CHMARL: A Multimodal Benchmark for Cooperative, Heterogeneous Multi-Agent Reinforcement Learning

Vasu Sharma*[1], Prasoon Goyal*[1], Kaixiang Lin*[1], Govind Thattai[1], Qiaozi Gao[1], Gaurav S. Sukhatme[1,2]
[1]Alexa AI, Amazon Inc [2]University of Southern California

***Abstract*—We propose a vision-and-language benchmark for cooperative and heterogeneous multi-agent learning. We introduce a benchmark multimodal dataset with tasks involving collaboration between multiple heterogeneous agents in a rich multiroom home environment. We provide an integrated learning framework, multimodal implementation of the state-of-the-art, and consistent evaluation protocol. Our experiments investigate the impact of different modalities on the learning performance. We also introduce a simple message passing method between agents. The results suggest that multi-modality introduces unique challenges for cooperative multi-agent learning and there is significant room for advancing MARL methods in such settings.**

I. Introduction

We posit that progress in multi-agent learning could be sped up with the introduction of standard, sophisticated environments for training and evaluation. Prior work on cooperative multi-agent learning has focused on simplified environments [12]. Visually rich environments that support multi-agent, cooperative tasks have not been explored until very recently [17, 7, 6, 14, 16]. We propose the first *multimodal* benchmark - Cooperative Heterogeneous Multi-Agent Reinforcement Learning (**CHMARL**) - wherein two agents must collaboratively find an object and place it at a target location.

CHMARL is built using visually rich scenes from VirtualHome [14], and includes language. We implement a language generator that procedurally provides feedback to guide embodied agents to achieve tasks. In addition to providing a novel large-scale vision and language dataset for collaborative task completion in simulated household environments, we conduct a comprehensive evaluation of several state of the art MARL algorithms under various setting for our benchmark task. We investigate and analyze the impact of various aspects of the collaborative MARL algorithms, including heterogeneity and multi-modality. We also propose and implement a message passing interface between agents to enable effective information sharing, especially in decentralized model setups where they would otherwise not have the ability to collaborate with each other. The results reveal interesting insights: 1) The multimodal (vision and language) setting presents an extra challenge for existing techniques; 2) Vision and language grounding helps the learning process; and 3) Even simple multi-agent communication protocols substantially improve task performance by allowing effective collaboration. We expect this work to contribute towards a standard multi-modal testbed for MARL and foster research in this direction.

II. Related Work

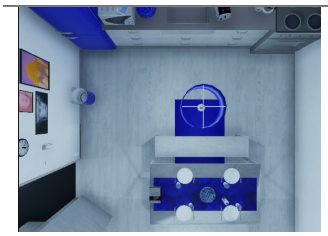
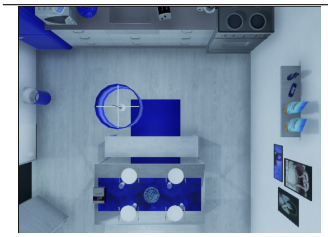
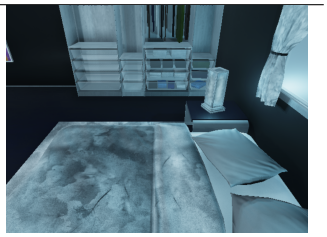
Collaborative multi-agent RL is well-studied problem. For brevity we only mention the most relevant work here. Kurenkov et al [9] consider the object finding problem, where the target object is described using natural language. However, their focus is to exploit semantic priors about object placements (e.g. cheese is likely to be found in a fridge, which in turn is likely to be in the kitchen). In our setup, objects are not placed according to semantic priors, since our focus is multi-agent collaboration to search for and move objects efficiently. Jain et al ([6], [7]) and Nachum et al [13] propose a setting where the agents must perform certain actions synchronously, e.g., for lifting a heavy object. This is different from our setting, which doesn't require synchronous actions; rather, the agents need to communicate with each other to explore the environment efficiently. These prior works assume homogeneous agents, whereas we consider heterogeneous agents. Zhu et al [18] study object finding an object in a multi-agent setup, but unlike us, their setting does not involve interactions with the environment, and the agents are homogeneous. Liu et al among others ([11], [10], [3], [4])) consider the problem of collaborative perception, where there are some degraded agents, and the goal is to learn an efficient communication strategy to improve the observation of the degraded agents. Unlike our setting, their agents are fixed and there is no interaction (navigation or manipulation) with the environment.

