# NAVIGATION TO MULTIPLE SEMANTIC TARGETS IN NOVEL INDOOR ENVIRONMENTS

Siddharth Goel

A THESIS

in

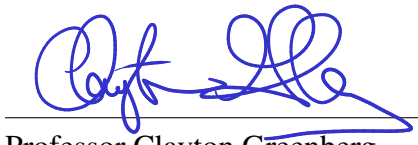
Data Science

Presented to the Faculties of the University of Pennsylvania in Partial

Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2021

Professor Kostas Daniilidis
Supervisor of Thesis Signature



Professor Clayton Greenberg
Data Science Projects Director Signature

# Acknowledgement

With a lot of pride, humility, and warmth, I'd like to express my gratitude to the following people who have made it possible for me to undertake and complete this work.

Georgios Georgakis and Bernadette Bucher, who have been an integral part of this work right from the start. Thank you for taking me under your mentorship and guiding me at every step with your invaluable feedback and advice. You guys have been extremely patient and helpful.

Special thanks to Prof. Kostas Daniilidis for giving me the opportunity to be a part of his group and pursue this research.

The Data Science department at UPenn - Prof. Clayton Greenberg and Staci Kaplan for their never ending support.

Finally, all the professors and instructors who have made big and small contributions to my UPenn journey in so many ways through courses, discussions, assignments, and above all by setting the success bar high.

# Abstract

Navigating to more than one semantic object using egocentric perception in an unseen indoor environment can be achieved using semantic map like memory and effective long-term goal planning. We propose the multi-object navigation task which requires the agent to navigate to more than one (unique and non-repetitive) semantic objects (e.g. fridge, TV, bed, chair, etc.) in an indoor environment. In our framework, the agent simultaneously predicts a semantic map of the environment based on egocentric RGB-D images and follows an uncertainty based goal selection policy that enforces it to pursue the closest object instance (from the target object list) at each time step and also balances exploitation of the semantic information along with exploration of the map. In addition, we leverage the novel method of predicting semantic maps outside the field of view of the agent as proposed in *Learning to Map (L2M)* [22]. Episodes for multi-object navigation are generated in Habitat simulator for Matterport3D dataset and experiments are conducted to demonstrate the validity of our proposed method. Finally, we improve the performance of L2M's semantic map prediction by countering the effect of extreme class imbalance through focal loss and incorporating temporal information (in the sequence of observations) by adding an LSTM layer in the architecture.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

The field of 'Internet AI'entails training models and agents on datasets of images, videos, and texts primarily curated from the internet and achieving state of the art results in each of those tasks. Some of the better performing algorithms/models and their corresponding tasks are - ResNet for image classification, transformer based language models such as BERT, GPT-2, and GPT-3 for language understanding, text classification, and text generation etc., and lastly Faster R-CNN, EfficientDet, and RetinaNet for the task of object detection are just to name a few.

The interpretability of these models and their consistent performance along with general advancements in the fields of deep learning, reinforcement learning, computer vision, and robotics have enabled research to take a step forward. Recently, there has been a surge in effort to train agents by making them interact with the environment, enabling the Gibsonian idea [24] that perceptual and robotic learning should be based on an ecological approach, solving tasks and problems approximating the real-world setting and scale.

Such ability of an agent to interact with it's surrounding environment and learn from those interactions is the overarching definition of Embodied AI. Some of the tasks which fall under the purview of Embodied AI include visual navigation[25, 1], vision-and-language navigation [2], and embodied question answering [17].

The Embodied AI research has also been bolstered by the development of various simulators such as Habitat-sim [40], AI2-THOR [30] and datasets such as Gibson [46] and Matterport3D [10]. These simulators and datasets aim to realistically replicate the physical world and allow to perform a faster, safer, cheaper and a more reproducible research in developing embodied AI agents.

As a result, it has become possible to train agents that can serve multiple purposes in indoor environments e.g. making life easier for elderly and disabled people if they could

just ask the agent to go and fetch the laptop from the bed or check if the gas stove is on. In order to accomplish such tasks, the agent should be able to perform auxiliary tasks such as ability to perceive and build a representation of its environment (semantic map prediction), select logical goal in the environment (long-term goal planning), and navigate to the goal locations (goal-driven navigation).

These auxiliary tasks constitute the sub-components of modern day visual navigation. In the context of Embodied AI, visual navigation task can be segregated based on the nature of goal - PointGoal, ObjectGoal, and AreaGoal [1, 5] where the goal is to navigate to specific points, objects, or areas using egocentric perception. In Object Goal Navigation [5], an agent is initialized at a random starting position and orientation in an unseen environment and asked to find an instance of an object category ('find/go to a chair') by navigating to it. If the agent is able to find any single instance of the target object category within the specified time steps the episode is considered successful.

However, practical real-life scenarios entail an agent to navigate to more than one logical location in our living spaces to fulfill an everyday task. For example, an agent getting an item from the kitchen to a person sitting on a bed or a chair. Such an agent needs to have the ability to start from its current location (random starting point) and navigate to the kitchen, identify objects such as oven, fridge, or sink, fetch the required item, and then navigate to bed/chair. Though completing this task in entirety involves solving for a combination of sub-tasks, we abstract this to the following navigation task - an agent starting from a random position in the environment and navigating to more than one semantic goal positions before ending the episode. Consequently, the multi-object navigation task is a more difficult task than object goal navigation given that the agent has to plan and traverse a longer path in order to navigate to more than one object. Apart from implicitly learning the relative semantics of the indoor environment e.g. bed and sofa are located in different rooms etc., the multi-object navigation agent must also be able to predict a robust semantic map of the

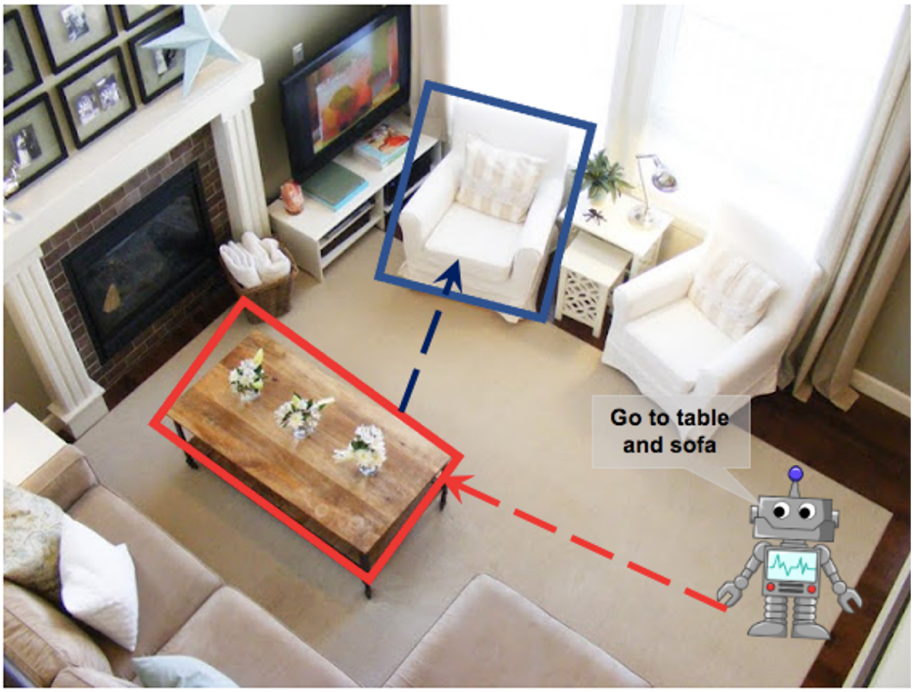


Figure 1.1: Navigating to two semantic objects in an indoor environment

environment.

We realize that a novel framework *Learning to Map (L2M)* [22] contains such a robust semantic map prediction module. L2M algorithm makes the agent learn to generate semantic maps outside the egocentric view of the agent using uncertainty estimated through disagreement of an ensemble of hierarchical segmentation models.

We propose a multi-object navigation technique which exploits the semantic map prediction module of L2M and leverages a novel goal-selection policy. The goal-selection policy balances exploitation of the semantic information and exploration of the map and enforces the agent to pursue closest semantic object instances. Additionally, we also delve deeper in the semantic map prediction module of L2M and make improvements to the semantic map results. Our main contributions in this work are as follows:

(i) Introduce a new framework for the multi-object navigation task in realistic 3D environments.

