A Appendix

A.1 Tasks

Here we give more details about the tasks, including the performance functions, teacher dataset, and sample images. Fig. 6 shows images all of simulation environments used for SOTA comparisons and generalizability, with one end-effector. In each environment, the end-effectors are pickers (white spheres).

1. CLOTH FOLD: Fold a square cloth in half, along a specified line. The performance metric is the distance of the cloth particles left of the folding line, to those on the right of the folding line. A fully folded cloth should have these two halves virtually overlap. Teacher demonstrations are from an agent with two pickers (*i.e.,* $\mathcal{D}_{Teacher} = \mathcal{D}^{2p}_{Demo}$); we solve the task on a student agent with one picker. Task variations are in cloth rotation.

2. DRY CLOTH: Pick up a square cloth from the ground and hang it on a plank to dry, variant of [46]. The performance metric is the number of cloth particles (in simulation) on either side of the plank and above the ground. Teacher demonstrations are from an agent with two pickers (*i.e.,* $\mathcal{D}_{Teacher} = \mathcal{D}^{2p}_{Demo}$); we solve the task on a student agent with one picker. Task variations are in cloth rotations and translations with respect to the plank.

3. THREE BOXES: A simple 2D environment where three boxes of different sizes are randomly placed and need to be moved to designated goal locations. Teacher demonstrations are from an agent with three pickers (*i.e.,* $\mathcal{D}_{Teacher} = \mathcal{D}^{3p}_{Demo}$); we solve the task on student agents with one picker and two pickers. Performance is measured by the distance of each object from its goal location. This task is used to illustrate the generalizability of **MAIL** with various n-to-m end-effector transfers, and is not used in the SOTA comparisons.

A.2 Ablations

A.2.1 Ablate the method for creating optimized dataset $\mathcal{D}_{Student}$

We answer the question: how do different methods perform in creating optimized dataset $\mathcal{D}_{Student}$? We ablate the optimizer used to create $\mathcal{D}_{Student}$ from the demonstrations, labeled ABL1 in Fig. 7, and compare the following methods, given state inputs from $\mathcal{D}_{Teacher}$.

- Random: A trivial random guesser, that serves as a lower benchmark.
- SAC: An RL algorithm that tries to reach the goal states of the demonstrations.
- Covariant Matrix Adaption Evolution Strategy (CMA-ES): An evolutionary strategy that samples optimization parameters from a multi-variate Gaussian, and updates the mean and covariance at each iteration.
- Cross-Entropy Method (CEM, ours): A well-known gradient-free optimizer, where we assume a Gaussian distribution for optimization parameters.

We did not use gradient-based trajectory optimizers since the contact-rich simulation will give rise to discontinuous dynamics and noisy gradients. As shown in Table 1a, SAC is unable to improve upon the random baseline, likely because of the very large state-space of our environment (> 15000 states for > 5000 cloth particles) and error accumulations from the imprecision of learned dynamics model. Trajectory optimizers achieve the highest performance, and we chose CEM as the best optimizer based on the performance of the optimized trajectory.

A.2.2 Ablate the dynamics model

We answer the question: what is the best architecture to learn the task dynamics? We ablate the learned dynamics model $\mathcal{T}_\psi$, labeled ABL2 in Fig. 7. The environment state is the state from

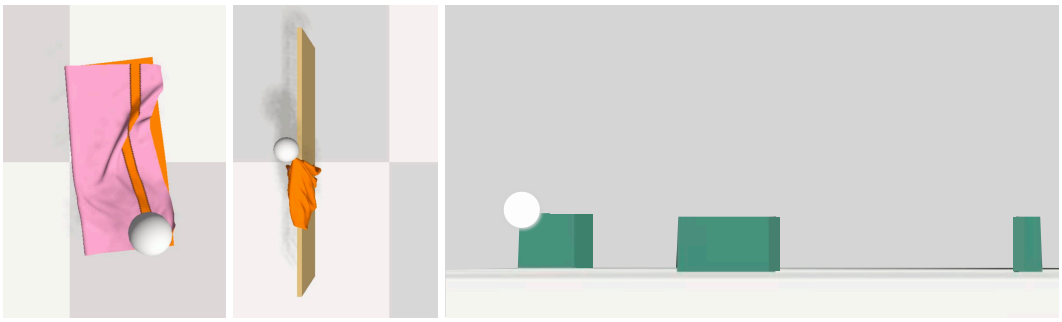

Figure 6: **Environments** used in our experiments, with one end-effector. The end-effectors are pickers (white spheres). In CLOTH FOLD (left) the robot has to fold the cloth (orange and pink) along an edge (inspired by the SoftGym [45] two-picker cloth fold task). In DRY CLOTH (middle) the robot has to hang the cloth (orange and pink) on the drying rack (brown plank). In THREE BOXES (right), the robot has to move three rigid boxes in a 2D environment.

