A Contextual Bandit Approach for Learning to Plan in Environments with Probabilistic Goal Configurations

Sohan Rudra*
Google Research

Saksham Goel*
Google Search

Anirban Santara*
Google Research

Claudio Gentile*
Google Research

Laurent Perron
Google Research

Fei Xia
Robotics@Google

Vikas Sindhwani
Robotics@Google

Carolina Parada
Robotics@Google

Gaurav Aggarwal
Google Research

Abstract

Object-goal navigation (Object-nav) entails searching, recognizing and navigating to a target object. Object-nav has been extensively studied by the Embodied-AI community, but most solutions are often restricted to considering static objects (e.g., television, fridge, etc.). We propose a modular framework for object-nav that is able to efficiently search indoor environments for not just static objects but also movable objects (e.g. fruits, glasses, phones, etc.) that frequently change their positions due to human intervention. Our contextual-bandit agent efficiently explores the environment by showing optimism in the face of uncertainty and learns a model of the likelihood of spotting different objects from each navigable location. The likelihoods are used as rewards in a weighted minimum latency solver to deduce a trajectory for the robot. We evaluate our algorithms in two simulated environments and a real-world setting, to demonstrate high sample efficiency and reliability.

1 Introduction

In this paper, we aim to achieve robust Object-Goal Navigation (object-nav) [15, 63] in indoor human-centric environments. Object-nav is a core component of many personal robotics applications like Mobile Manipulation [3], Embodied Question Answering [21], and Vision-and-Language Navigation [5]. It is defined as the task of searching (and optionally, retrieving) a given object within a designated space. Our algorithms are modular and map-based. They assume access to a binary 2D occupancy map of the environment with navigable and non-navigable parts marked out. Analogous to how humans search, we propose an algorithm that learns to look for objects that change their locations due to human interaction (e.g. cellphones, glasses and keys) in the most likely places first. Appendix A contains a brief review of different kinds of object-nav algorithms. A prominent challenge faced by learning-based algorithms in robotics is bridging the sim-to-real gap [35]. Learning agents are usually trained in a simulator (sim) like Matterport [13], AI2Thor [38], Gibson [62] and Habitat [52] before deploying in the real world. However, due to limited fidelity of a simulator, observations of the same event might be different in the simulator and in reality (domain mismatch). The modular design of our approach allows us to switch out object detectors, motion planners and point samplers for domain-specific models when our agent is transferred between sim and real – minimizing the effect of domain mismatch. Figure 1 provides an overview of our approach.

---

*equal contribution

NeurIPS 2022 Workshop on Robot Learning: Trustworthy Robotics, Virtual, Virtual

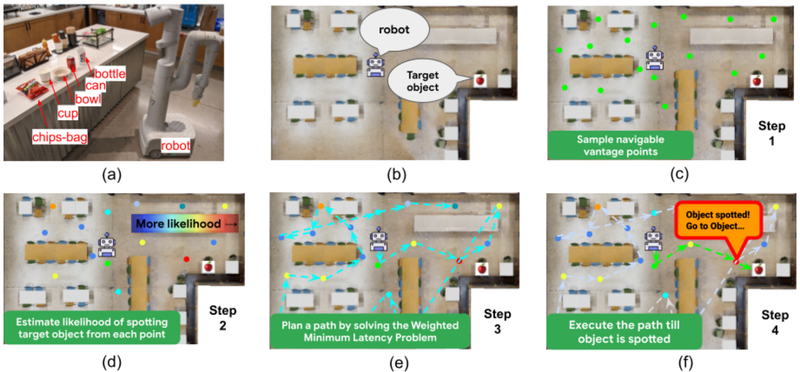

bottle
can
bowl
cup
chips-bag
robot
robot
Target object
Sample navigable vantage points
Step 1
More likelihood
Estimate likelihood of spotting target object from each point
Step 2
Plan a path by solving the Weighted Minimum Latency Problem
Step 3
Object spotted! Go to Object...
Execute the path till object is spotted
Step 4
(a)
(b)
(c)
(d)
(e)
(f)

Figure 1: Overview of our approach. (a) Picture of our robot and the target objects studied in our experiment. (b) The robot is randomly initialized in the environment with the task of finding a given target object. (c) A set of reachable vantage points (green dots) are sampled across the entire environment using the current $2D$ occupancy map via farthest point sub-sampling [24, 48]. (d) A Contextual Bandit Agent [41] estimates the likelihood of spotting the target object from each vantage point. (e) A Weighted Minimum Latency Problem (WMLP) solver [60] is used to generate an ordering of the vantage points taking into account their likelihood scores, the initial position of the robot and the geometry of the room. (f) The robot visits the vantage points in the planned order while inspecting its surroundings. As soon as it spots the object, it heads directly to it.

2 Proposed Methodology

Our "formal" algorithm is described as Algorithm 1. A practical implementation of this algorithm begins by **sampling** a sparse set of vantage points that are navigable and spread all over the scene in a way that the agent's vision is able to cover the space to a large extent by visiting these points and looking around. First, we extract all the navigable points (at a resolution of 0.1 meter) from the robot's occupancy map. Next, we select our vantage points using the Farthest Point Sub-sampling (FPS) algorithm [24, 48]. The second step is to **estimate the importance of each point** in the context of the target object. An important vantage point is one that has a high likelihood of the robot spotting an instance of the target object class within its visibility range r_{vis} standing at that point. In order to estimate the likelihood of the robot spotting an object (which can be movable) from a given point, we need to explore the environment. For efficient exploration, we formulate the problem as a contextual bandit with each vantage point as an arm and follow the principle of *optimism in the face of uncertainty* (e.g., [56]). The final step is deriving the sequence of visitation of the vantage points. We build a graph where each node is a vantage point and each edge contains the $A^\star$ distance (shortest collision-free path length) between the vantage points it connects. The visitation-sequence is obtained by solving the Weighted Minimum Latency Problem (WMLP) [7, 60] that minimizes the average waiting time of each node, weighted by its importance score. In our case, this minimizes the average distance traveled by the robot to reach the target object – the average being over the position of the object and the initial position of the robot. A detailed theoretical analysis of our formulation based on Algorithm 1 is presented in Appendix B. We show that the cumulative regret is sub-linear in the number of vantage points with high probability. We experiment with **generalized linear** (as in Algorithm 1) and simple two-layer fully connected neural network (with 100 hidden nodes) models for estimating the importance of points in our contextual bandit agent. Neural networks are approximated via Neural Tangent Kernels (NTK) [36], as contained in Algorithms 1-2 in [64] for computing uncertainties $\epsilon_t(x)$ (Step 2 in Algorithm 1) and updating $\widehat{\theta}$ (Step 5 in Algorithm 1).

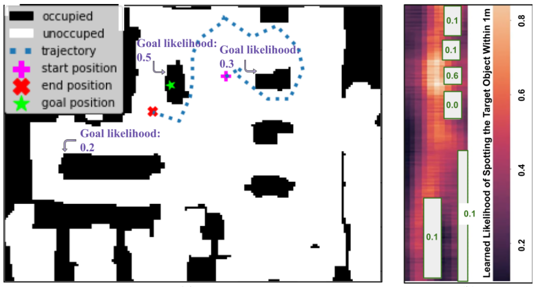
3 Experiments

We run experiments in two simulated and one real office kitchen environments. The two simulated kitchens have areas 80 sq.m. and 120 sq.m., respectively. Each experiment involves training the agent for 200 episodes followed by evaluation with frozen parameters in the same environment. For the real kitchen experiment, we train the bandit model on a snapshot of the map in our simulator and evaluate in the real environment.

