Object-Oriented Dynamics Learning through Multi-Level Abstraction

Guangxiang Zhu*, Jianhao Wang*, Zhizhou Ren*, and Chongjie Zhang
Institute for Interdisciplinary Information Sciences
Tsinghua University, Beijing, China
{zhu-gx15,jh-wang15,rzz16}@mails.tsinghua.edu.cn,chongjie@tsinghua.edu.cn

Abstract

Object-based approaches for learning action-conditioned dynamics has demonstrated promise for generalization and interpretability. However, existing approaches suffer from structural limitations and optimization difficulties for common environments with multiple dynamic objects. In this paper, we present a novel self-supervised learning framework, called *Multi-level Abstraction Object-oriented Predictor* (MAOP), for learning object-based dynamics models from raw visual observations. MAOP employs a three-level learning architecture that enables efficient dynamics learning for complex environments with a dynamic background. We also design a spatial-temporal relational reasoning mechanism to support instance-level dynamics learning and handle partial observability. Empirical results show that MAOP significantly outperforms previous methods in terms of sample efficiency and generalization over novel environments that have multiple controllable and uncontrollable dynamic objects and different static object layouts. In addition, MAOP learns semantically and visually interpretable disentangled representations.

1 Introduction

Model-based deep reinforcement learning (DRL) has recently attracted much attention for improving sample efficiency of DRL, such as [1, 2]. One of the core problems for model-based reinforcement learning is to learn action-conditioned dynamics models through interacting with environments. Pixel-based approaches have been proposed for such dynamics learning from raw visual perception, achieving remarkable performance in training environments [3].

To unlock sample efficiency of model-based DRL, learning action-conditioned dynamics models that generalize over unseen environments is critical yet challenging. Finn et al. [4] proposed an action-conditioned video prediction method that explicitly models pixel motion and thus is partially invariant to object appearances. Zhu et al. [5] developed an object-oriented dynamics predictor, taking a further step towards generalization over unseen environments with different object layouts. However, due to structural limitations and optimization difficulties, these methods do not learn and generalize well for common environments with a dynamic background, which contain multiple moving objects in addition to controllable objects.

To address these limitations, this paper presents a novel self-supervised, object-oriented dynamics learning framework, called *Multi-level Abstraction Object-oriented Predictor* (MAOP). This framework simultaneously learns disentangled object representations and predicts object motions conditioned on their historical states, their interactions to other objects, and an agent's actions. To reduce the complexity of such concurrent learning and improve sample efficiency, MAOP employs a three-level learning architecture from the most abstract level of motion detection, to dynamic instance segmentation, and to dynamics learning and prediction. A more abstract learning level solves an easier problem and has lower learning complexity, and its output provides a coarse-grained guidance

* indicates equal contribution.

for the less abstract learning level, improving its speed and quality of learning convergence. This multi-level architecture is inspired by humans' multi-level motion perception from cognitive science studies [6, 7] and multi-level abstraction search in constraint optimization [8]. In addition, we design a novel CNN-based spatial-temporal relational reasoning mechanism, which includes a Relation Net to reason about spatial relations between objects and an Inertia Net to learn temporal effects. This mechanism offers a disentangled way to handle physical reasoning in the setting with partial observability.

Empirical results show that MAOP significantly outperforms previous methods in terms of sample efficiency and generalization over novel environments that have multiple controllable and uncontrollable dynamic objects and different object layouts. It can learn from few examples and accurately predict the dynamics of objects as well as raw visual observations in previously unseen environments. In addition, MAOP learns disentangled representations and gains semantically and visually interpretable knowledge, including meaningful object masks, accurate object motions, disentangled reasoning process, and the discovery of the controllable agent.

2 Multi-level Abstraction Object-oriented Predictor (MAOP)

In this section, we will present a novel self-supervised deep learning framework, aiming to learn object-oriented dynamics models that are able to generalize over unseen environments with different object layouts and multiple dynamic objects. Such an object-oriented dynamics learning approach requires simultaneously learning object representations and motions conditioned on their historical states, their interactions to other objects, and an agent's actions. This concurrent learning is very challenging for an end-to-end approach in complex environments with a dynamic background. Evidences from cognitive science studies [6, 7] show that human beings are born with prior motion perception ability (in Cortical area MT) of perceiving moving and motionlessness, which enables learning more complex knowledge, such as object-level dynamics prediction. Inspired by these studies, we design a multi-level learning framework, called *Multi-level Abstraction Object-oriented Predictor* (MAOP), which incorporates motion perception levels to assist in dynamics learning.

