GEMA: Generative Memory for Robotic Manipulation

First Author **Second Author**
Institution Name
author@example.org

Abstract: Long-horizon robot manipulation requires memory of earlier interactions, yet the information needed from that memory changes as a task unfolds. We introduce GEMA (Generative Memory for Action), a modular framework that turns retained experience into task-specific references for control. Drawing on constructive accounts of human episodic memory, GEMA uses a multimodal large language model as a memory foundation model to interpret historical evidence and generate textual and visual guidance. This reference interface accommodates agentic controllers, world action models, and vision-language-action policies. A task agent selects subtasks, coordinates generation and execution, and assesses progress from observed outcomes. The framework supports recall, historical comparison, and construction of new targets under partial observability. On Task-Based evaluation, GEMA improves mean success by approximately eight percentage points over Gated Memory Policy across tasks requiring historical comparison, target construction, and sequence recall. We additionally introduce continuous-query (Con-Q) evaluation across sequential, spatial, and semantic memory, testing sustained execution under continuously generated queries with different downstream policies. These results motivate generative memory as a reusable interface between past experience and ongoing robot action. Project website: https://beautiful-genie-18580b.netlify.app/.

Keywords: Generative Memory, Partial Observability, Agentic Robotics, WAM

1 Introduction

Long-horizon manipulation requires information that is no longer visible. Language-model agents can decompose goals and revise plans from feedback [1, 2], but choosing the next action also depends on earlier observations. Under partial observability, a previously observed location, relation, or instruction may remain essential after the scene changes [3].

MemoryVLA consolidates perceptual and cognitive features, Gated Memory Policy (GMP) selectively attends to history, and MemoryVLA++ adds latent imagination [4, 5, 6]. These methods integrate history into learned control. Yet retrieval alone need not specify the requested action: placing a cube 10 cm right of its original location requires constructing a target absent from the record. An explicit interface can separate historical evidence from its interpretation for control.

Inspired by constructive accounts of episodic memory [7, 8], we introduce GEMA, *Generative Memory for Action*. A pretrained multimodal large language model (MLLM) interprets retained experience as task-specific textual and visual references; a downstream controller executes this guidance, while a task agent assesses observed outcomes (Figure 1). This interface can support vision–language–action (VLA), world action model (WAM), or agentic controllers. We evaluate GEMA-DreamZero, the DreamZero-based configuration.

Our contributions are the modular memory interface, its integration with execution feedback, and Task-Based and continuous-query (Con-Q) evaluation protocols for complementary memory demands.

10th Conference on Robot Learning (CoRL 2026), Austin, Texas, USA.

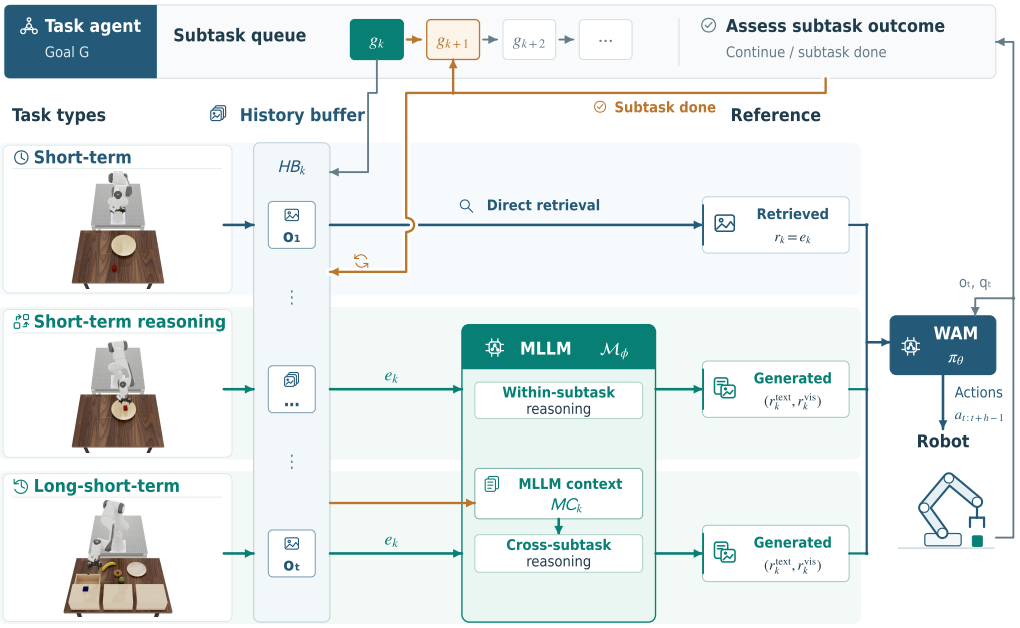


Figure 1: **Task-dependent memory for action.** The task agent selects subtasks and assesses execution. Retrieved evidence or MLLM-generated references guide the WAM alongside current observations. Amber arrows mark context updates, buffer refresh, and subtask advancement.

2 Problem and Benchmarks

A robot pursuing goal G observes image o_t and proprioception q_t. At execution round k, starting at t_k, the agent selects request g_k. Required information may be absent from the current view but available in earlier observations. We distinguish two complementary evaluation protocols.

Task-Based evaluation. *Transform* requires placing a cube 10 cm from its remembered original position after moving it to a plate. *Resume* requires approaching the drawer containing a previously stored cube after an intervening manipulation. Seven MIKASA variants test novelty comparison, color-sequence reproduction, and cue retention across intervening actions. Together, these nine tasks evaluate memory in controlled episodes.

Con-Q evaluation. Three custom benchmarks support procedurally generated query streams in a persistent scene, without resets between queries. Two LIBERO-derived settings emphasize *sequential memory* (event order and interrupted-task resumption) and *spatial memory* (historical locations and drawer associations); one RoboCasa-derived setting emphasizes *semantic memory* (order contents and historical associations). Each rollout ends after 15 minutes of simulated time or at the first task failure, whichever occurs first. Appendix C gives detailed protocols.

3 Method

GEMA separates the experience retained by memory from the guidance needed for the next action. Historical evidence grounds a task-specific reference; current observations ground its execution. The task agent connects these two stages as subtasks progress. Appendix B provides the complete formulation and algorithm.

3.1 Retained Experience

The history buffer HB_k stores the most recent subtask's observed history, including the active prefix for within-subtask queries, with a capacity of 250 frames. Longer subtask histories update retained MLLM context MC_k, which also carries selected information from preceding subtasks. We distinguish *short-term* recall, *short-term reasoning* through transformation or comparison, and *long-short-term* dependencies that additionally require earlier-subtask context (Appendix C.3).

3.2 Generating Task References

Generative memory constructs guidance from historical evidence conditioned on the current request. SigLIP text–image and image–image similarities rank buffer observations using g_k and o_{t_k} [9], yielding evidence e_k. Reference construction follows

$$r_k = \begin{cases} e_k, & \text{direct retrieval,} \\ \mathcal{M}_\phi(e_k, o_{t_k}, g_k), & \text{within-subtask generation,} \\ \mathcal{M}_\phi(\mathsf{MC}_k, e_k, o_{t_k}, g_k), & \text{cross-subtask generation,} \end{cases} \tag{1}$$

where $\mathcal{M}_\phi$ is the MLLM and generated $r_k = (r_k^{\text{text}}, r_k^{\text{vis}})$ combines textual guidance with a visual target.

