Multi-Object Navigation in real environments using hybrid policies

Assem Sadek[1], Guillaume Bono[1], Boris Chidlovskii[1], Atilla Baskurt[2] and Christian Wolf[1]

Abstract— **Navigation has been classically solved in robotics through the combination of SLAM and planning. More recently, beyond waypoint planning, problems involving significant components of (visual) high-level reasoning have been explored in simulated environments, mostly addressed with large-scale machine learning, in particular RL, offline-RL or imitation learning. These methods require the agent to learn various skills like local planning, mapping objects and querying the learned spatial representations. In contrast to simpler tasks like waypoint planning (PointGoal), for these more complex tasks the current state-of-the-art models have been thoroughly evaluated in simulation but, to our best knowledge, not yet in real environments.**

In this work we focus on sim2real transfer. We target the challenging Multi-Object Navigation (Multi-ON) task [41] and port it to a physical environment containing real replicas of the originally virtual Multi-ON objects. We introduce a hybrid navigation method, which decomposes the problem into two different skills: (1) waypoint navigation is addressed with classical SLAM combined with a symbolic planner, whereas (2) exploration, semantic mapping and goal retrieval are dealt with deep neural networks trained with a combination of supervised learning and RL. We show the advantages of this approach compared to end-to-end methods both in simulation and a real environment and outperform the SOTA for this task [28].

I. INTRODUCTION

Robot navigation has progressed from simple waypoint navigation problems in richly prepared environments, for which robustly working systems are now used in production, to complex tasks involving high-level reasoning with visual and semantic concepts. This has been made possible through large-scale machine learning, mostly in photo-realistic simulators and reinforcement learning from billions of interactions. The resulting robotic agents, implemented as high-capacity neural networks, which are however subject to performance drop and lack of robustness when the policies are transferred from simulation to real environments with physical robots. This is mostly due to the gap in realism between simulation and reality ("sim2real gap"), as well as the difficulty to explore a large amount of variation factors inherent in navigation problems, such as room layouts, furniture, textures and other room details, rare local scene geometries etc.

There is a recent trend towards modular approaches which decompose the problem into hierarchical parts [8], [4] and hybrid approaches, which combine a shortest path planner (symbolic or trained) with a trained policy. While these

[1]Naver Labs Europe, Meylan, France. firstname.lastname@naverlabs.com
[2]Université de Lyon, INSA-Lyon, LIRIS, Villeurbanne, France. atilla.baskurt@insa-lyon.fr

Fig. 1. We perform Multi-Object Navigation [41], i.e. the sequential visual search of multiple object in a given order, and are the first do this in real physical environments (a) characterized by a large sim2real gap. This is illustrated by two first-person views (b), real and (c), simulation. We propose a hybrid method combining classical mapping and deep learning, and compare to the SOTA methods on this task using end-to-end RL training and auxiliary losses [28].

approaches have been shown to be more sample efficient [32], state-of-the-art methods are still evaluated in simulation and lack thorough tests on real robots.

In this work we address the challenging problem of Multi-Object Navigation [41], which, similarly to the K-items scenario [5], requires an agent to sequentially navigate through a set of objects in an imposed order. This task definition favors agents capable of learning to map seen objects in an internal spatial representation, as navigating to them later in the episode can increase reward. This makes it stand out with respect to simpler tasks like ObjectNav, where the combined capacities of exploration and reactive local planning from the current observation are sufficient to solve the task[1].

We target sim2real transfer and, to our best knowledge, are the first to perform a thorough performance evaluation of a method on Multi-Object Navigation in a real physical environment, see Figure 1. While simpler tasks, such as PointGoal, have been evaluated on real robots [23], [34], evaluation of trained models on more complex tasks has been sparse or nonexistent. We present a new method for navigation, whose design choices have been driven by the objective of optimizing performance in real environments. We propose a new hybrid method which decomposes the problem into two parts:

① **"Good Old Fashioned Robotics"(GOFR)**, that deals with classical navigation aspects not related to semantics, such as detection of navigable space and localization

