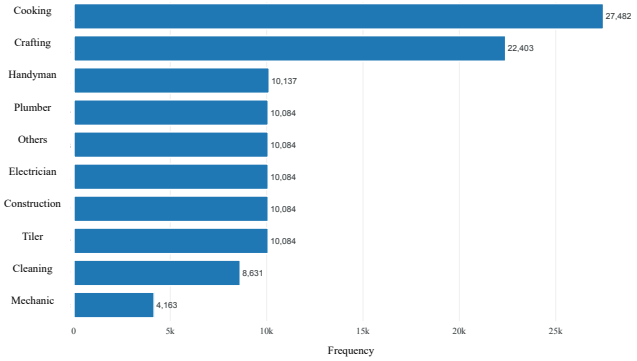


Figure 8. Distribution of most frequent scenarios in EgoEditData according to the original egocentric datasets [19, 20] categorization.

DiT-like [46] transformer backbones, and the Flow Matching [37, 41] framework. Our pretrained text-to-video model follows the same paradigm. We model videos in the latent space of a Wan 2.1 autoencoder [60] which performs an $8 \times 8 \times 4$ compression along the height, width, and time dimensions respectively. Input and output linear projections perform 2×2 spatial patchification to further increase compression. The main backbone consists of a 10.7B transformer backbone composed of 32 identical transformer blocks with 4096 hidden dimensions and 32 heads. Each block employs self attention, followed by cross attention for text conditioning, and a final MLP, and uses modulation [9] for timestep conditioning. QK normalization [15] and Flash Attention [12] are used for every attention operation to improve stability and speed. Text tokens are extracted using a combination of pretrained T5 [51] and CLIP [50] text encoders. The model performs inference in 40 steps using an Euler solver and is trained at a maximum resolution of 512 px.

B.2. Training Details

Training configuration.

We finetune the pretrained text to video model for 30k iterations with a total batch size of 96 videos using 48 H100 GPUs. We use AdamW [42] optimizer with lr 1e-5, a learning rate scheduler with linear warmup of 1000 steps, weight decay of 0.1, beta values of 0.99 and exponential moving average of 0.9999.

Editing data corpora. We collect a corpus of editing data to supplement EgoEditData for the non-egocentric case. We consider the following publicly available editing datasets: GPT-Image-Edit-1.5M [63], ShareGPT-4o-Image, Complex-Edit, HQEdit, OmniEdit [64], and UltraEdit. In addition, we employ internal data generation pipelines for the creation of additional editing training data. The pipelines employ Qwen-Image-Edit [65] to generate 2M image editing examples, and a combination of models including Wan VACE [24], Wan-Animate and MiniMax-Remover to generate an additional corpus of 210k video editing data containing edit categories including object addition and removal, object substitution, human edits. Generated editing pairs are assessed through a combination of automated and manual filtering. In addition, the pipeline synthesizes 1.1M video editing data pair by considering natural videos and pairing them with depth estimation, pose estimation, edge estimation, and optical flow estimation models. The training data is composed of multiple datasets. During training, we apply importance sampling with weights of 28% for EgoEditData, 52% for other video editing datasets, and 20% for image editing datasets. Within each category, individual datasets are weighted proportionally to their sizes.

B.3. Distillation Procedure Details

The distillation procedure is responsible for enabling real-time and low-latency execution of EgoEdit starting from the finetuned video editing method. The original model performs inference of 5s, 512×384px, 16 fps videos using 40 steps with classifier-free guidance, for a total of 80 model evaluations (NFEs). As every model evaluation takes 86ms on a single H100 GPU, the resulting throughput is of only 11.9fps, which additionally decreases to 9.68fps when

considering autoencoder source video encoding and target video decoding times, far from the 16fps required for real-time performance (see Table 2).

The first phase of distillation thus performs step and guidance distillation using DMD to yield a model capable of producing a 5s 512×384px 16fps video in 4 NFEs, for a $20\times$ increase in model throughput. While the model now possesses a throughput of 43.5fps after application of the autoencoder, the full video needs to be sampled before the first frame can be displayed, and the autoencoder needs to be run on the full source and target videos, creating a first frame latency of 6.93s, inhibiting interactive usage. We perform DMD distillation for 3500 iterations with a total batch size of 64 using 32 H100 GPUs. We use the AdamW [42] optimizer with lr 1e-6 for the generator and 4e-7 for the fake score model, 5 steps per generator update, weight decay of 0.1, beta values of 0.99 and exponential moving average of 0.99.

