References

[1] C. Pan, B. Okorn, H. Zhang, B. Eisner, and D. Held. Tax-pose: Task-specific cross-pose estimation for robot manipulation. In *Conference on Robot Learning*, pages 1783–1792. PMLR, 2023.

[2] 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)*, pages 6394–6400. IEEE, 2022.

[3] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018.

[4] A. Simeonov, A. Goyal, L. Manuelli, Y.-C. Lin, A. Sarmiento, A. R. Garcia, P. Agrawal, and D. Fox. Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement. In *Conference on Robot Learning*, pages 2030–2069. PMLR, 2023.

[5] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. *arXiv preprint arXiv:2310.16050*, 2023.

[6] 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.

[7] R. Antonova, P. Shi, H. Yin, Z. Weng, and D. K. Jensfelt. Dynamic environments with deformable objects. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[8] B. Eisner, Y. Yang, T. Davchev, M. Vecerik, J. Scholz, and D. Held. Deep se(3)-equivariant geometric reasoning for precise placement tasks, 2024.

[9] Y.-C. Lin, P. Florence, A. Zeng, J. T. Barron, Y. Du, W.-C. Ma, A. Simeonov, A. R. Garcia, and P. Isola. Mira: Mental imagery for robotic affordances. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1916–1927. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/lin23c.html.

[10] J. Seo, N. P. S. Prakash, X. Zhang, C. Wang, J. Choi, M. Tomizuka, and R. Horowitz. Contact-rich se(3)-equivariant robot manipulation task learning via geometric impedance control. *IEEE Robotics and Automation Letters*, 9(2):1508–1515, Feb. 2024. ISSN 2377-3774. doi:10.1109/lra.2023.3346748. URL http://dx.doi.org/10.1109/LRA.2023.3346748.

[11] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, A. Wahid, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation, 2022.

[12] A. Simeonov, Y. Du, L. Yen-Chen, A. Rodriguez, L. P. Kaelbling, T. Lozano-Perez, and P. Agrawal. Se(3)-equivariant relational rearrangement with neural descriptor fields, 2022.

[13] J. Wang, O. Donca, and D. Held. Learning distributional demonstration spaces for task-specific cross-pose estimation. *IEEE International Conference on Robotics and Automation (ICRA), 2024*, 2024.

[14] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[15] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[16] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[17] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021.

[18] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[19] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[20] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[21] L. Zhou, Y. Du, and J. Wu. 3d shape generation and completion through point-voxel diffusion, 2021.

[22] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis. Lion: Latent point diffusion models for 3d shape generation. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[23] S. Luo and W. Hu. Diffusion probabilistic models for 3d point cloud generation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[24] G. Nakayama, M. Uy, J. Huang, S. Hu, K. Li, and L. Guibas. Difffacto: Controllable part-based 3d point cloud generation with cross diffusion. *International Conference on Computer Vision (ICCV)*, 2023.

[25] W. Peebles and S. Xie. Scalable diffusion models with transformers. *International Conference on Computer Vision (ICCV)*, 2023.

[26] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. CoTracker: It is better to track together. 2023.

[27] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[28] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics, and machine learning, 2016-2020. URL `http://pybullet.org`.

Appendix

Table of Contents

A DEDO Environment Details

DEDO: Dynamic Environments with Deformable Object [7] is a suite of task-based simulation environments (hanging a bag, dressing a mannequin, etc.) involving highly deformable, topologically non-trivial objects. The environments are built on the PyBullet physics engine [28].