---

[1]Regularities in spatial layouts may be exploited with an additional form of higher reasoning, for instance with potential fields [32], but we do not focus on these aspects in this work.
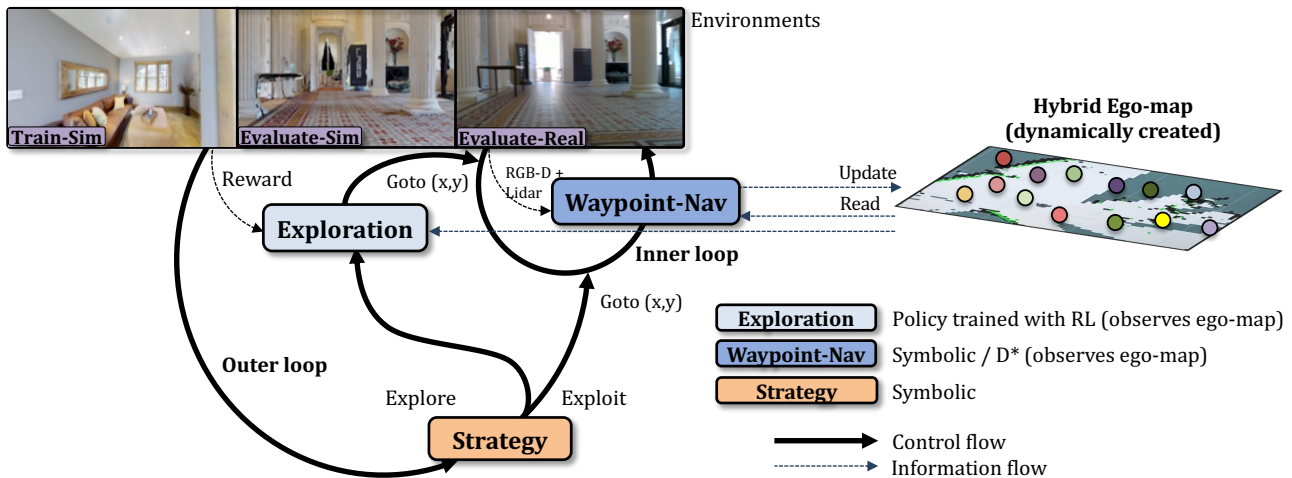
Fig. 2. An agent for multi-object navigation maintains a hybrid representation consisting of a metric bird's eye view map combined with a semantic point cloud. The agent switches between a trained exploration policy and symbolic waypoint selection, deferring low-level actions to a symbolic planner.

(geometric SLAM) combined with waypoint navigation on this map.

② **Semantics through Machine Learning**, i.e. mapping semantic concepts required for visual reasoning and exploiting them; exploration of the most promising areas of the environment exploiting layout regularities.

During navigation, a classical SLAM algorithm [24] creates and maintains a 2D metric representation in the form of a tensor/map and localizes the robot on it using Lidar input. High-level features, extracted from visual RGB-D observations with a deep neural network, form a spatial and semantic point cloud, whose spatial coordinates are aligned with the metric representation, see Figure 2. The combined hybrid representation satisfies the needs of relevant sub-skills the agent requires: (i) to determine whether a target object has been observed in the past, (ii) to plan optimal trajectories between the agent and explored areas, and (iii) to determine the frontiers of unexplored areas in the environment and thus the next intermediate sub-goals in case the environment needs to explore to find the next goal. All these sub-skills are designed and trained separately, which allows to limit sample complexity of training.

The contributions of this work are the following: (i) we introduce a hybrid method for Multi-Object navigation combining classical metric SLAM and path planning with learned components trained with supervised learning and RL; (ii) we reproduce the Multi-ON benchmark [41] in a real environment, where we place manufactured reproductions of the goal objects, used in originally simulated target environment; (iii) we compare the proposed method to end-to-end trained methods in this real environment, in particular with the winning entry of the CVPR 2021 Multi-ON competition [28], which we outperform in both real and simulated environments.

II. RELATED WORK

Modular Embodied Navigation — Casting a task as an end-to-end learning problem is widely used in CV and NLP.

Navigation has been addressed through this lens early on, e.g. for exploration [15], where a neural policy processes raw sensory observations and directly predicts agent actions. However, jointly learning mapping, state-estimation and path-planning purely from data has been shown to be expensive [8]. An alternative are hierarchical and hybrid architectures [4], [8], [7], [9] that compose a learned mapper with global and local policies, where all components interface via the map and an analytical path planner. Pushing modularity further, Ramakrishnan *et al.* [32] propose to disentangle the skills of '*where to look?*' from navigation itself. The network predicts potential functions conditioned on a semantic map and uses them to decide where to look for an unseen object.

Sim2Real — The sim2real gap can compromise strong performance achieved in simulation when agents are tested in the real-world [21], as perception and control policies often do not generalize well to real robots due to inaccuracies in modelling, simplifications and biases. To close the gap, domain randomization methods [31], [38] treat the discrepancy between the domains as variability in the simulation parameters. Alternatively, domain adaptation methods learn an invariant mapping for matching distributions between the simulator and the robot environment. Examples include transfer of visuo-motor policies by adversarial learning [42], adapting dynamics in RL [16], and adapting object recognition to new domains [43]. Bi-directional adaptation is proposed in [40]; Recently, Chattopadhyay *et al.* [11] benchmarked the robustness of embodied agents to visual and dynamics corruptions. Kadian *et al.* [23] investigated sim2real predictability of Habitat-Sim [36] for PointGoal navigation and proposed a new metric to quantify it, called Sim-vs-Real Correlation Coefficient (SRCC). The PointGoal task on real robots is also evaluated in [34]. To reach competitive performance without modelling any sim2real transfer, the agent is pre-trained on a wide variety of environments and then fine-tuned on a simulated version of the target environment.

