

Figure 6: ThreadInHole

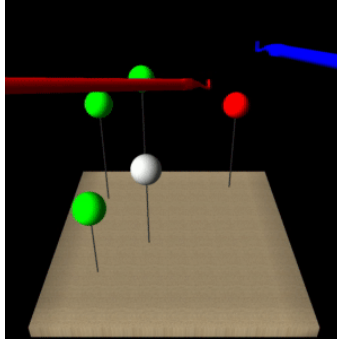


Figure 7: DeflectSpheres

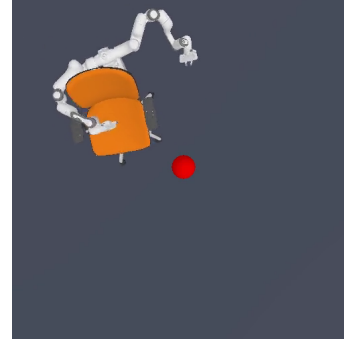


Figure 8: PushChair



Figure 9: OpenCabinetDrawer



Figure 10: OpenCabinetDoor



Figure 11: TurnFaucet

Figure 12: Image renderings of the 6 environments used. The ManiSkill12 environments are rendered from a dedicated, fixed rendering camera that the agent does not have access to.

**Task Descriptions:** All 6 environments are rendered in Figure Figure 12. In ThreadInHole, the goal is to feed the deformable thread into the (rigid) hole. In DeflectSpheres, the goal is to deflect the red or blue sphere with the tool of the corresponding color. Spheres turn green after they are deflected. In PushChair, the goal is to push the chair to the position marked by the red ball (not visible to the agent), without knocking it over. In OpenCabinetDrawer (OpenCabinetDoor), the goal is to grab the drawer (door) handle and open the drawer (door), then stabilize it in the open position. In TurnFaucet, the goal is to turn a faucet to the open position.

**sofaenv:** All sofaenv environments have point cloud observations from only a single camera, without a low-dimensional state. Upon each reset, the camera position and look-at are perturbed by a vector uniformly sampled from  $[-2\text{cm}, 2\text{cm}]$  in all directions. Reward weights are taken from the reference implementation.

**ManiSkill12:** All mobile manipulation tasks (OpenCabinetDrawer, OpenCabinetDoor, and PushChair) have 3 cameras mounted above the robot base facing outwards, spaced radially at  $120^\circ$ . In these tasks, the camera FOV was increased to 1.5, such that the cabinet/chair is almost always visible from at least one perspective. In OpenCabinetDoor, only the first 10 cabinet models are used for training and evaluation. In addition, the target door is not randomized but fixed for each cabinet model. OpenCabinetDrawer, because it is slightly easier, is not changed, using all 25 models and a randomly chosen target drawer for each episode. TurnFaucet is a static manipulation task with only 2 cameras, one mounted on the (static) base and one inside the gripper. In TurnFaucet, only the first 10 faucet models are used for training and evaluation. In all environments, joint position

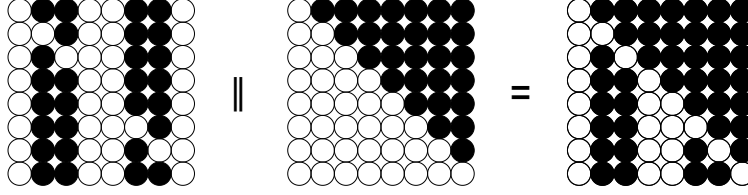


Figure 13: An example of an attention mask for a single token sequence. Black circles are masked (not visible), and white circles are unmasked. The  $i$ th row shows the tokens that are visible when encoding the  $i$ th token. The  $j$ th column shows which tokens may attend to the  $j$ th token. The first column is the start-of-sequence token, not the token of the first point patch. A token is masked if it is either randomly selected with a probability of  $m \in [0, 1]$  (random masking) or is in the present or future (causal masking). A token is always allowed to attend the token immediately preceding it. Additionally, a certain fraction of the tokens at the start of each sequence are never masked out.

473 delta control is used for the robot arm, and joint velocity control is used for the robot base, if ap-  
 474 plicable. Observations contain point clouds and a low-dimensional state vector containing robot  
 475 proprioception and the target location.

476 **Discounts and Time Limits:** Discounts for all tasks are: 0.99 for ThreadInHole and PushChair,  
 477 0.9 for DeflectSpheres, and 0.85 for OpenCabinetDoor, OpenCabinetDrawer, and TurnFaucet.  
 478 The time limits for all tasks are: 300 for ThreadInHole, 500 for DeflectSpheres, and 200 for all  
 479 ManiSkill12 tasks. In OpenCabinetDrawer, OpenCabinetDoor, and TurnFaucet, trajectories are  
 480 ended only on a timeout, not task success. In ThreadInHole, DeflectSpheres, and PushChair, trajec-  
 481 tories are ended either on a timeout or task success. PushChair is the only ManiSkill2 environment  
 482 treated like this, because we found that ending on task success is crucial for training.

