MetaCOG: A Hierarchical Probabilistic Model for Learning Meta-Cognitive Visual Representations (Supplementary Material)

Marlene D. Berke [1] Zhangir Azerbayev[2] Mario Belledonne[1] Zenna Tavares[3,4] Julian Jara-Ettinger[1,5]

[1]Psychology Dept., Yale University, New Haven, Connecticut, USA
[2]Computer Science Dept., Princeton University, Princeton, New Jersey, USA
[3]Basis Research Institute, New York, New York, USA
[4]Zuckerman Institute and Data Science Institute, Columbia University, New York, New York, USA
[5]Wu Tsai Institute, Yale University, New Haven, Connecticut, USA

A ADDITIONAL METACOG DETAILS

The MetaCOG model and inference were implemented in the Julia-based probablistic programming language Gen [Cusumano-Towner et al., 2019], licensed under the Apache License Version 2.0. Code and data are available in our repository: `https://github.com/marleneberke/MetaCOG/tree/main`

A.1 DETAILS OF PROCEDURE FOR TAKING THE DIFFERENCE BETWEEN W_t AND D_t

From a world state W_t and a corresponding collection of detections D_t, MetaCOG tries to infer which detection were hallucinations or misses so as to update the prior over the hallucination and miss rate. The procedure for this inference is as follows: first, the objects in W_t are projected from 3D space to the 2D image. Then, each detection is matched to the nearest projected object of the same category as long as the Euclidean distance between the pair is within radius $\sigma = 200$ pixels (equal to the spatial noise parameter). After matching, the number of unmatched objects per category are counted, and considered to have been missed. The number of unmatched detections are also counted, and considered to be hallucinations. This process informs the updating of the αs and βs in the priors described above.

A.2 OBJECT CONSTRAINTS AND GENERATIVE MODEL PARAMETERS

Implementation of object constraints were as follows. World states without object persistence were not included in $\mathbb{W}$, which is equivalent to implicitly setting their prior probability to 0. The assumption that two objects cannot occupy the same region in space was implemented through a prior over $\mathbb{W}$ where the probability of having one object near another decreased to 0 according to a Gaussian distribution with $\sigma^2 = 1$.

To account for noise in a detector's location detection D_t, each detection was modeled as having 2D spatial noise, following a Gaussian with $\sigma_{x,y} = 200$ pixels.

Finally, the full generative model requires specifying a prior distribution over camera positions and focal points (although these are observable), set as uniform over 3D space, and a prior distribution over expected number of objects in a scene, sampled from a geometric distribution with parameter $p = 0.9$ (with a uniform prior over category type).

A.3 INFERENCE PROCEDURE IMPLEMENTATION

We sequentially approximate the joint posterior given in 3.2 using a particle filter with 100 particles. The solution space to the inference problem that we consider is sparse, and Sequential Monte-Carlo methods can suffer from degeneracy and loss of diversity in these situations. Our inference approach solves these problems by implementing particle rejuvenation over objects, locations, and beliefs about V. Rejuvenation is conducted using a series of Metropolis-Hastings [Canini et al., 2009] moves with data-driven proposals designed to obtain samples from $Pr(\vec{V}, \vec{W}|\vec{O}, \vec{cj})$.

During object rejuvenation, a new state W_t' is proposed by either adding or removing an object from the current state W_t. The probability p_{add} of proposing to add an object depends on the belief about V_t and the observed detections. The number of detections across the five categories are summed. Call this sum k. The probability p of k or more detection being produced by hallucinations is calculated as $p = 1 - e^{-\lambda_{total}} \sum_{i=0}^{k} \frac{\lambda_{total}^i}{i!}$, where $\lambda_{total} = \sum_c(\lambda_c)$. (That $e^{-\lambda_{total}} \sum_{i=0}^{k} \frac{\lambda_{total}^i}{i!}$ term is the cdf of $Pois(\lambda_{total})$ from 0 to k.) p_{add} should depend on p such that, when the belief about the overall hallucination rate is high relative to the number of detections, the probability p_{add} of proposing adding a new object to the world state is low, but when the hallucination rate is low relative to the number of detections, the probability p_{add} of proposing adding a new objects is high. However, if p_{add} were to be set to p, then there is a danger that, when there are more detections than can be explained as hallucinations, the proposal function would only ever propose adding objects, and never remove them. To balance the relative proposals of adding and removing objects, we set $p_{add} = 0.5 * p$, effectively setting the maximum of p_{add} to 0.5.

Objects in W_t have a weighted probability of being removed to produce W_t', such that their probability of removal is inversely proportional to the number of times the object was observed in scene t. New objects are added based on a data-driven distribution. This distribution samples object categories from a categorical distribution biased towards categories observed in the current scene t (see Data-driven proposal functions below). With probability 0.5, location is sampled from a 3-D uniform distribution, and otherwise sampled from a data-driven function with location biased toward 3D points likely to have caused the 2D detections. The proposal for adding a new object or removing and existing object from the world state is accepted or rejected according to the MH algorithm.

After a proposed change to W, a second rejuvenation step is performed on locations, wherein an object in W is randomly selected (with equal probability) to have a new location proposed. With probability 0.5, the new location is drawn from a multivariate normal distribution centered on the previous location, and it is otherwise sampled from a data-driven proposal. The proposed world-state W' with a perturbed location is then accepted or rejected according to the MH algorithm.

