LLM2Swarm: Robot Swarms that Responsively Reason, Plan, and Collaborate through LLMs

Volker Strobel
IRIDIA
Université Libre de Bruxellles
Brussels, Belgium
volker.strobel@ulb.be

Marco Dorigo
IRIDIA
Université Libre de Bruxellles
Brussels, Belgium
mdorigo@ulb.ac.be

Mario Fritz
CISPA Helmholtz Center for Information Security
St. Ingbert, Germany
fritz@cispa.de

Abstract

Robot swarms are composed of many simple robots that communicate and collaborate to fulfill complex tasks. Robot controllers usually need to be specified by experts on a case-by-case basis via programming code. This process is time-consuming, prone to errors, and unable to take into account all situations that may be encountered during deployment. On the other hand, recent Large Language Models (LLMs) have demonstrated reasoning and planning capabilities, introduced new ways to interact with and program machines, and incorporate both domain-specific and commonsense knowledge. Hence, we propose to address the aforementioned challenges by integrating LLMs with robot swarms and show the potential in proofs of concept (showcases). For this integration, we explore two approaches. The first approach is 'indirect integration,' where LLMs are used to synthesize and validate the robot controllers. This approach may reduce development time and human error before deployment. Moreover, during deployment, it could be used for on-the-fly creation of new robot behaviors. The second approach is 'direct integration,' where each robot locally executes a separate LLM instance during deployment for robot-robot collaboration and human-swarm interaction. These local LLM instances enable each robot to reason, plan, and collaborate using natural language, as demonstrated in our showcases where the robots are able to detect a variety of anomalies, without prior information about the nature of these anomalies. To enable further research on our mainly conceptual contribution, we release the software and videos for our LLM2Swarm system: https://github.com/Pold87/LLM2Swarm.

1 Introduction

Robot swarms consist of many simple robots that accomplish complex tasks by collaborating with each other [11; 20]. They are characterized by the absence of a central control unit, which makes them more scalable and flexible than other multi-robot systems. Thanks to their decentralized control and redundancy, robot swarms can potentially continue functioning even if individual robots fail [42]. These features could make robot swarms ideal for applications where fast reactions to unforeseen events are required, such as environmental monitoring and disaster response [12; 43].

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

In robot swarms, communication and interaction protocols between robots usually need to be specified by experts on a case-by-case basis via programming code [28]: Given a specific task, a human designer needs to develop and implement the corresponding program logic. The process of writing such code is often time-consuming and prone to errors. Even though there are attempts to automate this process [15; 17; 40; 39; 14], these attempts usually still require expert knowledge at design time and the controllers cannot react to unforeseen events at deployment time.

Recent advances in Large Language Models (LLMs) have led to new possibilities and proposals for the integration of AI into different technologies, such as virtual assistants and search engines [18]. In particular, there have also been proposals to integrate LLMs into multi-robot systems [5; 32]. With this paper, we propose to integrate LLMs into a specific kind of multi-robot system: robot swarms.

We provide the first systematic exploration of the potential of LLM-enabled robot swarms (we call our system LLM2Swarm). We demonstrate the resulting opportunities in a series of proofs of concept (showcases), implemented in the ARGoS robot swarm simulator [34], and preliminary hardware tests using a Raspberry Pi 5. For this exploration, we propose two primary approaches of integrating LLMs into robot swarms.

The first approach is "indirect integration," where LLMs are used to synthesize and validate the robot controllers (before and/or during deployment). This approach may save development time for the designer by automating parts of the design process. In addition, it could dramatically simplify the process of writing controllers, as no programming knowledge is necessary. The approach ensures that robot swarms *responsively* react, as they execute their classical (synthesized) control software, instead of waiting for LLM-generated responses.

The second approach for integrating LLMs into robot swarms is "direct integration," where each robot locally executes a separate LLM instance during deployment. With this online integration, the local LLM instances enable each robot to reason, plan, and collaborate using natural language. With these new and improved capabilities, we aim to enhance robot swarms' adaptivity, efficiency, and intelligence, making them more capable of handling complex tasks in unpredictable environments, both for robot-robot collaboration and human-swarm interaction.

In summary, our contributions are as follows:

1. We introduce LLM2Swarm for integrating LLMs into robot swarms and explore the resulting opportunities and challenges.
2. We provide (video) showcases for controller synthesis, robot-robot collaboration, and human-swarm interaction in simulation.
3. We release our LLM2Swarm setup as open-source software. Videos and code can be downloaded at `https://github.com/Pold87/LLM2Swarm`.

