Learning to look around

Dinesh Jayaraman
UT Austin
dineshj@cs.utexas.edu

Kristen Grauman
UT Austin
grauman@cs.utexas.edu

Abstract

Visual perception requires not only making inferences from observations, but also making decisions about what to observe*. Though much of the computer vision literature implicitly assumes a well-captured visual observation as input, in reality a single view of a complex visual environment—or even multiple arbitrarily chosen views—may provide too little information for perception tasks. We aim to address the problem of "learning to look around" in the first place. Specifically, in a setting where a visual agent has the ability to voluntarily acquire new views to observe its environment, how can we train it to exhibit efficient exploratory behaviors to acquire informative observations? We treat this as a reinforcement learning problem, where a system is rewarded for actions that reduce its uncertainty about the unobserved portions of its environment. Based on this principle, we develop recurrent neural network-based systems to perform* active completion *of panoramic natural scenes and 3-D object shapes. Crucially, the learned policies are not closely tied 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 system learns useful and* generic *exploratory policies that transfer to new unseen tasks, an important step for autonomous embodied visual agents.*

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 if it is empty. 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 3-D shape [52, 51, 29], and one can often understand the primary contents of a room without literally scanning it [55]. 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 [22], depth from monocular views [48], or category labels of objects [33]. 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 [14, 44] and for machine perception [3, 62]. 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).

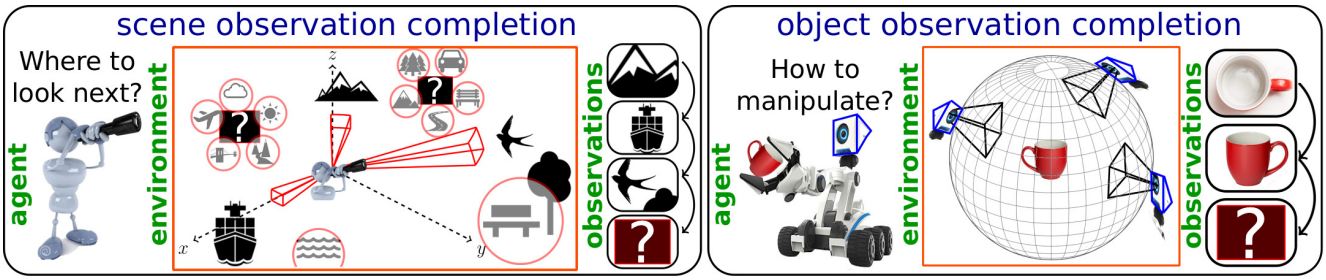

Figure 1. Looking around efficiently is a complex task requiring the ability to reason about regularities in the visual world using cues like context and geometry. (Left) An agent that has observed limited portions of its environment can reasonably hallucinate some unobserved portions (e.g. water near 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.

We address the general setting, where exploration is not specialized to one task, but should benefit perception tasks in general. To this end, we formulate the learning objective as *active observation completion*: a system must intelligently acquire a small set of observations, so that it is then able to hallucinate all other possible observations. The agent continuously updates its internal model of a target scene or 3-D 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 3-D shapes [51].

We develop a reinforcement learning solution for active visual completion. Our approach 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 two applications of our idea. See Figure 1. In the first, the agent scans an omnidirectional natural scene through its limited field of view camera; here the goal is to select efficient camera motions so that after a few glimpses, it has modeled unobserved portions of the scene well. In the second, the agent manipulates a 3-D object to inspect it; here 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 actions on portions that are difficult to hallucinate. Finally, we show how exploratory policies learned in this manner are generic enough to be *transferred* to entirely new unseen tasks set in unseen 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 [21, 54, 37, 1, 46, 41, 5, 63, 8], or to predict the gaze of a human observer [36]. 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. We compare our approach against a saliency-based method in our experiments.

Optimal sensor placement: The sensor placement literature studies how to place sensors in a distributed network to provide maximum coverage [12, 32, 56]. 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 sequential observations obtained by its sensors.

