Learning Long-term Visual Dynamics with Region Proposal Interaction Networks

Haozhi Qi[1] **Xiaolong Wang**[2] **Deepak Pathak**[3] **Yi Ma**[1] **Jitendra Malik**[1]
[1]UC Berkeley [2]UC San Diego [3]CMU
https://haozhiqi.github.io/RPIN/

Abstract

Learning long-term dynamics models is the key to understanding physical common sense. Most existing approaches on learning dynamics from visual input sidestep long-term predictions by resorting to rapid re-planning with short-term models. This not only requires such models to be super accurate but also limits them only to tasks where an agent can continuously obtain feedback and take action *at each step* until completion. In this paper, we aim to leverage the ideas from success stories in visual recognition tasks to build object representations that can capture inter-object and object-environment interactions over a long range. To this end, we propose *Region Proposal Interaction Networks (RPIN)*, which reason about each object's trajectory in a latent region-proposal feature space. Thanks to the simple yet effective object representation, our approach outperforms prior methods by a significant margin both in terms of prediction quality and their ability to plan for downstream tasks, and also generalize well to novel environments. Our code is available at https://github.com/HaozhiQi/RPIN.

1 Introduction

As argued by Kenneth Craik, *if an organism carries a model of external reality and its own possible actions within its head, it is able to react in much fuller, safer and more competent manner to emergencies which face it* [14]. Indeed, building prediction models has been long studied in computer vision and intuitive physics. In vision, most approaches make predictions in pixel-space [17, 19, 31, 41, 70], which ends up capturing the optical flow [70] and is difficult to generalize to long-horizon. In intuitive physics, a common approach is to learn the dynamics directly in an abstracted state space of objects to capture Newtonian physics [5, 12, 60]. However, the states end up being detached from raw sensory perception. Unfortunately, these two extremes have barely been connected. In this paper, we argue for a middle-ground to treat images as a window into the world, i.e., objects exist but can be accessed only via images. Images are neither to be used for predicting pixels nor to be isolated from dynamics. We operationalize it by learning to extract a rich state representation directly from images and build dynamics using the extracted state representations.

It is difficult to make predictions, especially about the future — Niels Bohr

Contrary to Niels Bohr, predictions are, in fact, easy if made only for the short-term. Predictions that are indeed difficult to make and actually matter are the ones made over the long-term. Consider the example of "Three-cushion Billiards" in Figure 1. The goal is to hit the cue in such a way that it touches the other two balls and contacts the wall thrice before hitting the last ball. This task is extremely challenging even for human experts because the number of successful trajectories is very sparse. Do players perform classical Newtonian physics calculations to obtain the best action before each shot, or do they just memorize the solution by practicing through exponentially many configurations? Both extremes are not impossible, but often impractical. Players rather build

Preprint. Under review.

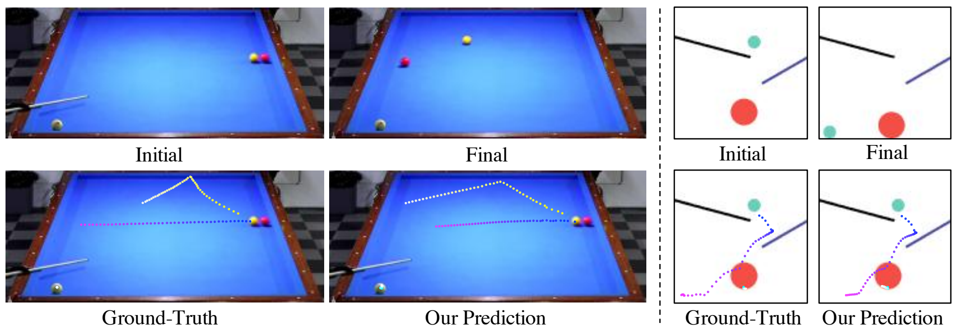


Figure 1: Long-term dynamics prediction tasks. Left: three-cushion billiards. Right: PHYRE intuitive-physics dataset [4]. Our proposed approach makes accurate long-term predictions that do not necessarily align with the ground truth but provide strong signal for planning.

a physical understanding by experience [39, 51, 52] and plan by making intuitive, yet accurate predictions in the long-term.

Learning such long-term prediction models is arguably the "Achilles' heel" of modern machine learning methods. Current approaches on learning physical dynamics of the world *cleverly* side-step the long-term dependency by re-planning at each step via model-predictive control (MPC) [2, 11]. The common practice is to train short-term dynamical models (usually 1-step) in a simulator. However, small errors in short-term predictions can accumulate over time in MPC. Hence, in this work, we focus primarily on the long-term aspect of prediction by just considering environments, such as the three-cushion billiards example or the PHYRE [4] in Figure 1, where an agent is allowed to take *only one* action in the beginning so as to preclude any scope of re-planning.

Our objective is to build data-driven prediction models for intuitive physics [51] that can both: (a) model long-term interactions over time to plan successfully for new instances, and (b) work from raw visual input in real-world scenarios. The question we ask is: how to leverage the ideas from success stories in computer vision tasks (e.g., object detection [24, 59]) to build long-term physical prediction models in a real-world environment? Our main idea is to represent each video frame with an object-centered representation where each object is treated as an individual entity. However, instead of performing prediction in the pixel space as most prior methods do [70, 80], we predict the state (e.g., location, shape) for each object entity in latent feature space which is similar in spirit to the idea of region proposals in detection methods [59].

How to extract object-centric features in an end-to-end fashion? We leverage the region of interests (RoI) pooling [24] to extract object representation from the frame-level feature, and build an interaction module to perform reasoning among the objects. Object feature extraction based on region proposals have achieved huge success in computer vision [15, 24, 25, 29], and yet, surprisingly under-explored in the field of intuitive physics. By using RoI pooling, each object feature not only contains its own object information but also the context of the environment. We will show in Section 5, the contextual information is critical in dealing with interactions in complex environments. The interaction module and the object feature extraction are trained end-to-end by minimizing the distance between predicted and ground-truth object trajectories. We name our approach Region Proposal Interaction Networks (RPIN), illustrated in Figure 2.

Notably, our approach is simple, yet outperforms the state-of-the-art object feature extraction methods in both simulation and real datasets. In Section 5, we thoroughly evaluate our approach across four datasets to study scientific questions related to a) prediction quality, b) generalization to time horizons longer than training, c) generalization to unseen configurations, d) the role of each design choice and, e) planning ability for downstream tasks.

2 Related Work

Physical Reasoning and Intuitive Physics. Learning models that can predict the changing dynamics of the scene is the key to building physical common-sense. Such models date back to "NeuroAnimator" [27] for simulating articulated objects. Several methods in recent years have leveraged deep networks to build data-driven models of intuitive physics [8, 12, 20, 23, 62]. However, these methods either require access to the underlying ground-truth state-space or do not scale to long-range due to absence of interaction reasoning. A more generic yet explicit approach has been to leverage graph neural networks [61] to capture interactions between entities in a scene [6, 12]. Closest to our approach are interaction models that scale to pixels and reason about object interaction [72, 80]. However, these approaches either reason about object crops with no context around or can only deal with a predetermined number and order of objects.

