Learning 3D Object-Oriented World Models from Unlabeled Videos

Eric Crawford [1 2] **Joelle Pineau** [1 2]

Abstract

The physical world can be decomposed into discrete 3D objects. Reasoning about the world in terms of these objects may provide a number of advantages to learning agents. For example, objects interact compositionally, and this can support a strong form of generalization. Knowing properties of individual objects and rules for how those properties interact, one can predict the effects that objects will have on one another even if one has never witnessed an interaction between the types of objects in question. The promise of object-level reasoning has fueled a recent surge of interest in systems capable of learning to extract object-oriented representations from perceptual input without supervision. However, the vast majority of such systems treat objects as 2-dimensional entities, effectively ignoring their 3-dimensional nature. In the current work, we propose a probabilistic, object-oriented model equipped with the inductive bias that the world is made up of 3D objects moving through a 3D world, and make a number of structural adaptations which take advantage of that bias. In a series of experiments we show that this system is capable not only of segmenting objects from the perceptual stream, but also of extracting 3D information about objects (e.g. depth) and of tracking them through 3D space.

1. Introduction

The study of intelligent systems in nature suggests that treating the world as made up of objects is worthwhile. For example, evolution has endowed humans and many animals with built-in (or at least developmentally inevitable) machinery for discovering, tracking and reasoning about objects (Carey, 2009). Thus, in building artificially intelligent systems, we should strive to equip our systems with the ability to extract object-like representations from perceptual input, and to reason in terms of those object-like representations in a way that exploits their compositionality.

The field of deep learning has recently seen a surge of interest in neural networks that make use of explicit object-like representations. In the current work we are interested, in particular, with what we will call *object-oriented world models* (OOWM). These are probablistic models which treat the world as being made up of objects, and typically allocate separate representational resources (e.g. latent variables) for each object. The per-object variables may themselves be further disentangled; they may, for example, contain explicit representations of object properties like position, velocity, or appearance. Such models are attractive in that they jointly learn how to extract object-like representations from raw perceptual input (images or videos) while at the same time learning how objects behave and interact. They often demonstrate improved scene-modeling performance (Burgess et al., 2019), and their learned representations can be useful in downstream tasks (Veerapaneni et al., 2019).

The majority of existing OOWMs fall into one of two camps: *scene-mixture models* and *spatial attention models* (using terminology from (Lin et al., 2020)), which differ primarily in how the latent object representations are grounded in visual input. Scene-mixture models represent each object with a soft, image-sized ownership mask, while spatial attention models model an object using a 2D bounding box. One salient feature of both these styles of representation is that they are fundamentally two-dimensional. In the current work, we aim to go beyond this assumption, proposing an OOWM that explicitly models objects as 3D entities existing in a 3D world.

Embracing the 3D nature of the world has a number of potential advantages for an OOWM. First, it enables the extraction of 3D information about objects (e.g. depth), which may be required for performing certain downstream tasks, and which is largely invisible to 2D models. Moreover, acknowledging that objects exist in a 3D world may support richer inferences, such as making reasonable predictions about what happens to objects that pass out of view (e.g. they keep moving in the same 3-dimensional direction until some external force intervenes).

In the current work we propose an OOWM that models dy-

[1]McGill University [2]Mila. Correspondence to: Eric Crawford <eric.crawford@mail.mcgill.ca>.

Workshop on Object-Oriented Learning at ICML 2020.

namic 3D objects moving through static 3D environments. For modeling dynamic objects, we employ a representation which disentangles 3D object position from other object properties. For modeling the static elements of a 3D scene, we propose a solution based on recent work on learning structured representations of static 3D environments from posed videos, namely Scene Representation Networks (Sitzmann et al., 2019). Finally, we present a neural network capable of learning to extract both dynamic and static representations from posed (but otherwise unlabeled) videos.

