Embodied Visual Navigation With Automatic Curriculum Learning in Real Environments

Steven D. Morad, Roberto Mecca, Rudra P. K. Poudel, Stephan Liwicki, and Roberto Cipolla, *Senior Member, IEEE*

***Abstract*—We present NavACL, a method of automatic curriculum learning tailored to the navigation task. NavACL is simple to train and efficiently selects relevant tasks using geometric features. In our experiments, deep reinforcement learning agents trained using NavACL significantly outperform state-of-the-art agents trained with uniform sampling – the current standard. Furthermore, our agents can navigate through unknown cluttered indoor environments to semantically-specified targets using only RGB images. Obstacle-avoiding policies and frozen feature networks support transfer to unseen real-world environments, without any modification or retraining requirements. We evaluate our policies in simulation, and in the real world on a ground robot and a quadrotor drone. Videos of real-world results are available in the supplementary material.**

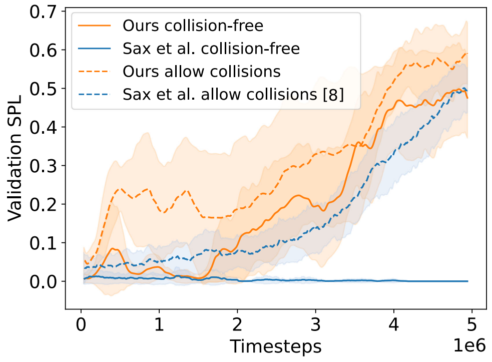
***Index Terms*—Autonomous agents, reinforcement learning, visual-based navigation.**

I. Introduction

Navigation forms a core challenge in embodied artificial intelligence (embodied AI) [1], [2]. Before the embodied AI renaissance, approaches such as active vision [3] and active visual simultaneous localization and mapping (active VSLAM) [4] were popular methods for building autonomous agents. They combined classical motion planning [5], [6] with non-learned exploration policies such as frontier expansion [7] to direct the agent. Active VSLAM and active vision work well in ideal circumstances, but are brittle and lack generalization ability in real-world situations.

Deep reinforcement learning (DRL) gained traction in the landmark paper [9], where DRL agents outperformed humans – all be it in relatively simple arcade games. Since then, the scope of DRL has expanded to real-world applications. In robotics and navigation, DRL shows promise as an alternative to classical models due to its surprising robustness and ability to generalize to real-world uncertainties. [10] trained visual navigation agents in a video-game maze, showing that DRL agents can memorize the layout of a maze from vision alone. Since then, there has been an explosion of DRL-based visual navigation, fuelled by the abundance of indoor photo-realistic simulators and datasets [11]–[15]. Evidence suggests DRL outperforms traditional methods in such cluttered, realistic indoor environments [16].

DRL visual navigation tasks can be broken down into point [8], [16], [17], object [8], [10], [18], or area [19], [20] navigation based on target description [21]. Object navigation goals can be represented with images [19], [22] or semantics [8], [23], [24]. Image-based representations match a reference image of the target object to the current agent observation, requiring a new image for each specific object instance. Semantic representations can be specified as a list of instructions [25],



(a) Performance Comparison

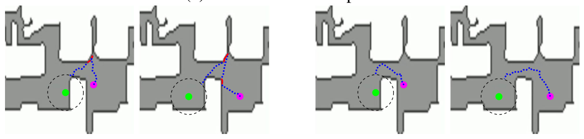

(b) Sax *et al.* [8] (c) Our Policy with NavACL

Fig. 1. (1a) The 2019 CVPR Habitat RGB Navigation challenge winner [8] does not learn an effective policy when the collision-free property is enforced (episode termination and negative reward upon collision with an obstacle). Plots represent the mean and standard deviation of a single train/test scene over five trials. Collision-free agents are important for Sim2Real transfer but are much harder to train (SPL defined in 3 – larger is better). (1b) Top-down view of fully-trained navigation agents exploiting collision mechanics to slide along walls. (1c) Our agents with collision avoidance and automatic curriculum learning. The agent starts at the pink point and moves towards the green goal, producing a blue path. Red dots indicate collisions.

Manuscript received September 11, 2020; accepted December 11, 2020. Date of publication January 1, 2021; date of current version January 20, 2021. This work was supported by Toshiba Europe Limited. This letter was recommended for publication by Associate Editor J. Kober and Editor T. Asfour upon evaluation of the reviewers' comments. *(Corresponding author: Stephan Liwicki.)*

Steven D. Morad is with the Toshiba Europe Limited, Cambridge, CB4 0GZ, U.K., and also with the Computer Laboratory, University of Cambridge, 15 JJ Thomson Ave, Cambridge CB3 0FD, U.K. (e-mail: stevenmorad@gmail.com).

Roberto Mecca, Rudra P. K. Poudel, and Stephan Liwicki are with the Toshiba Europe Limited, Cambridge, CB4 0GZ, U.K. (e-mail: roberto.mecca@crl.toshiba.co.uk; rudra.poudel@crl.toshiba.co.uk; stephan.liwicki@crl.toshiba.co.uk).

