I. Appendix

A. Ablation of Objective Terms

The objectives terms we propose are designed to be useful in many cases, but to understand this better, we ablate several of the objective terms. We evaluate the importance of the transformation validity objective, occupancy objective, and delta minimum distance objective by repeating our experiments. In each ablation, we omit one objective term. Each condition was run 10 times with different random seeds. The results are shown in Table I and Figure 1.

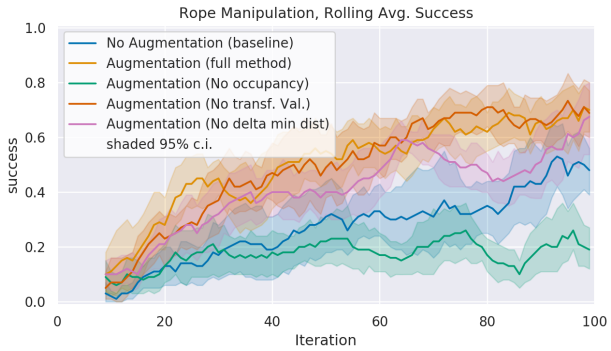

Fig. 1. Success vs Iterations for ablations. Bimanual rope manipulation, in simulation.

Method	Task Success % (↑) (rope)	Position Error (m) (↓) (planar pushing)
Full Method	0.700 (0.118)	0.0010 (0.0001)
No transf. valid.	0.700 (0.185)	0.0011 (0.0001)
No delta min dist	0.675 (0.185)	0.0012 (0.0002)
No occupancy	0.24 (0.136)	0.0028 (0.0011)

TABLE I
Ablations: Metrics on both tasks, with various objective terms removed. The standard deviation is shown in parentheses.

We find that across both experiments, the most important objective was the occupancy objective. Without this objective, our method produced augmentations where contacts and penetrations are not preserved. This is invalid, and training on these examples produces a worse model even than using no augmentations.

The next most important objective is the delta minimum distance objective. In our simulated rope experiment, the method without this objective performs slightly worse. By visualizing the different examples (Figure 2), we found that without this objective, the augmentation transforms examples which were near the hooks and

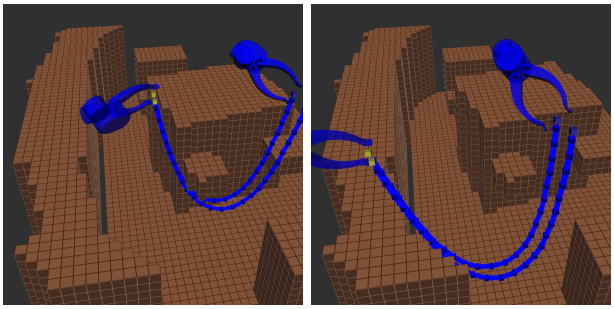

Fig. 2. Augmented rope data generated without the delta minimum distance objective. On the left is the original transition, under the hook. On the right is the augmented transition, which is far from the hook. Without the delta minimum distance objective, data can be augmented away from interesting regions, becoming less relevant. This is based on the heuristic that contact and near-contact events are relevant for manipulation.

protrusions far into free space. In contrast, with the delta minimum distance objective, these examples stay near the hooks. Since the planning tasks involve moving in narrow passages around these hooks, it's more relevant to produce more examples in this area. Hence, by preserving the distance to nearby obstacles, we also tend to keep examples in areas of interest. In the planar pushing task, however, omitting the delta minimum distance objective did not have a notable effect. This could be in part due to the cluttered nature of the scene, which means that most transformations are small.

Finally, we find that the valid transformation objective can be omitted without effecting performance in these two tasks. In the planar pushing case this is completely expected, because there are no SE(2) transformations which are always invalid or irrelevant. However, in the rope experiment, this result is less intuitive. We found that omitting this objective occasionally produced augmentations where the rope is floating sideways in physically impossible states. This result suggests that even if a few augmentations are invalid or irrelevant, training on the augmentations still significantly outperforms using no augmentations. Even if omitting the objective does not degrade performance in these experiments, it is a natural term to include and may be beneficial in other applications.

