LEARNED IMAGE COMPRESSION FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models increasingly rely on high-frame-rate multi-camera observations, making visual communication a major bottleneck for real-time robotic control in bandwidth-constrained or distributed deployment settings. Existing image and video codecs, however, are designed to preserve generic visual fidelity rather than the control performance of downstream VLA policies. In this work, we introduce SPARC (SPatially Adaptive Rate Control), a learned image compression framework tailored for VLA-driven robots. Our key observation is that the importance of visual information varies substantially across both camera views and spatial regions within an image frame. Based on this observation, SPARC employs a lightweight temporal mask selector that uses the current and preceding frames to decide which latent entries to prioritize, thereby allocating bitrate across views and regions according to task relevance. We further introduce a tilted rate loss that stabilizes training by reducing the tendency of entropy-based rate objectives to over-suppress rare yet task-critical visual patterns. Experiments on RoboCasa365, VLABench, and LIBERO with three VLA models show that SPARC consistently achieves better bitrate-success tradeoff than conventional image/video codecs and recent learned compression methods. We additionally demonstrate real-world deployment benefits in remote-control settings, where SPARC substantially improves the bitrate-success tradeoff.

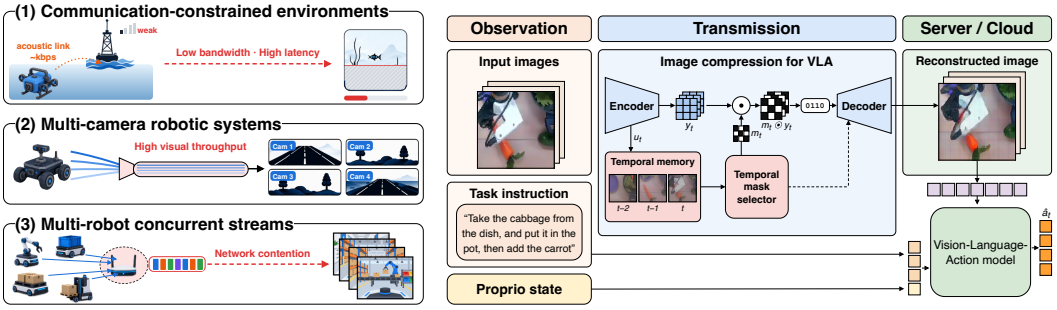
1 INTRODUCTION

How much visual information does a robot truly need to act effectively? This question is becoming increasingly important as vision-language-action (VLA) models emerge as a promising paradigm for general-purpose robotic control. Modern VLAs consume high-frame-rate visual streams, often from multiple cameras, to generate actions in real time (Ghosh et al., 2024; Kim et al., 2025; 2024; Black et al., 2025; Physical Intelligence et al., 2025; Bjorck et al., 2025).

In many practical scenarios, VLA inference relies on cameras that stream visual observations to a remotely located computing unit. As illustrated in Figure 1a, transmitting these visual streams incurs substantial transmission costs that hinder scalable real-time control, particularly in three scenarios: communication-constrained environments such as underwater or space robotics, robotic systems utilizing massive number of camera inputs, and multi-robot deployments with network contention such as in factories. In these settings, reducing the transmission cost of visual signals is essential for scalable real-time control. Yet, robotic visual observations are still compressed using generic image or video codecs such as JPEG (Wallace, 1992) or H.265 (Sullivan et al., 2012). These codecs naïvely allocate more bitrate to regions with more details to preserve perceptual image fidelity, while spatial regions with high complexity may not necessarily be relevant for the robotic task. Consequently, the bitrate allocation strategies of existing codecs may be inefficient for downstream control.

We believe that visual compression for VLAs should consider two major opportunities: localized task importance across cameras and spatial regions, and temporal sparsity due to action chunking. These points are particularly important because visual information a VLA relies on is highly non-uniform in both space and time. Spatially, modern VLAs often operate on multi-camera inputs, where different viewpoints contribute unequally to control (Mandlekar et al., 2021). Moreover, within each camera input, only a subset of spatial regions may be relevant to the current task (Xu et al., 2025). Temporally, action chunking means new observations are available every few control steps during inference. The

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Motivating scenarios (b) Framework of learned image compression for VLA

Figure 1: **Motivation and overview of SPARC.** (a) Visual transmission becomes a bottleneck in communication-constrained environments, multi-camera robotic systems, and multi-robot deployments sharing a network. (b) Trained with the action loss rather than pixel-level distortion, SPARC predicts a task-aware latent mask from temporal memory and transmits only the unmasked entries.

resulting gaps between transmitted frames make inter-frame change large and highly non-uniform, breaking the frame-to-frame continuity that conventional video codecs rely on.

We propose **SPARC** (SPatially Adaptive Rate Control), a learned image compression framework tailored for VLA systems. Reflecting highly localized task importance, SPARC performs policy-aware masking directly in the latent space of a neural image codec to allocate high bitrates to task-relevant regions. Moreover, it conditions on previously transmitted observations to compensate for the sparse temporal sampling induced by action chunking. In practice, it employs a lightweight temporal mask selector that predicts element-wise latent masks ($h \times w \times c$) from the temporal context, allowing adaptive bitrate allocation between camera streams and image regions. To train this adaptive masking mechanism stably under a VLA-guided rate objective, we introduce a tilted rate loss that prevents entropy-based optimization from aggressively suppressing statistically rare yet task-critical visual patterns. Built on top of a pretrained neural image codec, SPARC is trained end-to-end to directly optimize downstream action quality under a communication budget.

We validate SPARC on both simulation environments and real-world remote control deployments. On simulation benchmarks, such as RoboCasa365, VLABench, and LIBERO, SPARC consistently achieves stronger bitrate-success tradeoffs than conventional image/video codecs and recent learned compression methods. In real-world deployments, we employ split-computing architecture between local CPU and remote GPU server to verify that SPARC consistently achieves better success rates than MS-ILLM while using substantially lower bitrate. We show that reduced communication overhead translates into lower end-to-end latency in bandwidth-constrained deployments.

Our contributions are summarized as follows:

- We propose SPARC, the first learned image compression framework tailored for VLAs.
- We introduce a temporally conditioned latent masking framework for compression, combining a temporal mask selector for adaptive bitrate allocation and a tilted rate loss for stable optimization.
- We demonstrate superior bitrate-success tradeoffs and lower end-to-end latency across simulation and real-world robotic deployments.

2 Related Work

Neural compression of visual signals. Compared with traditional codecs such as JPEG (Wallace, 1992) and H.265 (Sullivan et al., 2012), learned compression methods utilize neural networks optimized under rate-distortion objectives (Ballé et al., 2017; Yang et al., 2023; Ballé et al., 2018; Minnen et al., 2018; Mentzer et al., 2020; Yang & Mandt, 2023). For example, prior work has improved compression quality through learned hyperpriors (Ballé et al., 2018; Minnen et al., 2018) and generative reconstruction techniques (Mentzer et al., 2020; Yang & Mandt, 2023). However, many of these methods are trained for a fixed compression rate, making it difficult to support multiple rates with a single model. To this end, variable-rate compression methods train a single model that

supports multiple bitrates using iterative refinement, conditioning, or latent selection (Toderici et al., 2016; 2017; Choi et al., 2019; Lee et al., 2022). Beyond global rate control, spatially adaptive codecs allocate different bit budgets to different image regions using external priors such as ROI masks or quality maps (Ma et al., 2021; Song et al., 2021). Yet, this reliance on external priors limits the applicability of existing methods to real-time VLA systems, where task-relevant regions should ideally be inferred automatically. SPARC therefore learns VLA-guided bit allocation across image regions without relying on external spatial priors.

Neural image compression for machines. A growing body of work adapts image compression for machine vision by using downstream-task losses instead of reconstruction distortion, enabling codecs to preserve task-relevant information rather than pixel fidelity (Duan et al., 2020). Representative approaches include content-adaptive codecs, ROI-based bit allocation with segmentation priors, and unified human-machine compression (Le et al., 2021; Shindo et al., 2024; Feng et al., 2025). Nevertheless, prior codecs for machines mainly target perception models that map images to labels, regions, or text, rather than policies that predict continuous robot actions. In contrast, to the best of our knowledge, SPARC is the first neural image codec designed specifically for VLA systems.

3 Problem Formulation

We formalize image compression for VLA models as follows. Suppose we have K cameras that capture a set of images $\mathbf{x}_t = \{\mathbf{x}_{t,i}\}_{i=1}^K$ at each timestep $t = 1, \dots, T$. Let $\mathbf{l}$ denote the language instruction for the task. At time t, the VLA policy generates an action $\mathbf{a}_t$ according to $\pi(\cdot|\mathbf{x}_{1:t}, \mathbf{l}, \mathbf{a}_{1:t-1})$. The policy is typically trained to minimize a loss measuring the quality of the generated action sequence $\mathbf{a}_{1:T}$. For example, given expert demonstrations $\mathbf{a}^*_{1:T}$, an off-policy objective may be

$$\min_{\pi} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\ell(\mathbf{a}_t, \mathbf{a}_t^*)] \quad \text{where} \quad \mathbf{a}_t \sim \pi(\cdot|\mathbf{x}_{1:t}, \mathbf{l}, \mathbf{a}^*_{1:t-1}), \tag{1}$$

where $\ell(\cdot, \cdot)$ is a loss function on actions, such as the flow matching loss.

Now consider *compressing* the image stream $\mathbf{x}_t$ before it is consumed by the VLA. Specifically, let (f_θ, g_ϕ) denote an encoder-decoder pair parameterized by (θ, ϕ). The encoder maps each image into a compact discrete latent representation $\mathbf{y}_{t,i} = f_\theta(\mathbf{x}_{t,i})$, and the decoder reconstructs the image as $\hat{\mathbf{x}}_{t,i} = g_\phi(\mathbf{y}_{t,i})$. In practice, the encoder may run on an edge device co-located with the cameras, while the decoder may run on a GPU server hosting the VLA model.

Our objective is to minimize the VLA loss under the compressed image stream $\{\hat{\mathbf{x}}_t\}_{t=1}^T$ while enforcing a compact latent representation $\mathbf{y}$. We formulate this objective as a Lagrangian relaxation of the rate-distortion objective under the off-policy setting:

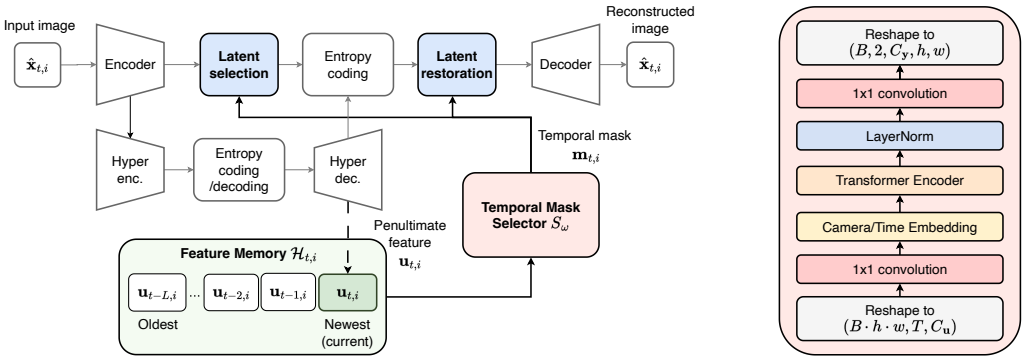
$$\min_{\theta,\phi,\psi} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\Big[\underbrace{\ell(\hat{\mathbf{a}}_t, \mathbf{a}_t^*)}_{\text{action distortion}} + \lambda \underbrace{\Big(\sum_{i=1}^{K} -\log p_\psi(\mathbf{y}_{t,i}) \Big)}_{\text{total rate}} \Big], \qquad \hat{\mathbf{a}}_t \sim \pi(\cdot|\hat{\mathbf{x}}_{1:t}, \mathbf{l}, \hat{\mathbf{a}}_{1:t-1}) \tag{2}$$

where p_ψ is a learnable entropy model over the latent, and $\lambda > 0$ controls the tradeoff between compression rate and action quality. Note that logarithms are base 2 throughout the paper.

Properties. A key distinction from prior work is twofold. First, the compression quality is measured by the action distortion $\ell(\hat{\mathbf{a}}_t, \mathbf{a}_t^*)$, rather than image-level distortion $d(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ or distortions computed from non-sequential downstream predictors $d(h(\mathbf{x}_t), h(\hat{\mathbf{x}}_t))$. Second, we minimize the aggregated total rate over the cameras, instead of applying the same rate individually.

4 Method

We propose a learned image codec framework for VLAs: **SPARC** (SPatially Adaptive Rate Control). Our design is motivated by the characteristics of images commonly encountered in VLA systems, as illustrated in Figure 2. In particular, we observe that the importance of visual information varies significantly both across cameras and across spatial regions within an image. To leverage this property, SPARC adaptively assigns different compression rates to different regions based on their



(a) Overall architecture of SPARC (b) Temporal mask selector

Figure 2: **Architecture of SPARC.** (a) SPARC builds on an autoencoder architecture with hyperpriors, incorporating a temporal mask selector which takes a history of hyperprior features ($\mathcal{H}_{t,i}$) and generates a spatial binary mask ($\mathbf{m}_{t,i}$) to selectively mask the latent. (b) Temporal mask selector is composed of a lightweight transformer, incorporating the camera and time embedding.

task relevance. Concretely, SPARC employs a lightweight temporal mask selector operating on latent representations (Section 4.1), trained with a specialized objective (Section 4.2).

4.1 Architecture

Motivated by their success in neural image compression, SPARC adopts an autoencoder architecture with hyperpriors (Figure 2). Specifically, we build upon the architecture of Ballé et al. (2018), and introduce a temporal mask selector that modulates the latent representation on a per-region basis.

Temporal mask selector. Given a latent representation $\mathbf{y}_{t,i} \in \mathbb{R}^{h \times w \times c}$, the temporal mask selector S_ω produces a binary mask $\mathbf{m}_{t,i} \in \{0,1\}^{h \times w \times c}$ of the same shape, indicating which entries of $\mathbf{y}_{t,i}$ should be retained. We then apply elementwise masking, $\mathbf{m}_{t,i} \odot \mathbf{y}_{t,i}$, so that only the unmasked entries are stored and transmitted, reducing the required bitrate.

To generate the mask, S_ω leverages information from past observations. Specifically, we maintain a feature memory $\mathcal{H}_{t,i} = \{\mathbf{u}_{t-L,i}, \ldots, \mathbf{u}_{t-1,i}, \mathbf{u}_{t,i}\}$, where L denotes the history length, and $\mathbf{u}_{t,i}$ denotes penultimate-layer feature of hyperprior decoder. Such features are known to contain informative signals about image content (Lee et al., 2022). The features in $\mathcal{H}_{t,i}$ are concatenated, projected, and augmented with learned temporal and camera embeddings. The resulting representation is processed by a lightweight transformer. Finally, the output is binarized into $\mathbf{m}_{t,i}$ using Gumbel-Softmax (Jang et al., 2017), allowing gradients to propagate through the discretization step during training.

Transmission. During transmission, SPARC sends two components: the entropy-coded image latent containing only the unmasked entries, and the hyper-latent. The binary mask $\mathbf{m}_{t,i}$ itself is not transmitted, avoiding additional communication overhead. Instead, the mask is deterministically rematerialized at the receiver side (i.e., GPU side) from the hyper-latent, using the same temporal mask selector and feature memory $\mathcal{H}_{t,i}$, producing the identical mask $\mathbf{m}_{t,i}$ at both the sender and receiver. The receiver then restores the sparse latent by inserting zeros at masked locations, yielding $\mathbf{m}_{t,i} \odot \mathbf{y}_{t,i}$. This procedure is applied independently to each camera stream.

4.2 Training

To leverage a codec pretrained on large-scale and diverse visual data, we initialize SPARC from a pretrained neural image codec, namely MS-ILLM (Muckley et al., 2023). Specifically, we augment the pretrained model with the temporal mask selector S_ω and the feature memory $\mathcal{H}$.