3.3 Downstream Policy

A downstream policy predicts an action chunk of length h under a variant-specific memory condition:

$$\begin{aligned} c_k^{(v)} &= \mathcal{C}_v(r_k), \\ a_{t:t+h-1} &\sim \pi_\theta^{(v)}(\cdot \mid o_t, q_t, g_k, c_k^{(v)}), \end{aligned} \tag{2}$$

where $\mathcal{C}_v$ formats the memory reference for downstream variant v. We instantiate three variants: **GEMA-Astra** supplies text/image references to an agentic controller; **GEMA-DreamZero** adapts DreamZero [10] with FL2V visual-goal conditioning, independently encoding current/goal images and supplying text through cross-attention; **GEMA-Pi05** uses semantic conditioning through memory-derived text in its instruction prompt, through its language interface. Each execution round holds the active request and reference fixed across action chunks while observations update. Appendix B gives implementation details.

4 Experiments

Setup. The task protocols are defined in Section 2. Task-Based experiments use `GPT-5.6-Sol` as the generative model. Training uses 200/300 demonstrations for Transform/Resume and 250 per MIKASA variant [11]. Task-Based evaluation uses 100 episodes per task. Con-Q uses `GPT-6 Astra` as the generative model, with 10 evaluation rollouts per benchmark for GEMA-Astra and 50 each for GEMA-DreamZero and GEMA-Pi05. Appendix C.4 gives reporting details.

Metrics. *Task-Based:* Task success rate is the percentage of evaluation episodes that complete the task. We report per-task rates and their unweighted mean over all nine tasks. *Con-Q:* Query survival measures the percentage of rollouts that successfully complete at least N queries:

$$S(N) = \frac{\text{rollouts completing at least } N \text{ queries}}{\text{total started rollouts}} \times 100\%.$$

Each rollout ends at the first task failure or 15 minutes of simulation time. We plot $S(N)$ against the number of completed queries N.

Task-Based Evaluation Results. Across Transform, Resume, and seven MIKASA variants, GEMA-DreamZero achieves 52.7% mean success, compared with GMP's 44.2% (Table 1). GEMA-DreamZero exceeds GMP on eight of nine tasks, with the largest gain on Transform (88% versus 65%). On nine-candidate color comparison, success is 25% versus 33% for GMP.

Table 1: **Task-Based success (%).** Each task uses 100 evaluation episodes; Transform/Resume use 20 episodes for each of five backgrounds. DreamZero is the controller-only baseline. Where available, $\pm$ denotes the success-rate standard deviation over five seeds. Bold marks the best central value.

Task	MemoryVLA	Long-DP	GMP	DreamZero	GEMA-DreamZero
Transform	21 ± 4.3	46	65 ± 5.1	12	$\mathbf{88} \pm 1.3$
Resume	27 ± 2.1	39	56 ± 3.2	25	$\mathbf{63} \pm 5.5$
FindImposter$_3$, shape	26	28	35	27	**44**
FindImposter$_3$, color	29	26	41	33	**50**
FindImposter$_9$, shape	20	12	27	10	**32**
FindImposter$_9$, color	22	12	**33**	18	25
ChainOfColors$_3$	18	24	37	5	**54**
ChainOfColors$_7$	12	10	22	0	**33**
GatherAndRecall	75	63	82	53	**85**
Mean	27.8	28.9	44.2	20.3	**52.7**

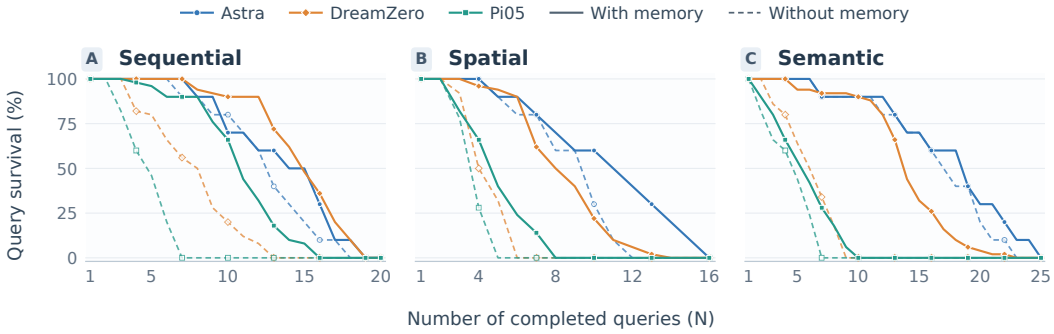


Figure 2: **Con-Q evaluation results.** The main Astra configuration is shown across the three memory forms. Solid/dashed lines indicate with/without memory; Pi05 uses semantic memory. Appendix E.2 gives reporting details.

Con-Q Evaluation Results. Figure 2 reports the main Astra configuration, comparing query survival with and without memory across the three settings and downstream policy families. At $N = 10$, DreamZero's sequential-memory survival increases from 20% to 90%. Pi05 uses semantic conditioning. The Opus configuration is evaluated in Appendix D.3.

5 Conclusion

We presented GEMA, a modular framework that connects past experience to robot action through explicit memory references. An MLLM interprets historical evidence when generation is needed, a downstream policy executes the resulting guidance, and a task agent incorporates observed outcomes into subsequent subtask decisions. Task-Based and Con-Q results support this interface for target construction, reuse across subtasks, and sustained execution. Crowded comparisons and longer sequences remain challenging. Future work will extend the framework to cross-session retention and autonomous memory scheduling.

References

[1] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. Jauregui Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as I can, not as I say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

[2] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022. URL https://arxiv.org/abs/2207.05608.

[3] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi:10.1016/S0004-3702(98)00023-X.

[4] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2025. URL https://arxiv.org/abs/2508.19236. Revised January 2026; ICLR 2026.

[5] Y. Gao, J. J. Liu, S. Li, and S. Song. Gated memory policy: In-context memorization and adaptation, 2026. URL https://arxiv.org/abs/2604.18933.

[6] H. Shi, W. Li, B. Xie, Y. Wang, R. Zhou, T. Wang, X. Zhang, P. Luo, and G. Huang. MemoryVLA++: Temporal modeling via memory and imagination in vision-language-action models, 2026. URL https://arxiv.org/abs/2606.09827.

[7] D. L. Schacter and D. R. Addis. The cognitive neuroscience of constructive memory: remembering the past and imagining the future. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):773–786, 2007. doi:10.1098/rstb.2007.2087.

[8] E. Spens and N. Burgess. A generative model of memory construction and consolidation. *Nature Human Behaviour*, 8(3):526–543, 2024. doi:10.1038/s41562-023-01799-z.

[9] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2303.15343.

