Supplementary Material

This supplementary material provides additional details on the STACK framework and our experiments. Section I offers a detailed description of the framework, including an expanded discussion of temporal and spatial segmentation, implementation details for the learned trajectory sampler and skill effect model, and pseudocode for the planning algorithm. Section II describes the experimental setup, including the robot platform, task design, and baseline implementations. Section III presents additional experimental results, including the original experiment with binary success counts, evaluations on articulated objects, validation of entity segmentation, an ablation study of the trajectory sampler, and a challenging long-horizon task with complex geometry that highlights the need for both task and motion planning. Section IV presents a failure analysis along with qualitative examples of challenging cases. Section V provides an expanded discussion of related work. Section VI expands the discussion of limitations. Finally, Section VII lists all prompts used in our experiments.

I. STACK Details

A. Temporal Segmentation

For temporal segmentation, we use RGB video data extracted from demonstration recordings. The raw video is provided directly to the `gemini-2.5-pro-preview-03-25` API, which internally subsamples and tokenizes each frame. To help the model detect contact changes, we incorporate two types of proprioceptive signals. First, gripper widths are binarized to indicate whether the gripper is open or closed, as raw values were empirically found to be difficult for the model to interpret. Second, joint torques, recorded from the Franka Emika arm's torque sensors, are reduced to a single scalar via averaging and rounded to two decimal places. Both signals are downsampled to 1 fps and annotated with `mm:ss` timestamps to align them with the video frames. The model is trained to associate these timestamps with the corresponding frames. For bimanual tasks, we additionally overlay transparent, color-coded masks on the video frames to distinguish the two arms. These masks are generated automatically using the robot's URDF model and camera parameters.

To improve accuracy, we perform temporal segmentation in multiple stages. In the proposal stage, the model generates coarse candidate segments (prompt shown in Listing 1). For bimanual tasks, the proposal stage is further divided into two substeps. First, the model is prompted to identify coarse segments involving bimanual coordination, such as picking up a book with two arms (prompt shown in Listing 2. Second, it is prompted to segment single-arm skills such as placing the book on a shelf (prompt shown in Listing 1). In the aggregation stage (prompt shown in Listing 3), the framework takes all the proposed segments and aggregates them into a single list of segments without overlaps. In the refinement stage, the list of aggregated temporal segments, along with all demonstration videos, is re-supplied to the model for further refinement (prompt shown in Listing 4).

After temporal segmentation, we identify the relevant entities in each skill segment. These entities are used for spatial segmentation and for constructing the arguments args of each skill. For each segment, we prompt the model to name the manipulated entities. All of the temporal segments of a specific skill, along with their video frames, are provided to the Video-LM. The video frames are subsampled to 1 fps and tokenized, while the temporal segment timestamps, names, and descriptions are included in the text prompt. The model then proposes a list of entities involved in that specific temporal segment. To ensure consistency, we provide in-context examples. A sample prompt is shown in Listing 5.

While the Video-LM exhibits strong temporal segmentation capabilities, it may include free-space motion within segments, which can dilute the meaningful, contact-rich parts of the demonstration.

B. Spatial Segmentation

For spatial segmentation, we use the open-vocabulary descriptions of relevant entities provided by the Video-LM for each skill segment. These descriptions serve as queries to an open-vocabulary object detector. Specifically, we use OWLv2 [69] and fall back to Lang-SAM [70] only when OWLv2 fails to detect any relevant objects. Once bounding boxes are obtained, they are used as prompts for the SAM2 [79] segmentation model, which produces pixel-wise masks for each object. To extract point clouds for the entities $\{p_e \; \forall \; e \in \mathcal{E}_s\}$, we repeat this segmentation process across all available camera views. For each view, we mask the depth image using the corresponding pixel-wise segmentation mask and project the masked depth image into 3D. Then we aggregate the resulting point clouds from all views. During training, spatial segmentation is performed for every frame. At test time, segmentation is run on each new observation.

In our preliminary experiments, we found that the open-vocabulary detector may occasionally generate bounding boxes for irrelevant objects. To mitigate this, we ensure that the entity descriptions generated by the Video-LM are as specific and descriptive as possible. In some cases, the detector fails to identify any target object. However, we expect that future improvements in open-vocabulary detection will continue to reduce these limitations.

C. Skill Sampler

Each trajectory sampler $\pi_s : \mathcal{O}_{\mathcal{E}_s} \to \mathcal{U}^{H_s}$ in a skill maps the segmented point clouds $\{p_e \; \forall \; e \in \mathcal{E}_s\}$ of the relevant entities $\mathcal{E}_s$ to a sequence of H_s low-level actions, additionally conditioned on robot proprioception.

All point clouds are concatenated and downsampled to 2048 points using farthest point sampling. Our preliminary experiments suggest that adding a per-point object index is unnecessary. We also uniformly subsample all trajectories so that $H_s = 32$ for every skill. For skills involving manipulation of a grasped object, we additionally provide the initial end-effector pose, allowing the generated actions to condition on

the grasp. For all other skills, the end-effector pose is set to the identity. Depending on the robot configuration, the model predicts end-effector poses for either a single arm or both arms jointly. For each arm, it outputs a 3D translation vector, an $SO(3)$ rotation represented using the 6D continuous representation [80] to avoid discontinuities, and a scalar value for the gripper width.

The trajectory sampler is based on the point cloud-conditioned diffusion model introduced in prior work [10]. The point cloud encoder processes the XYZ coordinates of the input point cloud and produces a 128-dimensional feature embedding, with layer normalization applied. The diffusion model is a 1D conditional U-Net that takes as input the sequence of noisy prediction targets and a global conditioning feature derived from the encoded observations. It uses a diffusion step embedding dimension of 128, downsampling channel dimensions of 128, 256, and 512, a kernel size of 5, and group normalization with 8 groups. The final feature embedding is decoded into a 10D or 20D action vector, depending on whether the skill is single-arm or bimanual.

The model is trained using the AdamW optimizer with a learning rate of 7×10^{-5}, betas of $(0.95, 0.999)$, and a weight decay of 1×10^{-6}. We use a batch size of 612 on a single NVIDIA RTX A5000 GPU with 24 GB of memory. The model is trained for 700 epochs. The diffusion process follows a DDIM scheduler [81] with 50 timesteps for both training and inference, a beta range of $[0.0001, 0.02]$, and a squared cosine noise schedule from GLIDE [82].

D. Skill Effect Model

The skill effect model, $f_s \colon \mathcal{O}_{\mathcal{E}_s} \times \mathcal{U}^{H_s} \to \mathcal{O}_{\mathcal{E}_s}$, predicts how the execution of a skill transforms the given entities. Given a sampled trajectory and the point clouds $\{p_e \; \forall \; e \in \mathcal{E}_s\}$, the model predicts a rigid-body transform $T \in SE(3)$ for each $e \in \mathcal{E}_s$ such that the updated point cloud is given by $p'_e = Tp_e$.

To generate training data, we take the extracted object point clouds from the initial and final frames of each skill segment and compute a rigid transformation in $SE(3)$ using Iterative Closest Point (ICP) [71] alignment. We first calculate the L2 distance between the centroids of the initial and final point clouds. If this distance exceeds a threshold ϵ, the motion is classified as large; otherwise, it is considered small. For small motions (e.g., pushing a book or grasping a mug), we apply ICP directly. For large motions (e.g., placing a book after grasping), we initialize the transformation by assuming that the object is rigidly attached to the gripper. The object point cloud is first transformed according to the gripper's initial and final poses, and the result is then refined using ICP.

Similar to the trajectory sampler, the effect model is built on a point cloud-conditioned diffusion model. It uses the same hyperparameters and training procedure as the trajectory sampler. The output omits the gripper state and predicts only the 3D translation and 6D rotation for the transform T.

E. Spatial Augmentation

During skill discovery, the vision-language model (VLM) identifies the relevant entities involved in each skill. For skills involving only a single entity, such as grasping a mug, the entity is treated as the anchor, denoted as e_a. For skills involving two entities, such as placing a book on a bookshelf, one entity is designated as the anchor (e_a) and the other as the moving entity (e_m), which is manipulated relative to the anchor. In all cases, we extract point clouds of the relevant entities and represent both observations and actions in the reference frame of the anchor entity. For example, in the bookshelf task, the bookshelf serves as the anchor and the book as the moving object. Using the anchor's reference frame improves learning efficiency and spatial generalization.

To further improve generalization, we apply several spatial augmentations during training. For each skill, we randomly perturb the anchor entity's point cloud. Specifically, we apply a rotation around the z-axis, with the angle sampled from a uniform distribution $\mathcal{U}(-10°, +10°)$. We also apply a random translation along the x, y, and z axes, where each component is independently sampled from $\mathcal{U}(-5, \text{cm}, +5, \text{cm})$. For skills involving a moving object, we apply additional augmentations. First, we sample new xy positions for the object by precomputing evenly spaced locations across the reachable workspace. We then apply a random z-axis rotation sampled from $\mathcal{U}(-36°, +36°)$, and a random translation along each x, y, and z axis, with components independently sampled from $\mathcal{U}(-5, \text{cm}, +5, \text{cm})$. To preserve the spatial relationship between the gripper and the object, we also apply the same transformation to the initial end-effector pose. These augmentations expose the policy to a wide range of spatial configurations and help prevent overfitting to the specific initial object poses observed in the demonstration data.

