# SAM2ACT: INTEGRATING VISUAL FOUNDATION MODEL WITH A MEMORY ARCHITECTURE FOR ROBOTIC MANIPULATION

**Haoquan Fang**[1] **Markus Grotz**[1] **Wilbert Pumacay**[2] **Yi Ru Wang**[1]
**Dieter Fox**[*1,3] **Ranjay Krishna**[*1,4] **Jiafei Duan**[*1,4]
[1]University of Washington, [2]Universidad Católica San Pablo
[3]NVIDIA [4]Allen Institute for Artificial Intelligence

sam2act.github.io

## ABSTRACT

Robotic manipulation systems operating in diverse, dynamic environments must exhibit three critical abilities: multitask interaction, generalization to unseen scenarios, and spatial memory. While significant progress has been made in robotic manipulation, existing approaches often fall short in generalization to complex environmental variations and addressing memory-dependent tasks. To bridge this gap, we introduce **SAM2Act**, a multi-view robotic transformer-based policy that leverages multi-resolution upsampling with visual representations from large-scale foundation model. SAM2Act achieves a state-of-the-art average success rate of **86.8% across 18 tasks** in the RLBench benchmark, and demonstrates robust generalization on `The Colosseum` benchmark, with only a **4.3% performance gap** under diverse environmental perturbations. Building on this foundation, we propose **SAM2Act+**, a memory-based architecture inspired by SAM2, which incorporates a memory bank, an encoder, and an attention mechanism to enhance spatial memory. To address the need for evaluating memory-dependent tasks, we introduce `MemoryBench`, a novel benchmark designed to assess spatial memory and action recall in robotic manipulation. SAM2Act+ achieves **competitive performance on `MemoryBench`**, significantly outperforming existing approaches and pushing the boundaries of memory-based robotic systems.

## 1 INTRODUCTION

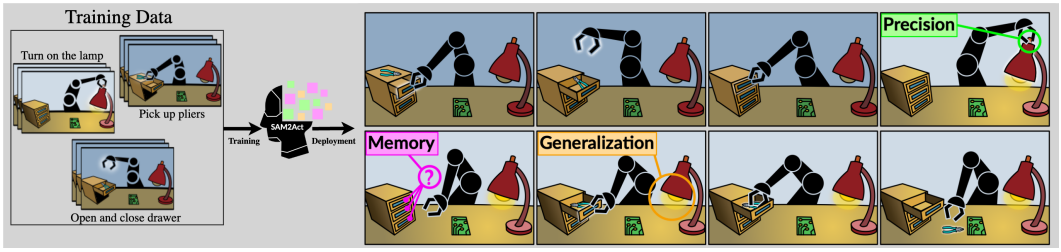


Figure 1: SAM2Act is a multi-view, language-conditioned behavior cloning policy trained with fewer demonstrations. Given a language instruction, it can execute high-precision tasks, such as turning the tiny knob on the lamp. It also generalizes to various environmental variations, such as changes in lighting conditions. Through further training with our proposed memory architecture, it now evolves into SAM2Act+, which is now capable of solving tasks that require implicit spatial memory—such as remembering where the robot previously stored the pliers, as depicted in the above figure.

The world in which we live is diverse and constantly changing, encompassing a wide variety of objects, scenes, and environmental conditions. Consider the seemingly simple task of following a

[*]Equal advising

recipe when cooking: we can seamlessly perform the action of picking it up and sprinkling it into the pan, recognize salt even if it comes in different types of container, and remember whether we have already added salt. Humans excel in such environments because they can interact with their surroundings to achieve specific goals, generalize to unseen scenarios, and retain knowledge from past experiences Smith & Gasser (2005). These abilities—multitask interaction, generalization, and memory—serve as guiding principles for developing robotic systems capable of operating in similarly complex environments.

Significant progress has been made in robotic manipulation through prior work. Early methods, such as the Transporter Network Zeng et al. (2021) and CLIPort Shridhar et al. (2022), demonstrated effective 2D action-centric manipulation but were limited in their ability to handle spatially complex tasks. More recent approaches, such as PerAct Shridhar et al. (2023) and RVT Goyal et al. (2023), have pushed toward 3D-based manipulation. PerAct employs a multitask transformer that interprets language commands and predicts keyframe poses, achieving strong results across a variety of tasks. RVT builds on this foundation by adopting a 2.5D representation, improving training efficiency and inference speed. Its successor, RVT-2, further enhances performance with a coarse-to-fine strategy, increasing precision for high-accuracy tasks. Despite these advances, important challenges remain, including improving multitask performance, enhancing generalization to novel environment configurations, and integrating memory mechanisms for tasks requiring episodic recall.

We introduce SAM2Act, a multi-view robotics transformer-based policy that enhances feature representation by integrating multi-resolution upsampling with visual embeddings from large-scale foundation models. Built on the RVT-2 multi-view transformer, SAM2Act achieves strong multitask success and generalization. Building on this foundation, we introduce SAM2Act+, which incorporates a memory-based architecture inspired by SAM2's approach. Using a memory bank, an encoder, and an attention mechanism, SAM2Act+ enables episodic recall to solve spatial memory-dependent manipulation tasks. We evaluate SAM2Act and SAM2Act+ using `MemoryBench`, a new benchmark suite that tests policies' spatial memory capabilities and the ability to retain and recall past actions. SAM2Act+ achieves competitive performance on `MemoryBench`, with an average accuracy of 94.3%, outperforming next highest baseline by a huge margin of 39.3%. Furthermore, we assess the generalization capabilities of SAM2Act on `The Colosseum` Pumacay et al. (2024), a benchmark designed to test robotic manipulation under various environmental perturbations. SAM2Act demonstrates robust performance on `The Colosseum` with an average decrease of 4.3% across all perturbations, highlighting its ability to generalize effectively in diverse and challenging scenarios. Lastly, our approach outperforms the baseline methods in real-world evaluations while exhibiting comparable generalization and spatial memory capabilities.

In summary, this work makes three key contributions. First, we introduce a **novel model formulation** that leverages visual foundation models to solve **high-precision, memory-dependent manipulation tasks**. Second, we propose `MemoryBench`, a evaluation benchmark for **assessing spatial memory in behavior cloning models**. Finally, we present **empirical results and insights** on the model's performance across both simulation and real-world tasks.

## 2 Related Work

### 2.1 3D-based Robotic Transformer for Manipulation

2D-based methods Zhao et al. (2023); Chi et al. (2023); Zeng et al. (2021); Brohan et al. (2022); Shridhar et al. (2022) are effective for simple pick-and-place tasks due to fast training, low hardware requirements, and minimal computational cost. However, they depend on pretrained image encoders and fail in tasks requiring high precision, robust spatial interaction, or resilience to environmental and camera variations Pumacay et al. (2024). Recent work addresses these limitations with 3D perception. Methods like PolarNet Chen et al. (2023), M2T2 Yuan et al. (2023), and Manipulate-Anything Duan et al. (2024) reconstruct point clouds, while C2F-ARM James & Abbeel (2022) and PerAct Shridhar et al. (2023) use voxel-based 3D representations. Act3D Gervet et al. (2023) and ChainedDiffuser Xian et al. (2023) adopt multi-scale 3D features. RVT Goyal et al. (2023) introduces 2.5D multi-view images for faster training, refined by RVT-2 Goyal et al. (2024) with a coarse-to-fine architecture for improved precision. Our work, SAM2Act, combines RVT-2's spatial reasoning with enhanced virtual images from the SAM2 visual encoder, achieving high precision and generalization across diverse tasks.

### 2.2 VISUAL REPRESENTATIONS FOR ROBOT LEARNING

Robotics research heavily relies on visual representations from computer vision to process high-dimensional inputs and improve policy learning. Visual representations are integrated into robot learning through pre-training Majumdar et al. (2023); Ma et al. (2022); Nair et al. (2022), co-training Laskin et al. (2020b); Yarats et al. (2021); Laskin et al. (2020a); Shang et al. (2024), or frozen encoders Shah & Kumar (2021); Wang et al. (2022a); Zhang et al. (2024), all of which effectively support policy training. These representations also enhance invariance, equivariance, and out-of-distribution generalization Wang et al. (2022b); Pumacay et al. (2024); Dasari et al. (2023). SAM-E Zhang et al. (2024) demonstrates the use of a pre-trained SAM encoder for robotic manipulation by leveraging image embeddings for policy learning. Expanding on this, our approach employs the SAM2 visual encoder to generate image embeddings for robotic transformers and utilizes its multi-resolution features to improve convex upsampling for next-action prediction.