Other common ways to measure physical understanding are to predict future judgments given a scene image, e.g., predicting the stability of a configuration [26, 33, 42–44]. Several hybrid methods take a data-driven approach to estimate Newtonian parameters from raw images [7, 9, 73, 74], or model Newtonian physics via latent variable to predict motion trajectory in images [53, 54, 79]. An extreme example is to use an actual simulator to do inference over objects [28]. The reliance on explicit Newtonian physics makes them infeasible on real-world data and un-instrumented settings. In contrast, we take into account the context around each object via RoIPooling and explicitly model their interaction with each other or with the environment without relying on Newtonian physics, and hence, easily scalable to real videos for long-range predictions.

Video Prediction. Instead of modeling physics from raw images, an alternative is to treat visual reasoning as an image translation problem. This approach has been adopted in the line of work that falls under video prediction. The most common theme is to leverage latent-variable models for predicting future [3, 17, 40]. Predicting pixels is difficult so several methods leverage auxiliary information like back/fore-ground [64, 66, 68], optical flow [48, 70], appearance transformation [13, 22, 32, 78], etc. These inductive biases help in a short interval but do not capture long-range behavior as needed in several scenarios, like playing billiards, due to lack of explicit reasoning. Some approaches can scale to relative longer term but are domain-specific, e.g., pre-defined human-pose space [67, 71]. Furthermore, the primary evaluation of these methods is either via rendering quality or representation [17, 50]. However, our goal is to model long-term interactions not only for prediction but also to facilitate planning for downstream tasks.

Learning Dynamics Models. Unlike video prediction, dynamics models take actions into account for predicting the future, also known as forward models [34]. Learning these forward dynamics models from images has recently become popular in robotics for both specific tasks [1, 22, 56, 69] and exploration [10, 57]. In contrast to these methods where a deep network directly predicts the whole outcome, we leverage our proposed region-proposal interaction module to capture each object trajectories explicitly to learn long-range forward dynamics as well as video prediction models.

Planning via Learned Models. Leveraging models to plan is the standard approach in control for obtaining task-specific behavior. Common approach is to re-plan after each action via Model Predictive Control [2, 11, 16]. Scaling the models and planning in a high dimensional space is a challenging problem. With deep learning, several approaches shown promising results on real-world robotic tasks [1, 21, 22, 58]. However, the horizon of these approaches is still very short, and replanning in long-term drifts away in practice. Some methods try to alleviate this issue via object modeling [30, 45] or skip connections [18] but assume the models are trained with state-action pairs. In contrast to prior works where a short-range dynamic model is unrolled in time, we learn our long-range models from passive data and then couple them with short-range forward models to infer actions during planning.

3 Region Proposal Interaction Networks

Our model takes N video frames as inputs and predicts the object locations for the future T timesteps, as illustrated in Figure 2. We first extract the image feature representation using a ConvNet for each frame, and then apply RoI pooling to obtain object-centric visual features. These object feature representations are forwarded to the interaction modules to perform interaction reasoning and predict future object locations. The whole pipeline is trained end-to-end by minimizing the loss between predicted and the ground-truth object locations. Since the parameters of each interaction module is shared so we can apply this process recurrently over time to an arbitrary T during testing.

3.1 Representation and Prediction Modules

Object-Centric Visual Representation. We first apply a houglass network [55] to extract the image features. Given an input image with size $3 \times H \times W$, the extracted feature map dimension will be $d \times \lfloor \frac{H}{s} \rfloor \times \lfloor \frac{W}{s} \rfloor$, where s is the spatial stride and d is the dimension of visual feature. On top of this

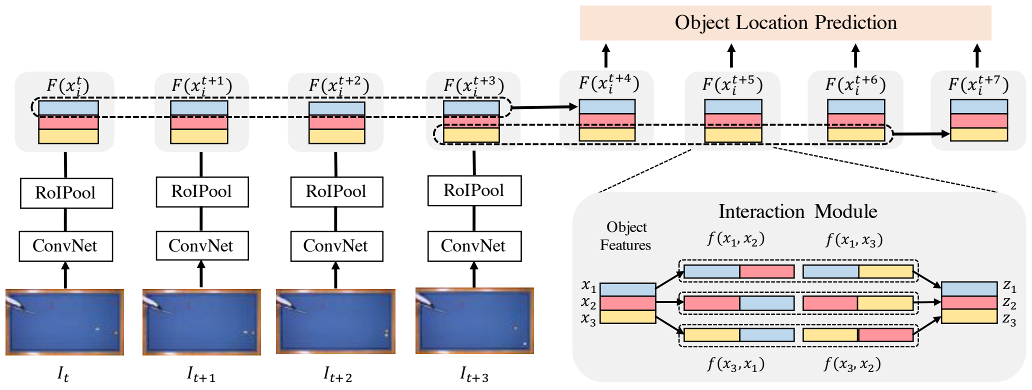


Figure 2: Our Region Proposal Interaction Network. Given N frames as inputs, we forward them to an encoder network, and then extract the foreground object features with RoIPooling (different colors represent different instances). We then perform interaction reasoning on top of the region proposal features (gray box on the bottom right). We predict each future object feature based on the previous k time steps. We then estimate the object location from each object feature.

feature map, we use the RoI pooling operator to extract the $d \times 3 \times 3$ object features. This feature is then flattened and forwarded to a fully connected layer to output a d-dimension feature vector. Besides the visual features, we also forward the object center location to a 2 layer fully-connected encoder to get object position embedding. The final object feature is the concatenation of visual feature and position feature. We will use x_i^t to represent the feature at t-th timestep for the i-th object.

Interaction Module. The interaction module is shown in the gray box on the bottom right of Figure 2. Our interaction reasoning is directly applied on the latent feature representation for each object. Assuming we have m object at time step t, with feature $X = \{x_1^t, x_2^t, ..., x_m^t\}$. The interaction reasoning is performed between every two objects:

$$f(x_i^t, x_j^t) = \text{ReLU}\big(W_f^T[x_i^t, x_j^t]\big), \tag{1}$$

where $W_f \in R^{2d \times d}$ is a learnable weight. Note that the interaction reasoning is only applied when the Euclidean distance between these two objects are smaller than a pre-defined threshold [12, 60]. For object i, $\mathcal{N}(i) = \{j \mid ||p_j - p_i||_2 \le \zeta, j \ne i\}$ as the set of objects satisfies this constraint, where p_i is the position of object i and ζ is the threshold hyperparameter. Then the updated feature for the i-th object by:

$$z_i^t = W_z^T\big(W_h^T(x_i^t) + \sum_{j \in \mathcal{N}(i)} f(x_i^t, x_j^t)\big), \tag{2}$$

where z_i^t is the updated feature and $W_z \in R^{d \times d}$ and $W_h \in R^{d \times d}$ are learnable weights implemented via a fully connected layer. The feature dimension of z_i the same as x_i. For simplicity, we denote the interaction reasoning process on x_i as $z_i^t = F(x_i^t)$.

Prediction Model. Given the individual object representation from the interaction module in a few time steps, we can predict the future object state representation x_i^{t+1} by:

$$x_i^{t+1} = \text{ReLU}\big(W_d^T[F(x_i^t), F(x_i^{t-1}), \dots, F(x_i^{t-k})]\big), \tag{3}$$

where we first concatenate the features for object i in the past k time steps, and then forward the concatenated feature to a fully connected layer with weights W_d. Note that although we show an example of using $k = 4$ in Figure 2 (dashed black rectangle), the model can be easily generalized to integration of more time steps. We apply this prediction model recurrently over different time steps until predicting all the object features in the future T frames.