To enable interactive usage, Self Forcing [23] performs distillation of the bidirectional DMD model into an autoregressive model capable of producing videos in a chunk-by-chunk fashion. In this way, the first frame can be displayed to the user right after the first chunk is sampled and decoded by the autoencoder, reducing first frame latency to 855ms. We use a configuration of Self Forcing where latent frames are denoised in chunks spanning three consecutive latent frames [23]. We conduct Self Forcing distillation starting directly from the DMD distilled checkpoint for 3500 iterations with a total batch size of 64 using 64 H100 GPUs. We use an AdamW [42] optimizer with lr 1e-6 for the generator and 4e-7 for the fake score model, 10 steps per generator update, weight decay of 0.1, beta values of 0.99 and exponential moving average of 0.99. We find that the model can quickly adapt to autoregressive modeling when initialized from the DMD checkpoint, so we skip the ODE initialization phase [23]. We use 7 chunks for generation and each chunk contains 3 latent frames with 21 latent frames in total. Following [23], to imitate the generation for longer videos, during training, we mask the first chunk of the latent frames when generating the last chunk and we use a window size of 5 chunks as condition during inference.

C. Additional Benchmark Details

C.1. Evaluation Tasks

Starting from 100 unique egocentric source videos, we construct a benchmark spanning 15 editing tasks. We ensure diversity by clustering source objects and scenario names using sentence embeddings and then uniformly sample source videos across clusters. Conditioned on each source video, its caption, and the source object, we use a GPT-5 Mini [45] to produce task-specific instruction prompts. Below we include details on constructing the source video and instruction prompt for each editing task.

- **Remove Object:** We aim to evaluate the model's ability to remove certain objects from the video while keeping the other parts of the video consistent with the source video. We select 50 source videos from the EgoEditData and their instruction prompts. We include them in the benchmark and remove them from the training set.
- **Add Object:** We aim to evaluate the model ability to insert a specified object into the scene while keeping the rest of the video consistent with the source. We select 50 source videos from the EgoEditData together with their instruction prompts. We use source videos from the *Add Object* task, where target object is absent and synthetically removed following the EgoEditData pipeline. We include these in the benchmark and exclude them from the training set.
- **Change Object:** We evaluate the model's ability to alter a specified object, either by modifying an attribute or replacing it with a new specified object while keeping the rest of the video consistent with the source. For each sampled source video, we create four instruction prompts: two that perform a pure replacement (*object:* $a \rightarrow b$) and two that pair the replacement with an added effect on the new object (e.g., fire or glow). In total, we obtain 400 videos for evaluating the *Change Object* task. Instruction prompts are designed following EgoEditData pipeline.
- **Change Background:** We evaluate the model's ability to replace or edit the background while preserving foreground identity and motion. To construct instructions, we provide GPT-5 Mini [45] with a few source frames and the video caption and request an editing instruction prompt with a semantically compatible target background.
- **Change Camera Pose:** We assess recomposition via a specified camera trajectory (pan/tilt/dolly/zoom) without altering scene events. Instruction prompts are generated by GPT-5 [45] Mini by prompting it with the video caption which contains a description of the camera pose in the source video.
- **Add Effect:** We evaluate adding post-processing effects while preserving scene content. These effects usually operate as global filters (e.g., motion blur, VHS, glow, film grain) that are independent of the particular video. Accordingly, we construct instructions by prompting GPT-5 Mini—primed with a few in-context examples from EditVerseBench [26] and to ask it to propose a diverse pool of effect editing instructions, from which we randomly sample one per clip.
- **Stylization:** We mirror the procedure of *Add Effect* by providing GPT-5 Mini with EditVerseBench [26] examples and ask it to propose diverse set of style editing instructions. We sample one randomly per source video.
- **Reasoning:** We evaluate edits that rely on spatial and

temporal reasoning. Given the source video and its caption, we prompt GPT-5 Mini [45] to propose an editing instruction tied to an explicit anchor (event or timestamp) or disambiguating relations (e.g., left-of/behind/before/after). The instruction deliberately avoids explicitly naming a unique target and instead focuses on giving an instruction where the correct object must be inferred from context.