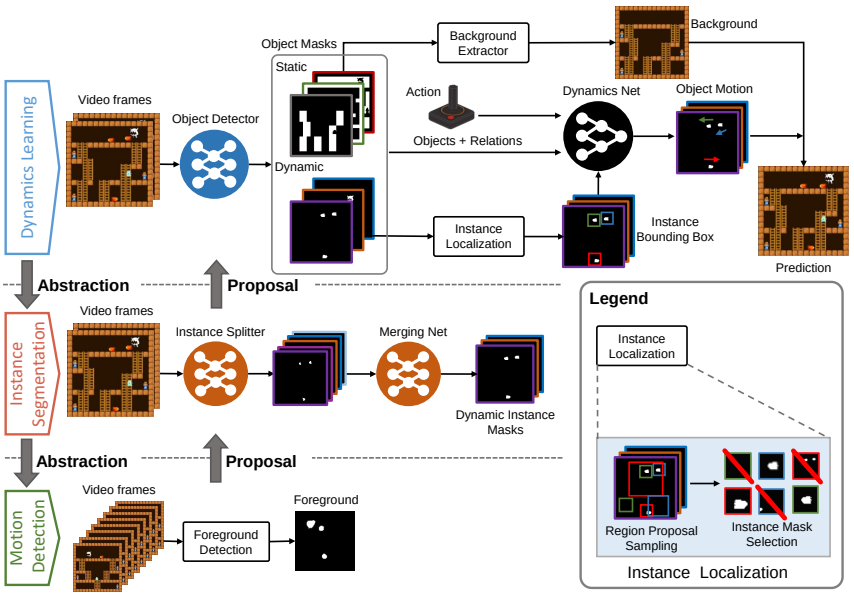

Figure 1: Multi-level abstraction framework from a top-down decomposition view. First, we perform motion detection to produce foreground masks. Then, we utilize the foreground masks as dynamic region proposals to guide the learning of dynamic instance segmentation. Finally, we use the learned dynamic instance segmentation networks (including Instance Splitter and Merging Net) as a guiding network to generate region proposals of dynamic instances and guide the learning of Object Detector in the level of dynamics learning.

Figure 1 illustrates three levels of MAOP framework: dynamics learning, dynamic instance segmentation, and motion detection. The dynamics learning level is an end-to-end, self-supervised neural network, aiming to learn object representations and instance-level dynamics, and predict the next visual observation conditioned on an agent's action. To guide the learning of the object representations and instance localization at the level of dynamics learning, the more abstracted level of dynamic instance segmentation learns a guiding network in a self-supervised manner, which can provide coarse dynamic instance mask proposals. It exploits the spatial-temporal information of locomotion property and appearance pattern to capture the region proposals of dynamic instances. To facilitate the learning of dynamic instance segmentation, MAOP employs the more coarse-grained level of motion detection, which detects changes in image sequences and provides guidance on proposing regions potentially containing dynamic instance. As the learning proceeds, the knowledge distilled from the more coarse-grained level are gradually refined at the more fine-grained level by considering additional information. When the training is finished, the coarse-grained levels of dynamic instance segmentation and motion detection will be removed at the testing stage. The details of the design of each level and their connections can be found in Appendix A.

3 Experiments

We compare MAOP with state-of-the-art action-conditioned dynamics learning baselines, AC Model [3], CDNA [4], and OODP [5] on two games, *Monster Kong* and *Flappy Bird*, from the Pygame Learning Environment, which allows us to test generalization ability over various scenes with different layouts. To test whether our model can truly learn the underlying physical mechanism behind the visual observations and perform relational reasoning, we set the k-to-m zero-shot generalization experiment, where we use k environments for training and other m unseen environments for testing.

As shown in Table 1, MAOP significantly outperforms other methods in all experiment settings in terms of generalization ability and sample efficiency of both object dynamics learning and image prediction. We also test our model on *Flappy Bird*, where we limit the training samples to 100 and 300 to form a sufficiently challenging generalization task. As shown in Table C2 (Appendix C), our performance is similar with that on *Monster Kong*. In Figure C4 (Appendix C), we plot the learning curve for better visualization of the comparison.

	Models	Training environments								Unseen environments							
		1-5†		1-5		2-5		3-5		1-5†		1-5		2-5		3-5	
		Agent	All	Agent	All	Agent	All	Agent	All	Agent	All	Agent	All	Agent	All	Agent	All
0-error accuracy	MAOP	0.67	0.80	0.88	0.87	0.86	0.87	0.80	0.83	**0.60**	**0.77**	**0.81**	**0.84**	**0.85**	**0.87**	**0.80**	**0.85**
	OODP	0.24	0.17	0.18	0.16	0.22	0.17	0.26	0.20	0.20	0.16	0.20	0.15	0.18	0.15	0.26	0.18
	AC Model	0.04	0.59	0.87	0.94	0.80	0.93	0.77	0.92	0.01	0.18	0.08	0.16	0.30	0.48	0.45	0.66
	CDNA	0.30	0.66	0.41	0.76	0.42	0.78	0.44	0.74	0.31	0.55	0.37	0.59	0.40	0.71	0.41	0.70
1-error accuracy	MAOP	0.90	0.91	0.97	0.94	0.97	0.93	0.96	0.93	**0.86**	**0.90**	**0.96**	**0.93**	**0.97**	**0.93**	**0.95**	**0.93**
	OODP	0.49	0.29	0.32	0.23	0.34	0.23	0.35	0.25	0.39	0.25	0.34	0.22	0.32	0.21	0.34	0.22
	AC Model	0.07	0.63	0.98	0.99	0.95	0.98	0.94	0.98	0.02	0.34	0.15	0.26	0.52	0.67	0.66	0.77
	CDNA	0.42	0.84	0.48	0.86	0.48	0.86	0.51	0.87	0.45	0.82	0.45	0.83	0.47	0.84	0.48	0.86
2-error accuracy	MAOP	0.95	0.94	0.99	0.96	0.99	0.95	0.98	0.94	**0.95**	**0.94**	**0.98**	**0.95**	**0.99**	**0.95**	**0.98**	**0.95**
	OODP	0.67	0.47	0.44	0.37	0.46	0.32	0.49	0.39	0.60	0.43	0.48	0.34	0.43	0.31	0.46	0.36
	AC Model	0.10	0.64	0.99	0.99	0.98	0.99	0.97	0.98	0.04	0.34	0.20	0.31	0.64	0.73	0.77	0.81
	CDNA	0.50	0.86	0.52	0.87	0.53	0.88	0.54	0.88	0.53	0.85	0.47	0.84	0.50	0.86	0.51	0.87
Object RMSE	MAOP	31.99		26.65		31.68		30.33		**34.14**		**29.78**		**31.32**		**30.80**	
	OODP	65.51		66.44		66.66		64.73		67.39		67.41		67.78		64.95	
	AC Model	62.02		18.88		22.39		21.30		85.46		57.41		55.45		43.48	
	CDNA	53.89		34.99		35.26		35.94		56.31		45.34		37.59		37.80	
Image RMSE	MAOP	6.90		5.64		6.68		6.46		**7.90**		**8.60**		8.73		**6.55**	
	OODP	14.70		15.08		14.89		14.42		15.42		24.68		26.39		14.52	
	AC Model	15.99		4.12		4.78		4.69		44.92		39.46		38.07		38.12	
	CDNA	11.47		7.41		7.58		7.68		12.23		9.87		**8.10**		8.16	

