COLMAN: Collaborative Multi-Agent Navigation using Textual-Visual Embeddings

Navneet Singh Arora
Department of Informatics
University of Hamburg
navneet.arora@informatik.uni-hamburg.de

Aida Usmanova
Department of Informatics
University of Hamburg
aida.usmanova@informatik.uni-hamburg.de

David Kraus
Department of Informatics
University of Hamburg
david.kraus@informatik.uni-hamburg.de

Sana Moin
Department of Informatics
University of Hamburg
sana.moin@informatik.uni-hamburg.de

Rana Abdullah
Department of Informatics
University of Hamburg
rana.abdullah@informatik.uni-hamburg.de

Advisor: Dr. Mikko Lauri

Master Project - Final Report
Computer Vision Group, CV
Department of Informatics, University of Hamburg

***Abstract*— In a multi-agent environment, collaboration is the key to performing joint tasks successfully. Despite the recent flourish in the research on multi-agent collaborative tasks, there is little emphasis on the construct towards the main task. Pre-task agent manoeuvres of (1) object identification, (2) path-finding and (3) visual navigation are always independently addressed. We hypothesise that including collaboration even in the build-up phase will significantly improve the pre-task performance with better inter-agent communication and, in turn, improve agent collaboration in the main task. This paper introduces *COLMAN: Collaborative Multi-Agent Navigation using Textual-Visual Embeddings*. It is a multi-agent collaboration in the build-up phase of the main task. We replace CNN with CLIP, fine-tuned on AI2-THOR scenes and see a increase of 5.92% accuracy by using the scene object embeddings. Taking a step further and paving the way for the entire AI community, we also consider the environmental impact of our entire model training process for benchmarking and initiating the process to move in the direction of sustainable AI research. Released model, pre-training, and fine-tuning code implemented in PyTorch are available on GitHub[1].**

***Keywords*—** Multi-agent, visual navigation, collaboration, AI2-THOR, CNN, CLIP, scene objects, embeddings

I. Introduction

Object goal navigation has been the research focus for the past decade in the machine learning paradigm. It is a task where the embodied AI agents learn to navigate towards an instance of an object in an unseen environment [40]. The CNN-based approaches [23, 39, 40] brought significant progress and achieved state-of-the-art (SOTA) performance on these tasks. However, these approaches follow the concept of environment mapping, inspired by the early work of Simultaneous Mapping and Localisation (SLAM) [4, 5] where a high degree of correlation builds between the objects and the map locations along with localising the robotic agents in the environment. This semantic mapping, along with the deep reinforcement learning (DRL) in photo-realistic 3D environments [13, 20, 29] poses many challenges. With the increase in environmental complexity and longer-horizon tasks, memory is the key bottleneck for the location mapping of the environment. It further induces the bias toward the kind of environments shown during the learning process [24, 30].

Recently, the emergence of Transformers [14] has shifted from the CNN-based, memory-intensive and semantic mapping-based approaches. The attention-inclusive, transformer-based approaches leverage the egocentric views to form a spatial relationship [36] of the setting. These transformers utilise the object information and simultaneously have scene understanding with multi-head attention [32, 42].

While the progress in this domain is significant, a multitude of these approaches involves working with a single agent in an environment [44, 47, 48] as multiple agents introduce a challenge of working in non-stationary environments. While deploying multiple agents, the work remains independent without any communication [6, 16]. Through the TBONE architecture of the two-body problem [18], authors Jain et al. bring in a system where multiple embodied AI agents coordinate to accomplish a specific task. TBONE focuses on the problem of furniture lifting and explores how communication can enable agents to better learn by synchronising the interaction with the environment while keeping the navigation independent with some hidden benefits from communication. Cordial Sync [27], a work based on TBONE by authors Jain et al., extends it further for furniture movement with an introduction of synchronised and joint policies.

While both works achieve SOTA, certain aspects are either ignored or kept for future work. Two such aspects include (1)

[1]COLMAN: https://github.com/NavneetSinghArora/Attention_and_Move

synchronised policies during agent navigation and (2) inclusion of natural language for task-based command. Wortsman et al., through the work of Self-Adaptive Visual Navigation (SAVN) [21], show how agents can generalise well in unseen environments through self-supervised learning using ResNet [9] as image features and Glove embedding [8] of the target object class as textual features, training LSTM networks to perform visual navigation. In the work of Spatial Attention [34], Mayo et al. add a fusion map of the features to represent attention probability that can help direct the agent towards the right goal location. These works strengthen the understanding that using textual features can help enhance the agent's learning capabilities.

As the main focus of the approaches mentioned above is to improve the collaboration between the agents through communication, they do not focus on using the textual embeddings of the target object that proved helpful in previous works. In the real-world scenario, the instructions are given orally as spoken sentences. In order to let the agent know about the object in concern, we focus on incorporating Natural Language Processing (NLP) module to explore the effectiveness of textual embeddings to make agents understand the object's underlying semantic meaning. It would subsequently lead to better object detection. The agents also remain independent during navigation despite the availability of communication and joint policies. Considering the impact of Sync policies suggested in Cordial Sync, we also focus on how joint policies can be utilised and contribute even during the navigation phase. Furthermore, we hypothesise that integrating a more complex object detection model would improve performance. Therefore, we aim to make the task of navigation better by enabling object recognition for multiple objects capturing the spatial information when placing the agent anywhere in the scene. In order to achieve that, we incorporate the work of Contrastive Language-Image Pre-Training (CLIP) [35], a SOTA model that uses both image and textual features to generate semantic embeddings, to perform the task of visual navigation.

For this work, we take Cordial Sync as our baseline for modifications and updates. The code of Cordial Sync is updated to make it compatible with the latest version of AI2-THOR and Python. We removed insignificant parts[2] that were only relevant for furniture movement. Alongside, we train the CLIP model on our custom-generated dataset (Section-IV-C) from the simulation environment for object detection using textual features of the object class. Finally, we apply the fine-tuned CLIP model to the Cordial Sync refined baseline and report our findings using different metrics.

A. Individual Contributions

1) Navneet
   - CLIP Dataset Creation (Section-IV-C)
   - CLIP Training (Section-II-C)
   - COLMAN Training (Section-III, Section-IV-D)
2) Aida
   - Base Cordial Sync Training - Old Version (run with old environment)
   - Assistance for CLIP Training (Section-IV-C)
   - COLMAN Training (Section-III, Section-IV-D)
   - Environmental Impact Analysis (Section-IV-H)
3) Sana
   - Spatial Attention Experimenting (run with old environment)
   - Base Cordial Sync Training - Upgraded Version (Section-IV-D)
   - COLMAN Training (Section-III)
4) David
   - System Upgrade and Environment (Section-IV-A)
   - Spatial Attention Experimenting (run with new environment)
   - Initial integration of Cordial Sync into our project
   - HPC Support (Section-IV-B)
   - Base Cordial Sync Training (run with new environment and still erroneous code) (Section-IV-D)
5) Rana
   - CLIP Dataset Creation and debugging Support (Section-IV-C)
   - CLIP Training Assistance (Section-II-C)
   - COLMAN Training (Section-III)

We divide the rest of the paper into multiple sections and subsections. Section-II gives an overall view of the system and provides the technical depths into the different components involved. Section-IV provides detailed information about the experiments conducted, along with the metrics, dataset and results. Finally, we conclude with Section-V.

