Imagined Subgoals for Hierarchical Goal-Conditioned Policies

Xuhui Kang, Wenqian Ye, Yen-Ling Kuo

***Abstract*— Humans form mental images of tasks being performed and their expected outcomes to support how they act in the world. We present a novel framework that generates visual subgoals based on the goal image and the robot's current state. The generated visual subgoals are used to guide the rollout of the policy. The subgoal generator is jointly trained with a progress encoder to track the progress of a task while constructing the next visual subgoals. At inference time, the robot adaptively samples a new visual subgoal from the generator once the current subgoal is achieved, or if the maximum step limit is reached without achieving the subgoal. We train our models and validate the proposed framework using the CALVIN manipulation dataset, demonstrating the improved success rate in various tasks.**

I. Introduction

Teaching robots to plan a task usually involves breaking down a task into incremental steps or solvable subtasks. This requires an understanding of the initial state, the key steps for completing a task, and the possible configurations of the key steps. This decomposition is not only needed for complex tasks but also for a single task "sliding the door"(Fig. 1). In order to complete the task successfully, the robot needs to reason about 1) *moving closer to the handle of the sliding window*, 2) *grasping the handle*, and 3) *moving it to the left*.

Several hierarchical methods have been proposed to enable robots to plan their actions effectively, e.g., task and motion planning [1], options in reinforcement learning [2], and learning hierarchical policies [3]. While this division of high-level and low-level policies helps learn skills, designing abstractions for hierarchical policies is still very challenging. Prior work has been exploring using latent embeddings [4], synthesized programs [5], or language [6] as abstractions for the high-level policies. On the other hand, humans and animals have demonstrated the ability to leverage mental imagery for planning [7–9]. Mental imagery enables them to plan efficiently by retrieving relevant memories from previous experiences and building representations of environments. In this paper, we propose to give this ability to robots by generating visual subgoals that guide the robot to follow a task specified in the goal image.

Taking the notion of using mental imagery as a method for high-level planning, we propose a framework, as illustrated in Fig. 2, which guides the rollout of the low-level policy using the generated visual subgoals. We cotrain the subgoal generator with the progress encoder to predict the image of the next subgoal a robot needs to reach given the image goal, current state, and the tracked progress. By leveraging the recent advances in conditional diffusion models [10, 11], we can generate realistic subgoal images to condition the low-level policy. We train and test our models on the CALVIN benchmark [12], a dataset with diverse tabletop manipulation tasks. We show that the generated subgoals guide the progression of tasks and can help with task completion.

Department of Computer Science, University of Virginia. {qhv6ku, wenqian, ylkuo}@virginia.edu.

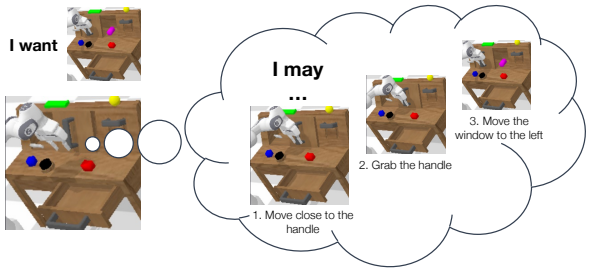

Fig. 1. An illustration of the idea that a robot can generate the visual subgoals incrementally (shown as the increasing numbers) for a task specified by an image goal. The generated subgoals reveal that the robot needs to understand the preconditions and steps, e.g., moving close to the handle, grabbing the handle, and sliding the window to the left.

This work makes the following contributions:

1) We present a novel framework that leverages the generated visual subgoals for goal-conditioned policies.
2) We show that the generated visual subgoals follow the progress of tasks.
3) We demonstrate that the generated subgoals can guide the rollout of the low-level policy and improve the success rate of tasks.