Finally, new parameters θ' for the meta-cognition are proposed by perturbing θ. Each parameter in the pair of length $|C|$ vectors is perturbed, with new values generated for $\theta(i, j)_t$ and for $\theta(i, j)_{t-1}$ $\forall i, j$, where i indexes the two vectors and j indexes their elements. The new value for each $\theta(i, j)_t$ is sampled from the appropriate distribution (Beta for the miss rate, Gamma for hallucination rate) with α_t and β_t, while the new value for $V(i, j)_{t-1}$ is sampled with α_{t-1} and β_{t-1}. These new values are accepted or rejected according to the MH algorithm.

This three-step rejuvenation process for world states, locations, and parameters of the meta-cognition is done for each particle for 200 iterations and the last state reached in the chain is used as the new rejuvenated particle.

A.4 DATA-DRIVEN PROPOSAL FUNCTIONS

During rejuvenation, new objects are proposed to be added to W_t' using a data-driven distribution. The category of the object comes from a categorical distribution. A category's weight in the categorical distribution depends on the number detections of that category in the current scene, and on the belief about the probability that those detections reflect a real object (as opposed to a hallucination). The meta-cognition can be used to calculate belief about the probability that this detection was generated by a real object. By Bayes rule, $P(real|obs) = P(obs|real) * P(real)/P(obs)$. Under the prior, each category has the same probability of appearing, so $P(real)$ will not affect the relative weighting of the categories. $P(obs|real)$ is simply the probability of detecting an object when it's present, p_c, for that category. $P(obs)$ is the probability of detecting an object of category c. Since a detection must be either true or a hallucination, $P(obs) = P(obs|real) + P(obs|not\ real) = p_c + (1 - e^{-\lambda_c})$, where the $(1 - e^{-\lambda_c})$ term is the probability of hallucinating one or more objects of category c. All together, the probability of sampling category c is proportional to $\frac{k_c * p_c}{p_c + (1 - e^{-\lambda_c})}$, where k_c is the number of detections of category c in this scene. So that its possible (though unlikely) to sample an object from an unseen category, k_c is set to have a minimum of one. This probability distribution for proposing object categories incoorporates both the number of times a category is detected, and the meta-cognitive belief about whether those detections were a mere hallucinations or generated from a real object.

In addition to a category, a location must be sampled for the proposed object. With probability 0.5, the location is sampled from a 3-D uniform distribution over the whole scene, and otherwise sampled from a data-driven function. Using the data-driven function, a point is sampled based on proximity to the line-segment that, when projected onto the 2D image, would result in the point where the detection was observed. The probability of proposing a particular point decreases with the distance from this line segment, following a Gaussian with $\sigma^2 = 0.01$.

In the second rejuvenation step a new location is proposed for an object. With probability 0.5, the new location is drawn from a multivariate normal distribution centered on the previous location, with $\sigma_{x,y,z} = 0.01$. Otherwise, it is sampled as described in the previous paragraph (based on proximity to the line segment that would results in the detection's 2D location).

B EXPERIMENT 1 ADDITIONAL DETAILS

B.1 EXPERIMENT 1 DATASET

Evaluating Exp. 1 we needed a dataset of videos of the same COCO object categories (so pre-trained NNs could detect them) across multiple videos. These videos had to include meta-data about the camera trajectory and position. Furthermore, so as to enable use to assess MetaCOG's accuracy locating objects in 3D space, we also needed ground-truth labels for the objects, including their locations in 3D space. Because we did not find any existing datasets to satisfy all these constraints, we rendered our own custom dataset.

B.1.1 TDW Scene Rendering

The TDW environment is licensed under the BSD 2-Clause "Simplified" License, Copyright 2020 Massachusetts Institute of Technology. We used the "box room 2018" model with a footprint with dimensions $\approx$ 12 x 8. For scale, the largest object model is about 1 unit along any face.

The number of objects in the scene is uniformly drawn from the counting numbers up to 3 and each object is uniformly assigned one of 5 object categories: potted plant, chair, bowl, tv, or umbrella. We sequentially place objects at locations drawn from a uniform distribution, resampling if two objects are within 1 units of each other or the walls so as to avoid collisions.

B.1.2 Camera Trajectory Sampling

A camera trajectory consists of a sequence of camera positions and focal points. We generate frames by querying the camera for an image at 20 linearly-spaced times.

Camera trajectories consisted of a circular path around the periphery of the room with noise, generated by a Gaussian process with a radial basis function (RBF) kernel with parameters $\sigma = 0.7$ and $\ell = 2.5$. The height of the camera is held constant at $y = 2$. The focal point trajectory is also sampled from a Gaussian process with an RBF kernel, mean above the center of the floor and component-wise parameters $\sigma = 0.7$ and $\ell = 2$. Fig. 6 shows some example trajectories.

The resulting images include objects passing in and out of frame and sometimes partly occluding each other. Fig. 7 shows five images from a scene in the dataset.

B.1.3 Video Counter-Balancing

To avoid order effects, all tests were run with four different videos orders. Videos were first randomly labeled from 1 to 50, and the four counterbalanced orders were: {1 ... 50}, {50 ... 1}, {26 ... 50, 1 ... 25}, and {25 ... 1, 26 ... 50}.

B.2 DETAILS ABOUT NEURAL NETWORKS USED IN EXPERIMENT 1

B.2.1 Pre-trained weights

For Faster R-CNN, we used the resnet50 pre-trained weights. For RetinaNet, we used the resnet50 pre-trained weights. And for DETR, we used the resnet101 pre-trained weights. These NNs were pre-trained on the COCO dataset.

Faster R-CNN is licensed under The MIT License (MIT), RetinaNet under Apache License 2.0, and DETR under Apache License 2.0.

