Generalizable Visual Reinforcement Learning with Segment Anything Model

Ziyu Wang[1*] Yanjie Ze[2*] Yifei Sun[23] Zhecheng Yuan[124] Huazhe Xu[124]
[1]Tsinghua University, IIIS [2]Shanghai Qi Zhi Institute [3]Tongji University [4]Shanghai AI Lab
[*]Equal contribution

yanjieze.com/SAM-G

Figure 1. **Left:** For each task, one image and its mask from the training environment are provided to indicate task-relevant objects. Additionally, we assign 3 extra points to each object. **Right:** In an unseen environment, point prompts are found by correspondence, after which Segment Anything Model (SAM) produces high-quality masked images for visual RL agents.

Abstract

Learning policies that can generalize to unseen environments is a fundamental challenge in visual reinforcement learning (RL). While most current methods focus on acquiring robust visual representations through auxiliary supervision, pre-training, or data augmentation, the potential of modern vision foundation models remains underleveraged. In this work, we introduce ***S****egment* ***A****nything* ***M****odel for* ***G****eneralizable visual RL (****SAM-G****), a novel framework that leverages the promptable segmentation ability of Segment Anything Model (SAM) to enhance the generalization capabilities of visual RL agents. We utilize image features from DINOv2 and SAM to find correspondence as point prompts to SAM, and then SAM produces high-quality masked images for agents directly. Evaluated across 8 DMControl tasks and 3 Adroit tasks, SAM-G significantly improves the visual generalization ability without altering the RL agents' architecture but merely their observations. Notably, SAM-G achieves* 44% *and* 29% *relative improvements on the challenging video hard setting on DMControl and Adroit respectively, compared to state-of-the-art methods. Video and code: yanjieze.com/SAM-G.*

1. Introduction

Visual reinforcement learning (RL) has achieved great success in various applications such as video games [15, 16], robotic manipulation [20, 32, 33], and robotic locomotion [25, 26]. Despite the progress, RL agents are known to easily overfit when the training environments are not diverse and thus face severe generalization problems [5, 34, 37].

To improve the generalization ability, recent works try to acquire a visual representation that is robust to environment changes, by using auxiliary loss functions [2, 10], data augmentation [9, 11], and pre-training [30, 33]. In contrast, humans exhibit a remarkable ability to perform complex tasks in a variety of real-world scenarios, whether it is in a kitchen, a factory, or the wild, without depending on such specific designs. A plausible explanation could be our innate understanding of object concepts [1]. We humans intuitively *identify* and *segment* task-relevant objects in cluttered environments, which is yet to be seamlessly replicated by RL-trained agents.

In this work, we equip RL agents with the capability to *identify* and *segment* for generalization across unseen environments, by utilizing Segment Anything Model (SAM, [14]), a segmentation foundation model that is promptable to receive vision prompts, such as points, bounding boxes, and languages, for user-given images. Our novel framework, **S**egment **A**nything **M**odel for **G**eneralizable visual RL (**SAM-G**), mainly consists of two parts: *identify* and *segment*. We first harness image features from vision foundation models, *i.e.*, DINOv2 [17] and SAM, to extract task-

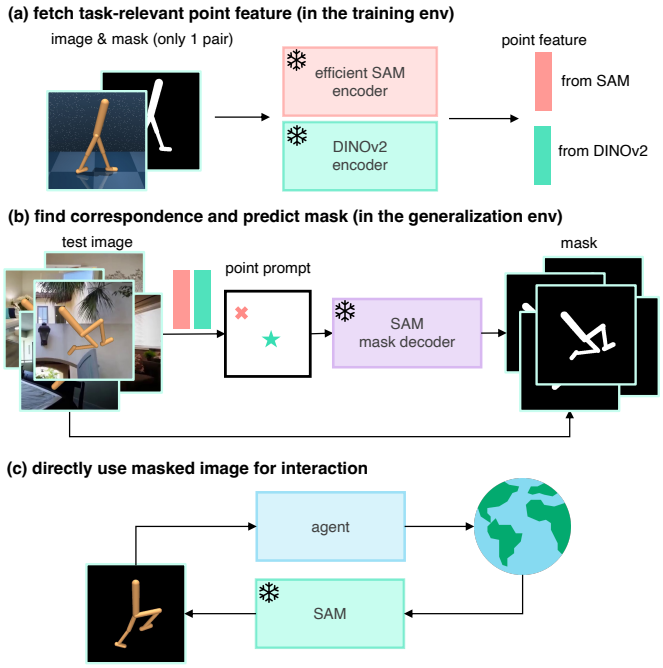

Figure 2. **Overview of SAM-G. (a)** We provide only one image and its mask from the training environment and use encoders of vision foundation models, *i.e.*, SAM [14] and DINOv2 [17], to extract point features. **(b)** We determine point prompts by finding correspondence in the test image and predict the mask with SAM. **(c)** The masked images are directly used for visual RL agents.

relevant features from the training environment, termed as *point feature*. Sparse points from human supervision are additionally provided to extract point features, only once for each task. Utilizing the point features, we compute the similarity map and determine point prompts in the test environment, which are then fed into SAM to accurately *segment* the objects that are critical to the task at hand, with iterative mask refinement. Moreover, SAM-G incorporates parameter-efficient finetuning techniques [36] for rapid adaptation of the SAM model (only 10 seconds). Subsequently, RL agents are directly fed with these high-quality masked images in both training and generalization environments. An overview of SAM-G is provided in Figure 2.

We evaluate SAM-G across a variety of tasks and domains, including 8 tasks from DeepMind Control Suite [21] and 3 dexterous manipulation tasks from Adroit [18], totaling **11** tasks. We use the generalization benchmark from DMC-GB [9] for DMControl tasks and RL-ViGen [31] for Adroit tasks. Extensive experiments show that our simple yet effective framework could significantly improve visual generalization capabilities, especially for the challenging *video hard* setting, where agents face dynamically changing backgrounds. Moreover, we note that SAM-G maintains consistent performance across settings of varying difficulty, in contrast to other baseline methods that exhibit significant performance degradation in challenging generalization settings. This observation aligns with our intuition, emphasizing the crucial importance of equipping agents with strong segmentation capabilities for robust visual generalization.

2. Related Work

Visual generalization in reinforcement learning. Reinforcement learning (RL) agents are known to be facing severe generalization issues [5, 11, 34, 37]. Contemporary research seeks to improve the visual generalization capabilities of RL agents [2, 9–11, 27, 30–32], with an emphasis on visual representations. Hansen and Wang [9] propose to decouple data augmentation from the policy learning process, thereby reducing the instability that augmentation may introduce during training. Hansen et al. [10] stabilize Q-function learning with a two-stream architecture. Yuan et al. [30] demonstrate the unexpected efficacy of utilizing an ImageNet [6] pre-trained encoder as representations. Ze et al. [32] pioneer the use of 3D pre-training for visual representations and facilitate a robust sim-to-real transfer. Bertoin et al. [2] employ saliency maps derived from Q-functions to guide agents in learning a mask. Yang et al. [27] tackle the issue of view generalization by exploiting environment dynamics. Orthogonal to these, our work primarily focuses on harnessing the segmentation foundation model to enhance the generalization ability of RL agents. Our method could be seamlessly incorporated into other algorithms.

Segment Anything Model (SAM, [14]) is a *promptable* segmentation foundation model, trained on over 11 million segmented images. It shows remarkable zero-shot segmentation abilities and can reproduce task-relevant masks with user-provided prompts, such as points, bounding boxes, and sparse masks. While concurrent research [4, 19, 36] has explored the application of SAM for object localization and tracking, our work diverges in two fundamental aspects: (1) we only provide images and masks in the training environment, emphasizing generalization to any unseen environments; and (2) our objective is to enable agents to accomplish desired tasks, more than simple object tracking.