(ii) Propose a new goal-selection policy for multi-object navigation.

(iii) Improve the semantic map prediction performance of L2M algorithm.

(iv) Perform experimentation for 2 object navigation scenarios, show success results on Matterport3D [10] dataset using the Habitat simulator [40].

# 2. Background

## 2.1 Related Work

***SLAM Approaches:*** A survey of visual navigation methods from the late 90s to 2008 can be found in [6]. One of the most popular and proven methods for robotic navigation in novel environments is Simultaneous Localization and Mapping (SLAM). By definition, SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. SLAM is one of the most prominent methods to create map representation of the environment. Prior works such as [18, 4, 8] survey the history and evolution of SLAM techniques over the years. [18] mentions the success of two of the most popular earlier SLAM methods - Extended Kalman Filter(EKF) a state-space model with additive Gaussian noise and Rao-Blackwellized Particle Filter, a vehicle motion model as a set of samples of a more general non-Gaussian probability distribution. These were followed by a great majority of work in SLAM research which focused on improving computational efficiency while ensuring consistent and accurate estimates for the map and agent pose as presented in [4].

There have been huge advancements in SLAM techniques in the past years. The latest modern day SLAM involves learning based methods [12, 39, 44] in which the agent is able to reconstruct topological semantic maps of the environment from egocentric visual perception. The L2M algorithm is a learning-based method for semantic map prediction in which the agent learns to construct a global semantic map of an unseen environment from egocentric RGB-D observations as the agent moves in the environment.

***Exploration methods for navigation:*** The process of simultaneously planning efficient paths during map building is referred to as Active SLAM. Some of the prior work in Active SLAM are presented in [19, 29, 9, 42]. [9, 42] use information gain objectives leveraging

estimated uncertainty over the map to decide future actions. Other works [16, 12] use area-coverage, the area seen in the environment during navigation, as a reward function to guide exploration. Several other works [37, 11, 31, 34] predict maps for unseen regions and as a result accelerate navigation and map creation along with increasing robustness in the decision making process. [37] anticipates occupancy in unseen regions by rewarding agent actions that yield accurate occupancy predictions for the global environment map. The L2M approach leverages the uncertainty over the semantic classes in the unobserved areas and shows its effectiveness on exploring a novel environment.

***Modular learning approach:*** The methods [13, 11] perform the overall navigation tasks by using separate modules for semantic map prediction and goal selection policy. This allows for using existing pre-trained models for object detection and object segmentation for semantic map prediction. A map-like memory has proven to aid in navigation performance of the agent in many prior works such as [25, 15, 47, 39, 23]. Further, the modular approach has been shown to perform better than end-to-end approaches as it leverages structural regularities of indoor environments, is robust to state-estimation errors, and is more sample-efficient. L2M also comprises of a modular structure with separate semantic map prediction and goal navigation policy modules.

***Uncertainty Estimation:*** In L2M, the model (epistemic) uncertainty is estimated from the variance between the output of an ensemble of models [22]. In our work we use the same approach to leverage uncertainty. Using uncertainty as an information-gain based objective in vision based reinforcement learning was motivated from [35, 20, 27]. This uncertainty based objective is also used in our extension of L2M for active exploration during training [14, 7] and target driven navigation using upper and lower confidence bounds [3, 21] at test time.

***Multi-object Navigation:*** In [44] *multiON* task is proposed that involves navigation to an ordered sequence of artificial objects placed programmatically within realistic 3D

environments. The main goal of the work in [44] is to investigate the impact of maps - oracle, semantic, neural on visual navigation tasks. It majorly focuses on analyzing what information is useful in maps and how different types of map-like memory impact navigation performance across different task complexities. In [44], artificial cylindrical objects, which act as targets, are programmatically placed in the environment. The justification for this modification is to control the task complexity. But such a modification makes the semantics in the scene irrelevant to the task. Therefore, we propose navigating to real semantic objects in the environment instead of synthetically created targets. The multi-object navigation task proposed in this work is more realistic and difficult than [44].

Another solution [32] of the multiON task defined by [44], proposes auxiliary supervised learning tasks to predict if a target object has already been observed since the beginning of the episode along with the relative direction and euclidean distance to the object if observed while predicting the occupancy map. A non-learning/classical SLAM based method [28] builds an occupancy map using the RGB-D observation and simultaneously localizes goal objects on the map. The agent navigates to the target if found else continues exploration.

To solve for multi-object navigation task, we leverage the semantic map prediction module of L2M which predicts beyond the field of view of the agent to build the environment map and a goal-selection policy which selects high-confidence target locations closest to the agent's position.

## 2.2 Simulators and Datasets

As per the embodiment hypothesis [41], intelligence emerges in the interaction of an agent with an environment and as a result of sensorimotor activity. Running embodied AI systems in a physical state is time consuming, tedious, and expensive. In addition, acquiring large-scale action and interaction data of real environments is not trivial via the traditional dataset

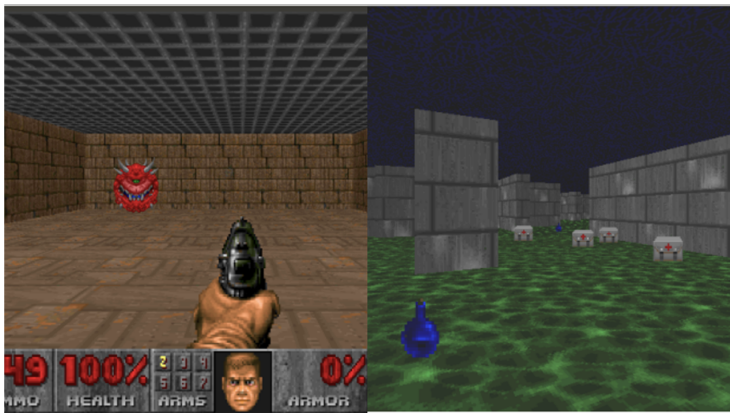


Figure 2.1: First-person perspective semi-realistic 3D scenes from ViZDoom

collection techniques. This produces the need for efficient simulators and datasets which replicate the real-world to the closest both in appearance and physics and are lightweight, customizable, re-configurable, easy-to-use, and fast. Some of the popular embodied AI simulators and datasets are discussed below.

### 2.2.1 Simulators

**ViZDoom:** The ViZDoom [26] was developed with an aim to provide a platform for visual reinforcement learning from first-person perspective. Till ViZDoom, Atari 2600 games were widely popular to train reinforcement learning bots from raw pixel information. ViZDoom attempted to improve upon Atari 2600 software toolkit, which provided non-realistic 2D environment with third person perspective, with its semi-realistic 3D environment and first-person perspective (as shown in figure 2.1). ViZDoom is built upon the popular first-person shooting (FPS) video game-Doom. ViZDoom provides raw visual information in the form of screen frames and a relatively realistic physics model to train the bot.

The bigger goal of our research is aimed at training embodied AI agents for indoor navigation. ViZDoom 3D environment is limited to the gaming environment of Doom and far from a realistic 3D scene of any indoor environment, making it less suitable for our use case.

Figure 2.2: Synthetic images scene renderings in AI2-THOR.

**AI2-THOR:** The House Of inteRactions (AI2-THOR) [48] is another popular simulation framework with high quality 3D scenes that enables visual interaction for agents. AI2-THOR has a variety of features including navigation, 3D asset library, object states, physics based interactions, dynamic lighting and multiple agents.

In AI2-THOR the agent can freely navigate (i.e. move and rotate) in various realistic indoor scenes, and is able to have low- and high-level interactions with the objects e.g., applying a force or opening/closing a microwave. This simulator has an integrated Unity3D physics engine that enables agent movement and object interactions. The physics engine has direct communication with Tensorflow (deep learning framework). AI2-THOR comprises of two main components iTHOR and RoboTHOR. In iTHOR the agent has the capability to interact with objects like toasting a piece of bread in a toaster. Currently, iTHOR has 120 rooms including bedroom, kitchen, bathroom, and living room with over 2000+ objects for various types of interactions such as throwing, pushing, opening etc. RoboTHOR is the platform to test embodied AI agents in simulation and physical world. RoboTHOR provides 89 simulated reconfigurable apartment scenes with 600+ objects out of which 14 have physical appearance. In its earlier versions, the main disadvantage of this simulator is the use of synthetic scenes (refer figure 2.2) which leads to a gap between the real-world and its representation. This has been improved in the later/current versions by including real counterparts for some apartment scenes.