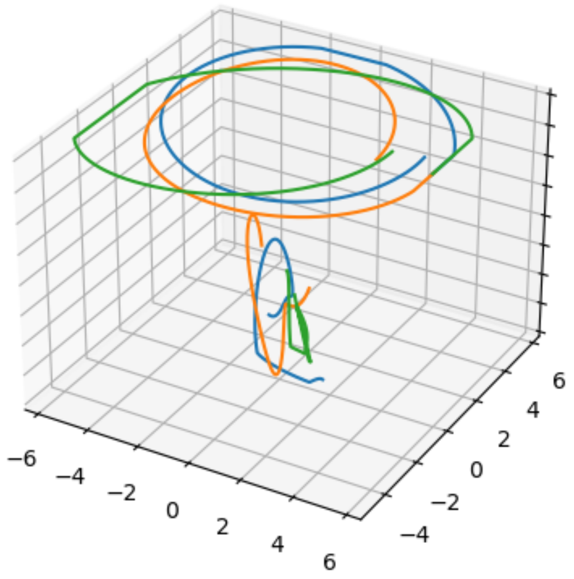

Figure 6: Three sampled trajectories. The wide circular patterns at the top are camera positions, and the smaller patterns near the bottom are camera focal points.

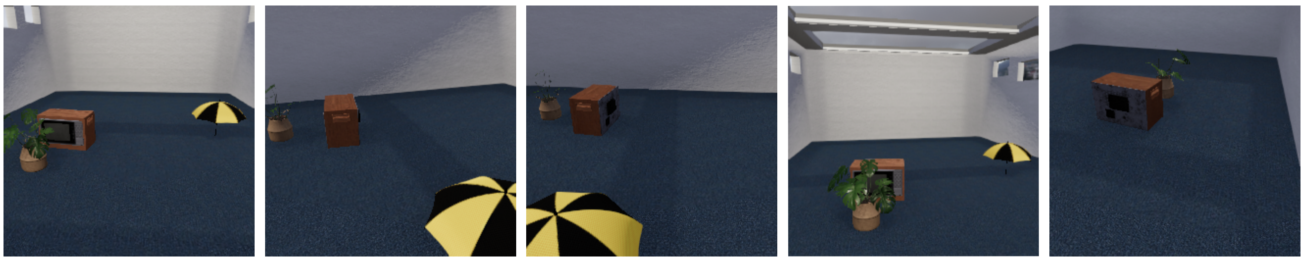

Figure 7: Five images from a scene in the dataset. The scene has three objects – a potted plant, a TV, and an umbrella. In the fourth image, the plant partly occludes the TV, and in the last image, the umbrella is out of view. The rest of the dataset and demo videos can be found at: https://github.com/marleneberke/MetaCOG/tree/main

B.2.2 Post-Processing

We process the outputs of the object detectors by filtering for the 5 object categories present in the scenes and then performed Non-Maximum Suppression (NMS) with an IoU (Intersection over Union) threshold of 0.4 (applied only for RetinaNet and Faster R-CNN, as it is not typically used for DETR). As input to MetaCOG, we took the top five highest-confidence detections per frame. The reason for taking the top five highest-confidence detection per frame, rather than all of the detections per frame is that sometimes some networks, especially DETR, can output many detections for a single frame. Typically, a confidence threshold is selected by fitting to ground-truth labels. But our setting is unsupervised and meant to apply to situations where ground-truth labels are perhaps unavailable. So, instead of fitting the confidence threshold using ground-truth, we took the top five highest-confidence detections. On most frames and with most networks, the NNs rarely output more than 5 detections, so in practice, taking the top 5 detections rarely affected the NN's output. DETR is the exception, and output more than 5 detections on many frames, resulting in its relatively poor baseline performance, as can be seen in Table 1.

B.2.3 mAP of NNs on our Dataset

To demonstrate that the neural networks are performing reasonably on our dataset, we calculated mAP values for the neural networks when applied to our dataset. We find that the object detectors perform similarly on our dataset as on the datasets on which they are typically evaluated. To show this, we evaluated the object detectors using the standard COCO evaluator (Lin et al. 2014). On the dataset used in Exp. 1, Faster R-CNN has a mAP of 39.4 on the training set, and 41.1 on the test set. This is similar to the mAPs reported for the most challenging object categories. For instance, Faster R-CNN's reported mAP for the plant category is 39.1 and 40.1 on the PASCAL VOC 2007 and 2012 test sets, respectively (Ren et al. 2015). RetinaNet had a mAP of 35.9 on the training set, and 36.3 on the test set. RetinaNet's reported mAP (across all object categories) on the COCO test dataset is 39.1 (Lin et al. 2017), so RetinaNet performs similarly on our dataset as on COCO. DETR had a mAP of 0.478 on the training set, and 0.492 on the test set. DETR's reported mAP on the COCO dataset is 43.5 (Carion et al. 2020). This confirms that the object detectors are operating within their expected performance ranges.

B.3 METRICS FOR EXPERIMENT 1

We evaluate MetaCOG's performance in three ways: 1) by testing whether it can learn an accurate meta-cognition of the object detection module, 2) by testing whether the learned meta-cognition improves inferences about which objects are where in 3D space, and 3) by testing whether the learned meta-cognition leads to improved accuracy at detecting objects in the 2D images.

B.3.1 MSE for θ

To measure how well MetaCOG learned a meta-cognition, we calculated the MSE between the inferred parameters $\hat{\theta}$ of the meta-cognition and the true parameters θ. The true θ was calculated for each object detection system as described in the following section B.3.2. The squared error was calculated for corresponding values in $\hat{\theta}$ and the true θ, then summed and divided by the total number of values ($2 * |C|$).

B.3.2 Calculating Ground-Truth θ

We needed a way, for a given image, to calculate how many times an object of each category was missed or hallucinated by the object detector. There are many possible ways to define and calculate ground-truth misses and hallucinations, but we chose a definition and procedure to match the definition and procedure MetaCOG uses (see A.1).