II. Related Work

Goal-Conditioned and Hierarchical Policy Learning Goal-conditioned policies enable robots to carry out actions that achieve a specified goal. One way to provide goal descriptions is to use images showing the desired outcome. They've aided in tasks like object placement [13] and offline learning [14, 15]. To learn to follow these visual goals effectively with unlabelled demonstrations, techniques like relabeling [16] are usually used in learning a goal-conditioned policy. It can combine with hierarchical methods to structure policies in layers: a low-level policy handles actions while a high-level one guides task-solving. [17–19]

Planning with Diffusion Models Diffusion models, particularly in image generation, have showcased their prowess in approximating data distributions [10, 11]. In robotics and reinforcement learning, they have been harnessed to view policy formulation as a generative act [20–24]. Approaches such as decision diffuser [21] and LCD [21] add layers of complexity, catering to constraints or integrating language cues. In contrast to these approaches, our method employs

diffusion models for proposing visual subgoals in a hierarchical policy structure. This facilitates an understanding and enhancement of policies.

III. Goal-Conditioned Policy with Imagined Subgoals

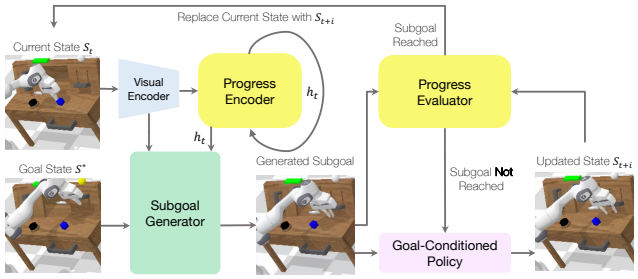

Fig. 2. An overview of the goal-conditioned policy with generated visual subgoals as its conditions to rollout the low-level policy.

A. Problem Formulation

Our aim is to develop a generative model $\mathcal{G}$ that generates the next visual subgoal f_{gen} for a task represented as an image goal s_g. For a sequence of generated images $\{f_{gen}\}$, it needs to be a sequence of subgoals to reach the specified image goal. The generative model is conditioned on the subgoals the robot has reached so far $s_0 \dots s_t$ and the corresponding image goal s^*. The objective is to learn a generative model that minimizes the distances between the generated visual subgoals and the corresponding ground-truth subgoals τ_t sampled from the trajectory $\mathcal{T}_{0:T}$ collected from demonstrations:

$$\min_{\theta} \mathbb{E}_{\tau_t \sim \mathcal{T}_{0:T}}[dist(\mathcal{G}_\theta(f_{gen} \mid s_0, \dots, s_t, s^*), \tau_t)] \quad (1)$$

Given the generated visual subgoal f_{gen}, we aim to have a low-level policy $\pi_\psi(a_t|s_t, f_{gen})$ that generates actions to achieve the subgoal f_{gen}. This policy can be learned from demonstrations.

B. Framework Overview

An overview of our proposed framework is depicted in Fig. 2. It consists of two stages: 1) visual subgoal generation and 2) a subgoal-conditioned policy. During the first stage, we generate subgoals based on the image of the initial observation and an embedding of the current progress from the progress encoder. The progress encoder can be a recurrent model that takes the subgoals observed so far as input to generate an embedding that encapsulates the current task progress (see Section III-C). These generated subgoals serve as intermediate objectives for the robot to achieve. In the second stage, the framework utilizes these generated subgoals to condition the low-level policy of the robot. Since we generate visual subgoals autoregressively, we will roll out the low-level policy until it reaches the current subgoal and use this updated state to generate the next subgoal. Ideally, the low-level policy π_ψ should predict the termination. However, as the generated visual subgoals may not perfectly match the ground-truth subgoals, it is hard to learn the termination for the generated subgoal from offline demonstrations. To accommodate this issue, we employ a progress evaluator to determine if the generated subgoal is reached or the low-level policy doesn't make progress so we need to generate the next visual subgoal for policy rollout (see Section III-D). Once the progress evaluator decides to advance to the next subgoal, the input of the subgoal generator is replaced with the updated state s_{t+i}. This update initiates a new round of subgoal generation, allowing the robot to proceed with a new round of subgoal execution.

