Stubborn: A Strong Baseline for Indoor Object Navigation*

Haokuan Luo[1] and Albert Yue.[1] and Zhang-Wei Hong[1] and Pulkit Agrawal[1]

Abstract— We present a strong baseline that surpasses the performance of previously published methods on the Habitat Challenge task of navigating to a target object in indoor environments. Our method is motivated from primary failure modes of prior state-of-the-art: poor exploration, inaccurate object identification, and agent getting trapped due to imprecise map construction. We make three contributions to mitigate these issues: (i) First, we show that existing map-based methods fail to effectively use semantic clues for exploration. We present a *semantic-agnostic* exploration strategy (called STUBBORN) without any learning that surprisingly outperforms prior work. (ii) We propose a strategy for integrating temporal information to improve object identification. (iii) Lastly, due to inaccurate depth observation the agent often gets trapped in small regions. We develop a multi-scale collision map for obstacle identification that mitigates this issue.
Website: https://github.com/Improbable-AI/Stubborn

I. INTRODUCTION

The ability to efficiently explore and navigate to objects in indoor environments is critical for many robotic applications. The Habitat Object Navigation Challenge [1] was designed to benchmark indoor object navigation ability of artificial agents. While there are several variants of the Habitat challenge, we are interested in the version where the agent is tasked to navigate to an object instance (e.g., a bed or a refrigerator) in an unseen indoor environment. This task is known as Object Goal Navigation (OBJECTNAV). In OBJECTNAV the agent only has access to RGBD observations from first person view of the environment and its pose in the global coordinate frame. Successfully solving the task requires the agent to explore the environment to find the target object and move close to it. The accuracy of state-of-the-art method is 24% and the efficiency measured as *Success weighted by Path Length (SPL)* [2] is 8.8% [1].

It is worth contrasting the performance of *object navigation* against *point navigation*, where the agent is tasked to reach target coordinates specified relative to agent's start location (e.g., *Go 3m South and 2m East relative to start*). In *point navigation* the accuracy of state-of-the-art systems is more than 90%. Because at every time step the agent is aware of the distance and direction to the goal, efficiently solving the point navigation task requires the agent to remember previously visited locations (i.e., map) and plan a path to the goal. This task is therefore entirely geometric – the agent

can safely ignore scene semantics other than the knowledge of free space and obstacles.

In OBJECTNAV, the agent must explore its environment to discover the goal resulting in additional challenges compared to point navigation: (i) identifying the target object from visual observations; (ii) exploiting scene semantics to speed up exploration (i.e., *semantic exploration*). For instance, if the goal is to reach a refrigerator, the agent should not explore bathrooms. Similarly, to locate beds the agent should directly navigate to a bedroom instead of exploring other parts of the house. The stark difference in performance between the *object* and *point* navigation tasks indicates that current methods are inadequate at either one or both of *target object identification* and *semantic exploration*.

To understand shortcomings of state-of-the-art methods, we analyzed winners of the 2020 and 2021 OBJECTNAV challenge: Goal-Oriented Semantic Policy Agent (SemExp) [3] and End-to-End Auxiliary Task Agent (EEAUX) [4] respectively. Given that image classification systems on the Imagenet dataset have surpassed 90% accuracy [5], one wouldn't expect object identification to be a major bottleneck in OBJECTNAV. However, even 90% per frame classification accuracy is insufficient. To see why, consider a scenario where the agent requires 100 actions to reach the goal. A *false detection* in any of the first 99 steps will result in failure. Our investigation reveals that *false detection* is the primary failure mode for both SemExp and EEAUX agents.

We found the second prominent error mode to be failure to explore. While a substantial fraction of mistakes are due to the agent getting *trapped* or going around *loops*, quite surprisingly we found that the SemExp agent is unable to utilize scene semantics to guide exploration (see Fig 1).

Based on these analysis we present an agent called as STUBBORN that surpasses the performance of previous winners. STUBBORN makes improvements on three fronts:
1) **Target Object Detection** is improved by accumulating detection scores across frames.
2) **Exploration** Given that SemExp agent isn't able to utilize semantics for exploration, we developed a much simpler and *semantics-agnostic* exploration method that is both superior and compute/memory efficient.
3) **Trap Avoidance** via multi-scale representation of collision map. It prevents the agent from getting trapped due to errors in (i) agent's location resulting from discretization [6]–[8]; (ii) inaccurate depth observations [9]; (iii) and artifacts in scans of indoor spaces used in OBJECTNAV challenge [1].