We used the 3D position of the object, the camera, and the camera's focus to determine whether and where the object would appear in the image. We considered an object to be missed if the object was in the image, but not detected within 200 pixels of its projected 2D location on the image. We calculated the miss rate for a category c as the number of misses divided by the number of times an object of category c was in view.

By hallucination, we mean a detection of an object of a particular category that was not caused by a ground-truth object of that category. To operationalize this, we considered a detection to be a hallucination if the detection occurred more than 200 pixels away from the projection of a ground-truth object of that category. We calculated the hallucination rate for a category c by counting the number of times an object of category c was hallucinated in our dataset, and then dividing by the total number of frames in the dataset.

The 200 pixel radius was set to match the radius used by MetaCOG in matching W_ts to D_ts (see A.1).

B.3.3 Jaccard Similarity

The Jaccard similarity coefficient is a useful metric for evaluating inferences about *which* objects are in a frame or scene. The Jaccard similarity coefficients measures the similarity between two sets by taking the size of the intersection of two sets and dividing it by the size of the union of the two sets. We can apply this measure to our setting by treating the object labels that were inferred as one set and the objects that were actually present as the second set. The Jaccard similarity coefficient is given by $J(\mathbb{I}, \mathbb{G}) = \frac{|\mathbb{I} \cap \mathbb{G}|}{|\mathbb{I} \cup \mathbb{G}|}$, where, in our setting, $\mathbb{I}$ is the set of object labels that were inferred (i.e. $\mathbb{I}$ = {chair, chair, bowl}) and $\mathbb{G}$ is the set of objects in ground-truth (i.e. $\mathbb{G}$ = {chair, bowl}). (In this example, $J(\mathbb{I}, \mathbb{G}) = \frac{2}{3}$.)

This Jaccard Similarity coefficient can be applied either at the frame or scene level, and is used in both contexts in this paper.

B.3.4 2D Inference Accuracy

To assess the accuracy of of the both MetaCOG and the NNs in a comparable way, we wanted an accuracy metric that could incooporate both *which* object categories are present, and *where* they are. The NNs operate over 2D images, not 3D spaces, so we must make this comparison at the level of 2D frames. The standard metric for object detection accuracy on images is mean Average Precision (mAP), which requires both a ground-truth 2D bounding box and an inferred/detected bounding box. Unfortunately, MetaCOG outputs a point in 3D space, which can be projected to a point in 2D space, but cannot be used to make a bounding box. But we still needed a metric by which to compare MetaCOG's object detection accuracy to those of the neural networks without a meta-cognition.

To solve this problem, we projected MetaCOG's inferences about the 3D location of objects to a point on each 2D image, and for the neural networks, we took the centroid of the bounding boxes. This way, MetaCOG and the neural networks' detections have the same format: an object label and a point on the image, or a tuple of (c, x, y). Then, for each object category, we used the Hungarian algorithm with Euclidean distances as cost to pair up the detections with the centroid of the ground-truth bounding box. For each pairing, we simply coded whether or not the detected point was within the ground-truth bounding box as 0 or 1. Last, we calculated the Jaccard similarity coefficient (see B.3.3) defining pairs as within the intersection of the sets if and only if they pair received the 1 coding. In other words, we counting up the number of pairs where the detection lay within the ground-truth bounding box, and divided by and the union of the detection and ground-truth sets: the sum of the number of unpaired detections, unpaired ground-truth bounding boxes, and number of total pairings. To extrapolate this accuracy metric from frames to videos, we simply average across frame accuracy for each frame in the video, and we average across videos to calculate overall accuracy. This accuracy metric encodes, for 2D frames, accuracy at detecting both *which* objects are present and *where* they are.

It is important to note that assessing accuracy at the 2D frame level is different from assessing accuracy at the 3D scene level. Accuracy at the 2D level should depend on the objects that are in the frame. For example, it could be the case that the scene contains a chair and a plant, but the chair is only in frame once, while the plant is in frame 20 times. In that case, 2D accuracy should depend more on identifying and locating the plant than the chair. This distinction will be important when we assess accuracy at the level of the 3D scene in the next section.

B.4 EXPERIMENT 1 SUPPLEMENTAL RESULTS AND DETAILS

B.4.1 3D Inference Accuracy

Our main results focused on MetaCOG's accuracy in 2D images. This was necessary so that we could evaluate its inferences against *NN Output*, because the object detection neural networks that we use only provide location on 2D images. Although not our main focus, here we report supplemental results evaluating MetaCOG's capacity to infer 3D world states in Exp. 1. This allows us to test MetaCOG's inferences at the scene level (i.e., what objects are in the room and where are they?), rather than at the frame level (i.e., what objects are visible at this time point and where are they in this 2D projection?).

Throughout, we compare MetaCOG only to Lesioned MetaCOG, as it is the only comparison model that also produces results in 3D space.

We first evaluate MetaCOG's accuracy in determining *which* objects are present in a scene (independent of location). To do this, we calculated the Jaccard similarity coefficent (see B.3.3) between the object categories inferred to be present in the scene, and those actually present. Table 2 shows Jaccard similarity coefficient averaged across the scenes. We find that on the whole, MetaCOG outperforms Lesioned MetaCOG, except for when paired with DETR, where the two perform comparably (as can be observed by noting that the confidence interval of each model's accuracy contains the mean of the other, indicating that there is no significant different between the means).