- **Depth-to-Video:** We convert the source video into a depth map using Depth Anything [69], and construct the instruction prompt as (*"Turn the depth map into a video with the following description: caption."*) where the caption is the source video caption describing the appearance of the scene.
- **Sketch-to-Video:** We convert the source video into Canny edge maps using OpenCV, and construct the instruction prompt as (*"Turn the canny edge map into a video with the following description: caption."*), where caption is the source video caption describing the scene's appearance and layout.
- **Pose-to-Video:** We convert the source video into 2D human poses using DWpose [70], and construct the instruction prompt as (*"Turn the DWpose pose map into a video with the following description: ⟨caption⟩."*), where caption is the source video caption specifying the subject's identity, attire, and overall scene appearance.
- **Video-to-Depth:** We prompt the model to convert a video into a temporally consistent depth map. For this task, we use fixed instruction prompt (*"Turn the video into a depth map."*).
- **Video-to-Sketch:** We use a fixed instruction prompt of (*"Turn the video into a Canny edge map."*).
- **Video-to-Pose:** We use a fixed instruction prompt (*"Turn the video into a DWpose pose map."*).
- **Combined (Multi-Task):** We compose multiple editing prompts from the same source video (e.g., Pose-to-Video + Change Background + Stylization) by sampling a subset of instruction prompts and prompting a GPT-5 Mini [45] to compose a single instruction prompt that combines all of the tasks together.

C.2. EgoEditBench-Human Alignment

To evaluate the reliability of the VLM score employed in EgoEditBench, we conduct a study on EgoEditBench comparing VLM and human preference alignment. Given 30 randomly sampled benchmark element per category, and a baseline method, we conduct VLM evaluation of EgoEdit and the baseline following EgoEditBench protocol and, for each sample, assign VLM preference to the method with highest VLM score. Simultaneously we ask a human evaluator to express preference for each sample for our method or the baseline. Results are shown in Table 4. VLM and user preferences are in high agreement, with 86.2% and 84.9% of cases on average, respectively when evaluated against LucyEdit and InsV2V. We thus rely on VLM score as the main benchmark metric.

D. Evaluation Details

D.1. Baseline Details

To ensure a fair comparison, we use each baseline's default inference hyperparameters including the number of frames, frames-per-second (FPS), spatial resolution, and inference settings such as guidance scale and sampling steps. Below, we include the specific hyperparameters we used for evaluating each baseline:

- **TokenFlow.** We rely on Stable Diffusion 1.5 as the backbone. We use 16 frames at a resolution of 512×512 px. We start editing from noise level of 0.9. We use the default guidance scale of 7.5 for 50 steps.
- **STDF.** We use 24 frames at 10 fps and 40 inference steps and 10 optimization steps. We use a guidance scale of 10 and inference at the resolution of 576×320.
- **SENORITA (Senorita-2M).** We use 33 frames at 8 fps and 768×448 resolution, with guidance scale of 4 and 30 inference steps. We use CogVideoX as the backbone. To obtain the edited first frame, we use the first frame generated by our model instead of relying on pretrained ControlNets, following [26]. We crop and resize the first frame generated by our method to the default resolution of SENORITA.
- **AnyV2V.** We use 16 frames at 8 fps and 512×512, guidance of 9, with 100 inversion steps plus 50 edit steps. Similarly, we use the first frame generated by our model instead of relying on pretrained ControlNets, following [26]. We crop and resize the first frame generated by our method to the default resolution of AnyV2V.
- **InsV2V.** We use 32 frames at 15 fps and 384×384, with the default guidance for the video branch (1.2) and text branch (7.5), over 20 inference steps.
- **Lucy-Edit.** We use 81 frames at 15 fps with 832×480 resolution, guidance scale of 5, and 50 inference steps.
- **EditVerse.** Since EditVerse is a closed source model, we only compare on EditVerseBench, where we rely on its published samples. We omit the results on EgoEditBench since we do not have access to the model to generate the required results.
- **StreamDiffusion.** We use the image-to-image editing pipeline for the video editing task. We process 81 frames and 16 fps at the resolution 832×480.
- **StreamDiffusionV2.** We use the streaming setup at 832×480 with an 81 frames sequence and 16 fps, using a very light 2–4 denoising steps.