Since STUBBORN does not use semantics for exploration and yet outperforms prior methods on OBJECTNAV challenge, it is a strong *semantic-agnostic* baseline for future

[1] All authors are with Improbable AI Lab in the Department of Electrical Engineering and Computer Science at MIT. {haokuan, ayue, zwhong, pulkitag}@mit.edu

TABLE I: Analyzing failure modes of previous winners of the Habitat Object Navigation Challenge [10]. Each row reports the percentage of failures attributed to a particular cause along with 95% confidence intervals. The primary cause of failure is false detection followed by exploration related issues of *loop*, *trapped* and *explore*. The last row reports the overall success rate.

Failure Mode	SemExp %	EEAux %	EEAux GT %	STUBBORN %
False Detection	47.7±9.0	35.3±6.3	1.4±1.1	45.1±8.8
Missed Detection	2.7±2.3	6.3±3.2	1.4±1.1	5.3±3.6
Loop	10.8±5.9	14.7±4.6	22.2±6.9	0.9±0.9
Trapped	10.8±5.4	12.5±4.3	13.9±5.8	8.0±5.0
Explore	8.1±5.0	8.9±3.7	12.5±5.4	13.3±6.1
Stairs	12.6±5.9	0.4±0.9	6.9±4.0	15.9±6.0
Misc	7.2±4.0	21.9±5.4	41.7±8.5	11.5±5.7
Success Rate	17.9	23.7	40.4[1]	23.7

[1] Estimated based on results from [4]

works claiming to utilize semantics for exploration.

II. EXPERIMENTAL SETUP

OBJECTNAV task requires the agent to navigate to an instance of the given target object category, such as "bed" or "table", in an unseen indoor environment. The input to the agent are first person observations from RGBD and GPS+Compass sensors providing location and orientation of the agent relative to its starting position. At each time step, the agent predicts one out of the four actions: move forward, turn left, turn right, and stop. The maximum number of steps in an episode is 500.

We follow standard evaluation protocols used in the OB-JECTNAV task of the Habitat Challenge [3], [4]. Performance of various models is evaluated using 2000 episodes of the validation split of the Matterport 3D dataset [11]. We test final performance on the Habitat's Challenge evaluation server, the results of which are publically available. Success Rate and SPL are the two primary performance metrics:

• **Success Rate** is the % of episodes in which the agent stops within a pre-defined distance threshold (1 meter) from an object belonging to the target category. The agent is not required to face the goal when stopping.

• **Success weighted by inverse Path Length (SPL)** [2]: Two agents that take paths of different lengths will have the same success rate. The SPL metric can distinguish between them by measuring agent's efficiency as:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

where N is the number of episodes, S_i is the indicator variable for success, l_i is the shortest path between the agent and the closest object instance belonging to the goal category (provided by an oracle for the purpose of evaluation only), and p_i is the agent's path length.

We use the bootstrap method [12] to report the 95% Confidence Intervals for results we collected.

III. ANALYZING PREVIOUS WINNERS

The winners of OBJECTNAV challenge in 2020 and 2021: SemExp [3] and EEAUX [4] used substantially different approaches. While SemExp explicitly builds the environment map, EEAUX is an end-to-end architecture using recurrent
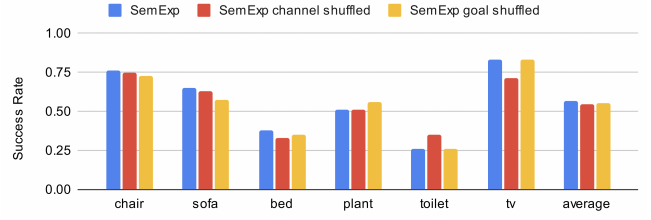

Fig. 1: The success rate of SemExp [3] when using ground-truth semantics (*blue*; 56.5% average accuracy) and two variants where the (i) semantic channels are randomly shuffled (*red*; 54.7% average accuracy) and (ii) the target goal of the planner is set to an incorrect object category (*yellow*; 55.0% average accuracy). The small drop in performance when semantic information is shuffled indicates that SemExp is unable to utilize semantics to guide exploration.

neural networks for spatial memory. As such these methods are representative of the state-of-the-art in both map-based and map-free algorithms.