Next, we evaluate MetaCOG's accuracy in determining *where* the objects are in the scene. To do this, we first paired the inferred and ground-truth object locations of the same category using the Hungarian algorithm with Euclidean distance as the cost. We then averaged the distance between all of the pairs, in both the training and test sets, resulting in an average distance between inferred and ground-truth objects. The results in are shown in Table 3, in the Avg. Distance All Pairs column.

On the whole, Lesioned MetaCOG's meta-cognition represents a system with a relatively high hallucination rate (higher than the rates inferred by MetaCOG). That belief leads Lesioned MetaCOG to only infer that an object was present when there were more detections of that object, and more spatial information, leading to greater location precision for the objects it inferred to be present. Consequently, Lesioned MetaCOG only infers the presence of the objects that are easiest to locate.

For a more fair comparison, we can examine the average distance between pairs of inferred and ground-truth objects only for ground-truth objects that both MetaCOG and Lesioned MetaCOG's inferred to be present. These average distances are reported in the Avg. Distance Shared Pairs column of Table 3. For objects that both MetaCOG and Lesioned MetaCOG inferred to be present, MetaCOG consistently infers slightly better locations than does the Lesioned model.

For reference, the footprint of the room is $\approx$ 12 x 8, and the largest object is 1 unit across.

B.4.2 Results on Additional Dataset

To address questions of how well our results will generalize to a new dataset, we rendered a new dataset in ThreeDWorld, this time using five different background rooms and including up to six objects per scene. This collection of room models varies in room size, texture and color (e.g. carpeting vs wood vs vinyl floors), windows, and even ceiling beams. With the inclusion of more objects and sometimes smaller rooms, this means that the new dataset also includes scenes that are more crowded and complex. As a conservative evaluation we used Faster R-CNN, which has the highest baseline performance of all three neural networks that we use, and for which MetaCOG showed the smallest improvement in the one-room dataset. Table 5 reports the results with Faster R-CNN and MetaCOG on this new dataset. As this dataset is more challenging, Faster R-CNN performs worse on this dataset, so the quality of the inputs to MetaCOG is lower. As the problem is harder, we increased the computation during inference from 200 iterations of the three-step rejuvenation process to 1000 iterations, but no other hyperparameters were changed.

B.4.3 Faster R-CNN Fine-tuning Details

We trained Faster R-CNN on the training set using MetaCOG's inferences as ground-truth labels. Since MetaCOG infers an object category and a 3D point, rather than a bounding box, we constructed a 50x50 pixel bounding box centered on the

Table 2: Scene-Level Accuracy Results for Exp. 1. Values in Parentheses Represent 95% Bootstrapped Confidence Intervals.

Object Detector	Model	Acc. Training	Acc. Test
RetinaNet	MetaCOG	**0.66** (0.58, 0.75)	**0.60** (0.51, 0.68)
	Lesioned MetaCOG	0.58 (0.49, 0.66)	0.48 (0.41, 0.55)
Faster R-CNN	MetaCOG	**0.76** (0.67, 0.84)	**0.66** (0.58, 0.74)
	Lesioned MetaCOG	0.58 (0.49, 0.66)	0.49 (0.43, 0.56)
DETR	MetaCOG	0.32 (0.26, 0.38)	**0.35** (0.29, 0.41)
	Lesioned MetaCOG	**0.38** (0.30, 0.46)	0.34 (0.28, 0.41)

Table 3: Average 3D Distance Between Inferred and Ground-Truth Object Locations.

OBJECT DETECTOR	MODEL	AVG. DISTANCE ALL PAIRS	AVG. DISTANCE SHARED PAIRS
RETINANET	METACOG	**0.37** (0.31, 0.42)	**0.30** (0.26, 0.34)
	LESIONED METACOG	0.58 (0.49, 0.66)	0.48 (0.41, 0.55)
FASTER R-CNN	METACOG	0.36 (0.30, 0.40)	**0.30** (0.25, 0.34)
	LESIONED METACOG	**0.31** (0.25, 0.35)	0.31 (0.25, 0.35)
DETR	METACOG	1.26 (0.90, 1.58)	**0.71** (0.45, 0.92)
	LESIONED METACOG	**0.83** (0.52, 1.11)	0.81 (0.51, 1.06)

MetaCOG inferred's 3D point projected onto the 2D image. Then, Faster R-CNN with initialized to its pre-trained weights was trained on the 1000 images (50 videos $*$ 20 frames) in the training set. The training set was augmented by flipping the images horizontally. Faster R-CNN was trained for 10 epochs with a SGR optimizer (learning rate = 0.005, momentum = 0.9, weight decay = 0.0005) and a learning rate scheduler (step size = 3, gamma = 0.1).

Faster R-CNN was trained using standard multi-task loss function given in the original Faster R-CNN paper [Ren et al., 2015], Eq. 1:

$$L(p_i, i_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

From [Ren et al., 2015]: "Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object vs. not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L1)... The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$). The outputs of the cls and reg layers consist of p_i and t_i respectively." Thus, MetaCOG's inferences are integrated into Faster R-CNN's training by substituting them in for ground-truth in the loss function.

Unlike in traditional fine-tuning regimes, the classification head layer was not replaced because we wished to retain the information about the COCO object categories Faster R-CNN had been pre-trained on, as the object categories in our dataset were a subset of COCO categories.

Recall from 4.1 that all reported results show averages across four runs of MetaCOG, with four different orderings of the videos. Each run of MetaCOG produced slightly different inferences, so we trained Faster R-CNN using the outputs of each of the four runs of MetaCOG, producing four sets of weights for Faster R-CNN. In the third line of Table 5, we report the accuracy of fine-tuned Faster R-CNN by averaging across the accuracy of Faster R-CNN with each of the four sets of weights.

