Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion

Dinesh Jayaraman, *UT Austin,* and Kristen Grauman, *UT Austin*

***Abstract*—Visual recognition systems mounted on autonomous moving agents face the challenge of unconstrained data, but simultaneously have the opportunity to improve their performance by moving to acquire new views of test data. In this work, we first show how a recurrent neural network-based system may be trained to perform end-to-end learning of motion policies suited for the "active recognition" setting. Further, we hypothesize that active vision requires an agent to have the capacity to reason about the effects of its motions on its view of the world. To verify this hypothesis, we attempt to induce this capacity in our active recognition pipeline, by simultaneously learning to forecast the effects of the agent's motions on its internal representation of its cumulative knowledge obtained from all past views. Results across two challenging datasets confirm both that our end-to-end system successfully learns meaningful policies for active recognition, and that "learning to look ahead" further boosts recognition performance.**[1]

I. Introduction

People consistently direct their senses in order to improve their chances of understanding their surroundings. For example, you might swivel around in your armchair to observe a person behind you, pick up and rotate a coffee mug on your desk to read an inscription on it, or walk to the window to better observe whether it is raining.

In sharp contrast to such scenarios, recent recognition research has been focused almost exclusively on static image recognition: the system takes a single snapshot as input, and produces a category label estimate as output. The ease of collecting large labeled datasets of images has enabled major advances on this task in recent years, as evident for example in the striking gains made by our community on the ImageNet challenge [1]. Yet, despite this recent progress, recognition performance remains low for more complex, unconstrained images [2].

Recognition systems mounted on autonomous moving agents acquire unconstrained visual input which may be difficult to recognize effectively for any *one image at a time*. However, similar to the human actor in the opening examples above, such systems have the opportunity to improve their performance by moving their camera apparatus or manipulating objects to acquire new information. In other words, a system's control over its own sensory input has tremendous potential to influence visual recognition. While such mobile agent settings (mobile robots, autonomous vehicles, etc.) are closer to reality today than ever before, the problem of *learning to actively move* to direct the acquisition of data remains underexplored in modern visual recognition research.

[1]A preliminary version of the material in this document was filed as University of Texas technical report no. UT AI15-06, December, 2015.

The problem we are describing fits into the realm of *active vision*, which has a rich history in the literature (e.g., [3], [4], [5], [6], [7], [8]). Active vision offers several technical challenges that are unaddressed in today's standard passive scenario. In order to perform active vision, a system must learn to intelligently direct the acquisition of input to be processed by its recognition pipeline. In addition, recognition in an active setting places different demands on a system than in the standard passive scenario. For example, a major difference is that "nuisance factors" in still image recognition—such as pose, lighting, and viewpoint changes—become *avoidable* factors in the active vision setting. In principle, such factors can often be overcome merely by moving to the right location.

This calls for a major change of approach. Rather than strive for invariance to nuisance factors as is the standard in static image recognition, an intriguing strategy is to learn to *identify when conditions are non-ideal for recognition* and to *actively select the correct agent motion* that will lead to better conditions. In addition, recognition decisions must be made based on intelligently fusing observations over time.

We contend that these three functions of an active vision system—control, per-view recognition, and evidence fusion over time—are closely intertwined, and must be tailored to work together. In particular, as the first contribution of this paper, we propose to learn all three modules of an active vision system simultaneously and end-to-end. We employ a stochastic neural network to learn intelligent motion policies (control), a standard neural network to process inputs at each time step (per-view recognition), and a modern recurrent neural network (RNN) to integrate evidence over time (evidence fusion). Given an initial view and a set of possible agent motions, our approach learns how to move in the 3-D environment to produce accurate categorization results.

Additionally, we hypothesize that motion planning for active vision requires an agent to internally "look before it leaps". That is, it ought to simultaneously reason about the effect of its motions on future inputs. To demonstrate this, as a second contribution, we jointly train our active vision system to have the ability to predict *how its knowledge will evolve* conditioned on its present state and its choice of motion. As we will explain below, this may be seen as preferring equivariance *i.e.* predictable feature responses to pose changes, rather than invariance as is standard in passive recognition pipelines.

Through experiments on two datasets, we empirically validate both our key ideas: (1) RNN-based end-to-end active categorization and (2) learning to forecast the effects of self-motion at the same time one learns how to move to solve the recognition task. We study both a scene categorization

scenario, where the system chooses how to move around a previously unseen 3-D scene, and an object categorization scenario, where the system chooses how to manipulate a previously unseen object that it holds. Our results comparing to passive and simpler active view selection methods show the high potential for such an approach.

II. Related work

a) Active vision: The idea that a subject's actions may play an important role in perception can be traced back almost 150 years [9] in the cognitive psychology literature [3]. "Active perception", the idea of exploiting *intelligent control strategies* (agent motion, object manipulation, camera parameter changes *etc*.) for *goal-directed data acquisition* to improve machine vision, was pioneered by [10], [8], [11], [4]. While most research in this area has targeted low-level vision problems such as segmentation, structure from motion, depth estimation, optical flow estimation [12], [11], [8], or known object localization ("semantic search") [13], [14], [15], [16], approaches targeting *active recognition* are most directly related to our work.

Most prior active recognition approaches attempt to identify during training those canonical/"special" views that minimize ambiguity among candidate labels [4], [5], [6], [7]. At test time, such systems iteratively estimate the current pose, then select the move that would take them to the pre-identified disambiguating view that would confirm their current beliefs. Such approaches are typically applicable only to *instance* recognition problems, since broader categories can be too diverse in appearance and shape to fix "special viewpoints".

In contrast, our approach handles complex real-world categories. To the best of our knowledge, very little prior work attempts the challenging task of active *category* recognition (as opposed to instance recognition) [17], [18], [19], [20], [21]. To actively categorize small toy categories [17], a naive Bayes model fuses information across time steps with observations treated as independent. An information-gain method proposed in [21] is used to select new views. The increased difficulty is due to the fact that with complex real world categories, it is much harder to anticipate new views conditioned on actions. Since new instances will be seen at test time, it is not sufficient to simply memorize the geometry of individual instances, as many active instance recognition methods effectively do.

Recently, [18], [19] use information gain in view planning for active categorization. Both methods learn to *predict* the next views of *unseen test objects* conditioned on various candidate agent motions starting from the current view, either by estimating 3D models from 2.5D RGBD images [18] or by learning to predict feature responses to camera motions [19]. They then estimate the information gain on their category beliefs from each such motion, and finally greedily select the estimated most informative "next-best" move. While our idea for learning to predict action-conditional future views of novel instances is similarly motivated, we refrain from explicit greedy reasoning about the next move. Instead, our approach uses reinforcement learning (RL) in a stochastic recurrent neural network to learn optimal *sequential* movement policies over multiple time-steps. The closest methods to ours in this respect are [22] and [23], both of which employ Q-learning in feedforward neural networks to perform view selection, and target relatively simpler visual tasks compared to this work.

In addition to the above, an important novelty of our approach is in learning the entire system end-to-end. Active recognition approaches must broadly perform three separate functions: action selection, per-instant view processing, and belief updates based on the history of observed views. While previous approaches have explored several choices for action selection, they typically train a "passive" per-instant view recognition module offline and fuse predictions across time using some manually defined heuristic [6], [5], [17], [23]. For example, recently a deep neural network was trained to learn action policies in [23] after pretraining a per-view classifier and using a simple Naive Bayes update heuristic for label belief fusion. In contrast, we train all three modules jointly within a single active recognition objective. This is important because, passive recognition systems may have demands such as invariance that are not suited to active recognition and would therefore "distract" the learned features.

b) Saliency and attention: Visual saliency and attention are related to active vision [24], [25], [26], [27], [28]. While active vision systems aim to form policies to acquire new data, saliency and attention systems aim to block out "distractors" in *existing* data by identifying portions of input images/video to focus on, often as a faster alternative to sliding window-based methods. Attention systems thus sometimes take a "foveated" approach [29], [24]. In contrast, in our setting, the system never holds a snapshot of the entire environment at once. Rather, its input at each time step is one portion of its complete physical 3D environment, and it must choose motions leading to more informative—possibly non-overlapping—viewpoints. Another difference between the two settings is that the focus of attention may move in arbitrary jumps (saccades) without continuity, whereas active vision agents may only move continuously.

Sequential attention systems using recurrent neural networks in particular have seen significant interest of late [24], with variants proving successful across several attention-based tasks [25], [26], [28]. We adopt the basic attention architecture of [24] as a starting point for our model, and develop it further to accommodate the active vision setting, instill lookahead capabilities, and select camera motions surrounding a 3D object that will most facilitate categorization.

c) Predicting related features: There is recent interest in "visual prediction" problems in various contexts, often using convolutional neural networks (CNNs) [30], [31], [32], [33], [18], [34], [19], [35]. For example, one can train CNNs to predict the features of the next video frame conditioned on the current one [30], or predict dense optical flow from a single image [35]. Recurrent networks can also learn to extrapolate videos based on previously observed frames [31]. These approaches do not attempt to reason about *causes* of view transformations *e.g*. camera motions.

Close to our work are methods for "next view" prediction, where the system generates the view change itself. In [32], [33], approaches are developed to allow the manipulation of the "factors of variation" (such as pose and lighting) of simple

synthetic images. High quality unseen views are predicted given surrounding views accompanied by exact camera poses, effectively learning 3D geometry end-to-end in a CNN [34]. The methods of [19], [36] learn to predict feature responses to a defined set of discrete observer motions. Different from all the above, we learn to predict the evolution of temporally aggregated features—computed from a complete history of seen views—as a function of observer motion choices. Furthermore, we integrate this idea with the closely tied active recognition problem.

d) Integrating sensors and actions: Our work is also related to research in sensorimotor feature embeddings. There the idea is to combine (possibly non-visual) sensor streams together with proprioception or other knowledge about the actions of the agent on which the sensors are mounted. Various methods learn features that transform in simple ways in response to an agent's actions [37], [19], [38] or reflect the geometry of an agent's environment [39]. In [40], a neural network learns to drive a simulated car in a video game. In [41], end-to-end reinforcement learning trains robotic agents to perform simple tasks. Perhaps conceptually most relevant among these is [42]. Their method learns an image feature space to determine control actions easily from visual inputs, with applications to simulated control tasks. In contrast, we learn embeddings encoding knowledge of complete *histories* of observations and agent actions, with the aim of exposing this knowledge to an active visual recognition controller.

III. Approach

First we define the setting and data flow for active recognition (Sec. III-A). Then we define our system architecture (Sec. III-B). Finally, we describe our look-ahead module (Sec. III-C).

A. Setting

We first describe our active vision setting at test time, using a 3-D object category recognition scenario as a running example. Our results consider both object and scene category recognition tasks. The active recognition system can issue motor commands to move a camera within a viewing sphere around the 3-D object X of interest. Each point on this viewing sphere is indexed by a corresponding 2-D camera pose vector $\boldsymbol{p}$ indexing elevation and azimuth.

The system is allowed T time steps to recognize every object instance X. At every time step $t = 1, 2, \ldots T$:

- The system issues a motor command $\boldsymbol{m}_t$ e.g. "increase camera elevation by 20 degrees, azimuth by 10 degrees", from a set $\mathcal{M}$ of available camera motions. In our experiments, $\mathcal{M}$ is a discrete set consisting of small camera motions to points on an elevation-azimuth grid centered at the previous camera pose $\boldsymbol{p}_{t-1}$. At time instance $t = 1$, the "previous" camera pose p_0 is set to some random unknown vector, corresponding to the agent initializing its recognition episode at some arbitrary position with respect to the object.
- Next, the system is presented a new 2-D view $\boldsymbol{x}_t = P(X, \boldsymbol{p}_t)$ of X captured from the new camera pose $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$, where $P(.,.)$ is a projection function. This new evidence is now available to the system while selecting its next action $\boldsymbol{m}_{t+1}$.

Finally, at the final time-step $t = T$, the system must additionally predict a category label $\hat{y}$ for X, e.g., the object category it believes is most probable. In our implementation, the number of time-steps T is fixed, and all valid motor commands are assumed to have uniform cost. The system is evaluated only on the accuracy of its prediction $\hat{y}$. However, the framework generalizes to the case of variable-length episodes.

B. Active recognition system architecture

Our basic active recognition system is modeled on the recurrent architecture first proposed in [24] for visual attention. Our system is composed of four basic modules: ACTOR, SENSOR, AGGREGATOR and CLASSIFIER, with weights W_a, W_s, W_r, W_c respectively. At each step t, ACTOR issues a motor command $\boldsymbol{m}_t$, which updates the camera pose vector to $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$. Next, a 2-D image $\boldsymbol{x}_t$ captured from this pose is fed into SENSOR together with the motor command $\boldsymbol{m}_t$. SENSOR produces a view-specific feature vector $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$, which is then fed into AGGREGATOR to produce aggregate feature vector $\boldsymbol{a}_t = \text{AGGREGATOR}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t)$. The cycle is completed when, at the next step $t + 1$, ACTOR processes the aggregate feature from the previous time step to issue $\boldsymbol{m}_{t+1} = \text{ACTOR}(\boldsymbol{a}_t)$. Finally, after T steps, the category label beliefs are predicted as $\hat{y}(W, X) = \text{CLASSIFIER}(\boldsymbol{a}_t)$, where $W = [W_a, W_s, W_r, W_c]$ is the vector of all learnable weights in the network, and for a C-class classification problem, $\hat{y}$ is a C-dimensional multinomial probability density function representing the likelihoods of the 3-D object X belonging to each of the C classes. See Fig 1 for a schematic showing how the modules are connected.

In our setup, AGGREGATOR is a recurrent neural network, CLASSIFIER is a simple fully-connected hidden layer followed by a log-softmax and SENSOR separately processes the view $\boldsymbol{x}_t$ and the motor signal $\boldsymbol{m}_t$ in disjoint neural network pipelines before merging them through more layers of processing to produce the per-instance view feature $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$. ACTOR has a non-standard neural net architecture involving stochastic units: at each time step, it internally produces an $|\mathcal{M}|$-dimensional multinomial density function $\pi(\boldsymbol{m}_t)$ over all candidate camera motions in $\mathcal{M}$, from which it samples one motion. For more details on the internal architecture of these modules, see Supp.

e) Training: At training time, the network weights W are trained jointly to maximize classifier accuracy at time T. Following [24], training W follows a hybrid procedure involving both standard backpropagation and "connectionist reinforcement learning" [43]. The modules with standard deterministic neural network connections (CLASSIFIER, AGGREGATOR and SENSOR) with joint weight vector denoted by $W_{\setminus a}$ can be trained directly by backpropagating gradients from a softmax classification loss, while the ACTOR module which contains stochastic units can only be trained using the REINFORCE procedure of [43].

Roughly, REINFORCE treats the ACTOR module containing stochastic units as a Partially Observable Markov Decision

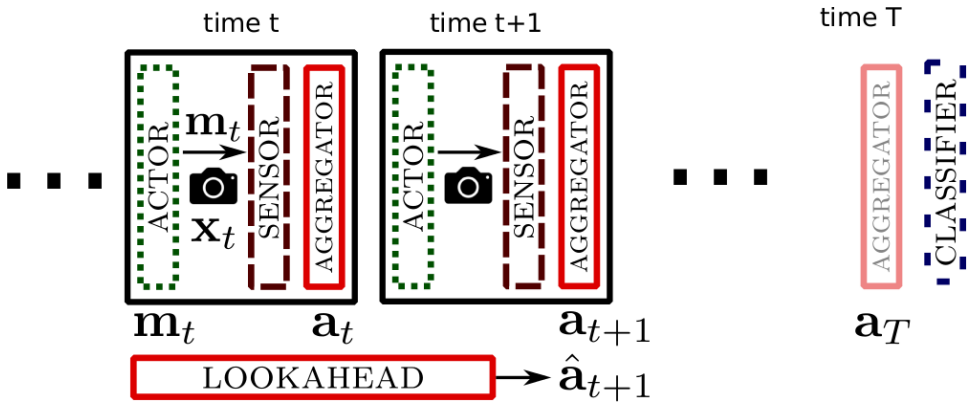


Fig. 1: A schematic of our system architecture depicting the interaction between ACTOR, SENSOR and AGGREGATOR and CLASSIFIER modules, unrolled over time-steps. Information flows from left to right. At training time, the additional LOOKAHEAD acts across two time-steps, learning to predict the evolution of the aggregate feature $\boldsymbol{a}_t$ into $\boldsymbol{a}_{t+1}$ conditional on the selected motion $\boldsymbol{m}_t$. See Sec III-B for details.

Process (POMDP), with the pdf $\pi(\boldsymbol{m}_t|\boldsymbol{a}_{t-1}, W)$ representing the policy to be learned. In a reinforcement learning (RL)-style approach, REINFORCE iteratively increases weights in the pdf $\pi(\boldsymbol{m})$ on those candidate motions $\boldsymbol{m} \in \mathcal{M}$ that have produced higher "rewards", as defined by a reward function. A simple REINFORCE reward function to promote classification accuracy could be $R_c(\hat{y}) = 1$ when the most likely label in $\hat{y}$ is correct, and 0 when not. To speed up training, we use a variance-reduced version of this loss $R(\hat{y}) = R_c(\hat{y}) - b$, where b is set to the reward for a simple baseline predictor that learns to predict the constant value that yields the most reward *i.e.* the most frequent class in training data. Beyond the stochastic units, the REINFORCE algorithm produces gradients that may be propagated to non-stochastic units through standard backpropagation. In our hybrid training approach, these REINFORCE gradients from ACTOR are therefore added to the softmax loss gradients from CLASSIFIER before back-propagation through AGGREGATOR and SENSOR.

More formally, given a training dataset of instance-label pairs $\{(X^i, y^i) : 1 \leq i \leq N\}$, the gradient updates are as follows. Let $W_{\setminus c}$ denote $[W_a, W_s, W_r]$, *i.e.* all the weights in W except the CLASSIFIER weights W_c, and similarly, let $W_{\setminus a}$ denote $[W_c, W_r, W_s]$. Then:

$$\Delta W_{\setminus c}^{RL} \approx \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{W_{\setminus c}} \log \pi(m_t^i | \boldsymbol{a}_{t-1}^i; W_{\setminus c}) R^i, \quad (1)$$

$$\Delta W_{\setminus a}^{SM} = -\sum_{i=1}^{N} \nabla_{W_{\setminus a}} L_{\text{softmax}}(\hat{y}^i(W, X), y^i), \quad (2)$$

where indices i in the superscripts denote correspondence to the i^{th} training sample X^i. Eq (1) and (2) show the gradients computed from the REINFORCE rewards (RL) and the softmax loss (SM) respectively, for different subsets of weights. The REINFORCE gradients ΔW^{RL} are computed using the approximation proposed in [43]. Final gradients with respect to the weights of each module used in weight updates are given by: $\Delta W_a = \Delta W_a^{RL}$, $\Delta W_s = \Delta W_s^{RL} + \Delta W_s^{SM}$, $\Delta W_r = \Delta W_r^{RL} + \Delta W_r^{SM}$, $\Delta W_c = \Delta W_c^{RL} + \Delta W_c^{SM}$. Training is through standard stochastic gradient descent with early stopping based on a validation set.