### 2.3 MEMORY IN ROBOTICS

Memory is a fundamental component of human cognition, and equipping generalist robotic agents with episodic and semantic memory is crucial for enabling them to perform complex tasks effectively Jockel et al. (2008). Early research on memory in robotics primarily addressed navigation tasks, relying on semantic maps that were often constrained in scope Henry et al. (2012); Bowman et al. (2017); Chaplot et al. (2020). Other work explicitly model the memory and its representation for a robot cognitive architecture Peller-Konrad et al. (2023). Recent advancements leverage representations derived from vision-language models (VLMs) and Large Vision Models (LVMs), utilizing voxel maps or neural feature fields to encode, store, and retrieve information Huang et al. (2024; 2023); Duan et al. (2024); Liu et al. (2024). Alternative methods represent semantic memory for manipulation tasks using Gaussian splats to encode spatial information Kerbl et al. (2023); Shorinwa et al. (2024). In contrast, our approach draws inspiration from the framework of Partially Observable Markov Decision Processes (POMDPs) Lauri et al. (2022), incorporating memory directly into the training process. By integrating spatial memory from past actions into the agent's belief state, we enhance the robustness and adaptability of learned policies.

## 3 MEMORYBENCH: A MEMORY BENCHMARK FOR ROBOTIC MANIPULATION

We introduce `MemoryBench`, a benchmark designed to systematically evaluate the spatial memory capabilities of robotic manipulation policies. In subsection 3.1, we begin by outlining the logic and rules behind task design. We will then describe the tasks we have developed in subsection 3.2.

### 3.1 TASK DESIGN

Unlike standard RLBench tasks James et al. (2020), many of which involve long-horizon scenarios, our tasks are specifically designed to require spatial memory. Without such memory, the agent would be forced to rely on random actions. To create these tasks, we intentionally violate the Markov assumption, which states that in a Markov Decision Process (MDP), the next observation depends solely on the current observation and action:

$$P\big(o_{t+1} \mid o_1, a_1, \ldots, o_t, a_t\big) \;=\; P\big(o_{t+1} \mid o_t, a_t\big).$$

This assumption implies that knowing only $o_t$ and $a_t$ is sufficient to predict $o_{t+1}$. However, in our tasks, we design scenarios where two distinct action histories lead to the same observation $o_t$, but require different subsequent actions. This forces the agent to recall which action history led to $o_t$ to perform the correct next action. Furthermore, we standardized the language instructions to prevent unintentional leakage of spatial information that could aid the model in memory-based tasks. These principles guided the development of our spatial memory-based tasks.

### 3.2 SPATIAL MEMORY-BASED TASKS

`MemoryBench` extends the RLBench simulator to provide scripted demonstrations for three spatial memory tasks: `reopen_drawer`, `put_block_back`, and `rearrange_block`. Each task is designed to evaluate a specific aspect of spatial memory and adheres to the principles outlined in

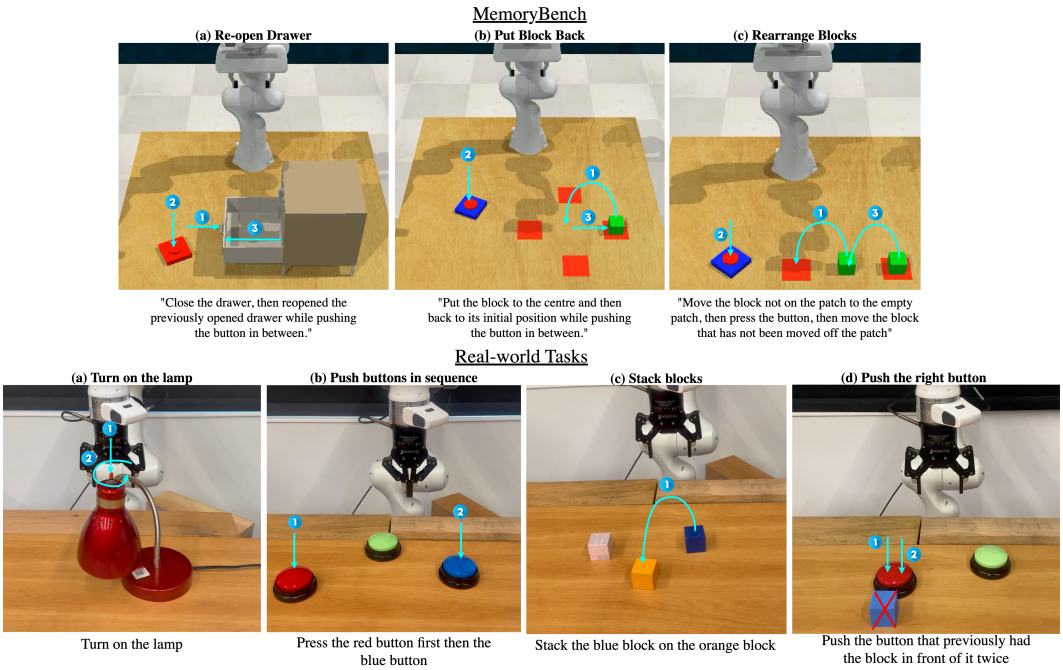


Figure 2: **Simulation and Real Tasks.** We demonstrate the effectiveness of SAM2Act+ in solving memory-based tasks by evaluating it against baselines on the three benchmark memory tasks (shown at the top). Additionally, we validate our approach using a Franka Panda robot on four real-world tasks (shown at the bottom), including tests under out-of-distribution perturbations.

Section 3.1. To introduce complexity, these tasks include two to four variations and additional steps—such as pressing a button mid-sequence—that disrupt the Markov property. This forces the agent to rely on memory rather than solely on immediate observations.

The `reopen_drawer` task evaluates the agent's ability to recall 3D spatial information along the z-axis. Initially, one of three drawers (top, middle, or bottom) is open. The agent must close the open drawer, press a button on the table, and then reopen the same drawer. After the button is pressed, all drawers are closed, and the scene becomes visually indistinguishable, requiring the agent to use memory to identify the correct drawer. This task tests the agent's ability to recall spatial states over a temporal sequence. The `put_block_back` task tests the agent's ability to remember 2D spatial information on the x-y plane. Four red patches are placed on a table, with a block initially positioned on one of them. The agent should move the block to the center of the patches, press a button, and return the block to its original position. The agent must rely on its memory of the block's initial location to succeed, demonstrating its capability to encode and retrieve 2D spatial information.

The `rearrange_block` task evaluates the agent's ability to perform backward reasoning by recalling and reversing prior actions. Initially, one block is placed on one of two red patches, while the other patch remains empty. A second block is positioned at the center of both patches. The agent must move the second block to the empty patch, press a button, and then relocate the first block off its patch. Successfully completing this task requires the agent to determine which block to move without having interacted with the correct one in previous actions, thereby testing its capacity for backward spatial memory reasoning. These tasks collectively evaluate both forward and backward spatial reasoning across 3D (z-axis) and 2D (x-y plane) spaces. By introducing non-Markovian elements, they emphasize the need for memory representations to solve complex sequential decision-making problems (more details in Appendix F).

# 4 METHOD

Our method, SAM2Act, enables precise 3D manipulation with strong generalization across environmental and object-level variations. Building upon the RVT-2 framework Goyal et al. (2024), SAM2Act introduces key architectural innovations that enhance visual feature representation and task-specific reasoning. The architecture reconstructs a point cloud of the scene, renders it from virtual cameras at orthogonal views, and employs a two-stage multi-view transformer (coarse-to-fine) to predict action heatmaps. The coarse branch generates zoom-in heatmaps to localize regions of interest, while the fine branch refines these into precise action heatmaps. SAM2Act leverages the pre-trained SAM2 encoder Ravi et al. (2024) to extract multi-resolution image embeddings, which are further refined through the multi-resolution upsampling technique to predict accurate translation heatmaps with minimal information loss. To address tasks requiring spatial memory, SAM2Act+ extends the SAM2Act architecture by incorporating memory-based components. These include Memory Bank, Memory Encoder, and Memory Attention, enabling the model to encode historical actions and condition current observations. This memory-based policy enhances the agent's ability to predict actions based on past contextual information, significantly improving performance in tasks that require sequential decision-making.

