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

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018

Dinesh Jayaraman, Kristen Grauman

Presenter: Shenghui Chen (Vivian)
Sept 30, 2021

The challenge: learning to look around

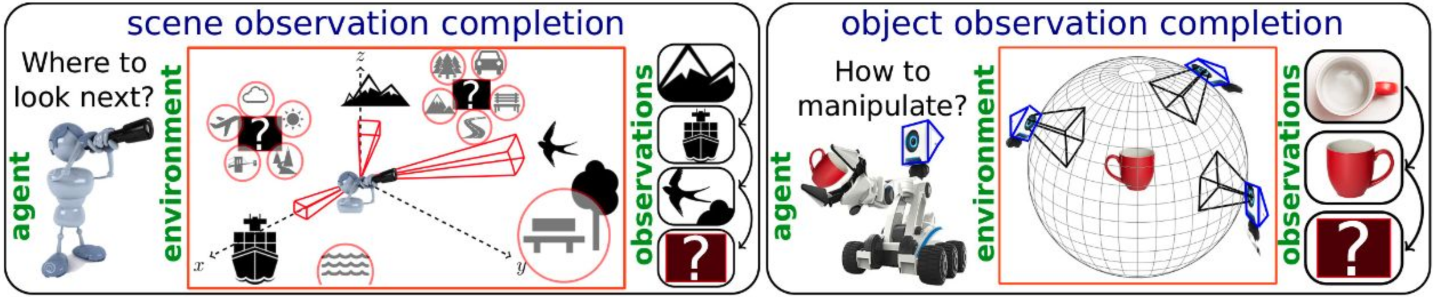
Problem:

How to autonomously capture good observations?

Why important?

- fixed observation → passive perception algorithm running on disembodied stationary machines
- embodied agent → active exploration and perception

Motivating tasks:



Proposed work

Motivation: infants' abilities to actively manipulate and inspect objects correlate with learning to complete 3D shapes [Soska et al. 2010]

Unsupervised learning based on **Active observation completion**

- small set of observations → all other possible observations
- the agent continuously updates its internal model of a target, and choose actions that lead to new views that will efficiently complete the internal model

Advantages:

- generality → policy can transfer to unseen tasks and environments
- low cost (label-free) data

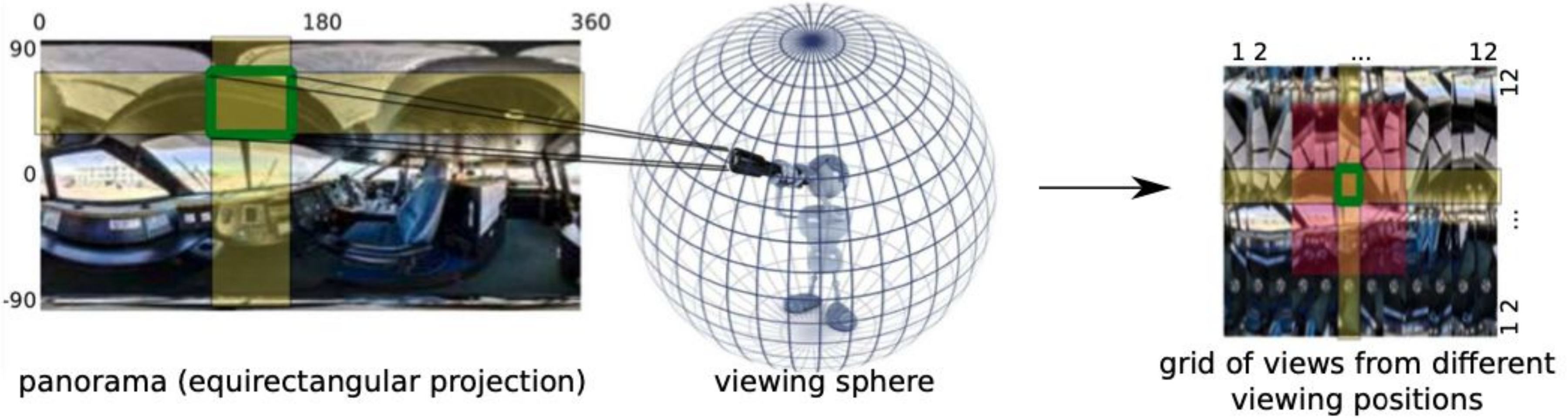
Related work (and their limitations)