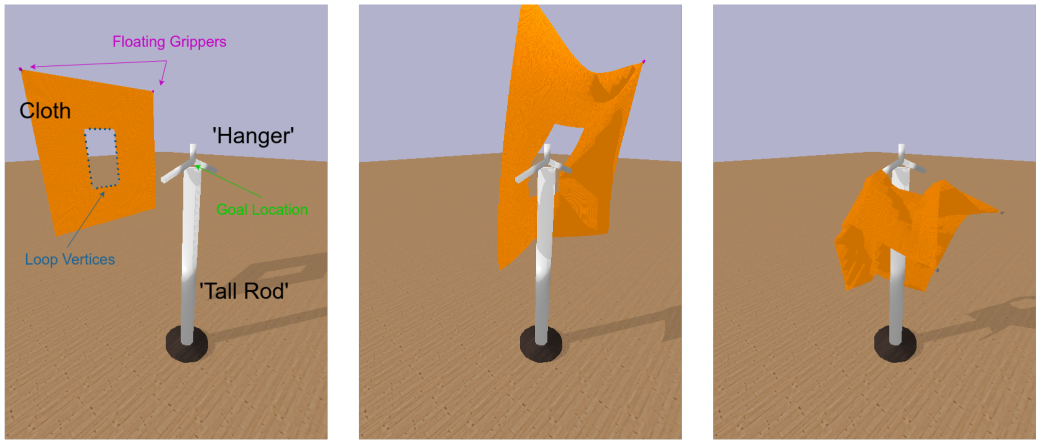


Figure 6: Sample demonstration of the `HangProcCloth task.`

A.1 `HangProcCloth` - Task Definition

For our experiments, we focus on the `HangProcCloth` task (Figure 6), in which a procedurally generated cloth must be placed on a hanger. More specifically, the cloth is generated to contain a hole in its topology - to successfully the complete the task, the vertical part of the hanger should be aligned *through* the hole.

The hanger (anchor) is loaded into the PyBullet engine as a pre-defined rigid body, and contains two components: a 'tall rod', and the 'hanger' itself. While we randomize the anchor pose throughout our experiments, this geometry remains fixed. The **goal** of the task is explicitly formulated in the environment as the center of the 'hanger' component (Figure 6) - while this goal definition is not passed as input to our models, it is used later by our success metric for evaluation.

A.2 Cloth Generation

Following DEDO's implementation, every cloth in our experiments is procedurally generated as a rectangular mesh, and can be represented using the following parameters:

`node_density`	25	The amount of vertices to initialize the cloth mesh with. Every cloth is initialized as an evenly spaced `node_density` $\times$ `node_density` grid (25 $\times$25 = 625 vertices for all of our cloths). Vertices are then removed during the hole generation process.
`width`	[0.25, 1.0]	The width of the cloth.
`height`	[0.25, 1.0]	The height of the cloth.
`num_holes`	(1..2)	The number of holes in the cloth.
`holes`	See A.2.1	See A.2.1

A.2.1 Hole Generation

Holes are created by removing mesh vertices. All generated holes are rectangular - as such, they can be represented topologically with respect to the procedurally generated cloth by their bottom-left and top-right corners. Accordingly, the `holes` parameter is a list, where each element corresponding to a specific hole in the cloth is a dictionary with elements:

`x0`	The x vertex coordinate of the bottom-left corner of the hole.
`y0`	The y vertex coordinate of the bottom-left corner of the hole.
`x1`	Similar to `x0`, for the top-right corner.
`y1`	Similar to `y0`, for the top-right corner.

For reference, the single-hole cloth used in our first experiment (Generalization to Unseen Scene Configuration) is defined as:

```
{
    "node_density": 25,
    "width": 1.0,
    "height": 1.0,
    "num_holes": 1,
    "holes": [
        {"x0": 8, "y0": 9, "x1": 16, "y1": 13}
    ]
}
```

In general, holes are randomly generated under the following constraints:

`x_range`	(2, `node_density` - 2)	The range of possible values for `x0`.
`y_range`	(2, `node_density` - 2)	The range of possible values for `y0`.
`width_range`	(1, `int(round(node_density * 0.3))`)	The range of possible values for w_h, such that $\texttt{x1} = \texttt{x0} + w_h$.
`height_range`	(1, `int(round(node_density * 0.3))`)	The range of possible values for w_h, such that $\texttt{x1} = \texttt{x0} + w_h$.

More precisely, when generating holes, `x0` and `y0` are first sampled based on `x_range` and `y_range`, respectively - `x1` and `y1` are then sampled based on `width_range` and `height_range`. To ensure that the resulting cloth geometry is valid topologically, DEDO generates cloths using a Monte Carlo method, only returning valid holes if they pass a boundary check (all vertices lie within the cloth boundary) and an overlap check (different holes do not overlap). For further implementation details, we refer to the DEDO codebase [7].

