Supplemental Material for "Unsupervised Segmentation in Real-World Images via Spelke Object Inference"

Honglin Chen[1], Rahul Venkatesh[1], Yoni Friedman[4], Jiajun Wu[1], Joshua B. Tenenbaum[4], Daniel L. K. Yamins[1,2,3]**, Daniel M. Bear[2,3]**

[1] Department of Computer Science, Stanford
[2] Department of Psychology, Stanford
[3] Wu Tsai Neurosciences Institute, Stanford
[4] Department of Brain and Cognitive Sciences and CBMM, MIT

A Additional Methods

A.1 Details and Extensions of EISEN

Learning a prior over Spelke objects. Relational supervision is natural for motion-based learning in part because motion is sparse. But the output segments of a well-trained EISEN are *not* sparse, so they can be used for learning *non*-relational features of Spelke objects. For instance, an image patch on the nearer side of a depth edge may "look like" it lies on the interior of an object segment – a cue known as *border ownership*. EISEN can take advantage of these non-relational features by learning a nonrandom, pixelwise initialization for KProp.

Specifically, we let the Q channels of the plateau map code for possible object centroid locations. Let Q_H, Q_W be the encoding resolutions of height and width, with $Q_H \cdot Q_W = Q$. Then the *centroid encoding* is constructed by creating the $Q_H \times Q_W \times Q$ feature tensor q defined by

$$q_{ijk} = 1 \textbf{ if } (k \ == \ iQ_W + j) \textbf{ else } 0, \tag{1}$$

then bilinearly upsampling this tensor from size (Q_H, Q_W, Q) to the usual plateau map resolution $(H', W', Q) = (H/4, W/4, Q)$ and ℓ^2 normalizing along the channel dimension. In this encoding, there is not just a ground truth segmentation map but also a ground truth plateau map – namely, the one in which each feature vector has a value equal to the encoded centroid of the true object segment it belongs to.

We train two modified RAFT networks[5] to (1) classify whether each pixel belongs in an EISEN-predicted Spelke object or not and, if it belongs to an object, (2) predict the relative offset between that pixel's location and the centroid of the

[1] ** = Equal senior authorship

[5] We simply replace the output head that predicts predicts a $H \times W \times 2$ flow map with one that predicts the $H \times W \times 1$ "objectness" logits or $H \times W \times 2$ centroid offsets.

object segment it belongs to. The plateau map initialization h_0 is then given by the "objectness"-masked, predicted centroid encodings of each pixel. In a loose analogy between KProp and Ising-like models of magnetic dipole dynamics, this initialization plays the role of a pulsed external field.

Interestingly, learning a KProp initialization does not improve quantitative results on the **Playroom** dataset, though in some cases it appears to help discover an object that is only partially segmented with random plateau map initialization (Figure S1.) It may be that the learned initialization is helpful in some ways but harmful in others, such as by degrading fine details. It is notable that passing the learned initializations directly through Competition without running any iterations of KProp (Figure S1, *-KProp*) performs better than all baselines, achieving an mIoU of 0.660 on the **Playroom** *val* set. This suggests that EISEN can best take advantage of *non-relational* cues to segment these scenes, but that there may be a low upper bound to performance when relational cues are not used.

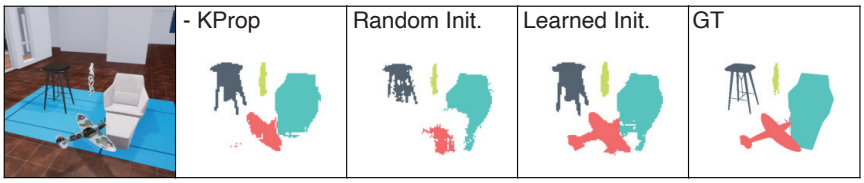

Fig. S1: **Learning a Spelke object prior for KProp.**

Constructing and segmenting a flow plateau map. The procedure for creating a flow plateau map to explain away background motion (see Figure S2) is similar to constructing the centroid encoding described above. Given a $H \times W \times 2$ optical flow map $\mathcal{F}$, we linearly normalize all flow values $(\mathcal{F}_x, \mathcal{F}_y)$ to the range $[-1, 1] \times [-1, 1]$. Then, each pixel's flow vector $\mathcal{F}_{ij}$ is embedded in a Q-dimensional space by constructing the $Q_H \times Q_W \times Q$ centroid encoding tensor q as above, then bilinearly sampling from this encoding with the normalized flow values,