2. Related Work

The vast majority of past work on OOWMs can be characterized as either scene-mixture models (Greff et al., 2017; van Steenkiste et al., 2018; Greff et al., 2019; Burgess et al., 2019; Engelcke et al., 2019; Kipf et al., 2019; Veerapaneni et al., 2019) or spatial attention models (Eslami et al., 2016; Kosiorek et al., 2018; He et al., 2018; Crawford & Pineau, 2019; 2020; Jiang et al., 2019; Wang et al., 2019), and some works even use both kinds of representation (Zhu et al., 2019; Lin et al., 2020). However, all these works effectively treat objects as 2D entities on a 2D plane. While a few of them do attempt to infer the relative depth of objects (Crawford & Pineau, 2019; Veerapaneni et al., 2019), this is only used to to determine z-order when rendering the objects to an output image, and does not constitute a true 3D representation. To date, the only OOWM that we are aware of with a truly 3D notion of objects is (Chen & Ahn, 2020). However, that model does not allow for moving objects, and does not employ a structured 3D representation of the non-object elements of a scene (e.g. walls, floors). It may also have scaling issues: it allocates a separate internal object slot for every cell of a 3D grid superimposed over the scene, so its runtime scales with the scene volume. BlockGAN is a related model with 3D objects (Nguyen-Phuoc et al., 2020); however, it only provides for scene generation, and cannot learn to detect objects from perceptual input.

3. Problem Setting

We assume episodic interactions with an environment with a particular structure. Let $\{S_1, \ldots, S_N\}$ be a set of static 3D scenes. At the start of each episode, a scene index $n \in \{1, \ldots, N\}$ is sampled uniformly at random, yielding static scene S_n. The sampled scene is then populated with dynamic elements, specifically a camera and a set of objects. For timestep t, let C_t be the camera pose, and let $O_t = \{o_{1,t}, \ldots o_{k,t}\}$ be the set of dynamic objects, where $o_{i,t}$ captures attributes of the i-th object. At the beginning of the episode, we sample an initial set of dynamic elements $C_0, O_0 \sim P_0(C_0, O_0|S_n)$. The objects are then propagated forward in time by sampling $C_t, O_t \sim P(C_t, O_t|C_{t-1}, O_{t-1}, S_n)$. That these distributions condition on S_n represents the fact that the dynamics of objects are determined in part by static elements; for example, moving objects may be obstructed by walls. Each timestep, a pinhole camera model F is used to render an image $I_t = F(C_t, O_t, S_n)$.

We will explore a setting in which an agent is trying to learn disentangled representations of the objects and static scene elements that it encounters. Thus we assume distributions P_0 and P governing the dynamic elements are unknown, as are the static scene representations S_n. For each episode, the agent has access only to the static index n for the episode, as well as I_t and C_t for each timestep. Overall the training data for an episode is:

$$n, (I_0, C_0), (I_1, C_1), \ldots, (I_{T-1}, C_{T-1}) \tag{1}$$

The fixed set of static scenes for training may be regarded as a low-complexity "nursery" environment in which the agent can discover objects and their properties, in preparation for operating in more complex environments in the future.

4. Learning to Model Static Scene Elements

For modeling static elements of a scene we employ the framework of Scene Representation Networks (SRN) (Sitzmann et al., 2019). Each static 3D scene S_n is modeled as a differentiable, parameterized function $\phi_n : \mathbb{R}^3 \to \mathbb{R}^f$ (e.g. multi-layer perceptron (MLP)) which maps points in 3D space to feature vectors.

The SRN framework provides a differentiable raymarching algorithm R which can be used to render the scene represented by function ϕ_n from the viewpoint of a given camera pose. We can use this algorithm, along with our posed training videos, to train ϕ_n to represent S_n. Letting $B_{(t)} = R(C_{(t)}, \phi_n)$, we minimize $\sum_{t=0}^{T-1} \mathcal{L}(B_{(t)}, I_{(t)})$ (where $\mathcal{L}$ is some image loss) using gradient descent. To generate the networks ϕ_n, we use a hypernetwork (a neural network that outputs neural networks) mapping from a scene index to an SRN, i.e. $\phi_n = H(n)$ (so in training ϕ_n we are actually training the weights of H).

