UNIVERSITY OF CALIFORNIA SAN DIEGO

Navigation in Everyday Settings Using Semantics and Language

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science

by

Srirangan Madhavan

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Henrik Christensen for his immense support and for giving me the freedom to pursue these projects. His guidance has been invaluable. I would also like to thank my mentor Anwesan Pal, for taking me under his wing and giving the opportunity to work in this amazing field. I am grateful for all that I learnt and for the numerous personal advise.

I also extend my gratitude to the members of my thesis committe, Professor Hao Su and Professor Xiaolong Wang for their valuable inputs and suggestions. I would also like to thank the members of the Cognitive Robotics Lab for the support at various times and for creating and maintaining a culture fostering growth and innovation.

I thank my family for the support and motivation through the journey of my higher education.

VITA

2019	Bachelor of Engineering (Honors), Birla Institute of Technology and Science, Pilani, India
2019-2021	System Software Engineer, Nvidia, India
2022-2023	Research Assistant, University of California San Diego
2023	Master of Science, University of California San Diego

PUBLICATIONS

Madhavan, Srirangan, Anwesan Pal, and Henrik I. Christensen. ”Role of reward shaping in object-goal navigation.” arXiv preprint arXiv:2207.08021 (2022). Embodied AI Workshop, CVPR 2022

ABSTRACT OF THE THESIS

Navigation in Everyday Settings Using Semantics and Language

by

Srirangan Madhavan

Master of Science in Computer Science

University of California San Diego, 2023

Professor Henrik Christensen, Chair

Advancements in deep learning have catalyzed growth in robotic applications, extending their utility beyond constrained settings. Nevertheless, a significant challenge remains in enabling robots to efficiently navigate and interact within unstructured and dynamic environments. Existing methods in robot navigation require the use of dense geometric representations such as high definition maps or full 3D reconstruction. But these methods are non trivial and consume significant resources in both creation and usage. They also become stale for environments with constant changes. To be able to scale in terms of size and novelty of the environment, new algorithms that use representations accounting for semantics is required. Besides that, to be able to interact and collaborate with humans, these representations need to be able to ground the visual information in other modalities such as text, while retaining a long term memory. This thesis presents work on these directions, development of new approach and the discussion on the experiments applying these methods to navigation and robot instruction following in home environments.

Chapter 1

Introduction

People have long sought to mechanize tasks that are tedious, dangerous or outright humanly impossible. Natural progression of that search has been the developments in robotics over the past five decades. Manufacturing and logistics have been one of the earliest and widest adapters of robots. Unmanned vehicles both aerial and terrestrial are being developed for last mile goods delivery. Besides these, due to COVID-19, [11] there has been faster adaptation of robots in offices and shops for sanitation work. One of the areas in which robot adaptation has been comparatively slow is inside our homes. The most common case of usage at homes has been vacuum and floor cleaners. There has also been some adaption for assisting people with reduced mobility and for educational support of children with different abilities. But most of these applications are limited and rigid in the set of activities they can perform and are therefore not able to scale to wider adaption. For true adaption in homes in helping elderly and children or collaborative home keeping, there is a need for big strides in the ability of robots to be robust, have increased awareness of the surroundings, act with common sense and be able to provide a wider set of services from a single platform.

There have been tremendous advancements in statistical learning, natural language processing, computer vision and other perception and control systems following the development of deep learning based approaches. These have had direct impact on the growth of robotic applications, leading to adaption outside of constrained environments. Yet, one of the most critical challenges hindering even wider adaption is the ability of the robots to navigate and interact in highly unstructured and dynamic environments. The task of navigating in highly dynamic scenarios requires understanding of the spatial details and the ability to take contextual decisions to reach the destination safely and efficiently. This means, being able to understand the causality between actions and changes in state and understanding semantics to an extent which allows the agent to take actions that help discover new knowledge on the fly. To that extend, one of the most easily available mode of information acquisition for the agent is through visual inputs, such as images and videos. While traditional vision systems have been able to infer details about the physical boundaries of the surroundings in which the agent operates, and even semantics to some extent, these require significant computational and manual resources and do not generalize to new scenarios.

The ability to interact with humans, adds significant complexities since it not only needs to be able to navigate from a pre-programmed source to destination, it must be able to do so by inferring these details from the data that is provided by human beings. Naturally, a very easy and well established modality for interaction between humans is natural language. The recent advancements in the field of natural language processing and language models have placed this goal within reach. Some of the works in this direction can be categorized into Embodied Instruction Following (EIF).

This thesis focuses on the above mentioned two directions of robotics in indoor scenarios: visual navigation and embodied instruction following. It aims to present a combination of theoretical analysis, algorithm development and experimental evaluations done in these two domains. In Chapter 2, discussion on some recent works in visual navigation, theoretical concepts utilized in the baseline works and the novel addition to the learning architecture have been discussed. Similarly, Chapter 3 discusses the same in the context of embodied instruction following.

Chapter 2

Object Oriented Navigation in Indoor Scenes

An integral component of any autonomous agent capable of performing tasks in the physical world is the capability to appropriately navigate to the desired points of interaction. Historical approaches to solve this problem involved planning based approaches and constructing metric semantic maps. With the evolution of deep learning, latest approaches have leveraged these techniques to develop algorithms that produce significantly generalizable results by addressing the gaps in traditional methods. In this chapter, discussion has been conducted on the latest approaches for navigation in indoor settings, tools and environments used for developing and testing these approaches and our contribution to learning based navigation.

2.1 Background

Navigation and specifically visual navigation has a storied history. In this work, the core navigation policy is learned as a deep neural network using the Actor-Critic model, under the Reinforcement Learning Algorithms. In this section, we cover the overview of broad categorization of navigation algorithms, some relevant work in these categories and background of the theoretical concept behind the Actor-Critic model.

2.1.1 Classification based on mapping

One way of broadly classifying navigation methods is map based vs map-less navigation approaches. Much of the the traditional approaches can be categorized as map based approaches [49][5][20][57]. In these, the navigation problem is formulated as obstacle avoidance problems, where the agents aided by the maps of the operating environment identify a collision-free trajectory using algorithms such as A*. But these methods are limited by the requirement for pre-computed maps of the environment, which may not be computationally feasible or even possible. Other approaches in this category create a map on-the-fly, which still suffer from some of the same drawbacks. These are further complicated by the need for including the semantics in the maps for more meaningful task execution. One potential solution to this problem is the class of approaches that are map-less [10][15][45]. In these methods, the agents learn representations over time through interaction with the environment. These representations act as some form of implicit maps that help the agent navigate.

2.1.2 Classifications based on goal type

Another classification criterion is what kind of goal the navigation algorithms require. This categorization is specifically applicable in the indoor settings. In the case where the agent is given a 2D/3D coordinate as the goal to reach, it can be categorized as pointgoal navigation[1][22]. If the agent's goal is to reach an object category while navigating around obstacles, it can be categorized as objectgoal navigation[63][61][58]. Pointgoal navigation is still not very well defined in realistic in indoor robotics tasks. Work discussed here, tackles objectgoal navigation. These algorithms usually require some prior knowledge about the environment, which aids in navigation.

2.1.3 Asynchronous Advantage Actor-Critic (A3C) Model

Deep reinforcement learning (DRL) has proved to be very successful in various tasks that require human level expertise ranging from object recognition [3], solving control problems

[18][26], to playing Atari games [36] and defeating champion Go-players [48]. Generally these algorithms can be designed based on the action probability learning or action value estimation. In policy based methods, the policy is parameterized as $\pi(a_t|s_t;\theta)$ where θ are the weights of the neural network, which get updated using gradient ascent on the estimated returns $E[R_t]$. The estimate of the value $V(s_t;\theta_)v$ is also modelled as neural network with weights θ_v Actor-Critic algorithms combine the two approaches, where a of policy gradient model acts as the actor doing action selection and the value estimation model acts as a critic evaluating the quality of the actions. In case of A3C[35], there are two additional improvements. In a typical implementation of policy gradient algorithms, the expected return $E[R_t|s_t = s, a]$ is the value of the agent's actions, where $R_t = \sum_{k=0}^{\infty} \gamma^k t_{t+k}$ is total accumulated return at time t with discount factor γ.

