

# Learning to Look Around: Intelligently Exploring Unseen Environments for Unknown Tasks

Dinesh Jayaraman  
UC Berkeley\*

Kristen Grauman  
UT Austin

## Abstract

*It is common to implicitly assume access to intelligently captured inputs (e.g., photos from a human photographer), yet autonomously capturing good observations is itself a major challenge. We address the problem of learning to look around: if an agent has the ability to voluntarily acquire new views to observe its environment, how can it learn efficient exploratory behaviors to acquire informative visual observations? We propose a reinforcement learning solution, where the agent is rewarded for actions that reduce its uncertainty about the unobserved portions of its environment. Based on this principle, we develop a recurrent neural network-based approach to perform active completion of panoramic natural scenes and 3D object shapes. Crucially, the learned policies are not tied to any recognition task nor to the particular semantic content seen during training. As a result, 1) the learned “look around” behavior is relevant even for new tasks in unseen environments, and 2) training data acquisition involves no manual labeling. Through tests in diverse settings, we demonstrate that our approach learns useful generic policies that transfer to new unseen tasks and environments.*

## 1. Introduction

Visual perception requires not only making inferences from observations, but also making decisions about *what to observe*. Individual views of an environment afford only a small fraction of all information relevant to a visual agent. For instance, an agent with a view of a television screen in front of it may not know if it is in a living room or a bedroom. An agent observing a mug from the side may have to move to see it from above to know what is inside. An agent surveying a rescue site may need to explore at the onset to get its bearings.

In principle, complete certainty in perception is only achieved by making every possible observation—that is, looking around in all directions, or systematically examining all sides of an object—yet observing all aspects is often

inconvenient if not intractable. In practice, however, not all views are equally informative. The natural visual world contains regularities, suggesting not every view needs to be sampled for near-perfect perception. For instance, humans rarely need to fully observe an object to understand its 3D shape [33, 58, 59], and one can often understand the primary contents of a room without literally scanning it [63]. Given a set of past observations, some new views are more useful than others. This leads us to investigate the question: *how can a learning system make intelligent decisions about how to acquire new exploratory visual observations?*

Today, much of the computer vision literature deals with inferring visual properties from a fixed observation. For instance, there are methods to infer shape from multiple views [24], depth from monocular views [54], or category labels of objects [37]. The implicit assumption is that the input visual observation is already appropriately captured. We contend that this assumption neglects a key part of the challenge: intelligence is often required to obtain proper inputs in the first place. Arbitrarily framed snapshots of the visual world are ill-suited both for human perception [16, 49] and for machine perception [3, 70]. Circumventing the acquisition problem is only viable for passive perception algorithms running on disembodied stationary machines, which are tasked only with processing human-captured imagery.

In contrast, we are interested in *learning to observe* efficiently—a critical yet understudied problem for autonomous embodied visual agents. An agent ought to be able to enter a new environment or pick up a new object and intelligently (non-exhaustively) look around. This capability would be valuable in both task-driven scenarios (e.g., a drone searches for signs of a particular activity) as well as scenarios where the task itself unfolds simultaneously with the agent’s exploratory actions (e.g., a search-and-rescue robot enters a burning building and dynamically decides its mission). While there is interesting recent headway in active object recognition [3, 11, 29, 43] and intelligent search mechanisms for detection [10, 32, 45, 73], such systems are supervised and task-specific—limited to accelerating a pre-defined recognition task.

We address the general setting, where exploration is not

---

\*Work done while at UT Austin.

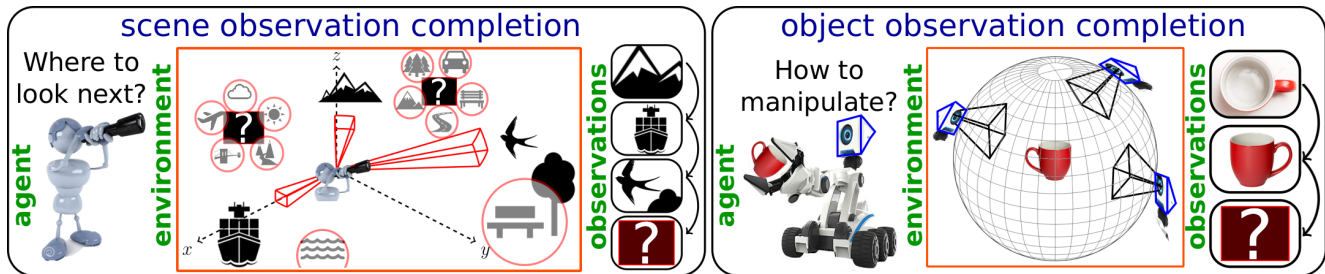


Figure 1. Looking around efficiently is a complex task requiring the ability to reason about regularities in the visual world using cues like context and geometry. (Left) An agent that has observed limited portions of its environment can reasonably hallucinate some unobserved portions (e.g. water near the ship), but is much more uncertain about other portions. Where should it look next? (Right) An agent inspecting a mug. Having seen a top view and a side view, how must it rotate the mug now to get maximum new information? Critically, we aim to learn policies that are not specific to a given object or scene, and not even to a specific individual task. Rather, the look-around policies ought to benefit the agent exploring new, unseen environments and performing tasks unspecified when learning the look-around behavior.

specialized to one task, but should benefit perception tasks in general. To this end, we formulate an unsupervised learning objective based on *active observation completion*: a system must intelligently acquire a small set of observations from which it can hallucinate all other possible observations. The agent continuously updates its internal model of a target scene or 3D shape based on all previously observed views. The goal is not to produce photorealistic predictions, but rather to represent the agent’s evolving internal state. Its task is to select actions leading to new views that will efficiently complete its internal model. Posing the active view acquisition problem in terms of observation completion has two key advantages: generality and low cost (label-free) training data. It is also well-motivated by findings that infants’ abilities to actively manipulate and inspect objects correlates with learning to complete 3D shapes [58].

