COMPUTER VISION

Emergence of exploratory look-around behaviors through active observation completion

Santhosh K. Ramakrishnan[1,2*†], Dinesh Jayaraman[3*], Kristen Grauman[1,2]

Standard computer vision systems assume access to intelligently captured inputs (e.g., photos from a human photographer), yet autonomously capturing good observations is a major challenge in itself. We address the problem of learning to look around: How can an agent learn to acquire informative visual observations? We propose a reinforcement learning solution, where the agent is rewarded for reducing its uncertainty about the unobserved portions of its environment. Specifically, the agent is trained to select a short sequence of glimpses, after which it must infer the appearance of its full environment. To address the challenge of sparse rewards, we further introduce sidekick policy learning, which exploits the asymmetry in observability between training and test time. The proposed methods learned observation policies that not only performed the completion task for which they were trained but also generalized to exhibit useful "look-around" behavior for a range of active perception tasks.

INTRODUCTION

Visual recognition has witnessed dramatic successes in recent years. Fueled by benchmarks composed of carefully curated web photos and videos, the focus has been on inferring semantic labels from human-captured images—whether classifying scenes, detecting objects, or recognizing activities (*1*–*3*). However, visual perception requires making not only inferences from observations but also decisions about what to observe. Methods that use human-captured images implicitly assume properties in their inputs, such as canonical poses of objects, no motion blur, or ideal lighting conditions. As a result, they gloss over important hurdles for robotic agents acting in the real world.

For an agent, individual views of an environment offer only a small fraction of all relevant information. For instance, an agent with a view of a television screen in front of it may not know whether 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 ought to be able to enter a new environment or pick up a new object and intelligently (non-exhaustively) "look around." The ability to actively explore would be valuable in both task-driven scenarios (e.g., a drone searches for signs of a particular activity) and 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). For example, consider a service robot that is moving around in an open environment without specific goals, waiting for future tasks like delivering a package from one person to another or picking up coffee from the kitchen. It needs to efficiently and constantly gather information so that it is well prepared to perform future tasks with minimal delays. Similarly, consider a search-and-rescue scenario, where a robot is deployed in a hostile environment, such as a burning building or earthquake collapse, where time is of the essence. The robot has to adapt to such new unseen environments and rapidly gather information that other robots and humans can use to effectively respond to situations that dynamically unfold over time (humans caught under debris, locations of fires, and presence of hazardous materials). Having a robot that knows how to explore intelligently can be critical in such scenarios, reducing risks for people while providing an effective response.

Any such scenario brings forth the question of how to collect visual information to benefit perception. A naïve strategy would be to gain full information by making every possible observation—that is, looking around in all directions or systematically examining all sides of an object. However, observing all aspects is often inconvenient if not intractable. Fortunately, in practice, not all views are equally informative. The natural visual world contains regularities, suggesting that not every view needs to be sampled for accurate perception. For instance, humans rarely need to fully observe an object to understand its three-dimensional (3D) shape (*4*–*6*), and one can often understand the primary contents of a room without literally scanning it (*7*). In short, given a set of past observations, some new views are more informative than others (Fig. 1).

This fact leads us to investigate the question of how to effectively look around: How can a learning system make intelligent decisions about how to acquire new exploratory visual observations? We propose a solution based on "active observation completion": An agent must actively observe a small fraction of its environment so that it can predict the pixelwise appearances of unseen portions of the environment.

Our problem setting relates to but is distinct from previous work in active perception, intrinsic motivation, and view synthesis. Although there is interesting recent headway in active object recognition (*8*–*11*) and intelligent search mechanisms for detection (*12*–*14*), such systems are supervised and task specific—limited to accelerating a predefined recognition task. In reinforcement learning (RL), intrinsic motivation methods define generic rewards, such as novelty or coverage (*15*–*17*), that encourage exploration for navigation agents, but they do not self-supervise policy learning in an observed visual environment, nor do they examine transfer beyond navigation tasks. View synthesis approaches use limited views of the environment along with geometric properties to generate unseen views (*18*–*22*). Whereas these methods assume individual human-captured images, our problem requires actively selecting the input views themselves. Our primary goal is not to synthesize unseen views but rather to use novel view inference as a means to elicit intelligent exploration policies that transfer well to other tasks.

In the following, we first formally define the learning task, overview our approach, and present results. Then, after the results, we discuss limitations of the current approach and key future directions, followed

[1]Department of Computer Science, University of Texas at Austin, Austin, TX, USA. [2]Facebook AI Research, Austin, TX, USA. [3]Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA, USA.
*These authors contributed equally to this work.
†Corresponding author. Email: srama@cs.utexas.edu

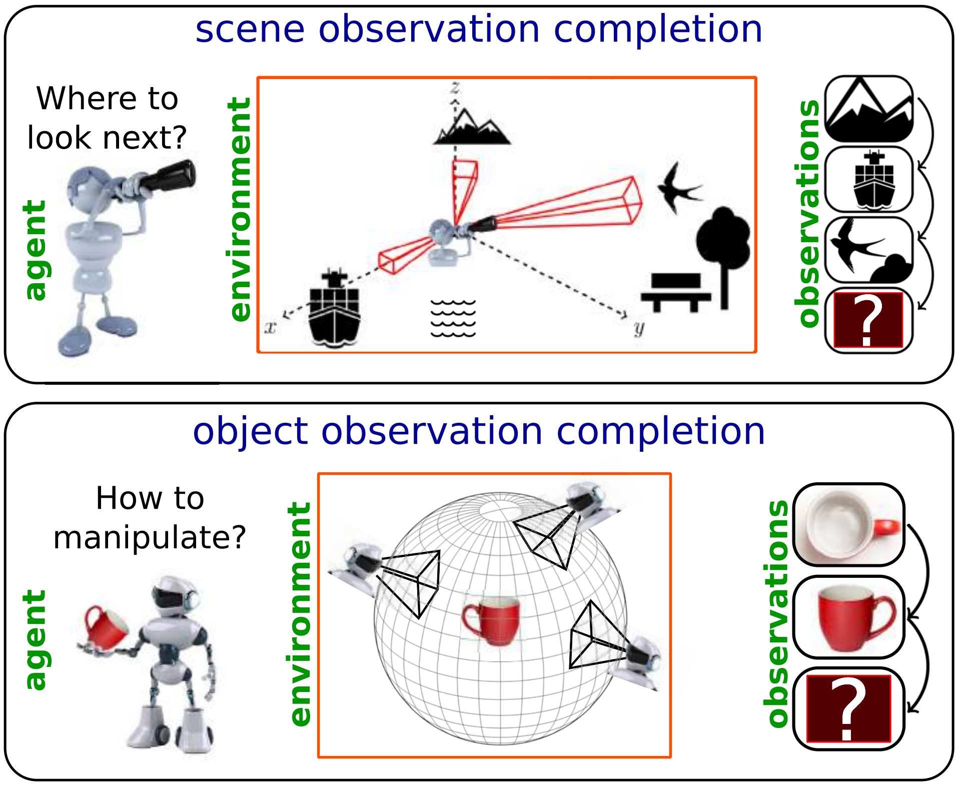


Fig. 1. Looking around efficiently is a complex task requiring the ability to reason about regularities in the visual world using cues like context and geometry. Top: An agent that has observed limited portions of its environment can reasonably predict some unobserved portions (e.g., water near the ship) but is much more uncertain about other portions. Where should it look next? Bottom: An agent inspecting a 3D object. 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, nor to a specific perception task. Instead, the look-around policies ought to benefit the agent exploring new, unseen environments and performing tasks unspecified when learning the look-around behavior.

by Materials and Methods—an overview of the specific deep networks and policy learning approaches we developed. This article expands upon our two previous conference papers (*23*, *24*).

Active observation completion

Our goal is to learn a policy for controlling an agent's camera motions such that it can explore novel environments and objects efficiently. To this end, we formulate an unsupervised learning objective based on active observation completion. The main idea is to favor sequences of camera motions that make the unseen parts of the agent's surroundings easier to predict. The output is a look-around policy equipped to gather new images in new environments. As we will demonstrate in results, it prepares the agent to perform intelligent exploration for a wide range of perception tasks, such as recognition, light source localization, and pose estimation.

Problem formulation

The problem setting is formally stated as follows. The agent starts by looking at a novel environment (or object) X from some unknown viewpoint (*25*). It has a budget T of time to explore the environment. The learning objective is to minimize the error in the agent's pixelwise reconstruction of the full—mostly unobserved—environment using only the sequence of views selected within that budget. To do this, the agent must maintain an internal representation of how the environment would look conditioned on the views it has seen so far.