II. System Overview

The system follows three different components, including:

1) (TBONE) Two Body Problem: Collaborative Visual Task Completion,
2) Multi-Agent Sync Policies, and
3) CLIP: Contrastive-Language Image Pre-training [35].

To study the algorithmic abilities of our system, we firstly introduce these architectures.

A. Two Body Problem (TBONE): Collaborative Visual Task Completion

TBONE architecture explores the visual aspects of collaboration through inter-agent communication. The main task is lifting a heavy target object in an environment that a single agent cannot handle [18]. In order to achieve that, the authors Jain et al. make a constraint where the agents visually and independently navigate in the environment. At the same time, analysing two different aspects of communication: (1) implicit and (2) explicit communication. The agents do explicit communication while processing; (1) the history of observations and actions, (2) current observations, and (3) the communication received from the other agent. While doing the explicit communication, the implicit communication occurs through the visual scene observation, where the agents learn the positions and actions performed by the other agents over time.

As shown in Figure-1, at any time t, the agents receive and perceive the AI2-THOR-based scene visual input as an image i_t[3]. The perception takes place using CNN [10] based visual encoder yielding a vector embedding v_t. This vector embedding, in combination with the historical representations (past observations and actions limited to the previous time step and accumulated over time) h_{t-1}, aggregated by LSTM (Long Short-Term Memory) [2], computes a policy π_θ over all the available actions[4] as a probability distribution and finally chooses the best possible action a_t for the

[2]The furniture movement task is discarded, due to the limited time, available resources and complexity of the tasks involved.

[3]Visual Scene Input: All the images are the first-person views obtained from AI2-THOR.

[4]Available Actions: Done, MoveAhead, MoveLeft, MoveRight, RotateLeft, RotateRight, PickupObject.

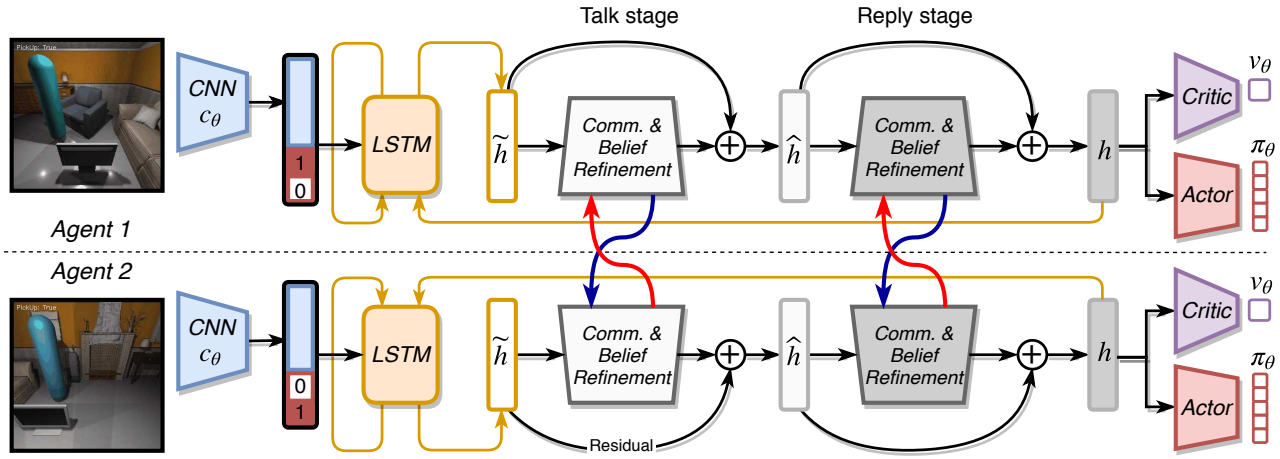


Figure 1: TBONE architecture. Figure from [18].

agent Equation-1. Along with the visual vector embeddings, the LSTM block also takes in a learnable embedding unique for each agent called an agent embedding e, concatenating it with visual vector embedding and producing the agent-specific beliefs. These beliefs get shared between the agents during the communication phase, allowing for better performance.

$$a_t = max[\pi_\theta(a_t|v_t, h_{t-1})] \quad (1)$$

As the main focus of this experiment was to analyse the effects of communication between the agents, the authors ignore the aspects of object detection by keeping a single target object. On top, the environment includes constraints, including (1) minimum distance between the object and the agent and (2) minimum distance between the two agents being the two most important ones.

The authors use the Reinforcement Learning (RL) based Asynchronous Advantage Actor Critic (A3C) [12] approach for model training. Furthermore, the work involves a warm start using Dagger [7] for efficient and faster model training, without which the training of the agents was infeasible.

B. Multi-Agent Sync Policies

Cordial Sync is an extension work of the TBONE authors to make agents collaboratively move the lifted object. The policies used in TBONE were calculated individually and were of rank one, which would diminish the ability of the model to learn complex policies. Therefore, the authors of Cordial Sync introduce Sync policies which enable more expressive joint policy [27]. The process is depicted in Figure-2.

In the usual multi-agent setup, each agent independently samples an action from its policy. However, this limits the policy learning to happen jointly among the agents. To enable some form of joint policy learning, the authors perform a two-step process. At first, for each agent at time step t, the output from the communication layers of TBONE C_t is used and fed into a shared function called f_θ which has θ learnable parameters. The function is represented as $\alpha_t = f_\theta(C_t)$. Here α_t is a multinomial consisting of shared probabilities and helps easily synchronise the actions' sampling across the agents. Moreover, shared seed computation takes place using the sum of each agent's randomly generated seed. Finally, this shared seed initiates sampling of j from the α_t.

Next, each agent generates multiple policies $\pi_{t,1}..\pi_{t,m}$ using $m-1$ additional linear layers in the TBONE, where $m <= |a|$ and a is the finite set of possible actions. The policy π_j for choosing the action is selected using the generated j value. This way, the model shared some part of the process of selecting an action, therefore, making the policy more expressive. Hence, the agents can generate multiple policies (high ranked) and synchronise the action through them when combined with the mixture weight α_t and the outer product from each agent's selected policies. This process is called a mixture-of-marginals, and for two agent setup, it equates to:

$$\text{mixture-of-marginals} = \Sigma_{j=1}^{m} \alpha_j * \pi_j^1 \otimes \pi_j^2 \quad (2)$$

This joint policy better represents a decentralised execution of the agents and is high rank and therefore can represent complex policy for complex tasks such as coordination.

C. CLIP: Contrastive-Language Image Pre-training

In the domain of NLP, pre-training methods, learning directly from the raw text on the web, created a new trend in the world of deep learning models. It leads to the emergence of models like BERT [15] and GPT-3 [22], suggesting that specific training datasets are not required to be competitive across various tasks. The authors, Radford et al., try to use this approach in the computer vision domain, where pre-training with standard datasets like ImageNet [11] is standard practice. They bring in another rare aspect of learning image representations using textual supervision.