Memory and Maps (Inductive Bias) — Memory is a crucial aspect of an intelligent agent's ability to reason about 3D

space and geometry. Neural memories like NeuralMap [29], MapNet [20] and propose latent metric maps, which are updated incrementally from the camera observations and odometry and act as inductive bias for end-to-end training. EgoMap [4] augments these maps with multi-step objectives and attention reads, trained with RL. Alternative neural maps are topological maps [35], [10], [4], transformers [30], [17], [13], [22], [14], [33], which break the Markovian assumption and attend to a large temporal horizon and implicit representations [25], [2].

Exploration — is at the core of all navigation tasks [8] and is in itself studied and evaluated [3]. Efficiently visiting the environment is useful for solving tasks in known environments and pre-mapping in unknown ones. Chen *et al.* [15] explored policies with spatial memory that are bootstrapped with imitation learning and finetuned with coverage reward. In [9], an exploration policy is trained by introducing semantic curiosity based on observation consistency. SEAL [6] trains perception models on internet images to learn an active exploration policy. They build 3D semantic maps to learn both action and perception models, and integrate intrinsic motivation. Episodic semantic maps are proposed in [7].

III. Hybrid planning and navigation

We target the task of Multi Object Navigation (Multi-ON) introduced by Wani *et al.* [41], in particular the 3 object variant: during each episode, the agent has to find 3 cylindrical objects G_n, $n = 1, 2, 3$, in a pre-defined order, where G_n is the nth object to find, and is required to call the *Found* action at each goal. The episode duration is limited to 2,500 environment steps. At each step t, the agent receives an egocentric RGB-D observation $\mathbf{O}_t \in \mathbb{R}^{h \times w \times 4}$, a Lidar frame, and the class label of the current target object taken from 8 classes. All training was performed in simulation only with the Habitat simulator [36], but the system was evaluated, both, in simulation and on a real Locobot robot in a real environment, more details are given in Section IV.

With operations on robots in real environment and conditions in mind, we follow a modular approach, outlined in Figure 2. The method is hybrid; it leverages both trained neural modules for perception and exploration, and classical algorithms for occupancy mapping, localization and waypoint navigation. The main motivation behind this approach is a maximum reduction of the sim2real gap, avoiding the main pitfalls of end-to-end training of navigation in simulation followed by a transfer of neural models to the real environment. We explore an approach that prefers classical methods based on sensor models and optimization, motivated by their robustness, and employ machine learning in a targeted way for parts of the system where its use is both necessary and beneficial. We also limit input to trained models to representations with a potentially low sim2real gap. For this reason, during navigation the agent builds a metric bird's eye view occupancy map from the Lidar input and localizes itself on it using metric SLAM [39]. This binary map is combined with an overlaid semantic point cloud, which contains the positions of key objects and their semantic

classes, which are detected from the RGB input with an object detector. Detection and mapping are aligned through the SLAM algorithm's localization module.

Navigation — is performed hierarchically on two different levels. On a higher level (outer loop in Figure 2), 2D waypoint coordinates $\mathbf{p}_t = (x, y)$ are produced and provided to the lower level controller (inner loop), whose task is to navigate to the waypoint using the maintained occupancy map. The high-level controller switches between two different strategies:

① **Exploration** — when the target object has not yet been observed, i.e. the robot explores the environment, maximizing coverage. This is done with a learned policy trained with RL, see below.

② **Exploitation** — when the target object has been observed and thus is part of the semantic point cloud, its location is taken as a new waypoint and given to the local planner.

Metric EgoMap — To gather navigability information along its path and more efficiently revisit previously seen areas, the agent builds what is called an EgoMap, an occupancy grid of fixed spatial resolution centered on its current position and aligned with its heading direction.

On the real robot, this map is obtained using the RTABMap [24] library. It uses a graph-based SLAM algorithm with loop closure, a flexible design taking advantage of RGB-D, Lidar and odometry sensor data. Lidar and/or depth are used to create a 2D/3D local occupancy grid, associated to a node whose initial position relies on odometry integration. Descriptors are then created from keypoints extracted from RGB frames in order to facilitate node comparison and loop closure detection. RTABMap also includes short- and long-term memory management, global map compression and multi-session mapping.