Finally, the outputs of fine-tuned Faster R-CNN were input again into MetaCOG, and MetaCOG inferred the objects present. The fourth line of Table 5 reports the accuracy of MetaCOG, averaged over four runs of MetaCOG, each with input from Faster R-CNN with different weights.

The results are shown in Fig. 4 and Table 5.

Table 4: Results on Dataset with Multiple Rooms

MODEL	ACC. TRAINING	ACC. TEST
METACOG	**0.69** (0.64, 0.74)	**0.60** (0.54, 0.67)
LESIONED METACOG	0.49 (0.43, 0.56)	0.46 (0.39, 0.53)
FASTER R-CNN	0.66 (0.61, 0.70)	0.57 (0.51, 0.63)

Table 5: Faster R-CNN Fine-tuned on MetaCOG's Inferred Labels.

Model	Acc. Training	Acc. Test
Pre-trained NN	0.75 (0.67, 0.80)	0.69 (0.63, 0.74)
MetaCOG	0.79 (0.73, 0.84)	0.76 (0.70, 0.82)
NN fine-tuned with MetaCOG	0.84 (0.81, 0.87)	0.81 (0.78, 0.84)
MetaCOG Round 2	**0.87** (0.83, 0.90)	**0.85** (0.82, 0.87)

B.4.4 Lesioned MetaCOG

To whether learning the correct parameters θ for the meta-cognition actually matter, or if the priors (over θ and over world states in the form of object constraints) do all the work of improving accuracy, we tested a *lesioned* version of MetaCOG fixing the values in θ to the mean of the prior. The initial prior over the hallucination rate for each object category is a Gamma distribution with with $\alpha = 1$ and $\beta = 1$. The mean of that prior is then 1.0, so in Lesioned MetaCOG, the hallucination rates are set to 1.0 The initial prior over the miss rate for each object category is a Beta distribution with $\alpha = 1$ and $\beta = 1$. The mean of that prior is then 0.5, so in Lesioned MetaCOG, the miss rates are set to 0.5. So this lesion produces the expectation that an object of each category will be hallucinated on average once per frame, and that, when an object is in view, it has a 50% chance of being missed.

B.4.5 Explaining Lesioned MetaCOG's Performance when Paired with DETR

As described in B.4.4, the Lesioned MetaCOG model's meta-cognitive representation of the behavior of the object detectors is fixed to the mean of the priors, which is a hallucination rate of 1.0 and a miss rate of 0.5 for each object category. It just so happens that, because of the post-processing procedure used for DETR (described in B.2.2), that representation are not too far off the ground-truth. The ground-truth miss rates for DETR are indeed around 0.5 for most categories (as can be seen in Fig. 3A), and for one category (umbrella), the hallucination rate is indeed very high $\lambda_{umbrella} = 0.79$. It's worth noting that the full MetaCOG model badly underestimates this category's hallucination rate (estimating it as $\hat{\lambda_{umbrella}} = 0.25$). It seems that DETR's poor baseline performance (Table 1) made it more difficult for MetaCOG to learn an accurate meta-cognitive representation. Since the Lesioned model's meta-cognition is not bad for DETR, and the full MetaCOG model struggled to learn a correct meta-cognition for DETR, the two models performed comparably when inferring the objects present. This shows that, in the case of DETR, learning the meta-cognition from experience did not result in a performance boost above and beyond MetaCOG's priors over meta-cognition and about the world (i.e. object constraints). In the case of DETR, the priors did all of the work.

B.4.6 Alignment Between the Statistics of the Dataset and MetaCOG's Meta-cognitive Representation

It is possible that the distributions in the MetaCOG's meta-cognitive representation might be overfit to the empirical distribution of detections on the dataset used in Experiment 1, and then inflate MetaCOG's performance. Here, we address this concern. First, the distributions forming the meta-cognitive representation (Poisson for hallucinations, Geometric for detections/misses) were set based on first-princiles, and before the dataset was even rendered. However, an overfit could still happen by coincidence. To address this, for each neural network and each object category, we tested whether the empirical distributions of hallucinations and detections could have come from a Poisson and Geometric distribution (respectively).

To test whether hallucinations followed the Poisson distribution, we tallied the number of hallucinations produced by each neural network on each image our dataset, fit a Poisson distribution to each network's distribution MLE, and then evaluated the quality of the match through a Chi-Squared goodness-of-fit test. Because the sum of Poisson distributions is also a Poisson distribution, we combined the distributions of all object categories and present one test per neural network. The results show that the distribution of hallucinations does not follow a Poisson distribution for Faster-RCNN ($\chi^2 = 29.5; df = 2; p < 1e-06$), RetinaNet ($\chi^2 = 163.4; df = 4; p < 1e-33$), or DETR ($\chi^2 = 2007.7; df = 5; p < 1e-99$).

To test whether detections followed a Geometric distribution, we likewise tallied the number of times each object in the camera's view was detected, and fitted a Geometric distribution to the detections. Because the sum of geometric distributions is not a geometric distribution, we present Chi-squared goodness-of-fit tests at the category level, for a total of 15 tests (five object categories, for three neural networks). These tests confirm that none of the detections for any NN and object category follow a geometric distribution.