$$\tilde{\mathcal{F}}_{ij} = q(\mathcal{F}_x^{\text{norm}}, \mathcal{F}_y^{\text{norm}}), \tag{2}$$

where $\tilde{\mathcal{F}}$ is the Q-channel "flow plateau map" and $q(i', j')$ denotes "soft" indexing with normalized image coordinates (i.e., bilinear sampling, as opposed to hard indexing, $q_{i'j'}$.)

Competition is run on the flow plateau map with $K = 32$ maximum segments and $R = 3$ rounds to detect the motion segments $\mathcal{S}_M$. The largest of these by area is assumed to be the background; the motion indicator tensor $\mathcal{I}$ is created by taking the complement of this background segment, and its segment identity in $\mathcal{S}_M$ is set to 0. Examples of motion segments computed on **DAVIS2016** are shown in Figure S2 (third column.) When these segments are used as self-supervision for

EISEN, the resulting *static segments* (Figure S2, fourth column) can sometimes be more accurate than the motion segments, likely because affinities computed from single-frame appearance cues (such as color or texture similarity) generalize better than motion similarity, which varies substantially from one frame pair to another.

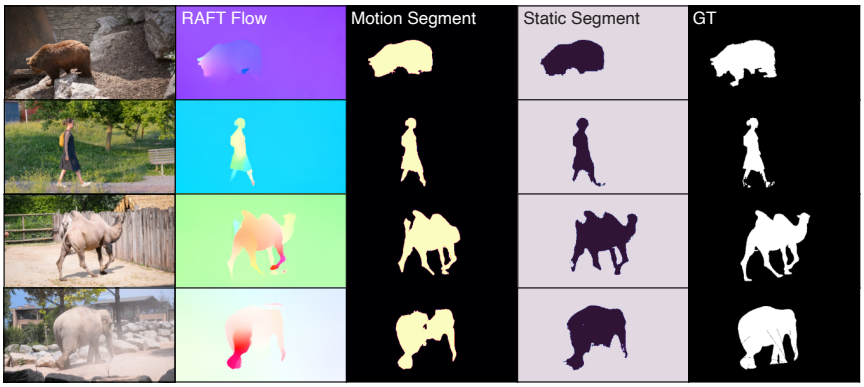

Fig. S2: **Explaining away background motion with Competition.** RAFT predictions cannot be threshholded directly. However, applying Competition to the flows, as though they were plateau maps, isolates the background and yields viable supervision targets. Training a *static* segmentation model with these targets can pick up details that the motion segments miss.

Computing confident segments. To isolate the confident segment predictions of an EISEN teacher model, we take advantage of the random initialization of the plateau map input to KProp: confident segments are those that are consistent across inference runs with different initializations. Specifically, If $\{\mathcal{S}_{\mathcal{T}}^{l} | \, l = 1, 2, ..., L\}$ are the segments output by $L = 5$ runs of the teacher model, then we compute a set of "meta-affinities" based on how often two scene elements belong to the same segment:

$$\hat{A}(a, b) = \frac{1}{L} \sum_{l} (\mathcal{S}_{\mathcal{T}}^{l}(a) == \mathcal{S}_{\mathcal{T}}^{l}(b)). \tag{3}$$

The meta-affinities are then converted to confident segments by applying KProp and Competition, keeping only the largest connected component of each predicted segment, and removing all segments of area < 10 pixels. This has the effect of filtering out low-confidence segments because, in practice, KProp only yields well-formed pixel clusters when the input (meta-)affinities are highly confident, i.e. close to 0 or 1. In the initial round of training, for which the EISEN teacher has not been pretrained, there are few if any confident segments, and the training target mainly reduces to Equation (9). Thus, successive rounds of bootstrapping tend to have more accurate training targets (such as by separating Spelke objects from agents and by providing pseudolabels for static objects)

even though all rounds use *the same rule* for inferring the connectivity target and loss mask.