Since holes are generated by directly manipulating the cloth mesh, they can also be represented as deformable loops, defined by a set of "loop vertices" (Figure 6) - while information about these vertices is not passed as input to our models, it is used later by our success metric for evaluation.

A.3 Cloth Control

The `HangProcCloth` environment does not model a robot grasp - instead, the cloth is manipulated by applying force controls to floating "grippers"[1] attached to the top-left and top-right corners of the cloth (Figure 6). The grippers themselves are zero-mass and collision free.

Pseudo-expert Policy: To generate demonstrations, we hard-code a pseudo-expert policy with access to privileged environment information. In particular, at the initial state of the scene, this policy computes the distance vector from the centroid of the loop vertices to the goal location in the

[1] DEDO refers to these grippers as "anchors." We refrain from this terminology since "anchor" denotes an entirely different object for our purposes.

hanger. If the cloth has multiple holes, a single hole is selected. This distance vector is then scaled by a hard-coded value (0.04) to convert it to a velocity within the DEDO action space, and passed as the target velocity for *both* grippers to DEDO's default velocity controls[2] (a simple proportional controller, where force is applied proportional to the velocity error) with a velocity gain of 50 and a maximum output force[3] of 5.

Evaluation Policy: To evaluate our models, we implement a separate evaluation policy without access to privileged environment information (e.g. goal location, deformable loop vertices). At the initial state of the scene, we run TAX3D on the full point cloud of the cloth[4], and obtain the predicted position in the world frame of the two grippers (which are attached to the top-left and top-right corners of the cloth). These target positions are then passed as inputs (in addition to a target velocity of zero) to a custom proportional-derivative controller, with a position gain of 50, a velocity gain of 50, and a maximum output force of 5.

A.4 Episode Rollout

A.4.1 Rollout Phases

Each episode rollout consists of two phases (Figure 6): a **manipulation phase**, in which the grippers receive force control inputs at each time step to manipulate the cloth, and a **release phase**, in which the grippers "release" the cloth and allow it to fall. The release phase is fixed at 500 simulation steps, whereas the manipulation phase has a variable episode length depending on the setting (with each environment step corresponding to 8 simulation steps).

If the task is completed successfully, the cloth should be supported by the rigid anchor *after* the release phase. However, because we are learning a goal-prediction module to condition a policy's control outputs, we use the post-manipulation, pre-release state of the cloth to label ground truth demonstrations.

A.4.2 Success Metric

To robustly determine whether or not the task has been successfully completed, we implement our own success metric consisting of two components:

1. **Centroid Check:** a *post-manipulation* binary metric that checks if the centroid of the deformable loop vertices is within a threshold distance (1.0) of the goal location.
2. **Polygon Check:** a *post-release* binary metric that projects the loop vertices and goal location onto the xy-plane, and then checks if the projected goal point lies on the interior of the polygon defined by the projected loop vertices. This is an intuitive heuristic that checks whether or not the hole "wraps" around the vertical rod of the hanger.

If the cloth has multiple holes, these metrics are computed individually for each hole - the task is considered successful if both are true for at least one hole.

A.5 Demonstration Generation

A.5.1 Randomizing Scene Configuration

For all demonstrations across all experiments, the objects in the scene are initialized to the following pose, shown in Figure 6:

[2]Before passing to the controller, we also add 0.01 to the z-component of both target velocities - we found that this empirically produced better aligned placements.

[3]Following the DEDO implementation, this is not an overall maximum force magnitude - it is the maximum magnitude of the force along the x-, y-, and z-axes.

[4]Note that this is different from our training procedure (B.2), where we downsample cloth point clouds to 512 points.

	position (xyz)	orientation (Euler)
cloth	(0, 5, 8)	$\left(-\frac{\pi}{2}, 0, \frac{3\pi}{2}\right)$
hanger	(0, 0, 8)	(0, 0, 0)
tall rod	(0, 0, 0)	(0, 0, 0)

For each demonstration, a `speed_factor` is also randomly sampled, such that `speed_factor` $= e^s, s \sim \mathrm{U}(0.0, 0.7)$. The episode length and the target velocity of the pseudo-expert policy are linearly scaled based on the `speed_factor` (the former by the `speed_factor` itself and the latter by its reciprocal), with a `speed_factor` of 1 corresponding to an episode length of 200. Intuitively, the `speed_factor` adds noise to the deformation undergone by the cloth during the task completion.