The simulated environments have photo-realistic scenes generated from Matterport scans [14] and Bullet [20] based physics simulation. Our robot is a differential-drive wheeled robot, which has a 3D LiDAR in the front, and depth sensors mounted on its head. It is capable of accurate localization and

Algorithm 1 Simplified contextual bandit planning algorithm.

```
Input: Learning rate $\eta > 0$, exploration parameter $\alpha \geq 0$, $\epsilon$-cover $\mathcal{F}_\epsilon$ of the set of feasible points $\mathcal{F}$, $\epsilon > 0$, path length $k$.
Init: $M_0 = kI \in \mathbb{R}^{D \times D}$, $\widehat{\theta}_1 = 0 \in \mathbb{R}^D$, $c_1 = 1$.
For $t = 1, 2, \ldots, T$
    1. Get object identity $i_t$ , and initial position of the robot $x_{0,t}$ ;
    2. For $x \in \mathcal{F}$, set
           $\widehat{\Delta}_t(x) = \widehat{\theta}_{c_t}^\top \phi(i_t, x)$   and   $\epsilon_t^2(x) = \alpha\, \phi(i_t, x)^\top M_{c_t-1}^{-1} \phi(i_t, x)$
    3. Compute $J_t = \langle x_{\pi_t(1),t}, \ldots, x_{\pi_t(k),t} \rangle$ as          //solve WMLP at episode $t$
           $J_t = \arg \min_{\substack{x_1 \ldots x_k \in \mathcal{F}_\epsilon \\ \text{permutation } \pi}} \sum_{\ell=1}^{k} \sigma\Big(\widehat{\Delta}_t(x_{\pi(\ell)}) - \epsilon_t(x_{\pi(\ell)})\Big) \sum_{j=1}^{\ell} \text{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})$
    4. Observe signal $\begin{cases} \langle s_{1,t}, \ldots, s_{k'_t,t} \rangle = \langle -1, \ldots, -1, +1 \rangle & \text{set } m_t = k'_t \\ \text{or} & \\ \langle s_{1,t}, \ldots, s_{k,t} \rangle = \langle -1, \ldots, -1, -1 \rangle & \text{set } m_t = 0 \end{cases}$
    5. For $j = 1, \ldots, m_t$ (in the order of occurrence of items $x_j$ in $J_t$) update:
           $M_{c_t+j-1} = M_{c_t+j-2} + \phi(i_t, x_j)\phi(i_t, x_j)^\top$,
           $\widehat{\theta}_{c_t+j} = \widehat{\theta}_{c_t+j-1} + \eta\, \sigma\Big(-s_{j,t}\, \widehat{\theta}_{c_t+j-1}^\top \phi(i_t, x_j)\Big)\, s_{j,t}\, M_{c_t+j-1}^{-1} \phi(i_t, x_j)$
    6. $c_{t+1} \leftarrow c_t + m_t$ .
```



(a) (b)

Figure 2: **(a)** Real-kitchen sample trajectory for CP-SAT planner with the Neural model. **(b)** Heat map of the estimated likelihood of spotting the goal object ("bottle") within a distance of 1m along with ground truth likelihoods (in green) of the object occurring on the surface of each furniture.

safe point-to-point navigation. We consider two different ways of solving the WMLP in our setting – a) directly solving the optimization problem using CP-SAT, a satisfiability (SAT)-based constraint programming (CP) solver [47, 54] from Google OR-Tools [31, 32]; and b) a one-step greedy approach that maximizes a weighted combination of the estimated likelihood of spotting the target object and the inverse of the $A^\star$ distance traveled to reach it – to choose the next vantage point in the visitation sequence. Although myopic, the latter is a faster approximation of the CP-SAT algorithm that despite being direct and more principled, suffers from drastic increases in running time with growing number of vantage points (See Figure E.1). We use Model Predictive Control [26, 11] to execute a path. Each vantage point x is described by a feature-vector $\phi(i, x)$ consisting of a one-hot encoding of the target object i and a flattened 16×16 patch of the wall-distance map centered at the point x (see also Figure D.3 in Appendix D). A sinusoidal positional encoding [58] vector is appended to represent the location of the point in the map. Map-resolution and positional encoding dimension are hyper-parameters. The normalizer for the feature-vector is also a hyper-parameter that is chosen from among: a) zero mean, unit standard-deviation, and b) unit l^2-norm. All hyper-parameters (η and α for Algorithm 1, α_p for One-step greedy, the learning rate and batch size in Algorithms 1-2 in [64] for Neural, the positional embededing size and the feature vector normalization for mapping $\phi(i, x)$, and the sigmoidal scale s for the sigmoid in Algorithm 1) are tuned across suitable ranges (Table E.5)

	Real Kitchen Env.			**Kitchen Environment 1**							**Kitchen Environment 2**						
	TSP	**Neural**		**TSP**	**Gen-Lin**		**Neural**		**GT-Scores**		**TSP**	**Gen-Lin**		**Neural**		**GT-Scores**	
		Greedy	**CP-SAT**		**Greedy**	**CP-SAT**	**Greedy**	**CP-SAT**	**Greedy**	**CP-SAT**		**Greedy**	**CP-SAT**	**Greedy**	**CP-SAT**	**Greedy**	**CP-SAT**
Train SPL	-	-	-	-	0.32	0.31	0.30	0.27	-	-	-	0.26	0.27	0.23	0.13	-	-
Eval. Succ.	0.88	0.80	0.92	0.89	0.88	0.89	0.90	0.90	0.90	0.90	0.56	0.80	0.78	0.74	0.67	0.82	0.80
Eval. SPL	0.37	0.38	0.42	0.38	0.42	0.47	0.40	0.39	0.43	0.51	0.22	0.26	0.29	0.25	0.20	0.33	0.34

Table 1: Empirical evaluation of our agents on 3 environments in terms of object-nav metrics.

using a Gaussian-Process Bandit based Blackbox optimizer [30] to maximize **success weighted by path length (SPL)** [6] over the training episodes (Appendix E).

For training in simulation, we consider the agent has successfully reached its goal if and when it visits a vantage point that is within r_{vis} radius from the target object. For learning good quality likelihood maps, we set $r_{\text{vis}} = 1\text{m}$ during training although the default value of $r_{\text{vis}} = 2.5\text{m}$ is used during evaluation in the simulated environments. For success during evaluation in the real environment, the robot must visually detect the target object and drive up to a grasping range of the object. For object detection, we use our implementation of the ViLD detector [33] that has a true positive rate of 84.6% across our test objects. We have five categories of target objects: “bottle”, “can”, “cup”, “bowl” and “chips-bag” each of which has the same frequency of occurrence across episodes and in any given episode we have a single instance of the target object in the environment. We compare the performances of our proposed framework using the generalized linear (“Gen-Lin”) and neural (“Neural”) models for training the contextual bandit agent and “CP-SAT” and one-step greedy (“Greedy”) algorithms for path planning to a purely geometric approach that solves the Travelling Salesman Problem [40] (“TSP”). We assign a time budget of 30 seconds to the CP-SAT solver to have a realistic bound on robot response time. Each algorithm is evaluated over 50 episodes in real and 300 episodes in simulated environments. During training, whenever Algorithm 1 receives a positive signal on a given vantage point, this signal is extended to nearby points (see Appendix C for details). Figure 2 (a) shows a sample trajectory during the real kitchen evaluation. Figure 2 (b) shows a sample learned likelihood map for the simulated kitchen-1 environment.

We compare the performance of the agents on the following metrics: a) rate of success during evaluation (“Eval. Succ.”), b) SPL [6] during training (“Train SPL”), and c) SPL during evaluation (“Eval. SPL”). Higher “Train SPL” indicates faster rate of convergence. Table 1 presents the results of our first study, where we compare different learning and planning approaches for an arbitrary spatial distribution of test objects. The columns labeled “GT-Scores” use ground truth likelihoods of the vantage points mapped in Figures D.1, and D.2 in Appendix D, and computed as shown in Figure B.1 (a) (Appendix B). For both training and evaluation, we use 25 vantage points for the real environment and the kitchen environment 1 and 50 for kitchen environment 2.

Our first observation is the significant improvement in performance achieved by our planners using “GT-Scores” over “TSP” in all the environments, and this validates the importance of estimating the importance of the vantage points in the context of the target object in addition to optimizing for the room geometry. The performances for “GT-Scores” provide an upper bound for the agents that learn the likelihood function through exploration. Although the performance of “CP-SAT” shines in the real evaluation[2], under the planning time budget of 30 seconds, the performance of our proposed one-step greedy solver regularly matches up and often beats the CP-SAT solver, especially in larger and more cluttered kitchen environment 2. The performance of the “Neural” model often lags behind the “Gen-Lin” model due to the computational limitations imposed by the NTK approximation.