To simulate out-of-distribution point cloud observations caused by extreme object placements at test time, we further augment the training data by projecting point clouds into pseudo camera viewpoints. These pseudo cameras are sampled from poses distributed around the workspace and are oriented similarly to the real cameras. This ensures that the resulting views are both realistic and meaningfully different from the actual camera perspectives. For each pseudo camera, we maintain a z-buffer to remove points that are occluded or outside the field of view. This augmentation prevents the model from overfitting to a narrow region of the point cloud and promotes robustness to occlusions and variations in observed object geometry.

F. Planning with Learned Skills

At deployment, given a task goal ℓ expressed in natural language and a point cloud observation of the scene $\bar{o}_0$, our objective is to generate a low-level action sequence $u_{1:T}$ composed of a sequence of K skills. The overall planning procedure is summarized in Algorithm 1.

We begin by using a video-language model to propose candidate skill sequences for the given task and scene. The model is prompted with videos of skill segments from demonstrations along with their corresponding descriptions. A sample prompt is shown in Listing 6. Along with the candidate skill sequences, we also generate the arguments for each skill, which consist of descriptions of the relevant

Algorithm 1 Planning with Learned Composable Skills

Input: Language goal ℓ, initial observation o_0, skill library Π, trajectory samples per skill M
Output: Sequence $\big[(s_0, \hat{\tau}_0), \ldots, (s_{K-1}, \hat{\tau}_{K-1})\big]$ satisfying equation III **or** FAIL

skeletons $\leftarrow$ GENERATEPLANSKELETONS(ℓ, Π, o_0) ▷ use VLM to generate plan skeletons
$\mathcal{E} \leftarrow$ SEGMENTSCENE$(o_0, \textit{skeletons})$ ▷ segment scene with open-vocab models
open $\leftarrow$ empty LIFO stack
for all $\sigma \in$ *skeletons* **do** ▷ initialize DFS
  push $\big(o_0,\ \sigma,\ 0,\ u_{\text{NULL}},\ \emptyset\big)$ onto *open*
while *open* not empty **do**
  $(o, \sigma, i, u_{\text{prev}}, \alpha) \leftarrow$ *open*.POP()
  if $i = |\sigma|$ **then** ▷ found a fully instantiated skeleton
    return α
  $s \leftarrow \sigma[i]$ ▷ get current skill
  $\mathcal{E}_s \leftarrow$ MATCHARGS$(s, \mathcal{E})$
  for all $\tau \in \pi_s(o_{\mathcal{E}_s})$ **do** ▷ sample candidate trajectories
    $\tilde{\tau} \leftarrow$ CFREETRAJ$\big(u_{\text{prev}},\ \tau[1],\ o, \alpha\big)$ ▷ plan for transit/move
    if $\tilde{\tau} = \emptyset$ **then** ▷ reject invalid skill trajectory
      continue
    $o' \leftarrow f_s(o_{\mathcal{E}_s}, \tau)$ ▷ predict skill effect
    if COLLIDES$(o',\ o \setminus o_{\mathcal{E}_s})$ **then** ▷ check for collision
      continue
    $\hat{\tau} \leftarrow \tilde{\tau} \,\|\, \tau$ ▷ prepend transit/move trajectory
    $\alpha' \leftarrow \alpha \cup \{(s, \hat{\tau})\}$
    push $\big(o' \cup (o \setminus o_{\mathcal{E}_s}),\ \sigma,\ i+1,\ \tau[-1],\ \alpha'\big)$ onto *open* ▷ move to the next skill
return FAIL

entities. These descriptions often include attributes such as color, which we found to be the most reliable cue for detecting specific entities during execution. The descriptions are used for spatial segmentation.

During planning, we search for a trajectory τ_s for each skill, sampled from its learned trajectory sampler π_s, subject to transition feasibility, kinematic, and collision constraints $\mathcal{G}$ and the current observation. Crucially, we predict the effect of each skill using the learned effect model f_s. For each valid trajectory, we update the scene geometry based on the predicted effect, allowing the next skill to generate its trajectory and effect using the updated observation.

To transition between any two consecutive learned skills, we generate a collision-free trajectory using the cuRobo planning framework [73]. We first provide the current depth images along with the corresponding camera intrinsics and extrinsics to cuRobo, which uses NVBlox [72] to construct an occupancy grid of the environment. Given the current and target joint configurations, we then use the cuRobo motion generation library to compute a collision-free joint trajectory. This trajectory is converted to a sequence of end-effector poses.

II. Experiment Details

A. Robot Setup

As shown in Figure A1, we use one or two Franka Emika Panda robots depending on the task. For single-arm tasks, one or two Intel RealSense cameras are positioned to capture the workspace. For bimanual tasks, two RealSense cameras are mounted between the two arms to provide a complete view of the scene.

Our framework generates trajectories as end-effector poses, which are converted to joint positions using inverse kinematics (IK) via PyBullet [83]. To ensure smooth motion, the IK solver minimizes the difference between successive joint configurations. The resulting joint trajectory is executed using the joint impedance controller implemented in the Deoxys library [35].

To determine the camera poses and robot base pose in a common world frame across all task settings, including the *Mug Tree* and *Bookshelf* tabletop tasks and the *Kitchen* in-the-wild task, we use a custom calibration procedure. An AprilTag board is placed in the workspace to define the world frame. Each camera, including an additional wrist-mounted gripper camera with a known transform to the end-effector, observes the board to estimate its pose relative to it. We then use forward kinematics to compute the robot base pose relative to the board.

B. Task Designs

[neil: clarify the partial success rate metrics for all of the domains]

We evaluate our method across three domains: Mug Tree, Bookshelf, and Kitchen. Each domain presents diverse objects, spatial layouts, and long-horizon manipulation goals. For each domain, we design test cases to investigate three types of generalization: new scene configurations (**SC**), new geometric constraints (**GC**), and longer task horizons (**LH**).

In the *Mug Tree* domain, the scene contains multiple colored plastic mugs placed on the table and a wooden mug tree. The mug tree has six total branches, five of which are within the robot's reach. The goal condition requires the robot to hang all plastic mugs on the table onto the mug tree. We collect five demonstrations of hanging a single mug on each peg. For **SC**, we vary the poses of the mugs and the tree. For **GC**, we place obstacles near the tree or move it to the table edge, limiting access to some pegs. For **LH**, we require sequentially hanging three or four mugs instead of one.

In the *Bookshelf* domain, the scene consists of one or more books placed on the table in front of a wooden bookshelf. The bookshelf has a brown finish and contains an additional book already leaning against it. The goal condition is to place all books from the table onto the bookshelf. We collect ten demonstrations of grasping and then inserting a book into an occupied compartment, requiring coordinated bimanual grasping and pushing an existing book to create space. For **SC**, we vary the positions of books on the table and shelf. For **GC**, we add obstacles blocking access to one of the

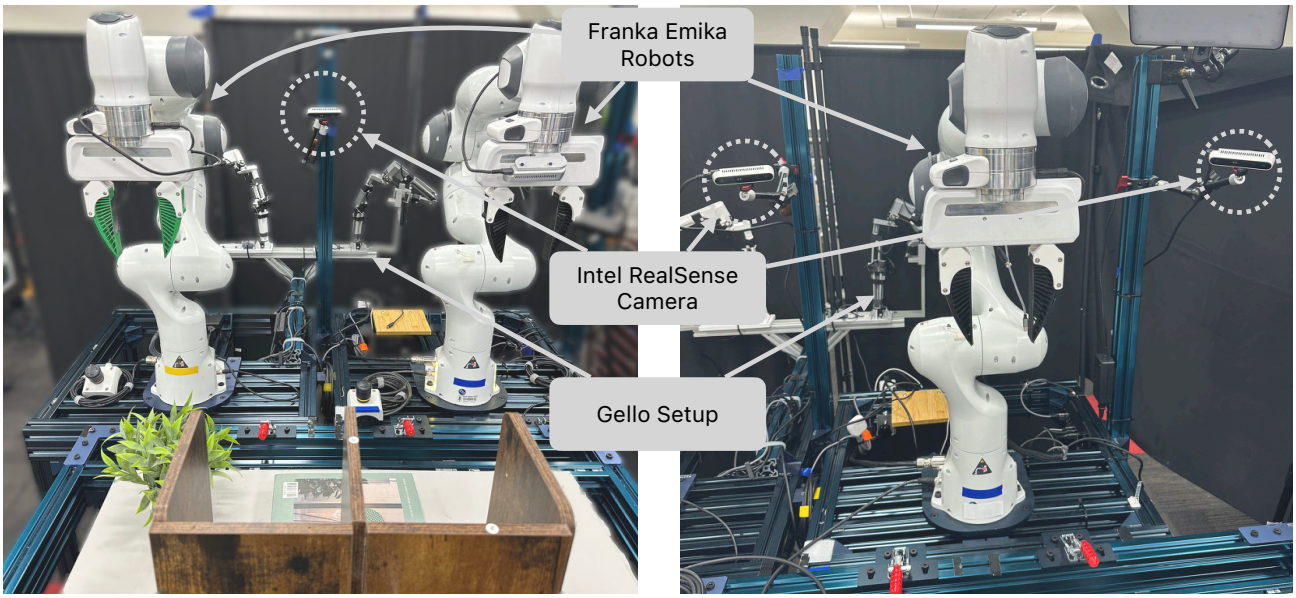

Fig. A1: Robot Setup. STACK uses two robot configurations: a bimanual setup and a single-arm setup, both based on Franka Emika Panda robot manipulators. Each setup is equipped with Intel RealSense cameras for RGB-D sensing and uses the Gello interface for teleoperation.

compartments. For **LH**, we require placing two books instead of one.