	VLM Mean Score			Preference (VLM)		Preference (User Study)		Agreement (%)	
Task	**EgoEdit**	**LucyEdit**	**InsV2V**	**vs LucyEdit**	**vs InsV2V**	**vs LucyEdit**	**vs InsV2V**	**LucyEdit**	**InsV2V**
Add Object	7.83	4.12	5.47	29 (100%)	29 (97%)	29 (100%)	29 (97%)	100.0	93.3
Change Camera Pose	7.11	6.93	7.30	20 (67%)	14 (47%)	27 (90%)	28 (93%)	70.0	40.0
Change Object	7.61	6.49	4.09	23 (77%)	26 (87%)	29 (97%)	30 (100%)	80.0	86.7
Change Background	7.28	5.10	4.46	29 (97%)	27 (90%)	28 (93%)	29 (97%)	90.0	86.7
Combined (Multi-Task)	7.96	6.36	4.66	30 (100%)	29 (97%)	29 (97%)	30 (100%)	96.7	96.7
Depth-to-Video	8.53	7.44	5.72	30 (100%)	30 (100%)	30 (100%)	30 (100%)	100.0	100.0
Add Effect	6.41	5.29	5.96	24 (80%)	19 (63%)	24 (80%)	27 (90%)	80.0	73.3
Video-to-Pose	7.90	4.63	4.39	30 (100%)	30 (100%)	28 (93%)	28 (93%)	93.3	93.3
Pose-to-Video	8.58	7.18	4.77	28 (93%)	30 (100%)	28 (93%)	29 (97%)	86.7	96.7
Reasoning	6.69	5.30	4.85	24 (80%)	23 (77%)	25 (83%)	29 (97%)	63.3	73.3
Remove Object	6.72	5.04	5.71	25 (83%)	23 (77%)	26 (87%)	29 (97%)	76.7	73.3
Sketch-to-Video	8.81	6.31	5.54	30 (100%)	30 (100%)	27 (90%)	30 (100%)	90.0	100.0
Stylization	7.88	4.68	6.70	29 (97%)	25 (83%)	29 (97%)	29 (97%)	93.3	80.0
Video-to-Depth	8.80	4.00	4.80	30 (100%)	30 (100%)	28 (93%)	28 (93%)	93.3	93.3
Video-to-Sketch	9.00	3.78	4.34	30 (100%)	30 (100%)	24 (80%)	26 (87%)	80.0	86.7
Overall	7.76	5.44	5.24	411 (91%)	395 (88%)	411 (91%)	431 (96%)	86.2	84.9

Table 4. Study on EgoEditBench comparing VLM and human preference alignment. We conduct VLM evaluation following EgoEditBench protocol and for each sample, assign VLM preference to the method with highest VLM score. Simultaneously we ask a human evaluator to express preference for a sample from one of two compared methods. We find VLM and user preference to be in agreement in 86.2% and 84.9% of cases on average, respectively when evaluated against LucyEdit and InsV2V.

D.2. Metrics Details

We closely follow [26] when computing metrics for our benchmark. We evaluate edit quality with a VLM-based judge, overall video quality with PickScore, text alignment at the frame and video levels, and temporal consistency with CLIP and DINO. Because frame- and video-level text alignment are highly correlated in our setting, we report only the video-level score for brevity. We also find the CLIP- and DINO-based consistency metrics to be strongly correlated, so we rely on the CLIP consistency metric for temporal consistency. Among all metrics, we found the VLM score to align well with human judgment (see Table 4), so we use it as our primary quality signal. For EgoEditBench, we use exactly the same evaluation prompts as [26] when computing the VLM editing-quality score.

E. Additional Results

E.1. Additional In-the-Wild Results

We present in Figure 10 and Figure 9 additional in-the-wild qualitative examples produced by EgoEdit-RT in real-time on a single H100 GPU for the egocentric and exocentric cases respectively. EgoEdit-RT can perform complex edits such as transforming markers placed on the floor into pillars of the Golden Gate Bridge, model fluid effects, change a location into a haunted mansion, add animals that interact with the environment and the user, or perform stylization.

E.2. Additional Comparison to Baselines

We present additional qualitative results in Figure 11 comparing EgoEdit to baseline video editing methods. EgoEdit and its real-time variant EgoEdit-RT are capable of performing a range of editing tasks including object attribute changes, object replacement, object insertion, style transfer, background changes, and conversion to depth map. We observe concurrent offline methods [58] and real-time video editing methods [17, 27] to often fail in the egocentric case, with failure modes often manifesting as inability to produce any change over the source video, or modifying the input beyond what is requested in the editing instruction. In addition, Table 5 reports per-category VLM scores on EgoEditBench for our method and baselines.

E.3. Additional Distillation Ablation Results

Figure 12 shows additional qualitative results comparing variants of EgoEdit at different stages of distillation: the starting 40-step (80 NFEs) video editing checkpoint obtained after finetuning of the pretrained video generation model (EgoEdit), the 4-step (4 NFEs) variant obtained as a result of the DMD distillation phase (EgoEdit-DMD), and the real-time streaming variant obtained at the end of Self Forcing training (EgoEdit-RT). We observe similar qualitative performance among the distilled variants as shown in the figure. We observe EgoEdit-RT to occasionally present temporal shifts at the boundaries between different chunks of predicted frames. In addition, when qualitatively evalu-