Roberto Cipolla is with the Toshiba Europe Limited, Cambridge, CB4 0GZ, U.K., and also with the Department of Engineering, University of Cambridge, Trumpington St, Cambridge CB2 1PZ, U.K. (e-mail: cipolla@eng.cam.ac.uk).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2020.3048662, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3048662

or as a target label [24]. Language instructions (*e.g.* left at staircase) can preclude scene generalization. Semantic labels show great promise for real-world applications – they are how humans describe targets to each other, and are not bound to a specific scene or object instance. Semantic label-based agents can be deployed to novel environments and duties; suitable for post-disaster recovery robots or embodied assistance technology with a wide range of task scenarios.

A. Contributions

We focus on indoor object-driven navigation to semantically-specified targets. We are interested in generalization to new environments and targets, including simulator-to-real-world (Sim2Real) generalization, without any retraining.

Utilizing semantic networks pretrained on large segmentation datasets like COCO [26], we show agents with test-time generalization to object classes not seen in training simulations. To the best of our knowledge, we are the first to demonstrate this test-time generalization ability in simulation, and further extend it to the real world. We call this test-time generalization *target-agnostic semantic navigation*.

While DRL visual navigation has proven itself in simulation, it rarely transfers to the real world. One challenge is overfitting to simulator-rendered images [27]. Using frozen feature encoders trained on real images, [8] shows compelling generalization ability across multiple simulators. In our work, we demonstrate frozen feature networks and collision avoidance help bridge the Sim2Real gap by showcasing our policies on real robots in real environments. Another issue present in almost every navigation simulator is collision modeling exploitation [1]. Agents drive into a wall at an angle and slide along it, covering the perimeter of a simulated building (Fig. 1(b)). [28] demonstrates collision-avoidance policies trained entirely in simulation can transfer to the real world, but stops short of investigating longer-term navigation policies. DRL for long-term planning with obstacle avoidance has yet to be perfected.

Enforcing collision-free paths for agents results in increased reward sparsity, making training more difficult with state-of-the-art navigation tools (Fig. 1(a)). We mitigate this elevated sparsity using automatic curriculum learning. The essence of curriculum learning is selecting and ordering training data in a way that produces desirable characteristics in the learner, such as generality, accuracy, and sample efficiency. Automatic curriculum learning (ACL) is the process of generating this curriculum without a human in the loop. Curriculum for neural networks was proposed by [29], and [30] affords a thorough overview of ACL applied to DRL. For navigation, tasks can be represented using low-dimensional Cartesian start and goal states. Some ACL methods that produce tasks of this form are asymmetric self-play [31] and GoalGAN [32]. Asymmetric self-play requires collecting distinct episodes for two separate policies, which is computationally expensive using 3D simulators. GoalGAN trades performance for generality. It can generate tasks for arbitrary problems, but uses a generative adversarial network (GAN) which is notoriously unstable and difficult to train. Instead, we trade generality for efficiency and propose a simple classification-based ACL method for navigation, termed NavACL.

In summary, we cast the visual navigation problem setup as follows:

1) the agent's observations consist of RGB images from an agent-mounted camera and the semantic label of the target (*e.g.* "football," "vase"),
2) the agent's actions consist of discrete, position-based motion primitives (*i.e.* move forward, turn left or right), without explicit loop closure outside of said primitives,
3) upon reaching the target, collision with the environment, or exceeding a preset time limit, the episode ends

and contribute:

1) a simple and efficient method to automatically generate curriculum for navigation agents,
2) target-agnostic semantic navigation – finding objects and object classes never seen during policy training,
3) a collision-free navigation policy for complex, unseen environments that bridges the Sim2Real gap without any sort of retraining.

TABLE I
NavACL Geometric Properties

Geodesic Distance	The shortest-path distance from s_0 to s_g
Path Complexity	The ratio of euclidean distance to geodesic distance of s_0, s_g
Turn Angle	The angle between the starting orientation and $\overrightarrow{s_0 s_g}$, represented as sine and cosine components
Agent/Goal Clearance	Distance from s_0 and s_g respectively to the nearest obstacle
Agent/Goal Island	Distance from the centroid of the navigation mesh to its furthest connected edge at s_0 and s_g respectively

II. Approach

Fig. 2 presents a flowchart of our contributions, which includes NavACL, the reward function for collision-avoidance, and the frozen feature networks. We discuss each piece in the following subsections.

A. NavACL

NavACL is based on evidence that intermediate difficulty tasks provide more learning signal than random tasks for policy improvement [30], [32] and that replaying easy tasks alleviates catastrophic forgetting [33]–[35]. NavACL filters down uniform random tasks to those that provide the most learning signal to the agent using predicted task success probability, described below.

Since our navigation problem has well-defined termination scenarios (agent reached the goal or not), we use binary task success as the signal metric. Let task $h = (s_0, s_g)$, with agent start position s_0 and goal position s_g. $f^*_\pi(h)$ denotes the probability of navigation policy π solving task h, zero for certain failure and one for certain success. We estimate task success probability f^*_π using a fully-connected deep neural network we call f_π. Before each forward pass, f_π preprocesses h into geometric properties (Table I), allowing f_π to generalize across scenes. We update f_π alongside π in the training loop (Algorithm 1, 2). We do not pretrain f_π – it learns only from rollouts produced by our agent, initially providing random tasks but quickly determining