For Faster-RCNN the results for the categories ["chair", “bowl”, "umbrella", "potted plant", "tv"] were [$(\chi^2 = 770.5; df = 3; p < 1e-99), (\chi^2 = 355.2; df = 1; p < 1e-78), (\chi^2 = 887.7; df = 2; p < 1e-99)*, (\chi^2 = 274.4; df = 3; p < 1e-58), (\chi^2 = 44.9; df = 2; p < 1e-09)$]. For RetinaNet the results for the categories ["chair", “bowl”, "umbrella", "potted plant", "tv"] were [$(\chi^2 = 325.9; df = 4; p < 1e-68), (\chi^2 = 290.8; df = 2; p < 1e-63), (\chi^2 = 635.9; df = 4; p < 1e-99), (\chi^2 = 247.5; df = 5; p < 1e-50), (\chi^2 = 70.2; df = 3; p < 1e-14)$]. For DETR the the results for the categories ["chair", “bowl”, "umbrella", "potted plant", "tv"] were [$(\chi^2 = 61.0; df = 5; p < 1e-11), (\chi^2 = 221.3; df = 5; p < 1e-45), (\chi^2 = 159.3; df = 5; p < 1e-31), (\chi^2 = 176.0; df = 5; p < 1e-35), (\chi^2 = 611.9; df = 5; p < 1e-99)$]. For all neural networks and object categories (15/15), we were able to reject the null that the detections caused by an object come from a geometric distribution.

Note also that all of the results are significant even after applying a Bonferroni correction for multiple comparisons (significant level of .0027 after correcting for 18 tests).

Taken together, this shows that, far from overfitting to the pattern of detections in the dataset, the chosen distributions chosen do not even fit the pattern of detections observed. This demonstrates that MetaCOG is robust to some level of misspecification of the distributions in the meta-cognition, providing further evidence that MetaCOG can handle some mis-alignment between its priors and the pattern of a NN’s detections a particular dataset.

B.4.7 Compute Resources and Runtime

We estimate the compute resources used in producing the final results reported in Exp. 1. Running the MetaCOG (and Lesioned) model four times per each of the three neural networks used 36 CPUs with about 5GB memory per CPU on an internal cluster for approximately 100 hours.

The goal of this work is to show an initial proof-of-concept that meta-cognition can be useful for object detection, and so we were not optimizing for efficiency at this time. That said, it is important to consider the resources that MetaCOG uses.

In Experiment 1, MetaCOG used 100 particle filters and 200 MCMC rejuvenation steps applied to each particle. We had access to 36 CPUs, so we were able to parallelize some of the particle filtering steps.

Under this setup, when paired with Faster-RCNN, MetaCOG took an average of 50 seconds per world state. There is a fair amount of variance (ranging from 20 to 102 seconds) per world state, owing to the complexity of the observations. To understand this variability, we have to delve deeper into the workings of the MetaCOG.

The most computationally costly piece of MetaCOG’s pipeline is in the inference procedure: specifically, calculating the possible associations between detections and objects in the world. MetaCOG models both the object in the world states and the detections as random finite sets. The bottleneck is in calculating the likelihoods of different possible matchings between detections and objects in the world state. Under the semantics of random finite sets [Vo and Vo, 2013], the conditional distribution over a set of observations (in our context, detections) X given a hypothesis of elements E (in our case, putative objects) is calculated and marginalized over all partitions of E to X where each element in X is associated with an element in E. Since, in our case, the distribution of mass over partitions is sparse (i.e. most of the probability mass is concentrated in a small percentage of partitions), we can employ a random walk approach to substantially reduce runtime without impacting accuracy [Lovász, 1993, Blum et al., 2016] by reducing the complexity from $O(N!)$ to $O(N^2)$.

Aside from the number of detections and putative objects in the world state, MetaCOG’s runtime depends on the amount of compute and the computational resources available. Our method uses a particle filter and MCMC rejuvenation steps applied to each particle. How many rejuvenation steps and particles are necessary for convergence depends on the dataset and the user’s trade-off between speed and accuracy. In Experiment 1, we were optimizing for accuracy in the speed-vs-accuracy trade-off, and so we erred on the side of overkill, using 100 particles and rejuvenating each particle using 200 MCMC steps. A use-case in which less compute is available could set a lower computational budget perhaps without compromising much accuracy. Extending the use-cases to situations in which speed is important requires studying how different computational budgets (i.e. different numbers of particles and MCMC steps) affect the speed-accuracy trade-off.

Finally, MetaCOG may not need to run on every single frame in a video. This approach of thinning frames has been taken to speed up other object detectors to real-time (`https://google.github.io/mediapipe/solutions/objectron.html`).

C EXPERIMENT 2 ADDITIONAL DETAILS

C.1 LIGHTWEIGHT METACOG

In addition to the full MetaCOG model described in the main text, we also implemented a simplified version for the simplified setting used in Exp. 2, eliminating spatial information so as to isolate the contribution of meta-cognition. This simplified setting is object detection without location, wherein a black-box object detector generates labels (without associated locations) for the objects present in a scene. So detections are simply object labels. Furthermore, the goal of inferences in this simplified setting is not to infer what objects are where in 3D space, but merely which objects are present in the scene. Camera trajectories are no longer included, and all objects are assumed to remain in view at all times.

Additionally, the meta-cognition in this Lightweight MetaCOG is simplified as well: the prior over the meta-cognition, G, does not vary with time, but is fixed as a Beta distribution with parameters $(\alpha = 2, \beta = 10)$. Hallucination and misses are treated as Bernoulli random variables, with their parameters θ sampled from the Beta prior, separately for each object category.

The inference targets and procedure are largely unchanged, with the estimate of joint posterior now sequentially approximated via:

$$Pr\left(\vec{V},\vec{W}|\vec{D}\right) \approx \prod_{t=1}^{T} Pr\left(D_t|\hat{V_t}^t, \hat{W_t}^t\right) Pr\left(\hat{W_t}^t\right) Pr\left(\hat{V_t}^t|\widehat{V_{t-1}}^t\right)$$

where the transition kernel, $Pr(\hat{V}^t|\hat{V}^{t-1})$, defines the identity function.