In A3C instead, to reduce the variance in the policy gradient (gradient of the expected return), a learned function of the state $b_t(s_t) \approx$, called baseline is subtracted from the return at time t. This difference can be seen as a measure of the advantage $A(a_t, s_t)$ in taking action a_t in state s_t. This can be written as follows.

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$$

This equation gives us the mathematical perspective of how the actor critic architecture works, since the policy π is the actor and the baseline b_t is the critic. The A3C algorithm operates in the forward view and uses a mix of n-step returns to update both the policy and the value function. They are both updated after every t_max actions or when the termination stage is achieved. The update performed can be written as:

$$\nabla_{\theta'} \, log\pi(a_t|s_t;\theta') \, A(s_t, a_t; \theta, \theta_v)$$

where,

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} \, + \, \gamma^k V(s_{t+k}; \theta_v) \, - \, V(s_t; \theta_v)$$

Here, k can vary from state to state and is upper-bounded by t_{max}. Further information can be found in [35].

2.2 Learning Stronger Semantic Contexts for Navigation through Reward Shaping

Semantic understanding and contextual learning have been studied in depth in the computer vision and robotics communities. But traditionally these have been approached within static settings such as semantic segmentation, object detection, recognizing activity, etc. But interactive navigation tasks require the visual input to be contextualized in terms of the current state and also the past states through which the agent has travelled. In case of indoor navigation, this requires the agent to learn the relationship between objects in the current and past frames with respect to the actions that resulted in them. Knowledge graphs and object-object relationships [34][28] have been utilized in computer vision tasks in the past but more recently, they are being used in the interactive learning and RL settings [61]. In the RL based interactive learning settings, reward is provided to the agent to act as a feedback for the success or failure of the set of action it takes. This reward signal is usually highly sparse and is provided at the end of a success full episode. This is partly due to the general non-trivial nature of crafting sub-goal or intermediate rewards. But in the case of objectgoal navigation which is the focus of this work, less-sparse intermediate goals can be crafted to aid the RL algorithm exploit the latent structure of the environment to learn the object - object relationships better. This can be taken a step further and the reward signal can be crafted as a dense, continuous function, resulting in improved success in reaching the goal. The first part of this section describes the basic architecture developed in [38] and the next part describes the continuous reward shaping mechanism.

2.3 Navigation Environment and Task Setup

Experiments using robots are typically done in physical world in a lab or a restricted test tracks. The inherently interactive nature of task like navigation and the data requirements of the deep learning based approaches make direct training of robots in physical world prohibitively tedious. There are also issues around generalization of the solutions and safety concerns besides the scalability. To work around these issues, there are open source data sets based on photorealistic 3D simulators such as AI2-THOR[23], Habitat[46], iGibson[27] etc. Depending on the environment, these can produce photorealistic RGB images, provide navigable and interactive actions as APIs that the agent can take, along with critical ground truth data points such as semantic and instance segmentation of the environment, 3D and 2D bounding boxes, depth information etc. This work is done on AI2-THOR [23].

The objectgoal navigation task setup for the iTHOR dataset of the AI2-THOR environment is as follows. The agent is tasked with navigating to a target object T. Here T is defines as a set $t_1, t_2, ..., t_N$. The agent does not have any access to the environment's semantic or topological map and the problem is defined as a purely vision based navigation. Within the boundaries of a given environment, the agent start at a random location at the beginning of an episode. The agent takes in as input the RGB image for the current time step t and the target object's word embedding rolled over the past time steps. As for the output, the agent then samples an action a from set A, where $a \in A = \{MoveAhead,\ RotateLeft,\ RotateRight,\ LookUp,\ LookDown\ and\ Done\}$. When the $MoveAhead$ action is sampled, the agent is moved forward by 0.25 meters in the simulated environment, while the two $Rotate$ actions rotate the agent view by 45 degrees. The camera is tilted up or down by 30 degrees when the two $Look$ actions are sampled. The end of an episode is marked by two cases. First, the episode ends in a "success" if at the final location of the agent, the target object is in the current frame, and is within a 1.5m distance in the environment. This criterion applies in the rest of this section whenever the object is called *visible*. For the episode to be called to an end in this manner, either the agent must sample a

”Done” action or there is an option in the simulator for the environment to sample this action once the target object is visible. The episode can also unsuccessfully end, if after a maximum number of actions, the object is never becomes visible to the agent.

2.4 Role of reward shaping in objectgoal navigation

In reinforcement learning based methods, reward shaping is a method used to provide additional local signals to motivate the agent to develop a behaviour that is consistent with prior knowledge of the domain [25]. In the task ob objectgoal navigation in indoor settings, frequent auxillary signals based on surrounding objects are highly useful in helping the agent reach the goal. This is even more important in case of large environments, where the number of steps required to reach the goal are quite high. A key piece of implementing reward shaping in this task setup is the hierarchical relationship between objects as noted in [38]. It can be observed that, in common indoor settings, a natural relationship between objects exists. As an example, a $toaster$ is in a $kitchen$ and is placed such that there are large salient objects such as *microwave* or $stove$ present close to it. Such salient objects can be called as *parent* objects for the *target* object. To facilitate the learning of these contextual relationships by the RL algorithms, the high dimensional embedding of each state must represent them. In the following section, the architecture introduced in [38] and named Memory-utilized Joint hierarchical Object Learning for Navigation in Indoor Rooms (MJOLNIR), is explained and in the subsequent section, the reward shaping mechanism of [33] is explained.

2.4.1 Learning Heirarchical Relationships for Object-goal navigation

In the task definition it was established that target object T is a set $t_1, t_2, ..., t_N$, when each t_i is an instance of target. In addition to these target objects, another set of objects called $parentsP = p_1, p_2.., p_M$ is introduced. This set is made up of the salient objects that are spatially and semantically related to the target object within a given room. This set is hand crafted and is used in the knowledge graph construction. The intuition behind this is that the agent learns to start the search for $t_i \in T$ by exploring areas around $p \in P$.

The construction of knowledge graph is similar to the work done in [61]. The object relationships are extracted by significantly pruning the image captions in Visual Genome(VG) dataset [24] and constructed as adjacency matrix. This work also introduced a $context vector \mathbf{c}_j$ for each object $o_j \in O$. Here O is the set of all objects present in the iTHOR dataset. The cardinality of O is 101. The context vector is a 5D vector that contains the information regarding the current state of each object in the current frame. c_j can be represented as $[b, x_c, y_c, bbox, CS]$. Here b specifies whether the object is present in the current frame and is binary. The bounding box center coordinates in the image and the size of the bounding box are the next three components. And the consine similarity between GloVe embeddings [40] of the object names are captured in the final component CS. This can be written as the following expression:

$$CS(g_{o_j}, g_t) = \frac{g_{o_j} \cdot g_t}{||g_{o_j}|| \cdot ||g_t||}$$

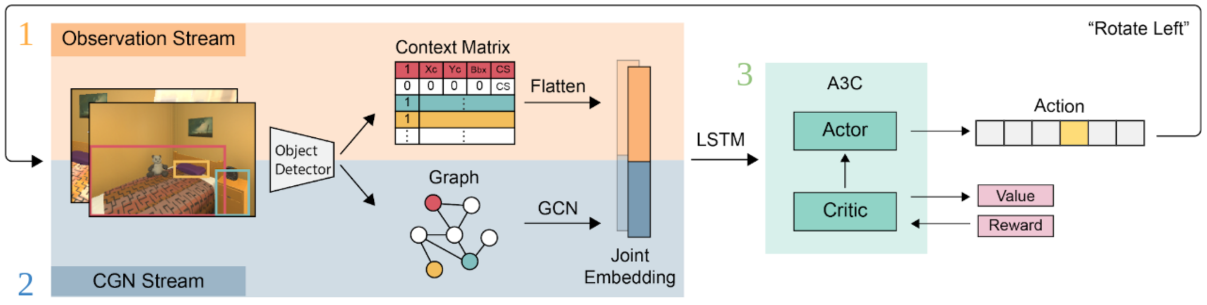


Figure 2.1. Architecture used to learn navigation policy as explained in [38].