In the following sections, we detail the SAM2Act architecture (subsection 4.1), including its multi-resolution upsampling mechanism (Figure 4). We also present the SAM2Act+ extension, which integrates memory-based components for solving spatial memory tasks (subsection 4.2).

## 4.1 SAM2ACT: MULTI-RESOLUTION UPSAMPLING FOR ENHANCED VISUAL FEATURE REPRESENTATION

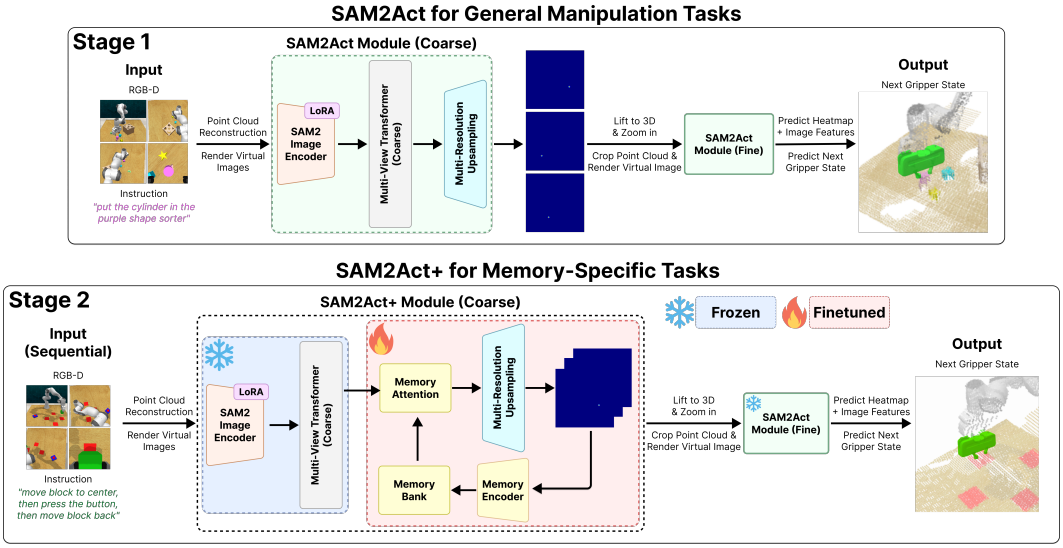


Figure 3: **Overview of the SAM2Act (top) and SAM2Act+ (bottom) architectures.** The SAM2Act architecture leverages the SAM2 image encoder to generate prompt-conditioned, multi-resolution embeddings, fine-tuned with LoRA for efficient adaptation to manipulation tasks. A multi-view transformer aligns spatial coordinates with language instructions, while a cascaded multi-resolution upsampling mechanism refines feature maps and generates accurate translation heatmaps. SAM2Act+ extends this architecture by incorporating memory-based components, including the Memory Encoder, Memory Attention, and Memory Bank, into the coarse branch. These components enable memory-driven reasoning by processing historical heatmaps and integrating prior observations, allowing the agent to predict actions based on stored contextual information. Observations are reconstructed into point clouds, rendered into three virtual images, and lifted into 3D translation points, enabling precise spatial reasoning across both architectures.

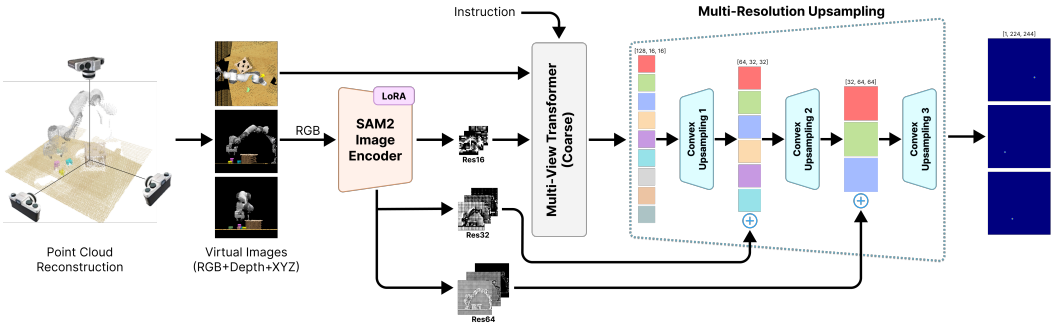


Figure 4: **SAM2Act Module and multi-resolution upsampling mechanism.** A cascade of three convex upsamplers processes feature maps at increasing resolutions, integrating multi-resolution embeddings from the SAM2 image encoder through elementwise addition and layer normalization. The upsamplers progressively refine features, doubling spatial dimensions at each stage, to generate accurate translation heatmaps while capturing fine-grained spatial details critical for manipulation tasks.

A distinctive feature of SAM2Act is the incorporation of the SAM2Act Module into the manipulation backbone for training, as illustrated in Figure 4. The coarse and fine SAM2Act Modules share the same architecture, with the fine branch generating additional features to predict actions beyond translation, while the coarse branch focuses exclusively on translation. Point-cloud representations are reconstructed from raw image inputs, and virtual images are generated from three viewpoints using virtual cameras. Instead of directly inputting these images into the multi-view transformer, their RGB channels are duplicated and processed by the SAM2 Ravi et al. (2024) image encoder, which produces object-centric multi-resolution embeddings. These embeddings, generated at three resolution levels, are combined with virtual images containing RGB, depth, 3D translation coordinates, and language instructions before being fed into the multi-view transformer. Details of how we adapt the MVT can be found in Appendix A.

To adapt the SAM2 image encoder to our domain, we fine-tune it using Low-Rank Adaptation (LoRA) Hu et al. (2021) with a default rank of 16, which enables domain adaptation with minimal computational cost while maintaining model efficiency. Additionally, to fully leverage the multi-resolution embeddings produced by the SAM2 image encoder, we introduce a multi-resolution upsampling method. This method uses the embeddings as auxiliary inputs to enhance the generation of translation heatmaps, thereby improving spatial precision and overall system performance. The multi-resolution upsampling mechanism, also detailed in Figure 4, leverages cascaded convex upsamplers to progressively refine feature maps across resolutions. Let $X^l \in \mathbb{R}^{B\times C^l\times H^l\times W^l}$ denote the feature maps at stage $l$ and $E^l \in \mathbb{R}^{B\times C^l\times H^l\times W^l}$ the corresponding multi-resolution embedding from SAM2. Also let $U(\cdot)$ denote the upsampling operator that doubles the spatial dimensions. The feature maps are updated at each stage as follows:

$$X^{l+1} = \text{LayerNorm}\big(U(X^l) \oplus E^l\big),$$

where $\oplus$ represents element-wise addition. The upsampling operator $U$ is defined as:

$$U : \mathbb{R}^{B\times C^l\times H^l\times W^l} \to \mathbb{R}^{B\times (C^l/2)\times (2H^l)\times (2W^l)}.$$

At each stage, the output of the upsampler is combined with the corresponding multi-resolution embedding $E^l$ from the SAM2 encoder, ensuring alignment between the multi-resolution features and the decoder's spatial refinement process. A layer normalization step follows each addition to stabilize training and maintain feature coherence. This results in direct integration of the embeddings into the translation heatmap generation process. The cascading structure refines features across multiple resolutions, capturing fine-grained spatial details critical for manipulation tasks.

**Algorithm 1** Forward Pass of SAM2Act+ Module

**Initialize:** Number of steps $N$, maximum number of memories $M$, number of views $V$, empty memory bank $Q$ with $V$ separate FIFO queues, input $X$
**for** $i = 1$ **to** $N$ **do**
  **for** $j = 1$ **to** $V$ **do**
    Get embeddings $\mathcal{E}_{raw}$ from MVT $T_{mv}(X_j)$
    Retrieve past memories $\mathcal{M}_{old}$ from $Q[j]$
    Get memory-conditioned embeddings $\mathcal{E}_{mem}$ from Memory Attention $T_{mem}(\mathcal{E}_{raw}, \mathcal{M}_{old})$
    Predict translation heatmap $\mathcal{H}$ with upsampler $U(\mathcal{E}_{mem})$
    Encode new memory $\mathcal{M}_{new}$ using Memory Encoder $E_{mem}(\mathcal{H}, \mathcal{E}_{raw})$
    Store new memory $Q[j] \leftarrow Q[j] \cup \{\mathcal{M}_{new}\}$
    **if** $|Q[j]| = M$ **then**
      $Q[j] \leftarrow Q[j]_{2:n}$
    **end if**
  **end for**
**end for**

### 4.2 SAM2Act+: Action Memory Architecture for Improved Spatial Awareness in Past Observations

To extend the SAM2Act architecture (subsection 4.1) with memory-based capabilities inspired by SAM2, we introduce SAM2Act+, a task-specific variant designed for solving memory-based tasks. SAM2Act+ integrates the three core memory components from SAM2—*Memory Attention, Memory Encoder, and Memory Bank*—into the coarse branch of SAM2Act. Originally developed for object tracking in SAM2, these components are adapted to align with the needs of SAM2Act+, enabling the agent to retain prior actions and observations for sequential decision-making. In SAM2, the Memory Encoder processes predicted object masks, while the Memory Attention module fuses image embeddings with positional information from previous frames. SAM2Act+ adopts a similar structure: the predicted heatmaps, which serve as binary indicators of spatial positions in the image, function analogously to object masks. This conceptual alignment ensures a seamless integration of memory mechanisms, allowing the agent to leverage stored information to predict subsequent actions based on historical context. A detailed description of the Memory Attention and Memory Encoder modules can be found in Appendix A.