Active perception: Intelligent control strategies for acquiring data for specific visual tasks were pioneered by [6, 2, 7, 57]. Recent work considers tasks such as active object localization [4, 24, 15, 50, 19, 69, 40], action detection in video [65], and object recognition (e.g., [49, 66, 43, 38, 28, 27, 3]) including foveated vision systems that selectively obtain higher resolution data [18, 9, 47]. Our idea stands out from this body of work in four aspects: (1) Most importantly, rather than target a 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 viewpoint-calibrated observations. Training good policies usually requires large amounts of data, and our unsupervised objective removes the substantial burden of manually labeling this data. Instead, the viewpoint-calibrated observations exploited in our approach serve as "free" annotations that an agent can acquire through its own explorations at training time. (3) With very few recent exceptions [38, 27, 3], prior active methods deal with simplified unrealistic (often virtual) settings, whereas we employ complex real-world imagery. (4) Whereas most prior methods involve explicit greedy and heuristic reasoning about views, we train an end-to-end data-driven policy for purposeful data acquisition. Only limited recent work studies end-to-end policy learning for active vision, and for the different and *supervised* tasks of object and action recognition [27, 65]. While our task and objective are very different, following the standard 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, training a recurrent neural network with a combination of backpropagation and REINFORCE as in [41, 27, 65] to minimize the training objective.

Active visual localization and mapping: Active visual SLAM aims to limit samples needed to densely reconstruct a 3-D environment using geometric methods [39, 11, 31, 30, 53]. Beyond measuring uncertainty in the current scene, our learning approach capitalizes on learned context from previous experiences with *different* scenes/objects. While purely geometric methods are confined to using exactly what they see, and hence typically require dense observations, our approach can infer substantial missing content using semantic and contextual cues. Finally, our scope is broader than SLAM, in that it also applies to image-based panoramic scene completion.

Image completion: Completion tasks appear in other contexts within vision and graphics. Inpainting and texture synthesis fill small holes in images by modeling local textures (e.g., [45, 16]). Influential scene completion work [23] showed that larger holes can be filled by pasting appropriate regions from similar-looking scenes. Recent work explores unsupervised "proxy tasks" to learn useful deep visual representations, based on various forms of observation completion like inpainting and colorization [45, 67, 35]. 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 photo-realistic rendering, and 4) we use completion to learn exploratory *behaviors* rather than features.

Learning to reconstruct: While 3-D vision has long been tackled with geometry and densely sampled views [22], recent work explores ways to inject learning into reconstruction and view synthesis [34, 13, 68, 10, 59, 64, 20, 26]. 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. Furthermore, existing methods concentrate on object-specific models (e.g., chairs) and/or 3-D CAD objects, making them inapplicable to unseen categories and environments. No prior work considers inferring the appearance of complete 360 degree scenes. Our model's view synthesis module builds on the one-shot reconstruction approach of [26], but our contribution is entirely different: whereas [26] infers a viewgrid from a single input view, our approach learns look-around behavior to select the sequence of views expected to best reconstruct all views. Further, while [26] targets image feature learning, we aim to learn exploratory *action policies*.

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 3-D 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 active agent is presented with an object X in a random, unknown pose[1]. At every timestep, it can perform one action to rotate the object and observe it from the resulting 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.

In practice, we discretize the space of all viewpoints into a "viewgrid" $V(X)$. To do this, we evenly sample M azimuths from 0° to 360° and N elevations from -90° to +90° and form all MN possible pairings. Each pairing of an azimuth and an elevation corresponds to one viewpoint

[1]We assume the elevation angle alone is known, since this is true of real-world settings due to gravity.

$\boldsymbol{\theta}_i$ on a viewing sphere focused on the object. Let $\boldsymbol{x}(X, \boldsymbol{\theta}_i)$ denote the 2-D image corresponding to the view of object X from viewpoint $\boldsymbol{\theta}_i$. The viewgrid $V(X)$ is the table of views $\boldsymbol{x}(X, \boldsymbol{\theta_i})$ 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 $\boldsymbol{x}_t$ and updates its *prediction* for the viewgrid $\hat{V}_t(\boldsymbol{x_1}, \cdots, \boldsymbol{x_t})$. Simplifying notation a little, 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 $T << MN$, the agent can see a maximum of T views out of all MN views (maximum because it is allowed to revisit old views). We 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. This is why we propose active observation completion in the pixel space as a means to learn policies that seek generic information useful to many tasks. Our formulation is easily adaptable to more specialized settings—e.g., if the target task only requires perceiving poses of people, the predictions could be in the keypoint space instead.