In order to pre-train the model, the authors prepare a custom dataset (400 million image-text pairs). Furthermore, to train this dataset, more efficient contrastive learning [45] is adopted compared to the supervised and generative learning approaches. As shown in Figure-3 the model jointly trains an image encoder for learning the image feature representations and the text encoder for extracting the textual encoding before contrastively matching the pairs (blue boxes in Figure-3).

Despite the success of contrastive learning approaches, the minimal sufficient representation [33, 37] is insufficient for a wide range of downstream tasks [46]. Similarly, Wang et al. [38] refer to the presence of under-clustering and over-clustering, depending on the number and quality of positive and negative samples available during training. Zheng et al. [41] also lead to a similar conclusion, but from a different perspective.

However, most of these approaches fine-tune the underlying model using the contrastive approach, which might not be feasible

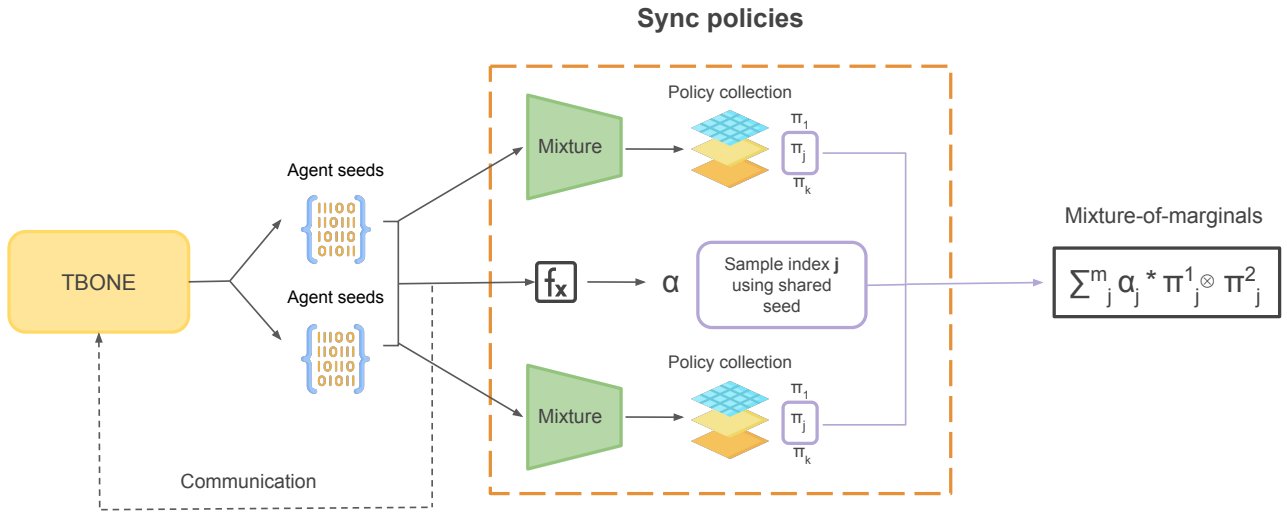


Figure 2: Using TBONE as a backbone to compute mixture-of-marginals using Sync policies through random seeds of each agent. Figure by the authors.

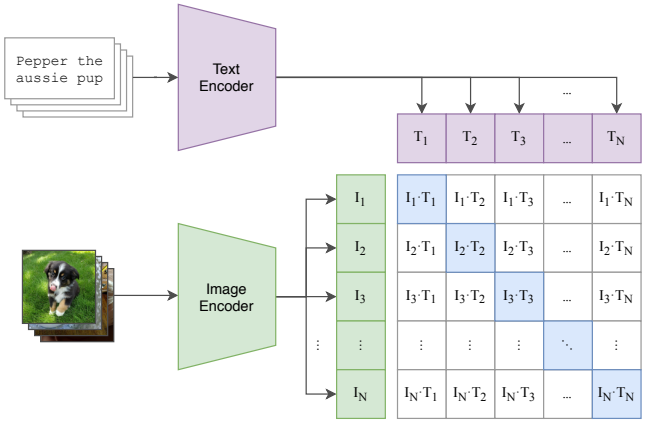


Figure 3: CLIP: Contrastive Learning pre-training representation schema using contrastive matching of the text and image features as shown in blue diagonal values. Figure from [35].

in many cases. In contrast, as suggested, explored and successfully proven in recent studies [25, 26, 33], fine-tuning the contrastive models in a supervised manner can lead to better performance in the underlying downstream task. Also, considering the embodied AI agents, simulator environments and the scenes/objects available, computing the good quality negative and positive samples is difficult, especially with a multi-agent setup.

Hence, we modify the underlying CLIP model for fine-tuning the embodied AI task using the custom dataset (Section-IV-C). Figure-4 (top) shows the modified architecture. The initial structure of the model remains the same, where the image-text[5] pair of AI2-THOR visual scenes (3x224x224 image resolution) are used as input from the collected dataset and passed onto the respective transformer-based image encoder (Vision Transformer [31], a ViT-B/32 having 32 layers with vector embedding of 1x1280) and transformer-based text encoder (Text Transformer [14], 12 layer model with eight attention heads with vector embedding of 1x512). The respective encoders result in the required visual scene's image feature embeddings (transformed to vector embedding of 1x512) and object-based scene's textual embeddings (transformed to vector embedding of 50x512). After that, the model computes the scene similarity to transform it further using the Softmax [1] layer (transformed to a vector embedding of 1x50, pertaining to 50 distinct objects available in the scene). It finally results in the combined probability distribution of the objects in the scene. As the fine-tuning works using a supervised approach, the contrastive loss is replaced by the probabilistic cross-entropy loss [17]. For computing the loss, the object probability distribution is computed from the scene metadata, containing the information related to the proportion of object presence and visibility in visual input. We observe that the model not only becomes capable of detecting the objects but also learns the spatial information (nearby objects in the scene to the target object). This is later explained in Section-IV-F.

[5]the text for every input image contains the class name of the objects available in the scene

III. COLMAN: Collaborative Multi-Agent Navigation using Textual-Visual Embeddings

We now describe the proposed architecture for the furniture lifting task, which we name COLMAN. The agents locate the object and then navigate towards it. The overall architecture is shown Figure-4.