A.2 Datasets

Playroom. The **Playroom** dataset was generated with ThreeDWorld [12] using custom code, which will be made public. The dataset consists of 40000 videos. Each video shows four objects placed on an immobile, randomly colored and textured "rug" in a tiled room. The objects are drawn from a pool of 2000 models and scaled so that they fit within the room. The camera is randomly positioned and pointed so that at least three of the objects are within view. At the fifth frame of each video, an invisible force is applied to one of the objects that pushes it toward another object; the scene ends when the pushed object comes to rest or leaves the field of view. Within a given video, only the pushed object is able to move.

Each object model is seen moving in $40000/2000 = 20$ videos. We hold out 4000 videos, use 500 of these as the *val* set, and train on the remaining 36000. EISEN and all baselines are trained only on the fifth frame of each video, with the supervising RAFT flow computed between the fifth and sixth frames.

In addition to the **Playroom** *train* and *val* datasets, we also generated a *test* dataset of 30 scenes that departs from the model training set in several ways. Specifically, it contains scenes with multiple copies of a particular object (e.g. the giraffes and zebras in the bottom two rows of Figure 5), scenes set in a different room (such that the background is different), and scenes with simply textured "primitive" objects containing, occluding, and colliding with each other; these primitive objects are not seen moving in the training set. Thus, the **Playroom** *test* set measures how well segmentation models generalize to new object arrangements and contexts. Both the *test* set and its generation script will be released along with all code.

Bridge. The Bridge dataset consists of 7200 demonstrations for 71 kitchen-themed tasks collected in 10 different environments [9]. Each demonstration shows a robotic arm executing a semantically meaningful task (e.g. put spoon into pot) in a household kitchen environment with different robotic positions, background, and lighting conditions. Each demonstration is collected with 3-5 camera viewpoints concurrently. 7 out of 10 environments were collected at the University of California, Berkeley. The three remaining environments were collected at the University of Pennsylvania. We train and evaluate models on the subset of the Bridge dataset collected at the University of California, Berkeley. In particular, we train on a total of 5881 randomly selected demonstrations. Since ground-truth segmentation annotations are not provided for the Bridge dataset, we manually annotate 50 held-out images for evaluating the validation performance.

A.3 Model Architecture Details

EISEN backbone. EISEN uses the ResNet50-DeepLab convolutional network as its feature extractor [22]. To ensure that EISEN is trained in an unsupervised manner, we randomly initialize the backbone parameters using He initialization [18], instead of using a ImageNet-pretrained backbone. The backbone is trained end-to-end along with the Affinity Prediction module.

EISEN input and output resolution. We use whole images as inputs without applying data augmentation. The input resolution is 512×512, 270×480, and 480×640 for the **Playroom**, **DAVIS**, and **Bridge** datasets respectively. The backbone outputs feature tensors at 1/4 of the input resolution, and EISEN predicts the affinities and segmentation masks at the output resolution of the backbone. Output segments are upsampled to the original resolution for evaluation.

EISEN hyperparameters. For all experiments on **Playroom** and **DAVIS**, we set the Affinity Prediction key and query dimension $D = 32$, the plateau map dimension $Q = 256$, and the maximum number of objects detectable by Competition $K = 32$. For **Bridge**, which contains more objects per scene, we increase to $K = 256$. By default we run KProp for $S = 40$ iterations and Competition for $R = 3$ rounds.

Baselines. The original baseline models are trained in a category-specific way with a separate semantic head for predicting object categories. However, given the absence of semantic supervision in a category-agnostic setting, we convert the semantic heads to binary objectness classifiers. In particular, we change the semantic loss function from the multi-class cross entropy to the binary cross entropy, which encourages the semantic head to predict 1 for Spelke objects and 0 otherwise. The semantic head architectures are identical to the original models, except for the output dimension in the final readout layer.