- **Saliency and attention**: find most salient regions of already captured image/video data; predict the gaze of human observer. Access to observation of the entire environment → to look for a new observation.
- **Optimal sensor placement**: how to place sensors so that they provide maximum coverage. Sensors are static → Active completion, reacting to past observations.
- **Active perception**: active object localization, action detection in video, object recognition. Pre-defined recognition tasks → general data acquisition strategy in perception; manually labeled data → unlabeled observations.

Related work (and their limitations)

- **Active visual localization and mapping**: to limit samples needed to densely reconstruct a 3D environment geometrically. Purely geometric methods require dense observations → infer missing content with semantic and contextual clues.
- **Learning to reconstruct**: one-short reconstruction. Single view → sequence of views; image feature learning → learn action policies.

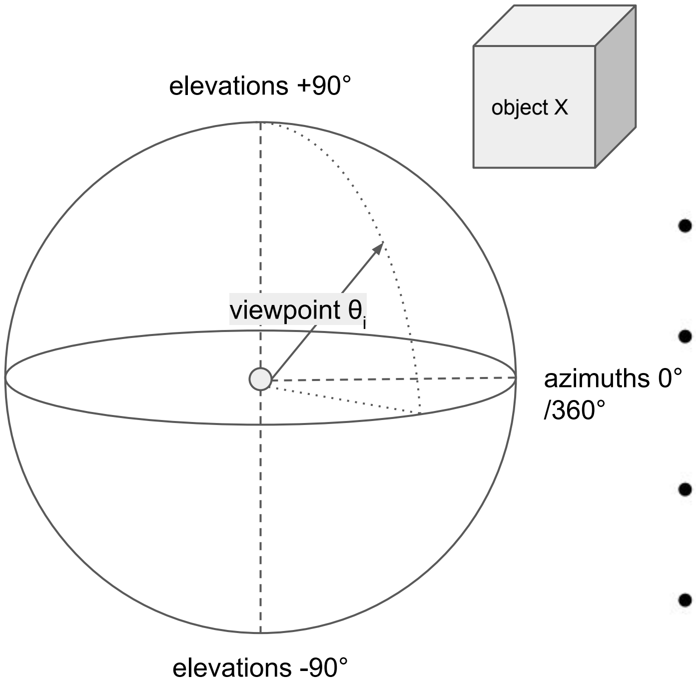
Problem setting



[Jayaraman et al. 2016]