**Architecture.** The SAM2Act+ architecture is illustrated in Figure 3. After pretraining SAM2Act in Stage 1, we freeze the SAM2 image encoder and the multi-view transformer in the coarse branch, as these components effectively generate robust embeddings for multi-view images in manipulation tasks. We also freeze the entire fine branch, given its proven ability to predict fine-grained actions accurately. The reason why we only fine-tune the coarse branch is because it focuses on generating heatmaps that provide richer contextual information for recalling past actions. The fine branch, in contrast, primarily emphasizes small objects or localized regions, which typically contain less information relevant to memory-based tasks.

**Training.** To train SAM2Act+, we fine-tune the coarse branch by integrating the three memory components (and train them from scratch) with the multi-resolution upsampling module. During fine-tuning, consecutive action keyframes are sampled as input, training the multi-resolution upsampler to predict new translations conditioned on memory. The memory components function similarly to their implementation in SAM2 for object tracking, with one key distinction: the input to the Memory Encoder. Instead of using image embeddings from the SAM2 image encoder, we input feature embeddings generated by the multi-view transformer (not conditioned by memory). This adaptation ensures that memory encoding incorporates multi-view information while maintaining independence in handling stored representations. Virtual images are treated independently during memory encoding and attention, with each view's memory encoded separately. Feature embeddings from each view are attended to using their corresponding stored memories, preserving spatial and contextual alignment while leveraging fused multi-view information. This structured approach prevents cross-view interference and enhances the model's ability to reason over sequential tasks. The memory-based forward pass for SAM2Act+ is outlined in Algorithm 1. By incorporating the

memory mechanism, SAM2Act+ enhances performance in scenarios requiring long-term reasoning, enabling the agent to make informed decisions based on historical context.

# 5 EXPERIMENTS

We study SAM2Act and SAM2Act+ in both simulated and real-world environments. Specifically, we are interested in answering the following questions:

§ 5.2 How does SAM2Act compare with state-of-the-art 3D manipulation policies?
§ 5.3 Can SAM2Act generalize across object and environmental perturbations?
§ 5.4 Can SAM2Act+ solve spatial memory-based tasks that other baselines cannot?
§ 5.5 How well does SAM2Act and SAM2Act+ perform on real-world tasks?

## 5.1 EXPERIMENTAL SETUP

We benchmark SAM2Act in both simulated and real-world environments. The simulated environments serve as a controlled platform to ensure reproducible and fair comparisons. The real-world experiments demonstrate the applicability of the method to real-world settings. Section 5.1 details our experimental setup and outlines the evaluation methodology. Training details can be found in Appendix B.

**Simulation Setup.** All simulated experiments were conducted in the CoppeliaSim environment via PyRep, using a 7-DoF Franka Emika Panda robot in a tabletop setting. Observations were captured from five RGB-D cameras—front, left shoulder, right shoulder, overhead and wrist—each at $128\,\text{px} \times 128\,\text{px}$. The robot receives a keyframe specifying translation and quaternion orientation and utilizes an OMPL-based motion planner to move to the target pose.

**Real-robot Setup.** We validate SAM2Act in real-world scenarios using a Franka Emika Panda robot with a Robotiq 2F-85 gripper and a exocentric Intel RealSense D455 depth sensor (more in Appendix G). We study four manipulation tasks, aligning three with RVT-2 for comparison and introducing a new memory-based task. We use the software stack as in Grotz et al. (2024). For each task, we collect 10–15 demonstrations via kinesthetic teaching and scripted execution with scene and object variations. As in Figure 2, we evaluate SAM2Act against RVT-2 for tasks (a)–(c) and SAM2Act+ for memory task (d). Each task undergoes 10 in-distribution and 10 out-of-distribution trials, including environmental perturbations, measuring total success.

**18 RLBench & `MemoryBench` Tasks.** To evaluate the general performance of SAM2Act and the memory capabilities of SAM2Act+, we conducted simulation experiments on two benchmarks: a subset of 18 tasks from RLBench and `MemoryBench`. RLBench is a standard multi-task manipulation benchmark, from which we selected 18 tasks well-studied in prior work. `MemoryBench` is a curated set of three tabletop manipulation tasks in CoppeliaSim that require the trained policy to have both semantic and spatial memory of past scenes and actions. In both benchmarks, each task is defined by a language instruction with 2–60 variations (e.g., handling objects, locations, and colors). We collected 100 demonstrations per task for training and held out 25 unseen demonstrations per task for testing. All policies are evaluated four times to obtain standard deviations. Tasks details can be found in Appendix E and Appendix F.

**3D Baselines.** We benchmark SAM2Act and SAM2Act+ against the current state-of-the-art 3D next-best-pose prediction model, RVT-2. RVT-2 is a multi-view robotics transformer that leverages a coarse-to-fine approach on the constructed point cloud to predict the next best action heatmap. We also compare with RVT Goyal et al. (2023), PerAct Shridhar et al. (2023), and SAM-E Zhang et al. (2024).

## 5.2 PERFORMANCES ACROSS 18 RLBENCH TASKS

Table 1 compares SAM2Act with prior keyframe-based 3D BC methods on the RLBench benchmark. Overall, SAM2Act achieves an average success rate of **86.8%**$\pm$0.5, surpassing the previous best (RVT-2) by **5.4%**. A closer look at individual tasks reveals that SAM2Act ranks **first in 9 out of 18 tasks** and remains highly competitive in **7 others**, coming within **one successful attempt or 4%** of the best performance. These tasks include Close Jar, Drag Stick, Meat Off Grill, Place Wine, Screw Bulb, Sweep to Dustpan, and Turn Tap. The largest margin of improvement occurs in `Insert Peg`,

where SAM2Act **exceeds RVT-2 by 44% (approximately 2.1×)**, and in `Sort Shape`, where it outperforms RVT-2 by 29%. Both tasks require precise manipulation, underscoring the effectiveness of SAM2Act's multi-resolution upsampling strategy. These results establish SAM2Act as a **leading policy for complex 3D tasks**, highlighting its ability to handle high-precision manipulations - an area where prior methods have struggled. Ablation studies are performed on SAM2Act in Appendix C.

Table 1: **Multi-Task Performance on RLBench.** We report the success rates for 18 RLBench tasks James et al. (2020), along with the average success rate and ranking across all tasks. Our method, SAM2Act, outperforms all baselines, achieving a significant performance margin of 5.8% over RVT-2 Goyal et al. (2024), the current state-of-the-art 3D keyframe-based behavior cloning (BC) policy.