The entire architecture contains two streams. The first stream titled Observation stream encodes the current observation and the second stream titled Contextualized Graph Network (CGN) is used for embedding the memory of the past frames using the knowledge graph $G = (V, E)$. There are further two variants of this architecture with modifications made to the observation stream. In first variant, the resnet-18 [17] feature of the current RGB frame is convolved with the word embedding and flattened to produce the vector used in observation stream. This is called as MJOLNIR-r. In second variant, the instead of the resnet features, the 5D context vector for every object is stacked to form a context matrix and is used for the observation stream. This version is named MJOLNIR-o. The CGN helps in bridging the gap between the information provided by knowledge graph and the domain difference between VG and the iTHOR datasets. A graph convolution [21] network learns the embeddings of the nodes. For a given graph G, each node vector $X \in R^{|O|+|g|}$ is concatenation of two vectors. The first part is a 101-dimensional binary vector representing the objects in the current frame similar to one hot encoding and the second vector is the glove embedding of that specific node object. Then, this graph passes through two layers of graph convolution network which generates intermediate embedding. The output of this is concatenated with the context vector and compressed through another layer of graph convolution network to create final graph embedding. These combined with the outputs of the observation through concatenation form the input embedding for the A3C RL model.

2.4.2 Reward Shaping

A new reward shaping mechanism for indoor navigation was introduced in Pal *et al.* [38]. In this mechanism when the agent navigates to a position where a parent object is visible, it receives a partial reward R_p. The partial reward is calculated as follows:

$$R_p = R_t * Pr(t|p) * k$$

where, R_t is the reward for the target object and $Pr(t|p)$ is the probability distribution that gives a measure of the relative closeness of the parent object to given target object. In this case, the R_t is fixed at 5 for all target values. k is a scaling factor that is also fixed at 0.1. These rewards are also provided only if the parent or the target object are within a distance of 1m from the agent in the environment and present in the frame. Because of this setup, the reward is independent of the distance between the effort the agent takes to reach a particular parent or target object. That is, the agent behavior that are aimed at reaching a parent or target object, but is unsuccessful due to external factors, goes unrewarded. This can be improved by reformulating the reward function to take in the distance between the object and the agent as a factor. Further, the new formulation for R_p can be extended to both the parent and the target objects.

Two methods are proposed for this extension in [33]. **(i) Utilizing metric depth:** The first approach is to use the metric depth information, in the form of depth maps ϕ obtained from the AI2-THOR simulator. An RGB-D sensor or any other depth estimation mechanism can be used in place of this depth map. In the depth map, the average value of the region bounded by the bounding box for given object is a reasonable approximation for measuring how far away the object is from the agent. Therefore, the scaling factor can be reformulated using the following logic:

$$R_p = R_t * Pr(t|p) * k'(d$$

$$k'(d) = k * (m * d + c)$$

$$d = \overline{\phi(i,j)} \; \forall \; i,j \; in \; bbox$$

The values for m and c are fixed in the experiments to -0.15 and 1 respectively. These values were chosen empirically so that, the partial reward scales linearly with the decreasing distance and $k' \in [0,1]$. **(i) Utilizing bounding box area:** The approach using the metric depth is intuitive and straight forward in theory. But in practice, it was observed that the computational load, specifically during training increased significantly due to the added sensor input. The dependence on additional sensor data for depth is also not desirable. A better alternative is to utilize a heuristic for relative distance base on the bounding box of a given object. The assumption here is that, as the agent moved closer to the object, it appears larger on the current frame and hence the area of the bounding box increases. For this approach, the scaling factor is given as follows:

$$R_p \ = \ R_t * Pr(t|p) * k'(d$$

$$k'(d) = k * (1 - (A_1/A_2(d))^{0.5})$$

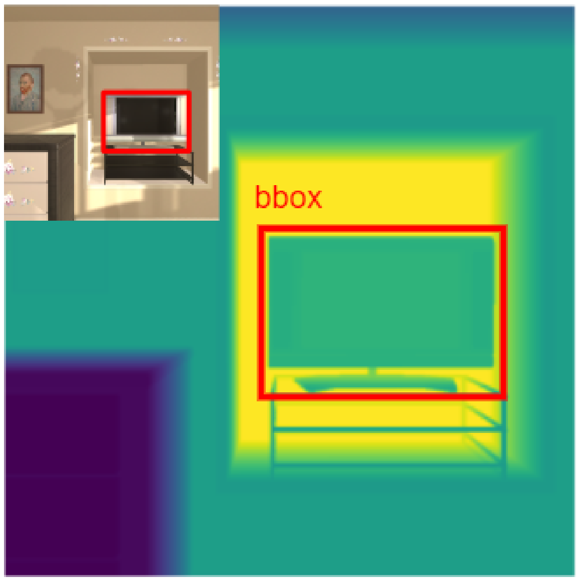
A_1 is bounding box areas of a particular object at the first frame it was observed in and A_2 is the area of the bounding box in the current frame. The implementation also keeps track of the observed/visible objects so as to provide a set of partial rewards for a given object only once. The two mechanisms are explained visually in Figure 2.2 and 2.3. The following section explains the experiments and the results.

2.4.3 Experiments and Results

The model was built on top of code made public by [58] using the PyTorch framework. The model was trained on the data from AI2THOR-v1.0.1 for 3 million episodes. The data was pre-collected and stored offline from the simulator to facilitate easier training. The evaluation is conducted for each of the 4 room types in the environment for 250 episodes each, totalling to a 1000 episodes. The floor plan, the target object and the initial stating state of the agent were all randomized for each episode. We evaluate the reward shaping mechanism by using GCN [61], SAVN [58], MJOLNIR-O and MJOLNIR-r [38] as baselines and using 4 different reward mechanisms: r_{bin} a binary reward mechanism, where upon successful navigation to target, the agent receives a reward, r_{base}, the partial reward mechanism from [38], and the two proposed dense partial rewards, r_{depth} and r_{area} using depth information and the bounding box alone respectively. The evaluation metrics are adapted from Anderson *et al*. [1].

Success rate (SR) - Table shows the performance for this metric. Almost all of the models, trained using the proposed reward mechanism end in better results. This is especially true for episodes in which agent needs to take larger path lengths, i.e $L \geq 5$, since further exploration of the environment might be needed for agent to discover the target.

Success weighted by Path Length (SPL) - In case of the SPL, as opposed to success rate, the



$$d = \overline{\phi[i, j]}, \quad \forall (i, j) \in bbox$$

Figure 2.2. Metric distance from depth map: Image shows depth map with a bounding box around the object. Inset contains the RGB image of the object. d is obtained by finding the average distance of each pixel in the bounding box.

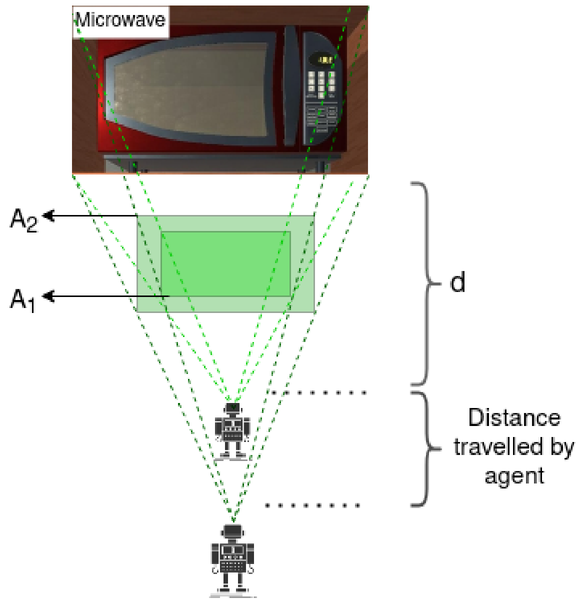


Figure 2.3. Relative distance from bbox area: Image on the right shows the relative increase in bounding box area of an object (A_1 to A_2) as the agent moves closer. d is object distance when area is A_2.

proposed method performs worse than the baseline models. This is shown in Table 2.2. An intuitive way to explain this, as observed in the visualisations of test cases, would be that the agent explores the areas around parent objects better compared to earlier. It proceeds to target objects after more number of steps due to this. But, this need not necessarily be viewed as a setback, because it is important for the agent to explore the environment in search oriented tasks and specifically for larger environments.

Another caveat to be noted here is that, for the models that take into account the inter-object relationships, the denser distance-based reward functions perform better. For example, (like GCN [61], and the MJOLNIRs [38]). This firther solidifies the hypothesis that auxiliary signal based on surrounding objects aid in the searching far-off target objects.

Table 2.1. Metric 1: Success rate (%). The mean score over 5 runs is provided with the standard deviation as sub-scripts.

