Supplemental document for: Learning 3D object-centric representation through prediction

January 25, 2024

The major codes of the OPPLE model are provided together with this document.

1 Additional demonstration of OPPLE's inference and prediction

1.1 Visualization of outputs by the learned networks and prediction for the next frame

As visualized in Figure S1, Frame 2 (column 1) is the basis of prediction (frame 1 is not shown). Frame 3 (ground truth in column 2) is predicted (column 3) by combining two approaches: warping-based prediction (column 4) and imagination (column 5). The warping-based prediction is made based on the optical flow predicted from the inferred depth (column 8), object segmentation (the last 4 columns), inferred object location and velocity, and the camera's motion. In this experiment, the maximum possible number of visible objects is set to 3.

As expected it can be seen that warping-based prediction generates sharper images (column 4) than imagination-based prediction (column 5). This is because the latter is based on learned statistical regularity of environments and is only needed for small regions in the image (thus receiving less teaching signal). This relaxation of the need to encode fine details of a scene in the imagination network is advantageous because the goal of object perception is primarily to obtain an estimation of 3D information and coarse features of an object. The warping-based prediction also does not require encoding fine details of each object. The warping weights (column 6) is the total contribution of warping-based prediction towards the final prediction in each pixel.

1.2 Depth perception

We demonstrate a few example images, their ground truth depth and the depth inferred by OPPLE. Our network can capture the global 3D structure of the

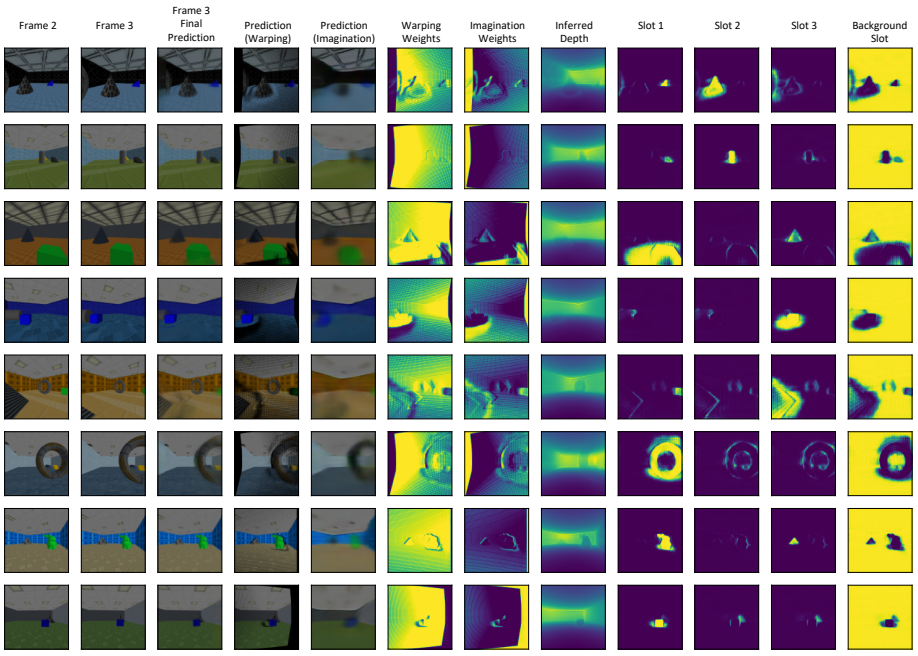

Figure S1: OPPLE's internal segmentation, depth inference and prediction for triplets in our dataset

scene, although the estimation of the depth of the objects tends to be biased towards the depth of the background behind them, consistent with the scatter plot in the main paper. We suspect that by introducing more translational motion of the camera parallel to the image plane will allow the depth network to learn more accurate depth inference for objects, as such camera movement induces strong motion parallax effect. We see evidence for this in our results on the GQN dataset, as discussed in the next section.

We would like to emphasize that the depth inference is performed on single images without the need of motion cues, unlike other unsupervised models such as O3V and Savi++ that need videos at both training and test time to learn and infer 3D information.

