CHALET: Cornell House Agent Learning Environment

Claudia Yan[♡♢] **Dipendra Misra**[♢] **Andrew Bennett**[♢]
Aaron Walsman[♣] **Yonatan Bisk**[♣] **Yoav Artzi**[♢]

[♢] Cornell University [♡] City College of New York [♣] University of Washington

[♢]{*dkm, awbennett, yoav*}@cs.cornell.edu [♡]*cyan000@citymail.cuny.edu* [♣]{*awalsman, ybisk*}@cs.washington.edu

https://github.com/lil-lab/chalet

arXiv:1801.07357v2 [cs.AI] 16 Sep 2019

***Abstract*—We present CHALET, a 3D house simulator with support for navigation and manipulation. CHALET includes 58 rooms and 10 house configuration, and allows to easily create new house and room layouts. CHALET supports a range of common household activities, including moving objects, toggling appliances, and placing objects inside closeable containers. The environment and actions available are designed to create a challenging domain to train and evaluate autonomous agents, including for tasks that combine language, vision, and planning in a dynamic environment.**

I. Introduction

Training autonomous agents poses challenges that go beyond the common use of annotated data in supervised learning. The large set of states an agent may observe and the importance of agent behavior in identifying states for learning require interactive training environments, where the agent observes the outcome of its behavior and receives feedback. While physical environments easily satisfy these requirements, they are costly, difficult to replicate, hard to scale, and require complex robotic agents. These challenges are further exacerbated by the increased focus on neural network policies for agent behavior, which require significant amounts of training data [15, 17]. Recently, these challenges are addressed with simulated environments [15, 13, 2, 22, 12, 4, 11, 20, 18, 1, 5]. In this report, we introduce the Cornell House Agent Learning EnvironmenT (CHALET), an interactive house environment. CHALET supports navigation and manipulation of both objects and the environment. It is implemented using the Unity game development engine, and can be deployed on various platforms, including web environments for crowdsourcing.

II. Environment

CHALET includes 58 rooms organized into 10 houses. Figure 1 shows a sample of the rooms. The environment contains 150 object types (e.g., fridge, sofa, plate). 71 types of objects can be manipulated: 60 picked and placed (e.g., plates and towels), 6 opened and closed (e.g., dishwashers and cabinets), and 5 change their state (e.g., opening or closing a faucet). Object types are used with different textures to generate 330 different objects. On average, each room includes 30 objects. Rooms often contain multiple objects of the same kind. For example, kitchens contain many plates and glasses, and bathrooms contain multiple towels. Objects that can be opened and closed are container objects, and can contain other objects. For example, opening a dishwasher exposes a set of racks, and pulling a rack out allows the agent access to the objects on that rack. The agent can also put an object on the rack, close the dishwasher, and open it later to retrieve the object. Figure 2 shows example object manipulations. The environment supports simple physics, including collision-detection and gravity.

The agent in CHALET observes the environment from a first-person perspective. At any given time, the agent observes only what is in front of it. The agent position is parameterized by two coordinates for its location and two coordinates for the orientation of its view. Changing the agent location is done in the direction of its orientation (i.e., first-person coordinate system). Whether the agent looks up or down does not influence location changes. All agent actions are continuous, but can be discretized to pre-defined precision by specifying the quantities of change for each step. Table I describes the agent actions.

CHALET provides a rich testbed for language, exploration, planning, and modeling challenges. We design rooms to often include many objects of the same types. Instructions or questions that refer to a specific objet must then use spatial relations and object properties to identify the exact instance. For example, to pick up a specific towel in a bathroom, the agent is likely to be given an instruction such as *pick up the yellow towel left of the sink*. In contrast, in an environment with a single object of each type, it would have been sufficient to ask to *pick up the towel*. The ability to open and close containers also creates several interesting challenges. Given an instruction, such as *put the glass from the cupboard on the table*, it is insufficient for an agent to simply align the word *glass* to an observed object. Instead, it must resolve the noun phrase *the cupboard* and the relation *from* to understand it must look for a glass in a specific location. Simply resolving the target object (*glass*) is insufficient. If multiple cupboards are available, the agent must also explore the different cupboards to find the one containing a glass. This requires both deciding