Models	$L \geq 1$				$L \geq 5$			
	r_{bin}	r_{base}	ours		r_{bin}	r_{base}	ours	
		[38]	r_{depth}	r_{bbox}		[38]	r_{depth}	r_{area}
GCN [61]	$33.1_{(0.8)}$	$33.3_{(1.4)}$	$31.7_{(0.7)}$	$\mathbf{35.3}_{(\mathbf{0.5})}$	$25.0_{(1.4)}$	$23.5_{(1.6)}$	$\mathbf{26.9}_{(\mathbf{1.1})}$	$24.6_{(0.8)}$
SAVN [58]	$34.7_{(0.5)}$	$\mathbf{40.7}_{(\mathbf{1.4})}$	$32.2_{(0.9)}$	$39.6_{(0.8)}$	$25.8_{(0.8)}$	$30.0_{(1.4)}$	$26.8_{1.3}$	$\mathbf{31.7}_{(\mathbf{1.5})}$
M_O [38]	$58.8_{(1.0)}$	$64.1_{(0.7)}$	$\mathbf{66.4}_{(\mathbf{0.3})}$	$66.3_{(1)}$	$40.6_{(0.6)}$	$46.6_{(1.6)}$	$50.5_{(0.7)}$	$\mathbf{51.5}_{(\mathbf{1.3})}$
M_R [38]	$65.5_{(0.6)}$	$68_{(0.9)}$	$\mathbf{77.1}_{(\mathbf{0.7})}$	$69.7_{(0.9)}$	$52.3_{(0.8)}$	$52.3_{(0.5)}$	$\mathbf{69.2}_{(\mathbf{0.8})}$	$57.3_{(1.3)}$

Table 2.2. Metric 2: SPL (%). The mean score over 5 runs is provided with the standard deviation as sub-scripts.

Models	$L \geq 1$				$L \geq 5$			
	r_{bin}	r_{base}	ours		r_{bin}	r_{base}	ours	
		[38]	r_{depth}	r_{bbox}		[38]	r_{depth}	r_{area}
GCN [61]	$10.0_{(0.4)}$	$\mathbf{10.8}_{(\mathbf{0.5})}$	$5.5_{(0.2)}$	$8.2_{(0.1)}$	$10.3_{(0.7)}$	$\mathbf{11.2}_{\mathbf{0.7}}$	$7.3_{(0.3)}$	$8.7_{(0.3)}$
SAVN [58]	$11.0_{(0.2)}$	$\mathbf{11.1}_{(\mathbf{0.3})}$	$6.6_{(0.3)}$	$10.5_{0.2}$	$11.7_{(0.1)}$	$12.4_{(0.5)}$	$10.5_{0.3}$	$\mathbf{12.8}_{(\mathbf{0.6})}$
M_O [38]	$18.5_{(0.3)}$	$\mathbf{20.7}_{(\mathbf{0.2})}$	$11.6_{(0.1)}$	$15.8_{0.4}$	$17.8_{(0.3)}$	$\mathbf{20.0}_{(\mathbf{0.6})}$	$13.7_{(0.3)}$	$17.3_{(0.5)}$
M_R [38]	$24.4_{(0.3)}$	$\mathbf{26.5}_{\mathbf{0.2}}$	$15.0_{(0.3)}$	$16.8_{(0.2)}$	$26.2_{(0.4)}$	$\mathbf{27.2}_{(\mathbf{0.3})}$	$20.3_{(0.4)}$	$19.3_{(0.4)}$

Chapter 3

Natural Language Guided Navigation and Interaction

One of the ultimate goals of the robotics is a collaborative environment where a humans can communicate using means that come naturally to us and in turn the agents take actions to satisfy the requirements. With the foremost means of communication between human beings being natural language, adding the capability to follow instructions given in this modality would be highly beneficial in advancing the adoption of robots. Consequently, this is a very active area of research. With recent advances in natural language processing and the usage of transformer models, the current research in this area focuses on leveraging those advances to solve vision-language navigation (VLN) and embodied instruction following (EIF) problems. In this chapter, within the first section, background details on latest advancements and core ideas utilized in this work are explained. In the sections following that, the problem setup, the architecture used to solve the problem and the observations are explained.

3.1 Background

Embodied instruction following is a composition of multiple concepts including language grounding, language based navigation, and the object interaction steps. Therefore complexity is added due to the necessity to manage the compositionality of the sub-tasks and actions, besides being able to understand the language. Research on systems to follow human instruction is not recent and has diverse background [9][6][4][52]. Variety of formats for language instructions have been utilized ranging from structured commands [41] to natural language or a mix of language and target state visual information[31]. Since the agent needs to be able to handle raw inputs from sensors, the research expanded in to the direction of vision and language navigation (VLN), where the agent is given unstructured input and needs to learn the contextualization process between the language and visual inputs.

These tasks are also long in terms of the time and the number of individual actions required to complete the instruction are quite high. This requires the agent to have memory of past observations and corresponding actions and learn the causality between them. Recurrent architectures have been investigated [32][51][55] since they have the desired property of maintaining track of past states in its internal state. But these also have the drawback of not being able to capture long term dependencies [54]. With the introduction of Vaswani *at al.* [53], these deficiencies could be addressed and that opened up the avenue for rapid advancements in both language understanding [8][12] and multimodal learning [50]. Paschevich *et al.* [39] successfully adapted this work recently for the case of vision and language instruction and act as the main inspiration for the work presented in following sections.

3.1.1 Transformer architecture for multimodal learning

Lately, there has been substantial research conducted on a wide array of Transformers, delving into various multimodal tasks. These studies have showcased the adaptability of Transformers across diverse modalities, both in tasks of discrimination and creation. In this section, we'll take a brief glace at techniques and designs employed in existing multimodal transformer, which is cruicial to the architecture presented in this report. We will look at two relevant aspects of the multi modal transformers. One is the way they handle multimodal inputs and the second aspect is how attention mechanism is implemented in the multimodal context.

The Transformer framework represents a versatile architectural paradigm that can be conceptualized as a form of general graph neural network. Particularly, the self-attention mechanism facilitates the treatment of each input as a fully interconnected graph. This process enables the identification of broader global patterns, thereby lending Transformers the capability to function within a modality-neutral framework. This adaptability is achieved by considering the embedding of each token as a distinct node within the graph.

When confronted with input from any given modality, users can take two primary tasks: (1) breaking down the input into tokens, and (2) choosing a suitable embedding space for token representation. These preparatory steps pave the way for the subsequent engagement with Transformers. In practice, while both tokenization and embedding selection are pivotal for Transformers, they offer considerable flexibility, accommodating various alternatives. For instance, consider the scenario of an image. Determining how to tokenize and embed the image is not a fixed process. Users can opt for or create tokenization strategies that operate across different levels of granularity—ranging from coarse-grained to fine-grained. For example, choices include employing regions of interest (ROIs) acquired through object detection, along with CNN features for tokens and their corresponding embeddings [30]. Alternatively, one could select patches along with linear projection for tokens and token embeddings [13]. Even the employment of graph nodes from object detection and graph generation, paired with GNN features, is a possibility for tokens and their embeddings [62]. With a tokenization plan in place, a wide array of subsequent embedding techniques can be embraced.

In multimodal Transformers, interactions between different modes (like fusion and alignment) are primarily managed through self-attention and its variations. Hence, this section undertakes a comprehensive review of principal practices in multimodal modeling within Transformers. Self-attention designs have multiple variations based on how the embeddings are combined. Based on the classifications explained in [60], there are 6 varieties. (1) early summation (weighted or token-wise), (2) early concatenation, (3) hierarchical attention (multi-stream to one-stream), (4) hierarchical attention (one-stream to multi-stream), (5) cross-attention, and (6) cross-attention leading to concatenation. In this work, keeping relevance in sight, we only look at summation and concatenation as they have been adapted to the proposed architecture.

It's worth noting that all discussed self-attention and its variations are remarkably adaptable, extending readily to scenarios with multiple modalities.

