Object Abstraction in Visual Model-Based Reinforcement Learning

Rishi Veerapaneni[*,1], **John D. Co-Reyes**[*,1], **Michael Chang**[*,1], **Michael Janner**[1]
Chelsea Finn[2], **Jiajun Wu**[3], **Joshua Tenenbaum**[3], **Sergey Levine**[1]

Learning to generalize to *any* novel compositional object manipulation task from raw visual input, such as stacking blocks into various towers, is challenging because the number of tasks grows combinatorially with the number and arrangement of discrete entities. To reduce this combinatorial complexity, this paper departs from the traditional paradigm of training models that globally process a single *scene* representation vector (and variants thereof; see Fig. 2b,c,d) and instead proposes to train models that locally and symmetrically process several *entity*-representations (Fig. 2a). We use the term *object abstraction* to refer to the abstraction barrier that isolates the entity-generic specification of the model from the identities of the entity-specific instances. Imposing object abstraction on a model allows permutation-invariant processing of the hidden entity-states and naturally scales to different numbers of entities without increasing the model's parameter count, thereby reducing modeling the combinatorial complexity of scenes to the simpler complexity of entities and their relations.

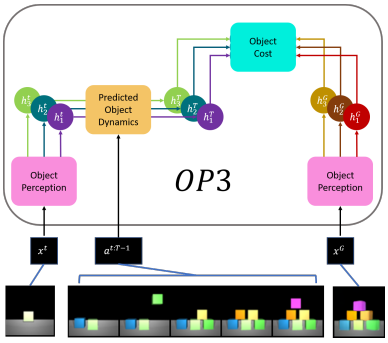

Figure 1: OP3 infers a set of hidden states $h^t_{1:K}$ from an observation x^t and predicts their future states given a sequence of actions $a^{t:T-1}$. Rolled-out plans are evaluated by scoring final predicted states against inferred goal hidden states h^g_k.

Prior works have proposed to model the dynamics of complex scenes with high-dimensional visual observations [4, 8, 20, 29], but particularly relevant to our model are methods with objects as primitives [1, 2]. Works that enforce object coherence typically require additional supervision [10, 13] or access to additional preprocessing, such as segmentations [14], crops [9], or a simulator [15, 30], while those that do not assume such additional information often factorize the entire scene into pixel-level entities [26, 32], which do not model objects as coherent wholes.

We present a framework for *object-centric perception, prediction, and planning* (OP3) that recovers representations of scenes that factorize effectively over coherent objects, and utilizes these entity-representations for planning, without requiring additional supervision besides the raw image pixels (Fig. 1) To *bind* the properties of the concrete physical entities to the factorized entity-representations, we model the problem as a state-factorized partially observable MDP and propose an amortized interactive inference algorithm for inferring the values of the hidden entity-representations.

Figure 2: Our approach to model-based reinforcement learning imposes **object abstraction**: **(a)** The hidden state is factorized into *local* entity states, symmetrically processed by the same function which handles *generic entities*. In contrast, prior work do not symmetrically process the hidden state. **(b)** The hidden state represents the scene *globally*, processed with a single function [12, 22, 23, 33]. **(c)** Different chunks of the hidden state represent different entities and the entire state processed with a single function [3, 18, 28]. **(d)** The hidden state is factorized into *local* entity states, processed by different functions [17, 19, 31].

* Equal contribution. [1]University of California Berkeley. [2]Stanford University. [3]MIT.
Workshop on Perception as Generative Reasoning, NeurIPS 2019, Vancouver, Canada.

1 Object-Centric Perception, Prediction, and Planning (OP3)

This section describes the general structure of the observation and dynamics models that implement object abstraction, the inference algorithm that binds values to the factorized hidden state, and the planning algorithm that plans with hidden entity-states as an intermediate representation. Under the language of factorized POMDPs, a set of latent entities $h^*_{1:K}$ generates an image observation x via an observation function $\mathcal{G}(x \mid h^*_{1:K})$. The dynamics function $\mathcal{D}(h^{t+1,*}_{1:K} \mid h^{t,*}_{1:K}, a^t)$ describes how an action a intervenes on the states of these entities to produce their states at the next timestep. Each task is specified by a reward function $\mathcal{R}(h^*_{1:K}, h^{G,*}_{1:K})$ which measures the distance between the current entities $h^*_{1:K}$ and the goal scene entities $h^{G,*}_{1:K}$. The goal scene specifies the task and depicts a particular arrangement of objects such as a block tower. We seek to train a model-based RL agent that can executing a sequence of actions to manipulate a physical scene with a variable number of entities to match the objects in the goal scene.

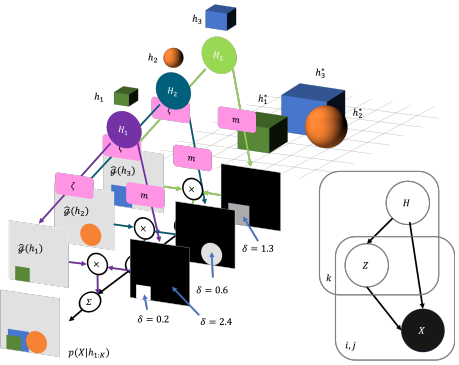

Figure 3: **(a)** The **observation model** $\hat{\mathcal{G}}$ models an observation image as a composition of sub-images weighted by segmentation masks. The shades of gray in the masks indicate the depth from the camera of the object that the sub-image depicts. **(b)** The graphical model of the generative model of observations k ranging from 1 to K, i from 1 to N, and J from 1 to M.