Figure 2.3: Different task simulation capabilities and photorealistic nature of Habitat simulator. Image credit https://aihabitat.org/.

**AI-Habitat:** Habitat [40, 43] is one of the most recent and popular simulator platform for research in Embodied AI. Habitat comprises of two main components - *1) Habitat-sim:* a flexible, high-performance 3D simulator with configurable agents, multiple sensors, and generic 3D dataset handling (with built-in support for Matterport3D, Gibson, and Replica datasets) and *2) Habitat-lab:* a modular high-level library for end-to-end development of embodied AI algorithms – defining embodied AI tasks (e.g. navigation, instruction following, question answering etc. as shown in figure 2.3), configuring and training embodied agents (via imitation or reinforcement learning, or via classic SLAM), and benchmarking using standard metrics. The main features of habitat simulator are -

1. Agnostic to 3D scene datasets and allows users to use their own datasets.

2. High performance in rendering thousands of scenes per second on single or multiple clusters.

3. Support for a diverse sensor suite - RGB, depth, contact, GPS, compass sensors.

4. Highly configurable. Allows user to specify agent, environment, and sensor properties as per needs.

All the above mentioned features along with the speed, efficiency, and robustness of habitat make it a desirable choice for experimentation.

**Gibson:** Another simulator developed with a similar goal to provide a good enough replica of the intricate real-world to train and test visual perception agents is Gibson [46]. Gibson also has an integrated physics engine which enable the constraints of space and physics (e.g. collision and gravity) on the agents. Both the real-world like images for training and physical constraints on the agents are aimed at facilitating smooth and accurate transfer of models trained in simulation to real-world. One large constraint of the Gibson environment is that it provides simulation capabilities only for vision tasks and not for language or vision-language based tasks unlike habitat simulator. The Gibson virtual environment also comprises of a dataset by the same name and is discussed in details in the section 2.2.2 below.

### 2.2.2 Datasets

**Gibson:** The Gibson dataset [46] comprises of reconstructed images from 3D scanned real scenes of 572 full buildings composed of 1447 floors covering a total area of 211k $m^2$. Each scene has an associated 3D reconstruction, RGB images, depth images, and surface normal, while some of the scenes also have semantic object annotations. The scanned real images are corrected for lighting inconsistency and misaligned or missing artifacts across different views by a combination of geometric point cloud rendering and a neural network to produce a more real looking image. Both the Gibson engine renderings from the neural network and scanned images are said to be statistically indistinguishable to the agent.

**Matterport3D:** Matterport3D (MP3D) [10] dataset contains reconstructions of real indoor

Figure 2.4: RGB frames from two different scenes in Matterport3D dataset along with their corresponding depth images.

scenes from 90 buildings. The size of reconstructed 194,400 RGB-D images is 1280 x 1024. The real images were captured in 10,800 panorama with a Matterport camera in living spaces in private homes. Some sample RGB-D frames from the matterport3D dataset scenes are displayed in figure 2.4. The dataset has the following main features -

1. Samples human-height viewpoints uniformly throughout the entire environment.

2. Provides camera poses that are globally consistent and aligned with a textured surface reconstruction.

3. Includes instance-level semantic segmentations into region and object categories.

The dataset features a globally consistent registration of images to a surface mesh which gives way to semantic annotation, enabling efficient 3D interfaces for object and region category annotation from which labels projected into images can train deep networks for semantic segmentation. These features make Matterport3D highly suitable to perform semantic map prediction efficiently and use the same further downstream for navigation.

## 2.3 Learning to Map (L2M)

Learning to Map (L2M) is a novel framework proposed to accomplish the object-goal navigation task. L2M lays emphasis on learning contextual semantic priors in the indoor environment setting. Some examples of contextual semantic priors can be - 1) oven and refrigerator are mostly close to one another, 2) a sofa will mostly be far away from a refrigerator. This learning of contextual semantic priors is achieved implicitly by actively learning to generate semantic maps outside the field of view of the agent and leveraging the uncertainty over the semantic classes in the unobserved areas to decide on long term goals.

L2M algorithm addresses some of the limitations of the current learning based navigation approaches. Firstly, L2M does not learn to map pixels directly to actions. End-to-end reactive approaches [33, 48] lead to poor generalization. On the other hand, L2M learns to generate a map representation that enables the encoding of prior information about the geometry and semantics of a scene. Secondly, the learned map representation is not target dependent since it does not involve a goal-oriented navigation policy [15]. It rather predicts the location of a semantic target by leveraging the uncertainty of the presence of the semantic target at that location. This in turn leads to predicting semantics in unobserved regions of the map. The semantic map representation is irrespective of the target object and can be leveraged for multiple tasks.

The L2M pipeline (i) predicts occupancy and semantic regions in the unseen environment using an ensemble of hierarchical segmentation models that operates on projected top down maps, and (ii) estimates the model uncertainty through the disagreement [36] of ensemble models. This quantified uncertainty is used for defining objectives in planners to actively select long-term goals for semantic navigation.

The L2M module can be divided into three sub-components namely - (i) semantic map predictor, (ii) goal-selection policy, and (iii) point-goal local navigation policy.

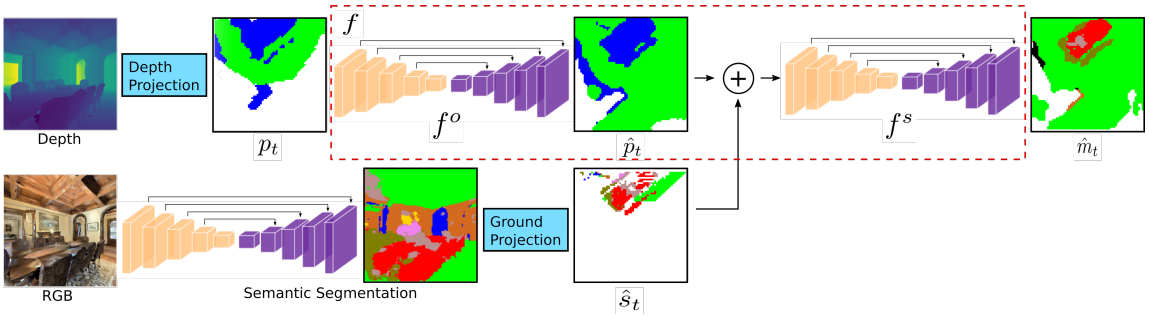



Figure 2.5: Predicting semantic maps from RGB-D images using an ensemble of models. Each model in the ensemble consists of two encoder decoder models for predicting occupancy and object semantics.

### 2.3.1 Semantic map prediction

The semantic map prediction is achieved through a hierarchical segmentation as shown in Figure 2.5. The semantic-map of the environment $m_t$ is predicted by passing the RGB-D observations through an ensemble of models $f = \{f_0, ...f_n\}$. Each model in the ensemble $f$ is a combination of two UNets $F_i^o$ and $F_i^s$. The model $F_i^o$ predicts the occupancy space - $C^o = \{occupied, free, unknown\}$ and the model $F_i^s$ predicts the top-down egocentric semantics of unobserved areas $\hat{m}_t$. Predicting the semantic map is a two-step procedure. In the first step, a UNet [38] $F_i^o$ is used to predict if a particular pixel in space is *occupied, free,* or *unknown* using the ground projection of the depth observations at time $t$. This gives the occupancy prediction $\hat{p}_t \in \mathbb{R}^{|C^o| \times h \times w}$ as the output.

$$\hat{p}_t = F_i^o(p_t; \theta_i^o) \qquad \hat{m}_t = F_i^s(\hat{p}_t \oplus \hat{s}_t; \theta_i^s) \tag{2.1}$$

In the next step, the occupancy prediction $\hat{p}_t$ is concatenated with a ground projection of the semantic segmentation $\hat{s}_t$ of the RGB observation at time $t$. The concatenated result is then provided as input to another UNet $F_i^s$ to predict the final semantic map crop $\hat{m}_t$ at time $t$. The segmentation model to predict $\hat{s}_t$ is pre-trained separately.