We develop a reinforcement learning approach for active visual completion. It uses recurrent neural networks to aggregate information over a sequence of views. The agent is rewarded based on its predictions of unobserved views.

We explore our idea in two settings. See Figure 1. In the first, the agent scans a scene through its limited field of view camera; the goal is to select efficient camera motions so that after a few glimpses, it can model unobserved portions of the scene well. In the second, the agent manipulates a 3D object to inspect it; the goal is to select efficient manipulations so that after only a small number of actions, it has a full model of the object’s 3D shape. In both cases, the system must learn to leverage visual regularities (shape primitives, context, etc.) that suggest the likely contents of unseen views, focusing on portions that are hard to hallucinate. Furthermore, we show our exploratory policies are generic enough to be *transferred* to entirely new unseen tasks and environments.

## 2. Related work

**Saliency and attention:** Previous work studies the question of “where to look” to prioritize portions of *already captured* image/video data, so as to reserve computation for the

most salient regions or block out distractors [1, 5, 8, 23, 41, 47, 52, 62, 71], or to predict the gaze or preference of a human observer [27, 40, 61]. In contrast, in our setting, the system can never observe a snapshot of its entire environment at once; its decision is not where to focus within a current observation, but rather where to look for a *new* observation.

**Optimal sensor placement:** The sensor placement literature studies how to place sensors in a distributed network to provide maximum coverage [14, 36, 64]. Unlike our active completion problem, the sensors are static, i.e., their positions are preset, and their number is fixed. Further, sensor placement is based on coverage properties of the sensors, whereas our model must *react* to past observations.

**Active perception:** Intelligent control strategies for visual tasks were pioneered by [2, 6, 7, 65]. Recent work considers tasks such as active object localization [4, 10, 17, 21, 32, 45, 46, 56, 77], action detection in video [73], and object recognition [3, 29, 31, 43] including foveated vision systems that selectively obtain higher resolution data [9, 20, 53].

Our idea stands out from this body of work in two key aspects: (1) Rather than target a pre-defined recognition task, we aim to learn a data acquisition strategy useful to perception in general, hence framing it as active “observation completion”. We show how policies trained on our task are useful for recognition tasks *for which the system has not been trained to optimize its look-around behavior*. (2) Rather than manually labeled data, our method learns from unlabeled observations. Training good policies usually requires large amounts of data; our unsupervised objective removes the substantial burden of manually labeling this data. Instead, our approach exploits viewpoint-calibrated observations as “free” annotations that an agent can acquire through its own explorations at training time.

Work on intrinsic motivation “pseudorewards” [42] also reduces the need for external supervision, but focuses on learning “options” for policies seeking reward signals in a specific task and fixed environment. Similarly motivated self-supervised work [50] learns policies to play sparse-reward video games by augmenting environmental reward

from the game engine with rewards for actions whose outcomes are unpredictable. Neither work explores problems with real natural images.

**Active visual localization and mapping:** Active visual SLAM aims to limit samples needed to densely reconstruct a 3D environment using geometric methods [13, 34, 35, 44, 60]. Beyond measuring uncertainty in the current scene, our learning approach capitalizes on learned context from previous experiences with *different* scenes/objects.

**Image completion:** Completion tasks appear in other contexts within vision and graphics. Inpainting and texture synthesis fill small holes (e.g., [18, 51]), and large holes can be filled by pasting in regions or textures from other images of the same scene or similar-looking scenes [25, 55, 75]. Recent work explores unsupervised “proxy tasks” to learn representations, via various forms of completion like inpainting and colorization [39, 51, 74]. Our observation completion setting differs from these in that 1) it requires agent *action*, 2) a much smaller fraction of the overall environment is observable at a time, 3) our target is a representation of multimodal beliefs, rather than a photorealistic rendering, and 4) we use completion to learn exploratory *behaviors* rather than features.

**Learning to reconstruct:** While 3D vision has long been tackled with geometry and densely sampled views [24], recent work explores ways to inject learning into reconstruction and view synthesis [12, 15, 22, 28, 38, 67, 72, 76]. Whereas prior work learns to aggregate and extrapolate from passively captured views in one shot, our work is the first to consider active, sequential acquisition of informative views. Our view synthesis module builds on the one-shot reconstruction approach of [28], but our contribution is entirely different. Whereas [28] *infers a viewgrid image* from a single input view, our approach *learns look-around behavior* to select the sequence of views expected to best reconstruct all views.

Shortly after our work was first released [30], the Im2Pano3D project [57] explored scene completion on 360 panoramas given a partial RGB-D view. Both our work and [57] represent first attempts to i) extrapolate to omnidirectional panoramas and ii) posit extrapolation as a *scene understanding* task beyond graphics. However, whereas [57] considers completion tasks where about half of the scene is observed, in our case only about 5% of the scene is observed at a time. More importantly, our idea is to learn exploratory, non-myopic *policies* for how an agent should direct its camera over time—even in a novel environment; the output is a policy for how to move. In contrast, the goal in [57] is to perform one-shot reconstruction of a given scene’s semantic segmentation and 3D structure; the output is the panorama itself.

### 3. Approach

We now present our approach for learning to actively look around. For ease of presentation, we present the problem setup as applied to a 3D object understanding task. With minor modifications (detailed in Sec. 4) our framework applies also to the panoramic scene understanding setting. Both will be tested in results.

#### 3.1. Problem setup and notation

The problem setting is as follows: At timestep  $t = 1$ , an agent is presented with an object  $X$  in a random, unknown pose<sup>1</sup>. At every timestep, it can perform one action to rotate the object and observe it from the new viewpoint. Its objective is to make efficient exploratory rotations to understand the object’s shape. It maintains an internal representation of the object shape, which it updates after every new observation. After a budget of  $T$  timesteps of exploration, it should have learned a model that can produce a view of the object as seen from any specified new viewing angle.