III. Proposed Benchmark

A. Problem Setting

Our setting consists of two embodied agents with different capabilities situated in a home environment. The agents need to cooperate to efficiently complete a task described in natural language. The first agent is a humanoid, while the second agent is a drone. The humanoid agent (H) has an egocentric field of view, and can physically interact with objects in the environment, while the drone agent (D) has a top-down view of a part of the environment, but cannot physically interact with objects. Given a task described in natural language, such as "Put a glass on the desk", the goal is to complete it in as few steps as possible. In order to complete the task, the agents need to find the objects of interest (i.e. *glass* and *desk*), and the ground agent needs to perform pick-and-place operations to accomplish the desired configuration. Effective inter-agent

TABLE I: **Drone and ground agent view examples from the dataset.** The message vector encodes which room the object of interest or the target receptacle is in, once either agent finds it. The state value encodes progress in the task (higher is better). In the first row, the language instruction is "Pick up the whipped cream", the corresponding message is $[0, 1, 0, 0]$, and the state value $\phi(s) = 2$. In the second row the instruction is "Place it on the bed", the corresponding message is $[0, 0, 0, 1]$, with state value $\phi(s) = 6$.

Drone view	Ground agent view
	
	

cooperation is required for efficiency; the drone with its larger field of view can explore the environment more effectively, while only the ground agent can interact with objects.

Environment We use VirtualHome [14], a Unity-based environment designed for embodied multi-agent collaborative tasks. VirtualHome consists of 7 different scenes; each scene contains multiple rooms. The environment allows initializing objects at different locations in the scene, which can be used to generate various configurations of object placements. The drone is simulated by an overhead camera attached to an invisible agent in VirtualHome. Examples of observations from the ground and drone agent are shown in Table I

Tasks Tasks in the environment involve placing a *graspable* object on a *receptacle* object. There are 45 graspable objects and 16 receptacle objects. For each task, a graspable object, initial receptacle object, and target receptacle object are randomly sampled. The graspable object is initialized at the initial receptacle object, and the goal is to move it to the target receptacle object. The task is described using natural language, such as "Put the ⟨graspable object⟩ on the ⟨receptacle object⟩."

State space VirtualHome provides both scene graph and visual representations. Hence, our benchmark consists of both — the scene graph representation that circumvents the object recognition problem (allows focus on multi-agent cooperation), with lower compute requirements, while the visual representation requires object recognition in addition to developing the multi-agent cooperation algorithms, and is closer to the real world. Each agent receives a local observation at every time step (humanoid: egocentric view of the environment and drone: top-down view of a part of the environment, depending on its current location). In the visual setting, the observations consist of RGB frames for each agent. In the scene graph setting, the observation consists of a graph where the nodes are all the objects present in the visual observation of the agent, and edges describe the relationships between them. For example a coffee mug placed on the dining table will be represented in the scene graph by the nodes "coffee mug" and "table" and the edge between them for the relationship "on".

Action space To make the setting amenable to reinforcement learning, the action space consists of high-level navigation and manipulation actions, as well as low-level navigation actions.

- High-level navigation actions: These are of the form `Goto [ROOM]`, where `ROOM` is one of the rooms in the scene (i.e. kitchen, bedroom, bathroom, livingroom). These actions are available to both the agents.
- High-level manipulation actions: Only available to the humanoid agent, and consist of `Pick` and `Place` operations. To keep the action space small, we do not require specifying the argument for these actions. Instead, if the agent executes the `Pick` action when the graspable object of interest is in its view, or the `Place` action when the receptacle object of interest is in its view (and the agent is holding the graspable object), the actions lead to picking up the target graspable object, and placing the object on the target receptacle object, respectively. Otherwise, the action fails, and results in no change to the environment.
- Low-level navigation actions: For the humanoid agent, these actions are `Move Forward`, `Turn Left`, and `Turn Right`, while for the drone agent, these actions are `Move Forward`, `Move Backward`, `Move Left`, and `Move Right`.
- The Stay action: Both agents also have a `Stay` action, which doesn't result in any movement of the agent or interaction with the environment.