Table S1: Comparison of backbones and parameter count

Model	Backbone	Parameters
SSAP [13]	ResNet34-FPN	48M
DETR [4]	ResNet50	41M
MaskRCNN [17]	ResNet50-FPN	43M
Panoptic-Deeplab [6]	ResNet50-DeepLab	30M
EISEN	ResNet50-DeepLab	40M

A.4 Model Training

EISEN training protocol. We adopt a similar training protocol in Panoptic-Deeplab[6]. In particular, we use the 'poly' learning rate policy [23] with an initial learning rate of 0.005, and optimize with Adam [25] without weight decay. On the

Playroom and **DAVIS2016** datasets, we train EISEN with a batch size of 8 for 200k iterations. On the **Bridge** dataset, we train EISEN with a batch size of 8 for 60k, 20k, 20k iterations for three rounds of bootstrapping, respectively. Training EISEN for 100k iterations on 8 GPUs takes 20 hours. Because **DAVIS2016** is not typically used to evaluate static segmentation (rather than video object segmentation and tracking), we developed a protocol in which 45 out of 50 scenes are used for (motion-based) training and 5 out of 50 are held-out and shown as *static images only* to the pretrained EISEN model for testing.

Baseline training protocol. For a fair comparison with EISEN, we train baselines with whole images as inputs and without applying data augmentation. The baseline models are trained from scratch without using ImageNet-pretrained weights. Other settings are the same as the original MaskRCNN[17], Panoptic-Deeplab[6], DETR[4] and SSAP [13] models. For evaluating the baseline models at inference time, we perform a grid search to find thresholds for pixelwise "objectness" classification that maximize mIoU on 500 images from the training set. Note that because of how object segment proposals are scored against ground truth segments, DETR and Mask-RCNN are not penalized for using a low objectness threshold to make many proposals. For running multiple rounds of bootstrapping with Mask-RCNN and Panoptic DeepLab, we apply the same teacher-student setup as with EISEN. Confident segments from baseline models are determined by taking all object proposals above their optimal cross-validated confidence thresholds (see Model Evaluation below.)

A.5 Model Evaluation

EISEN inference time. Although EISEN contains two RNNs (KProp and Competition) that may be unrolled for many iterations, its inference time is not substantially longer than that of baselines: EISEN takes 155ms to perform inference on a single 512 x 512 image with 30 iterations of KProp and 3 iterations of Competition, compared to 65 ms for Mask-RCNN. Because KProp iterations are implemented as sparse matrix multiplications, unrolling this RNN for many iterations is not particularly slow. Note also that during training, $\mathcal{L}_{EISEN}$ is applied directly to the *affinities* $A^{ij}_{i'j'}$, such that it is not necessary to perform expensive backpropagation-through-time on the KProp RNN. (KProp and Competition do need to be run to compute teacher object segments for bootstrapping, but no gradients need to be computed from the teacher model.)

Matched mIoU. Our metric for how well a model segments a scene's Spelke objects is the intersection over union (IoU) between predicted and ground truth segments, averaged over ground truth segments in each image, and then averaged across images in the evaluation dataset.

The mIoU for a given image is computed by finding the best one-to-one match between predicted and ground truth segments using linear sum assignment; a single predicted segment therefore cannot match to multiple ground truth segments. Because EISEN outputs a "panoptic" instance segmentation map (i.e. every pixel is assigned to exactly one segment), there is no ambiguity about

which predicted segment should be matched with the ground truth. For baselines that output overlapping object segment *proposals* (in this work, DETR [42] and Mask-RCNN [17]), we compute a pseudo-panoptic segmentation map by assigning each pixel that falls into *any* predicted segment to the highest confidence prediction. This ensures fair comparison to EISEN and other panoptic segmentation models (like SSAP [13] and Panoptic DeepLab [6]) that cannot benefit from making multiple segment proposals at each spatial location. We think that this segmentation metric is the one most appropriate to our goal of parsing scenes into Spelke objects, since an agent that wanted to *use* an object-centric scene representation would ultimately need to choose which single segmentation proposal to act on at any given time.

Computing an affinity map from Vision Transformers. To convert DINO [5] or other Vision Transformer attention maps to an affinity-like output, we use the `vit_small` architecture with a patch size of 8x8. We compute the attention map using the final self-attention layer. The affinity between two patches p_1 and p_2 is obtained by computing the normalized dot product between their respective query vectors, q_1^h and q_2^h for a given head h. Since Vision Transformers outputs have multiple attention heads, we use the average of the attention values computed across different heads,

$$\mathit{Affinity}(p_1, p_2) = \left(\sum_h \frac{q_1^h \cdot q_2^h}{|q_1^h||q_2^h|} \right) / N_{heads}, \tag{4}$$

where N_{heads} is the number of attention heads.