We grouped the failure modes to result from errors in (i) target object detection (top-most group in Table I); (ii) exploration (second group) and other reasons. Our analysis is inspired by categorization of failure modes presented in EEAux [4] and have the following meaning:

• **False Detection** Detecting a wrong object as the goal.
• **Missed Detection** Failures to detect the object even though it was in view.
• **Loop** Poor exploration due to looping over the same locations.
• **Trapped** Repeated collisions with the same or nearby objects cause the agent getting trapped in coverage. Includes when the agent is trapped in spawn.
• **Explore** Generic failures to find the goal although the agent explores new areas efficiently. Includes semantic failures e.g. going outdoors to find a bed.
• **Stairs** The goal is on a different floor and the agent is unable to navigate up/down stairs.
• **Misc.** Other causes of failure irrelevant to this paper such as mesh artifacts, agent randomly quitting the episode, etc.

A. Analysis of SemExp Agent

We analyzed failures of the the SemExp agent by evaluating performance on 200 validation episodes using pre-trained models released by Chaplot et al. [3] (see Table I). Since the publically released model only supports 6 of the 21 goal categories used in the Habitat challenge, the validation episodes only sample goals from these 6 object categories.

Results in Table I show that primary cause of failures are *false detections* accounting for 48% of the mistakes. The next ∼30% of failures are due to exploration challenges of the agent either going around in *loops*, getting *trapped* or simply being unable to find the target object though it was exploring the environment (i.e., *explore*). To probe the reason behind exploration failures we constructed a version of the SemExp agent using ground-truth semantics. The success rate of this agent was 56.5%. However, we found that the superior performance is results from mitigation of false detections and not due to better exploration. To understand
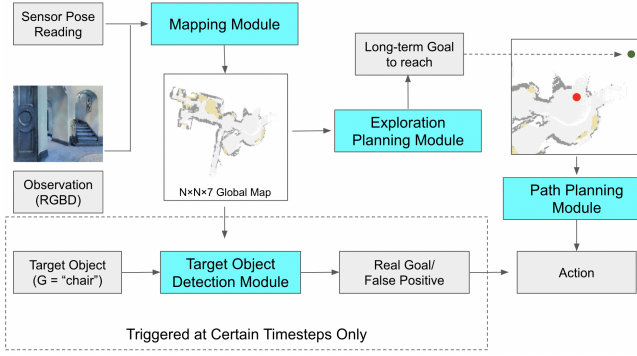
Fig. 2: Architecture of the STUBBORN agent. From sensory observations, the *Mapping* module constructs a top-down 2D map of size $N \times N \times 7$. The *Exploration Planning* module predicts the exploration goal (g_{exp}; green dot) in a local region around the agent. The *Path Planning* module plans sequence of actions from agent's current location (red dot) to g_{exp}. The target object detection module

if the agent is actually using semantics for exploration, we ran two ablation studies where during evaluation we either: (i) shuffled semantic channels or (ii) modified the input of SemExp's global planner that guides exploration to be an object category different from the actual goal. With this change, the agent will attempt to navigate to an incorrect object (e.g., "toilet" instead of a "table"). Surprisingly we found that both these ablations only lead to a 2% drop in performance suggesting that SemExp agent is unable to exploit semantics for exploration (see Figure 1).

B. Analysis of EEAux Agent [4]

We evaluated the 6-action model with tethering with the pre-trained weights released by Ye et al. [4] over the validation split of the Matterport3D (MP3D) scenes [11]. We collected 300 trajectories with each ground truth and predicted semantics. The dominant failure modes are similar to the SemExp agent. It is worth noting that using ground-truth (GT) semantics results in a much higher success rate indicating that accurate semantic segmentation is a major bottleneck in OBJECTNAV.

IV. METHOD

We propose a modular framework called STUBBORN illustrated in Fig. 2. STUBBORN consists of 4 modules for (i) mapping, (ii) exploration planner, (iii) path planner, and a (iv) multi-frame target object detector. The map is a top-down view of the environment discretized into a 2D grid. Each grid indicates if the space is free or occupied by an obstacle along with other meta-information used by the object detector. The exploration planner takes the map as the input and outputs a 2D grid coordinate indicating the goal for agent's exploration. The path planner outputs a sequence of actions to move the agent from its current location to the goal. The object detector runs at every step of the exploration to determine if the agent has reached the target object.