Reward The reward for taking action a at state s is defined in terms of a potential function, $R(s, a, s') = \phi(s') - \phi(s)$, where the potential function $\phi(\cdot)$ is defined as follows:

$$\phi(s) = \begin{cases} 10, & \text{if X placed on Y} \\ 6, & \text{if X grasped, and Y visible to G} \\ 5, & \text{if X grasped, and Y visible to D} \\ 4, & \text{if X grasped, and Y not visible to either G or D} \\ 2, & \text{if X not grasped, and X visible to G} \\ 1, & \text{if X not grasped, and X visible to D} \\ 0, & \text{if X not grasped, and X not visible to either G or D} \end{cases}$$

where X is the object of interest, and Y is the target location.

Language Feedback In addition to the reward, the agents might receive natural language feedback from the environment when they perform a suboptimal action. For instance, if the target object is visible to the ground agent, and it does not pick it, the feedback may be "You should have picked up the glass instead of going to the livingroom."

B. Implementation Details

Trajectory Generation using Planner To create a dataset for offline training, we implement a planner, that given a task, finds a trajectory to complete the task, using privileged

information. For instance, if the object of interest is visible to the drone, the ground agent is directed to the location of the object. Using the planner, we generate 6,100 trajectories, which we divide into training, validation, and test splits (subsection III-C).

Language Data We generate 100 task descriptions using a single template, and 100 feedback language instructions using 2 templates. We use Amazon Mechanical Turk to obtain 1 paraphrase for each description and feedback item, from which we generate additional templates and extract synonyms for objects. The resulting natural language descriptions and feedback have 183 unique words, and a mean sentence length of 14.04 words. See Table II for example task descriptions/feedback.

TABLE II: **Task descriptions (top) and feedback (bottom).**

1.	Grab the washing scrub and keep it on the kitchencounter
2.	Place the wine bottle on top of the work table
3.	Take the lotion and keep it on top of the towel rack
1.	You should have placed the notes on the kitchentable rather than going to the bedroom
2.	You didn't have to go to the livingroom, you should have placed the sportsball on the kitchen counter instead
3.	You had to place the boardgame on the bed instead of moving forward

C. Evaluation Protocol

Splits The VirtualHome environment has 7 scenes. The positions of objects can be modified to create different configurations. We create 6000 tasks across scenes 1-5, which are split into 5,500 training, 250 validation-seen and 250 test-seen tasks. 50 tasks are created for scenes 6 and 7 each, which are used as validation-unseen and test-unseen respectively.

Evaluation Metrics We compare approaches based on two evaluation metrics – the success rate of completing tasks, and the episode length for successful completion. These two metrics can be used to compute the path-length-weighted (PLW) score [1] $p_s = sL^*/\max(L, L^*)$, where s is 1 if the task was successfully completed, and 0 otherwise, L is the number of step taken by the approach to complete the task, and L^* is the optimal number of steps to complete the task, which is estimated as the number of steps in the trajectory generated by the planner (subsection III-B). The final score of the algorithm is computed as the average path-length-weighted score across all tasks in the test-unseen split.

Input settings We experiment with 2 variants – scene graph representation and visual representation (subsection III-A), and present results for both the settings (section IV).

IV. Experiments

We benchmark several approaches on our proposed problem setting, including behavior cloning and several state-of-the-art multi-agent RL algorithms. Our model architectures are shown in Figure 1.

A. Feature Extractors

Visual Observation. The input image is passed through a pre-trained ResNet-18 network ([5]), and the feature vector from the pre-final layer is further projected to a 512-dimensional vector using a linear layer, which is used as the visual representation of the scene.