C. Visual Subgoal Generation

1) Subgoal Generator: The subgoal generator is a conditional generative model. We implement this generator by combining the 2D latent diffusion model [25] and ControlNet as in [11] to ensure the generation follows the conditions:

- The current state s_t. This condition informs the generator to preserve the existing objects and the environment.
- The goal state s^*. This condition shows how an object should be manipulated.
- The progress embedding h_t. This condition adds time-relevant constraints and provides task-specific information to enable the generator to interpolate between the current and the goal state.

The image of the current state s_t is input to the ControlNet branch so the generated image includes objects in the current state. The goal image is encoded by the visual encoder and then an MLP. This embedding of the goal state is fused with the progress embedding h_t and the diffusion timestep to condition on each level of the U-Net. Similar to the language conditions in the original diffusion model, conditioning on goal and progress on each level allows us to generate images that match the next subgoal.

2) Progress Encoder: The progress encoder is a recurrent network that takes the image of the current state s_t and the last hidden state h_{t-1} of the progress encoder to update its hidden state. The output hidden state h_t is the progress embedding used in the subgoal generator. This progress encoder keeps track of the subgoals achieved so far and can provide guidance on the next subgoal.

D. Subgoal Conditioned Policy

1) Policy Network: A policy network is used to predict actions that achieve the generated subgoals. We parameterize the policy as a discretized logistic mixture distribution [26]. The policy network takes the current state s_t and the generated subgoal f_{gen} to decode into a mixture of logistic distributions where each of these distributions has separate means and scales and is weighted using a parameter α.

2) Progress Evaluator: The progress evaluator assesses the updated current state by comparing its representation with the generated visual subgoal. If the subgoal is nearly achieved or if it remains unachieved after a significant amount of time, the updated state is then fed into a new round of subgoal generation. The evaluation of progress is

to extract features from the current frame and the goal frame, s_i and s^*. We leverage a feature extraction model $\mathcal{M}$ such as R3M [27] to get the feature and compute the distance as $\|\mathcal{M}(s_i) - \mathcal{M}(s_j)\|_2$. If the distance is smaller than δ or the step for the current subgoal is larger than λ, it will move forward to the next subgoal. This process ensures continuous progress toward the achievement of the overall goal.

E. Training Pipeline

We train the subgoal generator and the goal-conditioned policy separately.

1) Training the Subgoal Generator: We select a sequence of ground-truth subgoals from demonstrations. To identify ground-truth subgoals from a demonstration, we utilize the R3M embedding [27] to compute the distances between the start frame and all frames in the demonstration. We observe that subgoals are more likely to be the frames whose distances to the start frame are relatively stable. So we select the frames where distance change is smaller than δ to be subgoals. Once the ground-truth subgoals are identified, we fine-tune the diffusion model with these ground-truth subgoal images. This fine-tuning process allows us to generate images that match the style of the environment. Finally, we initialize ControlNet with the weights obtained from the fine-tuned diffusion model and then co-train it with the progress encoder. This co-training process enables us to jointly learn the task progress and accurately predict the next visual subgoal in the sequence. To ensure the successful prediction of the next subgoal, we adopt the MSE loss, $\mathcal{L}_{gen}$, to compare the generated subgoal, f_{gen} and the ground-truth subgoal, f_{gt}:

$$\mathcal{L}_{gen}(f_{gen}, f_{gt}) = \sum (f_{gen} - f_{gt})^2 \tag{2}$$

2) Training the Goal-conditioned Policy: To train the goal-conditioned policy, we subsample trajectories of different lengths from a demonstration dataset D_{play}. The training objective is to predict the actions a_t taken in the demonstration. We define the imitation learning loss $\mathcal{L}_{IL}$ as follows:

$$\mathcal{L}_{IL} = \mathbb{E}_{(\tau, s_g) \sim D_{\text{play}}} [\sum_{t=0}^{|\tau|} \log \pi_\theta(a_t \mid s_t, s^*)] \tag{3}$$

The loss function is calculated over trajectories τ and goal states s^* drawn from the dataset D_{play}. It aggregates the log probabilities of actions a_t taken by the policy π_θ at each state s_t in the trajectory τ, conditioned on the respective goal s^*. The aim of this loss is to improve the alignment of the policy with the observed behaviors, guiding π_θ to better imitate the actions in the dataset.

