Stubborn: A Strong Baseline for the Indoor Object Navigation Task

by

Haokuan Luo

B.S. Computer Science and Engineering

Massachusetts Institute of Technology, 2021

Submitted to the

Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2022

Author . . . .
Department of Electrical Engineering and Computer Science
January 13, 2022

Certified by . . . .
Pulkit Agrawal
Assistant Professor
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Stubborn: A Strong Baseline for the Indoor Object Navigation Task

by

Haokuan Luo

Submitted to the Department of Electrical Engineering and Computer Science
on January 13, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This work studies the task of indoor object goal navigation, a widely-studied task that requires the agent to navigate to an instance of a given object category in unseen indoor environments. Previous state-of-the-art methods to this task include map-free end-to-end learning-based methods and methods that maintain and plan with spatial maps, but they both struggle to perform well in the task. Experiments show that the primary reasons for failures are poor exploration, agent getting trapped, and inaccurate object identification. For exploration strategy, we show that previous map-based methods fail to use semantic clues effectively and present our semantic-agnostic exploration strategy that proves to perform much better. For object identification, we show that using cumulative information across multiple frames leads to higher accuracy in object identification. We additionally present our methods for decreasing the agent's chance of getting stuck. The combination of our work leads to the winning entry on the leader board of the CVPR Habitat ObjectNav challenge.

Thesis Supervisor: Pulkit Agrawal
Title: Assistant Professor

Contents

1 Introduction

The task of indoor object goal navigation requires an agent to navigate to an instance of an object goal (i.e. bed or table) in an unseen indoor environment. While this task is easy for a human, it becomes much more challenging for computers. Consider a human being in an unseen environment with the task of finding the bed. Humans naturally observe the environment and build global semantic maps in their brains. With the global map, humans perform deduction on where to explore to maximize the chance of finding a bed. However, this reasoning process poses a considerable challenge to computers since it requires estimating accurate spatial and semantic map, efficient planning algorithms, and associating ego-centric views to the map.

The complexity of the indoor object navigation task makes it a good testbed for algorithms in robotics and machine learning, and the ability to efficiently reason over the internal semantic global map and navigate around the environment is critical for a variety of practical robotic applications (e.g. home robots). Map-based and Map-free approaches are both used to solve the task of object navigation and achieve similar performance. Map-free approaches [30, 15, 18, 19, 24, 10] usually use end-to-end reinforcement learning with methods that encode memory implicitly such as LSTM [12] or GRU [6]. While traditional end-to-end approaches suffer from many drawbacks such as low sample efficiency [26] and poor generalization[30], recent advancement in auxiliary tasks [31] addressed the low sample efficiency problem of end-to-end models, making end-to-end models a viable approach to object goal navigation[30]. Map-based approaches[4, 29], on the other hand, maintain explicit maps of the space and plan accordingly. Those approaches have the advantage that they are easier to reason about and more sample efficient than map-free approaches.

This work studies the failure modes of the state-of-the-art approaches of Object Goal Navigation in both the map-based branch [4] and map-free branch [30] and proposes a method that improved upon Goal-Oriented Semantic Policy(SemExp) [4], the previous state-of-the-art approach in the map-based branch. Experimental results show that most failures are caused by goal misidentification, getting stuck, or exploration. The proposed method, called Stubborn, makes three improvements corresponding to each of the major failure categories, which lead to the winning entry on the leaderboard of the CVPR Habitat Object Navigation Challenge [1]. The improvements are listed below.

1. **Exploration** We show that current exploration strategy [4] that exploits semantic information is unable to use it for object goal location inference. We present a much simpler method that not only works better but also takes less time and memory.
2. **Object Goal Detection** Current methods for object goal detection suffer from object goal misidentification. We present a method for accumulating information across multiple frames for detecting goal objects.
3. **Getting Trapped** We present 4 strategies for collision avoidance that overcome the issue of agent getting trapped (caused by discretization in agent's localization in the global map [7, 8, 20], inaccuracy in depth observations[14], and environment models of indoor spaces of the Habitat Challenge[22]).

2 Related Work

To find the goal object in an unseen environment, the agent needs to explore the environment, infer goal object locations, and identify the goal object. We briefly discuss prior work on the task of object navigation, as well as related work on semantic maps and object detection.

Object Navigation Some recent works have approached object navigation via end-to-end imitation or reinforcement learning approaches while implicitly learning semantic priors within the models. Ye et al. [30] learn a policy via reinforcement learning with auxiliary learning tasks and an exploration reward. Khandelwal et al. [15] incorporate CLIP features for object navigation. Furthermore, Maksymets et al. [18] augment scenes by inserting objects to increase object and trajectory diversity during training. Mousavian et al. [19] use imitation learning to learn a policy. Wahid et al. [24] address the similar problem of object navigation with continuous control using Soft Actor-Critic [10].

In contrast, other works have built explicit spatial representations that are then used in navigation, such as a map of the explored region in SLAM. Chaplot et al. [4] propose a hierarchical SLAM-based approach, winning the 2020 iteration of the Habitat Challenge [1]. The approach maintains a semantic map of the explored region and used a learned model to learn semantic priors and predict goal locations to find the target object. Similarly, Gupta et al. [9] construct a belief map to use for planning. On the other hand, Yang et al. [29] use a topological representation, constructing an object-to-object knowledge graph and using a graph convolutional network to generate semantic priors.

Semantic Mapping One of the fundamental components of SLAM is the construction of a 2D or 3D map. Particularly relevant in the case of object navigation is the incorporation of semantics. Prior works have done so via probabilistic graphical models [2] or learned computer vision models [32, 17]. As we follow the work of Chaplot et al. [4], we instead learn semantic mapping using differentiable projection operations.

Object Detection In addition to planning, object detection is a critical module towards success. Recent works have generally drawn upon pre-trained models from the computer vision community, mostly from semantic segmentation. For example, Chaplot et al. [4] use Mask-RCNN [11] and DeepLabv3 [5], whereas Ye et al. [30] use RedNet [13]. Additionally, with the increased interest in embodied tasks in recent years, video semantic segmentation has also become a field of interest. Classic semantic segmentation models do not necessarily perform as well in such an environment, as per-frame predictions can produce inconsistent results across time. Some prior works have tried to improve accuracy by exploiting these temporal relationships, such as [21] and [33], which use optical flow networks to propagate predictions. More recently, Liu et al. (2020) instead incorporates temporal consistency as a component of the loss and via knowledge distillation during training [16]. Similarly, Wang et al. (2021) construct a memory and use attention to relate the current frame and previous frames [25]. While we leave incorporating such techniques to future works, we try to improve temporal consistency via an additional learned model that refines semantic segmentation.