Removing redundant information for generalization. To achieve generalization, our intuition is to remove the redundant information across different scenes and only leave the task-relevant objects. Such intuition is shared among this work and several previous works [2, 8, 23, 29]. All these works try to achieve generalization by learning a mask: Wang et al. [23] utilize the keypoint detection and visual attention for segmentation; Fu et al. [8] learn to reconstruct foreground and background separately; Yuan et al. [29] preserve the larger Lipschitz constant areas and perform augmentation on other areas. Compared to these works, our method directly produces high-quality masks by SAM, without the need for time-consuming training, auxiliary learning objectives, or specific architecture design.

3. Preliminaries

Formulation. We model the problem as a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = \langle \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathbf{o} \in \mathcal{O}$ are high-dimensional observations (*e.g.*, images), $\mathbf{a} \in \mathcal{A}$ are actions, $\mathcal{F} : \mathcal{O} \times \mathcal{A} \mapsto \mathcal{O}$ is a transition function, $r \in \mathcal{R}$ are rewards, and $\gamma \in [0, 1)$ is a discount factor. During training time, the agent's goal is to learn a policy π that maximizes discounted cumulative rewards on $\mathcal{M}$, *i.e.*, $\max \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. During test time, the reward signal from the environment is not accessible to agents and only observations are available. These observations are possible to experience subtle changes such as appearance changes and background changes.

Segment Anything Model (SAM, [14]) is a vision foundation model designed for promptable image segmentation. Trained on 11 million labeled images, SAM has shown a notable zero-shot segmentation capability. It is particularly adept at refining predicted masks through the integration of user prompts, including sparse prompts like points, bounding boxes, and languages, and dense prompts such as masks. The *promptable* nature of SAM facilitates its application in a variety of downstream tasks through prompt engineering alone, such as edge detection and object detection.

The architecture of SAM is tripartite: it features a ViT-based image encoder [7], a prompt encoder, and a lightweight mask decoder. The image encoder processes images of size $1024 \times 1024 \times 3$ into image embeddings with dimensions of $64 \times 64 \times 256$. The prompt encoder subsequently converts user prompts into corresponding embeddings, which the mask decoder then integrates with the image embeddings to produce the segmentation *mask logits*. The mask is then generated by thresholding mask logits. Inference latency is primarily attributed to the image encoder. In this work, we alleviate this issue by the usage of EfficientViT [3].

4. Method

In this section, we introduce **S**egment **A**nything for Generalization (**SAM-G**), a framework that effectively utilizes SAM to segment out the task-relevant objects and help agents generalize to various scenes. SAM-G focuses on addressing the visual generalization problem of RL agents. Specifically, during the training phase, agents are exposed solely to static training environments, where the visual appearance and backgrounds are not changing. However, at test time, though the task objective does not change, the agents encounter environments with significantly altered visual properties, such as varied appearances and backgrounds. Hence, the core principle of SAM-G is to capture the consistent elements between the training and testing phases—namely, the object and the agent—while discounting the elements that are not pertinent to the task.

An overview of SAM-G is given in Figure 2. SAM-G mainly consists of two parts, *identify* and *segment*:

- **Identify.** Given only 1 image from the training environment and its mask, we extract *point feature* utilizing image features from vision foundation models. The point feature could be understood as the abstract of the task-relevant objects.
- **Segment.** Utilizing the point feature obtained from the training environment, we feed points found by correspondence as sparse prompts into SAM and leverage SAM to segment out task-relevant objects.

After images from environments are segmented by SAM, RL agents directly process these segmented images as input or into the replay buffer. SAM-G thus could be seamlessly incorporated into other visual RL algorithms. Details of each part are illustrated in the following sections.

4.1. Identify

For each task, we provide only 1 image from the training environment and its mask as additional information to help SAM identify task-relevant objects.

Extract image features from foundation models. We leverage the image encoder from DINOv2 [17] and SAM [14] to extract image feature from the given image. Notably, for fast inference speed, we use the Efficient ViT-L1 [3] architecture for SAM. We use the ViT-B/14 architecture for DINOv2.

Fetch point feature. We then fetch features that represent the task-relevant objects from image features, termed as *point feature*. Two types of point feature are extracted, as illustrated in Figure 5:

- *Type 1* point feature is automatically computed based on the masked image feature, using the average of the spatial average pooled feature and spatial max pooled feature on the masked image feature, termed as $\frac{avg+max}{2}$.
- *Type 2* point feature is fetched with human supervision. We manually label 3 points for each object, termed as *extra points*, visualized in Figure 3. Then we directly get the point feature in the corresponding coordinates.

Take a single-object task as an example. The stored point feature has the dimension of 4×768 and 4×256, where 4 means one *type 1* point feature and three *type 2* point features, and 768 and 256 are the feature dimension of DINOv2 and SAM respectively.

4.2. Segment

We now describe how to utilize the point feature to find point prompts and segment out the task-relevant objects with SAM. The segmentation process is the same for images from the training and generalization environments. An illustration of the segmentation process is given in Figure 4.

Determining point prompts via correspondence. To segment a novel image, we commence by extracting its im-

Figure 3. **Visualization of extra points.** We use a green marker to denote the extra points provided by humans. Each object is assigned with 3 extra points. It is important to note that for each task, extra points are provided only once, and the resultant point features are stored. Consequently, the time required for labeling these points can be considered negligible.

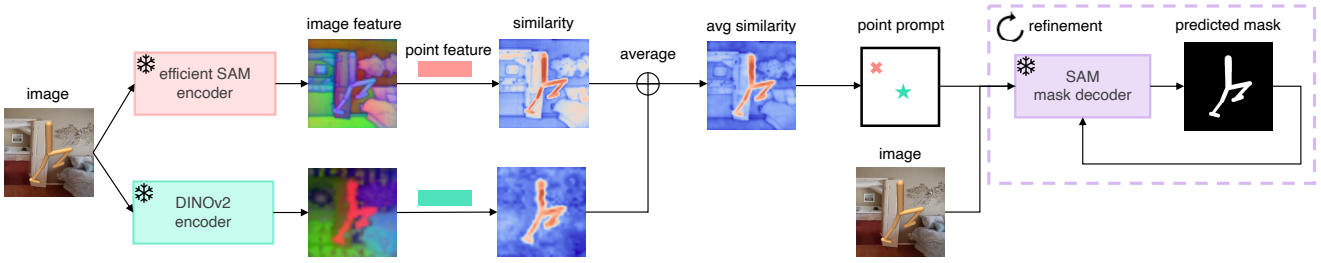

Figure 4. **Segmentation with point feature.** We use point features from the training environment to find correspondence in the test image and obtain point prompts. Then the mask decoder iteratively refines the predicted mask given point prompts.

