Grounding Artificial General Intelligence with Robotics: The PetitCat Project

Schneider, Howard[1][0000-0001-8052-6448] and Georgeon, Olivier L.[2][0000-0003-4883-8702]

[1] Sheppard Clinic North, Vaughan, Canada
hschneidermd@alum.mit.edu
[2] UR CONFLUENCE : Sciences et Humanités, UCLy, Lyon, France
ogeorgeon@univ-catholyon.fr

Abstract. The symbol grounding problem is a well-known philosophical problem. In the engineering of Artificial General Intelligence (AGI) systems, grounding becomes essential for interacting with a real-world environment. Grounding an AGI system or Brain-Inspired Cognitive Architecture (BICA) via mobile robotics is described. The PetitCat project includes an open-source software interface between the Python code of an AGI project or BICA project and a mobile robotics device. The interface allows real-time communication between the Python code on a desktop or server via wireless network to the lower-level C/C++ code used to control a robotic device. The PetitCat project is targeted at studying symbol grounding and autonomous bottom-up interactive learning in the framework of theories of enaction, active inference, predictive coding, and developmental and constructivist learning.
Keywords: Grounding, Artificial General Intelligence (AGI), Robotics, Brain-Inspired Cognitive Architecture (BICA)

1 Introduction: The Symbol Grounding Problem

This paper is a short technical communication describing the PetitCat Project. The project should be thought of as an open-source tool available to other cognitive and artificial intelligence researchers which can allow grounding, experimentation, and utilization of Python-based Artificial General Intelligence (AGI) projects, Brain-Inspired Cognitive Architecture (BICA) projects, and other cognitive projects, with real-world environments.

Barsalou [1] asks how abstract symbols of computing systems can actually understand the external world and produce cognition unlike people who actually experience the world. Harnad [2] gives a more direct example, asking how a person with no related knowledge of the Chinese language can learn it and understand it via a Chinese-Chinese dictionary. The person trying to learn Chinese via such a dictionary ends up going from one string of symbols without any meaning, to another set of such symbols. The issue is that these symbols (i.e., characters in the dictionary) are "grounded" in other Chinese symbols which cannot provide any meaning to the person who has no related knowledge of the language. This analogy holds for any other language if the user has no related knowledge of the language (i.e., does not have knowledge of another

language with some overlap). Similarly, this analogy holds for an artificial computing system. Such a system cannot truly ascribe meaning to its internal symbols.

The symbol grounding problem is effectively about how symbols get their meaning. Symbols in an artificial computing system are effectively obtained from following various sets of rules/procedures which really are based on what Harnad considers the symbols' "shapes" rather than their meaning. The rich relationships of symbols and their meaning which give much value are often missing in artificial computing systems. For example, an ungrounded AI or BICA agent may know all sorts of words and descriptions of kitchens, but when it actually has to explain how to interact with the real world in order to obtain a food item from a cupboard it may have a hard time doing so, i.e., where to stay in relation to the cupboard door, where to put one's hand, and so on, information those of us who interact with the real world have learned.

Large language models (LLM) have improved at present to the point where they are considered to have AGI-like properties. For example, the LLM ChatGPT without any specialized training was able to achieve passing grades in the United States medical licensing examinations [3]. However, what meaning does ChatGPT actually have about the symbols (which may even be just parts of words, i.e., tokens) it contains or produces? Despite the vast richness of an LLM's learning dataset, they have essentially been trained on text and have little actual awareness (i.e., sensory and motor interactions) of the external, real world. As Pavlick notes, it is still an open debate how this actual grounding affects the performance of such complex models [4].

Harnad's solution to the grounding problem is to ground symbols with their real-world sensations, interactions and the objects and actions they are linked to [2, 5]. With respect to the grounding of more abstract concepts, Reinboth and Farkaš note that effectively abstract concepts can also be grounded in experience, i.e., via the body's sensorimotor interactions with the real world [6].

In order to engineer more effective AGI or BICA systems, there will be the need to ground these systems. In keeping with Harnad's solution, the PetitCat open-source project provides a software interface between the Python code of an AGI or BICA project running on a personal computer (or other computing system) and a robotics device that can interact with the real world.

