Appendix

A Details on UVD

1. Decomposition

As we show in Algo. 1, UVD preprocesses the demonstration or user-provided raw video offline. We further provide low-level pseudocode in Python of UVD in Pseudocode. A.1. In practice, when identifying the temporally nearest observation where the monotonicity condition is not met, as per Eq. 3, it is equivalent to locating the most recent local maximum of the embedding distance curves. This is because the distance curve is anticipated to show an almost monotonically decreasing trend towards the final frame in each recursive iteration, as shown in Eq. 2. Due to the small-scale noise in pixel and high-dimensional feature space, we apply Nadaraya-Watson Kernel Regression [55] to first smooth the embedding curves before calculating the embedding distances. We benchmark our UVD implementation runtime in Appendix. A4., which shows a negligible time span, even when handling high-resolution videos with substantial duration in the wild.

```python
from scipy.signal import argrelextrema

def UVD(
    embeddings: np.ndarray | torch.Tensor,
    smooth_fn: Callable,
    min_interval: int = 18,
) -> list[int]:
    # last frame as the last subgoal
    cur_goal_idx = -1
    # saving (reversed) subgoal indices (timesteps)
    goal_indices = [cur_goal_idx]
    cur_emb = embeddings.copy()  # L, d
    while cur_goal_idx > min_interval:
        # smoothed embedding distance curve (L,)
        d = norm(cur_emb - cur_emb[-1], axis=-1)
        d = smooth_fn(d)
        # monotonicity breaks (e.g. maxima)
        extremas = argrelextrema(d, np.greater)[0]
        extremas = [
            e for e in extremas
            if cur_goal_idx - e > min_interval
        ]
        if extremas:
            # update subgoal by Eq.(3)
            cur_goal_idx = extremas[-1] - 1
            goal_indices.append(cur_goal_idx)
            cur_emb = embeddings[:cur_goal_idx + 1]
        else:
            break
    return embeddings[
        goal_indices[::-1]  # chronological
    ]
```

Pseudocode A.1: **UVD implementation in Python**

In all of our simulation and real-world experiments, we use `min_interval` = 18, and Radial Basis Function (RBF) with the bandwidth of 0.08 for Kernel Regression, to eliminate most of the visual and motion noise in the video. We provide UVD decomposition qualitative results in Appendix. F.

2. Inference

We now elucidate the specifics of applying UVD subgoals in a multi-task setting during inference. Remember that given a video demonstration represented as $\tau = (o_0, \cdots, o_T)$ and UVD-identified subgoals $\tau_{goal} = (g_0, \cdots, g_m)$, we can extract an augmented trajectory labeled with goals, represented as $\tau_{aug} = (o_a, a_0, g_0), \cdots, o_T, a_T, g_m$. This is useful for goal-conditioned policy training, as discussed in Sec. IV-B.

For inference, we can similarly produce an augmented offline trajectory without the necessity of ground-truth actions, i.e., $\tau_{aug,infer} = \{(o_0, g_0), \cdots, (o_T, g_m)\}$. In the online rollout, after resetting the environment to o_0, the agent continuously predicts and enacts actions conditioned on subgoal g_0 using the trained policy. This continues until the embedding distance between the current observation and the subgoal surpasses a pre-set positive threshold ϵ at a specific timestep i, *i.e.* $d_\phi(o_i; g_0) < \epsilon$, where ϕ is the same frozen visual backbone used in decomposition and training. Following this, the subgoal will be seamlessly transitioned to the next, continuing until success or failure is achieved.

In practice, the straightforward goal-relaying inference method might face accumulative errors during multiple subgoal transitions, especially due to noise from online rollouts. However, when an agent is guided explicitly by tasks depicted in a video, incorporating the duration dedicated to each subgoal can help reduce this vulnerability. To clarify, once we've aligned subgoals with observations from the video, we also draw a connection between the timesteps of observations and their corresponding subgoals. We denote the *subgoal budget* for subgoal $g_i = o_t$ as $\mathcal{B}_{g_i} := n + 1$ where $g_{i-1} = o_{t-n-1}$ based on Eq. 3. Building on this, we propose a secondary criterion for switching subgoals: verify if the relative steps completing the current stage is in the neighborhood of the subgoal budget. This measure ensures timely transitions: it avoids prematurely switching before completing a sub-stage or delaying the transition despite accomplishing the sub-stage in the environment. To sum up, given an ongoing observation o_t and subgoal g_i at timestep t, and considering the preceding subgoal g_{i-1} at timesteps $t - h$, the subgoal will transition to g_{i+1} if

$$d_\phi(o_t; g_i) < \epsilon \quad \text{and} \quad |h - \mathcal{B}_{g_i}| < \delta \tag{5}$$

We use $\epsilon = 0.2$ and $\delta = 2$ steps for all of our experiments, except in baseline tests that are conditioned solely on final goals.

3. Feature Continuity

We then visualize 3D t-SNE in feature space from different frozen visual backbones in Fig. A.1. We include VIP [32], R3M [38], CLIP [44], and ResNet [18] trained for ImageNet-1k classification [9]. As shown in Fig. A.1, representations pretrained with temporal objectives, *e.g.* VIP [32] and R3M [38], provide more smooth, continuous, and monotone clusters than other vision foundation models. In practice, those representations with more smooth and continuous embedding curves provide better UVD decomposition as well as better performance in downstream control.

4. Runtime of UVD

Finally, We present the average runtime of UVD for preprocessing and decomposing trajectories. We break the