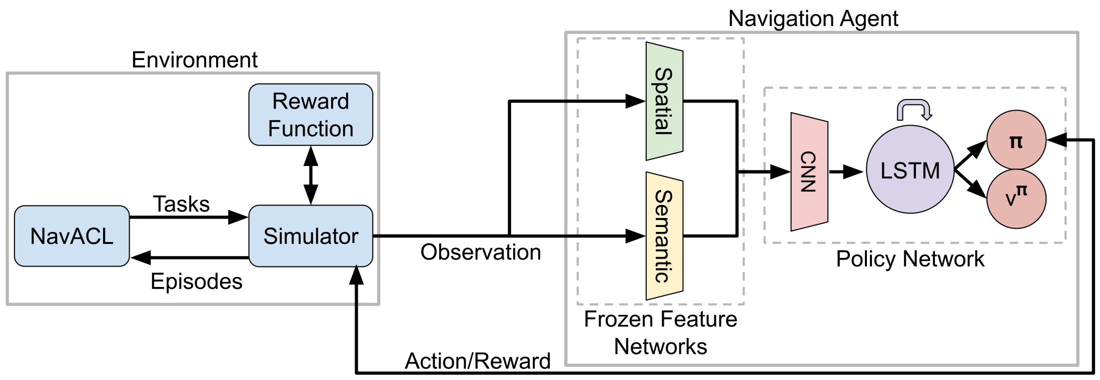


Fig. 2. Agent training pipeline, and how our contributions fit within it. We compress observations from the environment into latent features before passing them to the policy network. NavACL trains on navigation episodes and serves tasks back to the simulator.

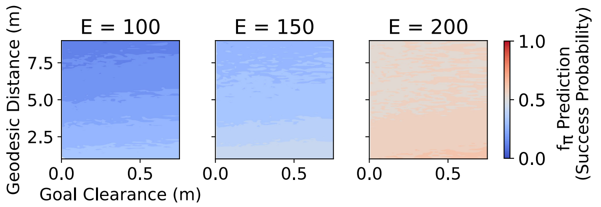


Fig. 3. Spatial interpolation of f_π across two geometric properties, at various training epochs E. Five thousand tasks are drawn randomly during each epoch and classified using f_π. Longer tasks (geodesic distance) are initially tough for the policy. As the policy improves, it can handle longer and harder tasks. Adaptive NavACL accounts for this, shifting its task distribution as the policy adapts.

regions of varying task difficulty. We define task *difficulty* as the complement of the estimated success probability, $1 - f_\pi(h)$. In contrast to [32] which formulates scenario generation with GANs for general frameworks, we optimize NavACL to generate scenarios efficiently for the navigation task using simple log loss.

Adaptive filtering: Now that we can estimate the difficulty of tasks, which tasks should we feed the agent? In one implementation, we produce *goals of intermediate difficulty* (GOID) [32], selecting tasks bounded between two success probabilities. GOID ensures we never select tasks that are too easy or too hard. However, GOID does not explicitly deal with catastrophic forgetting of easy tasks. Furthermore, the bounds do not change as the agent improves and task distribution shifts (Fig. 3). Instead, we provide a mixture of task types, where certain tasks *adapt* to the learner. *Easy* tasks provide adequate learning signal early in training and prevent catastrophic forgetting. *Frontier* tasks teach the agent to solve new tasks at its current ability. Uniformly sampled *random* tasks inject entropy and prevent overfitting to specific task types. Initially, easy and frontier tasks form the majority of the task mixture. The mixture decays into random sampling as the learning agent learns to generalize.

We draw many random tasks and estimate their difficulty using f_π, producing a difficulty estimate across the task space. We fit a normal distribution μ_f, σ_f to this distribution (Algorithm 1). μ_f, σ_f form an adaptive boundary in task space, partitioning it into easy and hard regions, predicated on policy π. In particular, task h is considered an *easy* task if $f_\pi(h) > \mu_f + \beta\sigma_f$ and a *frontier* task if $\mu_f - \gamma\sigma_f < f_\pi(h) < \mu_f + \gamma\sigma_f$, where β, γ are

Algorithm 1: Training Loop With f_π Update.

```
input  : $\emptyset$
output : $\pi$
$\pi, f_\pi, \mu_f, \sigma_f \leftarrow$ Init();
for $i \leftarrow 0$ to numEpochs do
    $tasks, successes, states, actions, rewards \leftarrow$ Rollouts($\pi, f_\pi, \mu_f, \sigma_f$);
    $\pi \leftarrow$ PPO($\pi, states, actions, rewards$);
    $f_\pi \leftarrow$ Train($f_\pi, tasks, successes$);
    $randomTasks \leftarrow$ GetRandomTasks();
    $\mu_f, \sigma_f \leftarrow$ FitNormal($f_\pi, randomTasks$);
return $\pi$;
```

Algorithm 2: Rollouts.

```
input  : $\pi, f_\pi; \mu_f; \sigma_f$;
output : $rollouts$
for $i \leftarrow 0$ to batchSize do
    $task \leftarrow$ GetDynamicTask($f_\pi, \mu_f, \sigma_f$);
    $rollouts[i] \leftarrow$ RunEpisode($\pi, task$);
return $rollouts$;
```

