

A Additional Experimental Results

A.1 Detailed Results

A.1.1 Tabletop and Kitchen Benchmark

Table 2a presents the cross-embodiment capabilities of UniSkill’s universal skill representation within the tabletop benchmark. For Franka prompts, UniSkill achieves the highest performance on most tasks compared to the baselines. While GCBC and XSkill show average success rate of 60% and 61%, UniSkill maintains a minimum success rate of 75%, indicating consistently strong performance. For human prompts, GCBC and XSkill fail to complete more than half of the tasks even once. In contrast, UniSkill succeeds on most tasks and achieves an average success rate more than three times higher than the baselines. This robustness, enabled by training on large-scale video data, demonstrates the generality of our skill representation across different embodiment, which is a central goal of our framework.

Table 2b illustrates the performance on the kitchen benchmark. UniSkill outperforms GCBC when evaluated with Franka prompts, which use an embodiment seen during both skill representation and policy learning. Even with Anubis prompts, which involve an unseen robot embodiment, UniSkill still surpasses GCBC. The performance gap is even more pronounced with human prompts, where UniSkill achieves more than twice the success rate of GCBC. Notably, GCBC exhibits biased performance with unseen prompts, succeeding on only one out of three tasks for each type of different embodiment prompt. This highlights GCBC’s difficulty in handling demonstration videos from unseen embodiments.

Prompt	Task	GCBC	XSkill	UniSkill
Franka	<i>Pull out the tissue</i>	0.43	0.42	0.93
	<i>Push the blue towel</i>	0.93	0.97	0.75
	<i>Close the trash bin</i>	0.13	0.58	0.65
	<i>Open the trash bin</i>	0.63	0.80	0.87
	<i>Pick the blue towel and place it in the bowl</i>	0.62	0.28	0.85
	Average	0.60	0.61	0.81
Human	<i>Pull out the tissue</i>	0.00	0.00	0.57
	<i>Push the blue towel</i>	0.00	0.00	0.37
	<i>Close the trash bin</i>	0.45	0.00	0.25
	<i>Open the trash bin</i>	0.10	0.00	0.62
	<i>Pick the blue towel and place it in the bowl</i>	0.00	0.00	0.00
	Average	0.11	0.00	0.36

(a) Tabletop

Prompt	Task	GCBC	UniSkill
Franka	<i>Put carrot on plate</i>	0.58	0.90
	<i>Turn faucet front to left</i>	1.00	1.00
	<i>Turn faucet front to right</i>	0.70	0.93
	Average	0.76	0.94
Anubis	<i>Put carrot on plate</i>	0.00	0.00
	<i>Turn faucet front to left</i>	1.00	0.83
	<i>Turn faucet front to right</i>	0.00	0.80
	Average	0.33	0.54
Human	<i>Put carrot on plate</i>	0.00	0.67
	<i>Turn faucet front to left</i>	1.00	0.97
	<i>Turn faucet front to right</i>	0.00	0.97
	Average	0.33	0.87

(b) Bridge

Table 2: Real-world robot experiment results comparing UniSkill with baselines. Each task is evaluated using three prompts, and success rates averaged over 20 rollouts per prompt. (a) Results on the tabletop benchmark using Franka and Human prompts. (b) Results on the kitchen benchmark using Franka, Anubis (a different robot embodiment), and Human prompts.

A.1.2 Task and Environment Generalization

Table 3a presents detailed results for compositional tasks. For Franka prompts, performance decreases as task complexity increases, but UniSkill still achieves a 42% success rate even when composing four tasks. In contrast, GCBC fails even on compositions of just two tasks. For human prompts, UniSkill achieves a 33% success rate on two-task compositions, despite the prompts involving both an unseen embodiment and unseen tasks. These results highlight the compositional nature of UniSkill’s skill representation. Although the composed tasks are not seen during policy learning, the skill-conditioned policy can still predict appropriate actions from the given skill representation. This shows that even when tasks are novel, the policy can generalize across skills by executing actions aligned with the inferred motion patterns, resulting in successful behavior.

Table 3b reports per-task results for experiments in unseen environments. With human prompts, GCBC fails on most tasks, showing biased results with success only on one or two out of five

Prompt	Task	GCBC	UniSkill	Task	Scene A		Scene B	
					GCBC	UniSkill	GCBC	UniSkill
Franka	A + B	0.00	0.83	<i>Pull out the tissue</i>	0.00	0.33	0.00	0.23
	A + B + C	0.00	0.72	<i>Push the blue towel</i>	0.03	0.18	0.00	0.17
	A + B + C + D	0.00	0.42	<i>Close the trash bin</i>	0.05	0.15	0.12	0.12
				<i>Open the trash bin</i>	0.02	0.62	0.02	0.57
Human	A + B	0.00	0.33	<i>Pick the blue towel and place it in the bowl</i>	0.35	0.00	0.30	0.00
				Average	0.09	0.26	0.12	0.23

Table 3: (a) Skill compositionality evaluation on the tabletop benchmark using Franka and human prompts. A composed task is considered successful only if all sub-tasks are completed. The sub-tasks are defined as follows: **A**: *Open the trash bin*, **B**: *Pull out the tissue*, **C**: *Pick the blue towel and place it in the bowl*, **D**: *Close the trash bin*. (b) Results on unseen scenes. Evaluation uses human prompts and follows the tabletop benchmark procedure.

tasks, and zero success on the rest. In contrast, UniSkill demonstrates generalization, successfully completing most tasks. Similarly, in Figure 4, Anubis prompts are collected in unseen environments with a novel embodiment. While GCBC fails on all tasks under these conditions, UniSkill succeeds across them, as shown in Table 2b. This is further supported by the results in Table 4, where UniSkill achieves 48% success with human prompts while GCBC reaches only 9% (see Appendix A.2). These results indicate that UniSkill is robust to scene variations in the prompt videos, consistently succeeding across the tabletop, kitchen, and simulation benchmarks. They also support the conclusion that UniSkill can imitate behaviors from demonstration videos, regardless of the environment in which they were collected.