1.3 Performance on GQN dataset

We evaluated our model on the ROOMS dataset using the rendering code provided by [5]. The dataset is based on the GQN rooms-ring-camera dataset [3] but we allowed all motion speeds to be uniformly sampled from an interval instead of being chosen from narrow Gaussian distributions as in [5]. It appears that the depth images (column 8) are sharper and exhibit less bias for objects' depth towards longer distance than in Figure S1 (on our custom dataset). This is potentially because in this dataset, the dominant motion of the camera is

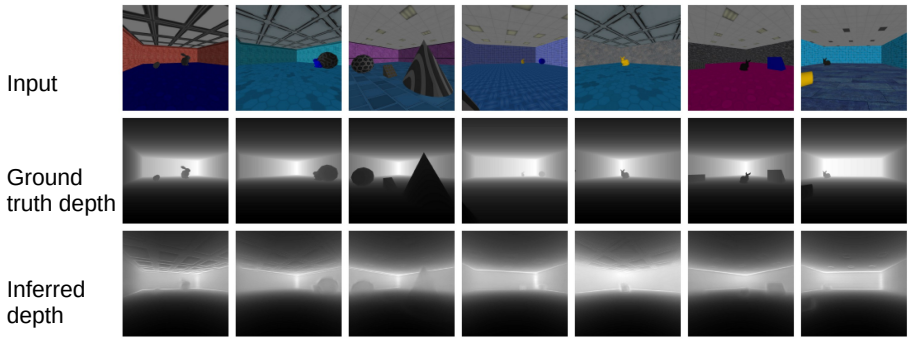

Figure S2: Comparison between ground truth depth and inferred depth

parallel to the image plane (along the ring while looking to a direction close to the center of the room), consistent with our hypothesis above. This visualization also demonstrates that although we set a maximum number of slots (objects) visible to the camera (3 in these experiments), the network is able to output empty object masks when the actual number of visible objects is smaller than the number of slots. In other words, it is not required for the network to know how many objects are in the scene. Only a maximum number of slots needs to be set for the LSTM within the object extraction network.

As shown below, the depth inference of OPPLE on GQN dataset reaches superior performance of r=0.99 between ground truth depth and inferred depth (excluding pixels in the sky, because the sky's distance is effectively infinity in the ground truth). A scatter plot of inferred vs. ground truth depth for pixels excluding the sky is shown below (the vertical stripes are likely due to rounding issue in saving depth map and down-sampling).

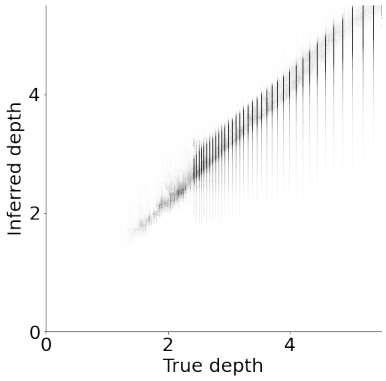

Figure S4: Performance of depth inference of OPPLE on GQN dataset.

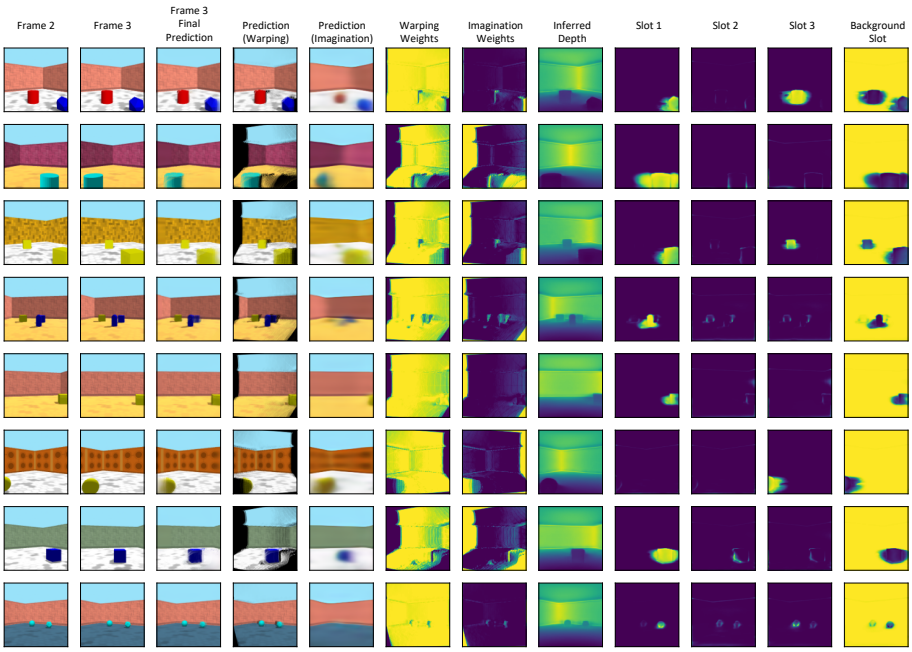

Figure S3: Visualization of the two approaches of prediction (warping and imagination), their combination weights, inferred depth, and probabilistic segmentation maps from OPPLE on scenes of moving objects in a simulated room based on the ring environment of GQN[3], based on code from [5].