1.1 OP3: A Model with Object Abstraction

We do not know the *true* state h^*_k of each entity k of the physical environment so will attempt to estimate the value h_k whose uncertainty is represented by the random variable H_k. The observation model $\hat{\mathcal{G}}\left(X \mid H_{1:K}\right)$ and dynamics model $\hat{\mathcal{D}}\left(H^{t+1}_{1:K} \mid H^t_{1:K}, A^t\right)$ operate symmetrically on $H_{1:K}$, which are assigned through a binding mechanism (Sec. 1.2).

Observation Model: The observation model $\hat{\mathcal{G}}$ models how the objects $H_{1:K}$ cause the image observation $X \in \mathbb{R}^{N \times M}$. Each object H_k is rendered independently using the same function $\hat{g}(H_k)$, and the resulting K sub-images are combined to form the final image observation X. The image is modeled as $p\left(X|H_{1:K}\right) = \sum_{k=1}^{K} m\left(H_k\right) \cdot \zeta\left(H_k\right)$ where mixture components $\zeta(H_k) = p\left(X|Z_k = 1, H_k\right)$ model the sub-images $\hat{g}(H_k)$, and mixture weights $m(H_k) = p\left(Z_k = 1|H_k\right)$ model the segmentation masks. See Appx. 3.5 for details.

Dynamics Model: The dynamics model $\hat{\mathcal{D}}$ models how each object H^t_k is affected by action A^t and the other objects $H^t_{\neq k}$. It applies the same function $\hat{d}(H^t_k, H^t_{\neq k}, A^t)$ to each state, composed of several functions illustrated and described in Appx. 3.1. A key feature is that an action is modeled as an intervention on each object *individually* rather than on the entire scene, which produces a finer-grained intervention mechanism than methods with global scene representations.

1.2 Interactive Inference for Binding Object Properties to Latent Variables

As OP3 is a latent variable model, we can bind properties of physical entities to the hidden states by inferring the parameters of the distributions of each H_k. We build on the IODINE framework [11], which iteratively refines [21] the posterior $p\left(H_{1:K}|X\right)$ via a recognition model $\mathcal{Q}(H_{1:K} \mid x, \hat{h}_{1:K})$ from a previous estimate $\hat{h}_{1:K}$. In real-world scenes, a single static image may be insufficient for disambiguating objects, which may require observing dynamic object interactions produced from an action sequence. We thus propose an *interactive inference* algorithm (Figs. 4,9) that incorporates temporal continuity and interactive feedback to approximate the posterior $p\left(H^t_{1:K} \mid X^{1:t}, A^{1:t-1}\right)$.

We decompose the inference problem as a variational inference problem at each timestep to approximate $p(H^t_{1:K} \mid x^t, a^{t-1}, h^{t-1}_{1:K})$, with an *action-conditioned* approximating distribution $q(H^t_{1:K} \mid x^t, a^{t-1}, h^{t-1}_{1:K})$ computed as follows: we first use our dynamics model to produce $\hat{h}^t_{1:K}$ as $\mathcal{D}(H^t_{1:K}|h^{t-1}_{1:K}, a^{t-1})$, and then use the recognition model from IODINE to compute the approximating distribution as $\mathcal{Q}(H^t_{1:K} \mid x^t, \hat{h}^t_{1:K})$. We apply multiple steps of amortized iterative inference every timestep to estimate $h^t_{1:K}$. We provide a derivation for the variational bound for the interactive inference algorithm in Appx. 3.2.1. We can train the entire OP3 system end-to-end by backpropagating

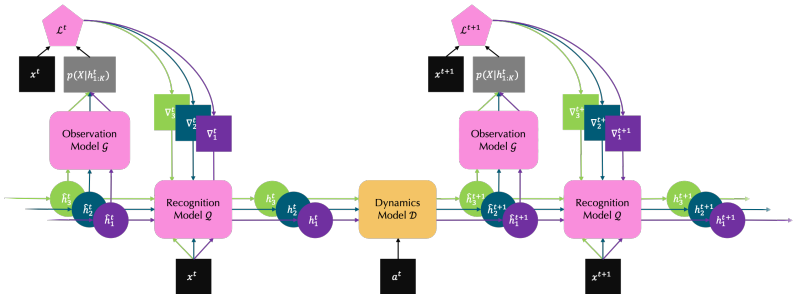

Figure 4: **Amortized interactive inference** alternates between perception (pink) and dynamics (orange) steps, iteratively updating the belief on the values of $H_{1:K}$ over time. Observed nodes are shaded in black. The pink functions that transform $\hat{H}^t_{1:K}$ to $H^t_{1:K}$ are equivalent to a single step of IODINE's iterative inference.

through the entire inference procedure, using the ELBO at every timestep as a training signal for the parameters of $\hat{\mathcal{G}}$, $\hat{\mathcal{D}}$, $\hat{\mathcal{Q}}$ in a similar manner as [27].

1.3 Planning

With grounded object-level representations, we can now perform prediction and planning with visual model-predictive control [7], summarized in Appx. Figure 10. The algorithm samples many sequences of actions, scores the predicted states from applying each action sequence with a cost function (Appx. 3.4), and chooses the first action of the best action sequence to take in the environment. This procedure repeats for a fixed number of steps or until the goal is achieved.