This active observation completion task poses three major challenges. Firstly, to predict unobserved views well, the agent must learn to understand 3-D from very few views. Classic geometric solutions (structure-from-motion/SLAM) do not work 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 2-D view $\boldsymbol{x}_t$ captured from a new viewpoint $\boldsymbol{\theta}_t$. We stress that absolute viewpoint coordinates $\boldsymbol{\theta}_t$ 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 $\boldsymbol{p}_t$ 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 $\boldsymbol{s}_t = \text{SENSE}(\boldsymbol{x}_t, \boldsymbol{p}_t)$ (see Fig 2, top left). FUSE combines information from both input streams and embeds it into $\boldsymbol{f}_t = \text{FUSE}(\boldsymbol{s}_t)$ (Fig 2, top center). Then this combined sensory information $\boldsymbol{f}_t$ from the current observation is fed into AGGREGATE, which is a long short term memory module (LSTM) [25]. AGGREGATE maintains an encoded internal model $\boldsymbol{a}_t$ 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 $\boldsymbol{a}_t = \text{AGGREGATE}(\boldsymbol{f}_1, \cdots, \boldsymbol{f}_t)$ (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 $\boldsymbol{a}_t$ 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 $\hat{V}_t(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t) = \text{DECODE}(\boldsymbol{a}_t)$. To do this, it first reshapes $\boldsymbol{a}_t$ into a sequence of small 2-D 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 to the appropriate viewgrid positions.

Acting to select the next viewpoint to observe Finally, ACT processes the aggregate code $\boldsymbol{a}_t$ to issue a motor command $\boldsymbol{\delta}_t = \text{ACT}(\boldsymbol{a}_t)$ (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 3-D environment). Upon execution, the observation's pose updates for the next timestep to $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\delta}_t$. For $t = 1$, $\boldsymbol{\theta}_1$ is randomly sampled.

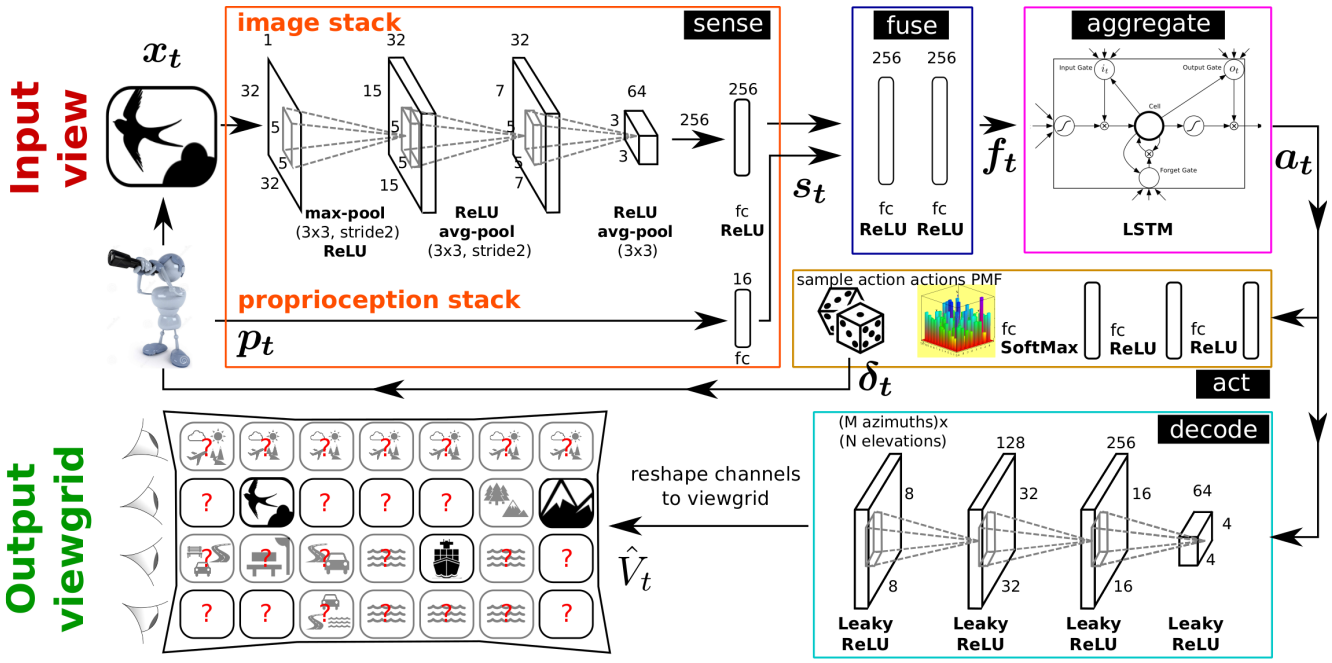

Figure 2. Architecture of our active observation completion system. While the input-output pair shown here is for the case of 360° scenes, we use the same architecture for ModelNet. See Sec. 3.2 for details.

