Reviewer 1 [Score: 6]

Eqn 1: Reward depends on history of the state. That's right. We can view the problem as a partially observable MDP (POMDP) where the state consists of the agent pose, interaction counts (visitation frequencies), object positions, etc. The recurrent policy network encodes the agent's observation history over time to arrive at a state-representation. Novelty rewards for visual exploration for mapping [57,51,7] are formulated similarly with RNNs.

Approach tries every single object. Actually, key to our approach is that our agents do *not* exhaustively try interactions — they learn to intelligently prioritize what to try (L43-57). The baselines that simply cycle through objects and actions yield very low precision (L264); our model discovers 2.5× more interactions for the same time budget (L251).

Taking a knife/apple is the same...# affordances is very low. The reward in Eqn 1 is provided for every new *interaction* executed by the agent, where an interaction consists of an action (take, slice) coupled with an object instance (apple1, lettuce) — i.e. the reward *does* treat taking a knife vs. taking an apple differently. There are 103 total interactions (L203). This number is defined by the AI2-iTHOR environments and is in no way limited by our approach.

Impact on environment if one tries every action/object. As mentioned in Sec 6 (L309), safety is important to consider when developing interaction exploration policies in the real world. Methods that simultaneously learn to *reset* the environment (e.g., Eysenbach, ICLR 2018) are promising for enabling both safe and efficient RL in these scenarios.

Reviewer 3 [Score: 7]

Noisy odometry. If we add truncated Gaussian noise to odometry readings similar to the popular LoCoBot noise model (Murali et al.), we find that our method's advantages still stand against baselines that rely on odometry observations. See Fig. R1 (middle).

How does IntExp(Obj) know object extents? We use the true object boundaries from the simulator. This corresponds to the agent understanding what visual boundaries are, without knowing object classes or affordances (L176).

Lines 167-172/Eqn 3 inconsistency: $\mathcal{L}(\hat{y}_A, \hat{y}_I, y) = \mathcal{L}_{ce}(\hat{y}_A, y, \forall y_{ij} \neq -1) + \mathcal{L}_{ce}(\hat{y}_I, \mathbb{1}[y = -1], \forall y_{ij})$
Thanks for pointing this out. We have revised the equation above. Each loss term is evaluated over a subset of pixels (third argument) — e.g., for y_A, the classification is ($y = 1$ vs. $y = 0$), and is evaluated over pixels where ($y \neq -1$).

Fully annotated images as an upper bound. This upper bound is already reported in the paper — IntExp(GT) in L226, yellow curve in Fig 3 (right). Fig. R1 (left) shows the equivalent upper bound for downstream tasks (Sec 4.2).

	Retr.	Store	Wash	Heat
IntExp	27.25	27.00	3.75	19.00
IntExp(Sal)	25.00	13.25	3.50	16.50
IntExp+ObjCov	23.00	11.75	4.50	20.25
IntExp(GT)	48.50	57.75	13.00	19.75

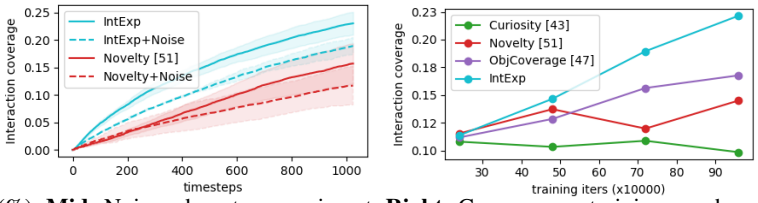

Figure R1: Left: Downstream success rates (%). **Mid:** Noisy odometry experiment. **Right:** Coverage vs. training epoch.

Reviewer 4 [Score: 6]

Training schedule for policy/affordance learning. We provide details in Sec S2 of the Supp. file. In short, we train the policy network for M=200k frames, then train the affordance model. Interleaving the affordance training more frequently results in better segmentation (+1.8% mAP) but no improvement in interaction exploration. Retraining models with reduced M=10k hurts performance (-2.8%), while larger M=500k results in similar coverage (+0.2%).

Performance vs. training time, not just the final quantities. Fig. R1 (right) shows coverage rate vs. training iters. The plot confirms R4's hypothesis. Note, Fig 6 (right) already showed a similar plot (reward vs. training epoch).

How about extending...to combine another exploration strategy? Fig. R1 (left) shows two such variants: (1) IntExp (Sal)[1] does interaction exploration, but rather than affordance maps it uses saliency maps, which are also interaction/object-oriented; (2) IntExp+ObjCov combines our reward with object coverage rewards. It performs slightly worse on the interaction exploration task (-0.27%) but marginally better downstream on *wash* and *heat*.

Why point-based > obj-based? It initially surprised us too. We found that the policy network tends to overfit more easily using the dense affordance predictions from IntExp(Obj) since the policy and affordance model use the same training environments. Its accuracy on training environments is indeed higher (25.16 vs. 23.61). The more conservative predictions from IntExp(Pt) generalize better to unseen test environments (Fig 3 right, L255-7).

[1] We already showed IntExp(Sal) in the submitted paper, Fig 3 (right); here we add it for the downstream tasks per R4's request.