2 Pseudo code of the OPPLE framework

Algorithm 1 Object Perception by Predictive LEarning (OPPLE)

initialize $f_{\theta_{\rm obj}}$, $h_{\theta_{\rm depth}}$, $g_{\theta_{\rm imag}}$

Input: images $\boldsymbol{I}^{(t-1)}, \boldsymbol{I}^{(t)}, \boldsymbol{I}^{(t+1)} \in \mathbb{R}^{w\times h\times 3}$, self-motion $\boldsymbol{v}_{\rm obs}^{(t-1)}, \omega_{\rm obs}^{(t-1)}, \boldsymbol{v}_{\rm obs}^{(t)}, \omega_{\rm obs}^{(t)}$

Output: prediction $\boldsymbol{I}'^{(t+1)}$, segmentation $\boldsymbol{\pi}_{1:K+1}^{(t-1)}$, $\boldsymbol{\pi}_{1:K+1}^{(t)}$, objects' codes $\boldsymbol{z}_{1:K}^{(t-1)}$, $\boldsymbol{z}_{1:K}^{(t)}$, objects' locations and poses $\hat{\boldsymbol{x}}_{1:K}^{(t-1)}, \boldsymbol{p}_{\phi_{1:K}}^{(t-1)}, \hat{\boldsymbol{x}}_{1:K}^{(t)}, \boldsymbol{p}_{\phi_{1:K}}^{(t)}$

for $\tau = \{t-1, t\}$ **do**

scene code $e^{(\tau)} \leftarrow$ U-NetEncoder$_{f_{\theta_{\rm obj}}}(\boldsymbol{I}^{(\tau)})$

object code $\boldsymbol{z}_{1:K}^{(\tau)}$, location $\hat{\boldsymbol{x}}_{1:K}^{(\tau)}$, pose $\boldsymbol{p}_{\phi_{1:K}}^{(\tau)} \leftarrow$ LSTM$_{f_{\theta_{\rm obj}}}(e^{(\tau)})$

background code $\boldsymbol{z}_{K+1} = 0$

depth $\boldsymbol{D}^{(\tau)} \leftarrow h_{\theta_{\rm depth}}(\boldsymbol{I}^{(\tau)})$

segmentation mask $\boldsymbol{\pi}_{1:K+1}^{(\tau)} \leftarrow$ Softmax ([U-NetDecoder$_{f_{\theta_{\rm obj}}}(\boldsymbol{I}^{(\tau)}, \boldsymbol{z}_{1:K}^{(\tau)}), 0$])

end for

object matching scores $r_{kl} \leftarrow \frac{\mathrm{RBF}(\boldsymbol{z}_k^{(t)}, \boldsymbol{z}_l^{(t-1)})}{\sum_{m=1}^{K+1}\mathrm{RBF}(\boldsymbol{z}_k^{(t)}, \boldsymbol{z}_m^{(t-1)})}, k, l \in 1:K+1$

for $k \leftarrow 1$ to K **do**

object motion $\hat{\boldsymbol{v}}_{1:K}, \boldsymbol{\omega}_{1:K} \leftarrow r_{k,l}, \hat{\boldsymbol{x}}_k^{(t)}, \hat{\boldsymbol{x}}_l^{(t-1)}, \boldsymbol{p}_{\phi_k}^{(t)}, \boldsymbol{p}_{\phi_l}^{(t-1)}, \boldsymbol{v}_{\rm obs}^{(t-1)}, \omega_{\rm obs}^{(t-1)}, l = 1:K+1$

object-specific optical flow$_k \leftarrow \hat{\boldsymbol{v}}_{1:K}, \boldsymbol{\omega}_{1:K}, \boldsymbol{v}_{\rm obs}^{(t)}, \omega_{\rm obs}^{(t)}, \boldsymbol{D}^{(t)}, \hat{\boldsymbol{x}}_k^{(t)}$

end for

warping-based prediction $\boldsymbol{I}'^{(t+1)}_{\rm warp}, \boldsymbol{W}_{\rm Warp} \leftarrow$ Warp$(\boldsymbol{I}^{(t)}, \text{optical flow}_{1:K+1}, \boldsymbol{\pi}_{1:K+1}^{(\tau)})$