IV. Experiments

A. Dataset

We train and test our proposed framework using the CALVIN benchmark [12]. Similar to prior works [17, 28, 18], we select Task D for evaluation, which consists of 34 different tabletop manipulation tasks. The training set contains more than 5,000 trajectories. In the validation dataset of Task D, each task boasts approximately 30 rollouts.

B. Hyperparameters

We use the CLIP [29] image encoder as our visual encoder. For the progress encoder, we use GRU [30] with two layers and a feature size of 768, matching the output size of the visual encoder.

In the progress evaluator, we use R3M to compute the distance between the current observation and the generated subgoals. If the distance is smaller than 0.3, we consider the robot reaches the subgoal. For each subgoal, we set a max step to 20 to avoid it stuck in a subgoal.

We used 4 40GB A100 GPUs to train the subgoal generator. We train with batch size 16 and at least 500k training steps to get reasonable subgoal images.

C. Experiment Setup

1) Baselines and Goal-Conditioned Policy: We consider two goal-conditioned policies for comparison.

- TACO-RL [19]: An offline reinforcement learning method that uses visual goals to guide its policy.
- Image-HULC [18]: Originally HULC was a language-goal-conditioned model learned from imitation learning. We adapted HULC to learn an image-goal-conditioned policy by replacing its language-goal encoder with a visual-goal encoder.
- D-GCBC [31]: A diffusion model based goal-conditioned policy. We follow the implementation in [32] to stack the current observation and the goal image, encode them by ResNet-50, and then use this embedding to condition a diffusion process to generate action distribution. Following [24], we predict four action sequences for temporal consistency.

The baselines are TACO-RL, Image-HULC, and D-GCBC conditioned on only the final image goal. While our method also employs the same set of low-level policies, we will use our generated visual subgoals to guide the policy rollout. For both TACO-RL and Image-HULC, We directly use the pretrained weights for CALVIN Task D released by the original authors. For the D-GCBC model, we trained it on the CALVIN Task D datasets, running 100,000 steps with a batch size of 256.

2) Training Generative Model: During our experiments, we found that initializing the model with pretrained weights greatly improves training speed and quality. Without using these pretrained weights, the model might not converge at all. So, we first initial the diffusion model with the weights from [33]. Once we train a diffusion model that is able to generate reasonable images, we use its weights to initialize the ControlNet. In the original ControlNet, only the ControlNet part needs training while the diffusion model part remains frozen. However, our training experience showed that training both parts together produces better results in the experiments.

D. Experimental Results

1) Impact of Generated Subgoals: Fig. 3 presents a comparison between using the ground-truth final goal only and our generated subgoals in three different goal-conditioned

Task\Method	TACO-RL		Image-HULC		D-GCBC	
	Final Goal only	**Ours**	Final Goal only	**Ours**	Final Goal only	**Ours**
turn_on_led	57.14 ± 2.42	77.38 ± 8.33	92.85 ± 1.79	95.24 ± 2.38	$\mathbf{96.43} \pm 3.57$	92.85 ± 7.15
move_slider_left	37.50 ± 0.00	71.88 ± 12.50	34.38 ± 3.13	46.88 ± 18.75	46.88 ± 0.00	$\mathbf{96.88} \pm 0.00$
push_pink_block_right	11.54 ± 0.00	26.92 ± 4.60	57.69 ± 3.85	64.10 ± 10.25	61.54 ± 3.85	$\mathbf{73.08} \pm 3.84$
unstack_block	31.03 ± 6.89	51.72 ± 6.89	72.41 ± 10.34	87.35 ± 4.60	86.20 ± 6.90	$\mathbf{89.66} \pm 3.45$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Average of 34 Tasks	19.61 ± 0.17	26.51 ± 1.27	49.06 ± 0.83	53.32 ± 1.07	49.37 ± 1.20	$\mathbf{75.56} \pm 0.34$