We represent the entire environment as a "viewgrid" containing views from a discrete set of viewpoints. To do this, we evenly sample N elevations from −90° to 90° and M azimuths from 0° to 360° and form all MN possible (elevation, azimuth) pairings. The viewgrid is then denoted by $V(X) = \{x(X, \theta^{(i)}) \mid 1 \leq i \leq MN\}$, where $x(X, \theta^{(i)})$ is the 2D view of X from viewpoint $\theta^{(i)}$, which is the ith pairing. More generally, $\theta^{(i)}$ could capture both camera angles and position; however, to best exploit existing datasets, we limited our experiments to camera rotations alone with no translation movements.

The agent expends its time budget T in discrete increments by selecting $T - 1$ camera motions in sequence. Each camera motion comprises an actively chosen "glimpse." At each time step, the agent gets an image observation x_t from the current viewpoint. It then makes an exploratory motion (a_t) based on its policy π. When the agent executes action $a_t \in \mathcal{A}$, the viewpoint changes according to $\theta_{t+1} = \theta_t + a_t$. For each camera motion a_t executed by the agent, a reward r_t is provided by the environment. Using the view x_t, the agent updates its internal representation of the environment, denoted $\hat{V}(X)$. Because camera motions are restricted to have proximity to the current camera angle and candidate viewpoints partially overlap, the discrete action space promotes efficiency without neglecting the physical realities of the problem [following (*8*, *9*, *23*, *26*)]. During training, the full viewgrids of the environments are available to the agent as supervision. During testing, the system must predict the complete viewgrid, having seen only a few views within it.

We explored our idea in two settings (Fig. 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" (i.e., predict pixelwise).

Posing the active view acquisition problem in terms of observation completion has two key advantages: generality and low-cost (label-free) training data. The objective is general in the sense that pixelwise reconstruction places no assumptions about the future task for which the glimpses will be used. The training data are low cost, because no manual annotations are required; the agent learns its look-around policy by exploring any visual scene or object. This assumes that capturing images is much more cost-effective than manually annotating images.

Approach overview

The active observation completion task poses three major challenges. First, to predict unobserved views well, the agent must learn to understand 3D relationships from very few views. Classic geometric solutions struggle under these conditions. Instead, our reconstruction must draw on semantic and contextual cues. Second, intelligent action selection is essential 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 never encountered during training, yet still must intelligently choose where it would be valuable to look next.

As a core solution to these challenges, we present an RL approach for active observation completion (Fig. 2) (*23*). Our RL approach uses a recurrent neural network to aggregate information over a sequence of views; a stochastic neural network uses that aggregated state and current observation to select a sequence of useful camera motions. The agent is rewarded on the basis of its predictions of unobserved views. It therefore learns a policy to intelligently select actions (camera motions) to maximize the quality of its predictions. During training, the complete viewgrid is known, thereby allowing the agent to "self-supervise" its policy learning, meaning it learns without any human-provided labels. See Materials and Methods below for the details of our approach.

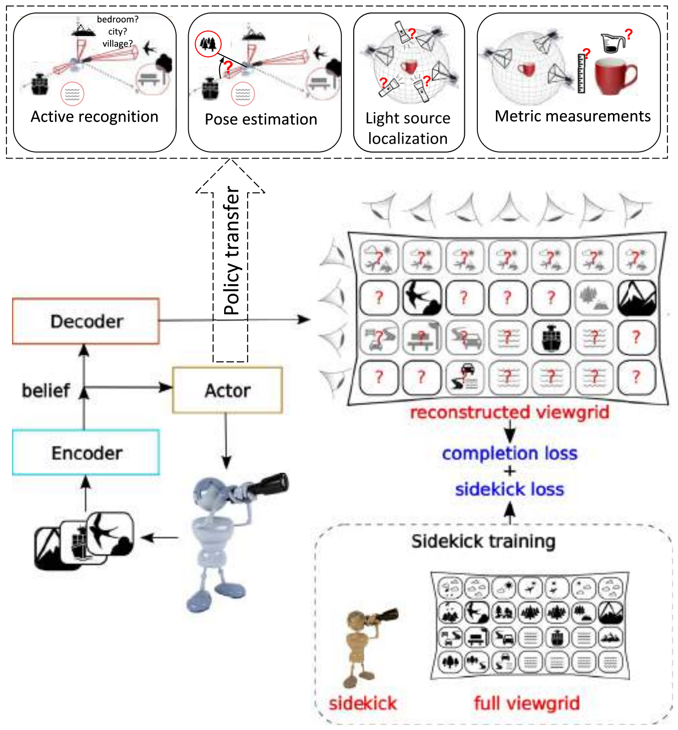


Fig. 2. Approach overview. The agent (actor) encodes individual views from the environment and aggregates them into a belief state vector. This belief is used by the decoder to get the reconstructed viewgrid. The agent's incomplete belief about the environment leads to uncertainty over some viewpoints (red question marks). To reduce this uncertainty, the agent intelligently samples more views based on its current belief within a fixed time budget *T*. The agent is penalized on the basis of the reconstruction error at the end of *T* steps (completion loss). In addition, we provide guidance through sidekicks (sidekick loss), which exploit the full viewgrid—only at training time—to alleviate uncertainty in training due to partial observability. The learned exploratory policy is then transferred to other tasks (top row shows four tasks we consider).

Our model judges the quality of viewgrid reconstruction in the pixel space so as to maintain generality: All pixels for the full scene (or 3D object) would encompass all potentially useful visual information for any task. Hence, 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. For example, if the target tasks only require semantic segmentation labels, then the predictions could be in the space of object labels instead.

RL approaches often suffer from costly exploration stages and partial state observability. In particular, an active visual agent has to take a long series of actions purely based on the limited information available from its first-person view (*23*, *27*–*29*). The most effective viewpoint trajectories are buried among many mediocre ones, impeding the agent's exploration in complex state-action spaces.

To address this challenge, as the second main technical contribution of this work, we introduce "sidekick policy learning." In the active observation completion task, there is a natural asymmetry in observability: Once deployed, an active exploration agent can only move the camera to look around nearby, yet during training, it can access omnidirectional viewpoints. Existing methods facing this asymmetry simply restrict the agent to the same partial observability during training (*8*, *10*, *23*, *26*, *27*, *30*). In contrast, our sidekick approach introduces reward shaping and demonstrations that leverage full observability during training to precompute the information content of each candidate glimpse. The sidekicks then guide the agent to visit information hot spots in the environment or sample information-rich trajectories while accounting for the fact that observability is only partial during testing (*24*). By doing so, sidekicks accelerate the training of the actual agent and improve the overall performance. We use the name "sidekick" to signify how a sidekick to a hero (e.g., in a comic or movie) provides alternate points of view, knowledge, and skills that the hero does not have. In contrast to an "expert" (*31*, *32*), a sidekick complements the hero (agent), yet cannot solve the main task at hand by itself. See Materials and Methods below for more details.

We show that the active observation completion policies learned by our approach serve as exploratory policies that are transferable to entirely new tasks and environments. Given a new task, rather than train a policy with task-specific rewards to direct the camera, we drop in the pretrained look-around policy. We demonstrate that policies learned via active observation completion transfer well to several semantic and geometric estimation tasks, and they even perform competitively with supervised task-specific policies (please see the look-around policy transfer section in Results).

RESULTS

We next present experiments to evaluate the behaviors learned by the proposed look-around agents.

Datasets

For benchmarking and reproducibility, we evaluated our approach on two widely used datasets.

SUN360 dataset for scenes

For this dataset, our limited field-of-view (60°) agent attempts to complete an omnidirectional scene. SUN360 (*33*) has spherical panoramas of 26 diverse categories. The dataset consists of 6174 training, 1013 validation, and 1805 testing examples. The viewgrid has 32 pixels–by–32 pixel resolution 2D images sampled from $M = 4$ camera elevations (−67.5°, −22.5°, 22.5°, 67.5°) and $N = 8$ azimuths (45°, 90°,…, 360°).

ModelNet dataset for objects

For this dataset, our agent manipulates a 3D object to complete its viewgrid of the object seen from all viewing directions. The viewgrid constitutes an implicit image-based 3D shape model. ModelNet (*10*) has two subsets of computer-aided design (CAD) models: ModelNet-40 (40 categories) and ModelNet-10 (a 10-category subset of ModelNet-40). Excluding the ModelNet-10 classes, ModelNet-40 consists of 6085 training, 327 validation, and 1310 testing examples. ModelNet-10 consists of 3991 training, 181 validation, and 727 testing examples. The viewgrid has 32 × 32 resolution 2D images sampled from $M = 6$ camera elevations (−75°, −45°,…, 45°, 75°) and $N = 10$ azimuths (20°, 56°, 92°,…, 344°) (*34*). We rendered the objects using substantial lighting variations to increase difficulty in perception. To test the agent's ability to generalize to previously unseen categories, we always tested on object categories in ModelNet-10, which are unseen during training.