To generate a demonstration, the pseudo-expert policy is rolled out under these initial conditions, with $(\mathbf{P}_\mathcal{A}, \mathbf{P}^*_\mathcal{A}, \mathbf{P}^*_\mathcal{B})$ all collected under the same initial configuration. The success metric is then used to evaluate the pseudo-expert rollout, with the entire demonstration discarded if unsuccessful. To randomize scene configuration, the goal point clouds $(\mathbf{P}^*_\mathcal{A}, \mathbf{P}^*_\mathcal{B})$ are then transformed with a randomly sampled translation, and a randomly sampled rotation about the z-axis.

	Unseen	Unseen (OOD)
x-translation	$(-5, 5)$	$(-10, -5) \cup (5, 10)$
y-translation	(0, -10)	(0, -10)
z-translation	0	$(1, 5)$
z-rotation	$\left(-\frac{\pi}{3}, \frac{\pi}{3}\right)$	$\left(-\frac{\pi}{3}, \frac{\pi}{3}\right)$

All transformations are sampled uniformly at random from their respective ranges, with one small caveat: x-translations are chosen such that their signs match the sign of the sampled rotation. That is, if the z-rotation is sampled to be non-negative, then the x-translations are only sampled from the non-negative subset of the corresponding range. This ensures that the anchor always "faces" the cloth, such that the cloth need not undergo significant rotations for a successful placement. Note that this simplification assumes a canonical configuration of the grippers with respect to the hanger for all demonstrations (namely, that the grippers are aligned along the length of the hanger's "shoulders"), despite the fact that a viable placement could be achieved with any arbitrary configuration (for example, if the grippers were aligned perpendicularly to the hanger's shoulders). We leave an exploration of more generalized placements for future work.

As for the point clouds themselves, we obtain $\mathbf{P}^*_\mathcal{B}$ as a partial point cloud from an RGB-D render of the initial state of the environment (since the anchor is static). To guarantee correspondences, $\mathbf{P}_\mathcal{A}$ and $\mathbf{P}^*_\mathcal{A}$ are directly extracted from the mesh vertices of the cloth at its initial and post-manipulation states, respectively.

A.5.2 Experiment Datasets

As a reminder, cloth geometry (including holes) is randomized by following the parameter ranges and procedures described in A.2.1. The datasets for each experiment are generated as follows, where each tuple entry corresponds to the (Train, Unseen, Unseen (OOD)) settings, respectively:

	# cloths	# holes per cloth	# demos per cloth	# total demonstrations
Unseen Scenes	(1, 1, 1)	(1, 1, 1)	(400, 40, 40)	(400, 40, 40)
Unseen Cloths	(100, 10, 10)	(1, 1, 1)	(4, 4, 4)	(400, 40, 40)
Multimodal Goals	(200, 20, 20)	(1/2, 1/2, 1/2)	(4, 4, 4)	(800, 80, 80)

For the Unseen Scenes experiment, all demonstrations use the same cloth geometry. For the Multimodal Goals experiment, training cloths are split evenly into 100 single-hole cloths and 100 double-hole cloths (and similarly for Unseen and Unseen(OOD)). For double-hole cloths, 2 demonstrations are generated for each hole, preserving a total of 4 demonstrations per cloth. For all demonstrations across all experiments, the anchor pose is randomized as described in A.5.1.