One important difference between our system and prior active view selection approaches is that we attempt to avoid manually defined heuristics to guide the camera motion. Instead, our method learns motion policies from scratch, based on the aggregated knowledge from the full history of past views, as contained in $\boldsymbol{a}_t$. With sufficient training data, better ultimate policies may result.

C. Look-ahead: predicting the effects of motions

Active recognition systems select the next motion based on some expectation of the next view. Though non-trivial even in the traditional instance recognition setting [4], [5], [6], [7], with instances one can exploit the fact that pose estimation in some canonical pose space is sufficient in itself to estimate properties of future views. In other words, with enough prior experience seeing the object instance, it is largely a 3-D (or implicit 3-D) geometric model formation problem.

In contrast, as discussed in Sec. II, this problem is much harder in active *categorization* with realistic categories—the domain we tackle. Predicting subsequent views in this setting is severely under-constrained, and requires reasoning about semantics and geometry together. In other words, next view planning requires some element of learning about how 3-D objects *in general* change in their appearance as a function of observer motion.

We hypothesize that the ability to predict the next view conditional on the next camera motion is closely tied to the ability to select optimal motions. Thus, rather than learn separately the model of view transitions and model of motion policies, we propose a unified approach to learn them jointly. Our idea is that knowledge transfer from a view prediction task will prove beneficial for active categorization. In this formulation, we retain the system from Sec. III-B, but simultaneously learn to predict, at every time step t, the impact on aggregate features $\boldsymbol{a}_{t+1}$ at the next time step, given $\boldsymbol{a}_t$ and any choice of motion $\boldsymbol{m}_t \in \mathcal{M}$. In other words, we simultaneously learn how the

accumulated history of learned features—not only the current view—will evolve as a function of our candidate motions.

To perform this auxiliary task, we introduce an additional module, LOOKAHEAD, with weights W_l into the setup of Sec. III-B at training time. At time step t, LOOKAHEAD takes as input $\boldsymbol{a}_{t-1}$ and $\boldsymbol{m}_{t-1}$ and predicts $\hat{\boldsymbol{a}}_t = \text{LOOKAHEAD}(\boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1})$. This module may be thought of as a "predictive auto-encoder" in the space of aggregate features $\boldsymbol{a}_t$ output by AGGREGATOR. A look-ahead error loss is computed at every time-step between the predicted and actual aggregate features: $d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t | \boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1})$. We use the cosine distance to compute this error. This per-time-step look-ahead loss provides a third source of training gradients for the network weights, as it is backpropagated through AGGREGATOR and SENSOR:

$$\Delta W_{\setminus ca}^{LA} = \sum_{i=1}^{N} \sum_{t=2}^{T} \nabla_{W_{\setminus ca}} d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t | \boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1}), \quad (3)$$

where W now includes W_l. The LOOKAHEAD module itself is trained solely from this error, so that $\Delta W_l = \Delta W_l^{LA}$. The final gradients used to train SENSOR and AGGREGATOR change to include this new loss: $\Delta W_s = \Delta W_s^{RL} + \Delta W_s^{SM} + \lambda \Delta W_s^{LA}$, $\Delta W_r = \Delta W_r^{RL} + \Delta W_r^{SM} + \lambda \Delta W_r^{LA}$. λ is a new hyperparameter that controls how much the weights in the core network are influenced by the look-ahead error loss.

The look-ahead error loss of Eq 3 may also be interpreted as an unsupervised regularizer on the classification objective of Eq 1 and 2. This regularizer encodes the hypothesis that good features for the active recognition task must respond in learnable, systematic ways to camera motions.

This is related to the role of "equivariant" image features in [19], where it was shown that regularizing image features to respond predictably to observer egomotions improves performance on a standard static image categorization tasks. We differ from [19] in several important ways. First, we explore the utility of lookahead for the active categorization problem, not recognition of individual static images. Second, the proposed lookahead module is conceptually distinct from that in prior work. In particular, we propose to regularize the aggregate features from a sequence of activity, not simply per-view features. Whereas in [19] the effect of a discrete egomotion on one image is estimated by linear transformations in the embedding space, the proposed lookahead module takes as input both the history of views and the selected motion when estimating the effects of hypothetical motions.

f) Proprioceptive knowledge: Another useful feature of our approach is that it allows for easy modeling of proprioceptive knowledge such as the current position $\boldsymbol{p}_t$ of a robotic arm. Since the ACTOR module is trained purely through REINFORCE rewards, all other modules may access its output $\boldsymbol{m}_t$ without having to backpropagate extra gradients from the softmax loss. For instance, while the sensor module is fed $\boldsymbol{m}_t$ as input, it does not directly backpropagate any gradients to train ACTOR. Since $\boldsymbol{p}_t$ is a function solely of $(\boldsymbol{m}_1 ... \boldsymbol{m}_t)$, this knowledge is readily available for use in other components of the system without any changes to the training procedure described above. We append appropriate proprioceptive information to the inputs of ACTOR and LOOKAHEAD, detailed in experiments.

g) Greedy softmax classification loss: We found it beneficial at training time to inject softmax classification gradients after every time-step, rather than only at the end of T time steps. To achieve this, the CLASSIFIER module is modified to contain a bank of T classification networks with identical architectures (but different weights, since in general, AGGREGATOR outputs $\boldsymbol{a}_t$ at different time steps may have domain differences). Note that the REINFORCE loss is still computed only at $t = T$. Thus, given that softmax gradients do not pass through the ACTOR module, it remains free to learn non-greedy motion policies.

IV. EXPERIMENTS

We evaluate our approach for object and scene categorization. In both cases, the system must choose how it will move in its 3-D environment such that the full sequence of its actions lead to the most accurate categorization results.

A. Datasets and baselines

While active vision systems have traditionally been tested on custom robotic setups [17] (or simple turn-table-style datasets [6]), we aim to test our system on realistic, off-the-shelf datasets in the interest of benchmarking and reproducibility. We work with two publicly available datasets, SUN360 [44] and GERMS [23].

Our **SUN360 [44]** experiments test a scenario where the agent is exploring a 3-D scene and must intelligently turn itself to see new parts of the scene that will enable accurate scene categorization (bedroom, living room, etc.). SUN360 consists of spherical panoramas of various indoor and outdoor scenes together with scene category labels. We use the 26-category subset (8992 panoramic images) used in [44]. Each panorama by itself represents a 3-D scene instance, around which an agent "moves" by rotating its head, as shown in Fig 2. For our experiments, the agent has a limited field of view (45 degrees) at each time step. We sample discrete views in a 12 elevations (camera pitch) $\times$ 12 azimuths (camera yaw) grid. The pitch and yaw steps are both spaced 30 degrees apart (12$\times$30=360), so that the entire viewing sphere is uniformly sampled on each axis. Starting from a full panorama of size 1024×2048, each 45 degree FOV view is represented first as a 224×224 image, from which 1024-dimensional GoogleNet [45] features are extracted from the penultimate layer. At each time step, the agent can choose to move to viewpoints on a 5$\times$7 grid centered at the current position. We use $T = 3$ time steps.[2] Proprioceptive knowledge in the form of the current elevation angle of the camera is fed into ACTOR and LOOKAHEAD. We use a random 80-20 train-test split. Our use of SUN360 to simulate an active agent in a 3D scene is new and offers a realistic scenario that we can benchmark rigorously; note that previous work on the dataset does a different task, i.e., recognition with the full panorama in hand at once [44], and results are therefore not comparable to our setting.

[2]Episode lengths were set based on learning time for efficient experimentation.

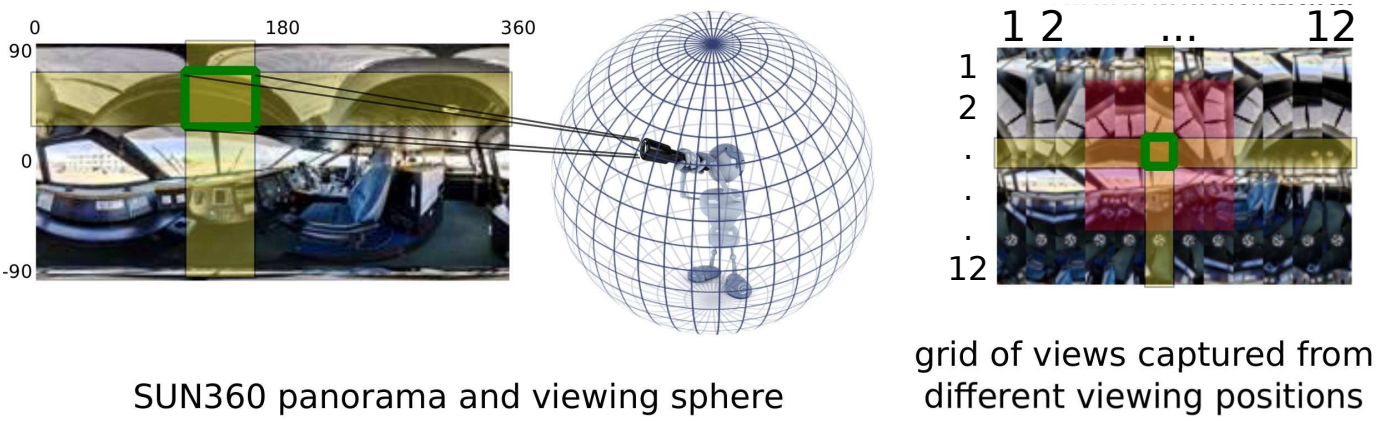


Fig. 2: (Best seen in color) An example of an "airplane interior" class showing how SUN360 spherical panoramas (shown here as a distorted rectangular image on the left) are converted into 12×12 view grids, with each view spanning a field of view of 45 degrees from the panorama. As an illustration, the view at grid coordinates $x = 4$, $y = 6$ outlined in green in the view grid on the right corresponds approximately to the overlap region (also outlined in green) on the left (approximate because of panorama distortions—rectangles in the panorama are not rectangles in the rectified views present in the grid). The 5×7 red shaded region in the view grid (right) shows the motions available to ACTOR when starting from the highlighted view.

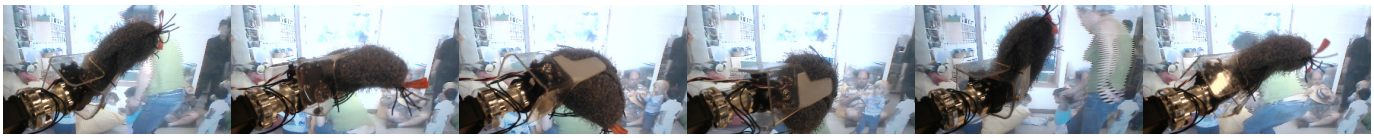


Fig. 3: The GERMS active instance recognition dataset [23] contains videos of a single-axis robotic hand rotating 136 toys against a moving background.

Our **GERMS [23]** experiments consider the scenario where a robot is holding an object and must decide on its next best motion relative to that object, e.g., to gain access to an unseen facet of the object, so as to recognize its instance label. GERMS has 6 videos each (3 train, 3 test) of 136 objects being rotated around different fixed axes, against a television screen displaying moving indoor scenes (see Fig 3). Each video frame is annotated by the angle at which the robotic arm is holding the object. Each video provides one collection of views that our active vision system can traverse at will, for a total of $136 \times 6 = 816$ train/test instances (compared to 8992 on SUN360). While GERMS is small and targets *instance* rather than category recognition, aside from SUN360 it is the most suitable prior dataset facilitating active recognition. Each frame is represented by a 4096-dimensional VGG-net feature vector [46], provided by the authors [23]. We again set episode lengths to $T = 3$ steps. As proprioceptive knowledge, we feed the current position of the robotic hand into ACTOR and LOOKAHEAD. We use the train-test subsets specified by the dataset authors.

h) Baselines:: Our "Look-ahead active RNN" (Sec III-C) and our simpler "Active RNN" (Sec III-B) are evaluated through comparison against three baselines:

- "Single view": which has access to only one view, like the starting view provided to the active systems. A feed-forward neural network is used for this baseline, composed from the appropriate components of the SENSOR and CLASSIFIER modules of our system.
- "Random views (average)": uses the same architecture as "Single view", but has access to T views, with successive views being related by randomly selected motions from the same motion set $\mathcal{M}$ available to the active systems. It uses an ensemble classifier: its output class likelihood at $t = T$ is the average of its independent estimates of class likelihood for each view.
- "Random views (recurrent)": this baseline uses the same core architecture as our Active RNN method, except for the ACTOR module. In its place, random motions are selected. Note that this should be a strong baseline, having nearly all aspects of the proposed approach except for the active view selection module. In particular, it has access to its selected motions in its SENSOR module, and can also learn to intelligently aggregate evidence over views in its AGGREGATOR RNN module.

Hyperparameters for all methods were optimized for overall accuracy on a validation set through iterative search over random combinations per [47].

B. Results

Table I shows the recognition accuracy results for scene categorization (SUN360) and object instance recognition (GERMS), and Figure 4 plots the results as a function of time steps. Both variants of our method outperform the baselines on both datasets, confirming that our active approach successfully learns intelligent view selection strategies. In addition, our Look-ahead active RNN outperforms our Active RNN variant on both datasets, showing the value in simultaneously learning to predict action-conditional next views at the same time we learn the active vision policy. By "looking before leaping" our

Method↓/Dataset→ Performance measure→	SUN360 T=2 acc.	 T=3 acc.	GERMS T=2 acc.	 T=3 acc.
Chance	14.08	14.08	0.74	0.74
Single view	40.12±0.45	40.12±0.45	40.31±0.23	40.31±0.23
Random views (average)	45.71±0.29	50.47±0.37	45.71±0.30	46.97±0.43
Random views (recurrent)	47.74±0.27	51.29±0.21	44.85±0.40	44.24±0.24
Active RNN (ours)	50.76±0.41	57.52±0.46	47.30±0.73	46.86±0.97
Look-ahead active RNN (ours)	**51.72±0.29**	**58.12±0.43**	**48.02±0.68**	**47.99±0.79**
Look-ahead active RNN+average (ours)	49.62±0.43	55.43±0.38	47.00±0.45	**48.31±0.72**

TABLE I: Recognition accuracy for both datasets (mean and std error over 5 runs)

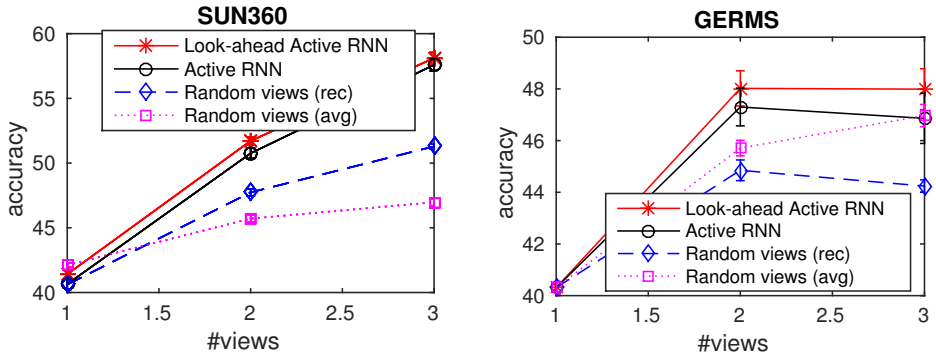


Fig. 4: Evolution of accuracy vs time for various active recognition methods, on SUN360 (left) and GERMS (right).

look-ahead module facilitates beneficial knowledge transfer for the active vision task.

On SUN360, though it represents a much harder active *category recognition* problem, the margins between passive and active methods are pronounced—even against the passive baseline "Random views (recurrent)" which benefits from all aspects of our architecture design except for the action selection. Recall that previous uses of the SUN360 data categorize the panorama as a whole, whereas we do active recognition through a series of actively chosen mini-glimpses in the scene; thus comparison to reported numbers (e.g.,[44]) are not applicable because the tasks are entirely different.

The margins of gains between active and passive are smaller on GERMS. Upon analysis, it became clear this is due to GERMS being a relatively small dataset. Not only is (1) the number of active recognition instances small compared to SUN360 (816 vs. 8992), but (2) different views of the same object instance are naturally closer to each other than different views from a SUN360 panorama view-grid (see Fig 2 and Fig 3) so that even single view diversity is low, and (3) because there is only a single degree of motion compared to two in SUN360. As a result, the number of possible reinforcement learning episodes is also much smaller. Upon inspection, we found that these factors can lead our end-to-end network to overfit to training data (which we countered with more aggressive regularization). In particular, it is problematic if our method achieves zero training error from just single views, so that the network has no incentive to learn to aggregate information across views well. Our active results are in line with those presented as a benchmark in the paper introducing the dataset [23], and we expect more training data is necessary to move further with end-to-end learning on this challenge.

As an upshot, interestingly, we see further improvements on GERMS by averaging the CLASSIFIER modules' outputs *i.e.* class likelihoods estimated from the aggregated features at each time step $t = 1, .., T$ ("Look-ahead active RNN+average (ours)"). Since the above factors make it difficult to learn the optimal AGGREGATOR in an end-to-end system like ours, a second tier of aggregation in the form of averaging over the outputs of our system can yield improvements. In contrast, since SUN offers much more training data, averaging over per-timestep CLASSIFIER outputs significantly *reduces* the performance of the system, compared to directly using the last timestep output. This is exactly as one would hope for a successful end-to-end training. This reasoning is further supported by the fact that "Random views (average)" shows slightly poorer performance than "Random views (recurrent)" on GERMS, but vastly better on SUN360.

Indeed, the large performance gap between "Random views (average)" and "Random views (recurrent)" on SUN360 points to an important advantage of treating object/scene categorization as a grounded, sequence-based decision process. The ability to intelligently fuse observations over time-steps based on both the views themselves and the camera motions relating them offers substantial rewards. In contrast, the current computer vision literature in visual categorization is largely focused on categorization strategies that process individual images outside of the context of any agent motion or sequential data, much like the "Single view" or "Random views (average)" baselines. We see these empirical results as an exciting prompt for future work in this space. They also suggest the need for increased efforts creating large 3-D and video benchmark datasets (in the spirit of SUN360 and GERMS and beyond) to support such vision research, allowing us to

systematically study these scenarios outside of robot platforms.

The result on SUN360 in particular is significant since, to the best of our knowledge, no prior active recognition approach has been shown to successfully handle any comparably complex dataset, containing real-world categories. While active categorization is technically challenging compared to recognition as discussed in Sec II, datasets like SUN360, containing complex visual data and requiring significant semantic understanding to plan motion sequences well for categorization, may actually be most suited to showing the advantages of the active recognition paradigm. By employing intelligent view selection and evidence fusion, they perform well even though individual views are often highly ambiguous (*e.g.*, say, patches of the sky).

V. Conclusions

We presented a new end-to-end approach for active visual categorization. Our framework simultaneously learns 1) how the system should move to improve its sequence of observations, and 2) how a sequence of future observations is likely to change conditioned on its possible actions. We show the impact on object and scene recognition, where our active approach makes sizeable strides over single view and passively moving systems. Furthermore, we establish the positive impact in treating all components of the active recognition system simultaneously. All together, the results are encouraging evidence that modern visual recognition algorithms can venture further into unconstrained, sequential data, moving beyond the static image snapshot labeling paradigm.

