

425 Appendix

426 **Loss Function** We use the PPO algorithm to optimize CSG. The policy objective function is given by

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (2)$$

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \quad (3)$$

427 where $r_t(\theta)$ is the probability ratio between the new policy and the old policy; \hat{A}_t is the advantage function
 428 estimate at timestep t ; $\text{clip}(\cdot)$ is a function that clips $r_t(\theta)$ within the range $[1 - \epsilon, 1 + \epsilon]$; ϵ is a small
 429 hyper-parameter that controls the clipping range.

430 The value function error and the entropy bonus are given by

$$L^{\text{VF}}(\theta) = \mathbb{E}_t \left[\left(V_\theta(s_t) - \hat{R}_t \right)^2 \right] \quad (4)$$

$$L^{\text{H}}(\theta) = \mathbb{E}_t [\mathcal{H}[\pi_\theta](s_t)] \quad (5)$$

431 where $V_\theta(s_t)$ is the value function parameterized by θ ; \hat{R}_t is the cumulative return at timestep t ; $\mathcal{H}[\pi_\theta](s_t)$
 432 is the entropy of the policy at state s_t .

433 The combined loss function is given by

$$L(\theta) = c_1 L^{\text{VF}}(\theta) - c_2 L^{\text{H}}(\theta) - L^{\text{CLIP}}(\theta) \quad (6)$$

434 where c_1 and c_2 are the coefficients that balance the importance of the value function error and the entropy
 435 bonus, respectively.

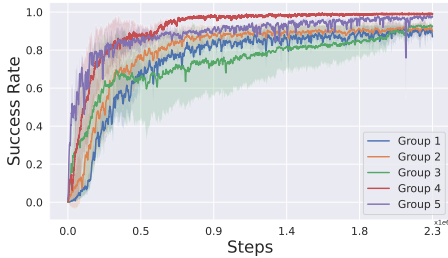


Fig. 9: **Learning curves of ClutterGen.** The x-axis shows the number of steps, and the y-axis shows the success rate. The solid curve represents the average success rate from three different initial random seeds for each group, with the semi-transparent area indicating the standard deviation.

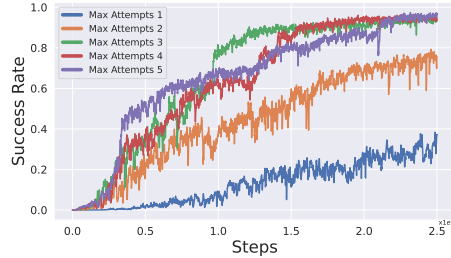


Fig. 10: **Maximum allowed attempts selection.** The performance will drop if the maximum allowed attempts ≤ 2 . We choose 5 attempts for the ClutterGen.

436 **Extra Explanation of the Stable Conditions** It is difficult to precisely define object stable conditions
 437 with single thresholds for the velocity and acceleration in the simulation. On one hand, the simulator is not
 438 perfect, meaning a visually stable pose usually has non-zero velocity and acceleration. Too small thresholds
 439 for the velocity and acceleration usually introduce false negatives (*i.e.*, the queried object that is already
 440 stable by human judgment but can not pass the stable conditions). On the other hand, too large thresholds
 441 usually introduce false positives. In this work, we applied small thresholds for both values, formulating a
 442 relatively strict checking condition.

443 **Implementation Details of the ClutterGen** The PointNet++ encoder was pre-trained on modelnet40
 444 dataset [52] and the weights were frozen during the training of ClutterGen. The attempt history was
 445 flattened and then padded by 0 if the current attempts were fewer than the maximum allowed attempts to