Joint optimization with tilted rate loss. We jointly optimize the codec decoder and the temporal mask selector using both the action distortion and the rate loss. To stabilize training, we introduce a modified rate objective, termed the ***tilted rate loss***, which prevents the masking policy from being overly biased toward regions with high local bitrate.

Let $\mathbf{y} \in \mathbb{R}^N$ denote a (flattened) latent representation with $N = h \times w \times c$ entries. We define the local bit vector $\mathbf{b} \in \mathbb{R}^N$ as the number of bits required to encode each latent entry:

$$\mathbf{b} = \big(-\log p_\psi(y_1), \ldots, -\log p_\psi(y_N) \big), \tag{3}$$

where p_ψ is the entropy model and y_j is the j-th entry of $\mathbf{y}$. Using this notation, the original rate loss is $\ell_{\text{rate}}(\mathbf{y}) := \frac{1}{N} \sum_{j=1}^{N} b_j$. Applying a binary mask $\mathbf{m} \in \{0,1\}^N$ yields the masked rate loss $\ell_{\text{rate}}(\mathbf{y}; \mathbf{m}) := \frac{1}{N} \sum_{j=1}^{N} m_j b_j$.

However, this objective implicitly encourages the temporal mask selector to suppress entries with the largest b_j, i.e., entries that are least likely under the entropy model. Such entries often correspond to rare or unusual visual patterns, which can nevertheless be highly important for robotic decision-making. Excessively masking these regions may therefore lead to unstable or degraded training.

To mitigate this issue, we introduce the tilted rate loss. Specifically, we define an exponentiated version of local bit vector:

$$\mathbf{b}^{(\alpha)} = \big(b_1^\alpha, \ldots, b_N^\alpha \big), \qquad \alpha \in [0,1], \tag{4}$$

which compresses the dynamic range of the bit penalties. We then replace the original penalty $\mathbf{b}$ with the scaled penalty $\mathbf{b}^{(\alpha)}$ when computing the masked rate:

$$\ell_{\text{rate}}^{(\alpha)}(\mathbf{y}; \mathbf{m}) = \left(\frac{1}{N} \sum_{j=1}^{N} m_j b_j^\alpha \right) \cdot \frac{\sum_{j=1}^{N} b_j}{\sum_{j=1}^{N} b_j^\alpha}. \tag{5}$$

The second term acts as a normalization factor, ensuring that the overall scale of the rate loss remains comparable across different values of α. The tilted rate loss is computed independently for each camera stream, summed, and incorporated into the overall optimization objective in Equation 2.

5 Experiments

To validate our ideas, we conduct several experiments on diverse simulation benchmarks, compression models and VLA models, as described in Section 5.1. We compare our method with several traditional and learned image compression methods and video compression methods (Section 5.2) and present several visual examples (Section 5.3). Additional ablation results are provided in Section C.

5.1 Experimental Setup

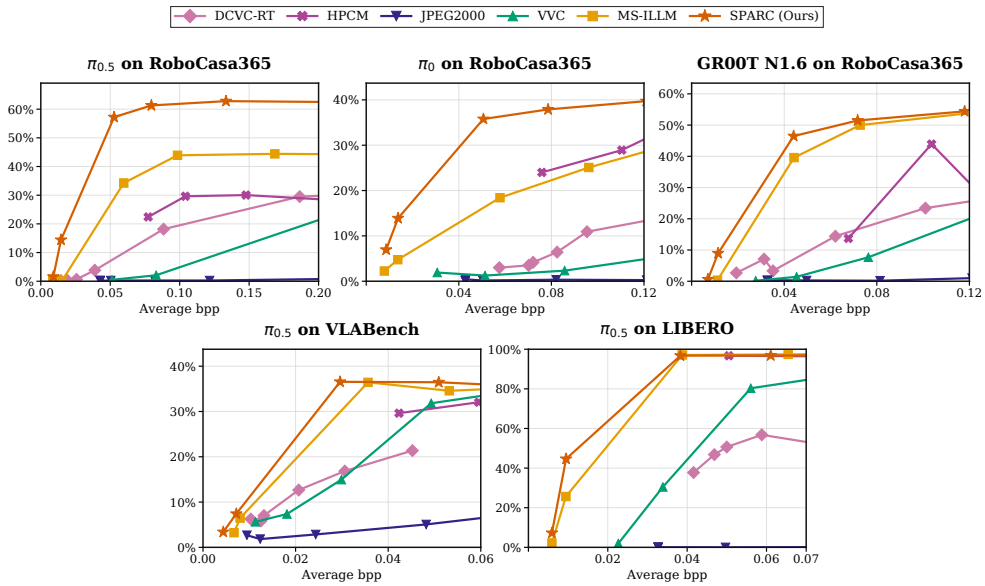
Benchmarks. We evaluate SPARC on three standard simulation benchmarks: (1) RoboCasa365 (Nasiriany et al., 2026): We use a VLA model fine-tuned on the Human300 pre-training tasks and report the average success rate across all 24 atomic tasks. (2) VLABench (Zhang et al., 2025): We evaluate the official VLA checkpoint fine-tuned on the 10 primitive tasks under the in-distribution setting. (3) LIBERO (Liu et al., 2023): We employ a checkpoint fine-tuned on all four task suites (Spatial, Object, Goal, and Long) and report the overall average success rate across them.

Baselines. We compare SPARC against multiple compression methods, categorized into two groups: (1) Image codecs: JPEG2000 (Christopoulos et al., 2000), a traditional handcrafted codec, VVC (Bross et al., 2021), a traditional standard codec, as well as MS-ILLM (Muckley et al., 2023), and HPCM (Li et al., 2025), which are recent learned neural codecs, and (2) Video codecs: DCVC-RT (Jia et al., 2025), a powerful learned real-time codec. For downstream VLA models, we use π_0 (Black et al., 2025), $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T N1.6 (Bjorck et al., 2025). See Section B for more details.

Training details. Across all datasets, we use a batch size of 12 and train for 15,000 steps. We report bpp based on the size of latent and hyper-latent representations. Also, VLA model-specific pre-processing is applied after visual signals are compressed. For each benchmark, SPARC is trained on the same dataset used to train the corresponding VLA model and the action distortion loss is equivalent to the action loss (i.e. flow-matching loss) used for fine-tuning VLA models. In terms of hyper-parameters, we tune α over $0.0, 0.2, 0.4, 0.6, 0.8, 1.0$ and set the temporal history to 1. For the rate-distortion trade-off, we use $\lambda \in \{1.0, 10.0, 20.0, 50.0\}$, with larger λ values fine-tuned from

	RoboCasa365			VLABench	LIBERO
VLA model	$\pi_{0.5}$	π_0	GR00T N1.6	$\pi_{0.5}$	$\pi_{0.5}$
SPARC vs. MS-ILLM	-43.2%	-70.4%	-24.0%	-17.2%	-18.7%

(a) BD-rate of SPARC relative to MS-ILLM. Negative values indicate bitrate savings.



(b) Bitrate–success trade-off across benchmarks and VLA models.

Figure 3: **Bitrate–success rates on diverse simulation benchmarks and models.** We report average simulation success rates over a range of average bpps (bits per pixel). To maintain the fairness of comparisons, we calculate average bpp by aggregating bpp statistics of all frames in every episode. For SPARC, we average over 3 seeds to guarantee the reliability of SPARC's performance.

pre-trained checkpoints targeting higher-quality operating points to reach lower-bpp regimes. See Section B.1 for more details.

Inference details. For MS-ILLM and HPCM, we utilize the official publicly available checkpoints. For both DCVC-RT and VVC, we select the best GOP from $\{1, 64\}$. VVC uses VVenC 1.13.0/VVdeC 3.0.0. See Section B.1 for more details.

5.2 Simulation Results

In Figure 3, SPARC consistently achieves higher simulation success rates under comparable bpp budgets across diverse VLA models and benchmarks. Compared with conventional codecs and learned codecs, SPARC reaches strong performance at lower bpps, indicating that preserving task-relevant visual information is more important than generic image fidelity. For RoboCasa365 benchmark, SPARC shows strong performance on π_0 and $\pi_{0.5}$. For VLABench and LIBERO, SPARC remains competitive and consistently lies on the upper envelope of the bitrate-success curve.

5.3 Qualitative Examples

In Figure 4, we compare visual samples of images from MS-ILLM and SPARC. While MS-ILLM and SPARC produce visually similar images at equivalent compression rates, their bit allocation strategies differ significantly. SPARC prioritizes task-critical regions, such as the gripper and target objects (e.g. an orange can), to facilitate robotic control. Conversely, MS-ILLM distributes bits uniformly across the entire image to optimize global quality, wasting bandwidth on task-irrelevant backgrounds like table textures. More examples can be found in Section K.

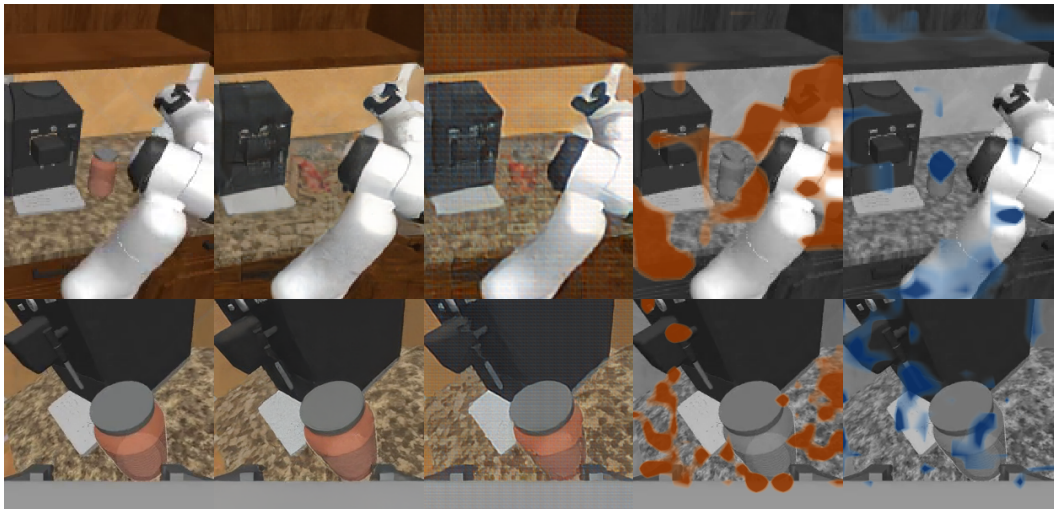


Figure 4: **Qualitative results on RoboCasa365.** In relative bit allocation map, Blue indicates spatial regions where SPARC allocates more bits under an identical total bit budget than MS-ILLM. Orange indicates spatial regions where MS-ILLM allocates more bits than SPARC.

Table 1: **SPARC vs. other masking strategies.** We compare SPARC against four alternative masking strategies applied at inference. Keep indicates the keep ratio of the mask. *Current-only* suppresses the past-frame token in the selector. *Saliency* keeps the top-K latents by the hyperprior's scale σ. *Spatially Uniform* applies a fixed deterministic pattern, and *Random* selects an arbitrary subset.

Masking	Keep	bpp	Succ. (%)	Keep	bpp	Succ. (%)	Keep	bpp	Succ. (%)
Current-only	0.235	0.015	14.2	0.667	0.053	54.4	0.644	0.080	59.2
Saliency	0.236	0.014	0.3	0.667	0.054	10.9	0.635	0.090	8.8
Spatially Uniform	0.236	0.008	0.2	0.667	0.041	0.1	0.635	0.065	1.2
Random	0.236	0.008	0.0	0.667	0.042	0.6	0.635	0.064	0.8
SPARC (Ours)	0.236	0.015	**14.4**	0.667	0.053	**57.2**	0.635	0.080	**61.3**

5.4 Ablation Studies

To verify the contribution of our learned masking strategies to performance, the following ablation studies were conducted on RoboCasa365 with $\pi_{0.5}$, a benchmark with the most dexterous tasks and abundant visual signals. All reported results are averaged over 3 seeds. Other ablation studies can be found in Section C.

Masking strategy. In Table 1, we compare SPARC against alternative masking strategies to verify that SPARC identifies and preserves task-critical regions rather than masking arbitrarily. To isolate the effect of the masking strategy alone, we fix the trained SPARC model and apply each alternative mask only at inference. As shown, SPARC achieves the highest success rate, outperforming rule-based strategies, as well as saliency-based approaches. These results confirm that SPARC's hyper-latent-driven masking is not incidental but is essential to its performance.

Temporal history length. In Table 2, we analyze the effect of the temporal history length L. Leveraging temporal history consistently improves the success rate over the baseline without history (L=0) at nearly identical bitrates, indicating that the gain stems from temporal context rather than additional bits. L=1 yields the best overall performance, improving the success rate over L=0 by 1.3, 2.5, and 3.8 percentage points, respectively, with the gain becoming more pronounced at higher bitrates. Notably, a longer history does not necessarily yield higher performance. We attribute this to action chunking, which executes multiple actions per inference step and thus induces large

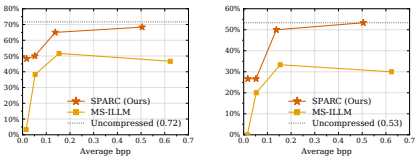
Table 2: **Temporal history ablation on RoboCasa $\pi_{0.5}$.** We analyze how changing temporal history affects the performance.

$L = 0$ (unused)		$L = 1$		$L = 2$		$L = 4$	
bpp	Succ. (%)	bpp	Succ. (%)	bpp	Succ. (%)	bpp	Succ. (%)
0.0146	13.1	0.0146	14.4	0.0146	13.9	0.0146	**14.8**
0.0528	54.7	0.0525	**57.2**	0.0521	55.6	0.0529	54.9
0.0788	57.5	0.0796	**61.3**	0.0792	59.0	0.0789	58.5



(a) side-view (b) top-view (c) wrist-view

Figure 5: **Camera & robot setup.** Three cameras with different viewpoints (256×256 resolution) and the single AgileX PiPER arm.



(a) sub-tasks (b) full-tasks

Figure 6: **Real-world results.** Average success scores for sub-tasks and full-tasks with 60 training episodes and 10 evaluation trials per task.

scene changes between consecutive observations. As a result, frames further in the past can differ substantially from the current frame, providing less useful or even misleading context. Moreover, since a single past frame suffices, our method incurs minimal memory and computational overhead.

6 Real-world Experiment

In Section 6, we conduct real-world experiments in a server-client split environment: Intel i7-10875H CPU on local laptop and the enterprise-level GPU workstation (RTX6000Ada) on a remote server. In Figure 5, we set up the environment with multiple cameras from different perspectives and a single 6-DoF AgileX PiPER arm equipped with commercially available two-finger parallel-jaw gripper (80 mm stroke). The dataset is collected using a leader-follower teleoperation setup. We design three challenging long-horizon tasks for training and evaluation. We also apply asynchronous inference and fix the input resolution to 256×256. Section 6.1 demonstrates our evaluation results, and Section 6.2 shows the latency gain of SPARC in the real-world and component-wise breakdown. See Section B.2 and Section B.3 for more details, Section I for example rollouts and Section J for failure case analysis.

6.1 Evaluation

In Figure 6a and Figure 6b, we report the success rate under our setting, showing that the practical gain using our compression method is notable compared to common high-quality JPEG q95 compression ($\sim$1.9 bpp). We measure the performance of SPARC on three long-horizon tasks, all of which are assessed using full-task success score and sub-task success scores. Each long-horizon task consists of two sub-tasks. We measure both average sub-task success scores and average full-task success scores. Our result shows that SPARC can maintain a greater success rate than MS-ILLM across a wide range of bpps for both sub-tasks and full tasks.

6.2 Latency

Component-wise latency breakdown. In Table 3, we report the component-wise latency comparison between MS-ILLM and SPARC. We omit the decoder-side mask re-materialization time since it can be performed simultaneously with encoder-side mask materialization and transmission, introducing no additional latency. While the temporal mask selector incurs some additional computational overhead, this is offset by the reduction in entropy coding, resulting in similar overall latency.

End-to-end latency analysis. In Table 4, we report the end-to-end latency of our split-computing architecture, where the network latency is computed from the transmitted bitstream size and the

Table 3: **Per-component latency.** We compare the measured latency (ms) of SPARC with MS-ILLM on 256×256 input resolution. Each measurement is a median over 50 trials.