A. Mapping Module

The agent maintains a map $\mathcal{M}$ of size $N \times N \times 7$, where $N \times N$ (7200×7200) denotes the number of mapping grid
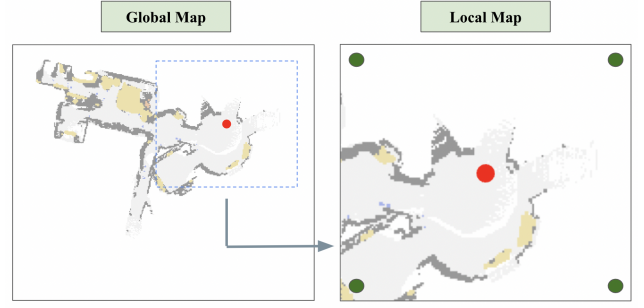

Fig. 3: Our agent maintains a global map of size $N \times N$, but plans path using a local map of size $K \times K$ around the agent (the red circle) to save computation time. The four green dots on the corners of the local map are the four possible exploration goal locations.

TABLE II: A description of the 7-channel map we maintain. The first three channels of the map are used to record obstacle-related information, while the rest of the channels are used to record goal-object-related information.

	Channel	Description
Obstacle Map	1	Obstacle Map Using Depth Input
	2	Pessimistic Obstacle Map using Collision Information
	3	Optimistic Obstacle Map using Collision Information
Object Identification Map	4	Total Number of Frames Appeared in View
	5	Sum of Confidence Score of Target Object
	6	Maximum Confidence Score of Target Object
	7	Maximum Confidence Score of Non-Target Object

units each of size $25cm^2$. The first three out of seven channels are binary, where 1 indicates obstacles and 0 represents free space. Unexplored space is treated as free space. Following the method proposed in SemExp, the first channel stores occupancy computed using depth from a RGBD camera.

Due to several reasons such as inaccurate depth observations, mesh artifacts in the Habitat simulator producing invisible obstacles, just relying on depth data to identify free space is insufficient. To mitigate these concerns, we additionally use agent's collisions to identify obstacles. A collision is defined as an event where the agent attempts to move forward but its location remains unchanged. Unfortunately, we cannot directly use collisions to construct an occupancy map because the agent's location is discretized and the the exact location and size of the obstacle is unobserved. If an obstacle is marked to occupy a size larger than its actual size, it will prevent the planner from planning a path. On the flip side, marking the obstacle to be smaller than actual size will yield infeasible plans.

We tackle this problem by maintaining a *multi-scale* obstacle map: a *pessimistic map* that marks larger areas (25×25 cm^2) and an *optimistic map* that marks smaller areas (15×15 cm^2) in front of the agent as obstacles. Given uncertainty in obstacle size and location, a useful rule of thumb we found was to always mark coordinates along the **visited path** of the agent as free space. It reduces the chances of an agent getting trapped due to incorrectly marked obstacles. These two collisions maps are encoded in channels 2 and 3 of $\mathcal{M}$ and are used by the path planner (see Sec. IV-C). Channels 4-7 of $\mathcal{M}$ (see Sec. IV-D).

Fig. 4: Tasked to go to a bed, the agent mistakes the sofa as a bed in the last frame and stops. Such a false detection results in failure.
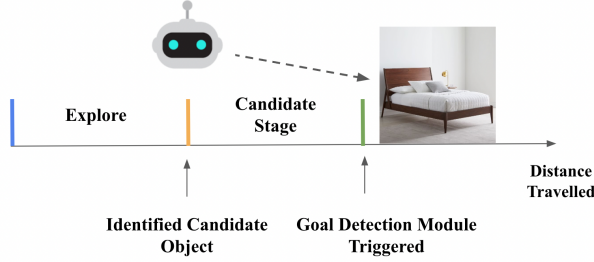

Fig. 5: Two stages of Goal Detection. After a candidate object is detected, the agent enters the candidate approach stage. After reaching the candidate object, the agent triggers the target detection module to decide whether to stop or continue exploring.

B. Exploration Planning Module

For discovering the target object the agent must explore its environment. Given we found that some state-of-the-art systems are unable to utilize semantics for exploration (Sec. III), we constructed a simple rule-based method for exploration that does not rely on any semantic information. Our intuition is that the agent will discover large areas of the environment without wasteful to-and-fro between two locations if it keeps moving along a particular direction until it runs into a dead end. At the dead-end, the agent can pick another direction and repeat the process.