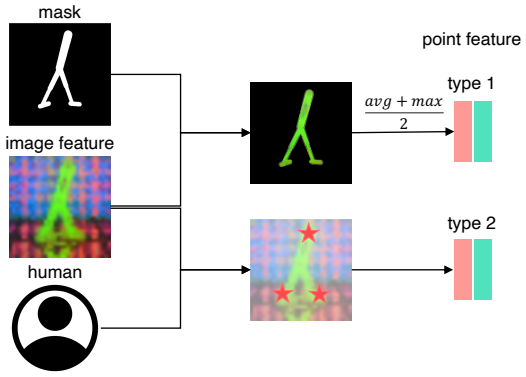

Figure 5. **Fetch point feature.** Given only 1 image in the training environment and its mask, we fetch two types of point features. *Type 1* is computed as the average of the spatial average pooled feature and spatial max pooled feature on the masked image feature; *type 2* is directly fetched given human-defined points, termed as *extra points*. Each object is assigned with 3 extra points.

age features using the encoders of SAM and DINOv2, the same as before. Subsequently, we construct a similarity map between these newly obtained image features and the pre-established point features. For each point feature, we average the similarity maps from two foundational models to achieve better object localization. As shown in Figure 4, the averaged similarity map distinctly highlights the target object. For the similarity map corresponding to the *type 1* point feature, we identify the point of highest similarity to serve as a positive point prompt and the point of least similarity as a negative point. In contrast, for the similarity map constructed by the *type 2* point feature, we solely select the point of highest similarity as a positive point prompt. This is because *type 2* point features only represent information from a specific part of the object, and the point of least similarity could be mistakenly put on other parts of the object. Features and similarity maps are visualized in Figure 6.

Mask prediction and refinement. After point prompts are obtained and encoded into embeddings by the prompt encoder of SAM, the mask decoder decodes the image feature and the prompt embedding into mask logits [14]. We further refine the prediction by repeating the mask decoding process with additional prompts. Specifically, after the 1st mask prediction, we compute the bounding box of the mask and select a point of least similarity in this box, based on the similarity map we obtain previously. Then we perform the 2nd and 3rd mask prediction with old prompts and newly added point prompts. Note that iterative mask refinement has been used in previous works such as SAM [14], PerSAM [36], and SAM-PT [19], while we propose to add additional negative points based on the similarity map in each refinement, to better separate the foreground and background in the generalization setting. The point prompts and masked images are visualized in Figure 1.

One shot adaptation. Before RL agents start to loop in the training environments, we fast adapt only two weights in SAM with our initially provided image and mask, following PerSAM [36]. Specifically, SAM produces 3 mask logits, denoted as M_1, M_2, M_3, and conducts a weighted summation,

$$M = w_1 \cdot M_1 + w_2 \cdot M_2 + (1 - w_1 - w_2) \cdot M_3 \,.$$

We use the provided image from the training environment as input and its mask as the supervision to adjust the weights w_1 and w_2. All other parts of SAM are frozen. This process

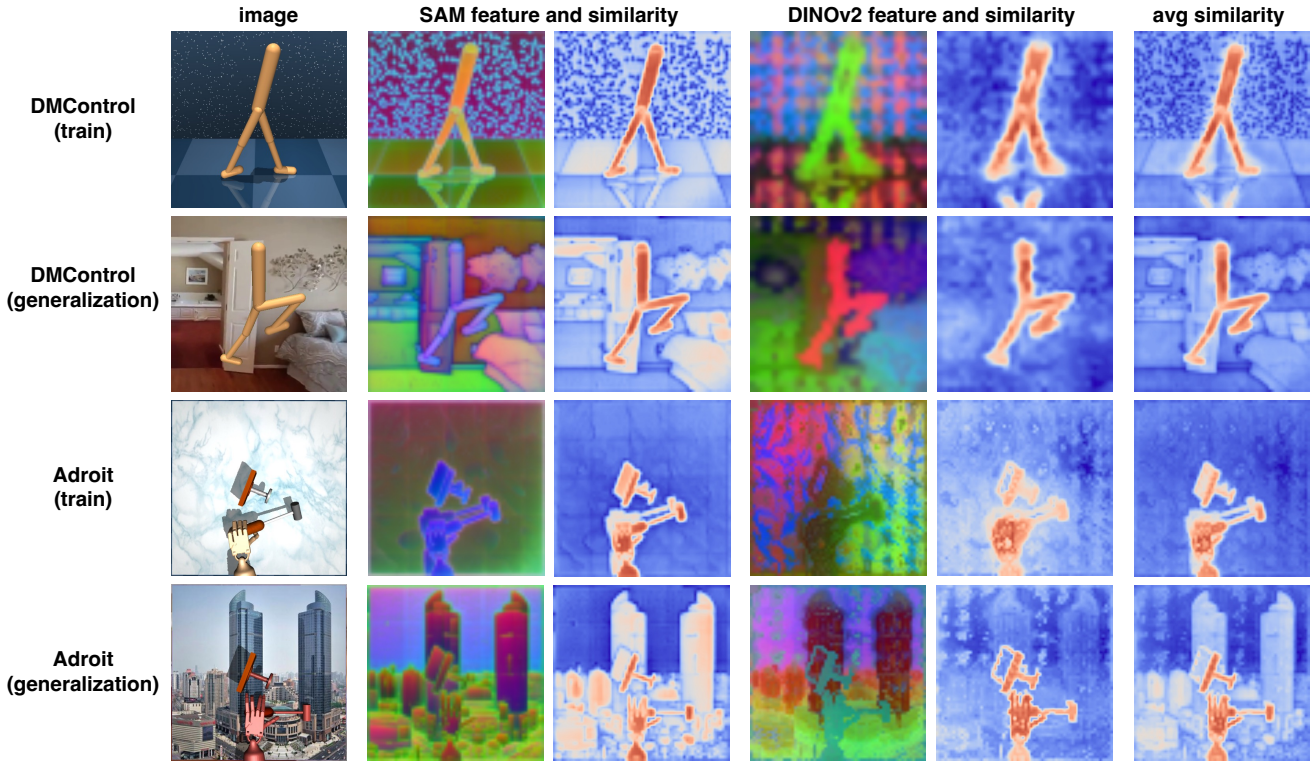

Figure 6. **Visualization of feature maps and similarity maps.** We visualize image features from foundation models and similarity maps that are used for finding correspondence. For image features, We employ Principal Component Analysis (PCA) to reduce the dimension of image features into 3 and visualize them as RGB images. For similarity maps, areas highlighted in red indicate high similarity, while blue regions represent the opposite. We only show two tasks from two domains here as the remaining tasks exhibit a similar pattern.

takes roughly 10 seconds on an Nvidia 3090 GPU, thus its time consumption could be almost neglected.

5. Experiments

In this section, we evaluate the generalization ability of SAM-G on 11 tasks, including 8 tasks from DMControl [21] and 3 dexterous manipulation tasks from Adroit [18]. We use the generalization setting from DMControl Generalization Benchmark (DMC-GB, [9]), where four settings are considered with increasing difficulty: *color easy*, *color hard*, *video easy*, and *video hard*. Detailed descriptions of tasks and generalization settings are in Appendix B. **Videos are given in supplementary files.**

5.1. Experiment Setup

Baselines. We benchmark SAM-G against the following strong baselines: (1) **DrQ-v2** [28] that applies random shift as data augmentation; (2) **VRL3** [22] that uses offline RL to pre-train for sample efficiency; (3) **SVEA** [10] that stabilizes the Q-function learning via an auxiliary loss; (4) **SGQN** [2] that leverages the saliency map to help remove redundant information; (5) **PIE-G** [30] that applies a pre-trained image encoder as the representation. Among these baselines, DrQ-v2 and VRL3 are designed for sample efficiency and thus their generalization ability is limited; DrQ-v2 and VRL3 serve as the backbone algorithms on DMControl tasks and VRL3 respectively. Other algorithms, including SVEA, PIE-G, SGQN, and SAM-G, are designed for visual generalization and built upon these backbone algorithms.

Implementation. We implement SAM-G on top of DrQ-v2, PIE-G (DrQ-v2 version), and PIE-G (VRL3 version), to show that SAM-G could be incorporated into other visual RL agents seamlessly. Visual observations are a stack of 3 RGB frames of size $84 \times 84 \times 3$. Mean and standard deviation of 3 random seeds are reported. For Adroit tasks, we use the implementation from RL-ViGen [31] since DMC-GB does not provide generalization settings on Adroit tasks. Hyperparameters are given in the supplementary material.