B Training Details

B.1 Model Architecture

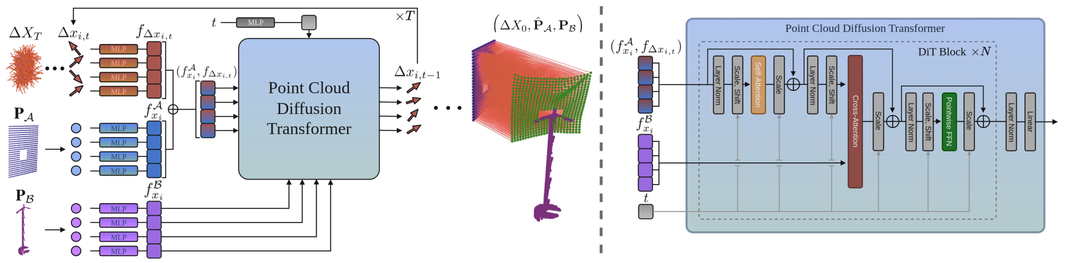


Figure 7: TAX3D model architecture. *(Left)*. During inference, randomly sampled displacements $\Delta X_T \sim \mathcal{N}(0, \mathbf{I})$ are de-noised conditioned on action ($\mathbf{P}_{\mathcal{A}}$) and anchor ($\mathbf{P}_{\mathcal{B}}$) features; the final ΔX_0 is predicted to displace the action into a goal configuration. *(Right)*. Our modified DiT [25] architecture combines self-attention and cross-attention for object-centric and scene-level reasoning.

As discussed in 5.2, we modify the standard DiT block [25] to include an additional cross-attention head 7. For all of our experiments, we train the same architecture:

`depth`	5	# of DiT blocks
`num_heads`	4	# heads per block
`hidden_size`	128	hidden size per block

These settings (namely, `num_heads` and `hidden_size`) are applied identically to the self-attention and cross-attenion layers. During training and inference, our model always uses 100 diffusion steps, with a linear noise schedule.

B.2 Training Pre-Processing & Hyperparameters

For training, both the action and anchor point clouds are downsampled to 512 points using furthest point sampling[5]. The anchor point cloud is additionally augmented with z-axis rotations sampled uniformly at random from $[0, 2\pi]$.

All models are trained under the same hyperparameters with AdamW optimization and cosine scheduling with warmup:

`learning_rate`	1×10^{-4}
`learning_rate_warmup_steps`	100
`weight decay`	1×10^{-5}
`epochs`	20,000
`batch_size`	16

C Evaluation Metrics

As discussed in Section 6, our method's modeling of point-wise displacements allows us to directly use root-mean-squared-error (RMSE) as a distance metric between predicted and ground truth configurations of the cloth. To appropriately evaluate distributional predictions in our setting, we define two evaluation metrics[6]:

[5]During policy evaluation, only the anchor point cloud is downsampled, as the full action point cloud is need to obtain target positions for the grippers.

[6]Both metrics bear strong similarity to the MMD metric, but are essentially modified to aggregate across different reference sets.

1. **Coverage RMSE:** For each demonstration with ground truth $\mathbf{P}^*_{\mathcal{A},i}$, we sample 20 predictions $\{\hat{\mathbf{P}}_{\mathcal{A},j}\}$, and keep the minimum RMSE. This is aggregated across all demonstrations in the dataset. Intuitively, this metric captures how well a model can produce all of the modes in a given dataset - that is, how well it *covers* a distribution.
2. **Precision RMSE:** We first collect demonstrations corresponding to a specific cloth geometry (for our experiments using this metric, there are 4 demonstrations per cloth) - for some cloth $\mathcal{C}$, this serves as a cloth-specific reference set $\{\mathbf{P}^*_{\mathcal{A},i}\}_{\mathcal{C}}$. We then sample 80 predictions conditioned on cloth $\mathcal{C}$, and compute for each prediction $\hat{\mathbf{P}}_{\mathcal{A},j}$ the minimum RMSE to ground truth point clouds in the reference set $\{\mathbf{P}^*_{\mathcal{A},i}\}_{\mathcal{C}}$[7]. This is aggregated across all 80 predictions, and then across all cloths. Intuitively, this metric captures how well a model can consistently produce predictions that are close to the dataset configurations - that is, how *precisely* it models a distribution.