3 Experimental Setup

We follow the standard experimental setup used in Object Goal Navigation papers [4, 30]: we run experiments on 2000 episodes of the validation split of the Matterport 3D dataset [3] in the Habitat Simulator[22]. The task config file is identical to the one used in the habitat challenge, and we do not enforce any run-time limit.

For the semantic segmentation model used in object detection and segmentation of the agents, we use a RedNet [13] model finetuned by Ye et al. on 100k randomly sampled forward-facing views from Matterport 3D [30].

Metrics We introduce below the metrics we use in our experiments that's referred to in the rest of the paper:

- **Success Rate** The rate of success, which is defined as the agent being within a certain distance threshold (1 meter) from the goal object category when it stops.

- **Success Measured by Path Length (SPL)** SPL is used to measure the agent's efficiency in completing the task. SPL is calculated 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 shorted path between the agent and the closest instance of the goal category object, and p_i is the path length taken by the agent.

- **Ground Truth (GT) Exploration Rate** This metric is used to measure the performance of the exploration strategy. It isolates exploration from other factors that affect success such as object detection and object localization. The agent explores successfully if it sees the goal object given the ground truth semantic information. Formally, an agent is considered successful if the target object appeared in the ground truth semantic mask of the agent's view and it occupies at least 0.8% of the pixels in the view. We call the success rate defined above the "Ground-Truth Exploration Rate" (GT Exploration Rate).

- **Plateau Rate** This rate is used to measure how often the agent gets trapped. An agent is considered trapped if it fails to move more than 1 meter for 100 steps. Different from other metrics, this metric is better when it is lower.

4 Agent Behavior Analysis

To improve the success rate in object navigation, it is important to understand the failure modes of agents. We analyze failure modes of Goal-Oriented Semantic Policy Agent (SemExp) [4] and End-to-End Auxiliary Task Agent (EEAux) [30]. We choose them because they won the Habitat ObjectNav Challenge in 2020 and 2021 respectively, and they used completely different methods, making them good representatives of map-based and map-free approaches to spatial memory.

We break down the failure modes into 14 categories described below, based on the ones defined in [30]. Note that we renamed the Detection mode Attention. Seven of them are common to both SemExp and EEAux, six of them are unique to EEAux, and one of them is unique to SemExp.

Common to Both:

- **Explore** A generic failure to find the goal despite steady exploration. Includes semantic failures e.g. going outdoors to find a bed.

- **Plateau** Repeated collisions with the same or nearby objects cause a plateau in coverage. Includes when the agent is trapped in spawn.

- **Loop** Poor exploration due to looping over the same locations or backtracking.

- **Detection** Didn't detect the object even though it was in view.
- **Misidentify** Misidentified something as the goal.
- **Stairs** Agent had to navigate up/down stairs to reach the goal.
- **Bad Mesh** The house mesh has artifacts, mistakes, practically invisible obstacles. (due to size) or other features that make navigation difficult.

Only in EEAux:

- **Attention** Despite positive SGE, the agent neither notices nor successfully navigates to the goal.
- **Last mile** Gets stuck near the goal.
- **Commitment** Sees and approaches the goal but passes it.
- **Open** Explores an open area without any objects.
- **Not Navigable** Agent started at a location that cannot reach a goal object.
- **Quit** Agent stopped within roughly 10s with no apparent reason.

Only in SemExp:

- **Localization** Wrongly predicted goal object location in the 2D map after identifying it.

We now describe the distribution of failure modes in each method.

4.1 Goal-Oriented Semantic Policy

The Goal-Oriented Semantic Policy (SemExp) [4] won the 2020 Habitat ObjectNav Challenge with a map-based approach. We ran the SemExp agent with the pre-trained weights released by Chaplot et al. [4]. We sampled 200 episodes from the validation episodes in the Matterport 3D [3] ObjectNav dataset for the agent to run. Since the pre-trained weights released by Chaplot et al. only support 6 of the 21

Table 1: Breakdown of Failures of SemExp using predicted semantics, EEAux using predicted semantics, EEAux using ground truth semantics, and Stubborn using predicted semantics

Failure Mode	SemExp	EEAux	EEAux GT	Stubborn
Misidentify	47.7 ± 9.0	32.6 ± 7.6	1.4 ± 1.1	45.1 ± 8.8
Stairs	12.6 ± 5.9	0.7 ± 1.0	6.9 ± 4.0	15.9 ± 6.0
Loop	10.8 ± 5.9	15.2 ± 5.6	22.2 ± 6.9	0.9 ± 0.9
Plateau	10.8 ± 5.4	23.2 ± 6.9	13.9 ± 5.8	8.0 ± 5.0
Bad Mesh	3.6 ± 3.2	0.7 ± 0.7	5.6 ± 3.6	7.1 ± 4.7
Detection	2.7 ± 2.3	3.6 ± 3.1	1.4 ± 1.1	5.3 ± 3.6
Explore	8.1 ± 5.0	5.8 ± 3.8	12.5 ± 5.4	13.3 ± 6.1
Localization	3.6 ± 3.2	-	-	4.4 ± 3.6
Attention	-	0.7 ± 0.7	3.5 ± 2.5	-
Last Mile	-	0.7 ± 0.7	3.4 ± 2.5	-
Commitment	-	2.1 ± 2.1	2.8 ± 2.5	-
Open	-	8 ± 4.2	2.8 ± 2.5	-
Not Navigable	-	5.8 ± 3.8	2.1 ± 1.8	-
Quit	-	0.7 ± 0.7	21.5 ± 6.9	-

goal categories used in the habitat challenge, we only sample from the 6 supported categories. We manually labeled the failure cases, and the breakdown of failure cases is presented in Table 1.

The primary reason for failures is due to false positives in object detection. In 48% of the failure cases, the agent identified the wrong object as the goal and stopped. Other failure cases are due to diverse reasons. 19% of failures are due to inefficient exploration strategy, which mainly comprises "explore" and "loop". 23% of failures are due to flaws in microscopic movement ("stairs" and "plateau"). Other 10% failures are due to miscellaneous reasons like flaws in the images generated by the simulator, locating the goal to the wrong position, and missing the goal even if it appeared in the view. Although the design of the SemExp agent attempts to utilize semantic priors, we do not notice any apparent signs of using semantic priors in the agent's exploration behavior.