2 Experiments

Our experiments aim to study how well our model can (1) learn object representations from images without any supervision, (2) generalize to novel combinations of objects and tasks, (3) perform multi-step planning for object manipulation tasks, and (4) infer objects and predict action outcome for perceptually complex scenes.

2.1 Combinatorial Generalization without Object Supervision

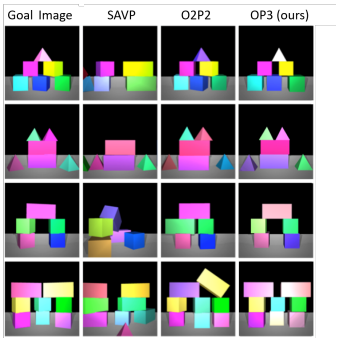

Figure 5: Respective results of our method in comparison to prior work. OP3 is able to plan at nearly the same level as O2P2 even though it does not have access to any object segmentations.

SAVP	O2P2	OP3 (ours)
24%	76%	**82%**

Table 1: Accuracy (%) of block tower builds by the SAVP baseline, the O2P2 oracle, and our approach. O2P2 uses image segmentations whereas OP3 uses only raw images as input.

Unfactorized	MLP Dynamics	OP3 (base)
7.3%	2.7%	**70%**

Table 2: Ablation showing the benefits of factorization. OP3 uses K=7 slots and a pair-wise dynamics model.

We first study how well OP3 can learn object-based representations without additional object supervision, as well as how well OP3's factorized representation can enable combinatorial generalization. We use the MuJoCo block stacking task introduced by Janner et al. [14] for the O2P2 model. This prior work demonstrates that an object-centric model that receives *ground truth* object segmentations can build varied structures out of blocks. We use the same training dataset of 60,000 trajectories of randomly dropped blocks, each showing the before and after image of a block being dropped. While the training set contains up to 5 objects, the test set contains up to 9 objects, placed in specific structures (bridge, pyramid, etc.) not seen during training. The actions are optimized using the cross-entropy method (CEM) [25].

Our two baselines, SAVP and O2P2, represent the state of the art in video prediction and object factorized planning methods, respectively. Our method obtains similar qualitative and quantitative performance to O2P2 without using object segmentations.

Furthermore our model learns to map the hidden states to objects in the scene without supervision, and can generalize to new permutations (e.g., configuration, height, colors, more objects) than seen in the training distribution. The appendix contains visualizations of OP3 factorizing the scene into individual object components (Fig. 11, 12). We additionally show how an unfactorized ablation of our model as a whole, and even an unfactorized ablation of just the dynamics model, lead to a severe degregation in performance in Table 2.

# Blocks	SAVP	OP3 (ours)
1	54%	**73%**
2	28%	**55%**
3	28%	**41%**

Table 3: Accuracy (%) of multi-step planning for building block towers by the SAVP baseline and our approach.

2.2 Multi-Step Planning

The goal of our second experiment is to understand how well OP3 can perform multi-step planning on objects already present in the scene. We modify the block stacking to require our model to reason over temporally extended action sequences, by changing the action space to represent a picking and dropping location. A pick only succeeds if it is within some distance of the center of a block. Goals are specified with a goal image, with the initial scene containing all the blocks needed to build the desired structure. This task is more difficult because the agent may have to move blocks out of the way before placing others which would require multi-step planning. We again optimize the actions using CEM, optimizing over multiple consecutive actions into the future, executing the first action in the sequence with lowest cost, and replanning at each time step.

We train on a dataset consisting of random picking and placing actions, in scenes with two blocks. The test dataset contains specific structures consisting of up to three blocks. We plan two steps into the future for the two block and three block goal environments. We evaluate the accuracy of our method in comparison to a non-object based approach, SAVP. Table 3 shows that while SAVP does well on two blocks, our method outperforms the non-object based approach on three blocks. This suggests an object-based approach can generalize better to more objects than seen during training.

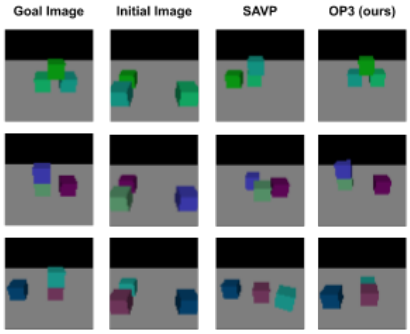

Figure 6: Comparison of our method against SAVP with end result shown.

2.3 Real World Evaluation

In this section, we study how well our method scales to real world data with perceptually complex objects. We evaluate OP3 on the Cloth dataset from Ebert et al. [5] which contains videos of a robotic arm moving cloths and other objects around. This dataset contain occlusions and include deformable and multipart objects with varying textures.

We evaluate qualitative performance by visualizing the object segmentations and compare against vanilla IODINE, which does not incorporate a dynamics model into the inference process. While the predictions are blurry we see that OP3 can obtain reasonable segmentations and in some cases performs better than IODINE as shown in Figure 7.

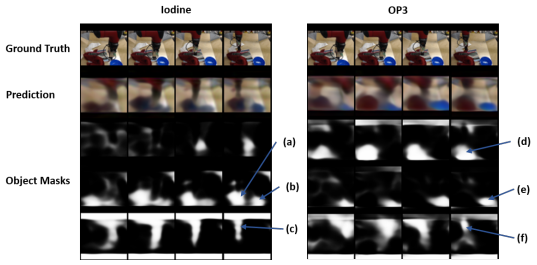

Figure 7: Qualitative results on learning object representations on real world data. We show the learned object masks for IODINE and our method. While Iodine represents two different objects with the same mask (a and b), OP3 is able to separate them into different masks (d and e).