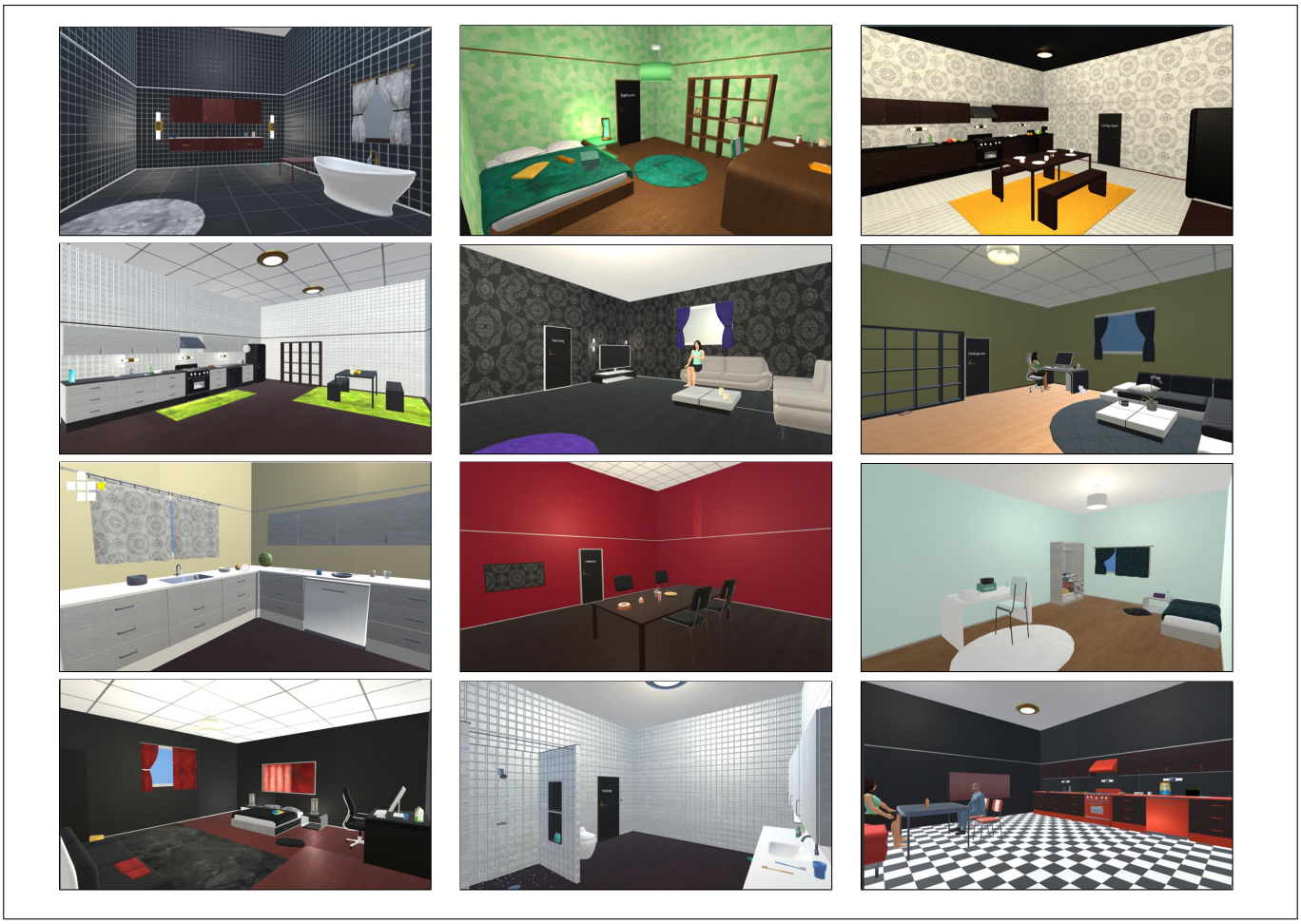

Fig. 1. Screenshots of various rooms from CHALET. Each house includes 4-7 rooms of various kinds, including bathrooms, bedrooms, and kitchens.

on an exploration policy, and planning a complex sequence of actions. Finally, the agent perspective requires models that support access to previous observations or a representation of them (i.e., memory) to overcome the partial observability challenge.