In simulation, we take advantage of privileged information to retrieve a complete top-down view of the scene navigability, through a projection of the NavMesh generated by the Recast&Detour [1] library in Habitat-Sim. A fog-of-war mask is then built by ray-tracing in the agent's field of view directly on this top-down view using perfect localization.

Both real and sim approaches generate a global map on which we apply a simple affine transformation parameterized by the agent's current pose to get the EgoMap.

Exploration — is the main module based on machine learning. In contrast to most recent work in embodied AI [4], [8], the policy does not take the first person RGB input, but the EgoMap $\mathbf{M}_t$ produced by the metric SLAM algorithm. This leads to a significant simplification of the task and increased sample efficiency, and it minimizes the sim2real gap, as changes in lightning, color and texture are avoided. The policy is a part of the outer loop and predicts 2D waypoint coordinates $\mathbf{p}_t$. The problem is partially observable for multiple reasons: (i) not all areas of the scene have been observed at any point in time; (ii) for efficiency reasons, the EgoMap $\mathbf{M}_t$ does not cover the full scene, observed areas can therefore be forgotten when the agent navigates sufficiently far away from them; (iii) even theoretically fully observable problems (MDPs) can
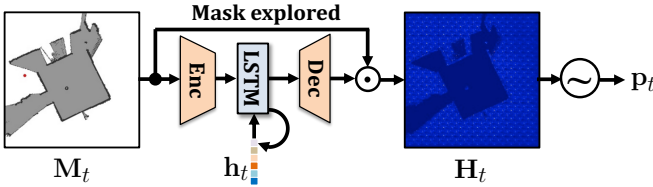
Fig. 3. The exploration policy takes as input EgoMaps $\mathbf{M}_t$ and predicts a heatmap, which is limited/masked ($\odot$) to unexplored areas. The next waypoint $\mathbf{p}_t$ is sampled ($\sim$) from the resulting map $\mathbf{H}_t$.

| Method | — LIDAR Usage — | | Aux | Obj. |
	Map	Low-lev cntrl	losses[28]	Segm.
ProjNMap+AUX [28]	–	✓	✓	–
Ours	✓	✓	–	✓

TABLE I

COMPARABILITY OF THE DIFFERENT METHODS IN TERMS OF SENSOR AND INFORMATION AVAILABILITY. BOTH METHODS USE LIDAR.

be transformed into POMDPs ("*Epistemic POMDPs*") in the presence of uncertainty in the environment, which is a standard case in robotics, as has recently been shown in [18]. We therefore imbued the policy with hidden memory $\mathbf{h}_t$ and made it recurrent.

The policy π needs to be able to predict multi-modal distributions, as there are multiple valid trajectories exploring an environment efficiently. We baked this into the policy through an inductive bias, which forces prediction to pass through a spatial heatmap $\mathbf{H}_t$, from which the chosen waypoint location is sampled. Before sampling, we restrict the heatmap to unexplored areas through masking. This choice also leads to a more interpretable model, as the distribution of targeted exploration points can be visualized (see Section IV). This can be formalized as follows (see also Figure 3):

$$\mathbf{h}_t = \phi(\mathbf{M}_t, \mathbf{h}_{t-1}; \theta_\phi) \tag{1}$$
$$\mathbf{H}'_t = \pi(\mathbf{h}_t; \theta_\pi), \ \mathbf{H}_t = \mathbf{H}'_t \odot [M_t == \text{"Unexplored"}]) \tag{2}$$
$$\mathbf{p}_t = \sim (\mathbf{H}_t), \tag{3}$$

where θ_π and θ_ϕ are trainable parameters and ϕ is the update recurrent function of an LSTM with hidden state $\mathbf{h}_t$; gates have been omitted in the notation for simplicity. Here, $\mathbf{p}_t = (x_t, y_t)$ is 2D coordinates of the point sampled in the spatial heatmap $\mathbf{H}_t$, '$\sim$' is the sampling operator.

We train the exploration policy with RL to maximize coverage and use the following reward function r_t:

$$r_t = \alpha r_{de}, \quad r_{de} = e_t - e_{t-1}, \tag{4}$$

where e_t denotes the explored area at step t, l is the number of inner environment steps necessary to navigate to the coordinates (x, y) predicted by the policy, α is a scaling hyper-parameter set to 0.01.

Local navigation — to the waypoint $\mathbf{p}_t$ is performed by an analytical planner that computes the shortest path on the current occupancy EgoMap $\mathbf{M}_t$. This is not necessarily the optimal path, as the map is not equal to the (unobserved) GT map and the intermediate regions to be traversed (and even the waypoint $\mathbf{p}_t$) might be unexplored. We employ a dynamic planner D* which calculates the shortest path under classical assumptions and replans when new information is available. Since we optimize our method to be robust and efficient in real environments, unlike recent work [32], [7] we choose a D* planner over the commonly used Fast Marching Method [37]. The path feasibility in real and the speed of planning are the two main reasons behind this design choice.