Given inputs $\mathbf{X}_A$ and $\mathbf{X}_B$ from two modalities, $\mathbf{Z}_{(A)}$ and $\mathbf{Z}_{(B)}$ respectively represent their token embeddings. $\mathbf{Z}$ signifies the token embedding (sequence) resulting from multimodal interactions. $Tf()$ denotes Transformer model. In practical terms, early summation represents a straightforward and effective multimodal interaction. Token embeddings from various modalities can be combined using weighted-sum at each token position and subsequently processed by Transformer layers. As summarized in [60]:

$$Z \leftarrow Tf(\alpha\mathbf{Z}_{(A)} \oplus \beta\mathbf{Z}_{(B)}) = MultiHeadSelfAttention(\mathbf{Q}_{(AB)}, \mathbf{K}_{(AB)}, \mathbf{V}_{(AB)}),$$

where $\oplus$ denotes element-wise sum, and α and β are weights. Expanding further,

$$\mathbf{Q}_{(AB)} = (\alpha\mathbf{Z}_{(A)} \oplus \beta\mathbf{Z}_{(B)})\mathbf{W}^{Q}_{(AB)},$$

$$\mathbf{K}_{(AB)} = (\alpha\mathbf{Z}_{(A)} \oplus \beta\mathbf{Z}_{(B)})\mathbf{W}^{K}_{(AB)},$$

and

$$\mathbf{V}_{(AB)} = (\alpha\mathbf{Z}_{(A)} \oplus \beta\mathbf{Z}_{(B)})\mathbf{W}^{V}_{(AB)}.$$

Its key advantage lies in its computational efficiency, yet its main drawback arises from manually determined weightings. The summation of position embeddings inherently represents a form of early summation.

The other straightforward approach is early concatenation, where token embedding sequences from various modalities are concatenated and fed into Transformer layers as:

$$Z \leftarrow Tf(C(\mathbf{Z}_{(A)}, \mathbf{Z}_{(B)})).$$

Hence, all positions of multimodal tokens are collectively attended to as a unified sequence, enabling thorough encoding of each modality's positions through the context of other modalities.

3.1.2 Scene Representation for Navigating Agent

As mentioned in the previous section, it is vital to adapt a good strategy to tokenize the input modalities to the the transformer model. In case where the two modalities are language and images, it is easier for the language part to be tokenized based on the words or character sequences. For the image part, creating good representations that capture the spatial and semantic information is goes a long way in improving the performance of the autonomous agent. These representations can be anywhere from CNN embeddings, to region proposal patches, to graphs constructed from the object detection. In this work, we use a graph based tokenization strategy. But the methods used for traditional scene graph constructions [19][29][59] do so as discrete connections and are static in time. But for an agent that moves through a house, we require a more robust scene graph representation that a) do not change when viewed from a different agent position and b) has changes in part of the graph that correspond to a displaced object. To facilitate that we adapt the pre-trained model from Gadre *et al*. [14].

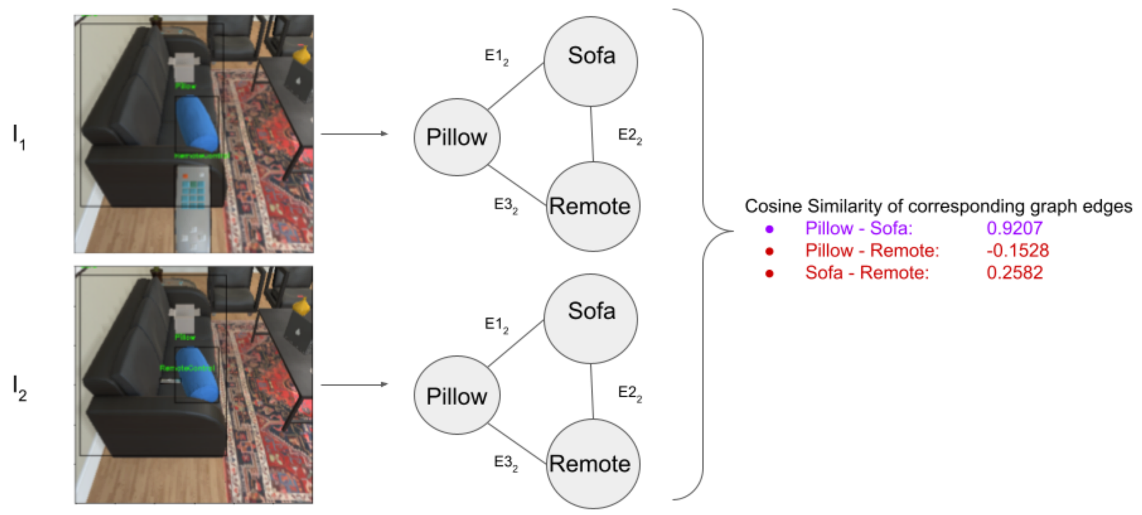


Figure 3.1. Comparison of graph features of same scene under rearrangement of single object.

The primary objective is to encapsulate object relationships through continuous vectors, updating this representation dynamically as the agent navigates. For seamless execution of interactive tasks without further training, we propose a straightforward planning strategy linked to this representation. Constructing such an interactive scene representation presents numerous challenges. The graph structure must accommodate new objects, inferring their relationships with existing ones. Furthermore, the algorithm must determine object correspondence across diverse perspectives. Addressing these complexities, our approach involves acquiring object relational embeddings using a contrastive loss. These embeddings serve to portray scene representation nodes and edges. Notably, this approach doesn't confine modeling to predefined symbols. The agent maintains a memory of previous embeddings, leveraging it to contrast newly extracted embeddings from egocentric observations, thus distinguishing new from existing embeddings.

In conventional scene representations, nodes are typically denoted by class labels, and edges are characterized by attributes linking these classes. In contrast, the CSR[14] approach employs continuous attributes for both nodes and edges. To derive nodes, a detector processes the current image (I_t) to obtain n detections. Employing the CSR encoder f_{CSR}, all n2 region pairs are mapped to a latent space, forming the local scene representation G^t_{CSR} with n node features and $n^2 - n$ edge features. The main concern is defining an effective objective for training the CSR encoder f_{CSR}. Here, the desired result is to ensure that the same relationship across different viewpoints converges to the same embedding location in the space. This stability assumption aligns with stable object trajectories. For instance, a kettle's relationship with a stove remains consistent across various angles, regardless of background objects. To create features capable of capturing relations from diverse viewpoints, CSR draws inspiration from momentum contrast strategies [16]. Using two views of an object pair, CSR extracts representations using f_{CSR} and a momentum encoder. This encoder, with architecture and initialization akin to f_{CSR}, updates via a slow-moving average of f_{CSR} weights. The InfoNCE loss [37] enforces relation similarity in feature space, expressed as

$$L_{CSR} = -log \frac{exp(CosSim(q,k^{+})/\tau)}{\sum_{K}^{i=0} exp(CosSim(q,k^{i})/\tau)}$$

Here, q denotes the f_{CSR} output for a node or edge, k^{+} signifies the momentum encoder's second view representation of the same relationship, each k^{i} represents a negative sample from K other momentum-encoded relationships, τ is a softmax temperature scaling parameter, and CosSim is the dot product of L2 normalized features. This objective encourages features to (1) exhibit high CosSim for the same relationship, (2) maintain multi-view consistency, and (3) differ from other relationships. In 3.1, a comparison between two sets of graph features are displayed where a single object has transition in space.

3.1.3 Dynamic Graph Attention

As the main architecture is based on a multimodal transformer, the generated scene graphs need to be further compressed into an embedding space where each scene graph corresponding to each frames can be viewed as tokens for the input to transformer model. These embeddings must have enough expressivity to compress the graphs, while also being able to distill the essence of the inter object relationships captured by the CSR. The version of Graph attention (GAT) layers, GATv2 developed in Brody *et al*. [7] are excellent at satisfying these requirements. GATv2 offers a dynamic mechanism for processing graph-structured data. By assigning attention scores to neighboring nodes, this layer selectively emphasizes pertinent information during aggregation, effectively focusing on the most relevant connections. Such adaptability enables more effective feature representation and information propagation, enhancing the overall performance of graph-based models in various tasks. In the standard GAT model, the linear layers are consecutively applied, without considering the sequence of attended nodes in relation to the query node. In contrast, GATv2 takes a distinct approach, enabling each node to potentially attend to any other node, breaking free from the constraint of fixed node rankings. The formulation can be written as follows. For each node vector:

$$x'_i = \alpha_{i,i}\Theta x_i + \sum_{j \in N(i)} \alpha_{i,j}\Theta x_j$$

the attention coefficients $\alpha_{i,j}$ are computed as:

$$\alpha_{i,j} = \frac{exp(\mathbf{a}^T LeakyReLU(\Theta[x_i||x_j]))}{\sum_{k \in N(i) U i} exp(\mathbf{a}^T LeakyReLU(\Theta[x_i||x_k]))}$$