elevations -90°

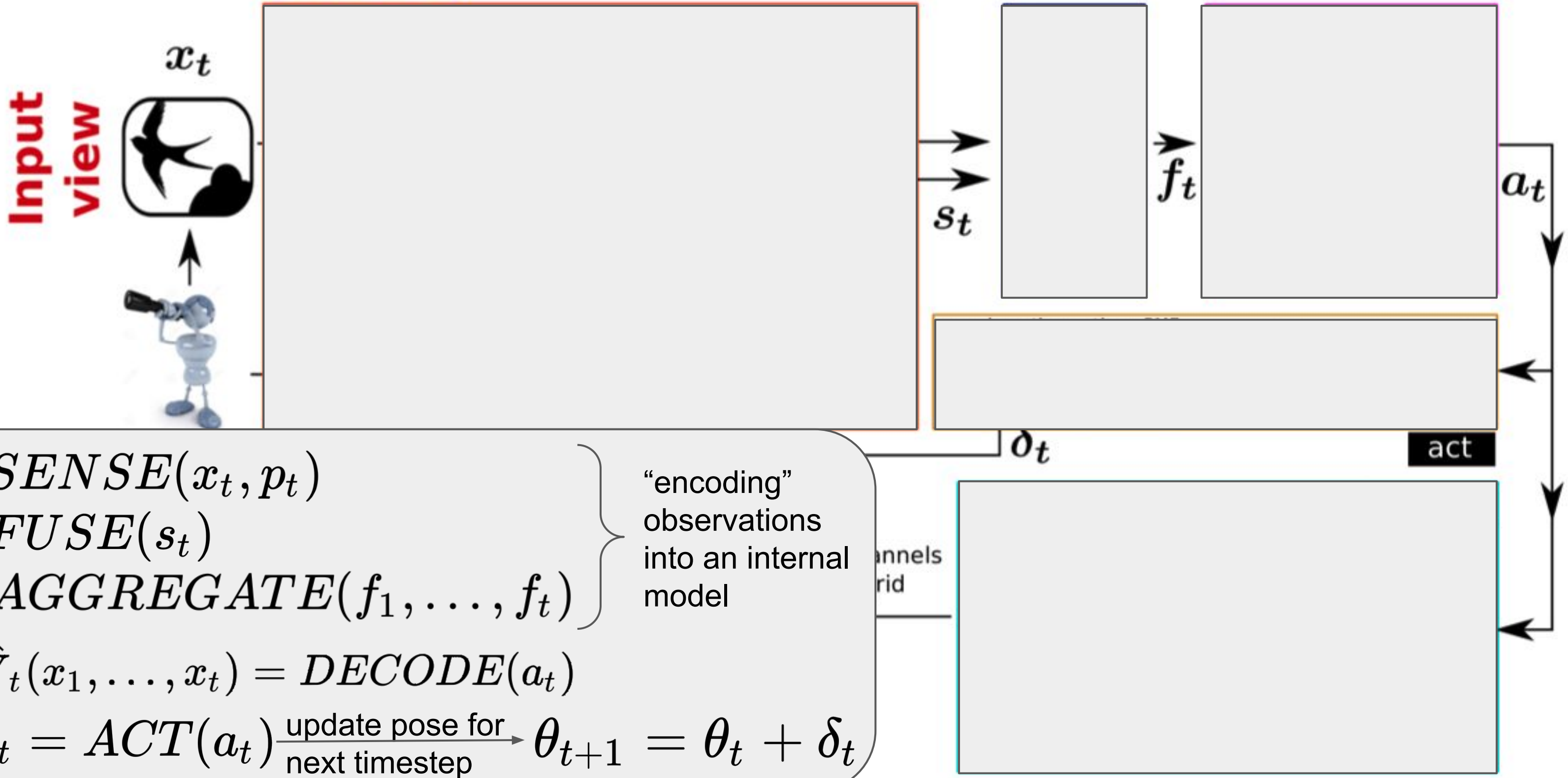
Problem setting



Object observation completion problem:

- At timestep $t = 1$, an active agent is presented with an object X in a random, unknown pose
- At every timestep, it can perform one action to rotate the object, make new observation x_t, then update its prediction for the viewgrid $\hat{V}_t(x_1, ..., x_t)$
- After $T << MN$ timesteps, it should have learned a model that can produce a view of the object as seen from any new viewing angle
- (Assume the agent is aware of the relative motion from the previous view, let p_t denote this proprioceptive metadata (elevation, relative motion))