Fig. 5: Examples of selected views (highlighted in red) by our method. Please see supplementary for more examples. Each row represents our method selecting views among a 12×12 view grid constructed from a SUN360 panorama. Our method intelligently decides where to look next to resolve scene class ambiguity. For example on the third row, the method moves from ambiguous sky patches to more detailed ground patches to identify the scene (Note the wrap-around from views 2 to 3).

REFERENCES

[1] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) (April 2015) 1–42

[2] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014. Springer (2014) 740–755

[3] Andreopoulos, A., Tsotsos, J.: 50 years of object recognition: Directions forward. In: CVIU. (2013)

[4] Wilkes, D., Tsotsos, J.: Active object recognition. In: CVPR. (1992)

[5] Dickinson, S., Christensen, H., Tsotsos, J., Olofsson, G.: Active object recognition integrating attention and viewpoint control. In: CVIU. (1997)

[6] Schiele, B., Crowley, J.: Transinformation for active object recognition. In: ICCV. (1998)

[7] Callari, F., Ferrie, F.: Active object recognition: Looking for differences. (2001)

[8] Aloimonos, J., Weiss, I., Bandyopadhyay, A.: Active vision. In: IJCV. (1988)

[9] Brentano, F.: Psychologie vom empirischen Standpunkte. (1874)

[10] Bajcsy, R.: Active perception. In: IEEE. (1988)

[11] Ballard, D.: Animate vision. In: AI. (1991)

[12] Mishra, A., Aloimonos, Y., Fermuller, C.: Active segmentation for robotics. In: IROS. (2009)

[13] Andreopoulos, A., Tsotsos, J.: A theory of active object localization. In: ICCV. (2009)

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

[15] Garcia, A.G., Vezhnevets, A., Ferrari, V.: An active search strategy for efficient object detection. (2015)

[16] Soatto, S.: Actionable information in vision. In: ICCV. (2009)

[17] Ramanathan, V., Pinz, A.: Active object categorization on a humanoid robot. In: VISAPP. (2011) 235–241

[18] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shape modeling. In: CVPR. (2015)

[19] Jayaraman, D., Grauman, K.: Learning image representations tied to ego-motion. In: ICCV. (2015)

[20] Yu, X., Fermuller, C., Teo, C.L., Yang, Y., Aloimonos, Y.: Active scene recognition with vision and language. In: CVPR. (2011)

[21] Borotschnig, H., Paletta, L., Prantl, M., Pinz, A., et al.: Active object recognition in parametric eigenspace. In: BMVC. (1998)

[22] Paletta, L., Pinz, A.: Active object recognition by view integration and reinforcement learning. In: RAS. (2000)

[23] Malmir, M., Sikka, K., Forster, D., Movellan, J., Cottrell, G.W.: Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning

[24] Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: NIPS. (2014)

[25] Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. In: ICLR. (2015)

[26] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML. (2015)

[27] Bazzani, L., Larochelle, H., Murino, V., Ting, J.A., Freitas, N.d.: Learning attentional policies for tracking and recognition in video with deep networks. In: ICML. (2011)

[28] Sermanet, P., Frome, A., Real, E.: Attention for fine-grained categorization. arXiv (2014)

[29] Butko, N., Movellan, J.: Optimal scanning for faster object detection. In: CVPR. (2009)

[30] Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating the future by watching unlabeled video. (29 April 2015)

[31] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)

[32] Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. (11 March 2015)

[33] Ding, W., Taylor, G.W.: "mental rotation" by optimizing transforming distance. In: NIPS DL Workshop. (2014)

[34] Flynn, J., Neulander, I., Philbin, J., Snavely, N.: DeepStereo: Learning to predict new views from the world's imagery. (22 June 2015)

[35] Walker, J., Gupta, A., Hebert, M.: Dense optical flow prediction from a static image. arXiv preprint arXiv:1505.00295 (2015)

[36] Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: ICCV. (2015)

[37] Bowling, M., Ghodsi, A., Wilkinson, D.: Action respecting embedding. In: ICML. (2005)

[38] Cohen, T.S., Welling, M.: Transformation properties of learned visual representations. arXiv preprint arXiv:1412.7659 (2014)

[39] Stober, J., Miikkulainen, R., Kuipers, B.: Learning geometry from sensorimotor experience. In: Development and Learning (ICDL), 2011 IEEE International Conference on. (2011)

[40] Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. arXiv preprint arXiv:1505.00256 (2015)

[41] Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-End training of deep visuomotor policies. (2 April 2015)

[42] Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M.: Embed to control: A locally linear latent dynamics model for control from raw images. (24 June 2015)

[43] Williams, R.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: JMLR. (1992)

[44] Xiao, J., Ehinger, K., Oliva, A., Torralba, A., et al.: Recognizing scene viewpoint using panoramic place representation. In: CVPR. (2012)

[45] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9

[46] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[47] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. In: JMLR. (2012)

This document provides information supplementary to the main paper. Procedure block 1 lists the steps involved in the forward pass during training/inference. Fig 6 provides a detailed schematic diagram explaining the architectures of and connections amongst the modules of our active vision system. Dataset-specific details of training-related choices are provided in Sec VI. Finally, at the end of this document, we present some examples of view sequences selected by our approach.

VI. Dataset-specific training details

Training for both datasets followed the general procedure described in Sec 3 of the main paper. In this section, we mention details of the training procedures specific to the two datasets. For SUN360, the full system shown in the schematic of Fig 6 was trained end-to-end in one shot. No dropout was employed in either of the Dropout layers in Fig 6.

For GERMS, the paucity of data and the relative simplicity of the task created some difficulties during training. First, for convergence, it was necessary to initialize end-to-end training at $T = 3$ with the weights of the same network trained end-to-end at $T = 1$. Given the shortcomings of the dataset though (small training set and ease of task), the $T = 1$ network had already achieved zero training error. This meant, the $T = 3$ network initialized with $T = 1$ weights did not have sufficient error gradients from the softmax classification loss for training to proceed. To handle this, we introduced dropout rates of 2e-5 and 0.5 at the CNN feature layer (as a kind of data augmentation) and preceding the AGGREGATOR input layer, respectively.

The other choices and hyperparameters involved in training are listed below: (marked with [*] when selected through cross-validation; others were fixed based on values known to work well for similar systems in the past)

- Gradient descent variant: Stochastic gradient descent (batch-size 32) with momentum (0.9)
- Starting learning rate [*]: 7e-3 (SUN360), 3e-3 (GERMS)
- Learning rate schedule: Linear decay towards a minimum learning rate of 1e-5 after 800 epochs (both datasets)
- Weight decay rate (L2 regularization on network weights): 5e-3 for both
- Lookahead λ [*]: 1.5 (SUN360), 0.05 (GERMS). Explanation for this hyperparameter: As specified in Sec 3 in the main paper, the net error from which derivatives are computed for modules other than ACTOR is:

$$\sum_{t=1}^{T}(\text{classification } L_{\text{softmax}} \text{ at } t) + \lambda \sum_{t=1}^{T} d_{\text{lookahead}}(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t) \tag{4}$$

- Weight initialization: uniform between 0 and 0.1 for all weights, except the recurrent network feedback weights (Linear(256,256) block inside AGGREGATOR in Fig 6), whose weights and biases were initialized to identity $\mathcal{I}_{256\times256}$ and zero $\mathbf{0}_{256\times1}$ respectively.
- Convergence criterion: Training is terminated when validation classification accuracy does not fall for 50 epochs. The rough number of epochs (full passes over training data) required for convergence are 250 for SUN360 and 150 for GERMS. Total training time is approximately 2 hours for SUN360 and 1.5 hours for GERMS.

If the paper is accepted, code and processed data will be made available for easy reproducibility.

VII. Examples of selected views

At the end of this document, we provide some examples of views selected by our approach when presented with SUN360 panoramas. Each example consists of two rows of $T = 3$ panels each. The top row corresponds to the views selected, and the bottom row shows where on the 12×12 view grid they are selected from. The eye graphics at the corners of the top row views indicate the elevation angle of each view (legend provided in the bottom row, left in each example). Note that views appear inverted when elevation angle is >90 or < -90 degrees, because these correspond to the agent "bending over backwards".

In each example, the first view is chosen at random for presentation to the system, and the remaining views are chosen automatically by the system itself, within a 5×7 grid neighborhood of the preceding view. In the bottom row of panels (view grids), the transparent red square shows the view that is currently being observed, and the 5×7 grid of transparent yellow squares shows the views that are available for selection at the next time step.

True category names are indicated above each example (left top), and the likelihood of the true category, as returned by the CLASSIFIER module of our system at every time-step, is shown in parentheses above each corresponding view. Note that these are *aggregated* likelihood scores *i.e.*, the score at time t is based on observations at times 1,2...t. In addition, we list the top-3 most likely categories (among 26 total) returned by our model above each view.

As can be seen from these examples (also explained in accompanying captions for a few cases, as an illustration), in many cases, (1) the automatic motion/view selection performed by our ACTOR module, serves to increase the confidence of the right category, and helps disambiguate among confusing categories. (2) AGGREGATOR efficiently fuses information across time-steps, often showing evidence of accounting for the agent's motions relative to previous views.

References

[1] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) (April 2015) 1–42

[2] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014. Springer (2014) 740–755

[3] Andreopoulos, A., Tsotsos, J.: 50 years of object recognition: Directions forward. In: CVIU. (2013)

[4] Wilkes, D., Tsotsos, J.: Active object recognition. In: CVPR. (1992)

[5] Dickinson, S., Christensen, H., Tsotsos, J., Olofsson, G.: Active object recognition integrating attention and viewpoint control. In: CVIU. (1997)

[6] Schiele, B., Crowley, J.: Transinformation for active object recognition. In: ICCV. (1998)

[7] Callari, F., Ferrie, F.: Active object recognition: Looking for differences. (2001)

[8] Aloimonos, J., Weiss, I., Bandyopadhyay, A.: Active vision. In: IJCV. (1988)

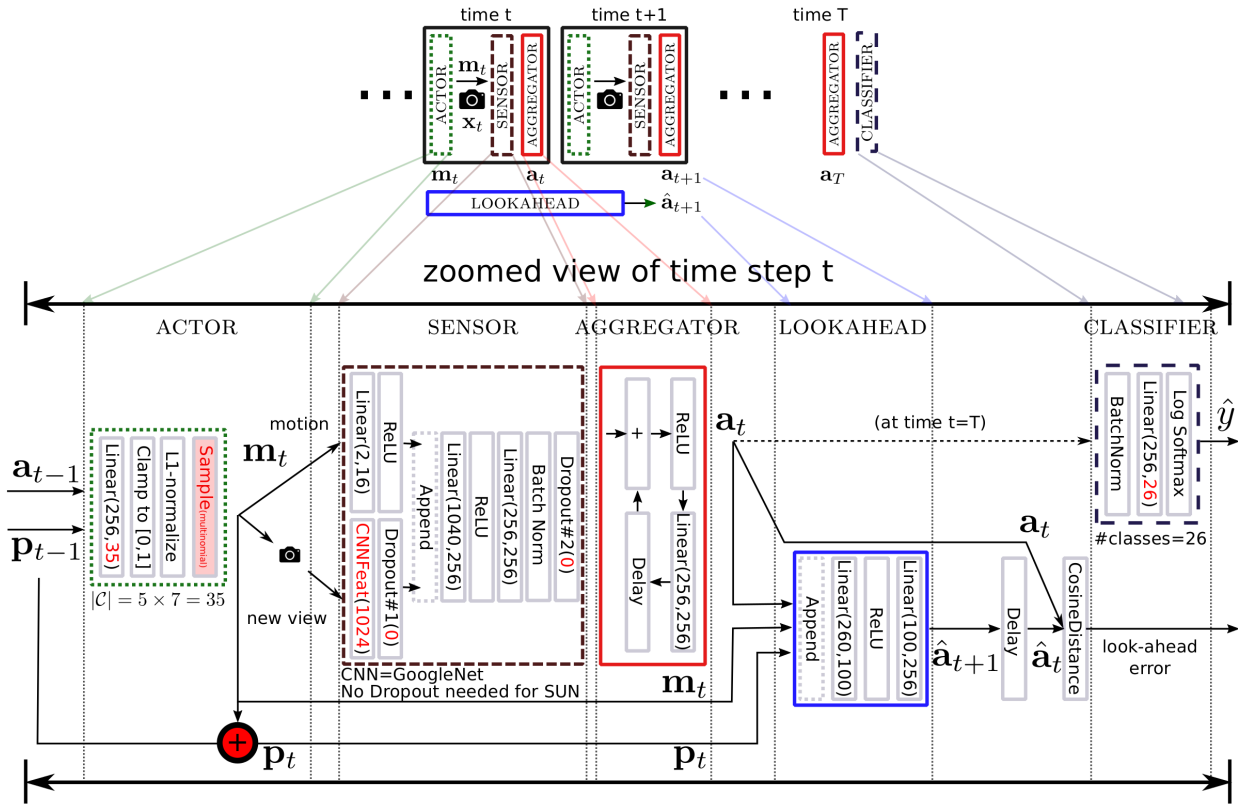


Fig. 6: A detailed schematic diagram showing the architectures of and connections amongst our active vision system modules. The small schematic at the top (repeated from Fig1 in the paper) presents a succinct bird's-eye view of information flow within as well as between time steps, and the large schematic below zooms into the operations at some given time step t in more detail. Processing proceeds from left to right, with arrows to disambiguate where necessary. In the bottom schematic, "Linear(a,b)" denotes fully connected layers which transform a-length vector inputs to b-length vector outputs. The "Clamp" operator in ACTOR is a squashing function that sets both upper and lower limits on its inputs. The red "Sample" layer in ACTOR takes the weights of a multinomial pdf as input and samples stochastically from the distribution to produce its output (gradients cannot be backpropagated through this layer; it is trained through REINFORCE [43] instead of SGD from the classification loss). "Delay" layers store inputs internally for one time-step and output them at the next time-step. Other layer names in the schematic are self-explanatory. Input and output sizes of some layers are marked in red to denote that these are parameters derived from dataset-related choices — these are set for our SUN360 experiments in this schematic, and explanations are shown below each module. Note that AGGREGATOR is a recurrent neural network, and LOOKAHEAD may be considered a "predictive" autoencoder, that reduces its input features (appended together with the current agent motion $\boldsymbol{m}_t$) to a more compact representation in its bottleneck layer before producing its prediction of its next time-step input.

[9] Brentano, F.: Psychologie vom empirischen Standpunkte. (1874)
[10] Bajcsy, R.: Active perception. In: IEEE. (1988)
[11] Ballard, D.: Animate vision. In: AI. (1991)
[12] Mishra, A., Aloimonos, Y., Fermuller, C.: Active segmentation for robotics. In: IROS. (2009)
[13] Andreopoulos, A., Tsotsos, J.: A theory of active object localization. In: ICCV. (2009)
[14] Helmer, S., Meger, D., Viswanathan, P., McCann, S., Dockrey, M., Fazli, P., Southey, T., Muja, M., Joya, M., Little, J., et al.: Semantic robot vision challenge: Current state and future directions. In: IJCAI workshop. (2009)
[15] Garcia, A.G., Vezhnevets, A., Ferrari, V.: An active search strategy for efficient object detection. (2015)
[16] Soatto, S.: Actionable information in vision. In: ICCV. (2009)
[17] Ramanathan, V., Pinz, A.: Active object categorization on a humanoid robot. In: VISAPP. (2011) 235–241
[18] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shape modeling. In: CVPR. (2015)
[19] Jayaraman, D., Grauman, K.: Learning image representations tied to ego-motion. In: ICCV. (2015)
[20] Yu, X., Fermuller, C., Teo, C.L., Yang, Y., Aloimonos, Y.: Active scene recognition with vision and language. In: CVPR. (2011)
[21] Borotschnig, H., Paletta, L., Prantl, M., Pinz, A., et al.: Active object recognition in parametric eigenspace. In: BMVC. (1998)
[22] Paletta, L., Pinz, A.: Active object recognition by view integration and reinforcement learning. In: RAS. (2000)
[23] Malmir, M., Sikka, K., Forster, D., Movellan, J., Cottrell, G.W.: Deep q-learning for active recognition of germs: Baseline performance on a standardized dataset for active learning
[24] Mnih, V., Heess, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: NIPS. (2014)
[25] Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. In: ICLR. (2015)
[26] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: ICML. (2015)
[27] Bazzani, L., Larochelle, H., Murino, V., Ting, J.A., Freitas, N.d.: Learning attentional policies for tracking and recognition in video with deep networks. In: ICML. (2011)
[28] Sermanet, P., Frome, A., Real, E.: Attention for fine-grained categorization. arXiv (2014)
[29] Butko, N., Movellan, J.: Optimal scanning for faster object detection.

In: CVPR. (2009)
[30] Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating the future by watching unlabeled video. (29 April 2015)
[31] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604 (2014)
[32] Kulkarni, T.D., Whitney, W., Kohli, P., Tenenbaum, J.B.: Deep convolutional inverse graphics network. (11 March 2015)
[33] Ding, W., Taylor, G.W.: "mental rotation" by optimizing transforming distance. In: NIPS DL Workshop. (2014)
[34] Flynn, J., Neulander, I., Philbin, J., Snavely, N.: DeepStereo: Learning to predict new views from the world's imagery. (22 June 2015)
[35] Walker, J., Gupta, A., Hebert, M.: Dense optical flow prediction from a static image. arXiv preprint arXiv:1505.00295 (2015)
[36] Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: ICCV. (2015)
[37] Bowling, M., Ghodsi, A., Wilkinson, D.: Action respecting embedding. In: ICML. (2005)
[38] Cohen, T.S., Welling, M.: Transformation properties of learned visual representations. arXiv preprint arXiv:1412.7659 (2014)
[39] Stober, J., Miikkulainen, R., Kuipers, B.: Learning geometry from sensorimotor experience. In: Development and Learning (ICDL), 2011 IEEE International Conference on. (2011)
[40] Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. arXiv preprint arXiv:1505.00256 (2015)
[41] Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-End training of deep visuomotor policies. (2 April 2015)
[42] Watter, M., Springenberg, J.T., Boedecker, J., Riedmiller, M.: Embed to control: A locally linear latent dynamics model for control from raw images. (24 June 2015)
[43] Williams, R.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: JMLR. (1992)
[44] Xiao, J., Ehinger, K., Oliva, A., Torralba, A., et al.: Recognizing scene viewpoint using panoramic place representation. In: CVPR. (2012)
[45] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
[46] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
[47] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. In: JMLR. (2012)

Procedure 1 Forward propagation (training/inference time)

Input: 3D instance X, together with:

- projection function $P(X, \boldsymbol{p})$ denoting the view captured from camera pose $\boldsymbol{p}$
- proprioception function $f(\boldsymbol{p})$ of the current position, which is known to the active vision system *e.g.* wrist position, or gravity direction.