4.2 End-to-End Auxiliary Task Agent

While in this paper we present a baseline based on SemExp, we still find it useful to profile an agent trained via end-to-end reinforcement learning to compare with a map-based approach. The model presented in Ye et al. [30], which we refer to as EEAux, won the 2021 Habitat ObjectNav Challenge [1] by introducing semantic features and multiple auxiliary tasks and adding an exploration reward during training.

We ran the 6-action model with tethering with the pre-trained weights released by Ye et al. [30] over the validation split of the Matterport3D (MP3D) scenes [3]. In total, we collected 299 trajectories with ground truth semantics and 189 trajectories with predicted semantics. Note that [30] includes some analysis of failure modes, they only do so for the agent provided with ground truth semantics, while we do so for both ground truth and predicted semantics.

The distribution of failure modes is in Table 1. In the ground truth semantics, we found a fairly similar breakdown as to what was reported in Ye et al. [30]. The primary failure modes were Quit and exploration related modes, namely Explore, Plateau, and Loop, with other modes contributing to a few failures each. On the other hand, with predicted semantics, misidentification becomes a significant failure mode, accounting for about 33% of all failures as well as seeming to add additional failures over the ground truth evaluation. While quitting is no longer a significant failure mode (likely because any possible cases of it are overshadowed by other issues), failures related to exploration still persist as the second most significant issue. Some modes of failure, such as stairs, decreased in the count, but this is mostly because other reasons predominated over them.

5 Method

In this section, we describe the Stubborn Agent, our map-based approach to solve the object navigation task.

5.1 Task definition

The Object Goal navigation task requires the agent to navigate to an instance of the given object goal category, such as “bed” or “table”, in an unseen indoor environment. The input to the agent is an RGB-D sensor and a GPS+Compass sensor. The RGB-D sensor provides the RGB image and the depth image of the current environment from the point of view of the agent, and the GPS+Compass sensor provides the location and orientation of the agent relative to its starting position. At each timestep, the agent receives the input from the sensors and needs to take one of the four actions: move forward, turn left, turn right, and stop. There is a limit on the number of steps the agent can take in each episode (500 steps). Success Rate and SPL are the two primary performance metrics for the task.

5.2 Overview

We propose a modular framework called “stubborn agent” (Stubborn). We divide the task of object goal navigation into several modules inspired by how humans solve the object navigation task: humans naturally construct and maintain maps of their surroundings, infer where the goal object is most likely to be, navigate to the inferred location, and repeat this process until the goal is identified. Therefore, the Stubborn agent consists of 4 modules: a mapping module, a rule-based global goal module, a rule-based path planning module, and a learned multi-frame goal detection module. The design of the mapping module and the path planning module are both inspired by SemExp [4]. The mapping module uses the RGB-D and location input from the sensors to build a top-down grid map of obstacles and semantic information over time. The global goal module receives the grid map as input and gives long-term goals, which are grid coordinates on the map, for the agent to reach, either for exploration or for approaching a potential goal object. The long-term goals could be either a single grid or multiple grids. The path planning module receives the grid map and the long-term goals and determines the action required to reach any of the long-term goals using the Fast Marching Method [23], a classic path planning algorithm. The goal-detection module is activated when the agent believes it reaches a goal candidate. The goal-detection module receives the map and the location of the goal candidate and outputs either True or False. True means the goal candidate is the real goal and the agent should stop, while False means the goal candidate is false positive and the agent should keep exploring. We now give details on each module.

5.3 Mapping Module

We maintain a map of size $M \times M \times 7$, where $M \times M$ (7200×7200) denotes the map size, and each grid corresponds to an area of size $25cm^2$. The first three channels of the map are obstacle maps. They record binary indicators for obstacles, where 1 represents obstacles and 0 represents free space, treating unknown space as free space. The first channel uses depth input from sensors to calculate the 3D point cloud and then projects the point cloud to the 2D map to get obstacle information. We noticed in experiments that depth information alone is not sufficient for constructing a complete obstacle map, as agents could still get trapped when navigating with obstacle map constructed from depth input, potentially due to inaccuracy in depth input and environment models (bad meshes) of indoor spaces in the habitat simulator. Therefore, we additionally use map channels to record obstacles indicated by collisions, which is defined as the agent attempting to move forward but failing to change its location. When collision happens, the obstacle's exact location (Is the obstacle on the left side of the agent or the right side?) and size is unclear, since visual information does not show any obstacles. Therefore, we cannot be certain how large of an area to mark as obstacles when a collision is detected. Marking an area that is too small will result in the path planner planning a path that is still blocked by obstacles; marking an area that is too large will result in the path planner failing to plan a path when there is one. We tackle this problem by maintaining two versions of the obstacle map: channel two, the pessimistic map, marks larger areas around the front of the agent as obstacles when collisions are detected, while channel three, the optimistic map, marks smaller areas as obstacles. This allows the path planner to first plan paths using the pessimistic map, and switch to the optimistic map in case it fails to plan a path using the pessimistic map.

One rule we found useful in implementing the obstacle maps is that coordinates along the visited path of the agent should always be marked as free space. This allows the agent to plan a path out of obstacles along the path it comes in when it gets trapped.

The rest of the map channels (Channel 4-7) are dedicated to object detection. Details are described in Sec. 5.6.

5.4 Global Goal Module

The global goal module receives the grid map as input and gives the agent the location of long-term goals to reach, similar to how humans infer "I should get to the other end of the corridor to see what's there". Human uses semantic information to make this inference (i.e. the dining table is in the kitchen or living room). Some previous approaches [4, 29, 27] attempted to use semantic information for better exploration. Among them, SemExp [4] won the 2020 Habitat Object Navigation Challenge, making it a good representative of the state-of-the-art in semantic exploration. However, we find that SemExp is slow and fails to fully utilize semantic information in experiments. We present a simple rule-based exploration strategy that is semantic-agnostic with stronger performance and faster speed.