Apart from the class probabilities predicted by each model $f_i$ and the mean of these

probabilities over the ensemble, the per class uncertainty $(u)$ is also estimated using the ensemble predictions.

$$u_t = \mathrm{Var} f(p_t, \hat{s}_t) := \frac{1}{n} \sum_{i=0}^{n} (f_i(p_t, \hat{s}_t) - f(p_t, \hat{s}_t)^2) \tag{2.2}$$

Each model $f_i$ in the ensemble $f$ is initialized with random weights $\theta_i^o$ and $\theta_i^s$ and trained end-to-end for both occupancy and semantic prediction using pixel-wise cross-entropy losses.

$$L^o = -\sum_{c \in C^o} p_{t,c} \log \hat{p_{t,c}} \qquad\qquad L^s = -\sum_{c \in C^s} m_{t,c} \log \hat{m_{t,c}} \tag{2.3}$$

where $p_{t,c}$ and $m_{t,c}$ are the ground-truth labels and $\hat{p}_{t,c}$ and $\hat{m}_{t,c}$ are the predicted classes for a particular pixel. The overall loss function for each $f_i$ is defined as

$$L_{sem} = \lambda^o L^o + \lambda^s L^s \tag{2.4}$$

where $\lambda^o$ and $\lambda^s$ are the hyperparameters to weight the component loss functions. Finally, the above intermediate results are accumulated in the global maps for occupancy $M^o \in \mathbb{R}^{|C^o| \times H \times W}$, semantics $M^s \in \mathbb{R}^{|C^s| \times H \times W}$ and uncertainty $M^u \in \mathbb{R}^{|C^s| \times H \times W}$ over the episode. Note that $C^o$ and $C^s$ represent occupancy and semantics space respectively.

$$M_t^o = \gamma^o(M_{t-1}^o, p_t, x_t); \qquad M_t^s = \gamma^s(M_{t-1}^s, \hat{m}_t, x_t); \qquad M_t^u = \gamma^u(M_{t-1}^u, u_t, x_t) \tag{2.5}$$

where each $\gamma$ function transforms the local egocentric region to geocentric with respect to the relative pose $x_t$ of the agent, and aggregates the information into the respective map. The observed regions of the map are updated with zero uncertainty.

### 2.3.2 Goal Selection Policy

After accumulating the RGB-D observations of the agent in the global semantic map, the next step is to select the goal location of the target class $c$ at each time step $t$. Let $f$ represent the ensemble used in the semantic map prediction module (refer figure 2.5). For class $c$, the ensemble estimates $P_c(m_t|p_t,\hat{s}_t)$, the probability of class $c$ for all locations on the semantic map $m_t$, given observation $p_t$ and semantic segmentation $\hat{s}_t$. The target class uncertainty is given by the variance - $\text{Var} f_c(p_t,\hat{s}_t;\theta)$ over the target class predictions by each ensemble model.

Goals are selected using the upper confidence bound (equation 2.6) of the estimate $P_c(m_t|p_t,\hat{s}_t)$ to select locations not only with high payoffs but also with high information gain (balancing exploration and exploitation).

$$P_c(m_t|p_t,\hat{s}_t) \leq \mu_c(p_t,\hat{s}_t) + \alpha_1 \sigma_c(p_t,\hat{s}_t) \tag{2.6}$$

where $\alpha_1$ is a hyper-parameter. The values of $\sigma_c$ - the standard deviation of the target class probability and $\mu_c(p_t,\hat{s}_t)$ the mean estimate of the ensemble models - are given in expression 2.7.

$$\sigma_c(p_t,\hat{s}_t) = \sqrt{\text{Var} f_c(p_t,\hat{s}_t;\theta)} \qquad \mu_c(p_t,\hat{s}_t) = \frac{1}{N}\sum_{i=1}^{N} f_c(p_t,\hat{s}_t;\theta_i) \tag{2.7}$$

Finally, the goals are selected using the policy given by equation 2.8.

$$\arg\max_{l_j}(\mu_c(p_t,\hat{s}_t) + \alpha_1 \sigma_c(p_t,\hat{s}_t)) \tag{2.8}$$

The rest of the combinations of the mean and the standard deviation estimate (shown in table 2.1) are considered as well in the L2M paper. The best results were given by the strategy which selects goals (location on the map) based on high probability estimate

$P_c(m_t|p_t,\hat{s}_t)$ and high uncertainty $\mathrm{Var} f_c(p_t,\hat{s}_t;\theta)$.

Table 2.1: Goal selection strategy options

| $\mu_c(p_t,\hat{s}_t)$ | $\sigma_c(p_t,\hat{s}_t)$ | **Interpretation** |
|---|---|---|
| High | Low | Found target with high confidence |
| High | High | Found target with low confidence |
| Low | High | Not certain that the target is found |
| Low | Low | Target not found with high confidence |

### 2.3.3 Local Policy

Once the goal is selected, a deep reinforcement learning, state-of-the-art algorithm for point-goal navigation task, called DD-PPO [45] is used to navigate the agent from its current location to the selected goal location in the environment by the most optimal route. The DD-PPO model takes as input the egocentric depth observation and the current goal converted from a 2D map coordinate to a relative distance and heading and outputs the next navigation action for the agent at each time-step $t$.

# 3. Methodology

## 3.1 Improving semantic map prediction in L2M

During the detailed investigation of the L2M architecture, it was realized that the L2M algorithm performance was being compromised by not considering two important factors. Firstly, there is a huge class imbalance between images of structural objects (floors, walls) and the images of semantic objects (chair, table, bed). In the original L2M paper, there is no acknowledgement of the drawbacks due to the class imbalance and hence no measure to counter the same.

Secondly, the images captured by the agent upon moving from one location to another in the environment naturally produces a sequence of observations. This sequence is meaningful in nature and can be leveraged by the agent, if captured, in learning semantic information such as table and chair or kitchen equipment such as oven and gas or oven and microwave most of the times occur together.

Based on prior knowledge of better deep learning practices for computer vision tasks, we made three modifications to the original L2M model. In this section, we demonstrate improvement in the semantic map prediction performance of L2M in the following ways.

(i) ***Use Focal Loss in place of Cross-Entropy Loss***

The L2M paper uses Cross Entropy (CE) Loss as the loss function for both the occupancy prediction $F_i^o$ and semantic map prediction $F_i^s$ network. Intuitively, we know that we have a class imbalance in the observations of the agent. Majority of the frames observed by the agent contain objects such as floors and walls whereas the number of images containing semantic objects to be detected such as chair, table, sofa, fridge, bed etc is comparatively very small.

As a result, even a small CE loss such as 0.2 for the correctly identified walls, floors,

and free space (easy negatives) overwhelm the loss function and the network is barely able to train on frames containing actual semantic objects (hard positives).

To counter the effect of extreme class imbalance in the presence of exponentially more number of easy negatives and lesser hard positives, we replace the CE loss with the focal loss. The focal loss achieves this by multiplying a modulating factor $(1-p)^{\gamma}$ to the CE loss as shown in (3.2) where $p$ is the probability value predicted for the target class.

The cross-entropy loss is defined in (3.1) and the focal loss is defined in (3.2) where N is the number of classes and $\gamma$ is a tunable hyper-parameter. Also, notice that when $\gamma = 0$ the focal loss is equal to cross-entropy loss.

$$\text{CE Loss} = -\sum_{i=1}^{N} y_i \log(p_i) \tag{3.1}$$

$$\text{Focal Loss} = -\sum_{i=1}^{N} y_i (1-p_i)^{\gamma} \log(p_i) \tag{3.2}$$

As a result, if an easy example is correctly predicted with probability $p_i = 0.9$ the loss is weighted down by multiplying $(1-0.9)^{\gamma}$ by the focal loss. For example, if $\gamma = 2$ then this loss will be 100 times smaller than the cross-entropy loss. On the other hand, a hard example with a predicted probability of 0.2 is given a weight 0.64. In summary, the correct easy example was multiplied by a factor of 0.01 whereas a mis-classified hard example was multiplied by a factor of 0.64. Therefore, countering the effect of disproportionate number of easy and hard examples.