2 The PetitCat Robotics Platform

The PetitCat free open-source software is maintained at Lyon Catholic University and can be downloaded from GitHub [7]. This software includes a Python layer to run on the personal computer (PC) where the AGI or BICA project is implemented, and a C/C++ layer (compiled by the Arduino development system [8]) that can run on various microcontrollers controlling a robotics device. The Python code sends motor instructions as well as queries to the robot system via a wireless network. In the other direction, the PetitCat robot sends sensory information and outcomes of instructions back to the PC. Its components are kept all together in a platform-like environment.

The Robot Operating System 2 (ROS2) is a popular open-source robotics framework that also provides hardware abstraction and a low-level code interface [9]. Exploring

the symbol grounding problem, as well as other work, however, can be done with simpler, less expensive equipment with the PetitCat project.

Another factor in the development of the PetitCat project was to have a robotics platform that is easily customizable by adding bespoke sensors and effectors. The choice of using low-computation sensors as opposed to intensive-computation sensors such as a camera or lidar is justified by the chosen sensorimotor approach that focuses on patterns of interaction implemented in the robot through control loops involving continuous control and sensory feedback. The choice of a microcontroller-based Arduino board (supported by the PetitCat project) rather than a computer-based operating system (often required by more complex robotic software frameworks) to control the robotic device was made for simplicity, adaptivity, learnability, and affordability.

Several Python projects also provide tools to control autonomous mobile robots. PythonRobotics [10] implements functionalities such as localization, mapping, and planning based on widely used algorithms. RatLab [11] and CoBeL-RL [12] provide brain-inspired tools to generate spatial behaviors in simulated robots. In contrast, PetitCat integrates the Arduino and Python layers in a single project that offers a tight coupling between action and sensory feedback, an advantage in the study sensorimotor grounding in the physical world.

3 Hardware Implementation

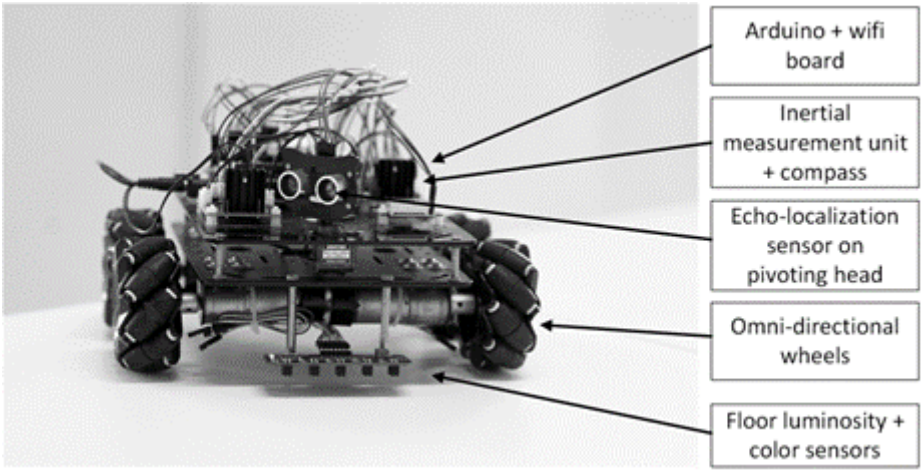

Fig. 1. The PetitCat robotics car in more detail.

Fig. 2. PetitCat platform allowing Python-based code to control the robot. The cognitive architecture selects behaviors and sends commands to the robot through User Datagram Protocol (UDP) packets. The robot enacts the behavior and sends the outcome back to the PC.

In the hardware implementation, an ordinary Microsoft Windows-based personal computer running Python version 3.11 is used. The robot is based on the robotics car manufactured by Osoyoo/Pinetree Electronics [13] shown in Figure 1. It includes an Arduino board with an ATmega2560 microcontroller, a Wi-Fi shield, a bar of five infrared-reflective sensors directed to the ground, and an ultrasonic transducer (for distance measurements) mounted on a pivoting servomechanism head. We added a 9-axis inertial measurement unit, a color sensor directed to the ground, and an RGB LED serving as a visual indicator of the robot's internal state. Figure 2 shows the PC controlling the robot in an example experimental setup.