One convenient feature of SRNs is that they lack any means of representing dynamic or contingent scene elements, e.g. objects that move or change appearance over time, or even static objects which are not present every episode. Notice, in particular, that the functions ϕ_n only take a spatial coordinate as input; the features assigned by the network to a given 3D location in a given scene are not permitted to vary with time or episode number. This property is useful to us because the SRN will only be able to account for static scene elements, while incurring a large error at pixels that belong to dynamic objects. We can thus think of the images rendered by the SRNs as backgrounds for our training videos. For handling the dynamic objects, we need a dif-

ferent kind of representation and accompanying inference network, outlined in the next section.

5. Learning to Model Dynamic Objects

We designed a novel neural network to handle the task of discovering and tracking dynamic objects. This network is inspired by SQAIR (Kosiorek et al., 2018), SILOT (Crawford & Pineau, 2020) and SCALOR (Jiang et al., 2019). However, whereas those models conceive of objects as 2D bounding boxes on a 2D plane, our proposed network models objects as 3D entities moving through a 3D world (only a fraction of which will be in view at any given time).

The primary data type in our network is a set of objects, each set having space for $K \in \mathbb{N}$ object slots. Each object set contains a collection of named variables, called attributes, and attributes can be indexed by slot k and timestep t. For a generic object set $\mathcal{O}$, the main attributes are:

$$\mathcal{O}^{\text{where-3D}}_{k,(t)} \in \mathbb{R}^3 \qquad \mathcal{O}^{\text{what}}_{k,(t)} \in \mathbb{R}^A \qquad \mathcal{O}^{\text{pres}}_{k,(t)} \in [0,1]$$

$\mathcal{O}^{\text{where-3D}}_{k,(t)}$ gives the 3D position of object k at time t, $\mathcal{O}^{\text{what}}_{k,(t)}$ is an unstructured vector capturing other features such as appearance, and $\mathcal{O}^{\text{pres}}_{k,(t)}$ specifies the extent to which the object exists, or is turned on. Different object sets may omit some of these attributes or add others.

Our network is composed of a number of modules (discovery, propagation and rendering as first proposed in SQAIR (Kosiorek et al., 2018), plus an additional selection module), each running once per timestep of the input video. Modules communicate by passing around object sets. We define three primary object sets: discovered objects $\mathcal{D}$, propagated objects $\mathcal{P}$ and selected objects $\mathcal{S}$. The flow of computation is depicted in Fig. 1, and runs as follows:

1. **Propagation** takes in selected objects from the previous timestep $\mathcal{S}_{(t-1)}$, and updates them using information from the new frame $I_{(t)}$ and camera pose $C_{(t)}$, yielding propagated objects $\mathcal{P}_{(t)}$.
2. **Discovery** takes in input frame $I_{(t)}$ and predicts a set of 3D objects for the frame. This module conditions on $\mathcal{P}_{(t)}$ which helps it to avoid discovering objects that are already accounted for by some object in $\mathcal{P}_{(t)}$. The resulting objects are $\mathcal{D}_{(t)}$.
3. **Selection** takes in $\mathcal{P}_{(t)}$ and $\mathcal{D}_{(t)}$, and selects the K objects with largest values for the *pres* attribute (ensuring that the number of objects propagated forward stays constant over time), yielding selected objects $\mathcal{S}_{(t)}$.
4. **Rendering** takes in selected objects $\mathcal{S}_{(t)}$ and predicts the 2D location, size, and appearance of all visible objects given camera pose $C_{(t)}$. These are blended with the background image $B_{(t)}$ rendered by the SRN to produce a final output image $\widehat{I}_{(t)}$.

We also have one additional module, Prior Propagation (also present in SQAIR), which is not directly part of the object tracking flow. This module is similar to Propagation (and is in fact trained to give the same results as Propagation), except that it does not have access to the input frames. Prior Propagation is thus forced to learn to *predict* object trajectories, rather than *tracking* them. This can support additional kinds of inference such as imagining future rollouts, which may be useful in planning. Prior propagation conditions its predictions on features of ϕ_n in a local neighbourhood around each object, which should allow it to predict interactions between dynamic objects and the static environment (e.g. objects bouncing off of walls). This is one additional benefit of using a structured 3D representation of the static scene.

Additional details on the implementation of each module are provided in Section C.