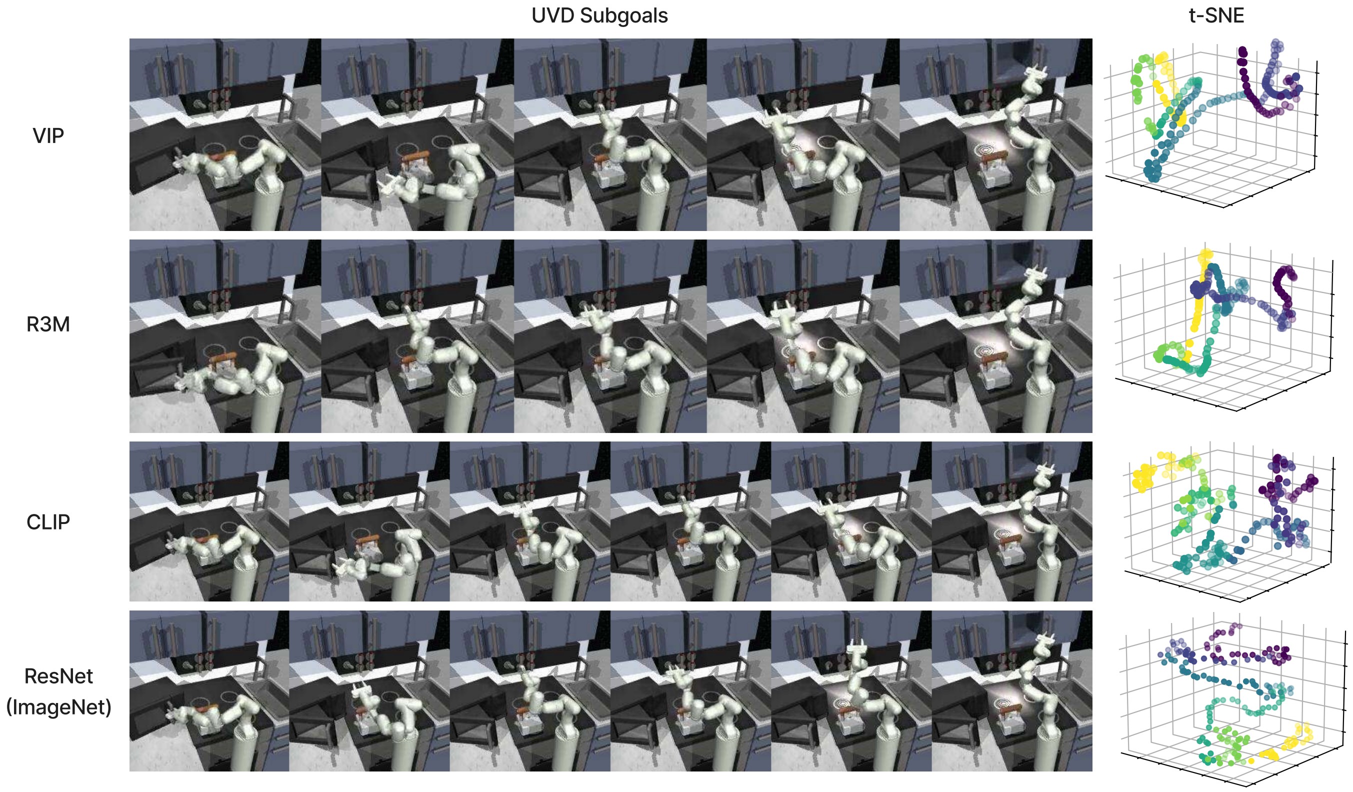

Fig. A.1: **UVD subgoals and 3D t-SNE visualizations of different frozen visual embeddings.** t-SNE colors are labeled by UVD subgoals. Representations pretrained with temporal objectives like VIP [32] and R3M [38] provide more smooth, continuous, and monotone clusters in feature space than others, whereas the ResNet trained for supervised classification on ImageNet-1k [9] provide the most sparse embeddings.

runtime by: 1) **load** from raw video file into an array; 2) **preprocess** the video array by the frozen visual encoder (including the tensor and device conversions); and 3) apply **UVD** to the preprocessed embeddings. In addition to the FrankaKitchen dataset and real-world data used for our experiments, we assessed the decomposition capabilities on a 720P MOV video. The video has a frame rate of 30 fps, contains 698 frames (equating to a duration of 23.3 seconds), and is decomposed into 16 subgoals. Visualizations can be found in Fig. F.15. Runtimes were calculated based on an average of 513 episodic data from the FrankaKitchen dataset and 100 trials for the `in the wild` video, all processed on an RTX A6000 GPU. Preprocessing is required only once offline before the policy training. As indicated in Tab. A.1, UVD operates in a negligible time span, even when handling high-resolution videos with substantial duration in the wild.

	# frames	Load	Preprocess	UVD
`FrankaKitchen`	226.9	0.023	0.155	0.0023
`In the wild`	698	1.011	0.450	0.011

TABLE A.1: **UVD offline preprocess runtimes (in seconds).**

B Models

1. *Policies*

To underscore that our method serves as an off-the-shelf method that is applicable to different policies, we ablate with a Multilayer Perceptron (MLP) based single-step policy and a GPT-like causal transformer policy. We summarize the MLP and GPT policies hyperparameters in Tab. C.4 C.5. The MLP policy, akin to the designs in [32,38] for downstream control tasks, employs a 3-layer MLP with hidden sizes of [1024, 512, 256] to produce deterministic actions. This MLP ingests a combination of the frozen visual embeddings from step-wise RGB observations and goal images followed by a 1D BatchNorm, as well as the 9D proprioceptive data encoded through a single layer complemented by a LayerNorm.

Our GPT policy removes the BatchNorm and replaces the MLP with the causal self-attention blocks consisting of 8 layers, 8 heads, and an embedding dimension of 768. We set an attention dropout rate of 0.1 and a context length of 10. The implementation is built upon [23,56]. We transition from the conventional LayerNorm to the Root Mean Square Layer Normalization (RMSNorm) [63] and enhance the transformer with rotary position embedding (RoPE) [53]. Actions are predicted via a linear similar to [5]. In practice, we generally found this recipe has more stable training and better performance than the original implementation from [23,48]. At inference time, we cache the keys and values of the self-attention at every step, ensuring that there's no bottleneck as the context length scales up. Nevertheless, in the FrankaKitchen tasks, we observed that a longer context length tends to overfit and performance drop. Therefore, we consistently use a context length of 10 for all experiments.

Policy	**MLP**	**MLP + UVD**	**GPT**	**GPT + UVD**
Episodic Runtime	6.03	6.17	7.43	7.50

TABLE B.2: **Benchmark the inference runtime (in seconds).** Runtimes are averaged across 100 rollouts with one GPU process and episodic horizon 300.

C Training Details

1. Imitation Learning

We summarize the hyperparameters of imitation learning in Tab. C.3. In the simulation, we conduct an online evaluation every 100 epochs using 10 parallel environments for each GPU machine. We choose the best checkpoints based on the combined IND and OOD performance, averaged across all multi-task training scenarios. For benchmarking inference time, we employ a single GPU, comparing our approach with the GCBC baseline as shown in Tab. B.2. This further underscores that UVD incurs negligible overhead throughout the preprocessing, training, and inference stages.