III. Evaluation in CHALET

Evaluating agent performance in CHALET is done by comparing the agent behavior to an annotated demonstration. A demonstration is a sequence of states and actions $\tau = \langle s_0, a_1, s_1, a_2, \cdots a_m, s_m \rangle$, where s_i is state and a_i is the action taken at that state. The start state of the demonstration is s_0 and the final state is s_m. A state s_i contains information about the position, the orientation, and the interaction state of every object in the house, including the agent. We use two metrics for evaluating navigation errors and manipulation accurcy. Navigation error is the sum of euclidean distances in each room the agent must travel to reach the goal. For the room the agent is currently located in, we take the euclidean distance between the agent and the door to the next room. For each intermediate room, we take the euclidean distance between the door where the agent enters and the door to the following room. For the room containing the goal (i.e., the agent position in s_m), we measure the distance between the entry door to the goal. In each room, the distance measured is a straight line. We compute manipulation accuracy by extracting an ordered list of interaction actions from the annotated demonstration τ. An interaction action may be picking an object, placing an object at a specific location, and changing the interaction state of an object (e.g., opening a drawer). All objects are uniquely identified. The manipulation accuracy is the F1-score computed for the list of actions extracted from the agent execution against the reference list of actions extracted from τ. We consider placing an object within a radius of 1.0m of the specified position in the same room as equivalent.

IV. Implementation Details

CHALET is implemented in Unity 3D,[1] a professional game development engine.[2] The environment logic is written in the C# scripting language, which supports high-level object-oriented programming constructs and is tightly integrated with

[1] https://unity3d.com/

[2] The community version of Unity, which was used to develop CHALET is publicly for education purposes.

Action	Description
move-forward	Change the agent location in the direction of its current orientation
move-back	Change the agent location in the direction opposite to its current orientation
strafe-right	Change the agent location in the direction of 90° to its current orientation
strafe-left	Change the agent location in the direction of 270° to its current orientation
look-left	Change the agent orientation to left
look-right	Change the agent orientation to right
look-up	Change the agent orientation up (when engaged with a container, change container towards closure)
look-down	Change the agent orientation down (when engaged with a container, change container towards open)
interact	Engage the container at the current orientation, pick the object at the current orientation, drop the object currently held, toggle state of object at current orientation (e.g., toggle TV power)

TABLE I
THE ACTIONS AVAILABLE TO THE AGENT IN CHALET.