SemExp While evaluating the performance of SemExp, we find that it has the following drawbacks: First, its execution speed is slow. The SemExp agent spent 43% of its running time maintaining the semantic map, which means the agent would be much faster without having to maintain semantic information. Execution speed is critical in the Habitat Challenge [1], as it has a constraint on the total number of time an agent can take to finish all episodes (42 hours for the 2000 test-std splits). While submitting to the Habitat Challenge, we found that we have to limit the maximum steps the SemExp agent can take to 300 steps per episode to avoid timeout, while 500 steps are allowed by the rules if the agents are fast enough.

Second, the SemExp agent fails to utilize the semantic information it saved. Fig. 1 presents the Ground Truth Exploration Rate of the SemExp agent and its two vari-

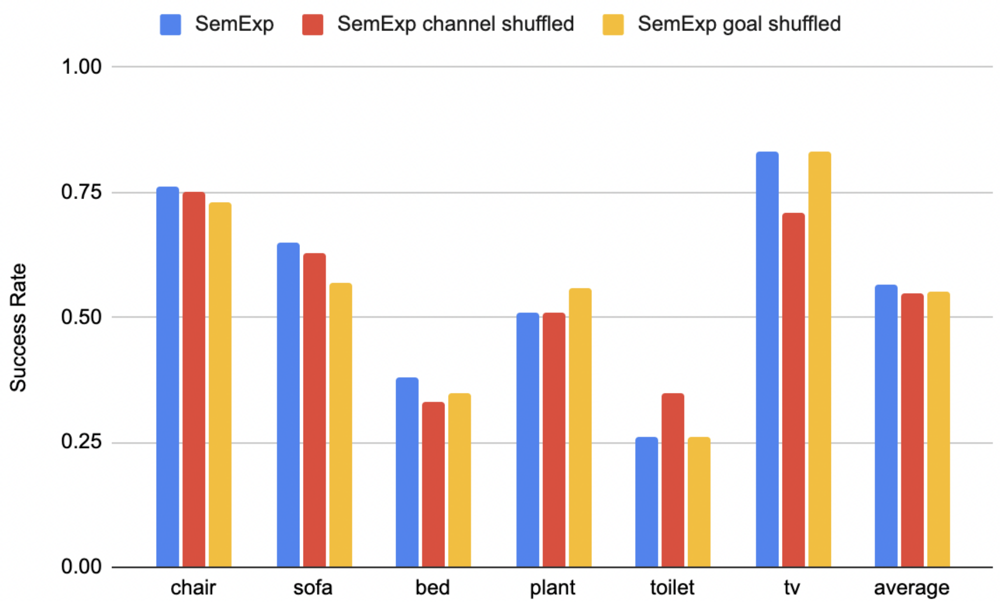

Figure 1: Success rate of SemExp and its two variants for ablation analysis, where success is defined as seeing the goal object with ground truth semantic map. The first variant's semantic information is randomly shuffled across categories, and the second variant's object goal category is randomly changed for the global planner module. SemExp achieved a 56.5% success rate, while the two variants achieved 54.7% and 55.0% success rates. SemExp does only slightly better than its ablation variants, suggesting semantic information is not fully utilized.

ants with ablations evaluated on the 6 semantic categories that were supported by the publicly released version. The first variant has the semantic map's channel information shuffled (so that the "bed" channel might contain the "table" channel's data). The idea behind this experiment is that, if the semantic channels are heavily used by SemExp, shuffling the semantic channel information would make the performance much worse. The second variant has the goal category fed into the global planner shuffled (so that the global planner will try to navigate to a "toilet" while "table" is the actual goal). The idea behind this experiment is similar: if semantic information is heavily used, attempting to navigate to the wrong object category should significantly decrease performance. The experiment result shows that the semantic information is indeed utilized by the agent, but it contributes only about 2% in performance improvement. This motivates us to discard semantic information in the global planner module.

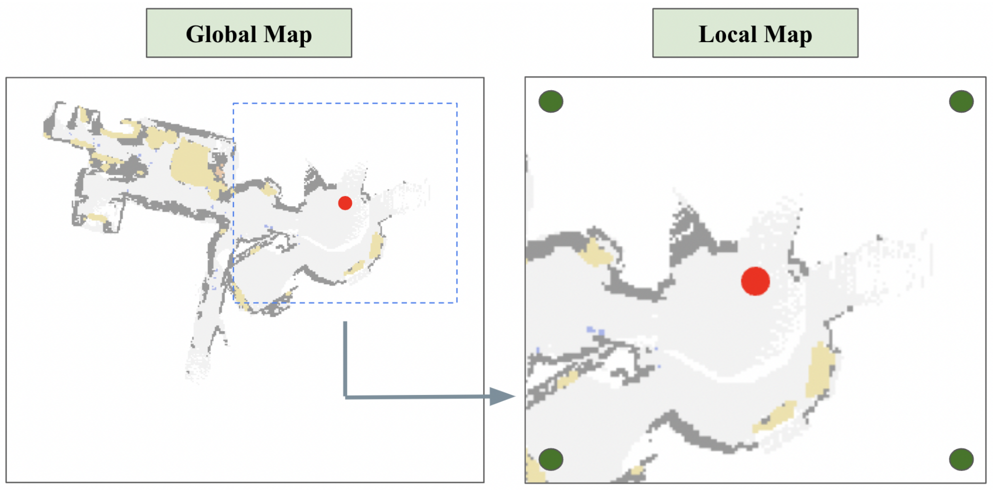

Figure 2: **Global and Local Map**. The Stubborn Agent maintains a global map of size $M \times M$, but only uses a local map of size $N \times N$ around the agent (the red circle) for the path planning module to save time. The four green circles on the corners of the local map are the four possible exploration goal locations outputted by the Stubborn Policy

Stubborn Policy The Stubborn Policy is a simple rule-based exploration strategy for the global planner module that is semantic-agnostic with stronger performance and faster speed. It takes the first three channels of the map (the obstacle map) as input, and output long-term goals for the agent to reach. As shown in Fig. 2, although the agent maintains a global map of size $M \times M$, the goal outputted by the global planner module is limited to grids on the local map of size $N \times N$ (1200×1200, the same value used in SemExp) around the agent, so that the path planning module can be faster by only performing path planning on the local map. The stubborn policy chooses one of the four corners of the local map as the global goal. The policy is "stubborn" in the sense that it does not switch its goal until the path planner can no longer plan a path to the goal (which usually means the agent entered a dead-end like one end of the corridor). The policy then rotates the global goal to the next corner and repeats the process. The motivation behind this policy is that we want to minimize the time steps wasted by the agent going back and forth by keeping the agent exploring one direction until it reaches a dead end.