We discretize the space of all viewpoints into a “viewgrid”  $V(X)$ , as in [29, 31, 43]. To do this, we evenly sample  $M$  azimuths from  $0^\circ$  to  $360^\circ$  and  $N$  elevations from  $-90^\circ$  to  $+90^\circ$  and form all  $MN$  possible pairings. Each pairing of an azimuth and an elevation corresponds to one viewpoint  $\theta_i$  on a viewing sphere focused on the object. Let  $x(X, \theta_i)$  denote the 2D image corresponding to the view of object  $X$  from viewpoint  $\theta_i$ . The viewgrid  $V(X)$  is the table of views  $x(X, \theta_i)$  for  $1 \leq i \leq MN$ . During training, the full viewgrid of each object is available to the agent as supervision. During testing, the system must predict the complete viewgrid, having seen only a few views within it.

At each timestep  $t$ , the agent observes a new view  $x_t$  and updates its *prediction* for the viewgrid  $\hat{V}_t(x_1, \dots, x_t)$ . Simplifying notation, the problem now reduces to sequentially exploring the viewgrid  $V$  to improve  $\hat{V}_t$  — in other words, actively *completing the observation* of the viewgrid  $V(X)$  of object  $X$ . Given the time budget  $T \ll MN$ , the agent can see a maximum of  $T$  views out of all  $MN$  views (maximum because it is allowed to revisit old views).

We explicitly choose to complete the viewgrid in the pixel-space so as to maintain generality—the full scene/3D object encompasses all potentially useful information for *any* task. Hence, by formulating active observation completion in the pixel space, our approach avoids committing to any intermediate semantic representation, in favor of learning policies that seek generic information useful to many tasks. That said, our formulation is easily adaptable to more specialized settings—e.g., if the target task only requires semantic segmentation labels, the predictions could be in the space of object labels instead.

<sup>1</sup>We assume the elevation angle alone is known, since this is true of real-world settings due to gravity.

The active observation completion task poses three major challenges. Firstly, to predict unobserved views well, the agent must learn to understand 3D from very few views. Classic geometric solutions struggle under these conditions. Instead, reconstruction must draw on semantic and contextual cues. Secondly, intelligent action is critical to this task. Given a set of past observations, the system must act based on which new views are likely to be most informative, i.e., determine which views would most improve its model of the full viewgrid. We stress that the system will be faced with objects and scenes it has never encountered during training, yet still must intelligently choose where it would be valuable to look next. Finally, the task is highly underconstrained—after only a few observations, there are typically many possibilities, and the agent must be able to handle this multimodality.

### 3.2. Active observation completion framework

Our solution to these challenges is a recurrent neural network, whose architecture naturally splits into five modules with distinct functions: SENSE, FUSE, AGGREGATE, DECODE, and ACT. We first present these modules and their connections; Sec. 3.3 below defines the learning objective and optimization. Architecture details for all modules are given in Fig 2.

**Encoding to an internal model of the target** First we define the core modules with which the agent encodes its internal model of the current environment. At each step  $t$ , the agent is presented with a 2D view  $x_t$  captured from a new viewpoint  $\theta_t$ . We stress that absolute viewpoint coordinates  $\theta_t$  are *not* fully known, and objects/scenes are *not* presented in any canonical orientation. All viewgrids inferred by our approach treat the first view’s azimuth as the origin. We assume only that the absolute elevation can be sensed using gravity, and that the agent is aware of the relative motion from the previous view. Let  $p_t$  denote this proprioceptive metadata (elevation, relative motion).

The SENSE module processes these inputs in separate neural network stacks to produce two vector outputs, which we jointly denote as  $s_t = \text{SENSE}(x_t, p_t)$  (see Fig 2, top left). FUSE combines information from both input streams and embeds it into  $f_t = \text{FUSE}(s_t)$  (Fig 2, top center). Then this combined sensory information  $f_t$  from the current observation is fed into AGGREGATE, which is a long short term memory module (LSTM) [26]. AGGREGATE maintains an encoded internal model  $a_t$  of the object/scene under observation to “remember” all relevant information from past observations. At each timestep, it updates this code, combining it with the current observation to produce  $a_t = \text{AGGREGATE}(f_1, \dots, f_t)$  (Fig 2, top right).

SENSE, FUSE, and AGGREGATE together may be thought of as performing the function of “encoding” observations into an internal model. This code  $a_t$  is now fed into two

modules, for producing the output viewgrid and selecting the action, respectively.

**Decoding to the inferred viewgrid** DECODE translates the aggregated code into the predicted viewgrid  $\hat{V}_t(x_1, \dots, x_t) = \text{DECODE}(a_t)$ . To do this, it first reshapes  $a_t$  into a sequence of small 2D feature maps (Fig 2, bottom right), before upsampling to the target dimensions using a series of learned up-convolutions. The final up-convolution produces  $MN$  maps, one for each of the  $MN$  views in the viewgrid. For color images, we produce  $3MN$  maps, one for each color channel of each view. This is then reshaped into the target viewgrid (Fig 2, bottom center). Seen views are pasted directly from memory.

**Acting to select the next viewpoint to observe** Finally, ACT processes the aggregate code  $a_t$  to issue a motor command  $\delta_t = \text{ACT}(a_t)$  (Fig 2, middle right). For objects, the motor commands rotate the object (i.e., agent manipulates the object or peers around it); for scenes, the motor commands move the camera (i.e., agent turns in the 3D environment). Upon execution, the observation’s pose updates for the next timestep to  $\theta_{t+1} = \theta_t + \delta_t$ . For  $t = 1$ ,  $\theta_1$  is randomly sampled.

Internally, ACT first produces a distribution over all possible actions, and then samples  $\delta_t$  from this distribution. To approximate the constraint that motions in the real world must be continuous, we restrict ACT to select “small” discrete actions (details in Sec 4) at each timestep. Due to the sampling operation, ACT is a *stochastic* neural network [48]. Once the new viewpoint  $\theta_{t+1}$  is set, a new view is captured and the whole process repeats. This happens until  $T$  timesteps have passed, involving  $T - 1$  actions.