5.2. Main Experiments

Considering the extensive scale of our experiments, we present a summary of our main generalization results in Figure 7, across 2 domains and 4 settings. We then detail our findings below.

DMControl. Detailed generalization results on 8 DMControl tasks are given in Table 1. We omit the *color easy* setting since this setting is too easy for all methods on DMCon-

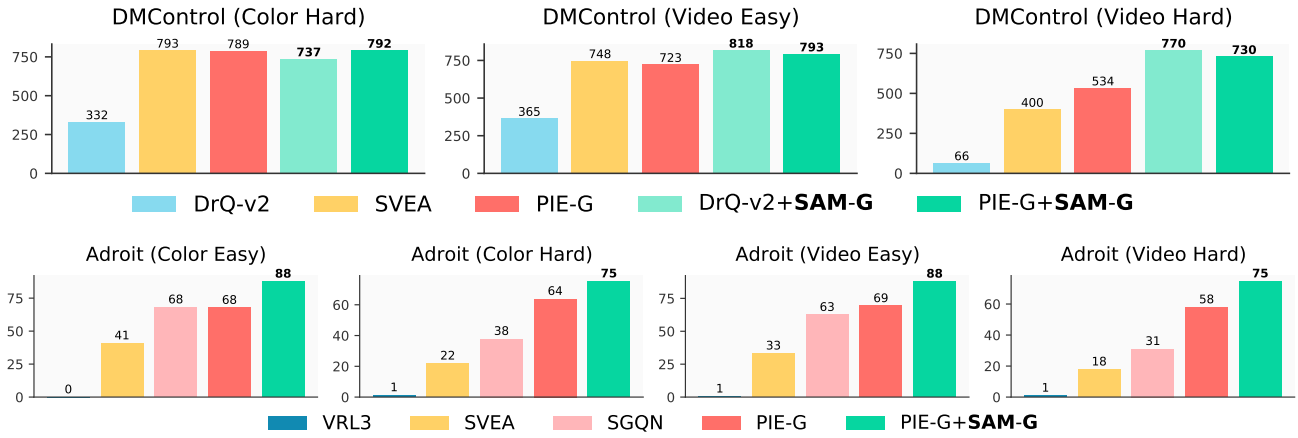

Figure 7. **Visual generalization results across 2 domains and 4 settings.** Our method SAM-G could robustly improve the visual generalization ability of visual RL agents such as DrQ-v2 [28] and PIE-G [30]. Notably, in the challenging *video hard* setting, SAM-G surpasses previous state-of-the-art method PIE-G with 44% and 29% relative improvement on DMControl and Adroit respectively.

trol. Training curves of 4 tasks from DMControl are displayed in Figure 8, where all the algorithms achieve similar sample efficiency and convergence. We could also observe that SAM-G slightly reduces the variance during training. Our observations are summarized as follows:

- Without specific design for generalization, the backbone algorithm DrQ-v2 encounters difficulties in all generalization settings. Other generalization methods such as SVEA and PIE-G exhibit robust performance in relatively easy scenarios, such as *color hard* and *video easy*, but exhibit significant performance drops in the more challenging *video hard* setting. This shows the limitations of current visual reinforcement learning algorithms and underscores the critical role of visual generalization.
- In contrast, SAM-G consistently delivers robust performance across all settings. Notably, in the *video hard* setting where other methods struggle, SAM-G achieves 770 average scores, largely surpassing PIE-G with 44% relative improvements. This verifies our intuition that the robust generalization capacity could arise from the robust segmentation capacity. Leveraging the capabilities of strong segmentation models such as SAM, agents should be adept at handling challenging generalization scenarios.

Adroit. We present the generalization results for 3 Adroit tasks in Table 2, spanning four challenging settings of increasing difficulty. The training curves for the Adroit tasks are displayed in Figure 8, where all algorithms exhibit similar convergence patterns, except for SVEA. This ensures a fair comparison in the generalization benchmark. Notably, we observe that SAM-G significantly improves sample efficiency on the Door task and does not hinder learning on the other two tasks. Our observations regarding the generalization results in Table 2 are summarized as follows:

- SAM-G demonstrates substantial performance enhancements, notably surpassing the previous state-of-the-art algorithm, PIE-G. This result is particularly significant because each Adroit task involves multiple objects, highlighting the generality of SAM-G for multi-object tasks,

Table 1. **Generalization results on DMControl.** Mean and std of 3 seeds are reported. Generalization settings are from DMC-GB [9].

DMControl (color hard)	DrQ-v2	SVEA	PIE-G	**DrQ-v2 + SAM-G**	**PIE-G + SAM-G**
Walker Walk	168±90	760±145	884±20	805±37	**895±24**
Walker Stand	413±61	942±26	960±15	839±80	**971±4**
Cartpole Swingup	277±80	**837±23**	749±46	728±15	751±57
Cheetah Run	109±45	273±23	**369±53**	280±48	349±28
Hopper Stand	383±41	**715±78**	681±72	659±7	706±55
Finger Spin	676±134	977±5	882±69	740±32	**935±66**
Ball_in_cup Catch	469±99	961±7	964±7	949±26	**970±5**
Quadruped Walk	162±40	879±8	822±8	**893±19**	759±44
Average	332	**793**	789	737	792

DMControl (video easy)	DrQ-v2	SVEA	PIE-G	**DrQ-v2 + SAM-G**	**PIE-G + SAM-G**
Walker Walk	175±117	819±71	871±22	887±33	**930±28**
Walker Stand	560±48	961±8	957±12	839±80	**970±7**
Cartpole Swingup	267±41	782±27	587±61	**855±7**	710±90
Cheetah Run	64±22	249±20	287±20	282±49	**345±8**
Hopper Stand	261±62	678±30	514±51	**971±2**	715±71
Finger Spin	456±15	808±33	837±107	**937±21**	928±65
Ball_in_cup Catch	454±60	871±106	922±20	858±79	**957±8**
Quadruped Walk	681±29	818±6	805±14	**916±14**	789±72
Average	365	748	723	**818**	793

DMControl (video hard)	DrQ-v2	SVEA	PIE-G	**DrQ-v2 + SAM-G**	**PIE-G + SAM-G**
Walker Walk	34±11	377±93	600±28	846±38	**857±31**
Walker Stand	151±13	834±46	852±56	966±2	**964±4**
Cartpole Swingup	130±3	393±45	401±21	**747±5**	653±58
Cheetah Run	23±5	105±37	154±17	283±46	**327±11**
Hopper Stand	5±5	221±13	235±17	**840±30**	618±63
Finger Spin	21±4	335±58	762±59	**932±22**	908±66
Ball_in_cup Catch	97±27	403±174	786±47	771±101	**880±54**
Quadruped Walk	69±21	532±23	483±62	**774±22**	635±21
Average	66	400	534	**770**	730

Table 2. **Generalization results on Adroit.** Mean and std of 3 seeds are reported. Generalization settings are from RL-ViGen [31].