Internally, ACT first produces a distribution over all possible actions, and then samples $\boldsymbol{\delta}_t$ from this distribution. Motions in the real world are constrained to be continuous, so we restrict ACT to select "small" actions (details below). Due to the sampling operation, ACT is a *stochastic* neural network [42]. Once the new viewpoint $\boldsymbol{\theta}_{t+1}$ 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 $\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_j)$ for all viewpoints $\boldsymbol{\theta}_j, 1 \leq j \leq MN$.

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, $L_T(X) = \sum_i d(\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_i), \boldsymbol{x}(X, \boldsymbol{\theta}_i))$, where $d(.)$ is a distance function. However, this loss function requires viewpoint coordinates to be registered exactly in the input 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 Δ_0 from the target viewgrid, and Δ_0 must be included in the loss function:

$$L_T(X) = \sum_{i=1}^{MN} d(\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_i + \Delta_0), \boldsymbol{x}(X, \boldsymbol{\theta}_i)). \quad (1)$$

We set $d(.)$ to be the per-pixel squared $\mathcal{L}_2$ distance. With this choice, the agent expresses its uncertainty by averaging over the modes of its beliefs about unseen views. In principle, $d(.)$ could be replaced with other metrics. In particular, a GAN-style loss [17] 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, we aim to resolve uncertainty over time to converge to the *correct* model.

Note that Δ_0 is used only at training time and only to compute the loss. The system cannot use azimuth knowledge while generating the reconstructions, since this will not be available at test time. As stated earlier, the system can only sense elevation (by measuring gravity), so Δ_0 is unknown. 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 <view, viewpoint> 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 this loss, we employ a combination of stochastic gradient descent (using backpropagation through time to handle recurrence) and REINFORCE [58], as in [41]. 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 neu-

ral 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, backpropagating to encourage ACT behaviors that led to high rewards. To backpropagate 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:

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

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 pretrained configurations. This helps avoid poor local minima and enables much faster convergence.

3.4. Unsupervised policy transfer to unseen tasks

The full scene/3D object encompasses all potentially useful information for *any* task. It is therefore reasonable to expect that our active observation-based policies seek generic information useful to many tasks.

However, we propose to also empirically verify this by *transferring* the learned observation completion policy to new unseen tasks in unseen environments. To do this, we plug in our unsupervised active observation completion policies into the active categorization system of [27]. At training time, we train two models: an end-to-end model for active categorization using random policies following the architecture of [27] (model "A"), and an active observation completion model ("model B"). Note that the observation completion model is trained on unsupervised unseen environments that may even be from categories that are not in model A's target set. 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 would produce explorations for the new categorization task too.

4. Experiments

We now validate our approach for learning efficient look-around behavior. We examine the effectiveness of the active completion policies for faster reconstruction (Sec 4.1), as well as their utility for transferring look-around policies trained without supervision to a specific 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** [61], our limited field-of-view (45°) agent attempts to complete an omnidirectional scene. SUN360 has spherical panoramas of diverse categories. We use the 26-category subset used in [61, 27]. The viewgrid has 32 px $\times$ 32 px views from 5 camera elevations (-90,-45,...,90°) and 8 azimuths (45,90,...,360°). At each timestep, the agent moves within a 3 elevations$\times$5 azimuths neighborhood from the current position. We set training episode length $T = 4$ for training speed.

On **ModelNet** [60], our agent manipulates a 3-D object to complete its image-based shape model of the object. ModelNet has two pre-specified subsets of object 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 32x32 views from 7 camera elevations (0,$\pm$30,$\pm$60,$\pm$90) and 12 azimuths (30,60,...,360°). Per-timestep motions are allowed within the 5$\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 uniformly 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.

- `large-action+` is the same as `large-action`, except that it chooses a different action at the highest/lowest elevation to avoid getting stuck. e.g., if `large-action` repeatedly chose to go up by 30°, it would get stuck at 90°. `large-action+` would instead change direction to go down.

- `peek-saliency` moves to the most salient view within reach at each timestep, using a popular saliency metric [21]. 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.

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 (6 on SUN360, 4 on ModelNet), 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 synthetic CAD model views; what is informative about an object's shape is much more complex than what low-level unsupervised saliency can measure.