In the *Kitchen* domain, the scene includes a plastic plate, a silver sink faucet with articulated handles, and a black dishrack with capacity for several plates and bowls. Plates are initially placed on the right-hand side of the counter. The goal condition requires the robot to wash the plates and stow them on the black dishrack located on the left-hand side of the counter. We collect ten demonstrations of washing a plate and placing it on the rack. The task involves articulated manipulation of the faucet lever and handling liquids. For **SC**, we vary the initial poses of the plate and rack. For **GC**, we add dishes to block parts of the rack. For **LH**, we require washing and placing two plates sequentially.

C. Baseline Implementation

For DP3-LONG, we follow the original DP3 [10] implementation in terms of model architecture and training parameters. The model uses an action horizon of 32 steps. Observations consist of downsampled point clouds with 2048 points, encoded using a multi-stage PointNet without color channels. The diffusion model uses a step embedding dimension of 128, downsampling channel dimensions of 128, 256, and 512, a kernel size of 5, and 8 groups for normalization. A DDIM noise scheduler is used with 50 diffusion steps and a squared cosine beta schedule. Training is conducted for 300 epochs using an AdamW optimizer with a learning rate of 7×10^{-5}, betas of $(0.95, 0.999)$, weight decay of 1×10^{-6}, and a batch size of 612. Proprioceptive information is excluded to remain consistent with the original DP3 setup. For bimanual tasks, to account for calibration-induced variations in the world origin between training and evaluation, we convert all observations and actions into the robot's base frame to prevent distribution shift.

TABLE A1: Success Count.

Method	Mug Tree			Bookshelf			Kitchen		
	SC	GC	LH	SC	GC	LH	SC	GC	LH
DP3-LONG	0	0	0	0	0	0	0	0	0
DP3-SHORT	1	0	0	0	0	0	0	0	0
BLADE	8	6	2	9	4	2	8	4	2
STACK (ours)	**9**	**8**	**3**	**9**	**8**	**7**	**8**	**7**	2
w/o Spatial Aug.	0	0	0	5	2	0	**8**	**7**	**3**
w/o Effect	8	5	0	8	1	1	6	5	2

For DP3-SHORT, we extend DP3-LONG by training the model to predict an additional scalar value between 0 and 1 that indicates task progress. At test time, this value is used to terminate execution once the policy predicts that the skill has been completed, enabling the transition to the next skill.

For BLADE, we generate a planning domain definition in PDDL [84] for each domain, using manually designed predicates. Example predicates include *is-plate-placed-on-rack*(plate) and *is-peg-occupied-or-blocked*(peg). For predicate classification, we use the vision-language model GPT-4o, which takes as input an RGB image of the scene and a predicate, and returns a boolean value indicating whether the predicate holds. Planning is performed using an off-the-shelf heuristic planner [85]. The planner generates a sequence of high-level actions, which are executed using trajectory samplers learned by our framework.

[neil: add the baselines for temporal segmentation]

III. Additional Experimental Results

A. Additional Results for Main Experiment

To complement the success rate reported in Table II, Table A1 reports the number of success over 10 trials using a

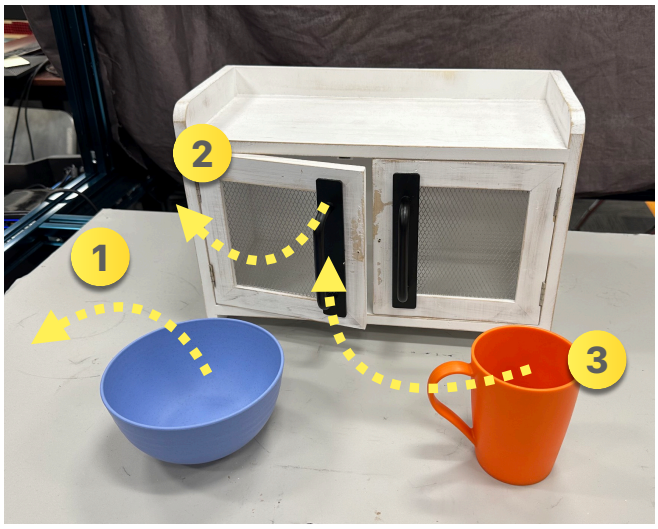

Fig. A2: Cabinet Domain. A new domain that involves manipulating an articulated object and modeling the associated geometric effect. The numbered yellow arrows illustrate the sequence of steps in the human demonstrations.

binary success criterion. In addition, we evaluate our method on training configurations following the same experiment protocol. Our method achieves partial success rates of 90%, 88%, 72%, and 90% in the *Mug Tree*, *Bookshelf*, *Kitchen*, and *Cabinet* domains, respectively. These results reinforce the conclusion we draw in the paper.

B. Experiment on Articulated Object

Our existing *Kitchen* domain demonstrates the trajectory sampler's ability to manipulate an articulated faucet handle. Here, we further introduce a new *Cabinet* domain as shown in Fig. A2, which requires modeling the geometric effects of opening an articulated cabinet door. Our method achieves average partial success rates of 85%, 80%, and 76% in the **SC**, **GC**, and **LH** settings.

C. Experiment on Entity Segmentation

In addition to the temporal segmentation experiment, we conduct an experiment to evaluate entity point cloud segmentation against expert-labeled ground truths. Our method achieves 97% mAcc and 91% mIoU across all frames and demonstrations. The performance is also robust in the *Kitchen* domain, despite challenges from clutter and occlusion.

D. Ablation on Diffusion-Based Trajectory Sampler

We perform an additional ablation in the *Mug Tree* domain by replacing the diffusion-based trajectory sampler with a Long Short-Term Memory (LSTM) network using a Gaussian Mixture Model (GMM) for action prediction, based on the implementation from RoboMimic [86]. We train this model with and without spatial augmentation. In both cases, the model fails to achieve meaningful performance, despite correctly predicting trajectories when trained on a single demonstration. We attribute this failure to the high diversity of trajectories in the training data, especially after augmentation. This experiment highlights the importance of using a diffusion model for trajectory sampling.

E. Experiment on Planning with Learning Skills

We further evaluate STACK 's ability to plan high-level skill sequences and low-level trajectories by introducing a more challenging generalization test case in the *Mug Tree* domain. In this test case, four mugs need to be placed on the mug tree and the two lower pegs are too short to accommodate tall mugs, introducing a constraint that the short mug must be placed on a lower peg. This test case requires our planner to select an alternative skill skeleton when the current one cannot be instantiated. An example of the search process and the resulting search tree is shown in Figure A3, where only the short white mug can be placed on the lower pegs. The figure illustrates a successful skill skeleton and the corresponding trajectories identified by our planner. We conducted 10 different trials and observed a partial success rate of 73.6%.

IV. Failure Analysis

We present a detailed breakdown of the failure cases and the success rate of each skill from our main experiment in Fig. A4. The analysis reveals that skill execution is the primary source of failure, followed by object detection, while the segmentation was successful in all trials. We also observe that *turn off faucet* and *hang mug* are the two most difficult skills, as the former requires applying force and the latter has low error tolerance. Since our tasks are inherently multi-step, 13.7% of failures result from error accumulation. Fig.A5 illustrates two representative challenging cases. In the left example, the bi-manual skill *grasp book* fails due to an imprecise grasp, while in the right example, the segmentation model omits faucet handles from the bounding box, resulting in incomplete point clouds and subsequent skill execution failures.

V. Expanded Discussion of Related Work

Visuomotor Skill Learning. Imitation learning is an prominent approach to learning visuomotor skills from demonstrations [8]. Diffusion-based BC improves multi-modal action modeling [9, 10] but tends to overfit to scene layouts seen in the demonstrations. Recent works tackle this by scaling up data [11] and generating synthetic demonstrations [12–14, 87]. In parallel, zero-shot labeling with foundation models reduces annotation cost at scale [15], and large multi-embodiment datasets broaden task and object diversity [16]. These methods obtain a single policy that is often difficult to generalize beyond the horizon of the training task. The discovery of reusable skills is promising to address this limitation. Some methods use RL to discover and then learn these skills [17–20, 88]. But they are restricted to simulated environments. Other methods [2, 21–23] discover skills unsupervised from demonstration and play data. However, these methods lack explicit reasoning about geometric dependencies between skills, which is imperative for effective long-horizon, sequential manipulation. More recently, language-supervised approaches