Table 1: Prediction performance on *Monster Kong*. k-m means the k-to-m generalization problem. † indicates training with only 1000 samples. ALL represents all dynamic objects.

MAOP takes a step towards interpretable deep learning and disentangled representation learning. We visualize the learned object masks in unseen environments to demonstrate the visual interpretability of MAOP. We highlight the attentions of the object masks by multiplying raw images by the binarized masks. As shown in Figure 2, MAOP captures all the key objects in the environments including the

controllable agents, the uncontrollable dynamic objects, and the static objects that have effects on the motions of dynamic objects. To further show the dynamical interpretability behind image prediction, we test our predicted motions by comparing RMSEs between the predicted and ground-truth motions in unseen environments (Table C3). Intriguingly, most predicted motions are quite accurate, with the RMSEs less than 1 pixel. Such a visually indistinguishable error also verifies our dynamics learning. With the learned knowledge in MAOP, we can easily uncover the action-controlled agent from all the dynamic objects because compared to other objects the action-controlled agent has the maximal variance of total effects over actions. In addition, with the learned knowledge in MAOP, we can easily uncover the action-controlled agent from all the dynamic objects because the action-controlled agent has the maximal variance of total effects over actions compared to other uncontrollable dynamic objects, as shown in Figure C5.

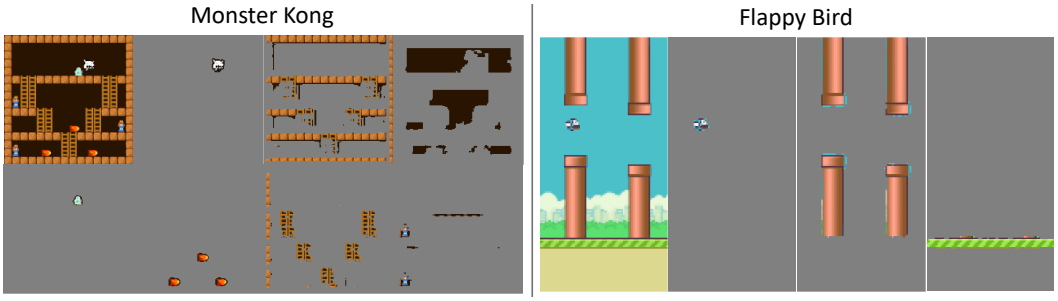

Figure 2: Visualization of the masked images in unseen environments. Left is the raw image.

4 Conclusions and Future Work

We present a self-supervised multi-level learning framework for learning action-conditioned object-based dynamics. This framework is example-efficient and generalizes object dynamics and prediction of raw visual observations to complex unseen environments with multiple dynamic objects. The learned dynamics model potentially enables an agent to directly plan or efficiently learn for unseen environments. Although a random policy or an expert's policy is used for exploration in our experiments, our framework can support smarter exploration strategies, e.g., curiosity-driven exploration. Our future work includes extending our model for deformation prediction (e.g., object appearing, disappearing and non-rigid deformation) and incorporating a camera motion prediction network module for applications such as FPS games and autonomous driving.

References

[1] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5694–5705, 2017.

[2] 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.

[3] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

[4] 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.

[5] Guangxiang Zhu, Zhiao Huang, and Chongjie Zhang. Object-oriented dynamics predictor. *Advances in Neural Information Processing Systems*, 2018.

[6] Gunnar Johansson. Visual motion perception. *Scientific American*, 232(6):76–89, 1975.

[7] Zhong-Lin Lu and George Sperling. The functional architecture of human visual motion perception. *Vision research*, 35(19):2697–2722, 1995.

[8] Chongjie Zhang and Julie A Shah. Co-optimizating multi-agent placement with task assignment and scheduling. In *IJCAI*, pages 3308–3314, 2016.

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[11] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[12] BPL Lo and SA Velastin. Automatic congestion detection system for underground platforms. In *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, pages 158–161. IEEE, 2001.

[13] Dar-Shyang Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):827–832, 2005.

[14] Lucia Maddalena, Alfredo Petrosino, et al. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168, 2008.