Stabilizing training — The potential failures and sub-optimal trajectories produced by local planning in uncertain conditions using D*, as described above, also negatively impact the training process of the exploration policy. This policy is a part of the outer loop and predicts waypoints $\mathbf{p}_t$, receiving a reward only upon completion of the full local navigation process. Noise in local planning impacts the stability of the RL training process and leads to lack of convergence.

We solved this by training the exploration policy interfacing a local policy on which we imposed a length limit. The full trajectory from the current position to the next waypoint $\mathbf{p}_t$ predicted by the exploration policy is split into a sequence of small sub goals distanced by 0.3m, and the local policy is limited to 5 of these sub goals. Control is given back to the outer loop if the waypoint $\mathbf{p}_t$ has been reached, or the limit of 5 subgoals is reached. This choice lead to stable training and the trained policy transferred well to the targeted exploration task, without changes. The same limitation on the length of local planning is also applied at deployment, which led to improved robustness in real conditions and makes complex recovery behavior obsolete.

Object Detection and Mapping — is framed as a semantic segmentation task from the current RGB-D frame o_t, which we supervise from GT masks calculated from privileged information in the simulator. The predictor is a DeepLab v3 network [12], detected objects in the mask are inversely projected and aligned with the EgoMap using depth information and the episodic odometry. Note that both, depth and odometry, are noisy in the real robot / real environment evaluation settings.

High-level decisions — are fully handcrafted, as this leads to a robust and transferable decision process where learning is arguably not required. Given our decision choices, only one type of decision is required, whether to perform exploration or exploitation (i.e. navigation towards the goal). This is taken on the basis whether the current target object has been observed at mapped, or not. If multiple objects of the same class have been detected, the location with the most probably detection (in terms of segmented object pixels) is chosen. A minimum number of pixels is required for an object to be mapped.

IV. EXPERIMENTAL RESULTS

Simulation — for training in simulation and for additional evaluation (complementary to experiments on the real platform) we used the photo-realistic Habitat simulator [36] and two datasets with 3D scanned environments, each with the

standard train/validation/test split: (i) the Gibson dataset (120 scenes) and the Matterport 3D dataset (90 scenes).

Real environment — To evaluate the method in a real environment, we used a LoCoBot robot [26] [🦾] equipped with an Intel *RealSense* RGB-D camera and a single-ray Lidar of type *RPLIDAR A2M8* (see Figure 1), which we restrict to Field-Of-View equivalent to the RGB camera. We used the publicly available *habitat_sim2real* library [19], which allows to connect a real robot under ROS to the Habitat simulator as an agent. We perform tests in a building with classical architecture covering one large conference room and several adjoining rooms, cf. Figure 1, and for a map, Figure 5. The environment features difficult conditions including windows and glass panels, thick carpets, textureless walls, etc.

Baselines — We compared with two end-to-end methods:

- **ProjNMap** (Projective Neural Map) is based on projective neural memory as inductive bias for neural networks [20] and has been explored for the specific task of Multi-ON in [41], providing the best results in the original experiments when the task was introduced.
- **ProjNMap+AUX** combines the original ProjNMap model with auxiliary losses on an additional head, which predicts the direction and distance to the current target during training [28]. This method achieved the winning performance in the CVPR 2021 M-ON Challenge [27] and is the current state-of-the-art on this task.

Table I summarizes sensor usage and information availability of the main baseline [28] compared to our method. All methods use Lidar: ours - to maintain a metric map, the baseline [28] - for a localization step necessary to perform closed-loop low-level control, mapping the discrete action space of the neural agent to the continuous motor space of the robot. All methods use information on object positions *during training*: ours - in the pre-training step of the visual encoder, the baseline [28] - through auxiliary losses.

Configurations — With different evaluation goals in mind, we created two configurations of the agent:

- **Locobot**: this configuration corresponds to the physical robot (Locobot) and its sensors. We also created a corresponding Habitat simulator configuration, which is equal to these settings: FOV of 56° (camera+Lidar), frame size of 160×120 and a compatible camera position. This configuration is of double use, i.e. can be used for evaluation in both simulation and the real environment.
- **Multi-ON-Compat**: we also reused the settings of the Multi-ON benchmark, making this configuration compatible with prior work like [28], [41]. This includes a FOV of 79° and camera frames of size 256×256. This configuration can be used in simulation only.

Setup and hyper-parameters — decision thresholds are set as follows: at least 0.7% of detected pixels is required for an object to be placed on a map; 5% or more of detected pixels is required for an object to trigger the *Found* action.