A.2 Simulation Results on LIBERO

A.2.1 Evaluation Protocol

We evaluate UniSkill on LIBERO [15], a benchmark designed for multi-task scenarios that features diverse object interactions, layouts, and tasks within a tabletop simulation environment using a Franka robot. Our evaluation encompasses 8 tasks across 2 distinct scenes in LIBERO. Detailed explanations of the tasks are provided in Appendix B.4. Each task includes 50 expert demonstrations, which we use for policy learning.

For evaluation, one demonstration per task is selected as the Franka prompt. To generate human prompts, we replicate the same tasks as those in the LIBERO benchmark. However, due to the nature of the simulation environment, it is not possible to create human prompts that exactly match the simulation settings. Instead, we align the number and positions of objects to closely resemble the LIBERO environment while ensuring a realistic human demonstrations, as shown in Figure 8. As a result, the objects presented in the human prompts are largely novel and introduce previously unseen scenarios. This implies that, while the behaviors demonstrated may align with those in the LIBERO benchmark, the semantic attributes of the objects and environment may differ.

To measure success rates, we use one demonstration per task and perform 20 rollouts per evaluation.

A.2.2 Cross-Embodiment Skill

Table 4 presents the evaluation results. In the top section, UniSkill outperforms the baselines on robot prompts across all tasks. Both GCBC and UniSkill are trained on expert demonstrations, but the result demonstrates the unique effectiveness of UniSkill’s skill representations compared to raw

Figure 8: We created a prompt video in which a human directly manipulates objects after arranging them in a real-world environment similar to the LIBERO task. Here, we visualize only 2 out of the 8 tasks here for clarity.

Prompt	Method	Scene1				Scene2				Avg
		Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	
LIBERO	GCBC	0.00	0.55	0.90	0.30	0.70	0.95	0.25	0.35	0.51
	UniSkill	0.90	0.80	1.00	1.00	1.00	1.00	0.90	0.70	0.91
Human	GCBC	0.05	0.00	0.25	0.00	0.25	0.00	0.05	0.15	0.09
	UniSkill	0.70	0.00	0.80	0.00	0.40	0.20	0.70	0.80	0.48
	GCBC-U	0.25	0.10	0.50	0.10	0.15	0.15	0.00	0.65	0.24

Table 4: Performance comparison on the LIBERO simulation benchmark. For each task, one demonstration is used with 20 rollouts, and success rates are averaged to evaluate the performance.

Figure 9: Comparison of the inference pipelines for GCBC, GCBC-U and UniSkill. All three methods use the same frame interval k . GCBC uses the I_{t+k} frames as the sub-goal and predictions the actions required to achieve that state. In contrast, GCBC-U employs ISD and FSD to predict the sub-goal based on the current observation. UniSkill is directly conditioned on the skill representation from ISD rather than relying on a pixel-level goal condition.

469 pixel inputs. This advantage is observed not only in real-world experiments but also in the simulation
470 benchmark.

471 The bottom section of Table 4 evaluates cross-embodiment performance. GCBC struggles to achieve
472 meaningful performance when transitioning from LIBERO prompts to human prompts. Espe-
473 cially, excluding tasks where both GCBC and UniSkill fail, GCBC’s maximum success rate is 25%,
474 whereas UniSkill’s minimum success rate is 30%, surpassing GCBC’s best result.

475 As shown in Figure 8, human prompts are not perfectly aligned with the simulation environment,
476 making GCBC highly sensitive to such discrepancies. Because GCBC predicts actions based on
477 sub-goal images, large visual mismatches lead to extremely poor performance. In contrast, UniSkill
478 models the demonstrator’s behavior from video prompts, allowing it to generalize across variations
479 in object semantics. This fundamental difference accounts for the significant performance gap be-
480 tween UniSkill and GCBC.

481 A.2.3 Improving GCBC with UniSkill

482 Due to the embodiment-agnostic nature of its skill representation, UniSkill enables FSD to gener-
483 ate future frames that reflect the encoded motion while preserving the original embodiment. This
484 property can help resolve the embodiment mismatch issue in GCBC, where a sub-goal image from
485 a human prompt not align with the robot’s embodiment.

486 Building on this insight, we introduce **GCBC-U**, a variation of GCBC where the sub-goal image
487 is replaced by one generated from FSD using UniSkill’s skill representation. As shown in Table 4,
488 GCBC-U significantly improves upon standard GCBC (from 9% to 24%), despite the only change
489 being the sub-goal input. This highlights that the major limitation of GCBC lies in the embodiment
490 discrepancy between the goal image and the target robot. UniSkill’s embodiment agnostic property
491 effectively resolves this issue.