In cases where the edge features are multi-dimensional, which is our case, the attention coefficients are calculated as follows by taking the edge vector $e_{i,j}$ into account.

$$\alpha_{i,j} = \frac{exp(\mathbf{a}^T LeakyReLU(\Theta[x_i||x_j||e_{i,j}]))}{\sum_{k \in N(i) U i} exp(\mathbf{a}^T LeakyReLU(\Theta[x_i||x_k||e_{i,k}]))}$$

3.2 Interaction Environment and Task Setup

As mentioned in previous chapter, since the task is interactive and involves robot navigation and rearrangement of objects in the environment, it is restrictive to train a deep learning based model in physical world. Doing so, raises concerns in terms of logistics, safety, solution quality, etc. Therefore, the need for workarounds, in the form of simulated environments arises. In the case of navigation the basic datasets provided by 3D simulators such as AI2-THOR[23], Habitat[46], iGibson[27] etc. were sufficient. But in the case of embodied instruction following, the agent needs to understand the context of the language in terms of the task, as illustrated in 3.2. This prompted the development of datasets such as TACoS[43], R2R[2], Matterport EQA[56], VirtualHome[42], etc. In this work, the ALFRED dataset [47] structured on the AI2THOR environment has been utilized. This is due to some of the advantages it provided such as, two-level instructions for trajectories, a semi-continuous grid based movement compared to either a lack of option or a navigation graph based movements in other datasets. ALFRED also requires the agent to locate the object that it wishes to interact with spatially using a mask. This adds another level of complexity which makes the problem non-trivial and more representative of real world scenarios.

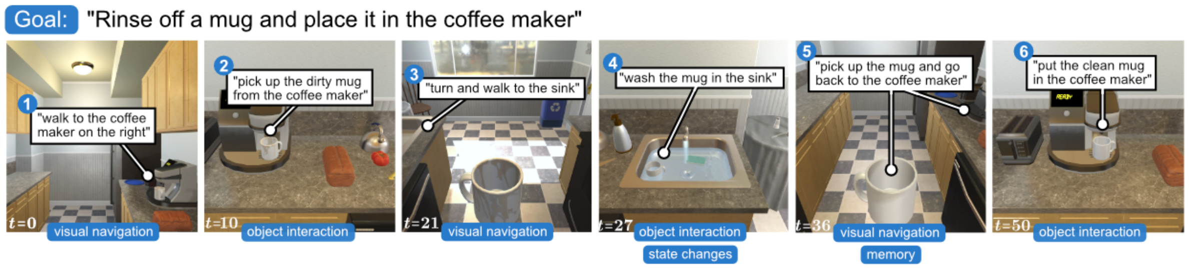

Goal: "Rinse off a mug and place it in the coffee maker"
1 "walk to the coffee maker on the right"
2 "pick up the dirty mug from the coffee maker"
3 "turn and walk to the sink"
4 "wash the mug in the sink"
5 "pick up the mug and go back to the coffee maker"
6 "put the clean mug in the coffee maker"
t=0
visual navigation
t=10
object interaction
t=21
visual navigation
t=27
object interaction
state changes
t=36
visual navigation
memory
t=50
object interaction

Figure 3.2. Metric distance from depth maps

The tasks are divided into 7 categories as follows: 1) Pick & Place 2) Stack & Place 3) Pick two & place 4) Clean & Place 5) Heat & Place 6) Cool & Place 7) Examine in light. Many of these tasks require the agent to identify the one or more objects that it needs to move or interact with and the receptacles that these objects go into or need to be picked from. For example a $fruit$ from $refrigerator$. Simultaneously the agent also needs to identify the intermediate state change in tasks that require heating, cooling etc. ALFRED dataset provides the instruction during test scenarios and also provides ground truth supervision trajectories to support supervised training approaches. This includes the state of the room at the beginning of simulation, language instructions corresponding to each episodes, planner goals, and expert demonstration of the task, in the form of step by step actions that were annotated by expert. This works mainly utilizes the instructions and the expert demonstrations. The instructions are provided in two levels. A high-level goal instruction I_g for each trajectory which briefly encapsulates the objective of the task. For example, in 3.2, I_g is *"Rinse off a mug and place it in the coffee maker"*. The second stage of instructions is made up of set of step by step instructions I_l. Following the same example, these instructions would be shown in the 3.2 as *"walk to the coffee maker on the right", "pick up the dirty mug from the coffee maker"*, etc.

In a typical episode, the agent at time $t = 0$ receives the set of instructions I_g and I_l. The instruction can be viewed as a constitution of words $x_{1:L}$. Besides that, at every time step t, the agent's only other input modality is the RGB images viewed from current position, v_t. The agent can, at every step choose one out of 13 actions which can be either from the *navigation* category or the *manipulation* category. There are 5 *navigation* actions (*Turn Left/Right, Move Forward, Look up/down*) and 7 manipulation actions (*Pickup, Put, Open, Close, Toggle On/Off and Slice*). As mentioned earlier, to manipulate a specific object, the agent needs to navigate such that the object is visible in the current frame and is within the manipulatable distance, and provide a manipulation mask M_t of the target object. The expert demonstrations also contain the actions taken at each time step $a_t \in 1, ..., A$, where A is the cardinality of the action set in ALFRED dataset.

The goal of the task is to learn a policy that can approximately mimic the expert policy.

3.3 Learning language grounded scene graphs for instruction following

The model used to learn the policy is a multimodal transformer as described in earlier section covering the background. This model has full visibility to the history. That is, it has access to all the visual frames $v_{1,...,t}$ and the actions $a_{1,...,t}$ the agent takes in the given episode. The model predicts the action to be taken for the current step as:

$$\hat{a}_t = f(x_{1,...,L}, v_{1,...,t}, \hat{a}_{1,...,t-1})$$

The proposed architecture is inspired from [39] and has 3 encoders for the language, visual and action inputs. The output of these encoders then act as the input to the multimodal encoder, where attention is paid between the different input modalities, therefore grounding the language in the visual inputs and past actions.

As specified earlier, the words in the instructions, the frames and the actions are transformed into an embedding space to act as the tokens for the transformer encoders. The encoder for the language stream shown in 3.3 comprises of a look up table to vectorize the text strings and a multi-layer transformer encoder. This takes in the words in the language instruction $x_{1:L}$

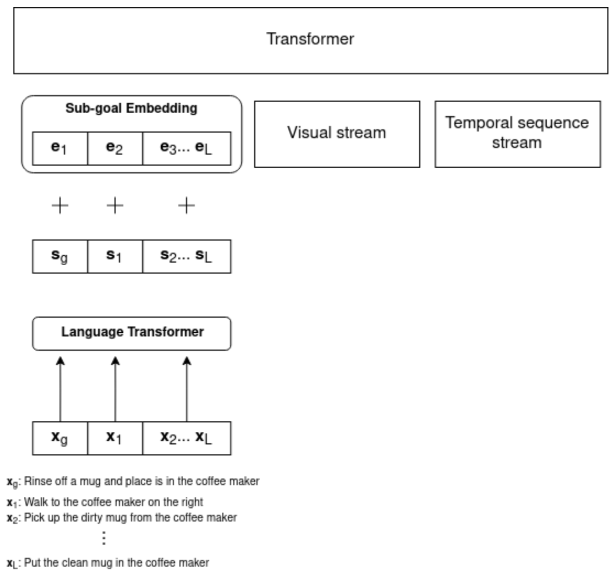

Transformer
Sub-goal Embedding
e_1 e_2 $e_3 \dots e_L$
Visual stream
Temporal sequence stream
+ + +
s_g s_1 $s_2 \dots s_L$
Language Transformer
x_g x_1 $x_2 \dots x_L$
x_g: Rinse off a mug and place is in the coffee maker
x_1: Walk to the coffee maker on the right
x_2: Pick up the dirty mug from the coffee maker
x_L: Put the clean mug in the coffee maker

Figure 3.3. A focused view of the language stream in the proposed architecture.

and output the language embeddings $h^x_{1:L}$. These tokens need information about their ordering in the sequence introduced. In the baseline version as illustrated in 3.3 sinusoidal encoding is added, which acts as the positional encoding for the *language stream* and temporal encoding for the *action* and *visual* streams. In the proposed version similar to the embedding proposed in [12], we add learned embedding to every token corresponding to words, to indicate which sentence in the low level instruction it belongs to. This is done as as to differentiate each part of the low level instruction into separate subgoals that the agent can learn to complete.