Results in Simulation — are shown in Table II. They have been obtained on two different datasets: (i) on the validation split of the Matterport 3D dataset, making these
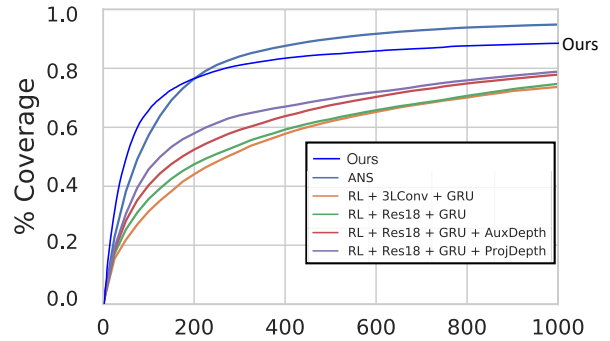

Fig. 4. Coverage (%) obtained by the exploration policy as a function of episode length (the number of simulation steps), compared to ANS [8] and end-to-end RL baselines using egocentric input taken from [8] on Gibson/Val.

runs comparable to the validation entries of the CVPR 2021 Multi-ON competition, and (ii) on 10 episodes in a 3D scanned version of our real environment, shown in Figure 2. These 10 episodes correspond to simulated versions of the episodes tested in the real environment, see further below. This simulated environment has *not* been used for training.

While our main design choices are biased toward a robust performance in real environments, we can see that it is also highly performant in simulation. In the LoCo Setup, where the (virtual) sensor configuration mirrors real sensors, the method outperforms the state of the art on the *Progress* and *Success* metrics and is competitive in the others.

Results in the real environment — are shown in Table III, on the same 10 episodes that we tested in simulation. Our hybrid agent was able to collect 43% of the targets successfully and even finished 2 out of 10 episodes retrieving all 3 required items. We conjecture that this is due to the strategy to disentangle perception, exploration, and waypoint navigation, which allows for keeping the sim2real gap lower than what can be done for the end-to-end (E2E) trained methods.

On the other hand, the performance of the baselines can be considered a failure. While it has been reported, that end-to-end training of the simpler PointGoal task in similar conditions can be successful [34], this did not apply to our experiences on the much more complex Multi-ON task. While [28] obtained the state of the art in the official benchmark, i.e. in simulation, not a single episode was successful in the real environment. Most failure cases were related to the poor exploration of the scene and high uncertainty in detecting the targets. The agent had a hard time changing rooms and repeatedly failed to apply 'Found' when the object is closely upfront. We conjecture that the high impact of the sim2real gap on raw sensor data(cf Figure 1) requires to adapt the E2E methods on real data for such complex tasks. Although training with real data is time-consuming, we believe that finetuning the E2E agents on offline real data with behavioral cloning is possible and would enhance the performance in upcoming real experiments.

Exploration performance — is provided as complementary information in Figure 4. On this task, we compare with *Active Neural SLAM* by Chaplot *et al*. [8]. We outperform it on the first 200 steps, making our method more robust for limited

Dataset	Agent	M-ON Setup				LoCo Setup			
		Progress	**PPL**	**Success**	**SPL**	**Progress**	**PPL**	**Success**	**SPL**
MP3D	ProjNMap	41.63	21.81	23.10	14.41	35.47	**23.97**	19.10	15.49
	ProjNMap+AUX [28]	**62.47**	**35.21**	**48.20**	**62.47**	59.97	23.29	33.40	**19.75**
	Hybrid (Ours)	55.23	12.41	41.80	11.72	**63.37**	19.72	**50.80**	18.44
10 episodes	ProjNMap	23.33	14.39	10.00	9.21	16.67	**13.63**	10.00	9.52
	ProjNMap+AUX [28]	43.33	**28.19**	30.00	**24.88**	40.00	12.77	30.00	10.65
	Hybrid (Ours)	**50.00**	9.20	**50.00**	9.20	**56.67**	12.09	**50.00**	**11.99**

"M-ON": CVPR 2021 Multi-On challenge sensor settings: FoV=79°, RGB size=256×256.
"LoCo": Sensor settings equivalent to the physical robot: FoV=56°, RGB size=160×120.

TABLE II

PERFORMANCE IN SIMULATION (HABITAT) FOR TWO DIFFERENT ENVIRONMENTS: THE MATTERPORT 3D VALIDATION SET (COMPARABLE WITH THE CVPR 2021 MULTI-ON CHALLENGE), AND 10 TEST EPISODES OF THE SIMULATED VERSION OF OUR REAL ENVIRONMENT.