[10] S. Ye, Y. Ge, K. Zheng, S. Gao, S. Yu, G. Kurian, S. Indupuru, Y. L. Tan, C. Zhu, J. Xiang, A. Malik, K. Lee, W. Liang, N. Ranawaka, J. Gu, Y. Xu, G. Wang, F. Hu, A. Narayan, J. Bjorck, J. Wang, G. Kim, D. Niu, R. Zheng, Y. Xie, J. Wu, Q. Wang, R. Julian, D. Xu, Y. Du, Y. Chebotar, S. Reed, J. Kautz, Y. Zhu, L. J. Fan, and J. Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[11] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9cLPurIZMj. Also arXiv:2502.10550.

[12] H. Nguyen, B. Daley, X. Song, C. Amato, and R. Platt. Belief-grounded networks for accelerated robot learning under partial observability. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1640–1653. PMLR, 2021. URL https://proceedings.mlr.press/v155/nguyen21a.html.

[13] H. Fang, M. Grotz, W. Pumacay, Y. R. Wang, D. Fox, R. Krishna, and J. Duan. SAM2Act: Integrating visual foundation model with a memory architecture for robotic manipulation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15925–15942. PMLR, 2025. URL https://proceedings.mlr.press/v267/fang25c.html.

[14] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.10639.

[15] Y. Hu, J. Zhang, Y. Luo, Y. Guo, X. Chen, X. Sun, K. Feng, Q. Lu, S. Chen, Y. Zhang, W. Li, and J. Chen. BagelVLA: Enhancing long-horizon manipulation via interleaved vision-language-action generation, 2026. URL https://arxiv.org/abs/2602.09849.

[16] D. R. Addis, A. T. Wong, and D. L. Schacter. Remembering the past and imagining the future: common and distinct neural substrates during event construction and elaboration. *Neuropsychologia*, 45(7):1363–1377, 2007. doi:10.1016/j.neuropsychologia.2006.10.016.

[17] C. Deng, D. Zhu, K. Li, C. Gou, F. Li, Z. Wang, S. Zhong, W. Yu, X. Nie, Z. Song, G. Shi, and H. Fan. Emerging properties in unified multimodal pretraining, 2025. URL https://arxiv.org/abs/2505.14683.

[18] E. Cherepanov, N. Kachaev, A. K. Kovalev, and A. I. Panov. MIKASA-Robo-VLA documentation. Project website and online task documentation, 2026. URL https://mikasarobo.github.io/. Online documentation. Accessed 2026-09-30.

A Related Work

Memory under partial observability. Belief-state methods summarize history for decision making under incomplete observations [3, 12]. Recent manipulation policies incorporate memory into learned control: SAM2Act+ uses a memory bank and attention [13]; MemoryVLA retrieves and consolidates perceptual and cognitive tokens [4]; GMP learns a memory gate and historical cross-attention [5]; and MemoryVLA++ augments memory with imagined latent states [6]. These systems emphasize representations used within action prediction. GEMA instead exposes task-relevant history through text and images. Recent observations reside in a subtask-local buffer, older information is retained in MLLM context, and references communicate the requested interpretation to a separate controller.

Agentic and generative robot control. SayCan grounds language-model decisions in executable skills [1], while Inner Monologue incorporates environment feedback into planning [2]. Visual generation provides another interface to control: SuSIE predicts image subgoals [14], BagelVLA interleaves language, visual prediction, and actions [15], and WAMs jointly model future video and actions [10]. GEMA contributes a memory interface to this setting. Its references depend on retained observations as well as the current request, allowing control to use historical information absent from the current scene.

Constructive memory and multimodal models. Constructive accounts link episodic recollection and imagination through the reconstruction of experiential details [7, 16]; generative models offer a computational account of memory construction and consolidation [8]. Unified MLLMs such as BAGEL combine visual understanding and generation [17]. GEMA uses these capabilities to interpret robot experience and express task-specific guidance through a common reference interface. A pretrained MLLM serves as the *memory foundation model*, combining multimodal interpretation and reference generation with history retention and robot control.

B Method and Implementation Details

GEMA connects past experience to action through an explicit memory reference. A task agent selects the active subtask; a memory component retrieves relevant evidence and, when necessary, uses an MLLM to express it as textual and visual guidance. A reference-conditioned WAM executes the subtask from this guidance and current observations. Execution feedback then informs the next subtask and becomes evidence for subsequent memory queries (Figure 1). We first define the information available under partial observability, then describe how references are constructed, used for control, and coordinated within this loop.

B.1 Problem Setup

Consider a robot pursuing a manipulation goal G. At control time t, it receives an image o_t and proprioceptive state q_t, and predicts an action chunk $a_{t:t+h-1}$ of length h. The current observation may omit information needed to complete G: distinct interaction histories can produce similar current views while requiring different actions [3]. We assume that the required historical information was observed earlier in the episode. The robot must recover it from recorded experience, without access to the hidden environment state.

Agent $\mathcal{A}$ maintains a queue Q_k of pending subtasks. At round k, beginning at control time t_k, it selects request g_k. An execution round spans selection and completion of one active request; the reference is constructed at the round's start and reused across its action chunks. A memory query is the internal retrieval-and-generation call, while a Con-Q query is an instruction issued by the benchmark. The history available at the start of a round can include observations collected during the task's preceding observation phase. Two stores provide historical context: a history buffer HB_k holding the most recent subtask history, including the observed prefix when that subtask is active, and retained MLLM context MC_k preserving selected information from earlier subtasks. The buffer

holds at most 250 frames; longer subtasks update MLLM memory. Implementation details appear in Appendix B.5.

We group memory demands by their historical dependency and the operation required. *Short-term* tasks recall an observed fact or sequence from the active subtask. *Short-term reasoning* tasks additionally transform or compare that evidence. *Long-short-term* tasks depend on information from a preceding subtask together with recent observations. Appendix C.3 gives the formal definitions and task mapping.

B.2 Task-Conditioned Memory References

Retrieving an earlier observation establishes what the robot previously saw, but does not always specify what it should do now. A request may require restoring an observed position, identifying a change, or placing an object relative to a previous arrangement. We define *generative memory* as constructing a task-conditioned reference from retained experience. It expresses the historical information needed for the current request. When the retrieved observation already supplies that information, GEMA can use it directly.

At the start of each memory-conditioned execution round, the request g_k and current image o_{t_k} rank buffered evidence using SigLIP text–image and image–image similarity [9]. The resulting evidence is $e_k = \mathrm{Retrieve}(\mathsf{HB}_k, o_{t_k}, g_k)$. This selector is separate from the controller. The reference $r_k = (r_k^{\mathrm{text}}, r_k^{\mathrm{vis}})$ contains textual guidance and a visual depiction of the relevant object or configuration. We represent retrieved evidence in the same format: recorded visual evidence with any accompanying text, leaving the text component empty when none is available. Reference construction follows three paths:

$$r_k = \begin{cases} e_k, & \text{direct recall,} \\ \mathcal{M}_\phi(e_k, o_{t_k}, g_k), & \text{local,} \\ \mathcal{M}_\phi(\mathsf{MC}_k, e_k, o_{t_k}, g_k), & \text{cross-subtask,} \end{cases} \tag{3}$$

where $\mathcal{M}_\phi$ is the MLLM memory model. The direct-retrieval path reuses evidence without generation, with g_k supplying instruction conditioning separately. All tasks in the present evaluation use the generation paths to construct textual and visual guidance from the evidence.