[15] Xiaowei Zhou, Can Yang, and Weichuan Yu. Moving object detection by detecting contiguous outliers in the low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):597–610, 2013.

[16] Xiaojie Guo, Xinggang Wang, Liang Yang, Xiaochun Cao, and Yi Ma. Robust foreground detection using smoothness and arbitrariness constraints. In *European Conference on Computer Vision*, pages 535–550. Springer, 2014.

[17] 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.

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Appendix A Details of Multi-level Abstraction Framework

In this section, we will describe in detail the design of each level of our multi-level abstraction framework and their connections.

A.1 Object-Oriented Dynamics Learning Level

The semantics of this level is formulated as learning an object-based dynamics model with the region proposals generated from the more abstracted level of dynamic instance segmentation. The whole architecture is shown in the top part of Figure 1, which is an end-to-end neural network and can be trained in a self-supervised manner. It takes a sequence of video frames and an agent's actions as input, learns the disentangled representations (including objects, relations and effects) and the dynamics of controllable and uncontrollable dynamic object instances conditioned on the actions and object relations, and produce the predictions of raw visual observations. The whole architecture includes four major components: A) an Object Detector that decomposes the input image into objects; B) an Instance Localization module responsible for localizing dynamic instances; C) a Dynamics Net for learning the motion of each dynamic instance conditioned on the effects from actions and object-level spatial-temporal relations; and D) a Background Extractor that computes the background image from the learned static object masks. Algorithm 1 illustrates the interactions of these components and the learning paradigm of object based dynamics, which is a general framework and agnostic to the concrete form of each component. In the following paragraphs, we will describe the detailed implementation of every components.

Algorithm 1 Basic paradigm of object-oriented dynamics learning.

Input: A sequence of video frames $\mathbf{I}^{t-h:t}$ with length h, input action $\boldsymbol{a}^t$ at time t.

Object masks $\mathbf{O}^{t-h:t} \leftarrow$ ObjectDetector($\mathbf{I}^{t-h:t}$), $\mathbf{O}$ include dynamic and static masks $\mathbf{D}, \mathbf{S}$
Instance masks $\mathbf{X}^{t-h:t} \leftarrow$ InstanceLocalization($\mathbf{I}^{t-h:t}, \mathbf{D}^{t-h:t}$)
Predicted instance masks $\hat{\mathbf{X}}^{t+1} \leftarrow \varnothing$
for each instance mask $\boldsymbol{x}$ in $\mathbf{X}$ **do**
    Effects from spatial relations $\boldsymbol{m}_1^t \leftarrow$ RelationNet($\boldsymbol{x}^t, \mathbf{O}^t, \boldsymbol{a}^t$)
    Effects from temporal relations $\boldsymbol{m}_2^t \leftarrow$ InertiaNet($\boldsymbol{x}^{t-h:t}, \boldsymbol{a}^t$)
    Total effects $\boldsymbol{m}^t \leftarrow \boldsymbol{m}_1^t + \boldsymbol{m}_2^t$
    Predicted instance mask $\hat{\boldsymbol{x}}^{t+1} \leftarrow$ Transformation($\boldsymbol{x}^t, \boldsymbol{m}^t$)
    $\hat{\mathbf{X}}^{t+1} \leftarrow \hat{\mathbf{X}}^{t+1} \bigcup \hat{\boldsymbol{x}}^{t+1}$
end for
Background image $\mathbf{B}^{t+1} \leftarrow$ BackgroundExtractor($\mathbf{I}^t, \mathbf{S}^t$)
Predicted next frame $\hat{\mathbf{I}}^{t+1} \leftarrow$ Merge($\hat{\mathbf{X}}^{t+1}, \mathbf{B}^{t+1}$)

Object Detector and Instance Localization Module. Object Detector is a CNN module aiming to learn object masks from input image. An object mask describes the spatial distribution of a class of objects, which forms the fundamental building block of our object-oriented framework. Considering that instances of the same class are likely to have different motions, we append an Instance Localization Module to Object Detector to localize each dynamic instance to support instance-level dynamics learning. The class-specific object masks in conjunction with instance localization build the bridge to connect visual perception (Object Detector) with dynamics learning (Dynamics Net), which allows learning objects based on both appearances and dynamics.

Specifically, Object Detector receives image $\mathbf{I}^t \in \mathbb{R}^{H \times W \times 3}$ at timestep t and then outputs object masks $\mathbf{O}^t \in [0,1]^{H \times W \times n_O}$, including dynamic object masks $\mathbf{D}^t \in [0,1]^{H \times W \times n_D}$ and static object masks $\mathbf{S}^t \in [0,1]^{H \times W \times n_S}$, where H and W denote the height and width of the input image, n_D and n_S denotes the maximum class number of dynamic and static objects respectively, and $n_O = n_D + n_S$. The entry $\mathbf{O}_{u,v,i}$ indicates the probability that the pixel $\mathbf{I}_{u,v,:}$ belongs to the i-th object class. Then, Instance Localization Module uses the dynamic object masks to compute each single instance mask $\mathbf{X}^t_{:,:,i} \in [0,1]^{H_M \times W_M} (1 \leq i \leq n_M)$, where H_M and W_M denote the height and width of the bounding box of this instance and n_M denotes the maximum number of localized instances. As shown in Figure 1, Instance Localization Module first samples a number of bounding boxes on the

dynamic object masks and then select the regions, each of which contains only one dynamic instance. As we focus on the motion of rigid objects, the affine transformation is approximatively consistent for all pixels of each dynamic instance mask. Inspired by this, we define a discrepancy loss $\mathcal{L}_{\text{instance}}$ for a sampled region that measures the motion consistence of its pixels and use it as a selection score for selecting instance masks. To compute this loss, we first compute an average rigid transformation of a sampled region between two time steps based on the instance masks and masked image in this region, then apply this transformation to this region at the previous time step, and finally compared this predicted region with the region at the current time. Obviously, when a sampled region contains exactly one dynamic instance, this loss will be extremely small, and even zero when the object masks are perfectly learned. More details of the region proposal sampling and instance mask selection can be found in Appendix B.