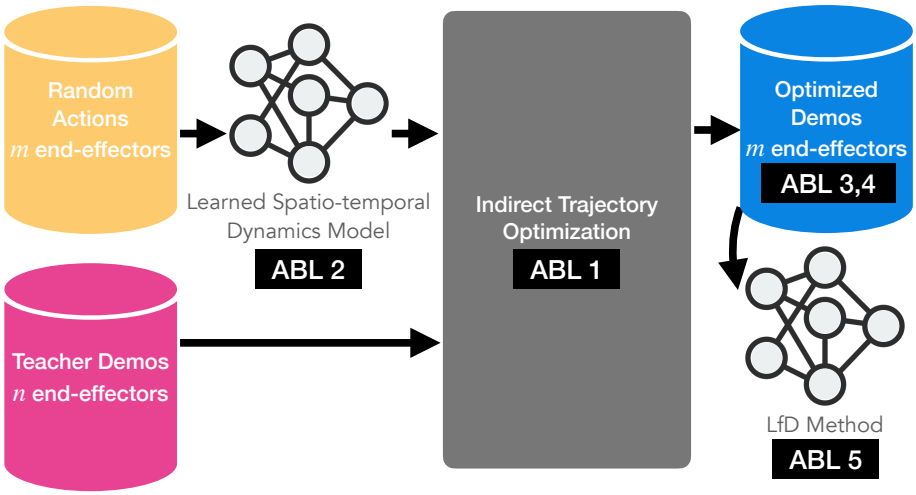

Figure 7: **Ablations** to **MAIL** components.

$\mathcal{D}_{Teacher}$ *i.e.,* positions of cloth particles. This is a structured but large state space since the cloth is discretized into $>$ 5000 particles.

Table 1b shows the performance of trajectories achieved by using the dynamics models. We see that CNN-LSTM models work better than models that contain only CNNs, graph networks (GNS), or LSTMs. We hypothesize that this is the case since we need to capture the spatial structure of cloth and capture a temporal element across the whole trajectory since particle velocity is not captured in the state. Further, a 1D CNN works better because the cloth state can be simply represented as a 2D vector ($N \times 3$ which represents the xyz for N particles). This is easier to learn with than the 3D state vector fed into 2D CNNs.

GNS performs poorly also due to the reasons of error accumulation from large displacements, discussed in Sec. 4.2. Our learned dynamics model $\mathcal{T}_\psi$ was significantly faster than the simulator. We tested it on a simple training run of SAC [5], without parallelization. Our learned dynamics gave 162 fps, about $50x$ faster than the 3.4 fps with the simulator. The accuracy was tolerable for trajectory optimization, as shown in Fig. 8.

A.2.3 Compare performance of optimized dataset $\mathcal{D}^{1p}_{Optim}$

We answer the question: how good is $\mathcal{D}_{Student}$ compared to the recorded demonstrations? This ablation gauges the performance of the optimized dataset that we used as the student dataset for