The local generation path serves both sequence recall and short-term reasoning. ChainOfColors asks the model to express an observed order; Transform asks it to construct a target offset from an earlier position; FindImposter requires comparing earlier and current objects. The cross-subtask path additionally supplies MC_k, allowing Resume to recover an earlier object–drawer association after intervening activity. Thus, the requested operation determines how evidence is interpreted, while its subtask origin determines which memory stores are needed.

The MLLM serves as a *memory foundation model*: pretrained multimodal understanding and generation support these requests through a shared interface across task families. Episode-specific facts come from the supplied evidence and context. Task-Based experiments use `GPT-5.6-Sol`; Con-Q experiments use Astra or Opus as described in Appendix D.3.

B.3 Downstream Policy Variants

GEMA-Astra: agentic policy. The Astra controller receives current RGB observations and measured robot state, together with generated textual and visual references. Its conversation and state are separate from the upstream memory generator. Predicted action segments are executed before new observations are used to assess progress.

GEMA-Pi05: VLA policy. The evaluated Pi05 interface receives current main/wrist observations, measured robot state, and an instruction prompt augmented with memory-derived semantic text. This variant supplies memory through the policy's language interface.

Algorithm 1 GEMA: memory references and execution feedback

Input: goal G, buffer HB, retained context MC, agent $\mathcal{A}$, memory model $\mathcal{M}_\phi$, and WAM π_θ.
Observe (o_0, q_0); initialize HB_0, $t \leftarrow 0$, $k \leftarrow 0$.
$Q_0 \leftarrow \mathcal{A}.\,\mathrm{Plan}(G, o_0)$.
while $Q_k \neq \varnothing$ and budget remains **do**
  Set $t_k \leftarrow t$ and $r_k \leftarrow \varnothing$; select $g_k \leftarrow \mathcal{A}.\,\mathrm{Select}(G, Q_k, o_t)$.
  if the selected request prescribes a memory query **then**
    Retrieve e_k from HB_k using (o_t, g_k); construct r_k by Eq. 3.
  end if
  repeat
    Predict $a_{t:t+h-1}$ using Eq. 4; execute the prescribed action segment.
    Observe $(o_{t'}, q_{t'})$; update HB_k with execution evidence; $t \leftarrow t'$.
    $b \leftarrow \mathcal{A}.\,\mathrm{Assess}(G, Q_k, g_k, o_t)$.
  until $b \neq$ `continue` or budget exhausted
  if $b =$ `done` or budget exhausted **then stop**.
  Archive selected history in MC_{k+1} before refreshing HB_{k+1}.
  Complete g_k, update Q_{k+1}; $k \leftarrow k + 1$.
end while

GEMA-DreamZero: WAM policy. A reference describes what matters for the subtask; it may depict an earlier state or a desired configuration that differs from the current scene. The controller therefore receives both reference and current sensory inputs:

$$a_{t:t+h-1} \sim \pi_\theta(\cdot \mid o_t, q_t, g_k, r_k), \tag{4}$$

where π_θ is a WAM instantiated with the DreamZero backbone [10]. The request and reference remain fixed within an execution round, while each policy call receives updated observations and proprioception. The same historical guidance supports successive action chunks throughout the round.

The current controller uses first/last-frame-to-video (FL2V) goal conditioning on a Wan2.2-TI2V-5B backbone within DreamZero. The first/current RGB image and the generated goal image are encoded independently by the frozen video VAE. The pretrained patch embedding maps the endpoint latents to spatial tokens, augmented with spatial positions and distinct trainable first/goal role embeddings. Endpoint tokens are concatenated with the embedded textual context and exposed through cross-attention to every video/action block, including the earliest action block.

Textual references augment the instruction, while the visual reference is passed explicitly as the goal image. Current observations and measured robot state ground action prediction. The reference remains fixed within an execution round, and the autoregressive state is reset when a new round begins. The implementation details below specify training targets and the inference interface. The Task-Based results evaluate this FL2V goal-conditioned controller.

B.4 Agentic Execution and Feedback

The agent coordinates memory and control as the overall goal progresses (Algorithm 1). It selects the active request from its pending queue and invokes memory at stages prescribed by the task protocol. Stages without a memory query use $r_k = \varnothing$.

After each action segment, new observations update the buffer and return to the agent. A *continue* decision retains (g_k, r_k) and calls the WAM again; *advance* completes the request and selects the next subtask; *done* terminates execution. Before the history buffer is refreshed, selected information is archived in retained MLLM context. Newly observed outcomes thereby become evidence for later queries, while older subtask information remains accessible through MC_k.

References guide action, and observed execution determines subsequent progress. Appendix B.6 details the loop, and Appendix C.4 identifies the evaluated execution settings.

B.5 Implementation Details

Evidence selection and retention. The buffer contains at most 250 frames from the most recent subtask. The current observation and request query this buffer using image–image and text–image similarity from `google/siglip-base-patch16-224` [9]. Highest-relevance buffered evidence forms e_k. Before the buffer is refreshed, selected information is retained in multimodal MLLM context. Subtasks longer than the frame budget also update MLLM memory. The buffer and retained context together provide recent visual evidence and information from earlier subtasks.

The evaluated tasks use the MLLM to generate textual and visual guidance, including sequence recall in ChainOfColors. Task-Based experiments use `GPT-5.6-Sol`, with GPT-6 and BAGEL compared in Appendix D. Con-Q uses the Astra and Opus configurations. Episode-specific facts are supplied through historical evidence and retained context.

B.6 FL2V Goal Conditioning

The current implementation uses DreamZero with the Wan2.2-TI2V-5B backbone. Let I^{first} denote the first/current RGB observation at the start of an execution round and $I^{\text{goal}} = r_k^{\text{vis}}$ the requested visual goal. A frozen VAE E independently encodes each endpoint, and the pretrained patch embedding P produces spatial tokens:

$$\begin{aligned} Z^u &= P(E(I^u)) + S + R^u, \quad u \in \{\text{first}, \text{goal}\}, \\ C_k &= [T(g_k, r_k^{\text{text}}); Z^{\text{first}}; Z^{\text{goal}}], \end{aligned} \tag{5}$$

where S is a spatial sinusoidal embedding, R^u is a trainable endpoint-role embedding, and T is embedded text conditioning. Every video/action transformer block receives C_k through cross-attention. Endpoint visibility therefore does not depend on the temporal ordering of the causal video tokens. In the default 160×320 packed-image configuration, each endpoint produces 50 spatial tokens. The video/action flow-matching objectives and teacher-forcing layout are retained. The adaptation trains the diffusion transformer and action/state modules while freezing T5 and the VAE.

Goal construction for training and inference. During training, the default goal is the last RGB frame of the sampled video window. A dataset may instead supply an explicit goal image. Encoding the goal independently prevents its VAE representation from containing intervening ground-truth video frames. During inference, the generated memory reference supplies the explicit `goal_images` input. The policy model expects packed uint8 RGB images with shape $[B, H, W, 3]$, using the same camera order and packing convention as observations. Text conditioning combines the active instruction and generated reference text. The goal is shared by both text-guidance branches. Each new execution round resets the autoregressive state. FL2V supplies endpoint features through cross-attention using the checkpoint's camera, state, and action conventions.