Recall that the system has access to the camera elevation. It is therefore interesting to ask: does our approach exploit this knowledge to simply sample some elevations more than others? For instance, perhaps views from a horizontal camera position (elevation $0°$) are more informative than others. Upon investigation, we find that this is not true of our approach in practice. In particular, our learned policy samples all elevations uniformly on both SUN360 and ModelNet data.

The plots in Figure 3 further show how error behaves as a function of time, including at time $t > T$, never experienced in training. With perfect information aggregation, all methods should asymptotically approach zero error at high t, which diminishes the value of intelligent exploration.[2] In practice, all methods show consistent improvement up to $t = T$, with sharpest error drops for `ours`. For $t > T$, behavior is more unpredictable since the LSTM cell in AGGREGATE is operating outside its training range, and may not effectively aggregate information. Despite this, `ours` shows strong improvement on SUN360. On ModelNet unseen classes, error flattens, and on ModelNet-seen classes, error begins to rise. In all cases, `ours` outperforms all baselines significantly at *all* t.

Fig 5 and Fig 6 present some completion episodes. As our system explores, the rough "shape" of the target scene or object emerges in its viewgrid predictions. 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.

4.3. Unsupervised policy transfer results

As we demonstrate above, our approach successfully trains unsupervised exploratory policies for efficient generic information acquisition using our active observation completion framework. Following the approach described in Sec 3.4, we now test how well this policy transfers to a new task with new data from unseen categories: the ModelNet-10 active categorization setting of [27].

[2] This fact, together with training speed considerations, is why we limit the training time budget to $T = 4$.

Table 1. Per-pixel mean-squared error (MSE×1000) with episode length set to training length $T = 4$ (after 3 actions), 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`	33.90	13.96%	3.46	9.66%	6.22	15.72%
`large-action`	31.14	20.96%	3.44	10.18%	6.16	16.53%
`large-action+`	29.70	24.62%	3.38	11.75%	6.00	18.70%
`peek-saliency` [21]	30.35	22.97%	3.47	9.40%	6.35	13.96%
`ours`	**26.25**	**33.38%**	**3.25**	**15.14%**	**5.65**	**23.44%**

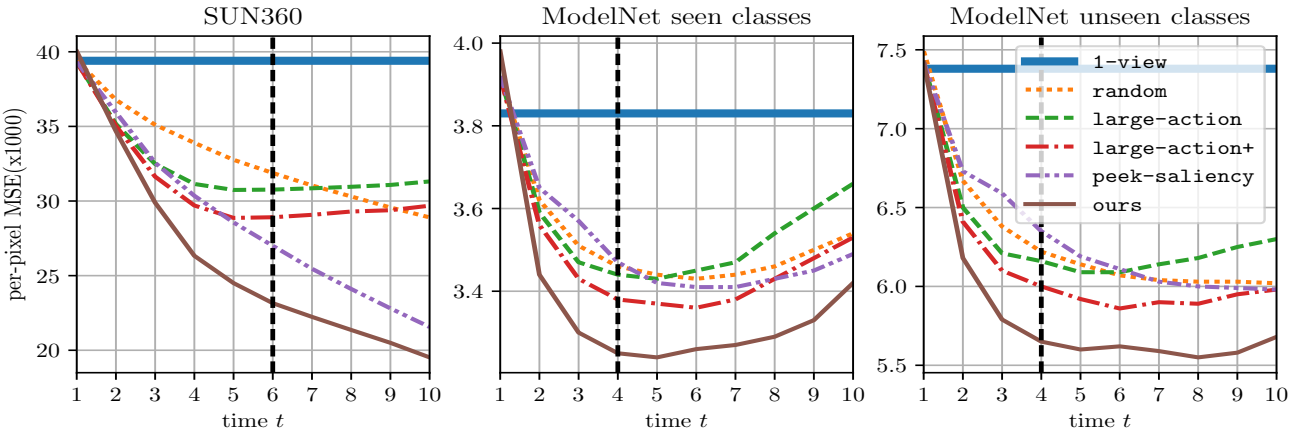

Figure 3. Active observation completion: per-pixel mean-squared error versus time for the three test datasets. Vertical black dotted line signifies the "look around budget" T targeted during training.