B. Choosing the Number of Augmentations

Our method produces multiple distinct augmentations for each original example. More augmentations should improve generalization after training, but at the cost of additional computation time. In this experiment, we explore how the number of augmentations per-example

affects generalization. We tested this in the rope domain on the task of learning the constraint checker. The original un-augmented dataset came from the first three iterations of learning. With these examples, we then generated varying numbers of augmentations for each example and evaluated on a held-out test example drawn from the fourth iteration. Figure 3 shows three rope transitions labeled 0 from the original three training examples, followed by the held-out test example. Intuitively we would expect, with enough augmentations, that from these original training examples we would be able to generate augmented examples which are very similar to the held-out test example.

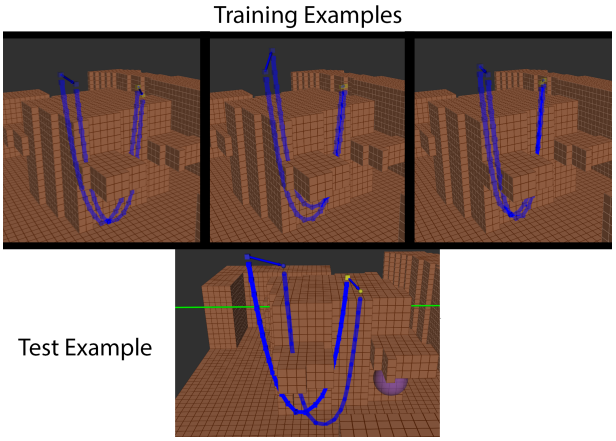

Fig. 3. (top) These three original training example show rope transitions with inaccurate predictions, where the rope is predicted to move inside the hook on the engine. (bottom) The test transition is similar, but is not identical. The proposed augmentation method can move the rope transition while ensuring it still intersects the hook in a similar way, which allows it to generate augmentations like the test example.

For each number of augmentations, we generated 3 augmented datasets (using different random seeds) and trained 3 models on each dataset (again using different random seeds), for a total of 9 data points for each number of augmentations. The results are shown in Figure 4. The y-axis is the classifier output on the test example, which has a true label of 0 (lower is better). Without any augmentations, the classifier has an output near 1, which is incorrect. Improvement began around 10 augmentations, and plateaued by 20. Since computation cost was not significant, we chose 25 augmentations per-example for all experiments in the main text.

C. Hyperparameters

The maximum number of iterations for stepping and projecting are $N_p = 5$ and $M_p = 25$, and we stop the outer loop if the change in transform is ever less than $\delta_p = 0.001$. When solving Problem 3, we stop if the gradient is smaller than $\epsilon_p = 0.0003$. There are also

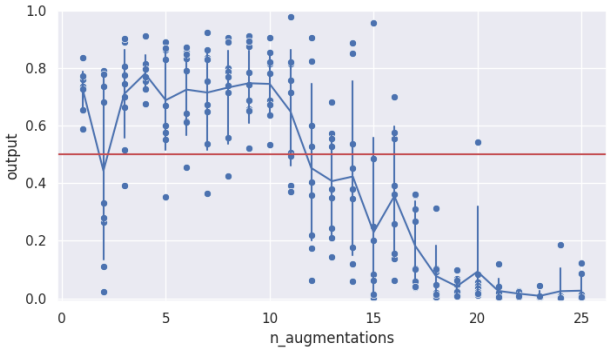

Fig. 4. Number of augmentations versus classifier output on a test example. Lower is better. Performance plateaus at around 25 augmentations, in this example.

learning rate, learning rate decay, and weighting parameters used when solving Problem 3. For the objective function weighting terms, we use $\beta_1 = 0.05, \beta_2 = 1, \beta_3 = 1, \beta_4 = 0.1$. The weighting terms were tuned so that the magnitude of the different weighted losses terms were comparable, and learning rate and number of iterations were tuned to maximize convergence. All values used are documented in our code, which can be found on our project website.