In the baseline model, in the visual stream, the images are transformed using the pre-trained ResNet-50 backbone [17] followed by convolution and fully connected layers, into the embedding space. Similar to the language stream, this can be notated as transforming image inputs $v_{1:t}$ into embeddings $h^v_{1:t}$. These primitive embeddings transform raw images into the vector embeddings and may not therefore capture the entire information available visually, including physical scene properties, context of the object relationships, etc. In order to better represent these, the proposed architecture additionally constructs a dynamic scene graph and learns representations from the graph to form the visual embeddings. This is done in two stages. First, the based on the detections masks from a Faster R-CNN object detector, [44] we use the

CSR network, f_{CSR} described in prior section to create the local CSR graph, $g_{1:t}$. Once we obtain the nodes, edges and edge features for the CSR graph, we use a graph attention module [7], again described in prior section to create the graph embedding $h^g_{1:t}$. The second step is to fuse the features obtained from the ResNet backbone with the graph embeddings. The hypothesis here is that while the ResNet features provide global information of the current frame, the graph feature provide the context missing from the former. To fuse these two features, we employ a simple concatenation followed by a fully connected layer to reduce the dimension.

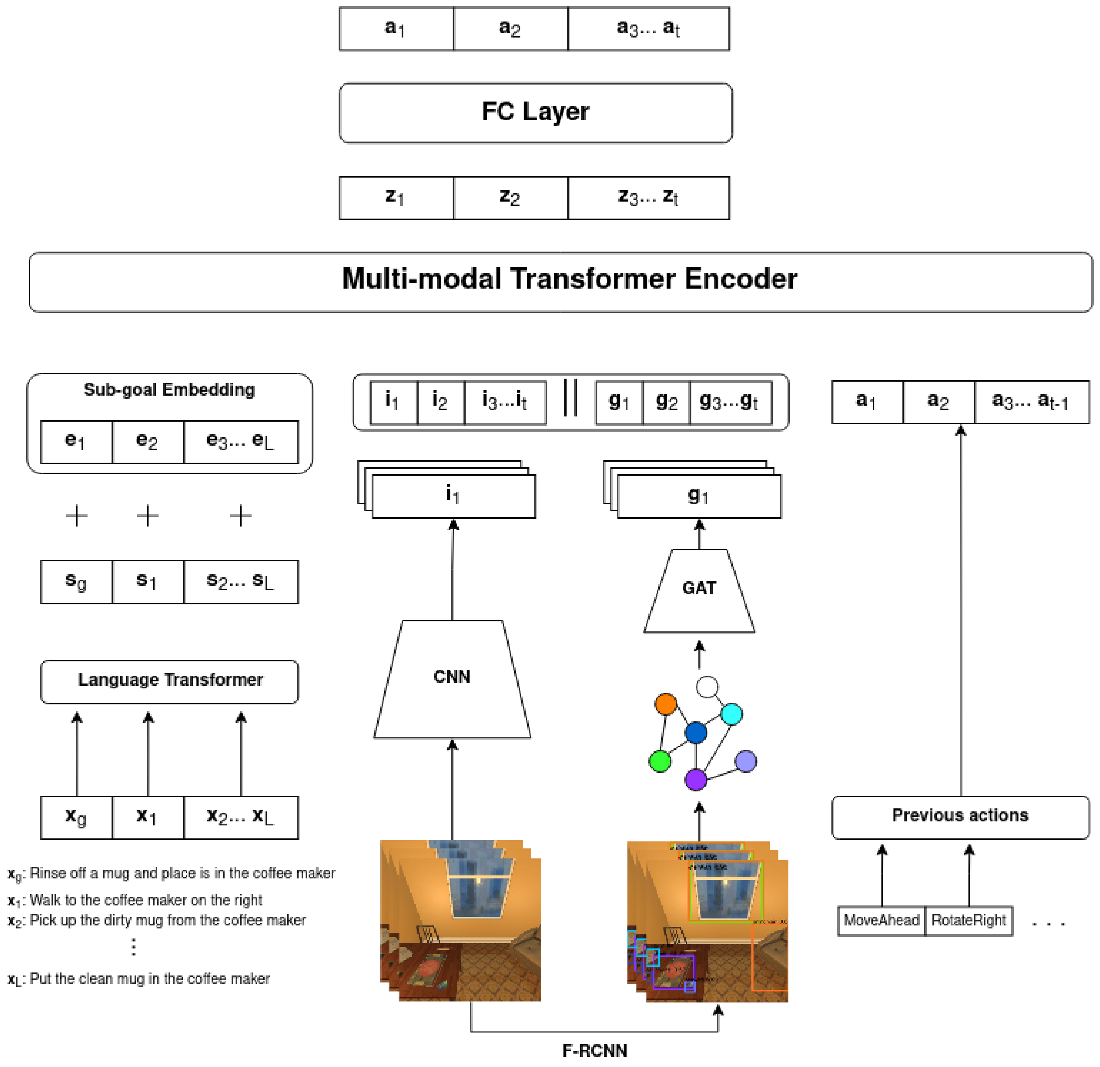

a_1 a_2 $a_3 \ldots a_t$
FC Layer
z_1 z_2 $z_3 \ldots z_t$
Multi-modal Transformer Encoder
Sub-goal Embedding
e_1 e_2 $e_3 \ldots e_L$
+ + +
s_g s_1 $s_2 \ldots s_L$
Language Transformer
x_g x_1 $x_2 \ldots x_L$
x_g: Rinse off a mug and place is in the coffee maker
x_1: Walk to the coffee maker on the right
x_2: Pick up the dirty mug from the coffee maker
⋮
x_L: Put the clean mug in the coffee maker
i_1 i_2 $i_3 \ldots i_t$ || g_1 g_2 $g_3 \ldots g_t$
i_1
CNN
g_1
GAT
F-RCNN
a_1 a_2 $a_3 \ldots a_{t-1}$
Previous actions
MoveAhead
RotateRight
. . .

Figure 3.4. Complete proposed architecture showing the 3 different input streams, their encoders and the multimodal transformer encoder used to learn the agent policy.

The actions are encoded using simple look up tables to vectorize the raw actions $a_{1:t-1}$ into their corresponding embeddings $h^a_{1:t-1}$. Finally, the embeddings from the modality specific encoders are concatenated and passed as the input to multimodal encounter that attend between different modalities using a causal attention mask [53]. This produces the output embeddings $(z^x_{1:L}, z^v_{1:t}, z^a_{1:t})$. Since the output embeddings of the visual features attend to the other modalities, $z^v_{1:t}$ is used as the input for a fully connected layer that predicts the action sequence till time t, $\hat{a}_{1:t}$ Since the transformer architecture is sequential, the output prediction is the sequence of actions from the beginning of the episode till current time step t. During training, these actions are compared with the actions from ground truth trajectory to compute a cross entropy loss, which guides the model update.

3.4 Experiment details and Results

Alfread Dataset: The dataset consists of expert deomstrations using 58 unique object classes and 26 receptacle object classes distributed over 120 different indoor scenes. In total there are 119 object categories in the environments. The scenes are divided into 30 each of the 4 available room types in AI2THOR simulator: kitchens, bathroom, bedroom and living room. There are 8055 expert trajectories and 25743 language instructions associated with them. This is divided into splits of 21023 training samples, 1641 validation and 3062 testing samples. Both the validation and the test splits are further divided into seen and unseen folds. Here the seen terminology denotes that the environment in which the agents acts has been observed among the samples in the training fold.

Architectural Details: The visual inputs are RGB images of size 224 X 224. The visual encoder and the mask generator are adapted from [39] and trained on 325K frames from train fold. The weights of the Faster R-CNN and the Mask R-CNN networks are frozen. The visual encoder features are reduced to a size of 512x7x7. The embedding for the language encoder, action encoder and the graph encoder are of length 768. The main transformer encoders have 2 blocks, 12 self attention heads and a hidden size of 768. The network is trained with an Adam optimizer for 20 epochs. The batch size for every epoch is 8 trajectories each, which means, 3750 batches per epoch.