Each agent gets an egocentric view of the object in the form of an RGB image of dimension 3x224x224. This local visual observation is first passed into the CLIP model and processed as described in Section-II-C. The scenes of the environment can be static or dynamic, further described in Section-IV-D. CLIP generates a vector embedding whose dimensions depend upon the use of textual features. If textual features are incorporated, then the output vector embedding is of dimension 1x50; otherwise, dimension 1x512. An additional learnable vector embedding of dimension 1x8 concatenates to this output vector embedding that gives the agents the capacity to develop distinct, complementary strategies [18]. The following two modules, LSTM and communication, are like how it

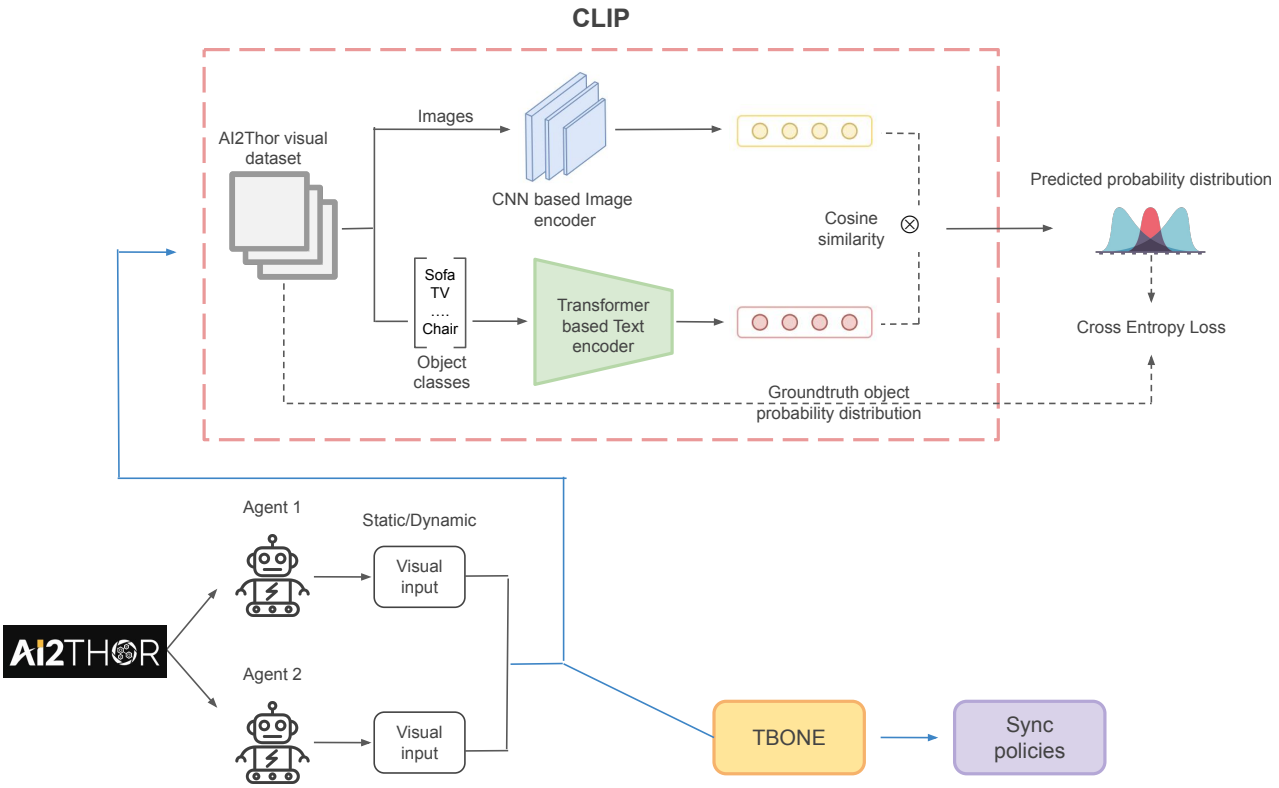


Figure 4: COLMAN architecture: Combining CLIP (top), TBONE and Sync Policies. Figure created by the authors.

is in TBONE after CNN layers, as described in Section-II-A. LSTM takes the CLIP output vector embedding and the history of previous observations to generate the agents' beliefs about the environment. In a multi-agent environment, it is imperative that the agents have some form of communication and coordination for a collaborative task of furniture lifting. Communication is incorporated among agents using communication modules, namely the talk and reply stage, to refine the local belief of the agents. It helps reduce the number of unsuccessful pickups and for better coordination [27]. This refined belief facilitates the generation of policy for learning. The policy generation happens using sync policies as described in Section-II-B. A3C algorithm processes this generated policy and enables learning in the agents.

The following section explains the critical components and related experiments to train this model.

IV. Empirical Analysis

A. System Upgrade and Environment

During the implementation phase of our project, when training the native Cordial Sync implementation on our systems for evaluating stability and capability, we encountered two bottlenecks. The first bottleneck was the inability to run the old AI2-THOR version 2.1.1 headless on a system without root privileges. Secondly, the old AI2-THOR version provokes various instabilities and performance issues. Despite working around these bottlenecks, we decided to update requirements to the recent and more stable version 4.3.0 of AI2-THOR that (1) allows using CloudRendering and (2) includes various stability fixes. When upgrading, other vital components such as Python and PyTorch have also been elevated to current versions to improve overall performance and stability. The code base of Cordial Sync was updated accordingly to ensure that the code behaviour stayed the same after accommodating upgrades. Moreover, we have reduced the code base to only support FurnLift and no FurnMove experiments.

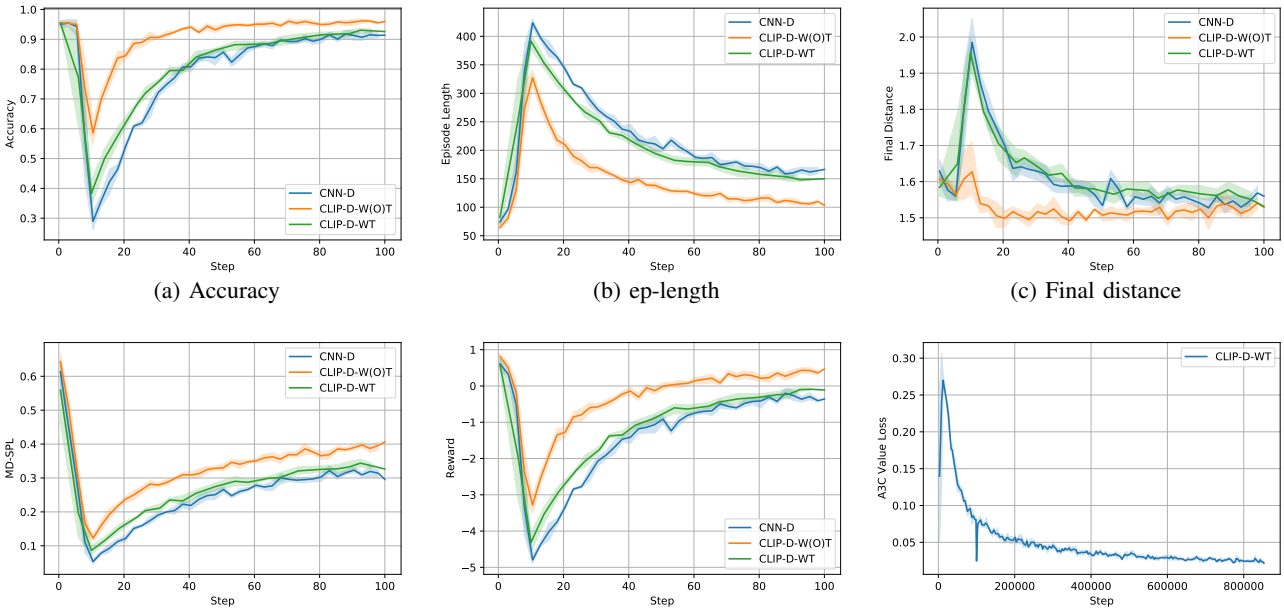
B. High-Performance Computing (HPC) Support