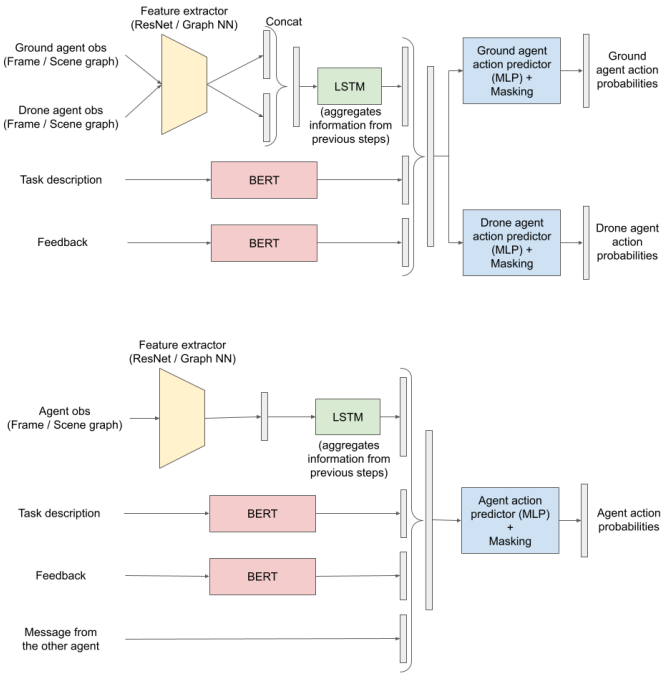


Fig. 1: **Models:** centralized (top) and decentralized(bottom).

Scene Graph Observation. The object class (e.g. bed, table, etc.) and the state (e.g. open, closed, etc.) of each node in the input scene graph is first encoded into vectors using an object class embedding layer and a state embedding layer respectively. These vectors are concatenated to obtain a vector representation for each node. We then apply a Graph Convolution Layer ([8]) to obtain contextualized embeddings for each node, which are aggregated using a mean-pooling operation to obtain the final vector representation of the input scene graph.

Message Passing A key component of a cooperative multi agent setup is for agents to communicate effectively. We propose a message passing method which allows agents to share information with each other. A message is a shared state between the two agents. It is a binary vector of length equal to the number of rooms. Each bit of the vector corresponding to the respective room is set to 1 if either of the agents identify the object of interest to be in that room. We use two such messages, one each for the object of interest and the target receptacle. The messages are used exclusively in the decentralized setups to allow agents to share information.

B. Algorithms

Behavior Cloning We use the (state, action) pairs in the dataset (subsection III-B) to train policy networks using supervised learning, for both centralized and decentralized scenarios. The decentralized scenario uses message passing between agents. For each scenario, we experiment with both visual and scene graph representations, where the states are encoded using the feature extractor architectures (subsection IV-A), and the networks are trained end-to-end using an Adam optimizer.

TABLE III: **Comparison of algorithms with visual observation.** (Ours) here refers to the algorithms with our message passing interface

Algorithm	Success rate				PLW score			
	Validation		Test		Validation		Test	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
BC; decentralized	37.21 ± 2.40	5.36 ± 1.50	42.02 ± 1.16	3.00 ± 1.00	30.34 ± 2.02	4.84 ± 0.78	35.43 ± 1.99	2.09 ± 0.81
BC; decentralized(ours)	40.89 ± 2.64	6.52 ± 1.81	48.54 ± 1.66	5.02 ± 0.82	37.42 ± 2.51	5.51 ± 0.82	40.11 ± 2.14	3.50 ± 1.31
BC; centralized	30.80 ± 2.00	4.35 ± 0.61	34.92 ± 2.96	4.00 ± 0.73	23.07 ± 2.55	4.19 ± 0.73	27.37 ± 3.06	2.29 ± 0.74
IQL	4.05 ± 5.02	2.01 ± 1.00	8.42 ± 8.88	2.33 ± 3.21	2.68 ± 2.88	1.79 ± 0.71	6.31 ± 6.91	1.73 ± 2.58
IQL(ours)	20.89 ± 2.01	7.56 ± 0.48	22.08 ± 4.02	**10.92 ± 2.11**	13.80 ± 2.55	5.54 ± 1.12	19.05 ± 2.55	6.10 ± 1.81
VDN	0.83 ± 0.41	0.67 ± 0.58	2.94 ± 3.26	0.00 ± 0.00	0.60 ± 0.58	0.51 ± 0.47	2.37 ± 3.03	0.00 ± 0.00
VDN(ours)	14.18 ± 1.72	2.55 ± 0.40	16.56 ± 2.08	5.21 ± 0.88	10.32 ± 1.14	3.76 ± 0.65	12.22 ± 1.02	3.00 ± 0.28
QMIX	17.73 ± 2.12	3.68 ± 0.56	17.96 ± 4.58	2.33 ± 2.31	14.09 ± 0.71	3.39 ± 1.06	15.21 ± 2.91	1.88 ± 2.29
QMIX(ours)	22.26 ± 2.12	8.01 ± 0.56	23.85 ± 4.58	**12.03 ± 2.31**	15.55 ± 2.71	6.34 ± 1.06	19.36 ± 2.91	**6.45 ± 2.29**
QTRAN	1.26 ± 0.73	2.01 ± 0.99	2.52 ± 2.22	0.33 ± 0.58	1.09 ± 0.91	1.45 ± 1.27	2.45 ± 1.92	0.33 ± 0.44
QTRAN(ours)	13.69 ± 1.22	7.07 ± 0.60	19.58 ± 1.66	8.00 ± 0.72	9.73 ± 0.89	5.94 ± 0.61	15.30 ± 1.25	4.82 ± 0.55