For both datasets, at each time step, the agent moved within a 5 elevations–by–5 azimuths neighborhood from the current position. Requiring nearby motions reflects that the agent cannot teleport, and it ensures that the actions have approximately uniform real-world cost. Balancing task difficulty (harder tasks require more views) and training speed (fewer views are faster) considerations, we set the training episode length $T = 4$ a priori. By training for a target budget T, the agent has to learn nonmyopic behaviors to best use the expected exploration time. Note that although further increasing T during training increased training costs considerably, doing so naturally led to better reconstructions (please see the Supplementary Materials for longer episode results).

Baselines

We tested our active completion approach with and without sidekick policy learning (*35*)—`lookaround` and `lookaround+spl`, respectively—compared with a variety of baselines:

1. `one-view` is our method trained with $T = 1$. No information aggregation or action selection was performed by this baseline.
2. `rnd-actions` is identical to our approach, except that the action selection module was replaced by randomly selected actions from the pool of all possible actions.
3. `large-actions` chooses the largest allowable action repeatedly. This tested whether far-apart views were sufficiently informative.
4. `peek-saliency` moves to the most salient view within reach at each time step, using a popular saliency metric (*36*). To avoid getting stuck in a local saliency maximum, it does not revisit seen views. Note that this baseline peeks at neighboring views before action selection to measure saliency, giving it an unfair and impossible advantage over our methods and the other baselines.

These baselines all used the same network architecture as our methods, differing only in the exploration policy that we sought to evaluate. In the interest of evaluating on a wide range of starting positions, we evaluated each method MN times on each test viewgrid, starting from all possible viewpoints.

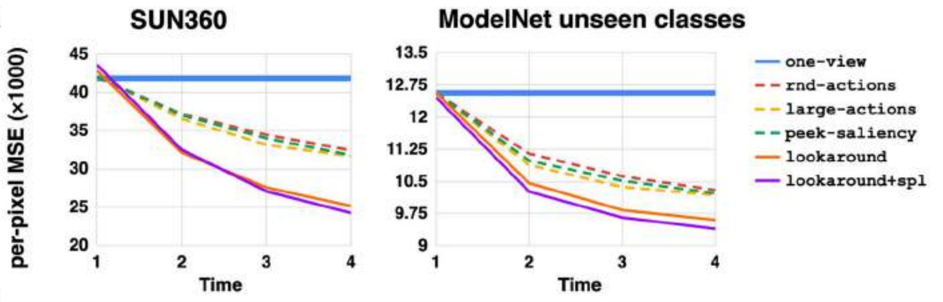
Active observation completion results

We show the results of scene and object completion on SUN360 and ModelNet (unseen classes) in Fig. 3B. The metrics "average" and "adversarial" measure the expected value of the average and maximum pixelwise mean squared errors (MSEs) over all starting points for a single sample, respectively. Whereas the former measures the average expected performance, the latter measures the worst-case performance when starting from the hardest place in each sample (averaged over examples). We additionally report the relative improvement of each model over `one-view` to isolate the gains obtained due to action selection over a pretrained $T = 1$ model. Because all methods shared the same pretraining stage of `one-view`, this metric provides an apples-to-apples measure of how well the different strategies for moving performed. All methods were evaluated over $T = 4$ time steps in accordance with the training budget unless stated otherwise.

As expected, all methods that acquired multiple glimpses outperformed `one-view` by taking advantage of the extra information that was available from additional views. Both the `lookaround` and `lookaround+spl` approaches substantially outperformed the others on all settings. The `peek-saliency` agent hovered near the most salient views in the neighborhood of the starting view because nearby views tended to have similar saliency scores. The `large-actions` agent's accuracy often tended to saturate near the top or bottom of the viewgrid after reaching the environment boundaries. Compared with these behaviors, intelligent sampling of actions using our learned policy led to substantial improvements. Using sidekicks in `lookaround+spl` improved performance and convergence speed. This is consistent with our results reported in (*24*) and demonstrates the advantage of using sidekicks. The faster convergence of `lookaround+spl` is shown in the Supplementary Materials.

Whereas Fig. 3B shows the agents' ultimate ability to infer the entire scene (object), Fig. 3A shows the reconstruction errors as a

A



B

Method	SUN360				ModelNet unseen classes			
	average		adversarial		average		adversarial	
	mean ↓	% ↑	mean ↓	% ↑	mean ↓	% ↑	mean ↓	% ↑
`one-view`	41.85	-	70.44	-	12.57	-	20.32	-
`rnd-actions`	32.42	22.52	54.66	22.39	10.30	18.06	14.01	31.04
`large-actions`	31.68	24.28	42.56	39.57	10.18	18.97	13.04	35.79
`peek-saliency`	31.80	24.02	45.24	35.76	10.22	18.67	12.78	37.05
`lookaround`	25.14	39.91	**30.44**	**56.79**	9.60	23.60	12.01	40.88
`lookaround+spl`	**24.24**	**42.06**	30.75	56.34	**9.40**	**25.23**	**11.85**	**41.67**

Fig. 3. Scene and object completion accuracy under different agent behaviors. (**A**) Pixelwise MSE errors versus time on both datasets as more glimpses are acquired. (**B**) Average/adversarial MSE error ×1000 (↓ lower is better) and corresponding improvements (%) over the one-view model (↑ higher is better) on both datasets after all T glimpses are acquired.

function of time. As we can see, the error reduced consistently over time for all methods, but it dropped most sharply for `lookaround` and `lookaround+spl`. Faster reduction in the reconstruction error indicates more efficient information aggregation.

Visualizations of the agent's evolving internal belief state echo this quantitative trend. Figure 4 shows observation completion episodes from the `lookaround` agent along with the ground truth viewgrid, viewing angles selected by the agent, and reconstruction errors over time. We show the SUN360 viewgrids in equirectangular projection for better visualization. Initially, the agent exhibited considerable uncertainty in its belief, as seen in the poorly decoded reconstructions and large MSE values. However, over time, it actively sampled views that quickly improved the reconstruction quality.

Figures 5 and 6 visualize the ultimate reconstructions after all *T* glimpses were acquired (*37*). For contrast, we also display the results for `rnd-actions` in Fig. 5. The policies learned by our agent led to more realistic and accurate reconstructions. Although the agent only saw about 15% of all the pixels, its choice of informative glimpses allowed it to anticipate the remainder of the novel scene or object. Movie S1 in the Supplementary Materials shows walkthroughs of the reconstructed environments from the agent's egocentric point of view.

Look-around policy transfer

Having shown that our unsupervised approach successfully trained policies to acquire visual observations useful for completion, we next tested how well the policies transfer to new tasks. Recall, our hypothesis is that the glimpses acquired to maximize completion accuracy will transfer well to solve perception tasks efficiently, because they were chosen to reveal maximal information about the full environment or object.

To demonstrate transfer, we first trained a `rnd-actions` model for each of the target tasks ("model A") and a `lookaround` model for the active observation completion task ("model B"). The policy from model B was then used to select actions for the target task using model A's task head (see details in the unsupervised policy transfer section in Materials and Methods). In this way, the agent learned to solve the task given arbitrary observations, then inherited our intelligent look-around