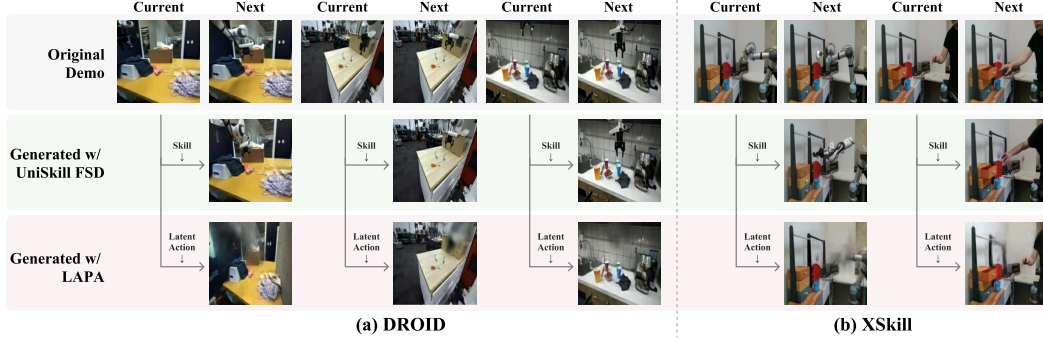


Figure 10: Comparison with Uniskill FSD and LAPA. A skill (UniSkill) or latent action (LAPA) was extracted between two frames, and the next frame was generated conditioned on the resulting vector. UniSkill FSD successfully reconstructs the video dynamics, while LAPA produces blurry images. (a): Result on DROID, (b): Result on XSkill.

The overall inference pipelines of GCBC, GCBC-U, and UniSkill are illustrated in Figure 9.

A.3 Comparison with LAPA

Both LAPA [18] and UniSkill utilize diverse video datasets, including human demonstrations, to learn latent action or skill representations. LAPA adopts Genie’s [17] transformer-based architecture trained with discrete latent actions.

In contrast, UniSkill employs an image editing [23] based pipeline to jointly train the Forward Skill Dynamics (FSD) and Inverse Skill Dynamics (ISD) models. In this framework, the edited frame serves as the next frame, while the original frame is treated as the current frame. This design encourages the ISD model to encode motion-specific features into the skill representation, rather than static features like background appearance. As a result, UniSkill effectively captures the motion between two frames, which we refer to as a skill.

Figure 10 compares future frame predictions from LAPA and UniSkill, using latent embeddings produced by their respective inverse models. We evaluate on two datasets: DROID [13], which is used for training, and XSkill [6], which is unseen during training.

In Figure 10(a), on the seen DROID dataset, LAPA generates blurry and less informative future frames, while UniSkill produces sharp and accurate predictions. In Figure 10(b), using the XSkill dataset—which is not used for training either method—only UniSkill accurately predicts the next frame, while LAPA continues to generate blurry outputs. Notably, when tested on human demonstration videos, UniSkill predicts precise future frames based on the extracted skill representation, whereas LAPA merely reproduces the input frame, failing to model motion dynamics. These results indicate that UniSkill’s skill representation effectively captures dynamic changes between frames, while LAPA fails to do so.

A.4 Additional Comparison with XSkill

A.4.1 Training XSkill on Large-Scale Datasets

We observe low success rates for XSkill on our tabletop benchmark, even when using scene-aligned datasets—i.e., human and robot videos collected in the same environment and covering the same sub-tasks. Although XSkill is typically used with scene-aligned data, it can be extended to unaligned, large-scale video datasets for skill discovery training, resembling the skill representation learning stage of UniSkill. To enable a fairer comparison, we extend XSkill’s training to include large-scale datasets and evaluate two variants that mirror UniSkill’s training setup:

- **XSkill-L** uses the same datasets as UniSkill for skill discovery. This includes robot datasets: Droid [13], Bridge [14], and LIBERO [15] as well as human datasets: Something-Something V2 [11] and H2O [12].
- **XSkill-A** builds on XSkill-L by additionally incorporating scene-aligned datasets collected in the tabletop environment.

A.4.2 Progress-Based Success Metric

In our primary evaluation, a task is marked as successful only if it is completed in full; otherwise, it is considered a failure. While this binary metric is effective for comparing compact and robust methods, it cannot distinguish between completely failed attempts and those that achieve partial progress.

To address this, we introduce intermediate evaluation points for each task to define **partial success**. Full task definitions are provided in Appendix B.3, and the partial success criteria are listed below:

- **Pull out the tissue:** Move to the left - 0.3; reach above the tissue - 0.5; grasp the tissue - 0.7; fully pull out the tissue - 1.0.
- **Close the trash bin:** Move to the right - 0.3; reach above the trash bin - 0.5; reach behind the lid - 0.7; fully close the lid - 1.0.
- **Open the trash bin:** Move to the right - 0.3; reach above the trash bin - 0.5; fully open the lid - 1.0.
- **Pick the towel and place it in the bowl:** Move downward - 0.3; touch the towel - 0.5; lift the towel - 0.7; place the towel in the bowl - 1.0.
- **Push the blue towel:** Move downward - 0.3; touch the towel - 0.5; push without covering the mark - 0.7; push and cover the mark - 1.0.

A.4.3 Effect of Dataset Scale and Alignment on XSkill

Table 5 presents comparison results on tabletop benchmark using the progress-based success metric. The results show that UniSkill significantly outperforms both XSkill variants on Franka and human prompts. UniSkill achieves near-perfect success (91%) across diverse tasks with Franka prompts and also attains the highest success rate on human prompts. In contrast, XSkill-L and XSkill-A perform poorly, despite being trained on the same or more datasets used for UniSkill’s skill representation learning. This highlights UniSkill’s scalability with large-scale training, whereas XSkill struggles to scale effectively. Notably, XSkill-L achieves only around 30% success, corresponding roughly to the first stage of the progress metric. This suggests that XSkill-L rarely completes tasks and often fails beyond the initial motion steps.