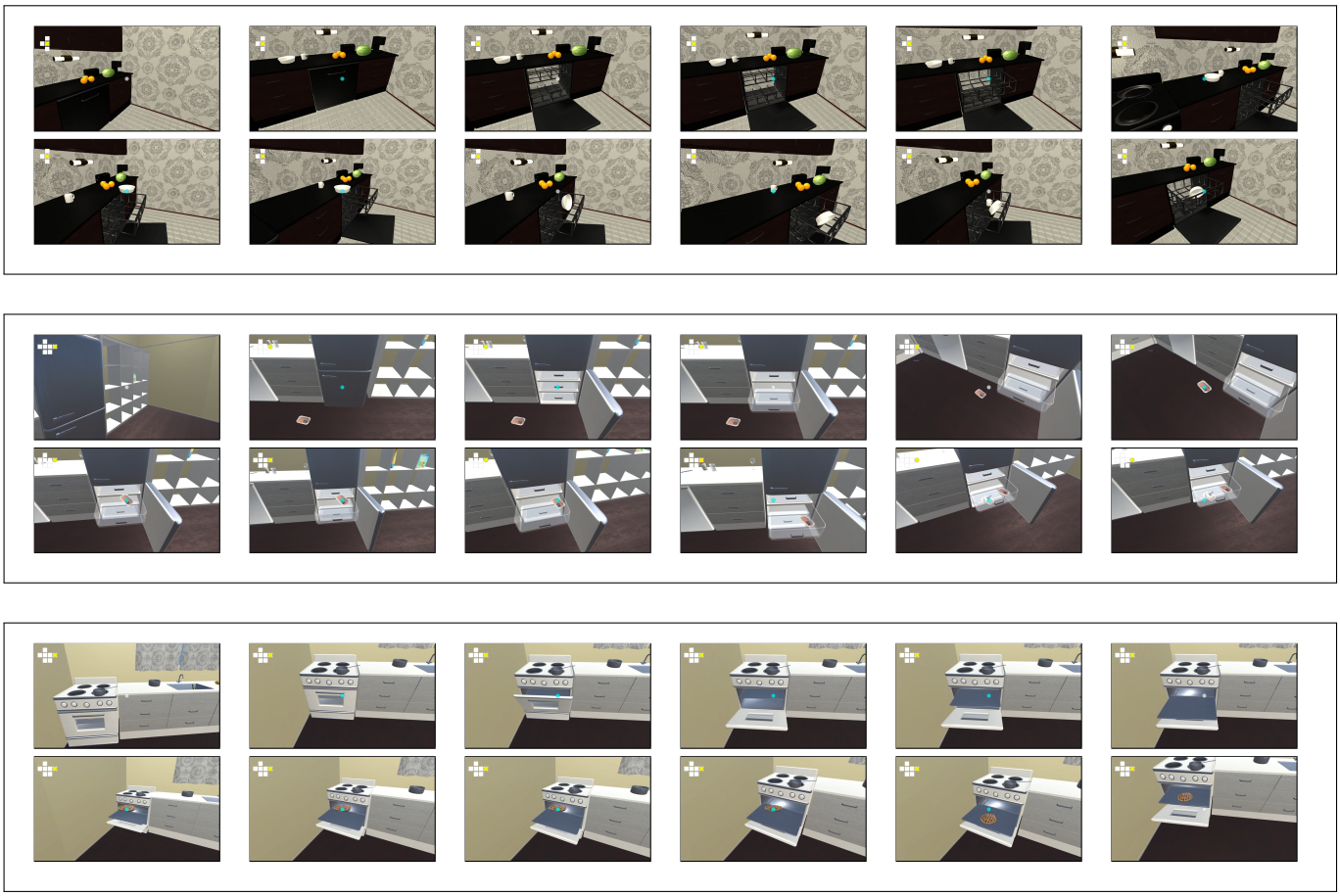

Fig. 2. Sampled observations from three sequences of object manipulation, from top to bottom: loading the dishwasher, placing a food item in the freezer, and placing a pie in the oven.

the Unity engine. Using Unity provides several advantages. CHALET can be easily compiled for different platforms, including Linux, MacOS, Windows, Android, iOS, and WebGL. Unity also provides a built in physics engine, and supports integration with augmented- and virtual-reality devices. Extending CHALET with new objects from the Unity Asset Store[3] is trivial.

CHALET supports three modes of operation:

- `standalone`: actions are provided using keyboard and mouse input. The generated trajectory is saved to a file. This model is used for crowdsourcing using a WebGL build.
- `simulator`: actions are read from a saved file and executed in sequence. This mode allows replaying previously recorded trajectories, for example during crowdsourcing.
- `client`: a separate process provides actions, and the framework returns the agent observations and information about the environment as required. Communication is

[3]https://www.assetstore.unity3d.com/en/

done over sockets. This mode enables interaction with machine learning frameworks.