Output: $\hat{y}$, the predicted label for instance X.

procedure FORWARDONESTEP($t, \boldsymbol{a}_{t-1}, \boldsymbol{p}_{t-1}, \hat{\boldsymbol{a}}_t$) ▷ 1 forward propagation step
  $\boldsymbol{m}_t \leftarrow$ ACTOR$(\boldsymbol{a}_{t-1}, f(\boldsymbol{p}_{t-1}))$ ▷ motor command sampled from $\mathcal{C}$ to adjust camera
  $\boldsymbol{p}_t \leftarrow \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$ ▷ camera pose update
  $\boldsymbol{x}_t \leftarrow P(X, \boldsymbol{p}_t)$ ▷ capture new view
  $\boldsymbol{s}_t \leftarrow$ SENSOR$(\boldsymbol{x}_t, \boldsymbol{m}_t)$ ▷ per-view processing
  $\boldsymbol{a}_t \leftarrow$ AGGREGATOR$(\boldsymbol{a}_{t-1}, \boldsymbol{s}_t)$ ▷ evidence fusion
  if $t > 1$ **then** ▷ relevant only at training time
    $\hat{\boldsymbol{a}}_t \leftarrow$ LOOKAHEAD$(\boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1}, f(\boldsymbol{p}_t))$ ▷ look-ahead prediction of current time-step feature
    look-ahead error $\zeta_t \leftarrow d(\boldsymbol{a}_t, \hat{\boldsymbol{a}}_t)$
  return $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t$

$\boldsymbol{a}_0 \leftarrow \mathbf{0}$ ▷ initialization
$\boldsymbol{p}_0 \leftarrow$ random position
for t =1,2,...T **do** ▷ move, observe, aggregate in a loop
  $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t \leftarrow$ FORWARDONESTEP$(t, \boldsymbol{a}_t, \boldsymbol{p}_{t-1})$
$\hat{y} \leftarrow$ CLASSIFIER$(\boldsymbol{a}_t)$ ▷ final class prediction
return $\hat{y}$

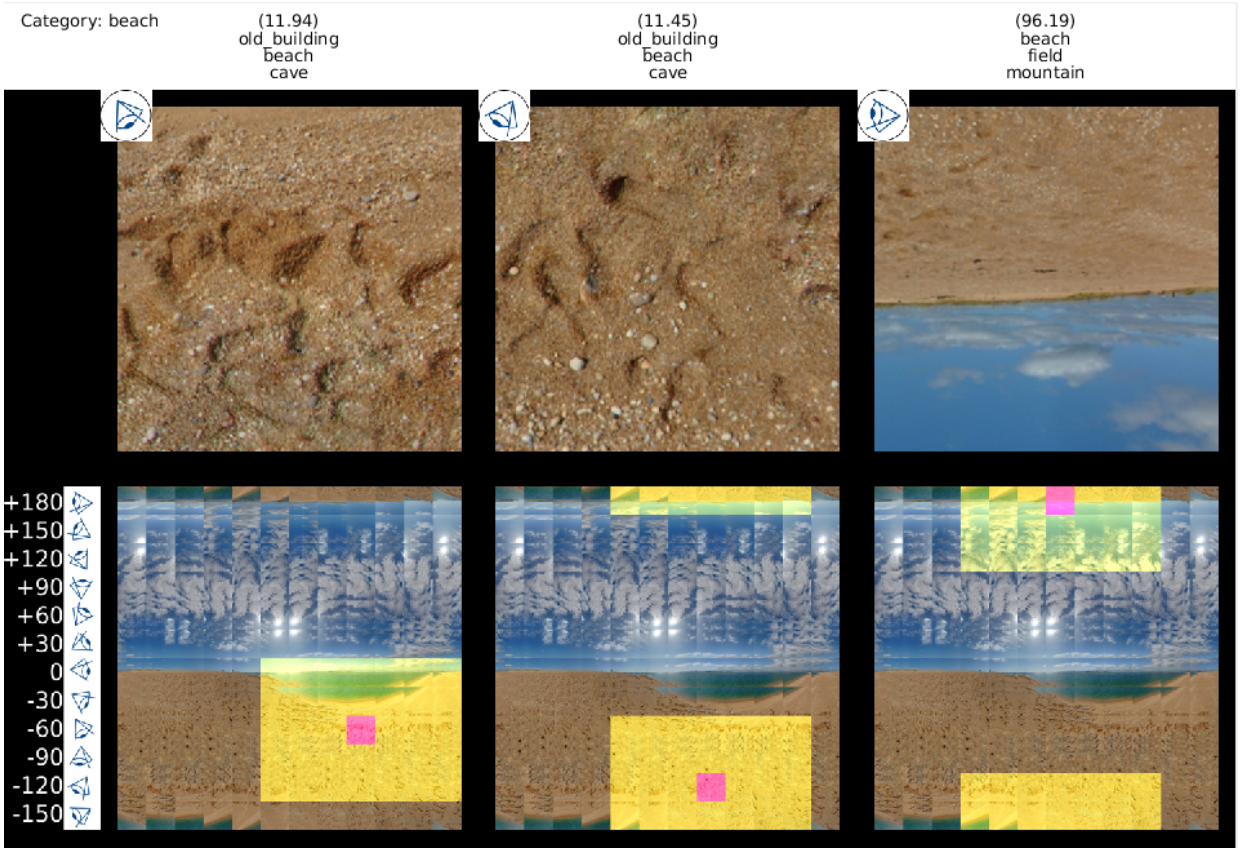


Fig. 7: The wet dry sand in this beach resembles old mud buildings, but once the agent looks up to see the sky and water at the horizon, it becomes 96% confident of "beach".

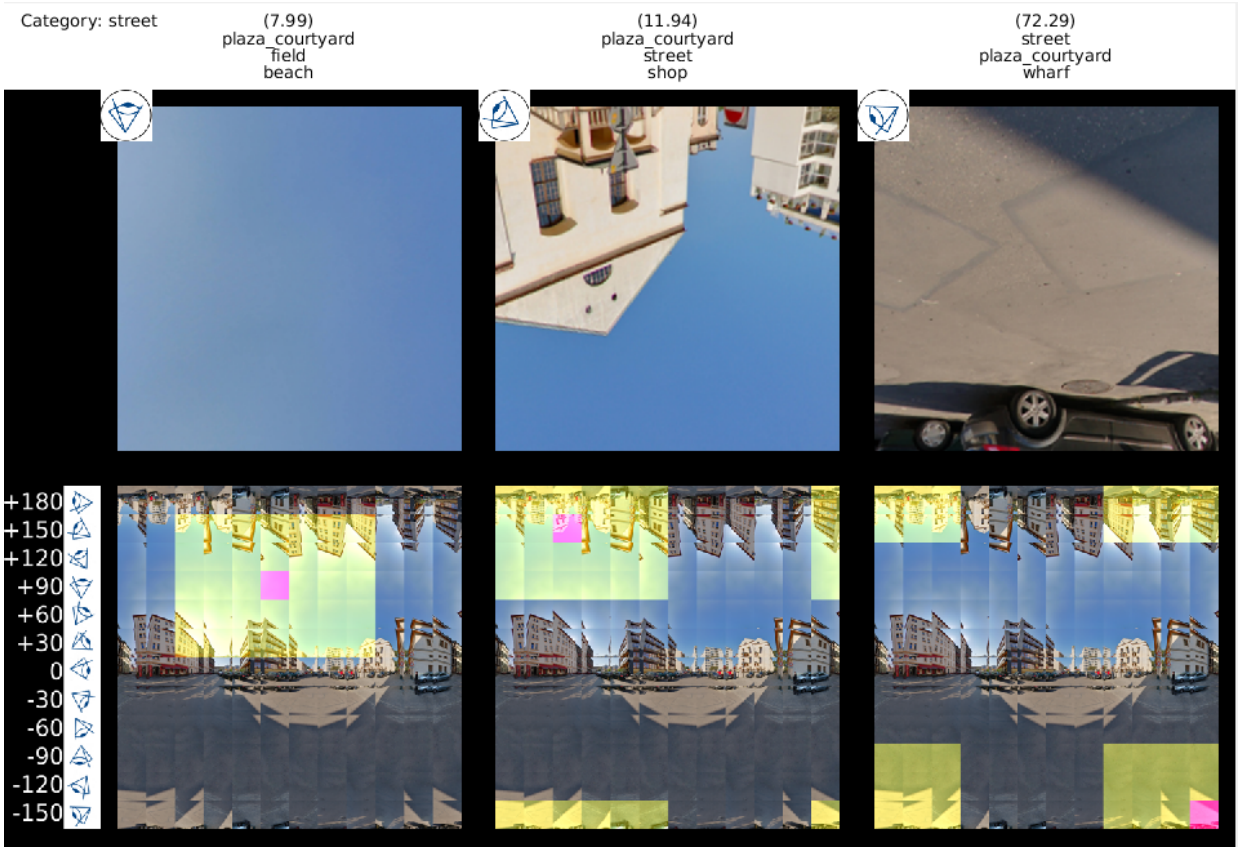


Fig. 8: The random starting view (at T=1) of the sky is not informative, but still enough to start guessing outdoor categories rather than indoor. The architecture of the building facade suggests "plaza", but with low confidence, at $T = 2$. When the system looks down to disambiguate its top guesses ("plaza"— typically paved with brick and "street"— typically tarmac), it sees car wheels and tarmac, immediately suggesting "street" with high confidence.

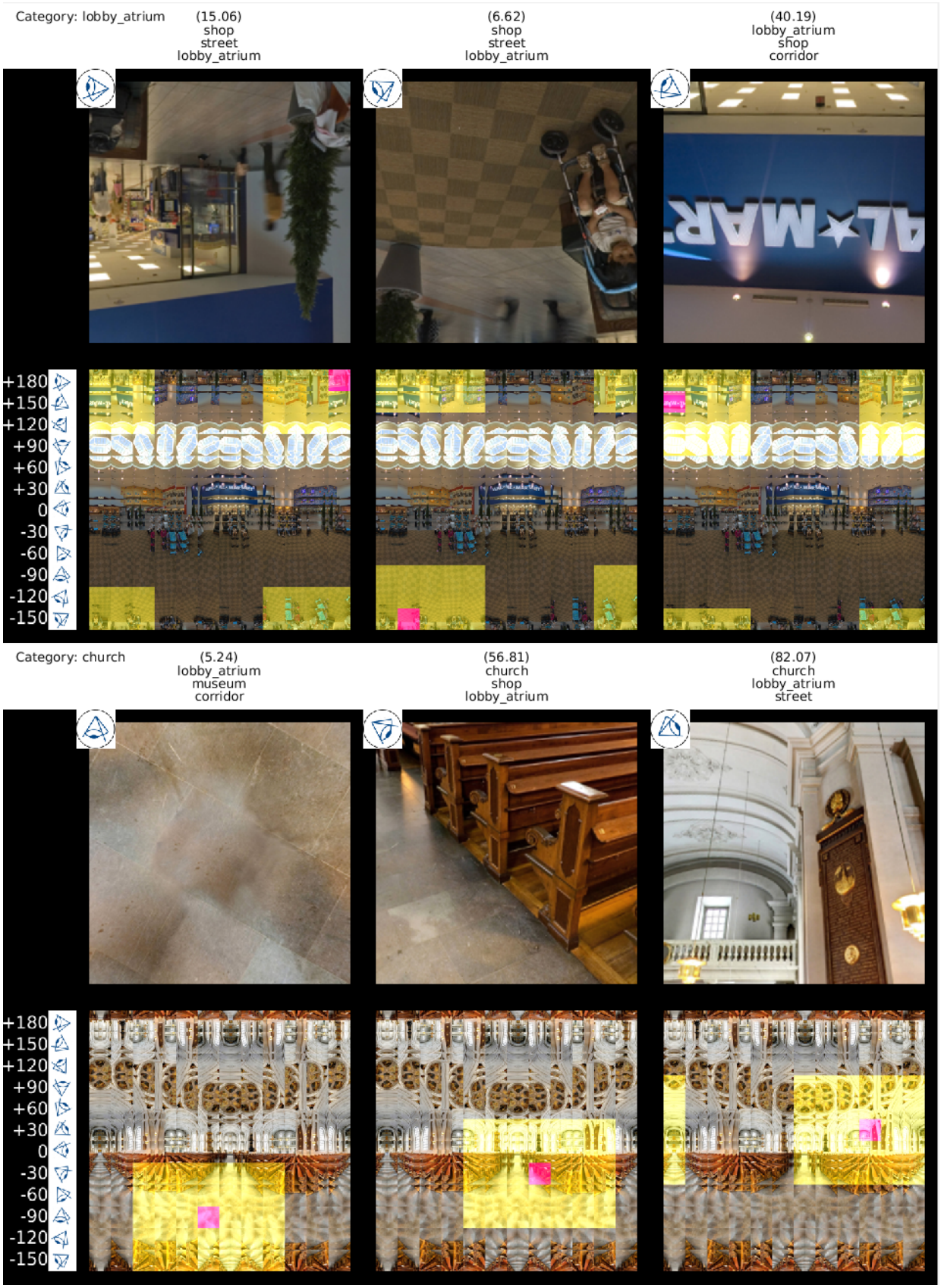
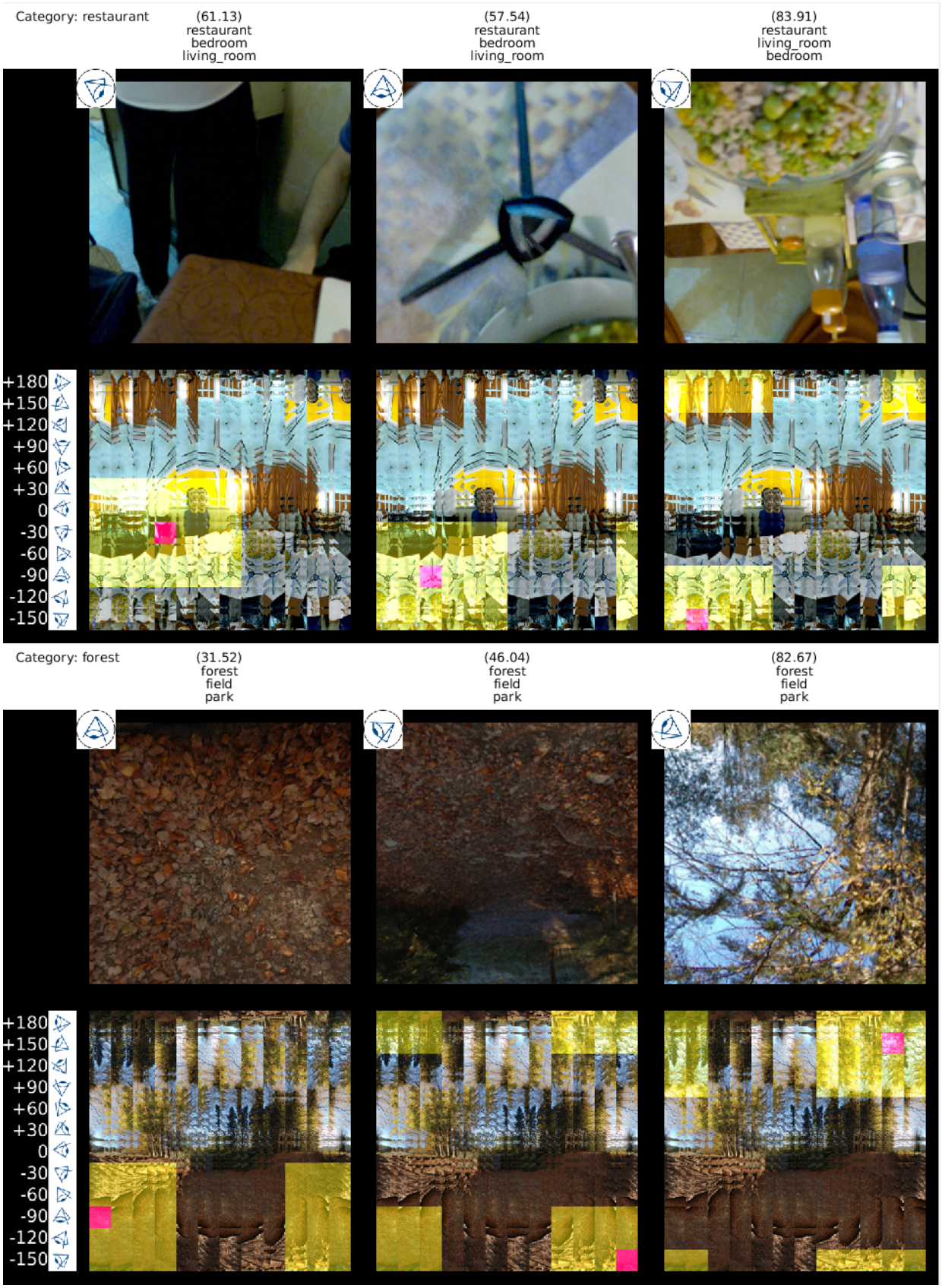
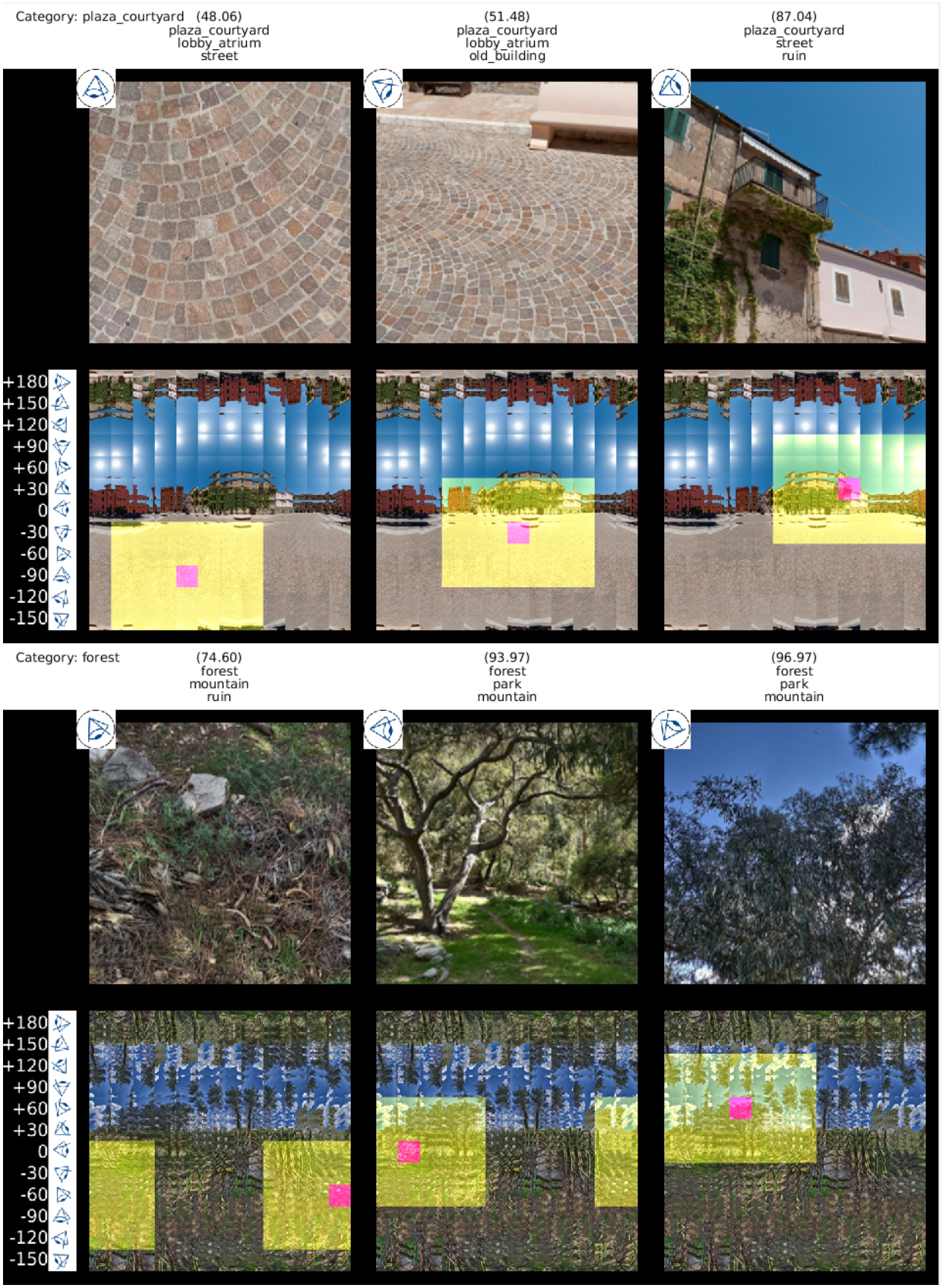
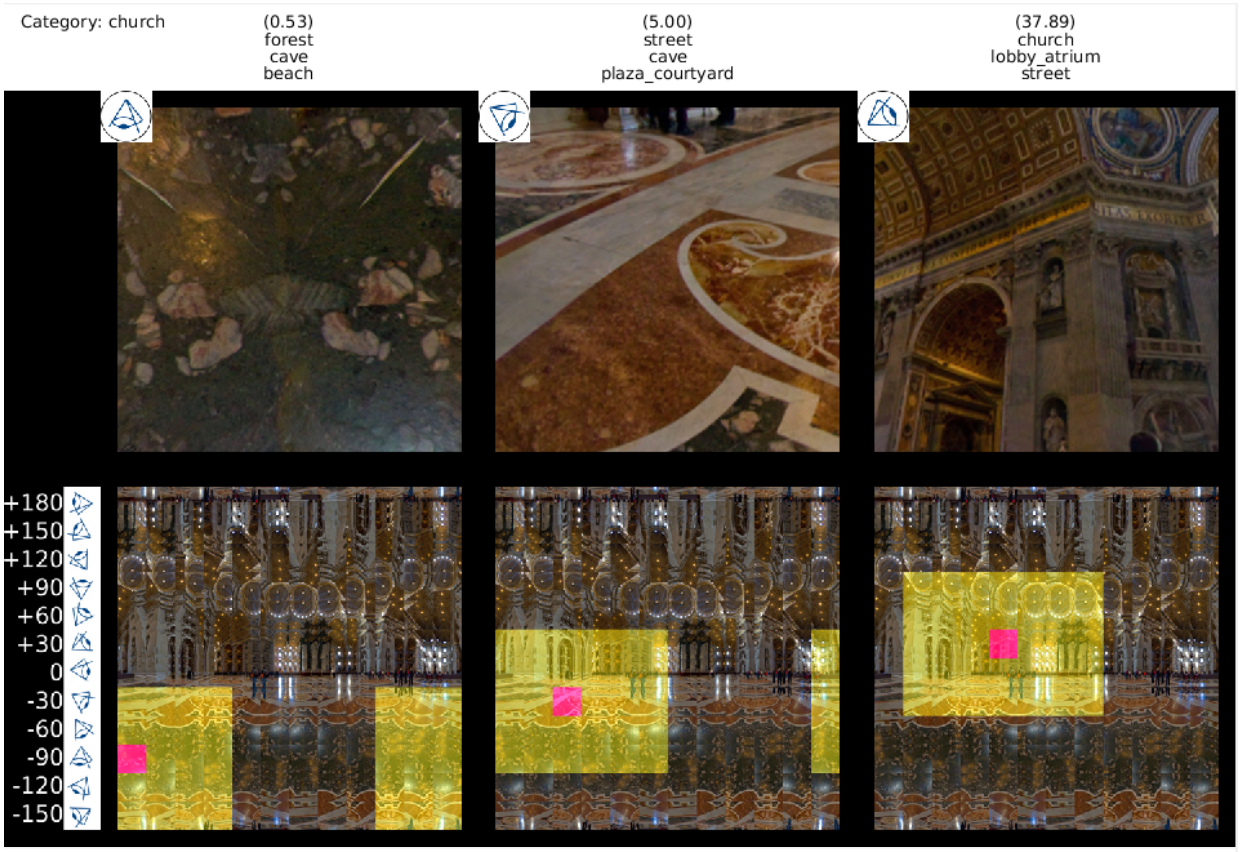