3.2 Learning Region Proposal Interaction Networks (RPIN)

Instead of predicting pixels, we train our model by predicting the future locations of each object since we believe this is the key of doing planning tasks. Given the predicted feature x_i^{t+1} for the ith object in time $t + 1$, we estimate its spatial location coordinates by a simple one layer decoder:

$\hat{p}_i^{t+1} = W_p^T x_i^{t+1}$. The ground-truth coordinate p_i^{t+1} is a 2-dimension coordinate normalized by the size of the input image. To facilitate training, besides the object location, we also predict its relative location (offset) $\Delta p_i^{t+1} = p_i^{t+1} - p_i^t$ with another fully connected layer. We apply Euclidean distance between the predicted object locations and the ground-truth locations as the training objective:

$$L_p = \sum_{t=1}^{T} \lambda_t \Big(\sum_{i=1}^{n} \|\hat{p}_i^{t+1} - p_i^{t+1}\|_2^2 + \alpha \|\Delta \hat{p}_i^{t+1} - \Delta p_i^{t+1}\|_2^2 \Big), \tag{4}$$

where α is a constant value to balance the two losses. We use discounted loss during training [72] to mitigate the effect of inaccurate prediction at early training stage and λ_t is the discounted factor.

Uncertainty Modeling. Our model can also be adopted in cases where only a single image input is available by setting $N = 1$. We also incorporate uncertainty estimation follows [80], by modeling the latent distribution using a variational auto-encoder [36]. For the complete details, we refer the reader to [80]. Here we only give a summary: we build an encoder h which takes the image feature from first $\mathcal{F}^0$ and last frame $\mathcal{F}^T$ of a video sequence as the input. The output of h is a distribution parameter, denoted by $h(\mathbf{u}|\mathcal{F}^0, \mathcal{F}^T)$. Given a particular sample from such distribution, we recover the latent variable by feeding them into a one-layer LSTM and merge into the object feature x_i^t. In this case, our pipeline is trained with an additional loss that minimize the KL divergence between the predicted distribution and normal distribution [36].

4 Experimental Setup

Datasets. We evaluate our method's prediction performance on four different datasets, and demonstrate the ability to perform planning for downstream tasks on two of them. We briefly introduce the four datasets below. The full dataset details are in the appendix.

Simulation Billiards (SimB): We use the simulation environment extended from [38, 63]. The image size is 64×64. Three different colored balls with a radius 2 are randomly placed in the image. To get initial velocity, we randomly sample 5 different magnitudes and 12 directions and apply it on one of the balls. We generate 1,000 video sequences for training and 1,000 video sequences for testing, with 100 frames per sequence. We will also evaluate the ability to generalize to more balls and different sized balls in the experiment section.

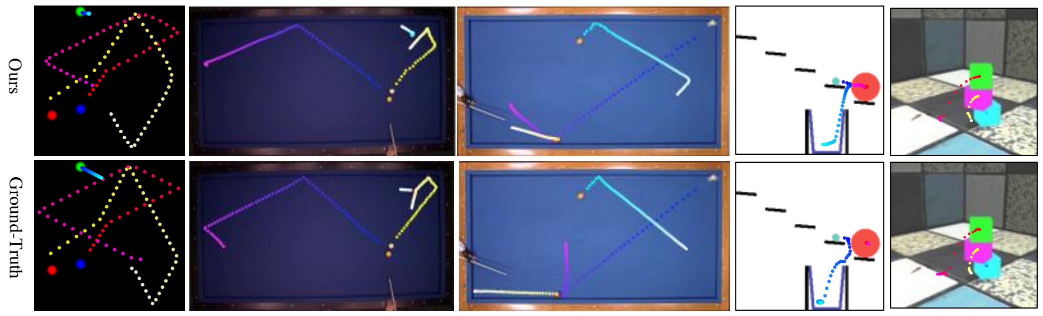
Real World Billiards (RealB): This dataset contains "Three-cushion Billiards" videos from three separate professional games with different viewpoints downloaded from YouTube. There are 62 training videos with $18,306$ frames, and 5 testing videos with $1,995$ frames. To get the bounding box annotations, we use off-the-shelf detector [47, 77] to detect the billiards. The detector is initialized from a ResNet-101 FPN model pretrained on COCO [46] dataset and fine-tuned on a subset of 30 images from our dataset.

PHYRE: We select 13 out of 25 tasks from the PHYRE benchmark [4]. We treat all the moving balls as objects and other static bodies as background. For each task, we split the provided 100 templates to 80 training templates and testing 20 templates. This setting is called within task generalization (PHYRE-W), where the testing environments contain the same object category but different sizes and positions. We will also evaluate our model's performance on environments containing objects and context it never seen during training (called cross task generalization (PHYRE-C)). In this setting, ten tasks are used as the training set. And the remaining three tasks are used for testing.

ShapeStacks (SS): This dataset is a synthetic dataset of multiple stacked objects (cubes, cylinders, or balls) [80]. Only objects' center positions provided. Following [80], we assume the object bounding box is square and of size 70×70. There are 1,320 training videos and 296 testing videos, with 32 frames per video. In this dataset, we set $N = 1$, and uncertainty estimation is incorporated.

Baseline Comparisons. We consider the following baselines. Since the considered baselines are usually tuned on different datasets using different architectures, it is hard to make a fair comparison. To mitigate this, we re-implement them using the same network structure and hyperparameter as ours so that only the way of getting visual object features are changed.

Visual Interaction Network (VIN) [37, 72]: Instead of using object-centric spatial pooling to extract object features, it directly assigns different channels of image feature to different objects. This approach requires specifying a fixed number of objects and a fixed mapping between feature channels



(a) Sim-Billiard (b) Real-Billiard (c) Real-Billiard (d) PHYRE (e) ShapeStacks

Figure 3: Visualization on all of the four datasets. The first row is our prediction results and the second row is the corresponding ground-truth trajectories. Our method accurately predicts long-term future even after complex interactions.

and object identity, which limits its generalization ability to a different number of objects and different appearances.

Object Masking (OM) [30, 65, 75]: This approach takes one image and m object proposals as input. For each proposal, only the pixels inside object proposals are kept while others are set to 0, leading to m masked images. This approach assumes no background information is needed thus fails to predict accurate trajectories in complex environments such as PHYRE. And it also cost m times computational resources.

Compositional Video Prediction (CVP) [80]: The object feature is extracted by cropping the object image patch and forwarding it to an encoder. Since the object features are directly extracted from the raw image patches, the context information is ignored. We use CVP* to denote our re-implementation. For the ShapeStack dataset, we consider both our re-implementation as well as the original model published with [80].

Metric. Given predictions for m objects $\hat{p} \in R^{T \times O \times 2}$. The prediction error for time step t is

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{2}(\hat{p}_{t,i,j} - p_{t,i,j})^2. \tag{5}$$

In the results, we report the average error for two horizons: $t \in [0, T_{\text{train}}]$ and $t \in [T_{\text{train}}, 2 \times T_{\text{train}}]$.

5 Evaluation Results: Prediction, Generalization, and Planning

Figure 3 shows some qualitative prediction results. More results are in the supplementary material and our project website. We organize this section and analyze our results by discussing five scientific questions related to the prediction quality, generalization ability across time & environment configurations, different design choices, and the ability to plan actions for downstream tasks.