4 Limitations and Ongoing Work

Our proposed approach can get adversely affected due to: 1) detection failure, 2) slowness of WMLP solvers, and 3) early stopping of CP-SAT solver (not running the CP-SAT solver until the end may give us feasible but poor solutions). Inclusion of orientation along with the robot’s base position can help in mitigating missed detections. Using richer feature embeddings can also improve object detection from distance. Related to that, the choice of the network architecture in the Neural model is severely limited by our usage of the NTK approximation to compute confidence bounds. Leveraging more time-efficient approximation schemes may allow for more complex (and potentially more accurate) network architectures. Faster convergence is possible in training by using a likelihood-guided sampling scheme but this may also create opportunities for local minima. These are among the missing aspects we are currently investigating.

---

[2] Visit `https://sites.google.com/view/find-my-glasses/home` for videos of real world tests.

References

[1] Yasin Abbasi-Yadkori, David Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *NIPS*, 2011.

[2] F. Afrati, S. Cosmadakis, C. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the traveling repairman problem. *Theoretical Informatics and Applications*, 20(1):79–86, 1986.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[4] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[5] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[6] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav revisited: On evaluation of embodied agents navigating to objects. *CoRR*, abs/2006.13171, 2020.

[7] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *STOC*, pages 163–171, 1994.

[8] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008.

[9] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[10] Johann Borenstein, Yoram Koren, et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[11] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[12] N. Cesa-Bianchi and C. Conconi, A. Gentile. A second-order perceptron algorithm. *SIAM J.Comput.*, 34(3):640–668, 2005.

[13] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[14] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[15] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *In Neural Information Processing Systems (NeurIPS)*, 2020.

[16] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *International Conference on Learning Representations (ICLR)*, 2020.

[17] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020.

[18] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proc. 44th Symposium on Foundations of Computer Science (FOCS 2003)*, pages 36–45, 2003.

[19] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *International Conference on Learning Representations*, 2019.

[20] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[21] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2018.

[22] Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002.

[23] Gamini Dissanayake, Shoudong Huang, Zhan Wang, and Ravindra Ranasinghe. A review of recent developments in simultaneous localization and mapping. In *2011 6th International Conference on Industrial and Information Systems*, pages 477–482. IEEE, 2011.

[24] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997.

[25] Kuan Fang, Fei-Fei Li, Silvio Savarese, and Alexander Toshev. Scene memory transformer for embodied agents in long time horizon tasks. In *CVPR 2019*, 2019.

[26] Anthony G Francis, Carolina Parada, Dmitry Kalashnikov, Edward Lee, Fei Xia, Jake Varley, Jie Tan, Krzysztof Marcin Choromanski, Leila Takayama, Mikael Persson, et al. Learning model predictive controllers with real-time attention for real-world navigation. In *CoRL*, 2022.

[27] A. Garcia, P. Jodra, and J. Tejel. A note on the traveling repairman problem. *Networks*, 40(1):27–31, 2002.

[28] C. Gentile and F. Orabona. On multilabel classification and ranking with partial feedback. In *Advances in Neural Information Processing Systems*, volume 25, pages 1151–1159. Curran Associates, Inc., 2012.

[29] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation. In *International Conference on Learning Representations*, 2022.

[30] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

[31] Google. Google OR-Tools. `https://developers.google.com/optimization`. [Online; accessed 10-June-2022].

[32] Google. OR-Tools - Google Optimization Tools. `https://github.com/google/or-tools`. [Online; accessed 10-June-2022].

[33] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[34] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Mach. Learn.*, 69(2-3):169–192, 2007.

[35] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE transactions on automation science and engineering*, 18(2):398–400, 2021.

[36] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[37] Dongsung Kim and Ramakant Nevatia. Symbolic navigation with a generic map. *Autonomous Robots*, 6(1):69–88, 1999.

[38] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474, 2017.

[39] E. Koutsoupias, C. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. 23nd Colloquium on Automata, Languages and Programming*, pages 280–289, 1996.

[40] Jan Karel Lenstra and AHG Rinnooy Kan. Some simple applications of the travelling salesman problem. *Journal of the Operational Research Society*, 26(4):717–733, 1975.

[41] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[42] M. Meng and A.C. Kak. Mobile robot navigation using neural networks and nonmetrical environmental models. *IEEE Control Systems Magazine*, 13(5):30–39, 1993.

[43] Min Meng and Avinash C Kak. Neuro-nav: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 750–757. IEEE, 1993.

[44] E. Minieka. The delivery man problem on a tree network. *Ann. Oper. Res.*, 18:261–266, 1989.

[45] Arsalan Mousavian, Alexander Toshev, Marek Fišer, Jana Košecká, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852. IEEE, 2019.

[46] J Pan, DJ Pack, A Kosaka, and AC Kak. Fuzzy-nav: A vision-based robot navigation architecture using fuzzy inference for uncertainty-reasoning. In *Procs. of the World Congress on Neural Networks*, pages 602–607, 1995.

[47] Laurent Perron and Vincent Furnon. Or-tools.

[48] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[49] Santhosh K. Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. Poni: Potential functions for objectgoal navigation with interaction-free learning. In *Computer Vision and Pattern Recognition (CVPR), 2022 IEEE Conference on*. IEEE, 2022.

[50] A. Santara, G. Aggarwal, S. Li, and C. Gentile. Learning to plan variable length sequences of actions with a cascading bandit click model of user feedback. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151, pages 767–797, 2022.

[51] J Santos-Victor and Giulio Sandini. Visual-based obstacle detection: a purposive approach using the normal ow. In *Proc. of the International Conference on Intelligent Autonomous Systems, Karlsruhe, Germany*. Citeseer, 1995.

[52] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[53] Shai Shalev-Shwartz and Amnon Shashua. On the sample complexity of end-to-end training vs. semantic abstraction training. *arXiv preprint arXiv:1604.06915*, 2016.

[54] Paul Shaw, Vincent Furnon, and Bruno De Backer. *A Constraint Programming Toolkit for Local Search*. Springer US, Boston, MA, 2002.

[55] R. Sitters. The minimum latency problem is np-hard for weighted trees. In *Proc. 9th International IPCO Conference*, pages 230–239, 2002.

[56] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[57] Ashit Talukder, S Goldberg, Larry Matthies, and Adnan Ansar. Real-time detection of moving objects in a dynamic scene from moving robotic vehicles. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 2, pages 1308–1313. IEEE, 2003.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[59] Ayzaan Wahid, Austin Stone, Kevin Chen, Brian Ichter, and Alexander Toshev. Learning object-conditioned exploration using distributed soft actor critic. *CoRR*, abs/2007.14545, 2020.

[60] Ziqi Wei. New methods for solving the minimum weighted latency problem, 2018.

[61] B.Y. Wu. Polynomial time algorithms for some minimum latency problems. *Inf. Process. Lett.*, 75(5):225–229, 2000.

[62] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[63] Xin Ye and Yezhou Yang. From seeing to moving: A survey on learning for visual indoor navigation (vin). *arXiv preprint arXiv:2002.11310*, 2020.

[64] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

APPENDIX

A Review of Object-Nav Algorithms

Object-nav algorithms work by learning semantic relationships between the target object and the topology of the environment. They can be classified into two categories: map-based and map-free [8]. Map-free algorithms [57, 51, 45, 19, 49, 59, 29] do not require a map of the environment and can decide where to go based directly on the current observations and past memories without having to maintain a global representation of the environment. This requires them to solve the problem of localization and mapping in conjunction with the object-nav problem, making their sample complexity very high. As a result, these algorithms are often trained in simulation, with inevitably simplified human and environment representations, resulting in regressions in real environments due to the sim-to-real gap [35], which is further exacerbated by the fact that many of them work with low-level observations.