Algorithm 3: GetDynamicTask.

```
input  : Training timestep $t$; $f_\pi$; $\mu_f$; $\sigma_f$; Hyperparameters $\beta, \gamma$
output : Task $h$
$taskType \leftarrow$ GetTaskType($t$);
while true do
    $h \leftarrow$ RandomTask() ;
    switch $taskType$ do
        case easy do
            if $f_\pi(h) > \mu_f + \beta\sigma_f$ then
                return $h$;
        case frontier do
            if $\mu_f - \gamma\sigma_f < f_\pi(h) < \mu_f + \gamma\sigma_f$ then
                return $h$;
        case random do
            return $h$;
```

hyperparameters. In other words, task difficulty is relative to the current ability of the agent – if we expect π to do better on task h than an average task, it is easy. If h is near the difficulty of the average task, straddling the adaptive boundary, we call it a frontier task. Intuitively, this should provide a more conservative mixture of tasks than pure random sampling, promoting stable learning in difficult environments. The full algorithm is detailed in Algorithm 1–3.

B. Reward Shaping

Our reward function provides negative rewards to discourage collision and intrinsic rewards to encourage exploration and movement. We define it as:

$$r(s) = \mathbb{1}_{succ} + \delta(-\mathbb{1}_{coll} + \mathbb{1}_{expl}) + 0.01\, d. \tag{1}$$

The binary indicator $\mathbb{1}_{succ}$ is true upon reaching the target, and false otherwise. $\mathbb{1}_{coll}$ is true upon collision and false otherwise. The hyperparameter $0 < \delta < 1$ controls the agent's affinity for learning exploration and motor skills compared to target-seeking behavior. $\mathbb{1}_{expl}$ is an intrinsic reward for exploration. We keep a buffer of past agent positions over an episode, and provide a reward if the current position of the agent is some distance from all previous positions. We find that without the intrinsic exploration term, the agent falls into a local maxima of spinning in place to avoid the negative reward from collisions, which is difficult to escape. d is the distance traveled in the current step, expressing the prior that the agent should be trying to cover as large a search area as possible. We do not use a distance to target term, as others have shown binary success to be sufficient [8].

C. Frozen Feature Networks

Traditional visual DRL agents use an autoencoder to transform input RGB images into a latent representation, where the autoencoder is trained end-to-end with the policy network [36], [37]. End-to-end training can overfit the policy network to simulation artifacts, and hurt real-world transfer [27]. We use *spatial* autoencoders pretrained on real images [8] and freeze their weights to prevent overfitting to simulation renders during training. High-polygon meshes scanned by [11] and photorealistic renderers provided by [16] produce detailed enough visualizations to work with encoders trained on real-world datasets. Freezing also speeds up policy convergence, as the gradient backpropagates through fewer layers.

D. Semantic Target Network

We produce the semantic target feature using a Mask R-CNN with an FPN backbone trained on the COCO dataset [26], [38]. We introduce a small postprocessing layer that enables swapping target classes without retraining. Given an image, the Mask R-CNN predicts a binary mask M for each object class, along with its confidence. We extract the mask with target label l and do scalar multiplication of the binary mask with the prediction confidence to get output O.

$$O(x, y) = P(M(x, y)_{label=l}). \tag{2}$$

We can change l at runtime to search for different target classes. Pixels of O still contain shape information on the target object (*e.g.* a ball will produce a round mask but a box will produce a square one). To prevent the downstream policy from overfitting to one specific object shape, as well as reduce latent size, we apply a max-pool operation to O which is then stacked along with the other features into a latent representation, which is fed to the policy network.

TABLE II
PPO PARAMETERS

# of Minibatches	1	Learning Rate	0.005
Clipping Range (ϵ)	0.10	Discount Factor (γ)	0.99
Value Function Coef. (c_1)	0.5	Entropy Coef. (β or c_2)	0.01
Timesteps per Update	4000	Rollout Workers	12
Inner-Loop Epochs	4	GAE λ	0.95

III. Model Description

Our learner model consists of an actor-critic model with policy $\pi(s)$ and value function $V^{\pi}(s)$ optimized using proximal policy optimization (PPO) with clipping (Table II) [39], [40]. The policy network and value function take latent representations from the feature encoders as input, and produce an action and value estimate as output. The policy networks consist of feature-compression and memory sections. The feature-compression section compresses spatially-coherent latent features into a more compact representation using convolutional layers. Receiving features instead of full RGB images reduces time to train and the likelihood of overfitting to the simulator.

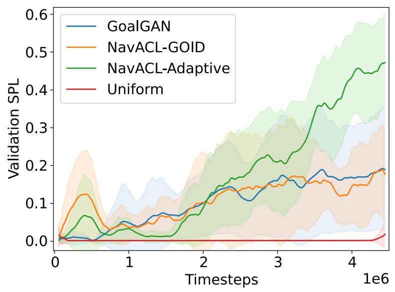
To keep the navigation problem Markovian, the state must contain information on where the agent has been, and if it has previously seen the target. The purpose of the memory section is to store this information. The memory section uses long short-term memory (LSTM) [41] cells to represent state in the partially observable Markov decision process (POMDP) [42]. With this, we aim to reduce the likelihood of revisiting previously explored areas and to remember the target location if it leaves the view. While rotating to circumvent obstacles, the agent may lose sight of the target.