Collision Aversion of Global Planner The stubborn planner we described above switches its goal when the path planner fails to plan a path to the goal. The obstacle map fed into the path planner mentioned above uses the combination of the depth obstacle map and the pessimistic collision obstacle map (which marks collision obstacles as larger areas compared to the optimistic one). The motivation for using the pessimistic collision obstacle map is that we do not want to plan paths too close to collision points for exploration. We simply change global goals instead in that case. This design makes sure that the agent only uses the pessimistic obstacle map for exploration to stay away from the obstacles during exploration.

In the case that a potential goal candidate is detected, the policy outputs locations of potential goal candidates instead, and details are described in Sec. 5.6.

5.5 Path Planning Module

Shortest Path Algorithm The path planning module receives the local obstacle map and gird coordinates of the goals, and outputs the action agent needs to take to reach one of the goals. We use the Fast Marching Method [23] similar to SemExp [4] for path planning. The fast marching method takes the obstacle map and the goals and outputs a sequence of coordinates of the path to one of the goals. The agent then uses the coordinates to decide which of the four actions to take.

Collision Obstacle Map to Use The mapping module described three channels of the obstacle map. Channel 1 is the obstacle map calculated from depth, and channel 2 and 3 are the pessimistic and optimistic versions of the obstacle map calculated from collisions. The obstacle map that is fed into the Fast Marching Method is calculated by the element-wise OR operation of the depth obstacle map and the collision obstacle map. Since there are two versions of the collision obstacle map, the path planning module first uses the pessimistic collision obstacle map, and switch to the optimistic collision obstacle map in case it fails to plan a path using the pessimistic map.

Brute Force Untrap Mode Sometimes the open space is so small that the path planner fails to plan a path given the obstacle map due to discretization in the agent's localization in the grid map. In this case, the path planning module enters the brute force untrapping mode: the agent will turn left and right and attempt to move forward.

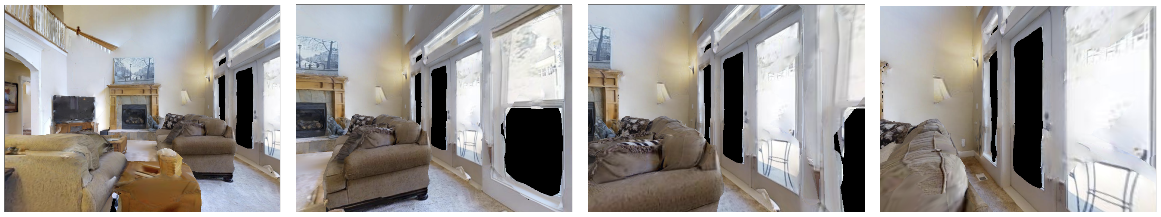

Figure 3: Failure Example. The goal is to navigate to an instance of bed, but the agent mistakes the sofa as a bed in the last frame, resulting in failure.

5.6 Object Detection Module

The previous approach to object goal detection used by SemExp [4] only uses 1 map channel for object goal detection, which is a binary indicator channel that records whether goal object is detected on the corresponding area. They first train a semantic segmentation model like Mask-RCNN [11] or RedNet [13], then using the semantic segmentation mask returned by the model to assign labels to the point clouds calculated from depth input, which allow them to finally project the point clouds to the 2D map to update the goal object binary indicator map, where 1 means the goal is detected on the corresponding area and 0 means no goal is detected. Whenever there are non-zero entries on the goal object map, the agent will navigate to the closest non-zero grid entry and stop when it reaches the grid location.

However, the approach above is subject to noises in the segmentation model and high levels of false positives. Figure 3 demonstrates one failure example where the goal is to navigate to an instance of bed. The agent successfully identified the sofa in the first three pictures, but wrongly identified the sofa as a bed in the last picture, resulting in failure.

The last frame alone is indeed confusing: even from a human's perspective, the last image does look like a bed. However, a human won't make this mistake if they are also presented with the previous pictures: they would remember that this object is actually a sofa, so it is not a bed. We name this situation a spatial-temporal conflict: the same location in the global map was identified as different objects in different frames. In the case of spatial-temporal conflict, why do humans know that the object is not a bed? Because in the many frames we see the object, only the

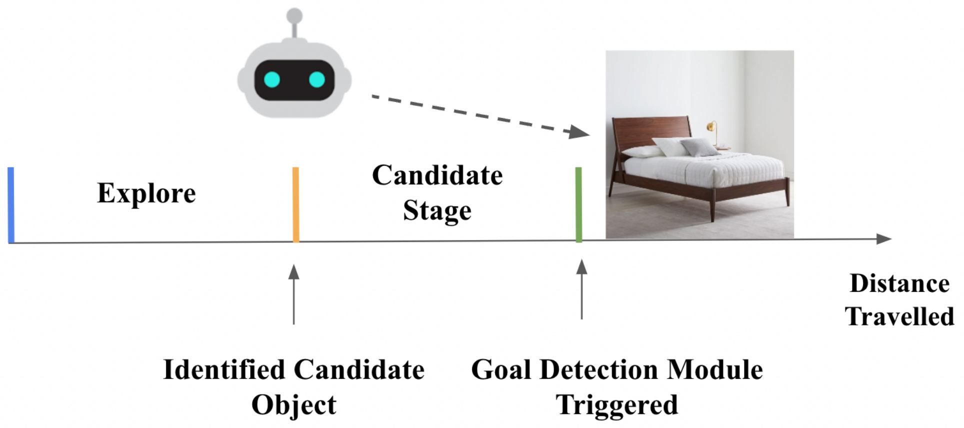

Figure 4: Two stages of Goal Detection. After a candidate object is detected, the agent enters the candidate stage and approaches it. After reaching the candidate object, the agent triggers the goal detection module to decide whether to stop or continue exploring.

last frame was identified as a bed. We call this concept the cumulative confidence. This motivates us to accumulate information across multiple frames to make more informed distinctions between objects. We extend object-detection-related mapping channels to four and construct feature vectors for objects using the extended map channels to capture accumulated information of objects.

Mapping We use channels 4 - 7 of the map to keep information useful in goal object detection. Channel 4 is the total view channel: it records the total number of frames the corresponding area appeared in the view of the agent. Channel 5 is the cumulative confidence channel: it represents the sum of the confidence scores reported by the object segmentation model for the goal object across all frames in that grid. Channel 6 is the highest confidence channel: it represents the highest confidence scores for the goal object in that grid. Channel 7 is the spatial-temporal conflict channel: it represents the highest confidence score for non-goal objects (which suggests spatial-temporal conflict) in that grid.