6. Training

To maximize learning stability, we train the network in 3 stages. In the first stage, we train the SRN network that models the static scene elements, specifically the hypernetwork (which outputs ϕ_n) and learnable elements of the raymarching algorithm R. After this phase, the network is able to achieve low reconstruction error for pixels that belong to static elements, but is unable to model dynamic elements. In the second stage, we train only the Discovery and Rendering modules, hooking the output of the former up to the input of the latter (essentially constituting a SPAIR network (Crawford & Pineau, 2019), roughly an autoencoder with an object-like latent representation). In this stage the network learns to segment the objects in two dimensions, but does not predict object depth and does not track objects over time (each frame is handled in isolation). Finally, in the third stage the network is fully wired up as shown in Fig 1, and trained to reconstruct the input frames. Weights of SRN, Rendering, and parts of Discovery are frozen, while Propagation, Prior Propagation and the remaining parts of Discovery are trained. In this stage the network learns to predict object depth and to track objects over time in 3D.

7. Experiments

We performed experiments in two simulated domains to test whether our method is able to learn to discover and track objects. Results are shown in Table 1 (metrics are described in Section B). Qualitative results are shown in Section A.

Stationary Objects. In our first experiment, we test our system on an environment containing stationary objects only,

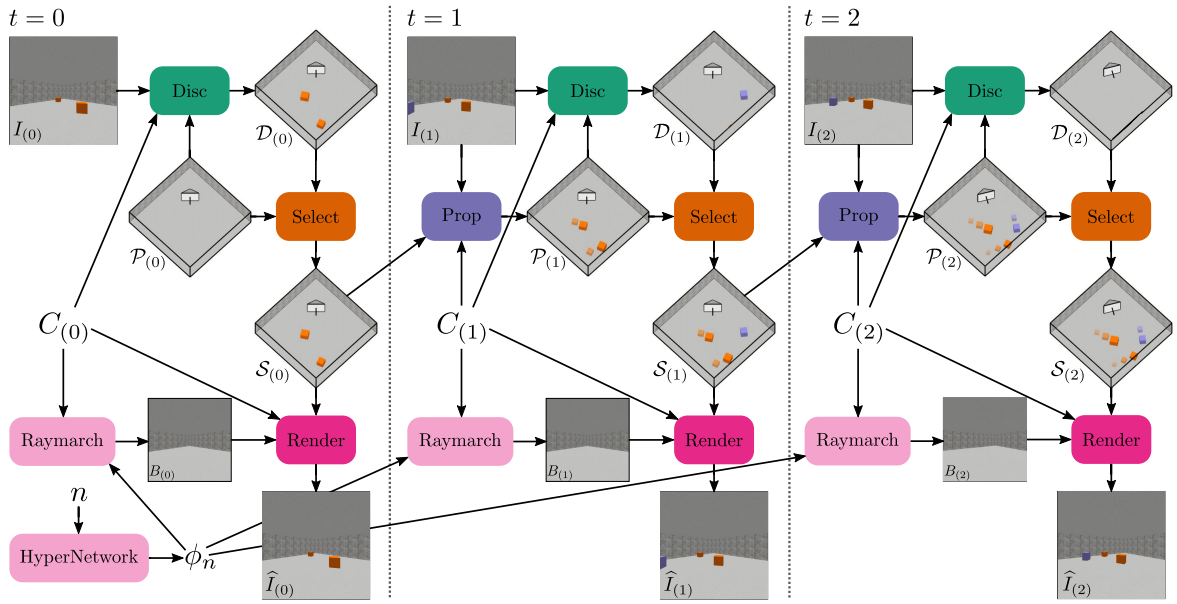

Figure 1. Schematic depicting the modules in our object discovery and tracking network. Each gray diamond represents a set of 3D internal objects. Within the gray diamonds, the white pyramid represents the (known) camera pose, while the colored blocks are objects tracked by the agent. The translucent object "tails" are meant to indicate where the objects were previously estimated to be, and correspond to perceived object motion. The Prior Propagation module is not shown here.

in order to determine whether the network is capable of segmenting objects, estimating their depth, and tracking them through 3D space in this relatively simple scenario. To test this, we generated a set of environments in Miniworld (Chevalier-Boisvert, 2018) similar to the "Health Gathering" scenario from Vizdoom (Kempka et al., 2016). To generate the dataset, we first randomly generate $N = 20$ rooms, all square but with different randomly selected textures. Each episode, one of these rooms is sampled uniformly at random and populated with "medkit" objects placed in random locations. In the Vizdoom scenario, the agent's health decreases over time, and the agent must collect medkits to stay alive. Being able to estimate the 3D positions of the objects in this scenario is highly relevant, as it would allow, for example, planning a path for collecting all medkits as fast as possible.