Adroit (color easy)	VRL3	SVEA	SGQN	PIE-G	**PIE-G + SAM-G**
Pen	1.7±0.7	53.3±7.6	71.3±5.0	76.0±7.0	**79.0±4.4**
Door	0.0±0.0	45.4±9.7	58.2±12.9	81.6±6.7	**96.0±3.6**
Hammer	0.0±0.0	24.0±15.6	75.0±8.6	45.8±19.3	**88.0±9.3**
Average	0.6	40.9	68.2	67.8	**87.7**

Adroit (color hard)	VRL3	SVEA	SGQN	PIE-G	**PIE-G + SAM-G**
Pen	3.7±2.1	44.7±5.8	54.3±7.1	70.3±4.6	**75.3±6.7**
Door	0.0±0.0	11.8±3.6	31.4±9.8	67.8±9.8	**90.3±4.2**
Hammer	0.0±0.0	9.0±6.3	27.8±6.4	53.2±12.5	**60.0±12.8**
Average	1.2	21.8	37.8	63.8	**75.2**

Adroit (video easy)	VRL3	SVEA	SGQN	PIE-G	**PIE-G + SAM-G**
Pen	1.7±0.6	46.7±3.8	68.7±8.1	76.0±1.7	**82.3±5.1**
Door	0.0±0.0	44.8±8.5	58.2±12.3	81.6±4.4	**98.0±1.0**
Hammer	0.0±0.0	8.4±8.6	61.0±9.4	50.4±21.2	**83.3±7.0**
Average	0.6	33.3	62.6	69.3	**87.9**

Adroit (video hard)	VRL3	SVEA	SGQN	PIE-G	**PIE-G + SAM-G**
Pen	2.7±1.5	41.7±6.1	52.3±0.6	60.7±6.0	**76.7±2.1**
Door	0.0±0.0	7.6±1.8	21.6±6.4	60.4±12.3	**88.3±3.1**
Hammer	0.0±0.0	4.2±3.7	19.2±7.4	52.6±10.2	**58.7±11.2**
Average	0.9	17.8	31.0	57.9	**74.6**

extending beyond the simpler single-object tasks in DMControl.

- The performance of other baselines, including the previous state-of-the-art algorithm SVEA, is subpar. This once again underscores the critical importance of robust visual generalization.

5.3. Imitation Learning

Since SAM-G could be viewed as a general framework for generalizable visuomotor policy learning, we conduct initial experiments that apply SAM-G for imitation learning (IL) on challenging Adroit tasks.

Setup. We use the policy architecture of PIE-G [30]. We collect 300 expert demonstrations for each task using RL agents and train 80 epochs to ensure convergence. The training objective is simply behavior cloning with a mean squared error. We use Adam [13] optimizer with the learning rate 5×10^{-5} and the batch size 256. The data augmentation in PIE-G is also applied. Our baseline is the policy part of PIE-G, which is called PIE-G as well in this section.

Results. As indicated in Table 3, our observations on Adroit tasks remain consistent across both RL and IL settings. In both cases, SAM-G outperforms PIE-G by a significant margin on Door and Hammer tasks and achieves results that are comparable to PIE-G on the Pen task. These preliminary experiments in IL highlight the potential of our framework for visuomotor policy learning.

Table 3. **Imitation learning and generalization results on Adroit.** Mean and std of 3 seeds are reported. Generalization settings are the same as the RL experiments.

Imitation (Door)	Train	Color easy	Color hard	Video easy	Video hard	**Average**
PIE-G	93.0±3.6	85.7±2.3	62.5±16.5	78.7±7.5	61.0±8.5	76.2
SAM-G	96.7±3.5	97.0±3.0	91.3±8.1	96.3±0.6	88.3±1.5	**93.9**

Imitation (Hammer)	Train	Color easy	Color hard	Video easy	Video hard	**Average**
PIE-G	96.7±4.2	75.3±4.2	44.3±4.0	55.0±6.2	48.0±5.3	63.9
SAM-G	96.7±3.5	75.0±2.0	63.7±3.5	62.0±4.4	55.0±4.4	**70.5**

Imitation (Pen)	Train	Color easy	Color hard	Video easy	Video hard	**Average**
PIE-G	76.7±4.2	62.7±6.1	57.3±9.5	61.3±3.2	50.7±1.2	61.7
SAM-G	72.7±1.2	65.3±5.0	57.0±7.0	62.7±11.0	57.3±9.0	**63.0**

Table 4. **Ablations on DMControl.** Three main design choices of SAM-G are ablated: using DINOv2 to extract image features; adding extra points for more point features; and iterative mask refinement. We remove each component from SAM-G and observe performance drops compared to full SAM-G.

DMControl (color hard)	Walker Walk	Cartpole Swingup	Finger Spin	Ball_in_cup Catch	**Average**
SAM-G	805±37	728±51	739±32	949±26	805
w/o. DINOv2	726±39	589±75	785±90	921±10	755 (↓ 50)
w/o. extra points	761±64	649±78	783±34	929±17	781 (↓ 24)
w/o. refinement	708±73	790±123	617±27	935±15	763 (↓ 42)
w/o. PerSam adapt	659±93	501±28	833±136	894±91	722 (↓ 172)

DMControl (video easy)	Walker Walk	Cartpole Swingup	Finger Spin	Ball_in_cup Catch	**Average**
SAM-G	887±33	855±6	937±21	858±79	884
w/o. DINOv2	871±13	611±82	949±15	706±59	784 (↓ 100)
w/o. extra points	843±55	659±58	930±24	885±18	829 (↓ 55)
w/o. refinement	834±136	791±39	680±24	923±18	807 (↓ 77)
w/o. PerSam adapt	817±35	469±75	939±42	911±56	784 (↓ 100)

DMControl (video hard)	Walker Walk	Cartpole Swingup	Finger Spin	Ball_in_cup Catch	**Average**
SAM-G	870±28	747±5	932±22	771±101	830
w/o. DINOv2	782±27	307±61	862±22	404±97	589 (↓ 241)
w/o. extra points	805±41	491±45	688±11	847±53	708 (↓ 122)
w/o. refinement	819±140	717±75	591±18	883±29	753 (↓ 77)
w/o. PerSam adapt	803±26	336±107	919±22	802±112	715 (↓ 115)

5.4. Ablations

To verify the necessity of designs in SAM-G, we conduct a series of ablations on 4 DMControl tasks, including two single-object tasks and two multi-object tasks. The quantitative results are in Table 4. We conclude our observations below. More ablations are in the supplementary material.

Usage of DINOv2. SAM-G leverages both the DINOv2 encoder and the SAM encoder to extract point features. This design stems from our observation that combining image features from two foundational models enhances the local-