It was unclear at the project's early stages if our department could provide enough resources for model training, evaluation and testing. Therefore the project has been set up to support the HPC cluster run by the University of Hamburg. HPC support is enabled by providing Python and Bash scripts for (1) pre-fetching prerequisites such as the AI2-THOR environment binary and (2) creating a SingularityCE [43] container that allows the execution of our project on the HPC cluster using Simple Linux Utility for Resource Management (SLURM) [3] and (3) copying the pre-fetched data, pre-build container and necessary scripts onto the Hummel cluster needed for remote setup and execution of our project[6]. As the project progressed, the department provided us with a workstation that made further development of better Hummel integration obsolete. However, we believe that the provided implementation and documentation will guide other scientists in setting up a complex project on HPC and is, therefore, valuable.

C. CLIP Dataset

In order to fine-tune the pre-trained CLIP model, we create a custom dataset. The data capturing for the dataset takes place

[6]We base our work on the scripts for the creation of a SingularityCE container and the execution on Hummel via SLURM on work kindly provided by Dr Mikko Lauri.



(a) Accuracy (b) ep-length (c) Final distance

(d) MD SPL (e) Reward (f) A3C Loss

Figure 5: Metric Comparison using dynamic scenes. From top-left to bottom-right; the figure shows (a) accuracy (b) episode-length (c) final distance (d) MD-SPL (e) reward and (f) A3C Loss

using the simulation environment of AI2-THOR. The data collection process involves creating a multi-agent environment[7].

The visual inputs collected in the dataset are first-eye views from the agents. For each scene, both agents get randomly initialised[8] in the environment. As the degree of rotation for the agent is 30 degrees, each agent captures 12 distinct images at a given position in the scene. While capturing the visual scene, one agent is always static. This constraint ensures that no two images captured are redundant in the process. The dynamic agent can only do a set of actions, limited to *MoveAhead, MoveBack, MoveLeft, MoveRight, RotateRight and Done*.

Along with capturing the visual scene input, the process also saves the visual scene metadata information, including (1) available objects in the scene, (2) visible objects in the scene and (3) bounding box information of the scene. It is necessary to create ground truth information while fine-tuning the model. In this order, a total of 400,000+ images with the ground-truth data for model training have been collected, pertaining to a set of 30 LivingRoom scenes. Finally, the dataset gets split into three sets of train/val/test in the ratio of 60:20:20.

D. Experimentation

The training arguments of Cordial Sync and COLMAN are almost the same as they were for TBONE. Warm-start occurs for the first 10,000 episodes as explained in Section-II-A. We set the number of agents as two. We set television as the object of interest. The scene types that the agents explore can be static or dynamic. If the scenes are static, then the location of the television will always be the same in every iteration of an episode for that particular scene. If it is dynamic, then the television is placed on the floor in a randomly chosen location in the scene. After this, the agents choose their actions purely using the generated policy. An episode is successful if both the agents reach within 500 time steps(250 per agent) and are on either side of the television within a 1.5m distance. We set the horizon of the agents to 0 degrees as opposed to 30 degrees in TBONE. Rewards are discrete, where a successful pickup leads to 1 for each agent for successful pickup, -0.01 step penalty to discourage long trajectories and -0.02 for failed action [18].

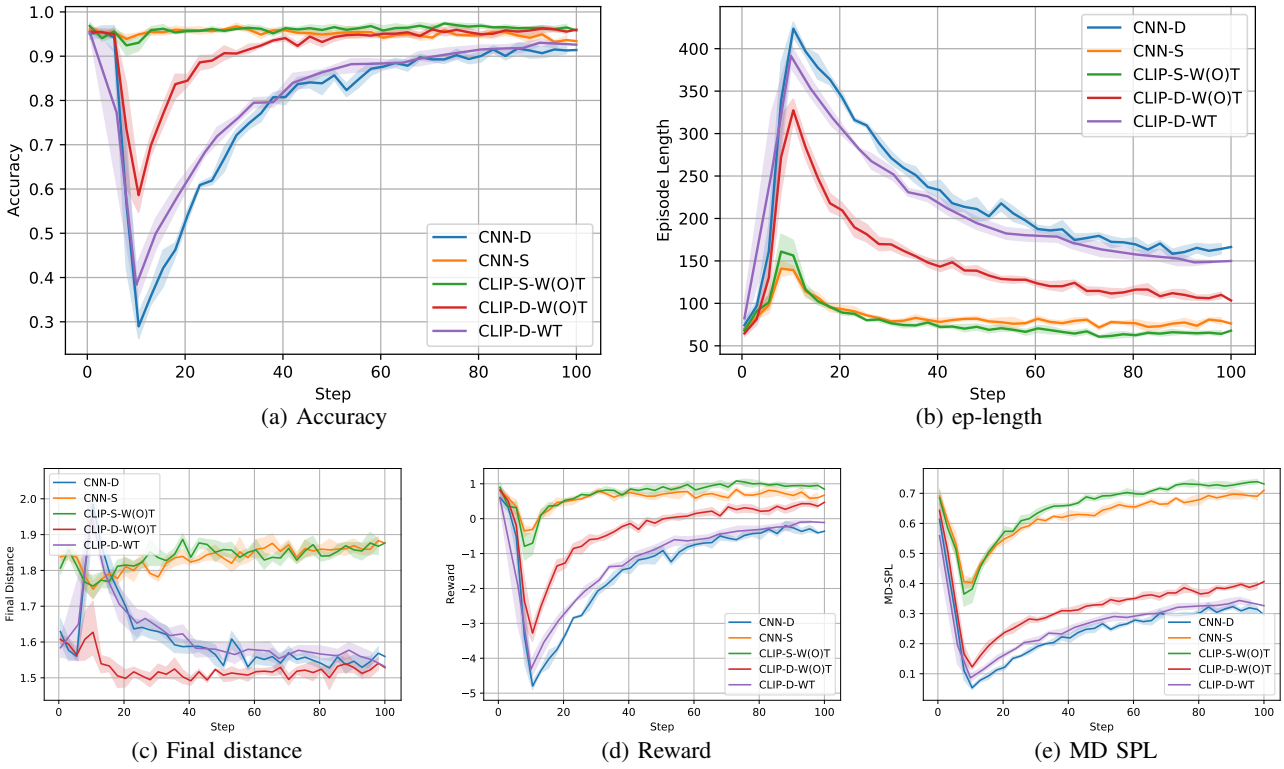
As mentioned in Section-IV-A. Our first experiment is to replicate the baseline defined in the paper Cordial Sync, which uses CNN along with dynamic scenes on the old AI2-THOR version. Following that, we use the upgraded AI2-THOR to perform our next experiments. To verify the model's functioning using the upgraded version, we perform the following two experiments: with CNN for object detection using static scenes (CNN-S) and dynamic scenes (CNN-D). These experiments are a part of our ablation study to observe the potential benefits the model could gain in performance by upgrading the versions. We report the results in Table-I.