Fig. 3. Success rate (%) for TACO-RL, Image-HULC, and D-GCBC, w/ and w/o the generated subgoals. The results are based on three random seeds.

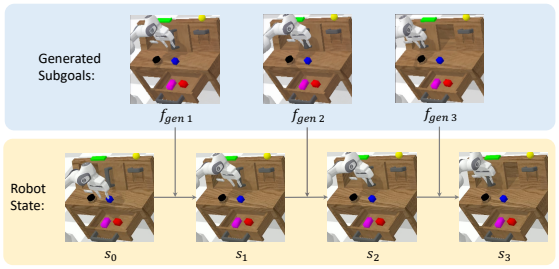

Fig. 4. An example demonstrating how the generated subgoals guide the robot to perform a task: sliding the window to the left. The first generated subgoal, $f_{gen,1}$, is to get close to the handle, and by time $t = 30$, this subgoal is achieved. The next subgoal, $f_{gen,2}$, is to grasp the handle, and by time $t = 34$, the robot successfully grasps it. The final subgoal, $f_{gen,3}$, directs the robot to successfully slide the window to the left.

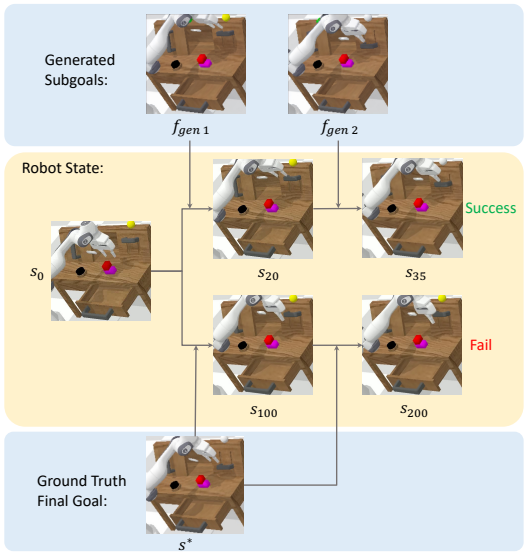

Fig. 5. For the task "slide down the switch," we find that the task fails if we only use the final goal. This is because the robot's gripper is not raised high enough to push the switch down. However, with the generated subgoals, the first goal is to raise the robot arm higher than the gripper, and thus, lead to successfully turn off the switch.

policy networks. In the bottom row, we average performance across 34 tasks. We show that our proposed framework can apply to any goal-conditioned policies to improve their performance. There is a noticeable boost in performance for the runs with the generated subgoals. In addition to the average performance, we additionally present 4 tasks as examples to show that the improvement of some tasks can be significant. For example, for task turn_on_led task, TACO-RL can only achieve a 57.14% success rate when running with a ground-truth final goal only. After using the generated subgoals, the success rate raised to 77.38%. Even in Image-HULC and D-GCBC, adding subgoals resulted in improved performance, e.g., task move_slider_left moves from 34.38% to 46.88% in Image-HULC and from 46.88% to 96.88% in D-GCBC.

However, some tasks such as lift_blue_block_drawer have 0% success rate in TACO-RL no matter if we consider the final goal only or generated subgoals. This is why the average improvement is not as big as individual tasks. In these cases, we find it necessary to improve the goal-conditioned policy first so it is possible to leverage our generated subgoals to improve task completion.