Map-based algorithms [43, 42, 46, 37, 10, 9] assume that a map of the environment is available at the time of path-planning. The map could be an occupancy map showing the probability of obstacles being at each location. It could also be a topological map [22] that is comprised of a graph where nodes represent characteristic places and edges contain reachability information (distances, times, etc.) between pairs of nodes. A few of these algorithms construct a map of the current region before planning in that region [16, 17, 25, 15]. Map-based methods are typically modular, hence, sample-efficient [53] and easier to deploy in the real world. However, the validity of the solution devised by these algorithms is a strong function of the accuracy of the map and localization of the robot. Thanks to advancements in Simultaneous Localization and Mapping (SLAM) algorithms [23], constructing an occupancy map of reasonable accuracy has become relatively easy in modern robotic systems.

B Theoretical Underpinning

Model. We formalize our problem as follows. In the 2D occupancy map of the environment, let us denote all the points by a set $\mathcal{X}$. Set $\mathcal{X}$ is partitioned into the set of *feasible* points $\mathcal{F}$ (the points the robot can freely navigate) and the set of *non-feasible* points $\mathcal{N}$ (the points occupied by obstacles like furnitures), so that $\mathcal{X} = \mathcal{F} \cup \mathcal{N}$, and $\mathcal{F} \cap \mathcal{N} = \emptyset$. The three sets $\mathcal{X}$, $\mathcal{F}$, and $\mathcal{N}$ are available to us before planning. Each $x \in \mathcal{F}$ comes with a *visibility set* $V(x) \subseteq \mathcal{X}$, that is, a set of points that the robot can inspect while standing[3] at x. For concreteness, visibility is defined in terms of Euclidean distance as $V(x) = \{x' \in \mathcal{X} \, : \, ||x - x'|| \le r_{\text{vis}}\}$, for some visibility range $r_{\text{vis}} > 0$ which represents the effective range of the object detectors on the system (e.g., $r_{\text{vis}} = 2.5$ meters for our robot platform).

We have n movable objects of interest (glasses, keys, etc.). We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For each pair $(i, x) \in [n] \times \mathcal{F}$, denote by $p_i(x)$ the probability that the robot spots object i while standing at position x. These probabilities are in turn defined by n (unknown) probability distributions $\{\mathbb{P}_i, i \in [n]\}$, with support $\mathcal{X}$, that determine where objects are located, so that $p_i(x) = \mathbb{P}_{y \sim \mathbb{P}_i}(y \in V(x))$. Notice that an object can, in principle, be anywhere in the scene. The probabilities $p_i(x)$ are unknown to the planner, and have to be learned through interactions with the environment. We model them as $p_i(x) = f(\phi(i, x); \theta)$, where $\phi \, : \, [n] \times \mathbb{R}^d \to \mathbb{R}^D$ is a mapping that featurizes the pair (i, x) into a D-dimensional real vector, for some feature dimension $D \ge d$, $f \, : \, \mathbb{R}^D \times \mathbb{R}^m \to [0, 1]$ is a known function, and $\theta \in \mathbb{R}^m$ is an unknown vector of parameters, for some parameter dimension m. For example, in a *generalized linear* model, we have $f(\phi; \theta) = \sigma(\theta^\top \phi)$, where σ is the sigmoidal function $\sigma(z) = \frac{e^{zs}}{1+e^{zs}}$ with slope s at the origin, and $m = D$. Our theoretical analysis uses this generalized linear model, while in our experimental evaluation, we compare both linear and neural models for $f(\cdot; \theta)$.

Planning and Regret. In each navigation episode $t = 1, 2, \ldots$, there will be only one object $i_t \in [n]$ in the scene[4], and the identity of this object is *known* to the robot. The environment generates the position y_t of object i_t by drawing y_t from $\mathbb{P}_{i_t}$. Let $x_{0,t} \in \mathcal{F}$ be the starting position of the robot

---

[3] We are assuming here that the robot can sample from x any pose via 360 degree rotation.

[4] This is not a strict requirement, our analysis can be seamlessly extended to the case where multiple instances of the same object are simultaneously present on the scene.

in episode t. The algorithm begins by sampling a total of k vantage points $x_{1,t}, \ldots, x_{k,t} \in \mathcal{F}$. The planner has to generate a path $J_t = \langle x_{\pi_t(1),t}, \ldots, x_{\pi_t(k),t} \rangle$ across them, where $\pi_t(\cdot)$ is a permutation of the indices $\{0\} \cup [k]$, with $\pi_t(0) = 0$. The robot traverses the path in the order dictated by J_t, and stops as soon as the object is spotted, as allowed by the visibility structure $V(x_{\ell,t})$, $\ell \in \{0\} \cup [k]$. It is also reasonable to admit that the robot may incur some detection failures during an episode, an event we denote by $\mathcal{E}_t$, to which we shall assign an independent probability $\mathbb{P}(\mathcal{E}_t)$ to occur. We henceforth drop the episode subscript t for notational convenience.

The *path length* loss $L(y, J)$ of J is the actual distance traversed by the robot over the points $x_0, x_1, \ldots, x_k$ *before* spotting object i in position y. On the other hand, if the object is not found, it is reasonable to stipulate that the loss incurred will be a large number $L_M > \sum_{\ell=1}^{k} \sum_{j=1}^{\ell}, \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})$, bigger than the total path length of any length-k path $\langle x_{\pi(1)}, \ldots, x_{\pi(k)} \rangle$. Overall

$$
\begin{aligned}
L(y, J) &= \mathbb{1}\{\mathcal{E}\} L_M \qquad\qquad (1)\\
&+(1 - \mathbb{1}\{\mathcal{E}\}) \left(\sum_{\ell=1}^{k} \underbrace{\mathbb{1}\{y \in V(x_{\pi(\ell)}) \setminus (V(x_{\pi(0)}) \cup \ldots \cup V(x_{\pi(\ell-1)}))\}}_{V(x_{\pi(\ell)}) \text{ is the first ball in the traversal order where object is located}} \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)}) \right)
\end{aligned}
$$

where $\mathbb{1}\{\cdot\}$ is the indicator function of the predicate at argument, and $\mathrm{dist}_\star(x_1, x_2)$ denotes the $A^\star$ distance between the two vantage points x_1 and x_2 on the scene, i.e., the robot path length between x_1 and x_2 (notice that $\mathrm{dist}_\star(x_1, x_2) \geq ||x_1 - x_2||$). Observe that, in the absence of a failure, we are assuming the object will eventually be found during each episode. Hence, a failure will be ascertained only at the end of an episode. We approximate the above by disregarding the overlap among the visibility balls $V(x_{\pi(\ell)})$ (hence somehow assuming these balls do not influence one another), and then take expectation over $y \sim \mathbb{P}_i$ and the independent Bernoulli variables $\mathbb{1}\{\mathcal{E}\}$, with expectation $\mathbb{P}(\mathcal{E})$. This yields the (approximate) average path length

$$
\mathbb{E}_i[L(y, J)] = \mathbb{P}(\mathcal{E})\, L_M + (1 - \mathbb{P}(\mathcal{E})) \left(\sum_{\ell=1}^{k} p_i(x_{\pi(\ell)}) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)}) \right) \qquad (2)
$$

where $\mathbb{E}_i[\cdot]$ is a short-hand for $\mathbb{E}_{y \sim \mathbb{P}_i, \mathcal{E}}[\cdot]$. We are now in a position to define our benchmark performance measure against which regret will be defined. The *benchmark planner* knows the distributions $\{\mathbb{P}_i,\ i \in [n]\}$ and the probability $\mathbb{P}(\mathcal{E})$, and computes a path $J^\star = \langle x_1^*, \ldots, x_k^\star \rangle$ whose elements are taken from $\mathcal{F}$, such that $J^\star$ minimizes $\mathbb{E}_i[L(y, J)]$ over all length-k paths (and permutations thereof) that can be constructed out of points from $\mathcal{F}$. Given a sequence of episodes $t = 1, \ldots, T$ with corresponding objects $i_1, \ldots, i_T$ (and starting positions $x_{0,1}, \ldots, x_{0,T}$), we define the cumulative *regret* $R_T(J_1, \ldots, J_T)$ of a planner that generates paths $J_1, \ldots, J_T$ as

$$
R_T(J_1, \ldots, J_T) = \sum_{t=1}^{T} \mathbb{E}_{i_t}[L(y_t, J_t)] - \mathbb{E}_{i_t}[L(y_t, J_t^\star)]\,.
$$

We would like this cumulative regret to be *sublinear* in T with high probability (in the random draw of position y_t at the beginning of each episode t). Notice that the last term in the RHS of (2) is independent of J, hence L_M will play no role in the regret computation.