Method	Add	Cam. Mov.	Change	Chg. BG	Combined	Depth2Vid	Effect	Vid2Pose.	Pose2Vid	Reasoning	Remove	Sketch2Vid	Stylization	Vid2Depth	Vid2Sketch	Overall
TokenFlow	5.56	5.67	5.37	5.02	5.39	4.05	5.80	5.62	0.52	5.24	5.49	3.73	6.01	5.88	5.54	4.99
STDF	4.59	4.40	4.83	4.82	5.27	4.71	4.67	5.33	2.58	4.13	4.55	4.00	5.33	5.17	4.45	4.59
Señorita-2M	7.85	7.34	7.35	6.64	7.10	8.48	7.24	6.68	7.85	7.61	6.93	8.49	6.75	8.41	8.05	7.52
AnyV2V	7.72	7.55	7.22	6.58	6.42	7.14	**7.32**	6.66	2.62	7.52	**7.21**	5.29	6.58	7.35	7.56	6.72
InsV2V	5.67	7.26	3.99	4.42	4.85	5.60	6.30	4.39	4.69	4.41	5.52	5.55	6.73	4.79	4.40	5.22
Lucy Edit	4.31	6.92	6.25	4.67	6.15	7.31	5.16	4.49	7.13	5.68	4.81	6.52	4.65	3.88	3.72	5.46
StreamDiffusion	3.95	3.06	3.44	5.29	5.60	5.53	3.62	5.43	2.66	3.26	2.91	5.20	6.32	4.45	4.02	4.31
StreamDiffusionV2	1.63	1.76	2.42	2.21	3.62	4.66	2.42	3.52	2.45	1.10	1.02	3.33	3.19	3.53	1.37	2.55
EgoEdit	**7.89**	6.79	7.84	**7.45**	**7.74**	**8.57**	6.32	7.87	**8.57**	6.79	6.82	**8.70**	**7.46**	**8.70**	**8.93**	**7.76**
EgoEdit-DMD	7.91	6.76	8.04	5.36	7.29	8.01	5.92	8.07	5.60	7.16	7.46	7.76	6.45	8.95	8.96	7.42
EgoEdit-RT	7.79	6.67	**8.09**	7.57	7.83	8.52	6.40	**8.48**	8.31	7.54	4.01	7.92	7.43	8.89	8.98	7.83

Table 5. Breakdown of per-category EgoEditBench VLM scores for EgoEdit and baselines.

Method	VLM ↑	PS ↑	TA ↑	TC ↑
EgoEdit	7.80	**19.09**	**16.91**	96.74
EgoEdit-DMD	7.42	18.95	16.52	**96.87**
EgoEdit-RT	**7.83**	19.04	16.49	96.49

Table 6. Quantitative comparison of EgoEdit variants on EgoEditBench: "VLM" is VLM evaluation score, "PS" is Pick Score, "TA" is Text Alignment, "TC" is Temporal Consistency. EgoEdit-RT is the real-time version, and EgoEdit-DMD is the bidirectional DMD variant.

ated on in-the-wild cases, distilled variants (EgoEdit-DMD, EgoEdit-RT) exhibited a lower capacity of handling complex out-of-distribution editing instructions. In addition, Table 6 reports full evaluation scores on EgoEditBench for all distilled variants of EgoEdit.

E.4. Additional Dataset Ablation Results

In Figure 13, we show qualitative examples produced by different variants of EgoEdit trained on our training data mixture same as the main experiment (Table 1) with reduced portions of EgoEditData. As the portion of EgoEditData increases, we notice increase quality of edits and improved alignment to the source video. We additionally evaluate whether increasing the amount of data in EgoEditData can result in improved performance in non-egocentric benchmarks. Table 7 compares variants of EgoEdit trained with a data mixture same as the main experiment including versions of EgoEditData of reduced size, on the EditVerseBench benchmark, which mostly consists of exocentric videos. We find that the introduction of EgoEditData consistently improves the overall benchmark score, with particular regards to the *Reasoning*, *Remove*, *Camera Movement*, and *Change* categories. We speculate these improvements originate from strict quality filtering and descriptive captions, whose benefits extend beyond the egocentric case.

F. Failed Experiments

Usage of unfiltered data. We initially experiment with a video editing dataset corpora consisting of lightly filtered video editing pairs. In this setting, we notice weak video editing performance, with the model often reproducing artifacts encountered in low-quality video editing data pairs such as failure of an object in being replaced with a different object, or failure in adding an object. This motivates us to integrate extensive automated and manual curation into EgoEditData at the expense of dataset size, which resulted in increased editing performance.

Usage of detailed editing instructions. We initially experiment with simple editing instructions for EgoEditData that are obtained by filling templates with known source and target object names such as *"Replace ⟨source object⟩ with ⟨target object⟩"*, where source and target objects are represented by strings obtained during the *Object names extraction* and *Object Editing* stages of the data curation pipeline. We observe low editing performance when training on such prompts. Such captions suffer from several issues: (i) their variety is limited, (ii) object names extracted in the *Object names extraction* stage are often short, (iii) objects in generated target videos produced in the *Object Editing* stage might not faithfully correspond to the requested edited object description. After training on editing instructions using GPT-5 Mini [45], we observe increased ability to follow instruction prompts.