The framework provides a simple API to compute reward and feedback signals, as required for learning, and provide information about the environment, including the position and state of objects. CHALET also provides programmatic generation of rich scenarios by adding, removing, and replacing objects during runtime without the use of Unity or reloading the simulator. To enable this, each room is annotated with a set of surface locations where items may be placed. Placing an objects requires specifying its type and orientation, and the target surface and coordinates.

V. Related Environments

Table II compares CHALET to existing simulators. Savva et al. [18], Wu et al. [20], and Beattie et al. [2] provide similar observations to CHALET in navigation-only environments. In contrast, CHALET emphasizes manipulation of both objects and the environment to support complex tasks. Anderson et al. [1] use real images with a discrete state space for navigation. While CHALET includes 3D rendered environments, it provides a continuous environment with a variety of actions. The most related environments to ours are HoME [5] and AI2-Thor [7], both provide 3D rendered houses with object manipulation. Unlike HoME, which only supports moving objects, CHALET enables toggling the state of objects and changing the environment by modifying containers. In contrast to Thor, CHALET supports moving between rooms in complete houses, while the current version of Thor supports a single room.

There is also significant work on using simulators for other domains. Atari [15], OpenAI Gym [4], Project Malmo [11], Minecraft [16], Gazebo [21], Viz Doom [12] are commonly used for testing reinforcement learning algorithms. Simulators have also been used to evaluate natural language instruction following [14, 3, 13, 9, 8] and question answering [7, 6, 10, 19]. The manipulation features and partial observability challenges of CHALET provide a more realistic testbed for studying language, including for instruction following, visual reasoning, and question answering.

Acknowledgments

This work was supported by NSF under Grant No. 1750499, the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program, and the Women in Technology and Entrepreneurship in New York (WiTNY) initiative. We also thank Porrith Suong for help with Unity3D development.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *arXiv preprint arXiv:1711.07280*, 2017.

[2] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[3] Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural language communication with robots. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–761, San Diego, CA, June 2016.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[5] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub, Jean Rouat, Hugo Larochelle, and Aaron C. Courville. HoME: a Household Multimodal Environment. *arXiv preprint arXiv:1711.11017*, 2017.

[6] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. *arXiv preprint arXiv:1711.11543*, 2017.

[7] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual Question Answering in Interactive Environments. *arXiv preprint arXiv:1712.03316*, 2017.

[8] Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded Language Learning in a Simulated 3D World. *arXiv preprint arXiv:1706.06551*, 2017.

[9] Michaela Jänner, Karthik Narasimhan, and Regina Barzilay. Representation learning for grounded spatial reasoning. *CoRR*, abs/1707.03938, 2017.

[10] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016.

[11] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *International Joint Conferences on Artificial Intelligence*, pages 4246–4247, 2016.

[12] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaskowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. *arXiv preprint arXiv:1605.02097*, 2016.

[13] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2017.

[14] Kumar Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd*

Software	Type	Number of Environments	Navigation	Manipulation
MINOS [18]	Simulated	45K houses + variations	Yes	No
House3D [20]	Simulated	Thousand houses	Yes	No
AI2-Thor [22, 7]	Simulated	120 rooms	Yes	Yes
Matterport3D [1]	Real Images	90 houses	Yes	No
HoME [5]	Simulated	45000 houses	Yes	Yes
DeepMind Lab [2]	Simulated	Few (procedural)	Yes	No
CHALET	Simulated	58 rooms and 10 default houses	Yes	Yes

TABLE II
COMPARISON OF CHALET WITH OTHER 3D HOUSE SIMULATORS

Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015. doi: 10.3115/v1/P15-1096.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2013.

[16] Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. In *Proceedings of the International Conference on Machine Learning*, 2016.

[17] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[18] Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments. *arXiv preprint arXiv:1712.03931v1*, 2017.

[19] Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017.

[20] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building Generalizable Agents with a Realistic and Rich 3D Environment. *arXiv preprint arXiv:1801.02209v1*, 2017.

[21] Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.

[22] 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 *IEEE International Conference on Robotics and Automation*, 2017.