Method	Keep	bpp	Main Net	Hyper Net	Hyper-Entropy	Temporal	Entropy	Total
(a) Encoding								
SPARC	0.667	0.0454	68.58	16.68	3.42	5.35	11.28	106.24
MS-ILLM	—	0.0497	69.09	16.68	3.34	—	12.60	102.16
(b) Decoding								
SPARC	0.667	0.0454	5.55	0.43	1.51	—	10.94	18.43
MS-ILLM	—	0.0497	5.54	0.42	1.49	—	14.01	21.46

Table 4: **Per-camera end-to-end latency under varying bandwidth and number of streams.** We show how bandwidth-limited settings highlight the practical effectiveness of SPARC by reducing network latency. † denotes latency exceeding the required transmission deadline.

	JPEG (q95)			SPARC		
	Enc/Dec	Network	Total	Enc/Dec	Network	Total
(a) Varying bandwidth C (N=1)						
250 kbps	0.5	—	—	124.7	33.1	**157.8**
500 kbps	0.5	446.1	446.6†	124.7	15.8	**140.4**
750 kbps	0.5	205.6	206.1	124.7	10.3	**135.0**
1 Mbps	0.5	133.6	134.1	124.7	7.7	**132.4**
2 Mbps	0.5	55.6	**56.1**	124.7	3.8	128.5
(b) Varying streams N (C=100 Mbps)						
N=1	0.5	0.96	**1.5**	124.7	0.08	124.7
N=128	0.5	1.51	**2.0**	124.7	0.08	124.7
N=256	0.5	3.57	**4.1**	124.7	0.08	124.7
N=347	0.5	145.0	145.5	124.7	0.08	**124.8**
N=1024	0.5	—	—	124.7	0.10	**124.8**
N=4096	0.5	—	—	124.7	1.01	**125.7**

available link bandwidth. Since the policy executes action chunks of 10 steps at 30 Hz, each observation must be delivered within 333 ms. As a baseline, we adopt JPEG with a quality factor of 95, the default setting in ROS 2, which records 1.9 bpp on our real-world images. Since SPARC maintains competitive performance down to 0.15 bpp in our real-world experiments, we set its bitrate to 0.15 bpp. The results reveal a clear trade-off between communication and computation. Although SPARC incurs additional neural encoding overhead, its bitrate reduction substantially lowers the transmission latency under bandwidth-constrained conditions, yielding lower latency than JPEG below approximately 1 Mbps or beyond 347 concurrent streams on a 100 Mbps link. In particular, JPEG fails to meet the deadline at 500 kbps or below, whereas SPARC remains within the deadline even at 250 kbps. Moreover, the latency of SPARC stays nearly constant as the number of streams increases, while that of JPEG diverges beyond 347 streams due to link saturation. We note that deploying SPARC on more capable edge-computing platforms, such as NVIDIA Jetson devices with GPU acceleration, could further reduce the encoding latency and extend its advantage to higher-bandwidth settings.

7 Conclusion

In this work, we propose SPARC, a learned image compression framework that adaptively allocates bitrate across camera views and spatial regions using task-aware spatiotemporal latent masks. By prioritizing control-relevant visual information, SPARC achieves a stronger bitrate–success tradeoff than conventional image/video codecs and learned compression baselines across simulation and real-world experiments. Our analysis further shows that the additional latency introduced by the temporal mask selector on CPU is offset by the reduction in the size of transmitted latent representation resulting in lower entropy coding and decoding overhead. We further believe that additional reduction of latency is possible with more powerful on-device compute resources, making SPARC a practical step toward bandwidth-efficient VLA deployment.

AI USE STATEMENT

In this work, we used generative AI tools for helping develop theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results. We have not used generative AI tools for proposing or refining hypotheses and cleaning and reformatting dataset and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. For example, we checked LLM-generated research ideas for potential plagiarism through a manual literature survey and LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For the implementation of SPARC, we submit a link to an anonymous downloadable source code as supplementary materials.

REFERENCES

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

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. In *Robotics: Science and Systems*, 2025.

Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *International Conference on Computer Vision*, 2019.

Charilaos Christopoulos, Athanassios Skodras, and Touradj Ebrahimi. The JPEG2000 still image coding system: an overview. *IEEE Transactions on Consumer Electronics*, 2000.

Lingyu Duan, Jiaying Liu, Wenhan Yang, Tiejun Huang, and Wen Gao. Video coding for machines: A paradigm of collaborative compression and intelligent analytics. *IEEE Transactions on Image Processing*, 2020.

Ruoyu Feng, Yunpeng Qi, Jinming Liu, Yixin Gao, Xin Li, Xin Jin, and Zhibo Chen. Diff-ICMH: Harmonizing machine and human vision in image compression with generative prior. In *Advances in Neural Information Processing Systems*, 2025.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Zhaoyang Jia, Bin Li, Jiahao Li, Wenxuan Xie, Linfeng Qi, Houqiang Li, and Yan Lu. Towards practical real-time neural video compression. In *Computer Vision and Pattern Recognition*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.

Nam Le, Honglei Zhang, Francesco Cricri, Ramin Ghaznavi-Youvalari, Hamed Rezazadegan Tavakoli, and Esa Rahtu. Learned image coding for machines: A content-adaptive approach. In *IEEE International Conference on Multimedia and Expo*, 2021.

Jooyoung Lee, Seyoon Jeong, and Munchurl Kim. Selective compression learning of latent representations for variable-rate image compression. In *Advances in Neural Information Processing Systems*, 2022.

Yuqi Li, Haotian Zhang, Li Li, and Dong Liu. Learned image compression with hierarchical progressive context modeling. In *International Conference on Computer Vision*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

Yi Ma, Yongqi Zhai, Chunhui Yang, Jiayu Yang, Ruofan Wang, Jing Zhou, Kai Li, Ying Chen, and Ronggang Wang. Variable rate roi image compression optimized for visual quality. In *Computer Vision and Pattern Recognition*, 2021.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In *Advances in Neural Information Processing Systems*, 2020.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, 2018.

Matthew J Muckley, Alaaeldin El-Nouby, Karen Ullrich, Hervé Jégou, and Jakob Verbeek. Improving statistical fidelity for neural image compression with implicit local likelihood models. In *International Conference on Machine Learning*, 2023.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Takahiro Shindo, Kein Yamada, Taiju Watanabe, and Hiroshi Watanabe. Image coding for machines with edge information learning using segment anything. In *IEEE International Conference on Image Processing*, 2024.

Myungseo Song, Jinyoung Choi, and Bohyung Han. Variable-rate deep image compression through spatially-adaptive feature transform. In *International Conference on Computer Vision*, 2021.

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.

George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, 2025.

Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

Yibo Yang, Stephan Mandt, and Lucas Theis. An introduction to neural data compression. *Foundations and Trends in Computer Graphics and Vision*, 2023.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. VLAbench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *International Conference on Computer Vision*, 2025.

APPENDIX

A Limitations **15**

B Additional Experimental Details **15**

B.1 Implementation Details 15

B.2 Real-world Experiment Setup 16

B.3 Task Information 16

B.4 Baseline Reproduction 17

C Additional Ablation Results **18**

C.1 Tilted Rate Loss 18

C.2 Comparison with Alternative Training Recipes 19

C.3 Network robustness 19

C.4 Freezing components 20

D Transferability Analysis **20**

E LIBERO-Plus Benchmark **21**

F Task-criticality of the mask selector **22**

G Visual Quality Assessment **23**

H Per-task Analysis **24**

I Real-World Rollout Examples **25**

J Failure Case Analysis **38**

K Additional Visual Examples **39**

A LIMITATIONS

The absence of language grounding constrains interpretive flexibility; incorporating it would require larger models that may introduce non-trivial latency costs. Second, erroneous masking of task-critical details could pose a potential safety concern and may require precautionary measures such as hardware-level safety locks against unexpected robot motion. Finally, while remote and resource-constrained environments—such as deep-ocean settings—represent a compelling frontier for this framework, their evaluation remains out of reach due to spatiotemporal constraints.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 IMPLEMENTATION DETAILS

In Table 5, we provide additional training, evaluation, and model configurations of temporal mask selector model. Also, our implementation is anonymously available at `https://anonymous.4open.science/r/SPARC-B226`

Table 5: Additional implementation details

Category	**Configuration**
Training	
Optimizer	AdamW (β_1 = 0.9, β_2 = 0.999)
Weight decay	0.01
Gradient clipping	1.0
Batch size	12
Training steps	15000
Scheduler	cosine annealing with `min_lr = 1e-06`
Image resolution (LIBERO, RoboCasa, Real-World)	256×256
Image resolution (VLABench)	480×480
Codec training scope	A single codec for all camera views
Precision	fp32
Evaluation	
Action chunk size	Official default settings for each benchmark
Action chunk size (Real World)	10
SPARC	
Latent channels	320 (= MS-ILLM)
Num. of Temporal Mask Selector Parameters	1.4M
Num. of Encoder Parameters	7.42M
Num. of Hyperprior-Entropy Parameters	17.3M
Num. of Decoder Parameters	156M
Layers	1
Attention heads	8
Hidden dimension	1280 ($4\times$ the input dimension)

B.2 Real-world Experiment Setup

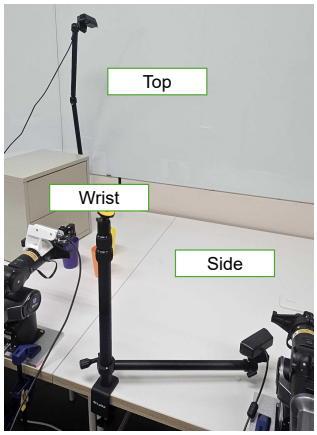


Figure 7: Real-world environment setups.

In Figure 7, we provide an overview of our real world experimental setup. We utilize three cameras (top, side, and wrist) to capture multi-view visual observations. Our platform consists of a single 6-DoF AgileX PiPER arm and three RGB cameras providing top, side, and wrist views (highlighted in red). Observations are encoded on-board and transmitted over the emulated link via SSH Tunnel to a remote policy server.

B.3 Task Information

In Table 6 and Table 7, we provide task composition information for simulation benchmarks and real-world evaluations. For LIBERO and VLABench, we evaluated on all available tasks. For RoboCasa365, we evaluated on 24 selected atomic tasks. For real-robot tasks, we give the details of tasks used for fine-tuning and evaluation.

Table 6: **RoboCasa atomic tasks.** The 24 atomic tasks used for every simulation result, grouped by skill family. Each task is evaluated with the official pretrained-split scene layouts.

Skill family	Tasks	#
Pick-and-place	`PickPlaceCounterToCabinet`, `PickPlaceCabinetToCounter`, `PickPlaceCounterToSink`, `PickPlaceSinkToCounter`, `PickPlaceCounterToMicrowave`, `PickPlaceMicrowaveToCounter`, `PickPlaceCounterToStove`, `PickPlaceStoveToCounter`	8
Open / close doors	`OpenCabinet`, `CloseCabinet`, `OpenMicrowave`, `CloseMicrowave`	4
Open / close drawers	`OpenDrawer`, `CloseDrawer`	2
Turning levers	`TurnOnSinkFaucet`, `TurnOffSinkFaucet`	2
Twisting knobs	`TurnOnStove`, `TurnOffStove`	2
Pressing buttons	`TurnOnMicrowave`, `TurnOffMicrowave`, `StartCoffeeMachine`	3
Insertion	`CoffeeSetupMug`, `CoffeeServeMug`	2
Navigation	`NavigateKitchen`	1
Total		24

Table 7: **Real-robot tasks.** Language instruction given to the policy, the sub-tasks it chains, and the demonstrations used for fine-tuning (single-arm PiPER, three cameras: top, side, wrist; 30 Hz).

Task	Instruction	Sub-tasks	Demos
Stack cups	*"stack the purple cup on the orange one, and put the yellow one on top"*	(1) pick the purple cup → place it on the orange cup (2) pick the yellow cup → place it on the purple cup	60
Cabbage & carrot to pot	*"take the cabbage from the dish and put it in the pot, then add the carrot"*	(1) pick the cabbage from the dish → place it in the pot (2) pick the carrot → place it in the pot	60
Carrot to drawer	*"pick up the carrot and put it in the upper drawer"*	(1) move to the upper drawer → open the upper drawer (2) pick the carrot → place it in the upper drawer	60
Total			180

B.4 Baseline Reproduction

In this section, we describe how we reproduced the existing codec baselines in our setting. All baselines are evaluated with exactly the pipeline used for SPARC: the policy is frozen and untouched, and the codec is inserted between the simulator and the policy, so every camera observation is encoded to a bitstream and decoded back before it enters the policy's own preprocessing.

JPEG 2000. We use OpenJPEG through Pillow with the irreversible 9/7 wavelet and a single quality layer at a fixed compression ratio $\mathrm{CR} \in \{50, 100, 200, 300, 500, 1000, 2000, 5000\}$, i.e. a target of $24/\mathrm{CR}$ bpp for 24-bit RGB; the rate is the size of the emitted codestream.

VVC. We use the Fraunhofer VVenC 1.13.0 encoder and VVdeC 3.0.0 decoder, both built from source in release configuration. RGB frames are converted to 8-bit YUV 4:2:0 with the `faster` preset and a fixed QP$\in \{22, 27, 32, 37, 42, 47, 51, 55\}$, decoded, and converted back to RGB with nearest-neighbor chroma upsampling.

DCVC-RT. We use the authors' release: the image model codes I-frames and the video model codes P-frames. Each camera maintains its own independent decoded picture buffer, which is cleared at every episode boundary, so the first policy call of every episode is coded as an I-frame and every subsequent call as a P-frame predicted from the previously coded frame of the same camera. Frames are replicate-padded to a multiple of 16 pixels, inference runs in half precision as in the authors' real-time setting. We sweep the model's continuous rate index $q \in \{0, 4, 8, 16, 24, 32, 40, 48, 56, 63\}$ (larger is higher rate).

HPCM. We use the authors' LIC-HPCM release with the HPCM-Large architecture and its MS-SSIM-optimised checkpoints, giving six rate points ($\lambda \in \{2.4, 4.58, 8.73, 16.64, 31.73, 60.5\}$).The entropy model uses the authors' scale table (60 levels between 0.12 and 64). HPCM is run in full precision.

MS-ILLM. We use the authors' release through its `torch.hub` entry points. The release names its operating points by target rate; we evaluate `vlo1` (0.0022 bpp), `vlo2` (0.0044 bpp), q=1 (0.035 bpp), q=2 (0.07 bpp), q=3. Frames are reflect-padded to a multiple of 64 pixels and the reconstruction is cropped back; the rate is the size of the actual rANS bitstream (latent and hyper-latent strings).

C Additional Ablation Results

C.1 Tilted Rate Loss

Table 8: **Performance across different tasks varying α.**

α	Succ. (%)	bpp	Succ. (%)	bpp	Succ. (%)	bpp
0.0	54.9	0.0546	58.9	0.0891	15.2	0.0146
0.2	53.7	0.0543	59.1	0.0863	**15.9**	0.0146
0.4	**57.2**	**0.0525**	**61.3**	**0.0788**	14.4	0.0146
0.6	55.6	0.0535	59.2	0.0847	0.2	0.0081
0.8	56.3	0.0530	59.8	0.0819	0.3	**0.0065**
1.0	55.8	0.0545	59.3	0.0828	0.2	**0.0065**

We conduct an ablation study of tilted rate loss, as described in Section 4.2. In tilted rate loss, the hyper-parameter α determines the degree to which the temporal mask selector aggressively masks regions with high local bitrates. A lower value of α encourages the temporal mask selector to mask based more on the action distortion loss than on local bitrates. In other words, it masks more task-irrelevant regions regardless of how many bits are allocated to them, which leads to an increase in the number of latent channels masked by the temporal mask selector. To verify the advantages of tilted rate loss, we report the average success rate and latency of SPARC with different values of α on the RoboCasa365 benchmark with $\pi_{0.5}$ as the downstream VLA model across a range of different bpp values. For latency measurements, note that all reported points correspond to models initialized from a pretrained model targeting a fixed quality level and trained with fixed hyperparameters to directly analyze the effect of α. We use A6000 GPU for both encoding and decoding for latency measurements.