The experimental setup is identical to the active categorization setup described in [27], except that we use $T = 4$ timesteps and neighborhood size 5×5 (rather than $T = 6$ and 3×3 neighborhoods in [27]) to be consistent with our setup so far. The methods compared are as follows:

- `sup-policy` is a full end-to-end active categorization system trained using the "Lookahead active RNN" approach of [27] on ModelNet-10 training data.
- `1-view` is 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.
- `random-policy` is an active categorization system that selects random actions. This 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`. This is trained on ModelNet-10 training data.
- `ours (policy transfer)` follows the method described in Sec 3.4 to plug in our unsupervised active observation completion policies into the active categorization system of [27]. The active categorization model ("model A" in the description of Sec 3.4) is trained on ModelNet-10 training data with random policies — this is the same as the `random-policy` baseline above. We train the active observation completion model (model "B" in the notation of Sec 3.4) on *ModelNet-30* training data, disjoint from the target ModelNet-10 dataset classes.`ours (policy transfer)` thus attempts to transfer policies trained on an unsupervised task, to a supervised active task on disjoint data from novel classes.

Fig 4 shows the results. Our unsupervised policy transfer approach `ours (policy transfer)` performs on par with our earlier end-to-end active categorization policy on this task, easily outperforming `random-policy` and `1-view`. This is remarkable because the policy being employed in `ours (policy transfer)` is not only trained for the separate, unsupervised active observation completion task but it was also trained on data from disjoint classes.

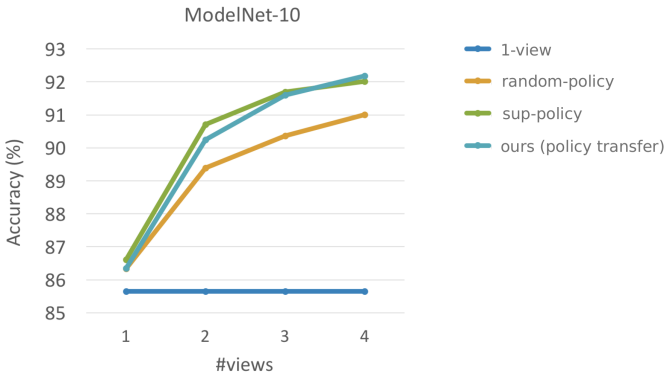

Figure 4. Evolution of ModelNet-10 active object categorization accuracy over time. `ours (policy transfer)` represents our unsupervised generic exploratory policy applied to the active categorization task. As seen here, it performs on par with `sup-policy`, which is the end-to-end supervised policy trained specifically for active categorization using the approach of [27].

This suggests that unsupervised exploratory tasks such as active observation completion could facilitate policy learning on massive unlabeled datasets in the future. Since policy learning is expensive in terms of data, computation and time, such exploratory policies once trained 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 which is widely employed in the passive 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 strong baselines. Our results showing successful application of our unsupervised exploratory policy for active recognition are the first demonstration of "policy transfer" between tasks to the best of our knowledge. These results hold great promise of general curiosity-driven exploration, an important step towards autonomous embodied visual agents.

References

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. In *IJCV*, 1988.

[3] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017.

[4] A. Andreopoulos and J. Tsotsos. A theory of active object localization. In *ICCV*, 2009.

[5] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[6] R. Bajcsy. Active perception. In *Proceedings of the IEEE*, 1988.

[7] D. Ballard. Animate vision. In *Artificial Intelligence*, 1991.

[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.

[9] N. Butko and J. Movellan. Optimal scanning for faster object detection. In *CVPR*, 2009.

[10] C. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstrution. In *ECCV*, 2016.

[11] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. In *TPAMI*, 2002.

[12] S. S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *WCNC 2003*, 2003.

[13] A. Dosovitskiy, J. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015.

[14] S. Edelman and H. H. Bülthoff. Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. In *Vision research*, 1992.

[15] A. G. Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object detection. In *CVPR*, 2015.

[16] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015.

[17] 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.

[18] 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.

[19] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017.

[20] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *CVPR*, 2017.

[21] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *NIPS*, 2006.

[22] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[23] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Graphics (TOG)*, 2007.

[24] S. Helmer, D. Meger, P. Viswanathan, S. McCann, M. Dockrey, P. Fazli, T. Southey, M. Muja, M. Joya, J. Little, et al. Semantic robot vision challenge: Current state and future directions. In *IJCAI workshop*, 2009.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, 1997.

[26] D. Jayaraman, R. Gao, and K. Grauman. Unsupervised learning through one-shot image-based shape reconstruction. In *arXiv*, 2017.