5.1 How accurate is the predicted dynamics?

To evaluate how well the world dynamics is modeled, we report the prediction errors on the test split over similar time-horizon as which model is trained on, i.e., $t \in [0, T_{\text{train}}]$. The results are shown in Table 1 (left half). In this setting, the OM method performs relatively better than other baselines in the billiard and PHYRE datasets since it explicitly models objects by instance masking. In contrast, VIN needs to learn to attend on object features from a global image, which may make learning accurate object representation harder, leading to worse performance. Meanwhile, VIN requires a fixed number of objects, so it is not even trainable in the PHYRE dataset which contains a variable number of objects. For the CVP* method, it performs poorly on simulated billiard due to the complex dynamics in this dataset (i.e., there are more interactions, as shown in 3). The performance of CVP is reasonable in RealB and PHYRE, but still worse than OM. In the SS dataset, all the baselines including CVP work decently well. One possible explanation is that the object size is large, and cropped image regions already provide enough context. Note that the re-implemented CVP method has similar performance with the original number reported, which shows our re-implementation is proper. Finally, our method, with both explicit object modeling and context feature learning, achieves the best results on all of the four datasets. This demonstrates the advantage of using rich state representations. Note that in ShapeStack datasets, neither the baseline nor our method uses the pixel-wise supervision and

method	$t \in [0, T_{\text{train}}]$				$t \in [T_{\text{train}}, 2 \times T_{\text{train}}]$			
	SimB	RealB	PHYRE	SS	SimB	RealB	PHYRE	SS
VIN [37, 72]	3.89	1.02	N.A.	2.47	29.51	5.11	N.A.	7.77
OM [30, 75]	3.58	0.59	11.31	3.01	27.87	3.23	22.96	9.51
CVP* [80]	82.15	0.85	22.63	2.84	112.09	4.26	35.84	7.72
CVP [80]	-	-	-	1.95	-	-	-	11.42
Ours	**2.44**	**0.34**	**4.46**	**1.59**	**22.20**	**2.19**	**12.20**	**6.83**

Table 1: We compare our method with different baselines as well as [80] on all four datasets. The left part shows the prediction error when rollout timesteps is the same as training time. The right part shows the generalization ability to longer horizon unseen during training. The error is scaled by 1,000. * denotes re-implementation for fair comparison with ours.

method	SimB-5	SimB-L	PHYRE-C	SS-4
VIN [37, 72]	N.A.	54.77	N.A.	N.A.
OM [30, 75]	59.70	39.42	19.83	36.30
CVP* [80]	113.39	102.34	73.72	36.02
CVP [80]	-	-	-	15.96
Ours	**15.56**	**38.65**	**11.36**	**13.97**

Table 2: The ability to generalize to novel environments. We show the average prediction error for $t \in [0, 2 \times T_{\text{train}}]$. Our method achieves significantly better results compared to previous methods.

	visual	local	pos	$t \in [0, T_{train}]$
(a)			✓	14.45
(b)		✓	✓	15.18
(c)	✓			4.86
(d)	✓		✓	4.63
(e)	✓	✓	✓	**4.46**

Table 3: Ablation on PHYRE-W. We compare the effect of applying local interaction and position features to our baseline.

stacked interaction networks as in [80]. Thanks to the rich state representation, we can achieve much more accurate object trajectory prediction even with a much simpler interaction modeling.

5.2 Does learned model generalize to longer horizon than training?

As the parameters of our interaction module and prediction module are shared over time, our model can predict a longer sequence than training time. In Table 1 (right half), we show the average prediction error for $t \in [T_{\text{train}}, 2 \times T_{\text{train}}]$. The results in this setting are consistent with what we found in Section 5.1: OM performs better on both the billiard and PHYRE datasets, and the CVP* method performs poorly on SimB and a little bit better on RealB and PHYRE. On the ShapeStacks dataset, an interesting observation is that all the baselines are better than [80]. We hypothesize this is because the network representation power is efficiently spent on predicting accurate locations, instead of achieving balance with visual quality as in [80]. Still, our method achieves the best performance against all baselines as well as [80]. This again validates our hypothesis that the key to making accurate long-term feature prediction is the rich state representation extracted from an image.

5.3 Does learned mode generalize to unseen configurations?

The general applicability of RoIPool has been extensively verified in the computer vision community. Our method can generalize to novel environments configurations without any modifications, thanks to the object-centric representations. We test such a claim by testing on several novel environments unseen during training. Specifically, we construct 1) simulation billiard dataset contains 5 balls with radius 2 (SimB-5); 2) simulation billiard dataset contains 3 balls and larger radius from 2 to 5 (SimB-L); 3) PHYRE-C where the test tasks are not seen during training; 4) ShapeStacks with 4 stacked blocks (SS-4). The results are shown in Table 2.

Since VIN needs a fixed number of objects as input, it cannot generalize to a different number of objects, thus we don't report its performance on SimB-5, PHYRE-C, and SS-4. Its generalization to larger objects is also poor for lack of explicit object modeling. The OM method performs better than other baselines. One surprising finding is that the baselines are worse than [80] on SS-4, which is in contrast to our findings in Table 1. We hypothesize this indicates the pixel-wise supervision provides regularization to the model, thus helps reduce overfitting and improve generalization to novel environments. Our method, although without such regularization, still achieves better performance.

	Target State Error	Hitting Accuracy	PHYRE-W	PHYRE-C
Random Policy	36.91	9.50%	0.0% / 46.9%	0.0% / 30.0%
VIN [37, 72]	8.03	62.1%	N.A.	N.A.
OM [30, 75]	7.79	64.5%	29.2% / 80.4%	15.3% / 45.0%
CVP* [80]	29.65	23.8%	4.2% / 40.0%	2.7% / 34.3%
Ours	**6.86**	**68.8%**	**33.1% / 83.5%**	**18.3% / 74.7%**

Table 4: We show planning results for Simulation Billiards and PHYRE. From left to right (i) Init-End State dataset (lower number the better); (ii) Hitting Accuracy (higher number the better); (iii) PHYRE Within task success rate (high number the better). (iv) PHYRE Cross task success rate. For PHYRE, we shows success rate for 100 action trials.

5.4 How does model performance vary with respect to different design choices?

In Table 3, we analyze the effect of several network components on the PHYRE-W dataset, including position encoding and local interaction constraint. Firstly, (a) shows that with only position features, the prediction error is very high. This is because the position of objects cannot represent the complex environment setting in PHYRE dataset. With the local interaction constraint, the error is even higher. In contrast, (c) shows that our method achieves significantly better results using only visual features, which demonstrates the effectiveness of simultaneously model the environment and the object. (d) shows that adding position encoding features to our baseline leads to another 0.23 improvement, which indicates these two features are complementary to each other. Finally, adding local interaction constraints can improve performance by about 0.17, suggesting the effectiveness of prior knowledge to facilitate interaction learning.

5.5 How well can the learned model be used for planning actions?

The advantage of using a general-purpose prediction model is that it can be used to do downstream planning tasks without any adaptation. We evaluate our prediction model in simulation billiards and a subset of PHYRE. To analyze the long-term prediction ability of our model under a controlled setting, we will use the environment to generate the first $N = 4$ frames given one initial configuration and one candidate action. The resulting frames will be used as the input to our predictive model. We score each action according to the similarity between the generated trajectory and the goal state. Then the action with the highest score is selected. The full planning algorithm and implementation details will be included in the appendix. We evaluate the planning performance on the following tasks:

Billiard Target State. Given an initial and final configuration after 40 timesteps, the goal is to find one action that will lead to the target configuration. We report the smallest distances between the trajectory between timestep 35-45 and the final position.

Billiard Hitting. Given the initial configurations, the goal is to find an action that can hit the other two balls within 50 timesteps. We report the average success rate over all different configurations.