TABLE IV: **Comparison of algorithms with scene graph observation.** (Ours) here refers to the algorithms with our message passing interface

Algorithm	Success rate				PLW score			
	Validation		Test		Validation		Test	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
BC; decentralized	41.87 ± 3.94	11.47 ± 2.66	47.19 ± 0.55	13.67 ± 3.21	35.93 ± 2.95	9.51 ± 1.80	41.78 ± 1.33	11.36 ± 3.08
BC; decentralized(ours)	46.25 ± 3.94	21.08 ± 2.66	53.55 ± 0.55	18.04 ± 3.21	38.17 ± 2.95	15.06 ± 1.80	45.35 ± 1.33	13.72 ± 3.08
BC; centralized	40.08 ± 3.81	15.15 ± 1.13	33.61 ± 2.96	10.01 ± 1.93	32.73 ± 3.60	11.89 ± 0.39	26.77 ± 16.95	7.18 ± 3.11
IQL	14.18 ± 5.18	14.49 ± 2.59	14.53 ± 5.33	13.67 ± 5.13	9.91 ± 4.45	10.01 ± 1.91	10.80 ± 4.22	10.78 ± 3.12
IQL(ours)	24.16 ± 3.18	23.46 ± 2.59	24.89 ± 4.33	21.26 ± 4.13	19.94 ± 2.45	19.21 ± 1.91	19.73 ± 3.22	17.05 ± 3.12
VDN	16.85 ± 1.89	17.74 ± 3.61	19.03 ± 2.08	14.67 ± 4.73	11.36 ± 2.06	13.76 ± 2.37	13.68 ± 1.54	11.52 ± 2.87
VDN(ours)	34.87 ± 1.89	26.04 ± 3.61	36.70 ± 2.08	**27.51 ± 3.73**	23.37 ± 2.06	18.24 ± 2.37	27.86 ± 1.54	17.16 ± 2.87
QMIX	20.37 ± 2.90	14.78 ± 1.67	23.48 ± 2.23	18.67 ± 4.51	13.73 ± 3.56	12.84 ± 0.58	16.71 ± 3.50	14.67 ± 5.98
QMIX(ours)	39.25 ± 2.90	27.85 ± 1.67	38.13 ± 2.23	**28.95 ± 3.51**	25.36 ± 3.56	21.44 ± 0.58	28.68 ± 3.50	**20.05 ± 5.98**
QTRAN	3.04 ± 5.27	2.69 ± 3.83	2.94 ± 4.03	3.00 ± 5.20	2.57 ± 4.46	2.64 ± 3.74	2.57 ± 4.13	2.67 ± 4.63
QTRAN(ours)	12.42 ± 2.27	10.07 ± 3.83	11.45 ± 3.03	10.05 ± 2.20	10.57 ± 3.46	6.34 ± 3.74	9.12 ± 4.13	7.42 ± 1.63

Decentralized RL Algorithms Next, we benchmark several state-of-the-art decentralized RL algorithms, described below.
IQL: Independent Q-learning trains the Q-function of each agent on its history of local observations.
VDN: Value Decomposition Network decomposes the joint Q-function into a sum of Q-functions of the individual agents, and trains each agent on its own Q-function using DQN loss.
QMIX: extends VDN by relaxing the decomposition of the joint Q-function to be any monotonic function of the individual Q-functions, and trains as in VDN.
QTRAN: extends both VDN and QMIX by transforming the joint Q-function into an alternate that is expected to be easier to factorize. We base our implementation on [15], [2] and extend it for explicit message-passing (subsection IV-A).