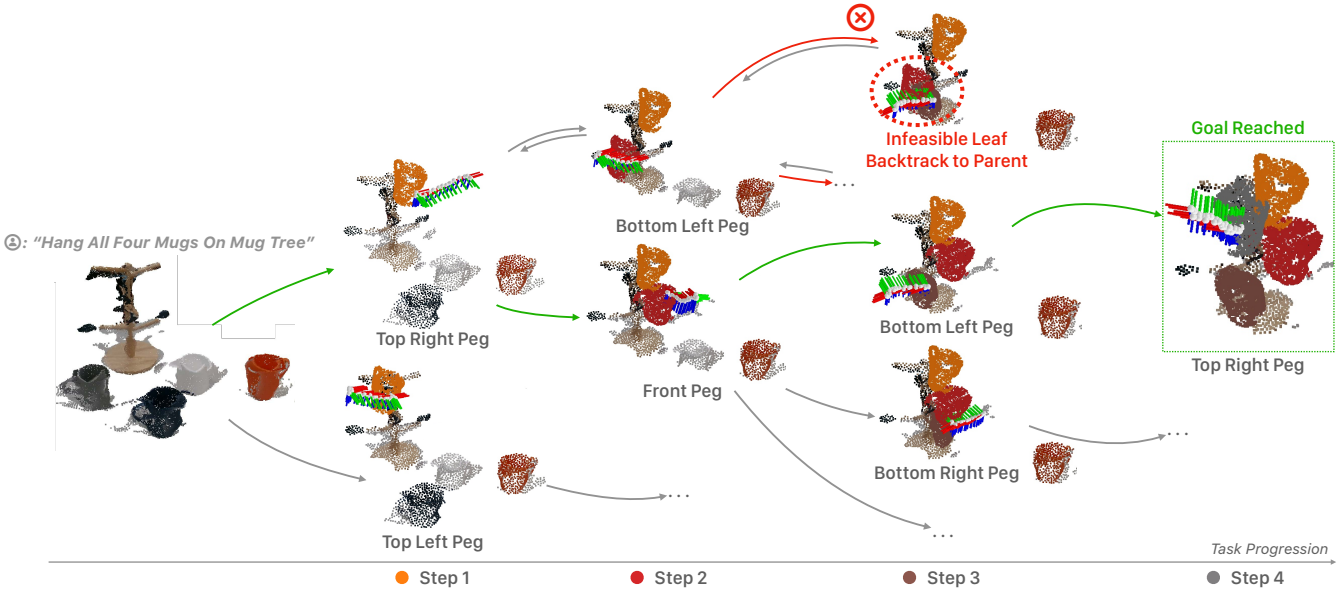

Fig. A3: Visualization of the Planning Process. Example search tree from planning for a four-mug generalization test case in the *Mug Tree* domain. STACK explores multiple skill skeletons and backtracks upon encountering infeasible ones caused by physical constraints, specifically that the two lower pegs are too short to hold tall mugs. The figure highlights a successful plan in which the short white mug is correctly assigned to a lower peg. Each node shows the expanded skill along with its sampled trajectory and predicted effect. Each step in the skill sequence is assigned a unique color, as indicated at the bottom of the figure, and the same color is used to highlight the corresponding manipulated mug. Nodes where collisions are detected are highlighted in red. The final solution, shown in green, is obtained by tracing back from a successful terminal node to the root of the search tree.

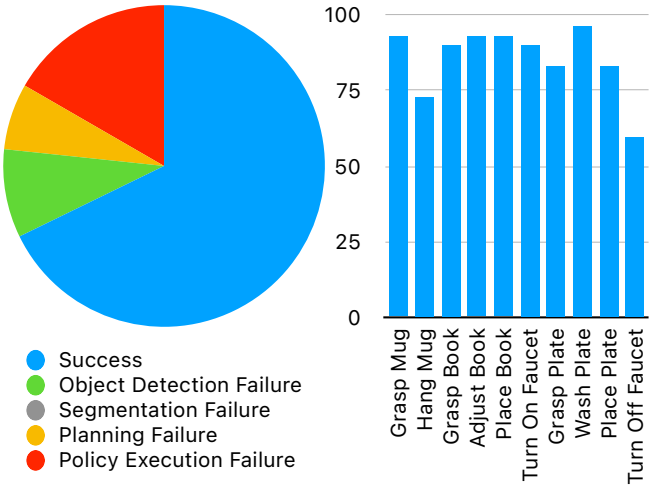

Fig. A4: Error Analysis. A detailed breakdown of the failure cases and the success rate of each skill from our main experiment

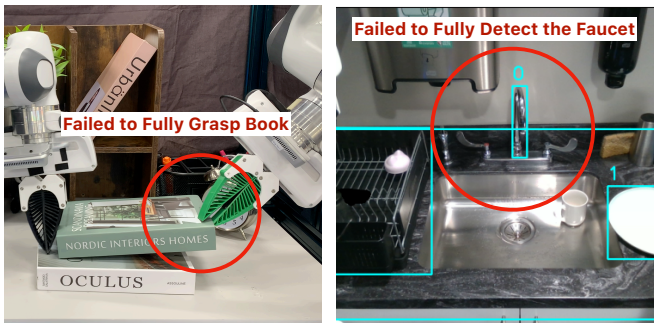

Fig. A5: Failure Examples. We present two representative failure cases. In the left example, the bi-manual skill *grasp book* fails due to an imprecise grasp, while in the right example, the segmentation model omits faucet handles from the bounding box

have been introduced to guide trajectory segmentation [89], showing that linguistic grounding can improve the composability of learned sub-tasks.

Compositional Generalization for Robot Manipulation. Generalization to novel states and goals by recombining previously learned skills is essential for robot manipulation. Towards temporal compositionality, researchers have introduced hierarchical structures into models [2, 24–27]. However, in practice, these methods often fail to model spatial generalization and therefore are limited to following sequentially specified goals. Another approach is to use symbolic action representations to compose skills, including manually-defined symbolic abstractions [28–31], learned symbolic abstractions [32–34], and approaches that incorporate additional segmentation and language annotations [5]. Recent works have also explored incorporating object-centric priors for improving spatial compositional generalization in learning-based policies and models [6, 12, 35–39, 90–99]. By contrast, we aim to construct spatiotemporal compositional action representations directly from data using the rich priors of foundation models. Crucially, it explicitly reasons about geometric constraints, thereby avoiding the limitations of purely discrete symbolic representations in physical domains.

Foundation Models for Robotics. Emerging foundation models, pre-trained on Internet-scale data, have internalized

rich semantics priors useful for robotics applications [40–43]. A recent body of literature focuses on adapting pre-trained vision-language models (VLMs) as vision-language-action (VLA) models for end-to-end visuomotor learning [44–48, 100–102]. Due to the lack of large-scale robotics data, a parallel line of works introduces structural priors into the robotics system stack, where foundation models are often leveraged for open-world perception [5, 49–51, 103–112], goal interpretation and reward specification [52–57, 113–117], model or domain specification [58–60], (visual) dynamics prediction [61–64, 118], and high-level decision-making [4, 5, 65–67, 119–123]. VLMs have also been shown to generate semantically meaningful subgoals for task and motion planning [68] and robotic affordance-grounded behaviors [4], highlighting their potential for temporal decomposition and long-horizon reasoning. Whereas prior works rely on carefully designed abstractions to apply off-the-shelf foundation models, we uniquely explore the question of *whether foundation models can provide such structural abstractions themselves* by discovering spatial and temporal structure from raw demonstrations.

VI. Expanded Discussion of Limitations

Besides limitations discussed in Section VI, we discuss the following two additional limitations and potential future work to address them. We focus on modeling geometric effects, so effects such as water boiling or a plate being cleaned are not handled. For now, we rely on the Video-LM to propose correct skill skeletons. In the future, we can extend STACK by integrating symbolic representations of non-geometric predicates, as demonstrated in BLADE [5], which uses symbolic predicates to explicitly model preconditions and effects. This would allow STACK to reason jointly over both geometric and non-geometric outcomes.

We don't directly tackle generalization to new object instances and categories. However, we observe that our 3D point cloud representations and training strategies offer some robustness to unseen instances, particularly when test-time objects share similar geometry with those encountered during training. Achieving broader category-level generalization, especially to objects with significantly different shapes or affordances, would likely require training on more diverse objects, or incorporating priors from 3D vision foundation models.

VII. Prompts

Here, we list all prompts used in the experiments, as discussed in the sections above.