imagination $\boldsymbol{I}'^{(t+1)}_{\rm imagine} \leftarrow g_{\theta_{\rm obj}}(\boldsymbol{I}^{(t)} \odot \boldsymbol{\pi}_{1:K+1}^{(t)}, \log(\boldsymbol{D}^{(t)}) \odot \boldsymbol{\pi}_{1:K+1}^{(t)}, \boldsymbol{v}_{\rm obs}, \omega_{\rm obs}, \hat{\boldsymbol{v}}_{1:K}, \hat{\boldsymbol{x}}_{1:K})$

final image prediction: $\boldsymbol{I}'^{(t+1)} \leftarrow \boldsymbol{I}'^{(t+1)}_{\rm warp}, \boldsymbol{I}'^{(t+1)}_{\rm imagine}, \boldsymbol{W}_{\rm Warp}$

update parameters: $\theta \leftarrow \theta - \gamma\nabla_\theta[|\boldsymbol{I}'^{(t+1)} - \boldsymbol{I}^{(t+1)}|^2 + \text{regularization loss}]$, $\theta \in [\theta_{\rm obj}, \theta_{\rm depth}, \theta_{\rm imag}]$

3 Network training and dataset

We trained the three networks jointly using ADAM optimization [6] with a learning rate of $3e-4$, $\epsilon = 1e-6$ and other default setting in PyTorch, with a batch size of 20. 55 epochs were trained on the dataset. We set $\lambda = 1.0$.

The model was implemented in PyTorch and trained on NVidia RTX 6000 for about 8 days. We will release the dataset upon publication of the manuscript.

In consideration that other models originally demonstrated on simple datasets with homogeneous colors may need to increase their size for our dataset, we tested MONet[1] and slot attention[7] with both the original network sizes and

increased sizes. For MONet, we tested the original number of channels ([32, 32, 64, 64]) for the hidden layers of encoder of the component VAE (IoU=0.12, ARI-fg=0.27), and increased them to [32, 64, 128, 256, 256] with one more layer while also increasing the channel number in its attention network from 64 to 128 (MONet-bigger as reported in the main text: IoU=0.20, ARI-fg=0.36). The decoder layers' sizes were increased accordingly. For slot attention, we tested a variant which increased the number of features in the attention component from 64 to 128 (slot-attention-128, which took 15 days to train) in addition to the original setting (slot-attention) and found the performance to be similar. For our model, we tuned sizes of networks based on performance on the training set.

Images in the dataset are rendered in a custom Unity environment at a resolution of 512 × 512 and downsampled to 128 × 128 resolution for training. Images are rendered in sequence of 7 time steps for each scene. In each scene, a room with newly selected textures and objects is created. The starting location and orientation of the camera and objects are initialized randomly. The camera moves with random steps and pivots with random angles between consecutive frames; each object moves with random constant velocity drawn independently. All possible valid sequential triplets out of the recorded 7 frames (e.g., frames 1,2,3; 1,3,5; 1,4,7, etc.) form our training samples, excluding the frame sets where objects collide with each other or the camera. For our dataset we use 90k scenes for training which give us 275k valid triplets.

4 Details of prediction methods

4.1 Prediction of new location and pose of an object

We first explain how to use the inferred 3D spatial states of an object in two frame to estimate its velocity and predict its location in the next frame. Following the notation in 2.1 of the main text, in order to calculate the instantaneous velocity of object k at $t-1$ from the perspective of the camera at t (with a static reference frame), we need to first calculate the 3D location that object k observed at $t-1$ should appear at t to the camera if it is static relative to the background: $\hat{x}_k^{t-1\to t} = \boldsymbol{M}_{-\omega_{\text{obs}}}^{(t-1)}(\hat{\boldsymbol{x}}_k^{(t-1)} - \boldsymbol{v}_{\text{obs}}^{(t-1)})$, where $\hat{\boldsymbol{x}}_k^{(t-1)}$ is the inferred 3D location of object k at $t-1$; $\boldsymbol{v}_{\text{obs}}^{(t-1)}$ is the velocity of the camera from $t-1$ to t viewed from a static reference frame centered at its original location; and $\boldsymbol{M}_{-\omega_{\text{obs}}}^{(t-1)}$ is the rotation matrix $\begin{bmatrix} \cos\omega_{\text{obs}}^{(t-1)} & \sin\omega_{\text{obs}}^{(t-1)} & 0 \\ -\sin\omega_{\text{obs}}^{(t-1)} & \cos\omega_{\text{obs}}^{(t-1)} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ induced by the rotational speed of the camera from $t-1$ to t. Thus, the spatial offset from any object j at $t-1$ to object k at t, as viewed from the camera, is $\hat{\boldsymbol{v}}_{j\to k}^{(t)} = \hat{x}_j^{(t)} - \hat{x}_j^{t-1\to t}$. We calculate the velocity of object k as the weighted average of the spatial offset against all candidate objects detected at $t-1$ using the matching scores r_{kj} between object k and those candidate objects: $\hat{\boldsymbol{v}}_k^{(t)} = \sum_{j=0}^{K} r_{kj}\hat{\boldsymbol{v}}_{j\to k}^{(t)}$