Subtask coordination. Algorithm 1 connects goal decomposition, reference construction, and action execution. The agent uses observed outcomes to continue the current request, advance to another subtask, or terminate. A continue decision keeps the request and reference fixed while updating current observations. An advance decision archives selected history before refreshing the buffer and selecting the next request. Later memory queries use the resulting execution history and archived context.

C Benchmark Definitions and Evaluation Protocols

Evaluation structure. Section 2 distinguishes Task-Based evaluation from Con-Q evaluation. The Task-Based suite includes Transform, Resume, and seven MIKASA variants. Transform and Resume test historical target transformation and recovery across intervening subtasks, while MIKASA covers comparison, sequence reproduction, and retained cues; Appendix E expands their results. The Con-Q suite consists of three benchmarks: a LIBERO sequential-memory stream, a LIBERO spatial-memory

Table 2: **Benchmark organization.** Protocol families describe how requests are issued; memory forms describe the primary information dependency. Con-Q benchmarks can require several forms simultaneously.

Protocol	Benchmark	Primary memory requirement
Task-Based	Transform	Transform a remembered position into a new target.
	Resume	Recover a stored object–drawer association after another task.
	MIKASA (7 variants)	Compare past/current observations, reproduce sequences, and retain cues.
Con-Q	LIBERO sequential	Preserve event order and suspended intentions.
	LIBERO spatial	Recover historical locations and drawer assignments.
	RoboCasa semantic	Retain and update historical order contents.

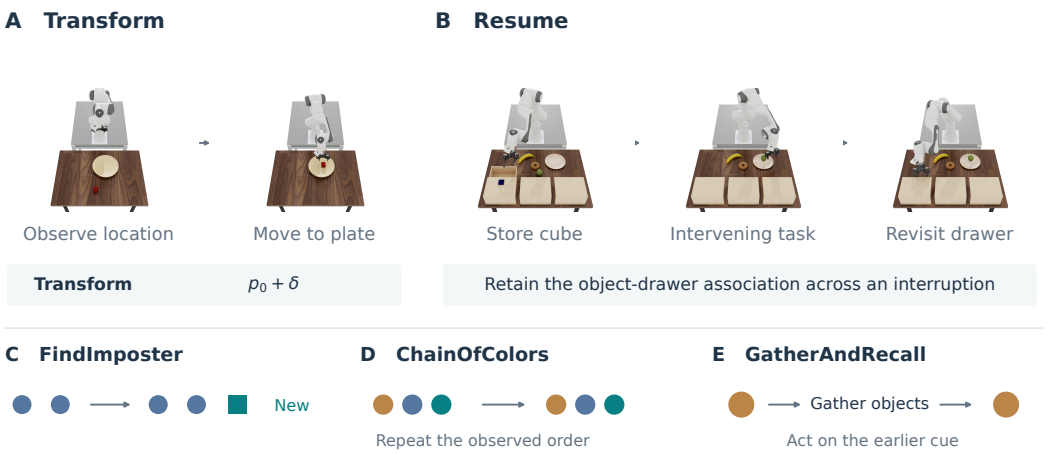


Figure 3: **Tasks and memory requirements.** Transform constructs an offset target; Resume retains a drawer association across an intervening subtask. MIKASA evaluates novelty comparison, sequence recall, and cue retention across object gathering. The schematics illustrate each task's stages.

stream, and a RoboCasa semantic-memory stream. We construct these query-stream protocols using LIBERO and RoboCasa environments.

C.1 Task-Based Benchmarks

Manipulation benchmark. We evaluate two tasks with complementary memory demands (Figure 3). In *Transform*, the robot moves a red cube onto a plate and then requests placement 10 cm left, right, in front of, or behind its original position. The required target must therefore be constructed from a remembered position and a new spatial instruction. In *Resume*, the robot stores a blue cube in one of three identical opaque drawers, completes an unrelated object-to-plate subtask, and approaches the drawer containing the cube. Success is defined by approaching the correct drawer. These tasks instantiate short-term reasoning and long-short-term memory, respectively.

MIKASA. Seven variants broaden the evaluation beyond spatial manipulation [11, 18]. *FindImposter* requires identifying an object absent from an earlier set, with shape or color variation and three or nine candidates. *ChainOfColors* requires reproducing a previously demonstrated sequence of three or seven colors after a waiting phase. *GatherAndRecall* briefly presents a lamp color, inserts an object-gathering subtask, and then requests the matching button. These correspond to short-term reasoning, short-term recall, and long-short-term memory. This classification follows the historical evidence required by the active request.

Systems and protocol. We compare MemoryVLA [4], Gated Memory Policy (GMP) and its long-history Diffusion Policy baseline (Long-DP) [5], and DreamZero fine-tuned on raw demonstrations. GEMA-DreamZero uses `GPT-5.6-Sol` for generation on the evaluated tasks, including ChainOf-Colors. The controller uses FL2V goal conditioning. Training uses 200 and 300 demonstrations for Transform and Resume, and 250 episodes per MIKASA variant. Manipulation evaluation uses 20 episodes for each of five backgrounds; MIKASA uses 100 episodes per variant. Table captions state reporting conventions; Appendix C.4 documents execution settings.

C.2 Con-Q Benchmarks

Figure 4 summarizes the three scene settings, object sets, and query types. Sequential memory preserves event order and suspended tasks; spatial memory tracks historical locations across two cabinets; semantic memory retains and restores order contents.