(ii) ***Tune the weights for spatial loss and object loss***

The total loss for the semantic map prediction model is the sum of the weighted losses

corresponding to the occupancy prediction model ($L^o$) and the semantic prediction model ($L^s$) and can be denoted as — $\mathbf{L_{sem}} = \lambda^{\mathbf{o}}\mathbf{L^o} + \lambda^{\mathbf{s}}\mathbf{L^s}$.

Though the L2M paper proposes the idea of weighting the two losses differently but the values for $\lambda^o$ and $\lambda^s$ are kept as 1 for all the experiments. We fine-tune these hyper-parameters to improve the overall performance of the model. Going by the same intuition as for using focal loss, we believe that the observations comprising objects are much less in number as compared to their counter observations. Therefore, the weight for the object/semantic loss should be much higher than the weight for the spatial/occupancy loss. Using a higher weight for $L^s$ *i.e.* ($\lambda^s >> \lambda^o$) makes the model put more emphasis on identifying objects and does not let the high frequency samples dominate the loss function.

(iii) ***Incorporate temporal information in the map prediction model by using LSTM layer***

A Recurrent Neural Network (RNN) is known to be good at modelling sequence data. In the object goal navigation scenario, each episode is a sequence of observations. The L2M paper only uses a ResNetUNet architecture for both occupancy and semantic map prediction. This architecture fails to take temporal information into consideration. We hypothesize that introducing an LSTM layer in the neural net architecture will maintain temporal consistency among the sequence of RGB-D egocentric observations and improve the performance of semantic map prediction. Figure A.1 shows our proposed network modification by placing an LSTM layer between the encoder and decoder of the ResNetUNet.

## 3.2 Multi-Object Navigation

A logical extension of object goal navigation is multi-object navigation. Simply put, multi-object navigation entails the agent navigating to more than one unique object categories. In this case, the agent is provided a list of target objects (unique and non-repetitive) and is expected to navigate to any single instance of all the specified target objects within the permissible steps limit starting from a random position in the environment. In this task definition, we *do not enforce a sequence* and the agent can navigate to the unique object instances in any order.

The basic assumption of the object goal navigation stays true for multi-object goal navigation case i.e. a map of the environment is not provided and the agent must only use its sensory input (RGB-D camera and a (noiseless) GPS+Compass sensor) to navigate.

The multi-object goal navigation task can be defined for $N$ objects. The difficulty of the task increases with increasing number of target objects in the scene. Some examples for 2-object navigation tasks are find a table and a chair, or find a fridge and an oven simultaneously. Note that the target object categories must be unique e.g a scenario in which the agent is expected to navigate to bed, fridge, and bed is not valid. Similarly, we can define a task for $N$ object navigation by defining $N$ unique semantic target object categories.

We extend the Learning to Map (L2M) (Section 2.3) algorithm for multi-object navigation. The multi-object navigation pipeline leverages the ensemble of hierarchical segmentation models to predict a semantic map of the unseen environment as described in section 2.3.1. Using the semantic map and the output of the ensemble of hierarchical segmentation models the agent selects its long-term goal using the following policy objective.

### 3.2.1 Multi-Object Policy Objective

We seek to incentivize pursuing success over short length paths. To this end, we propose selecting paths in the L2M navigation framework with an objective balancing exploitation of semantic information with exploration of the map.

Let $C$ be the collection of $N$ distinct semantic targets (object classes are not repeated). For each class $c \in C$, we choose the top $k$ values of $\mu_c(p_t, \hat{s}_t) + \alpha_1 \sigma_c(p_t, \hat{s}_t)$, the upper bound of $P_c(m_t | p_t, \hat{s}_t)$ where $\mu_c(p_t, \hat{s}_t)$ is the mean output of the ensemble models, $\sigma_c(p_t, \hat{s}_t)$ is the standard deviation of the output of the ensemble models for class $c$, $p_t$ is the observation, and $\hat{s}_t$ is the semantic segmentation ground projection (refer figure 2.5). Note that these notations are same as the notation and quantities introduced in L2M (section 2.3.2).

We use these upper bounds and associated locations on the global map $M_t$ to construct and score a set of candidate paths $\rho$. The candidate path with the highest score is selected for traversal. On a fixed interval, set by a hyperparameter, the path is re-planned using the above strategy for the set of semantic targets which are yet to be navigated. Candidate paths are constructed by choosing every combination of locations where one candidate for each class is selected.

Each of the $kN$ paths is ordered by path length measured in euclidean distance (for which we have access to ground truth values). To elaborate the point further, the sum of euclidean distance from agent's position to class 1 instance and the euclidean distance from class 1 instance to class 2 instance constitute a path length for 2-object navigation. Similarly, path length is calculated for all the combinations of traversal i.e. agent's position to class 1 instance to class 2 instance or agent's position to class 2 instance to class 1 instance (again for 2-object navigation) whichever is shorter. Paths which are not traversable are not considered and discarded.

Let us denote the set of all $kN$ candidate paths by $\rho$ and $j$ denote the node index in the

path $\rho_i$. For example, consider the path: agent's start position -> table -> chair. Then $j = 0$ corresponds to agent's start position, $j = 1$ represents table, and for chair $j$ is equal to 2.

With the notation established as above, a policy which chooses the shortest path connecting the agent's position to all the semantic target objects at each time step and takes advantage of the agent's iterative selection of best path as it navigates is given by 3.3.

$$\arg\max_{\rho_i \in \rho} \sum_{j=0}^{N-1} \alpha_0^{-j} \left( \mu_j(p_t, \hat{s}_t) + \alpha_1 \sigma_j(p_t, \hat{s}_t) - \alpha_2 d_{j,j+1} \right) \tag{3.3}$$

where $d_{j,j+1}$ is the euclidean distance between $j^{\text{th}}$ and $(j+1)^{\text{th}}$ node and $\alpha_0 > 1$ is another hyperparameter. Note that $d_{0,1}$ will be the euclidean distance between agent's starting position and the first node on the path. The quantities $\mu$ and $\sigma$ are the same as described above and correspond to the mean and standard deviation of the assigned semantic object class at node $j$ during candidate path construction.

# 4. Experimental Results

## 4.1 L2M - Semantic Map Prediction

We implemented the L2M code along with our proposed changes. The implementation details of the network are mentioned in table A.2 along with the habitat simulator configuration in table A.1 in the appendix section.

Both training and validation were performed on the Matterport3D (MP3D) [10] dataset using the Habitat [40] simulator. MP3D contains reconstructions of real indoor scenes with large appearance and layout variation, and Habitat provides continuous control for realistic agent movement. The standard train/val split, which contains 56 scenes for training and 11 for validation, is used for experimentation. During testing, the model is evaluated on 17900 test examples collected in the 11 validation scenes which had not been observed during training and the results are compared to the original L2M paper results.

The performance of semantic map prediction model is evaluated using the three common image segmentation metrics - F1 score, Intersection over Union (IoU), and Accuracy.

A qualitative example of a predicted semantic map of a bedroom scene is shown in figure 4.1.

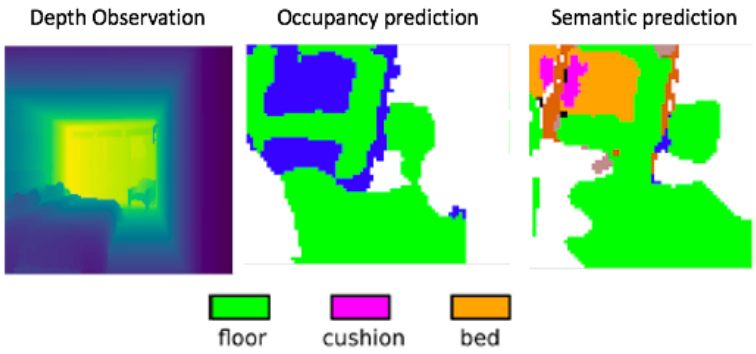



Figure 4.1: Semantic map prediction results of a bedroom scene

### 4.1.1 Selecting the optimal weights for occupancy and object detection loss function

We conducted experiments by training different models for different values of $\lambda^o$ and $\lambda^s$. Initially, we started with both the weights $\lambda^o$ and $\lambda^s$ equal to 1 for the purpose of replicating the L2M paper results. This also served as a baseline to compare the results using different weights.

