Sketch RL: Interactive Sketch Generation for Long-Horizon Tasks via Vision-based Skill Predictor

APPENDIX

A. Ablation on diversity of primitive skills

To provide a more comprehensive understanding of the VSP's performance, we perform an ablation study on the diversity of primitive skills. We compare variants of our VSP module using 3, 10, or 20 primitive skills to see whether the complexity of the skill library plays an important role on the algorithm's performance. The test data spans four environments including three seen tasks NutAssembly (N), Stack (S), PickPlaceMilk (M), and one unseen task StackThree (ST). This ablation examines the robustness of our method under skill libraries with different complexities, which shows the VSP's scalability to large-scale libraries in real-world scenarios. We outline all results in Table I.

TABLE I

ABLATION ON DIVERSITY OF PRIMITIVE SKILLS

Num of skills		3	10	20
seen	N	100±0.0%	100±0.0%	100±0.0%
	S	100±0.0%	99.9±0.1%	100±0.0%
	M	99.9±0.1%	100±0.0%	100±0.0%
unseen	ST	99.5±0.2%	98.8±0.2%	97.2±0.3%

B. Albation on the composition of primitive skills

We conduct additional ablation experiments to assess the VSP's performance in situations where visually similar skills exhibit different dynamics. We compare the performance of VSP in the {*Grasp, Reach, Open*} and {*Push, Grasp, Reach*} task respectively. Note that the key frames of *Push* and *Grasp* skill share similar visual features. In our work we choose to sample the key observations around bottleneck states of a task, aiming to emphasize the differences in observations. To assess the role of keyframes in VSP, we also consider using the full trajectories for the pre-training VSP module. All results are presented in Table II. We make the following observations for the VSP module:

- Sampling the key observations is important. The full trajectories include more visually similar frames indicating different skills, especially at the junction of two skills. The performance significantly degrades as a result.
- When the number of skills becomes large, it is inevitable to encounter visually similar skills exhibiting different dynamics. We see a slight drop in performance of VSP under this situation.

TABLE II

ABLATION ON THE COMPOSITION OF PRIMITIVE SKILLS

Composition of skills	Sample key frames	Full trajectory
Grasp, Reach, Open	100±0.0%	87.2±0.9%
Push, Grasp, Reach	96.6±0.7%	92.6±0.8%

C. Datasets

1) Key Frames: To train the VSP module, we collect 100 demonstrations for each manipulation task in the robosuite environment with scripted policies. Each demonstration for a task consists of 30-50 sampled key frames around bottleneck states which represent subgoals of the task. For instance, the bottleneck states in pick-and-place tasks specifically refer to the initial state of an episode, pick the object successfully, and hover right above the target position. A detailed description for key frames of each primitive skill is shown below:

- Grasp: the initial 10 steps of an episode;
- Reach: the steps that the gripper is closing;
- Open: the end-effector is approaching the target position and the distance from the target position is within a certain threshold.

2) Scripted Policies: We adopt scripted policies π_{script} for the key frame collection. A scripted policy refers to a predetermined set of instructions or rules that dictate how a robot should behave in a given situation. The scripted policy here concretely simplifies a pick-and-place task by moving the end-effector to the position of the object to be grasped, closing the gripper, lifting the object, and then reaching the target position, finally opening the gripper.

D. Overview of SAC

SAC considers an infinite-horizon Markov decision process (MDP), specified by the tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are continuous, and the unknown state transition probability $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$ represents the probability density of the next state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$. The environment emits a bounded reward r on each transition.

SAC uses function approximators for both the Q-function and the policy, and instead of running evaluation and improvement to convergence, alternates between optimizing both networks with stochastic gradient descent. Considering a parameterized Q function $Q_\psi(s, a)$ and a tractable policy $\pi_\theta(a|s)$. For example, the value functions can be modeled as expressive neural networks, and the policy as a Gaussian with mean and covariance given by neural networks. The Q-function parameters can be trained to minimize the soft

Bellman residual:

$$J_Q(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{B}}\left[\left(Q_\psi(s,a) - r(s,a) - \gamma\mathbb{E}_{s'\sim p}[V_{\tilde{\psi}}(s')]\right)^2\right] \tag{1}$$

with

$$V_{\tilde{\psi}}(s) = \mathbb{E}_{\pi_\theta}\left[Q_{\tilde{\psi}}(s,a) - \alpha\log\pi_\theta(a|s)\right] \tag{2}$$

where α is the regularization parameter for controlling the max entropy objective, $\mathcal{B}$ is the distribution of previously sampled states and actions, or a replay buffer and $\tilde{\psi}$ can be an exponentially moving average of the action value network weight.

Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence:

$$J_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{B}}\left[\mathbb{E}_{a\sim\pi_\theta}\left[\alpha\log\pi_\theta(a|s) - Q_\psi(s,a)\right]\right] \tag{3}$$