Given the first three channels of the map as input ($\mathscr{M}[:,:,1:3]$), the *exploration planner* outputs one out of the four corners of a region of size $K \times K$ (local map; $\mathscr{M}_L$) centered around the agent as the exploration goal (g_{exp}). Same as in SemExp, we choose $K = 1200$ and the exploration goal is predicted in a local map instead of the global map to speedup computation. The agent is "**stubborn**" in the sense that it continues exploring until the *path planning module* (Sec. IV-C) can find no feasible path to the current goal. When such a situation occurs, the agent chooses the next corner in a clockwise direction as its exploration goal.

Collision Averse Exploration In order to pursue the exploration goal, the path planner uses the depth obstacle map and the pessimistic collision map (see Sec. IV-A). The use of pessimistic map keeps the agent away from obstacles and thereby reduces the chances of the agent getting trapped (see results in Table III) and prematurely endings its exploration.

C. Path Planning Module

Given the local map of obstacles ($\mathscr{M}_L[:,:,1:3]$) and grid coordinates of a goal (g), following the work of SemExp, we use fast-marching method [13] to compute the shortest path from agent's current location to g. The path is a sequence of 2D coordinates and subsequent coordinates are used infer one out of the four agent actions. The path is recomputed after every action (i.e., closed loop control).

Choice of Obstacle Map The path planner uses obstacle map that is computed by the element-wise OR operation between the depth obstacle map and the *pessimistic* collision obstacle map $\left(\mathscr{M}_L[:,:,1] \lor \mathscr{M}_L[:,:,2] \right)$. While the path planned using the pessimistic map maybe sub-optimal, it significantly decreases agent's chance of getting stuck and thereby improving the exploration efficiency.

If goal of the planner is g_{exp}, the pessimistic map is always used. However, if the planner's goal is the location of target object, then if the pessimistic planner leads to no feasible plan, the path is planned using the optimistic collision obstacle map $\left(\mathscr{M}_L[:,:,1] \lor \mathscr{M}_L[:,:,3] \right)$.

Brute Force Untrap Mode Sometimes the free space is so small that the path planner fails to find any feasible plan due to discretization of the agent's location to grid elements. If the agent fails to move forward (i.e., it is stuck) for ten consecutive steps, the path planner enters a *brute force untrap mode*: the agent turns either left (L) or right (R) and then attempts to move forward (F). Suppose it chose, left at the first step, then the executed actions are $L, F, L, F, L, F, L, F...$, until one of the forward action results in a forward motion. In such a scenario, the untrap mode ends and the agent resumes normal operation. However, if the agent never moves forward, it stays in the untrap mode until the end of the episode. The choice of left or right motion for untrapping alternates each time the agent enters the untrap mode.

D. Target Object Detection Module

Prior approaches to target object detection such as the one used by SemExp rely on single-frame confidence score. Because the agent decides at every step whether it reached the target object or not, relying on per-frame decisions leads to a high false detection rate as discussed in Sec. III-A. E.g., consider the sequence of four frames in Fig. 4 encountered by the agent trying to reach a "bed". The agent successfully identifies the sofa in the first three pictures, but mistakes the sofa as a bed in the fourth frame. Consequently the agent stops, believing it reached its target, but in reality it has failed. Even to the human eye, in the fourth frame the sofa looks a little like a bed. However, humans won't mistake the sofa as a bed due to the memory of "sofa" from previous frames [14].

TABLE III: The four methods used to avoid collisions help the STUBBORN Agent reduces the trap rate and improves exploration efficiency measured by the GT exploration rate. The full definition of abbreviated method names can be found in Sec. V-B.

Agent	Collision Avoidance Update				Performance	
	BFUntrap	ColObs	ColAver	VisPath	Trapped%	GT Exploration Rate %
I					10.0	54
II	✓				8.7	59
III	✓	✓			6.7	64
IV	✓	✓	✓		5.2	64
V	✓	✓	✓	✓	**5.0**	**67**

TABLE IV: Comparing the exploration performance of Frontier, SemExp, and STUBBORN agents measured as the GT Exploration Rate (Sec. V-A) with 95% Confidence Intervals. SemExp and STUBBORN are both tested with and without the Collision Avoidance Update described in Sec. V-B