Prompt	Task	XSkill-L	XSkill-A	UniSkill
Franka	<i>Pull out the tissue</i>	0.17	0.00	0.90
	<i>Push the blue towel</i>	0.03	0.00	0.84
	<i>Close the trash bin</i>	0.61	0.19	0.91
	<i>Open the trash bin</i>	0.57	0.19	1.00
	<i>Pick the blue towel and place it in the bowl</i>	0.33	0.06	0.90
	Average	0.34	0.09	0.91
Human	<i>Pull out the tissue</i>	0.63	0.10	0.75
	<i>Push the blue towel</i>	0.20	0.20	0.37
	<i>Close the trash bin</i>	0.50	0.32	0.66
	<i>Open the trash bin</i>	0.50	0.17	0.90
	<i>Pick the blue towel and place it in the bowl</i>	0.00	0.04	0.21
	Average	0.37	0.17	0.58

Table 5: Real-world robot experiment results on tabletop benchmark using the progress-based metric. For both XSkill and UniSkill, each task is evaluated with three prompts, and success rates are averaged over five rollouts per prompt (UniSkill results are re-evaluated accordingly).

When comparing XSkill-L and XSkill-A, where the only difference is the inclusion of scene-aligned tabletop data, XSkill-L actually performs better, even though XSkill-A uses additional data. This is likely because the added tabletop dataset contains only 1K videos, which is much smaller than the large-scale training set of over 200K videos. As a result, the additional data has minimal effect on performance. Moreover, this outcome reveals that XSkill’s training becomes unstable when scaled to large and diverse datasets.

Droid	Robot	XSkill	Human	Avg				
					Depth	Augmentation	Avg	
✓				0.25				
✓				0.19				
✓	✓			0.19		✓	0.44	
✓	✓	✓	✓	0.49	✓		0.00	
✓	✓		✓	0.48	✓	✓	0.48	

(a)

		k		Prompt	
Stage 1	Stage 2	LIBERO	Human		
[1, 20]	1	0.19	0.08		
	20	0.18	0.05		
[20, 40]	20	0.91	0.45		
	40	0.79	0.34		
[40, 60]	40	0.80	0.30		
	60	0.43	0.19		

(b) (c)

Table 6: Ablation studies on the LIBERO benchmark using human video prompts. All experiments evaluate variations of UniSkill without relying on scene-aligned human-robot datasets. (a) Effect of training datasets. The last row shows our method trained without the scene-aligned dataset (XSkill), yet achieving comparable performance. **Robot:** Bridge [14] and LIBERO [15]. **Human:** Something-SomethingV2 [11] and H2O [12]. (b) Effect of training strategies. Both using augmentation and depth improve performance. (c) Effect of skill interval k . Stage 1 (skill representation learning) samples k from a range, while Stage 2 (policy learning) uses a fixed interval.

These results highlight a key limitation of XSkill’s approach, which maps skill embeddings into a shared space using a fixed set of predefined prototypes. While this mechanism allows mapping into a continuous skill representation, the limited number of prototypes restricts the model’s ability to represent the wide variety of skills found in large-scale video datasets. This limitation contributes to failures in completing full tasks and leads to instability during training. For example, even under the progress-based metric, XSkill-L completely fails one task with a human prompt.

In contrast, UniSkill emphasizes motion by focusing on the dynamic parts of a video through an image-editing pipeline. On the other hand, XSkill relies on learning objectives such as prototypes loss and time-contrastive learning, which are less effective at capturing motion patterns across video frames. By directly encoding motion, UniSkill captures features that generalize well across diverse embodiments. This allows it to demonstrate strong flexibility, even when responding to human demonstrations.

A.5 Ablation Studies

All ablation studies are conducted in LIBERO benchmark with human prompt. To conduct ablation studies, evaluation protocol is the same as simulation experiment.

Effect of Dataset for Pre-training. As shown in Table 1, we already observe the importance of scaling up dataset size with robot prompts. To further investigate the effect of pretraining datasets on cross-embodiment skill learning, we conduct ablation studies on various datasets with human prompt, with results presented in Table 6a. When large-scale human video datasets are used, performance more than doubles (from 19% to 49%), highlighting the importance of including human data. Interestingly, incorporating the XSkill dataset [6], which is scene-aligned, does not lead to meaningful improvements—likely due to its relatively small size. These findings suggest that the size and diversity of the dataset are more critical than whether it is scene-aligned.

Effect of Depth Prediction. Table 6b demonstrates the importance of incorporating depth prediction. Since our core objective is to encode dynamic information for cross-embodiment, the representation should not overly rely on semantic information, as this could lead to embodiment-specific features. As demonstrated in Figure 12(b), removing depth prediction results in a substantial drop in K-means clustering accuracy from 82.0 to 31.7, indicating reduced skill separation across embodiments. To mitigate this dependency, we incorporate depth prediction into ISD. When depth prediction is not used, overall performance decreases, highlighting its importance in ensuring an embodiment-agnostic skill representation.

Task	Scene B	
	GCBC	UniSkill [†]
<i>Pull out the tissue</i>	0.42	0.92
<i>Push the blue towel</i>	0.43	0.33
<i>Close the trash bin</i>	0.08	0.48
<i>Open the trash bin</i>	0.52	0.78
<i>Pick the blue towel and place it in the bowl</i>	0.43	0.90
Average	0.38	0.68

Table 7: Real-world robot experiments on the tabletop benchmark comparing the performance of UniSkill and XSkill using robot prompts. [†] indicates robot-only training, where only the DROID dataset is used for skill representation learning.

Effect of Augmentation. We evaluate the effectiveness of augmenting ISD inputs during policy learning. As shown in Table 6b, eliminating this augmentation leads to large drop in performance. Because UniSkill encodes the dynamics of the video prompt, it is sensitive to changes in viewpoint or object arrangements. These results indicate that our augmentation strategy effectively mitigates these challenges.