PHYRE. In this task, we need to place a red ball to solve a specific goal for each environment (see figure 1 right for an example). The action space contains the position and size of the red ball. We uniformly sample 2000 actions from the continuous action space and score each action according to the similarity. We report the success rate for both top-1 and top-100 actions.

The results are shown in Table 4. The planning accuracy is consistent with the prediction performance. Our method performs significantly better than baselines in both simulated billiard planning and PHYRE tasks, especially on the PHYRE cross-task generalization tasks.

6 Conclusions

In this paper, we leverage the modern computer vision techniques to propose *Region Proposal Interaction Networks* for physical interaction reasoning with visual inputs. We show that our general, yet simple method achieves a significant improvement and can generalize across both simulation and real-world environments for long-range prediction and planning. We believe this method may serve as a good benchmark for developing future methods in the field of learning intuitive physics, as well as their application to real-world robotics.

Acknowledgement

This work is supported in part by DARPA MCS and DARPA LwLL. We would like to thank the members of BAIR for fruitful discussions and comments.

References

[1] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NeurIPS*, 2016. 3

[2] F. Allgöwer and A. Zheng. *Nonlinear model predictive control*. Birkhäuser, 2012. 2, 3

[3] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. *ICLR*, 2017. 3

[4] A. Bakhtin, L. van der Maaten, J. Johnson, L. Gustafson, and R. Girshick. Phyre: A new benchmark for physical reasoning. *arXiv*, 2019. 2, 5

[5] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016. 1

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. 3

[7] K. S. Bhat, S. M. Seitz, J. Popović, and P. K. Khosla. Computing the physical parameters of rigid-body motion from video. In *ECCV*, 2002. 3

[8] A. Bhattacharyya, M. Malinowski, B. Schiele, and M. Fritz. Long-term image boundary extrapolation. *arXiv*, 2016. 2

[9] M. A. Brubaker, L. Sigal, and D. J. Fleet. Estimating contact dynamics. In *ICCV*. IEEE, 2009. 3

[10] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019. 3

[11] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013. 2, 3

[12] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *ICLR*, 2016. 1, 2, 3, 4

[13] B. Chen, W. Wang, and J. Wang. Video imagination from a single image with transformation generation. In *ACMMM Workshop*, 2017. 3

[14] K. J. W. Craik. *The nature of explanation*. CUP Archive, 1952. 1

[15] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017. 2

[16] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011. 3

[17] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018. 1, 3

[18] F. Ebert, S. Dasari, A. X. Lee, S. Levine, and C. Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. *arXiv*, 2018. 3

[19] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv*, 2018. 1

[20] S. Ehrhardt, A. Monszpart, N. J. Mitra, and A. Vedaldi. Learning a physical long-term predictor. *arXiv*, 2017. 2

[21] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017. 3

[22] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016. 3

[23] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *ICLR*, 2015. 2, 13

[24] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2

[25] G. Gkioxari, J. Malik, and J. Johnson. Mesh r-cnn. In *ICCV*, 2019. 2

[26] O. Groth, F. Fuchs, I. Posner, and A. Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. *ECCV*, 2018. 3

[27] R. Grzeszczuk, D. Terzopoulos, and G. Hinton. Neuroanimator: Fast neural network emulation and control of physics-based models. In *SIGGRAPH*, 1998. 2

[28] J. Hamrick, P. Battaglia, and J. B. Tenenbaum. Internal physics models guide probabilistic judgments about object dynamics. In *Proceedings of the 33rd annual conference of the cognitive science society*, 2011. 3

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 2

[30] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-oriented prediction and planning. *ICLR*, 2019. 3, 6, 7, 8

[31] D. Jayaraman, F. Ebert, A. A. Efros, and S. Levine. Time-agnostic prediction: Predicting predictable video frames. *arXiv*, 2018. 1

[32] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NeurIPS*, 2016. 3

[33] Z. Jia, A. C. Gallagher, A. Saxena, and T. Chen. 3d reasoning from blocks to stability. *PAMI*, 2015. 3

[34] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 1992. 3

[35] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 13

[36] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *ICLR*, 2014. 5

[37] T. Kipf, E. van der Pol, and M. Welling. Contrastive learning of structured world models. *ICLR*, 2020. 5, 7, 8

[38] J. Kossen, K. Stelzner, M. Hussing, C. Voelcker, and K. Kersting. Structured object-aware physics prediction for video modeling and planning. *arXiv*, 2019. 5

[39] J. R. Kubricht, K. J. Holyoak, and H. Lu. Intuitive physics: Current research and controversies. *Trends in cognitive sciences*, 2017. 2

[40] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv*, 2018. 3

[41] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv*, 2018. 1

[42] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. *ICML*, 2016. 3

[43] W. Li, S. Azimi, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv*, 2016.

[44] W. Li, A. Leonardis, and M. Fritz. Visual stability prediction and its application to manipulation. *AAAI*, 2016. 3

[45] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. Propagation networks for model-based control under partial observation. In *ICRA*, 2019. 3

[46] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[47] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 5

[48] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 3

[49] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*, 2016. 13

[50] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv*, 2015. 3

[51] M. McCloskey. Intuitive physics. *Scientific american*, 1983. 2

[52] M. McCloskey, A. Washburn, and L. Felch. Intuitive physics: the straight-down belief and its origin. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 1983. 2

[53] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *CVPR*, 2016. 3

[54] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi. "what happens if..." learning to predict the effect of forces in images. In *ECCV*, 2016. 3

[55] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 3

[56] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *NeurIPS*, 2015. 3

[57] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *ICML*, 2017. 3

[58] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 3

[59] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2

[60] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. *arXiv*, 2020. 1, 4

[61] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Network*, 2009. 3

[62] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, 2017. 2

[63] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In *NeurIPS*, 2009. 5

[64] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. *CVPR*, 2017. 3

[65] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. B. Tenenbaum, and S. Levine. Entity abstraction in visual model-based reinforcement learning. In *CoRL*, 2019. 6

[66] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017. 3

[67] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *ICML*, 2017. 3

[68] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016. 3

[69] N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *ICML*, 2015. 3

[70] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016. 1, 2, 3

[71] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017. 3

[72] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *NeurIPS*, 2017. 3, 5, 7, 8

[73] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NeurIPS*, 2015. 3

[74] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016. 3

[75] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. In *NeurIPS*, 2017. 6, 7, 8

[76] Y. Wu and K. He. Group normalization. In *ECCV*, 2018. 13

[77] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5

[78] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NeurIPS*, 2016. 3

[79] T. Ye, X. Wang, J. Davidson, and A. Gupta. Interpretable intuitive physics model. *ECCV*, 2018. 3

[80] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani. Compositional video prediction. *ICCV*, 2019. 2, 3, 5, 6, 7, 8, 13

A Implementation Details

A.1 Network Backbone

To keep the comparison as fair as possible, we use the same hourglass network as the image feature extractor for our method and the baselines (VIN, OM, CVP*). Given an input image, we apply a 7×7 stride-2 convolution, three residual blocks with channel dimension C, and a stride-2 max pooling on it. Then this intermediate feature representation is fed into one hourglass modules. In each hourglass, the feature maps are down-sampled with 3 stride-2 residual blocks and then up-sampled with nearest neighbor interpolation. The dimensions of both the input channel and the output channel of each residual block are $4C$. Finally, the output features are transformed to object-centric representations. For the Simulated Billiard dataset, we use $C = 16$ since the environment is relatively simple. For Real Billiard, PHYRE, and ShapeStack dataset, $C = 64$. We use batch normalization before each convolutional layer for the Simulated Billiard and ShapeStack dataset. And since the batch size of PHYRE and Real Billiard is relatively small, we use group normalization [76]. Normalization layers are not used after the network backbone.