Moving Objects. In the next experiment we test whether our method can handle moving objects, which may present additional challanges. If objects are guaranteed to stay still, we can get a form of "temporal stereo" when the agent undergoes translational motion, which can be used as a signal about depth. When objects can move, that signal becomes weaker. Additionally, moving objects can make inter-object occlusion more common. In our moving object scenario, we first created 10 separate static environments (small mazes, rather than square rooms, to show that SRNs can handle the added complexity). Each episode we pick one at random and populate it with random colored blocks. Each object flips a coin to determine whether it moves or not. Objects that move pick a horizontal movement direction at random, bouncing off of any walls they encounter.

	AP ↑	2D MOTA ↑	3D MOTA ↑	2-norm ↓
Stationary	0.715	0.726	0.730	0.240
Moving	0.726	0.718	0.602	0.758

Table 1. Object detection and tracking metrics (described in Section B). Arrows indicate which direction is better.

8. Discussion

The quantitative results suggests that our method is generally able to segment objects, estimate their depth, and track them over time. However, inspection of qualitative results suggests a number of areas for improvement. The network is generally good at tracking object motion parallel to the camera plane (i.e. side-to-side), but poor at tracking motion where the depth of the object changes. This is likely due to the former kind of motion being much more visually obvious than the latter. The network also struggles somewhat with object permanence; for example, it often lowers *pres* values to 0 when an object passes out of sight or behind another object. To remedy this in future development, we intend to include a prior which encourages *pres* values to stay constant over time.

In follow-up work, we also intend to more fully characterize the sensitivity of the system to properties of the training data, e.g. the fraction of objects that are moving, the fraction of agent movements that are translational. We also intend to investigate the quality of the learned prior, and the extent to which the learned object trackers can generalize to more complex environments than they were trained on.

References

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Carey, S. *The origin of concepts*. Oxford University Press, 2009.

Chen, C. and Ahn, S. Object-oriented representation of 3d scenes. https://openreview.net/forum?id=BJg8_xHtPr, 2020.

Chevalier-Boisvert, M. gym-miniworld environment for openai gym. https://github.com/maximecb/gym-miniworld, 2018.

Crawford, E. and Pineau, J. Spatially invariant, unsupervised object detection with convolutional neural networks. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

Crawford, E. and Pineau, J. Exploiting spatial invariance for scalable unsupervised object tracking. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

Eslami, A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

Greff, K., van Steenkiste, S., and Schmidhuber, J. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pp. 6691–6701, 2017.

Greff, K., Kaufmann, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

He, Z., Li, J., Liu, D., He, H., and Barber, D. Tracking by animation: Unsupervised learning of multi-object attentive trackers. *arXiv preprint arXiv:1809.03137*, 2018.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2019.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, 2016.

Kipf, T., van der Pol, E., and Welling, M. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pp. 8606–8616, 2018.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*, 2020.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

Nguyen-Phuoc, T., Richardt, C., Mai, L., Yang, Y.-L., and Mitra, N. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *arXiv preprint arXiv:2002.08988*, 2020.

Redmon, J. and Farhadi, A. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pp. 1119–1130, 2019.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J. B., and Levine, S. Entity abstraction in visual model-based reinforcement learning. *arXiv preprint arXiv:1910.12827*, 2019.

Wang, D., Jamnik, M., and Lio, P. Unsupervised and interpretable scene discovery with discrete-attend-infer-repeat. *arXiv preprint arXiv:1903.06581*, 2019.

Zhu, G., Wang, J., Ren, Z., Lin, Z., and Zhang, C. Object-oriented dynamics learning through multi-level abstraction. *arXiv preprint arXiv:1904.07482*, 2019.