483 **Point Clouds:** All environments are rendered with a resolution of 128x128, except PushChair,  
 484 which has a resolution of 64x64. Egocentric point clouds are created from depth images using  
 485 the linear transformations given by the camera intrinsic (focal length) and extrinsic (position and  
 486 orientation) parameters. Point clouds are then post-processed using the following steps: cropping,  
 487 appending target points, downsampling, and normalization (note that not all environments use all of  
 488 these steps). First, points beyond the boundaries of the scene are discarded. In ManiSkill2 environ-  
 489 ments, the floor is also cropped out, and furthermore in OpenCabinetDrawer and OpenCabinetDoor,  
 490 any points more than 0.5m in front of the cabinet are also cropped out. In OpenCabinetDrawer,  
 491 OpenCabinetDoor, and PushChair, 50 additional points are appended to the point cloud, sampled  
 492 from a uniform cube (of side length 7cm for OpenCabinetDrawer and OpenCabinetDoor and 14cm  
 493 for PushChair). Although this information is also present in the state vector observation, these points  
 494 serve to indicate the visual target position within the observation directly. In ThreadInHole, Deflect-  
 495 Spheres, and PushChair, we use voxel grid sampling (with a voxel size of 5mm for sofaenv and  
 496 5cm for PushChair) to downsample the point clouds, followed by random downsampling (without  
 497 replacement) to a maximum point cloud size of 1000 points. In OpenCabinetDrawer, OpenCab-  
 498 inetDoor, and TurnFaucet, we randomly downsample each point cloud (without replacement) to  
 499 a maximum point cloud size of 800 points. For DeflectSpheres, we normalize all observed point  
 500 clouds by subtracting a fixed center point and dividing by a fixed scale factor. The scale and center  
 501 point are chosen such that the extrema of a point cloud from a typical initial observation are roughly  
 502 within  $[-1, 1]$ . For ThreadInHole, we normalize each observed point cloud independently by sub-  
 503 tracting the mean and then dividing by the maximum absolute value over all 3 coordinates, such that  
 504 the longest axis is mapped to  $[-1, 1]$ . This method of normalization (compared to that of Deflect-  
 505 Spheres) was crucial for training. Point clouds for ManiSkill2 environments are not normalized as  
 506 they already lie roughly within  $[-1, 1]$ .

## 507 B Hyperparameters and Architectures

508 **PPRL** We use hyperparameters for the tokenizer, masking, encoder, decoder, and prediction head  
509 as defined in PointGPT. We reduce the number of transformer layers to 3 to speed up inference. For  
510 environments with color, the point feature dimension is set to 3 in the tokenizer and the prediction  
511 head; otherwise it is 0. We disable dropout.

512 **Actor and Critic Networks** Actor and critic networks are fully-connected networks with 3 layers  
513 and 1024 neurons per layer, using ELU as the non-linearity.

514 **Masking** Figure 13 provides an overview of the attention mask used for the auxiliary reconstruc-  
515 tion loss.

516 **SAC** We use SAC with a replay buffer of 500000. We use a replay ratio of 64, except for sofaenv  
517 environments with a replay ratio of 32, and PushChair where we use 8. We update target networks  
518 after every learning update with a  $\tau$  of 0.005, except PushChair, which uses a  $\tau$  of 0.01. By default,  
519 the batch size is 1024 when training without auxiliary loss and 256 when training with auxiliary loss,  
520 except for PushChair, where we use 128 in both cases. Due to compute constraints, we reduce the  
521 batch size for PointNet++ to 256, and for PointTransformer to 512. Also due to compute constraints,  
522 we reduce the replay ratio for PointNet++ on OpenCabinetDrawer and OpenCabinetDoor to 32, as  
523 these environments have the largest point clouds on average. The learning rate is 1e-4, except for  
524 DeflectSpheres and PushChair with 5e-5, and we use the Adam optimizer. The starting entropy  
525 coefficient is 0.1 except for DeflectSpheres with 0.2 and PushChair for 0.05. The learning rate for  
526 the entropy coefficient is 1e-4, except for PushChair with 2e-5.

527 For all baselines, we use hyperparameters as defined by the baselines, except for the environment-  
528 dependent discount, initial entropy coefficient, and entropy learning rate, where we use the same  
529 values as for our method.

530 **DrQ-v2** A single CNN encoder is shared among all cameras, and the features are concatenated  
531 together, resulting in an image encoding of size 117600. Both the image embedding and the state  
532 are projected down to a dimensionality of 50 and then concatenated.

533 **Dreamer and Multi-View Masked World Models** For these world model-based methods, we  
534 follow the approach of [41, 4] to handle the additional state information. They first encode both  
535 image and state using separate encoders and concatenate their outputs before providing them to the  
536 RSSM. For training, they then consider two separate output heads to reconstruct both the image  
537 and state. Dreamer uses mostly the hyperparameters proposed by [42]. However, we assume a  
538 Gaussian latent variable and use the same discount and entropy control as PPRL. For Multi-View  
539 Masked World Models, we run training in multi-view mode, which masks random viewpoints and  
540 reconstructs them. We disable the behavioural cloning loss, as our method does not have access to  
541 demonstrations.