Performance. Table 8 shows the effect of scaling α on the bit-success rate trade-off curve. Selecting a moderate value of α ($0.4 \leq \alpha \leq 0.8$) improves the performance of VLA models by preventing the temporal mask selector from aggressively masking high-frequency details to which a high proportion of bits is allocated, while preserving important details that require high local bitrates. This elucidates the advantage of tilted rate loss in maintaining high performance within a similar bpp range.

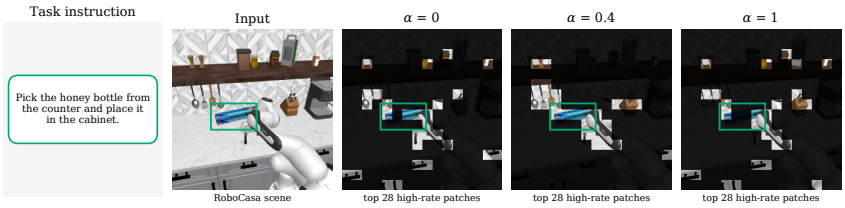


Figure 8: Example visualization of sweeping α on rate preservation

Visualization. Figure 8 illustrates how the rate-preservation coefficient α reshapes the spatial allocation of the token budget. For a representative RoboCasa manipulation episode with the instruction "Pick the honey bottle from the counter and place it in the cabinet", we visualize the top-28 patches ranked by estimated rate under three settings of α, where the green box denotes the task-relevant region containing the target object. At α = 0 and α = 1, a substantial fraction is absorbed by background clutter such as the shelf and surrounding utensils, leaving the target region only sparsely covered. As α increases to 0.4, the retained patches progressively concentrate around the target object and the manipulator, indicating that rate preservation reallocates capacity toward regions that are causally relevant to the instruction rather than merely photometrically complex. This qualitative trend is consistent with the quantitative gains reported in Table 8, and suggests that the improvements stem from better-targeted token allocation rather than from an overall increase in retained information.

C.2 Comparison with Alternative Training Recipes

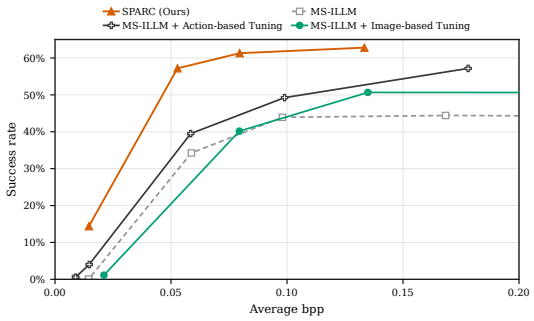


Figure 9: MS-ILLM variants on RoboCasa365 benchmarks.

We compare with other simple training methods to verify that the training method of SPARC is robot-oriented and effective. To do this, we compare SPARC with the following plausible training recipes.

- **Action-based Tuning**: Decoder-only fine-tuning with VLA optimization objectives
- **Image-based Tuning**: Original training objectives of MS-ILLM

As with SPARC, we apply these training recipes to pretrained MS-ILLM models with different quality levels. We report the average success rate of SPARC on 24 atomic tasks in the RoboCasa365 benchmark with $\pi_{0.5}$ as the downstream VLA model across a range of different bpp values.

Figure 9 compares average success rates across a wide range of bpp values among SPARC, MS-ILLM, and MS-ILLM variants trained with the two different recipes. It shows that our method consistently outperforms the baseline across a wide range of bpp values. In particular, SPARC achieves the highest performance at lower bpp values than the other training methods before entering the very-low-bpp regime, where success rates drop substantially. Moreover, it consistently outperforms other training methods at low bpp values. Other training methods, regardless of whether they are perceptually oriented or action-distortion-oriented, cannot match the performance gain achieved by SPARC, which benefits from the spatial allocation of local bitrates for downstream VLA models.

C.3 Network robustness

Table 9: **Network robustness.** On RoboCasa365 benchmark with $\pi_{0.5}$ backbone, we evaluate SPARC under a range of network-distorted conditions emulated with a FIFO link emulator, covering bandwidth throttling, fixed RTT, jitter, and frame drop. All results are averaged over 3 seeds.

		Clean	Bandwidth (kbps)				Fixed RTT (ms)		Jitter (ms)		Drop (%)	
Method	Metric		50	100	300	500	250	500	250±50	500±100	10	30
JPEG	Succ. (%)	38.67	1.67	2.00	2.17	6.08	30.17	18.67	25.25	13.25	40.58	37.58
	bpp	1.081	1.054	1.054	1.046	1.076	1.085	1.084	1.083	1.074	1.080	1.066
SPARC (Ours)	Succ. (%)	**57.2**	**49.9**	**50.7**	**49.8**	**49.0**	**49.0**	**39.9**	**44.4**	**32.8**	**55.4**	**52.3**
	bpp	0.053	0.053	0.053	0.053	0.053	0.053	0.052	0.052	0.052	0.053	0.053

In Table 9, we demonstrate the robustness of SPARC against JPEG, which is a common compression method on robot foundation models, In a variety of limited channel conditions, unlike JPEG, SPARC maintains its performance under matched or slightly higher success rate.

C.4 FREEZING COMPONENTS

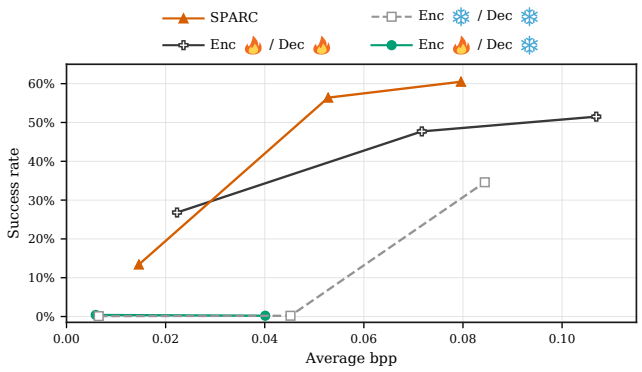


Figure 10: Freezing encoder or decoder on SPARC

We test the effects of freezing key components on the performance of SPARC to verify why we choose to freeze encoder, and train decoder. Figure 10 compares average success rates across a range of bpp values between SPARC and variants in which the encoder and/or decoder are frozen or fine-tuned. SPARC reaches a success rate of 56% at around 0.035 bpp and 60% at 0.08 bpp. It outperforms the fully fine-tuned variant across most of the bpp range, and the fully fine-tuned variant attains lower success rates (47% and 51%) even at higher bitrates (0.072 and 0.107 bpp). SPARC falls behind the fully fine-tuned variant only in the very-low-bpp regime (below roughly 0.03 bpp), where its success rate drops substantially. Freezing the decoder has a severe impact. The variant with a trainable encoder and a frozen decoder fails completely, with 0% success at all evaluated bitrates. When both the encoder and decoder are frozen, the success rate stays at 0% up to about 0.045 bpp and reaches only 34% at 0.085 bpp. These results indicate that adapting the decoder is essential for preserving action-relevant information for downstream VLA models. They also show that SPARC's training design achieves a better rate–performance trade-off than naively fine-tuning all components.

D TRANSFERABILITY ANALYSIS

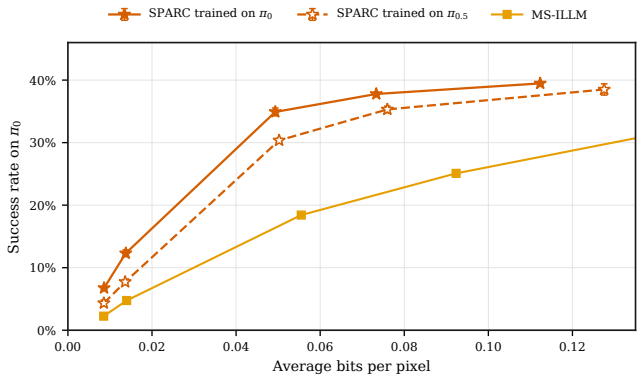


Figure 11: **Transfer to another VLA backbone.** We train SPARC on $\pi_{0.5}$ and run inference with different VLA backbone, π_0.

Figure 11 evaluates whether SPARC's learned compression strategy generalizes to a VLA backbone unseen during training. We train SPARC on $\pi_{0.5}$ and evaluate it on π_0, comparing against the oracle setting (SPARC trained directly on π_0) and the MS-ILLM baseline. Despite never observing π_0 during training, SPARC trained on $\pi_{0.5}$ consistently outperforms MS-ILLM across the full bitrate range, and the gap to the oracle narrows as bitrate increases. We attribute this to the fact that the

visual regions critical to task success—e.g., the gripper and target object—are largely determined by the manipulation task itself rather than by the specific VLA backbone, so a codec trained to preserve task-relevant information on one backbone transfers reasonably well to another. The residual gap to the oracle likely reflects backbone-specific differences in attention granularity rather than a failure of transfer.

E LIBERO-PLUS BENCHMARK

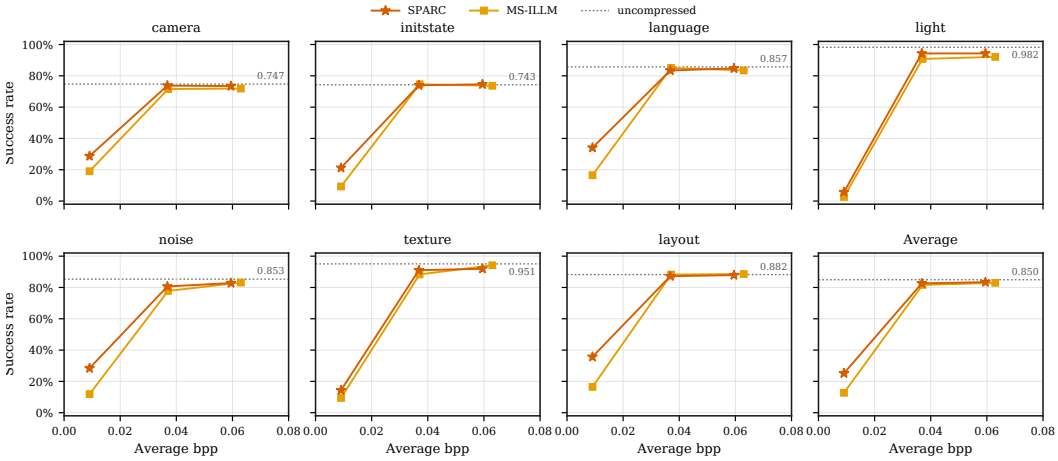


Figure 12: **LIBERO-Plus results.** *Average* represents the average of other seven perturbation dimensions.

To assess robustness beyond the standard LIBERO suite, we evaluate on LIBERO-Plus, which perturbs each episode along seven axes—camera pose, initial state, language instruction, lighting, sensor noise, texture, and object layout—as well as an aggregate setting combining all perturbations (`Average`). Figure 12 shows that the gains observed on standard LIBERO transfer consistently to this more challenging setting: across nearly all perturbation types, SPARC matches or exceeds MS-ILLM at every bitrate and closes the gap to the uncompressed baseline more quickly. The advantage is most pronounced in the low-bitrate regime ($\sim$0.01 bpp), where SPARC roughly doubles the success rate of MS-ILLM under `initstate`, `noise`, `texture`, and the `Average` aggregate, indicating that SPARC's compression better preserves task-relevant information precisely when bit budgets are tightest. The two methods converge as bitrate increases, with both approaching the uncompressed success rate by $\sim$0.03–0.05 bpp in most categories; the `light` perturbation is a partial exception, where the uncompressed baseline is already near-ceiling (0.982) and both compressed variants converge to it almost immediately. Overall, these results indicate that SPARC's gains are not an artifact of the standard LIBERO setup but generalize across a broad range of visual and semantic perturbations.

F TASK-CRITICALITY OF THE MASK SELECTOR

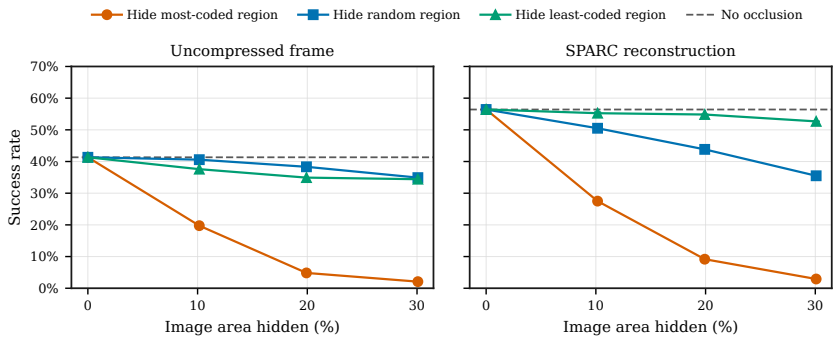


Figure 13: Effect of occluding masked regions.



Figure 14: Input images examples.

Figure 13 examines whether the regions prioritized by the mask selector are indeed task-critical. We rank image regions by the bitrate allocated by the mask selector and occlude the most-coded, least-coded, or randomly chosen regions, covering 10–30% of the image area. Occlusion is applied both to uncompressed frames (left) and to SPARC reconstructions (right). In both settings, hiding the most-coded regions is by far the most harmful: the success rate is roughly halved already at 10% occlusion and collapses from 41% to 2% on uncompressed frames and from 57% to 3% on SPARC reconstructions at 30% occlusion. Hiding random regions causes only a moderate drop, to about 35% and 36%, respectively. Hiding the least-coded regions is nearly harmless for SPARC reconstructions (56% $\rightarrow$ 53%), whereas on uncompressed frames it degrades performance comparably to random occlusion (41% $\rightarrow$ 34%). This gap is likely because the least-coded regions are already reconstructed at low fidelity by SPARC, so occluding them removes little additional information. Notably, the most-coded regions are equally critical on uncompressed frames, indicating that the selector identifies content that is intrinsically important for the downstream task rather than an artifact of compression. These results confirm that the mask selector concentrates bits on the regions that matter most for downstream task performance.

G VISUAL QUALITY ASSESSMENT

To assess how image quality changes with our training recipe, we compare the quality of images produced by MS-ILLM and SPARC using image quality assessment metrics. On RoboCasa365 benchmarks, we collect 800 sample images of SPARC and MS-ILLM, where images samples of SPARC are optimized for $\pi_{0.5}$ across a range of different bpps and measure their average qualities using three well-known image quality assessment methods: (1) PSNR (Peak Signal-to-Noise Ratio), (2) SSIM (Structural Similarity Index Measure) and (3) LPIPS (Learned Perceptual Image Patch Similarity).

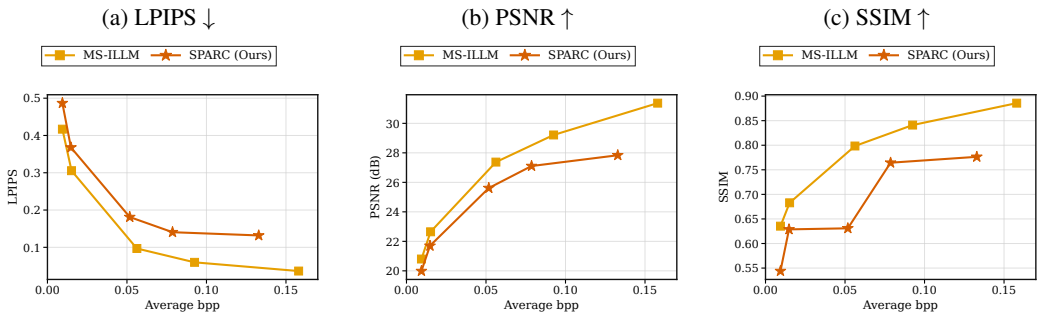


Figure 15: Visual quality assessment between SPARC and MS-ILLM

Figure 15 compares SPARC and MS-ILLM in terms of image quality. We observe that SPARC generates images with lower perceptual quality across a wide range of bpp values compared to MS-ILLM. An interesting characteristic shared by all three metrics is that the improvement in the quality of SPARC images follows that of MS-ILLM at low bpp values, but soon stagnates beyond a certain bpp range ($0.03 \sim 0.05$ bpp), where the performance of downstream VLA models becomes sufficiently high. This implies that, beyond a certain point, improving perceptual quality does not contribute to further improvements in VLA model performance. SPARC is instead optimized for reconstructing task-relevant details for downstream VLA models.