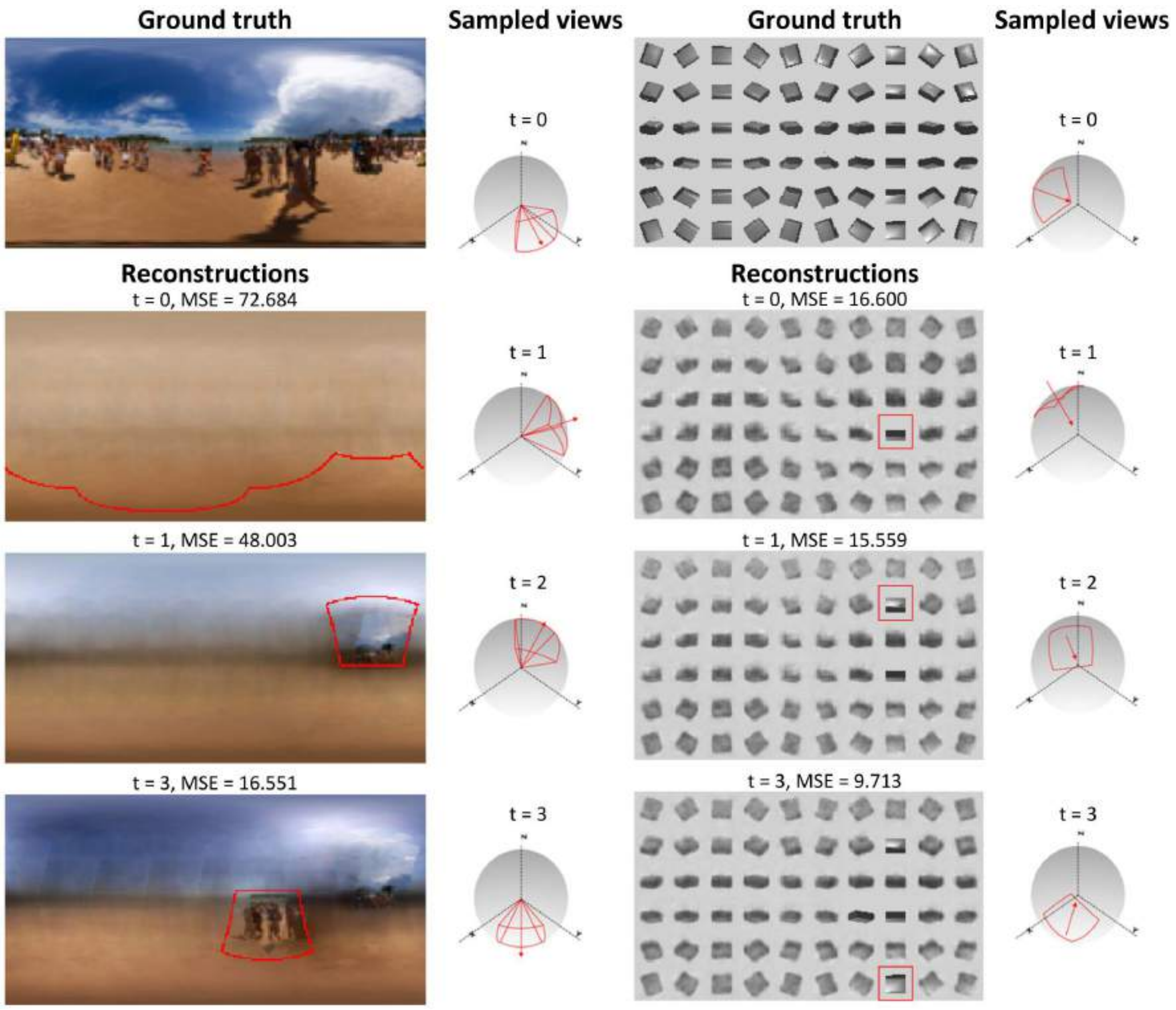


Fig. 4. Episodes of active observation completion for SUN360 (left) and ModelNet (right). For each example, the first row on the left shows the ground-truth viewgrid; the subsequent rows on the left show the reconstructions at times $t = 0,1, T - 1 = 3$ along with the pixelwise MSE error (×1000) and the agent's current glimpse (marked in red). On the right, the sampled viewing angles of the agent at each time step are shown on the viewing sphere (marking the agent's viewpoint and field of view using a red arrow and outline on the sphere). The reconstruction quality improves over time as it quickly refines the scene structure and object shape.

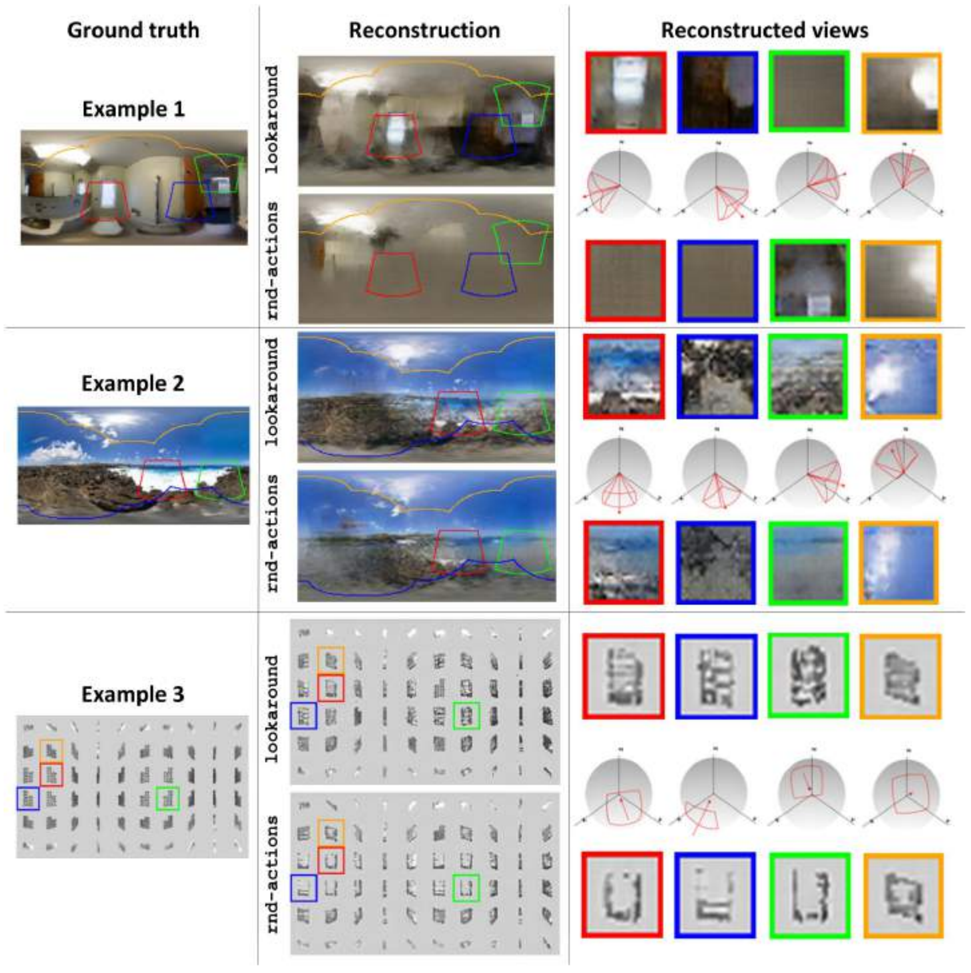


Fig. 5. Three examples of reconstructions after $T = 6$ glimpses. The first column shows the ground-truth viewgrids (equirectangular projections for SUN), the second column shows the corresponding generative adversarial network (GAN)–refined reconstructions of the `lookaround` and `rnd-actions` agents, and the third column shows handpicked unseen views (marked on the ground-truth) and the corresponding angles. We chose $T = 6$ to generate more complete images. Please see the Supplementary Materials for more GAN refinement details. Best viewed on PDF with zoom. Using an intelligent policy, `lookaround` captures more information from the scene, leading to more realistic reconstructions (examples 1 and 3). Although `rnd-actions` leads to realistic reconstructions on example 2, its textures and content differ from the ground truth, especially on the ground. Note that the bounding boxes over views are warped to curves on the equirectangular projection for SUN360.

policy to (potentially) solve the task more quickly—with fewer glimpses. The transfer is considered a success if the look-around agent can solve the task with similar efficiency as a supervised task-specific policy, despite being unsupervised and task agnostic. We tested policy transferability for the following four tasks.

Task 1: Active categorization

The first task is category recognition: The agent must produce the category name of the object or scene it is exploring. We plugged look-around policies into the active categorization system from (*8*) and followed a similar setup. For ModelNet, we trained model A on ModelNet-10 training objects and the active observation completion model (model B) on ModelNet-40 training objects, which were disjoint classes from those in the target ModelNet-10 dataset. For SUN360, both models were trained on SUN360 training data. We replicated the results from (*8*) and used the corresponding architecture and training strategies. In particular, the classification head was trained with a cross-entropy loss over the set of classes, and the supervised reward function for policy learning was the negative of the classification loss at the end of the episode. We refer the readers to (*8*) for the full details. Performance was measured using classification accuracy on the test set.

Task 2: Active surface area estimation

The second task is surface area estimation. The agent starts by looking at some view of the object and must intelligently select subsequent viewing angles to estimate the 3D object's surface area. The task is relevant for a robot that needs to interact with an unfamiliar object. The 3D models from ModelNet-10 were converted into 50 voxel–by–50 voxel–by–50 voxel occupancy grids. The true surface area was the number of unoccupied voxels that were adjacent to occupied voxels. Estimation was posed as a regression task where the agent predicted a normalized metric value between 0 and 1. Performance was measured using the relative MSE between predicted and ground truth areas on the test set; i.e., if the ground truth and predicted areas are m_g and m_p, respectively, then the error for one example is $((m_g - m_p)/m_g)^2$. This normalized the error so that it remained comparable across objects of different sizes.