3 Discussion

We have introduced a framework for model-based reinforcement learning that predicts and plans with inferred object representations from raw visual input. Our key idea is using *object abstraction*, with which we model a factorized POMDP by symmetrically processing a set of hidden states. By defining models *locally* on entities rather than globally on scenes we achieve almost three times the accuracy of a SOTA video prediction model, and perform comparably to an oracle model that uses object segmentations. We have presented an *interactive inference* algorithm with which we have shown multi-step planning with object representations grounded in the physical scene, on a set of block stacking tasks. We hope this work motivates future research in grounding object representations from visual input.

References

[1] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016.

[2] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv:1612.00341*, 2016.

[3] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[4] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in neural information processing systems*, pages 4414–4423, 2017.

[5] Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. *arXiv:1810.03043*, 2018.

[6] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv:1812.00568*, 2018.

[7] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017.

[8] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016.

[9] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv:1511.07404*, 2015.

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[11] Klaus Greff, Raphael Lopez Kaufmann, Rishabh Kabra, Nicholas Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *arXiv:1903.00450*, 2019.

[12] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv:1811.04551*, 2018.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[14] Michael Janner, Sergey Levine, William T Freeman, Joshua B Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-oriented prediction and planning. *arXiv:1812.10972*, 2018.

[15] Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. *arXiv:1706.04317*, 2017.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[17] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *arXiv:1906.11883*, 2019.

[18] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015.

[19] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, 2018.

[20] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *arXiv:1603.01312*, 2016.

[21] Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. *arXiv:1807.09356*, 2018.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[23] Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *arXiv:1605.09128*, 2016.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *ArXiv*, abs/1211.5063, 2012.

[25] Reuven Y. Rubinstein and Dirk P. Kroese. The cross-entropy method. In *Information Science and Statistics*, 2004.

[26] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv:1706.01427*, 2017.

[27] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv:1802.10353*, 2018.

[28] William F Whitney, Michael Chang, Tejas Kulkarni, and Joshua B Tenenbaum. Understanding visual concepts with continuation learning. *arXiv:1602.06822*, 2016.

[29] Nevan Wichers, Ruben Villegas, Dumitru Erhan, and Honglak Lee. Hierarchical long-term video prediction without supervision. *arXiv:1806.04768*, 2018.

[30] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems*, pages 153–164, 2017.

[31] Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Unsupervised discovery of parts, structure, and dynamics. *arXiv:1903.05136*, 2019.

[32] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. 2018.

[33] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J Johnson, and Sergey Levine. Solar: Deep structured latent representations for model-based reinforcement learning. *arXiv:1808.09105*, 2018.

Supplementary Material

3.1 Dynamics Model

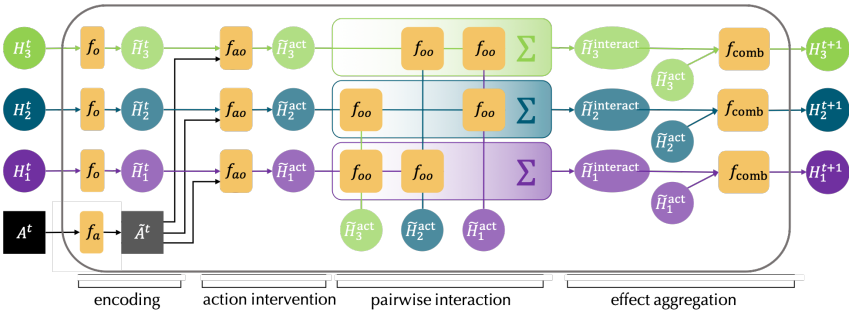

Figure 8: The **dynamics model** $\hat{\mathcal{D}}$ models the time evolution of every object by symmetrically applying the function $\hat{d}$ to each object. For a given object, $\hat{d}$ models the individual dynamics of that object (f_o), embeds the action vector (f_a), computes the action's effect on that object (f_{ao}), computes each of the other objects' effect on that object (f_{oo}), and aggregates these effects together (f_{comb}).

The dynamics model computes pairwise interactions between the hidden states and the effect of an action on each hidden state. The dynamics model is comprised of the functions:

$$\tilde{H}_k = f_o(H_k^t) \qquad \tilde{A} = f_a(A^t) \qquad \tilde{H}_k^{\text{act}} = f_{ao}(\tilde{H}_k \tilde{A})$$
$$H_k^{\text{interact}} = \sum_{i \neq k}^{K} f_{oo}(\tilde{H}_i^{\text{act}}, \tilde{H}_k^{\text{act}}) \qquad H_k^{t+1} = f_{\text{comb}}(\tilde{H}_k^{\text{act}}, H_k^{\text{interact}}),$$

where for a given object k, $f_{ao}(\tilde{H}_k \tilde{A}) := f_{\text{act-eff}}(\tilde{H}_k, \tilde{A}) \cdot f_{\text{act-att}}(\tilde{H}_k, \tilde{A})$ computes how $(f_{\text{act-eff}})$ and to what degree $(f_{\text{act-att}})$ an action affects the object and $f_{oo}(\tilde{H}_i^{\text{act}}, \tilde{H}_k^{\text{act}}) := f_{\text{obj-eff}}(\tilde{H}_i^{\text{act}}, \tilde{H}_k^{\text{act}}) \cdot f_{\text{obj-att}}(\tilde{H}_i^{\text{act}}, \tilde{H}_k^{\text{act}})$ computes how $(f_{\text{obj-eff}})$ and to what degree $(f_{\text{obj-att}})$ other objects affect that object. $f_{\text{obj-eff}}$ and $f_{\text{obj-att}}$ are shared across all object pairs. The other functions are shared across all objects.