Active observation completion framework



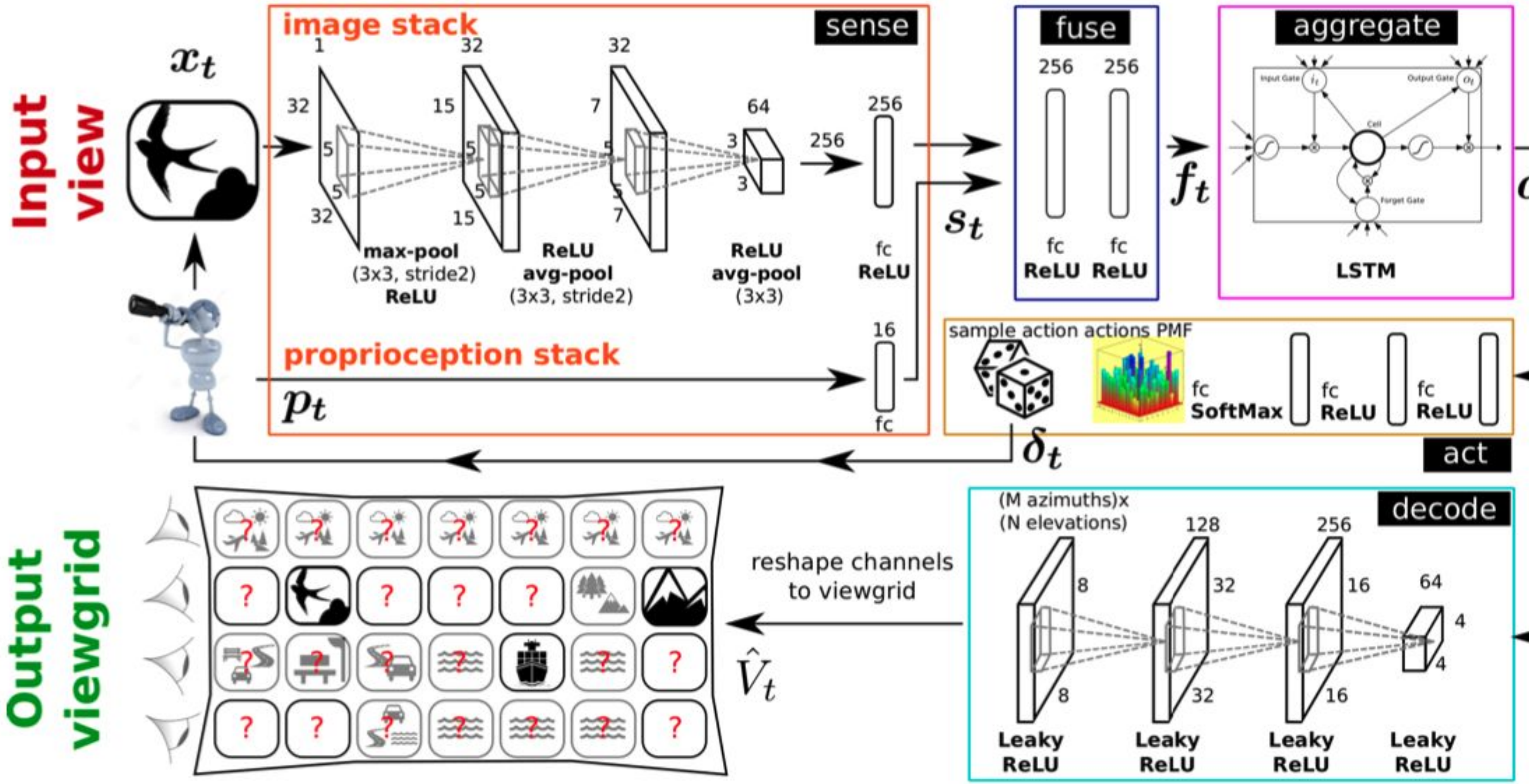
$$s_t = SENSE(x_t, p_t)$$
$$f_t = FUSE(s_t)$$
$$a_t = AGGREGATE(f_1, \ldots, f_t)$$

"encoding" observations into an internal model

$$\hat{V}_t(x_1, \ldots, x_t) = DECODE(a_t)$$

$$\delta_t = ACT(a_t) \xrightarrow[\text{next timestep}]{\text{update pose for}} \theta_{t+1} = \theta_t + \delta_t$$

Active observation completion framework



Objective function and model optimization

- to minimize the distance between predicted and target views at the same viewpoint at time T

$$L_T(X) = \sum_i d(\hat{x}_T(X, \theta_i), x(X, \theta_i))$$

Output viewgrids are shifted by an angle $\triangle_0$ from the target viewgrid

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

- To minimize this loss, employ stochastic gradient descent (BPTT) + REINFORCE [Williams 1992]