Listing 1: Prompt for Proposing Coarse Temporal Segments (Single Arm)

```
You are a robot arm with 2 cameras and a gripper. The left
 and right camera can see the environment from the third
person point of view.
The robot is trying to {task_name} once in this video.

To help you perform this task, here I have provided you
with the gripper width data. The gripper width data is a 1
D array that contains the gripper width at each timestamp.
When the gripper is closed, the robot could be grasping an
 object. BUT THE ROBOT COULD ALSO BE NOT GRASPING ANYTHING
 when the gripper is closed.
You must use the video frames and additional information
to verify if the robot is actually grasping an object or
not.

{gripper_widths}

To further help you perform this task, here I have
provided you with the robot's overall joint torque data.
The joint torque data is a 1D array that contains the
joint torque at each timestamp.
The higher the joint torque, the more force the robot is
exerting.

{joint_torque}

Please look at the video content very carefully. Make sure
 the segmentation you come up with is very fine-grained,
and have meaningful segments and describe each segment.


Your job is to segment the video into meaningful **
temporal segments **.

Here is the general instruction (with the detailed
instructions below):

**Step 1**: Rough Segmentation:
Each segment should be open gripper --> close gripper -->
open gripper. This is a new segment.
Each segment should be a dictionary with the following
keys.
- approach_start (format mm:ss):
   A second or two before the robot arm comes into contact
    with the target object / environment
   OR a second or two seconds before the gripper closes
   and marks the start of free-space motion.
   Both of these cases are acceptable, but you should
   choose the one that comes last.
- approach_end (format mm:ss):
   the end time of the the robot arm fully grasped the
   target and the gripper is closed.
   This segment MUST END when the gripper is closed.
- interaction_start (format mm:ss):
   - The start of the meaningful interaction, when the
   gripper is holding / touching / releasing / grasping an
    object.
   - This can occur after a free-space motion period
   following the gripper closure.
   - This is when the gripper is first closing, and AFTER
   FREE-SPACE MOTION IS COMPLETE IF THERE'S FREE-SPACE
   MOTION.
- interaction_end (format mm:ss):
   - The end of the meaningful interaction.
   - If grasping, the moment AFTER the robot releases the
   target object (the gripper is open).
   - If interacting without grasping, after the gripper
   opens and the robot is no longer in contact with the
   target object.
   - In some cases, the interaction_end may occur before
   the gripper is fully open (if justified, please explain
   ).

Here are a few examples of the rough segmentation based on
 gripper open and close signal, and the torque signals.
Example 1: pour tea
1. approach_start: just before the robot grasps the teapot
2. approach_end: after the gripper closes
3. interaction_start: just after the gripper closes
4. interaction_end: the gripper releases the teapot after
placing it on the table

First, watch the video from start to finish and determine
how many segments there are in the video by counting the
number of times the gripper opens --> closes --> opens.

** Detailed Instructions for Step 1 Rough Segmentation **
You must follow these important guidelines very very
carefully.
1. Watch the video from the beginning to the end first
before making any segmentations or before answering any
questions. This step is very important.
2. Count the number of times the gripper opens --> closes
--> opens. Let the count be N. The number of output
temporal segments should be at least N segments.
3. For each of the segments, explain WHAT and WHY the
robot is doing what it is doing. Consider the overall
context of the video.
4. Try not to include free-space motion in your
segmentation. This is very important. Free-space motion is
 when the robot is holding the object the robot is grasing
 is not in contact with the environment. In other words,
pure transport motion. If the robot is not in contact with
 the environment the torque should be low.
If you determine the segment contains free-space motion,
but the torque is high during the segment, then the high
torque part is not free space motion, so you must include
this segment.
5. If the robot is trying to use the object in the robot
gripper, or the gripper itself to interact with some other
 object, that is NOT free space motion.
6. If the segment begins with the robot grasping and
object, BE SURE TO INCLUDE THE GRASPING time segment in
your segment as well. And explain this.
7. Remember WHICH ARM YOU ARE LOOKING AT, and don't let
the other arm distract you from returning the correct
segmentations.
8. The interaction start to end duration should be at
least 4 or more seconds. This is very important.
9. If the segment is grasping an object, interaction_start
 should be RIGHT BEFORE the gripper closes!

**Step 2**: Segmentation Refinement:
For each segment, determine if it's necessary to split the
 segment into new chunks.
Usually, if a segment is longer than 15 seconds and
interacts with more than 1 object, then you should split
the segment.
If you should split, ALWAYS split the segment into THREE
NEW segments.
Here are a few examples:
Example 1:
If the segment is grasp the tea pot, pour tea into the cup
, and return the tea pot to the table, then you should
split this into three segments:
1. grasp the tea pot
2. pour tea into the cup
3. return the tea pot to the table
Example 2:
If the segment is grasp the plate from table, hold the
plate wash it under water, and place the plate onto the
drying rack, then you should split this into three
segments:
1. grasp the plate from table
2. hold the plate wash it under water
3. place the plate onto the drying rack
Example 3:
If the segment is pouring water from a bowl to another
bowl, and then placing the bowl back on the table, then it
 should be split into three segments
1. grasping the bowl
2. pouring water from a bowl to another bowl
3. placing the bowl back on the table
where you shouldn't split:
Example 4:
If the segment is place book onto the bookshelf, then it
SHOULD NOT be split, because placing the book onto the
bookshelf is a single action.
Example 5:
If the segment is turn on the light by flipping a switch,
then it SHOULD NOT be split, because flipping the light
switch is a single action.
Example 6:
If the segment is using the arm to wipe the table
repeatedly, then it SHOULD NOT be split, because wiping
the table is a single action.
```

```
Example 7:
If the left arm is grasp the plate and transporting it to
the dishrack, this should be split into
- grasping the plate
- transporting the plate
- and placing the plate, and make sure to capture to the
full insertion motion
This is very important to split this segment into these
meaningful segments

** Detailed Instructions for Step 2 Segmentation
Refinement **
- For each segment, determine if it's necessary to split
the segment into new chunks. Look at the examples and
justify your reasoning by describing how it relates to the
 examples.
- If you can confidently determine the robot is grasping
something as part of the rough segment, then the grasping
phase must be split into its own segment.
- Explain your reasoning on why to split it or not
- If a segment is long, and similar to the examples above
where splitting is reasonable, then it should probably be
split.
- Split the segment by providing a list of new
interaction_start and interaction_end timestamps.

**Output Validation**
- First, follow the instruction above and complete step 1
and step 2 very carefully. Consolidate all of your
segments.
- You are a reasoning model so make sure to reason. Please
 explain your reasoning and think carefully step by step
before answering.
- For each segment you generate in the output, make sure
the segment is at least 4 seconds long.
- If no segment is found but you see the gripper state
changed in the video, you want to double check everything
from start to finish to re-propose segments
- Verify for each segment, from approach_start to
interaction_end, the gripper state changes from open to
close to open. This is very important

**Error Checking**
If no segments found, re-watch the video and re-evaluate
everything from the first instruction to the last
instruction.

** Output Format **
Your output should contain your reasoning, and 1 code
block with all of the segments in the following format:
No need to include approach and reset away segments.
'''
[
   {
      "interaction_start": "xxx",
      "interaction_end": "xxx",
      "skill_description": "description of the skill. The
      description should only include the specific arm.
      LEFT or RIGHT arm.",
      "skill_name": "snake_case_name_for_the_skill"
   },
   ...
]
'''
```

Listing 2: Prompt for Proposing Coarse Temporal Segments (Bimanual)

```
The robot is trying to complete the following task:
{task_name}

** Your Primary Objective: **
Temporally segment video into bimanual coordination phases
 using sensor data and visual context.

**Robot Capabilities:**
- Two arms (left=black gripper, right=light green gripper)
 with parallel jaw grippers
- One camera: Center camera provide third-person view

The left arm is masked in a light blue color, and the
right arm is masked in red light red color.
Remember this crucial information when looking at the
video.
This information can help you disambiguate between the
left and right arm.

**Data Stream Definitions:**
1. **Gripper Widths** (Left/Right):
   - Closing/opening = object interaction (grasp/release).

2. **Joint Torque** (Left/Right):
   - Large value might indicate forceful coordination.
   - Synchronized torque spikes might indicate
   collaborative manipulation.

Analyze the following data streams with these guidelines:
- Left gripper open / closed, in [MM:SS, o/c ... ] format.
 o is for open and c is for closed. Here is the data: {
gripper_widths_left},
- Left torque, in [MM:SS, torque value ...] format. Here
is the data {joint_torque_left}

- Right gripper open / closed, in [MM:SS, o/c ... ] format
. o is for open and c is for closed. Here is the data: {
gripper_widths_right}
- Right torque, in [MM:SS, torque value ...] format. Here
is the data {joint_torque_right}


**Objective:**
Analyze a robot manipulation video and identify motion
segments that should be categorized as bimanual
coordinated tasks. Return timestamps in "mm:ss" format.

**Bimanual Coordination Manipulation Definition and
Guidelines**:
   - When trying to determine if the motion is bimanual
   coordinated manipulation, ask yourself the question, "
   Is this possible without the second arm?" "Can I still
   achieve the desired outcome with a single arm?"
   - Look at the surrounding context of the video.
   - Both arms must be actively contributing to the task
   through the following criteria. These are suggestions,
   not strict rules, not exhaustive list:
     - both arms are in contact with the same object
     - force-coupled motion (torque synchronization)
     - and/or spatial coordination (working in shared
     workspace region)
     - and/or complementary roles (one stabilizes while
     other manipulates)
     - anything that can't be done with a single arm
   - Some examples of bimanual coordinated manipulation:
      - Left arm holding a bowl while the right arm is
      stirring the soup. Both arms are actively
      contributing to the task of stirring the soup.
      - The left arm is stabilizing the plate/book/object
      while the right arm is trying to grasp the plate/
      book/object.
   - Some examples of not bimanual coordinated
   manipulation:
      - One arm is doing something while the other arm is
      just holding an object, not really doing anything.
      - MOVING the object around is not bimanual
      coordinated manipulation. It's free-space motion.
   - DO NOT INCLUDE IF the two arms are MOVING AWAY from
   each other

**Identification Guidelines:**
- **Start Time:** First frame showing bimanual coordinated
 motion is starting (not exhaustive list).
   - Increasing torque values on both arms is usually a
   good indicator of bimanual coordinated motion.
   - The start time should be the GRIPPER CLOSING TIME of
   ONE OF THE ARMS, or when one of the arms comes into
   contact with the target object.
   - Both arms are APPROACHING the target object.
   - You can give a 2 second buffer by subtracting 2
   seconds from the start time, this is because you want
   to include some of the approach motion.
   - If the segment ends with the grippers all closed,
   then the segment should start with when the grippers
   are open.
- **End Time:** Last frame before either:
  1) Object fully grasped or placed down
  2) Transition to the next motion segment
  3) There is no longer two arm coordinated motion. Only
```

```
  one arm is doing moving motion. Pay attention to the
  context of the video.
  4) Torque values dropping significantly
  5) The robot is clearly doing something else now. Pay
  attention to the context of the video.
- **Minimum Segment Duration:** 3 seconds for meaningful
actions
- **Maximum Segment Duration:** 10 seconds. If the segment
 is longer than 10 seconds, then it's probably not a
bimanual coordinated manipulation.

**Processing Instructions:**
1. **Video Analysis:**
   a) Perform full video scan (00:00 - end). When scanning
   , as part of your output, explain very carefully not
   just what you are seeing, and why you think the robot
   is doing what it is doing, and why this motion
   contribute to the overall goal of the task.
   b) Identify all potential bimanual coordinated motion
   segments. Find everything you possibly can, don't miss
   anything.
   c) For each candidate segment, make sure to explain why
    it is bimanual coordinated manipulation, and why the
   robot is doing what it is doing, not just what you are
   seeing.
   d) For each segment, perform the per-segment evaluation
   .

2. **Per-Segment Evaluation:**
   For each candidate segment:
   a) Review the motion definition very very carefully,
   follow the definition make sure the segment is bimanual
    coordinated manipulation.
   First, look at the left arm. Then, look at the right
   arm. Then look at them together.
   Explain WHY it is bimanual coordinated manipulation,
   and why the robot is doing what it is doing, not just
   what you are seeing.
   b) Check both arm gripper state, torque values.
   c) Analyze object interaction context
   d) Verify the segment is in fact bimanual coordinated
   manipulation. If not, discard it. Do not include it in
   the output.
   e) Make sure the end time is not too late. We don't
   want to include too much free-space motion. The segment
    shouldn't be too long.

**Output Format:**
Return list of dictionaries with strict structure:
```
[
    {
        "interaction_start": "mm:ss",
        "interaction_end": "mm:ss",
        "skill_name": "snake_case_name_for_the_skill",
        "skill_description": "description of the skill."
    },
    ...
]
```

**Critical Requirements:**
1. Explicitly state conflict resolutions when:
   - Arm movements are ambiguous
   - Torque values border threshold
   - Spatial relationships are unclear
2. Prioritize task context over raw motion metrics
3. There should usually be at least 1 segment. If there's
no bimanual segment then you should really double check
everything.

Think extremely carefully, explain your reasoning step by
step. In your explanation, do not include code blocks.
Explain your reasoning in the output in great detail in
the output.
```

