## A Algorithm

We list a detailed step-by-step overview of M3L in Algorithm 1.

**Algorithm 1** Masked Multimodal Learning (M3L)

```
1:  Initialize MAE and PPO parameters
2:  repeat
3:      // Collect rollouts
4:      Initialize rollout buffer $\mathcal{B} \leftarrow 0$
5:      repeat
6:          Read latest visual and tactile inputs
7:          Feed inputs through frozen networks without masking to compute representations $z$
8:          Use the current policy to compute the control action from $z$
9:          Store transition (with original inputs) to the rollout buffer
10:     until maximum of $N^{\text{PPO}}$ environment interactions
11:     // Update networks
12:     Train MAE and PPO using the latest rollouts for $M$ epochs following Figure 1
13: until maximum number $N^{\text{PPO}}_{\text{max}}$ of environment interactions
```

## B Background for Proximal Policy Optimization

Let $\pi_\theta$ be a policy (or actor function) parameterized by $\theta$ that outputs an action $a_t$ given a state $s_t$, $V_{\theta_V}$ be a value (or critic) function parameterized by $\theta_V$, and $\hat{A}_t$ be be an estimator of the advantage function (Sutton & Barto, 2018) at a timestep $t$. The goal of Proximal Policy Optimization (PPO) (Schulman et al., 2017) is to address the problem of vanilla policy gradient update (Sutton & Barto, 2018) that often causes destructively large parameter updates. To this end, PPO introduces a new objective for training the actor that minimizes the following clipped surrogate objective, which is a lower bound of the conservative policy iteration objective (Kakade & Langford, 2002):

$$\mathcal{L}^{\texttt{clip}} = \hat{\mathbb{E}}_t \left[ \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right] \tag{3}$$

where $\hat{\mathbb{E}}_t$ denotes an empirical average over a minibatch, $\epsilon$ is a hyperparameter, $r_t(\theta)$ is a probability ratio $\pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$, and $\theta_{\text{old}}$ are the actor parameters before the update. Note that this objective discourages large updates that would make the $r_t$ be outside the range of $(1-\epsilon, 1+\epsilon)$.

For training the critic function, PPO minimizes the following objective:

$$\mathcal{L}^{\texttt{critic}} = \hat{\mathbb{E}}_t \left[ \frac{1}{2}(V_{\theta_V}(s_t) - \hat{V}_t^{\texttt{targ}})^2 \right] \tag{4}$$

where $\hat{V}_t^{\texttt{targ}}$ is a target value estimate computed with the generalized advantage estimation (GAE) (Schulman et al., 2016), which is also used for computing $\hat{A}_t$. The final objective of PPO is given as follows:

$$\mathcal{L}^{\texttt{PPO}} = \mathcal{L}^{\texttt{clip}} + \beta_V \cdot \mathcal{L}^{\texttt{critic}} + \beta_H \cdot H[\pi] \tag{5}$$

where $H[\pi]$ is an action entropy bonus for exploration and $\beta_V, \beta_H$ are scale hyperparameters.

## C Hyperparameters

The training hyperparameters are listed in Table 1.

## D Tactile Insertion Environment

In Figure 8 we show all pegs and targets used for randomizing training in the tactile insertion environments. The dense reward used for the environment is the following:

$$r = -\log(100 \cdot d + 1) \tag{6}$$

where $d$ is the distance of the peg from the target.

Table 1: Hyperparameters

| Symbol | Description | Value |
|---|---|---|
| | number of parallel PPO envs | 8 |
| | masking ratio | 95% |
| $B$ | batch size | 512 |
| $n$ | number of representation learning steps for hand | 16 |
| $N^{\text{PPO}}$ | PPO rollout buffer length | 32768 for insertion and hand, 4800 for door |
| $M$ | PPO n. epochs | 10 |
| | learning rate PPO | $10^{-4}$ |
| $\beta_T$ | tactile reconstruction weight | 10 |