IV. Experiments

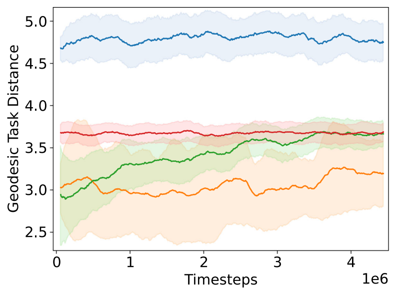
We present three experiments: an ablation study of NavACL, a simulation benchmark of our model on unseen environments and target objects, and a benchmark of our agent operating in the real world.

A. Evaluating NavACL

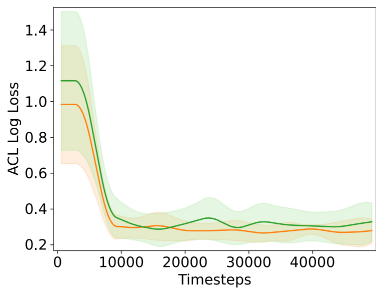
Our first experiment compares the impact of NavACL on visual navigation to GoalGAN as well as the current standard of uniform random task sampling. We evaluate NavACL with GOID (NavACL-GOID) and with adaptive filtering (NavACL-Adaptive). We hold all policy parameters the same, and run five navigation trials of five million samples on the Cooperstown environment from the Gibson dataset [11]. Uniform sampling uses Habitat's built-in task generator to generate tasks [16]. GoalGAN uses an intermediate difficulty value between 0.1 and 0.9, used in their MazeAnt navigation experiment. When GoalGAN selects points outside of the scene, we select the closest valid point to it within the scene. We do not pretrain GoalGAN on past experiments – we instead use the random initialization provided by the GoalGAN library, similar to NavACL initialization. For NavACL-GOID, we filter uniformly random tasks using our f_{π} framework, and target tasks with an intermediate difficulty value



(a) Validation SPL



(b) Task Selection



(c) NavACL Convergence

Fig. 4. (4a) Validation SPL over five trials on a single test-train environment. Solid lines and shaded areas refer to mean and one standard deviation, respectively. (4b) As the policy improves over time, NavACL increases the distance from start to goal – ratcheting up the task difficulty. (4c) NavACL's simple architecture provides valid predictions very quickly.

Fig. 5. Simulation test scenes, left to right: Cooperstown (40 m^2), Avonia (60 m^2), Hometown (61 m^2).

of $0.4 \leq f_\pi(h) \leq 0.6$. NavACL-Adaptive uses hyperparameters $\beta = 1, \gamma = 0.1$. For NavACL-Adaptive, `GetTaskType` uses random task probability $\min(0.15, t/t_{\max})$, where $t_{\max}$ is the maximum number of training timesteps. Frontier and easy tasks split the remaining probability equally. NavACL-Adaptive samples 100 tasks to estimate μ_f and σ_f. In our reward function, we use $\delta = 0.25, d = 0.2$.

NavACL-GOID performs on-par with GoalGAN, both outperforming uniform sampling. NavACL-Adaptive performs best, beating both NavACL-GOID and GoalGAN by a sizeable margin (Fig. 4). GoalGAN produces difficult tasks with long paths (Fig. 4(b)), suggesting it has not learned the spatial layout of the complex scene (*e.g.* where obstacles lie). Its performance being on-par with NavACL-GOID suggests that the adaptive portion of NavACL-Adaptive is responsible for the performance disparity.

B. Evaluating Model Performance

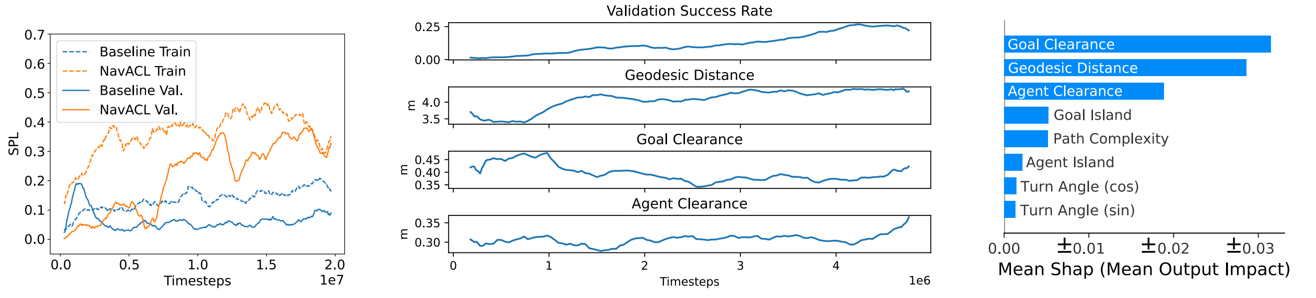
Using our methodology, we train a policy over twenty million timesteps using the Habitat 2019 challenge split of the Gibson dataset. Policies are evaluated over ten trials of 30 episodes spread across three unseen test environments, held out from the Habitat split (Fig. 5). The test tasks are randomly generated using the same uniform sampling as the Habitat challenge datasets [16]. The target object is an 11 cm radius football (soccer ball). All policies are limited to 150 moves, and all tasks have a maximum start to goal distance of 10 m. We find increasing the number of moves beyond 150 results in little improvement. The action space consists of a rotation of $\pm 30°$ and forward translation of 0.2 m.