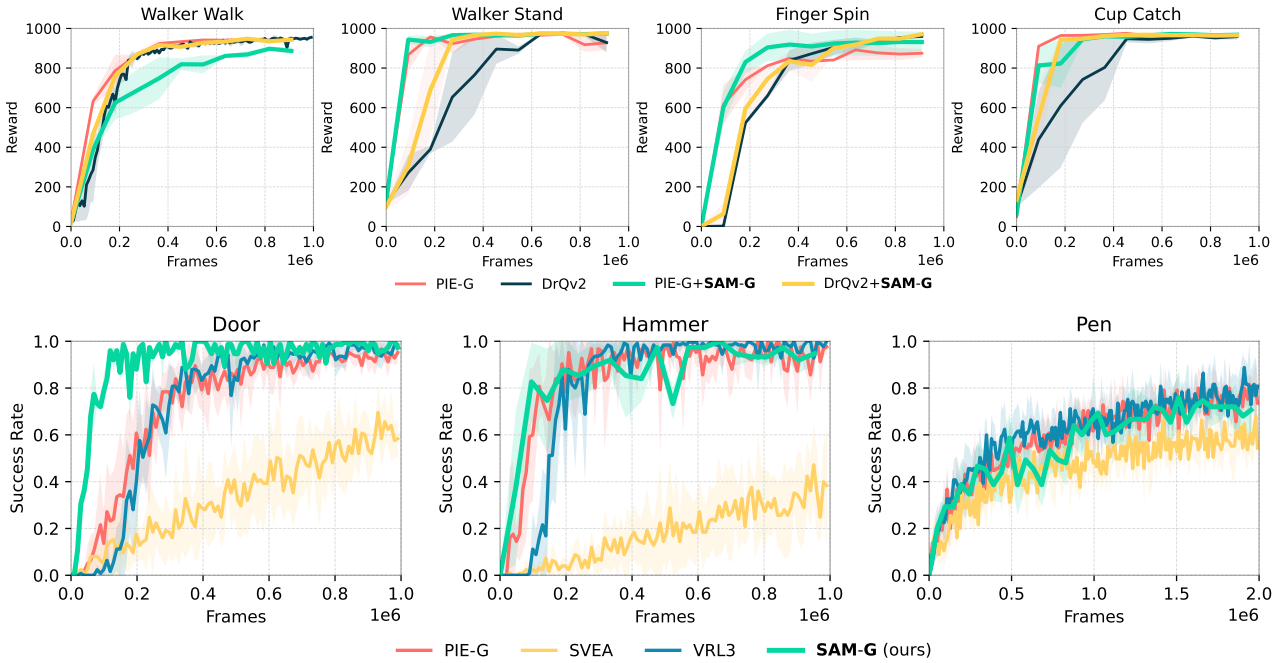

Figure 8. **Training curves on Adroit and DMControl.** Mean of 3 random seeds, shaded area is ± 1 std. We observe that SAM-G (PIE-G version) achieves better sample efficiency on Door and competitive results on Hammer and Pen compared to other strong baselines. On DMControl, SAM-G (both DrQ-v2 version and PIE-G version) achieves compared efficiency to other strong baselines.This indicates that applying masked images directly would not negatively impact the training performance.

ization capability of point features, as depicted in Figure 6. Our quantitative evaluation results further underscore the efficacy of incorporating DINOv2, particularly in the challenging *video hard* setting. Notably, when we exclude DINOv2, the average performance of SAM-G drops significantly, decreasing from 830 to 589.

Extra points. In addition to the automatically computed point features, we also include coordinates provided by humans for direct extraction of point features from image features, as detailed in Section 4.1. As illustrated in Table 4, these extra points play a pivotal role in certain tasks, particularly in Cartpole Swingup. Without the inclusion of extra points, the performance of SAM-G in the *video hard* setting declines from 747 to 491. This observation underscores two key insights: (1) the accurate identification of correspondence is of paramount importance, and (2) the *type 1* point features may sometimes fail to provide precise correspondence and our proposed extra points serve as a remedy to mitigate this issue.

Mask refinement. Similar to extra points, we note the significance of mask refinement particularly in specific tasks such as Finger Spin. While mask refinement generally helps, we have identified a nuanced scenario in the case of Ball_in_cup Catch, where mask refinement appears to have a slightly adverse effect. This could be possibly attributed to the characteristics of this task: we observe that the cup frequently moves to the corner of the image, where mask refinement might be even hurtful in identifying such cases.

PerSAM loss. SAM-G adopts the efficient parameter finetuning technique from PerSAM [36] which finetunes 2 learnable weights, for better adaptation to target tasks. We ablate the necessity of the usage. As shown in Table 4, we observe that the usage of this technique is helpful for SAM-G. From our observation in experiments, this technique helps to find a better understanding of our target object, reducing confusion about what foreground and background are.

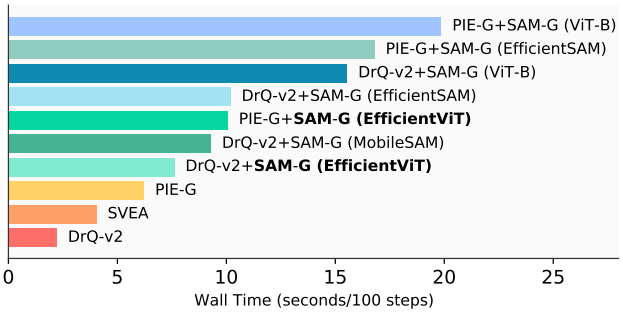

Figure 9. **Wall time** of different visual RL algorithms, tested on an NVIDIA A800 GPU on Walker Walk task. The usage of EfficientViT largely improves the inference speed, while it is still costly due to the usage of the large segmentation model.

Speedup inference by EfficientViT. The original ViT model from SAM imposes a large computational overhead when encoding images. To address this issue, we incorpo-

Table 5. **Ablation on different segmentation models.** We replace the EfficientViT [3] model in SAM-G with Mask-RCNN [12], SAM (ViT-B) [14], and EfficientSAM [24] respectively.

Ablations (color hard)	Walker walk	Cartpole swingup	Finger spin	Ball_in_cup catch	Average
EfficientViT	805±37	728±51	739±32	949±26	805
Mask-RCNN	566±124	318±294	487±79	448±163	455
SAM (ViT-B)	718±17	642±17	752±44	887±14	750
EfficientSAM	865±9	767±31	896±17	951±18	870
MobileSAM	906±22	582±156	844±83	617±128	737

DMControl (video easy)	Walker Walk	Cartpole Swingup	Finger Spin	Ball_in_cup Catch	Average
EfficientViT	887±33	855±6	937±21	858±79	884
Mask-RCNN	650±161	257±60	482±29	536±48	481
SAM (ViT-B)	820±27	624±8	940±34	966±5	838
EfficientSAM	869±25	824±10	959±18	951±16	901
MobileSAM	883±67	650±105	932±30	790±81	814

DMControl (video hard)	Walker Walk	Cartpole Swingup	Finger Spin	Ball_in_cup Catch	Average
EfficientViT	870±28	747±5	932±22	771±101	830
Mask-RCNN	62±20	150±30	37±5	100±41	87
SAM (ViT-B)	735±10	517±21	896±35	927±2	769
EfficientSAM	872±5	732±33	862±15	901±35	842
MobileSAM	867±57	543±95	858±24	534±84	701

rate the EfficientViT [3] architecture. We test the wall time of several visual RL algorithms for comparison, as shown in Figure 9. It is observed that the adoption of the EfficientViT architecture has made our method more practical, while SAM-G still exhibits a longer wall time when compared to PIE-G and SVEA. We hope that this overhead can be mitigated in the future through further advancements in model architecture and system optimization.

Different segmentation models. One key design of SAM-G is to apply SAM, the most powerful segmentation foundation model as far as we know. We have also tried other segmentation models for comparison, including a Mask R-CNN [12] pre-trained on COCO, a ViT-B version of original SAM, EfficientSAM [24], and MobileSAM [35]. EfficientSAM and MobileSAM are two distilled versions of SAM for faster inference. We use the EfficientSAM-S model.

As shown in Table 5, SAM-G that leverages an EfficientViT version of SAM largely surpasses that with Mask-RCNN. This is not surprising since Mask-RCNN can not make competitive segmentation results compared to SAM, as visualized in Figure 10. We also observe that the original SAM (ViT-B) works slightly worse than SAM-G, since SAM-G is using the Efficient ViT-L1 model, which is distilled from the more accurate ViT-H SAM model. EfficientSAM is also trained to distill the ViT-H model in SAM and it gets an even better result than SAM-G, while due to its larger wall time (see Figure 9), we apply EfficientViT in SAM-G. MobileSAM achieves similar wall time but is less accurate, compared to EfficientViT.