Task 3: Active light source localization

In the third task, the agent is required to localize the sources of light surrounding the 3D object. To design a controlled experimental setting when rendering the ModelNet objects, we placed a single light source randomly at any one of two possible azimuths and four possible elevations relative to the object (see fig. S2). The task was posed as a four-way classification problem where the agent was required to identify the correct elevation (irrespective of the azimuth such that there can be no unfair orientation bias). Performance was measured using localization accuracy on the test set.

Task 4: Active pose estimation

The fourth task is camera pose estimation. Having explored the environment, the agent is required to identify the elevation and relative azimuth of a given reference view. We used a simple solution to this problem. By using the agent's reconstruction after T time steps, we measured the ℓ_2 distance between the given view and each of the reconstructed views. The elevation and azimuth of the reconstructed view leading to the smallest ℓ_2 distance was predicted as the pose. The agent used its own decoder as opposed to the decoder from `rnd-actions` as done in previous tasks. We did not evaluate pose

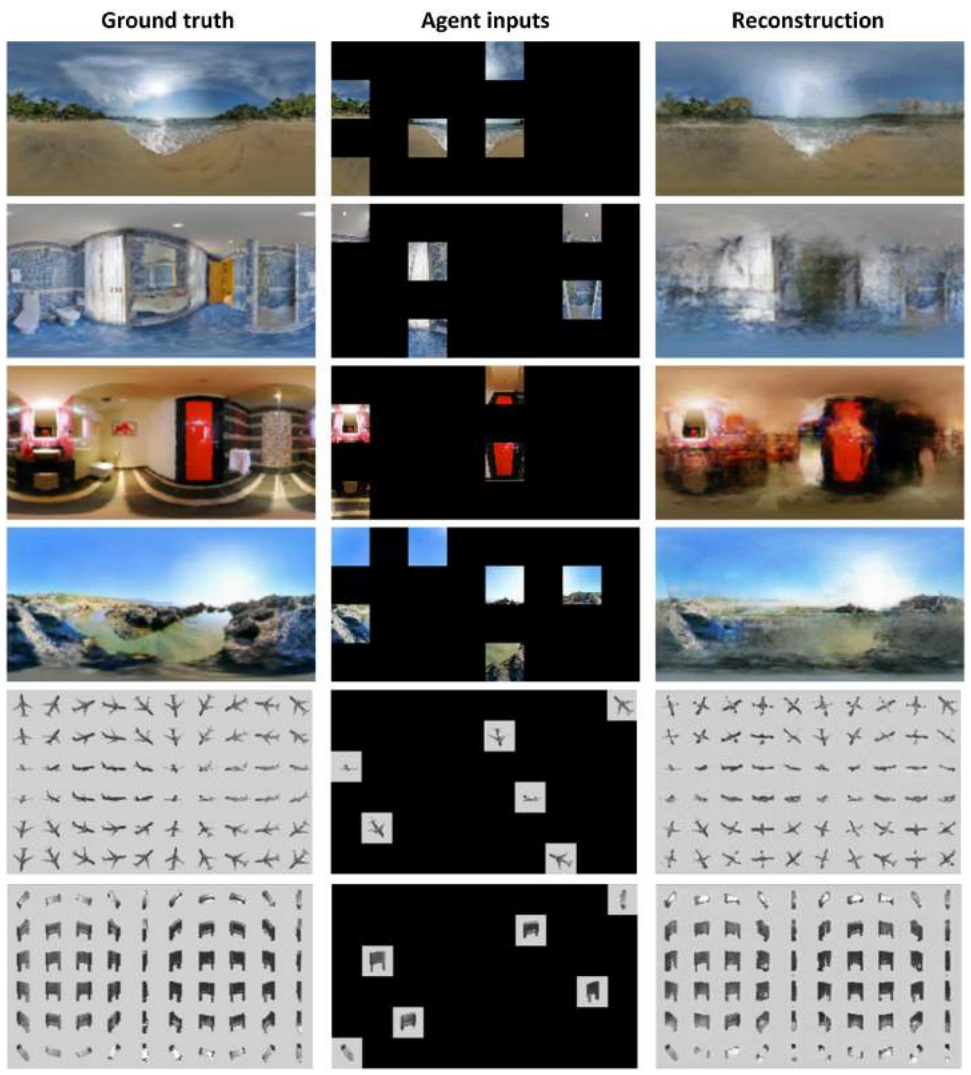


Fig. 6. The ground-truth 360 panorama or viewgrid, agent glimpse inputs, and final GAN-refined reconstructions for multiple environments from SUN360 and ModelNet. See also movie S1 provided in the Supplementary Materials.

estimation on ModelNet due to the ambiguity arising from symmetric objects. The models were evaluated using the absolute angular error (AE) in (i) elevation and (ii) azimuth predictions, denoted by "AE azim." and "AE elev." in Table 1. During evaluation, the starting positions of the agent were selected uniformly over the grid of views. The reference view was sampled randomly from the viewgrid for each episode.

For baselines, we used `one-view`, `rnd-actions`, `large-actions`, `peek-saliency` (defined in the previous section), and `supervised`. `supervised` is a policy that was trained specifically on the training objective for each task, i.e., with task-specific rewards.

We compared the transfer of `lookaround` and `lookaround+spl` with these baselines in Table 1. The transfer performance of our policies was better than that of `rnd-actions` on all tasks. This shows that intelligent sequential camera control has scope for improving these perception tasks' efficiency. Overall, our look-around policy transferred well across tasks, competing with or even outperforming the supervised task-specific policies. Furthermore, our look-around policies consistently performed the tasks better than the baseline policies for glimpse selection based on saliency or large actions.

For active recognition on ModelNet, most of the methods performed similarly. On that dataset, recognition with a single view was already fairly high, leaving limited headroom for improving with additional views, intelligently selected or otherwise. On pose estimation, our learned policies outperformed the baselines as expected, because the reconstructions generated by our agents were more accurate. On light source localization, our policies showed competitive results and came close to the performance of `supervised`. They also substantially outperformed the remaining baselines, demonstrating successful transfer. For surface area estimation, we observed that all methods, including the supervised policies, managed only marginal gains over `one-view`. We believe that this is an indication of the difficulty of this task, as well as the necessity for more 3D-specific architectures such as those that produce voxel grids, point clouds, or surface meshes as output (*38*–*40*).

These results demonstrate the effectiveness of learning active look-around policies via observation completion on unlabeled datasets—without task-specific rewards. As we see in Table 1, such policies could successfully transfer to a wide range of perception tasks and often performed on par with supervised task-specific policies.

CONCLUSION

We propose the task of active observation completion to facilitate learning look-around behaviors in a task-independent way. Our proposed approach outperformed several baselines and effectively anticipated the high-level properties of the environment, having observed only a small fraction of the scene or 3D object. We further showed that adding the proposed RL sidekicks led to faster training and convergence to better policies (Fig. 3 and fig. S3). Once look-around behaviors were learned, we showed that they could be effectively transferred to a wide range of semantic and geometric tasks where they at times outperformed supervised policies trained in a traditional task-specific manner (Table 1).

Although we are motivated to devise sidekick policy learning for active visual exploration, it is more generally applicable whenever an RL agent can access greater observability during training than during deployment. For example, agents may operate on first-person observations during test time, yet have access to multiple sensors during training in simulation environments (*41*–*43*). Similarly, an active

Table 1. Transfer results. `lookaround` and `lookaround+spl` are transferred to the `rnd-actions` task-heads from each task. The same unsupervised look-around policy successfully accelerates a variety of tasks—even competing well with the fully supervised task-specific policy (`supervised`). Note that RMSE here denotes the root mean squared error in the surface area prediction.

	SUN360			ModelNet		
		Pose estimation				
Task method	Active recogn. accuracy ↑	AE azim. ↓	AE elev. ↓	Active recogn. accuracy ↑	Light source loc. accuracy ↑	Surface area RMSE × 100↓
`one-view`	51.94	75.74	30.32	83.60	58.74	21.22
`rnd-actions`	62.90	66.18	19.53	88.46	72.97	19.04
`large-actions`	63.73	67.57	19.94	89.05	75.14	18.38
`peek-saliency`	64.20	65.46	19.76	88.74	71.19	18.85
`supervised`	68.21	51.36	9.81	88.58	**86.30**	18.43
`lookaround`	68.89	50.00	9.94	89.00	83.29	18.82
`lookaround+spl`	**69.32**	**47.13**	**9.36**	**89.38**	83.08	**18.14**

object recognition system (*8*, *10*, *11*, *26*, *30*) can only see its previously selected views of the object; yet, if trained with CAD models, it could observe all possible views while learning. Future work can explore sidekicks in such scenarios.