The spatial location where the object k observed at t would be at $t+1$ can be

predicted as $\boldsymbol{x}'^{(t+1)}_k = \boldsymbol{M}^{(t)}_{-\omega_{\text{obs}}}(\hat{\boldsymbol{x}}^{(t)}_k + \hat{\boldsymbol{v}}^{(t)}_k - \boldsymbol{v}^{(t)}_{\text{obs}})$, with $\boldsymbol{v}^{(t)}_{\text{obs}}$ and $\boldsymbol{M}^{(t)}_{-\omega_{\text{obs}}}$ defined similarly as above, but for camera movement between t and $t+1$.

As the matrix multiplication involved in this prediction is almost impossible to be hard-wired in the brain, we also tested the possibility of learning to approximate this computation with a fully-connected neural network, as is further discussed in section 5 of this Supplementary.

As described in the main text, we use $\boldsymbol{p}^{(t)}_{\phi_k} = [p^{(t)}_{\phi_{k,1}}, p^{(t)}_{\phi_{k,2}}, \cdots, p^{(t)}_{\phi_{k,b}}]$ to represent the probabilities that the pose of object k at t falls into b equally spaced bins of angles in $(0, 2\pi)$, where $\sum_i p^{(t)}_{\phi_{k,i}} = 1$. With this notation, if we consider b possible discrete rotational speeds $-\frac{(b-2)\pi}{b}, -\frac{(b-4)\pi}{b}, \cdots, 0, \cdots, \frac{b\pi}{b}$, the probability that the rotational speed $\omega^{(t)}_{j\to k}$ is equal to any γ_1 among these discrete values if the same object is assigned index j at $t-1$ and k at t is $p(\omega^{(t)}_{j\to k} = \gamma_1) \propto \sum_{l,m} p^{(t-1)}_{\phi_{j,l}} p^{(t-1)}_{\phi_{k,m}} \mathbb{1}(\frac{(m-l)2\pi}{b} \in \{\gamma_1 - 2\pi, \gamma_1, \gamma_1 + 2\pi\})$. We then calculate the likelihood of rotational speed for object k similarly by weighting the likelihoods calculated for all possible candidate pairs j, k with their matching scores.

The probability that the pose of object k is equal to any possible discrete yaw angle bin γ_2 at $t+1$ can be predicted through $p'(\phi^{(t+1)}_k + \omega^{(t)}_{\text{obs}} = \gamma_2) = \sum_{\substack{\gamma_1,\omega,\\ \gamma_2-\gamma_1\in\{\omega-2\pi,\omega,\omega+2\pi\}}} p(\omega^{(t)}_{j\to k} = \omega) p(\phi^{(t)}_k = \gamma_1)$, where $\omega^{(t)}_{\text{obs}}$ is the angular velocity of the observer. Since we model $\phi^{(t)}_k$ and $\omega^{(t)}_k$ as probability distributions on discrete bins of angle while ω_{obs} can take continuous values, we convert $p'(\phi^{(t+1)}_k + \omega_{\text{obs}})$ to $p'(\phi^{(t+1)}_k)$ on the same set of bins as $\phi^{(t)}_k$ by interpolation with a Von Mises kernel.

For the purpose of obtaining a point estimation of the rotational speed of ω_k to be used in predicting pixel-wise optical flow, we multiply the estimated probability for each angle bin with a vector pointing to the direction of that angle and take the angle of the sum of all these products (resembling the population vector code commonly observed in the brain for angular variables such as arm reaching direction or visual motion direction[8]).