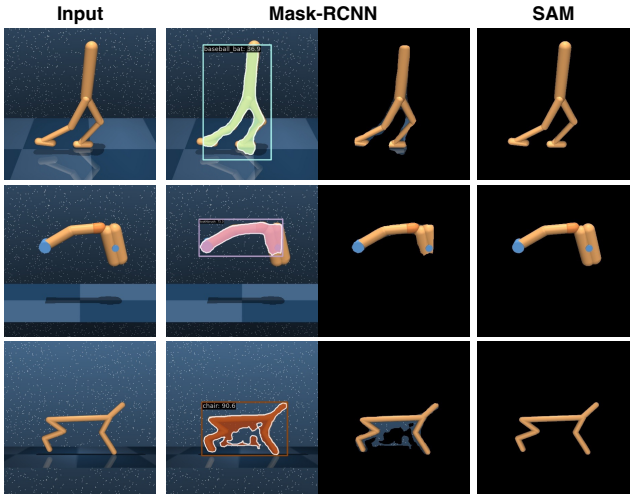

Figure 10. **Visualization of segmentation from Mask-RCNN and SAM for comparison.** Mask-RCNN only roughly detects the object in the center, while SAM produces high-quality fine-grained masks.

6. Conclusion

In this work, we introduce **S**egment **A**nything **M**odel for **G**eneralizable visual RL (**SAM-G**), a framework that harnesses vision foundation models to find correspondence and subsequently leverages Segment Anything Model (SAM) to segment out task-relevant objects for the benefit of visual reinforcement learning (RL) agents. The high-quality masked images produced by SAM are directly utilized by agents. We conduct evaluations of SAM-G on the generalization benchmark of 11 visual RL tasks and observe the robust generalization capabilities of SAM-G across settings of varying difficulty. Notably, in the most challenging setting, *video hard*, we achieve relative improvements of 44% and 29% across two domains when compared to the state-of-the-art method. Our work highlights the significance of leveraging vision foundation models for enhancing generalization in decision-making.

One limitation of our work is the extended wall time when employing SAM. To mitigate this issue, we have incorporated EfficientViT in this work, and we anticipate that future advancements in machine learning systems and architectures will further address this challenge.

Acknowledgement

This work is supported by National Key R&D Program of China (2022ZD0161700).

References

[1] Renee Baillargeon, Elizabeth S Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 1985. 1

[2] David Bertoin, Adil Zouitine, Mehdi Zouitine, and Emmanuel Rachelson. Look where you look! saliency-guided q-networks for generalization in visual reinforcement learning. *NeurIPS*, 2022. 1, 2, 5

[3] Han Cai, Chuang Gan, and Song Han. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. *arXiv*, 2022. 3, 9

[4] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv*, 2023. 2

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019. 1, 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020. 3

[8] Xiang Fu, Ge Yang, Pulkit Agrawal, and Tommi Jaakkola. Learning task informed abstractions. In *ICML*, 2021. 2

[9] Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *ICRA*, 2021. 1, 2, 5, 6

[10] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *NeurIPS*, 2021. 1, 2, 5

[11] Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. *ICML*, 2023. 1, 2

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 9

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 7

[14] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *ICCV*, 2023. 1, 2, 3, 4, 9

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv*, 2013. 1

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015. 1

[17] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv*, 2023. 1, 2, 3

[18] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv*, 2017. 2, 5, 1

[19] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Segment anything meets point tracking. *arXiv*, 2023. 2, 4

[20] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. *ICML*, 2021. 1

[21] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv*, 2018. 2, 5, 1

[22] Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *NeurIPS*, 2022. 5

[23] Xudong Wang, Long Lian, and Stella X Yu. Unsupervised visual attention and invariance for reinforcement learning. In *CVPR*, 2021. 2

[24] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. *arXiv*, 2023. 9

[25] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv*, 2021. 1

[26] Ruihan Yang, Ge Yang, and Xiaolong Wang. Neural volumetric memory for visual locomotion control. In *CVPR*, 2023. 1

[27] Sizhe Yang, Yanjie Ze, and Huazhe Xu. Movie: Visual model-based policy adaptation for view generalization. *arXiv*, 2023. 2

[28] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv*, 2021. 5, 6

[29] Zhecheng Yuan, Guozheng Ma, Yao Mu, Bo Xia, Bo Yuan, Xueqian Wang, Ping Luo, and Huazhe Xu. Don't touch what matters: Task-aware lipschitz data augmentation for visual reinforcement learning. *arXiv*, 2022. 2

[30] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. *NeurIPS*, 2022. 1, 2, 5, 6, 7

[31] Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, Xiaolong Wang, and Huazhe Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization. *arXiv*, 2023. 2, 5, 7, 1

[32] Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 2023. 1, 2

[33] Yanjie Ze, Yuyao Liu, Ruizhe Shi, Jiaxin Qin, Zhecheng Yuan, Jiashun Wang, and Huazhe Xu. H-index: Visual reinforcement learning with hand-informed representations for dexterous manipulation. *NeurIPS*, 2023. 1

[34] Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv*, 2018. 1, 2

[35] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv*, 2023. 9

[36] Renrui Zhang, Zhengkai Jiang, Ziyu Guo, Shilin Yan, Junting Pan, Hao Dong, Peng Gao, and Hongsheng Li. Personalize segment anything model with one shot. *arXiv preprint arXiv:2305.03048*, 2023. 2, 4, 8

[37] Chenyang Zhao, Olivier Sigaud, Freek Stulp, and Timothy M Hospedales. Investigating generalisation in continuous deep reinforcement learning. *arXiv*, 2019. 1, 2

Generalizable Visual Reinforcement Learning with Segment Anything Model

Supplementary Material

A. Implementation Details

In this section, we describe more implementation details of SAM-G, mainly describing how to obtain embeddings, point features, and SAM prompts. An overview of SAM-G has been shown in Figure 2. Our official implementation is available at https://github.com/wadiuvatzy/SAM-G.

Obtain feature embeddings. To get the point prompts, we first extract feature embeddings from the visual encoder of SAM and DINOv2, and the resulting embedding is used to compute the similarity map. Taking the image observation of size $84 \times 84 \times 3$ as input, the encoder of SAM resizes the image into $1024 \times 1024 \times 3$ and obtains an embedding of size $64 \times 64 \times 256$. Similarly, the encoder of DINOv2 resizes the image into $448 \times 448 \times 3$ and obtains an embedding of size $32 \times 32 \times 768$. The DINOv2 embedding is resized to $64 \times 64 \times 768$ for alignment.

Obtain point features. Given a pair of image and its mask from the training environment, we obtain 2 types of target point feature as shown in Figure 5. *Type 1* point features are computed on the masked embeddings using spatial average and max operations. *Type 2* point features are fetched directly from original embeddings coordinated by human-given points. We observe that *type 2* point features are more necessary for multi-object scenarios and confusing backgrounds, and for easier settings, *type 2* point features are shown to be not very necessary.

Prompt for segmentation. Once we prepare the point features, we are ready to segment images. Take the case with 1 *type 1* point feature and 3 *type 2* point features as an example. For an image we want to segment, we first get the feature embeddings through the method discussed above. And then we will use *mask decoder* of SAM 3 times to get the final segmentation result.

• **1st segmentation.** for each point feature, calculating the normalized inner product with the feature embedding gives us two 64×64 similarity maps (one from SAM feature and the other from DINOv2 feature). By taking the mean of each pair of two similarity maps, we get 4 final similarity maps: one from *type 1* point feature and three from *type 2* point feature. All the 4 similarity maps give us a positive point prompt by finding the maxima in each map, while only the similarity map from *type 1* point feature gets the minima as a negative point prompt. We do not use *type 2* similarity map to get negative point prompts because *type 2* point feature only contains local information, the minima may lie on another object we want. Feeding those 5 point prompts together with the feature embedding got before to *mask decoder* of SAM is the final step to get the first result.