### 3.3. Objective function and model optimization

All modules are jointly optimized end-to-end to improve the final reconstructed viewgrid  $\hat{V}_T$ , which contains predicted views  $\hat{x}_T(X, \theta_j)$  for all viewpoints  $\theta_j, 1 \leq j \leq MN$ .

A simple objective would be to minimize the distance between predicted and target views at the same viewpoint coordinate at time  $T$ : for each training object  $X$ ,  $L_T(X) = \sum_i d(\hat{x}_T(X, \theta_i), x(X, \theta_i))$ , where  $d(\cdot)$  is a distance function. However, this loss function requires viewpoint coordinates to be registered exactly in the output and target viewgrids, whereas the agent has only partial knowledge of the object’s pose (known elevation but unknown azimuth) and thus must output viewgrids assuming the azimuth coordinate of the first view to be the origin. Therefore, output viewgrids are shifted by an angle  $\Delta_0$  from the target viewgrid, and  $\Delta_0$  must be included in the loss function:

$$L_T(X) = \sum_{i=1}^{MN} d(\hat{x}_T(X, \theta_i + \Delta_0), x(X, \theta_i)). \quad (1)$$

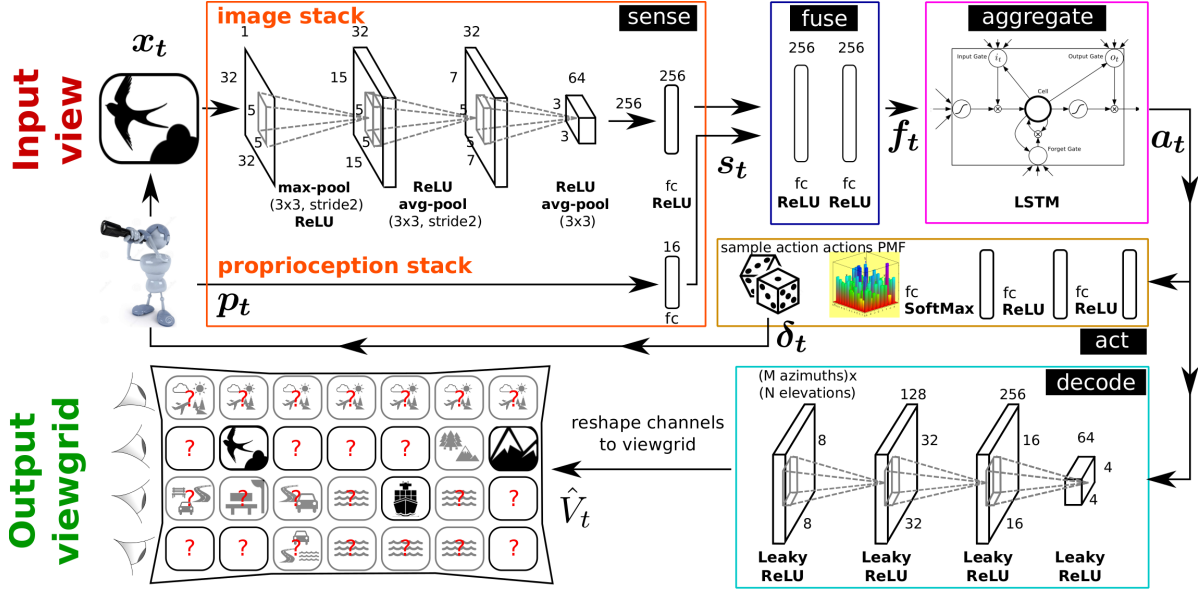


Figure 2. Architecture of our active observation completion system. While the input-output pair shown here is for the case of  $360^\circ$  scenes, we use the same architecture for the case of 3D objects. In the output viewgrid, solid black portions denote observed views, question marks denote unobserved views, and transparent black portions denote the system’s uncertain contextual guesses. See Sec. 3.2 for details.

We set  $d(\cdot)$  to be the per-pixel squared  $\mathcal{L}_2$  distance, so the agent expresses its uncertainty by averaging over the modes of its beliefs about unseen views. In principle,  $d(\cdot)$  could be replaced with other metrics. In particular, a GAN loss [19] would force the agent to select one belief mode to produce a photorealistic viewgrid, but the selected mode might not match the ground truth. Rather than one *plausible* photorealistic rendering (GAN), we aim to resolve uncertainty over time to converge to the *correct* model ( $\mathcal{L}_2$ ).

Note that  $\Delta_0$  is used only at training time and only to compute the loss. This choice has the effect of making the setting more realistic and also significantly improving generalization ability. If the viewpoint were fully known, the system might minimize the training objective by memorizing a mapping from  $\langle \text{view}, \text{viewpoint} \rangle$  to viewgrid, which would not generalize. Instead, with our unknown viewpoint setting and training objective (Eq 1), the system is incentivized to learn the harder but more generalizable skill of mental object rotation to produce the target viewgrids.

To minimize the loss, we employ a combination of stochastic gradient descent and REINFORCE [66], as in [47]. Specifically, the gradient of the loss in Eq 1 is backpropagated via the DECODE, AGGREGATE, FUSE, and SENSE modules. If ACT were a standard deterministic neural network module, it could receive gradients from SENSE. However, ACT is stochastic as it involves a sampling operation. To handle this, we use the REINFORCE technique: we compute reward  $R(X) = -L_T(X)$ , and apply it to the outputs of ACT at all timesteps<sup>2</sup>, backpropagating to encourage ACT behaviors that led to high rewards. To backprop-

agate through time (BPTT) to the previous timestep, the reward gradient from ACT is now passed to AGGREGATE for the previous timestep. BPTT for the LSTM module inside AGGREGATE proceeds normally with incoming gradients from the various timesteps—namely, the DECODE loss gradient for  $t = T$ , and the ACT reward gradients for previous timesteps.