2. Reinforcement Learning

All RL experiments are trained using the Proximal Policy Optimization (PPO) [46] RL algorithm implemented within the AllenAct [58] RL training framework. We summarize the hyperparameters of reinforcement learning in Tab. C.6.

In our RL setting, the configurations for both training and inference remain consistent. This is analogous to the inference for IL as detailed in Appendix. A2.. Specifically, the task is also specified by an unlabeled video trajectory τ. Given the initial observation o_0 and UVD subgoal $g_0 \in \tau_{goal}$, the agent continuously predicts and executes actions conditioned on subgoal g_0 using the online policy with frozen visual encoder ϕ, until the condition $d_\phi(o_t; g_0) < \epsilon$ satisfied for some timestep t and positive threshold ϵ.

As shown in Eq. 4, we provide progressive rewards defined as goal-embedding distance *difference* using UVD subgoals. Recognizing that the distance between consecutive subgoals can vary, we employ the normalized distance function: $\bar{d_\phi}(o_t; g_i) := d_\phi(o_t; g_i)/d_\phi(g_{i-1}; g_i)$. This ensures that $\bar{d_\phi}(o_{t-h}; g_i) \approx 1$ for some timestep $t-h$ that the subgoal was transitioned from g_{i-1} to g_i. Additionally, we provide modest discrete rewards for encouraging (chronically) subgoal transitions, and larger terminal rewards for the full completion of task sub-stages, which is equivalent as the embedding distance between the observation and the final subgoal becomes sufficiently small. To sum up, at timestep t, the agent is receiving a weighted reward

$$\begin{aligned} R_t = \alpha \cdot \big(\bar{d_\phi}(o_{t-1}; g_i) - d_\phi(o_t; g_i)\big) \\ + \beta \cdot \mathbf{1}_{\bar{d_\phi}(o_t; g_i) < \epsilon} \\ + \gamma \cdot \mathbf{1}_{\bar{d_\phi}(o_t; g_m) < \epsilon} \end{aligned} \tag{6}$$

based on the RGB observations o_t, o_{t-1}, corresponding UVD subgoal $g_i \in \tau_{goal}$, and final subgoal $g_m \in \tau_{goal}$. While similar reward formulations appear in works such as [27,28,32,40,65], we are the first in delivering optimally monotonic implicit rewards unsupervisedly by UVD, derived directly from RGB features. In our experiments, we use $\alpha = 5, \beta = 3, \gamma = 6, \epsilon = 0.2$, and confine the first term within the range $[-\alpha, \alpha]$ in case edge cases in feature space. For the final-goal-conditioned RL baseline, it is equivalent as $g_i = g_m = o_T \in \tau = \{o_0, \cdots, o_T\}$ and $\beta = 0$ in Eq. 6.

Tab.II illustrates that the simple incorporation of UVD-rewards greatly enhances performance. We also showcase a comparison of evaluation rewards between GCRL and GCRL augmented with our UVD rewards. This is done using the R3M [38] and VIP [32] backbones, as seen in Fig. C.2. This highlights the capability of our UVD to offer more streamlined progressive rewards. This capability is pivotal for the agent to adeptly manage the challenging, multi-stage tasks presented in FrankaKitchen. To the best of our knowledge, ours is the first work to achieve such a high success rate in the FrankaKitchen task without human reward engineering and additional training. Notably, our RL agent, trained with the optimally monotonic UVD-reward, can complete 4 sequential tasks in as few as **90 steps** — a stark contrast to the over 200 steps observed in human-teleoperated demonstrations. This further illustrates the UVD-reward's potential to encourage agents to accomplish multi-stage goals more efficiently. The videos of rollouts can be found on our website.

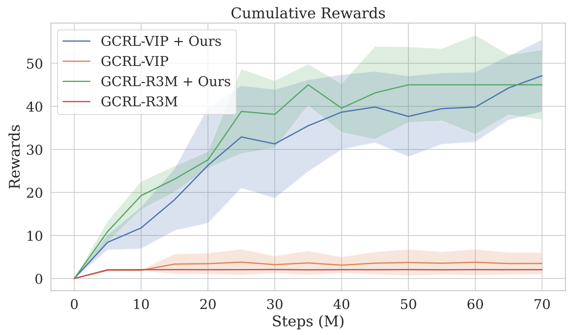

Fig. C.2: **RL evaluation cumulative rewards.** Note that different visual backbones may not be comparable due to different representation spaces, but we show significant progressive signals in comparison with the final-goal-conditioned baseline.

Failures in the GCRL baselines predominantly stem from the agent's tendency to get trapped in local minima, usually when it achieves a task sub-stage that results in the most significant visual changes in the feature space, *e.g.* sliding the cabinet in this case. Since it causes the most noticeable shifts both in pixel and feature representations, the baseline agent often fixates on this subtask alone, making no further progress. Conversely, when employing UVD subgoals and rewards, we have observed a marked difference during training. The agent incrementally learns to navigate the entire task, approaching it stage by stage. These stages are not

Hyperparameter/Value	MLP-Policy	GPT-Policy
Optimizer	AdamW [29]	AdamW [29]
Learning Rate	3e-4	3e-4
Learning Rate Schedule	cos decay	cos decay
Warmup Steps	0	1000
Decay Steps	150k	200k
Weight Decay	0.01	0.1
Betas	[0.9, 0.999]	[0.9, 0.99]
Max Gradient Norm	1.0	1.0
Batch Size	512	128

TABLE C.3: **IL training hyperparameters**

Hyperparameter	Value
Hidden Dim.	[1024, 512, 256]
Activation	ReLU
Proprio. Hidden dim.	512
Proprio. Activation	Tanh
Visual Norm.	Batchnorm1d
Proprio. Norm.	LayerNorm
Action Activation	Tanh
Trainable Parameters	3.3M

TABLE C.4: **MLP policy hyperparameters**

Hyperparameter	Value
Context Length	10
Embedding Dim.	768
Layers	8
Heads	8
Embedding Dropout	0.0
Attention Dropout	0.1
Normalization	RMSNorm [63]
Action Activation	Tanh
Trainable Parameters	58.6M

TABLE C.5: **GPT policy hyperparameters**

Hyperparameter	Value
GPU instances	8x RTX A6000
Environments per GPU	8
Optimizer	AdamW [29]
Learning rate	3e-4
Learning rate schedule	linear decay
Max gradient norm	0.5
Discount factor γ	0.99
GAE τ	0.95
Value loss coefficient	0.5
Normalized advantages	True
Entropy coefficient	0.001
Rollout length	200
PPO epochs	10
Number of mini-batches	1
PPO Clip	0.1