The **Random** policy selects random actions to provide a lower bound on performance. The **PPO Baseline** policy is trained using depth, reshading (de-texturing and re-lighting), and semantic features, along with intrinsic rewards. The **NavACL** policy is trained identically to the PPO Baseline policy, with the addition of NavACL-Adaptive. **NavACL Target-Agnostic** is identical to NavACL, but during testing, we change the target from the ball to a large vase to evaluate target-agnostic semantic generalization to unseen targets of different shapes and sizes. We recruit ten volunteers from varying backgrounds to establish an upper-bound on performance. The **Human** policies are trained and tested just like the agent policies. The volunteers played the training set until they were comfortable with the controls, receiving the same RGB observations and action space as the agents. Once comfortable, the volunteers played through the same test set as the agents. We use the SPL metric defined by [21]

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{succ,i} \frac{l_i}{\max(l_i, p_i)} \tag{3}$$

where l is the length of the agent's path, p is the length of the shortest path from start to goal, and N is the number of episodes. Results are presented in Fig. 6 and Tabel III.

Our agents are able to find semantically specified targets in simulation, performing drastically better than random. Our NavACL methods improve upon PPO Baseline on average, and particularly in peak performance results of Cooperstown and Avonia. On Hometown, the NavACL agent is confused by a photoscanned mirror at the end of a long corridor, producing a mirage nearly 60 timesteps in length. The baseline policy behaves more randomly and falls for this trick less often. On unseen semantic classes, performance decreases slightly, but the policy shows target-agnostic semantic generalization capability. On average, humans outperform agents in unseen environments. However, humans can memorize test scenes during the first few episodes, giving them an advantage over agents during later episodes. On Cooperstown, agents are within one standard deviation of human-level performance in success rate, suggesting they outperform some humans in some cases. We found agents had trouble navigating to new spaces in larger, unseen environments. Agents did not have as much trouble when navigating



(a) SPL (b) Geometric Feature Development (c) Feature Impact

Fig. 6. (6a) Training and validation results of our model trained with and without NavACL. (6b) As the policy improves, NavACL produces harder tasks. Qualitatively, the agent learns obstacle avoidance motor skills before learning exploration behavior. Initially, NavACL selects shorter paths (short geodesic distance) in narrow corridors (low agent clearance) to guide the agent toward the goal. As the policy learns exploration strategies, the agent navigates without corridor guidance (high agent clearance), and even reaches goals behind obstacles (low goal clearance). (6c) The mean prediction impact each feature had on NavACL's task prediction. Mean Shapley values [43] were determined using 1800 background samples, and feature impact was meaned over 200 test samples at 5M timesteps. At 5M timesteps, the agent has learned basic motion, so placing the target behind the agent (turn angle) has little impact. Path complexity has a smaller than expected impact, suggesting the measure could be improved.

TABLE III
SIMULATION RESULTS OF COLLISION-FREE AGENTS OVER TEN TRIALS ON UNSEEN ENVIRONMENTS

Policy	Scene	Success Rate		SPL	
		μ	σ	μ	σ
Random	Cooperstown	0.07	0.00	0.06	0.00
	Avonia	0.01	0.00	0.01	0.00
	Hometown	0.00	0.00	0.00	0.00
	Total	**0.03**	**0.03**	**0.02**	**0.03**
PPO Baseline	Cooperstown	0.41	0.12	0.16	0.05
	Avonia	0.28	0.09	0.12	0.04
	Hometown	0.23	0.10	0.13	0.05
	Total	**0.31**	**0.13**	**0.14**	**0.05**
NavACL Target-Agnostic	Cooperstown	0.55	0.11	0.25	0.08
	Avonia	0.43	0.04	0.20	0.02
	Hometown	0.09	0.06	0.06	0.03
	Total	**0.36**	**0.21**	**0.17**	**0.10**
NavACL	Cooperstown	0.63	0.14	0.31	0.09
	Avonia	0.50	0.06	0.21	0.04
	Hometown	0.12	0.08	0.09	0.06
	Total	**0.42**	**0.19**	**0.24**	**0.09**
Human	Cooperstown	0.76	0.22	0.54	0.17
	Avonia	0.89	0.13	0.63	0.13
	Hometown	0.72	0.19	0.51	0.13
	Total	**0.79**	**0.20**	**0.56**	**0.15**

in large, previously-seen environments. Memory for embodied visual agents is an active area of research [19], [44]–[46], and we expect leveraging these memory modules will improve performance in larger environments. Another limitation was model throughput – it took roughly two weeks to train twenty million timesteps on a GPU machine, and previous experiments were still showing policy improvement at sixty million timesteps. With memory improvements and distributed computing, we believe our models could approach human performance.