As our next step was integrating CLIP in Cordial Sync, we fine-tuned the CLIP model on the AI2-THOR dataset we created to have a pre-trained CLIP model that we use in COLMAN. We report the training results of this fine-tuning in Table-I.

Our next set of experiments consists of training COLMAN where we incorporate our enhancements, i.e., using the fine-tuned CLIP model instead of CNN. We conducted three experiments varying the models based on scene types and the use of textual features with CLIP, where we expect CLIP to perform better than CNN. We use two different variants of CLIP that depends on the use of textual features. We perform the experiments on these variants

[7]The environment is limited to the living room scenes available in AI2-THOR for reduced complexity and to control the size of the dataset

[8]Random position initialisation takes place using the available reachable positions of a scene loaded in the environment



(a) Accuracy (b) ep-length

(c) Final distance (d) Reward (e) MD SPL

Figure 6: Metric Comparison for all experimentation scenarios. From top-left to bottom-right; the figure shows (a) accuracy (b) episode-length (c) final distance (d) reward and (e) MD-SPL.

using static and dynamic scenes. To sum up, these make up our three experiments with their results in the tables mentioned, namely CLIP without textual features and static scenes(CLIP-S-W(O)T), CLIP without textual features and dynamic scenes (CLIP-D-W(O)T), and CLIP with textual features and dynamic scenes (CLIP-D-WT). The results are reported in Table-I. We do not include CLIP with textual features and static scenes for the reason mentioned in Section-IV-H. We also use the first two experiments for the ablation study to analyse the differences in scene types. We hypothesise that CLIP performs better with textual features and that dynamic scenes would help prevent overfitting and provide a large variety of scene settings for optimal learning. Hence, the model that uses CLIP with textual features and dynamic scenes would give us the best overall performance.

E. Metrics

We use the same evaluation metrics as the Cordial Sync experiment to compare our results qualitatively and quantitatively to baseline results. The first metric we use is the Manhattan-Distance based Success Path Length (MD-SPL) which is adapted from regular SPL and is as follows:

$$\text{MD-SPL} = {N_{\text{ep}}}^{-1} \sum_{i=1}^{N_{\text{ep}}} S_i \frac{m_i \;/\; d_{\text{grid}}}{\max\left(p_i, m_i \;/\; d_{\text{grid}}\right)} \tag{3}$$

where N_{ep} represents the test episodes, S_i stands for the respective episode's i success or failure, m_i brings in the Manhattan distance from the start location of the target to the goal location, d_{grid} is the distance between adjacent grid points, and p_i is the number of actions executed by each agent.

Besides accuracy and reward, we also include episode length and final distance in the metrics. Episode length is determined based on the average number of actions each agent takes in an episode. At the end of each episode, the final distance indicates the distance left to cover by the agents to reach the target.

F. Results

When comparing CLIP-D-WT and CNN-D on their training values, we observe that the accuracy trend is similar, as seen in Figure-5(a). However, when we look at the trend on the validation set in Figure-7, we observe that CLIP-D-WT performs much better than CNN-D on unseen scenes, which is also reflected by smaller episode lengths and final distance of CLIP-D-WT as can be seen on Table-I.

Moreover, we can see a slower convergence of CLIP-D-WT when looking at the accuracy, episode length, and reward values reported on Table-I, which is visible through the loss trend in Figure-5(f). Therefore when running CLIP-D-WT for an extreme 500k episodes, the results are comparable to those of CLIP-D-W(O)T in terms of the episode length, training accuracy and reward and a much better result for MD-SPL.

When comparing validation results of CNN-D with CLIP-D-WT in Table-II, we observe a similar trend as seen on training values, but here CLIP-D-WT outperforms CLIP-D-W(O)T on the episode

Table I: Quantitative results for Training

Experiment	MD-SPL ↑[1]	Ep len ↓[1]	Final dist ↓[1]	Accuracy (Success) ↑[1]	Reward ↑[1]
Static scenes (100K episodes)					
CNN-S	0.1390	176.9	2.228	0.59	0.6749
CLIP-S-W(O)T	0.7489	67.85	1.876	0.96	0.844
Dynamic scenes (100K episodes)					
CNN-D	0.3057	165.4	1.570	0.91	-0.3592
CLIP-D-W(O)T	0.4064	103.4	1.529	0.96	0.4692
CLIP-D-WT	0.3235	149.6	1.535	0.93	-0.1106
Dynamic scenes (300K episodes)					
CLIP-D-WT	0.4052	118	1.588	0.95	0.3069
Dynamic scenes (500K episodes)					
CLIP-D-WT	0.4302	102.2	1.544	0.96	0.552

Table II: Quantitative results for Validation

Experiment	MD-SPL ↑[1]	Ep len ↓[1]	Final dist ↓[1]	Accuracy (Success) ↑[1]	Reward ↑[1]
Static scenes (100K episodes)					
CNN-S	1.158e-27	500	3.61	4.0018e-3	-8.938
CLIP-S-W(O)T	1.8807e-4	499.7	3.088	0.02211	-7.851
Dynamic scenes (100K episodes)					
CNN-D	0.1641	321.8	2.286	0.3603	-2.539
CLIP-D-W(O)T	6.1424e-7	500	2.977	0.09728	-7.143
CLIP-D-WT	0.1582	290.1	1.798	0.4195	-3.334
Dynamic scenes (300K episodes)					
CLIP-D-WT	0.08395	310	2.047	0.4196	-2.788
Dynamic scenes (500K episodes)					
CLIP-D-WT	0.03844	462.1	1.839	0.4194	-5.781

[1] An arrow points downwards or upwards according to the desired values of the respective metric. For MD-SPL, Accuracy, and Reward, the higher is better. However, the Episode length and Final distance follow the lower the better pattern. All the performance metrics are recorded and analyzed through tensorboard.

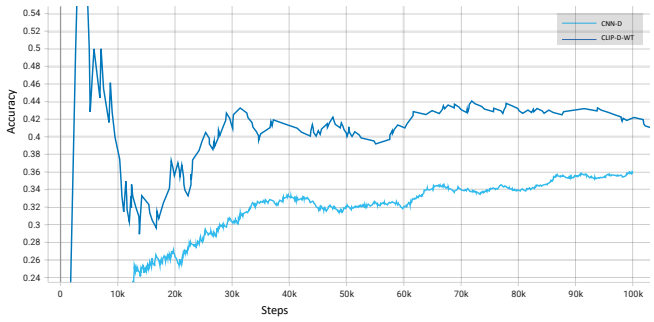


Figure 7: Accuracy (Validation)

length, final distance and accuracy. Any further training of CLIP-D-WT resulted in improved MD-SPL and reward metrics; however, between 300k and 500k episodes, an increase in episode length and decrease in reward indicates overfitting. Furthermore, Figure-5 shows trends of other metrics computed on the dynamic scene environment. In all these evaluation metrics, CLIP-D-WT is seen as outperforming in all scenarios.