Dynamics Net. Dynamics Net is designed to learn instance-based motion effects of actions, object-to-object spatial relations (Relation Net) and temporal relations of spatial states (Inertia Net), and then reason about the motion of each dynamic instance based on these effects. Its architecture is illustrated in Figure A3, which has an Effect Net for each class of objects. An Effect Net consists of one Inertia Net and several Relation Nets. As shown in the left subfigure, instance-level dynamics learning is performed, which means the motion of each dynamic instance is individually computed. We take as an example the computation of the motion of the i-th instance $\mathbf{X}^t_{:,:,i}$ to show the detailed structure of the Effect Net.

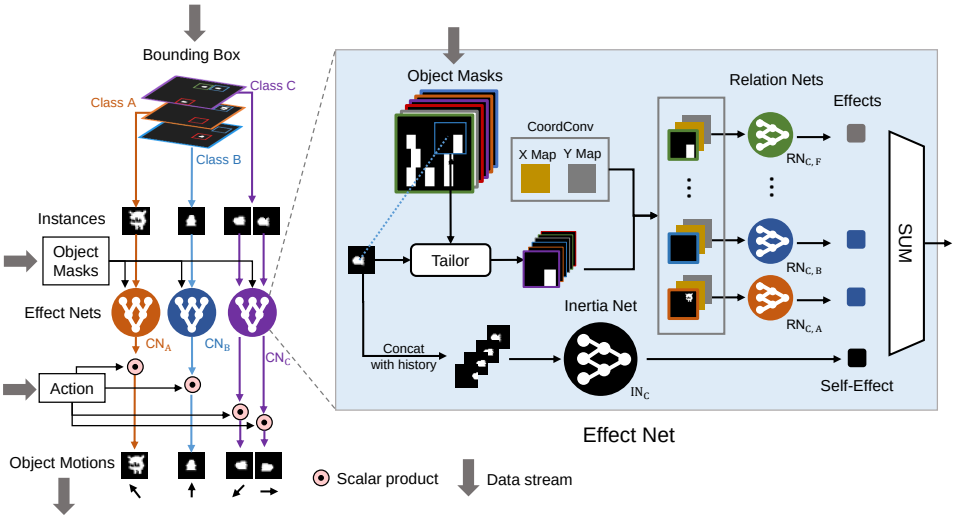

Figure A3: Architecture of Dynamics Net (left) and its component of Effect Net (right). Different classes are distinguished by different letters (e.g., A, B, ... , F).

As shown in the right subfigure of Figure A3, we first use a sub-differentiable tailor module introduced by Zhu et al. [5] to enable the inference of dynamics focusing on the relations with neighbour objects. This module crops a w-size "horizon" window from the concatenated masks of all objects $\mathbf{O}^t$ centered on the expected location of $\mathbf{X}^t_{:,:,i}$, where w denotes the maximum effective range of relations. Then, the cropped object masks are respectively concatenated with the constant x-coordinate and y-coordinate meshgrid map (to make networks more sensitive to the spatial information) and fed into the corresponding Relation Nets (RN) according to their classes. We use $\mathbf{C}^t_{:,:,i,j}$ to denote the cropped mask that crops the j-th object class $\mathbf{O}^t_{:,:,j}$ centered on the expected location of the i-th dynamic instance (the class it belongs to is denoted as $c_i, 1 \leq c_i \leq n_D$). The effect of object class j on class c_i, $E^t(c_i, j) \in \mathbb{R}^{2\times n_a}$ (n_a denotes the number of actions) is calculated as, $E^t(c_i, j) = \text{RN}_{c_i,j}\Big(\text{concat}\big(\mathbf{C}^t_{:,:,i,j}, \text{Xmap}, \text{Ymap}\big)\Big)$. Note that there are total $n_D \times n_O$ RNs for $n_D \times n_O$ pairs of object classes that share the same architecture but not their weights. To handle the partial observation problem, we add an Inertia Nets (IN) to learn the self-effects of an object class through historical states, $E^t_{\text{self}}(c_i) = \text{IN}_{c_i}\Big(\text{concat}\big(\mathbf{X}^t_{:,:,i}, \mathbf{X}^{t+1}_{:,:,i}, \ldots, \mathbf{X}^{t+h}_{:,:,i}\big)\Big)$, where h is the history length and there are total n_D INs for n_D dynamic object classes that share the same architecture but

not their weights. To predict the motion vector $\boldsymbol{m}_i^t \in \mathbb{R}^2$ for the i-th dynamic instance, all these effects are summed up and then multiplied by the one-hot coding of action $\boldsymbol{a}^t \in \{0,1\}^{n_a}$, that is, $\boldsymbol{m}_i^t = \Big(\big(\sum_j E^t(c_i, j) \big) + E^t_{\text{self}}(c_i) \Big) \cdot \boldsymbol{a}^t$.