TABLE C.6: **RL hyperparameters.**

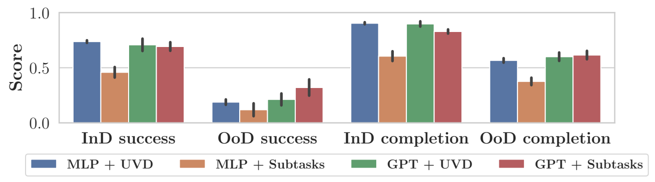

Fig. D.3: **Comparison with the decomposition from human pre-defined**

isolated, as they share pertinent information. For example, UVD breaks down task sequences into phases characterized by nearly monotonic motions. These can be categorized as the "hand reaching" or "object interaction" phases. This shared knowledge framework means that once the agent masters the initial "hand reaching" phase, subsequent similar hand-reaching motions become more intuitive to learn and execute. Nevertheless, we do occasionally observe instances where applying UVD results in failure. In these cases, the agent often oscillates its gripper back and forth, seemingly hacking the reward shaping, which in turn leads to an irreversible state. We speculate that incorporating supervised human intervention [52] or unsupervised near-irreversible detection [65], could address this issue and further enhance performance.

D Extended Experiments and Ablations

1. Simulation

We present numerical results for ablations from Sec. V-A in Tab. D.7, with extended comparison with GCBC baselines. Without surprise, our method consistently outperforms baselines in compositional generalization settings, when varying the dataset size to 5 demonstrations for each FrankaKitchen task, or adjusting the seen-unseen partitions, which doubles the count of unseen sequences while halving the number of seen ones.

In the FrankaKitchen demonstrations, there are 24 task sequences encompassing 4 subtasks each, translating to 4 ordered object interactions. We further compare GCBC+UVD with GCBC conditioned on the final frames of each human-defined subtask in FrankaKitchen, utilizing both MLP and GPT policies. The main distinction between UVD-decomposed subgoals and the subgoals for each subtask is that UVD furnishes milestones emphasizing monotone motions, while subtasks yield subgoals with oracle semantic meanings. Both methodologies employ the identical inference approach as described in Appendix 2., and share the same tasks partitions from three distinct seeds. The average successes and subtask completion rates are shown in Fig. D.3. Surprisingly, even when subgoals from subtasks offer ground-truth semantic meaning (and, notably, share identical conditioned subgoal frames across different task sequences), only GPT policy allows for performance surpassing our method in the OoD setting. In the InD setting, however, using subgoals from pre-defined sub-tasks leads to a substantial performance drop for MLP policies. This decline might be due to potential confusion (*e.g.* mis-manipulated top and bottom burners) arising from similar subgoals when jointly training multiple multi-stage tasks when utilizing a lightweight, single-step MLP policy.

2. Real-Robot

In the OoD settings, even though we have not achieved exceptionally high success rates for all of the tasks, we still managed a completion rate of around or above 50%. This outcome can be attributed to the fact that policies trained using our method consistently exhibit the right intent. Instead of overfitting to the InD settings, they tend to successfully complete intermediate steps and occasionally face challenges only at later stages. In contrast, the GCBC baseline always overfits the InD initial state. For example, in `Fold-Cloth` generalization experiments, the baseline still goes to the corner that is already folded. For more rollout visualizations,

Representation	Method	IND success	IND completion	OOD success	OOD completion
VIP (5 demos)	GCBC-GPT	0.409 (0.102)	0.702 (0.066)	0.005 (0.005)	0.285 (0.024)
	GCBC-GPT + Ours	**0.419 (0.027)**	**0.763 (0.016)**	**0.13 (0.033)**	**0.533 (0.026)**
VIP (5 demos)	GCBC-MLP	**0.668 (0.024)**	0.82 (0.035)	0.016 (0.016)	0.208 (0.006)
	GCBC-MLP + Ours	0.643 (0.058)	**0.848 (0.028)**	**0.104 (0.048)**	**0.458 (0.038)**
VIP (8 seen - 16 unseen)	GCBC-MLP	0.724 (0.057)	0.851 (0.036)	0.001 (0.001)	0.102 (0.020)
	GCBC-MLP + Ours	0.717 (0.051)	0.853 (0.048)	**0.084 (0.007)**	**0.497 (0.055)**
VIP (8 seen - 16 unseen)	GCBC-GPT	0.602 (0.114)	0.554 (0.48)	0.003 (0.003)	0.143 (0.124)
	GCBC-GPT + Ours	0.587 (0.049)	0.558 (0.483)	**0.037 (0.040)**	**0.307 (0.268)**

TABLE D.7: **Ablations on dataset size and compositions, with comparisons with GCBC baselines and UVD**

please refer to the videos available on our website.

We further extend our OOD setting, which aimed for unseen initial states in Sec. V-B, also encompass more diverse intermediate states. Our objective during deployment is to ensure the agent remains resilient to tasks, even in the presence of human interference. In `Apple-in-Oven` and `Fries-and-Rack` tasks, we introduce two more OOD scenarios. In the first, we revert the scene by placing the apple back to its original position, challenging the agent to recover from this change. In the second, we manually circumvent an intermediate step. For instance, after the robot has grasped the bowl of fries, we manually transfer all the fries to the plate. This alteration means the agent should subsequently place the bowl directly on the rack without the need for pouring.

Method	`Apple-in-Oven`	`Fries-and-Rack`
GCBC	0.0	0.2
GCBC + Ours	**0.5**	**0.9**

TABLE D.8: **Success rate over 10 rollouts with human interference.**

E Real-World Robot Experiment Details

The robot learning environment is illustrated in Fig. E.4. We use a 7-DoF Franka robot with a continuous joint-control action space. A Zed 2 camera is positioned on the table's right edge, and only its RGB image stream—excluding depth information—is employed for data collection and policy learning. Another Zed mini camera is affixed to the robot's wrist. For the `Apple-in-Oven` task, we utilize the right view from both cameras, while for the `Fries-and-Rack` and `Fold-Cloth` tasks, we rely on their left views.