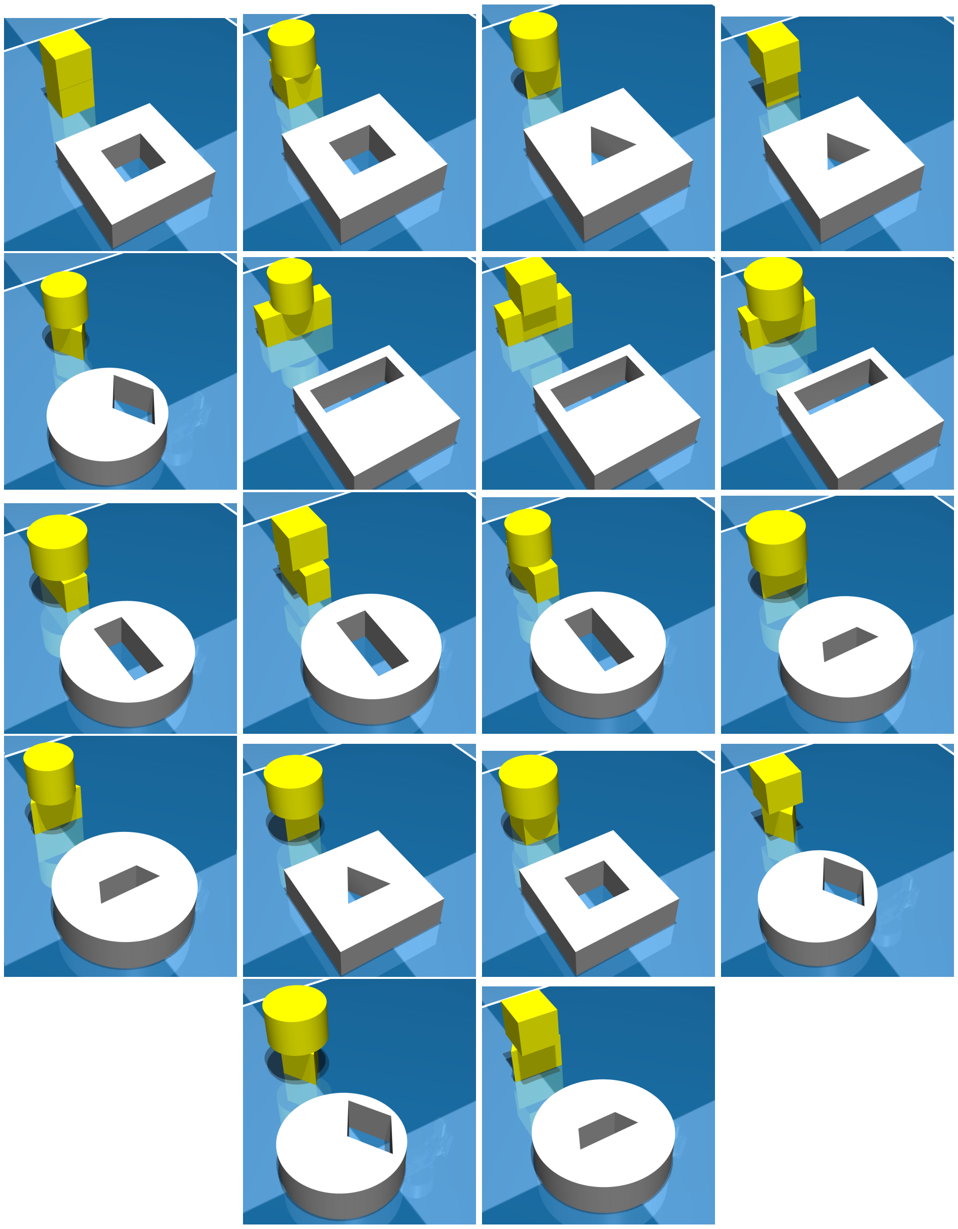


Figure 8: Objects (pegs) and targets used to train the tactile insertion task.