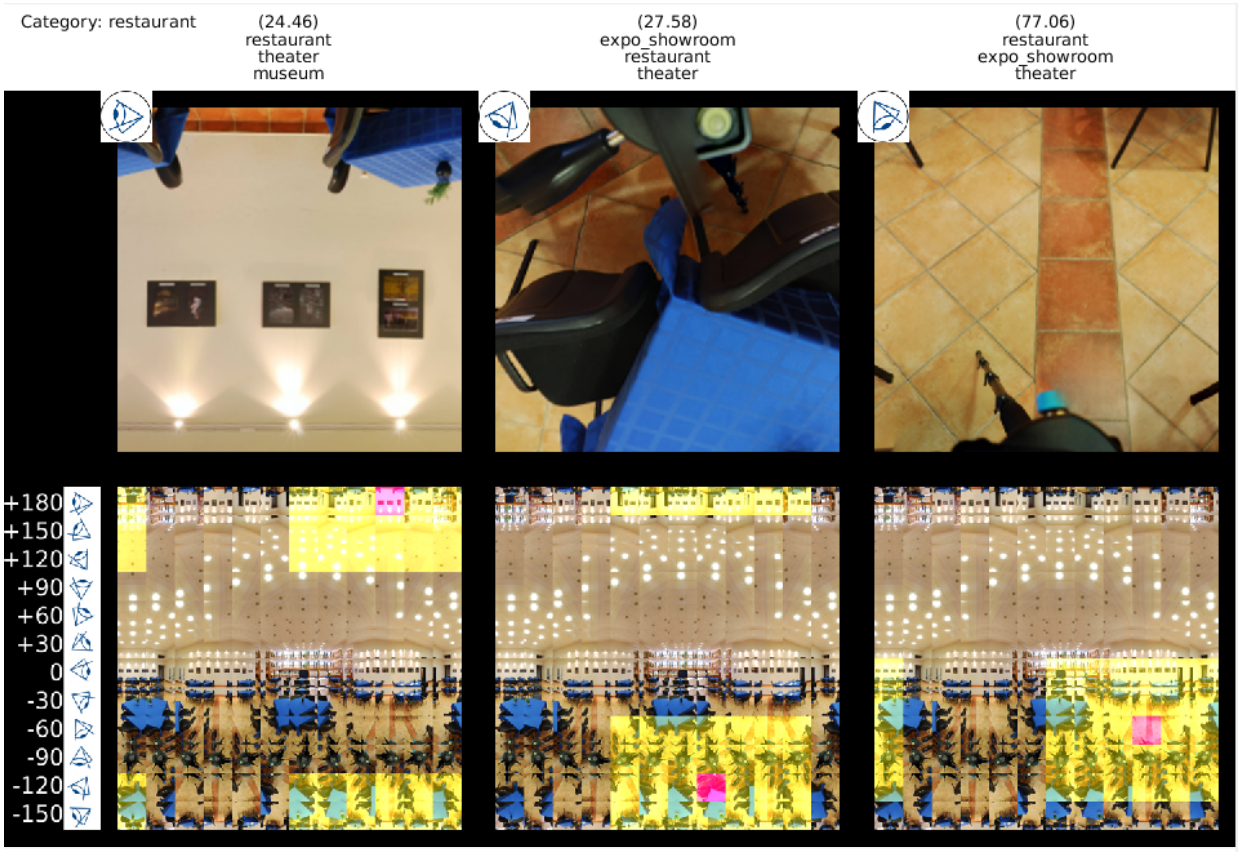
Fig. 9: The view shown to the agent at $T = 1$ suggests a store-front, and thus "shop", but after exploring the scene and finding a large indoor space near the shop, the agent revises its guess correctly to "lobby atrium".


Category: restaurant
(61.13)
restaurant
bedroom
living_room
(57.54)
restaurant
bedroom
living_room
(83.91)
restaurant
living_room
bedroom
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150
Category: forest
(31.52)
forest
field
park
(46.04)
forest
field
park
(82.67)
forest
field
park
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

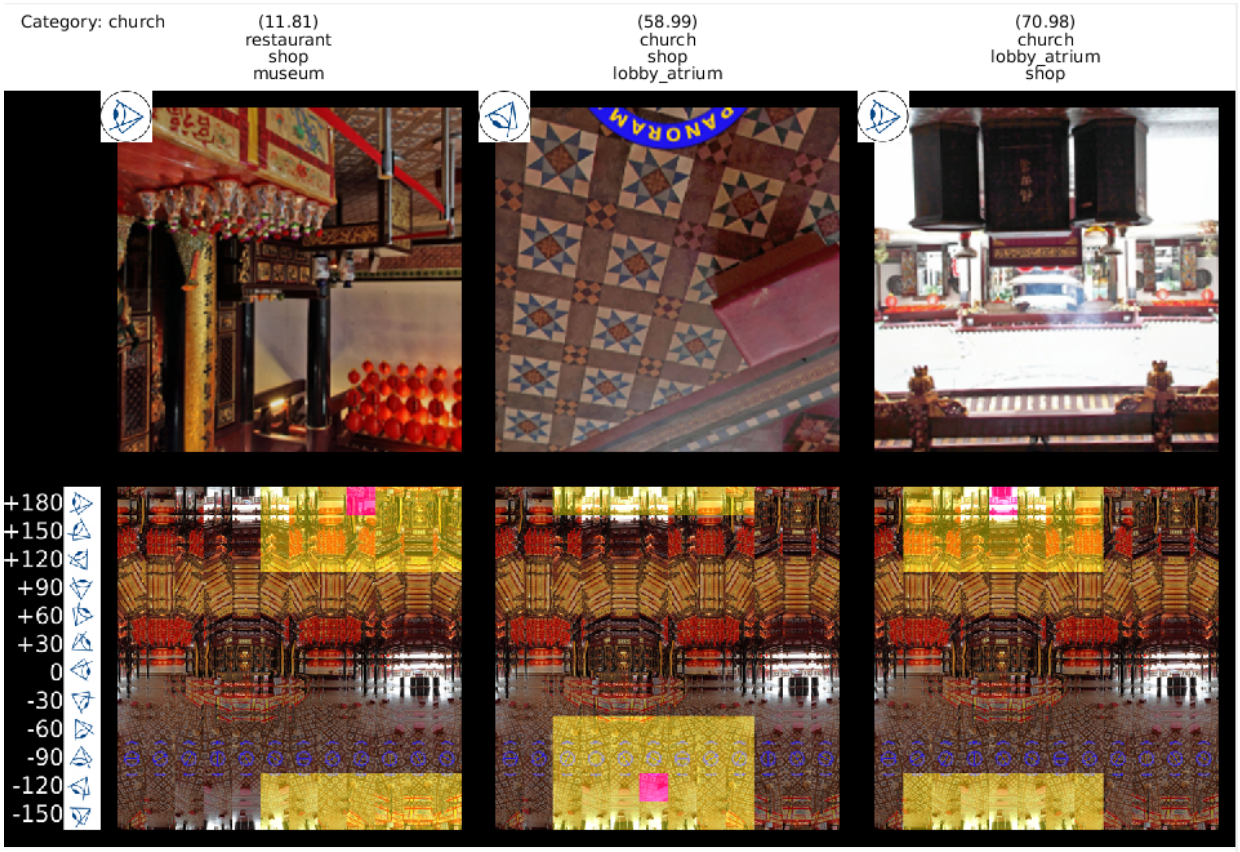

Category: plaza_courtyard (48.06)
plaza_courtyard
lobby_atrium
street
(51.48)
plaza_courtyard
lobby_atrium
old_building
(87.04)
plaza_courtyard
street
ruin
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150
Category: forest
(74.60)
forest
mountain
ruin
(93.97)
forest
park
mountain
(96.97)
forest
park
mountain
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

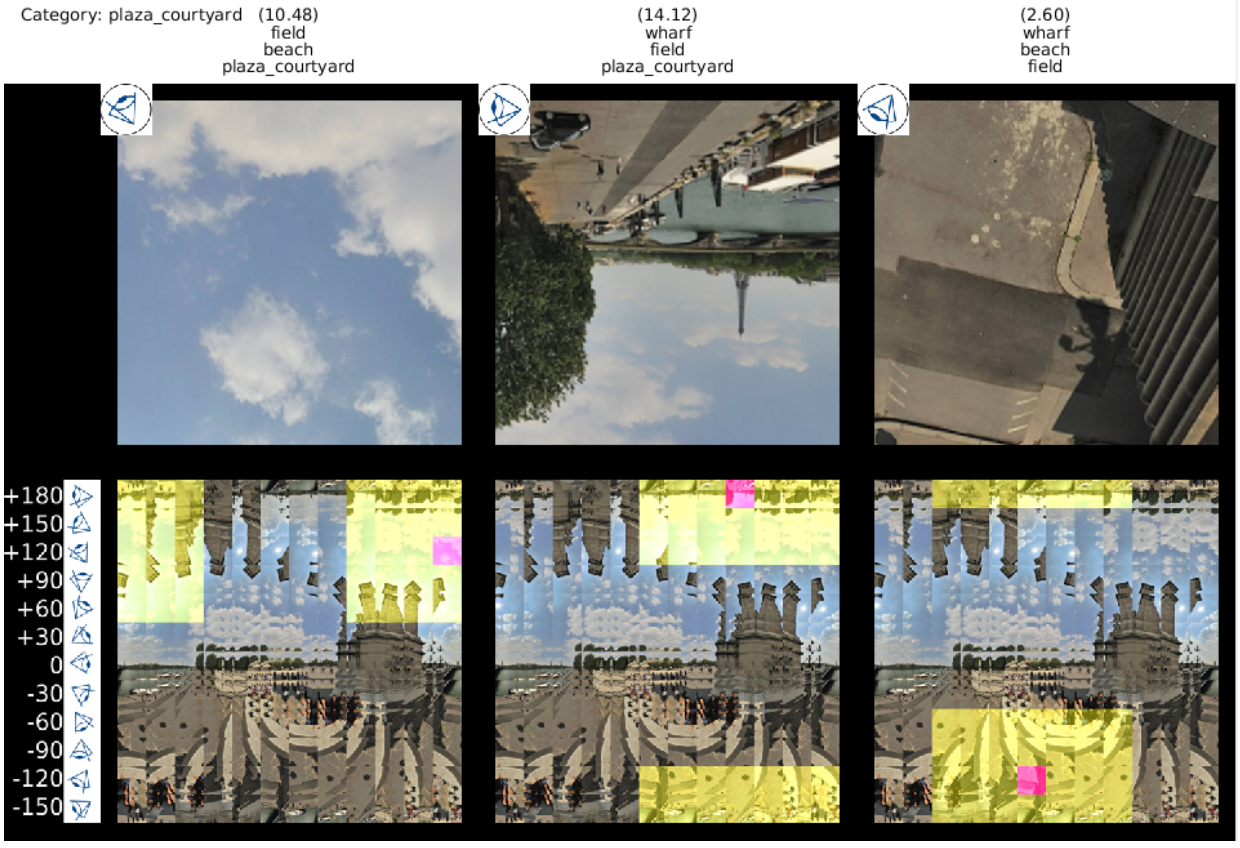

Category: church
(0.53)
forest
cave
beach
(5.00)
street
cave
plaza_courtyard
(37.89)
church
lobby_atrium
street
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

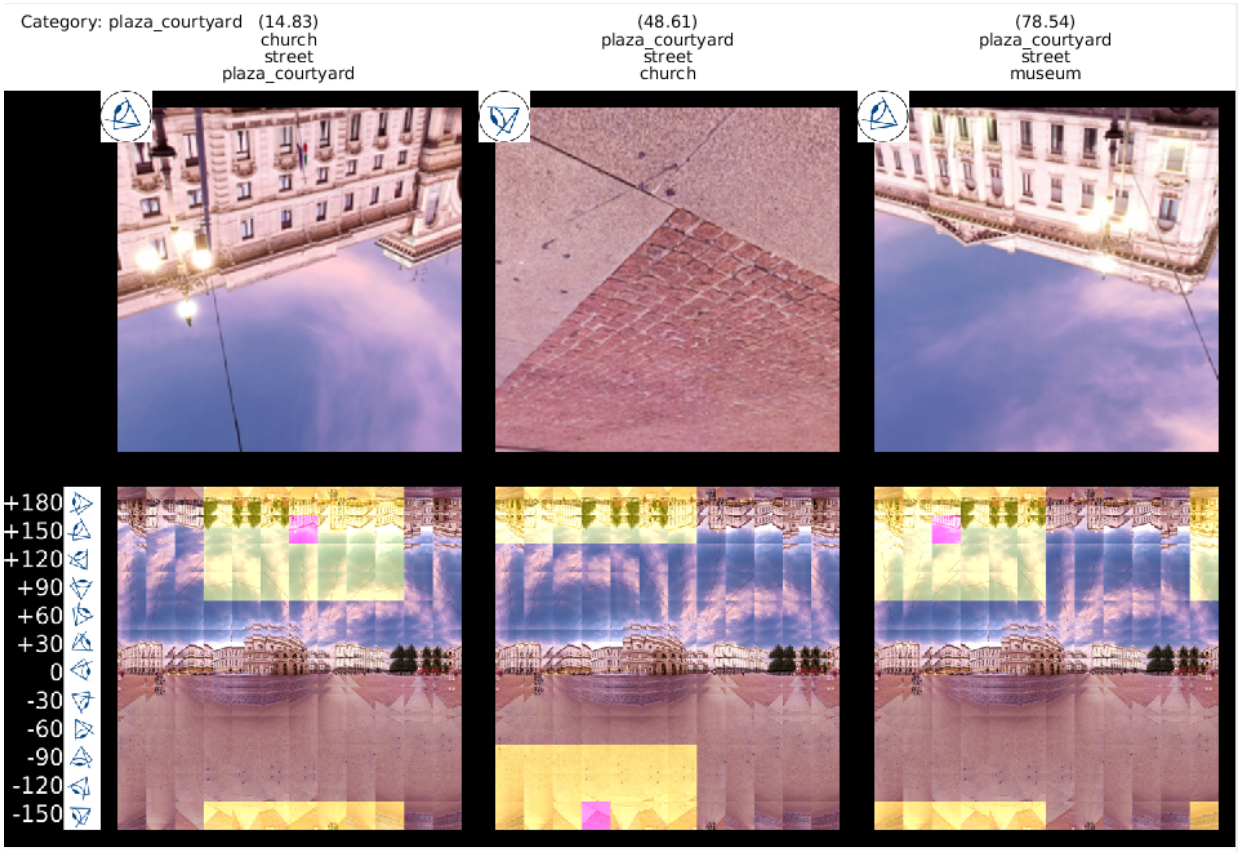

Category: plaza_courtyard
(6.28)
restaurant
train_interior
shop
(11.95)
theater
restaurant
plaza_courtyard
(68.38)
plaza_courtyard
street
theater
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: restaurant
(24.46)
restaurant
theater
museum
(27.58)
expo_showroom
restaurant
theater
(77.06)
restaurant
expo_showroom
theater
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