In the following experiments, we started to increase the weight corresponding to the semantic map prediction model in factors of 10. In other words, the ratio of the two losses - $\frac{\lambda^s}{\lambda^o}$ was increased in the order 1, 10, 100, and 1000, putting this much times more weight on the semantic loss.

From figure 4.2 we observe the following things. Initially, there is no considerable improvement in the mean F1 scores for semantic/object and occupancy/spatial as the weights ratio — $\frac{\lambda^s}{\lambda^o}$ is increased to 10.

The mean F1 score metric is almost same as the baseline for both semantic and occupancy models. Upon gradually increasing the weights ratio to 100 a drastic improvement in performance of the semantic map model is observed along with a very small improvement in the occupancy model mean F1 score. Finally, as the weights ratio is increased beyond 100 we see a dip in the mean F1 score for both semantic and occupancy models.

The occupancy mean F1 score remains almost same for weights ratio 1, 10, and 100. This indicates that the occupancy prediction is a relatively easier problem to solve and putting a 100 times lower weight on the occupancy loss than its counterpart doesn't affect the performance of the model on predicting occupancy. These observations are in line with our intuition. Increasing the weight for the semantic loss makes the model learn object detection better since it is a relatively harder problem than occupancy prediction. At the same time, increasing the weight beyond 100 leads to a dip in performance as the model is focusing

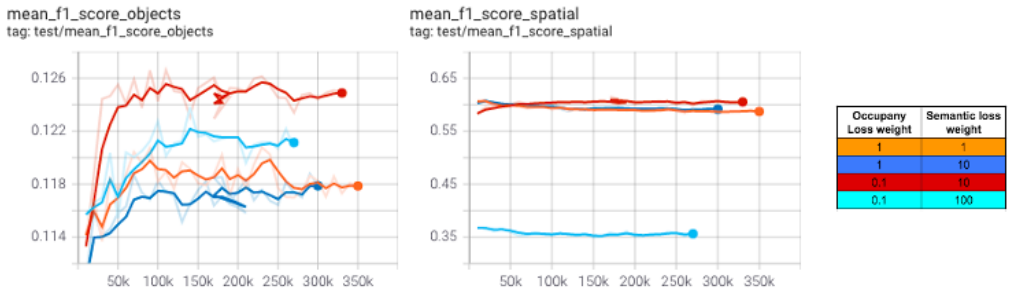



Figure 4.2: Mean F1 score for spatial and object prediction for different loss weights

entirely on object detection and the learning corresponding to occupancy prediction is barely taking place. We must recall that the semantic map model takes the output of occupancy prediction as input. Therefore, it is important for the end-to-end model to maintain a balance and do well on both the sub-components.

For the purpose of ablation studies, we also trained models by giving more weights to occupancy prediction loss $\lambda^o$ than semantic prediction loss $\lambda^s$. As expected, we observed worse performance than the baseline for these models.

### 4.1.2 Selecting the best loss function for occupancy prediction and semantic prediction models

We replaced the cross-entropy (CE) loss for semantic prediction model by focal loss and observed a considerable improvement in the mean F1 scores for semantic/object and occupancy/spatial as compared to the baseline L2M model results.

The focal loss prevents the high number of easy examples from overwhelming the loss function and propagates a balance in training of easy negatives and hard positives. This is very well confirmed by our results in figure 4.3.

Next, we also replaced the CE loss with focal-loss for the occupancy prediction model along with semantic prediction model. Again an improvement in the performance of the mean F1 score for objects was observed as compared to the L2M baseline. But, the

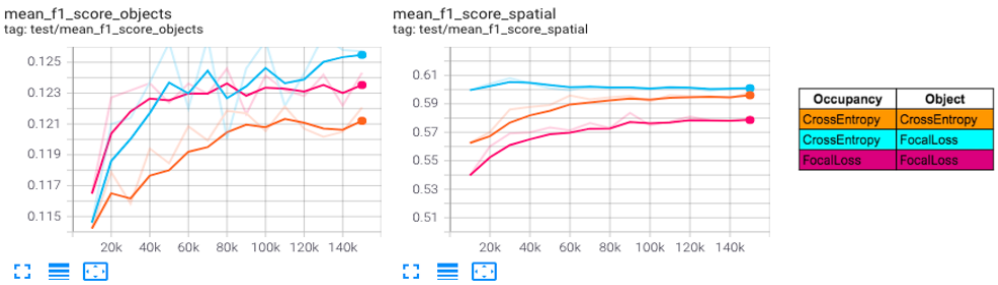



Figure 4.3: Mean F1 score for spatial and object prediction for different loss functions

performance was not optimal for the mean F1 score for spatial/occupancy. We believe the following reasons account for these observations.

The spatial space has just three labels - {*free, occupied, unknown*}. Also, the three labels are quite proportionate to each other in the frames. The modulating factor of the focal loss weighs down the spatial space loss more than required and hurts the performance of the model. In figure 4.3 we observe that the mean F1 score across models is best for the combination of CE loss (occupancy prediction) and focal loss (semantic prediction). The weights of the two losses were fine-tuned in a separate experiment and the values $\lambda^o = 0.1$ and $\lambda^s = 10$ give the best performance metrics.

### 4.1.3 Adding an LSTM layer to the network architecture

Finally, we performed the third set of experiments by modifying the network architecture (figure A.1) to capture temporal information. We added an LSTM layer between the encoder and decoder of the ResNetUNet to achieve this. The results of this experiment in figure 4.4 indicate that adding an LSTM layer does improve the performance of the baseline L2M model. As per our earlier hypothesis, we believe that using an LSTM layer helps the model in building an episodic memory and retain structural information better *e.g.* pair of objects like table and chairs, sofa and cushion, or fridge and oven are usually placed together. In case of occupancy model, as well the spatial information *e.g.* the continuous nature of floor

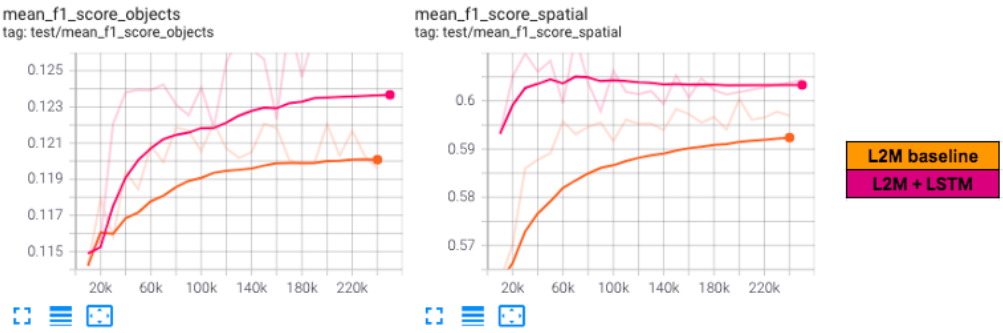



Figure 4.4: Mean F1 score for spatial and object prediction with LSTM layer

and walls are captured better. This led to an improved mean F1 score for both semantic and occupancy model as compared to the baseline. It is also noted that the mean F1 score for

Table 4.1: Best performing model loss function and weights configuration

| **Parameter** | **Value** |
|---|---|
| Occupancy Loss | Cross-entropy |
| Semantic Loss | Focal Loss |
| $\lambda^o$ | 0.1 |
| $\lambda^s$ | 10 |
| LSTM layer | yes |

Table 4.2: Semantic map prediction results

| Occupancy Prediction | | | |
|---|---|---|---|
| **Method** | **Acc(%)** | **IoU(%)** | **F1(%)** |
| **L2M** | 65.2 | 45.5 | 61.9 |
| **L2M+FocalLoss+LSTM** | **66.0** | **46.5** | **63.0** |
| Semantic Prediction | | | |
| **L2M** | **31.2** | 20.1 | 30.5 |
| **L2M+FocalLoss+LSTM** | 29.0 | **21.1** | **31.7** |

occupancy prediction curve is almost a flat line across experiments after 60,000-70,000 steps. This indicates that occupancy prediction model is easier to train and reaches its optimal state faster as compared to the semantic prediction model.