| Method | Avg. Success ↑ | Avg. Rank ↓ | Close Jar | Drag Stick | Insert Peg | Meat off Grill | Open Drawer | Place Cups | Place Wine | Push Buttons |
|---|---|---|---|---|---|---|---|---|---|---|
| PerAct Shridhar et al. (2023) | 49.4 ± 4.3 | 4.6 | 55.2 ± 4.7 | 89.6 ± 4.1 | 5.6 ± 4.1 | 70.4 ± 2.0 | 88.0 ± 5.7 | 2.4 ± 3.2 | 44.8 ± 7.8 | 92.8 ± 3.0 |
| RVT Goyal et al. (2023) | 62.9 ± 3.7 | 3.6 | 52.0 ± 2.5 | 99.2 ± 1.6 | 11.2 ± 3.0 | 88.0 ± 2.5 | 71.2 ± 6.9 | 4.0 ± 2.5 | 91.0 ± 5.2 | **100.0** ± 0.0 |
| RVT-2 Goyal et al. (2024) | 81.4 ± 3.1 | 1.9 | **100.0** ± 0.0 | 99.0 ± 1.7 | 40.0 ± 0.0 | **99.0** ± 1.7 | 74.0 ± 11.8 | 38.0 ± 4.5 | **95.0** ± 3.3 | **100.0** ± 0.0 |
| SAM-E Zhang et al. (2024) | 70.6 ± 0.7 | 2.6 | 82.4 ± 3.6 | **100.0** ± 0.0 | 18.4 ± 4.6 | 95.2 ± 3.3 | **95.2** ± 5.2 | 0.0 ± 0.0 | 94.4 ± 4.6 | **100.0** ± 0.0 |
| SAM2Act (Ours) | **86.8** ± 0.5 | **1.8** | 99.0 ± 2.0 | 99.0 ± 2.0 | **84.0** ± 5.7 | 98.0 ± 2.3 | 83.0 ± 6.0 | **47.0** ± 6.0 | 93.0 ± 3.8 | **100.0** ± 0.0 |

| Method | Put in Cupboard | Put in Drawer | Put in Safe | Screw Bulb | Slide Block | Sort Shape | Stack Blocks | Stack Cups | Sweep to Dustpan | Turn Tap |
|---|---|---|---|---|---|---|---|---|---|---|
| PerAct Shridhar et al. (2023) | 28.0 ± 4.4 | 51.2 ± 4.7 | 84.0 ± 3.6 | 17.6 ± 2.0 | 74.0 ± 13.0 | 16.8 ± 4.7 | 26.4 ± 3.2 | 2.4 ± 2.0 | 52.0 ± 0.0 | 88.0 ± 4.4 |
| RVT Goyal et al. (2023) | 49.6 ± 3.2 | 88.0 ± 5.7 | 91.2 ± 3.0 | 48.0 ± 5.7 | 81.6 ± 5.4 | 36.0 ± 2.5 | 28.8 ± 3.9 | 26.4 ± 8.2 | 72.0 ± 0.0 | 93.6 ± 4.1 |
| RVT-2 Goyal et al. (2024) | 66.0 ± 4.5 | 96.0 ± 0.0 | 96.0 ± 2.8 | 88.0 ± 4.9 | 92.0 ± 2.8 | 35.0 ± 7.1 | **80.0** ± 2.8 | 69.0 ± 5.9 | **100.0** ± 0.0 | 99.0 ± 1.7 |
| SAM-E Zhang et al. (2024) | 64.0 ± 2.8 | 92.0 ± 5.7 | 95.2 ± 3.3 | 78.4 ± 3.6 | **95.2**± 1.8 | 34.4 ± 6.1 | 26.4 ± 4.6 | 0.0 ± 0.0 | **100.0** ± 0.0 | **100.0** ± 0.0 |
| SAM2Act (Ours) | **75.0** ± 3.8 | **99.0** ± 2.0 | **98.0** ± 2.3 | **89.0** ± 2.0 | 86.0 ± 4.0 | **64.0** ± 4.6 | 76.0 ± 8.6 | **78.0** ± 4.0 | 99.0 ± 2.0 | 96.0 ± 5.7 |

### 5.3 Semantic Generalization across Tasks

The results evaluated in subsection 5.2 were obtained by training and testing models within the same environment. However, to truly assess **generalization performance**, policies must remain robust against both environmental and object-level perturbations. We therefore trained SAM2Act and the baseline methods on 20 tasks from `The Colosseum` benchmark and tested them under 13 different perturbation categories over three runs. **SAM2Act exhibits the smallest performance drop compared to the baselines**, with an average decrease of 4.3% (standard deviation of 3.59%). Notably, it proves particularly robust to environmental perturbations – such as changes in lighting, table color/texture, the addition of distractors, and even camera pose – while also maintaining competitive performance under object-level perturbations (see more analysis in subsection C.2).

Table 2: **`The Colosseum` results**. Task-average success rate percentage change for SAM2Act and other baselines across 13 perturbation factors from `The Colosseum`, relative to evaluations without perturbations. Our approach, SAM2Act, demonstrates the lowest average percentage change across all perturbations, with a minimal drop of -4.3±3.6%, highlighting its robustness in handling various environmental and object-level perturbations.

| Method | Average ↑ | MO-Color ↑ | RO-Color ↑ | MO-Texture ↑ | RO-Texture ↑ | MO-Size ↑ | RO-Size ↑ |
|---|---|---|---|---|---|---|---|
| RVT-2 Goyal et al. (2024) | -19.5±2.8 | -20.7±1.0 | -11.8±0.8 | -13.3±4.6 | -11.4±3.7 | **-13.2**±3.1 | -17.7±0.1 |
| SAM2Act (SAM2 → SAM) | -20.7±1.2 | -26.1±0.7 | -15.7±2.9 | -15.0±3.3 | -16.5±6.2 | -18.7±1.9 | -19.8±1.3 |
| SAM2Act (w/o Multi-res Input) | -19.1±4.5 | -15.5±6.4 | -13.5±4.6 | -20.4±0.5 | -16.6±6.1 | -21.3±7.5 | -12.6±7.5 |
| SAM2Act (Ours) | **-4.3**±3.6 | **-1.1**±2.5 | **-0.7**±7.2 | **-3.3**±2.4 | **24.72**±6.1 | -15.9±5.0 | **0.9**±6.8 |

| Method | Light Color ↑ | Table Color ↑ | Table Texture ↑ | Distractor ↑ | Background Texture ↑ | Camera Pose ↑ | All Perturbations ↑ |
|---|---|---|---|---|---|---|---|
| RVT-2 Goyal et al. (2024) | -15.6±1.3 | -26.5±4.4 | -14.6±4.4 | -4.9±5.3 | -4.4±4.0 | -19.5±2.8 | -77.9±1.7 |
| SAM2Act (SAM2 → SAM) | -16.3±1.2 | -23.5±5.3 | -12.3±3.1 | 0.6±2.9 | -5.4±3.2 | -20.7±1.2 | -79.5±2.5 |
| SAM2Act (w/o Multi-res Input) | -7.2±3.6 | -18.3±6.1 | -17.5±3.3 | -4.6±3.5 | -5.7±3.5 | -19.1±4.5 | -73.8 ±2.2 |
| SAM2Act (Ours) | **4.5**±4.4 | **1.1**±2.5 | **-3.7**±5.2 | **1.7**±1.7 | **-1.5**±2.7 | **-4.3**±3.6 | **-58.3**±4.4 |

### 5.4 Performance on MemoryBench

In Table 3, we evaluate SAM2Act+ against SoTA 3D BC model, RVT-2 on `MemoryBench`, training all models in a single-task setting to isolate memory-related challenges (e.g., opening the wrong drawer rather than unrelated mid-task failures). This setup ensures that performance differences stem from memory capabilities. For a random agent, the expected success rates are determined by the number of possible choices per task: 33% for `reopen_drawer` (three drawers), 25% for `put_block_back` (four patches), and 50% for `rearrange_block` (two blocks). However, variations in task complexity, fixed training data, and imbalanced task distributions lead to slight deviations from these baselines. Our proposed memory-based model, SAM2Act+, demonstrates **a strong understanding of spatial memory**, achieving an average success rate of 94.3% across all tasks. It **outperforms SAM2Act (without memory) by a huge margin of 39.3% on `MemoryBench`**, highlighting the significant impact of explicit memory modeling.

Table 3: **Performance on `MemoryBench`.** We report the success rates for the three spatial memory tasks in `MemoryBench`. Our method, SAM2Act+, significantly outperforms all baseline methods that lack an explicit memory mechanism, achieving an average improvement of 37.6% across all three tasks. Note that there is an update with `MemoryBench`, see more in Appendix D.

| Methods / Tasks | Avg. Success ↑ | (a) Reopen Drawer | (b) Put Block Back | (c) Rearrange Block |
|---|---|---|---|---|
| RVT-2 | 54.0 ± 5.3 | 60.0 ± 0.0 | 50.0 ± 2.3 | 52.0 ± 3.3 |
| SAM2Act (Ours) | 55.0 ± 24.3 | 48.0 ± 0.0 | 35.0 ± 3.8 | 82.0 ± 2.3 |
| SAM2Act+ (Ours) | **94.3 ± 9.0** | **84.0** ± 0.0 | **100.0** ± 0.0 | **99.0** ± 2.0 |

### 5.5 Real-robot Evaluations

Table 4 presents our real-world experiment results, where our method achieves a 75% task success rate, compared to 43% for RVT-2. SAM2Act significantly outperforms the baseline in high-precision tasks (60% vs 0%). It excels in memory-based tasks, such as `(d) Push the same button`, which requires recalling the button's previous location. Here, SAM2Act achieves 70% success, while RVT-2, relying on random guessing, scores 40%. We also test models' generalization against perturbations like lighting changes, distractors, and position variations. Additional details are in the Appendix G, with real-world rollout videos available on our project website.