% of EgoEditData	Add	Cam. Mov.	Change	Chg. BG	Combined	Depth2Vid	Pose2Vid	Reasoning	Remove	Sketch2Vid	Stylization	Overall
0%	7.80	6.27	6.38	**8.33**	5.54	7.63	8.17	6.43	4.30	7.40	7.87	6.89
25%	7.23	6.43	6.82	8.23	**7.42**	6.00	8.63	6.73	5.00	6.63	8.20	7.00
75%	7.73	7.37	6.88	7.80	5.29	8.07	**8.63**	7.53	5.27	**7.53**	8.60	7.30
100%	**7.92**	**7.60**	**7.46**	8.27	7.17	**8.13**	8.57	**8.37**	**7.00**	7.17	**8.77**	**7.79**

Table 7. **EditVerseBench [26]** results for our model trained on a data mixture where the number of samples retained in EgoEditData is varied. As the amount of retained samples in EgoEditData increases, non-egocentric editing performance as measured by EditVerseBench [26] increases, showing usefulness of EgoEditData beyond the egocentric case.

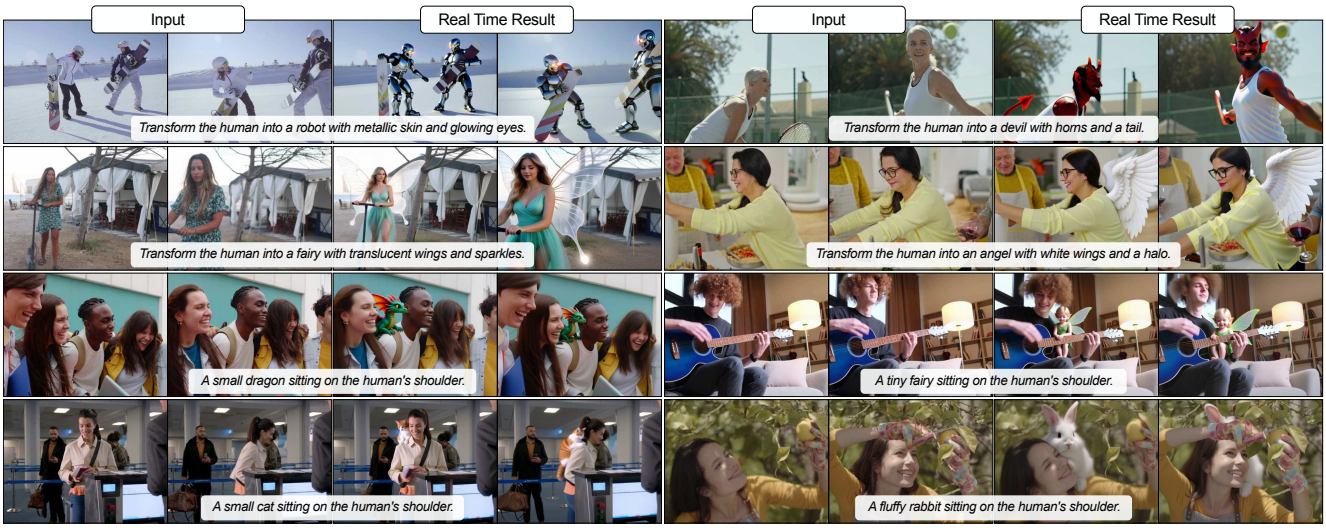


Figure 9. Exocentric video edits produced in real time by EgoEdit's streaming variant EgoEdit-RT on a single H100 GPU.

Figure 10. In-the-wild video edits produced in real time by EgoEdit's streaming variant EgoEdit-RT on a single H100 GPU.