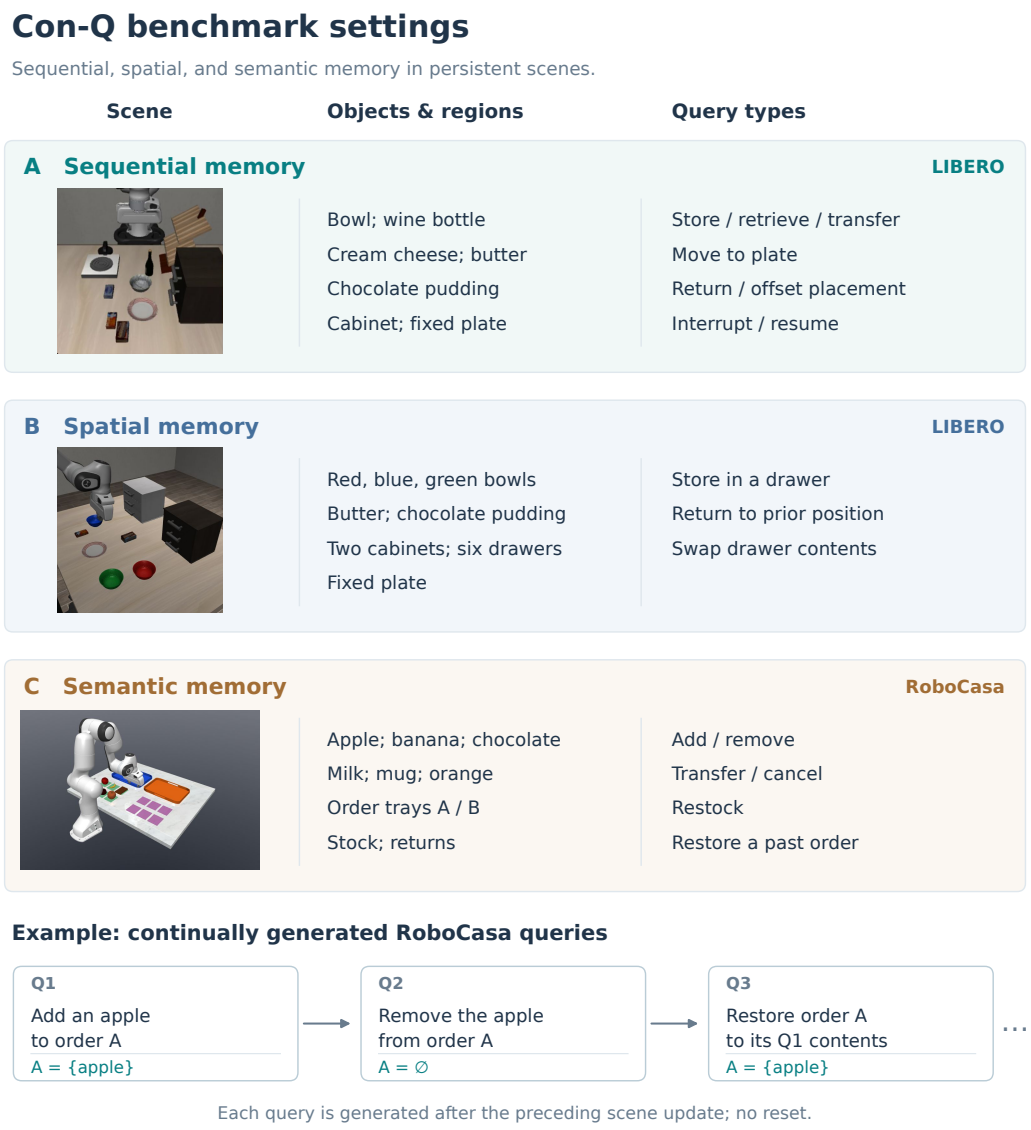


Figure 4: **Con-Q settings and continual query generation.** One recorded view per setting is paired with its objects and query types. The bottom row gives a schematic RoboCasa example: add an apple to order A, remove it, then restore the order to its contents after Q1. Each query follows the preceding scene update, with no scene reset.

Only the current instruction and allowed observation history are public; future queries and evaluator state remain private. Each rollout ends at 15 minutes of simulated time or the first task failure. Success is determined from physical execution, including required support, containment, release, and preservation of unaffected objects or orders. Query-survival at N is the fraction of all started rollouts completing at least N queries. Model-inference wall time is recorded separately.

Table 3: **Task-family mapping.** The family is determined by the historical dependency and requested operation.

Task	Evidence and requested use	Family
ChainOfColors	Express and reproduce a color sequence after a waiting phase.	Short-term
Transform	Derive an offset target from an earlier object position.	Short-term reasoning
FindImposter	Compare current objects with a previously observed set.	Short-term reasoning
Resume	Recover an object–drawer association across an intervening subtask.	Long-short-term
GatherAndRecall	Retain a lamp cue across gathering, then select a matching button.	Long-short-term

C.3 Memory Task Families

Let $H^{(j)}$ denote the observations, executed actions, and instructions associated with subtask j. At execution round k, the history buffer HB_k contains the most recent subtask history available within its capacity, including the observed prefix when that subtask is active. Retained MLLM context MC_k preserves selected information beyond this buffer, including earlier subtasks. Retrieval uses the active request and current observation:

$$e_k = \mathrm{Retrieve}(\mathsf{HB}_k, o_{t_k}, g_k). \tag{6}$$

The retrieved evidence is represented in the same text–visual format as r_k, with an empty text component when no accompanying text is available.

A *short-term* task recalls an observed fact or sequence from the active subtask. ChainOfColors uses the MLLM to express a remembered sequence. A *short-term reasoning* task requires a new relation, comparison, or target to be inferred from that evidence. A *long-short-term* task additionally depends on information acquired before the active subtask. These families describe the operation applied to historical evidence and its subtask origin.

Table 3 maps the evaluated tasks into these families. MIKASA's descriptions of negative, sequential, procedural, and prospective memory characterize the content or function of memory [11, 18]; our mapping characterizes how an active subtask depends on historical information. Its Short/Medium/Long environment splits describe episode length, complementing our dependency-based task families.

C.4 Evaluation Settings

Data and reporting. The manipulation tasks use 200 training episodes for Transform and 300 for Resume. Each MIKASA variant uses 250 training episodes. Manipulation evaluation uses 20 episodes for each of five backgrounds, totaling 100 episodes per task; MIKASA uses 100 evaluation episodes per variant. Reported $\pm$ values and error bars denote the standard deviation of success rates over five random seeds. Dispersion is shown only where supplied.

The model-level image-generation study uses 20 episodes per task and assesses references separately from downstream execution.

Con-Q configurations. Astra and Opus each serve as the reference generator across downstream policy families and as the controller for the agentic policy. For each configuration and benchmark, evaluation uses 10 agentic rollouts and 50 rollouts each for DreamZero and Pi05, for both memory conditions.

D Ablation and Sensitivity Studies

We group the reference-generator comparison, training-data sensitivity, supplementary Opus configuration, and inference efficiency here. The generator study varies the reference model; the data

Table 4: **Reference-model configurations.** Manipulation success (%); 100 evaluation episodes per task. The main configuration uses GPT-5.6-Sol for reference generation.

Configuration	Transform	Resume
GPT-5.6-Sol (main)	88	63
GPT-6	**94**	**67**
BAGEL	71	59

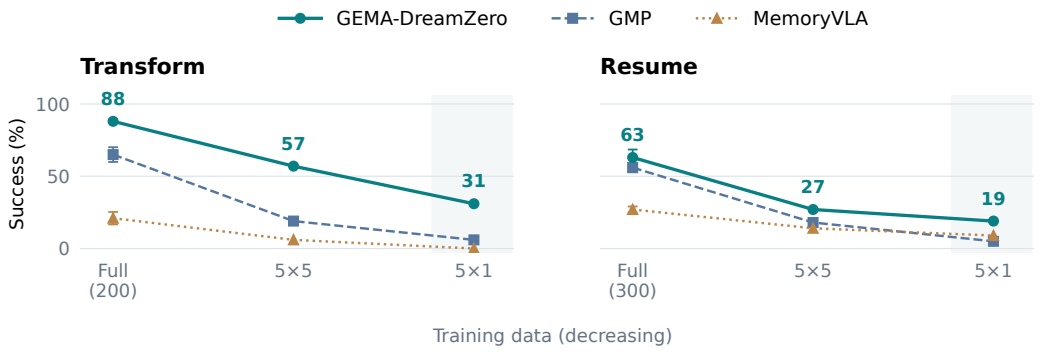


Figure 5: **Reduced-data performance.** Full uses 200/300 demonstrations for Transform/Resume. Error bars show five-seed standard deviations where available.

study varies the demonstration budget. Together, these studies characterize reference-model choice, demonstration budget, and execution efficiency.