Background Extractor. This module extracts the static background of input image based on the static object masks learned by Object Detector and then it is combined with the predicted dynamic instances to predict the next visual observation. As Object Detector can decompose its observation into objects in an unseen environment with a different object layout, Background Extractor is able to generate a corresponding static background and support the visual observation prediction in novel environments. Specifically, Background Extractor maintains an external background memory $\mathbf{B} \in \mathbb{R}^{H \times W \times 3}$ which is continuously updated (via moving average) by the static object mask learned by Object Detector. Denoting α as the decay rate, the updating formula is given by, $\mathbf{B}^t = \alpha \mathbf{B}^{t-1} + (1-\alpha)\mathbf{I}^t \sum_i \mathbf{S}^t_{:,:,i},\ \mathbf{B}^0 = 0$.

Prediction and Training Loss. At the output end of our model, the prediction of the next frame is produced by merging the learned object motions and the background $\mathbf{B}^t$. The pixels of a dynamic instance can be calculated by masking the raw image with the corresponding instance mask and we can use Spatial Transformer Network (STN) [9] to apply the learned instance motion vector $\boldsymbol{m}_i^t$ on these pixels. First, we transform all the dynamic instances according to the learned instance-level motions. Then, we merge all the transformed dynamic instances with the background image calculated from Background Extractor to generate the prediction of the next frame. In this paper, we use the pixel-wise l_2 loss to restrain image prediction error, denoted as $\mathcal{L}_{\text{prediction}}$. To get earlier feedback before reconstructing images and facilitate the training process, we add a highway loss, $\mathcal{L}_{\text{highway}} = \sum_i \left\| (\bar{u}_i, \bar{v}_i)^t + \boldsymbol{m}_i^t - (\bar{u}_i, \bar{v}_i)^{t+1} \right\|_2^2$, where $(\bar{u}_i, \bar{v}_i)^t$ is the excepted location of i-th instance mask $\mathbf{X}^t_{:,:,i}$. In addition, we add a proposal loss to utilize the dynamic instance proposals provided by the abstracted problem to guide our optimization, which is given by $\mathcal{L}_{\text{proposal}} = \left\| \sum_i (\mathbf{D}^t_{:,:,i} - \mathbf{P}^t_{:,:,i}) \right\|_2^2$, where $\mathbf{P}$ denotes the dynamic instance region proposals computed by the more abstract learning level (i.e., dynamic instance segmentation level). The total loss of the dynamics learning level is given by combining the previous losses with different weights,

$$\mathcal{L}_{\text{DL}} = \mathcal{L}_{\text{highway}} + \lambda_1 \mathcal{L}_{\text{prediction}} + \lambda_2 \mathcal{L}_{\text{proposal}}$$

A.2 Dynamic Instance Segmentation Level

This level aims to generate region proposals of dynamic instances to guide the learning of object masks and facilitate instance localization at the level of dynamics learning. The architecture is shown in Figure 1. Instance Splitter aims to identify regions, each of which potentially contains one dynamic instance. To learn to divide different dynamic object instances onto different masks, we use the discrepancy loss $\mathcal{L}_{\text{instance}}$ described in Section A.1 to train Instance Splitter. Considering that one object instance may be split into smaller patches on different masks, we append a Merging Net (i.e., a two-layer CNN with 1 kernel size and 1 stride) to Instance Splitter to learn to merge redundant masks by a merging loss $\mathcal{L}_{\text{merge}}$ based on the prior that the patches of the same instance are adjacent to each other and share the same motion. In addition, we add a foreground proposal loss $\mathcal{L}_{\text{forground}}$ to encourage attentions on the dynamic regions. The total loss of this level is given by combining these losses with different weights,

$$\mathcal{L}_{\text{DIS}} = \mathcal{L}_{\text{instance}} + \lambda_3 \mathcal{L}_{\text{merge}} + \lambda_4 \mathcal{L}_{\text{forground}}$$

For more complex domains with arbitrary deformation and appearance change, MAOP is also readily to incorporate the vanilla Mask R-CNN [10] or other off-the-shelf supervised object detection methods [11] as a plug-and-play module into our framework to generate region proposals of dynamic instances. In addition, although the network structure of this level is similar to Object Detector in the level of dynamics learning, we do not integrated them together as a whole network because the concurrent learning of both object representations and dynamics model is not stable. Instead, we first learn the coarse object instances only based on the spatial-temporal consistency of locomotion and appearance pattern, and then use them as proposal regions to perform object-oriented dynamics learning at the more fine-grained level, which in turn fine-tunes the object representations.

A.3 Motion Detection Level

At this level, we employ foreground detection to detect changing regions from a sequence of image frames and provide coarse dynamic region proposals $\mathbf{F}_p$ for assisting in dynamic instance segmentation. In our experiments, we use a simple unsupervised foreground detection approach proposed by Lo and Velastin [12]. Our framework is also compatible with many advanced unsupervised foreground detection methods [13, 14, 15, 16] that are more efficient or more robust to moving camera. These complex unsupervised foreground detection methods have the potential to improve the performance but are not the focus of this work.