A. Qualitative Results

A.1. Health Gathering

Qualitative results are shown for the Health Gathering environment in the top of Fig A2. This example shows that the network is largely able to learn to segment and track these stationary objects. Looking at the overhead view (row 4 in each example), we can study some of the errors made by the network. We see the network's lack of object permanence; once objects have moved out of view, the network lowers their *pres* value to near 0 and the objects cease to be selected by the Selection module. As stated in Section 8, we expect this can be resolved by using a prior which encourages *pres* values to stay constant over time (encoding the fact that objects tend not to spontaneously disappear).

A.2. Moving Boxes

Qualitative results are shown for the Moving Boxes environment in the bottom of Fig A2. As before, looking at the overhead view can reveal some of the model's deficiencies. Consider the object assigned the gray color by the network. The network does well at tracking the projections of these objects onto the image plane, but in both cases has a difficult time tracking object motion that is perpendicular to the image plane (i.e. tracking change in depth). Improving these aspects of performance will be a target of future work.

B. Metrics

We employed 4 metrics for measuring the performance of the object tracking network.

Average Precision (AP) is a standard measure of object detection performance (Everingham et al., 2010). This assesses the quality of the bounding boxes proposed by the network, independent of the network's ability to track objects coherently over time. We use a relatively permissive IOU (Intersection over Union, a measure of how well a ground-truth bounding box is accounted for by a predicted bounding box) threshold, since we do not require the agent to output pixel-perfect bounding boxes.

Multi-object Tracking Accuracy in 2D (MOTA 2D) is a standard measure of object tracking performance (Milan et al., 2016). This assesses the networks ability to assign consistent identifiers to objects over time, as well as ability to localize objects. This measures 2D object tracking, meaning it only cares about 2D location of objects on the camera plane, and doesn't take depth into account. At each frame we only take into account ground-truth objects that are either currently visible, or were visible at some past time step. We only consider objects that have a *pres* value of at least 0.5, and use a low threshold on IOU.

Multi-object Tracking Accuracy in 3D (MOTA 3D) is a version of MOTA adapted to 3 dimensions. Since our objects are modeled only as 3D points rather than 3D bounding boxes, we use Euclidean distance to measure the distance between a ground-truth object and a predicted object. Again, we only consider objects with *pres* values at least 0.5, and consider a predicted object to be tracking a ground-truth object when the distance between them is less than 3 units (where 1 unit roughly corresponds to a meter).

2-norm. Measures the average 2-norm between ground-truth objects and the predicted objects assigned to them. Assignment is done using the Hungarian algorithm (Kuhn, 1955) on the objects' 2D bounding boxes.

C. Module Details

The idea of an Object-Oriented World Model with separate modules for discovering objects, propagating them forward in time (both with and without information from the current frame), and finally rendering them to an output image, was first introduced in SQAIR (Kosiorek et al., 2018). SCALOR (Jiang et al., 2019) and SILOT (Crawford & Pineau, 2020) subsequently made scalability-related improvements, and added the selection module. Our model can roughly be regarded as a 3D version of these past models.

Here we go into more detail about each of the modules.

C.1. Discovery

This module is very similar in structure to the convolutional networks used in supervised object detection, such as YOLO (Redmon & Farhadi, 2017). It takes in an input image and processes it with a convolutional neural network. At each cell of the output of that network, a structured 2D object representation is created, specifically a bounding box (which is required to be inside the receptive field of that convolutional cell), a presence value, and an unstructured feature vector that is intended to capture the object's appearance. A depth value is also predicted for each object, and is used along with $C_{(t)}$ to predict the object's location in 3D space.

In order to avoid rediscovering objects that are already accounted for by a propagated object, this module's prediction for the *pres* attribute conditions on nearby propagated objects, achieved using a Gaussian attention step. Each potential discovered objects "pays attention to" any propagated objects in a local neighbourhood around its 2D location on the current camera, just before predicting the *pres* attribute.

In the second stage of training (see Section 6), this network is hooked up directly to the Rendering module. The combination of these two modules essentially constitutes a SPAIR network (Crawford & Pineau, 2019), a network designed for scalable unsupervised object detection in 2D images.