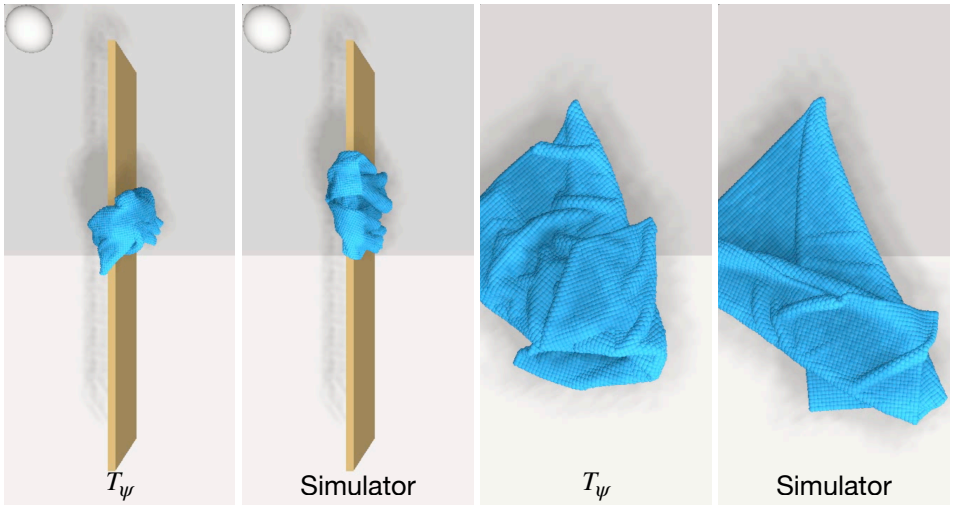

Figure 8: **Predictions of the learned spatio-temporal dynamics model $\mathcal{T}_\psi$ and the FleX simulator.** Predictions are made for the same state and action, shown for both cloth tasks. The learned model supports optimization approximately $50x$ faster than the simulator, albeit at the cost of accuracy.

LfD, $\mathcal{D}_{Student} = \mathcal{D}^{1p}_{Optim}$. We compare this to other relevant datasets to solve the task, as shown in Table 1c. It is labeled ABL3 in Fig. 7. The two-picker demonstrations $\mathcal{D}^{2p}_{Demo}$ are recorded for an agent with two pickers as end-effectors. This is used as the teacher demonstrations in our experiment $\mathcal{D}_{Teacher} = \mathcal{D}^{2p}_{Demo}$. The one-picker demonstrations $\mathcal{D}^{1p}_{Demo}$ are recorded for an agent with one picker as an end-effector. This is to contrast against the optimized demonstrations in the same morphology, $\mathcal{D}^{1p}_{Optim}$. The random action trajectories are with a one-picker agent, added as a lower performance benchmark. They are the same random trajectories used to train the spatio-temporal dynamics model $\mathcal{T}_\psi$. Naturally, the teacher dataset is the best, as it is trivial to do this task with two pickers. The one-picker dataset has about the same performance as the optimized dataset $\mathcal{D}^{1p}_{Optim}$, both of which are suboptimal, as it is not trivial to manipulate cloth with one hand. *This is the kind of task we wish to unlock with this work: tasks that are easy to do for teachers in one morphology but difficult to program or record demonstrations for in the student's morphology.* Note that $\mathcal{D}^{1p}_{Optim}$ has been optimized on the fast but inaccurate learned dynamics model, which is one reason for the reduced performance. This is why the downstream LfD method uses the simulator, as accuracy is very important in the final policy.

A.2.4 Ablate modality of demonstrations

We answer the question: how well does the downstream LfD method handle different kinds of demonstrations? This ablates the composition of the student dataset fed into LfD, and is labeled ABL4 in Fig. 7. We compare the following datasets for $\mathcal{D}_{Student}$, using the notation for datasets explained in Sec. 3.1:

- Demonstrations in one-picker morphology, $\mathcal{D}^{1p}_{Demo}$: These are non-trivial to create and are thus not as performant, discussed above. Creating these is increasingly difficult as the task becomes more challenging.
- Optimized demos, $\mathcal{D}^{1p}_{Optim}$: This is optimized from the two-picker teacher demonstrations ($\mathcal{D}_{Teacher} = \mathcal{D}^{2p}_{Demo}$), which are easy to collect as the task is trivial with two pickers.
- 50% $\mathcal{D}^{1p}_{Demo}$ and 50% $\mathcal{D}^{1p}_{Optim}$: A mix of trajectories from the two cases above. This is an example of handling multiple demonstrators with different morphologies.

Method	$25^{th}\%$	$\mu \pm \sigma$	median	$75^{th}\%$
Random	0.000	0.003±0.088	0.000	0.000
SAC	0.000	0.000±0.006	0.000	0.000
CMA-ES	0.104	0.270±0.258	0.286	0.489
CEM	0.351	0.502±0.242	0.501	0.702

(a) Ablation on the method chosen for creating demonstrations.

Method	$25^{th}\%$	$\mu \pm \sigma$	median	$75^{th}\%$
GNS	-0.182	0.002±0.223	-0.042	0.149
2D CNN, LSTM	0.157	0.376±0.305	0.382	0.602
No CNN, LSTM	0.327	0.465±0.213	0.463	0.595
1D CNN, No LSTM	0.202	0.407±0.237	0.387	0.587
1D CNN, LSTM (ours)	0.351	0.502±0.242	0.501	0.702

(b) Ablation on the dynamics network architecture.