Listing 3: Prompt for Aggregating Proposed Temporal Segments

```
You are watch a bimanual robot video.

**Scene configuration:**
- 2 arms with parallel jaw grippers (LEFT has BLACK
gripper)
- 1 camera: Center camera provide third-person view. This
is the view you are watching.
- Tabletop environment
- The robot is trying to {task_name}
- The left robot is masked in blue, and the right robot is
 masked in red. Remember this crucial information when
looking at the video.

**Proprioceptive data as context:**
- Left gripper open / closed, in MM:SS : open/closed
format. {gripper_widths_left},
- Left torque, in MM:SS : torque value format. {
joint_torque_left}

- Right gripper open / closed, in MM:SS : open/closed
format. {gripper_widths_right}
- Right torque, in MM:SS : torque value format. {
joint_torque_right}
You might jointly use the video and the proprioceptive
data to determine if the temporal segments are correct.

**Temporal Segmentation Explanation:**
- You are provided with a list of temporal segments.
- Conceptually, a temporal segment is defined as the
period of time when the robot is doing meaningful
manipulation, towards the goal of the task: {task_name}.
- Each temporal segment is a dictionary containing at
least the following keys:
  - "interaction_start": "mm:ss", the start time of the
  segment.
  - "interaction_end": "mm:ss", the end time of the
  segment.
- There could be multiple temporal segments in the video.
- The temporal segments that you are given might contain
overlaps, might contain gaps, and might contain semantic
errors.

The temporal segments sorted in time:
{temporal_segments}

**Task:**
You will watch the video and the corresponding context
information from start to end.
You will look at each of the temporal segments and
determine if they are correct.
The goal is to return a new list of temporal segments that
 are correct. If you look at the
new list of temporal segments, sequentially along with the
 video, they should tell a complete story and make sense.

**Guidelines:**
1. Start by watching the video from start to end while
considering the sensor data as well as the temporal
segments.
2. For each temporal segment, think about and explain why
the robot is doing what it is doing. Then look at the
video as a whole, and think very very deeply about if this
 temporal segment is truly correct or not.
3. Identify overlapping temporal segments.
If the overlapping amount is small, and it makes sense to
keep both segments, keep both segments. Especially if the
overlap is between two single arm segments.
If the overlapping amount is significant (3+ seconds),
then you need to make a decision on which segment to keep.
If the overlapping segments are one bimanual one single
arm, and the overlapping amount is greater than 2 seconds,
 then you probably want to keep the bimanual segment.
you need to make a decision on which segment to keep.
Remember, if segments are overlapping, you can only keep
1.
4. DO NOT MODIFY the start and end time of the temporal
segments. Only determine if the temporal segment is
correct or not.
5. Look at the output temporal segments, and explain why
you made the decisions you made. Finally, order the
temporal segments by the start time.
6. Discard any segment that is less or equal to 3 seconds
long. If a segment is too short, that's usually an
indicator that the segment is not meaningful and incorrect
.
7. Under absolutely no circumstances should you modify the
 segment starting and end time. You should only determine
if the segment is correct or not.
```

```
8. The segments MUST BE LONGER THAN 2.5 SECONDS!! IF NOT
DISCARD IT!
9. The skill and skill description fields could be
inherited from the input, if the segment is correct. Read
the skill description and skill name fields very carefully
 and understand them.

**Output Format:**
list of dictionaries in a code block with the following
keys:
"interaction_start": "mm:ss",
"interaction_end": "mm:ss",
"skill_name": "a short name for the skill, such as
pick_up_book_bimanul, or turn_off_faucet_left_arm, or
pick_up_mug. The name should be snake case"
"skill_description": "a description of this skill and how
this skill is used in the demonstration video. If this is
a single arm skill, only mention the arm that is being
used."

Final Instructions:
- Follow the above guidelines very very carefully. For
each segment, follow the guidelines EXACTLY and EXPLAIN
- The output should be a valid JSON object, with valid
explanations.
- Be sure to explain your reasoning in great detail.
Provide your reasoning steps in the output!
```

Listing 4: Prompt for Refining Aggregated Temporal Segments

```
Ultra-deep thinking mode. Greater rigor, attention to
detail, and multi-angle verification.
Start by outlining the task and breaking down the problem
into subtasks. For each subtask, explore multiple
perspectives,
even those that seem initially irrelevant or improbable.
Purposefully attempt to disprove or challenge your own
assumptions at every step.
Triple-verify everything. Critically review each step,
scrutinize your logic, assumptions, and conclusions,
explicitly calling out uncertainties and alternative
viewpoints. Independently verify your reasoning using
alternative methodologies or tools,
cross-checking every fact, inference, and conclusion
against external data, calculation, or authoritative
sources.
Deliberately seek out and employ at least twice as many
verification tools or methods as you typically would.
Use mathematical validations, web searches, logic
evaluation frameworks, and additional resources explicitly
 and liberally to cross-verify your claims.
Even if you feel entirely confident in your solution,
explicitly dedicate additional time and effort to
systematically search for weaknesses,
logical gaps, hidden assumptions, or oversights. Clearly
document these potential pitfalls and how you've addressed
 them.
Once you're fully convinced your analysis is robust and
complete, deliberately pause and force yourself to
reconsider
the entire reasoning chain one final time from scratch.
Explicitly detail this last reflective step.

<task>

** Setup **
You are given a list of videos. Each video is a robot
demonstration.
The video contains a robot performing a task with goal {
task_name}.

** Input **
You are given {num_videos} videos. Associated with each is
 a list of temporal
segements. A temporal segment is defined as a period of
time (mm:ss format) such that
the robot is doing something meaningful. For example, if
the video shows the robot
grasping a cup, then placing the cup on a shelf, you would
 expect to have two temporal segments,
one for grasping the cup, and one for placing the cup on a
 shelf.
The expected number of temporal segments for each
demonstration should be exactly the same
as the other demonstrations.

** Your Task **
Some of the videos may have inconsistent temporal segments
 compared to others.
Detect which of the videos have inconsistent temporal
segments compared to others?
Think very carefully step by step before answering.

It is possible that all videos have consistent temporal
segments. In that case,
provide an empty list. But you still have to explain your
reasoning in great detail.

First, identify the videos that have consistent temporal
segments.
Second, understand what is the correct temporal
segmentation by looking at all of the input.
At least one or several of the videos should have the
correct temporal segmentation. Therefore
start with those videos and understand the task, videos,
context, and everything in the input prompt.
Third, re-propose the temporal segments for the videos
that have inconsistent temporal segments.

** Output **

1. Provide the list of videos that have inconsistent
temporal segments.
2. For each video that you believe that have inconsistent
temporal segments, do the following:
    1. Re-propose the temporal segments for this video.
    2. Check and verify the re-proposed temporal segments,
     such that they are consistent with the other videos.
    3. Explain in great detail why the re-proposed
    temporal segments are consistent with the other videos
    .
3. For the newly proposed temporal segments, provide them
in JSON format, in code.

'''json
[
    open curly brackets
        "video_id": the integer id of the video (remember
        to use zero-indexing)
        "segments": the list of the NEW temporal segments
    close curly brackets
]
'''

** Verification and Check **

1. For each video that you believe that have inconsistent
temporal segments, verify that the re-proposed temporal
segments are consistent with the other videos.
2. If the re-proposed temporal segments are consistent
with the other videos, then you have successfully detected
 the inconsistent temporal segments.
3. If the re-proposed temporal segments are not consistent
 with the other videos, then you have failed to detect the
 inconsistent temporal segments.
4. For each video that you believe that have consistent
temporal segments, verify that the temporal segments are
consistent with the other videos.
5. If the temporal segments are consistent with the other
videos, then you have successfully detected the consistent
 temporal segments.
6. If the temporal segments are not consistent with the
other videos, then you have failed to detect the
consistent temporal segments.

** Final Instructions **

- Think very carefully step by step
- Provide your detailed reasoning
- Read every instruction carefully
- Watch all the videos and read the temporal segments
carefully before answering

</task>
```