Despite the promising results, our approach does have several shortcomings, and our work points to several interesting directions for future work. Although the agent is moving from one view to another, it does not use the information available during this motion. This is reasonable, because allowable actions are confined to a neighborhood of the current observation and hence relatively close in 3D world space. Still, an interesting setting would be to use the sequence of views obtained while the action is being executed.

Second, our current action space was discretized to promote training efficiency, and we assumed that each action had unit cost and optimized the agent to perform well for a fixed cost budget. The unit cost was approximately correct given the locality of the action space. Nonetheless, it could be interesting to adapt to free-range actions with action-specific costs by allowing the agent to sample any action (continuous or discrete) and penalizing it based on the cost of that action. Such costs could be embodiment specific. For example, humanoid robots may find it easier to move forward when compared with turning and walking, whereas wheeled robots can perform both motions equally well. Such a formulation would also naturally account for the sequence of views seen during action execution. Furthermore, as an alternative to training the agent to make nonmyopic camera motions to best reduce reconstruction error in a fixed budget of glimpses, one could instead formulate the objective in terms of a fixed threshold on reconstruction error and allow the agent to move until that threshold is reached. The former (our formulation) is valuable for scenarios with hard resource constraints; the latter is valuable for scenarios with hard accuracy constraints.

A third limitation of the current approach is that in practice we found that the diversity of actions selected by our learned policies was sometimes limited. The agent often tended to prefer a reduced action space of two or three actions depending on the starting point and the environment, despite using a loss term explicitly encouraging high entropy of selected actions. We believe that this could be related to optimization difficulties commonly associated with policy gradient–based RL, and improvements on this front would also improve the performance of our approach.

Our approach was also affected by a well-known limitation associated with rectangular representations of spherical environments (*44*) where information at the poles are oversampled compared with the central elevations, resulting in redundant information across different azimuths at the poles. This is further exacerbated in realistic scenes where the poles often represent the sky, floor, and ceiling, which tend to have limited diversity. Because of this issue, we observed that heuristic policies that sample constant actions while avoiding the poles competed strongly with learned approaches and even outperformed supervised policies in some cases. We found that incorporating priors that encourage the agent to move away from the poles resulted in consistent performance gains for our method as well. One future direction to avoid the issue would be to design environments that have varying azimuths across elevations.

Another drawback is that our current testbeds handle only camera rotations, not translations. In future work, we will extend our approach to 3D environments that also permit camera translations (*45*, *46*). In such scenarios, intelligent look-around behavior becomes even more essential, because no matter what visual sensors it has, an agent must move its camera to observe another room. We also plan to consider other tasks for transfer such as target-driven navigation (*47*) and model-based RL (*48*, *49*), where a preliminary exploratory stage is crucial for performing well on downstream tasks.

Last, it will be interesting to explore how multiple sensing modalities could work together to learn look-around behavior. For example, an agent that hears a sudden noise from one direction might learn to look there to gain new information about dynamic objects in the scene, or an agent that sees an unfamiliar texture might reach out to touch the object surface to better anticipate its shape.

MATERIALS AND METHODS

In this final section, we summarize the implementation of our approach. Complete implementation details are provided in the Supplementary Materials.

Recurrent observation completion network

We now discuss the recurrent neural network used for active observation completion. The architecture naturally splits into five modules with distinct functions: SENSE, FUSE, AGGREGATE, DECODE, and ACT. Architecture details for all modules are given in Fig. 7.

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 $\mathbf{x}_t$ captured from a new viewpoint θ_t. We stress that absolute viewpoint coordinates θ_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 $\mathbf{p}_t$ 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 $\mathbf{o}_t$ = SENSE ($\mathbf{x}_t$, $\mathbf{p}_t$) (see Fig. 7, top left). FUSE combines information from both input streams and embeds it into $\mathbf{f}_t$ = fuse ($\mathbf{o}_t$) (Fig. 7, top center). Then, this combined sensory information $\mathbf{f}_t$ from the current observation is fed into AGGREGATE, which is a long short-term memory module (*50*). AGGREGATE maintains an encoded internal model $\mathbf{s}_t$ of the object/scene under observation to "remember" all relevant information from past observations. At each time step, it updates this code, combining it with the current observation to produce $\mathbf{s}_t$ = AGGREGATE ($\mathbf{f}_1$, $\cdots$, $\mathbf{f}_t$) (Fig. 7, top right).

SENSE, FUSE, and AGGREGATE together encode observations into an internal state $\mathbf{s}_t$ that is used to produce the output viewgrid and select the action, respectively, as we detail next.

Decoding to the inferred viewgrid

DECODE translates the aggregated code into the predicted viewgrid $\hat{V}_t(\mathbf{x}_1, \cdots, \mathbf{x}_t) = \text{DECODE}(\mathbf{s}_t)$. To do this, it first reshapes $\mathbf{s}_t$ into a sequence of small 2D feature maps (Fig. 7, 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. 7, bottom center). Seen views are pasted directly from memory.

Acting to select the next viewpoint to observe

Last, ACT processes the aggregate code $\mathbf{s}_t$ to issue a motor command $\mathbf{a}_t$ = ACT ($\mathbf{s}_t$) (Fig. 7, 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 time step to $\theta_{t+1} = \theta_t + \mathbf{a}_t$. For $t = 1$, θ_1 is randomly sampled, corresponding to the agent initially encountering the new environment or object from an arbitrary pose.

Internally, ACT first produces a distribution over all possible actions and then samples $\mathbf{a}_t$ from this distribution. We restrict ACT to select small discrete actions at each time step to approximately simulate continuous motion. Once the new viewpoint θ_{t+1} is set, a new view is captured and the whole process is repeated. This happens until T time steps have passed, involving $T - 1$ actions. These modules are learned end to end in a policy learning framework as described in the section below on the policy learning formulation.

Sidekick policy learning

We now describe the sidekicks used to learn faster and converge to better policies under partial observability. To effectively learn to perform the task, the agent has to use the limited information available from its egocentric view to (i) aggregate information, (ii) select intelligent actions to improve its training, and (iii) decode the entire viewgrid. This poses considerable hurdles for policy learning under partial observability, that is, making decisions while lacking full state knowledge. In particular, our agent does not know the entire 360° environment before it has to decide where to look next.

To address these issues, we propose sidekicks that exploit full observability available exclusively during training to aid policy learning of the ultimate agent. The key idea is to solve a simpler problem with relevance to the actual look-around task using full observability and then transfer the knowledge to the main agent. We define two types of sidekicks, reward-based and demonstration-based.

Reward-based sidekick

The reward-based sidekick aims to identify a set $\{x(X, \theta_i)\}_{i=1}^{K}$ of K highly informative views in the environment X by exploiting full observability during training. It considers a simplified completion problem where the goal is to evaluate the information content of individual views themselves, i.e., to identify information hot spots in the environment that strongly suggest other parts of the environment. For example, it might learn that facing the blank ceiling of a kitchen is less informative than looking at the contents of the refrigerator or stove.

To evaluate the informativeness of a candidate view, the sidekick sees how well

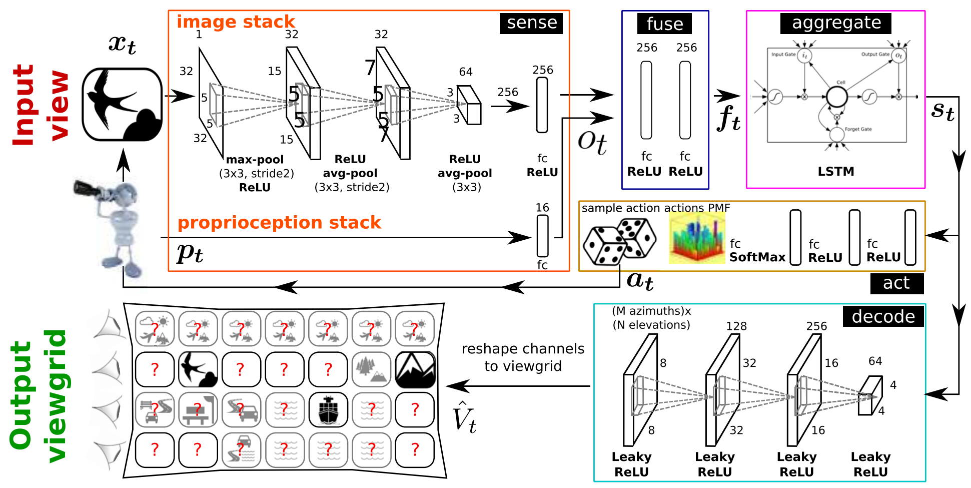


Fig. 7. Architecture of our active observation completion system. Although the input-output pair shown here is for the case of 360° scenes, we used the same architecture for the case of 3D objects. In the output viewgrid, solid black portions denote observed views, question marks denote unobserved views, and transparent black portions denote the system's uncertain contextual guesses.

the entire environment can be reconstructed given only that view. We train a completion model that can reconstruct $\hat{V}(X)$ from any single view (i.e., we set $T = 1$). The score assigned to a candidate view is inversely proportional to the reconstruction error of the entire environment given only that view. The sidekick conveys the results to the agent during policy learning in the form of an augmented reward r_t^s at each time step. Please see the section on sidekick policy learning in the Supplementary Materials for more details.