G. Ablation Studies

1) Effect of upgrades on Cordial Sync: After incorporating the updates to the original Cordial Sync FurnLift experiment, we could see that the upgrades did not cause any significant problems. We observed that the model was stable; however, the accuracy for Cordial Sync remains higher with a value of 0.423 compared to 0.3603 for CNN-D. Interestingly, while Cordial Sync performed better for most metrics, MD-SPL for CNN-D is considerably higher than that for Cordial Sync.

2) Effect of changes in scene type: One of our configuration setups was static or dynamic scene type. As the target object (television) in the old AI2-THOR is always on the table, it is fascinating to see the upgraded version with a more realistic scenario of placing it on the wall and other locations. It provides a reason to explore the environments where the object is not directly on the horizon of the agents. However, the real-life environment is not static, and agents must learn how to navigate and perform in the dynamic one.

As shown in Figure-6, the models executed in the static environment converged faster and over-fitted on the training dataset in the early stages. In contrast, the models in the dynamic environment allowed more options for explorations and provided multiple scenarios within the limited number of scenes. It resulted in models taking much more time to learn and converge but were more adaptable to not yet seen scenes from the validation set.

Nevertheless, we noticed that CLIP-S-W(O)T outperformed CNN-S. For 100K steps, MD-SPL increases from 0.1390 to 0.7489,

and accuracy jumps from 0.59 to 0.96. Due to CLIP's ability to memorise the target object's location (TV), it starts overfitting. Therefore, we observe a substantial reduction in episode length, from 176.9 to 67.85 and in final distance from 2.228 to 1.876. Missed pickups value reduces from 0.08 to 0.062. In contrast, Failed pickups seem to be higher than 0.144 at exactly 100K episodes. However, if smoothing is applied, we can see the trend line is downward, indicating CLIP performs better when replaced with CNN in static scene configuration.

3) Effect of using textual features in CLIP: During training, all CLIP-D-W(O)T metrics are better than the CLIP-D-WT for 100K episodes. It happens due to the slow convergence of the CLIP model. However, if we keep training CLIP-D-WT for 300K and even for 500K episodes, as shown in Table-I, then CLIP with text start surpassing CLIP without text. It occurs because the CLIP model with textual features takes a long time to converge, but once trained fully, it performs better.

By looking at the validation accuracy, we can see differences between CLIP-D-W(O)T and CLIP-D-WT. CNN-D and CLIP-D-WT both outperform CLIP-D-W(O)T on accuracy over the validation set, which shows that using textual features with CLIP has a positive effect and is necessary for better performance.

H. Environmental Impact Analysis

With computers becoming more powerful, there is a growing demand for training large models on massive datasets. The results of such pre-training lead to the best-performing model. However, this comes with the cost of energy consumption and the number of emissions. Hence, it becomes essential for scientific papers to report not only experiment details and model performance but the model's efficiency as an evaluation criterion (as proposed by [28]). In this section, we report hardware details, the corresponding time required to train the model with five different configurations and the estimated carbon emissions.

We perform all the experiments using a private infrastructure on NVIDIA TITAN X hardware. To conduct a more efficient study, we made a few arrangements. The first one is while incorporating CLIP. While training the CLIP model, we decided to freeze all but the last fully-connected layers to speed up the process. CLIP model has been pre-trained on a huge dataset with a wide variety of objects; hence for our task, it was enough to fine-tune the last layers of the model on the collected photo-realistic dataset. The model trained with frozen layers was still robust for the image recognition task; however, the training with such a setup took less time to be executed.

Overall, we conducted two experiments with static scenes using CNN and CLIP, respectively. As one of our interests was to see the impact of textual embeddings on the model's performance, we had to test CLIP with and without text features. However, we noticed that the model was overfitting with CNN and CLIP without text with the static scenes scenario. Therefore we have decided not to execute COLMAN using CLIP with text features to save time and resources.

By the end of our experiments, we could observe that COLMAN with CLIP object detection was performing slightly better than with CNN. However, we noticed a considerable difference in training times in both static and dynamic scenarios. COLMAN with CNN in dynamic scenes was executed for 37 hours, which resulted in approximately 4 kg of CO_2eq. emissions, the model with CLIP and text features completed training 6 hours earlier and resulted in approximately 3.35 kg of CO_2eq.. These results are equivalent to 16.1 and 13.5 km of a standard ICE drive, as seen in Table III. The model with CLIP and without text features had the worst performance with respect to time, we assume that such behaviour occurred due to the fact that the size of the embedding with this setup was larger than for others; hence, it resulted for a model to take more time to learn.

Table III: Experiment time and estimated emissions.

Experiment	Time (hours)↓	CO_2eq. emissions (kg) ↓	Equivalence in ICE drive (km)↓
Static scenes			
CNN-S	15	1.62	6.55
CLIP-S-W(O)T	10	1.08	4.36
Dynamic scenes			
CNN-D	37	4	16.1
CLIP-D-W(O)T	43	4.64	18.7
CLIP-D-WT	31	3.35	13.5

Throughout the study, we faced multiple hardware issues, which resulted in many unsuccessful runs that the paper does not report. Another point is that the paper this study is based on report their results after 100K epochs. Hence to make our results comparable with theirs, we also ran the experiments for 100K epochs; however, we could observe promising results even before. It brings us to the fact that the studies primarily focus on performance disregarding the resources it takes to achieve them, so hopefully, new studies will aim to find a balance between performance and efficiency of the model.

The estimations were conducted using the MachineLearning Impact calculator presented in [19]. We understand that the results are only approximations of real carbon emissions, and there are much more factors that are not yet involved in calculations. However, we believe that the works should report the environmental impact of ML models.

V. Conclusion and Discussion

We propose COLMAN, a model that collaboratively navigates towards a target object. We use Cordial Sync as the baseline and extend the work to include the CLIP model for performing object detection and navigation while incorporating textual features. The upgrades in the model help in the further development and extension of the modules. We observed through the experiments that using textual features can improve the overall task performance on the unseen dataset. We also see that COLMAN with textual features take longer to converge than the baseline model, which can be attributed to the fact that transformers take longer to learn. That could also be the reason why our training trend was similar to the CNN baseline. Another aspect is the reduced vector dimensions which influence the overall learning of the CLIP model and results in slower model convergence.

Due to lack of time and technical challenges, the task of FurnMove is considered as the future work, where the agents collaborate to pickup and move the objects in the environment. Thus, limiting the current scope to enhancing visual object detection and navigation. Despite the progress, there is a lot which can be achieved and have not been explored in the multi-agent environment. As seen with the results, the model performance improves with the inclusion of textual embeddings of the target object. This can be further enhanced to include the text along the command based textual embeddings to incorporating complete sentences to generate a more semantic understanding of the visible environment. We can extend this work to involve graph algorithms for better object search

and aiding in navigation. Another opportunity presented by this result is to add the speech modality along with the depth images. Considering the amount time taken to generalise the models, it proves that the communication between the agent to create joint policies needs further improvement.