Agent	**Progress**	**PPL**	**Success**	**SPL**
ProjNMap	3.00	0.85	0.00	0.00
ProjNMap+AUX [28]	0.00	0.00	0.00	0.00
Hybrid (Ours)	43.10	6.02	20.00	4.99

TABLE III

PERFORMANCE IN THE **REAL ENVIRONMENT** BY THE PHYSICAL LOCOBOT ON 10 TEST EPISODES. WE COMPARE WITH THE CVPR 2021 MULTI-ON CHALLENGE WINNER [28] (CURRENT SOTA).

Agent	Coverage (%)	
	Gibson	MP3D (domain generaliz.)
RL+3LConv+GRU	73.7	33.2
RL+Res18+GRU	74.7	34.1
RL+Res18+GRU+AuxDepth	77.9	35.6
RL+Res18+GRU+ProjDepth	78.9	37.8
ANS [8]	**94.8**	52.1
Ours	88.4	**67.14**

TABLE IV

COVERAGE OBTAINED BY DIFFERENT EXPLORATION POLICIES ON GIBSON AND MP3D. ALL AGENTS WERE TRAINED ON THE GIBSON TRAIN SPLIT (RESULTS ON COMPETING METHODS TAKEN FROM [8]).

Fig. 5. A rollout of an episode with the hybrid model. From left to right: (1) RGB observation; (2) GT map with the GT goal positions ▪▪▪, the current agent position ◁, the current waypoint p_t ▪; (3) EgoMap M_t with the planned local path and (4) a zoomed version. The initial goal is blue. At $t=6$, an exploration goal is predicted. The agent enters a new room, and at $t=17$ it detects the **blue goal** and switches to exploitation mode advancing towards it. At $t=23$, it observes the very dark **green goal** and maps it for future use. A false positive example (white cylinder) was also detected.

Fig. 6. For a given time step, we plot the predicted spatial heatmaps $\mathbf{H}_t$ for different training checkpoints, after 0,1,2,5 and 10 million updates. The lowest and highest probabilities are in dark blue and red, respectively.

time-budget exploration. More importantly, given our main objective of optimizing sim2real performance, we outperform the state-of-the-art on domain generalization by +15% margin, as shown in Table IV.

Qualitative results — are given in Figure 5, which shows a rollout for a single episode. We see that the agent first explores the scene, observes the goal quickly, and switches to exploitation mode. While navigating to the first goal, it observes a potential future goal and correctly maps it. Some drawbacks of the hybrid agent are presented in the detection of false positives such as the white cylinder.

In Figure 6, we visualize, for a chosen episode in a given environment step, the evolution of the predicted heatmaps $\mathbf{H}_t$ as training evolves (hence, different checkpoints). Heatmaps correspond to high-entropy distributions at the beginning, with high uncertainty on exploration targets. As training goes on, the distribution gets narrower and peaky, making the predictions more and more certain.
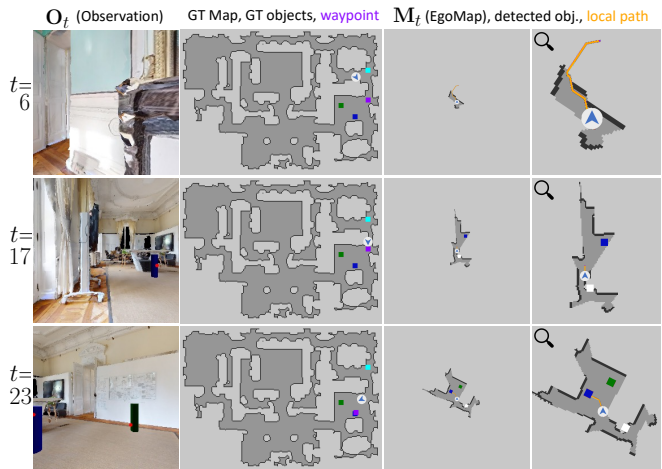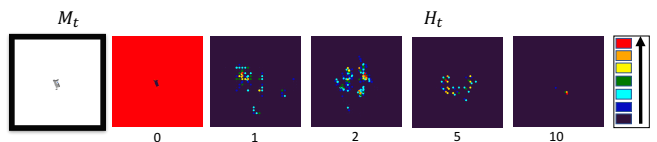
V. CONCLUSION

We have extended the Multi-ON task to real environments and up to our knowledge we present the first experimental evaluation of this task in these settings. We have introduced a hybrid model, which disentangles waypoint planning and semantics, significantly decreases the sim2real gap, and outperforms E2E trained models which were the current SOTA in simulation. Future work will focus on the enhancement of the handcrafted high-level strategy, which needs to be robust to the false positive detections, a common challenge also in other navigation tasks (e.g. ObjectNav).