C. Sim2Real Transfer

We transfer our policy without modification to a Turtlebot3 wheeled robot (AGV) and a DJI Tello quadrotor (UAV). The AGV uses wheel encoders for closed-loop control for motion primitives (single actions), but does not estimate odometry across actions. We tested the AGV on seven tasks spanning

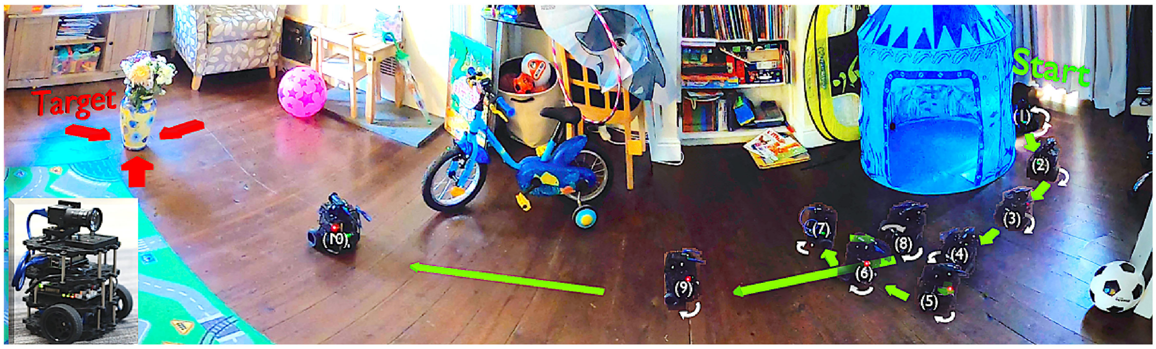
TABLE IV
REAL-WORLD RESULTS FROM THE AGV

Scene	Target	Task Dist. (m)	SPL
Office 1	Football	2.95	0.92
	Football	3.09	0.53
	Football	2.92	0.42
	Football	1.67	0.18
Office 2	Football	9.10	0.65
House	Orange Football	4.19	0.92
House	Vase	5.14	0.54

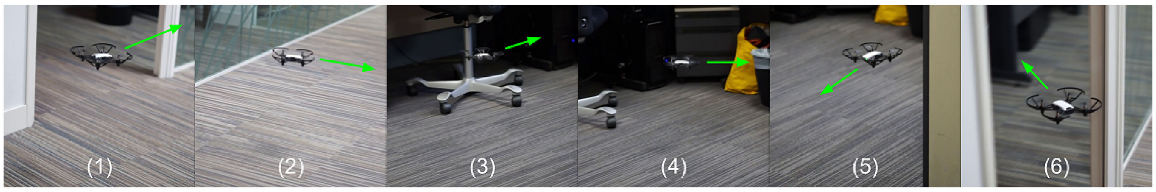
three environments and three objects, one being an unseen object and one being from an unseen semantic class. We use wheel odometry to measure SPL for the AGV (Table IV). The AGV did not experience a single collision over the 29 m it traveled during tests and was robust to actuator noise as well as wheel slip caused by terrain (hardwood, carpet, and rugs).

The UAV uses IR sensors to determine its height and an IMU to obtain very noisy position estimates for motion primitives and hovering stability. *We did not train a separate model for the UAV*. We used the model trained with the AGV height and camera field of view, which drastically differ from the UAV (0.2 m vs $\sim$ 0.75–1.5m and 68° vs 47° respectively). Policies trained for the AGV seemed surprisingly effective on the UAV, suggesting greater model generalization than we anticipated. The UAV was able to fly in-between legs of a camera tripod, through doorways, and even around moving people on many occasions without a single collision. Unfortunately, hover instability resulted in varying target height making target navigation unstable. While small changes in height were tolerated, larger differences were regarded as spurious detections. Still, this surprising generalization implies it may be possible to train a single navigation model for use on diverse types of robots that implement similar motion primitives. We provide video results of both the AGV and the UAV in the supplementary material,[1] and illustrations in Fig. 7.

[1] [Online]. Available: https://www.youtube.com/playlist?list=PLkG_dDkoI9pjPdOGyTec-sSu20pB7iayC



(a)



(b)

Fig. 7. (7a) The AGV navigating to the vase target in the house scene, with several never-before-seen obstacles littering the path to the target. Previous semantic targets (football and pink star ball) are present to emphasize target-agnostic semantic navigation capability. The agent turned 360° (1) to evaluate its options, then took the path between the desk and tent (2), adjusted the trajectory towards the wide-open area in front of the blue tent (3), and rotated 360° (4). The agent explored the areas surrounding the bike, bookshelves, and the blue tent (5, 6, 7). Target detection occurred at (8), and the AGV made a beeline for the target (9,10). (7b) While flying down a hallway (1), the UAV notices an empty cubicle (2). It threads the needle, flying between the chair wheels and seat (3). After exploring the cubicle (4, 5), it leaves and heads into the adjacent open office without collision (6).

V. Conclusion

We introduce NavACL and present two variants, NavACL-GOID and NavACL-Adaptive, for navigation task generation. Both methods significantly improve upon uniform sampling (the current standard) as well as GoalGAN in sparse-reward settings. Combining NavACL with frozen feature networks and collision-free policies produces agents capable of target-agnostic semantic navigation in simulation and the real world.

A. Future Work

We found LSTMs had issues with generalization to new environments with the compute power available to us. Future work will focus on integrating more structured and efficient memory modules [19], [44]–[46] into our learning pipeline.

The unexpected real-world generalization ability between mobility types warrants further investigation. Training an agent with an actuator abstraction layer allows transfer to disparate, never-before-seen robots. It may be prudent to invest computational resources in training a single model with abstract actuation that can be applied to drones, wheeled robots, walking robots, blimps, etc., rather than train each model individually.

We evaluated NavACL using on-policy reinforcement learning, but NavACL may be useful for selecting which episodes to replay when using off-policy methods. It may also prove useful in selecting and ordering training episodes for imitation learning.