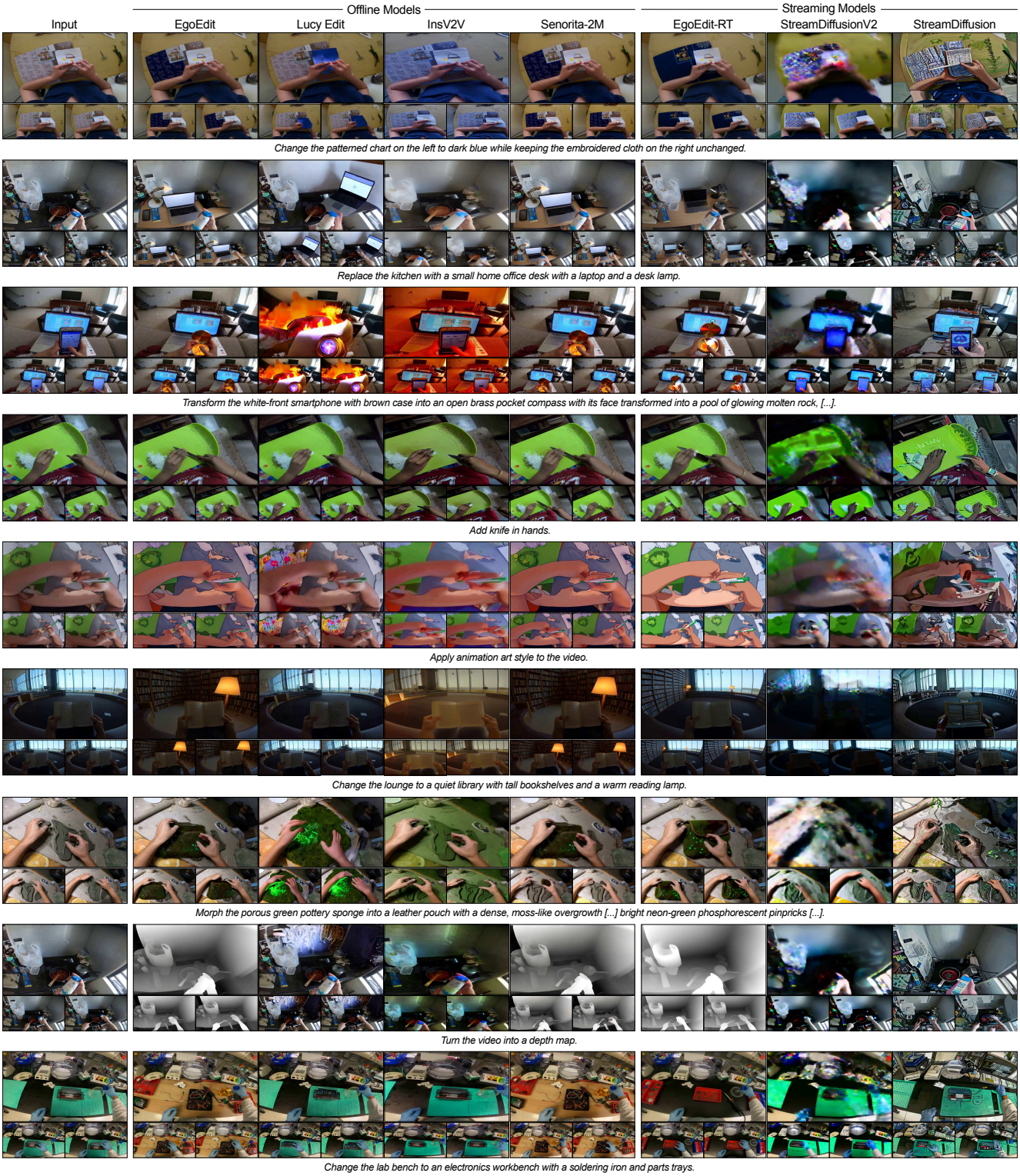


Figure 11. Qualitative comparison of EgoEdit and its real-time streaming variant EgoEdit-RT against baselines on EgoEditBench. EgoEdit and EgoEdit-RT consistently perform better than their baselines. Note that Señorita-2M uses the first frame from EgoEdit for frame propagation.