A.2 Dataset Details

SimB: To get the initial velocity, the magnitude (number of pixels moved per timestep) is sampled from $[2, 3, 4, 5, 6]$ and the direction is sampled from $\{6i\pi, i = 0, 1, \ldots, 11\}$.

RealB: We found that the bounding box prediction results are accurate enough to serve as the ground-truth. After running the detector, we also manually go through the dataset and filter out images with incorrect detections.

PHYRE: The id for the 13 selected tasks is {0, 1, 2, 7, 11, 12, 13, 14, 15, 16, 19, 20, 24}. We use the fold id 0 provided by the dataset to split it into the training/testing dataset. For within task generalization (PHYRE-W), the training set contains 80 templates from each task. The testing set contains the remaining 20 templates from each task. For cross task generalization (PHYRE-C), the training set contains 100 templates from {0, 1, 2, 7, 11, 12, 13, 16, 20, 24} while the test set contains 100 templates from {14, 15, 19}. For each template, we randomly sample a maximum 50 success and 50 fail actions to collect the trajectories to train our model. The image sequence is temporally downsampled by 20.

A.3 Hyperparameters

We use Adam optimizer [35] with cosine decay [49] to train our networks. The default input frames is $N = 6$ except $N = 1$ for ShapeStacks. We set d to be 256 except for simulation billiard d is 64. During training, T (denoted as T_{train}) is set to be 20 except for ShapeStacks where we use 15 for fair comparison with [80]. The discounted factor λ_t is set to be $(\frac{\text{current\_iter}}{\text{max\_iter}})^t$.

Simulation Billiards. The image size is 64×64. We train the model for 100K iterations with a learning rate 2×10^{-3} and batch size 200. The local constraint threshold ζ is 1.5 times object size.

Real World Billiards. The image is resized to 192×64. We train the model for 240K iterations with a learning rate 1×10^{-4} and batch size 20. The local constraint threshold ζ is 1.5 times object size.

PHYRE. The image is resized to 128×128. We train the model for 150K iterations with a learning rate 2×10^{-4} and batch size 20. The local constraint threshold ζ is 2.5 times object size.

ShapeStacks. The image is resized to 224×224. We train the model for 25K iterations with a learning rate 2×10^{-4} and batch size 40. In this dataset, we apply uncertainty modeling as described in section 3.2. The loss weight of KL-divergence is 3×10^{-5}. During inference, following [80], we randomly sample 100 outputs from our model, and select the best (in terms of the distance to ground-truth) of them as our model's output. The local constraint threshold ζ is 1.5 times object size.

B Planning Details

Given an initial state (represented by an image) and a goal, we aim to produce an action that can lead to the goal from the initial state. Our planning algorithm works in a similar way as visual imagination [23]: Firstly, we select a candidate action $\mathbf{a}$ from a candidate action set $\mathcal{A}$. Then we generate six input images $\mathcal{I} = \{I_0, \ldots, I_5\}$ using the corresponding simulator. After that, we can forward the images to our prediction model to get future object trajectories for each object i and each timestep t: $\{\hat{p}_i^t\}_{i=1,\ldots,m}^{t=1,\ldots,T}$. The score of each action can be calculated by a score function designed for each task. We then select the action that can maximize the score.

We introduce the action set for each task in section B.1, and how to design distance function in B.2. A summary of our algorithm is in Algorithm 1.

B.1 Candidate Action Sets

For simulation billiard, the action is 3 dimensional. The first two dimensions stand for the direction of the force. The last dimension stands for the magnitude of the force. During doing planning, we enumerate over 5 different magnitudes and 12 different angles, leading to 60 possible actions. All of the initial condition is guaranteed to have a solution.

For PHYRE, the action is also 3 dimensional. The first two dimensions stand for the location placing the red ball. The last dimension stands for the radius of the ball. During doing planning, we randomly draw 2000 actions from a uniform distribution.

B.2 Distance Function

Init-End State Error. Denote the given target location of m objects as $y \in R^{m\times 2}$. We use the following distance function, which measures the distance between the final rollout location and the target location:

$$D = \sum_{i=1}^{m}\sum_{j=1}^{2}(\hat{p}_{T,i,j} - y_{i,j})^2 \tag{6}$$

Hitting Accuracy. Denote the given initial location of m objects as $x \in R^{m\times 2}$. We apply force at the object i'. We use the following distance function, which prefer the larger moving distance for objects other than i':

$$D = -\min_i \sum_{i=1, i\neq i'}^{m}\sum_{j=1}^{2}(\hat{p}_{T,i,j} - x_{i,j})^2 \tag{7}$$

PHYRE task. In this task, we are required to place a red ball in a way that can make a given green ball touch a certain goal object, either another moving ball or a purple wall. We denote the center position of the goal object as $(y_1, y_2) \in R^2$ and the index of green ball as i'. Then we define the following distance function, which consider the distance between the green ball and the goal position in the horizontal and vertical distance respectively:

$$D_1 = -\sum(\hat{p}_{T,i',1} - y_1)^2 \tag{8}$$

$$D_2 = -\sum(\hat{p}_{T,i',2} - y_2)^2 \tag{9}$$

For task 1, we use D_1 as our score function. For task $\{2, 12, 13, 15, 24\}$, we use D_2 as our score function. For the remaining tasks, we use $D_1 + D_2$ as our score function.

B.3 Planning Algorithm

Algorithm 1: Planning Algorithm for Simulated Billiard and PHYRE

```
Input: candidate actions $\mathcal{A} = \{\mathbf{a}_i\}$, initial state $x$, end state $y$ (optional)
Output: action $\mathbf{a}^*$
for $\mathbf{a}$ in $\mathcal{A}$ do
    $\mathcal{I}$ = Simulation$(x, \mathbf{a})$ ;
    $\hat{p}$ = PredictionModel$(\mathcal{I})$ ;
    calculate $D$ according to task as in B.2;
    if $D < D^*$ then
        $D^* = D$ ;
        $a^* = a$ ;
    end
end
```