- Specifically,

  $\nabla L_T(X)$ backpropagated via the DECODE, AGGREGATE, FUSE, SENSE modules

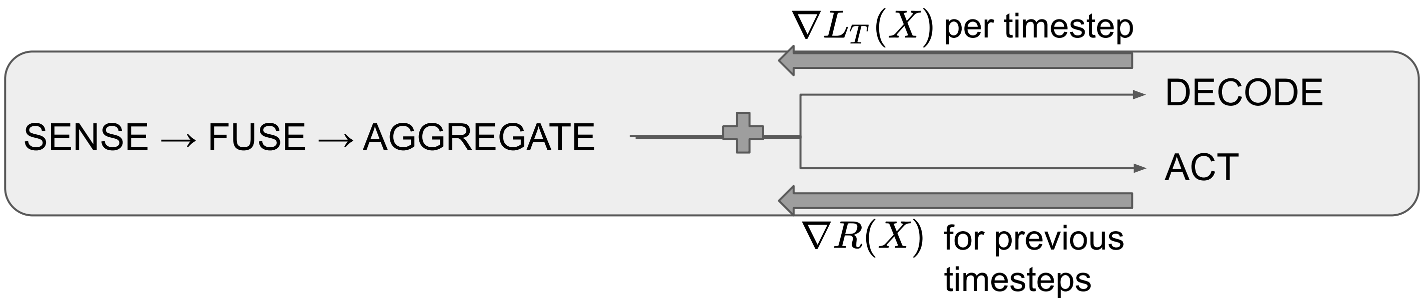
- ACT is stochastic as it involves sampling → use REINFORCE to handle this:

  $R(X) = -L_T(X)$ applied to outputs of ACT at all timesteps, backpropagating to encourage ACT behaviors that led to high R(X) (i.e. low loss);

  $\nabla R(X)$ at timestep t from ACT passed to AGGREGATE for timestep t-1

- In practice, it is beneficial to penalize errors in the predicted viewgrid at every timestep rather than just at t=T

$$L(X) = \sum_{t=1}^{T} L_T(X)$$



Experiments

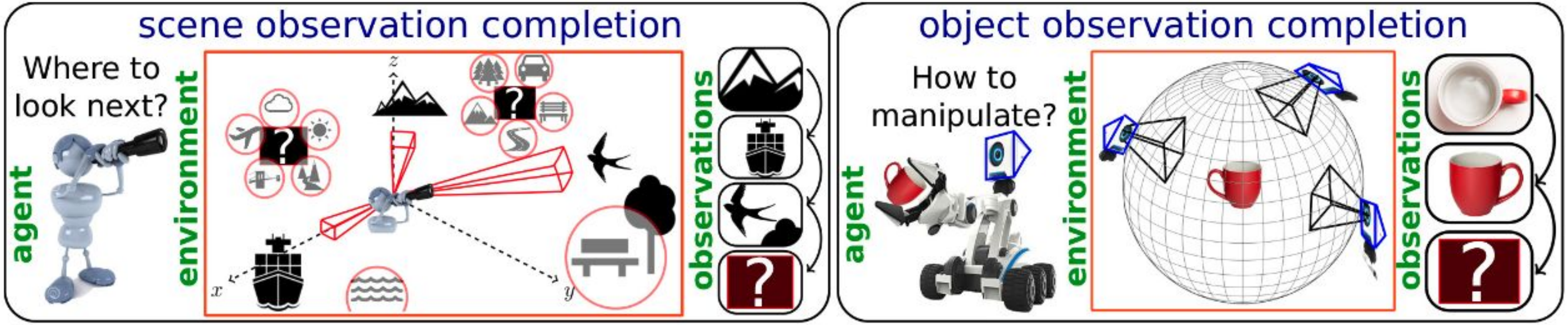
Active Observation Completion (scene, object)

⇒ to show look-around policy performs better on motivating tasks

Transfer to unseen tasks

⇒ to show the advantages of the approach: generality and low-cost

Active Observation Completion



SUN360

45° agent to complete an omnidirectional scene

ModelNet

Agent manipulates a 3D object to complete model of the object

Experimental Setup: Active observation completion

Datasets:

SUN360 (scene)	ModelNet (object)
● 26 category ● 32x32 views from 5 camera elevations and 8 azimuths ● per-timestep motions within 3x5 ● Training episode length T = 6	● Train on seen (ModelNet-40 \ ModelNet-10); unseen (ModelNet-10) ● 32x32 views from 7 camera elevations and 12 azimuths ● per-timestep motions within 5x5 ● Training episode length T = 4