Finally, we combined the results of our previous experiments and found the best performing model configuration as mentioned in table 4.1. The final metric scores for this configuration are recorded in table 4.2.

## 4.2 Multi-Object Navigation

### 4.2.1 Experimental Setup

Episodes for 2-object navigation are generated for the 10 scenes in the Matterport3D (MP3D) validation set using the Habitat simulator. The five semantic objects - 'chair', 'sofa', 'table', 'bed', and 'cabinet' comprise of the target object set. A total of 680 episodes are generated across all the 10 validation scenes by taking all combination pairs of each of these target objects - ['chair', 'table'], ['bed','cabinet'], ['bed','sofa'],['table','cabinet'], ['table','sofa'], ['cabinet', 'sofa'], ['table', 'bed'], ['chair','cabinet'], ['chair','sofa'], ['chair', 'bed']. Combinations of semantic object categories for which no navigable path existed between any two logical nodes in a scene were ignored. The agent is allowed to take a maximum of 500 steps in order to navigate to each of the semantic targets in an episode. The agent's action space comprises of moving a distance of $0.25m$ in the forward direction, turning $10^{\circ}$ to the left or right directions, and stop. These details are consolidated in table A.1. The agent can consider successful navigation to the semantic object if it is within $1m$ distance from the target. The episode begins with agent starting at a random location in the environment. The agent has access to RGB-D observations, gps, and compass (noiseless) from the simulator. It has no map of the novel environment and nor it has seen the environment before. The agent follows the policy introduced in section 3.2.1 and simultaneously builds the semantic map of the environment and navigates to specified semantic targets.

### 4.2.2 Metrics

The navigation performance of the agent for multi-object scenario is evaluated for the following metrics.

1. **Success:** An episode is considered successful '1' if the agent is able to navigate to all the target object categories with in the max number of steps before ending the episode. The success metric takes a value of '0' - *i)* if the agent ends the episode before navigating to all the target objects or *ii)* if the agent takes maximum number of steps before navigating to all the target object categories and the episode fails.

2. **Progress:** This is the ratio of target object categories that the agent is able to navigate to the total number of target object categories. For example, if the agent is able to navigate to 2 target object categories in a 3-object navigation scenario then progress is equal to $\frac{2}{3} = 0.66$. Progress is equal to success in one-object navigation scenario.

3. **Success weighted by Path Length (SPL):** The SPL metric quantifies the distance covered by an agent in a successful episode. SPL is calculated using the following expression - $Succ \cdot d/max(p,d)$ where *Succ* is the binary success indicator as defined above, $p$ is the total distance travelled by the agent in that episode, and $d$ is the length of the shortest route spanning agent's starting position and any one instance of all the target objects.

4. **Progress weighted by Path Length (PPL):** This metric is the progress analogue of SPL. It is to measure the distance covered by an agent in an unsuccessful episode. The distance ratio $d/max(p,d)$ is calculated same as in SPL where $p$ is the total distance travelled by the agent in that episode and $d$ being the length of the shortest route connecting the agent's starting position to the object instances navigated successfully by the agent. This ratio is multiplied by the progress metric to obtain PPL. The expression for PPL is written as - PPL $= Prog \cdot d/max(p,d)$ where *Prog* is the progress metric value for that episode.

We average the metric values across episodes and all object categories in our final results.

Table 4.3: Multi-object Navigation Experimental Results

| **Method** | **Success**(%) | **Progress**(%) | **SPL**(%) | **PPL**(%) |
|---|---|---|---|---|
| Multi-obj-L2M | 2.35 | 11.98 | 2.46 | 9.27 |
| Multi-obj-L2M-OS | 17.60 | 41.53 | 15.98 | 35.30 |

### 4.2.3 Results

Multi-object navigation experiments were conducted using the pre-trained semantic map prediction model of L2M and a new goal-selection policy as discussed in section 3.2.1. The mean success, progress, SPL, and PPL were recorded for all the 680 episodes across 10 validation set scenes of Matterport3D and all 10 semantic object categories combinations. Two sets of experiments were conducted. In the first experiment, *oracle_stop=False*, the agent must take stop decision by itself after recognizing a goal state. In the second set of experiments, *oracle_stop=True*, the agent has access to stop oracle. The experimental results across four target metrics for both set of experiments are tabulated in table 4.3.

The agent performs considerably better when it uses an oracle for the stop decision. The difficulty of the task is evident from the metrics values when the agent has no access to stop oracle and is totally dependent on the predicted semantic map to recognize a goal state. It is observed that the agent is able to navigate to both the semantic objects in a scene and end the episode successfully only 2% of the total 680 experiments. On top of it, the agent doesn't perform very well in navigating to just one object as well. The progress metric for *oracle_stop=False* is just 12%.

There is a drastic gain in all the four metrics when we replace the stop decision with an oracle. The success % increases from 2% to 17%, progress from 12% to 41.5%, SPL increases almost 6 times, and PPL gains 25% points. These results indicate that in the absence of oracle stop information the agent is not able to decide by itself that the target semantic object has been reached.

We also performed an analysis on the relative performance of the agent on the target

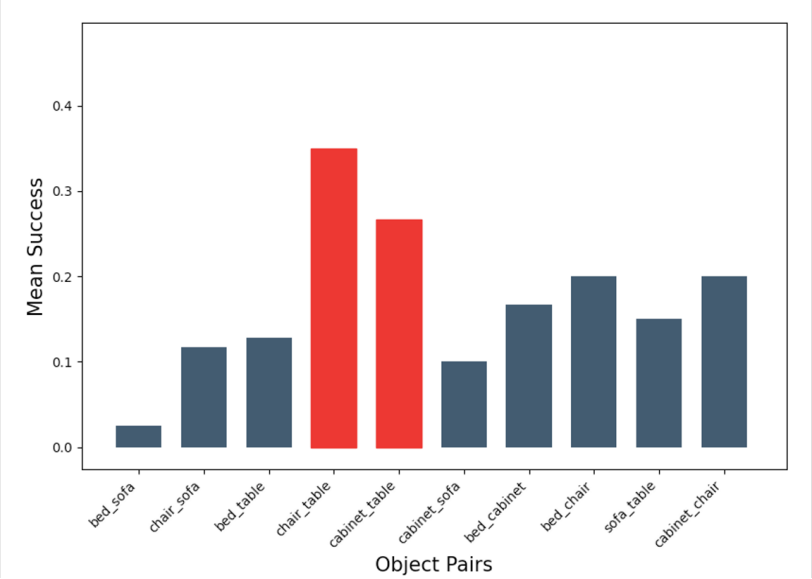


Figure 4.5: Mean success values across object pairs

object categories. From the graph in figure 4.5 it is observed that the agent is able to navigate to both the objects relatively more number of times when the target object categories are - ['chair', 'table'] and ['cabinet', 'table']. This is in agreement with the real-world layout of majority of the common living room spaces. In majority of times and environments, chairs and tables co-occur. Further, cabinets are mostly found in living rooms where it is highly probable to find a table. Additionally, object combinations such as *chair*, *table* and *cabinet*, *table* are in proximity to each other as compared to *bed*, *sofa* which are mostly in different rooms. We would like to point out that as per the success metric results in L2M [22] the agent performs reasonably well in object goal navigation task to semantic objects such as 'bed' and 'sofa'. Therefore, we believe that co-occurrence of semantic objects plays an important role in the success rate of multi-object navigation episodes.