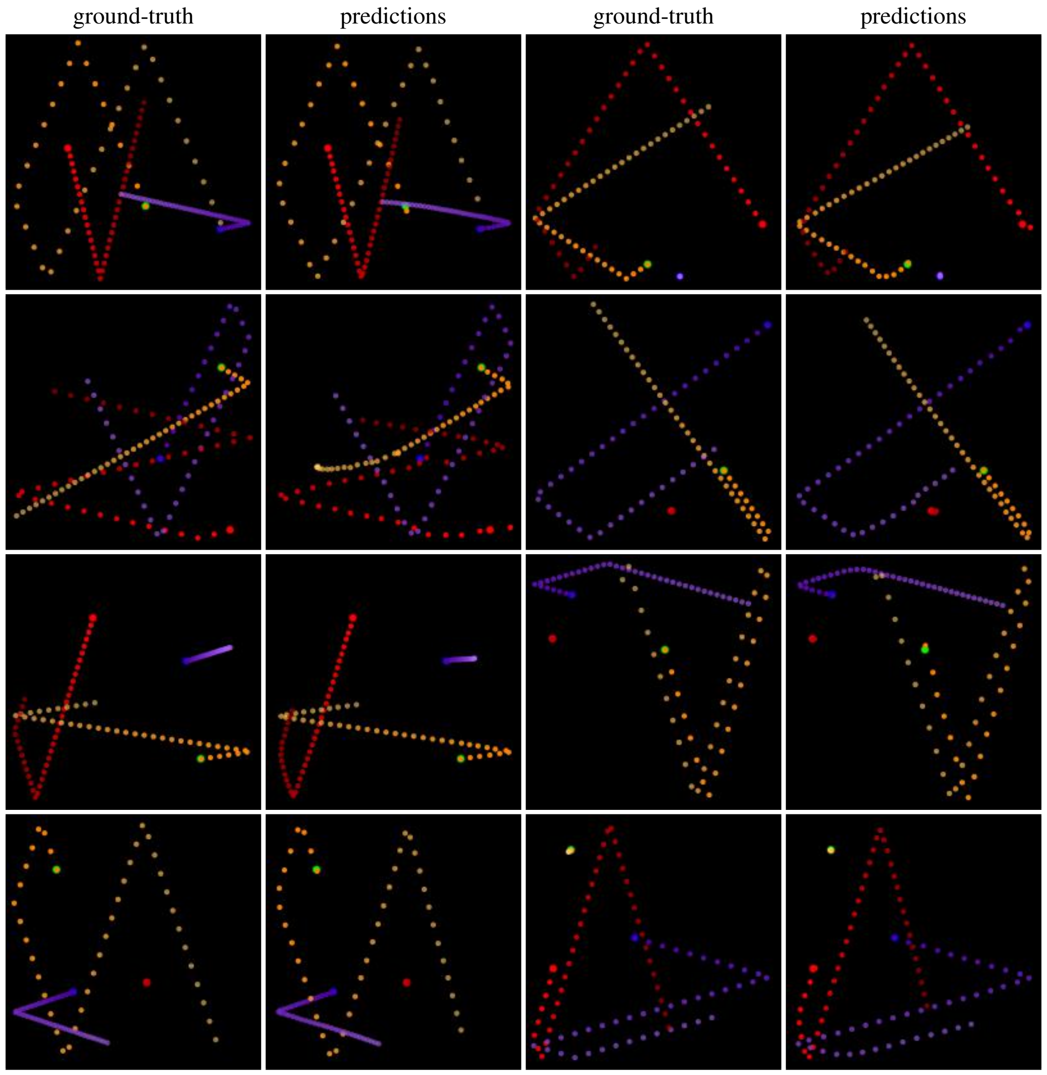


Figure 4: Visualization results of the Simulated Billiard dataset. We visualize the first input image and the trajectories in future 40 timesteps.

C Qualitative Experiments

We show some of the qualitative results in this section. For more results, we refer reader to our project website: https://haozhiqi.github.io/RPIN/.

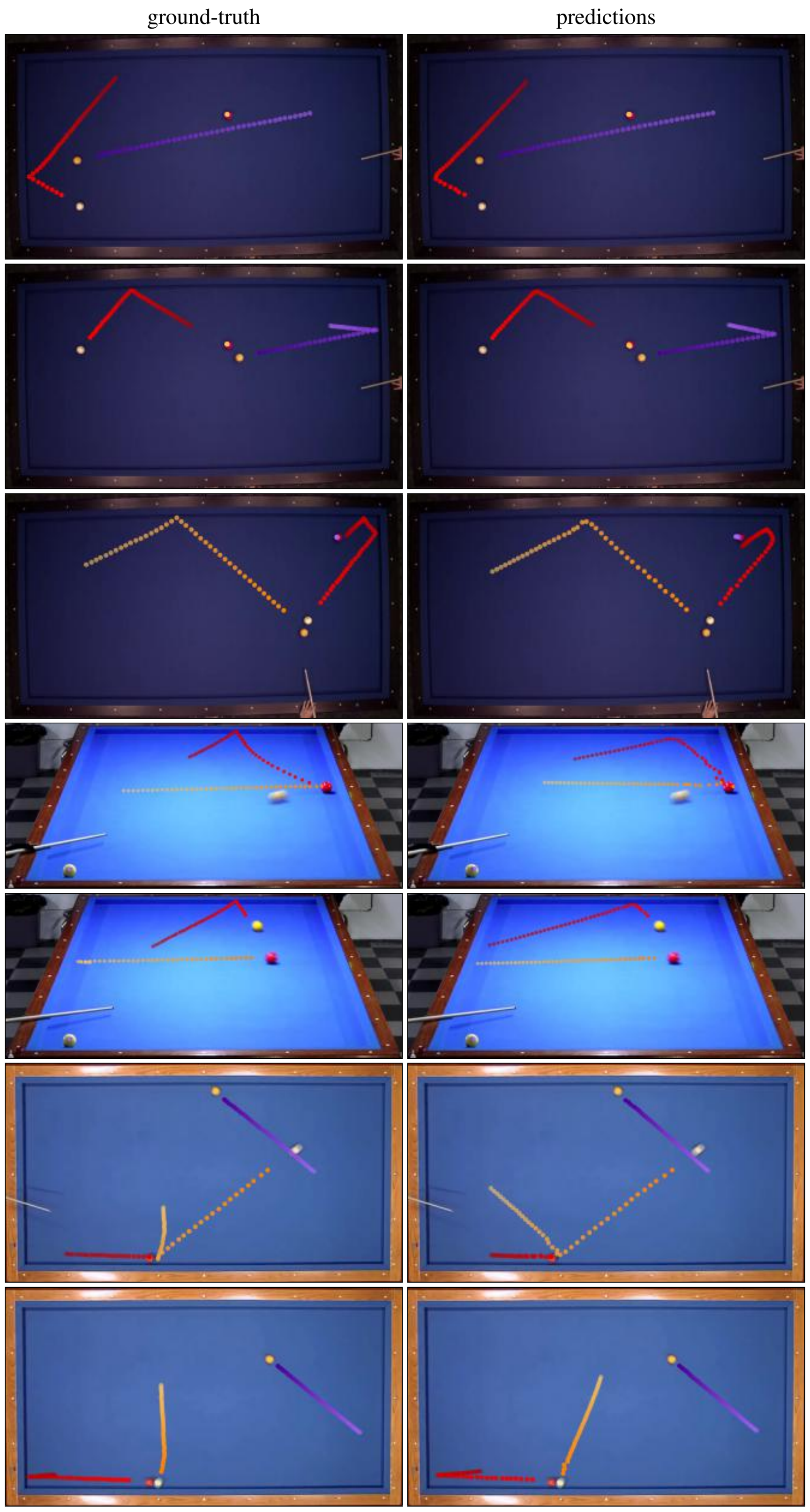


Figure 5: Visualization results of the Real-World Billiard dataset. We visualize the first input image and the trajectories in future 40 timesteps.