Demonstration-based sidekick

Our second sidekick generates trajectories of informative views through a sidekick policy π_s. In a trajectory, the informativeness of the current view is conditioned on the past views selected, as opposed to sampling individually informative views. To condition the informativeness on past views, we use a cumulative coverage score (see eqs. S9 and S10) that measures the amount of information gathered about different parts of the environment until time t. The goodness of a view is measured by the increase in cumulative coverage obtained upon selecting that view, i.e., how well it complements the previously selected views. Please see the section on sidekick policy learning in the Supplementary Materials for full details.

The demonstration sidekick uses this coverage score to sample informative trajectories. Given a starting view in X, the demonstration sidekick selects a trajectory of T views that jointly maximize the coverage of X. At each time step, the demonstration sidekick evaluates the gain in cumulative coverage obtained by sampling each view in its neighborhood and then greedily samples the best view (see eq. S11).

We use sidekick-generated trajectories as supervision to the agent for a short preparatory period. The goal is to initialize the agent with useful insights learned by the sidekick to accelerate training of better policies. We achieve this through a hybrid training procedure that combines imitation and reinforcement, as described in the demonstration-based sidekick section in the Supplementary Materials.

Policy learning formulation

Having defined the recurrent network model and the sidekick policy preparation, we now describe the policy learning framework used to train our agent as well as the mechanisms used to incorporate sidekick rewards (r_t^s) and demonstrations (obtained from π_s). All modules are jointly optimized end to end to improve the final reconstructed viewgrid $\hat{V}_T$, which contains predicted views $\hat{\mathbf{x}}_\mathbf{T}(X, \theta_j)$ for all viewpoints θ_j, $1 \le j \le MN$. The agent learns a policy $\pi(a \mid s_t)$ that returns a distribution over actions for the aggregated internal representation s_t at time t. Let $\mathcal{A} = \{a_i\}$ denote the set of camera motions available to the agent. Our agent seeks the policy that minimizes reconstruction error for the environment given a budget of T camera motions (views). Let W_s, W_f, W_r, W_d, W_a represent the weights of the SENSE, FUSE, AGGREGATE, DECODE, and ACT modules. If we denote the set of weights of the network $[W_s, W_f, W_r, W_d, W_a]$ by W and W excluding W_a by $W_{/a}$ and W excluding W_d by $W_{/d}$, then the overall weight update is

$$\Delta W = \frac{1}{n}\sum_{j=1}^{n}\lambda_r \Delta W_{/a}^{rec} + \lambda_a \Delta W_{/d}^{\text{act}} \tag{1}$$

where n is the number of training samples, j indexes over the training samples, λ_r and λ_a are constants, and $\Delta W_{/a}^{\text{rec}}$ and $\Delta W_{/d}^{\text{act}}$ update all parameters except W_a and W_d, respectively.

The pixelwise MSE reconstruction loss ($\mathcal{L}_t^{\text{rec}}$) and corresponding weight update at time t are as follows

$$\begin{aligned} \mathcal{L}_{\text{rec}}^t(X) &= \sum_{i=1}^{MN} d(\hat{x}_t(X, \theta^{(i)} + \Delta_0), x(X, \theta^{(i)})), \\ \Delta W_{/a}^{\text{rec}} &= -\sum_{t=1}^{T} \nabla_{W_{/a}} \mathcal{L}_{\text{rec}}^t(X) \end{aligned} \tag{2}$$

where $\hat{x}_t(X, \theta^{(i)})$ denotes the reconstructed view at viewpoint $\theta^{(i)}$ and time t, d denotes the pixelwise reconstruction MSE, and Δ_0 denotes the offset to account for the unknown starting azimuth (*23*).

The agent's reward at time t consists of the intrinsic reward from the sidekick $r_t^s = \text{Info}(x(X, \theta_t), X)$ and the negated final reconstruction loss, $-\mathcal{L}_{\text{rec}}^T(X)$

$$r_t = \begin{pmatrix} r_t^s & 1 \le t \le T-2 \\ -\mathcal{L}_{\text{rec}}^T(X) + r_t^s & t = T-1. \end{pmatrix} \tag{3}$$

The sidekick reward r_t^s serves to densify the rewards by exploiting full observability, thereby reducing uncertainty during policy learning. Please see the Supplementary Materials for the exact form of r_t^s. The update from the policy consists of an actor-critic update, with a baseline b to reduce variance, and supervision from the demonstration sidekick:

$$\Delta W_{/d}^{\text{act}} = \sum_{t=1}^{T-1} \nabla_{W_{/d}} \log \pi(a_t \mid s_t) \left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t) \right) + \Delta W_{/d}^{\text{demo}}. \tag{4}$$

We adapt the baseline b as the value function from an actor-critic (*51*) method to update the ACT module. The demonstration sidekick's supervision is defined below in Eq. 5. The ACT term additionally includes a loss to update the learned value network and entropy regularization to promote diversity in action selection (please see additional loss functions in the Supplementary Materials).

Whereas the reward sidekick augments rewards, the demonstration sidekick instead influences policy learning by directly supervising the early rounds of action selection. This is achieved through a cross-entropy loss between the sidekick's policy π_s and the agent's policy π:

$$\Delta W_{/d}^{\text{demo}} = \sum_{t=1}^{T-1} \sum_{a \in \mathcal{A}} \nabla_{/d}(\pi_s(a \mid s_t) \log \pi(a \mid s_t)). - 0.10cm \tag{5}$$

Please see the sidekick policy learning section in the Supplementary Materials for the exact form of π_s.

We pretrain the SENSE, FUSE, and DECODE modules with $T = 1$. The full network is then trained end to end (with SENSE and FUSE frozen). For training with sidekicks, the agent is augmented either with additional rewards from the reward sidekick (Eq. 3) or an additional supervised loss from the demonstration sidekick (Eq. 5).

Unsupervised policy transfer to unseen tasks

We now describe the mechanism used to transfer policies learned in an unsupervised fashion via active observation completion to new perception tasks requiring sequential observations. This section details the process overviewed above in the look-around policy transfer section. 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 with neither manual supervision nor task-specific reward—into various semantic and geometric recognition tasks for which the agent was not specifically trained. Recall, we considered four different tasks: recognition, surface area estimation, light source localization, and camera pose estimation.

At training time, we train an end-to-end task-specific model (model A) with a random policy (`rnd-actions`) and an active observation completion model (model B). Note that our completion model is trained without supervision to look around environments that have zero overlap with model A's test set. Furthermore, even the categories seen during training may differ from those during testing. For example, the agent might see various furniture categories during training (bookcase, bed, etc.) but never a chair, yet it must generalize well to look around a chair.