In practice, we find it beneficial to penalize errors in the predicted viewgrid at *every* timestep, rather than only at  $t = T$ , so that the loss  $L_T(X)$  of Eq 1 changes to:

$$L(X) = \sum_{t=1}^T \sum_{i=1}^{MN} d(\hat{x}_t(X, \theta_i + \Delta_0), x(X, \theta_i)). \quad (2)$$

Note that this loss  $L(X)$  would reduce to the loss  $L_T(X)$  of Eq 1 if, instead of the summation over  $t$ ,  $t$  were held fixed at  $T$ . Since there are now incoming loss gradients to DECODE at every timestep, BPTT involves adding reward gradients from ACT to per-timestep loss gradients from DECODE before passing through AGGREGATE. BPTT through AGGREGATE is unaffected. Our approach learns a non-myopic policy to best utilize the budget  $T$ , meaning it can learn behaviors more complex than simply choosing the next most promising observation. Accordingly, we retain the reward  $R(X) = -L_T(X)$  for REINFORCE updates to ACT, based only on the final prediction; per-timestep rewards would induce greedy short-term behavior and disincentivize actions that yield gains in the long term, but not immediately.

Further, we find it useful to pretrain the entire network with  $T = 1$ , before training AGGREGATE and ACT with more timesteps, while other modules are frozen at their pre-trained configurations. This helps avoid poor local minima and enables much faster convergence.

<sup>2</sup>In practice, we reduce the variance of  $R$  for stable gradients by subtracting the “baseline” expected reward over the last few iterations.

There are prior methods that use recurrent neural networks and REINFORCE to achieve some notion of visual attention [29, 47, 73]. Following the best practice of adopting well-honed architectures in the literature, we retain broadly similar architectural choices to these recent instantiations of neural network policy learning where possible. This also facilitates fair comparisons with [29] for testing our policy transfer idea (defined below). However, in addition to all the technical details presented above, our approach differs significantly in its objective (see Sec. 2).

### 3.4. Unsupervised policy transfer to unseen tasks

The complete scene or 3D object encompasses all potentially useful information for any task. To capitalize on this property, we next propose an unsupervised policy transfer approach. The main idea is to inject our generic look-around policy into new unseen tasks in unseen environments. In particular, we consider transferring our policy—trained without supervision—into a specific recognition task that targets objects unseen by the policy learner.

To do this, we plug in our unsupervised active observation completion policies into the active categorization system of [29]. At training time, we train two models: an end-to-end model for active categorization using random policies following [29] (“model A”), and an active observation completion model (“model B”). Note that our completion model is, without supervision, trained to look around environments/objects that have zero overlap with model A’s target set. Furthermore, even the *categories* of objects seen during training may differ from those during testing.

At test time, we run forward passes through both models A and B simultaneously. At every timestep, both models observe the same input view. They then communicate as follows: the observation completion model B selects actions to complete its internal model of the new environment. At each timestep, this action is transmitted to model A, in place of the randomly sampled actions that it was trained with. Model A now produces the labels from the correct target label set. If the policy learned in model A is truly generic, it will intelligently explore to solve the new (unseen) categorization task.

## 4. Experiments

To validate our approach, we examine the effectiveness of active completion policies for faster reconstruction (Sec 4.2), as well as their utility for transferring unsupervised look-around policies to a recognition task (Sec 4.2).

### 4.1. Datasets and experimental setups

For benchmarking and reproducibility, we evaluate active settings with two widely used datasets:

On **SUN360** [69], our limited field-of-view ( $45^\circ$ ) agent attempts to complete an omnidirectional scene. SUN360 has spherical panoramas of diverse categories. We use the 26-category subset used in [29, 69]. The viewgrid has  $32 \times 32$  views from 5 camera elevations ( $-90, -45, \dots, 90^\circ$ ) and 8 azimuths ( $45, 90, \dots, 360^\circ$ ). At each timestep, the agent moves within a  $3 \text{ elevations} \times 5 \text{ azimuths}$  neighborhood from the current position. Balancing task difficulty (harder tasks require more views) and training speed (fewer views is faster) considerations, we set training episode length  $T = 6$  a priori.

On **ModelNet** [68], our agent manipulates a 3D object to complete its image-based shape model of the object. ModelNet has two subsets of CAD models: ModelNet-40 (40 categories) and ModelNet-10 (10 category-subset of ModelNet-40). To help test our ability to generalize to previously unseen categories, we train on categories in ModelNet-40 that are not in ModelNet-10. We then test both on new instances from the seen categories, and on the unseen categories from ModelNet-10. The viewgrid has  $32 \times 32$  views from 7 camera elevations ( $0, \pm 30, \pm 60, \pm 90^\circ$ ) and 12 azimuths ( $30, 60, \dots, 360^\circ$ ). Per-timestep motions are allowed within the  $5 \times 5$  neighboring angles of the current viewing angle. The training episode length is  $T = 4$ .

**Baselines** We test our active completion approach *ours* against a variety of baselines:

- **1-view** is our method trained with  $T = 1$ . No information aggregation or action selection is performed by this baseline.
- **random** is identical to our approach, except that the action selection module is replaced by randomly selected actions from the pool of all possible actions.
- **large-action** chooses the largest allowable action repeatedly. This tests if “informative” views are just far-apart views. Since there is no one largest action, we test all actions along the perimeter of the grid of allowable actions, and report results for the best-performing action on the test set.
- **peek-saliency** moves to the most salient view within reach at each timestep, using a popular saliency metric [23]. To avoid getting stuck in a local saliency maximum, it does not revisit seen views. **peek-saliency** tests if salient views are informative for observation completion. Note that this baseline “peeks” at neighboring views prior to action selection to measure saliency, giving it an unfair and impossible advantage over *ours* and the other baselines.

These baselines all use the same network architecture as *ours*, differing only in the exploration policy which we seek to evaluate. All results sample every possible starting position for all methods.