Effect of Skill Interval. To validate our choice of the skill interval k , we conduct ablation studies on the LIBERO [15] benchmark. Training consists of two stages: skill representation learning and skill-conditioned policy learning. During skill representation learning, we define a range for k and sample a value from this range at each training iteration. For policy learning, we use a fixed skill interval denoted by k . As shown in Table 6c, our default setting for k achieves the best performance across both robot and human prompts. When k is too small, it becomes difficult to extract meaningful skill information between frames, leading to degraded performance. On the other hand, if k is too large, it may exceed the feasible execution horizon of a skill, which also harms performance, as seen when the policy learning interval is set to 60.

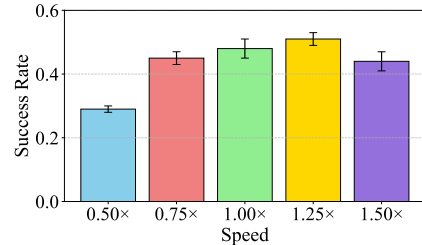


Figure 11: Ablation studies on camera speeds using human prompts on the LIBERO benchmark. Each success rate represents the average across all tasks in the benchmark.

Effect of Speed. In Figure 11, we evaluate the robustness of UniSkill by varying the video speed of the human prompts. The best performance occurs at speeds of 1.00x and 1.25x. Notably, performance decreases as the speed slows down. When the speed is too low, the encoded skill interval becomes short to capture meaningful action sequences.

A.6 Additional Analyses

A.6.1 Analysis of Skill-Conditioned Policy

In addition to the embodiment-agnostic property of our skill representation, we further evaluate its effectiveness in policy learning. To isolate this effect, we train the skill representation using only the DROID [13] dataset.

Table 7 presents the results on the Scene B environment, which features a different background, different objects, and added distractors, as introduced in Section 4.3. The strong performance in this unseen setting demonstrates the generalization capability and effectiveness of our skill representation, even when trained on a limited, robot-only dataset.

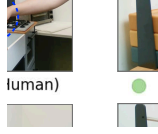


Figure 12: t-SNE visualization of UniSkill embeddings with and without depth on the XSkill dataset. Circle markers represent skill embeddings from human prompts, while cross markers represent those from robot prompts. Each color corresponds to a different task, with visual examples shown above for both human and robot executions.

A.6.2 Analysis of Cross-Embodiment Skill

We further analyze UniSkill’s ability to generalize across embodiments. As discussed in Section 4, UniSkill’s skill representations cluster by skill rather than embodiment.

To investigate this further, we visualize the t-SNE plots of skill embeddings with and without depth information. As shown in Figure 12, the embeddings learned with depth are more compact and clearly separated by skill, while the embeddings without depth are more dispersed and overlapping.

Quantitatively, using K-means clustering with $K = 3$, the depth-enabled model achieves higher clustering accuracy. This suggests that incorporating depth improves the quality of the learned skill representation and enhances its embodiment-agnostic property.

A.6.3 Analysis of Spatial Sensitivity of UniSkill

UniSkill performs tasks by imitating motion patterns from demonstration videos. As a result, the difference of positions of interacted objects between prompt video and test environment can influence task performance. To investigate this, we design an experiment varying the position of the target object.

We select the task *Push the blue towel* and modify the initial position of the towel in the evaluation environment relative to its position in the prompt video. While the towel’s position is already randomized during evaluation, we extend this variation to more extreme displacements to test the limits of spatial generalization. As shown in Figure 13, the towel’s center is shifted by 0 cm, 4 cm, 8 cm, and 12 cm from the original prompt position. The results show that as the position deviates further from the original, the success rate declines.

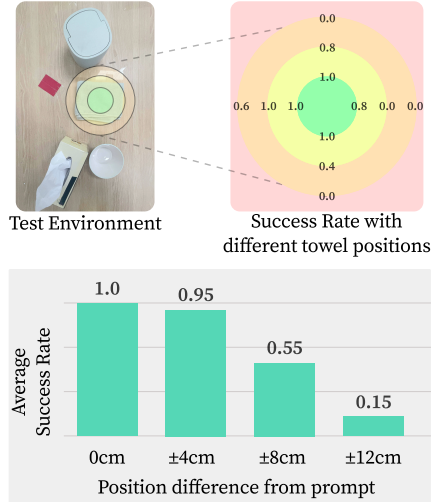


Figure 13: Visualization of the test environment and success rates across different initial towel positions. The towel’s position is shifted up to ± 12 cm from the prompt location to evaluate UniSkill’s spatial sensitivity.

This drop in performance suggests that UniSkill, which emphasizes motion patterns over semantic cues, can be sensitive to spatial changes. Nevertheless, it still demonstrates a reasonable level of robustness, successfully completing the task across a range of varied object positions.

B Experiment Details

B.1 Hardware Setup

We adopt the hardware configuration utilized in DROID [13]. Specifically, our setup comprises a Franka Research 3 robot arm paired with a 2F-85 Robotiq gripper. For the camera setting, we use two cameras: a side camera and a wrist-mounted camera. The side camera employed is the Zed 2i, while the wrist-mounted camera is Zed Mini. Both cameras capture RGB images at a resolution of 720×1280 at 15 Hz. The overall settings are depicted in Figure 14.