Algorithm. Our "formal" algorithm is described as Algorithm 1. The algorithm operates on an ϵ-cover[5] $\mathcal{F}_\epsilon$ of $\mathcal{F}$ and generalized linear probabilities $p_{i_t}(x) = \sigma(\theta^\top \phi(i_t, x))$. The algorithm replaces the above true probabilities in the average path length (2) with lower confidence estimations $\sigma\Big(\widehat{\Delta}_t(x) - \epsilon_t(x)\Big)$, and then computes J_t by minimizing (2) over the choice of k points within $\mathcal{F}_\epsilon$, as well as their order (permutation π). A sequence of signals $\langle s_{1,t}, \ldots, s_{k_t',t} \rangle = \langle -1, \ldots, -1, +1 \rangle$ observed during episode t is associated with the successful path J_t in that a sequence of $k_t' - 1$ negative signals ("-1" = object not spotted from $x_{\pi_t(\ell),t}$, for $\ell = 1, \ldots, k_t' - 1$) precede a positive signal ("$+1$" = object spotted at $x_{\pi_t(k_t'),t}$). On the other hand, when the object is not found throughout the length-k path, the sequence of signals becomes $\langle s_{1,t}, \ldots, s_{k,t} \rangle = \langle -1, \ldots, -1, -1 \rangle$ (this happens with probability $\mathbb{P}(\mathcal{E})$).

---

[5] Recall that $\mathcal{F}_\epsilon$ is an ϵ-cover of $\mathcal{F}$ if $\mathcal{F}_\epsilon \subseteq \mathcal{F}$ and for all $x \in \mathcal{F}$ there is $x' \in \mathcal{F}_\epsilon$ for which $\mathrm{dist}_\star(x, x') \leq \epsilon$. It is easy to see that, given a 2D-scene, the cardinality $|\mathcal{F}_\epsilon|$ of $\mathcal{F}_\epsilon$ is $O(1/\epsilon^2)$.

When the object is found, Algorithm 1 uses the observed signals to update over time a D-dimensional weight vector $\widehat{\theta}$, and a $(D \times D)$-dimensional matrix M. Vector $\widehat{\theta}_t$ is used to estimate $\theta^\top \phi(i_t, x)$, through $\widehat{\Delta}_t(x)$, while matrix M_t delivers a standard confidence bound via $\epsilon_t^2(x)$. The update rule implements a second-order descent method on logistic loss trying to learn the unknown vector θ out of the signals $s_{j,t}$. In particular, the update $\widehat{\theta}_{c_t+j-1} \to \widehat{\theta}_{c_t+j}$ is done by computing a standard online Newton step (e.g., [34]). Notice that at each episode t, both matrix M and vector $\widehat{\theta}$ get updated $m_t = k'_t$ times, which corresponds to the number of (valid) signals received in that episode. Counter c_t accumulates the number of such updates across (non-failing) episodes. On the contrary, when the object is not found, we know that there has been a failure, hence we disregard all (negative) signals received and jump to the next episode with no updates ($m_t = 0$).

From a computational standpoint, calculating J_t as described in Algorithm 1 is hard, since the planning problem the algorithm is solving at each episode is essentially equivalent to a *Weighted Minimum Latency Problem* (WMLP) [60], also called *traveling repairman problem*, which, on a generic metric space is NP-hard and also MAX-SNP-hard [7]. Fast algorithms are available only for very special metric graphs, like paths [2, 27] edge-unweighted trees [44], trees of diameter 3 [7], trees of constant number of leaves [39], and the like [61]. Even for weighted trees the problem remains NP-hard [55]. Approximation algorithms are indeed available [18], but they are not practical enough for real-world deployment. In our experiments (Sections 2–3), we implement and compare fast planning approximations to Algorithm 1.

In both the planning and the training of the contextual bandit algorithm, we are using the average path length as a minimization objective because it is easier for the WMLP solver to handle. However, when it comes to evaluating performance in our experiments, we use the Success weighted by Path Length (SPL) metric [4] because it is a more established metric in the object-nav literature.

Regret Analysis. From a statistical standpoint, Algorithm 1 is a simplified version of the slightly more complex Algorithm 2 which is the one our regret analysis applies to. Specifically, we replace the exploration parameter α with an exact expression $\alpha(k, D, T, \delta, B)$ that is required for our analysis to go through. Moreover, Algorithm 2 includes a projection step, where the update $\widehat{\theta}_{c_t+j-1} \to \widehat{\theta}_{c_t+j}$ is performed by first projecting $\widehat{\theta}_{c_t+j-1}$ onto the set

$$\{\theta \in \mathbb{R}^D \,:\, |\theta^\top \phi(i_t, x_j)| \le B\}$$

w.r.t. the Mahalanobis distance

$$d_{c_t}(\theta_1, \theta_2) = (\theta_1 - \theta_2)^\top M_{c_t} (\theta_1 - \theta_2)\ ,$$

so as to obtain $\widehat{\theta}'_{c_t+j-1}$, and then using a standard Newton step with $\widehat{\theta}'_{c_t+j-1}$.

We now turn to the proof of Theorem 1 in the main body of the paper. We first need a couple of ancillary lemmas.

Lemma 1 *Let Algorithm 2 be run on an ϵ-cover $\mathcal{F}_\epsilon$ with $\epsilon = O(1/\sqrt{t})$. Let episode t be such that $|\theta^\top \phi(i_t, x) - \phi(i_t, x)^\top \widehat{\theta}'_{c_t}| \le \epsilon_t(x)$ for all $x \in \mathcal{F}$. Also, let $D_M = \max_{x,x' \in \mathcal{F}} \mathrm{dist}_\star(x, x')$ denote the diameter of the scene. Then*

$$\mathbb{E}_{i_t}[L(y_t, J_t)] - \mathbb{E}_{i_t}[L(y_t, J^\star)] = O\left((1 - \mathbb{P}(\mathcal{E}_t))\left(\frac{k^2}{\sqrt{t}} + D_M\, k \sum_{\ell=1}^{k} \epsilon_t(x_{\pi_t(\ell),t})\right)\right)\ ,$$

where $\mathcal{E}_t$ denotes the failure event during episode t.

Proof. We fix episode t and remove subscript t and c_t for convenience. As short-hands, let us denote by $J = \langle x_1, \ldots, x_k \rangle$ the path computed by Algorithm 2 in episode t and by $J^\star_\epsilon = \langle x^\star_{1,\epsilon}, \ldots, x^\star_{k,\epsilon} \rangle$ the minimizer of (2) when the k vantage points are constrained to lie in the ϵ-cover $\mathcal{F}_\epsilon$. We clearly have

$$\begin{aligned}|\mathbb{E}_i[L(y, J^\star_\epsilon)] - \mathbb{E}_i[L(y, J^\star)]| &\le (1 - \mathbb{P}(\mathcal{E}_t))k(k+1)\epsilon \\ &= O\left((1 - \mathbb{P}(\mathcal{E}_t))\frac{k^2}{\sqrt{t}}\right)\ .\end{aligned}$$