Methods	Collision Avoidance Update	chair	sofa	bed	plant	toilet	tv	average %
Frontier	✓	75	65	24	56	29	**100**	58±6
SemExp	✗	76	65	38	51	26	83	57±7
	✓	79	65	38	57	26	83	58±6
STUBBORN	✗	74	58	38	50	36	66	54±6
	✓	**80**	**72**	**39**	**61**	**67**	83	**67** ±6

To mitigate false detection we designed a *two-stage approach* that detects the target object based on *sequence of frames*. Two stages illustrated in Fig. 5 are: (a) Similar to SemExp, the first stage predicts for every grid location in $\mathscr{M}_L$ visible in the agent's view (V) the confidence score $\left(c_l \in [0,1]; l \in [1,L]\right)$ for L object categories: $\left\{(x,y) \rightarrow [c_1^{x,y}, c_2^{x,y}, ..., c_L^{x,y}] \forall (x,y) \in V\right\}$. Following EEAUX, we use a RedNet [15] model finetuned on 100K randomly sampled views from Matterport 3D [4] for these predictions. If the presence of target object is detected at a particular grid location $\left(g_{tgt}: (x,y)\, \text{s.t.}\, c_{tgt}^{x,y} > 0.5\right)$, then agent starts approaching the detected target object by setting the target object location as the output of the exploration planning module. When the agent is within a distance threshold (1 meter) to the target object, it enters the *verification* stage to reduce the chances of a false detection.

Verification Stage is a binary classifier using a manually designed 4D feature representation that captures temporal information about identity of the target object to decide if its a false detection. The 4D feature vector is stored as 4 channels of the map that the agent builds as it explores the environment. Concretely, given $\mathscr{M}_L[g_{tgt}(y), g_{tgt}(x), 4:7]$, the binary classifier outputs 0/1 to indicate absence/presence of target object. $\mathscr{M}_L[:,:,4:7]$ has the following information:

• **Total View**($\mathscr{M}_L[y,x,4]$): Number of frames in the episode that $\mathscr{M}_L[y,x,4]$ appeared in agent's view.
• **Cumulative Confidence** ($\mathscr{M}_L[y,x,5]$): cumulative confidence scores of target object at location (x,y) until current time step (t) computed as $\sum_{i=1}^{t} c_{tgt}^{x,y}(i)$.
• **Max. Confidence** ($\mathscr{M}_L[y,x,6]$): $\max_{i \in [1,t]} c_{tgt}^{x,y}(i)$
• **Max. Other Confidence**($\mathscr{M}_L[y,x,7]$): maximum score among non-target categories, $\max_{l \neq tgt} \left[\max_{i \in [1,t]} c_l^{x,y}(i)\right]$

The verification system is a binary Naive Bayes Classifier trained using data collected by running the agent on the validation episodes of Matterport 3D to prevent overfitting the classifier to the training episodes which were used to train the base object segmentation model. A separate classifier is trained for each target object category using trajectories generated by the agent in pursuit of the same object category. At test-time, if the output of classifier is 1, agent believes it has reached the target and stops. Otherwise, the agent marks g_{tgt} under consideration as a false positive to prevent itself from being tricked by the same location in future exploration. If the first stage of object detection finds the target after 200 time steps (which we choose as a hyperparameter), the verification system is not used. Instead the agent is stopped

once it reaches the detected target. The rationale is that close to the end of an episode there is not enough time to search other locations for the target. Therefore, the agent might as well navigate to the detected target location.

V. RESULTS

Our main contributions are: (i) semantic-agnostic exploration method that is competitive with state-of-the-art; (ii) multi-scale collision maps for improving exploration and (iii) multi-frame object detection system to reduce false positives. The empirical evaluation presented in subsequent sections supports the importance of these contributions.

A. Exploration Efficiency

The overall success rate of the agent entangles the performance of target detection and exploration. To study exploration efficiency in isolation, we use the following metrics:
• **Ground Truth (GT) Exploration Rate** To remove the effect of target detection in evaluating exploration performance, we consider the agent to be successful if target object occupies at least 0.8% of the pixels in the *ground truth semantic mask* in the agent's view.
• **Trapped Rate** is the percentage of episodes the agent gets trapped. If the agent moves less than 1 meter in 100 time steps it is considered trapped. Lower trapped rate is better.