3.2 Inference

The joint distribution of the hidden object states $H_{1:K}^{1:T}$ and observations $X^{1:T}$ given actions $A^{1:T-1}$ is given as

$$p\left(X^{1:T}, H_{1:K}^{1:T} \mid A^{1:T-1}\right) = p\left(H_{1:K}^{1}\right) p\left(X^{1} \mid H_{1:K}^{1}\right) \prod_{t=2}^{T} p\left(H_{1:K}^{t} \mid H_{1:K}^{t-1}, A^{t-1}\right). \tag{1}$$

Consider the problem of inferring the posterior distribution $p\left(H_{1:K}^{1:T} | x^{1:T}, a^{1:T-1}\right)$ of the hidden object states $H_{1:K}^{1:T}$ given observations $x^{1:T}$ and actions $a^{1:T-1}$. In general this posterior is intractable to compute because it involves marginalizing over all values of $H_{1:K}^{1:T}$ so we approximate the posterior as the solution to a variational inference problem. In Section 3.2.1 we set up the variational inference objective and in Figure 9 we present the interactive inference algorithm for optimizing the objective.

3.2.1 Amortized Interactive Variational Inference

We approximate the posterior as the solution to a variational inference problem that minimizes the Kullback-Leibler (KL) divergence

$$KL\big(q\left(H_{1:K}^{1:T} | x^{1:T}, a^{1:T-1}\right) \parallel p\left(H_{1:K}^{1:T} | x^{1:T}, a^{1:T-1}\right)\big) \tag{2}$$

between a variational distribution $q\left(H_{1:K}^{1:T} | x^{1:T}, a^{1:T-1}\right)$ and the true posterior. As we do not have access to the true posterior, we can equivalently optimize the KL by maximizing the evidence lower bound

$$\mathbb{E}_{h_{1:K}^{1:T} \sim q\left(H_{1:K}^{1:T} | x^{1:T}, a^{1:T-1}\right)} \left[\log p\left(x^{1:T} \mid h_{1:K}^{1:T}\right)\right] - KL\big(q\left(H_{1:K}^{1:T} \mid x^{1:T}, a^{1:T-1}\right) \parallel p\left(H_{1:K}^{1:T} \mid a^{1:T}\right)\big) \tag{3}$$

Algorithm 1 Amortized Variational Interactive Inference

Input: observations $x^{1:T}$, actions $a^{1:T-1}$, hyperparameters T, N
Input: trainable parameters $\hat{h}^{1,1}, \phi_{\mathcal{G}}, \phi_{\mathcal{D}}, \phi_{\mathcal{Q}}$
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do** ▷ N steps of iterative inference
    $p(X|\hat{h}^{t,i}_{1:K}) \leftarrow \mathcal{G}(\hat{h}^{t,i}_{1:K})$
    $\mathcal{L}^t \leftarrow KL\Big(q(\hat{H}^t_{1:K}|x^t, \hat{h}^{t,i}_{1:K}) \parallel p(\hat{h}^{t,i}_{1:K})\Big) - \log p(x^t|\hat{h}^{t,i}_{1:K})$
    $\nabla^{t,i}_{1:K} \leftarrow \text{feedback}(x^t, \hat{h}^{t,i}_{1:K}, \mathcal{L}^t)$
    $H^{t,i+1}_{1:K} \leftarrow \mathcal{Q}(x^t, \hat{h}^{t,i}_{1:K}, \nabla^{t,i}_{1:K})$
  end for
  $\hat{h}^{t+1,1}_{1:K} \leftarrow \mathcal{D}(h^{t,N}_{1:K}, a^t)$
end for

Figure 9: Given object states $h^t_{1:K}$ and actions a^t, the dynamics model predicts the next state $\hat{h}^{t+1}_{1:K}$ from which the observation model predicts the observation $\hat{x}^{t+1}$. Given a new observation x^{t+1}, feedback $\gamma^t_{1:K}$ from the ELBO $\mathcal{L}^t$ updates the belief of the state to $h^{t+1}_{1:K}$. The observation, dynamics, and recognition models are parameterized by $\phi_{\hat{\mathcal{G}}}, \phi_{\hat{\mathcal{D}}}, \phi_{\hat{\mathcal{Q}}}$.

with respect to q, where we simplified $p\left(x^{1:T} \mid h^{1:T}_{1:K}, a^{1:T-1}\right)$ to $p\left(x^{1:T} \mid h^{1:T}_{1:K}\right)$ because $x^{1:T}$ and $a^{1:T-1}$ are conditionally independent given $h^{1:T}_{1:K}$. Let us factorize the variational distribution as

$$q\left(H^{1:T}_{1:K} \mid x^{1:T}, a^{1:T}\right) = \prod_{t=1}^{T} q\left(H^t_{1:K} \mid x^{1:t}, a^{1:t-1}\right). \tag{4}$$