Dataset	$25^{th}\%$	$\mu \pm \sigma$	median	$75^{th}\%$
$\mathcal{D}_{Random}$	0.000	0.003±0.088	0.000	0.000
$\mathcal{D}^{1p}_{Demo}$	0.344	0.484±0.169	0.446	0.641
$\mathcal{D}^{2p}_{Demo}$	0.696	0.744±0.068	0.724	0.785
$\mathcal{D}^{1p}_{Optim}$	0.351	0.502±0.242	0.501	0.702

(c) Compare the performance of the optimized dataset.

Table 1: **Ablation results for MAIL**

Fig. 9 illustrates that all three variants achieve similar final performance. This demonstrates that the downstream LfD method is capable of solving the task with a variety of suboptimal demonstrations. This could be from one dataset of demonstrations, or even a combination of datasets obtained from a heterogeneous set of teachers.

An interesting observation here is that by comparing Fig. 9 and Table 1c, *we see that the final policy is better than the suboptimal demonstrations by a considerable margin, and also slightly improves upon the performance of the teacher demonstrations.* This improvement comes from the LfD method's ability to effectively utilize demonstrations and generalize across task variations. This result, combined with the ablation that we need demonstrations in Sec. 4.2, shows that our downstream LfD method is well adapted to work with suboptimal demonstrations to solve a task.

A.2.5 Ablate Reference State Initialization in DMfD

We answer the question: how does the use of demonstration state matching affect the downstream LfD? An improvement we made over the original DMfD algorithm is to disable matching with expert states, known as RSI-IR, first proposed in [42]. We justify this improvement in this ablation, labeled ABL5 in Fig. 7.

As shown in Fig. 10, removing RSI and IR has a net positive effect throughout training, and around 10% on the final policy performance. This means that matching expert states exactly via imitation reward does not help, even during the initial stages of training when the policy is randomly initialized. We believe this is because RSI helps when there are hard-to-reach intermediate states that the policy cannot reach during the initial stages of training. This is true for dynamic or long-horizon tasks, such as karate chops and roundhouse kicks. However, our tasks are quasi-static, and also have a short horizon of 3 for the cloth tasks. In other words, removing this technique allows the policy to

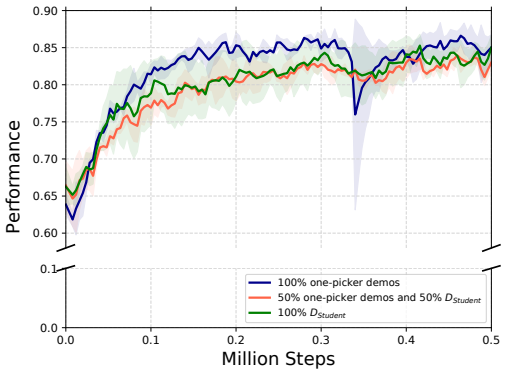

Figure 9: **Ablation on the modality of demonstrations on LfD performance.** Similar performance shows that **MAIL** can learn from a wide variety of demonstrations, or even a mixture of them, without loss in performance. See Sec. A.2.4.

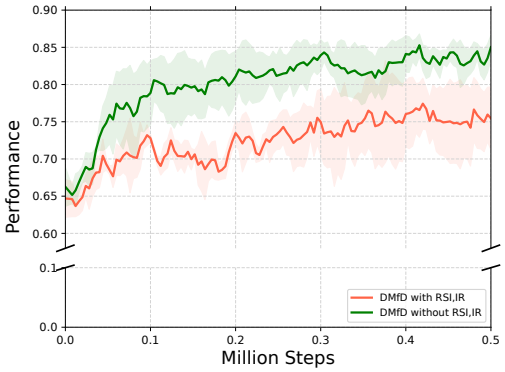

Figure 10: **Ablation on the effect of reference state initialization (RSI) and imitation reward (IR) on LfD performance.** RSI is not helpful here because our tasks are not as dynamic or long horizon as DeepMimic [42]. See Sec. A.2.5.

freely explore the state space while the demonstrations can still guide the RL policy learning via the advantage-weighted loss from DMfD.