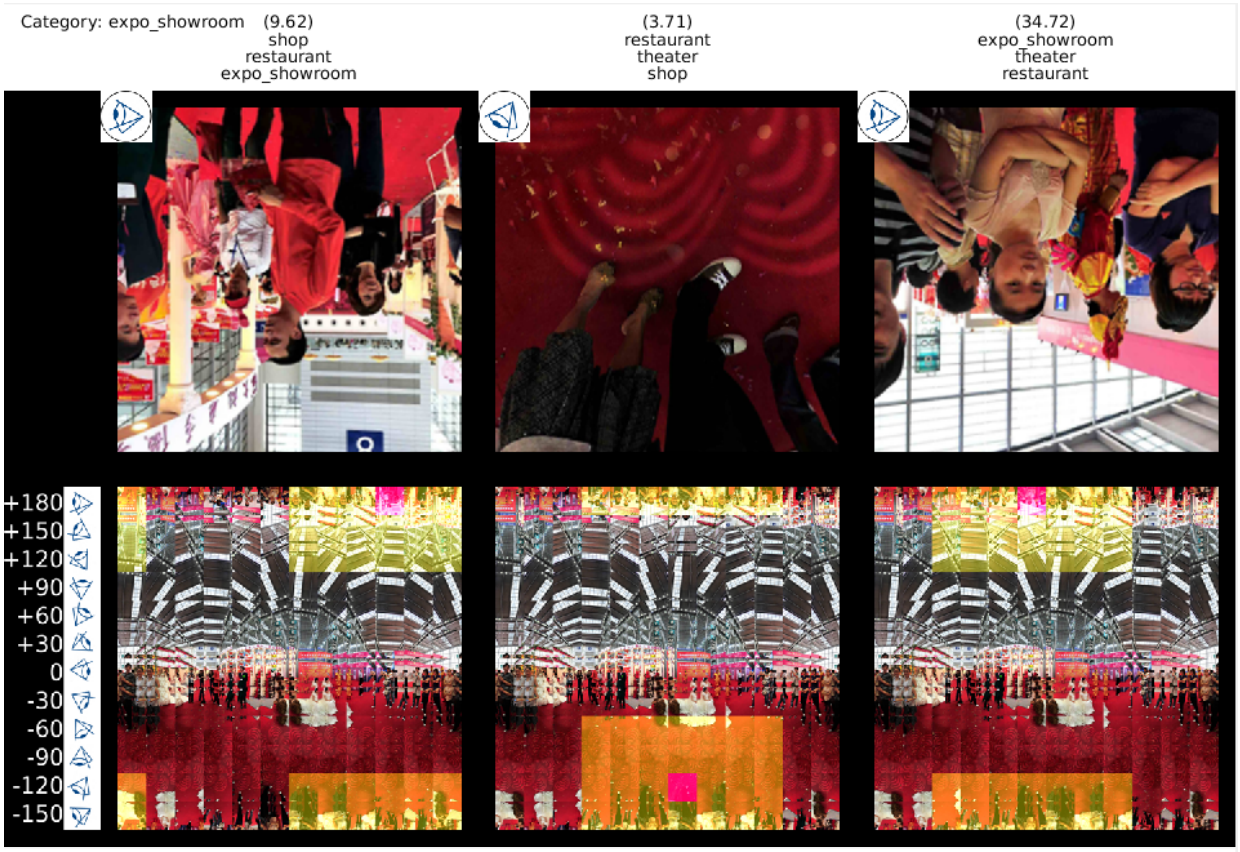

Category: restaurant
(47.37)
restaurant
bedroom
church
(68.14)
restaurant
church
bathroom
(77.90)
restaurant
church
lobby_atrium
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

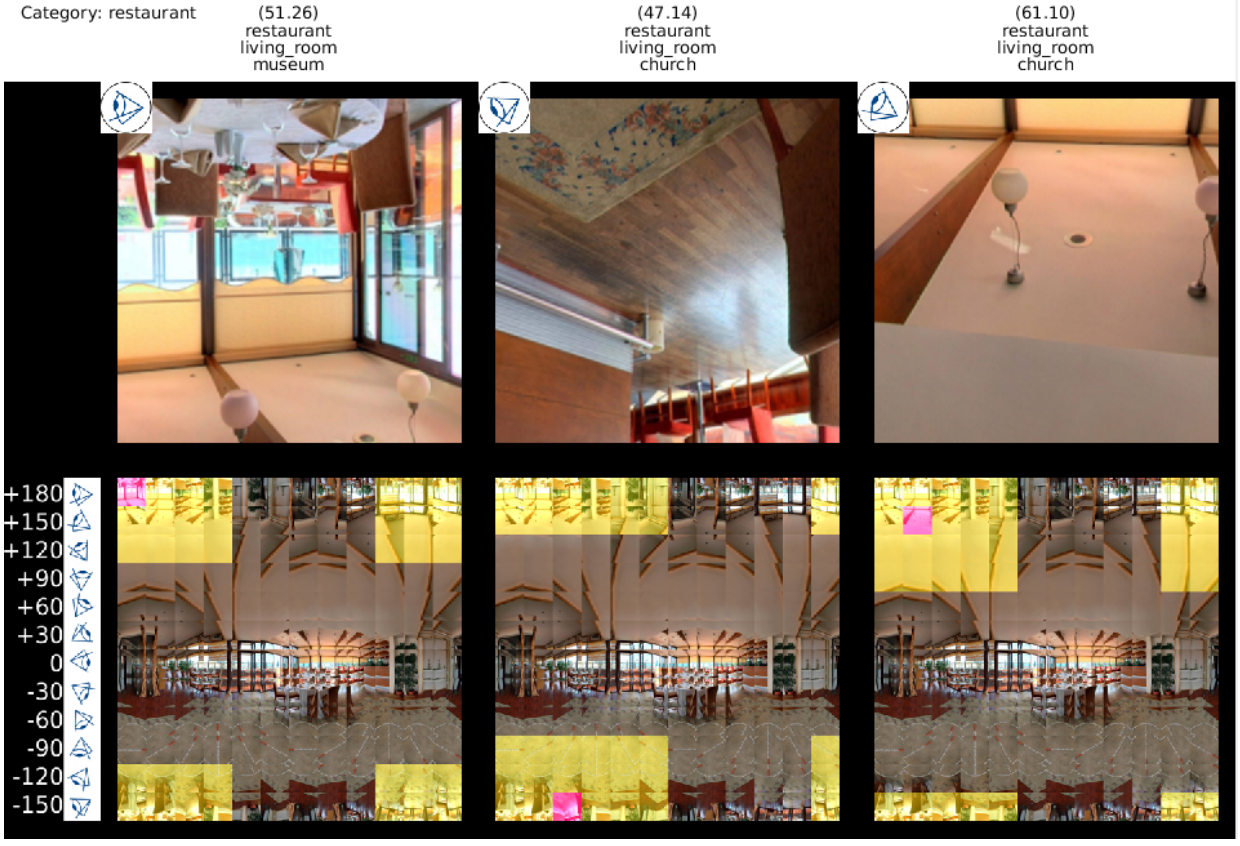

Category: church
(11.81)
restaurant
shop
museum
(58.99)
church
shop
lobby_atrium
(70.98)
church
lobby_atrium
shop
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

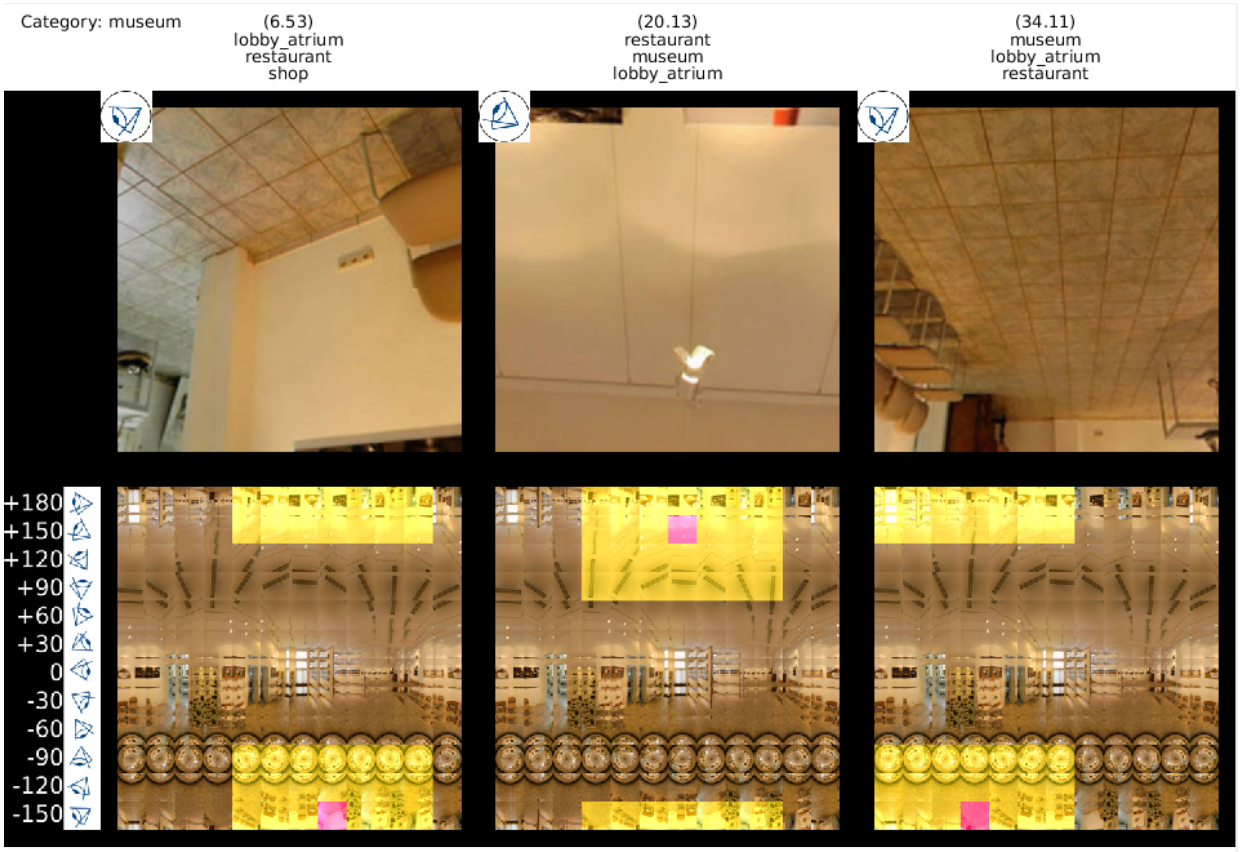

Category: plaza_courtyard
(10.48)
field
beach
plaza_courtyard
(14.12)
wharf
field
plaza_courtyard
(2.60)
wharf
beach
field
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

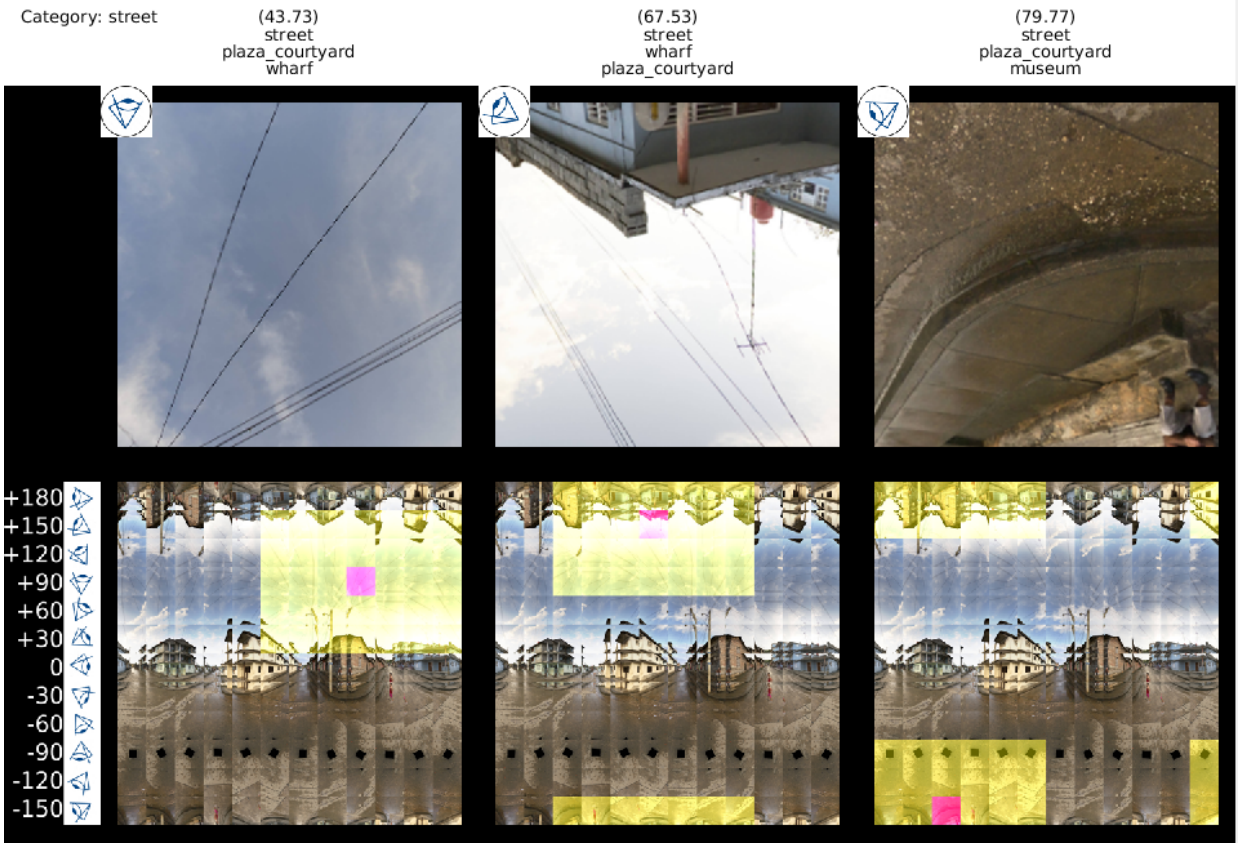

Category: plaza_courtyard (14.83)
church
street
plaza_courtyard
(48.61)
plaza_courtyard
street
church
(78.54)
plaza_courtyard
street
museum
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

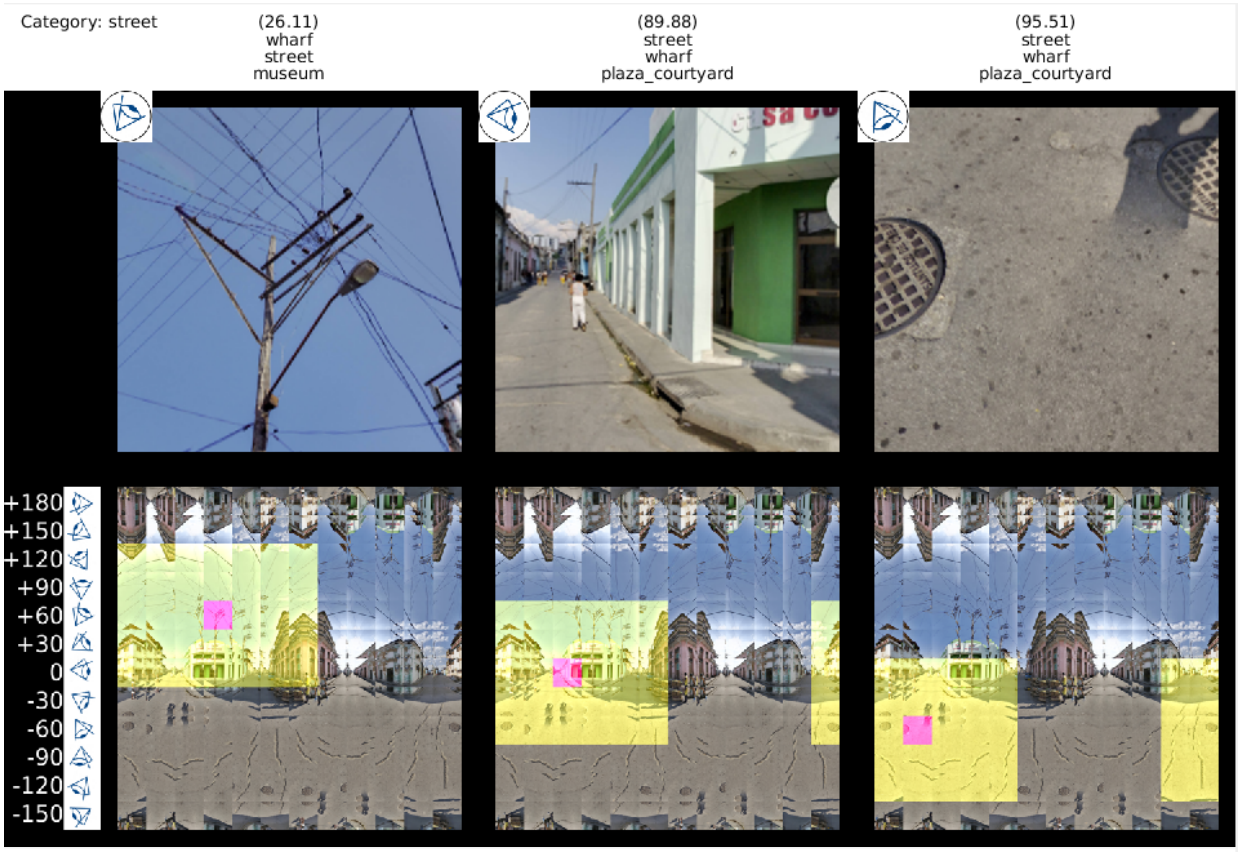

Category: plaza_courtyard (56.54)
plaza_courtyard
lobby_atrium
street
(94.84)
plaza_courtyard
street
lobby_atrium
(71.84)
plaza_courtyard
lobby_atrium
restaurant
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