This Arduino board as well as the actual robotics car provides an inexpensive basis on which researchers can configure their PetitCat implementation according to their needs. The capacity to express emotions through body and head movements and a color indicator makes PetitCat also suitable for studying emotional robotics [14].

4 Usage

The `PetitCat.ino` is C/C++ code compiled by the Arduino IDE [8] into machine code that runs directly on a variety of Arduino and non-Arduino microcontrollers. It contains many routines for a robotics car and can be used as is by the developer. However, some researchers may want to modify the C/C++ code, which again can easily be modified.

The `PetitCatMain.py` module integrates with any Python AGI, BICA, or other cognitive architecture code. The developer can use it as is or can modify it—the level of complexity is significantly simpler than a more elaborate framework such as the Robot Operating System [9].

The Python project includes an internal simulator of the robot interacting with simple

objects (Fig. 3). The cognitive architecture can use the simulator to compute predictions of outcomes of different possible behaviors before selecting the behavior. As inspired by Ullman and colleagues [15], the simulator plays the role of the "game engine in the head" that the cognitive architecture can use to reflect on the physical world. On a more pragmatic level, the developer using the PetitCat project can use the simulator to help debug a particular project implementation.

5 Implementations and Discussion

Harnad's solution to the grounding problem is to ground symbols with their real-world sensations and interactions [2, 5]. The PetitCat project addresses this requirement by providing a software interface between the Python code of an AGI or BICA project running on a personal computer (or other computing system) and a robotics device, thus allowing sensorimotor interactions with the real world based on control loops of interactions.

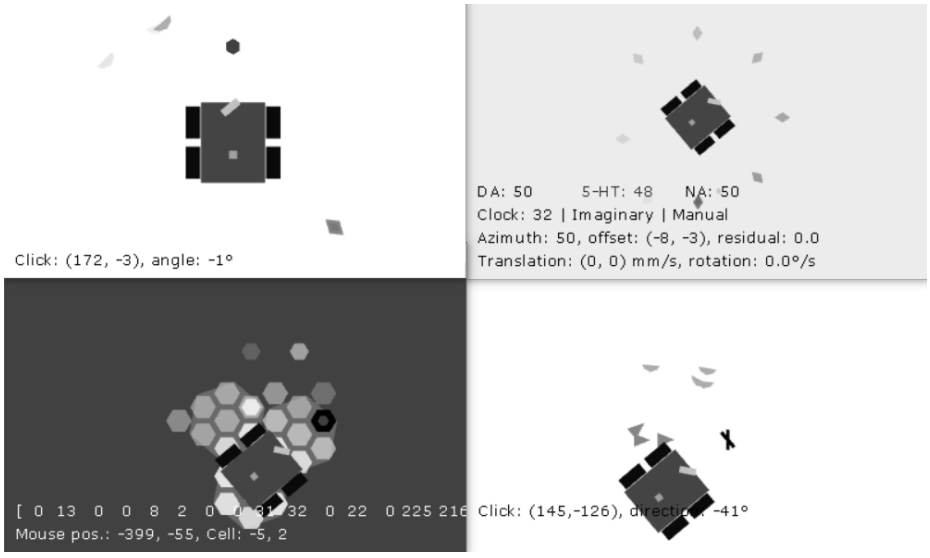

Fig. 3. Screenshots of the PetitCat Internal Simulator. Top left: egocentric memory. Hexagon: the robot's focus of attention. Darker half-circles: echoes. Top right: body memory. Diamonds: compass points over time as the robot turns around. Bottom left: allocentric memory. Dark hexagon: black mark on the floor. Bottom-right: object-centric memory. Darker segments: interactions with the black mark on the floor.

PetitCat has been used to demonstrate artificial enactive inference in the three-dimensional world [16]. As opposed to utilizing detached symbols about the external world, in enactive inference, an agent learns about the world through a continuous loop of action and perception. The agent develops a grounded understanding of the external

world in this manner. In another implementation, Gay and colleagues are using PetitCat to study autonomous bio-inspired spatial orientation and navigation [17].