Algorithm 2 Contextual bandit planning algorithm.

```
Input: $\epsilon$-cover $\mathcal{F}_\epsilon$ of $\mathcal{F}$, $\epsilon > 0$, path length $k$, maximal range $B > 0$.
Init: $M_0 = kI \in \mathbb{R}^{D\times D}$, $\widehat{\theta}_1 = 0 \in \mathbb{R}^D$, $c_1 = 1$.
For $t = 1, 2, \dots, T$
    1. Get object identity $i_t$ , and initial position of the robot $x_{0,t}$ ;
    2. For $x \in \mathcal{F}$, set
           $\widehat{\Delta}_t(x) = \phi(i_t, x)^\top \widehat{\theta}'_{c_t}(x)$   and   $\epsilon_t^2(x) = \alpha(k, D, T, \delta, B)\, \phi(i_t, x)^\top M_{c_t-1}^{-1} \phi(i_t, x)$ ,
       where
           $\widehat{\theta}'_{c_t}(x) = \arg \min_{\theta \,:\, -B \le \theta^\top \phi(i_t, x) \le B} d_{c_t-1}(\theta, \widehat{\theta}_{c_t})$ ;
       and
           $\alpha(k, D, T, \delta, B) = O\Bigg( kB^2 + \left(\frac{c_\sigma}{c_{\sigma'}}\right)^2 D \log\left(1 + \frac{1}{k}\Big(\frac{t\, c_\sigma}{1 - c_\sigma} + \log\frac{t+1}{\delta}\Big)\right) + \left(\left(\frac{c_\sigma}{c_{\sigma'}}\right)^2 + \frac{1+B}{c_{\sigma'}}\right) \log\frac{k(t+1)}{\delta}\Bigg)$
    3. Compute $J_t = \langle x_{\pi_t(1),t}, \dots, x_{\pi_t(k),t} \rangle$ as        //solve WMLP at episode $t$
           $J_t = \arg \min_{\substack{x_1 \dots x_k \in \mathcal{F}_\epsilon \\ \text{permutation } \pi}} \sum_{\ell=1}^{k} \sigma\Big(\widehat{\Delta}_t(x_{\pi(\ell)}) + \epsilon_t(x_{\pi(\ell)})\Big) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{\pi(j-1)}, x_{\pi(j)})$
    4. Observe signal $\begin{cases} \langle s_{1,t}, \dots, s_{k'_t,t} \rangle = \langle -1, \dots, -1, +1 \rangle & \text{set } m_t = k'_t \\ \text{or} & \\ \langle s_{1,t}, \dots, s_{k,t} \rangle = \langle -1, \dots, -1, -1 \rangle & \text{set } m_t = 0 \end{cases}$
    5. For $j = 1, \dots, m_t$ (in the order of occurrence of items $x_j$ in $J_t$) update:
           $M_{c_t+j-1} = M_{c_t+j-2} + \phi(i_t, x_j)\phi(i_t, x_j)^\top,$
           $\widehat{\theta}_{c_t+j} = \widehat{\theta}'_{c_t+j-1} + \frac{1}{c_{\sigma'}} M_{c_t+j-1}^{-1} \nabla_{j,t}$ ,
       where $\nabla_{j,t} = \sigma(-s_{j,t}\, \widehat{\Delta}'_t(x_j))\, s_{j,t}\, \phi(i_t, x_j)$ , where $\widehat{\Delta}'_t(x_j) = \phi(i_t, x_j)^\top \widehat{\theta}'_{c_t+j-1}$
       with
           $\widehat{\theta}'_{c_t+j-1} = \arg \min_{\theta \,:\, -B \le \theta^\top \phi(i_t, x_j) \le B} d_{c_t+j-2}(\theta, \widehat{\theta}_{c_t+j-1})$ ;
    6. $c_{t+1} \leftarrow c_t + m_t$ .
```

Moreover,

$$
\begin{aligned}
&\frac{1}{(1 - \mathbb{P}(\mathcal{E}_t))}\, \mathbb{E}_i[L(y, J)] - \mathbb{E}_i[L(y, J^\star_\epsilon)] \\
&= \sum_{\ell=1}^{k} p_i(x_\ell) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{j-1}, x_j) - \sum_{\ell=1}^{k} p_i(x^\star_{\ell,\epsilon}) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x^\star_{j-1,\epsilon}, x^\star_{j,\epsilon}) \\
&\le \sum_{\ell=1}^{k} \sigma\Big(\theta^\top \phi(i, x_\ell)\Big) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x_{j-1}, x_j) - \sum_{\ell=1}^{k} \sigma\Big(\widehat{\theta}^\top \phi(i, x^\star_{\ell,\epsilon}) - \epsilon(x^\star_{\ell,\epsilon})\Big) \sum_{j=1}^{\ell} \mathrm{dist}_\star(x^\star_{j-1,\epsilon}, x^\star_{j,\epsilon}) \,.
\end{aligned}
$$

In turn, the above is upper bounded by

$$\sum_{\ell=1}^{k}\sigma\Big(\theta^\top\phi(i,x_\ell)\Big)\sum_{j=1}^{\ell}\mathrm{dist}_\star(x_{j-1},x_j)-\sum_{\ell=1}^{k}\sigma\Big(\widehat{\theta}^\top\phi(i,x_\ell)-\epsilon(x_\ell)\Big)\sum_{j=1}^{\ell}\mathrm{dist}_\star(x_{j-1},x_j)$$

$$\le\sum_{\ell=1}^{k}\sigma\Big(\widehat{\theta}^\top\phi(i,x_\ell)+\epsilon(x_\ell)\Big)\sum_{j=1}^{\ell}\mathrm{dist}_\star(x_{j-1},x_j)-\sum_{\ell=1}^{k}\sigma\Big(\widehat{\theta}^\top\phi(i,x_\ell)-\epsilon(x_\ell)\Big)\sum_{j=1}^{\ell}\mathrm{dist}_\star(x_{j-1},x_j)$$

$$\le\sum_{\ell=1}^{k}2\epsilon(x_\ell)\sum_{j=1}^{\ell}\mathrm{dist}_\star(x_{j-1},x_j)$$

$$=O\left(D_M\,k\sum_{\ell=1}^{k}\epsilon(x_\ell)\right)\,.$$

Putting together proves the claim.

Lemma 2 *Let $B>0$ be such that $\theta^\top\phi(i,x)\in[-B,B]$ for all $i\in[n]$ and $x\in\mathcal{F}$. Moreover, let c_σ and $c_{\sigma'}$ be two positive constants such that, for all $\Delta\in[-D,D]$ the conditions $0<1-c_\sigma\le\sigma(\Delta)\le c_\sigma<1$ and $\sigma'(\Delta)\ge c_{\sigma'}$ hold. Then with probability at least $1-\delta$, with $\delta<1/e$, we have*

$$d_{c_t-1}(\theta,\widehat{\theta}'_{c_t})\le\alpha(k,D,T,\delta,B)\,,$$

uniformly over $c_t\in[kT]$, where

$$\alpha(k,D,T,\delta,B)$$
$$=O\left(kB^2+\left(\frac{c_\sigma}{c_{\sigma'}}\right)^2D\log\left(1+\frac{1}{k}\Big(\frac{t\,c_\sigma}{1-c_\sigma}+\log\frac{t+1}{\delta}\Big)\right)+\left(\left(\frac{c_\sigma}{c_{\sigma'}}\right)^2+\frac{1+B}{c_{\sigma'}}\right)\log\frac{k(t+1)}{\delta}\right)\,.$$

Proof. The proof follows from standard concentration arguments applied to the logistic loss, which Algorithm 2 implicitly operates on. See, e.g., [50], Lemma 5 therein which, in turn, relies on [34] and [28]. The argument therein can be applied to the non-failing episodes, that is, those episodes on which state updates occur. In our bound above we are simply over-approximating the number of non-failing episodes within the first t episodes with t itself.

Theorem 1 *Let $D_M=\max_{x,x'\in\mathcal{F}}\mathrm{dist}_\star(x,x')$ be the diameter of the scene. Also, let the feature mapping $\phi:[n]\times\mathbb{R}^d\to\mathbb{R}^D$ be such that $||\phi(i,x)||\le1$ for all $i\in[n]$ and $x\in\mathcal{F}$, and let constant B be such that $||\theta||\le B$. Then a variant of Algorithm 1 exists that operates in episode t with an ϵ-covering $\mathcal{F}_\epsilon$ of $\mathcal{F}$ with $\epsilon=k/\sqrt{t}$, such that with probability at least $1-\delta$, with $\delta<1/e$, the cumulative regret of this algorithm satisfies, on any sequence of objects $i_1,\ldots,i_T$,*

$$R_T(J_1,\ldots,J_T)=O\Big((1-p)k^2\sqrt{T}+D_M\,k\,\sqrt{(1-p)kT\,\alpha(k,D,T,\delta,B)\,D\log(1+kT)}\Big)\,,$$

where $\alpha(k,D,T,\delta,B)=O\left[e^{2B}\left(k+D\,\log\left(1+\frac{kDT}{\delta}\right)\right)\right]$, and $p=\mathbb{P}(\mathcal{E}_t)$ is the (constant) failure probability. In the above, the big-oh notation hides additive and multiplicative constants independent of T, D, B, k, p, and δ.