### SUN360 scene and ModelNet object observation completion examples

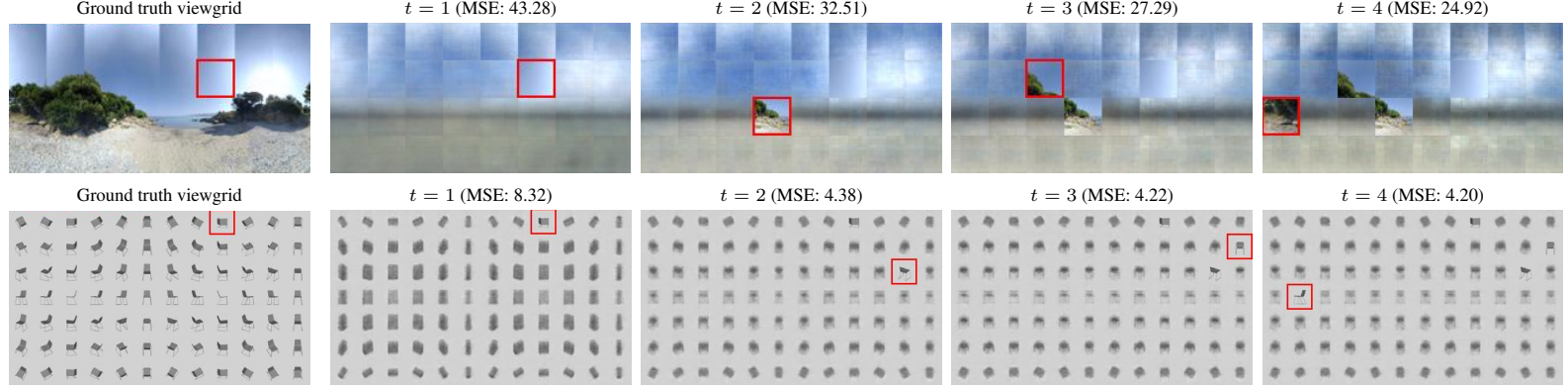


Figure 3. Best viewed on pdf with zoom. Episodes of active observation completion for a scene (top) and object (bottom). Column 1 shows the ground truth viewgrid with a red square around the random starting view. Columns 2-5 show our method’s viewgrid completions for  $t = 1, \dots, 4$  with red squares around selected views. As the model’s beliefs evolve, the space of possibilities grows more constrained, and the shape of the ground truth viewgrid begins to emerge. **Row 1:** The system correctly estimates a flat outdoor scene at  $t = 1$ , inferring the position of a horizon and even the sun from just one view of a gradient in the sky. At  $t = 2$ , it sees rocks and sand, and updates the viewgrid to begin resembling a beach. It then continues to focus on the most interesting (and unpredictable) region of the scene containing the rocks and shrubs. **Row 2:** The first view is overhead, and azimuthally aligned with one of the sides of an unseen category object (chair). Our agent chooses to move as far from this view as possible at  $t = 2$ , instantly forming a much more chair-like predicted viewgrid, which continues to improve gradually afterwards.

Table 1. Per-pixel mean squared error (MSE $\times 1000$ ) with episode length set to training length  $T$  (6 on SUN360, 4 on ModelNet), and corresponding improvement over 1-view baseline. Lower error and higher improvement is better. RGB (luminance) values in color (gray) images are normalized to  $[0,1]$ , so error values are on scale of 0 to 1000.

Dataset→	SUN360		ModelNet (seen classes)		ModelNet (unseen classes)	
Method↓ — Metric→	MSE(x1000)	Improvement	MSE(x1000)	Improvement	MSE(x1000)	Improvement
1-view	39.40	-	3.83	-	7.38	-
random	31.88	19.09%	3.46	9.66%	6.22	15.72%
large-action	30.76	21.93%	3.44	10.18%	6.16	16.53%
peek-saliency	27.00	31.47%	3.47	9.40%	6.35	13.96%
ours	<b>23.16</b>	<b>41.22%</b>	<b>3.25</b>	<b>15.14%</b>	<b>5.65</b>	<b>23.44%</b>

## 4.2. Active observation completion results

Tab 1 shows the scene and object completion mean-squared error on SUN360 and ModelNet (seen and unseen classes). For these results, episode lengths are held constant to  $T$  timesteps, same as during training. While all the multi-view methods improve over 1-view, our method outperforms all baselines by large margins. To isolate the impact of view selection, we report improvement over 1-view for all methods. Compared to *random*, *ours* consistently yields approximately 2x improvement; our gains over *large-action* are also substantial in all cases, meaning that simply looking at well-spaced views is not enough. Both outcomes highlight the major value in learning to intelligently look around. Improvements are larger on more difficult datasets, where errors are larger (SUN360 > ModelNet unseen > ModelNet seen). This is as expected, since additional views are most critical where one view produces very poor results. On SUN360, *peek-saliency*, which has unfair access to neighboring views for action selection, is the strongest baseline, but still falls short of *ours*. On ModelNet data, *peek-saliency* performs poorly, likely because saliency fails to differentiate well between the syn-

thetic CAD model views; what is informative about an object’s shape is much more complex than what low-level unsupervised saliency can measure. Importantly, our advantages hold *even for unseen categories* (rightmost), emphasizing the task-independence of our look-around policies.

Does our approach simply exploit its knowledge of camera elevation to sample useful elevations more than others? For instance, perhaps views from a horizontal camera position (elevation  $0^\circ$ ) are more informative than others. Upon investigation, we find that this is not the case in practice. In particular, our learned policy samples all elevations uniformly on both SUN360 and ModelNet data. Hence, the ability to sense gravity alone offers no advantage over the *random* baseline.

Figure 4 further shows how error drops over time. With perfect information aggregation, all methods should asymptotically approach zero error at high  $t$ , which diminishes the value of intelligent exploration. All methods show consistent improvement, with sharpest error drops for *ours*.

Fig 3 presents some completion episodes (see Supp for more). As our system explores, the rough “shape” of the target scene or object emerges in its viewgrid predictions.