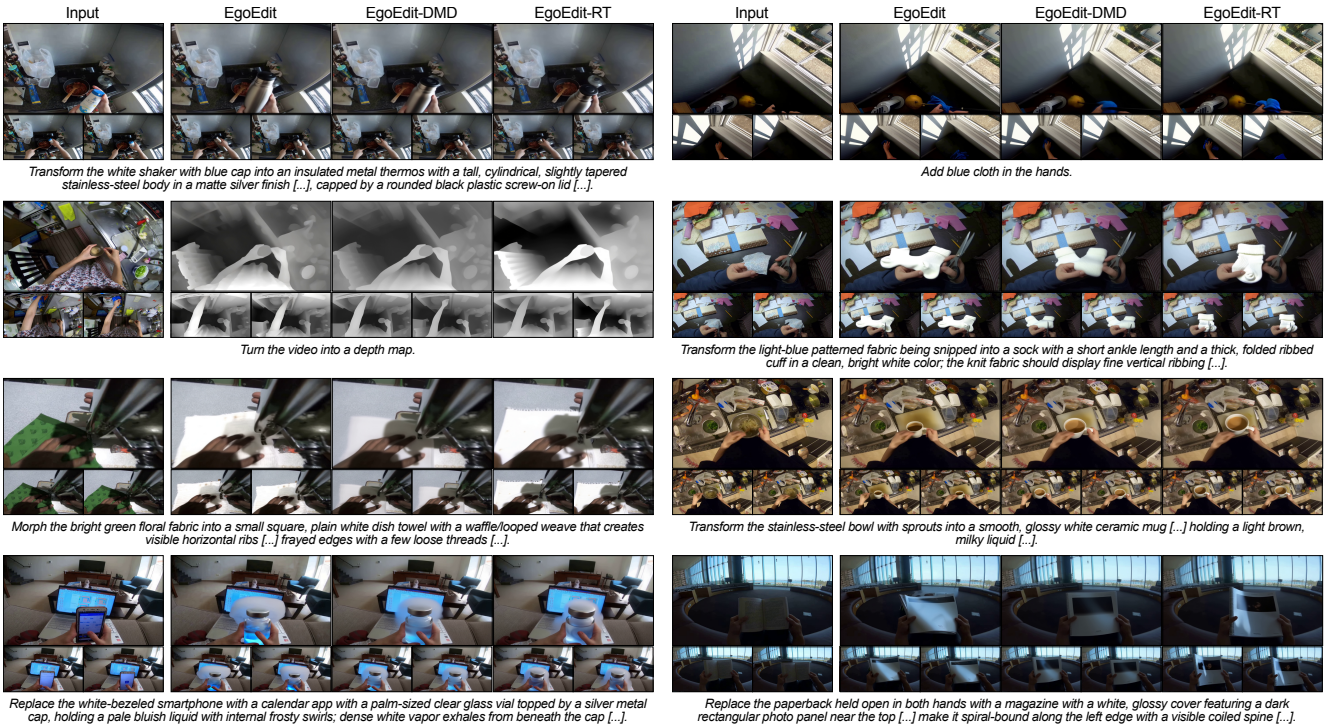


Figure 12. Qualitative comparison of EgoEdit at different stages of distillation. EgoEdit indicates the original 80 NFEs model, EgoEdit-DMD represents the model after the 4-step DMD distillation, EgoEdit-RT represents the final real-time streaming model obtained after Self Forcing distillation.

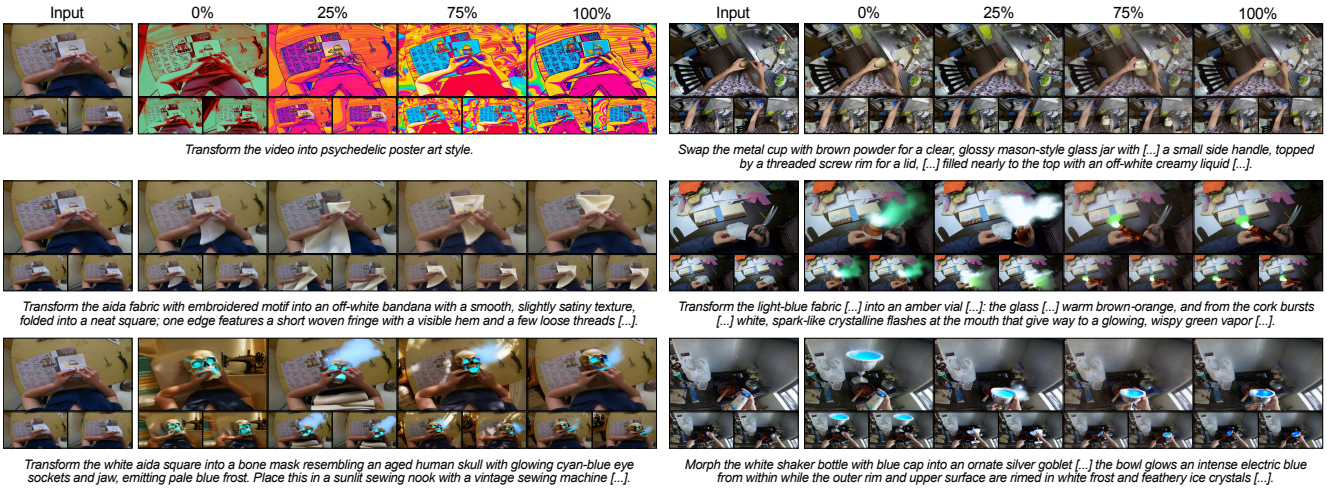


Figure 13. Qualitative comparison of different variants of EgoEdit trained on a data mixture with reduced portions of EgoEditData. Percentages indicate the proportion of unique source video samples in EgoEditData retained for training.