Apart from co-occurrence, the frequency of the semantic objects also accounts for higher success %. We know from common knowledge that usually there are more instances of chairs and tables (can be found in both living room and bedroom) in an indoor environment as compared to instances of other semantic objects such as bed, sofa, and cabinet. The

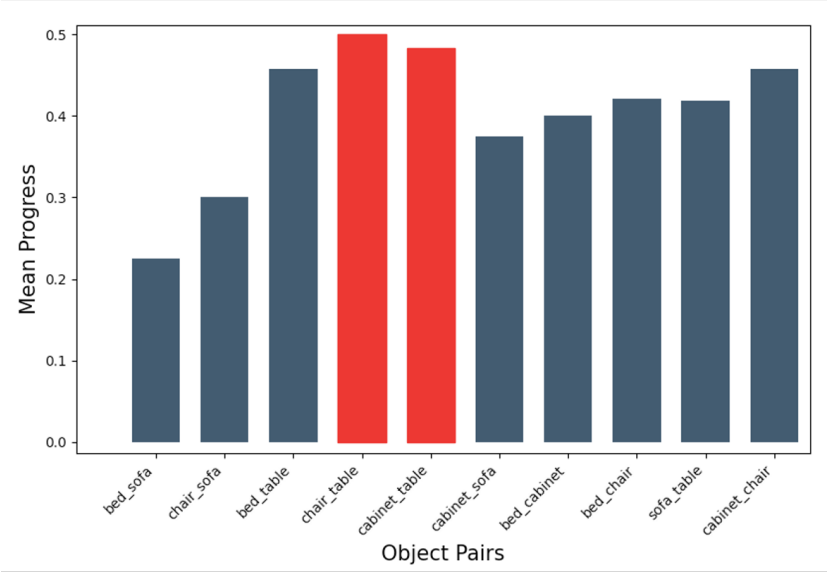


Figure 4.6: Mean progress values across object pairs

consistent higher success % for target object combinations containing higher frequency semantic objects - in this case *chair* and *table*.

On the other hand, we notice in figure 4.6 the progress values don't have a significant difference across target object combinations. It is seen in the bar plot that seven out of ten progress values are above 0.4 with the highest progress value being 0.5. These results point that the agent is able to navigate to one object across scenes 50% of the times. This result revalidates the difficult nature of navigating to more than one object. It shall be noted that since we are experimenting for 2-object navigation the progress value of 0.5 indicates successful navigation to one object out of two.

# 5. Conclusion

In this work, we formulate the multi-object navigation task and define it as navigating to more than one (unique and non-repetitive) semantic objects, following no specific order sequence between objects, in an indoor environment using egocentric RGB-D observations. We propose a novel goal selection policy which balances between exploitation of semantic information and exploration of the map and enforces the agent to pursue closest target object in the environment. This goal selection policy is leveraged in conjunction with the semantic map prediction module of L2M [22] algorithm. The L2M algorithm makes the agent learn to predict semantic map beyond the field of view of the agent. We implement the multi-object navigation framework in Habitat simulator [40] and conduct experiments using Matterport3D [10] dataset to demonstrate validity of our proposed approach. Finally, we delve deeper in the Learning to Map (L2M) algorithm and improve the semantic map prediction module by fine tuning its sub-components.

# A. Appendix

Table A.1: Habitat Simulator configuration

| Parameter | Value |
|---|---|
| Max. episode steps | 500 |
| Sensors | ['RGB', 'Depth'] |
| Height of agent(m) | 1.5 |
| Task type | ObjectNav-v1 |
| Possible Actions | ['Stop', 'Move Forward', 'Turn Left', 'Turn Right'] |
| Move forward distance (cm) | 25 |
| Turn left/right angle | 10° |

Table A.2: L2M semantic map prediction model details

| Parameter | Value |
|---|---|
| Epoch | 40 |
| Batch Size | 12 |
| Episode Length | 10 |
| Initial Learning Rate | 2e-4 |
| Learning Rate decay | 0.99 |
| Optimizer | Adam |
| no. of spatial classes | 3 |
| no. of object classes | 27 |
| image crop size | 64 x 64 |

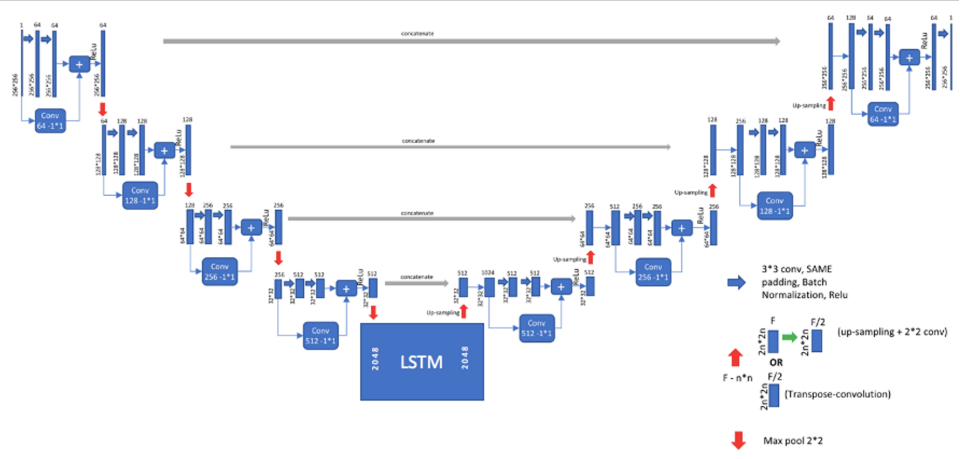



Figure A.1: Model architecture with LSTM layer between the encoder and decoder

# Bibliography


[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, 2018.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

[4] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117, 2006.

[5] 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, 2020.

[6] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53:263–296, 2008.

[7] Bernadette Bucher, Karl Schmeckpeper, Nikolai Matni, and Kostas Daniilidis. An adversarial objective for scalable exploration. *International Conference on Intelligent Robots and Systems*, 2021.

[8] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[9] L. Carlone, J. Du, M. Kaouk Ng, M. Indri, and B. Bona. Active SLAM and exploration with particle filters using Kullback-Leibler divergence. 75(2):291–311, 2014.

[10] 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.

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

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

[13] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. *CoRR*, abs/2004.05155, 2020.

[14] Devendra Singh Chaplot, Helen Jiang, Saurabh Gupta, and Abhinav Gupta. Semantic curiosity for active visual learning, 2020.

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

[16] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation, 2019.

[17] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering, 2017.

[18] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[19] Hans Jacob S. Feder, John J. Leonard, and Christopher M. Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.

[20] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[21] Nicolas Galichet, Michèle Sebag, and Olivier Teytaud. Exploration vs exploitation vs safety: Risk-averse multi-armed bandits, 2014.

[22] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation, 2021.

[23] Georgios Georgakis, Yimeng Li, and Jana Kosecka. Simultaneous mapping and target driven navigation. *arXiv preprint arXiv:1911.07980*, 2019.

[24] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[25] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation, 2019.

[26] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning, 2016.

[27] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision?, 2017.

[28] Junho Kim, Eun Sun Lee, Mingi Lee, Donsu Zhang, and Young Min Kim. Sgolam: Simultaneous goal localization and mapping for multi-object goal navigation, 2021.

[29] Thomas Kollar and Nicholas Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27(2):175–196, 2008.

[30] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2019.

[31] Yiqing Liang, Boyuan Chen, and Shuran Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation, 2021.

[32] Pierre Marza, Laetitia Matignon, Olivier Simonin, and Christian Wolf. Teaching agents how to map: Spatial reasoning for multi-object navigation, 2021.

[33] Arsalan Mousavian, Alexander Toshev, Marek Fiser, Jana Kosecka, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation, 2019.

[34] Medhini Narasimhan, Erik Wijmans, Xinlei Chen, Trevor Darrell, Dhruv Batra, Devi Parikh, and Amanpreet Singh. Seeing the un-scene: Learning amodal semantic maps for room navigation, 2020.

[35] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *CoRR*, abs/1906.04161, 2019.

[36] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement, 2019.

[37] Santhosh K. Ramakrishnan, Ziad Al-Halah, and Kristen Grauman. Occupancy anticipation for efficient exploration and navigation, 2020.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[39] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation, 2018.

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

[41] Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artif. Life*, 11(1–2):13–30, jan 2005.

[42] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. pages 65–72, 06 2005.

[43] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021.

[44] Saim Wani, Shivansh Patel, Unnat Jain, Angel X. Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation, 2020.

[45] 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, 2020.

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

[47] Jingwei Zhang, Lei Tai, Ming Liu, Joschka Boedecker, and Wolfram Burgard. Neural slam: Learning to explore with external memory, 2020.

[48] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning, 2016.