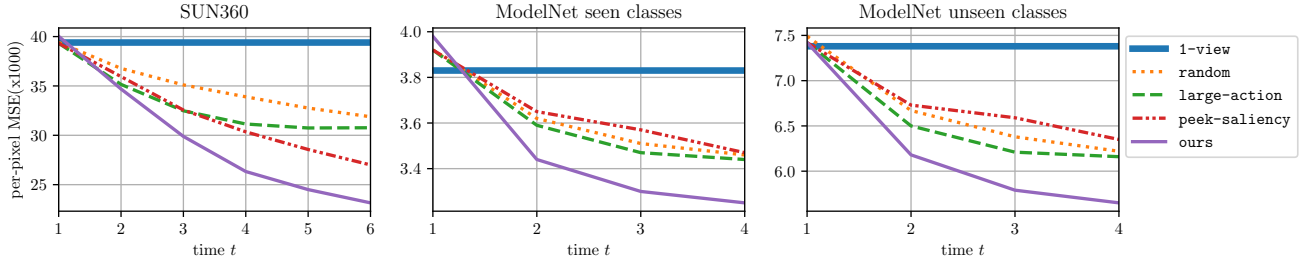


Figure 4. Active observation completion: per-pixel mean-squared error versus time for the three test datasets.

We stress that the goal of our work is not to obtain photorealistic images. Rather, the goal is to learn policies for looking around that efficiently resolve model uncertainty in novel environments; the predicted viewgrids visualize the agent’s beliefs over time. The key product of our method is a *policy*, not an image—as the next result emphasizes.

### 4.3. Unsupervised policy transfer results

Having shown our approach successfully trains unsupervised policies to acquire useful visual observations, we next test how well this policy transfers to a new task with new data from unseen categories (cf. Sec 3.4).

We closely follow the active categorization experimental setups in [29]. Using our method presented in Sec 3.4, we plug our unsupervised active observation completion policies into the active categorization system of [29]. The active categorization model (“model A”) is trained with random policies—this is the same as the *random-policy* baseline below. For ModelNet, we train “model A” on ModelNet-10 training objects, and the active observation completion model (“model B”) on *ModelNet-30* training objects, disjoint from the target ModelNet-10 dataset classes. For SUN360, both models are trained on SUN360 training data.

**Baselines** We compare: 1) *sup-policy*, the full end-to-end active categorization system trained using the “Lookahead active RNN” approach [29]; 2) *1-view*, a passive feed-forward neural network which only processes one randomly presented view and predicts its category. Its architecture is identical to *sup-policy* minus the action selection and information aggregation modules; and 3) *random-policy*, an active categorization system trained on the target classes that selects random actions. It uses the same core architecture as *sup-policy*, except for the action selection module; in place of learned actions, it selects random legal motions from the same motion neighborhood as *sup-policy*. We also compare 4) *large-action* and 5) *peek-saliency* from the last section.

Fig 5 shows the results. For both SUN360 active scene recognition and ModelNet-10 active object recognition, our unsupervised policies perform on par with the end-to-end active categorization policy of [29], easily outperforming *random-policy*, *1-view*, and *large-action*. This is remarkable because our policy is only trained for the separate, unsupervised active observation completion task. Fur-

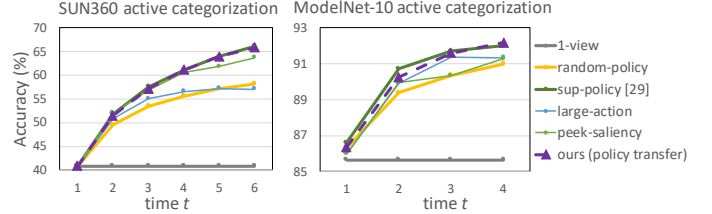


Figure 5. Policy transfer: Active categorization accuracy vs. time on SUN360 scenes (left) and ModelNet-10 objects (right).

ther, in the ModelNet case, it is also trained on data from disjoint classes. Recall that *peek-saliency* is not actually a viable solution; it “cheats” by trying out all moves and measuring saliency before selecting a move at each timestep. Still, even this strategy falls short of our method.

These results show the potential of unsupervised exploratory tasks to facilitate policy learning on massive unlabeled datasets. Policy learning is famously expensive in terms of data, computation, and time. Once trained, exploratory policies like the proposed active completion framework could be transferred to arbitrary new tasks with much smaller datasets. Performance may further improve if instead of directly transferring the policy, the policy could be finetuned for the new task, analogous to feature finetuning as widely employed in the passive recognition setting.

## 5. Conclusions

Our work tackles a new problem: how can a visual agent learn to look around, independent of a recognition task? We presented a new active observation completion framework for general exploratory behavior learning. Our reinforcement learning solution demonstrates consistently strong results across very different settings for realistic scene and object completion, compared to multiple revealing baselines. Our results showing successful application of our unsupervised exploratory policy for active recognition are the first demonstration of “policy transfer” between tasks to our knowledge. These results hold great promise for task-agnostic exploration, an important step towards autonomous embodied visual agents.

**Acknowledgements:** This research is supported in part by a DARPA Lifelong Learning Machines award, an AWS ML Research Award, and a Samsung Fellowship. We thank TACC for providing computing resources.