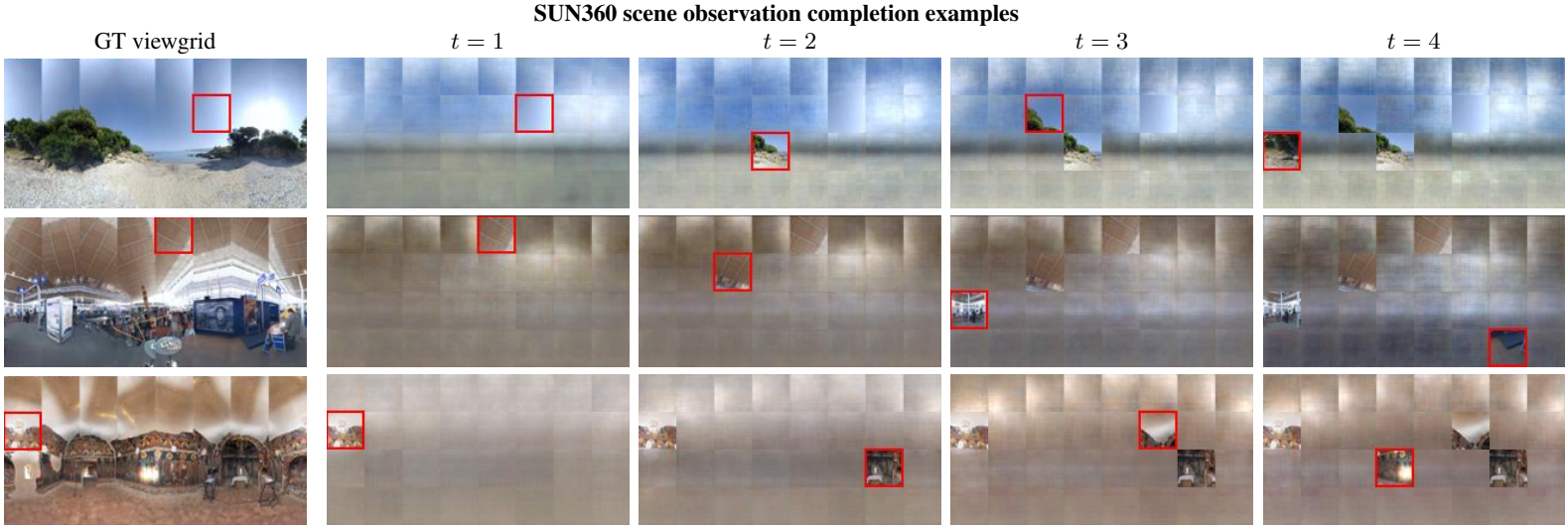

Figure 5. Best viewed on pdf. Each row is one episode of SUN360 panorama observation. Column 1 shows the ground truth viewgrid with a red square around the random starting view. Columns 2-5 show our method's viewgrid completions for $t = 1, \ldots, 4$ with red squares around selected views. As the model's 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 bright white light in the windows is never directly observed, yet the system has inferred its presence and rough shape by $t = 4$. **Row 3:** Observing a view naturally improves nearby view predictions, but more remarkably, sometimes even improves views that are far away. Here, at $t = 2$, after having seen an eye-level view, the agent appears to re-evaluate the implications of its first observation and correctly draw ceiling lights. The light background in the view observed at $t = 1$ could plausibly have been the color of a wall but, after the second view of the dark wall, the light background of the first view must be attributed to lighting.