## E IN-HAND CUBE ROTATION ENVIRONMENT

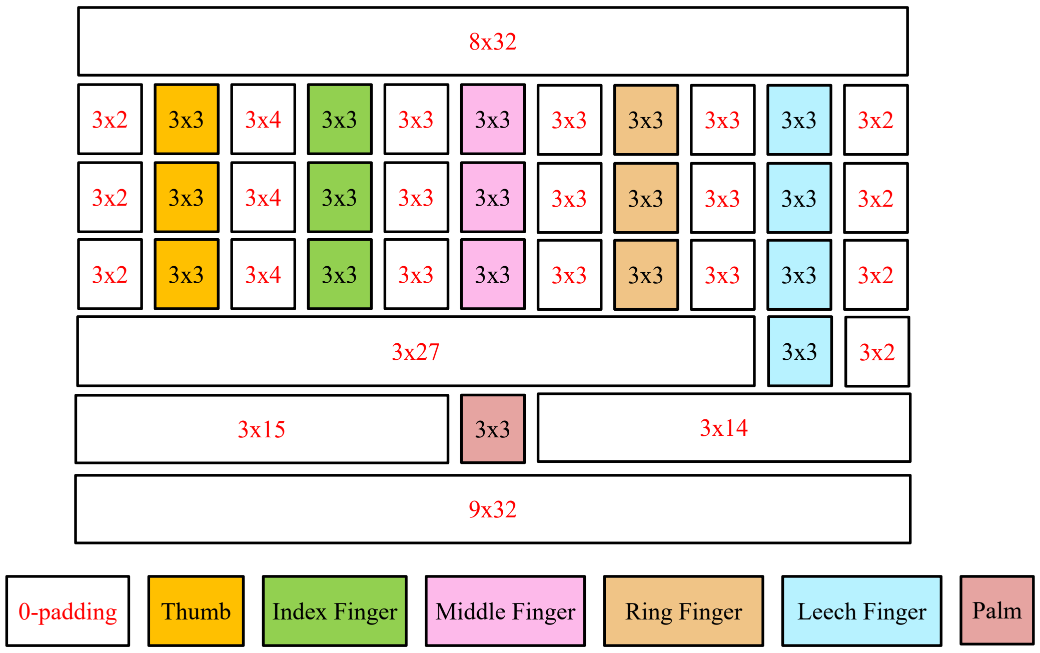


Figure 9: $32 \times 32$ tactile grid for the tactile observation in the in-hand cube rotation environment.

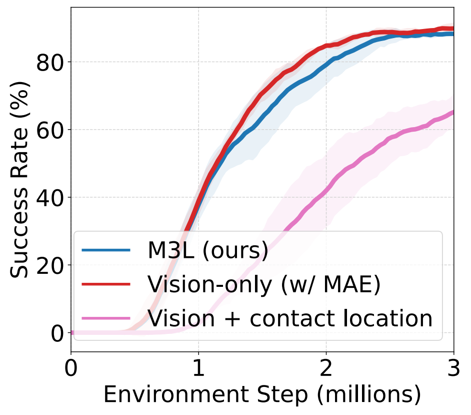
## F ADDITIONAL BASELINES

This section presents additional baselines in addition to those described in Section 6.1. Note that experiments in this section have been executed on three random seeds.

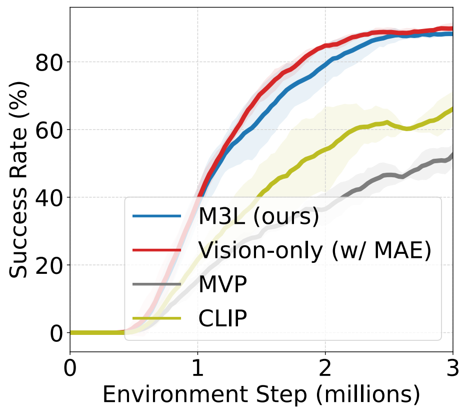
In particular, we implemented a baseline that follows the concurrent work in Qi et al. (2023). In fact, we discretize the contact location, and feed a binarized representation to an MLP that creates a tactile embedding of the stacked frames. Similarly, we feed the visual images to a CNN and concatenate the visual embedding with the tactile embedding. We feed the concatenation into a similar transformer encoder that outputs representations that we input to PPO. This baseline is then trained end-to-end on the insertion task (without employing auxiliary objectives or privileged learning). The results are shown in Figure 10a, with such a baseline underperforming both M3L and the vision-only baseline co-trained with an MAE objective.