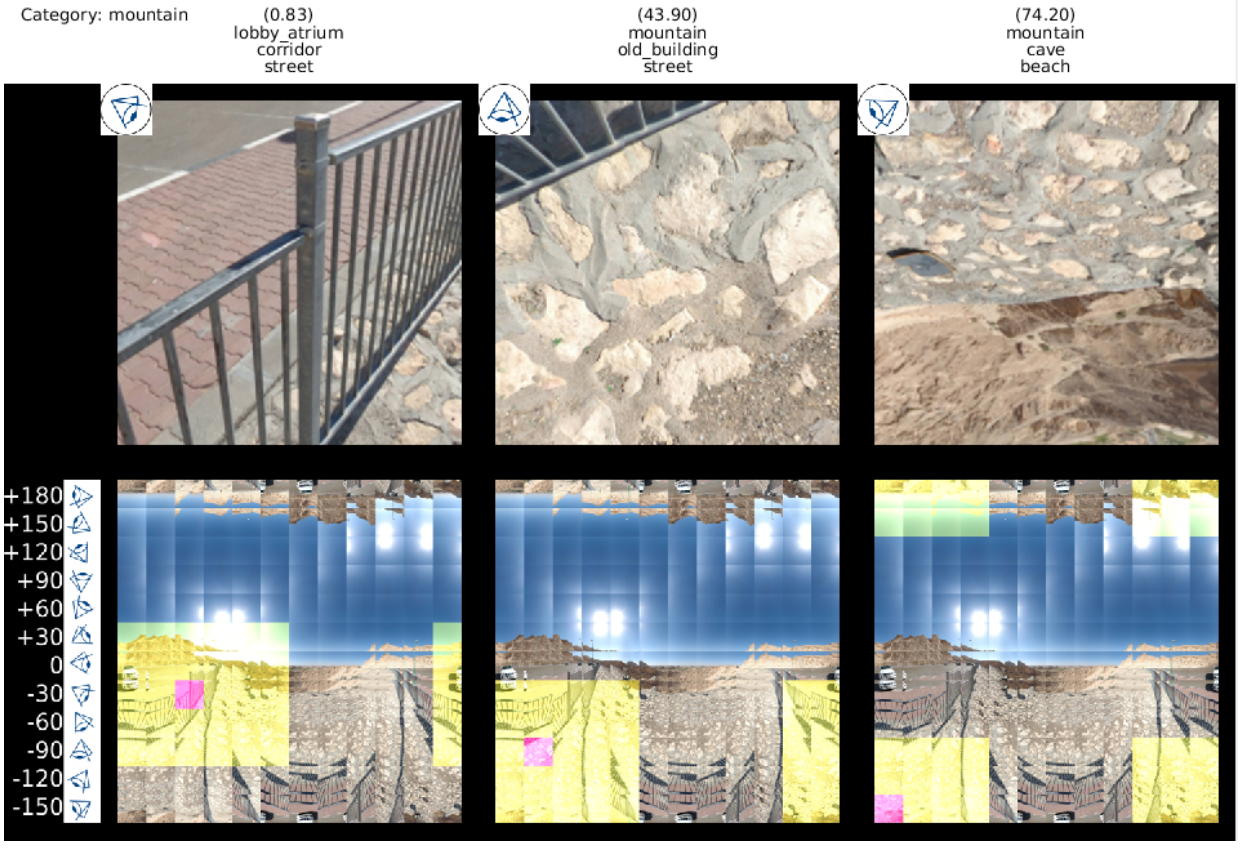

Category: expo_showroom (9.62)
shop
restaurant
expo_showroom
(3.71)
restaurant
theater
shop
(34.72)
expo_showroom
theater
restaurant
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

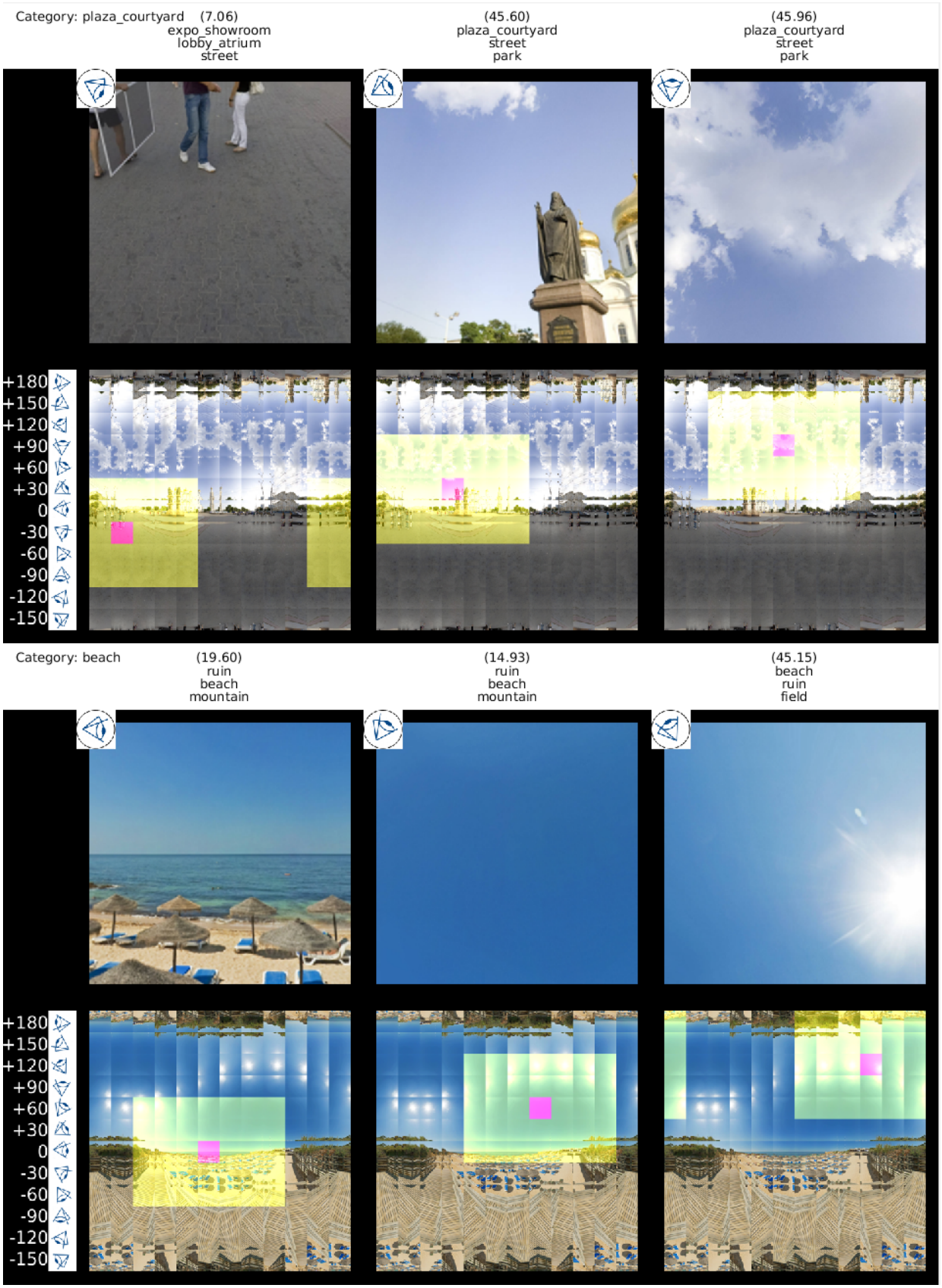

Category: restaurant (51.26)
restaurant
living_room
museum
(47.14)
restaurant
living_room
church
(61.10)
restaurant
living_room
church
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

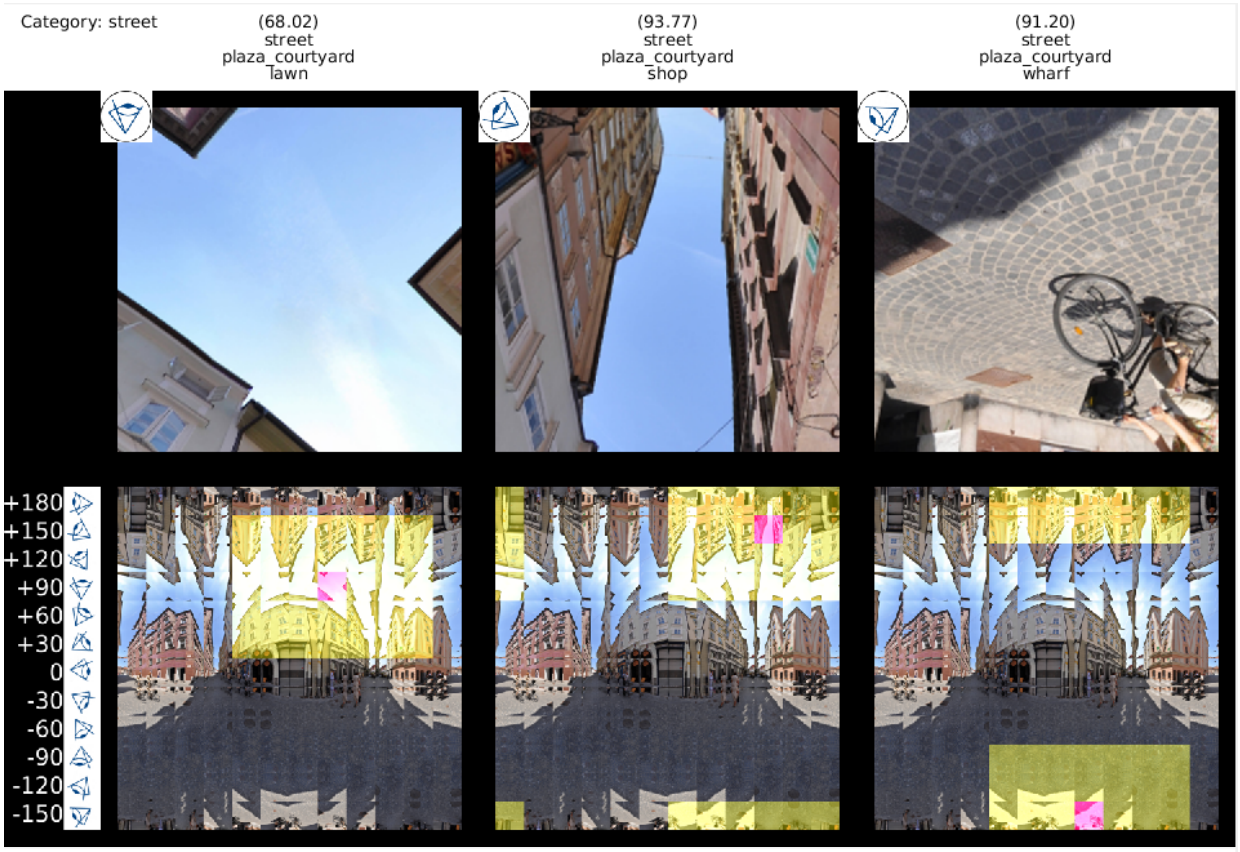

Category: museum
(6.53)
lobby_atrium
restaurant
shop
(20.13)
restaurant
museum
lobby_atrium
(34.11)
museum
lobby_atrium
restaurant
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

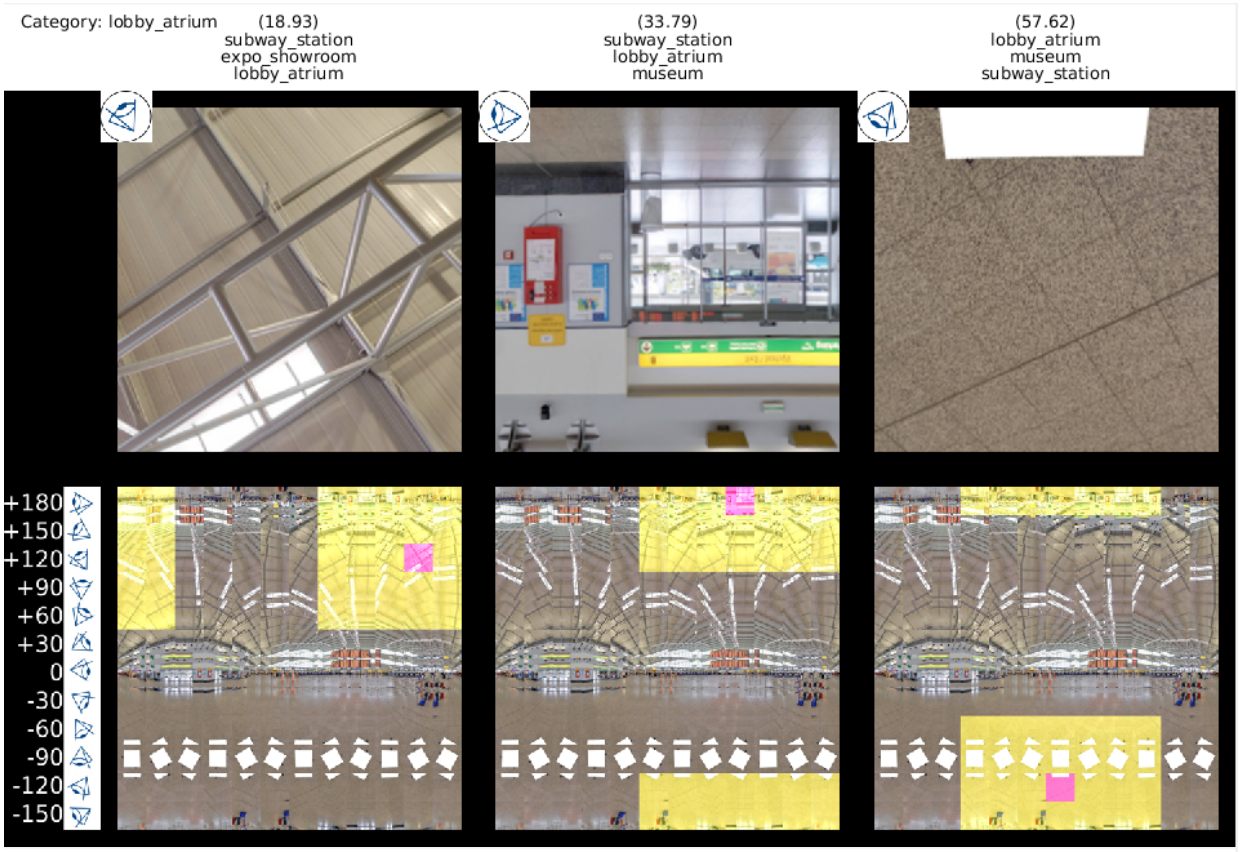

Category: street
(43.73)
street
plaza_courtyard
wharf
(67.53)
street
wharf
plaza_courtyard
(79.77)
street
plaza_courtyard
museum
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

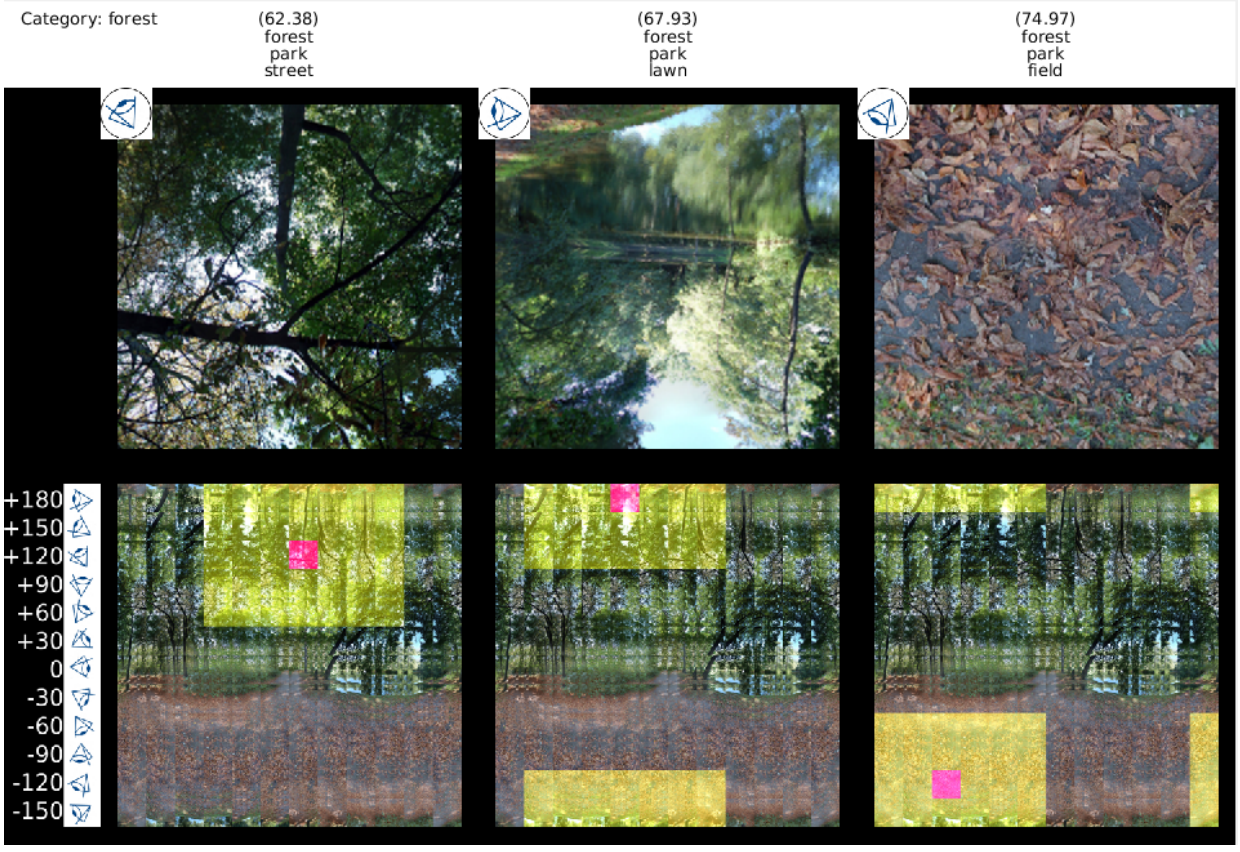

Category: street
(26.11)
wharf
street
museum
(89.88)
street
wharf
plaza_courtyard
(95.51)
street
wharf
plaza_courtyard
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: mountain
(0.83)
lobby_atrium
corridor
street
(43.90)
mountain
old_building
street
(74.20)
mountain
cave
beach
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: plaza_courtyard (7.06)
expo_showroom
lobby_atrium
street
(45.60)
plaza_courtyard
street
park
(45.96)
plaza_courtyard
street
park
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150
Category: beach
(19.60)
ruin
beach
mountain
(14.93)
ruin
beach
mountain
(45.15)
beach
ruin
field
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: street
(68.02)
street
plaza_courtyard
lawn
(93.77)
street
plaza_courtyard
shop
(91.20)
street
plaza_courtyard
wharf
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150
Category: shop
(5.76)
restaurant
church
lobby_atrium
(7.28)
restaurant
street
train_interior
(21.59)
restaurant
shop
street
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: lobby_atrium
(18.93)
subway_station
expo_showroom
lobby_atrium
(33.79)
subway_station
lobby_atrium
museum
(57.62)
lobby_atrium
museum
subway_station
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150