The evaluation metrics are adapted from [47] and are aimed at measuring both the success of the overall task and the sub goals on a more fine grained level. When the agent creates the state changes and object positions corresponding to the expected goal conditions for the task, the end is considered to be in success and failed otherwise. This is marked by binary count of 1 or 0 respectively. In addition to the success of the task, the goal-condition success is also measured. GC success is the ratio of goal conditions completed at the end of the episode to those necessary to have the task completed. The tasks success can be 1 only if the GC success is 1. [47] also specifies the path length weighted version of both the metrics. Using these metrics, we compare the baseline model without any segment embedding or changes to the visual stream, to the model after incorporating the proposed changes. The results are presented in **??**. It can be inferred from the table that the segment embedding alone results in 14.9% increase in success rate, 12% increase in GC success and 14.2 and 10% increase in their path weighted counterparts. The introduction of the graph stream results in a slight decrease in the success metrics. One possible explanation for this is that while the additional modality is expected to increase the contextualized learning, more training data and sophisticated loss signals might be required to actually realize the gains.

Table 3.1. Metrics (%). .

	S.R.	G.C.	Path Length Weighted S.R.	P.L.W G.C.
Baseline	31.4	39.2	23.2	30.5
Segment Embedding	36.1	43.0	26.5	33.0
Graph Stream + Seg E	34.9	41.8	25.6	33.1

Chapter 4

Conclusion

In this report we discuss methods addressing the problem of building intelligent autonomous agents for indoor use cases. This is tackled by creating end-to-end learning models that map visual and text inputs to the actions space for different task setups. Chapter 2 presents the discussion on an Reinforcement Learning based approach to the task of object goal navigation in simulated household environment. Previous work on utilizing an end-to-end RL model introduced the logic that there exists spatial relationships between objects in houses. And that a salient object can be treated as a parent object to identify smaller objects that frequently occur together. Building on that, this work developed a continuous function for rewarding the RL model based on object relationships, and the distance of the target and its related objects from agent. This helps the agent make meaningful connections between the state, the objects in current state and actions. As a result, when searching for a target object, the agent focuses the search in areas that are logically most likely to contain them. Experiments with multiple RL based models show that the success rate of reaching the target object improves significantly when adapting the reward functions, accompanied by increase in the path length due to the added exploration. Chapter 3 presents work on end-to-end learning model for Embodied Instruction Following, the natural evolution of an agent capable of autonomous navigation. As earlier, this is structured in the form of a simulated indoor environment where agent takes navigation actions and manipulation actions as instructed by human language. This chapter provides the detailed explanation of the proposed multimodal transformer model for taking in human language and grounding them in visual inputs and agent actions. The proposed changes involve different embedding mechanism for the language and a new pipeline for processing visual inputs. To help the transformer model in better understanding the spatial organization of the objects in the scene, vectorized representations of scene graphs are extracted and a graph attention model attends to them before being processed by the multimodal transformer.

While the results are promising, due to the limited availability of time, some areas of further experimentation could not be addressed within this work. One of the issues with the end-to-end learning system is the limited availability of the signals guiding the optimization of network. Although the cross entropy loss for the predicted actions captures the core objective of the model, it may not be sufficient. Other supplementary signals like predicting the sub-goal stage prediction are not robust enough and the model overall will benefit from better loss functions such as generative losses, guiding the training. This also applies for the graph attention module in the visual stream. This stage compresses high dimensional tensors of the scene graphs into lower dimensional representation and takes place independent of the language instructions. Recent works have shown potential of contrastive losses for learning tasks such as visual question answering, which could be scene as a counterpart for the instruction following task. Another line of investigation that will be included in future work is incorporating the latest advances in the large language models for understanding the instructions. Pre-trained variants of LLMs such as BERT are being investigated currently to augment the language streams but this involves addressing practical concerns in bridging domain gaps and architectural differences. On the other hand, an entirely different and exciting line of investigation for the visual stream is utilizing recent advances in neural scene reconstructions and image-language contrastive representations such as CLIP to enable the agent to understand the environment in 3D.

In conclusion, the works presented here take some small steps in achieving the goal of intelligent autonomous agents collaborating with humans and identify gaps that need to be addressed in future work. While there are rapid advancements towards this goal everyday, potential for drastic improvement both in terms of model capabilities and efficiently executing them in constrained on-board compute devices do exist.

Bibliography

[1] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir Roshan Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[3] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[4] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, pages 481–495. Springer, 2013.

[5] Johann Borenstein, Yoram Koren, and Koray Kavukcuoglu. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 1991.

[6] Satchuthananthavale RK Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, 2009.

[7] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020.

[9] Guido Bugmann, Stanislao Lauria, Theocharis Kyriacou, Ewan Klein, Johan Bos, and Kenny Coventry. Using verbal instructions for route learning: Instruction analysis. *Proc. TIMR*, 1(96103):59, 2001.

[10] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages 2722–2730, 2015.

[11] Henrik Christensen, Nancy Amato, Holly Yanco, Maja Mataric, Howie Choset, Ann Drobnis, Ken Goldberg, Jessy Grizzle, Gregory Hager, John Hollerbach, Seth Hutchinson, Venkat Krovi, Daniel Lee, Bill Smart, Jeff Trinkle, and Gaurav Sukhatme. A roadmap for us robotics – from internet to robotics 2020 edition. *Foundations and Trends® in Robotics*, 8(4):307–424, 2021.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[14] Samir Yitzhak Gadre, Kiana Ehsani, Shuran Song, and Roozbeh Mottaghi. Continuous scene representations for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14849–14859, 2022.

[15] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2616–2625, 2017.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[19] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. Action genome: Actions as compositions of spatio-temporal scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10236–10247, 2020.

[20] Yiannis Kantaros and George J Pappas. Optimal temporal logic planning for multi-robot systems in uncertain semantic maps. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4127–4132. IEEE, 2019.

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[22] Noriyuki Kojima and Jia Deng. To learn or not to learn: Analyzing the role of learning for navigation in virtual environments. *arXiv preprint arXiv:1907.11770*, 2019.

[23] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, abs/1712.05474, 2017.

[24] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123:32–73, 2017.

[25] Adam Daniel Laud. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.

[26] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[27] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, C. Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

[28] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation recognition with graph neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4173–4182, 2017.

[29] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 852–869. Springer, 2016.

[30] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[31] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[32] Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*, 2019.

[33] Srirangan Madhavan, Anwesan Pal, and Henrik I. Christensen. Role of reward shaping in object-goal navigation. *CVPR Workshop on Embodied AI, New Orleans*, 2022.

[34] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016.

[35] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[37] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[38] Anwesan Pal, Yiding Qiu, and Henrik Christensen. Learning hierarchical relationships for object-goal navigation. In *Proceedings of the 2020 Conference on Robot Learning*, Proceedings of Machine Learning Research. PMLR, 16–18 Nov 2021.

[39] Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952, 2021.

[40] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[41] Amir Pnueli and Zohar Manna. The temporal logic of reactive and concurrent systems. *Springer*, 16:12, 1992.

[42] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[43] Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36, 2013.

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[45] Parvaneh Saeedi, Peter D Lawrence, and David G Lowe. Vision-based 3-d trajectory tracking for unknown environments. *IEEE transactions on robotics*, 22(1):119–136, 2006.

[46] 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)*, October 2019.

[47] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[48] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[49] Robert Sim and James J. Little. Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2082–2089, 2006.

[50] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7464–7473, 2019.

[51] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. *arXiv preprint arXiv:1904.04195*, 2019.

[52] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 ieee international conference on robotics and automation*, pages 1486–1491. IEEE, 2010.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. *Advances in neural information processing systems*, 28, 2015.

[55] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[56] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019.

[57] David Wooden. A guide to vision-based map building. *IEEE Robotics & Automation Magazine*, 13(2):94–98, 2006.

[58] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to Learn How to Learn: Self-Adaptive Visual Navigation Using Meta-Learning. In *IEEE CVPR*, June 2019.

[59] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5410–5419, 2017.

[60] Peng Xu, Xiatian Zhu, and David A Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[61] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[62] Xiaocui Yang, Shi Feng, Yifei Zhang, and Daling Wang. Multimodal sentiment detection based on multi-channel graph neural networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 328–339, 2021.

[63] 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. In *ICRA*, pages 3357–3364. IEEE, 2017.