Moreover, we compare M3L with two vision-only baselines that rely on frozen pretrained representations, specifically using the visual encoders from Masked Visual Pretraining (MVP, Xiao et al. (2022); Radosavovic et al. (2023)) and Contrastive Language-Image Pretraining (CLIP, Radford et al. (2021)). Note that both encoders are much larger than the one used in M3L (5M for M3L, 22M for MVP, and 88M for CLIP), leading to considerably slower training. The results are shown in Figure 10b, with both M3L and its vision-only MAE variation outperforming MVP and CLIP representation on the in-domain learning task.

Finally, we compare M3L with a touch-only MAE-based baseline (similar to vision-only w/ MAE) on the tactile insertion and door opening tasks. Note that the insertion and task is misspecified when vision is missing, because of missing target information, while the door task is not necessarily requiring vision to be solved. Results are shown in Figure 11, with such a baseline underperforming M3L, particularly on the insertion task.

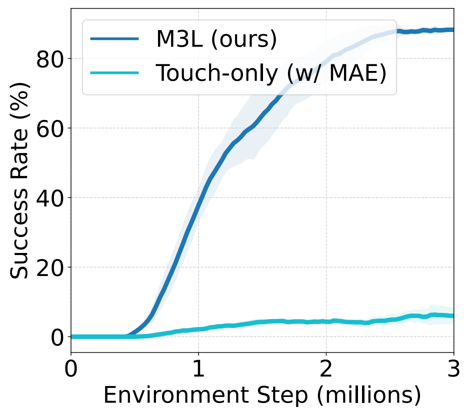


(a) vs contact location only

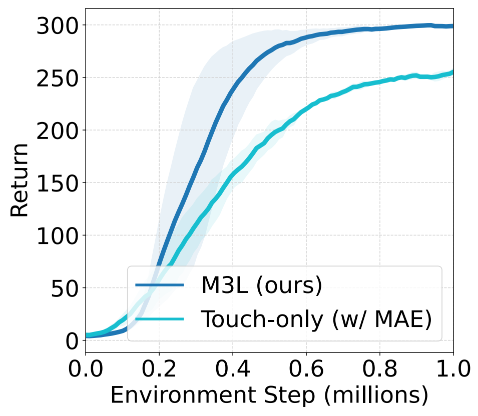


(b) vs pretrained representations

Figure 10: In-domain (tactile insertion) learning curves comparing M3L with baselines.



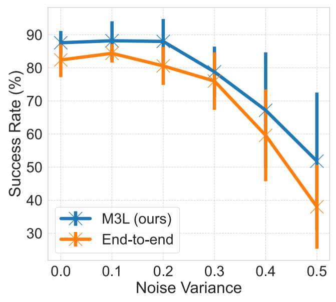
(a) Tactile Insertion



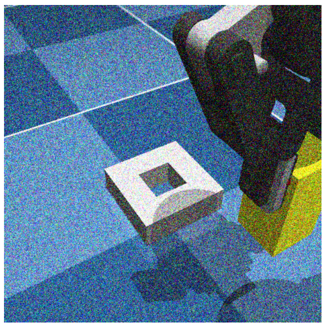
(b) Door Opening

Figure 11: In-domain learning curves comparing M3L with touch-only baseline.

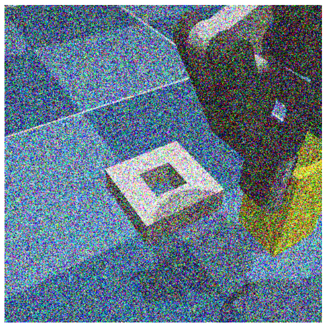
## G Robustness to Noise



(a) Performance vs noise



(b) Image with 0.2 noise variance



(c) Image with 0.5 noise variance

Figure 12: Noise robustness analysis.

We test the robustness of the trained policies by adding zero-mean Gaussian noise to both visual and tactile data at inference time. Figure 12a shows the performance of M3L vs the end-to-end for increasing noise variance, highlighting how M3L success rates do not degrade up to a reasonable degree of disturbances and also consistently outperforms the end-to-end approach also in term of robustness. Note from Figure 12b and Figure 12c how performance only degrades for considerable perturbations of the original image and tactile data.