H PER-TASK ANALYSIS

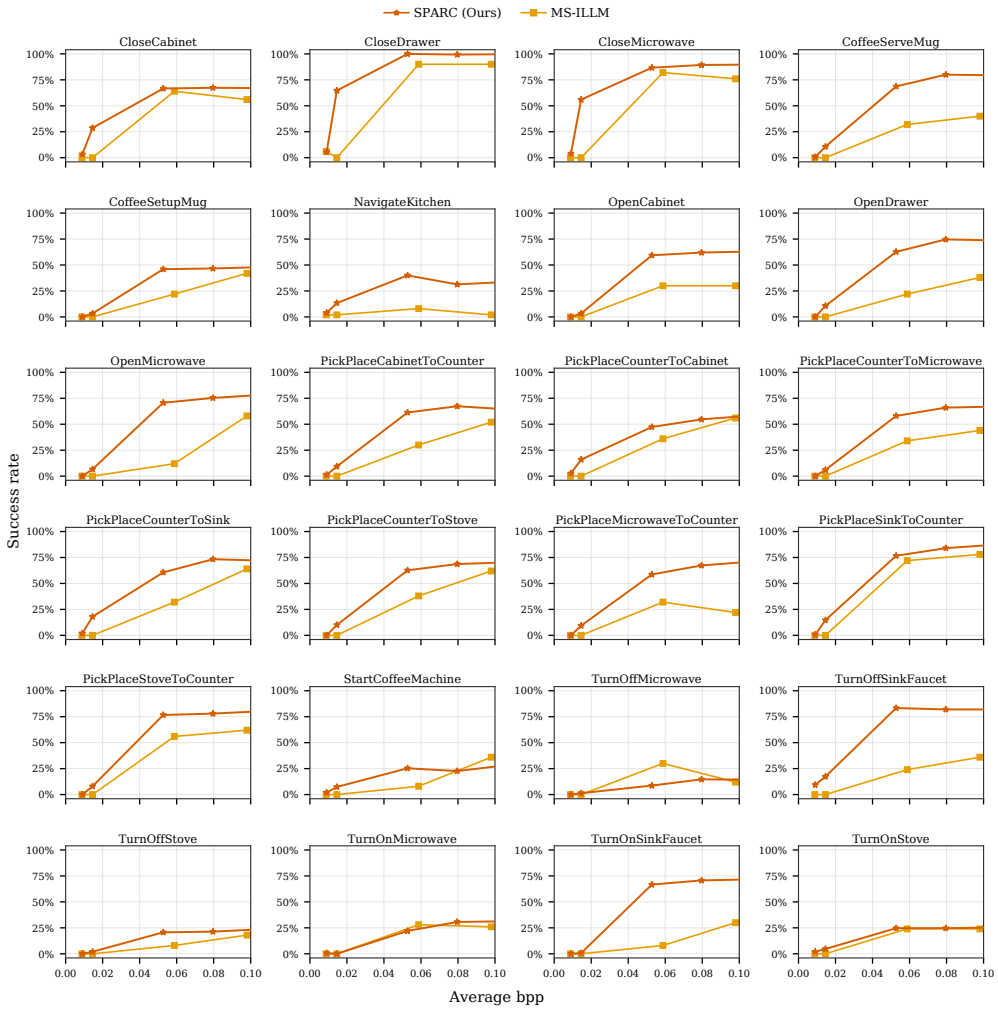


Figure 16: Per-task analysis of RoboCasa365 benchmarks

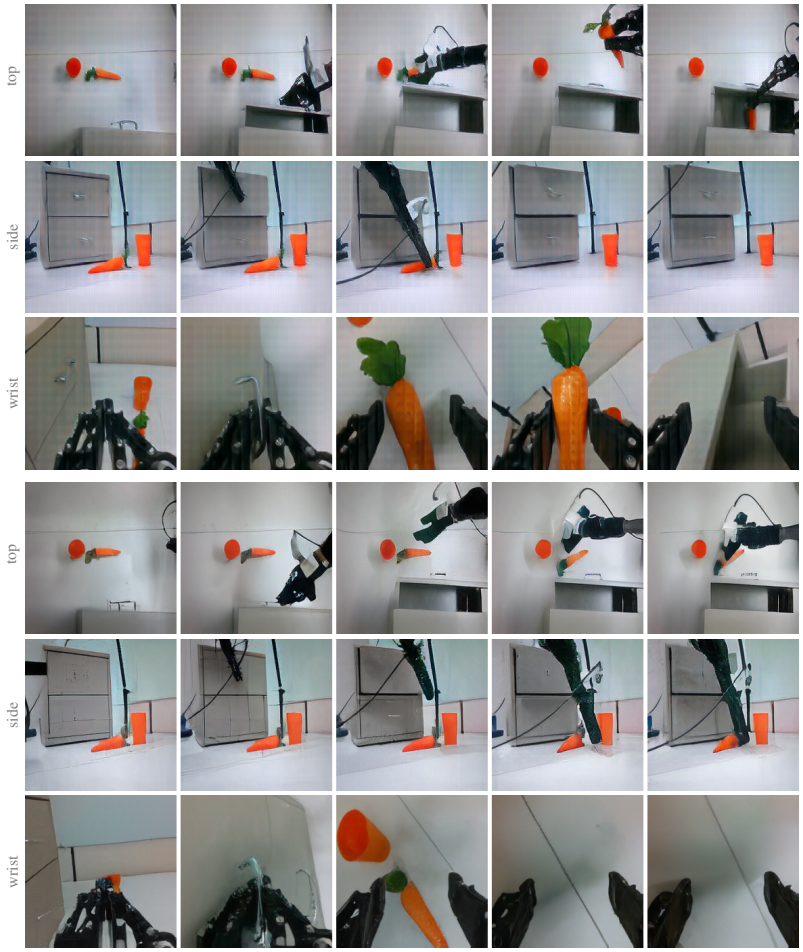
To analyze the advantages of SPARC in depth, we conduct a per-task analysis of SPARC compared to MS-ILLM on RoboCasa365 benchmarks with $\pi_{0.5}$ VLA model as the downstream VLA model. Following the evaluation pipeline in Section 5.1, we analyze 24 atomic tasks in RoboCasa365 benchmark. In Figure 16, we report the task success rate across a wide range of bpp values. We observe that SPARC achieves higher performance than MS-ILLM on most tasks, but the magnitude of the performance gain differs across tasks. SPARC tends to achieve greater performance gains on tasks for which identifying objects and the environment with more textural high-frequency details is important, such as *CloseMicrowave*, *CoffeeServeMug*, *CloseDrawer* and *TurnOffSinkFaucet*. In contrast, the performance gain from SPARC is not significant for tasks that require more complex dexterous operations, such as *TurnOffStove*, *TurnOnMicrowave*, and *TurnOffMicrowave*. We hypothesize that removing complicated background details and focusing on task-specific regions helps identify target objects so that VLA models are not distracted by unnecessary background details. However, dexterous tasks that test fine-grained manipulation details *rather* require more detailed information about how to manipulate the target objects. Nonetheless, SPARC is trained not to mask important latent features, which explains why the performance of SPARC is at least similar to or better than that of MS-ILLM on these tasks.

I REAL-WORLD ROLLOUT EXAMPLES

This section shows sample real-world rollouts of the $\pi_{0.5}$ policy on the Piper arm when its camera observations are compressed by SPARC or by MS-ILLM, on the three tasks *cabbage & carrot to pot*, *carrot to drawer*, and *stack cups*. Each episode is shown from its top, side, and wrist cameras at the same five timesteps, and the frames are the decoded images the policy actually received. The label next to each episode gives the method, its outcome, and the measured bitrate; for failed episodes it also names the failure mode.

Figure 17 pairs, for each task, a successful SPARC episode with a failed MS-ILLM episode at a comparable bitrate. Figure 20 collects successful SPARC episodes across tasks and bitrates, and Figure 24 shows representative SPARC failures. Figure 25 shows MS-ILLM failures across tasks and bitrates for comparison.

Carrot to drawer *"pick up the carrot and put it in the upper drawer"*



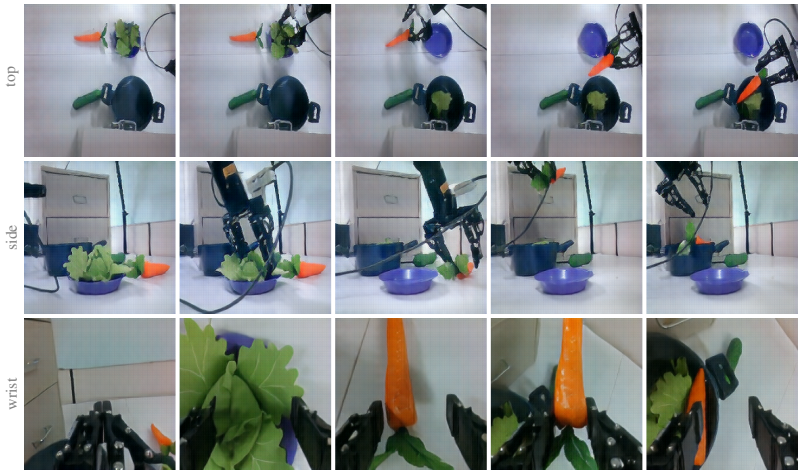
✓ **SPARC**
0.051 bpp

✗ **MS-ILLM**
0.052 bpp
misses the grasp

Figure 17: **Real-world rollouts: SPARC vs. MS-ILLM.** For each task, SPARC (ours; top) succeeds while MS-ILLM (bottom) fails at a comparable bitrate. The italic note under each label names the failure mode. Each episode is shown by its top, side, and wrist cameras (three rows) at the same timesteps; time flows left to right, frames are the decoded images the $\pi_{0.5}$ policy received on the Piper arm, and the gray label gives the measured bitrate.

Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✓ **SPARC**
0.152 bpp



✗ **MS-ILLM**
0.178 bpp
cannot lift the cabbage

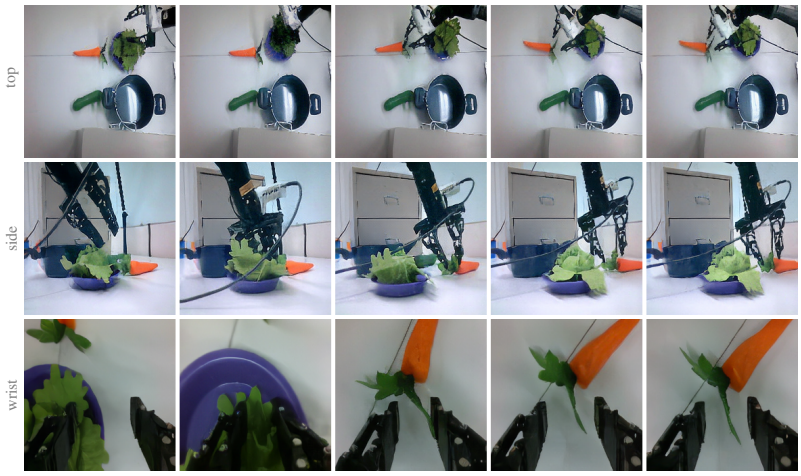
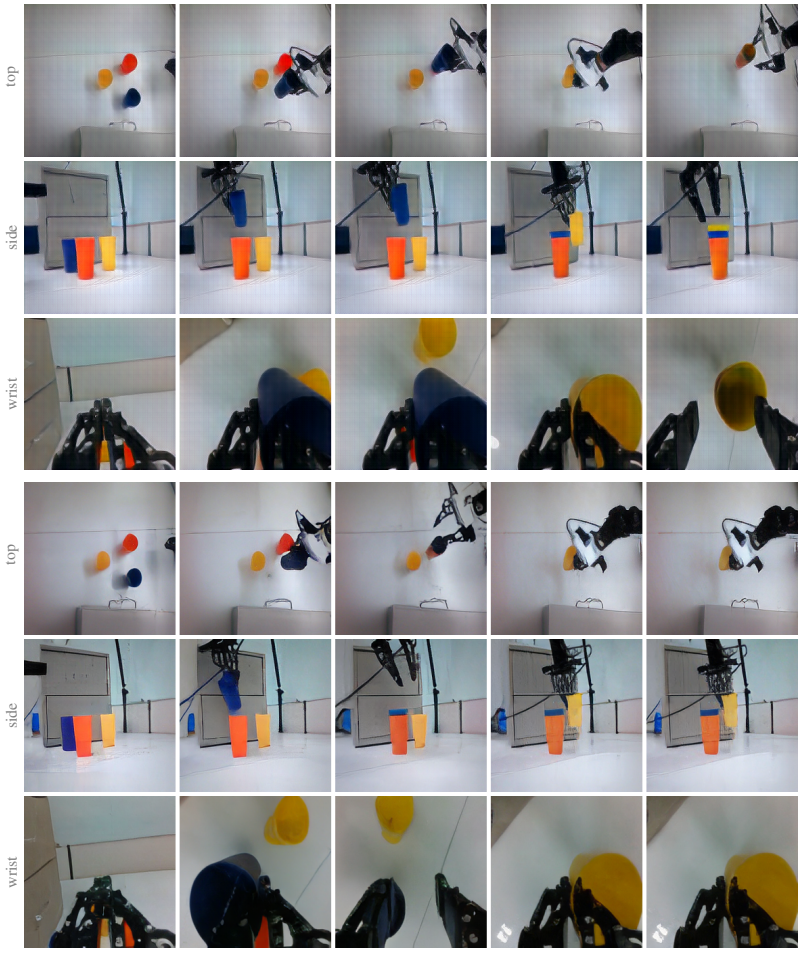


Figure 18: **Real-world rollouts: SPARC vs. MS-ILLM.** (continued from Figure 17).

Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*



✓ **SPARC**
0.047 bpp

✗ **MS-ILLM**
0.052 bpp
never places the yellow cup

Figure 19: **Real-world rollouts: SPARC vs. MS-ILLM.** (continued from Figure 17).