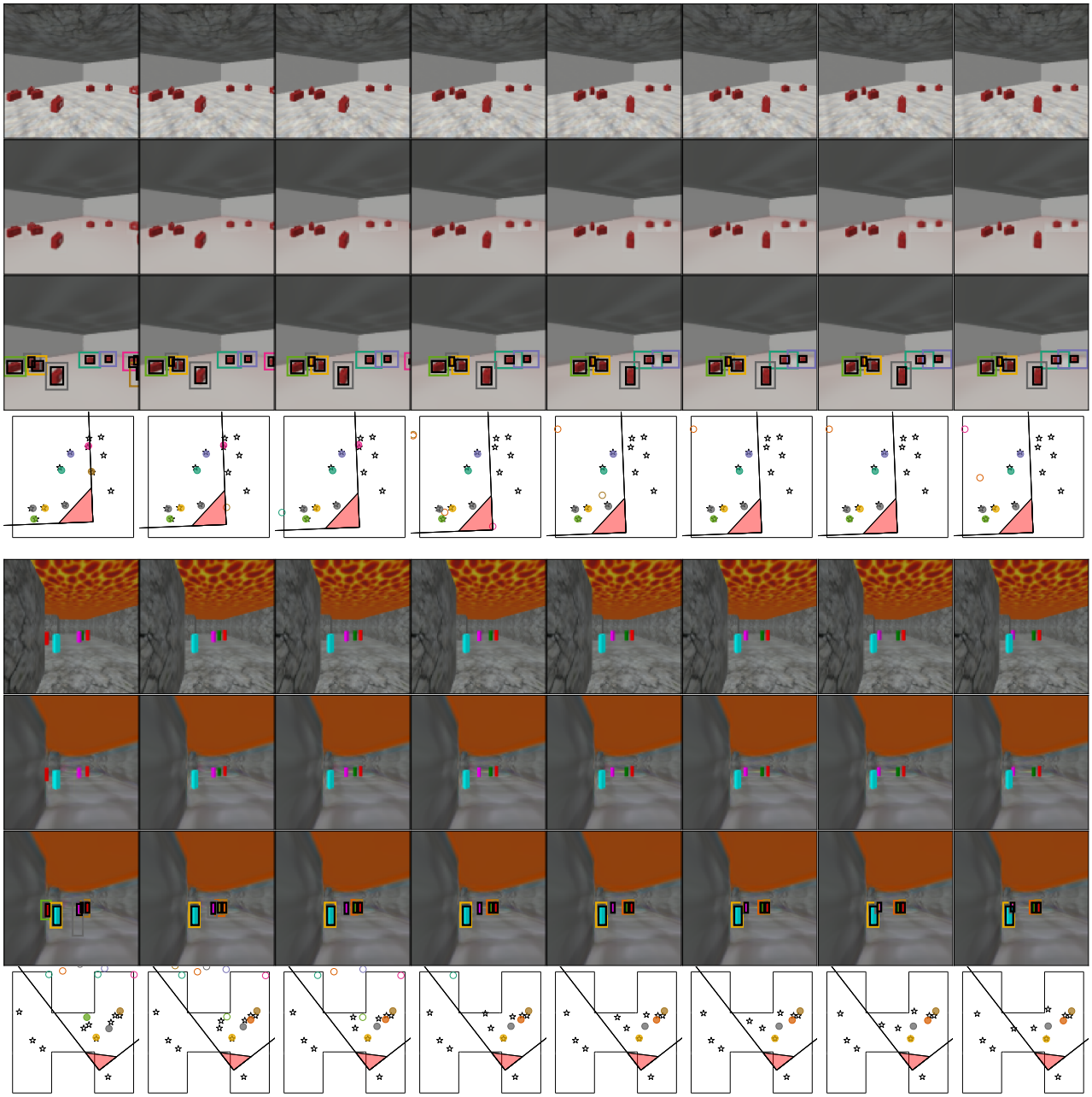

Figure A2. Qualitative results on the *Health Gathering* environment (top) and Moving Boxes environment (bottom). Timestep increases from left to right. Rows, in order from top to bottom are: 1. ground truth image, 2. reconstructed image, 3. reconstructed image with object bounding boxes, predicted (color) and ground-truth (black), 4. top-down view of environment showing objects. The red triangle is the camera. Ground-truth objects are stars, predicted objects are circles. Transparency of circles indicates *pres* value predicted by the network. Colors of predicted objects are the same in rows 3 and 4. Refer to Section A.1 for a discussion of the errors made in these examples.