Proof. From Lemma 2 and the Cauchy-Schwarz inequality it follows that

$$\begin{aligned}(\theta^\top\phi(i,x)-\phi(i,x)^\top\widehat{\theta}'_{c_t})^2&\le\phi(i,x)^\top M_{c_t-1}^{-1}\phi(i,x)\,d_{c_t-1}(\theta,\widehat{\theta}'_{c_t})\\&\le\left(\phi(i,x)^\top M_{c_t-1}^{-1}\phi(i,x)\right)\,\alpha(k,D,T,\delta,B)\end{aligned}$$

for all $i\in[n]$ and $x\in\mathcal{F}$. Hence we can apply Lemma 1 with

$$\epsilon_t^2(x)=\left(\phi(i,x)^\top M_{c_t-1}^{-1}\phi(i,x)\right)\,\alpha(k,D,T,\delta,B)\,.$$

Let $\mathcal{E}_t$ denote the failure event at episode t, with $\mathbb{P}(\mathcal{E}_t)=p$ for all t. Summing over $t=1,\ldots,T$, we can write

$$\sum_{t=1}^{T}\Big(\mathbb{E}_{i_t}[L(y_t,J_t)]-\mathbb{E}_{i_t}[L(y_t,J^\star)]\Big)$$

$$=O\left((1-p)k^2\sqrt{T}+D_M\,k\,\mathbb{E}\left[\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\epsilon_t(x_{\pi_t(\ell),t})\right]\right)=O\left((1-p)k^2\sqrt{T}+D_M\,k\,\sqrt{\alpha(k,D,T,\delta,B)}\;\mathbb{E}\right)\,,$$

where $\mathbb{E}$ is a short-hand for

$$\mathbb{E}\left[\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\sqrt{\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1}^{-1}\phi(i_t,x_{\pi_t(\ell),t})}\right] .$$

We now follow similar arguments as in the proof of Theorem 1 in [50] by focusing on

$$\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1}^{-1}\phi(i_t,x_{\pi_t(\ell),t}) .$$

First, by virtue of Lemma 6 in [50], we have, for each t,

$$\sum_{\ell=1}^{k}\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1}^{-1}\phi(i_t,x_{\pi_t(\ell),t}) \le e\sum_{\ell=1}^{k}\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1+\ell}^{-1}\,\phi(i_t,x_{\pi_t(\ell),t}) ,$$

so that

$$\begin{aligned}\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1}^{-1}\phi(i_t,x_{\pi_t(\ell),t}) &\le e\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1+\ell}^{-1}\,\phi(i_t,x_{\pi_t(\ell),t})\\ &= O\left(D\log(1+kT)\right) ,\end{aligned}$$

the last inequality following from standard upper bounds (e.g., [12, 1]). As a consequence

$$\sum_{t=1,\,\mathcal{E}_t=0}^{T}\sum_{\ell=1}^{k}\sqrt{\phi(i_t,x_{\pi_t(\ell),t})^\top M_{c_t-1}^{-1}\phi(i_t,x_{\pi_t(\ell),t})} = O\left(\sqrt{kD\log(1+kT)\sum_{t=1}^{T}(1-\mathcal{E}_t)}\right) ,$$

which we plug back. Using the concavity of the square root, this allows us to obtain

$$\begin{aligned}&\sum_{t=1}^{T}\Big(\mathbb{E}_{i_t}[L(y_t,J_t)]-\mathbb{E}_{i_t}[L(y_t,J^\star)]\Big)\\ &\qquad= O\Big((1-p)k^2\sqrt{T}+D_M\,k\,\sqrt{(1-p)kT\,\alpha(k,D,T,\delta,B)\,D\log(1+kT)}\Big) .\end{aligned}$$

Finally, observe that, since $\sigma(z)=\frac{\exp(z)}{1+\exp(z)}$, we have within the expression for $\alpha(k,D,T,\delta,B)$ in Lemma 2, $c_\sigma=\frac{e^B}{1+e^B}$ (hence $\frac{c_\sigma}{1-c_\sigma}=e^B$), and $c_{\sigma'}=e^{-B}/(1+e^{-B})^2\ge e^{-B}/4$. Plugging back concludes the proof.

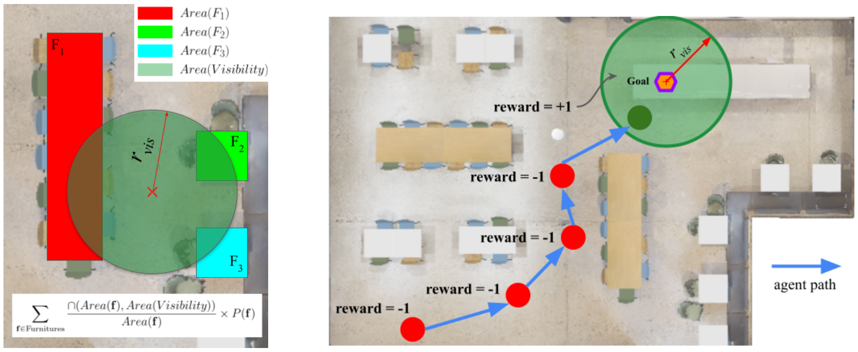
In a nutshell, the above analysis provides a high-probability regret guarantee of the form $k^2D\sqrt{T}$, when hyperparameters η and α in Algorithm 1 are assigned specific values, as detailed in Algorithm 2.

C Data augmentation

Each vantage point is described by a vector with positional, geometric and semantic features of the point along with the identity of the target object. We triangulate the position of the target object once the agent spots it from a vantage point. In order to improve learning efficiency, we assign positive training signal ("+1") to all the navigable points within r_{vis} radius from the object. Figure B.1 (b) illustrates this procedure.

D Object distributions

In this section, we describe the way objects are spawned in the environment in simulation. We only consider objects that are kept on table-tops. As shown in Figures D.1 and D.2, each table in the environment has a certain probability of housing the object. In each episode, for each object category, a table is sampled from the corresponding probability distribution. A location on the surface of the selected table is then picked uniformly at random to determine the object location within the environment. We also experimented with a peaky object distribution, where each object category was assigned a different table to be spawned exclusively on. We present the results in Table D.1 and the observations are similar to those reported in Section 3.


Area(F1)
Area(F2)
Area(F3)
Area(Visibility)
F1
F2
F3
r vis
$\sum_{\mathbf{f}\in\text{Furnitures}} \frac{\cap(Area(\mathbf{f}), Area(Visibility))}{Area(\mathbf{f})} \times P(\mathbf{f})$
Goal
r vis
reward = +1
reward = -1
reward = -1
reward = -1
reward = -1
agent path

(a) Calculation of ground-truth likelihood scores. $P(\mathbf{f})$ is the likelihood of the object appearing on furniture $\mathbf{f}$. $\sum_{\mathbf{f}\in\text{Furnitures}} P(\mathbf{f}) = 1$.

(b) Positive sample augmentation for improved sample efficiency. Radius r_{vis} is the robot's visibility range.

Figure B.1: **(a)** Ground Truth (GT) likelihood extraction. **(b)** Data augmentation used in training.

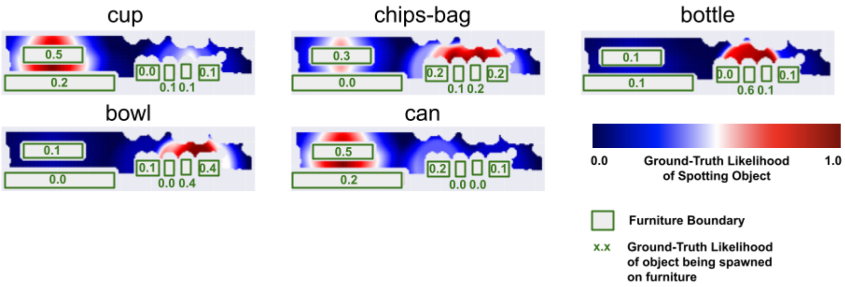


Figure D.1: Map of Simulated Kitchen 1 with distribution of occurrence of the five target objects categories "cup", "chips-bag", "bottle", "bowl", and "can".