Table 4: **Real-world results.** We compare RVT2 against SAM2Act for the first three tasks and SAM2Act+ on the last real-world tasks (indicated with *), evaluating performance both in-distribution and out-of-distribution during test time.

| | **In-Distribution** | | **Out-Distribution** | |
|---|---|---|---|---|
| **Task** | **RVT-2** | **SAM2Act** | **RVT-2** | **SAM2Act** |
| (a) turn on the lamp | 0/10 | **6/10** | 0/10 | **6/10** |
| (b) push button sequence | 4/10 | **9/10** | 1/10 | **9/10** |
| (c) stack cubes | 8/10 | 8/10 | 3/10 | 3/10 |
| (d) push the same button * | 4/10 | **7/10** | 2/10 | **6/10** |

## 6 Conclusion & Limitation

We introduce SAM2Act, a multi-view, language-conditioned behavior cloning policy for 6-DoF 3D manipulation, enabling high-precision manipulations while generalizing effectively to unseen perturbations. Building on this foundation, we propose SAM2Act+, a memory-based multi-view language-conditioned robotic transformer-based policy that equips the agent with spatial memory awareness, allowing it to solve spatial memory-based tasks. While both SAM2Act and SAM2Act+ achieve SOTA performance across multiple benchmarks, challenges remain in extending them to dexterous continuous control. Additionally, SAM2Act+ relies on a fixed memory window length, which differs from task to task, limiting its adaptability to tasks of varying length. We also examined whether our memory architecture could retain semantic information (e.g., color), but unfortunately, it appears to be limited to storing spatial information. Despite these challenges, we believe SAM2Act+ is an important step towards memory-based generalist manipulation policies.

## REFERENCES


Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1722–1729. IEEE, 2017.

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

Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

Shizhe Chen, Ricardo Garcia, Cordelia Schmid, and Ivan Laptev. Polarnet: 3d point clouds for language-guided robotic manipulation. *arXiv preprint arXiv:2309.15596*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *ArXiv*, abs/2307.08691, 2023. URL https://api.semanticscholar.org/CorpusID:259936734.

Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pp. 1183–1198. PMLR, 2023.

Jiafei Duan, Wentao Yuan, Wilbert Pumacay, Yi Ru Wang, Kiana Ehsani, Dieter Fox, and Ranjay Krishna. Manipulate-anything: Automating real-world robots using vision-language models. *arXiv preprint arXiv:2406.18915*, 2024.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: Infinite resolution action detection transformer for robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pp. 694–710. PMLR, 2023.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

Markus Grotz, Mohit Shridhar, Yu-Wei Chao, Tamim Asfour, and Dieter Fox. Peract2: Benchmarking and learning for robotic bimanual manipulation tasks. In *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*, 2024. URL https://openreview.net/forum?id=nIU0ZFmptX.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The international journal of Robotics Research*, 31(5):647–663, 2012.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision*, 2024. URL https://api.semanticscholar.org/CorpusID:268536717.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Haoxu Huang, Fanqi Lin, Yingdong Hu, Shengjie Wang, and Yang Gao. Copa: General robotic manipulation through spatial constraints of parts with foundation models. *arXiv preprint arXiv:2403.08248*, 2024.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Stephen James and Pieter Abbeel. Coarse-to-fine q-attention with learned path ranking. *arXiv preprint arXiv:2204.01571*, 2022.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Sascha Jockel, Martin Weser, Daniel Westhoff, and Jianwei Zhang. Towards an episodic memory for cognitive robots. In *Proc. of 6th Cognitive Robotics workshop at 18th European Conf. on Artificial Intelligence (ECAI)*, pp. 68–74. Citeseer, 2008.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020b.

Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

Peiqi Liu, Zhanqiu Guo, Mohit Warke, Soumith Chintala, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. *arXiv preprint arXiv:2411.04999*, 2024.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36: 655–677, 2023.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

Fabian Peller-Konrad, Rainer Kartmann, Christian RG Dreher, Andre Meixner, Fabian Reister, Markus Grotz, and Tamim Asfour. A memory system of a robot cognitive architecture and its implementation in armarx. *Robotics and Autonomous Systems*, 164:104415, 2023.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. URL `https://api.semanticscholar.org/CorpusID:231591445`.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *arXiv preprint arXiv:2107.03380*, 2021.

Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.

Olaolu Shorinwa, Johnathan Tucker, Aliyah Smith, Aiden Swann, Timothy Chen, Roya Firoozi, Monroe David Kennedy, and Mac Schwager. Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. In *8th Annual Conference on Robot Learning*, 2024.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. PMLR, 2022.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *ArXiv*, abs/2104.09864, 2021. URL `https://api.semanticscholar.org/CorpusID:233307138`.

Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 32974–32988, 2022a.

Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant $q$ learning in spatial action spaces. In *Conference on Robot Learning*, pp. 1713–1723. PMLR, 2022b.

Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International conference on learning representations*, 2021.

Wentao Yuan, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. M2t2: Multi-task masked transformer for object-centric pick and place. *arXiv preprint arXiv:2311.00926*, 2023.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pp. 726–747. PMLR, 2021.

Junjie Zhang, Chenjia Bai, Haoran He, Wenke Xia, Zhigang Wang, Bin Zhao, Xiu Li, and Xuelong Li. Sam-e: Leveraging visual foundation model with sequence imitation for embodied manipulation. *arXiv preprint arXiv:2405.19586*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

# A MODEL ARCHITECTURE

We will explain our model architecture in detail, including Multi-View Transformer, Memory Attention, Memory Encoder, and Memory Bank. The multi-resolution is already explained in subsection 4.1.

**Multi-View Transformer.** The two MVTs used in the coarse and fine branches have the same architecture. Very similar to the MVT proposed by Goyal et al. (2023), the input to the transformer consists of a language description of the task, virtual images of the scene point cloud, and the image embeddings (at the lowest resolution) generated by the SAM2 image encoder. The text is transformed into token embeddings using the pre-trained CLIP Radford et al. (2021) model, while the virtual images are converted into token embeddings through patchify and projection operations. Similarly, the image embeddings are converted into token embeddings via a projection layer. For each virtual image, tokens corresponding to the same image are processed through four attention layers. Finally, the processed image tokens, along with the language tokens, are jointly processed using an additional four attention layers. The resulting image tokens are then used to infer the 3D action.

**Memory Attention.** Akin to the memory attention in SAM2 Ravi et al. (2024), the purpose of this module is to condition the current observation features on both past observation features and predicted actions, specifically translation. Notably, features from each view are processed independently. We stack four transformer blocks, with the first one taking the image embedding output of MVT from the current observation as input. Each block applies self-attention, followed by cross-attention to memories of past observation features and predicted actions, stored in a memory bank (described below), and ends with a multi-layer perceptron (MLP). For both self- and cross-attention, we use vanilla attention operations, enabling us to leverage recent advances in efficient attention kernels Dao (2023). In addition to sinusoidal absolute positional embeddings, 2D spatial Rotary Positional Embedding (RoPE) Su et al. (2021); Heo et al. (2024) are incorporated in both self-attention and cross-attention layers. We also reduce the dimension size from the original 256 to 128 to align with the image embedding dimension of the MVT output.

**Memory Encoder.** The memory encoder constructs memory features by downsampling the output translation heatmap using a convolutional module and summing it element-wise with the unconditioned observation embedding from the multi-view transformer (not shown in Figure 3). This is followed by lightweight convolutional layers to integrate the information. Instead of employing an additional image encoder, our memory encoder reuses the image embeddings produced by the MVT (not the SAM2 image encoder) and fuses them with the predicted translation information to generate memory features. This design enables the memory features to leverage rich representations that incorporate language, semantic, and spatial features from multiple views, making them more suitable for encoding action memories. Originally, this module was designed to encode an image embedding with multiple object masks within the same frame. However, we do not utilize this functionality. Instead, we encode one memory per view, where each memory is generated by encoding a single heatmap with a corresponding image embedding from each view.