C.2. Propagation

The role of the propagation module is to take in the selected objects from the previous timestep $\mathcal{S}_{(t-1)}$, and use information provided by the current frame $I_{(t)}$ to propagate those objects forward in time, yielding the set of propagated objects for the current timestep $\mathcal{P}_{(t)}$.

In simple terms, the problem solved by the propagation module is this: given the object's 3D location and appearance from the previous timestep, how does one intelligently use information contained in the new frame $I_{(t)}$ to update one's estimate of the object's location and appearance at the current timestep? One possibility would be to just condition on the entirety of the new frame; however, it is likely that the majority of the information contained in that new frame is irrelevant to updating one's estimate of a single object. Under the assumption that the object will not move far between frames, we need only pay attention to a small region of the new frame, corresponding to where the object could feasibly have moved.

More concretely, we take the 3D location of the object from the previous timestep, and project it onto the *current* (known) camera location; this essentially says where we would expect to find the object in frame $I_{(t)}$ assuming that it didn't move. Next, we delineate a region around this location where the object could conceivably have moved. To do this, we use the object size from the previous timestep, multiplied by a fixed constant. The use of the object size from the previous timestep is motivated here by the assumption that the size of an object's silhouette cannot change too much between frames provided that neither the camera nor the object are rotating too quickly (relative to the framerate at which the videos were captured). Next, we differentiably extract the pixels inside the delineated region using Spatial Transformers (Jaderberg et al., 2015), i.e. extract a glimpse. This glimpse is then processed by a convolutional neural network.

Based on the output of that network, we then predict the new 2D location of the object in a coordinate frame determined by the original glimpse's dimensions. A second glimpse is taken at this new location, and a second network processes that glimpse. Nearing the end, we autoregressively predict changes to the $what$, $pres$ and $depth$ attributes based on the information extracted from the glimpses. And, finally, we use the object's predicted 2D location, in combination with the predicted depth and known camera pose, to compute the object's new 3D location.

C.3. Selection

Selection is a simple module which takes in the discovered objects $\mathcal{D}_{(t)}$ and propagated objects $\mathcal{P}_{(t)}$, and selects the K objects with largest values for *pres*. This is intended to keep the number of objects propagated forward fixed at K as time increases. While this hard selection step is not differentiable, we have not found this to cause problems as long as K is selected to be large enough (e.g. larger than the maximum amount of objects one expects the network to have to track at any given time).

C.4. Rendering

The rendering module takes in the selected objects $\mathcal{S}_{(t)}$, finds their 2D location and size on the current camera $C_{(t)}$, and predicts an appearance for each object (i.e. a small grid of pixels, with color and alpha values at each pixel). The appearances of the different objects are then combined together, using Spatial Transformers (Jaderberg et al., 2015) to place the objects in the correct location with the correct size. Objects are rendered in order of decreasing depth, so that closer objects may occlude further objects. The objects are rendered on top of the background image $B_{(t)}$ output by the SRN raymarching algorithm. This is similar to the differentiable object-based renderers used in past work (Crawford & Pineau, 2019; 2020; Lin et al., 2020; Jiang et al., 2019).

C.5. Propagation Prior

This module is similar to the Propagation module, except that it doesn't have access to the input frames. Thus, rather than looking at the new input frame to see how the object has moved, the Propagation Prior has to make a *prediction* about where the object has moved. It can base this prediction on the object's past, features of other objects, and even on features of the local static environment defined by the structured scene representation ϕ_n. The Prior Propagation module is trained to match the predictions of the Propagation module. The Prior Propagation module is the only module that does not require knowledge of the pose of the camera; the prior predictions are made purely in 3D space, and do not need to depend on where the camera is (though, of course, we may want to allow the camera/agent to affect predictions about objects behavior; imagine a case where objects in the world chase after or flee from the agent).