At test time, both the task-specific model A and the active observation model B receive and process the same inputs at each time step. The task-specific model does not have a learned policy of its own, because it is trained with a policy that samples random actions. At each time step, model B selects actions to complete its internal model of the new environment based on its look-around policy. This action is then communicated to model A in place of the random actions with which it was trained. Therefore, model A gathers its information based on the actions provided by model B. Model A then makes a prediction for the target task. If the policy learned in model B is truly generic, then it will intelligently explore to solve the new (unseen) tasks despite never receiving task-specific reward for any one of them during training.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/4/30/eaaw6326/DC1
Text to augment the implementation details in Materials and Methods
Fig. S1. Sidekick framework.
Fig. S2. Light source localization example.
Fig. S3. Convergence of sidekick policy learning.
Fig. S4. Training on different target budgets *T*.
Fig. S5. Episodes of active observation completion.
Fig. S6. GAN refinement.
Movie S1. Sample walkthroughs in reconstructed environments.
Movie S2. Active observation completion on SUN360.
Movie S3. Active observation completion on ModelNet.
References (*53*–*56*)

REFERENCES AND NOTES

1. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge. *Int. J. Comp. Vis.* **115**, 211–252 (2015).
2. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common objects in context. *Eur. Conf. Comp. Vis.* **8693**, 740–755 (2014).
3. K. Soomro, A. R. Zamir, M. Shah, Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402 [cs.CV] (3 December 2012).
4. K. C. Soska, S. P. Johnson, Development of three-dimensional object completion in infancy. *Child Dev.* **79**, 1230–1236 (2008).
5. K. C. Soska, K. E. Adolph, S. P. Johnson, Systems in development: Motor skill acquisition facilitates three-dimensional object completion. *Dev. Psychol.* **46**, 129–138 (2010).
6. P. J. Kellman, E. S. Spelke, Perception of partly occluded objects in infancy. *Cogn. Psychol.* **15**, 483–524 (1983).
7. A. Torralba, A. Oliva, M. S. Castelhano, J. M. Henderson, Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychol. Rev.* **113**, 766–786 (2006).
8. D. Jayaraman, K. Grauman, Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion, in *European Conference on Computer Vision* (Springer, 2016).
9. M. Malmir, K. Sikka, D. Forster, J. R. Movellan, G. Cottrell, Deep Q-learning for active recognition of GERMS: Baseline performance on a standardized dataset for active learning. *British Machine Vision Conference* (BMVA, 2015).
10. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D ShapeNets: A deep representation for volumetric shapes, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2015).
11. P. Ammirato, P. Poirson, E. Park, J. Košecká, A. C. Berg, A dataset for developing and benchmarking active vision, in *IEEE International Conference on Robotics and Automation* (IEEE, 2017).
12. S. Yeung, O. Russakovsky, G. Mori, L. Fei-Fei, End-to-end learning of action detection from frame glimpses in videos, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016).
13. S. Mathe, A. Pirinen, C. Sminchisescu, Reinforcement learning for visual object detection, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016).
14. S. Karayev, T. Baumgartner, M. Fritz, T. Darrell, Timely object recognition, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2012).
15. D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in *International Conference on Machine Learning* (PMLR, 2017).
16. T. Chen, S. Gupta, A. Gupta, Learning exploration policies for navigation, in *International Conference on Learning Representations* (2019).
17. B. Hepp, D. Dey, S. N. Sinha, A. Kapoor, N. Joshi, O. Hilliges, Learn-to-score: Efficient 3D scene exploration by predicting view utility, in *European Conference on Computer Vision* (Springer, 2018).
18. S. Song, A. Zeng, A. X. Chang, M. Savva, S. Savarese, T. Funkhouser, Im2pano3D: Extrapolating 360° structure and semantics beyond the field of view, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018).
19. D. Ji, J. Kwon, M. McFarland, S. Savarese, Deep view morphing, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017).
20. T. D. Kulkarni, W. Whitney, P. Kohli, J. B. Tenenbaum, Deep convolutional inverse graphics network, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2015).
21. D. Jayaraman, R. Gao, K. Grauman, ShapeCodes: Self-supervised feature learning by lifting views to viewgrids, in *European Conference on Computer Vision* (Springer, 2018).
22. S. M. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, D. Hassabis, Neural scene representation and rendering. *Science* **360**, 1204–1210 (2018).
23. D. Jayaraman, K. Grauman, Learning to look around: Intelligently exploring unseen environments for unknown tasks, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018).
24. S. K. Ramakrishnan, K. Grauman, Sidekick policy learning for active visual exploration, in *European Conference on Computer Vision* (Springer, 2018).
25. For simplicity of presentation, we represent an "environment" as *X* where the agent explores a novel scene, looking outward in new viewing directions. However, experiments will also use *X* as an object where the agent moves around an object, looking inward at it from new viewing angles. Figure 1 illustrates the two scenarios.
26. E. Johns, S. Leutenegger, A. J. Davison, Pairwise decomposition of image sequences for active multi-view recognition, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016).
27. Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, A. Farhadi, Visual semantic planning using deep successor representations, in *IEEE International Conference on Computer Vision* (IEEE, 2017).
28. S. Gupta, D. Fouhey, S. Levine, J. Malik, Unifying map and landmark based representations for visual navigation. arXiv:1712.08125 [cs.CV] (21 December 2017).
29. Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in *IEEE International Conference on Robotics and Automation* (IEEE, 2017).
30. D. Jayaraman, K. Grauman, End-to-end policy learning for active visual categorization. *IEEE Trans. Pattern Anal. Mach. Intell.* (2018).
31. X. Guo, S. Singh, H. Lee, R. Lewis, X. Wang, Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2014).
32. V. Vapnik, R. Izmailov, Learning with intelligent teacher, in *Symposium on Conformal and Probabilistic Prediction with Applications* (Springer, 2016).

33. J. Xiao, K. A. Ehinger, A. Oliva, A. Torralba, Recognizing scene viewpoint using panoramic place representation, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2012).
34. The angles were selected to break symmetry and reduce redundancy of views.
35. For the sake of brevity, we report the best performances among the two sidekick variants we proposed in (*24*).
36. J. Harel, C. Koch, P. Perona, Graph-based visual saliency, in *Advances in Neural Information Processing Systems* (MIT Press, 2006).
37. We refine the decoded viewgrids (for both our method and the baseline) with a pix2pix (*52*)–style conditional generative adversarial network (GAN), detailed in the Supplementary Materials.
38. C. B. Choy, D. Xu, J. Gwak, K. Chen, S. Savarese, 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction, in *Proceedings of the European Conference on Computer Vision (ECCV)* (Springer, 2016).
39. H. Fan, H. Su, L. Guibas, A point set generation network for 3D object reconstruction from a single image, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017).
40. N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, Y.-G. Jiang, Pixel2mesh: Generating 3D mesh models from single RGB images. arXiv:1804.01654 [cs.CV] (5 April 2018).
41. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in *Conference on Robot Learning* (PMLR, 2017).
42. L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, P. Abbeel, Asymmetric actor critic for image-based robot learning, in *Robotics: Science and Systems* (Robotics Proceedings, 2018).
43. A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, D. Batra, Embodied question answering, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018).
44. B. Coors, A. P. Condurache, A. Geiger, SphereNet: Learning spherical representations for detection and classification in omnidirectional images. *Proc. Eur. Conf. Comput. Vis.* **11213**, 525–541 (2018).
45. Y. Wu, Y. Wu, G. Gkioxari, Y. Tian, Building generalizable agents with a realistic and rich 3D environment. arXiv:1801.02209 [cs.LG] (7 January 2018).
46. P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, A. van den Hengel, Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018).
47. N. Savinov, A. Dosovitskiy, V. Koltun, Semi-parametric topological memory for navigation, in *International Conference on Learning Representations* (2018).
48. D. Ha, J. Schmidhuber, World models. arXiv:1803.10122 [cs.LG] (27 March 2018).
49. A. J. Piergiovanni, A. Wu, M. S. Ryoo, Learning real-world robot policies by dreaming. arXiv:1805.07813 [cs.RO] (20 May 2018).
50. S. Hochreiter, J. Schmidhuber, Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
51. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 2014).
52. P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017).
53. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars. arXiv:1604.07316 [cs.CV] (25 April 2016).
54. A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, L. M. Gambardella, A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot. Autom. Lett.* **1**, 661–667 (2016).
55. Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, W. Zaremba, One-shot imitation learning, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2017).
56. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2014).

Acknowledgments: We thank Y.-C. Su, K. Hsiao, B. Xiong, and P. Krähenbühl for helpful discussions. **Funding:** The University of Texas at Austin was supported, in part, by DARPA Lifelong Learning Machines (L2M), an AFOSR DURIP equipment grant, an IBM OCR Award, and a Sony Research Award. This material is based on research sponsored by the Air Force Research Laboratory and DARPA under agreement number FA8750-18-2-0126. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. **Author contributions:** The project ideas were conceived by D.J., S.K.R., and K.G. The experiments were designed by D.J., S.K.R., and K.G. The experiments were performed by D.J. and S.K.R. The paper was written by D.J., S.K.R., and K.G. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** The datasets for this study are available at http://vision.cs.utexas.edu/projects/sidekicks/scirobo-2019-data.zip, and the source code is available at https://github.com/srama2512/sidekicks.

Submitted 15 January 2019
Accepted 19 April 2019
Published 15 May 2019
10.1126/scirobotics.aaw6326

Citation: S. K. Ramakrishnan, D. Jayaraman, K. Grauman, Emergence of exploratory look-around behaviors through active observation completion. *Sci. Robot.* **4**, eaaw6326 (2019).

Science Robotics

Emergence of exploratory look-around behaviors through active observation completion

Santhosh K. Ramakrishnan, Dinesh Jayaraman and Kristen Grauman

Sci. Robotics **4**, eaaw6326.
DOI: 10.1126/scirobotics.aaw6326