We compare exploration performance of STUBBORN against SemExp [3] and the well known *Frontier Based Exploration* baseline [16] where the agent explores by navigating to the frontier of previously visited locations. We used the publicly released pre-trained model of SemExp. The global map size in the SemExp model is 2400×2400, which we found to be small for Matterport 3D dataset and results in poor exploration. We modified the global map size to 7200×7200 to match with STUBBORN. Since public version of SemExp only supports 6 semantic categories, we evaluate the performance on these categories, with and without the *Collision Avoidance* method mentioned below.

B. Collision Avoidance Method

The STUBBORN Agent used 4 methods to prevent agent from getting trapped: (i) Brute Force Untrap (BFUntrap; Sec. IV-C); (ii) Multi-Scale collision obstacle maps (ColObs; Sec. IV-C); (iii) collision aversion in exploration planner (ColAver; Sec. IV-B); and (iv) marking visited paths as free space (VisPath; Sec. IV-A). Results in Table III show that all four factors are essential for improving exploration and reducing trap frequency. The most significant performance gain result from multi-scale collisions representation (ColObs).

TABLE V: Comparing success Rate of STUBBORN against the baseline Single-Frame Agent on 15 semantic categories, with 95% Confidence Intervals.

Goal	STUBBORN %	Single-Frame %	Improvement %
stool	**16**	9	77.78
toilet	**43**	35	22.86
sink	**31**	26	19.23
chest_of_drawers	**19**	16	18.75
plant	**33**	28	17.86
bed	**51**	44	15.91
cabinet	**34**	30	13.33
table	**61**	56	8.93
counter	**39**	36	8.33
chair	**47**	46	2.17
sofa	31	31	0
seating	69	69	0
picture	27	**30**	-10
cushion	46	**56**	-17.86
towel	12	**15**	-20
Average	**37** ±3	35 ±3	6.55

Results in Table IV show that without collision avoidance, SemExp and STUBBORN achieve comparable exploration performance. With collision avoidance, SemExp is no better than semantics-agnostic *Frontier* exploration strategy. STUBBORN with collision avoidance outperforms all baselines by a large margin. One reason why STUBBORN benefits from collision avoidance, but SemExp doesnot is that the STUBBORN agent is prone to getting trapped. This is because its exploration goals are chosen to be corners of a local map which forces the agent to go to the corners of the rooms more often, where it is more likely to be trapped. Collision avoidance reduces the chances of the agent getting trapped and therefore benefits STUBBORN. Note that the reported results are using data from the validation set and experiments are run with no constraint on execution time. However, for the actual test on the Habitat Challenge evaluation server, the total execution time is constrained.

C. Target Object Identification

Baseline We compare STUBBORN with a single frame baseline (Single-Frame) that only uses the first stage of the target object detection described in Sec. IV-D. This baseline is akin to the kind of object detection used in SemExp.

Results Table V compares the performance of the proposed multi-frame object detector (STUBBORN) against the single frame baseline on 15 object categories. We do not report performance on 6 out of the 21 categories because these target object categories were contained in 4 or less house environments. Results show that the proposed object detection scheme improves performance on most object categories. While the average performance of STUBBORN is higher than the Single-Frame agent, it is not statistically significant measured by 95% confidence intervals. At the same time, we consistently noticed performance improvements from the multi-frame object detector. On the official test set of Habitat Challenge, the agent's success rate increased from 22.2% to 23.7%, by switching from Single-frame to STUBBORN.

TABLE VI: We report the leaderboard entries on standard test split of the **2021 Habitat Object Navigation Challenge** Entries with the superscript 2020 and 2021 indicate challenge winners in the corresponding year. Our Method (named Yuumi_the_magic_cat) is ranked second by a minute margin based on SPL and accuracy. It substantially outperforms SemExp, the agent we have built upon.

Models	SPL	Success Rate %
(1) Habitat on Web (IL-HD)	**0.099**	**27.8**
(2) yuumi_the_magic_cat(Our Method)	0.098	23.7
(3) TreasureHunt [17]	0.089	21.4
(4) PONI (PF)	0.088	20
(5) EmbCLIP [18]	0.078	18.1
(6) Arnold (SemExp)[2020] [3]	0.0707	17.85
(7) OVRL (RGBD)	0.076	23.2
(8) Red Rabbit 6-Act Base (EEAux)[2021] [4]	0.062	23.7

D. Performance on the Habitat Challenge

We evaluate STUBBORN on the 2021 Habitat Object Navigation Challenge with the test-standard split. Results in Table VI show that our agent (with code name *yummi_the_magic_cat*) has the second best performance in SPL (trailing by a very small margin) and Success Rate. These results convincingly show that our method that doesnot uses semantics for exploration is a strong baseline for the OBJECTNAV task.