Baselines: *ours* compared with

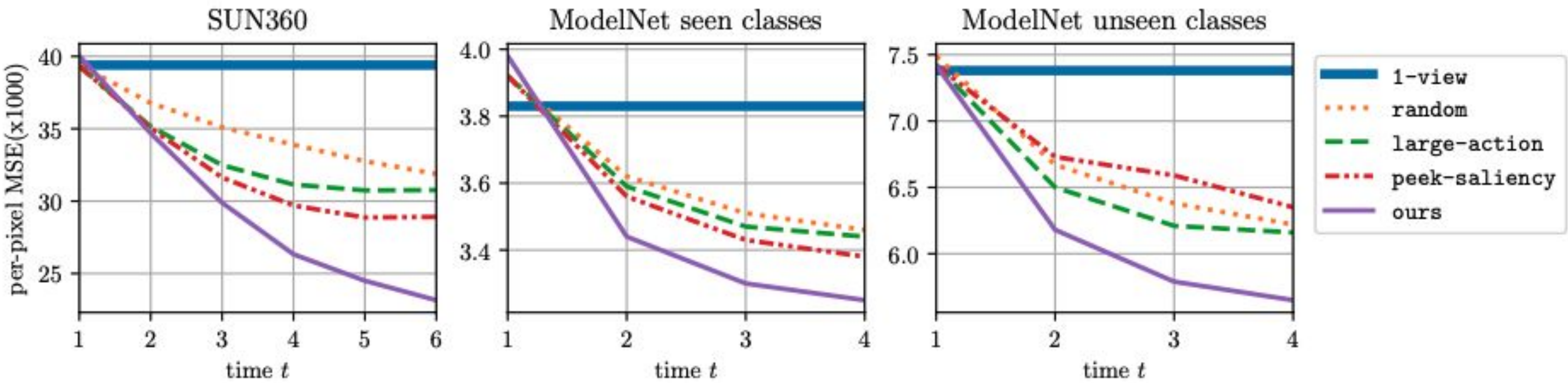
- *1-view*: the method trained with T=1
- *random*: the method with randomly action selection module
- *large-action*: largest allowable action
- *peek-saliency*: most salient view within reach at each timestep

Experimental Results: Active observation completion

Dataset→	SUN360		ModelNet (seen classes)		ModelNet (unseen classes)	
Method↓ — Metric→	MSE(x1000)	Improvement	MSE(x1000)	Improvement	MSE(x1000)	Improvement
1-view	39.40	-	3.83	-	7.38	-
random	31.88	19.09%	3.46	9.66%	6.22	15.72%
large-action	30.76	21.93%	3.44	10.18%	6.16	16.53%
peek-saliency [23]	27.00	31.47%	3.47	9.40%	6.35	13.96%
ours	**23.16**	**41.22%**	**3.25**	**15.14%**	**5.65**	**23.44%**