## H MAE Reconstructions

Example reconstruction of both visual and tactile inputs are shown in Figure 13.

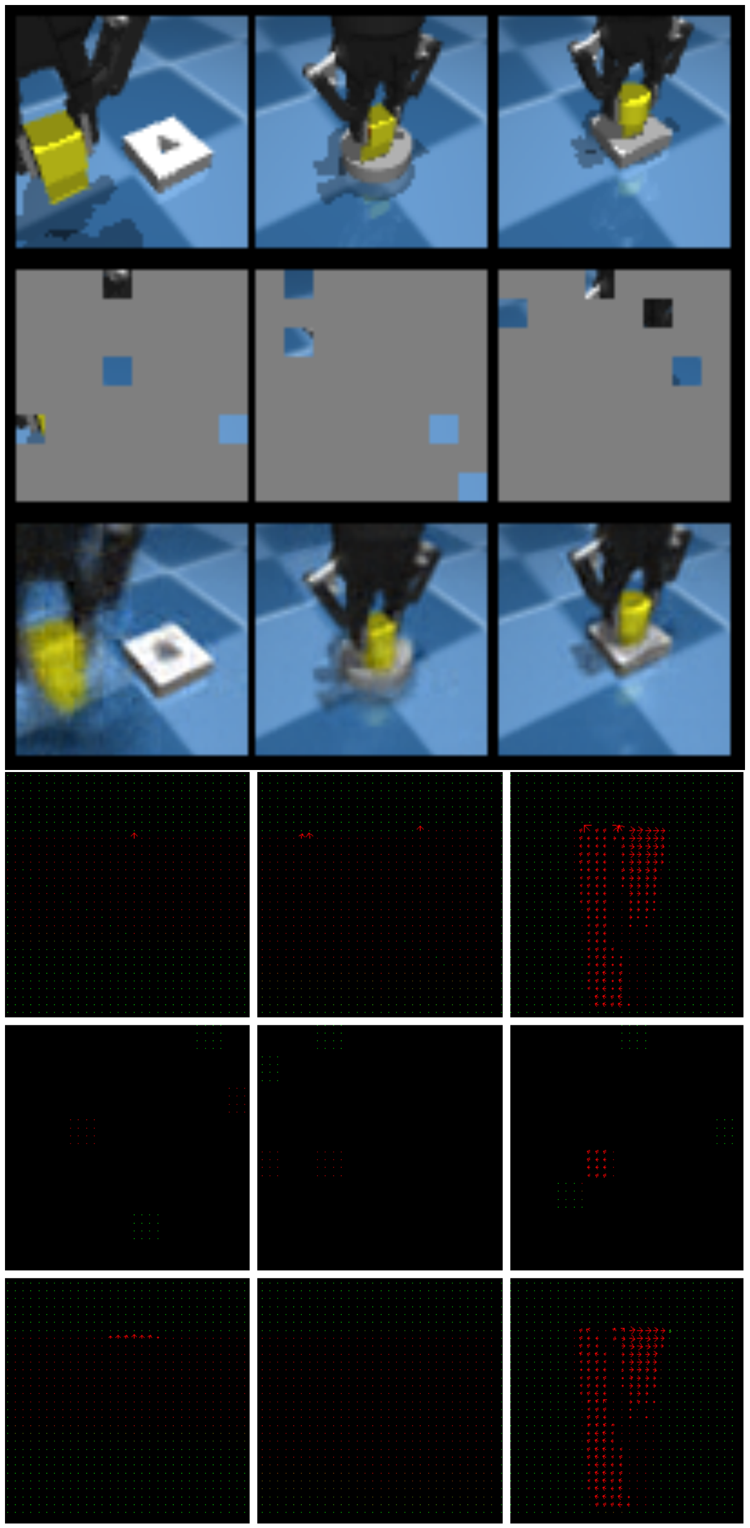


Figure 13: Examples of MAE reconstructions for visual and tactile inputs. First and fourth rows are original inputs. Second and fifth rows are visualizations of the masked and unmasked features (note that we mask convolutional features and not raw input). Third and sixth rows are reconstructions.