• **2nd and 3rd segmentation**. After the first segmentation, we can get 3 masks (multimask return by SAM) together with their mask logits. Conducting a weighted summation of those mask logits with our trained 3 weights gives us a rough mask whose bounding box is also easily got. We can find the minima of *type 1* similarity map inside the box and take that as another negative point prompt (may be the same point as the previous negative prompt). Further we will feed those 6 point prompts together with the box, the logits into *mask decoder* of SAM to get the 2nd result. For the last time of segmentation, repeating similar process in 2nd segmentation, we can get a bounding box of a rough mask, a mask logits and another negative prompt. Once again, feeding 7 point prompt, 1 box and 1 mask logits to the *mask decoder* of SAM gives out 3 masks. We choose the one with highest score (given by SAM) as our final segmentation result.

B. Descriptions of Tasks

Our generalization setting follows [9, 31], where the *color* setting changes the background, object color and table texture, while the *video* setting changes the background to a natural video and introduces moving light. Compared with *color easy* setting, *color hard* setting presents a higher level of randomness and complexity, with more varied and unpredictable visual features. Similarly, *video easy* setting has simpler video background dynamics, while *video hard* setting consists of intricate and swiftly alternating video backgrounds which are substantially dissimilar to the training setting.

We describe the tasks used in this work as follows, mainly from DMControl [21] and Adroit [18].

- *Walker, walk* ($\mathbf{a} \in \mathbb{R}^6$). A planar walker that is rewarded for walking forward successfully at a set speed. Dense rewards.
- *Walker, stand* ($\mathbf{a} \in \mathbb{R}^6$). A planar walker that is rewarded for maintaining a vertical position and a steady height above a specified minimum. Dense rewards.
- *Cartpole, swingup*($\mathbf{a} \in \mathbb{R}$). Swing up and stabilize a free-standing rod by exerting forces on a cart at the foundation. The agent is rewarded for keeping the rod aligned within a set angular limit. Dense rewards. Dense rewards.
- *Cheetah, run* ($\mathbf{a} \in \mathbb{R}^6$). A planar cheetah model that is rewarded for sprinting forward rapidly, with the objective of achieving and maintaining a high velocity. Dense rewards.
- *Hopper, stand* ($\mathbf{a} \in \mathbb{R}^4$). A one-legged planar hopper that is rewarded for achieving and maintaining an upright

Table 6. **Hyperparameters** for SAM-G.

Hyperparameter	DMControl-GB	Adroit
Input size	84×84	84×84 (door, hammer), 160×160 (pen)
Discount factor γ	0.99	0.99
Action repeat	2	2
Frame stack	3	3
Learning rate	1e-3	1e-4
Random shifting padding	4	4
Episode length	1000	200 (door, hammer), 100 (pen)
Evaluation episodes	100	100
Batch size	128	256
Replay buffer size	5e5	1e6
Optimizer	Adam	Adam

Table 7. **Task parameters** for DMControl and Adroit tasks.

Task	Action Dim	Action Repeat	# Objects	# Extra points	# Frames
Walker Walk	6	2	1	3	1M
Walker Stand	6	2	1	3	1M
Cartpole Swingup	1	2	1	3	1M
Ball_in_cup Catch	2	2	2	6	1M
Hopper Stand	4	2	1	3	1M
Finger Spin	2	2	1	3	1M
Cheetah Run	6	2	1	3	3M
Quadruped Walk	12	2	1	3	3M
Door	28	2	2	6	1M
Hammer	26	2	3	9	1M
Pen	24	2	2	6	2M

position, with its torso held vertically to a minimal height. Dense rewards.

- *Finger, spin* ($\mathbf{a} \in \mathbb{R}^2$). A planar robotic finger that is rewarded for spinning a body affixed to a surface, with the objective of achieving and maintaining a continuous rotational velocity. Dense rewards.
- *Ball_in_cup, catch* ($\mathbf{a} \in \mathbb{R}^2$). A motorized planar container that is rewarded for oscillating and catching a ball tethered to its base by a string. Sparse rewards.
- *Quadruped, walk* ($\mathbf{a} \in \mathbb{R}^{12}$). A four-legged robotic entity that is rewarded for ambulating forward, aiming to achieve a targeted gait and speed. Dense rewards.
- *Door* ($\mathbf{a} \in \mathbb{R}^{28}$). The task to be completed consists on unlocking the door and swing the door. The task is considered complete when the door touches the door stopper. Sparse rewards.
- *Pen* ($\mathbf{a} \in \mathbb{R}^{24}$). The task to be completed consists on repositioning the blue pen to match the orientation at the green target pen which is randomly chosen from all configurations. The task is considered complete when the orientations match with some tolerance. Sparse rewards.
- *Hammer* ($\mathbf{a} \in \mathbb{R}^{26}$). The task to be completed consists on picking up a hammer and drive a randomly positioned nail into a board. The task is considered complete when the neil is entirely in the board. Sparse rewards.

C. Hyperparameters

Our algorithms are mainly based on DrQ-v2 and PIE-G and most of the training hyperparameters are the same. Table 6 shows detailed hyperparameters. Task-relevant parameters are in Table 7.

D. More Ablations

D.1. Higher Resolution for Pen

In Table 2, we mentioned that SAM-G falls short of performance in the Pen task in 84×84 resolution. The detailed evalation result can be found in Table 8. We postulate that the unsatisfying performance of SAM-G in the Pen task may be attributed to the low image resolution. In the Pen task, the dexterous hand must execute precise manipulation, and the challenges in segmenting objects become pronounced in low-resolution images. To substantiate our hypothesis and enhance SAM-G' performance, we increase the input image resolution from 84×84 to 160×160 for

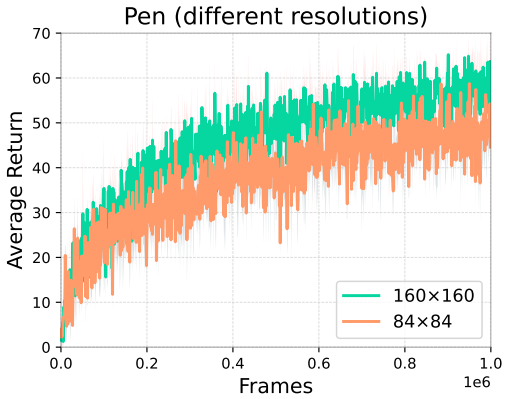

Figure 11. **Training curves in different resolution in Pen.** Mean of 3 random seeds, shaded area is ± 1 std. Training in 160×160 input get better performance than training in 84×84 for SAM-G.

(a) 84×84 (b) 160×160

Figure 12. **Visualization of segmentation under different resolutions.** SAM segments more accurately under a higher resolution.

Table 8. **The low resolution result for pen.** All the resolution of the visual observation are 84×84.

Pen high resolution	Color easy	Color hard	Video easy	Video hard	**Average**
VRL3	0.1±0.0	0.1±0.0	0.0±0.0	3.6±0.9	1.0
SVEA	55.0±8.6	47.0±6.8	50.2±8.6	46.8±9.7	49.8
SGQN	51.4±18.1	36.8±13.7	51.4±18.1	54.0±3.7	48.4
PIE-G	70.6±2.9	57.4±4.2	67.0±8.8	56.6±6.3	62.9
SAM-G	60.0±12.8	53.0±5.0	61.3±9.3	55.3±4.2	57.4

improved segmentation. The training progress is shown in Figure 11, and the final corresponding evaluation results are presented in Table 2. We also compare the segmentation results in Figure 12.

Notably, the success rate in the *train* setting has substantially improved, rising from 69% in the low-resolution setting to an impressive 82% in the high-resolution setting. Moreover, with the adoption of this higher resolution, SAM-G has now significantly outperformed all other baselines, providing strong validation for our initial hypothesis.