Appendix B Instance Localization

Instance localization is a common technique in context of supervised region-based object detection [17, 18, 19, 10, 11], which localizes objects on raw images with regression between the predicted bounding box and the ground truth. Here, we propose an unsupervised approach to perform dynamic instance localization on dynamic object masks learned by Object Detector. Our objective is to sample a number of region proposals on the dynamic object masks and then select the regions, each of which has exactly one dynamic instance. In the rest of this section, we will describe these two steps in details.

Region proposal sampling. We design a learning-free sampling algorithm for sampling region proposals on object masks. This algorithm generates multi-scale region proposals with a full coverage over the input mask. Actually, we adopt multi-fold full coverage to ensure that pixels of the potential instances are covered at each scale. The detailed algorithm is described in Algorithm 2.

Instance mask selection. Instance mask selection aims at selecting the regions, each of which contains exactly one dynamic instance, based on the discrepancy loss $\mathcal{L}_{\text{instance}}$ (Section A.1). To screen out high-consistency, non-overlapping and non-empty instance masks at the same time, we integrate Non-Maximum Suppression (NMS) and Selective Search (SS) [20] in the context of region-based object detection [17, 18, 19, 10, 11] into our algorithm.

Algorithm 2 Region proposal sampling.

Input: Dynamic object mask $\mathbf{D} \in [0,1]^{H \times W}$, the number of region proposal scales n_S, the folds of full coverage T .

Initialize proposal set $\mathbb{P} = \varnothing$.
Binarize $\mathbf{D}$ to get the indicator for the existence of objects
for $l = 1 \dots n_S$ **do**
    Select scale dx, dy depend on the level l.
    for $t = 1 \dots T$ **do**
        Initialize candidate set $\mathbb{C} = \{(i,j) | \mathbf{D}_{i,j} = 1\}$.
        while $\mathbb{C} \neq \varnothing$ **do**
            Sample a pixel coordinate (x, y) from $\mathbb{C}$.
            Get a box $\mathbb{B} = \{(i,j) |\ |i - x| \leq dx, |j - y| \leq dy\}$.
            if $\mathbb{B}$ is not empty **then**
                Insert $\mathbb{B}$ into the proposal set $\mathbb{P} \leftarrow \mathbb{P} \cup \{\mathbb{B}\}$.
            end if
            Update the remain candidate set $\mathbb{C} \leftarrow \mathbb{C} \setminus \mathbb{B}$.
        end while
    end for
end for
return $\mathbb{P}$

Appendix C Tables and Figures

	Models	Training environments				Unseen environments			
		1-5†		1-5‡		1-5†		1-5‡	
		Agent	All	Agent	All	Agent	All	Agent	All
0-error accuracy	MAOP	**0.84**	**0.90**	**0.87**	**0.93**	**0.83**	**0.89**	**0.83**	**0.92**
	OODP	0.01	0.29	0.01	0.32	0.01	0.18	0.02	0.15
	AC Model	0.39	0.64	0.48	0.75	0.03	0.18	0.04	0.23
	CDNA	0.13	0.78	0.41	0.84	0.10	0.77	0.16	0.79
1-error accuracy	MAOP	**0.99**	**1.00**	**0.97**	**0.97**	**0.99**	**0.99**	**0.98**	**0.97**
	OODP	0.05	0.52	0.04	0.56	0.06	0.39	0.07	0.39
	AC Model	0.48	0.80	0.57	0.87	0.07	0.37	0.14	0.45
	CDNA	0.26	0.82	0.57	0.89	0.22	0.81	0.36	0.84
2-error accuracy	MAOP	**1.00**	**1.00**	**0.99**	**0.99**	**1.00**	**1.00**	**0.99**	**0.98**
	OODP	0.14	0.66	0.12	0.67	0.16	0.59	0.16	0.56
	AC Model	0.53	0.85	0.63	0.90	0.12	0.53	0.24	0.64
	CDNA	0.37	0.84	0.66	0.92	0.36	0.84	0.49	0.87

Table C2: Performance of the object dynamics prediction on 1-5 generalization problem in *Flappy Bird*. † and ‡ indicates training with only 100 and 300 samples.

Model	Monster Kong				Flappy Bird	
	1-5†	1-5	2-5	3-5	1-5†	1-5‡
MAOP	1.96	0.34	0.38	0.42	0.30	0.34

Table C3: Average motion prediction error in two experiment environments. † and ‡ correspond to the same sample restriction experiments in Table 1 and C2