2) Qualitative Examples: Fig. 5 shows an example that without the subgoal, the task fails. Fig. 4 shows an example of how the generated subgoals guide the robot to achieve the subgoals one by one and the final goal. These examples follow the subgoal almost perfectly. However, we also have some failure cases in Appendix C. Our analysis shows that the failure cases are either issues in subgoal generation or low-level policy. For instance, in Fig. 7, the task was to "rotate blue block to the left," but a generated subgoal showed the block being dropped. However, more frequently, the problem lies in the policy itself. For example, in Fig. 8, the task is "rotate pink block left," the generated subgoal is correct, but the policy cannot guide the robot to reach the specified correct subgoal image.

V. Conclusion & Future Work

We have demonstrated that it is possible to generate visual subgoals to guide the rollout of policies. Instead of learning a policy that needs to reason about steps and preconditions, we lift this part of the reasoning to the subgoal generator to break down a task into smaller steps. Our experiments in simulated tabletop manipulation tasks showed promising results. However, our analysis showed that the low-level policy contributes to most failure cases. In the future, we may consider how to leverage the generated subgoals or cotrain subgoal generation and the policy network to improve the low-level policy. Experiments with real robots and more diverse tasks will help us understand how the proposed method can be applied to more complex scenarios.

References

[1] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[2] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[3] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.

[4] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.

[5] S.-H. Sun, T.-L. Wu, and J. J. Lim, "Program guided agent," in *International Conference on Learning Representations*, 2019.

[6] Y. Jiang, S. S. Gu, K. P. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] A. D. Redish, *Beyond the cognitive map: from place cells to episodic memory*. MIT press, 1999.

[8] L. H. Phillips, V. Wynn, S. McPherson, and K. Gilhooly, "Mental planning and the tower of london task," *The Quarterly Journal of Experimental Psychology Section A*, vol. 54, no. 2, pp. 579–597, 2001.

[9] A. Bocchi, M. Carrieri, S. Lancia, V. Quaresima, and L. Piccardi, "The key of the maze: The role of mental imagery and cognitive flexibility in navigational planning," *Neuroscience Letters*, vol. 651, pp. 146–150, 2017.

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 6840–6851.

[11] L. Zhang and M. Agrawala, "Adding conditional control to text-to-image diffusion models," *arXiv preprint arXiv:2302.05543*, 2023.

[12] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.

[13] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4568–4575.

[14] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*. PMLR, 2020, pp. 1113–1132.

[15] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard, "Latent plans for task-agnostic offline reinforcement learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1838–1849.

[16] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," *arXiv preprint arXiv:1912.06088*, 2019.

[17] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[18] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 11 205–11 212, 2022.

[19] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard, "Latent plans for task agnostic offline reinforcement learning," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[20] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9902–9915.

[21] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision-making?" *arXiv preprint arXiv:2211.15657*, 2022.

[22] E. Zhang, Y. Lu, W. Wang, and A. Zhang, "Language control diffusion: Efficiently scaling through space, time, and tasks," *arXiv*, 2023.

[23] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine, "Idql: Implicit q-learning as an actor-critic method with diffusion policies," 2023.

[24] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," 2023.

[25] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[26] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," *arXiv preprint arXiv:1701.05517*, 2017.

[27] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 892–909.

[28] E. Zhang, Y. Lu, W. Wang, and A. Zhang, "Lad: Language control diffusion: efficiently scaling through space, time, and tasks," *arXiv preprint arXiv:2210.15629*, 2023.