Two Stages of Goal Detection The object goal detection framework has two stages as shown in Fig. 4. Stage I is called the candidate stage. This is the stage where the agent identified the potential goal object and approaches it. Whenever the confidence score of grids in Channel 6 (maximum confidence channel) exceeds a certain threshold, the agent marks the grids as potential goals, and the global goal module output those grids as long-term goals. After the agent reached the long-term goals, it enters stage II and the goal detection module is triggered to decide whether the goal candidate is the true goal or a false positive. The reason why the goal detection module is triggered after the agent has reached the goal candidate instead of at the beginning of the candidate stage is that we want to accumulate more information regarding the goal object through the process of approaching the goal candidate.

Object Goal Detection Module When the goal detection module is triggered, the agent needs to decide whether the goal candidate is the real goal or a false positive using information such as spatial-temporal conflict or cumulative confidence. We treat this as a binary classification learning problem. For the candidate goal object, we construct a feature vector for it (described below) and use the ground-truth map to get the ground-truth result on whether the candidate is the real goal or false positive. We run the agent in the habitat simulator to collect training data and then train binary classifiers with the training data. The trained binary classifier is then used in the object goal detection module. If the trained classifier returns True, the agent believes the candidate is the real goal and stops. Otherwise, the agent marks the corresponding areas of the candidate object as false positive so that it will not be tricked by it in the future, and continues exploring.

In practice, we let the agent STOP regardless of what the object goal detection module outputs if the agent is already past a threshold of time steps (200) in the episode and reached a goal candidate. The rationale is that if the agent is already in the later stage of the episode, it will have little time to explore and find another potential goal, so it is better for the agent to stick with the current goal candidate even if the current goal candidate is not ideal.

Define "Object" In the description above, we have not defined what an "object" is from the computational perspective. An object in a human's concept, such as a bed or a table, usually corresponds to multiple grid points that are next to each other on the 2D grid map. Therefore, we treat each connected component of the non-zero entries of Channel 6 (maximum confidence channel) as an object.

Object Feature Vector To construct the feature vector for an object, we want to include information related to confidence score, cumulative confidence score, and spatial-temporal conflict. Therefore, we construct a 5-dimension feature vector with each dimension described below:

- **Maximum Confidence Score** This is the maximum confidence score reported by the segmentation model for the candidate object.
- **Cumulative Confidence Score** This is the sum of confidence scores of the candidate object across all the frames.
- **Total Number of Frames** This is the total number of frames the candidate object appeared in the view of the agent, regardless of whether the candidate object is identified as a goal or not by the semantic segmentation model.
- **Average Confidence Score** This number is calculated as

  $$\frac{Cumulative\ Confidence\ Score}{Total\ Number\ of\ Frames}$$

  . It is the average confidence score of the candidate among all the frames it appeared in the view of the agent.
- **Conflict Object Score** It is the highest confidence score of non-goal categories reported by the semantic segmentation model in the same location of the goal candidate object.

5.7 Summary

The Stubborn Agent makes 3 improvement upon SemExp [4] described below:

- **Global Goal Module** SemExp uses a learned global goal module that attempts to use semantic information, while Stubborn uses a rule-based global goal module that is semantic-agnostic.

- **Object Goal Detection Module** SemExp uses the confidence score reported by the semantic segmentation model of every single frame to determine whether the goal object is found and reached, while Stubborn uses information accumulated from multiple frames to determine whether the goal object is found and reached.

- **Collision Avoidance** Four updates are applied to the Stubborn agent to decrease failures caused by Plateau. While the four updates are already discussed in various subsections in Sec. 5, we summarize them below.

  - **Collision Obstacle Size** Two maps are maintained to record collision obstacles, the pessimistic one marking larger areas as obstacles, while the optimistic marking smaller areas. The path planner uses the pessimistic map first, and switch to the optimistic map if it fails with the pessimistic map.
  - **Marking visited path as Free Space on Obstacle Maps**
  - **Collision Aversion in Global Planner** The stubborn planner we described in the global goal module switches its goal when the path planner fails to plan a path to the goal. To decide when to switch global planner's goal, the path planner uses the pessimistic obstacle map to avoid planning paths too close to collision points for exploration.
  - **Brute Force Untrapping Mode** Sometimes the open space is so small that the planner fails to accurately plan a path given the obstacle map. In this case, we enter the brute force mode: the agent will turn left and right and attempt to move forward.

6 Results

In this section, we evaluate the performance of the Stubborn Agent. As the Stubborn Agent made 3 primary changes from SemExp [4] described in Sec. 5.7, we evaluate

Methods	Collision Avoidance Update	chair	sofa	bed	plant	toilet	tv	average
Frontier	✓	0.75	0.65	0.24	0.56	0.29	**1.00**	0.58 ± 0.06
SemExp	✗	0.76	0.65	0.38	0.51	0.26	0.83	0.57 ± 0.07
	✓	0.79	0.65	0.38	0.57	0.26	0.83	0.58 ± 0.06
Stubborn	✗	0.74	0.58	0.38	0.5	0.36	0.66	0.54 ± 0.06
	✓	**0.8**	**0.72**	**0.39**	**0.61**	**0.67**	0.83	**0.67** ± 0.06

Table 2: Exploration Performance of Frontier, SemExp, and Stubborn. SemExp and Stubborn are both tested with and without the Collision Avoidance Update described in Sec. 5.7

the performance change brought by each change respectively. We then present the Stubborn Agent's performance on the Habitat Challenge [22].

6.1 Exploration Efficiency

We evaluate the exploration strategy of the Stubborn Agent using GT Exploration Rate as the metric. Table 2 compares the Stubborn Agent with the baselines. We use SemExp[4] and the Frontier Based Exploration [28], a classic baseline exploration strategy where robots always try to navigate to the intersections of observed and unobserved space (the frontier), as baselines. We run SemExp with the pre-trained model that is publicly released. We change the global map size of SemExp to 7200×7200 to match with Stubborn (The original global map size in the released version of SemExp is 2400×2400, which is too small for Matterport 3D dataset). While Stubborn requires training data for its Object Detection Module, the Object Detection Module is not relevant when using GT Exploration Rate as metrics, so both Frontier and Stubborn Agent requires no training.