1. Task Descriptions

	EL	# demos
`Apple-in-Oven`	197.5	105
`Fries-and-Rack`	170.1	110
`Fold-Cloth`	246.8	105

TABLE E.9: **Real Tasks average episode length (EL) and the number of demos (# demos).**

We specify the average episode lengths and the number of demonstrations we used for experiments for each task in

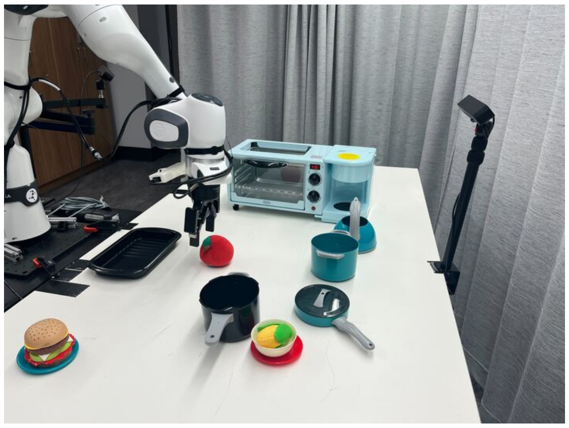

Fig. E.4: **Real robot experiments setup.**

Tab. E.9. The criteria for successful task completion are as follows:

- `Apple-in-Oven`: pick up the apple on the table; place the apple in the bowl without tipping it over; push the bowl into the oven; close the oven door.
- `Fries-and-Rack`: pour fries onto the plate, ensure at least half of them are on the plate; place the bowl on the rack without causing any collisions.
- `Fold-Cloth`: grasp the corner of the cloth and fold the cloth in the directions shown in the demonstrated video multiple times.

During evaluation, we assess the successful completion of each sub-stage over 20 rollouts to determine the overall success and completion rates. To evaluate our policy's compositional generalization abilities, we introduce unseen initial states for each task. While the success criteria remain consistent with prior assessments, the initial step has been pre-completed by humans. The extended OOD setting, which includes human interference in intermediate states, is detailed in Appendix 2..

2. Training and evaluation details

UVD in training: As with our simulation experiments, we preprocess all the demos using UVD. The behavior cloning policy further incorporates the view from the wrist camera besides the view from the side camera and decomposed

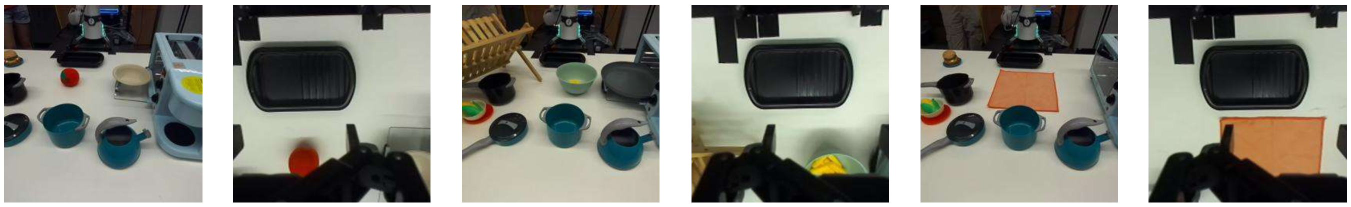
Fig. E.5: **Raw observations from two different cameras for three tasks.** Three (No.1, 3, 5 from the left) are from the side-view Zed2 camera and the others (No.2, 4, 6) are from Zed mini on the wrist.

subgoals during training. Operating under velocity control, our robot's action space encompasses a 6-DoF joint velocity and a singular dimension of the gripper action (open or close). Consequently, the policy produces 7D continuous actions. The robot control frequency is set as 15 Hz.

UVD in evaluating: In the training stage, we save the set of subgoals and corresponding observations over all demos in a task. During inference, every time the robot gets a pair of observations, we retrieve the subgoal that has the nearest observation (l_2 norm over observation embeddings) with the current one as the current subgoal. Then we concatenate the current observation with the retrieved subgoal together as input then get real-time joint velocity action.

Our method frequently results in the development of more robust policies, enabling recovery actions when initial attempts fail. For example, in the failure scenario of `Apple-in-Oven` and one of the successful cases in `Fold-Cloth` as showcased on our website, the policy opts for a reattempt, pushing the bowl further if not adequately placed inside the oven, or re-folding if the cloth's corner is not grasped properly. In contrast, such recovery behaviors are conspicuously absent in the GCBC baselines, further highlighting its propensity to overfit to IND training setting.

BC Model details: We train our policy on a laptop with RTX 3080 GPU. For both the baseline policy and our method, we add proprioception to help learning and augment each training dataset by randomly cropping the input images. Since we have limited demonstrations in the real world, we only set MLP size to be [256,256]. Please refer to Tab. E.10 for details.

Hyperparameter	**Value**
GPU Instances	RTX 3080 Ti Laptop GPU
MLP Architecture	[256, 256]
Non-Linear Activation	ReLU
Optimizer	AdamW [29]
Gradient Steps	10k
Batch Size	64
Learning Rate	1e-3
Proprioception	Yes
Augmentation	Random crop

TABLE E.10: **Real robot BC hyperparameters.**

F Qualitative Subgoal Decomposition Results

We show decomposition results with UVD on simulation videos in Fig. F.6 F.7, real robots videos in Fig. F.8, F.9, F.10, and wild videos in Fig. F.11, F.12, F.13, F.14, F.15. From the subgoal decomposition results, we can have a clear overview of the key frames within a video. For instance, in `Fold-Cloth` video, UVD precisely catches the picking and placing key frames for three times.

Fig. F.6 F.7 are from our simluation experiments, Fig. F.8, F.9, F.10 are from real robot experiments. Fig. F.11 video depicts a human opening a cabinet and rearranging items, while Fig. F.12 video showcases unlocking a computer in an office. Furthermore, Fig. F.13 demonstrates the process of opening a drawer and charging a device, and Fig. F.14 illustrates washing and then wiping hands in a bathroom. Lastly, Fig. F.15 presents activities shot in the kitchen with relatively longer duration. Based on the analysis of all video decomposition results, it is evident that UVD extends beyond robotic settings, proving to be highly effective in household scenarios captured in human videos.

G Limitation and Future Works

While UVD offers the advantage of not necessitating any task-specific knowledge or training, its efficacy is well-demonstrated across both simulated and real-robot environments. However, as we only validate on fully observable manipulation tasks, direct application to navigation tasks, especially those embodied tasks involving partial observations, may not yield intuitive or explainable subgoals (even though representations are pretrained with temporal objective using egocentric datasets [31,32,38]).