C. Results

Our results (Table III and Table IV) show that:

- The message passing-based decentralized models (labelled (ours)) are significantly better than their non message-passing based versions. Decentralized behavior-cloning performs best in seen environments in PLW score, both in the visual and scene-graph representations. Both highlight the importance of effective communication
- Each of the RL models perform better on unseen test and val environments (best PLW 20.05) compared to behavior cloning methods (best PLW 13.72), demonstrating their ability to generalize to newer and unseen environments
- The visual observation (best test-unseen PLW score: 6.45) is significantly harder than scene graph observation (best test-unseen PLW score: 20.05). A potential reason for this could be the need for better representing visual features using a feature extractor trained on in-domain data.
- All the existing algorithms achieve a relatively low PLW score on the unseen splits, suggesting **CHMARL** could spur creation of new algorithms/models.

V. Conclusions

We proposed **CHMARL**, a new multimodal, multiagent, cooperative learning benchmark, with heterogeneous agents – a ground agent with an egocentric field of view that can interact with objects in the environment, and a drone agent with a larger field of view that cannot physically interact with the environment. We create tasks that require effective collaboration between agents. We introduce a simple message passing-based communication interface to allow efficient collaboration between agents, leading to significant performance gains over the non communicative baselines, highlighting the need for better communication between the agents to improve task success. We benchmark existing algorithms on the proposed problem; there is significant room for improvement in a multimodal setup to solve tasks effectively by developing new algorithms that leverage the strengths of each agent, and learn an efficient cooperative policy.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

[3] Swaminathan Gurumurthy, Akshat Agarwal, Vasu Sharma, and Katia P. Sycara. Mind your language: Learning visually grounded dialog in a multi-agent setting. *Adaptive Learning Agents (ALA) 2018*, 2018.

[4] Swaminathan Gurumurthy, Akshat Agarwal, Vasu Sharma, Mike Lewis, and Katia Sycara. Community regularization of visually-grounded dialog. *International Conference on Autonomous Agents and Multiagent Systems(AAMAS 2019)*, 2019.

[5] 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.

[6] Unnat Jain, Luca Weihs, Eric Kolve, Mohammad Rastegari, Svetlana Lazebnik, Ali Farhadi, Alexander G Schwing, and Aniruddha Kembhavi. Two body problem: Collaborative visual task completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6689–6699, 2019.

[7] Unnat Jain, Luca Weihs, Eric Kolve, Ali Farhadi, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *European Conference on Computer Vision*, pages 471–490. Springer, 2020.

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

[9] Andrey Kurenkov, Roberto Martín-Martín, Jeff Ichnowski, Ken Goldberg, and Silvio Savarese. Semantic and geometric modeling with neural message passing in 3d scene graphs for hierarchical mechanical search. *arXiv preprint arXiv:2012.04060*, 2020.

[10] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: multi-agent perception via communication graph grouping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4106–4115, 2020.

[11] Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6876–6883. IEEE, 2020.

[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[13] Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

[14] 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.

[15] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[16] Sinan Tan, Weilai Xiang, Huaping Liu, Di Guo, and Fuchun Sun. Multi-agent embodied question answering in interactive environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 663–678. Springer, 2020.

[17] Haiyang Wang, Wenguan Wang, Xizhou Zhu, Jifeng Dai, and Liwei Wang. Collaborative visual navigation. *arXiv preprint arXiv:2107.01151*, 2021.

[18] Fengda Zhu, Siyi Hu, Yi Zhang, Haodong Hong, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Main: A multi-agent indoor navigation benchmark for cooperative learning. 2021.