[29] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.
[30] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.
[31] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine, "Bridgedata v2: A dataset for robot learning at scale," 2023.
[32] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine, "Zero-shot robotic manipulation with pretrained image-editing diffusion models," 2023.
[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *CoRR*, vol. abs/2112.10752, 2021. [Online]. Available: https://arxiv.org/abs/2112.10752

APPENDIX

A. Ground-truth Subgoals Selection

In human cognition, we naturally visualize tasks and anticipate outcomes. For robots, an analogous mechanism is what we want in the form of visual subgoals. However, when human-labeled ground-truth subgoals aren't available – which would be the ideal dataset – an algorithmic approach to selecting the ground-truth subgoals becomes crucial, that's where our ground-truth subgoals selection algorithm comes in. It aims to replicate the human process of selecting significant visual moments in a sequence. The algorithm initiates with the video's first frame and computes the 'difference' between consecutive frames employing a distance evaluation model. To extract features from two frames, s_i and s_j, we leverage a feature extraction model $\mathcal{M}$ such as R3M [27]. The l_2 norm of the difference between s_i and s_j is then calculated as $\|\mathcal{M}(s_i) - \mathcal{M}(s_j)\|_2$. Upon plotting this computed distance, the data is smoothed to recognize patterns more clearly. The algorithm then identifies subgoals at points where the distance gradient lower than a threshold δ approaches zero. In this experiment, we choose $\delta = 0.02$ and minimal subgoals interval $\lambda = 7$. From the two examples in Fig. 6, it can be observed that the selected ground-truth subgoals align with human intuition, demonstrating their validity.

Algorithm 1 Ground-truth Subgoals Selection Algorithm

Input: Frame sequence $F = \{f_1, f_2, \ldots, f_n\}$, Gradient threshold δ, Minimal keyframe interval λ, Distance evaluation function $Dist(\cdot, \cdot)$

Output: Keyframe sequence K

Set initial keyframe $f_{key} = f_1, key = 1$
Initialize keyframe sequence $K = \{\}$
Initialize distance sequence $D = \{\}$
for each frame $f_i \in F$ **do**
    Calculate frame distance: $d_i = Dist(f_1, f_i)$
    $D = D \cup \{d_i\}$
Smooth D to get $D' = \{d'_1, d'_2, \ldots, d'_n\}$
for $i = 1 \rightarrow n$ and $key \leq n$ **do**
    if gradient$(d'_i) < \delta$ and $i - key > \lambda$ and $i + \lambda <= n$ **then**
        $K = K \cup \{f_i\}$
        Update $f_{\text{key}} = f_i, key = i$
return K

B. Example of Selected Subgoals

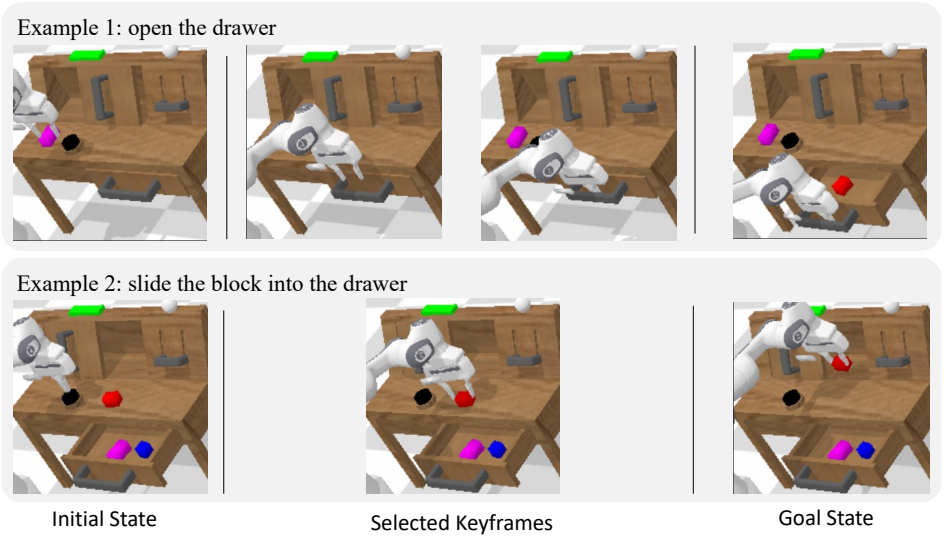

Fig. 6. Selected subgoals and their initial and goal statuses for two different tasks: Example 1 illustrates the subgoals chosen for the task "open the drawer," while Example 2 demonstrates the selected subgoals for the task "slide the block into the drawer."