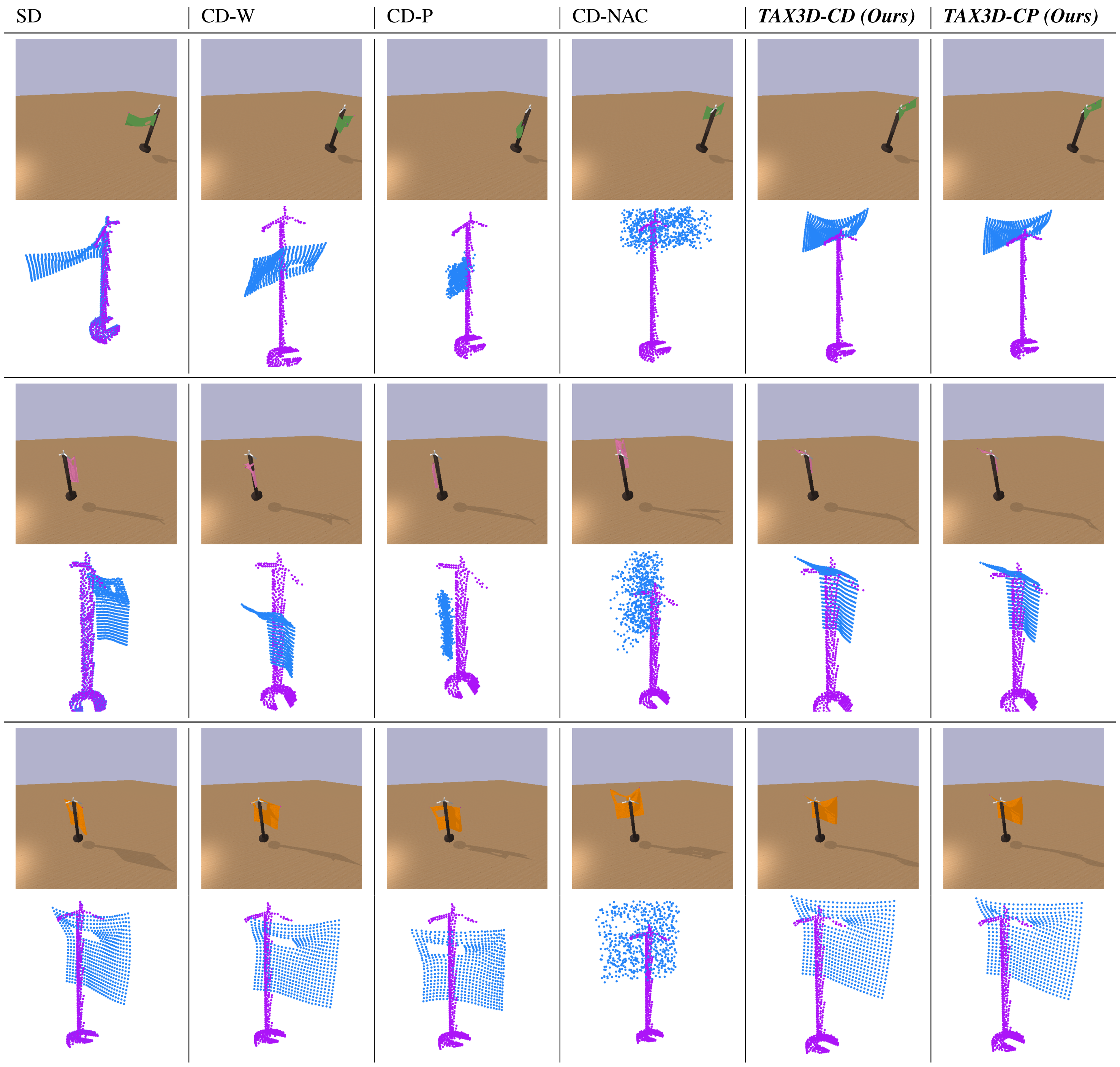
D Experiments

The following pages contain visualizations of our method (as well as all baseline methods) across classes of cloth geometry (single- and double-hole) on out-of-distribution scene configurations. For every visualized prediction, both the cloth and configuration were unseen by the model during training.