Looking ahead, we are eager to broaden the applications of UVD, diving deeper into its capabilities within egocentric scenarios, and even the key-frame extraction for video understanding and dense video caption tasks. On another front, while task graphs are widely used in Reset-Free RL [17,62], acquiring milestones as subgoals is resource-intensive and lacks scalability. By integrating our off-the-shelf UVD subgoals into the task-graph, we are interested in seeing agents that, with minimal resets, can adeptly handle a wide range of tasks across various sequences and horizons in the wild.

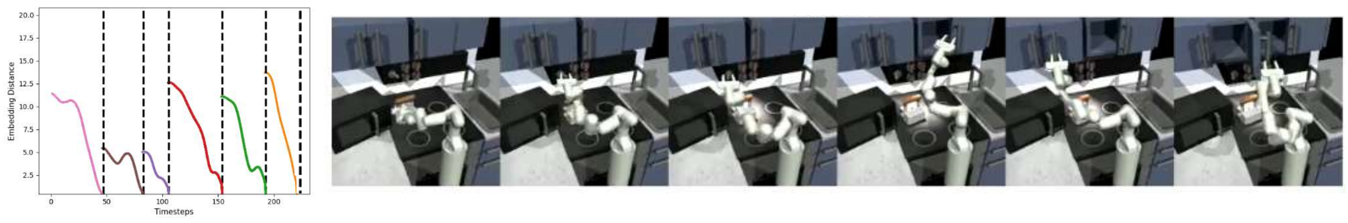

Fig. F.6: **Video sequence**: moving a kettle, turning on light switch, operating slide cabinet, operating hinge cabinet.

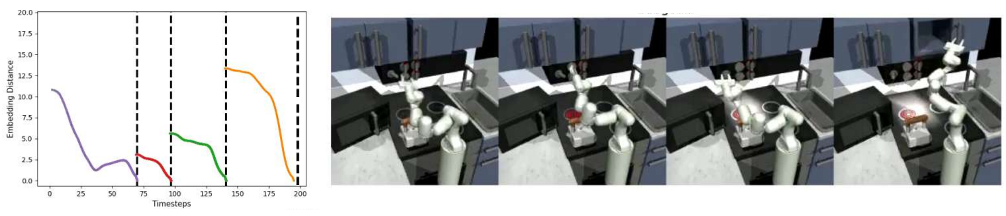

Fig. F.7: **Video sequence**: rotating bottom burner, rotating top burner, turning on light switch, operating slide cabinet.

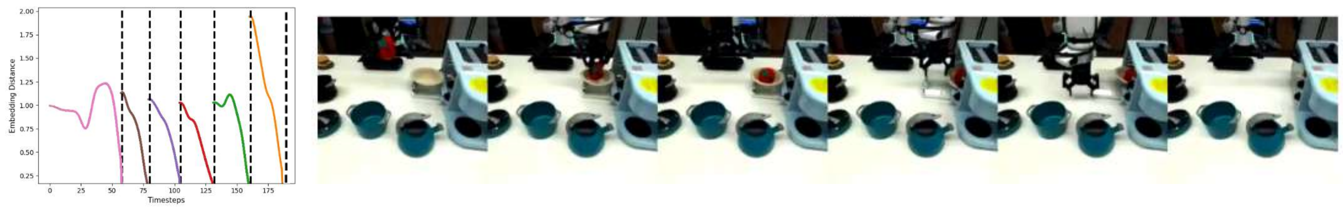

Fig. F.8: **Video sequence**: picking apple, placing apple in the bowl, pushing the bowl into the oven, closing the oven.

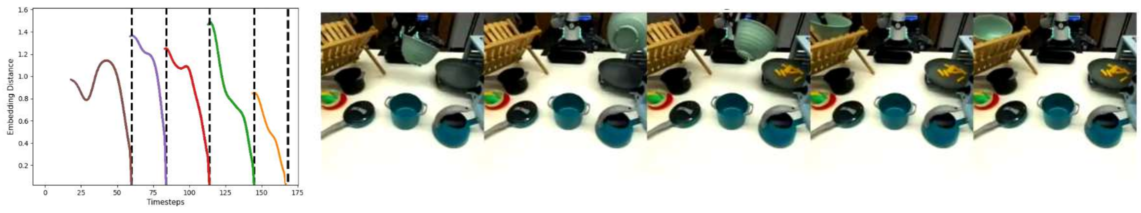

Fig. F.9: **Video sequence**: picking a bowl, pour fries out of the bowl, placing the bowl on the rack.

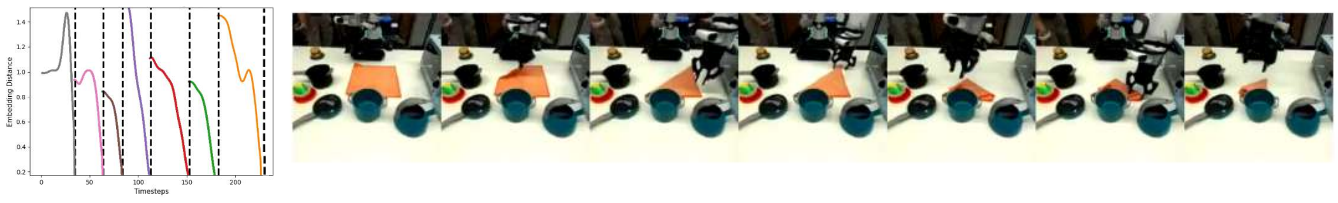

Fig. F.10: **Video sequence**: diagnal fold, quater fold, eighth fold.

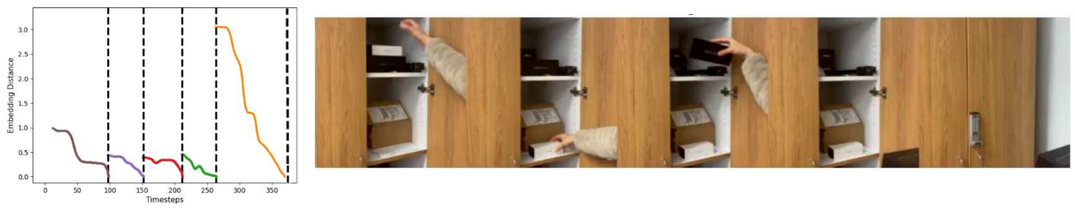

Fig. F.11: **Video sequence**: picking upper white box, placing white box, picking upper black box, closing the cabinet.