B.2 Implementation Detail

For pre-training, we initialize the FSD using the Instruct-Pix2Pix model [23] and train the ISD from scratch. For the visual encoder, we adopt the ResNet-18 [28], and for depth prediction, we utilize the pre-trained DepthAnythingV2 model [22] without further training. During pre-training, skill interval k is randomly selected between 1.0s and 2.0s, with the specific values determined by the frame rate of the video datasets. The image resolution is set to 256×256 . For policy learning, we employ diffusion policy [24] as the policy network. In real-world experiments, the policy network is pre-trained on the DROID dataset [13] and fine-tuned on the collected dataset. During both training and inference, the skill interval k is fixed at 20 frames, the image resolution is 128×128 , and the action dimension is set to 7. The hyperparameters are provided in Appendix C.

Figure 14: Our experiments are conducted in the DROID [13] environment.

B.3 Real-world Environments

B.3.1 tabletop Benchmark

For the tabletop benchmark, we utilize four objects: a tissue box, bowl, towel, and trash bin. The positions of the tissue box and trash bin are fixed, while the pose of the tissue varies. For the bowl and towel, we define fixed regions and randomize their locations within those areas. The towel’s pose is also varied for each trial. Additionally, we standardize the initial trajectory for each task to prevent the policy from becoming conditioned on specific initial movements.

Task definitions are as follows:

- **Pull out the tissue.** Pull out the tissue from the tissue box. *Success Criterion:* The entire tissue is removed from the tissue box.
- **Close the trash bin.** Close the lid of the trash bin. *Success Criterion:* The lid is fully closed.
- **Open the trash bin.** Open the lid of the trash bin. *Success Criterion:* The lid is clearly opened without any partial closure.
- **Pick the towel and place it in the bowl.** Pick up the blue towel and place it into the bowl. *Success Criterion:* More than half of the bowl’s area is covered by the towel.
- **Push the blue towel.** Push the blue towel to the red mark. *Success Criterion:* The red mark is entirely covered by the blue towel.

Figure 15: We designed five tasks within a real-world scene. The tasks were designed to share similar trajectories or involve the same objects across different tasks. This task setup allows for the evaluation of various actions, including push, pull, and pick-and-place.

Figure 15 shows the examples of task execution from both robot and human videos. Note that human videos are not used during training.

For the skill robustness experiments in Section 4.3, we construct two new scenes.

- **Scene A:** As shown in Figure 16(a), we change the background (table), use a completely different towel in terms of shape, size, and color, and replace the trash bin and bowl with different colors.
- **Scene B:** As shown in Figure 16(b), we introduce various distractor objects, including puppets, extra bowls, towels, and unrelated items, to increase visual complexity.

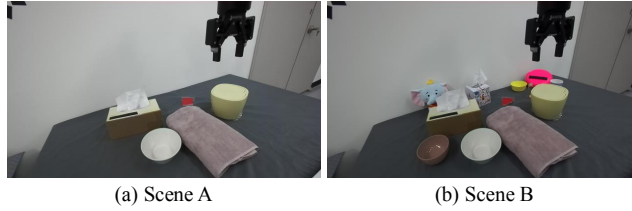


Figure 16: Unseen environments of tabletop benchmark used for skill robustness evaluation.

B.3.2 Kitchen Benchmark

For the kitchen benchmark, we employ a toy sink similar to that presented in the BridgeV2 dataset [14] and utilize three objects: faucet, carrot and plate. The data collection process mirrors that of the tabletop benchmark to maintain consistency across benchmarks.

The task definitions are as follows:

- **Turn faucet front to right.** Turn the head of faucet to the right direction. *Success Criterion:* The faucet is moved to the right relative to its original position.
- **Turn faucet front to left.** Turn the head of faucet to the left direction. *Success Criterion:* The faucet is moved to the left relative to its original position.
- **Put the carrot on the plate.** Pick up carrot and put it on the plate. *Success Criterion:* The entire carrot is placed on the plate without any part touching the sink surface.

Figure 17 demonstrates examples of task executions using Franka, Anubis, and human embodiments. Anubis is an unseen robot embodiment not included in skill representation learning, and its demonstrations are collected in an unseen environment with a completely different background.

Figure 17: Prompt videos with different embodiments. To evaluate cross-embodiment imitation using UniSkill, we record prompt videos with 3 different embodiments. Prompt videos are recorded using a Anubis Robot (Top row), Franka arm (middle row) and a human hand (bottom row).

Anubis Prompt. Anubis is a custom-built robot inspired by Mobile ALOHA system [26]. It is a mobile, bimanual robot with two 6-DoF arms, each equipped with a wrist-cam-mounted parallel gripper, and a 3-wheel omni chassis. As a non-commercial, custom-designed platform, Anubis does not appear in any existing robot datasets. Therefore, it serves as a fully unseen embodiment in our evaluation.

B.4 Simulation Environments

For the LIBERO benchmark, we conduct experiments on four tasks within each of two distinct scenes, resulting in a total of eight tasks. These tasks are predefined within the LIBERO simulation environment, and their success is automatically determined by the simulation system.

The task definitions are as follows:

- **Task1.** put the red mug on the left plate.
- **Task2.** put the red mug on the right plate.
- **Task3.** put the white mug on the left plate.
- **Task4.** put the yellow and white mug on the right plate.
- **Task5.** put the chocolate pudding to the left of the plate.
- **Task6.** put the chocolate pudding to the right of the plate.
- **Task7.** put the red mug on the plate.
- **Task8.** put the white mug on the plate.

C Implementation Details

C.1 Skill Dynamics Modeling

For skill dynamic modeling, UniSkill jointly trains the Inver Skill Dynamics (ISD) model and the Forward Skill Dynamics (FSD) model. The hyperparameters used for training are listed in Table 8a.