Carrot to drawer *"pick up the carrot and put it in the upper drawer"*

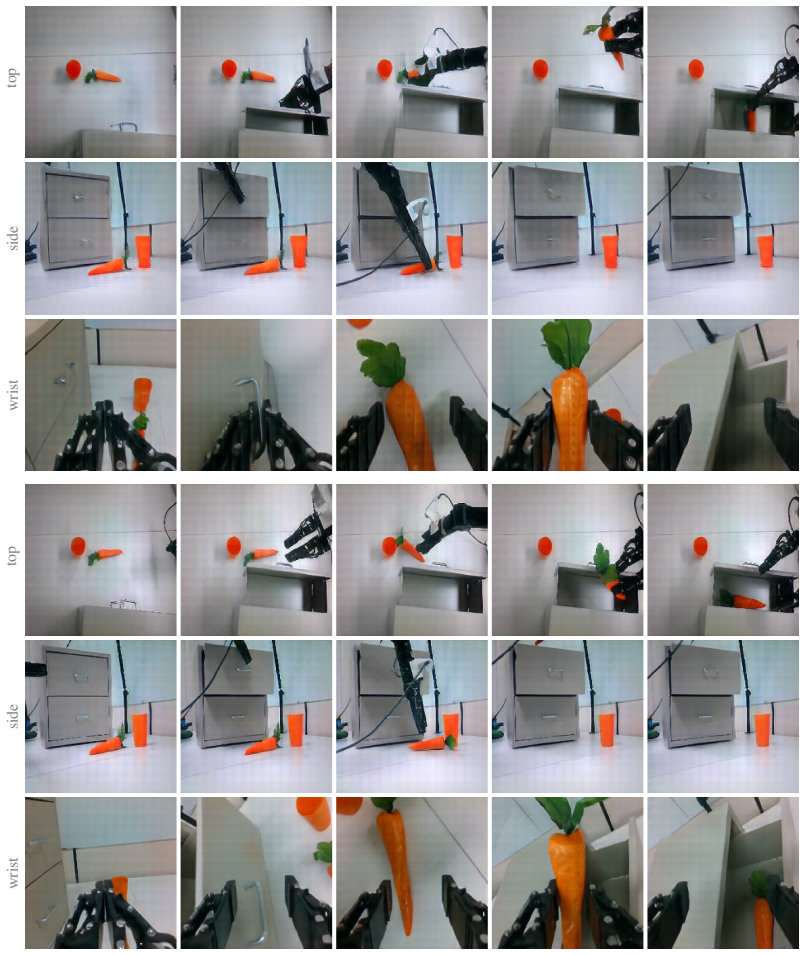
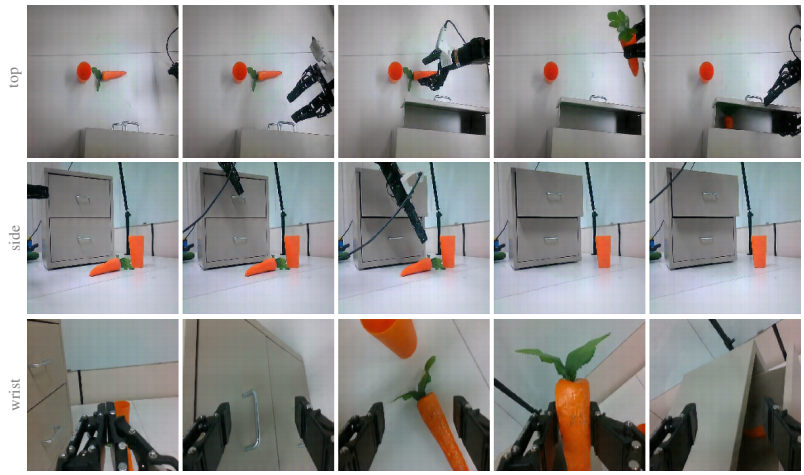


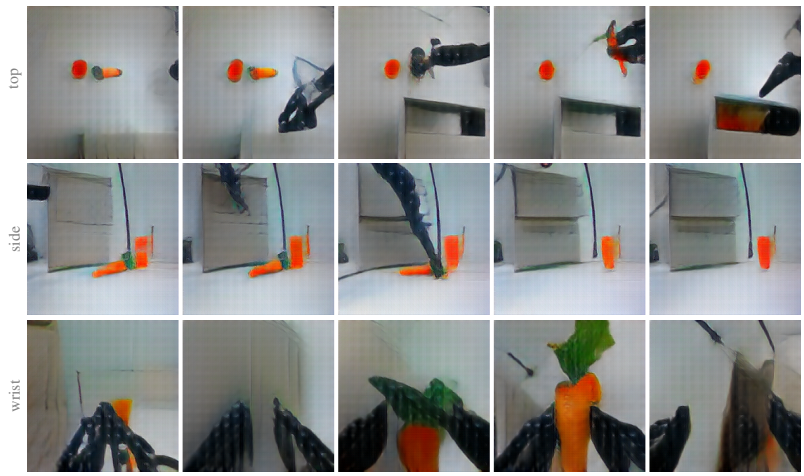
Figure 20: **Real-world rollouts with SPARC: successful episodes** across tasks and bitrates. Each episode is shown by its top, side, and wrist cameras (three rows) at the same timesteps; time flows left to right, frames are the decoded images the $\pi_{0.5}$ policy received on the Piper arm, and the gray label gives the measured bitrate.

Carrot to drawer *"pick up the carrot and put it in the upper drawer"*

✓ **SPARC**
0.496 bpp



✓ **SPARC**
0.017 bpp



Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✓ **SPARC**
0.057 bpp

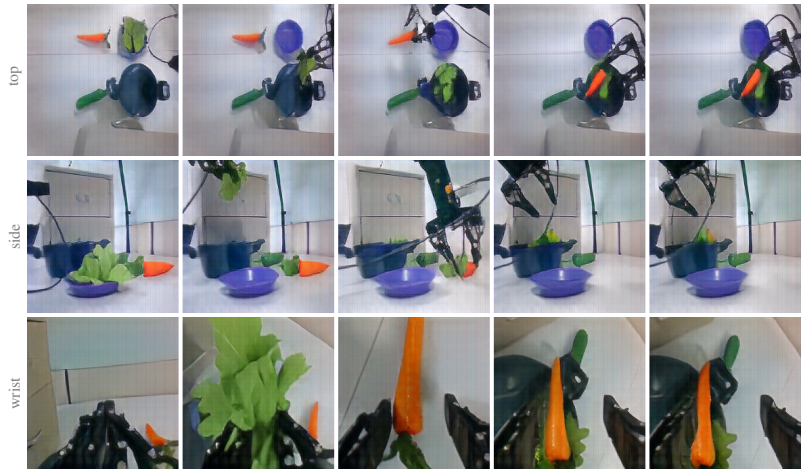
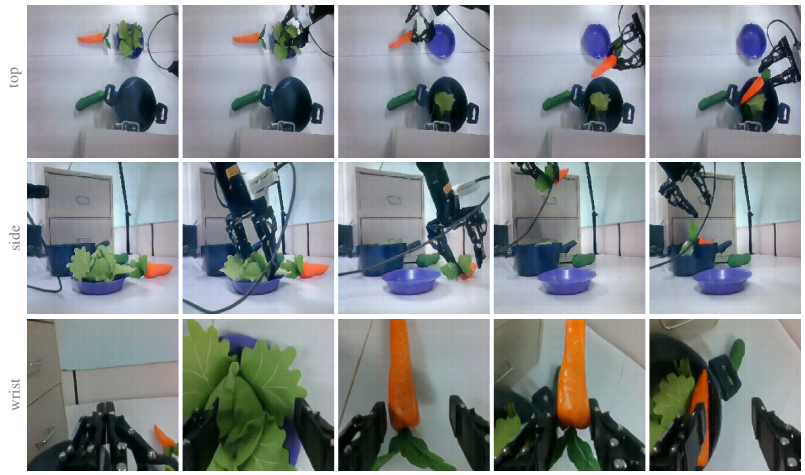


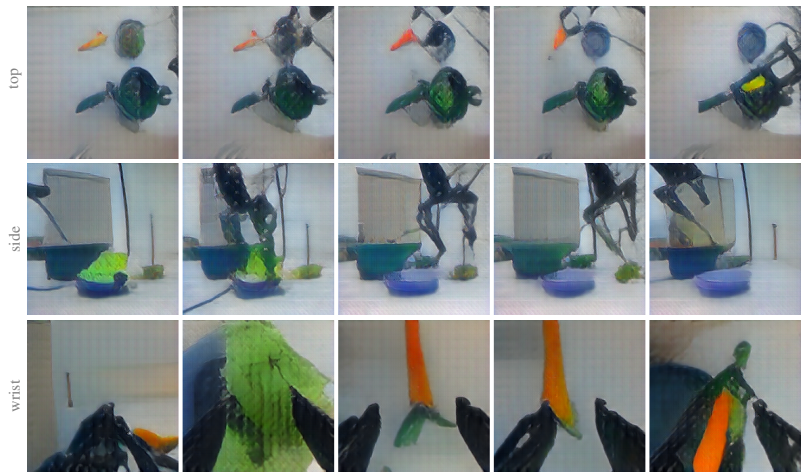
Figure 21: **Real-world rollouts with SPARC: successful episodes** (continued from Figure 20).

Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✓ **SPARC**
0.152 bpp



✓ **SPARC**
0.019 bpp



Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*

✓ **SPARC**
0.047 bpp

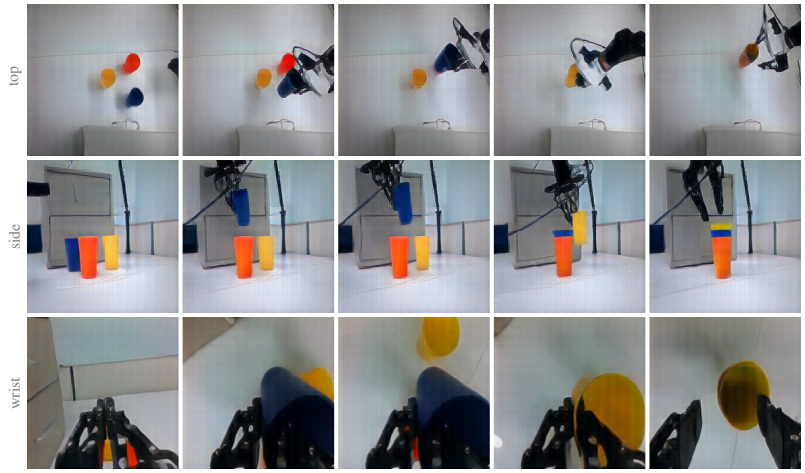
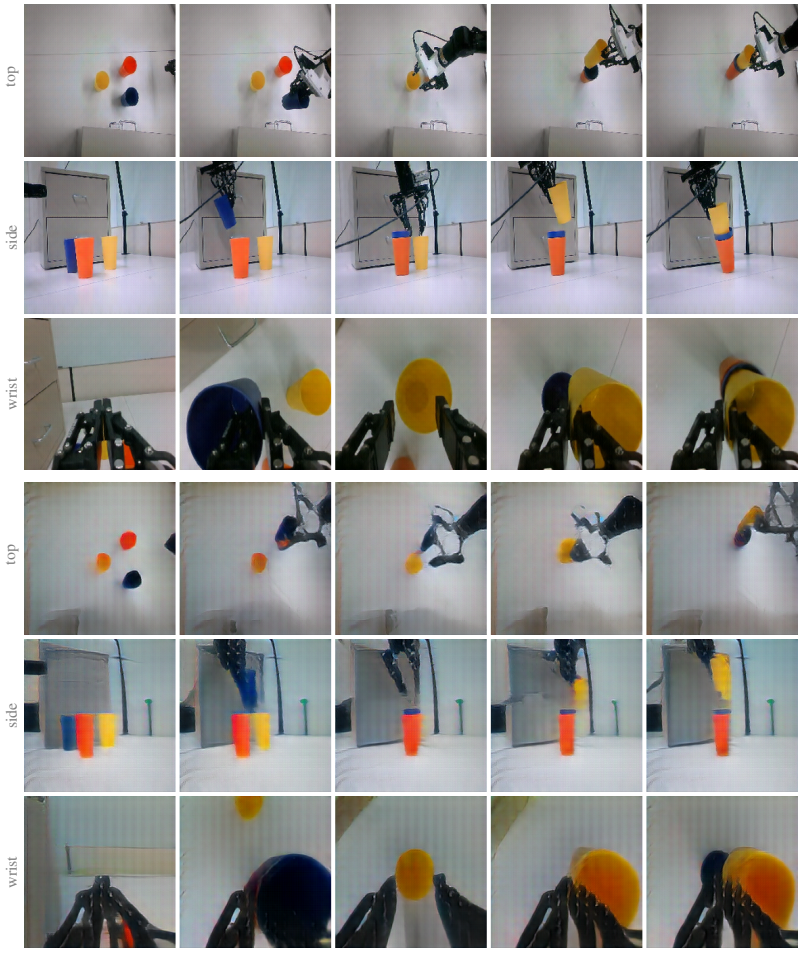


Figure 22: **Real-world rollouts with SPARC: successful episodes** (continued from Figure 20).

Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*



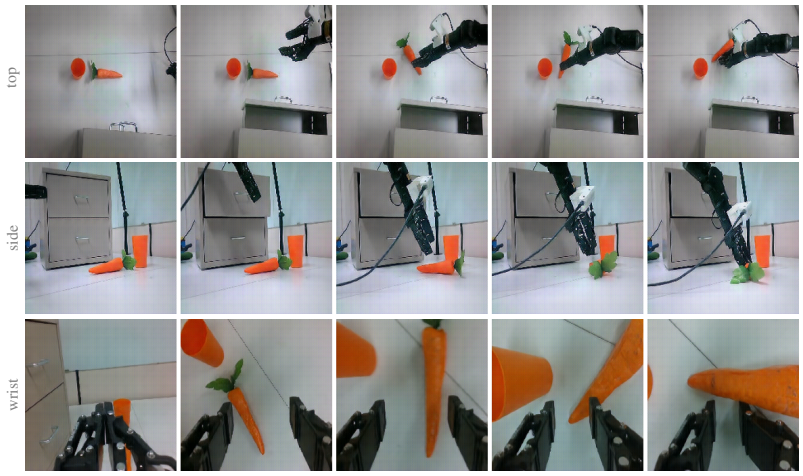
✓ **SPARC**
0.443 bpp

✓ **SPARC**
0.016 bpp

Figure 23: **Real-world rollouts with SPARC: successful episodes** (continued from Figure 20).

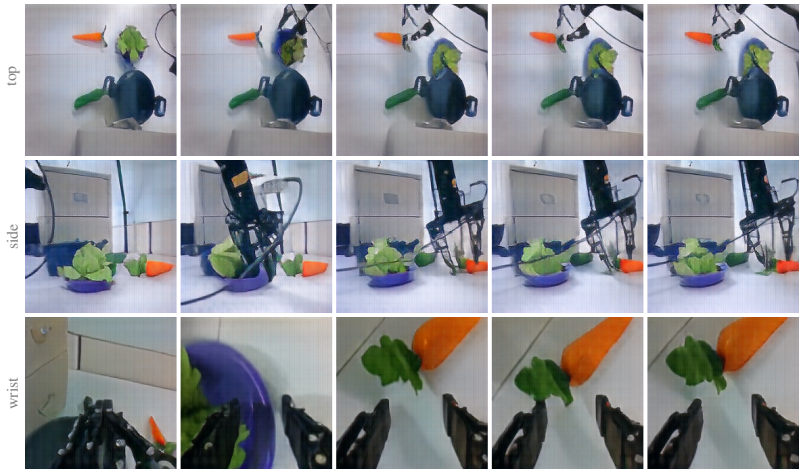
Carrot to drawer *"pick up the carrot and put it in the upper drawer"*

✗ **SPARC**
0.504 bpp
misses the grasp



Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✗ **SPARC**
0.057 bpp
never picks up the carrot



Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*

✗ **SPARC**
0.127 bpp
stacks on the wrong cup

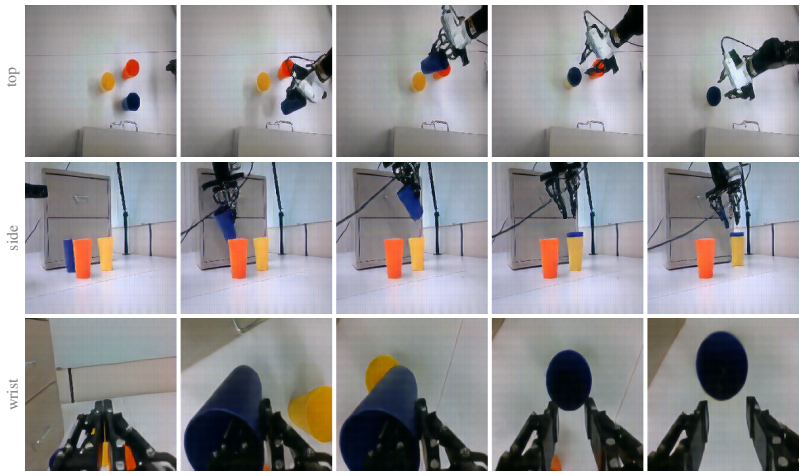


Figure 24: **Real-world rollouts with SPARC: failure cases.** The italic note under each label names the failure mode. Each episode is shown by its top, side, and wrist cameras (three rows) at the same timesteps; time flows left to right, frames are the decoded images the $\pi_{0.5}$ policy received on the Piper arm, and the gray label gives the measured bitrate.