C. More Running Cases

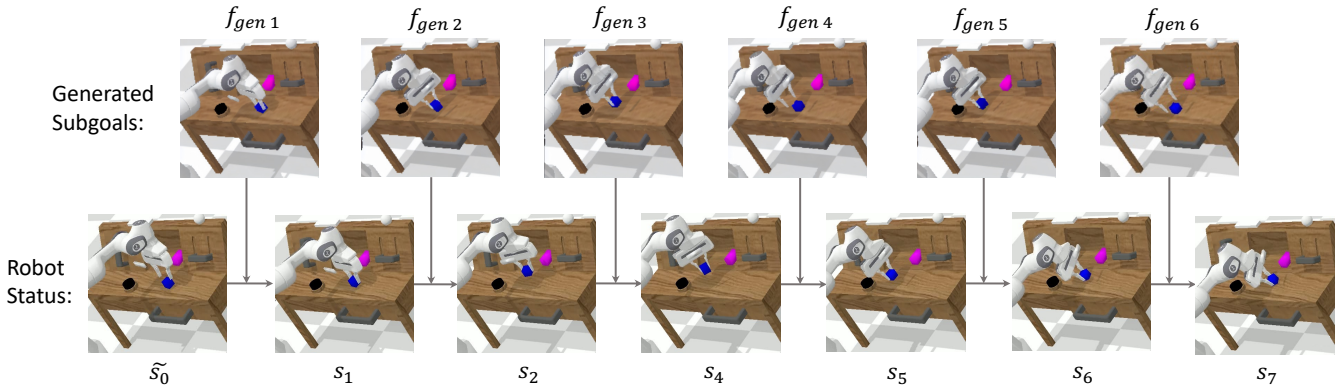

Fig. 7. Success case of the rollout on the task "rotate the blue block left": although f_{gen4} is not precisely generated (dropped the blue block), it can still provide valuable directional guidance for the robot's subsequent movements toward achieving the final goal.

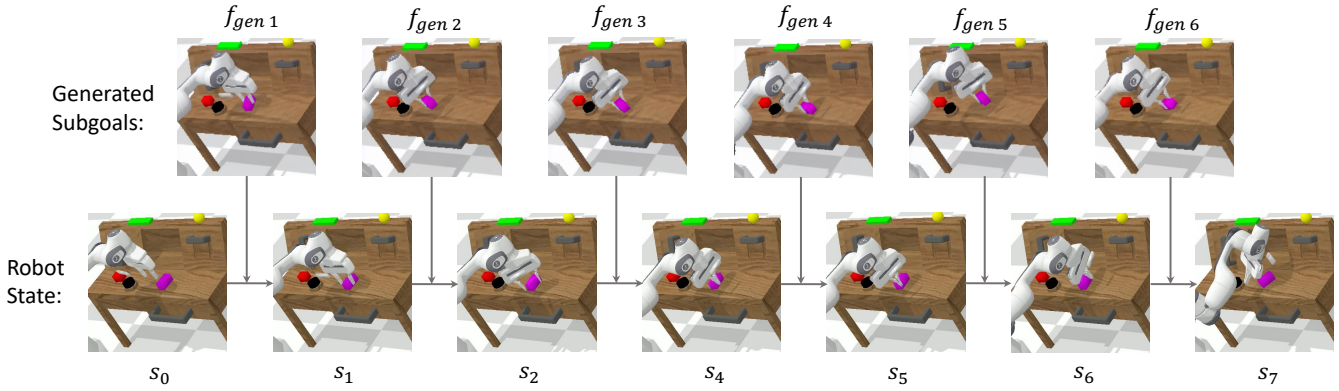

Fig. 8. Failure case of the "rotate the pink block left" task rollout: The policy fails to adapt to the changes in the generated subgoals, resulting in a series of failures.