C.1.1 Inverse Skill Dynamics Model

For the Inverse Skill Dynamics model, we employ ResNet-18 [28] as the visual encoder and utilize the pre-trained DepthAnythingV2-small model [22] as the monocular depth estimator. During pre-training, the depth estimator remains fixed, and only the visual encoder is trained to encode visual features.

Hyperparameter	Value	Hyperparameter	Value
Batch Size	1024	Batch Size	128 (DROID, tabletop, Bridge) 256 (LIBERO)
Training Epoch	50	Training Steps	50000 (DROID) 25000 (tabletop) 5000 (Bridge) 200000 (LIBERO)
Learning Rate	$1e - 4$	Learning Rate	$1e - 4$
k	[20, 40]	k	20
skill dim	256	Optimizer	Adam
Optimizer	AdamW	Betas	(0.9, 0.999)
Betas	(0.9, 0.999)	Weight Decay	0.01
Weight Decay	0.01	Image Resolution	(128, 128)
Image Resolution	(256, 256)	Crop Size	(116, 116)
		Diffusion Model	DDIM
		Denoising Step	20
		Observation Horizon	2
		Prediction Horizon	16
		Action Horizon	8

(a)
(b)

Table 8: Hyperparameters used in UniSkill: (a) FSD/ISD during pre-training and (b) policy learning.

To effectively capture spatial and temporal dependencies between frames I_t and I_{t+k} , we integrate ST-Transformer blocks [29]. Each ST-Transformer block comprises a spatial attention layer, a causal temporal attention layer, and a MLP layer. The ISD model incorporates a total of eight ST-Transformer blocks, enabling robust encoding of dynamic interactions between the sampled frame pairs. Prior to processing with the ST-Transformer blocks, the predicted depth maps are projected into depth features, which are then concatenated channel-wise with the visual features by the visual encoder for each timestep t and $t + k$.

C.1.2 Forward Skill Dynamics Model

The Forward Skill Dynamics model adopts the architecture of InstructPix2Pix [23], with a key modification: FSD is conditioned on the universal skill representation \mathbf{z}_t instead of language instructions. Consequently, while InstructPix2Pix freezes the text encoder and does not propagate gradients to it, we replace the text encoder with ISD and condition FSD on \mathbf{z}_t . Additionally, the ISD receives gradient updates from FSD during training. This adjustment ensures that FSD generates future frames based on the encoded motion patterns in \mathbf{z}_t , facilitating effective cross-embodiment imitation.

C.2 Universal Skill-Conditioned Policy

We employed a diffusion policy[24] as our policy architecture and utilized a codebase based on Robomimic [30] and DROID [13]. In the training process, we first resize all image observations to 128×128 and use a resnet [28] visual encoder to extract visual features. These visual features are then concatenated with other observations to form a single vector. This observation vector is passed through an MLP to obtain a global condition. Typically, this global condition is fed into the Unet diffusion head to generate an action trajectory. However, in the case of our universal skill-conditioned policy, the global condition vector is concatenated with a universal skill representation before being processed by the diffusion Unet. We use an observation horizon of 2 and generated an action trajectory spanning 16 timesteps. During inference, the action prediction length is set to 8, meaning that 8 steps of actions are executed in an open-loop manner. For the diffusion model, we employ DDIM [31] with 20 denoising steps for action prediction. The hyperparameters used for policy learning are reported in Table 8b.

C.2.1 Real-World

In real-world experiments, we use images from the ZED 2i camera and ZED Mini as image observations, along with the 3D Cartesian position of the gripper and the gripper state as proprioception. ImageNet pretrained ResNet-50 [28] is used to encode the image observations. During training, skills are extracted from the left ZED 2i camera images of expert trajectories at intervals of 20 timesteps. These skill representations are concatenated with the global condition for action denoising. During inference, skills are first extracted from the prompt video at intervals of 20 timesteps using ISD. These pre-extracted skills are then utilized as conditions corresponding to the current timestep for action denoising. Specifically, the skill relevant to the current timestep of prompt video is concatenated with the global condition, and the diffusion process is performed to predict the action trajectory.

C.2.2 Simulation

In the LIBERO [15] simulation setup, we use the robot’s agent view and wrist view as image observations, with ResNet-18 [28] as the visual encoder. For low-dimensional observations, we include the end-effector’s orientation, position, gripper states, and joint states. Similar to the real-world experiment, we extract skills only from the agent view and not from the wrist view during training. These extracted skills are then concatenated with the global condition to predict actions. During inference, we extract skills from the prompt video with skill interval 20 and use the skill corresponding to the current timestep as a condition for action prediction.

C.3 Goal-conditioned Behavioral Cloning

The policy architecture of a goal-conditioned behavioral cloning policy (GCBC) employs the same diffusion policy as the Universal skill-conditioned policy, with all components being identical except for the conditioning. In the Universal skill-conditioned policy, the global condition is concatenated with the Universal skill representation for denoising, whereas in GCBC, the global condition is concatenated with the goal image feature for denoising. The goal image feature is obtained by passing the corresponding view image through the same visual encoder used for encoding observations.

During training, the goal image is sampled from the expert dataset using hindsight relabeling and is concatenated with the global condition for action prediction. During inference, to maintain consistency with the Universal skill-conditioned policy setup, the image from 20 timesteps ahead in the prompt video is used as the sub-goal image for the current timestep.

C.3.1 Real-World

The real-world setup for GCBC is largely consistent with that of the Universal Skill-Conditioned Policy. Similar to the UniSkill setup, the image from wrist camera is not used as the goal image, and the image from the ZED 2i camera is utilized. During inference, the image from 20 timesteps ahead in the prompt video is used as the goal image for action denoising.