## References

- [1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009. 2
- [2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. In *IJCV*, 1988. 2
- [3] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017. 1, 2
- [4] A. Andreopoulos and J. Tsotsos. 50 years of object recognition: Directions forward. In *CVIU*, 2013. 2
- [5] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015. 2
- [6] R. Bajcsy. Active perception. In *Proceedings of the IEEE*, 1988. 2
- [7] D. Ballard. Animate vision. In *Artificial Intelligence*, 1991. 2
- [8] L. Bazzani, H. Larochelle, V. Murino, J.-A. Ting, and N. d. Freitas. Learning attentional policies for tracking and recognition in video with deep networks. In *ICML*, 2011. 2
- [9] N. Butko and J. Movellan. Optimal scanning for faster object detection. In *CVPR*, 2009. 2
- [10] J. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 1, 2
- [11] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. In *arXiv preprint arXiv:1512.03012*, 2015. 1
- [12] C. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 3
- [13] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. In *TPAMI*, 2002. 3
- [14] S. S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *WCNC*, 2003. 2
- [15] A. Dosovitskiy, J. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 3
- [16] S. Edelman and H. H. Bülthoff. Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. In *Vision research*, 1992. 1
- [17] A. G. Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object detection. In *CVPR*, 2015. 2
- [18] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 3
- [19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 5
- [20] S. Gould, J. Arfvidsson, A. Kaehler, B. Sapp, M. Messner, G. Bradski, P. Baumstarck, S. Chung, and A. Ng. Peripheral-foveal vision for real-time object recognition and tracking in video. In *IJCAI*, 2007. 2
- [21] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017. 2
- [22] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *CVPR*, 2017. 3
- [23] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *NIPS*, 2006. 2, 6
- [24] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 3
- [25] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Graphics (TOG)*, 2007. 3
- [26] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, 1997. 4
- [27] H. Hu, Y. Lin, M. Liu, H. Cheng, Y. Chang, and M. Sun. Deep 360 pilot: Learning a deep agent for piloting through 360 sports videos. In *CVPR*, 2017. 2
- [28] D. Jayaraman, R. Gao, and K. Grauman. Unsupervised learning through one-shot image-based shape reconstruction. In *arXiv*, 2017. 3
- [29] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *ECCV*, 2016. 1, 2, 3, 6, 8
- [30] D. Jayaraman and K. Grauman. Learning to look around. *arXiv preprint arXiv:1709.00507*, 2017. 3
- [31] E. Johns, S. Leutenegger, and A. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016. 2, 3
- [32] S. Karayev, T. Baumgartner, M. Fritz, and T. Darrell. Timely object recognition. In *NIPS*, 2012. 1, 2
- [33] P. J. Kellman and E. S. Spelke. Perception of partly occluded objects in infancy. In *Cognitive psychology*, 1983. 1
- [34] A. Kim and R. M. Eustice. Perception-driven navigation: Active visual slam for robotic area coverage. In *ICRA*, 2013. 3
- [35] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. In *IJRR*, 2008. 3
- [36] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, 2007. 2
- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1
- [38] T. Kulkarni, W. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015. 3
- [39] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 3
- [40] Y. Li, A. Fathi, and J. M. Rehg. Learning to predict gaze in egocentric video. In *ICCV*, 2013. 2
- [41] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *PAMI*, 2011. 2
- [42] M. Machado and M. Bowling. Learning purposeful behaviour in the absence of rewards. In *ICML*, 2016. 2
- [43] M. Malmir, K. Sikka, D. Forster, J. Movellan, and G. W. Cottrell. Deep Q-learning for active recognition of GERMS. In *BMVC*, 2015. 1, 2, 3
- [44] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *RSS*, 2007. 3
- [45] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, 2016. 1, 2

- [46] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. In *ICLR*, 2017. 2
- [47] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014. 2, 5, 6
- [48] R. M. Neal. Learning stochastic feedforward networks. In *Tech Report*, 1990. 4
- [49] S. Palmer, E. Rosch, and P. Chase. Canonical perspective and the perception of objects. In *Attention and performance IX*, 1981. 1
- [50] D. Pathak, P. Agrawal, A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 2
- [51] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 3
- [52] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 2012. 2
- [53] M. Ranzato. On learning where to look. In *arXiv preprint arXiv:1405.5488*, 2014. 2
- [54] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. In *TPAMI*, 2009. 1
- [55] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz. Photo uncrop. In *ECCV*, 2014. 3
- [56] S. Soatto. Actionable information in vision. In *ICCV*, 2009. 2
- [57] S. Song, A. Zeng, A. X. Chang, M. Savva, S. Savarese, and T. Funkhouser. Im2pano3d: Extrapolating 360 structure and semantics beyond the field of view. *arXiv preprint arXiv:1712.04569 (to appear at CVPR 2018)*, 2017. 3
- [58] K. C. Soska, K. E. Adolph, and S. P. Johnson. Systems in development: motor skill acquisition facilitates three-dimensional object completion. In *Developmental psychology*, 2010. 1, 2
- [59] K. C. Soska and S. P. Johnson. Development of three-dimensional object completion in infancy. In *Child development*, 2008. 1
- [60] R. Spica, P. R. Giordano, and F. Chaumette. Active structure from motion: application to point, sphere, and cylinder. 2014. 3
- [61] Y.-C. Su, D. Jayaraman, and K. Grauman. Pano2vid: Automatic cinematography for watching 360 videos. In *ACCV*, 2016. 2
- [62] A. Torralba. Neurobiology of attention, chapter contextual influences on saliency. 2005. 2
- [63] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. In *Psychological review*, 2006. 1
- [64] B. Wang. Coverage problems in sensor networks: A survey. In *ACM CSUR*, 2011. 2
- [65] D. Wilkes and J. Tsotsos. Active object recognition. In *CVPR*, 1992. 2
- [66] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning*, 1992. 5
- [67] J. Wu, T. Xue, J. Lim, Y. Tian, J. Tenenbaum, A. Torralba, and W. Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 3
- [68] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 6
- [69] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012. 6
- [70] B. Xiong and K. Grauman. Detecting snap points in egocentric video with a web photo prior. In *ECCV*, 2014. 1
- [71] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 2
- [72] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016. 3
- [73] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016. 1, 2, 6
- [74] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2016. 3
- [75] Y. Zhang, J. Xiao, J. Hays, and P. Tan. Framebreak: Dramatic image extrapolation by guided shift-maps. In *CVPR*, 2013. 3
- [76] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. Efros. View synthesis by appearance flow. In *ECCV*, 2016. 3
- [77] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017. 2