**Memory Bank.** The memory bank preserves past translation predictions associated with previous observations in the video by maintaining a FIFO queue of up to $N$ recent memories. Each view has its own independent memory bank, as memories are stored and retrieved separately for different views. These memories are represented as spatial feature maps. Additionally, in our memory bank, the memory features are projected to a dimension of 64.

# B TRAINING IMPLEMENTATION

All models are trained on 32 NVIDIA H100/A100 GPUs. In some cases, we also train on 16 or 8 NVIDIA H100/A100 GPUs, but we ensure fairness by maintaining the same total batch size across all settings.

## B.1 SAM2ACT

We use the same way to data and demo augmentation methods and training pipeline as in RVT2 Goyal et al. (2024) to train SAM2Act (stage 1). The training hyperparameters are shown in Table 5. We use this set of hyperparameters to train on RLBench and `The Colosseum`.

Table 5: Training Hyperparameters of SAM2Act on RLBench and `The Colosseum`. The batch size stands for total batch size across all GPUs. For the learning rate, we follow the scaling strategy used in RVT2 Goyal et al. (2024), where the learning rate is scaled by the batch size as $1.25e-5 \times bs$.

| Hyperparameters | SAM2Act Training |
|---|---|
| batch size | 256 |
| learning rate | 3.2e-3 |
| optimizer | LAMB |
| learning rate schedule | cosine decay |
| weight decay | 1e-4 |
| warmup steps | 2000 |
| training steps | 56.25K |
| training epochs | 90 |
| LoRA rank | 16 |

### B.2 SAM2ACT+

We use a different strategy for sampling a batch of data for training. Previous sampling strategies randomly select a batch of independent observations, allowing the model to predict the next action based on each observation independently. However, for SAM2Act+, we aim for the agent to predict the next action based on both the current and past observations. To achieve this, we must sample a batch of data that is spatio-temporally consistent. To implement this, we randomly sample $n$ consecutive observations from a random episode. The forward pass is then performed sequentially from the first to the last observation. The details of the forward pass are provided in Algorithm 1.

When adopting this new sampling method during training, one immediate effect is a significant reduction in data diversity per batch. This can be detrimental, especially when dealing with tasks with numerous variations. We attempted to train the standard SAM2Act model on RLBench tasks using this new sampling method, but the convergence time was excessively long. To address this, we propose a new training pipeline: first, we pre-train the model using the previous sampling method, then fine-tune it with the new sampling approach. This strategy effectively mitigates the issue of slow convergence, significantly reducing training time.

As mentioned in subsection 5.4, we train all methods on `MemoryBench` in a single-task setting. However, finding a training configuration that optimizes all tasks is challenging. To address this, we use a universal set of hyperparameters for training but evaluate models across all epochs and select the best-performing one for evaluation. We follow the same approach to determine the optimal pre-trained weights for SAM2Act before fine-tuning on SAM2Act+. In addition, the window size of the memory mechanism is fixed to be 10 in all tasks in `MemoryBench`. We keep the batch size the same as the window size during training, and thus the learning rate will be a bit different as they are related with batch size. The detailed training hyperparameters are listed in Table 6.

## C ABLATION ON SAM2ACT

### C.1 RLBENCH

We conduct ablation experiments on the proposed SAM2Act, focusing on two key aspects: the SAM2 image Encoder and multi-resolution upsampling. We evaluate the model under three different configurations:

(i) Replacing the SAM2 image encoder with the SAM image encoder and removing the multi-resolution upsampling, as the SAM image encoder does not produce multi-resolution outputs. (ii) Replacing the multi-resolution upsampling with the original convex upsampling from RVT-2 Goyal et al. (2024). (iii) Removing SAM2's multi-resolution image embedding inputs to the multi-resolution upsampling while keeping the multi-resolution upsampling itself.

Note that SAM-E Zhang et al. (2024) proposed a 3D behavior cloning policy that integrates RVT and the SAM image encoder, along with an action-sequence policy head. We attempted to extend

Table 6: Training Hyperparameters of SAM2Act and SAM2Act+ on `MemoryBench`. Note that the batch size refers to the total batch size across all GPUs. For SAM2Act+, we use a maximum window size of 10 across all tasks, resulting in a per-GPU batch size of 10 and a total batch size of $10 \times 32 = 320$. The learning rate follows the same scaling rule mentioned in Table 5.

| Hyperparameters | SAM2Act Training | SAM2Act+ Training |
|---|---|---|
| batch size | 256 | 320 |
| learning rate | 3.2e-3 | 4e-3 |
| optimizer | LAMB | LAMB |
| learning rate schedule | cosine decay | cosine decay |
| weight decay | 1e-4 | 1e-4 |
| warmup steps | 2000 | 2000 |
| training steps | 6.25K | 12.5K |
| training epochs | 10 | 20 |
| LoRA rank | 16 | 16 |

this method to the more powerful RVT2 backbone for comparison. However, its action-sequence policy proved incompatible with the coarse-to-fine pipeline, resulting in very slow convergence under SAM-E's training setup. To ensure a fair comparison, we also extended SAM-E while keeping its original hyperparameters (notably, a LoRA rank of 4, whereas ours is 16). We trained both versions and found that SAM-E's configuration performed better. Therefore, we adopted their configuration and reported the results accordingly, which also applies to subsection 5.3. For all other ablation experiments, the training configuration are kept the same.

Ablation results on RLBench are presented in Table 7. All three variants of SAM2Act exhibit lower performance than the original version. Removing SAM2's multi-resolution image embedding inputs results in a 1.1% drop in the average success rate. Replacing the entire multi-resolution upsampling with the original convex upsampling leads to a 2.6% decrease. Substituting the SAM2 image encoder with the SAM image encoder causes a 6.0% drop compared to SAM2Act and a 3.4% drop compared to SAM2Act with the original convex upsampling—where the only differences are the image encoder and some training hyperparameters. These results indicate that all of our architectural innovations significantly enhance the agent's ability across multiple manipulation tasks.

Table 7: **SAM2Act Abaltion Performance on RLBench.** We report the success rates for 18 RLBench tasks James et al. (2020), along with the average success rate and ranking across all tasks. Table shows that SAM2Act outperforms all of its variations.

| Method | Avg. Success ↑ | Avg. Rank ↓ | Close Jar | Drag Stick | Insert Peg | Meat off Grill | Open Drawer | Place Cups | Place Wine | Push Buttons |
|---|---|---|---|---|---|---|---|---|---|---|
| SAM2Act (SAM2 → SAM) | 80.8 ± 1.9 | 2.8 | 96.0 ± 3.3 | 94.0 ± 4.0 | 28.0 ± 8.6 | 98.0 ± 2.3 | 72.0 ± 7.3 | 42.0 ± 6.9 | 95.0 ± 3.8 | **100.0** ± 0.0 |
| SAM2Act (Original Upsampling) | 84.2 ± 0.9 | 2.7 | **100.0** ± 0.0 | **100.0** ± 0.0 | **91.0** ± 3.8 | **99.0** ± 2.0 | 78.0 ± 9.5 | 29.0 ± 6.0 | 88.0 ± 5.7 | 96.0± 0.0 |
| SAM2Act (w/o Multi-res Input) | 85.7 ± 0.3 | 2.1 | 99.0 ± 2.0 | 96.0 ± 0.0 | 86.0 ± 8.3 | 98.0 ± 2.3 | **99.0** ± 2.0 | 43.0 ±10.5 | **96.0** ± 0.0 | **100.0** ± 0.0 |
| SAM2Act | **86.8** ± 0.5 | **1.8** | 99.0 ± 2.0 | 99.0 ± 2.0 | 84.0 ± 5.7 | 98.0 ± 2.3 | 83.0 ± 6.0 | **47.0** ± 6.0 | 93.0 ± 3.8 | **100.0** ± 0.0 |
| **Method** | **Put in Cupboard** | **Put in Drawer** | **Put in Safe** | **Screw Bulb** | **Slide Block** | **Sort Shape** | **Stack Blocks** | **Stack Cups** | **Sweep to Dustpan** | **Turn Tap** |
| SAM2Act (SAM2 → SAM) | 72.0 ± 8.6 | 94.0 ± 2.3 | **99.0** ± 2.0 | **92.0** ± 5.7 | 97.0 ± 3.8 | 41.0 ± 3.8 | 73.0 ± 3.8 | 71.0 ± 2.0 | 96.0 ± 3.3 | 95.0 ± 2.0 |
| SAM2Act (Original Upsampling) | 69.0 ± 5.0 | 98.0 ± 2.3 | 96.0 ± 3.3 | 84.0 ± 3.3 | **99.0** ± 2.0 | 52.0 ± 3.3 | 71.0 ± 3.8 | 80.0 ± 3.3 | **99.0** ± 2.0 | 87.0 ± 6.0 |
| SAM2Act (w/o Multi-res Input) | 72.0 ± 4.6 | **100.0** ± 0.0 | 96.0 ± 4.6 | 87.0 ± 2.0 | 82.0 ± 5.2 | 54.0 ± 5.2 | 74.0 ± 2.3 | **90.0** ± 6.9 | 97.0 ± 3.8 | 92.0 ± 4.6 |
| SAM2Act | **75.0** ± 3.8 | 99.0 ± 2.0 | 98.0 ± 2.3 | 89.0 ± 2.0 | 86.0 ± 4.0 | **64.0** ± 4.6 | **76.0** ± 8.6 | 78.0 ± 4.0 | **99.0** ± 2.0 | **96.0** ± 5.7 |