Category: forest
(62.38)
forest
park
street
(67.93)
forest
park
lawn
(74.97)
forest
park
field
+180
+150
+120
+90
+60
+30
0
-30
-60
-90
-120
-150

VIII. Failure cases

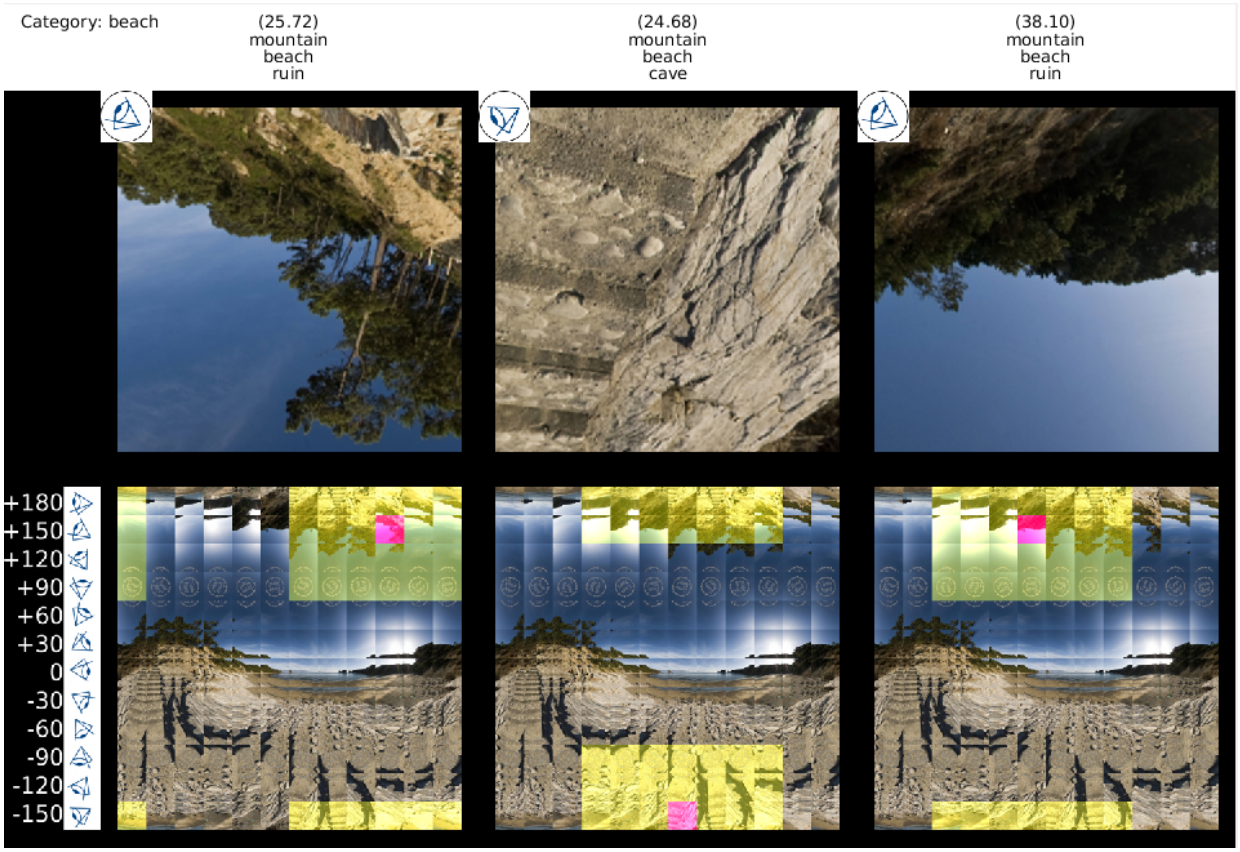


Fig. 10: Failure case: the agent grows increasingly confident about its starting "mountain" guess as the sea and sand are restricted to small portion of the original panorama (near the center of the view grid), and the agent observes neither of these.

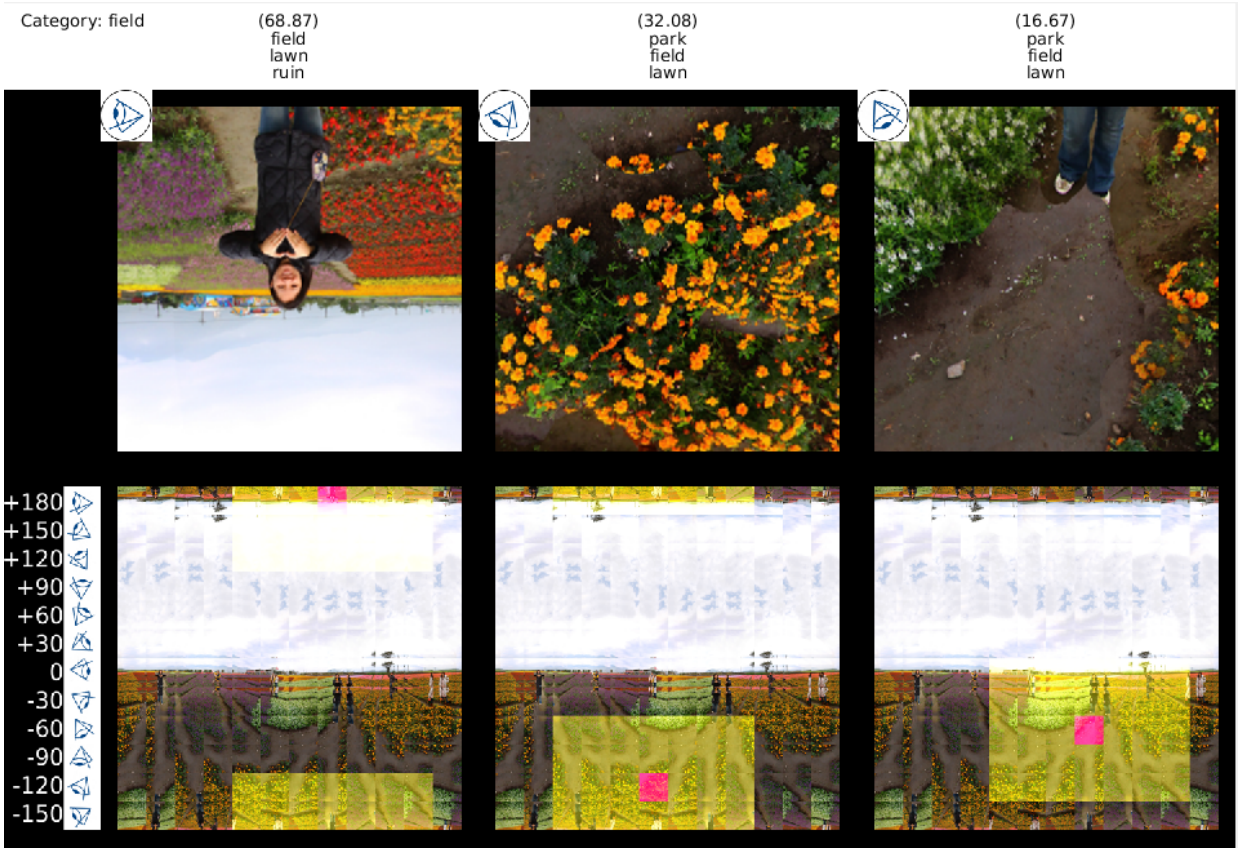


Fig. 11: Failure case: ambiguous categories. This field with multi-colored flowers, neat paths for walking (seen at T=2 and 3), is easily mistaken for a park.

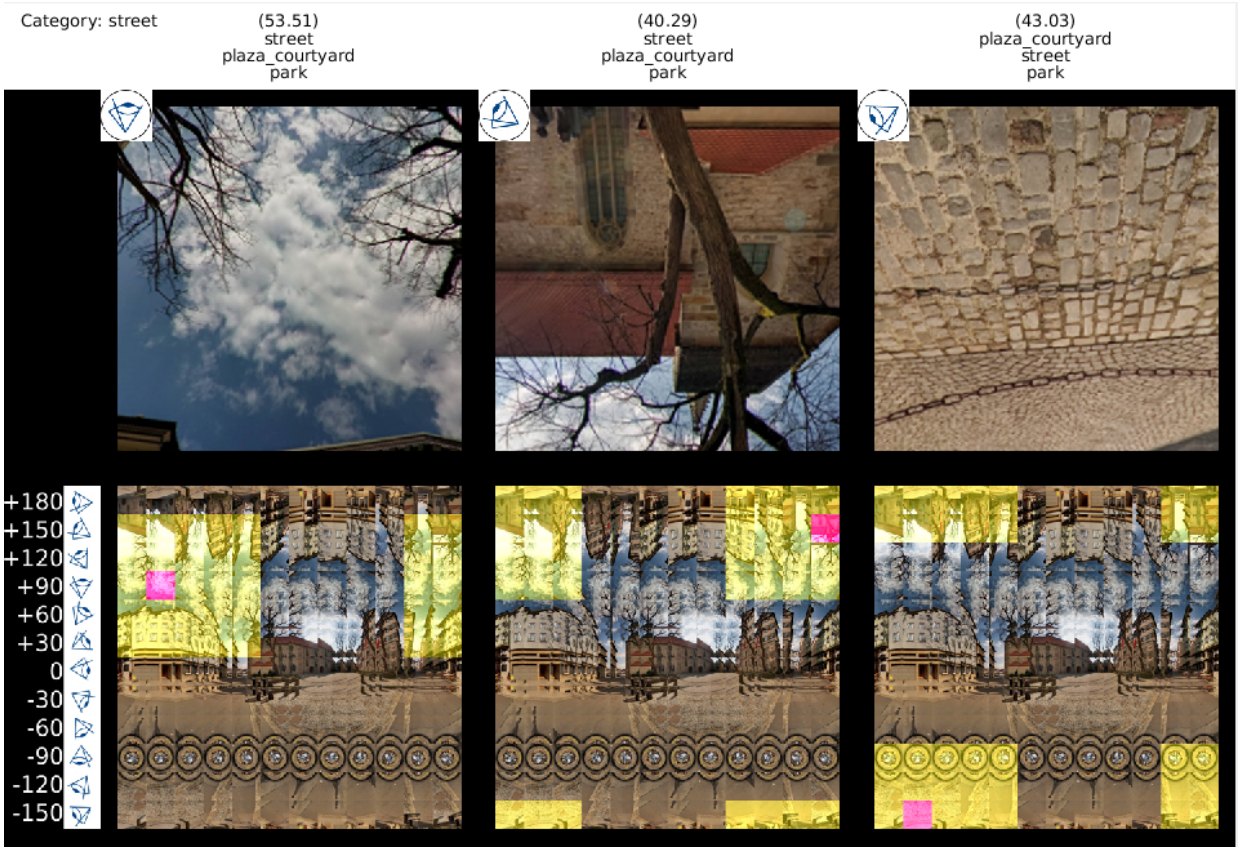


Fig. 12: Failure case: overcorrecting during category disambiguation. The top category after the first two views is "street", but confidence is relatively low (40.29%) and "plaza courtyard" is the top competing category. The system therefore opts to look at the ground where streets typically have tarmac patches, and plazas often have brick pavements. Unfortunately, this European street is brick-paved, and the system over-corrects to guess "plaza courtyard".

IX. Random action selection

To illustrate the difficulty of the action selection task, we show views selected by a random action selection baseline, next to those selected by our method, starting from the same starting view.

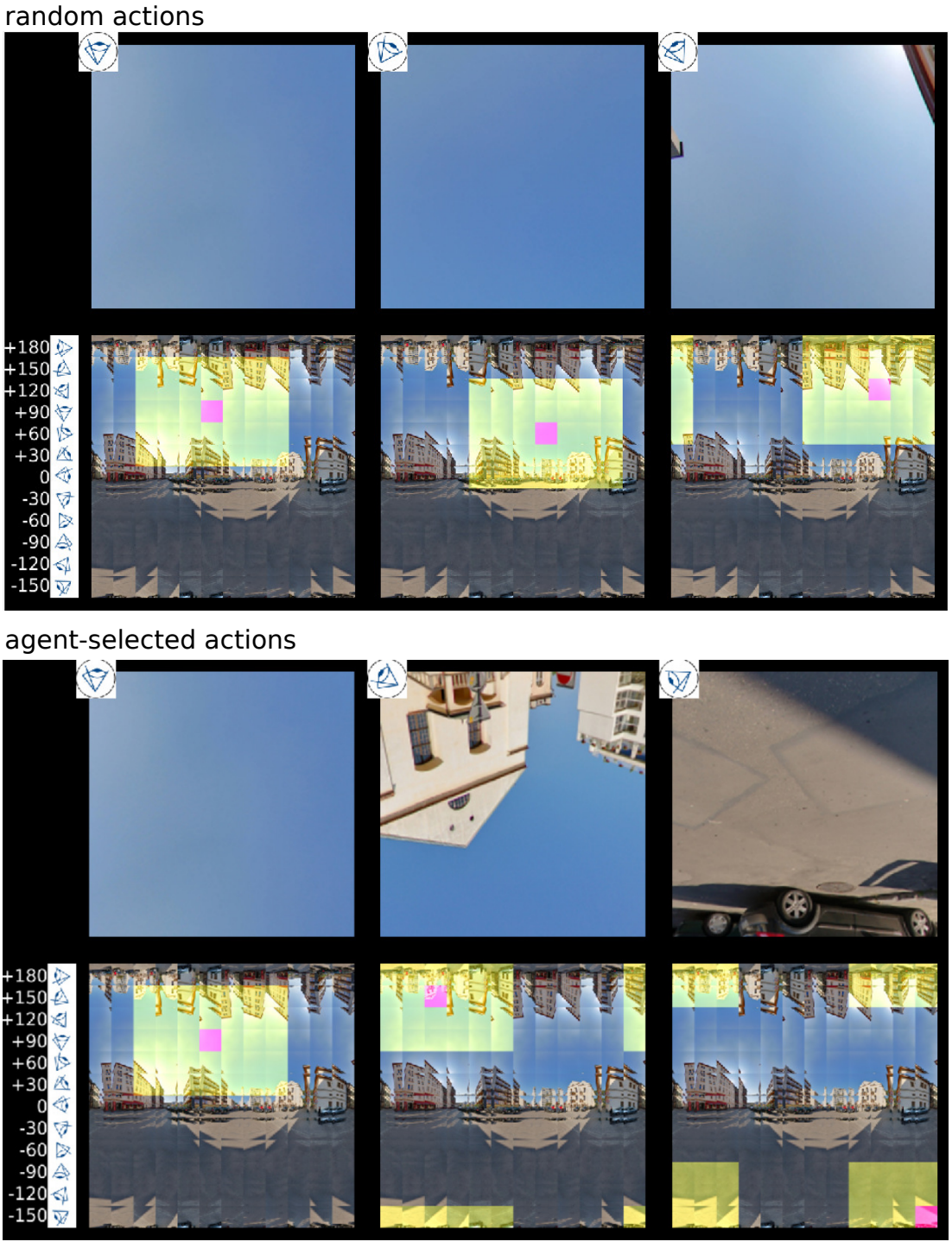


Fig. 13: The random agent selects all uninformative sky views, while the our agent selects disambiguating views

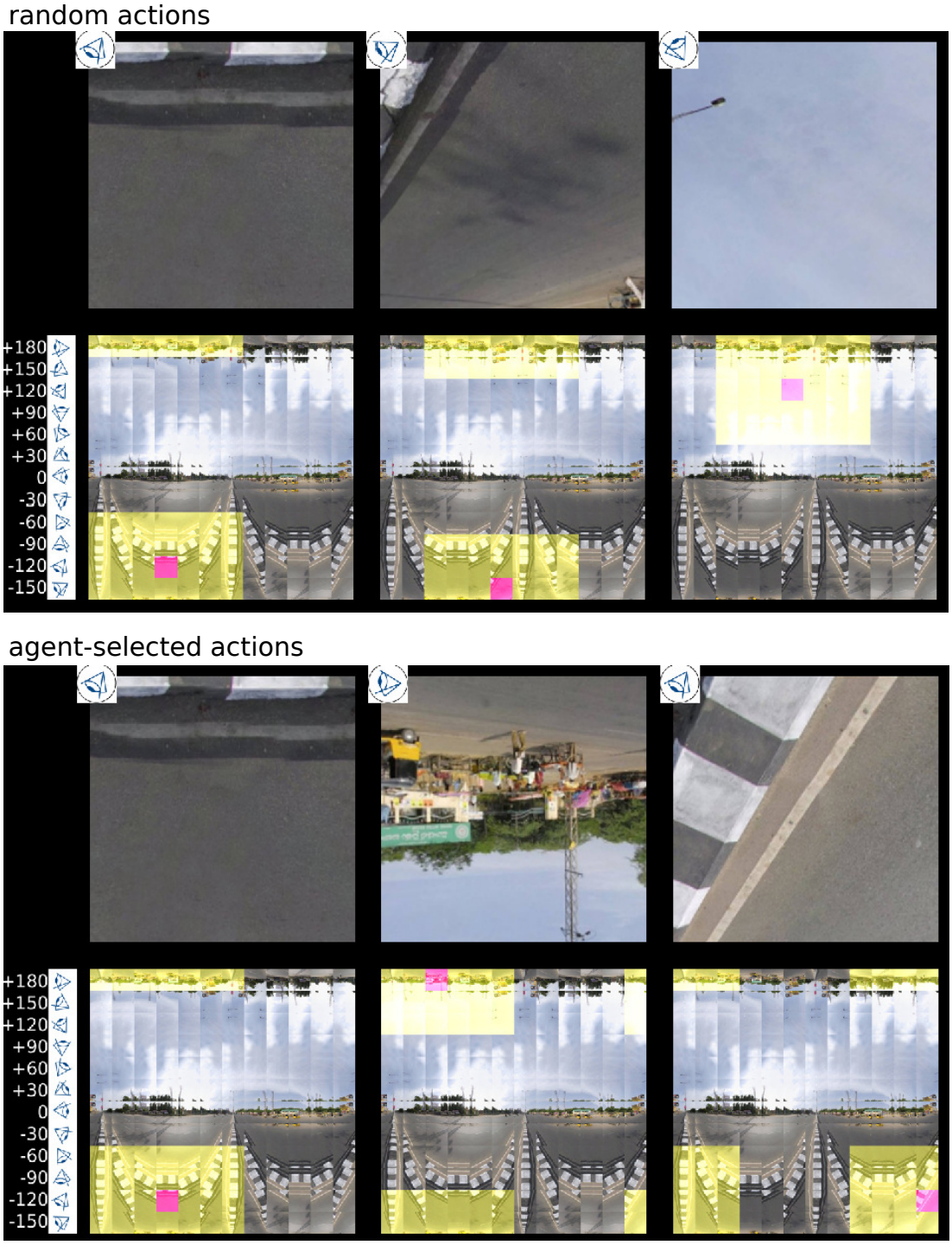


Fig. 14: The random agent views uninformative ground and sky views, while our agent starting at the same view, selects an informative view of the street, and then reviews the pavement curb from a more favorable viewpoint.