Listing 5: Prompt for Naming Relevant Entities in Each Skill Segment

```
You are given a list of videos of the robot doing a task.
These videos are
all demonstrating the same task {task_name}.

You are also given a list of temporal segments. One
segment per video. Please focus on the temporal segment,
but also
pay attention to the broader context of the video.

Your task is to identify the object of interest in the
temporal segment. The object(s) of interest is the object
that is
most relevant to the task and the temporal segment. You
must also identify the parameters for the object.
Here are some examples with explanations:

** Example 1 **:
Task: Grasping a orange cut that is sitting on the table.
Target object: cup
Target object parameters: orange
Moving object: None
Moving object parameters: None
Explanation: because the robot is just grasping the cup,
there is no moving object.
By the end of this segment, the robot is expected to be
grasping the mug.

** Example 2 **:
Task: placing a blue bow onto the wooden shelf
Target object: shelf
Target object parameters: wooden color
Moving object: bow
Moving object parameters: blue
Explanation: because the robot is placing the bow onto the
 shelf, the bow is the moving object.
In this case, the robot should be clearly holding the bow
in its hand in the beginning of this segment.

** Example 3 **:
Task: grasping green bow using both robot arms from a
countertop before washing it in the bathroom
Target object: bow
Target object parameters: green
Moving object: None
Moving object parameters: None
Explanation: because the two robot arms are both working
together to grasp the bow, there is no moving object.
In this case, the robot should be clearly not holding
anything in the beginning of the segment.
By the end of the segment, the robot should be holding
something.

** Example 4 **:
Task: washing a black frying pan under the kitchen sink
faucet with running water.
Moving object: frying pan
Moving object parameters: black
Target object: sink faucet hand and handles
Target object parameters: silver
Explanation: because the robot is washing the frying pan,
the frying pan is the moving object.
We specify the target object as the sink faucet because
the robot is washing the frying pan under the faucet.
This is very important. No other object is relevant
besides the position of the faucet.

** Additional Instructions **:
- The object of interest names must be as descriptive as
possible.
- You must watch all of the videos and the temporal
segments before answering.
- For each skill, you must identify the target object, and
 the moving object.
- There MUST be a target object.
    - To determine the target object, focus on the object
    that the position matters the most for the robot to
    pay attention to. Look at the examples above carefully
    .
- If the interaction is between the robot arm(s) and an
object, then there could
be no moving object. So it's possible that there's no
moving object!
- You must consider all of the videos and provide one
common proposal for one moving object (possibly none) and
one target object.
- The moving and target cannot be the same object.
Therefore, if the moving and target are the same, keep the
 target object
and set the moving object to null.

** Output format **

Output the list of objects as a python code block.

'''json
open_curly_bracket
    "moving": "object name",
    "moving_parameters": "something about the object,
    focus on color, shape position",
    "target": "object name",
    "target_parameters": "something about the object,
    focus on color, shape position"
close_curly_bracket
'''

Watch the videos carefully and think step-by-step. You
must also provide your detailed reasoning.
```

Listing 6: Prompt for Generating Candidate Skill Skeletons

```
You are a helpful robot agent trying to generate a list of
 possible task plans. Read this prompt very carefully
and following the instructions.

** Background Information **
You are in an environment with robot arm(s). The robot
arms is trying to following the human instruction to
complete
some tasks. You are given an video of the robot going
these tasks, as well as the timestamps where the robot
arms
are completing certain actions.

You are given {video_count} demo videos above. Each video
is a demonstration of the robot arms

** Viable Actions **
You are given a list of actions the robots can take. Each
action has a name, and a short description of the purpose
of such
actions. The format of the actions is a list of
dictionaries with the following keys:

"interaction_start": "MM:SS", the start time of this
specific action in the video
"interaction_end": "MM:SS", the end time of this specific
action in the video
"reasoning": "why this action is important...",
"skill_name": "snake case name for the action",
"skill_description": "description of the action"

** Viable Actions From Demonstrations **
You are provided with several demonstrations. You are
provided a list of actions for each demonstration.
Here is the list of actions in JSON format:
{actions}

** Completing a Task with the Robots **
The human is interested in completing this task {task}.
You must think about how you can complete this task
with the given set of actions. You need to look at the
demonstration video and the actions, and think about how
the actions can be used to complete the task.

** Plan Generation **
- First, you need to list out all of the unique actions
from all of the actions provided. You can merge actions
that are similar.
To perform this step, you need to look at the videos
carefully, and also correspond the actions to the videos.
Then identify the unique actions and merge them.
- Read the actions very carefully, understand each action.
```

```
- Watch the video and understand the task. Explain how
each action is used in the video.
- Think about the task very carefully. For each action
provided, explain how the action contributes to completing
 the task.
- Generate a list of possible task plans with the given
set of possible actions. Each plan should be able to
complete the task upon plan verification.
- For each plan, explain why you generated this plan.

** Output Format **
- You must output a list of possible task plans.
- Each task plan must be a list of actions. The output
must be a JSON code block. the list should look like this:
```
[
    [# this is the start of the first plan
        "action_1",
        "action_2",
        ...
    ],
    [# this is the start of the second plan...
        "action_1",
        "action_2",
        ...
    ],
]
```
- Each action in the task plan must be a snake case name
for the action.
- You must also provide a short explanation of why you
generated this plan.
- There must only be 1 code block in your output.

** Additional Instructions **
- Please propose at least 2 task plans.
- Try to change things around, you can try to skip an
action, or repeat a sequence of actions, if it makes sense
.
- You must not use any other actions that are not provided
 in the input.
- You must use the actions provided in the input.
- You must not use any other actions that are not provided
 in the input.
- You must think extremely carefully before answering.
"""
```

REFERENCES

[1] 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] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar, "Mimicplay: Long-horizon imitation learning by watching human play," in *CoRL*, 2023. 1, 2, 12, 13

[3] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," *arXiv preprint arXiv:2410.13126*, 2024. 1

[4] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes *et al.*, "Do as i can and not as i say: Grounding language in robotic affordances," in *arXiv preprint arXiv:2204.01691*, 2022. 1, 2, 14

[5] W. Liu, N. Nie, R. Zhang, J. Mao, and J. Wu, "Blade: Learning compositional behaviors from demonstration and language," in *CoRL*, 2024. 1, 2, 5, 13, 14

[6] S. Cheng, C. Garrett, A. Mandlekar, and D. Xu, "Nod-tamp: Multi-step manipulation planning with neural object descriptors," *arXiv preprint arXiv:2311.01530*, 2023. 1, 2, 13

[7] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6394–6400. 1

[8] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *ICML*. PMLR, 2016, pp. 49–58. 2, 12

[9] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 2, 12

[10] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *RSS*, 2024. 2, 5, 9, 11, 12

[11] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024. 2, 12

[12] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023. 2, 12, 13

[13] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu, "Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning," in *RSS*, 2025. 2, 12

[14] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," in *ICRA*, 2025. 2, 12

[15] N. Blank, M. Reuss, M. Rühle, Ömer Erdinç Yağmurlu, F. Wenzel, O. Mees, and R. Lioutikov, "Scaling robot policy learning via zero-shot labeling with foundation models," 2024. 2, 12

[16] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju *et al.*, "Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation," 2025. 2, 12

[17] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, 2009. 2, 12

[18] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman, "Dynamics-aware unsupervised discovery of skills," 2020. 2, 12

[19] M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel, "Cic: Contrastive intrinsic control for unsupervised skill discovery," *arXiv preprint arXiv:2202.00161*, 2022. 2, 12

[20] S. Rho, L. Smith, T. Li, S. Levine, X. B. Peng, and S. Ha, "Language guided skill discovery," *arXiv preprint arXiv:2406.06615*, 2024. 2, 12

[21] 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. 2, 12

[22] S. Nair and C. Finn, "Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation," in *ICLR*, 2020. 2, 12

[23] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song, "Xskill: Cross embodiment skill discovery," in *CoRL*, 2023. 2, 12

[24] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn, "Yell at your robot: Improving on-the-fly from language corrections," *arXiv:2403.12910*, 2024. 2, 13

[25] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine, "Multi-stage cable routing through hierarchical imitation learning," *IEEE Transactions on Robotics*, 2024. 2, 13

[26] S. Pirk, K. Hausman, A. Toshev, and M. Khansari, "Modeling long-horizon tasks as sequential interaction landscapes," in *CoRL*, 2020. 2, 13

[27] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," in *ICRA*, 2023. 2, 13

[28] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined Task and Motion Planning through an Extensible Planner-Independent Interface Layer," in *ICRA*, 2014. 2, 13

[29] M. Toussaint, "Logic-Geometric Programming: An optimization-based approach to combined task and motion planning," in *IJCAI*, 2015. 2, 13

[30] A. Curtis, X. Fang, L. P. Kaelbling, T. Lozano-Pérez, and C. R. Garrett, "Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances," in *ICRA*, 2022. 2, 13

[31] Z. Yang, C. R. Garrett, T. Lozano-Pérez, L. Kaelbling, and D. Fox, "Sequence-based plan feasibility prediction for efficient task and motion planning," in *RSS*, 2023. 2, 13

[32] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018. 2, 13

[33] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. Kaelbling, and J. B. Tenenbaum, "Predicate invention for bilevel planning," in *AAAI*, 2023. 2, 13

[34] A. Ahmetoglu, E. Oztop, and E. Ugur, "Symbolic manipulation planning with discovered object and relational predicates," *arXiv preprint arXiv:2401.01123*, 2024. 2, 13