4.2 Camera intrinsic

The Depth Perception network uses visual features in the image $\boldsymbol{I}^{(t)}$ to infer the distance of all pixels to the camera (depth) $\boldsymbol{D}^{(t)} \in \mathbb{R}^{w\times h} = h_\theta(\boldsymbol{I}^{(t)})$. With the inferred depth $D^{(t)}(i,j)$ for a pixel at any coordinate (i,j) in the image and the focal length f of the camera, the pixel's 3D location as viewed by the camera can be determined as $\hat{\boldsymbol{m}}^{(t)}_{(i,j)} = \frac{D^{(t)}(i,j)}{\sqrt{i^2+j^2+f^2}} \cdot [i, f, j]$. Here, we take the pixel coordinate of the center of an image as (0,0) and pixel indices are assumed to be symmetric around the center of the image: $|i| \le \frac{w-1}{2}$,$|j| \le \frac{h-1}{2}$. Conversely, given a 3D coordinate location x, y, z relative to the camera, the pixel coordinate it appears on the image can be calculated as well: $[i, j] = \frac{f}{y}[x, z]$. And its depth $d = \sqrt{x^2 + y^2 + z^2}$. This is the knowledge of camera intrinsic, which we assume

is known. However, since these are relatively simple mapping functions, we think it is possible to learn to approximate them with neural networks together with other networks in future works.

4.3 Weights in warping-based prediction and combination of two streams of predictions

The details of the procedure of predicting where a pixel at t would land at $t+1$ on the image has been detailed in the main text. Here we describe the calculation of the weights when drawing the colors for every pixel at $t+1$.

As the object attribution of each pixel is not known but is inferred by $f_{\text{obj}}(\boldsymbol{I}^{(t)})$, it is represented for every pixel as a probability of belonging to each object and the background $\boldsymbol{\pi}_k^{(t)}$, $k = 1, 2, \cdots, K+1$. Therefore, the predicted motion of each pixel should be described as a probability distribution over $K+1$ discrete target locations $p(\boldsymbol{m'}_{(i,j)}^{(t+1)}) = \sum_{k=1}^{K+1} \pi_{kij}^{(t)} \cdot \delta(\boldsymbol{m'}_{k,(i,j)}^{(t+1)})$, i.e., pixel (i,j) has a probability of $\pi_{kij}^{(t)}$ to move to location $\boldsymbol{m'}_{k,(i,j)}^{(t+1)}$ at the next time point, for $k = 1, 2, \cdots, K+1$. With such probabilistic prediction of pixel movement for all visible pixel $(i,j)^{(t)}$, we can partially predict the colors of the next image at the pixel grids where some original pixels from the current view will land nearby by weighting their contribution:

$$\boldsymbol{I'}_{\text{Warp}}^{(t+1)}(p,q) = \begin{cases} \frac{\sum_{k,i,j} w_k(i,j,p,q) I^{(t)}(i,j)}{\sum_{k,i,j} w_k(i,j,p,q)}, & \text{if } \sum_{k,i,j} w_k(i,j,p,q) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We define the weight of the contribution from any source pixel (i,j) to a target pixel (p,q) as

$$w_k(i,j,p,q) = \pi_{kij}^{(t)} \cdot e^{-\beta \cdot {D'}_k^{(t+1)}(i,j)} \cdot \max\{1 - |{i'}_k^{(t+1)} - p|, 0\} \cdot \max\{1 - |{j'}_k^{(t+1)} - q|, 0\} \tag{2}$$

Here, $({i'}_k^{(t+1)}, {j'}_k^{(t+1)})$ is the pixel coordinate corresponding to the predicted new 3D location $\boldsymbol{m'}_{k,(i,j)}^{(t+1)}$ for pixel (i,j) from the current frame. The first term $\pi_{kij}^{(t)}$ in the definition incorporates the uncertainty of which object a pixel belongs to. The second term $e^{-\beta \cdot {D'}_k^{(t+1)}(i,j)}$ resolves the issue of occlusion when multiple pixels are predicted to move close to the same pixel grid by down-weighting the pixels predicted to land farther from the camera. These last two terms mean that only the source pixels predicted to land within a square of of 2×2 pixels centered at any target location (p,q) will contribute to the color $I'^{(t+1)}_{\text{Warp}}(p,q)$. The depth map $\boldsymbol{D'}_{\text{Warp}}^{(t+1)}$ can be predicted by the same weighting scheme after replacing $I^{(t)}(i,j)$ with each predicted depth ${D'}_k^{(t+1)}(i,j)$ assuming the pixel belongs to object k.

Note that the formula above guarantees that the weights from all contributing original pixels sum to 1. However, because we assume that only a portion $\pi_{kij}^{(t)}$ of a pixel might land nearby a target pixel, the sum of all the portions of the

pixels that would land in the 2×2 square near a target pixel may be less than 1, in which case, the above prediction is combined with the prediction based on imagination to form the final prediction for such target pixels. Therefore, the weight on warping-based prediction is calculated as $\boldsymbol{W}_{\text{warp}}(p,q) = \min\{1, \pi_{kij}^{(t)} \cdot \max\{1 - |i'^{(t+1)}_k - p|, 0\} \cdot \max\{1 - |j'^{(t+1)}_k - q|, 0\}\}$.