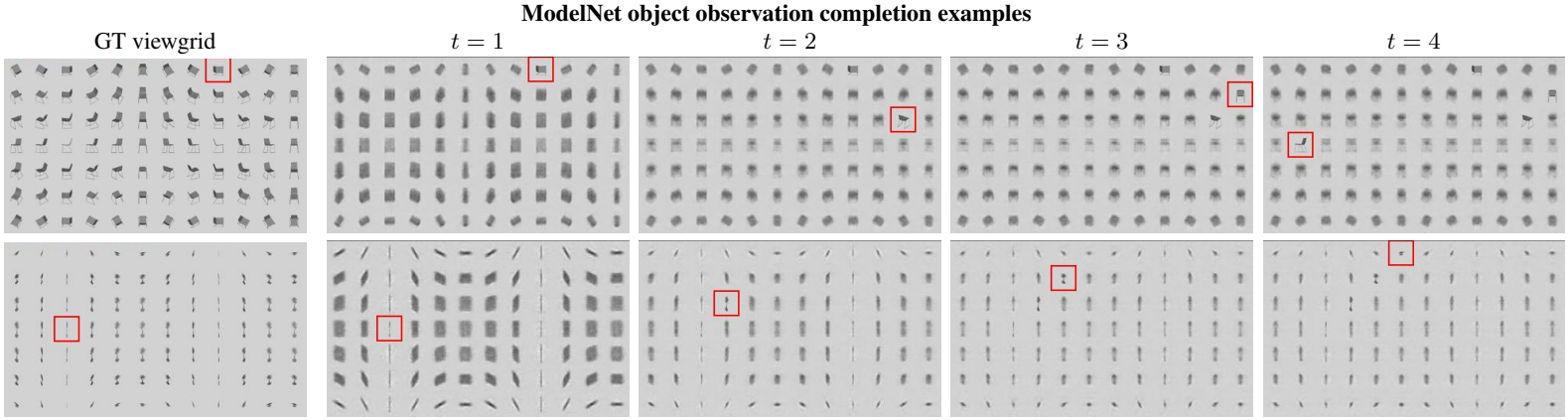

Figure 6. Best viewed on pdf with zoom. Each row represents one episode for a ModelNet object. Formatting is as in Fig 5. **Row 1:** The first view is overhead, and azimuthally aligned with one of the sides of an unseen category object (chair). Our agent chooses to move as far from this view as possible at $t = 2$, instantly forming a much more chair-like predicted viewgrid, which continues to improve afterwards. **Row 2:** Views of narrow faces are very uninformative, and at $t = 1$, the system guesses some a flat board-like surface. After successfully moving to better views, its predictions start resembling the ground truth tree object.

[27] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In *ECCV*, 2016.

[28] E. Johns, S. Leutenegger, and A. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016.

[29] P. J. Kellman and E. S. Spelke. Perception of partly occluded objects in infancy. In *Cognitive psychology*, 1983.

[30] A. Kim and R. M. Eustice. Perception-driven navigation: Active visual slam for robotic area coverage. In *ICRA*, 2013.

[31] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. In *IJRR*, 2008.

[32] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, 2007.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[34] T. Kulkarni, W. Whitney, P. Kohli, and J. Tenenbaum. Deep

convolutional inverse graphics network. In *NIPS*, 2015.

[35] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017.

[36] Y. Li, A. Fathi, and J. M. Rehg. Learning to predict gaze in egocentric video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3216–3223, 2013.

[37] 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.

[38] M. Malmir, K. Sikka, D. Forster, J. Movellan, and G. W. Cottrell. Deep Q-learning for active recognition of GERMS. In *BMVC*, 2015.

[39] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, pages 321–328, 2007.

[40] 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.

[41] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014.

[42] R. M. Neal. Learning stochastic feedforward networks. In *Tech Report*, 1990.

[43] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. In *RAS*, 2000.

[44] S. Palmer, E. Rosch, and P. Chase. Canonical perspective and the perception of objects. In *Attention and performance IX*, 1981.

[45] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[46] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 2012.

[47] M. Ranzato. On learning where to look. In *arXiv preprint arXiv:1405.5488*, 2014.

[48] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. In *TPAMI*, 2009.

[49] B. Schiele and J. Crowley. Transinformation for active object recognition. In *ICCV*, 1998.

[50] S. Soatto. Actionable information in vision. In *ICCV*, 2009.

[51] 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.

[52] K. C. Soska and S. P. Johnson. Development of three-dimensional object completion in infancy. In *Child development*, 2008.

[53] R. Spica, P. R. Giordano, and F. Chaumette. Active structure from motion: application to point, sphere, and cylinder. 2014.

[54] A. Torralba. Neurobiology of attention, chapter contextual influences on saliency. 2005.

[55] 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.

[56] B. Wang. Coverage problems in sensor networks: A survey. In *ACM CSUR*, 2011.

[57] D. Wilkes and J. Tsotsos. Active object recognition. In *CVPR*, 1992.

[58] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine learning*, 1992.

[59] J. Wu, T. Xue, J. Lim, Y. Tian, J. Tenenbaum, A. Torralba, and W. Freeman. Single image 3d interpreter network. In *ECCV*, 2016.

[60] 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.

[61] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012.

[62] B. Xiong and K. Grauman. Detecting snap points in egocentric video with a web photo prior. In *ECCV*, 2014.

[63] 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.

[64] 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.

[65] 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.

[66] X. Yu, C. Fermuller, C. L. Teo, Y. Yang, and Y. Aloimonos. Active scene recognition with vision and language. In *CVPR*, 2011.

[67] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2016.

[68] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. Efros. View synthesis by appearance flow. In *ECCV*, 2016.

[69] 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.