[35] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *6th Annual Conference on Robot Learning*, 2022. 2, 10, 13

[36] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende *et al.*, "Interaction networks for learning about objects, relations and physics," *Advances in neural information processing systems*, vol. 29, 2016. 2, 13

[37] E. Jang, C. Devin, V. Vanhoucke, and S. Levine, "Grasp2vec: Learning object representations from self-supervised grasping," *arXiv preprint arXiv:1811.06964*, 2018. 2, 13

[38] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," *Advances in neural information processing systems*, vol. 33, pp. 11 525–11 538, 2020. 2, 13

[39] N. Heravi, A. Wahid, C. Lynch, P. Florence, T. Armstrong, J. Tompson, P. Sermanet, J. Bohg, and D. Dwibedi, "Visuomotor control in multi-object scenes using object-aware representations," in *ICRA*. IEEE, 2023, pp. 9515–9522. 2, 13

[40] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, Z. Zhao *et al.*, "Toward general-purpose robots via foundation models: A survey and meta-analysis," *arXiv:2312.08782*, 2023. 2, 14

[41] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023. 2, 14

[42] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," *arXiv preprint arXiv:2402.05741*, 2024. 2, 14

[43] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," *arXiv preprint arXiv:2303.04129*, 2023. 2, 14

[44] A. B. et al, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," 2023. 2, 14

[45] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *RSS*, Delft, Netherlands, 2024. 2, 14

[46] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Open-vla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024. 2, 14

[47] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024. 2, 14

[48] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan, "3d-vla: A 3d vision-language-action generative world model," *arXiv preprint arXiv:2403.09631*, 2024. 2, 14

[49] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009. 2, 14

[50] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in neural information processing systems*, vol. 33, pp. 9912–9924, 2020. 2, 14

[51] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, "Physically grounded vision-language models for robotic manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 462–12 469. 2, 14

[52] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," *arXiv preprint arXiv:2306.08647*, 2023. 2, 14

[53] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv:2310.12931*, 2023. 2, 14

[54] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *CoRL*, 2024. 2, 14

[55] Y. Wang, T.-H. Wang, J. Mao, M. Hagenow, and J. Shah, "Grounding language plans in demonstrations through counterfactual perturbations," 2024. 2, 14

[56] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Automated dense reward function generation for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023. 2, 14

[57] S. Patel, X. Yin, W. Huang, S. Garg, H. Nayyeri, L. Fei-Fei, S. Lazebnik, and Y. Li, "A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards," *arXiv preprint arXiv:2502.08643*, 2025. 2, 14

[58] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023. 2, 14

[59] J. Hsu, J. Mao, and J. Wu, "Ns3d: Neuro-symbolic grounding of 3d objects and relations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2614–2623. 2, 14

[60] A. Athalye, N. Kumar, T. Silver, Y. Liang, T. Lozano-Pérez, and L. P. Kaelbling, "Predicate invention from pixels via pretrained vision-language models," *arXiv preprint arXiv:2501.00296*, 2024. 2, 14

[61] Y. Du, M. Yang, P. Florence, F. Xia, A. Wahid, B. Ichter, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum *et al.*, "Video language planning," *arXiv preprint arXiv:2310.10625*, 2023. 2, 14

[62] S. Li, Y. Gao, D. Sadigh, and S. Song, "Unified video action model," *arXiv preprint arXiv:2503.00200*, 2025. 2, 14

[63] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann, "Diffusion forcing: Next-token prediction meets full-sequence diffusion," *Advances in Neural Information Processing Systems*, vol. 37, pp. 24 081–24 125, 2024. 2, 14

[64] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel, "Learning universal policies via text-guided video generation," *Advances in neural information processing systems*, vol. 36, pp. 9156–9172, 2023. 2, 14

[65] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022. 2, 14

[66] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-anything: Automating real-world robots using vision-language models," in *CoRL*, 2024. 2, 14

[67] C. Wang, F. Xia, W. Yu, T. Zhang, R. Zhang, C. K. Liu, L. Fei-Fei, J. Tan, and J. Liang, "Chain-of-modality: Learning manipulation programs from multimodal human videos with vision-language-models," *arXiv preprint arXiv:2504.13351*, 2025. 2, 14

[68] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, "Guiding long-horizon task and motion planning with vision language models," in *ICRA*, 2024. 2, 14

[69] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," in *NeurIPs*, 2023. 3, 8

[70] L. Medeiros, "Lang-Segment-Anything: Language-guided instance segmentation," 2023. 3, 8

[71] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. 3, 9

[72] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart, "nvblox: Gpu-accelerated incremental signed distance field mapping," 2024. 4, 10

[73] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, "curobo: Parallelized collision-free minimum-jerk robot motion generation," 2023. 4, 10

[74] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators," 2023. 4

[75] C. Lea, A. Reiter, R. Vidal, and G. Hager, "Segmental spatiotemporal cnns for fine-grained action segmentation," in *ECCV*, 2016. 4

[76] G. Ding, F. Sener, and A. Yao, "Temporal action segmentation: An analysis of modern techniques," 2023. 5

[77] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs, "Universal visual decomposer: Long-horizon manipulation made easy," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6973–6980. 5

[78] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *ICRA*, 2011, pp. 1470–1477. 7

[79] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala,

N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," in *ICLR*, 2025. 8

[80] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks. 2019 ieee," in *CVPR*, vol. 3, 2018. 9

[81] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021. 9

[82] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," 2021. 9

[83] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016. 10

[84] D. M. McDermott, "The 1998 ai planning systems competition," *AI magazine*, vol. 21, no. 2, pp. 35–35, 2000. 11

[85] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *JAIR*, vol. 14, pp. 253–302, 2001. 11

[86] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021. 12

[87] C. Garrett, A. Mandlekar, B. Wen, and D. Fox, "Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment," in *CoRL*, 2024. 12

[88] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," 2020. 12

[89] D. Raj, O. Patil, W. Gu, C. Baral, and N. Gopalan, "Learning temporally composable task segmentations with language," in *IROS*, 2024. 13

[90] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *arXiv preprint arXiv:1612.00341*, 2016. 13

[91] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *ICML*. PMLR, 2018, pp. 4470–4479. 13

[92] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," *arXiv preprint arXiv:1809.10790*, 2018. 13

[93] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions," in *RSS*, 2019. 13

[94] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *arXiv preprint arXiv:1904.12584*, 2019. 13

[95] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner, "Monet: Unsupervised scene decomposition and representation," *arXiv preprint arXiv:1901.11390*, 2019. 13

[96] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020. 13

[97] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," in *CoRL*, 2023. 13

[98] W. Yuan, C. Paxton, K. Desingh, and D. Fox, "Sornet: Spatial object-centric representations for sequential manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 148–157. 13

[99] Y. Chen, C. Wang, Y. Yang, and C. K. Liu, "Object-centric dexterous manipulation from human motion data," in *CoRL*, 2024. 13

[100] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl *et al.*, "Gemini robotics: Bringing ai into the physical world," *arXiv preprint arXiv:2503.20020*, 2025. 14

[101] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025. 14

[102] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine, "Fast: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025. 14

[103] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "Data2vec: A general framework for self-supervised learning in speech, vision and language," in *International Conference on Machine Learning*. PMLR, 2022, pp. 1298–1312. 14

[104] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607. 14

[105] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *ICCV*, 2021. 14

[106] 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. 14

[107] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, "Vision transformers need registers," in *ICLR*, 2023. 14

[108] X. Wang, J. Yang, and T. Darrell, "Segment anything without supervision," in *NeurIPS*, 2024. 14

[109] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan, "3d-llm: Injecting the 3d world into large language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 20 482–20 494, 2023. 14

[110] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," in *CVPR*, 2024, pp. 14 455–14 465. 14

[111] Y. Wang, Z. Li, M. Zhang, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li, "D^3fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation," *arXiv preprint arXiv:2309.16118*, 2023. 14

[112] Y. Tang, W. Huang, Y. Wang, C. Li, R. Yuan, R. Zhang, J. Wu, and L. Fei-Fei, "Uad: Unsupervised affordance distillation for generalization in robotic manipulation," in *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*. 14

[113] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *CoRL*, 2023. 14

[114] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson, "Rl-vlm-f: Reinforcement learning from vision language foundation model feedback," *arXiv preprint arXiv:2402.03681*, 2024. 14

[115] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh, "Reward design with language models," in *ICLR*, 2023. 14

[116] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao *et al.*, "Vision language models are in-context value learners," in *The Thirteenth ICLR*, 2024. 14

[117] X. Fang, B.-R. Huang, J. Mao, J. Shone, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Keypoint abstraction using large models for object-relative imitation learning," *arXiv preprint arXiv:2410.23254*, 2024. 14

[118] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick, "Dreamitate: Real-world visuomotor policy learning via video generation," *arXiv preprint arXiv:2406.16862*, 2024. 14

[119] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022. 14

[120] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox, "Robopoint: A vision-language model for spatial affordance prediction for robotics," in *CoRL*, 2024. 14

[121] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," in *ICML*, 2024. 14

[122] Y. Hu, F. Lin, T. Zhang, L. Yi, and Y. Gao, "Look before you leap: Unveiling the power of GPT-4v in robotic vision-language planning," *arXiv:2311.17842*, 2023. 14

[123] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, R. Yu, C. R. Garrett, F. Ramos, D. Fox, A. Li *et al.*, "Hamster: Hierarchical action models for open-world robot manipulation," *arXiv preprint arXiv:2502.05485*, 2025. 14