D.1 Reference-Generator Comparison

The GPT-6 configuration reaches 94% on Transform and 67% on Resume, compared with 88% and 63% for the main configuration; BAGEL reaches 71% and 59% (Table 4).

D.2 Training-Data Sensitivity

Figure 5 reports few-shot training across Full, 5×5, and 5×1 conditions. The first factor denotes five backgrounds and the second denotes demonstrations per background: five demonstrations yield 25 total, and one yields 5 total. At 5×1, GEMA-DreamZero reaches 31% on Transform and 19% on Resume, compared with GMP's 6% and 5%. With five demonstrations, GEMA-DreamZero exceeds GMP by 25 percentage points on Transform and 14 on Resume.

Table 5: **Few-shot training: success (%).** Full uses 200 and 300 training episodes for Transform and Resume. The 5×5 and 5×1 conditions use five backgrounds with five or one training demonstrations per background, totaling 25 or 5 demonstrations. Evaluation uses 100 episodes per task. Where supplied, $\pm$ is the success-rate standard deviation over five random seeds.

Task	Method	Full	5×5	5×1
Transform	GEMA-DreamZero	88 ± 1.3	57	31
	GMP	65 ± 5.1	19	6
	MemoryVLA	21 ± 4.3	6	0
Resume	GEMA-DreamZero	63 ± 5.5	27	19
	GMP	56 ± 3.2	18	5
	MemoryVLA	27 ± 2.1	14	9

D.3 Opus Configuration

We use Astra for reference generation and agentic control in the main Con-Q results. The Opus configuration replaces the reference generator in all variants and also replaces the controller in the agentic variant (Figure 6). DreamZero and Pi05 retain their downstream policies. Each benchmark uses 10 agentic rollouts and 50 rollouts each for DreamZero and Pi05 under both memory conditions. At $N = 10$, sequential-memory survival with memory is 80% for the agentic policy and 88% for DreamZero, compared with 70% and 90% in the main Astra configuration.

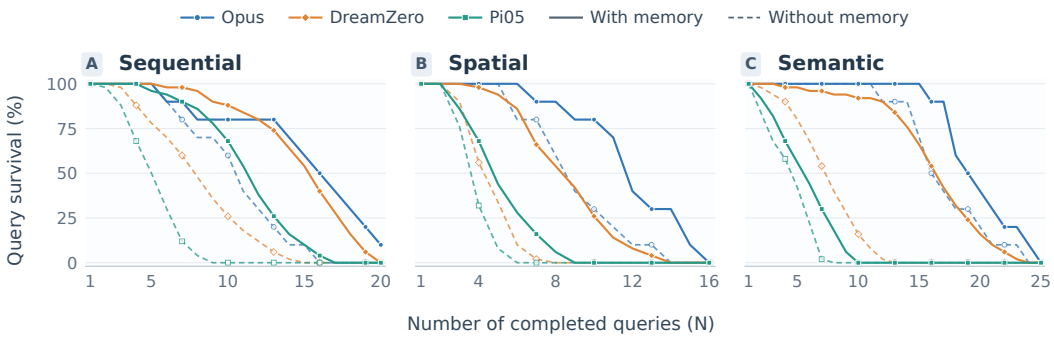


Figure 6: **Opus configuration on Con-Q.** Opus provides reference generation and agentic control. The three panels show sequential, spatial, and semantic query survival. Colors identify Opus, DreamZero, and Pi05; solid/dashed lines indicate with/without memory. Pi05 uses semantic conditioning. The corresponding main Astra configuration appears in Figure 2.

D.4 Token Usage and Inference Speed

Table 6 compares the Astra, Opus, and available Gemini configurations across the three Con-Q benchmarks. We report token usage per query and model-inference wall time per task.

Table 6: **Con-Q token usage and inference time.** Each cell reports token usage (thousands per query) / inference time (seconds per task). Lower is better for both. A dash denotes an unreported configuration.

Downstream policy	Model	Sequential	Spatial	Semantic
DreamZero	Astra	68 / 135	88 / 168	76 / 148
	Opus	105 / 38	140 / 48	120 / 42
	Gemini	—	—	74 / 480
Pi05	Astra	56 / 112	72 / 140	64 / 124
	Opus	88 / 30	116 / 39	100 / 34
	Gemini	—	—	62 / 420
Agentic	Astra	215 / 190	280 / 235	240 / 210
	Opus	310 / 55	410 / 70	350 / 62
	Gemini	—	—	235 / 650

Downstream policy. Under Astra, DreamZero uses 68–69% fewer tokens per query than the agentic policy across the three benchmarks; Pi05 uses 73–74% fewer. Both also have lower reported inference times. Pi05 reduces inference cost but exhibits rapid survival declines in the spatial and semantic settings (Figure 2).

Model configuration. Across the three downstream families and benchmarks, Opus uses 44–61% more tokens than Astra, but its reported inference time is 70–73% lower (a 3.4–$3.7\times$ speedup). Together with Figure 6, these measurements characterize the accuracy–efficiency tradeoff across tasks and downstream policies. Gemini is reported only for semantic memory: token usage is close to Astra (74 versus 76 thousand for DreamZero, 62 versus 64 for Pi05, and 235 versus 240 for the agentic policy), but inference takes 3.1–$3.4\times$ as long.

E Additional Results and Diagnostics

This section expands the Task-Based results and distinguishes reference-output correctness from robot execution. GEMA-DreamZero uses the DreamZero FL2V goal-conditioned controller. Generator and data-budget comparisons are grouped in Appendix D.

E.1 Task Success across Memory Demands

Overall performance. Table 1 reports the nine-task Task-Based evaluation, comprising Transform, Resume, and seven MIKASA variants. The unweighted mean across these nine tasks is 52.7% for GEMA-DreamZero and 44.2% for GMP. GEMA-DreamZero exceeds GMP on eight of nine tasks. Table 1 also reports seed standard deviations for Transform and Resume.

Short-term recall. ChainOfColors uses the MLLM to express a remembered sequence through the same interface. Success is 54% at length three and 33% at length seven, exceeding GMP by 17 and 11 points. The decline with sequence length indicates that longer temporal dependencies remain difficult despite the improvement over the baselines.

Reasoning from recent experience. Transform shows the largest manipulation gain over GMP: 88% versus 65%. The target position does not appear directly in the historical observation. The reference makes the requested transformation available to the controller before action prediction. Find-Imposter provides a complementary test of comparing past and current scenes. GEMA-DreamZero

Table 7: **Model-level memory and image-generation correctness (%).** Chain_3 and Chain_7 measure sequence-memory capability. Generated-image correctness uses 20 test episodes per task. The columns assess sequence recall and reference-image quality.