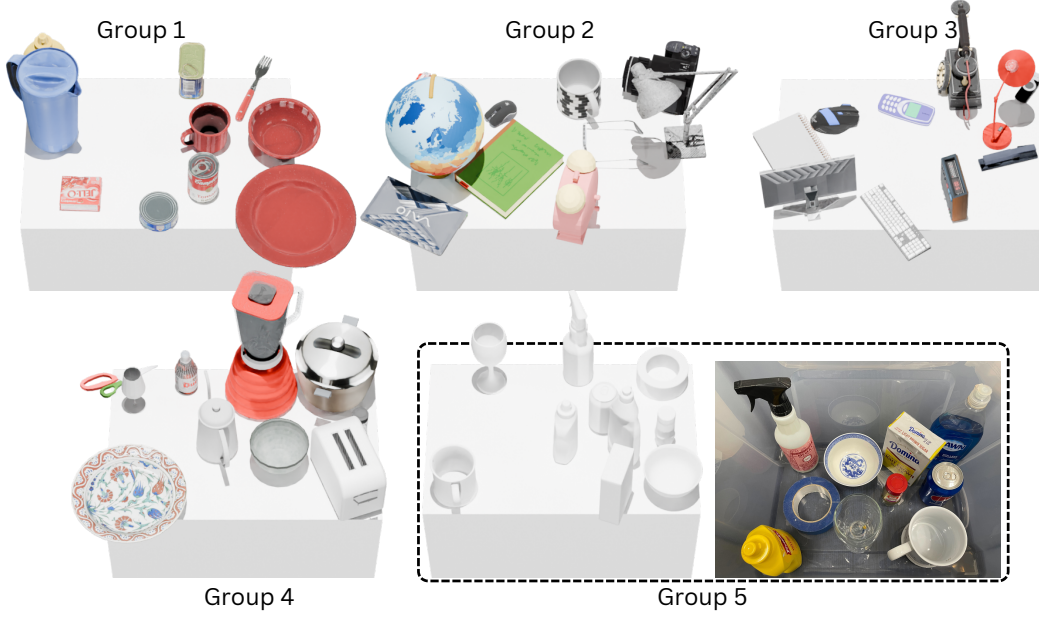


Fig. 11: **5 groups objects dataset**. Each group contains 10 objects. The group 1 to 4 are selected from the existing object dataset. Group 5 is 3D-scanned everyday objects that are also used in our physical experiments.



Fig. 12: **10 test tables dataset**. We select 10 different tables from the existing object dataset for the scene-level randomization test.

446 guarantee fixed-size input before being sent to the history sequence encoder. All the training and test was
 447 conducted in the PyBullet simulator [51].

448 **ClutterGen Training Results** We trained ClutterGen on different group objects. The training results
 449 are shown in Fig.9. ClutterGen could reach above 0.85 success rate after 2.3M steps optimization for all
 450 groups, which shows the high adaptation of different objects.

451 **Selection of Maximum Allowed Attempts.** We test the performance of ClutterGen with different maxi-
 452 mum attempts on group 1 objects. Increasing allowed attempts will boost the performance. However, too
 453 many attempts will slow down the training process and focus on only place one object. We set 5 attempts
 454 at most to balance the performance and training speed.

455 **Implementation Details of Stable Placement Policy** We used two pre-trained PointNet++ models as the
 456 perception encoders for the queried scene point cloud and the queried object point cloud respectively. We
 457 did not freeze their weights during the training in this task. The perception features were concatenated and
 458 sent to a 4-layer MLP to generate the predicted placement pose into the scene surface center frame. This
 459 pose was then transformed into the robot base frame for motion planning.

Tab. 4: PPO Hyperparameters

Hyperparameter	Value	Description
Learning Rate	0.0001	The step size at each iteration while moving toward a minimum of a loss function.
Batch Size	1000	Number of steps per gradient update.
Update Epochs	5	Number of epochs to update the policy.
γ	0.99	The discount factor for reward.
λ_{gae}	0.95	The discount for the general advantage estimation.
ϵ	0.2	The surrogate clipping coefficient.
c_1	0.5	The coefficient of the value function.
c_2	0.01	The coefficient of the entropy.
$g_{clip.max}$	0.5	The maximum norm for the gradient clipping.
Hidden Layer Size	256	The number of units of hidden layer in MLP.
Optimizer	Adam	Optimization algorithm for updating weights.

Tab. 5: ClutterGen Hyperparameters

Hyperparameter	Value	Description
P_{obj}	1024	Number of points of queried object point cloud.
P_{sce}	20480	Number of points of queried scene point cloud.
Max Attempts	5	Number of allowed attempts of placing the queried object.
c	0.005	The coefficient of the velocity and acceleration penalty.
R_0	100	The scalar reward.