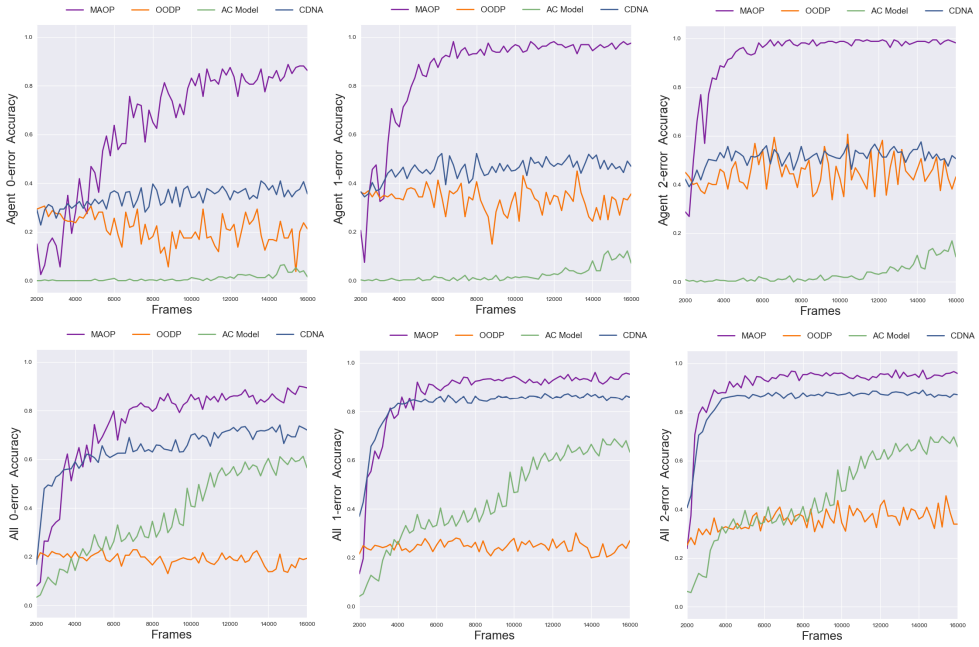

Figure C4: Learning curves for the dynamics prediction in unseen environments on *Monster Kong*. The curves with "Agent" notation represent the learning curves of the agent, while those with "All" notation indicate the learning curves of all dynamic objects.

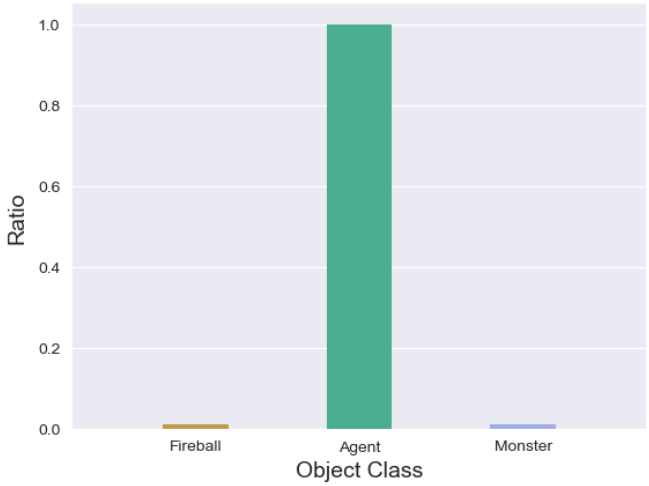

Figure C5: Our discovery of the controllable agent. This histogram plotting the ground-truth label distribution of our discovered action-controlled agents clearly demonstrates that our discovery of the controllable agent achieves perfect 100% accuracy.

Appendix D Implementation details for experiments

Object Detector in the dynamics learning level and Instance Splitter in the dynamic instance segmentation level have similar architectures with Object Detector in OODP [5]. To augment the interactions of instances when training Instance Splitter, we random sample two region proposals and combine them into a single region proposal with double size.

Denote $Conv(F, K, S)$ as the convolutional layer with the number of filters F, kernel size K and stride S. Let $R(), S()$ and $BN()$ denote the ReLU layer, sigmoid layer and batch normalization layer [21]. The 5 convolutional layers in Object Detector can be indicated as $R(BN(Conv(16, 5, 2)))$, $R(BN(Conv(32, 3, 2)))$, $R(BN(Conv(64, 3, 1)))$, $R(BN(Conv(32, 1, 1)))$, and $BN(Conv(1, 3, 1))$, respectively. The 5 convolutional layers in Instance Detector can be indicated as $R(BN(Conv(32, 5, 2)))$, $R(BN(Conv(32, 3, 2)))$, $R(BN(Conv(32, 3, 1)))$, $R(BN(Conv(32, 1, 1)))$, and $BN(Conv(1, 3, 1))$, respectively. The architecture of Foreground Detector is similar to binary-class Object Detector and the 5 convolutional layers in Foreground Detector can be indicated as $R(BN(Conv(32, 5, 2)))$, $R(BN(Conv(32, 3, 2)))$, $R(BN(Conv(32, 3, 1)))$, $R(BN(Conv(32, 1, 1)))$, and $S(BN(Conv(1, 3, 1)))$, respectively. The CNNs in Relation Net are connected in the order: $R(BN(Conv(16, 3, 2)))$, $R(BN(Conv(32, 3, 2)))$, $R(BN(Conv(32, 3, 2)))$, and $R(BN(Conv(32, 3, 2)))$. The last convolutional layer is reshaped and fully connected by the 64-dimensional hidden layer and the 2-dimensional output layer successively. Inertia Net has the same architecture and hyperparameters as Relation Net.

The hyperparameters for training MAOP in *Monster Kong* and *Flappy Bird* are listed as follows:

- The weights of losses, λ_1, λ_2, and λ_3, are 100, 1, and 10, respectively. In addition, all the l_2 losses are divided by HW to keep invariance to the image size.
- Batch size is 16 and the maximum number of training steps is set to be 1×10^5.
- The optimizer is Adam [22] with learning rate 1×10^{-3}.
- The raw images of *Monster Kong* and *Flappy Bird* are resized to $160 \times 160 \times 3$ and $160 \times 80 \times 3$, respectively.
- The size of the horizon window w is 33 on *Monster Kong*, 41 on *Flappy Bird*.
- The maximum number of static and dynamic masks is 4 and 12 on *Monster Kong*, on *Flappy Bird*.
- The maximum instance number of each object class is set to be 15.