Model	Chain_3	Chain_7	Transform image	Resume image
GPT-6	**95**	**90**	**100**	**80**
GPT-5.6-Sol	65	55	90	**80**
BAGEL	15	12	75	75

improves over GMP by nine points on both three-candidate variants and five points on nine-candidate shape comparison, but falls eight points behind on nine-candidate color comparison. These results show gains on target construction and shape comparison, with lower success on crowded color comparison.

Retaining information across subtasks. Resume reaches 63%, seven points above GMP and 38 above the DreamZero baseline. The identical drawers conceal the earlier storage decision, and an intervening manipulation separates that decision from the final approach. GatherAndRecall similarly requires retaining a cue across a different activity; GEMA-DreamZero reaches 85%, compared with 82% for GMP. Together, these tasks test use of earlier-subtask information alongside the most recent observations.

E.2 Con-Q Results and Reporting

Figure 2 reports the three main Astra panels; Figure 6 reports the corresponding Opus configuration in Appendix D.3. Both cover sequential, spatial, and semantic memory. Each panel compares agentic, DreamZero, and Pi05 policies with and without memory; Pi05's memory input is semantic text. The main Con-Q configuration uses `GPT-6 Astra` for reference generation and agentic control. The Opus configuration replaces both roles with Opus; DreamZero and Pi05 retain their downstream policies. Each rollout ends at the first task failure or after 15 minutes of simulated time. Survival at N uses all started rollouts as the denominator, including earlier failures.

The evaluation curves cover $N = 1, \ldots, 20$ for sequential memory, $1, \ldots, 16$ for spatial memory, and $1, \ldots, 25$ for semantic memory. Each Con-Q configuration uses 10 rollouts per benchmark for the agentic policy and 50 each for DreamZero and Pi05, with and without memory.

E.3 Reference-Output Diagnostics

Separate reference-level measurements also distinguish generation from execution: GPT-6 attains 100% judged image correctness on Transform over 20 episodes, while downstream task success is 94%. Table 7 reports sequence-memory capability and generated-image correctness, with 20 image-generation test episodes per task.

F Qualitative Analysis

The following panels show retrieved observations and generated task references. Figure 8 separates a retrieved historical anchor from a generated spatial target. Figure 9 changes the query while retaining the same history, illustrating how different prior facts can become relevant. Figure 10 compares reference models and illustrates a spatial-target error. These examples illustrate task conditioning and reference fidelity.

What the memory reference expresses. Figure 7 holds the historical scene fixed while varying the spatial request. The generated target changes with that request, exposing the operation required from memory instead of merely displaying the retrieved scene. The panels below compare retrieved evidence and generated targets, different queries on shared history, and correct and incorrect refer-

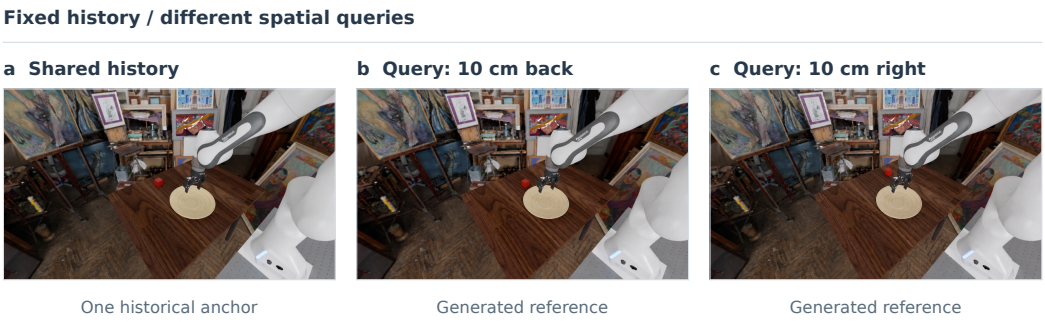


Figure 7: **One history, different targets.** Different spatial requests yield different references from shared history. Each generated image depicts the target specified by its query.

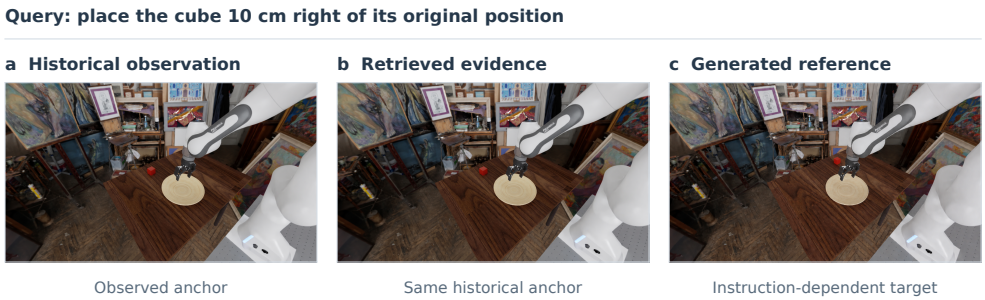


Figure 8: **Retrieved evidence and generated guidance.** The request places the cube 10 cm to the right of its original position. Retrieval preserves the observed position; generation depicts the requested target.

ences across models. Together, the panels illustrate how the reference interface expresses spatial transformations and historical associations for downstream control.



Figure 9: **Shared history, different memory queries.** The queries ask for the drawer that received the blue cube and the pre-grasp location of the object manipulated during the intermediate step. GPT-6 and GPT-5.6 outputs illustrate query-dependent references. The historical frame illustrates the shared scene context.

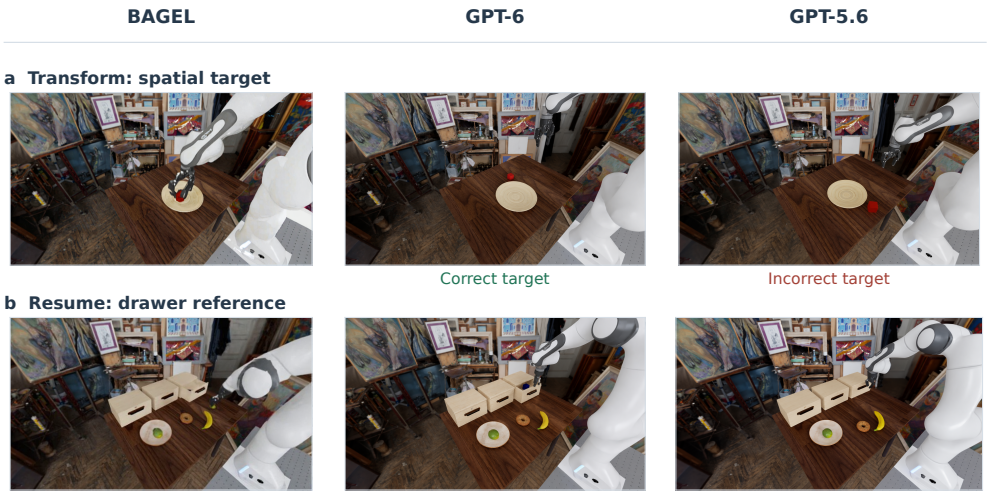


Figure 10: **Generator-dependent reference fidelity.** BAGEL, GPT-6, and GPT-5.6 outputs for spatial transformation and drawer memory. Annotations identify correct references and a BAGEL spatial-target error.