A.2.6 Ablate the effect of cross-morphology on SOTA

We answer the question: how do established LfD baselines perform across morphologies? We studied the effect of why baselines such as GAIfO and GPIL performed so poorly on our tasks. In our experiments, we noticed a number of factors (such as variations in the task, diversity of demonstrations, etc.). This ablation studies the effect of cross-morphology in the demonstrations, where we compare the performance of GAIfO, when provided demonstrations from the teacher dataset $\mathcal{D}_{Teacher}$ and student dataset $\mathcal{D}_{Student}$.

As we can see in Table 2, there is a 36% performance improvement when using the (suboptimal) student dataset. Obviously, since the demonstration actions are not available to learn from, the primary difference that the agent sees during training is the richness of demonstration states. Thus, improvement is because of the demonstration states seen in the student dataset. Since the student morphology has only one picker, any demonstration for the task (DryCloth) includes multiple intermediate states of the cloth in various conditions of being partially hung for drying. By contrast, the teacher requires fewer pick-place steps to complete the task, and thus there are fewer intermediate states in the demonstrations.

A.2.7 Ablate the effect of environment difficulty on LfD baselines

We answer the question: how do established LfD baselines perform across environments? Given the subpar performance of the LfD baselines GAIfO and GPIL on our SOTA environments, we ablated the effect of environment difficulty. We took the easy cloth environment (CLOTH FOLD) and used an easier variant of it, CLOTH FOLD DIAGONAL PINNED [41]. In this variant, the agent has to perform an easier fold, but one corner of the cloth is pinned to prevent sliding. Moreover, the desired fold is across the diagonal of the cloth, which can be done by manipulating only one corner of the cloth. We used the state-based observations, and the action space is the small-displacement action space, where the agent outputs incremental picker displacements instead of pick-and-place locations. This action space is similar to those seen in the experiments of GNS, GAIfO and GPIL, where they worked with rigid objects in simulation. This is an easy version of our CLOTH FOLD environment. We can see in Table 3 that the same baselines are able to perform significantly better in this environment. Hence, we believe manipulating with long-horizon pick-place actions, with an

image observation, makes it challenging for the baselines to work in challenging cloth environments described in Sec. 4.1.

Method	$25^{th}\%$	$\mu \pm \sigma$	median	$75^{th}\%$
$\mathbf{D_{Teacher}}$	-0.198	-0.055±0.183	-0.043	0.078
$\mathbf{D_{Student}}$	0.199	0.363±0.245	0.409	0.528

Table 2: Ablation of GAIfO on the effect of cross-morphology. We compare the normalized performance, measured at the end of the task.

Method	$25^{th}\%$	$\mu \pm \sigma$	median	$75^{th}\%$
GPIL	0.356	0.427±0.162	0.487	0.553
GAIfO	0.115	0.374±0.267	0.471	0.592

Table 3: Measuring performance on the easy cloth task, CLOTH FOLD DIAGONAL PINNED. We compare the normalized performance, measured at the end of the task.

A.3 List of environments we tried for LfD baseline ablations

Two LfD baselines, GAIfO and GPIL, seemed to perform quite poorly, although we expected better performance. In an effort to understand why these fail, we performed a host of studies with different varieties of easier environments, to isolate the properties of the environment that make it the most challenging to succeed. A list of the different task variants we tried are given below. The ones with the most striking difference in performance are described in further detail in Sec. A.2.6 and Sec. A.2.7.

1. Used the easier CLOTH FOLD environment instead of DRY CLOTH.
2. Used state-based environments instead of image-based environments.
3. Reduced the number of variations of the task distribution $\mathcal{V}$.
4. Used the small-displacement action-space that is used in GNS and GAIfO experiments, instead of the large-displacement pick-place action spaces.
5. Removed the effect of cross-morphology, by providing demonstrations in the students morphology.

A.4 Hyperparameter choices for MAIL

In this section, Table 4 shows the hyperparameters chosen for training the inverse dynamics model $\mathcal{T}_\psi$. Table 5 shows the details of CEM hyperparameter choices. Table 6 shows the hyperparameters for our chosen LfD method (DMfD).

Parameter	Description
CNN	4 layers, 32 channels, 3x3 kernel, leaky ReLU activation. stride = 2 for the first layer, stride = 1 for subsequent layers
LSTM	One layer Hidden size = 32
Other Parameters	Learning rate α =1e-5 Batch size = 128

Table 4: Hyper-parameters for training the forward dynamics model.