With this factorization, we can use the linearity of expectation to decouple equation 3 as a variational inference problem at every timestep. At the first timestep we have

$$\mathbb{E}_{h^1_{1:K} \sim q\left(H^1_{1:K} \mid x^1\right)}\left[\log p\left(x^1 \mid h^1_{1:K}\right)\right] - KL\big(q\left(H^1_{1:K} \mid x^1\right) \parallel p\left(H^1_{1:K}\right)\big) \tag{5}$$

and in subsequent timesteps t we have

$$\begin{aligned}&\mathbb{E}_{h^t_{1:K} \sim q\left(H^t_{1:K} \mid x^{1:t}, a^{1:t-1}\right)}\left[\log p\left(x^t | h^t_{1:K}\right)\right] \hookleftarrow \\ -&\mathbb{E}_{h^{t-1}_{1:K} \sim q\left(H^{t-1}_{1:K} \mid x^{1:t-1}, a^{1:t-2}\right)}\left[KL\big(q\left(H^t_{1:K} \mid x^{1:t}, a^{1:t-1}\right) \parallel p\left(H^t_{1:K} \mid h^{t-1}, a^{t-1}\right)\big)\right].\end{aligned} \tag{6}$$

Note that the objective at timestep 1 is equivalent to amortized variational inference on a static image and is the objective in IODINE. We outline the algorithm in detail in Figure 9.

3.3 Planning

We use visual model predictive control for planning. The planning algorithm and action scoring algorithm are outlined in Figure 10. The actions are optimized using the cross-entropy method (CEM), with each sampled action evaluated by the model using the action-scoring algorithm. CEM begins from a uniform distribution on the first iteration, uses a population size of 1000 samples per iteration, and uses 10% of the best samples to fit a Gaussian distribution for each successive iteration.

3.4 Cost Function

Building good cost functions for image-based control is generally difficult without access to the underlying state [6]. Our learned representations can alleviate this problem, since they better reflect the underlying state of objects in the scene. In our evaluation, we study tasks that require arranging physical objects into goal configurations (Fig. 11). Our cost function will compute some measure of distance between the inferred hidden states of the goal image $h^G_{1:k}$ with the predicted hidden states $h^P_{1:k}$.

We use two different cost functions. If we are performing single-step greedy planning, then we assume a single action will achieve one of the goal states, and can remove the matching goal state

Algorithm 2 OBJECT-CENTRIC-PLANNING

Input: $x^{1:\tau}$, $a^{1:\tau-1}$, x^G; **Output:** $a^{\tau:\tau+T}$
$h^G_{1:K} \leftarrow$ GOAL INFERENCE(x^G)
$h^\tau_{1:K} \leftarrow$ STATE ACQUISITION($x^{1:\tau}$, $a^{1:\tau-1}$)
for $t \leftarrow \tau$ **to** T **do**
    $a^t \leftarrow$ ACTION SELECTION($h^{\tau+t}_{1:K}$, h^G_K)
    $x^{\tau+t+1} \leftarrow$ ENVIRONMENT-STEP(a^t)
    $\hat{h}^{\tau+t+1}_{1:K} \leftarrow \mathcal{D}(h^{\tau+t}_{1:K}, a^t)$
    $h^{\tau+t+1}_{1:K} \leftarrow \mathcal{Q}(\hat{h}^{\tau+t}_{1:K}, x^{\tau+t+1})$
end for
return $a^{\tau:T}$

Algorithm 3 ACTION-SCORING

Input: $h^t_{1:K}$, $h^G_{1:K}$, $a^{t:T-1}$; **Output:** c
for $t' \leftarrow t$ **to** T-1 **do**
    $h^{t'+1}_{1:K} \leftarrow \mathcal{D}(h^{t'}_{1:K}, a^{t'})$
end for
score $c = \mathcal{C}(\hat{h}^T_{1:K}, h^G_{1:K})$
return c

Figure 10: GOAL INFERENCE: estimate the goal hidden states $h^G_{1:K}$ with IODINE. STATE ACQUISTION: estimate the hidden states $h^\tau_{1:K}$ of the current scene with amortized interactive inference over τ *seed steps*.

from future comparison. Specifically, we compute the distances over all pairs of goal and predicted hidden states and find the pair with minimum distance. This is defined as

$$\mathcal{C}(h^G_{1:k}, h^P_{1:k}) = \min_{a \in K', b \in K} D(h^G_a, h^P_b), \tag{7}$$

where K' and K are the set of hidden states in the goal and predicted image respectively. The specifics of what we use for D are explained in the appendix. Once an action is chosen, the hidden state $\text{argmin}_{a \in K'}$ in $H^G_{K'}$ is removed from the goal states K', representing how the action has succeeded in placing a particular object at that goal state and no longer needs to be compared to in the cost. While this works for greedy planning, it may not work in general for multi-step planning. We instead sum over the distances of each matching pair and no longer prune the set of goal states using $\mathcal{C}(h^G_{1:k}, h^P_{1:k}) = \sum_{a \in K'} \min_{b \in K} D(h^G_a, h^P_b)$.