For both SemExp and Stubborn, we evaluate the performance of those agents both with and without the Collision Avoidance Update described in Sec. 5.7. Since one of the baseline methods, SemExp, only supports 6 semantic categories in its publicly released version, we perform experiments only on those 6 categories: "chair", "couch", "plant", "bed", "toilet", "tv". The Stubborn Agent with the collision avoidance update outperforms all baselines by a considerable margin and the difference in success rate is statistically significant.

Comparing Stubborn and SemExp The Stubborn Agent performs similarly with

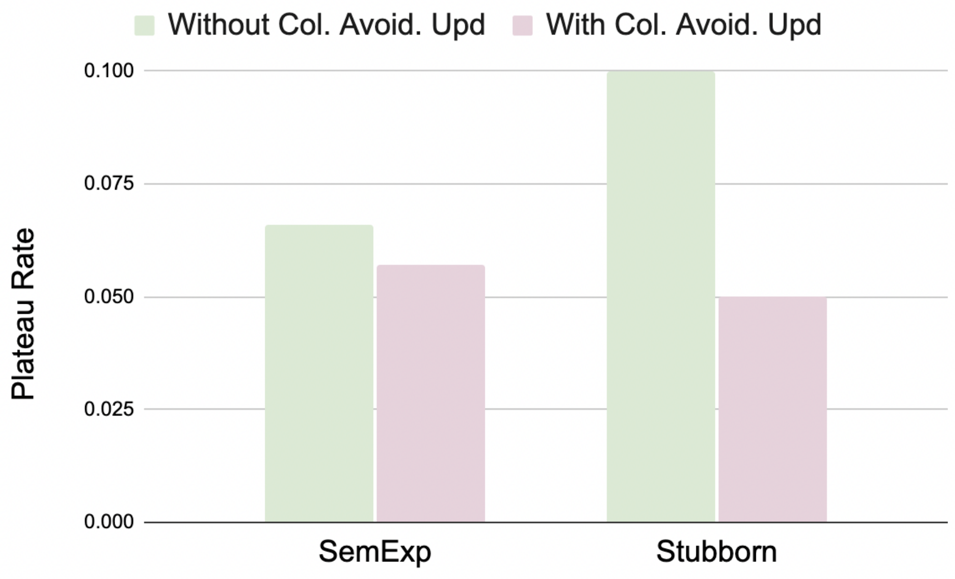

Figure 5: Plateau Rate of the SemExp Agent and the Stubborn Agent before and after the Collision Avoidance Update. The lower the Plateau Rate is, the better. Stubborn benefited from Collision Avoidance Update more than SemExp did.

SemExp when both agents are without the collision avoidance update, but performs much stronger when both agents receive the collision avoidance update. This is because the Stubborn Agent gets trapped more easily since its goal is fixed, which forces the agent to go to the corners of the rooms more often. Therefore the Stubborn Agent benefited more from the Collision Avoidance Update, as demonstrated in Fig. 5.

6.2 Collision Avoidance

The Stubborn Agent used 4 methods to decrease failures caused by plateau, as discussed in Sec. 5.7. We present an ablation analysis to show how much improvement each method made. Table 3 shows the performance of the Stubborn Agent with different ablations on the collision avoidance update. All of the 4 collision avoidance strategies decreased the plateau rate and improved the GT exploration rate, with the biggest improvement made by maintaining two versions of the collision obstacle map (ObsSize).

	Collision Avoidance Update				Performance	
Agent	**BFUntrap**	**ObsSize**	**ColAver**	**VisPath**	**Plateau Rate**	**GT Exploration Rate**
I					0.100	0.54
II	✓				0.087	0.59
III	✓	✓			0.067	0.64
IV	✓	✓	✓		0.052	0.64
V	✓	✓	✓	✓	**0.050**	**0.67**

Table 3: The Stubborn Agent with ablations indicated on the left and performance on the right. A lower Plateau Rate and a higher GT Exploration Rate indicates stronger performance. Each part in the Collision Avoidance Update correspond to a method described in Sec. 5.7: BFUntrap is the "Brute Force Untrapping Mode", ObsSize is the "Obstacle Size", ColAver is the "Collision Aversion in Global Planner", and VisPath is "Marking visited path as traversible".

6.3 Object Identification

Training We use the Naive Bayes Classifier as the binary classifier described in Sec. 5.6. We collect training data using the validation episodes of Matterport 3D. We collect training data in the validation episode instead of the training episode because the training episodes are already used to train the object segmentation model and therefore not accurate indicators of how the object segmentation model behaves in the testing split.

We train a separate Naive Bayes Classifier for each object goal category and each testing scene. The Classifier for an object goal category only uses data gathered from the object goal category for training, while the classifier for a testing scene excludes data gathered from the scene to prevent training and testing on the same scene.

Baseline We compare the Stubborn agent with a naive baseline (Naive). The Naive baseline differs from the Stubborn agent only in the object detection module: it uses the simple object detection baseline that is similar to SemExp described at the beginning of Sec. 5.6, where the agent uses the maximum confidence score reported by the semantic segmentation model as the only indicator for whether the goal is found.

Results We consider results only from object goal categories that appeared in more than 4 scenes in the 2000 validation episodes to make sure the classifier is trained with sufficient data. Among the 21 object categories, 15 of them satisfied the requirement.

Goal	Stubborn	Naive	Improvement %
stool	**0.16**	0.09	77.78
toilet	**0.43**	0.35	22.86
sink	**0.31**	0.26	19.23
chest_of_drawers	**0.19**	0.16	18.75
plant	**0.33**	0.28	17.86
bed	**0.51**	0.44	15.91
cabinet	**0.34**	0.3	13.33
table	**0.61**	0.56	8.93
counter	**0.39**	0.36	8.33
chair	**0.47**	0.46	2.17
sofa	0.31	0.31	0
seating	0.69	0.69	0
picture	0.27	**0.3**	-10
cushion	0.46	**0.56**	-17.86
towel	0.12	**0.15**	-20
Average	**0.37** $\pm$0.03	0.35 $\pm$ 0.03	6.55

Table 4: Success Rate of Stubborn and Naive Agent on 15 semantic categories. Categories with improvements are marked green, while categories with worse performance are marked red.

Results are presented in Table 4. The agent has increased performance in most categories with the detection module update, but not all categories. Overall, the agent performance increased by about 7%.