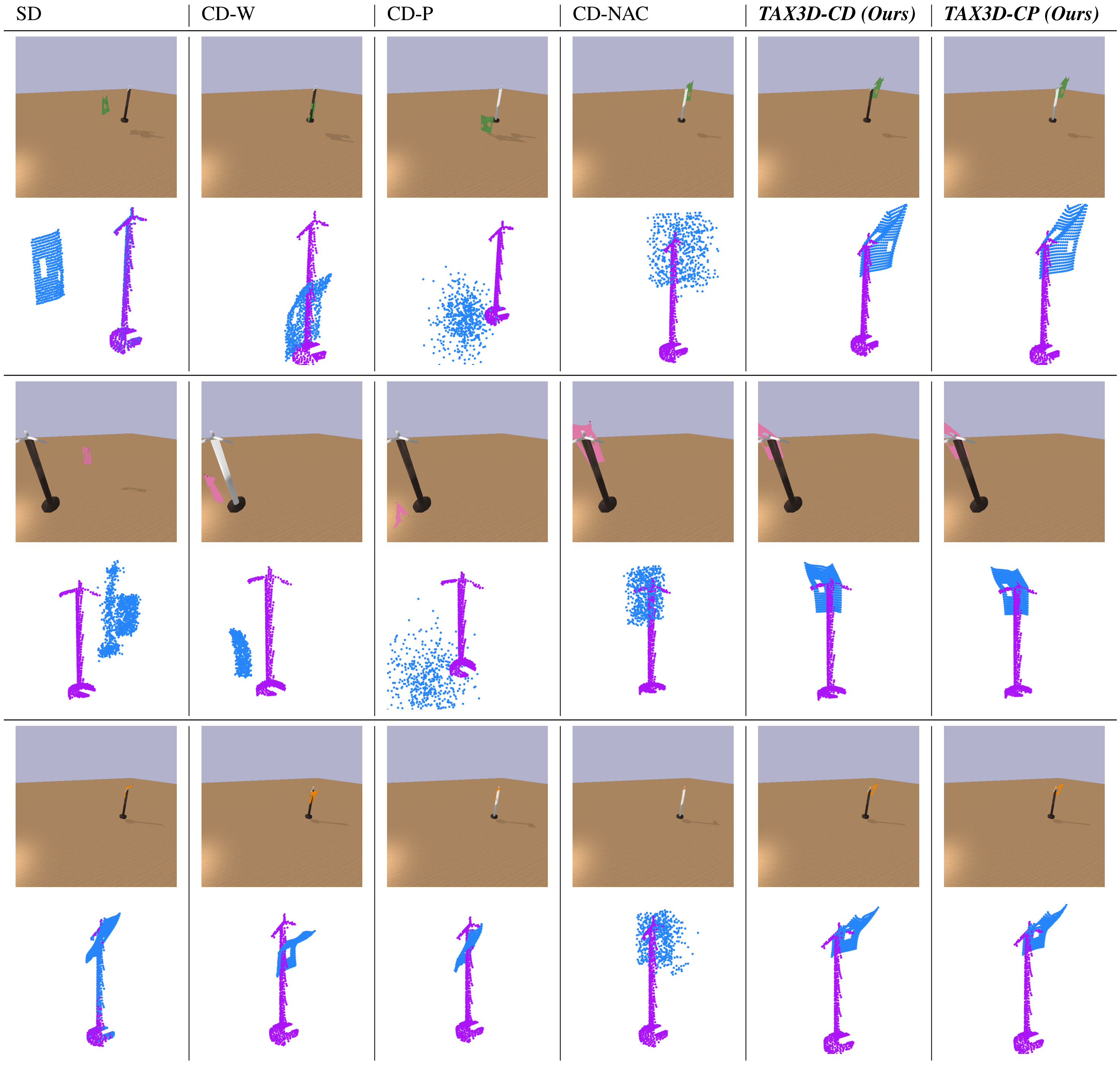
Within each table row, the top displays the result of the evaluation policy rollout on the corresponding model's predicted cloth configuration; the bottom displays the predicted configuration itself. Zooming in may be necessary to properly view the policy executions. For more visualizations, including videos of the policy rollout and the full reverse diffusion process, see our anonymized project page.

[7]Because demonstrations are sampled with random scene configurations, we additionally invert the anchor transformation from A.5.1 so that RMSEs are computed relative to the same anchor pose. Without this step, the RMSEs would be meaningless, as different ground truth point clouds $\mathbf{P}^*_{\mathcal{A},i}$ would be in different positions in the world frame.

D.1 Single-Hole Cloth, Unseen Configuration (Out-of-distribution)



544 **D.2 Double-Hole Cloth, Unseen Configuration (Out-of-distribution)**


SD
CD-W
CD-P
CD-NAC
TAX3D-CD (Ours)
TAX3D-CP (Ours)