References

[1] A. Kadian *et al.*, “Sim2Real predictivity: Does evaluation in simulation predict real-world performance?,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6670–6677, 2020.

[2] F. Xia *et al.*, “Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 713–720, Apr. 2020.

[3] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, “Active vision,” *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 333–356, 1988.

[4] C. Leung, S. Huang, and G. Dissanayake, “Active slam in structured environments,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 1898–1903.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[6] S. M. LaValle, “Rapidly-Exploring random Trees: A new tool for path planning,” Comput. Sci. Dept., Iowa State Univ., Tech. Rep., 1998.

[7] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Automat. CIRA’97.’Towards New Comput. Princ. Robot. Automat.*, 1997, pp. 146–151.

[8] A. Sax, J. O. Zhang, B. Emi, A. R. Zamir, L. J. Guibas, S. Savarese, and J. Malik, “Learning to navigate using mid-level visual priors,” in *Proc. Conf. Robot Learn.*, 2020, pp. 791–812.

[9] V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” in *Proc. NIPS Deep Learn. Workshop*, 2013.

[10] P. Mirowski *et al.*, “Learning to navigate in complex environments,” in *Proc. Int. Conf. Learn. Represent.*, 2017.

[11] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: Real-world perception for embodied agents,” in *Proc. Comput. Vis. Pattern Recognit., IEEE Conf.*, 2018, pp. 9068–9079.

[12] J. Straub *et al.*, “The replica dataset: A digital replica of indoor spaces,” Tech. Rep., 2019, *arXiv:1906.05797*.

[13] A. Chang *et al.*, “Matterport3d: Learning from rgb-d data in indoor environments,” in *Proc. 7th IEEE Int. Conf. 3D Vis.*, 3DV 2017. Inst. Elect. Electron. Engineers Inc., 2018, pp. 667–676.

[14] K. Mo, H. Li, Z. Lin, and J.-Y. Lee, “The adobeindoornav dataset: Towards deep reinforcement learning based real-world indoor robot visual navigation,” Tech. Rep., 2018, *arXiv:1802.08824*.

[15] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun, “Minos: Multimodal indoor simulator for navigation in complex environments,” Tech. Rep., 2017, *arXiv:1712.03931*.

[16] M. Savva *et al.*, “Habitat: A. platform for embodied ai research,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9339–9347.

[17] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, “Learning to explore using active neural slam,” 2020, *arXiv:2004.05155*.

[18] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2017, pp. 2371–2378.

[19] L. Mezghani, S. Sukhbaatar, A. Szlam, A. Joulin, and P. Bojanowski, "Learning to visually navigate in photorealistic environments without any supervision," Tech. Rep., 2020, *arXiv:2004.04954*.

[20] Y. Wu, Y. Wu, A. Tamar, S. Russell, G. Gkioxari, and Y. Tian, "Bayesian relational memory for semantic visual navigation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 2769–2779.

[21] P. Anderson *et al.*, "On evaluation of embodied navigation agents," Tech. Rep., 2018, *arXiv:1807.06757*.

[22] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat*., 2017, pp. 3357–3364.

[23] F. Hill, S. Mokra, N. Wong, and T. Harley, "Human instruction-following with deep reinforcement learning via transfer-learning from text," *CoRR*, vol. abs/2005.09382, 2020.

[24] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *Proc. IEEE Int. Conf. Robot. Automat*., 2019, pp. 8846–8852.

[25] X. Wang *et al.*, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6629–6638.

[26] T.-Y. Lin *et al.*, "Microsoft coco: common objects in context," in *Proc. Euro. Conf. Comput. Vis.*, Berlin, Germany: Springer, 2014, pp. 740–755.

[27] D. Gordon, A. Kadian, D. Parikh, J. Hoffman, and D. Batra, "Splitnet: Sim2sim and task2task transfer for embodied visual navigation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1022–1031.

[28] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Proc. Robot. Sci. Syst.*, Cambridge, MA, USA, Jul. 2017, doi: 10.15607/RSS.2017.XIII.034.

[29] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.

[30] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic curriculum learning for deep RL: A short survey," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 4819–4825.

[31] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *Proc. 6th Int. Conf. Learn. Represent.*, 2018.

[32] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1515–1528.

[33] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proc. 30nd AAAI Conf. Artif. Intell.*, 2018.

[34] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Adv. Neural Inf. Process. Sys.*, 2019, pp. 348–358.

[35] L. A. Atherton, D. Dupret, and J. R. Mellor, "Memory trace replay: The shaping of memory consolidation by neuromodulation," *Trends Neurosci.*, vol. 38, no. 9, pp. 560–570, 2015.

[36] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," *Advances Neural Info. Proc. Syst. 31*, 2018, pp. 2451–2463.

[37] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 512–519.

[38] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron 2," 2019. [Online]. Available:https://github.com/facebookresearch/detectron2.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[40] A. Hill *et al.*, "Stable Baselines," 2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[43] S. M. Lundberg *et al.*, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[44] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2616–2625.

[45] S. Gupta, D. Fouhey, S. Levine, and J. Malik, "Unifying map and landmark 526 based representations for visual navigation," Tech. Rep., 2017, *arXiv:1712.08125*.

[46] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *Proc. Int. Conf. Learn. Representations*, 2018.