REFERENCES

[1] Recast & Detour library. https://github.com/recastnavigation/recastnavigation.

[2] Vision-Only Robot Navigation in a Neural Radiance World. *IEEE Robotics and Automation Letters*.

[3] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018.

[4] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf. Learning to reason on uncertain topological maps. In *European Conference on Computer Vision (ECCV)*, 2020.

[5] E. Beeching, C. Wolf, J. Dibangoye, and O. Simonin. Deep Reinforcement Learning on a Budget: 3D Control and Reasoning Without a Supercomputer. In *ICPR*, 2020.

[6] Devendra Singh Chaplot, Murtaza Dalal, Saurabh Gupta, Jitendra Malik, and Ruslan Salakhutdinov. SEAL: self-supervised embodied active learning using exploration and 3d consistency. In *NeurIPS*, pages 13086–13098, 2021.

[7] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020.

[8] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. In *ICLR*, 2020.

[9] Devendra Singh Chaplot, Helen Jiang, Saurabh Gupta, and Abhinav Gupta. Semantic curiosity for active visual learning. In *ECCV*, volume 12351, pages 309–326. Springer, 2020.

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

[11] Prithvijit Chattopadhyay, Judy Hoffman, Roozbeh Mottaghi, and Aniruddha Kembhavi. Robustnav: Towards benchmarking robustness in embodied navigation. *CoRR*, 2106.04531, 2021.

[12] L.-C. Chen, G. Papandreou, F. Schroff, and A. Hartwig. Rethinking Atrous Convolution for Semantic Image Segmentation. In *CVPR*, 2017.

[13] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

[14] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think Global, Act Local: Dual-scale Graph Transformer for Vision-and-Language Navigation. 2022.

[15] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. In *ICLR*, 2019.

[16] Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *Intern. Conf. Learning Representations, ICLR*, 2021.

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

[18] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R.P. Adams, and S. Levine. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. In *NeurIPS*, 2021.

[19] Habitat sim2real. https://github.com/wgw101/habitat_sim2real.

[20] João F. Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *CVPR*, pages 8476–8484, 2018.

[21] Sebastian Höfer, Kostas E. Bekris, Ankur Handa, Juan Camilo Gamboa Higuera, Florian Golemo, Melissa Mozifian, Christopher G. Atkeson, Dieter Fox, Ken Goldberg, John Leonard, C. Karen Liu, Jan Peters, Shuran Song, Peter Welinder, and Martha White. Perspectives on sim2real transfer for robotics: A summary of the R: SS 2020 workshop. *CoRR*, abs/2012.03806, 2020.

[22] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *NeurIPS*, 2021.

[23] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, October 2020.

[24] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[25] Xueting Li, Shalini De Mello, Xiaolong Wang, Ming-Hsuan Yang, Jan Kautz, and Sifei Liu. Learning continuous environment fields via implicit functions. In *ICLR*, 2022.

[26] LoCoBot: An Open Source Low Cost Robot. http://www.locobot.org.

[27] The cvpr 2021 multi-object navigation challenge. http://multion-challenge.cs.sfu.ca/2021.html.

[28] Pierre Marza, L. Matignon, O. Simonin, and C. Wolf. Teaching Agents how to Map: Spatial Reasoning for Multi-Object Navigation. In *IROS*, 2022.

[29] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. In *ICLR*, 2018.

[30] A. Pashevich, C. Schmid, and C. Sun. Episodic transformer for vision-and-language navigation. In *ICCV*, 2021.

[31] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation, ICRA*, pages 1–8, 2018.

[32] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning. In *CVPR*, 2022.

[33] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A Generalist Agent. *arXiv:2205.06175*, 2022.

[34] A. Sadek, G. Bono, B. Chidlovskii, and C. Wolf. An in-depth experimental study of sensor usage and visual reasoning of robots navigating in real environments. In *ICRA*, 2022.

[35] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *ICLR*, 2018.

[36] 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 *IEEE/CVF International Conference on Computer Vision*, 2019.

[37] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[38] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems*, 2018.

[39] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[40] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bi-directional Domain Adaptation for Sim2Real Transfer of Embodied Navigation Agents. *IEEE Robotics and Automation Letters*, 6(2), 2021.

[41] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. In *Advances in Neural Information Processing Systems*, 2020.

[42] Fangyi Zhang, Jürgen Leitner, Zongyuan Ge, Michael Milford, and Peter Corke. Adversarial discriminative sim-to-real transfer of visuo-motor policies. *Int. J. Robotics Res.*, 38(10-11), 2019.

[43] Xinge Zhu, Jiangmiao Pang, Ceyuan Yang, Jianping Shi, and Dahua Lin. Adapting object detectors via selective cross-domain alignment. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 687–696, 2019.