Our contributions to executing training on Hummel nodes will also be helpful for the future in order to fully utilise with the multi-processing capabilities. Moreover, this research also shows the amount of impact these large machine learning models have on the environment, which can help us to plan and execute more sustainable studies in the future.

References

[1] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2, Morgan-Kaufmann, 1989. [Online]. Available: https://proceedings.neurips.cc/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[3] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60, ISBN: 978-3-540-39727-4.

[4] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006. DOI: 10.1109/MRA.2006.1678144.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. DOI: 10.1109/MRA.2006.1638022.

[6] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489.

[7] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162.

[9] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385. [Online]. Available: https://arxiv.org/abs/1512.03385.

[10] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. DOI: 10.48550/ARXIV.1511.08458. [Online]. Available: https://arxiv.org/abs/1511.08458.

[11] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[12] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, Jun. 2016, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html.

[13] E. Kolve *et al.*, *Ai2-thor: An interactive 3d environment for visual ai*, 2017. DOI: 10.48550/ARXIV.1712.05474. [Online]. Available: https://arxiv.org/abs/1712.05474.

[14] A. Vaswani *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805. [Online]. Available: https://arxiv.org/abs/1810.04805.

[16] J. Godoy, T. Chen, S. J. Guy, I. Karamouzas, and M. Gini, "Alan: Adaptive learning for multi-agent navigation," *Autonomous Robots*, vol. 42, no. 8, pp. 1543–1562, 2018.

[17] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/f2925f97bc13ad2852a7a551802feea0-Paper.pdf.

[18] U. Jain *et al.*, "Two body problem: Collaborative visual task completion," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6682–6692. DOI: 10.1109/CVPR.2019.00685.

[19] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," *arXiv preprint arXiv:1910.09700*, 2019.

[20] M. Savva *et al.*, *Habitat: A platform for embodied ai research*, 2019. DOI: 10.48550/ARXIV.1904.01201. [Online]. Available: https://arxiv.org/abs/1904.01201.

[21] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6743–6752. DOI: 10.1109/CVPR.2019.00691.

[22] T. B. Brown *et al.*, *Language models are few-shot learners*, 2020. DOI: 10.48550/ARXIV.2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[23] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 4247–4258. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/2c75cf2681788adaca63aa95ae028b22-Paper.pdf.

[24] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[25] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jul. 2020, pp. 1597–1607. [Online]. Available: https://proceedings.mlr.press/v119/chen20j.html.

[26] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[27] U. Jain *et al.*, "A cordial sync: Going beyond marginal policies for multi-agent embodied tasks," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V*, Glasgow, United Kingdom: Springer-Verlag, 2020, pp. 471–490, ISBN: 978-3-030-58557-0. DOI: 10.1007/978-3-030-58558-7_28. [Online]. Available: https://doi.org/10.1007/978-3-030-58558-7_28.

[28] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.

[29] B. Shen *et al.*, "Igibson 1.0: A simulation environment for interactive tasks in large realistic scenes," 2020. DOI: 10.48550/ARXIV.2012.02924. [Online]. Available: https://arxiv.org/abs/2012.02924.

[30] S. Wani, S. Patel, U. Jain, A. Chang, and M. Savva, "Multion: Benchmarking semantic map memory using multi-object navigation," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 9700–9712. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/6e01383fd96a17ae51cc3e15447e7533-Paper.pdf.

[31] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.

[32] H. Du, X. Yu, and L. Zheng, *Vtnet: Visual transformer network for object goal navigation*, 2021. DOI: 10.48550/ARXIV.2105.09447. [Online]. Available: https://arxiv.org/abs/2105.09447.

[33] L. Ericsson, H. Gouk, and T. M. Hospedales, "How well do self-supervised models transfer?" In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 5414–5423.

[34] B. Mayo, T. Hazan, and A. Tal, "Visual navigation with spatial attention," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 16 893–16 902. DOI: 10.1109/CVPR46437.2021.01662.

[35] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, Jul. 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html.

[36] Z. Seymour, K. Thopalli, N. Mithun, H.-P. Chiu, S. Samarasekera, and R. Kumar, "Maast: Map attention with semantic transformers for efficient visual navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 223–13 230. DOI: 10.1109/ICRA48506.2021.9561058.

[37] Y.-H. H. Tsai, Y. Wu, R. Salakhutdinov, and L.-P. Morency, "Self-supervised learning from a multi-view perspective," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=-bdp_8Itjwp.

[38] G. Wang, K. Wang, G. Wang, P. H. Torr, and L. Lin, "Solving inefficiency of self-supervised representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 9505–9515.

[39] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recogni-*

tion (CVPR), 2021, pp. 5918–5927. DOI: 10.1109/CVPR46437.2021.00586.

[40] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectgoal navigation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 16 097–16 106. DOI: 10.1109/ICCV48922.2021.01581.

[41] M. Zheng *et al.*, "Ressl: Relational self-supervised learning with weak augmentation," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 2543–2555. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/14c4f36143b4b09cbc320d7c95a50ee7-Paper.pdf.

[42] R. Fukushima, K. Ota, A. Kanezaki, Y. Sasaki, and Y. Yoshiyasu, *Object memory transformer for object goal navigation*, 2022. DOI: 10.48550/ARXIV.2203.14708. [Online]. Available: https://arxiv.org/abs/2203.14708.

[43] G. M. Kurtzer *et al.*, *sylabs/singularity: SingularityCE 3.9.8*, version v3.9.8, Apr. 2022. DOI: 10.5281/zenodo.6423401. [Online]. Available: https://doi.org/10.5281/zenodo.6423401.

[44] M. K. Moghaddam, E. Abbasnejad, Q. Wu, J. Q. Shi, and A. Van Den Hengel, "Foresi: Success-aware visual navigation agent," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 691–700.

[45] N. Saunshi *et al.*, *Understanding contrastive learning requires incorporating inductive biases*, 2022. DOI: 10.48550/ARXIV.2202.14037. [Online]. Available: https://arxiv.org/abs/2202.14037.

[46] H. Wang, X. Guo, Z.-H. Deng, and Y. Lu, "Rethinking minimal sufficient representation in contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 16 041–16 050.

[47] Q. Wu, J. Wang, J. Liang, X. Gong, and D. Manocha, "Image-goal navigation in complex environments via modular learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6902–6909, 2022. DOI: 10.1109/LRA.2022.3178810.

[48] X. Hou, H. Zhao, C. Wang, and H. Liu, "Knowledge driven indoor object-goal navigation aid for visually impaired people," *Cognitive Computation and Systems*, vol. n/a, no. n/a, DOI: https://doi.org/10.1049/ccs2.12061. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ccs2.12061. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ccs2.12061.