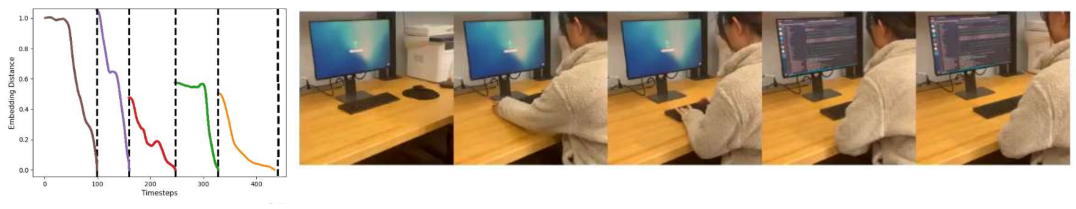

Fig. F.12: **Video sequence**: grabing a chair, moving the keyboard towrads human, typing password, unlocking a computer.

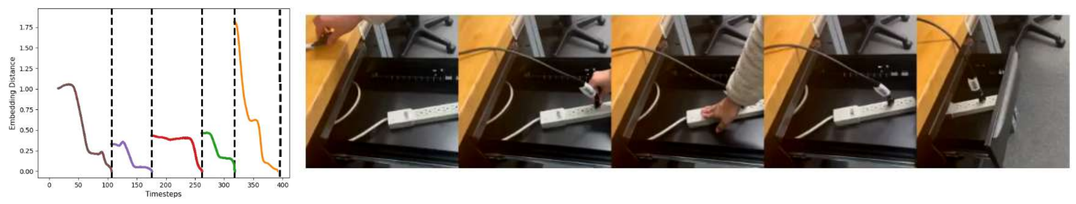

Fig. F.13: **Video sequence**: opening a drawer, picking the charger, pluging the charger, turing on the power strip, closing the drawer partially.

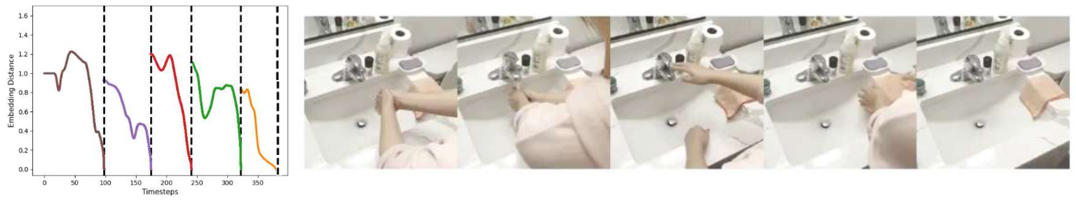

Fig. F.14: **Video sequence**: lathering hands, washing hands, turning off the tap, wiping hands with towel, placing back the cloth.

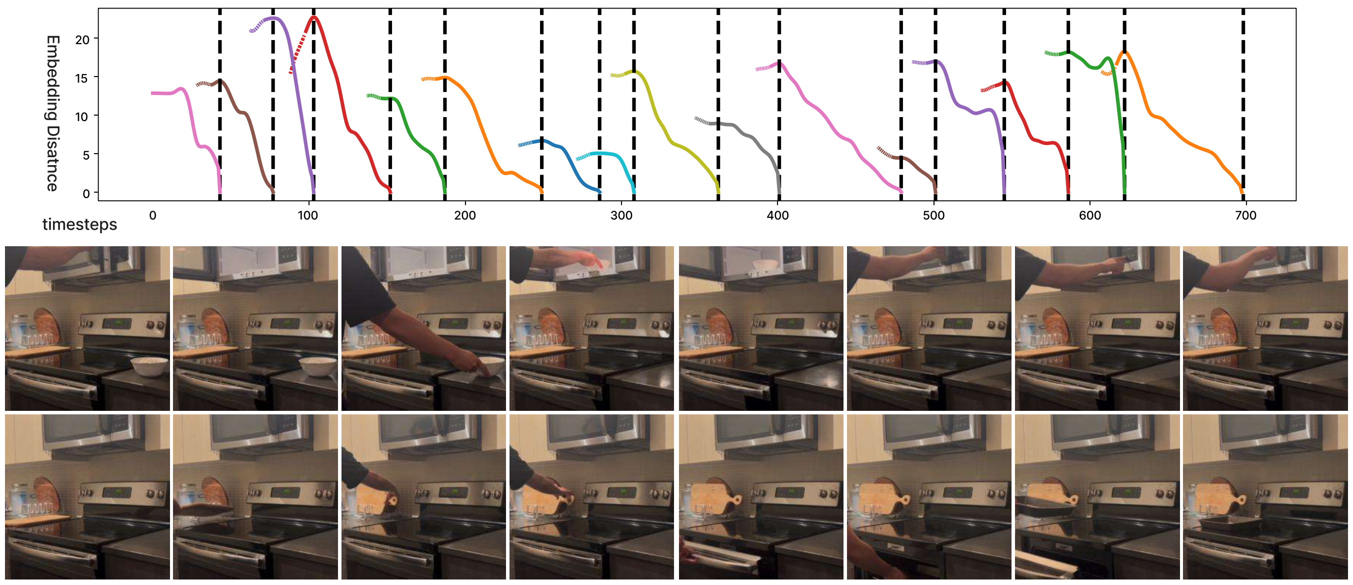

Fig. F.15: **UVD decomposes long video into subgoals.** The video sequence demonstrates: opening the microwave, placing a bowl with rice inside the microwave, closing the microwave, activating the microwave to heat the rice, placing the cutting board onto its rack, opening the oven, and putting the baking tray on the burner.