5 Jointly learning rules of rigid body motion and apparent movement of objects induced by ego-motion

As mentioned above, the expected apparent change of 3D location of a static object induced by the motion of the camera itself, and the expected movement of all pixels on an object given the object's 3D location and velocity and pixels' locations (rigid-body motion) are assumed to be known in our main model. However, it is unnatural to assume these are hard-wired in the brain. To test whether they can be jointly learned together with all other networks by optimizing for the same objective, we approximated them with neural networks.

We used a series of 3 fully-connected layers, each with 16 units, with a residual connection from input to output, to learn the function that maps from an object's location $\hat{\boldsymbol{x}}_k^{(t-1)}$, the camera's horizontal movement velocity $\boldsymbol{v}_{\text{obs}}^{(t-1)}$ and rotational speed ω_{obs} to the 3D location where the object would appear after the camera's motion: $\hat{x}_k^{t-1\to t}$. We then used two convolutional networks, each with channel sizes [64,64,32,4] and a residual connection from input to output layer, to approximate the prediction of pixels' 3D motion. One network is dedicated for the background and the other for objects. The network for predicting pixels belonging to objects takes as input a pixel's inferred 3D location, the location of the object it belongs to, the camera's motion parameters, the object's inferred 3D location and its velocity and rotational speed, and then predicts the new 3D location of the pixel in the next frame. The network for predicting background pixels' motion only takes the pixels' inferred 3D locations and camera's motion parameters as input.

By jointly optimizing all networks for the same objective function, we found that the relaxed model obtains a segmentation performance of ARI-fg=0.58, IoU=0.44, which is on par with the full model (ARI-fg=0.58, IoU=0.45) (Figure S5). Further we found that the objects' 3D locations are also still inferred well when the objects are segmented well (Figure S6). The correlation between ground truth polar angle and inferred polar angle for objects segmented with IoU>0.5 (mainly red dots) is 0.79 (while the correlation of all objects regardless of whether they can be segmented correctly by the model is 0.62). The correlation between the ground truth distance and the inferred distance for objects with IoU>0.5 is 0.48 (while the correlation without such filtering is -0.01). In regard to depth inference, we find that this network struggles to learn the background depth, which is reflected in the decrease in the overall correlation to the ground truth

depth ($r = 0.33$)

As discussed in the main text, this suggests that the assumption of rigid body motion may not be strictly necessary for segmentation and 3D localization of objects together with depth perception. Note that we did make an assumption that objects move with inertia (keeping their speed) as is true in the data. Future works may test whether this assumption still allows learning all the capacities when objects actually change speed (e.g, due to collision or gravity).

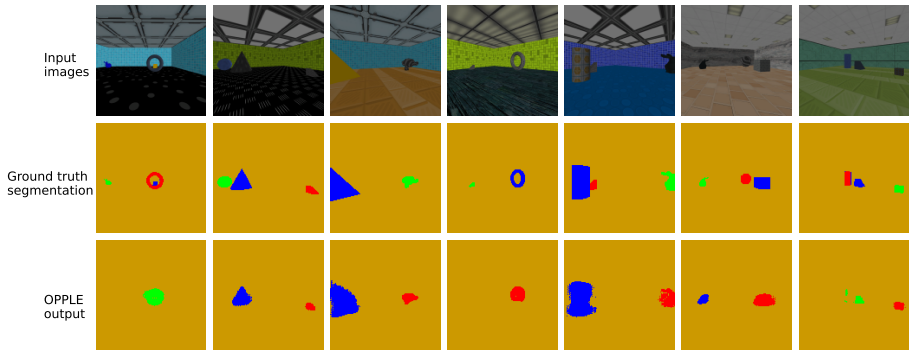

Figure S5: Segmentation performance for relaxed model (the rule of rigid body motion and apparent object motion induced by egomotion are learned).

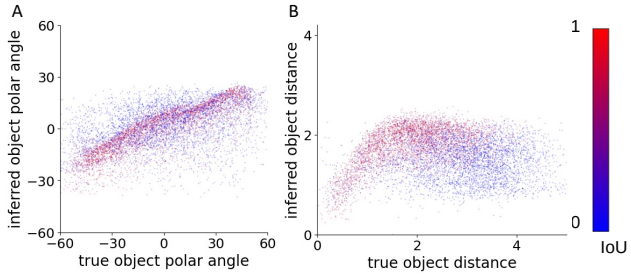

Figure S6: Performance of object 3D localization color-coded by segmentation performance for all objects in the relaxed model.