	Planning Horizon	Number of optimization iterations	Number of env interactions
1	1	2	21,000
2	2	2	15,000
3	2	2	21,000
4	2	2	31,000
5	2	2	34,000
6	2	10	21,000
7	2	1	21,000
8	2	1	15,000
9	2	1	32,000
10	3	2	21,000
11	3	10	21,000
12	4	2	21,000
13	4	10	21,000

Table 5: CEM hyper-parameters tested for tuning the trajectory optimization. We conducted ten rollouts for each parameter set and used the set with the highest average normalized performance on the teacher demonstrations. Population size is determined by the number of environment interactions. The number of elites for each CEM iteration is 10% of population size.

Parameter	Description
State encoding	Fully connected network (FCN) 2 hidden layers of 1024, ReLU activation
Image encoding	32x32 RGB input, with random crops. CNN: 4 layers, 32 channels, stride 1, 3x3 kernel, leaky ReLU activation FCN: 1 layer of 1024 neurons, $tanh$ activation
Actor	Fully connected network 2 hidden layers of 1024, leaky ReLU activation
Critic	Fully connected network 2 hidden layers of 1024, leaky ReLU activation
Other parameters	Discount factor: $\gamma = 0.9$ Entropy loss weight: $w_E = 0.1$ Entropy regularizer coefficient: $\alpha = 0.5$ Batch size = 256 Replay buffer size = 600,000 RSI-IR probability = 0 (disabled)

Table 6: Hyper-parameters used in the LfD method (DMfD).

A.5 Performance metrics for real-world cloth experiments

In this section, we explain the metrics for measuring performance of the cloth, to explain the sim2real results discussed in Fig. 4.2

For CLOTH FOLD task, we measure performance at time t by the number of pixels of the top color $pix_{top,t}$ and bottom color $pix_{bot,t}$ of the flattened cloth, compared to the maximum number of pixels, pix_{max} (Fig. 11).

For DRY CLOTH task, it is challenging to measure pixels on the sides and top of the plank. Moreover, we could be double counting pixels if they are visible in both side and top views. Hence, we measure the cloth to determine whether the length of the cloth *on top of* the plank is equal to or greater than the side of the square cloth. We call this the spread metric.

The policies achieve $\sim 80\%$ performance, which is about the average performance of our method in simulation, for both tasks. However, since these performance metrics are different in the simulation and real world, we cannot *quantify* the sim2real gap through these numbers.

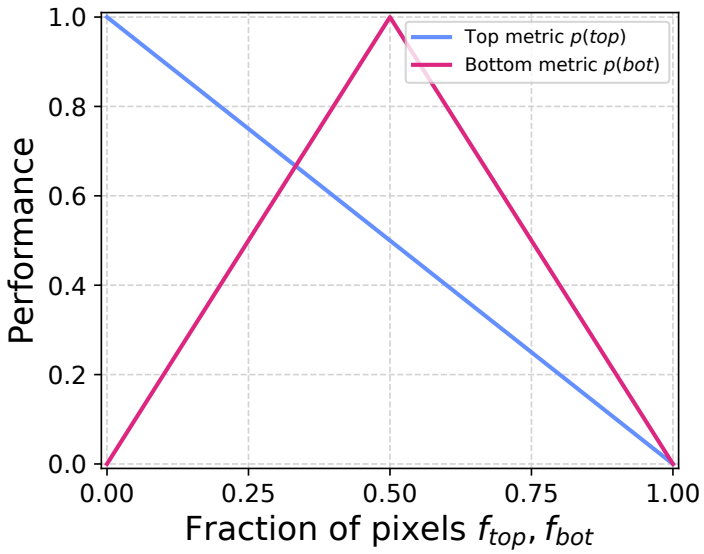

Figure 11: **Performance function for** CLOTH FOLD **on the real robot.** At time t, we measure the fraction of pixels visible to the maximum number of pixels visible $f_{top} = pix_{top,t}/pix_{max}$ and $f_{bot} = pix_{bot,t}/pix_{max}$. Performance for the top of the cloth should be 1 when it is not visible, $p(top) = 1 - f_{top}$. Performance for the bottom of the cloth should be 1 when it is exactly half-folded on top of the top side, $p(bot) = \min\left[2\left(1 - f_{bot}\right), 2f_{bot}\right]$. Final performance is an average of both metrics, $p(s_t) = p(top) + p(bottom)/2$. Note that the cloth is flattened at the start, thus $pix_{max} = pix_{top,0}$.