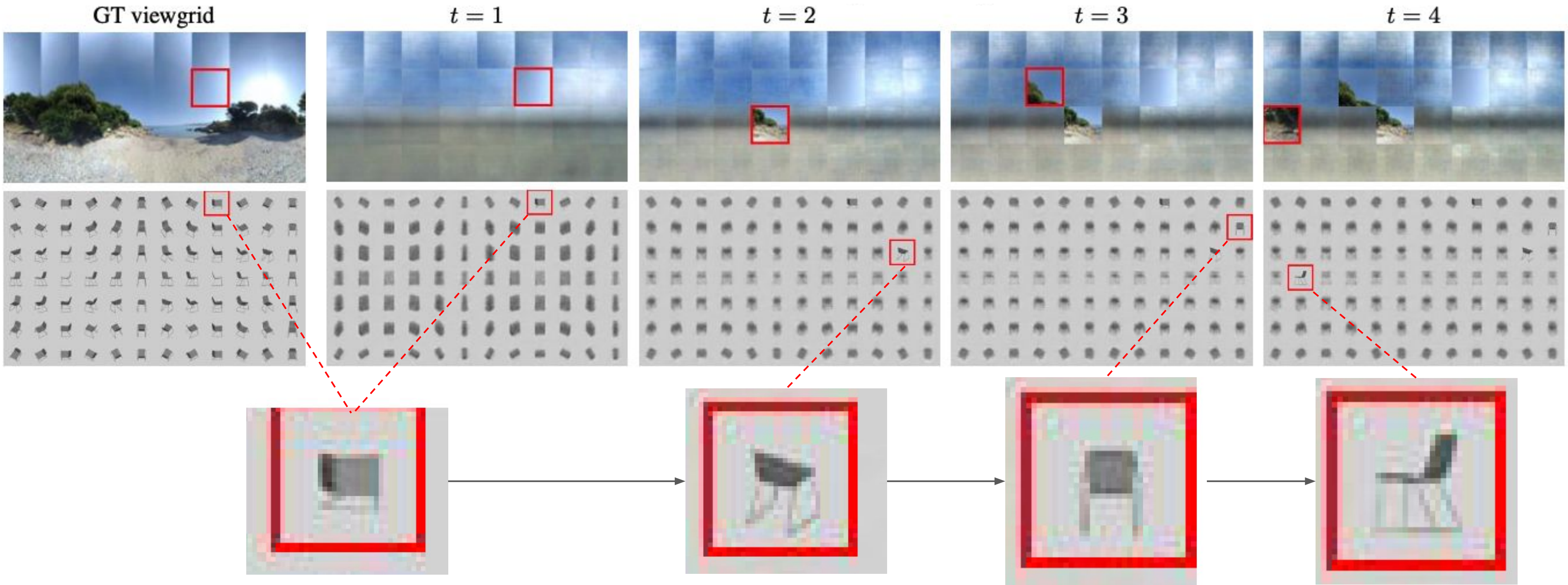
- lower error
- higher improvement
- improvements larger on more difficult datasets (SUN360 > unseen ModelNet > seen ModelNet)

Experimental Results: Active observation completion



- *ours* method has the sharpest MSE drop

SUN360 scene observation completion examples



Transfer to unseen tasks

Objective: to inject the generic look-around policy into unseen tasks in unseen environments.

Active Categorization system [Jayaraman et al. 2016]:

Approach:

s_t

"model A": an end-to-end model using random policies

zero overlap, different categories

"model B": an active observation completion model

a_t

labels from correct target label set

Experimental Setup: Unsupervised policy transfer

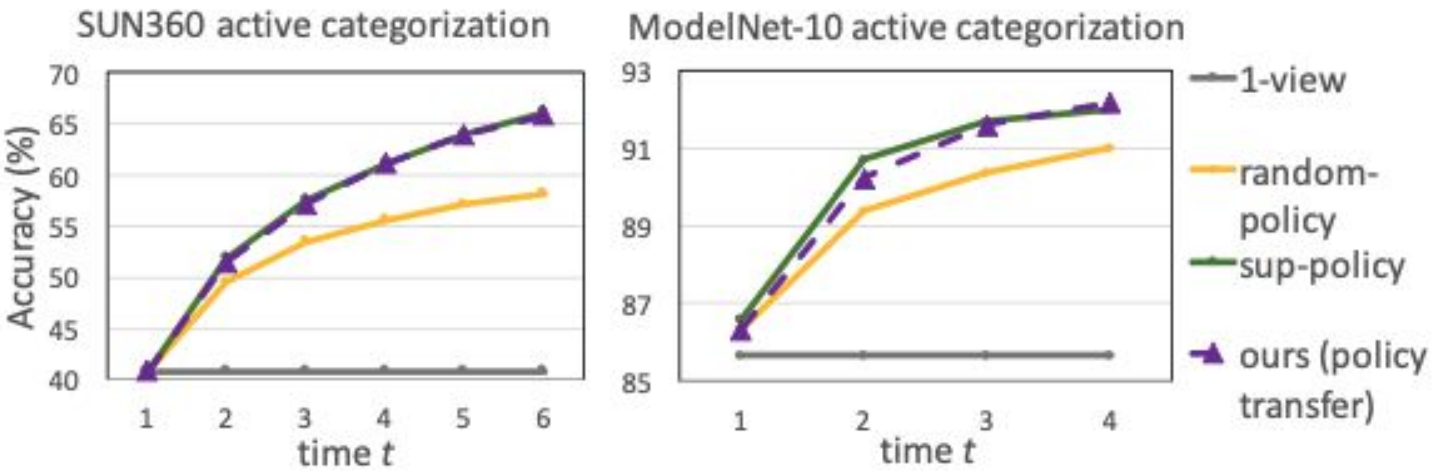
Datasets:

	SUN360 (scene)	ModelNet (object)
“model A” (*random-policy*)	● SUN360 training data	● ModelNet-10 training objects
“model B”	● SUN360 training data	● ModelNet-30 training objects

Baselines: *ours* compared with

- *sup-policy*: “Lookahead active RNN” [Jayaraman et al. 2016]
- *1-view*: passive feed-forward neural network which only processes one randomly presented view and predicts its category
- *random-policy*: *sup-policy* with random legal motions

Experimental Results: Unsupervised policy transfer



- *ours* outperforms *1-view* and *random-policy*, on par with *sup-policy*
  - remarkable b/c *ours* only trained for the separate, unsupervised active observation completion task

Discussion of Results

Active observation completion

- look-around policy decreases error and boosts improvement
- results hold even for unseen ModelNet → advantage of task-independence

Unsupervised policy transfer

- Transferred policy achieves good accuracy → the potential of unsupervised exploratory tasks to facilitate policy learning on massive unlabeled datasets → advantage of generality and low-cost

Limitation and Future work

Limitation & followed-up work

- Train faster and converge to better policies?
  - Sidekick policy learning [Ramakrishnan et al. 2018]
- Geometry awareness (cross-object occlusion)?
  - Geometry-aware RNN [Cheng et al. 2018]
- Inference about occluded region?
  - Occupancy anticipation [Ramakrishnan et al. 2020]
- Viewing sphere in more refined discretization (Large M,N)?

Future work

- Fine-tune policy when transferred for new tasks
- Look around → move around?
  - reconstruction-based exploratory policies [54,61]
  - learning to move to perceive [Yang et al. 2019]

Extended Readings

- Yang, J., Ren, Z., Xu, M., Chen, X., Crandall, D., Parikh, D., & Batra, D. (2019). Embodied amodal recognition: Learning to move to perceive objects. Proceedings of the IEEE International Conference on Computer Vision, 2019-October, 2040–2050. https://doi.org/10.1109/ICCV.2019.00213
- Ramakrishnan, S. K., Al-Halah, Z., & Grauman, K. (2020). Occupancy Anticipation for Efficient Exploration and Navigation. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12350 LNCS, 400–418. https://doi.org/10.1007/978-3-030-58558-7_24
- Ramakrishnan, S. K., Jayaraman, D., & Grauman, K. (2021). An Exploration of Embodied Visual Exploration. International Journal of Computer Vision, 129(5), 1616–1649. https://doi.org/10.1007/s11263-021-01437-z
- Cheng, R., Wang, Z., & Fragkiadaki, K. (2018). Geometry-aware recurrent neural networks for active visual recognition. Advances in Neural Information Processing Systems, 2018-Decem(Nips), 5081–5091. https://arxiv.org/pdf/1811.01292.pdf
- Ramakrishnan, S. K., & Grauman, K. (2018). Sidekick policy learning for active visual exploration. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11216 LNCS, 424–442. https://doi.org/10.1007/978-3-030-01258-8_26

Summary

- **Problem:** How can a visual agent autonomously capture good observations?
- **Why important?** Crucial step towards embodied, active agents in novel environments
- **key limitation:** lack of geometry awareness, coarse viewing sphere discretization
- **Advantages:** generality, low-cost
- **key insights:** the agent is rewarded for actions that reduce its uncertainty about the unobserved portions of the environment
- **What did they demonstrate by this insight?**
  - SOTA performance on active observation completion tasks
  - first to accomplish “policy transfer” between tasks