6 Performance of OPPLE on MOVi dataset

The recently published MOVi dataset [4] poses more challenges to OCRL models. It introduces more complex and realistic object motion following gravity and including interaction between objects. We therefore also test OPPLE on the most challenging version of MOVi dataset (MOVi-E) tested in a previous work SAVi++ [2]. We generated scenes including 3 objects following the default data generation procedure. In order to incorporate 3D object rotation, we made modification to the object extraction network such that it outputs cosines and

sines of each Euler angle characterizing the current pose (α, β, γ) of each object viewed by the camera. These values are used to compute a rotation matrix corresponding to the pose. Then, using the rule of apparent object movement induced by camera motion that incorporates 3D rotation, the pose of any object at $t-1$ that would appear at t if it did not move is calculated. This allows the calculation of the rotation matrix for any pair of detected objects between $t-1$ and t. Using the same weighting introduced in Section 4.3, the expected rotation for each object from t to $t+1$ is calculated with an assumption that the rotation from $t-1$ to t carries over.

On testing data of MOVi-E, OPPLE achieved a slightly lower segmentation performance (ARF-fg=0.38 and IoU=0.44) than on our custom dataset. Figure S7 illustrates the segmentation performance on MOVi-E dataset. It can be seen although overall the union of all segmentation masks often cover the majority of the true object masks, the model often attribute different objects into the same mask. We think the lower performance in segmentation may be due to the difficulty of learning a to assign a pose to each object relative to a canonical pose that is consistent to all objects. Additionally, because all objects follow gravitational force and fall at a similar time, their motions are often synchronized and the speed may thus appear close to each other, which encourages our network to attribute them to the same objects.

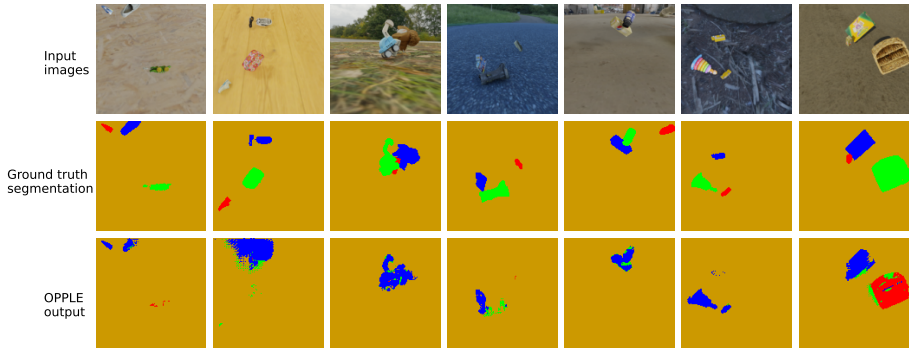

Figure S7: OPPLE's segmentation performance on 3-object version of MOVi-E dataset.

7 Depth perception performance of O3V model on our dataset

As shown below, the O3V model generates blurrier depth maps for our dataset and often fails to capture the relative distance relationship between objects and background, even though it has the advantage of using 3 consecutive frames to infer 3D scene structure. In contrast, our model can make inference of depth based on a single image and produce more accurate depth estimation (Figure S1).

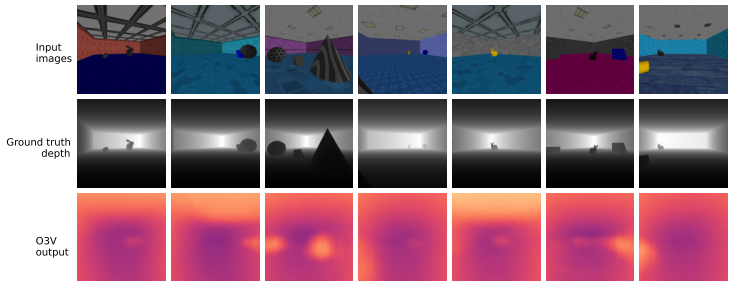

Figure S8: Example frames (first row), the ground truth depth map (second row) and the depth map inferred by O3V (third row).

References

[1] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[2] Gamaleldin F Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *arXiv preprint arXiv:2206.07764*, 2022.

[3] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[4] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022.

[5] Paul Henderson and Christoph H Lampert. Unsupervised object-centric video generation and decomposition in 3d. *Advances in Neural Information Processing Systems*, 33:3106–3117, 2020.

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

[7] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[8] Alexandre Pouget, Peter Dayan, and Richard Zemel. Information processing with population codes. *Nature Reviews Neuroscience*, 1(2):125–132, 2000.