Carrot to drawer *"pick up the carrot and put it in the upper drawer"*

✗ **MS-ILLM**
0.052 bpp
misses the grasp

✗ **MS-ILLM**
0.574 bpp
misses the grasp

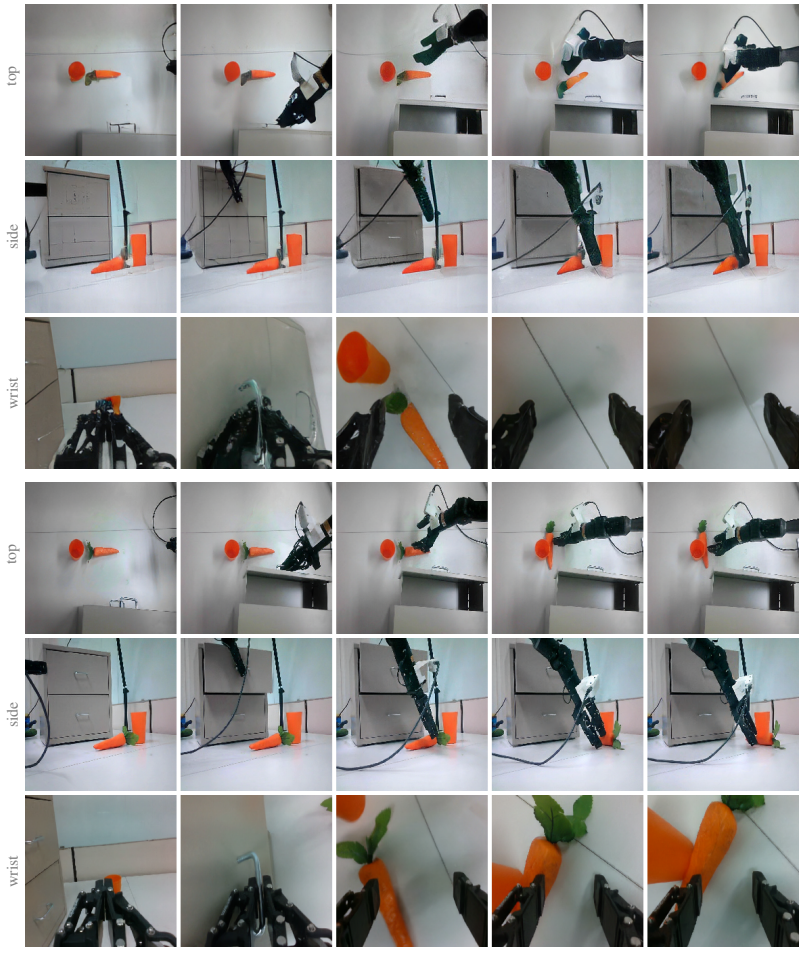


Figure 25: **Real-world rollouts with MS-ILLM: failure cases** across tasks and bitrates. The italic note under each label names the failure mode. Each episode is shown by its top, side, and wrist cameras (three rows) at the same timesteps; time flows left to right, frames are the decoded images the $\pi_{0.5}$ policy received on the Piper arm, and the gray label gives the measured bitrate.

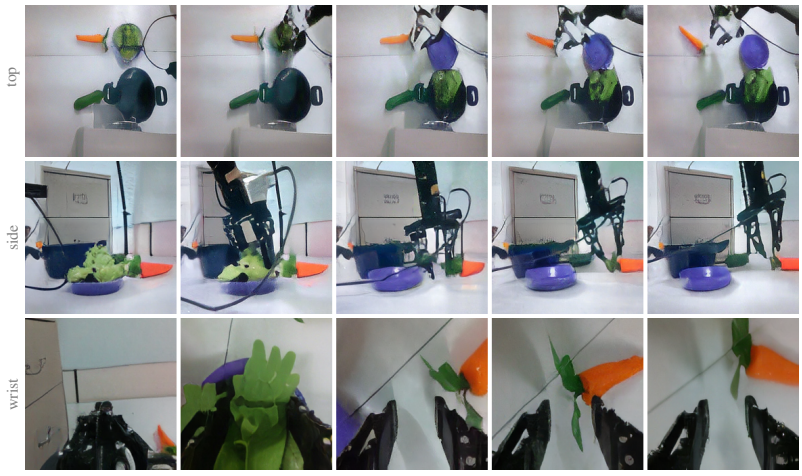
Carrot to drawer *"pick up the carrot and put it in the upper drawer"*

✗ **MS-ILLM**
0.015 bpp
never opens the drawer



Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✗ **MS-ILLM**
0.064 bpp
never places the carrot



✗ **MS-ILLM**
0.178 bpp
cannot lift the cabbage

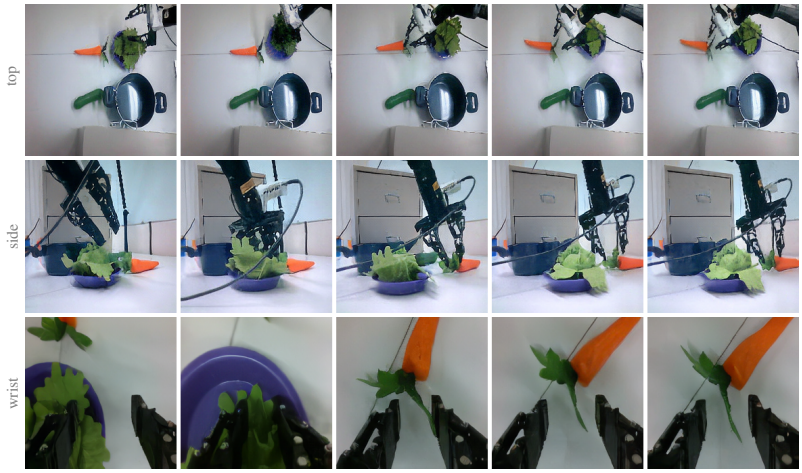
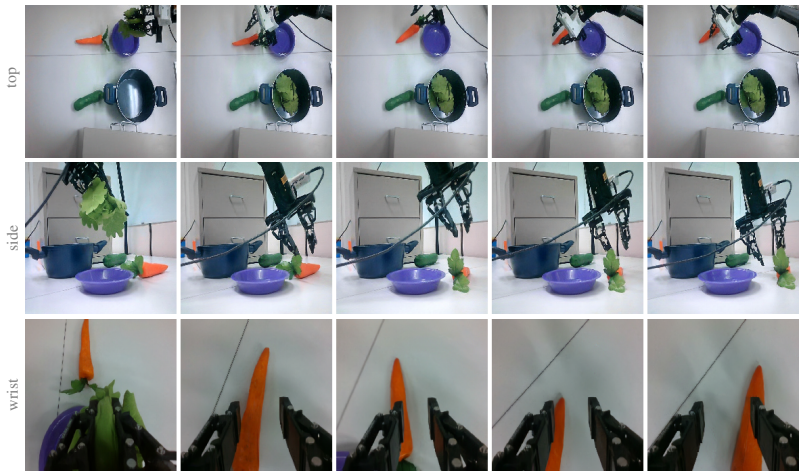


Figure 26: **Real-world rollouts with MS-ILLM: failure cases** (continued from Figure 25).

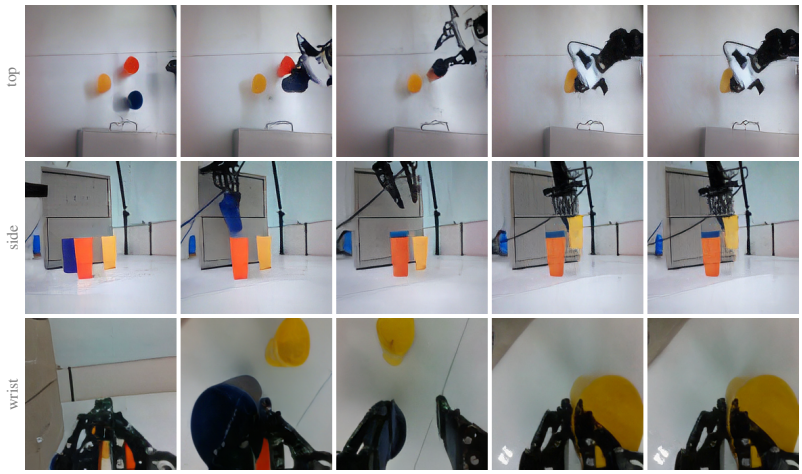
Cabbage & carrot to pot *"take the cabbage from the dish, and put it in the pot, then add the carrot"*

✗ **MS-ILLM**
0.683 bpp
never places the carrot



Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*

✗ **MS-ILLM**
0.052 bpp
never places the yellow cup



✗ **MS-ILLM**
0.149 bpp
knocks the yellow cup over

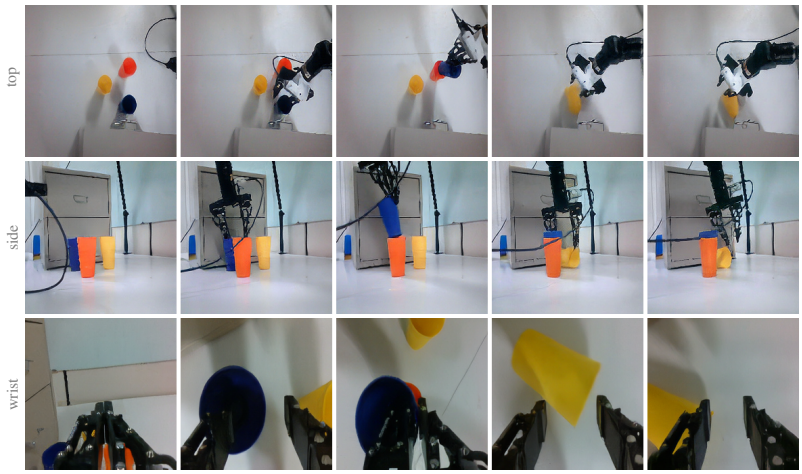
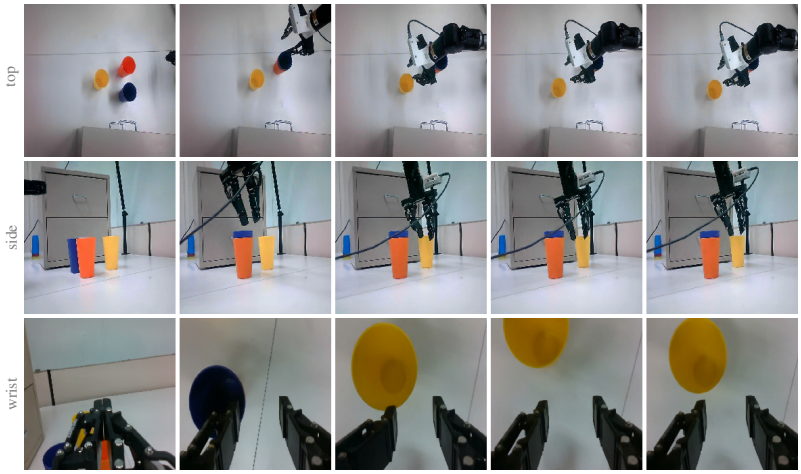


Figure 27: **Real-world rollouts with MS-ILLM: failure cases** (continued from Figure 25).

Stack cups *"stack the purple cup on the orange one, and put the yellow one on top"*

✗ **MS-ILLM**
0.564 bpp
never places the yellow cup



✗ **MS-ILLM**
0.015 bpp
fails to grasp a cup

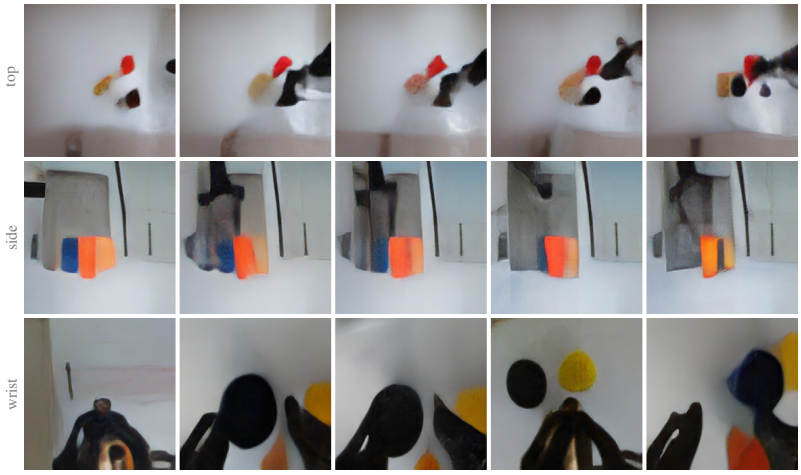


Figure 28: **Real-world rollouts with MS-ILLM: failure cases** (continued from Figure 25).

J Failure Case Analysis

SPARC rarely fails outright. On the two pick-and-place tasks, SPARC fails the first stage in only 2 of 80 attempts, against 31 of 80 for MS-ILLM. This matches the uncompressed policy, which also never fails the first stage there. SPARC's remaining failures on these tasks are therefore second-stage failures, the same kind the uncompressed policy makes in 40–50% of its attempts. The gap is widest at the lowest bitrate: MS-ILLM fails the first task in 28 of 30 attempts, and SPARC in 9 of 30, 8 of which are on *stack cups*. SPARC reaches this while spending 3–19% fewer bits than MS-ILLM at the three higher quality levels.

Failure modes. We inspected the decoded frames of every episode and grouped the failures. SPARC's failures are concentrated in the precise placement of *stack cups*, where the gripper must line up a cup rim with another in the side view. This task is also the hardest for the uncompressed policy (30% zero-score attempts), so part of this failure rate reflects the policy rather than the codec. MS-ILLM's failures instead extend to the first stage of every task and, at the lowest bitrate, to losing the scene altogether.

K Additional Visual Examples

Following Section 5.3, we provide additional visualization examples for tasks where the average performance gain of SPARC is high or low. For both visualizations, the bit allocation map shows that SPARC allocates more bits to task-relevant regions (i.e. near the microwave and the gripper's end-effector), supporting the analysis in Section H. In addition, SPARC also tends to allocate more bits to objects that are generally not used in other tasks. This may imply that allocating more bits to static objects provides reference points and is required for recognizing distinctive scene features in simulation environments where object locations are randomized. Overall, these examples show that SPARC is capable of allocating more bits to task-centric regions to improve the performance of VLA models.

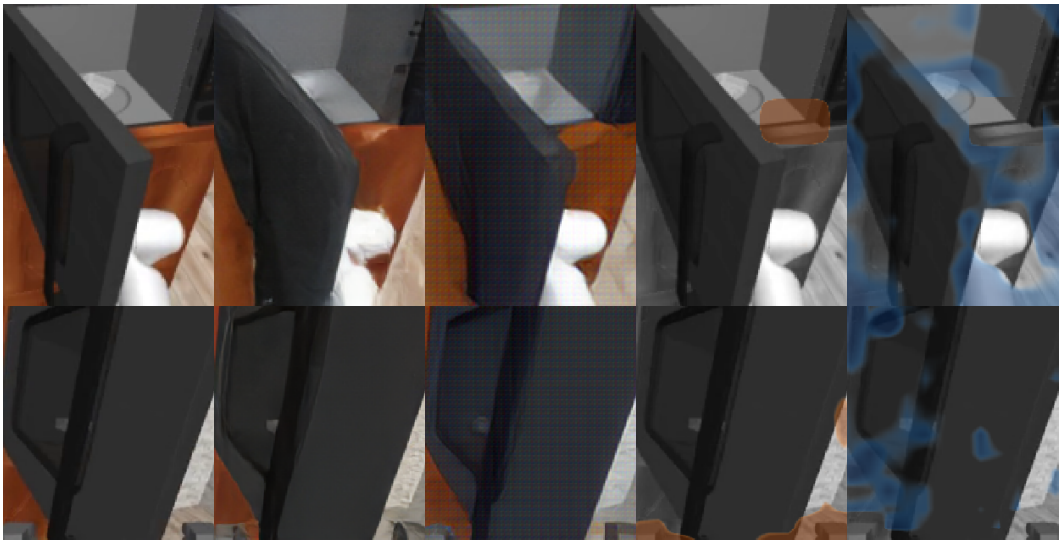


Figure 29: CloseMicrowave,"Close the microwave door.".