C.3.2 Simulation

The LIBERO [15] simulation setup is also identical to that of the Universal Skill-Conditioned Policy. The LIBERO simulation agent-view is used as the goal image, hence the goal image feature is obtained by passing the goal image through the visual encoder for agent-view observations.

C.4 XSkill

XSkill [6] proposes a cross-embodiment skill representation using a feature clustering approach. While it does not require strictly paired datasets with identical motions between humans and robot demonstrations, it still imposes certain constraints. Specifically, XSkill requires human demonstration videos for training, and those videos must be recorded in the same environment and cover the same tasks as the target robot setup.

828 To meet these requirements, we additionally col-
 829 lect 100 human demonstrations per task in the same
 830 scene as the target evaluation setup.

831 Since the official XSkill codebase¹ does not include
 832 complete inference code or training configurations
 833 for real-world, we re-implement the method for real-
 834 world experiments, following the paper and avail-
 835 able codebase as closely as possible. For the re-
 836 ported results, we train XSkill three times and report
 837 the best performance among the runs. The training
 838 configurations used for XSkill are reported in Ta-
 839 ble 9a and Table 9b.

840 D Failure Cases

841 We analyze failure cases in the real-world tabletop
 842 benchmark. Figure 18 displays both successful and
 843 unsuccessful rollouts derived from the same human
 844 video prompts. A common failure mode is the in-
 845 ability to make proper contact with the target objects.
 846 Although our skill representation encodes motion
 847 patterns and the robot faithfully follows the demon-
 848 strated trajectory, it does not adapt when object in-
 849 teraction fails.

850 In failure case (a), for example, the gripper slightly
 851 retracts toward the tissue, attempts to grasp it, but
 852 only opens without securing the object. In failure
 853 case (b), the gripper descends to grasp the towel but
 854 misses, resulting in a failure to secure the towel even
 855 though the robot proceeds to push toward the desired
 856 location.

857 These observations suggest that while UniSkill effectively replicates the demonstrated trajectory,
 858 further enhancements in object interaction may yield additional performance gains.

Hyperparameter	Value
Video Clip Length l	8
Sample Frames T	100
Sinkhorn Iterations	3
Sinkhorn Epsilon	0.03
Prototype Loss Coef	0.5
Prototype Loss Temperature	0.1
TCN Loss Coef	1
TCN Positive Window w_p	16
TCN Negative Window w_p	16
TCN Positive Samples	1
TCN Temperature τ_{tcn}	0.1
Batch Size	20
Training Iteration	500
Learning Rate	$1e - 4$
Optimizer	ADAM

(a)

Hyperparameter	Value
Observation Horizon	2
Observation Dimension	7
Action Dimension	7
Batch Size	128
Training Iteration	150
Learning Rate	$1e - 4$
Optimizer	ADAM

(b)

Table 9: Hyperparameters used for XSkill: (a) Skill Discovery and (b) Skill Transfer Composing.

¹<https://github.com/real-stanford/xskill>

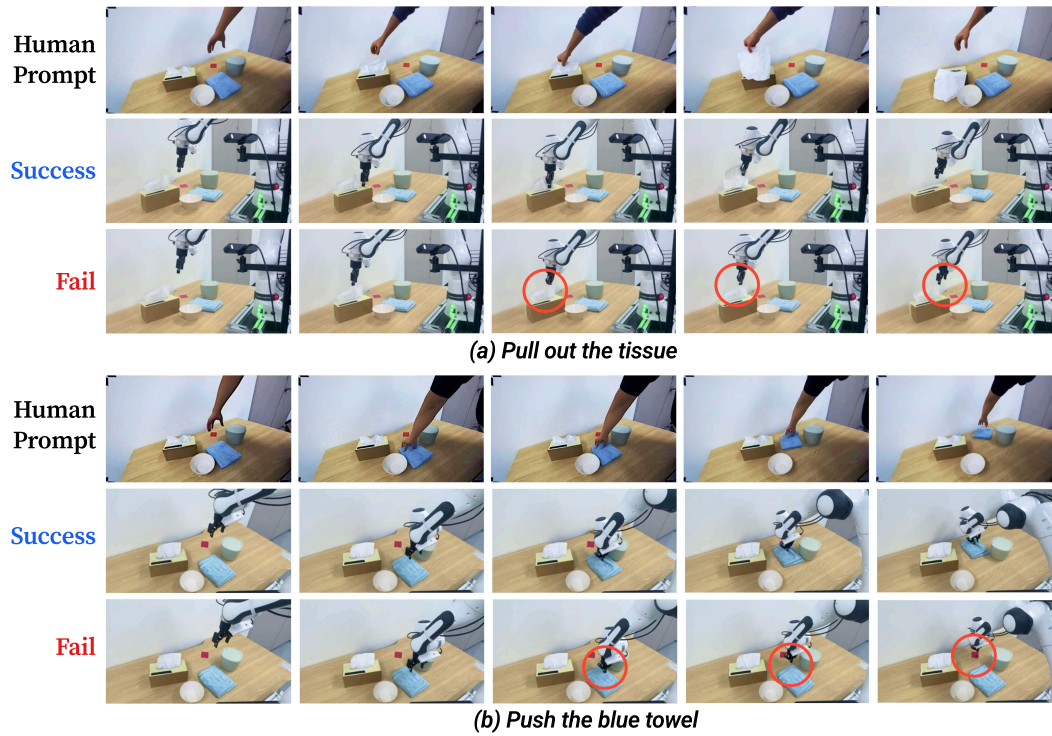


Figure 18: Analysis of failure cases for UniSkill on the tabletop tasks *Pull out the tissue* and *Push the blue towel*. In these cases, the primary failure mode is inaccurate contact with the target object.