C.1.1 Details of Inference Procedure for Lightweight MetaCOG

The above equation is estimated via particles filtering, with 100 particles.

We implemented rejuvenation using a series of Metropolis-Hastings MCMC perturbation moves over $\hat{\theta}$. The proposal function is defined as a truncated normal distribution with bounds $(0, 1)$:

$$\widehat{\theta_{i,j}}^{t'} \sim \mathcal{N}(\mu = \widehat{\theta_{i,j}}^{t}, \sigma^2 = 0.01)$$

where $\theta_{i,j}$ is the jth element of vector i in the pair of vectors of parameters θ. A proposal is accepted or rejected according to the Metropolis-Hastings algorithm. Each element in θ is rejuvenated separately and in randomized order.

C.2 EXPERIMENT 2 DATASET

Here we discuss how we synthesized the dataset for evaluating Lightweight MetaCOG in Exp. 2.

In this context, the object detector can be represented by its parameters, θ, which is a pair of vectors of length $|C|$ containing the hallucination and miss rates for each object category. The parameters for each object detector was generated by drawing ten independent samples from a beta distribution, $\sim \mathrm{B}(\alpha = 2, \beta = 10)$. This distribution allows us to sample object detectors with variable error rates (mean value = $\frac{1}{6}$) while maintaining a low probability of sampling object detectors that produced hallucinations or misses more often than chance (0.005 chance of sampling values above 0.5; 0.06 chance that complete sampled object detector has at least one hallucination or miss rate above 0.5).

For each object detector, we sampled 75 world states. A Poisson distribution $N \sim Poisson(\lambda = 1)$ truncated with bounds $[1, 5]$ determined the number of objects in a world state. The object categories were samples from a uniform distribution. Each world state was a hypothetical collections of objects, summarized as a vector of 1s and 0s indicating the presence or absence of each category of objects. For each world state we used the object detector to synthesize the detections from $5 - 15$ simulated frames (number sampled from a uniform distribution), producing a total of $375 - 1125$ simulated frames per object detector. Inferences about the hallucination and miss rates of each object category are independent, and we thus considered situations with only five categories.

C.3 METRICS FOR EXPERIMENT 2

To measure how well MetaCOG learned a meta-cognition, we calculated the mean squared error (MSE) between the inferred parameters of the object detector $\hat{\theta}$ and the true parameters θ generating the percepts given by

$$\text{MSE} = \frac{1}{2|C|} \sum_{c \in C} \left((H_c - \hat{H}_c)^2 + (M_c - \hat{M}_c)^2 \right)$$

where C is the the set of object classes, H_c is the hallucination rate for category c, and M_c is the miss rate for category c.

Accuracy was measured using the Jaccard similarity coefficient of the set of object classes in the ground-truth world state W_t and the set of object classes in the inferred world state $\hat{W}_t$, see B.3.3.

To analyze MetaCOG's accuracy as a function of faultiness in the detected labels, we computed the average faultiness in a collection of detections D_t as

$$\zeta_t = \frac{1}{|D_t||C|} \sum_{c \in C} \sum_{x \in D_t} |\mathbb{1}_{W_t}(c) - \mathbb{1}_x(c)| \tag{1}$$

where C is the set of object classes, D_t is the collection of detected labels generated from world state W_t, $\mathbb{1}_{W_t}(c)$ is an indicator for whether an object of class c is in W_t, and $\mathbb{1}_x(c)$ is an indicator for whether an object of class c is in the detection x.

Because our sampling-based dataset generation process (C.2) does not guarantee enough data points for every possible faultiness value, analysis in Fig. 5C and D were computed using a rolling window such that each point shows average accuracy on the $[\zeta - .05, \zeta + .05]$ range.

C.4 COMPARISON MODELS FOR EXPERIMENT 2

To test whether learning a meta-cognition improves inferences about the world state, we compare MetaCOG with and without the meta-cognitive learning. We call MetaCOG without learned meta-cognition Lesioned MetaCOG. Like the other MetaCOG models, Lesioned MetaCOG has a meta-cognitive representation of V and uses an assumption of object permanence to infer the world states causing the detected labels. Lesioned MetaCOG, however, does not learn or adjust the parameters θ in its meta-represetation based on the observed labels. Formally, Lesioned MetaCOG assumes that the hallucination and miss rates for every category are the mean of the beta prior over hallucination and miss rates, call it $\hat{\theta}_{0,\mu}$. Lesioned MetaCOG then uses the same particle filtering process described in C.1, except that it conditions on $\hat{\theta}_{0,\mu}$ instead of $\hat{\theta}_{T,\mu}$.

As described in C.1, the prior over both the hallucination rate and the miss rate is a Beta distribution parameters $(\alpha = 2, \beta = 10)$. The mean of this prior is then $\frac{1}{6}$. So in the Lesioned MetaCOG for Exp. 2, the hallucination rate and the miss rate are both set to $\frac{1}{6}$.

We also compare two variations of MetaCOG with learning. MetaCOG Learning performs a joint inference over θ and $\vec{W}$ based on collections of detections $\vec{D}$ generated from the sequence of world states $\vec{W}$. This model allows us to evaluate how MetaCOG's inferences improve as a function of the observations it has received. We name this model observation-by-observation inference MetaCOG Learning.

After having received all T observations, MetaCOG could retrospectively re-infer the world states causing the T observations. This model, simply called MetaCOG, re-infers the world states that caused its observations conditioned on its estimate $\hat{\theta}^T_{T,\mu}$, as described in 3.2.

MetaCOG Learning lets us interpret how MetaCOG learns a meta-cognition, and MetaCOG lets us test how MetaCOG performs after having learned that meta-cognition. Lesioned MetaCOG serve as a baseline model for comparison.