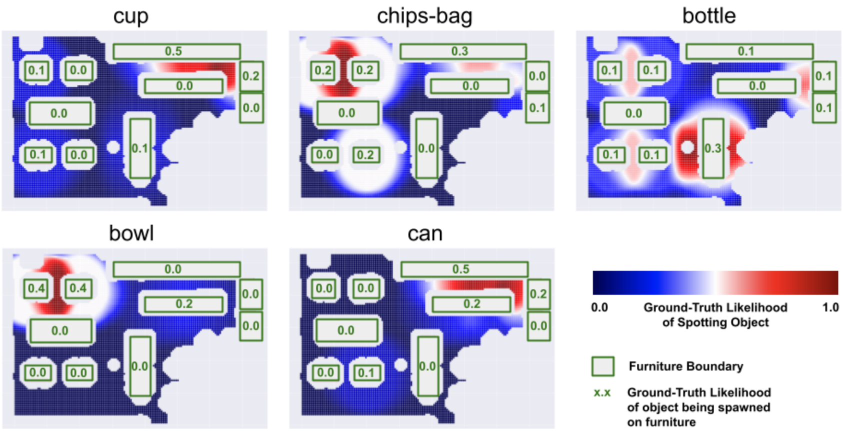


Figure D.2: Map of Simulated Kitchen 2 with distribution of occurrence of the five target objects categories "cup", "chips-bag", "bottle", "bowl", and "can".

	Kitchen Environment 1 (Peaky distributions)						
	TSP	**Gen-Lin**		**Neural**		**GT-Scores**	
		Greedy	**CPSAT**	**Greedy**	**CP-SAT**	**Greedy**	**CP-SAT**
Train SPL	-	0.39	0.37	0.35	0.28	-	-
Eval Succ Rate	0.88	0.9	0.86	0.85	0.88	0.88	0.91
Eval SPL	0.40	0.46	0.55	0.41	0.45	0.56	0.65

Table D.1: Experimental comparison of performance of our agents on a peaky object distribution in Kitchen Environment 1 against the metrics mentioned in Section 3.

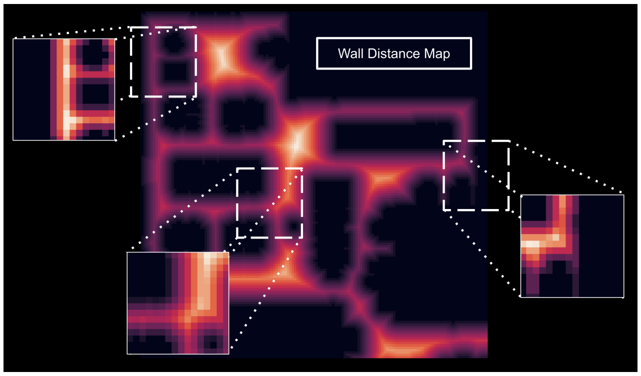


Figure D.3: The image shows grid based feature space where a local image patch from various places in the wall distance map are shown. The dotted white boxes indicated from which region the patch features are coming and the patches show how the features look when scaled to a 16 × 16 grid.

Table E.1: Hyperparameters for experiments with Non-Peaky object distributions in Kitchen Environment 1.

	Gen-Lin		**Neural**	
Hyperparameter	Greedy	CP-SAT	Greedy	CP-SAT
Learning Rate (η)	0.44	100	0.01	0.01
Exploration parameter (α)	0.1	0.1	0.1	3.44
Number of vantage points (k)	25	25	25	25
Alpha planner (α_p)	0.48	-	0.43	-
Map Resolution	75	75	75	75
Positional Embedding Size	50	15	50	10
Feature Vector Normalization	l^2-norm	l^2-norm	l^2-norm	l^2-norm
Sigmoid Scale (s)	1.0	1.0	20.0	19.8

Table E.2: Hyperparameters for experiments with Non-Peaky object distributions in Kitchen Environment 2.

	Gen-Lin		**Neural**	
Hyperparameter	Greedy	CP-SAT	Greedy	CP-SAT
Learning Rate (η)	16.62	10.44	0.01	0.01
Exploration parameter (α)	10	0.1	0.1	2.04
Number of vantage points (k)	50	50	50	50
Alpha planner (α_p)	0.49	-	0.38	-
Map Resolution	37	37	75	75
Positional Embedding Size	20	50	50	15
Feature Vector Normalization	mean-var	mean-var	mean-var	mean-var
Sigmoid Scale (s)	-	-	10.00	17.56

E Hyper-parameters

Tables E.1, E.2 and E.3 contain the hyperparameters for each of the algorithms tested in our experiments. Table E.5 gives the values searched for each hyperparameter.

Table E.3: Hyperparameters for experiments with Peaky object distributions in Kitchen Environment 1.

	Gen-Lin		**Neural**	
Hyperparameter	Greedy	CP-SAT	Greedy	CP-SAT
Learning Rate (η)	3.98	75.41	0.01	0.01
Exploration parameter (α)	7.15	2.59	0.10	1.40
Number of vantage points (k)	25	25	25	25
Alpha planner (α_p)	0.59	-	0.57	-
Map Resolution	75	75	75	75
Positional Embedding Size	10	20	10	10
Feature Vector Normalization	l^2-norm	l^2-norm	l^2-norm	l^2-norm
Sigmoid Scale	-	-	20.00	10.67

Table E.4: Hyperparameter for experiments in the real world.

Hyperparameter	Greedy	CP-SAT
Learning Rate (η)	0.01	0.01
Exploration parameter (α)	0.11	0.1
Number of vantage points (k)	25	25
Alpha planner (α_p)	0.49	-
Map Resolution	75	75
Positional Embedding Size	20	30
Feature Vector Normalization	l^2-norm	l^2-norm
Sigmoid Scale	18.38	15.78

Table E.5: Hyperparameter search ranges and scales.

Hyperparameter	**Values Searched**	**Search Scale**
Learning Rate (η) (Gen-Lin model only)	$[0.01, 100.0]$	Log
Exploration parameter (α)	$[0.1, 10.0]$	Linear
Number of vantage points (k)	$\{25, 50\}$	–
Alpha planner (α_p) (Greedy only)	$[0.1, 0.9]$	Linear
Map Resolution	$\{37, 75, 150\}$	–
Positional Embedding Size	$\{5, 10, 15, 20, 30, 50\}$	–
Feature Vector Normalization	$\{l^2$-norm, mean-var$\}$	–
Sigmoid Scale (for Neural model only)	$[10, 20]$	Linear

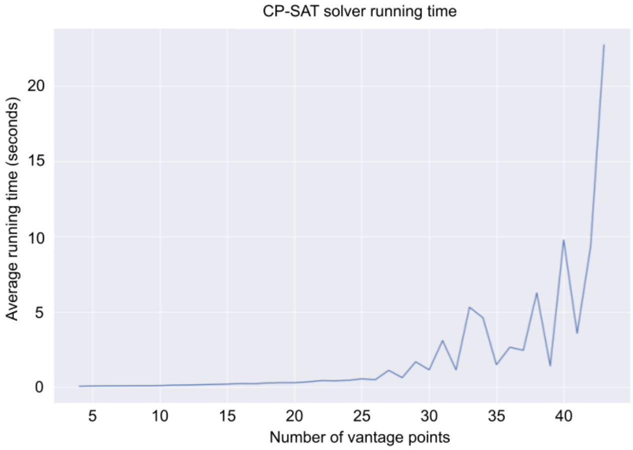


Figure E.1: Running time of CP-SAT solver on Intel Xeon 8-core CPU for different numbers of vantage points in Kitchen Environment 2 environment.