To compute the cost we use a distance function between hidden states, $D(H_a, H_b)$. For the first environment with single-step planning we use L2 distance of the corresponding subimages. $D(H_a, H_b) = L_2(I(H_a), I(H_b))$ where the masked sub-image of an object is the mask times the pixel means $I(H_k) = m_{ij}(H_k) \cdot \hat{g}(H_k)_{(ij)}$. For the second environment with multi-step planning we a different distance function since the previous one may care more about if a shape matches than if the color matches. We instead use a form of intersection over union but that counts intersection if the mask aligns and pixel color values are close $D(H_a, H_b) = 1 - \frac{\sum_{i,j} m_{ij}(H_a) > 0.01 \text{ and } m_{ij}(H_b) > 0.01 \text{ and } L2(\hat{g}(H_a)_{(ij)}, \hat{g}(H_b)_{(ij)}) < 0.1}{\sum_{i,j} m_{ij}(H_a) > 0.01 \text{ or } m_{ij}(H_b) > 0.01}$. We found this version to work better since it will not give low cost to moving a wrong color block to the position of a different color goal block.

3.5 Model and Hyperparameter Details

We use similar model architectures as in [11] and so have rewritten some details from their appendix here. Differences include the dynamics model, inclusion of actions, and training procedure over sequences of data. The posterior distribution $p(h|x)$ is a diagonal Gaussian. The the output distribution $p(x|h)$ is also a diagonal Gaussian with means μ and global scale $\sigma = 0.1$. The decoder outputs the means μ and mask m_k.

Training All models are trained with the ADAM optimizer [16] with default parameters and a learning rate of 0.0001. We use gradient clipping as in [24] where if the norm of global gradient exceeds 5.0 then the gradient is scaled down to that norm.

Inputs For all models, we use the following inputs to the refinement network, where LN means Layernorm and SG means stop gradients. The following image-sized inputs are concatenated and fed to the corresponding convolutional network:

Description	Formula	LN	SG	Ch.
image	$\boldsymbol{x}$			3
means	$\boldsymbol{\mu}$			3
mask	$\boldsymbol{m}_k$			1
mask-logits	$\hat{\boldsymbol{m}}_k$			1
mask posterior	$p(\boldsymbol{m}_k \vert \boldsymbol{x},)$			1
gradient of means	$\nabla_k \mathcal{L}$	✓	✓	3
gradient of mask	$\nabla_{\boldsymbol{m}_k} \mathcal{L}$	✓	✓	1
pixelwise likelihood	$p(\boldsymbol{x} \vert \boldsymbol{h})$	✓	✓	1
leave-one-out likelih.	$p(\boldsymbol{x} \vert \boldsymbol{h}_{i \neq k})$	✓	✓	1
coordinate channels				2
			total:	17

The posterior parameters $\boldsymbol{h}$ and their gradients are flat vectors, and we concatenate them with the output of the convolutional part of the refinement network and use the result as input to the refinement LSTM:

Description	Formula	LN	SG
gradient of posterior	$\nabla_{\boldsymbol{h}_k} \mathcal{L}$	✓	✓
posterior	$\boldsymbol{h}_k$		

3.5.1 Architecture

All models use the ELU activation function and the Convolutional layers use a stride equal to 1 unless otherwise noted.

Observation Model Decoder

Type	Size/*Ch.*	Act. Func.	Comment
Input: H_i	128		
Broadcast	*130*		+ coordinates
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Conv 3×3	*4*	Linear	RGB + Mask

Refinement Network

Type	Size/*Ch.*	Act. Func.	Comment
MLP	128	Linear	
LSTM	256	Tanh	
Concat $[H_i, \nabla_{H_i}]$	512		
MLP	256	ELU	
Avg. Pool	64		
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Conv 3×3	*64*	ELU	
Inputs	*17*		

Dynamics Network The posterior distribution is represented as a diagonal Gaussian and so we choose the dynamics network to only operate on the mean parameters of this distribution. We carry over the variance parameters from the previous step to the next step. All models including the final layer uses ELU activations unless otherwise stated. MLP(128) would denote a multilayer peceptron with a hidden layer of size 128.

- f_o: This inputs H_i^t of size 128, is a MLP(128), and outputs $\tilde{H}_i$ of size 128.
- $f_{\text{obj-eff}}$: This inputs the concatenated pair of $\tilde{H}_i, \tilde{H}_j$ each of size 128, is a MLP(256), and outputs features of size 128.
- $f_{\text{obj-att}}$: This inputs the concatenated $\tilde{H}_i, \tilde{H}_j$ each of size 128, is a MLP(256), and applies a sigmoid activation to output a single attention scalar of size 1.
- f_{comb}: This inputs the concatenated $\tilde{H}_i, \tilde{H}_i^{\text{eff}}$ each of size 128, is a MLP(256), and applies an identity activation to output H_i^{t+1}.

3.5.2 Single-Step Block-Stacking

The training dataset has 60,000 trajectories each containing before and after images of size 64x64 from [14]. Factorized OP3 models were trained on scenes with 1 to 5 blocks with $K = 7$ slots. The unfactorized OP3 ablation is naturally trained with $K = 1$ slots, while the unfactorized dynamics ablation replaces the regular dynamics network with an MLP whose input and output are the concatenated latents. We evaluate OP3 on the same set of 110 goal images as Janner et al. [14].

3.5.3 Multi-Step Block-Stacking

The training dataset has 10,000 trajectories each from a separate environment with two different colored blocks. Each trajectory contains five frames (64x64) of randomly picking and placing blocks. We bias the dataset such that 30% of actions will pick up a block and place it somewhere randomly, 40% of actions will pick up a block and place it on top of a another random block, and 30% of actions contain random pick and place locations. Models were trained with K = 5 slots.