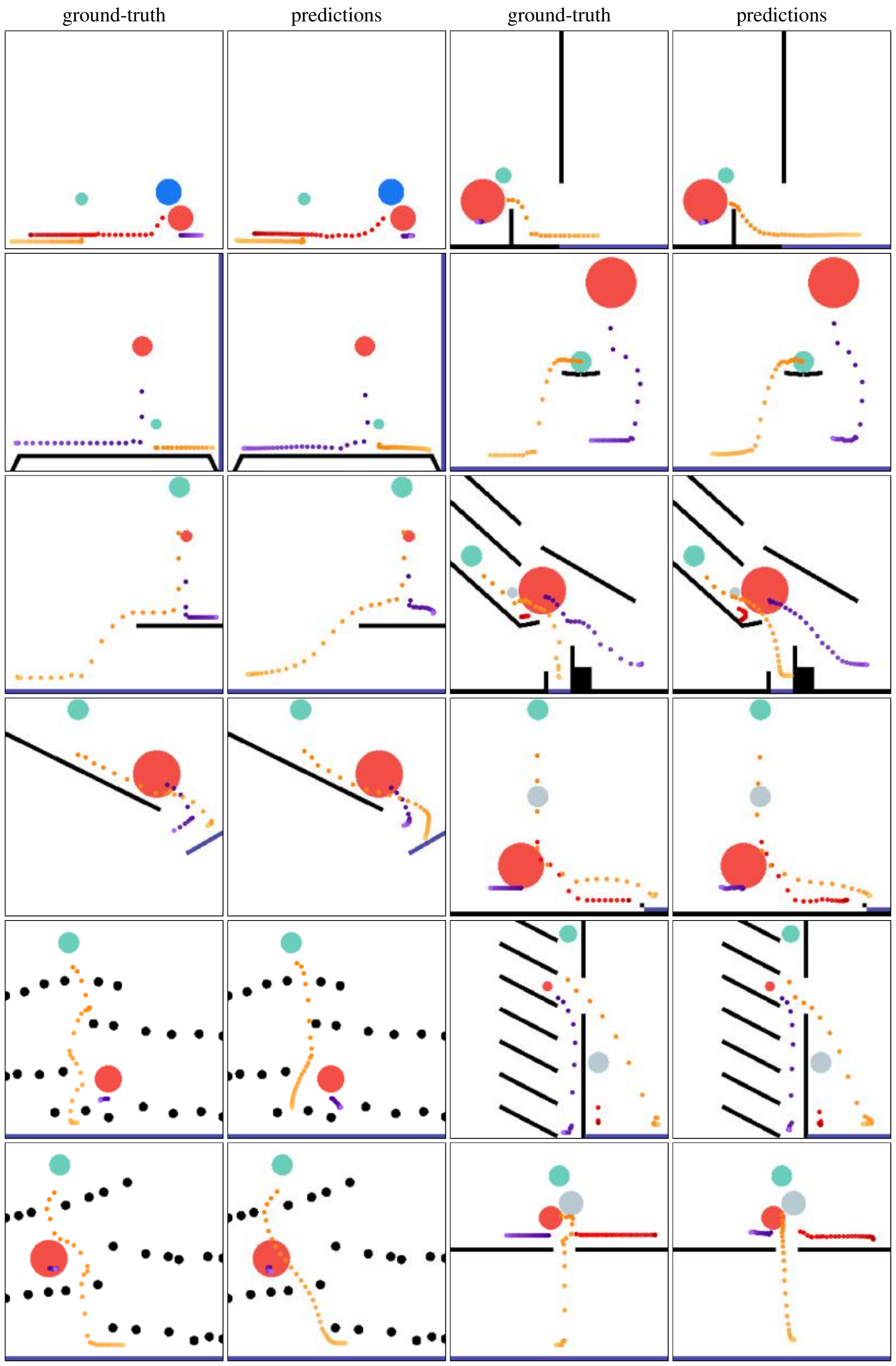


Figure 6: Visualization results of the PHYRE-W dataset. We visualize the first input image and the trajectories in future 40 timesteps.

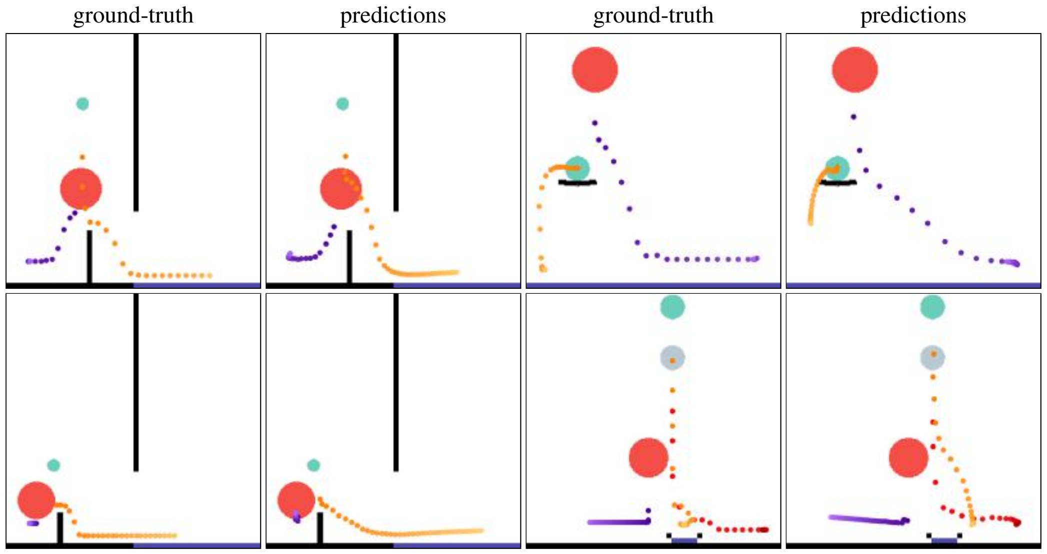


Figure 7: Visualization results of the PHYRE-C dataset. These environments are never shown in the training set. We visualize the first input image and the trajectories in future 40 timesteps.

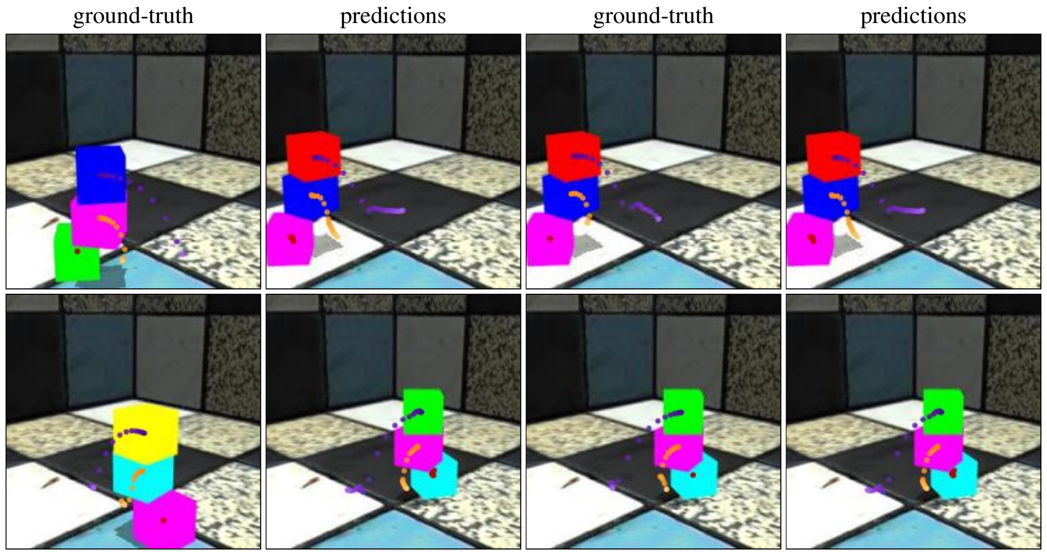


Figure 8: Visualization results of the ShapeStack dataset. We visualize the first input image and the trajectories in future 30 timesteps.