### C.2 THE COLOSSEUM

We also conducted the same ablation experiments on `The Colosseum` generalization benchmark, as shown in Table 2. The experimental setup remains the same as in Table 7, except that we did not test the variant of SAM2Act with the original convex upsampling. The results in Table 7 show that removing SAM2's multi-resolution image embedding inputs leads to a 14.8% drop in performance, representing a relative decrease of 344.2%. This highlights the effectiveness of SAM2's multi-resolution image embeddings in providing robust visual representations, significantly enhancing SAM2Act's generalization ability.

## D MEMORYBENCH UPDATE

We updated the `reopen_drawer` task in `MemoryBench` for the following reasons. During training on the original data, we observed that the gripper often collided with the drawer handle when closing the drawer. To prevent this, we introduced an additional waypoint for the closing motion, mirroring the procedure used for opening the drawer. Consequently, we retrained all policies specifically on this updated task. Furthermore, to standardize the memory window size across all three tasks, we also retrained SAM2Act+ on this task using a window size of 10, which led to improved performance. All results are updated to Table 3.

## E RLBENCH TASKS

We follow the multi-task, multi-variation simulated experiment setup of PerAct Shridhar et al. (2023), RVT Goyal et al. (2023), and RVT-2 Goyal et al. (2024), using 18 RLBench tasks with 249 unique variations in object placement, color, size, category, count, and shape. A summary of the 18 RLBench tasks is provided in Table 8. For a more detailed description of each task, please refer to PerAct Shridhar et al. (2023).

Table 8: The 18 RLBench tasks for multi-task experiment

| Task name | Language Template | Avg. Keyframes | #of Variations | Variation Type |
|---|---|---|---|---|
| put in drawer | "put the item in the __ drawer" | 12.0 | 3 | placement |
| reach and drag | "use the stick to drag the cube onto the __ target" | 6.0 | 20 | color |
| turn tap | "turn __ tap" | 2.0 | 2 | placement |
| slide to target | "slide the block to __ target" | 4.7 | 4 | color |
| open drawer | "open the __ drawer" | 3.0 | 3 | placement |
| put in cupboard | "put the __ in the cupboard" | 5.0 | 9 | category |
| place in shape sorter | "put the __ in the shape sorter" | 5.0 | 5 | shape |
| put money in safe | "put the money away in the safe on the __ shelf" | 5.0 | 3 | placement |
| push buttons | "push the __ button, [then the __ button]" | 3.8 | 50 | color |
| close jar | "close the __ jar" | 6.0 | 20 | color |
| stack block | "stack __ __ blocks" | 14.6 | 60 | color,count |
| place cups | "place __ cups on the cup holder" | 11.5 | 3 | count |
| place wine at rack | "stack the wine bottle to the __ of the rack" | 5.0 | 3 | placement |
| screw bulb | "screw in the __ light bulb" | 7.0 | 20 | color |
| sweep to dustpan | "sweep dirt to the __ dustpan" | 4.6 | 2 | size |
| insert peg | "put the ring on the __ spoke" | 5.0 | 20 | color |
| meat off grill | "take the __ off the grill" | 5.0 | 2 | category |
| stack cups | "stack the other cups on top of the __ cup" | 10.0 | 20 | color |

## F MEMORYBENCH TASKS

In the following we provide details of the `MemoryBench` tasks.

### (A) REOPEN DRAWER

**Task Description:** The robot is instructed remember the drawer slot that was initially opened, and closed it and then press the button on the table, before finding back the previously opened drawer to re-open it.

**Success Metric:** The task is considered successful once the initial opened drawer has been re-opened.

**Objects:** A drawer and button.

**Variation Number:** 3

**Keyframes:** 8

**Language Instructions:** *"Close the drawer, then reopened the previously opened drawer while pushing the button in between."*

Table 9: Properties of the real-world tasks. We report on language template, the average number of extracted keyframes, the number of items that the robot can interact with, the task variations and the variation type.

| Task name | Language template | # keyframes | # items | # variations | variation type |
|---|---|---|---|---|---|
| (a) turn on the lamp | "turn on the lamp" | 4.5 | 1 | 1 | placement |
| (b) push buttons in sequence | "push the red button, then the green button" | 5 | 3 | 1 | placement |
| (c) stack cubes | "stack the __ cube on the __ cube" | 4.0 | 5 | 3 | category,placement |
| (d) push the right button | "push the button closest to the blue block" | 6 | 3 | 1 | color,placement |

#### (B) PUT BLOCK BACK

**Task Description:** The robot is instructed move the block the centre, then push the button, then move the block back to its initial position.

**Success Metric:** The task is considered successful once the initial block has been moved back to its initial pose.

**Objects:** Four patch, one block and one button.

**Variation Number:** 4

**Keyframes:** 11

**Language Instructions:** *""Put the block to the centre and then back to its initial position while pushing the button in between.""*

#### (C) REARRANGE BLOCK

**Task Description:** The robot is instructed move the block in the centre to the empty patch, and then press the button, and then move the alternative block to the centre..

**Success Metric:** The task is considered successful once the alternative block has been moved to the centre.

**Objects:** Two patch, two blocks and one button.

**Variation Number:** 2

**Keyframes:** 10

**Language Instructions:** *"Move the block not on the patch to the empty patch, then press the button, then move the block that has not been moved off the patch."*

## G REAL-WORLD EXPERIMENTS

In the following we provide details of the real-world setup and tasks. Figure 5 illustrates the real-world setup. Table 9 summarizes the properties of the real-world tasks.

#### (A) TURN ON THE LAMP

**Task Description:** The robot is instructed to turn on a lamp by rotating its knob.

**Success Metric:** The task is considered successful once the lamp has been turned on by rotating the knob.

**Objects:** A single lamp.

**Coordination Challenges:** High precision is required to properly rotate the knob.

**Language Instructions:** *"Turn on the lamp."*

Figure 5: Robot setup. A Franka Panda robot with a Robotiq Gripper. A RealSense D455 depth sensor captures the scene.

#### (B) PUSH BUTTONS IN SEQUENCE

**Task Description:** The robot must press the red button first and then the blue button.

**Success Metric:** The task is considered successful if the buttons are pressed in the specified order: red, then blue. A third button is present but should remain unpressed.

**Objects:** Three buttons in front of the robot.

**Coordination Challenges:** Ensuring the robot presses the correct buttons in sequence without pressing the third button.

**Language Instructions:** *"Push the red button and then the blue button."*

#### (C) STACK BLOCKS

**Task Description:** The robot must place one specified block on top of another specified block.

**Success Metric:** The task is successful if the designated block is stacked on the correct target block.

**Objects:** Three single-colored blocks.

**Coordination Challenges:** Precision in picking and placing, plus correct language understanding to identify which block goes where.

**Language Instructions:** *"Stack the <item> block on the <item> block."*

#### (D) PUSH THE SAME BUTTON

**Task Description:** The robot must first identify and press the button closest to the blue block, then press the same button again after the block is removed.

**Success Metric:** The task is successful if the robot presses the correct button twice. Pressing the other button at any point results in failure.

**Objects:** Two buttons and one blue block (marking proximity).

**Coordination Challenges:** After the first button press, the blue block is removed; the robot must remember the button location to press it again.

**Language Instructions:** *"Push the button that is closest to the blue block. Press the same button again."*