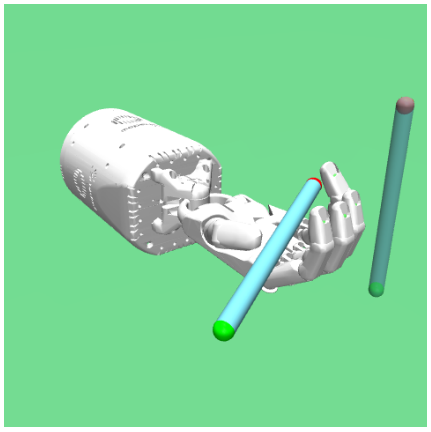
## I Additional Tasks

In this section, we present two additional tasks, **Egg Rotation** and **Pen Rotation**. These tasks are essentially performed in the same environment as the in-hand rotation task described in Section 5.3. However, we replace the cube with an egg and a pen, respectively (see Figure 14). In these new tasks, we test *visual variations*, e.g., the camera pose in the egg task and the color of the pen in the pen task.

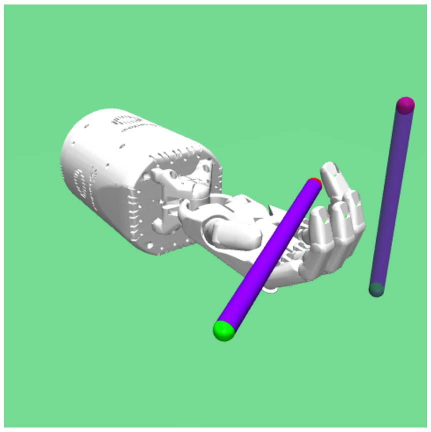
The results are shown in Figure 15. M3L outperforms the baselines in these environments in terms of generalization. In the egg rotation task, where we perturbed the camera pose, M3L vision policy outperforms the vision-only (w/ MAE) approach during training as well as testing. Interestingly, touch is even more crucial for generalization in the pen task, where the pen has an unseen color, largely outperforming M3L vision policy. We believe this is due to the fact that the multimodal policy learns to rely less on vision (having the possibility of using touch too), which leads to more visual robustness even compared to M3L vision-policy. As a sanity check for this, we also tested feeding a constant visual input (i.e. an initial image) to M3L, which led to a major drop in performance (below all other baselines and not shown in the figure). This confirms that M3L effectively leverages both vision and touch to learn robust policies.

(a) Egg Rotation (training observation)

(b) Egg Rotation (test observation)

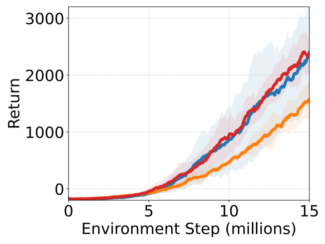


(c) Pen Rotation (training observation)

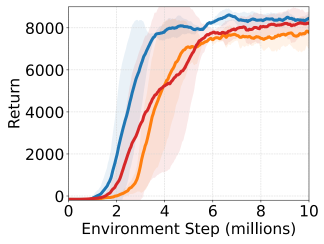


(d) Pen Rotation (test observation)

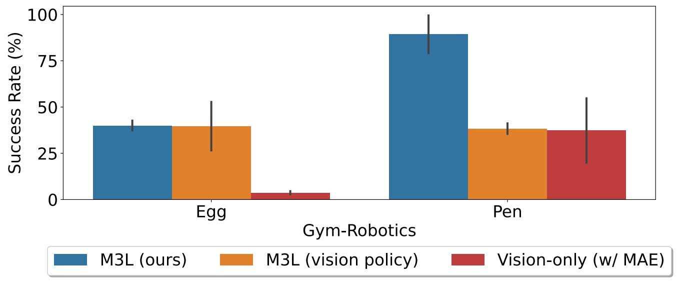
Figure 14: Snapshots from the in-hand egg and pen rotation environments.



(a) In-Hand Egg Rotation



(b) In-Hand Pen Rotation



(c) Generalization results

Figure 15: In-domain (first row) and generalization (second row) performance on the in-hand egg and pen rotation tasks.