Feature Importance The feature vector of an object has 5 dimensions described in Sec. 5.6. We present in Fig. 6 the permutation feature importance of each dimension. Cumulative Confidence Score, Total Number of Frames, and Conflict Object Score are the most important features used by the Global Goal Detection module, which aligns with how humans make decisions using information across multiple frames.

6.4 Performance on the Habitat Challenge

We evaluate the Stubborn Agent on the 2021 Habitat Object Navigation Challenge with the test-standard split. Results in Table 5 show that our agent has the best performance in SPL and the second best performance in Success Rate.

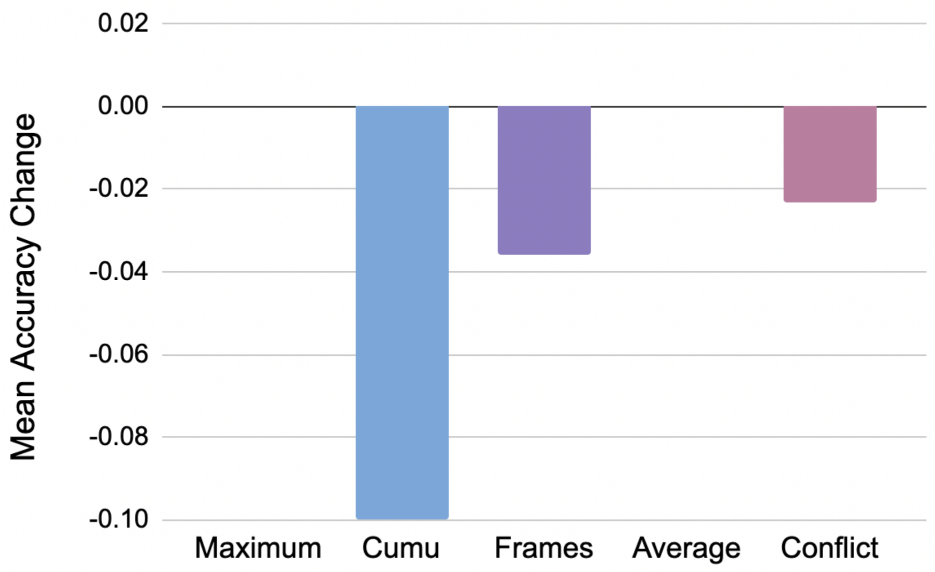

Figure 6: Permutation Feature Importance of the Object Feature Vector Described in Sec. 5.6. More drop in accuracy suggests higher importance. Maximum stands for Maximum Confidence Score, Cumu stands for Cumulative Confidence Score, Frames stand for Total Number of Frames, Average stands for Average Confidence Score, and Conflict stands for Conflict Object Score.

Models	**SPL**	**Success Rate**
(1) yuumi_the_magic_cat(Our Method)	**0.098**	0.237
(2) PONI (PF)	0.088	0.2
(3) TreasureHunt[18]	0.087	0.211
(4) Habitat on Web (IL-HD)	0.082	**0.244**
(5) EmbCLIP [15]	0.078	0.181
(6) Arnold (SemExp)^{2020} [4]	0.0707	0.1785
(7) Red Rabbit 6-Act Base (EEAux)^{2021} [30]	0.062	0.237
(8) Alstar (2RLs_sl_400)	0.054	0.144
(9) Clueless-Wanderers (PeterBot)	0.018	0.065
(10) BEyond-VRI-UFPR	0.002	0.004

Table 5: **2021 Habitat Object Navigation Challenge** We report the leaderboard entries on the Test Standard split. Entries with the superscript 2020 and 2021 indicate challenge winners in the corresponding year. Our Method (named Yuumi_the_magic_cat) is first in SPL and second in Success Rate, with a large improvement from SemExp which we built upon.

Licenses for Matterport3D:

References

[1] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

[2] Sean L Bowman, Nikolay Atanasov, Kostas Daniilidis, and George J Pappas. Probabilistic data association for semantic slam. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1722–1729. IEEE, 2017.

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

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

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[6] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[7] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[8] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram 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*, pages 2432–2437. IEEE, 2005.

[9] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *International Journal of Computer Vision*, 128:1311–1330, 2019.

[10] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, G. Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, P. Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Jindong Jiang, Lunan Zheng, Fei Luo, and Zhijun Zhang. Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation. *arXiv preprint arXiv:1806.01054*, 2018.

[14] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[15] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *arXiv preprint arXiv:2111.09888*, 2021.

[16] Yifan Liu, Chunhua Shen, Changqian Yu, and Jingdong Wang. Efficient semantic video segmentation with per-frame inference. In *ECCV*, 2020.

[17] Lingni Ma, J. Stückler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605, 2017.

[18] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *ICCV*, 2021.

[19] Arsalan Mousavian, Alexander Toshev, Marek Fiser, Jana Kosecka, and James Davidson. Visual representations for semantic target driven navigation. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8846–8852, 2019.

[20] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[21] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *CVPR*, 2018.

[22] 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. *ICCV*, pages 9338–9346, 2019.

[23] 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.

[24] Ayzaan Wahid, Austin Stone, Kevin Chen, Brian Ichter, and Alexander Toshev. Learning object-conditioned exploration using distributed soft actor critic. 2020.

[25] H. Wang, Weining Wang, and Jing Liu. Temporal memory attention for video semantic segmentation. *arXiv preprint arXiv:2102.08643*, 2021.

[26] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[27] Yi Wu, Yuxin Wu, Aviv Tamar, Stuart Russell, Georgia Gkioxari, and Yuandong Tian. Learning and planning with a semantic model. *arXiv preprint arXiv:1809.10842*, 2018.

[28] Brian 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'*, pages 146–151. IEEE, 1997.

[29] Wei Yang, X. Wang, Ali Farhadi, Abhinav Kumar Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2019.

[30] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable object navigation. *ICCV*, 2021.

[31] Joel Ye, Dhruv Batra, Erik Wijmans, and Abhishek Das. Auxiliary tasks speed up learning pointgoal navigation. *arXiv preprint arXiv:2007.04561*, 2020.

[32] Liang Zhang, Leqi Wei, Peiyi Shen, Wei Wei, Guangming Zhu, and Juan Song. Semantic slam based on object detection and improved octomap. *IEEE Access*, 6:75545–75559, 2018.

[33] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, 2017.