4 Additional Experimental Evaluation

Figure 11 shows a visualization of the planning procedure that OP3 executes on single-step block stacking, included the predicted images that OP3 generates. Given an action image, which depicts an intervention on the scene (a block raised in the air), OP3 predicts the steady state result from dropping the block. The figure also shows that OP3 makes predictions that reasonably model the true dynamics of the environment.

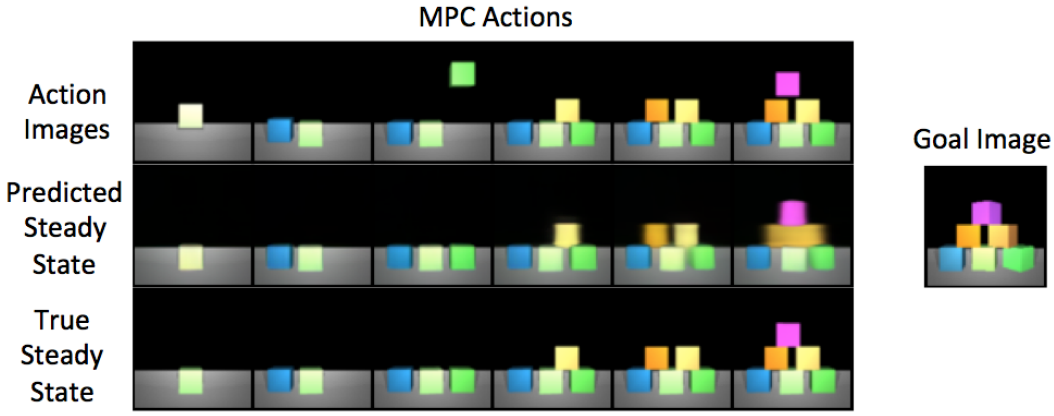

Figure 11: Qualitative results on building a structure. We see how our method is able to accurately and consistently predict the outcome of the action image, successively capturing the effect of inertial dynamics (gravity) and interactions with other objects.

Figure 12 shows a demonstration of the execution of the inference procedure on single-step block stacking. OP3 learns to split the objects in the scene into several hidden entity-representations, which are decoded into the object sub-images shown. When OP3 predicts how the yellow block falls in the figure, notice that only the sub-image depicting the yellow block changes while the other sub-images remain unperturbed, showcasing that modeling object separately may provide a benefit for isolating the relevant variables to a prediction from the irrelevant ones.

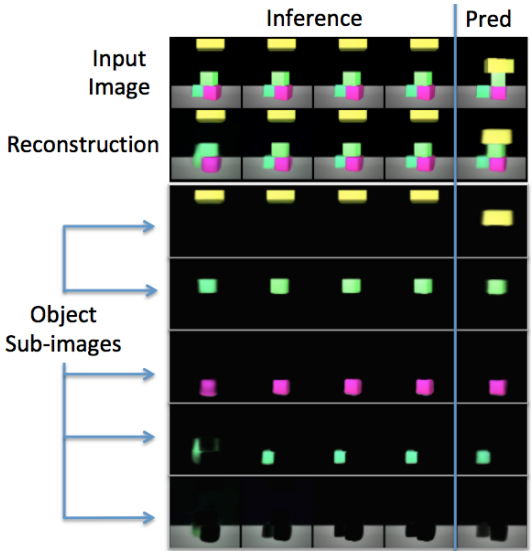

Figure 12: We show a demonstration of a rollout. The first four columns show inference iterations on the single input image, while the last column shows the predicted results using the dynamics module on the learnt hidden states (top right image is not given as input and shows the true outcome). The bottom 5 rows show $I(H_i)$ at each iteration, demonstrating how the model is able to capture individual objects, and the dynamics afterwards.

Figure 13 shows several plans executed by OP3, starting from an initial scene of three blocks. Note that OP3 was trained only on modeling 1 or 2 blocks, so modeling three blocks requires OP3 to extrapolate its knowledge of object interactions to a different number of objects from what it has been trained on.

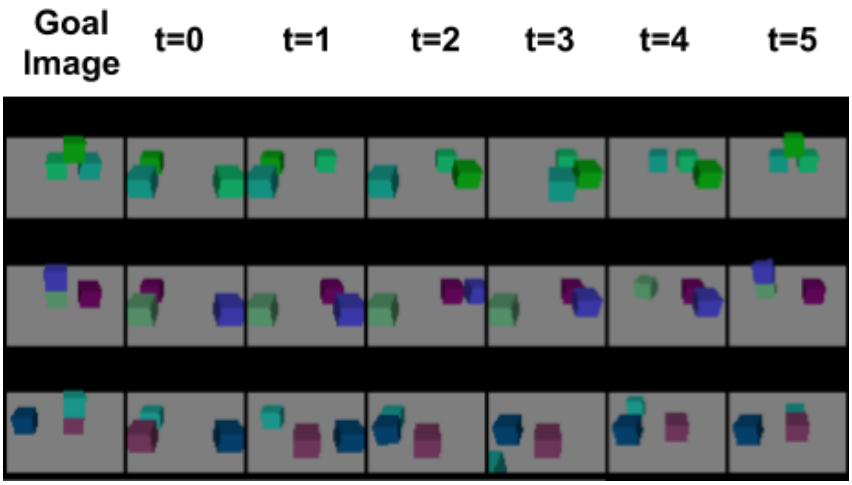

Figure 13: Demonstration of our method on several goals. $t = 0$ denotes the initial scene that must be reconfigured to match the goal image. $t = 1...5$ show the executed actions.