The PetitCat project is also currently being interfaced with the Causal Cognitive Architecture (CCA) [18]. This architecture possesses abstract navigation planning procedures but does not have the experience to navigate in the real world to a series of waypoints.

Future work on the PetitCat project specifically includes further developing the sensorimotor layer related to the internal simulator to facilitate the control of behaviors in time and space from a more abstract layer in which third-party cognitive architectures are implemented. More generally, work is ongoing in enhancing the bidirectional motor and sensory interface between the AGI or BICA Python program and the robotic system it is interfacing with. Researchers and developers are welcome to contribute to this open-source project [7].

Acknowledgments: OLG: This work benefited from the contribution of Karim Assi.

References

1. Barsalou L. W.: Challenges and Opportunities for Grounding Cognition. Journal of Cognition, **3**(1),31(2020). doi.org/10.5334/joc.116
2. Harnad, S.: The symbol grounding problem. Physica D, **42** (1-3), 335-346 (1990). doi:10.1016/0167-2789(90)90087-6
3. Kung, Tiffany H., et al.: Performance of ChatGPT on USMLE. medRxiv (2022). https://doi.org/10.1101/2022.12.19.22283643
4. Pavlick, E.: Symbols and grounding in large language models. Philosophical Transactions of the Royal Society A, **381**(2251), 20220041 (2023).
5. Harnad, S.: Computation Is Just Interpretable Symbol Manipulation: Cognition Isn't. Minds and Machines **4**:379-390 (1994).
6. Reinboth, T., Farkaš, I.: Ultimate Grounding of Abstract Concepts: A Graded Account. Journal of Cognition. **5**(10):21 (2022). doi.org/10.5334/joc.214
7. Georgeon, O.: PetitCat. GitHub (2024). Retrieved April 11, 2024, from https://github.com/UCLy/INIT2
8. Arduino: What is Arduino? (2024). Retrieved April 11, 2024, from https://www.arduino.cc/
9. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot Operating System 2: Design, architecture, and uses in the wild. Science robotics, **7**(66), eabm6074 (2022).
10. Sakai, A., Ingram, D., Dinius, J., Chawla, K., Raffin, A., Paques, A. : PythonRobotics: A Python code collection of robotics algorithms. arXiv:1808.10703 (2018).
11. Schoenfeld, F., Wiskott, L.: RatLab: An easy to use tool for place code simulations. Frontiers in Computational Neuroscience, **7** (2013). doi: 10.3389/fncom.2013.00104
12. Diekmann, N., Vijayabaskaran, S., Zeng, X., Kappel, D., Menezes, M. C., Cheng, S.: CoBeL-RL: A neuroscience-oriented simulation framework for complex behavior and learning. Frontiers in Neuroinformatics, **17** (2023). doi: 10.3389/fninf.2023.1134405
13. Osoyoo: M2.0 metal chassis mecanum wheel robotic, retrieved April 11, 2024, from https://osoyoo.com/2022/07/05/v2-metal-chassis-mecanum-wheel-robotic-for-arduino
14. Chebotareva, E., Safin, R., Shafikov, A., …, Talanov, M.: Emotional social robot "emotico". In: 2019 12th International Conference on Developments in eSystems Engineering (DeSE). pp. 247-252. IEEE (2019). https://doi.org/10.1109/DeSE.2019.00054

15. Ullman, T.D., Spelke, E., Battaglia, P., Tenenbaum, J.B.: Mind games: Game engines as an architecture for intuitive physics. Trends in cognitive sciences **21**(9): 649-665 (2017).
16. Georgeon, O. L., Lurie, D., Robertson, P.: Artificial enactive inference in three-dimensional world. Cog Sys Research, 101234 (2024). doi:10.1016/j.cogsys.2024.101234
17. Gay S.: A platform for studying navigation models, (2023). Retrieved April 11, 2024, from https://gaysimon.github.io/robot/robot_navigation_en.html
18. Schneider, H.: The Emergence of an Enhanced Intelligence in a Brain-Inspired Cognitive Architecture, Frontiers in Computational Neuroscience, **18**, 1367712 (2024). doi: 10.3389/fncom.2024.1367712