References

[1] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017. 2

[2] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2019. 2

[3] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard, "Affordance learning from play for sample-efficient policy learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6372–6378. 1, 2

[4] E. Chane-Sane, C. Schmid, and I. Laptev, "Goal-conditioned reinforcement learning with imagined subgoals," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1430–1440. 2

[5] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021. 11

[6] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings," in *International conference on machine learning*. PMLR, 2018, pp. 1009–1018. 2

[7] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, *et al.*, "Scaling egocentric vision: The epic-kitchens dataset," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 720–736. 1

[8] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, "Robothor: An open simulation-to-real embodied AI platform," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 3161–3171. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Deitke_RoboTHOR_An_Open_Simulation-to-Real_Embodied_AI_Platform_CVPR_2020_paper.html 3

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255. 10, 11

[10] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *Advances in neural information processing systems*, vol. 32, 2019. 3

[11] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019. 2

[12] K. Fang, P. Yin, A. Nair, and S. Levine, "Planning to practice: Efficient online fine-tuning by composing goals in latent space," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4076–4083. 1, 2

[13] K. Fang, P. Yin, A. Nair, H. R. Walke, G. Yan, and S. Levine, "Generalization with lossy affordances: Leveraging broad offline data for learning visuomotor tasks," in *Conference on Robot Learning*. PMLR, 2023, pp. 106–117. 1, 2

[14] 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. 3

[15] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, *et al.*, "Ego4d: Around the world in 3,000 hours of egocentric video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 995–19 012. 1

[16] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," *arXiv preprint arXiv:1910.11956*, 2019. 2, 4, 5

[17] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6664–6671. 15

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 10

[19] M. Heo, Y. Lee, D. Lee, and J. J. Lim, "Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation," *arXiv preprint arXiv:2305.12821*, 2023. 2

[20] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8565–8574. 1, 2

[21] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612–1619, 2022. 1, 2

[22] D. Jayaraman, F. Ebert, A. A. Efros, and S. Levine, "Time-agnostic prediction: Predicting predictable video frames," *ICLR*, 2019. 2

[23] A. Karpathy, "nanogpt: The simplest, fastest repository for training/finetuning medium-sized gpts." 2023. [Online]. Available: https://github.com/karpathy/nanoGPT 11

[24] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 829–14 838. 1

[25] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020. 5

[26] Y. Lee, E. S. Hu, and J. J. Lim, "Ikea furniture assembly environment for long-horizon complex manipulation tasks," in *2021 ieee international conference on robotics and automation (icra)*. IEEE, 2021, pp. 6343–6349. 2

[27] Y. Lee, A. Szot, S.-H. Sun, and J. J. Lim, "Generalizable imitation learning from observation via inferring goal proximity," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=lp9foO8AFoD 12

[28] Y. Li, T. Gao, J. Yang, H. Xu, and Y. Wu, "Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 765–12 781. 12

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017. 13, 15

[30] 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. 5

[31] Y. J. Ma, W. Liang, V. Som, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman, "Liv: Language-image representations and rewards for robotic control," *arXiv preprint arXiv:2306.00958*, 2023. 1, 3, 4, 5, 15

[32] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang, "Vip: Towards universal visual reward and representation via value-implicit pre-training," *arXiv preprint arXiv:2210.00030*, 2022. 1, 2, 3, 4, 5, 10, 11, 12, 15

[33] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, P. Abbeel, J. Malik, *et al.*, "Where are we in the search for an artificial visual cortex for embodied intelligence?" *arXiv preprint arXiv:2303.18240*, 2023. 1

[34] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, "Learning to generalize across long-horizon tasks from human demonstrations," *arXiv preprint arXiv:2003.06085*, 2020. 1, 2

[35] O. Nachum, S. Gu, H. Lee, and S. Levine, "Near-optimal representation learning for hierarchical reinforcement learning," *arXiv preprint arXiv:1810.01257*, 2018. 2

[36] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems*, vol. 31, 2018. 2

[37] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," *arXiv preprint arXiv:1909.05829*, 2019. 1, 2

[38] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," *arXiv preprint arXiv:2203.12601*, 2022. 1, 3, 4, 5, 10, 11, 12, 15

[39] S. Nasiriany, V. Pong, S. Lin, and S. Levine, "Planning with goal-conditioned policies," *Advances in Neural Information Processing Systems*, vol. 32, 2019. 2

[40] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, 1999, pp. 278–287. 12

[41] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023. 4, 5

[42] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, "The unsurprising effectiveness of pre-trained vision models for control," *arXiv preprint arXiv:2203.03580*, 2022. 1

[43] K. Pertsch, O. Rybkin, F. Ebert, S. Zhou, D. Jayaraman, C. Finn, and S. Levine, "Long-horizon visual planning with goal-conditioned hierarchical predictors," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 321–17 333, 2020. 1, 2

[44] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763. 4, 10

[45] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. 5

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. 5, 12

[47] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1134–1141. 2

[48] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in neural information processing systems*, vol. 35, pp. 22 955–22 968, 2022. 11

[49] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn, "Waypoint-based imitation learning for robotic manipulation," *arXiv preprint arXiv:2307.14326*, 2023. 1, 2

[50] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner, "Taco: Learning task decomposition via temporal alignment for control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4654–4663. 1, 2

[51] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799. 1, 2

[52] K. P. Singh, L. Weihs, A. Herrasti, J. Choi, A. Kembhavi, and R. Mottaghi, "Ask4help: Learning to leverage an expert for embodied tasks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 221–16 232, 2022. 13

[53] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021. 11

[54] 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. 2

[55] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on image processing*, vol. 16, no. 2, pp. 349–366, 2007. 10

[56] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023. 11

[57] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 5922–5931. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Weihs_Visual_Room_Rearrangement_CVPR_2021_paper.html 3

[58] L. Weihs, J. Salvador, K. Kotar, U. Jain, K.-H. Zeng, R. Mottaghi, and A. Kembhavi, "Allenact: A framework for embodied ai research," *arXiv preprint arXiv:2008.12760*, 2020. 12

[59] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=H1gX8C4YPr 3

[60] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik, "Masked visual pre-training for motor control," *arXiv preprint arXiv:2203.06173*, 2022. 1

[61] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3795–3802. 1, 2

[62] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine, "Dexterous manipulation from images: Autonomous real-world rl via substep guidance," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5938–5945. 15

[63] B. Zhang and R. Sennrich, "Root Mean Square Layer Normalization," in *Advances in Neural Information Processing Systems 32*, Vancouver, Canada, 2019. [Online]. Available: https://openreview.net/references/pdf?id=S1qBAf6rr 11, 13

[64] L. Zhang, G. Yang, and B. C. Stadie, "World model as a graph: Learning latent landmarks for planning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 611–12 620. 2

[65] Z. Zhang and L. Weihs, "When learning is out of reach, reset: Generalization in autonomous visuomotor reinforcement learning," *arXiv preprint arXiv:2303.17600*, 2023. 12, 13