VI. RELATED WORK

Object Navigation Reinforcement learning (RL) has been employed to tackle object navigation in several recent works [4], [17]–[19]. These methods take goal objects as the policy's inputs and reward the policy upon reaching goal. In addition, Mousavian et al. [20] use imitation learning to learn a policy matching the behaviors of the optimal path planner with privileged information of the environment. In contrast, other works build explicit spatial representations. Chaplot et al. [3] maintains a semantic map for predicting locations of goal objects. Gupta et al. [21] construct a belief map of goal locations for planning. Yang et al. [22] construct an object-to-object knowledge graph and using a graph convolutional network to generate semantic priors for object locations.

Object Detection Recent works have drawn upon pre-trained computer vision models [15], [23], [24]. As per-frame predictions could produce inconsistent results across time, prior works tried improving accuracy by temporal relationships. David et al. [25] and Zhu et al. [26] use optical flow networks to propagate predictions over time. Liu et al. [27] use temporal consistency loss during training. Wang et al. [28] employ attention to relate the current frame and previous frames. While we leave incorporating such techniques to future works, we try to improve temporal consistency via an additional learned model that refines the single-frame object detection system used in [3].

Exploration for Navigation Exploration is a critical component for navigation tasks. Chen et al. [29] train an exploration policy by RL with coverage rewards. Kollar et al. [30] uses trajectory optimization to derive an efficient exploration strategy. Pathak et al. [31] train an exploration policy using prediction errors of the forward dynamics model. In

contrast, we use a rule-based exploration strategy alternating among corners of the map.

REFERENCES

[1] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," *ICCV*, pp. 9338–9346, 2019.

[2] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[3] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *NeurIPS*, 2020.

[4] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable object navigation," *ICCV*, 2021.

[5] Z. Dai, H. Liu, Q. Le, and M. Tan, "Coatnet: Marrying convolution and attention for all data sizes," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[7] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE, 2005, pp. 2432–2437.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[9] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 1–10.

[10] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects," in *arXiv:2006.13171*, 2020.

[11] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017, license available at http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf.

[12] B. Efron and R. Tibshirani, "The bootstrap method for assessing statistical accuracy," *Behaviormetrika*, vol. 12, no. 17, pp. 1–35, 1985.

[13] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[14] M.-Q. Zhu, Z.-L. Wang, and Z.-H. Chen, "Human visual intelligence and particle filter based robust object tracking algorithm," *Control and Decision*, vol. 27, no. 11, pp. 1720–1724, 2012.

[15] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, "Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation," *arXiv preprint arXiv:1806.01054*, 2018.

[16] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[17] O. Maksymets, V. Cartillier, A. Gokaslan, E. Wijmans, W. Galuba, S. Lee, and D. Batra, "Thda: Treasure hunt data augmentation for semantic navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 374–15 383.

[18] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," *arXiv preprint arXiv:2111.09888*, 2021.

[19] A. Wahid, A. Stone, K. Chen, B. Ichter, and A. Toshev, "Learning object-conditioned exploration using distributed soft actor critic," 2020.

[20] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, and J. Davidson, "Visual representations for semantic target driven navigation," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8846–8852, 2019.

[21] S. Gupta, V. Tolani, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *International Journal of Computer Vision*, vol. 128, pp. 1311–1330, 2019.

[22] W. Yang, X. Wang, A. Farhadi, A. K. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2019.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[24] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[25] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," in *CVPR*, 2018.

[26] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *CVPR*, 2017.

[27] Y. Liu, C. Shen, C. Yu, and J. Wang, "Efficient semantic video segmentation with per-frame inference," in *ECCV*, 2020.

[28] H. Wang, W. Wang, and J. Liu, "Temporal memory attention for video semantic segmentation," *arXiv preprint arXiv:2102.08643*, 2021.

[29] T. Chen, S. Gupta, and A. Gupta, "Learning exploration policies for navigation," *arXiv preprint arXiv:1903.01959*, 2019.

[30] T. Kollar and N. Roy, "Trajectory optimization using reinforcement learning for map exploration," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 175–196, 2008.

[31] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.