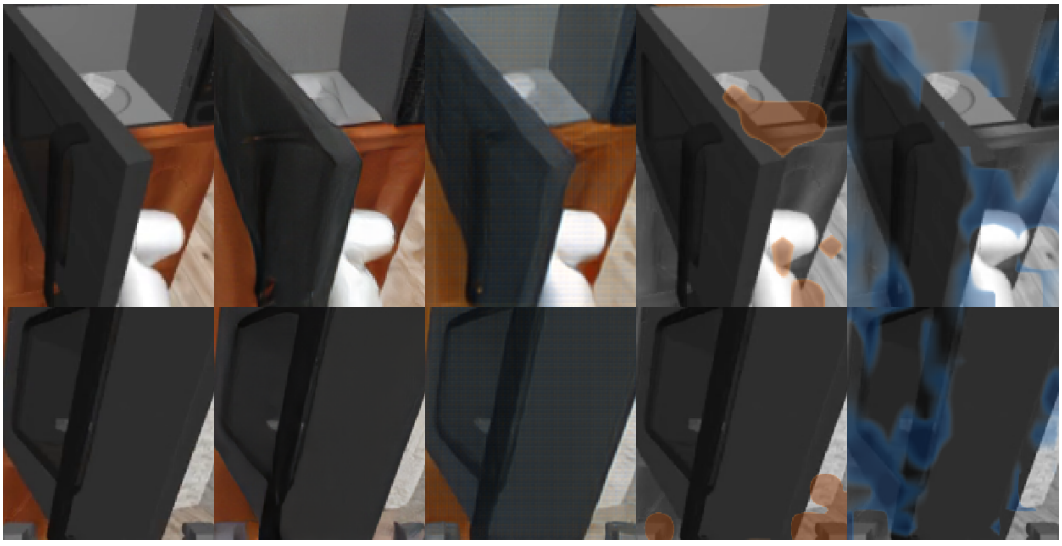


Figure 30: CloseMicrowave, "Close the microwave door.".

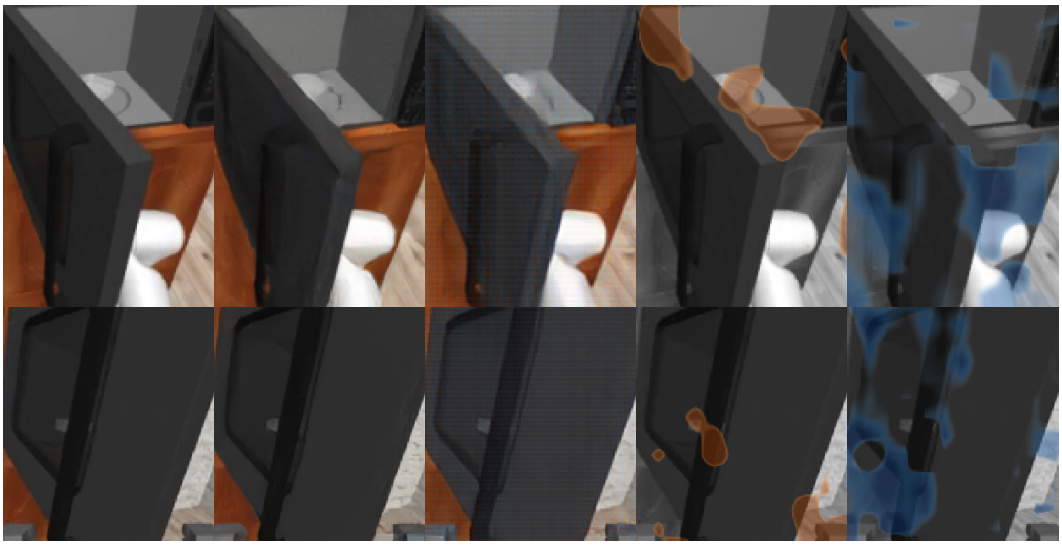


Figure 31: CloseMicrowave, task instruction "Close the microwave door.".

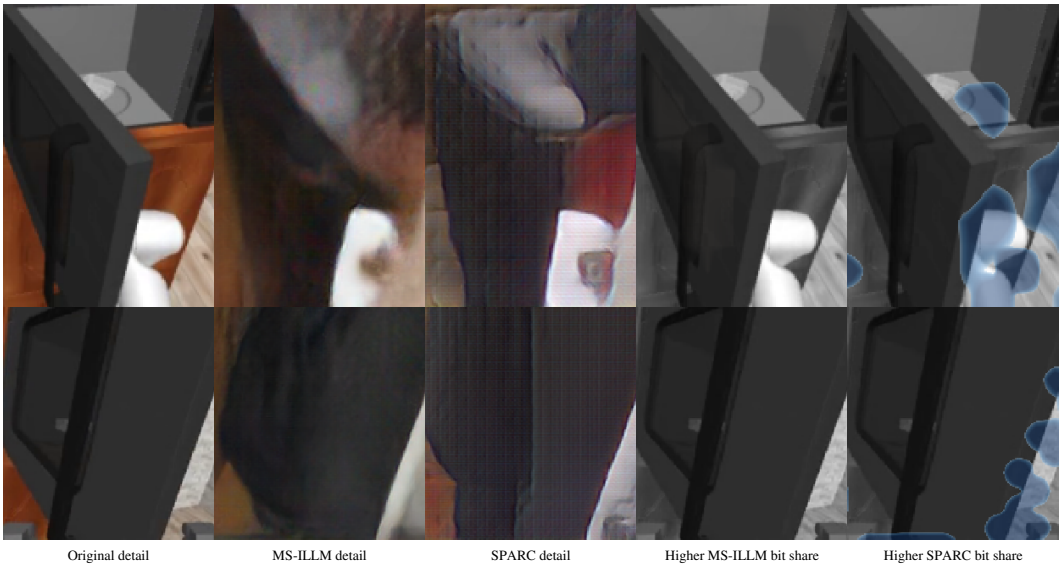
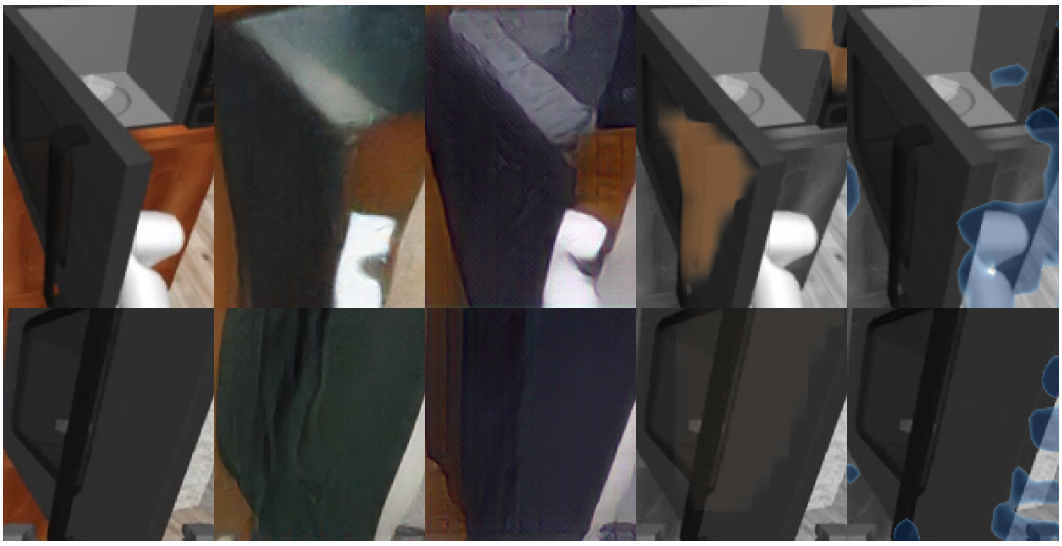
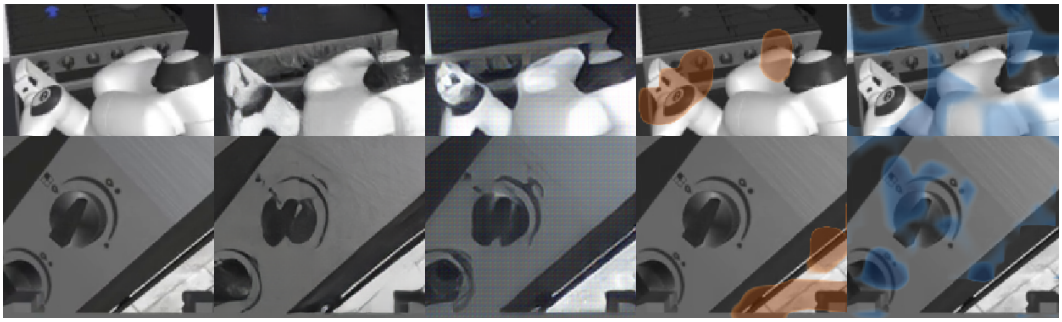


Figure 32: CloseMicrowave, "Close the microwave door.".



Task: "Close the microwave door."

Figure 33: CloseMicrowave, "Close the microwave door.".



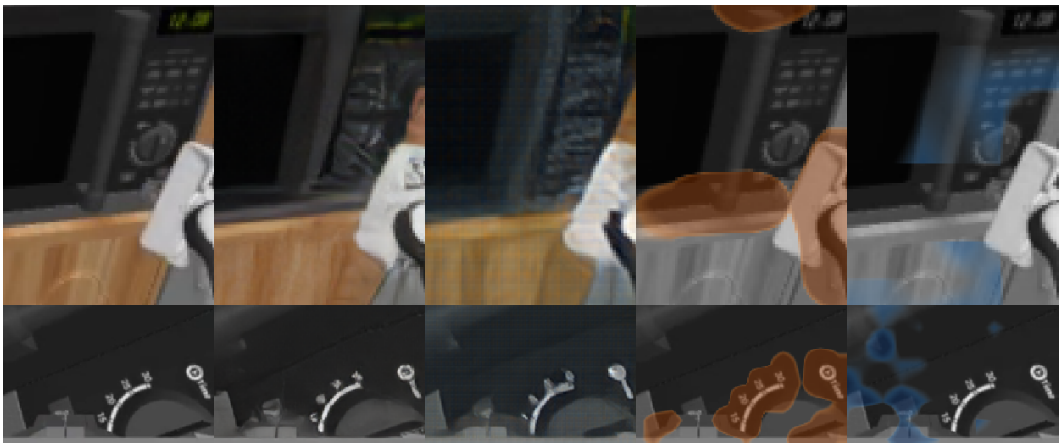
Task: "Turn off the front left burner of the stove."

Figure 34: TurnOffStove, "Turn off the front left burner of the stove.".



Task: "Press the start button on the microwave."

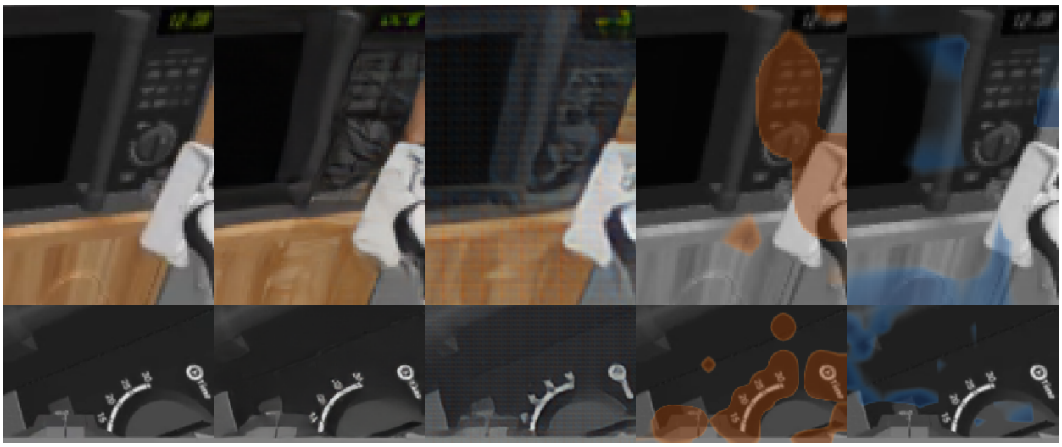
Figure 35: TurnOnMicrowave, "Press the start button on the microwave.".



Original detail | MS-ILLM detail | SPARC detail | Higher MS-ILLM bit share | Higher SPARC bit share

Task: "Press the start button on the microwave."

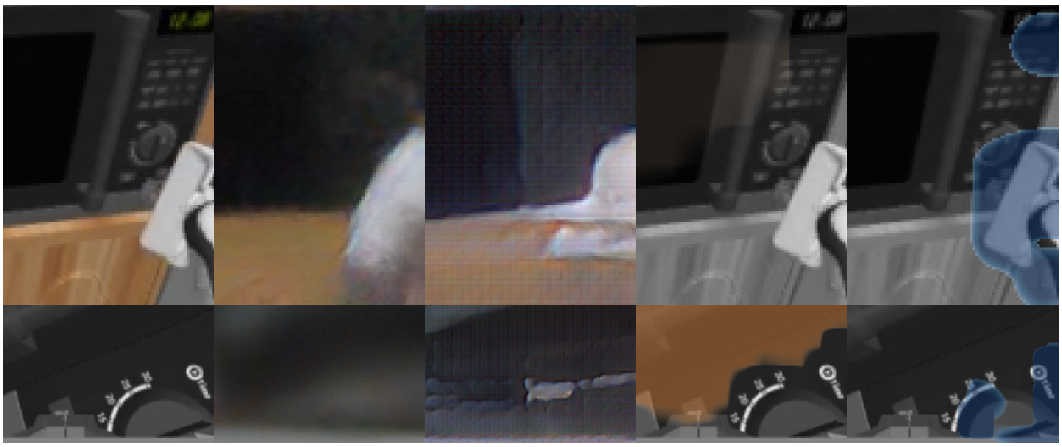
Figure 36: TurnOnMicrowave, "Press the start button on the microwave.".



Original detail | MS-ILLM detail | SPARC detail | Higher MS-ILLM bit share | Higher SPARC bit share

Task: "Press the start button on the microwave."

Figure 37: TurnOnMicrowave, "Press the start button on the microwave.".



Original detail | MS-ILLM detail | SPARC detail | Higher MS-ILLM bit share | Higher SPARC bit share

Task: "Press the start button on the microwave."

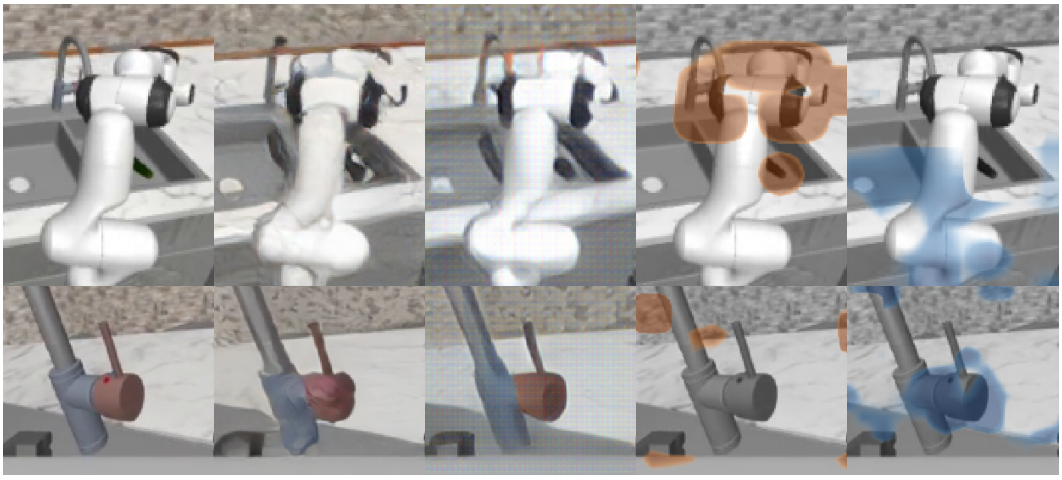
Figure 38: TurnOnMicrowave, "Press the start button on the microwave.".



Original detail | MS-ILLM detail | SPARC detail | Higher MS-ILLM bit share | Higher SPARC bit share

Task: "Press the start button on the microwave."

Figure 39: TurnOnMicrowave, "Press the start button on the microwave.".



Original detail | MS-ILLM detail | SPARC detail | Higher MS-ILLM bit share | Higher SPARC bit share

Task: "Turn on the sink faucet."

Figure 40: TurnOnSinkFaucet, "Turn on the sink faucet.".