2 Related work

Our work is related to three topics: controller synthesis, LLMs as agents, and LLMs in robotics. In the following, we briefly present related work from these fields and highlight how our vision differs.

Controller synthesis. There are several related works on controller synthesis, both from the field of swarm robotics (e.g., AutoMoDe [15; 17], evolutionary approaches [40], neural network controllers [39]) and from more theoretical fields, such as computer aided verification [14]. LLMs have also already been used for the controller synthesis of single robots [41]. However, these approaches still rely on expert knowledge (e.g., formal specification languages) during design time, cannot flexibly react to unforeseen situations during deployment, or do not support multiple agents or robots.

LLMs in multi-agent systems. LLMs are increasingly being integrated into other systems to enhance their capabilities (e.g., as part of search engines) [18]. In the recent past, there has been an increasing interest in letting *multiple* LLMs interact and discuss [1; 6]. An overview of the integration of LLMs into multi-agent systems for various tasks has been given by Guo et al. [19]. In contrast to these works, our target platforms are physical robots. The previous works did not take into account how to synthesize robot controllers, how to deal with mobility, sensor information, and actuators, and how to map sensors and actuators to LLM inputs and responses.

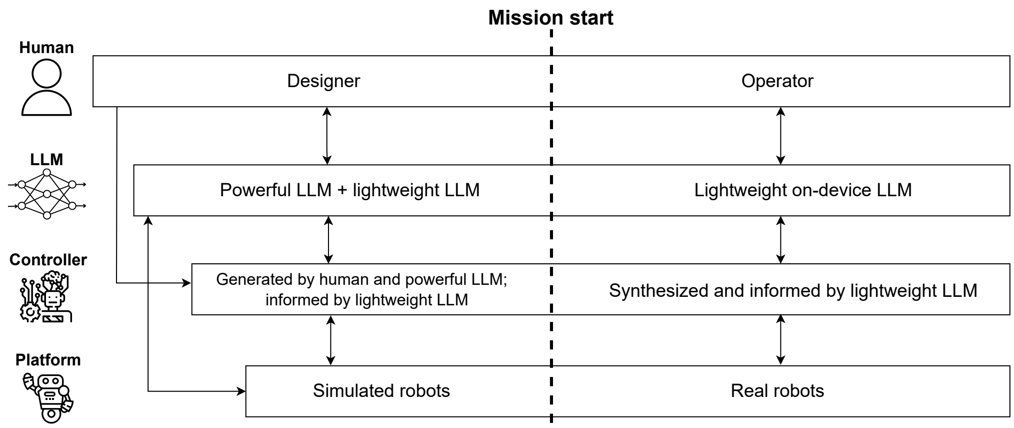

Figure 1: **Overview of LLM2Swarm – LLM-enabled robot swarms**. LLM2Swarm involves four key system components: humans, LLMs, controllers, and platforms. *Before mission start*: a human designer uses both manual design and LLM2Swarm's controller synthesis module (which prompts a powerful LLM) to generate a robot controller. This controller is executed in simulation, and uses one lightweight LLM per robot to simulate on-device execution of LLM2Swarm's direct integration module. *After mission start*: a human operator interacts with the real robots' lightweight on-device LLMs to instruct the swarm and to receive information about the swarm state. These LLMs also interact with other robots' controllers in order to reason, plan, and collaborate. In addition, the lightweight LLMs can synthesize new robot controllers on-the-fly during the mission.

LLMs in robotics. Integrating LLMs into other systems has also led to several new applications in robotics [26], for example for task planning [9], centralized motion planning [24], and human-robot interaction [41]. As our target platforms are robot swarms, in the following, we are considering works that use a separate LLM instance for each embodied agent. Chen et al. [5] perform motion planning (e.g., to move a box to a target), with up to 32 simulated robots (in 2D grid environments and using robotic arms). Mandi et al. [32] perform motion planning (of robotic arms) with up to three robots. Zhang et al. [44] study how two virtual humans can communicate via LLMs. These research efforts, however, have one or more of the following limitations: a low number of robots/agents, lack of decentralization, no consideration of mobile robots with physics-based dynamics, no consideration of typical swarm robotics tasks, or no consideration of hardware limitations; therefore, their applicability to swarm robotics is limited.

3 LLM-enabled robot swarms: LLM2Swarm

To give an overview of how our LLM2Swarm system yields LLM-enabled robot swarms, we illustrate the proposed temporal flow and interactions in Figure 1. LLM2Swarm provides two main modules: an *indirect integration module* for automating controller design (both before and during deployment) and a *direct integration module* for enhancing robots' reasoning, planning, and collaboration capabilities during deployment.

3.1 Indirect Integration

We propose to use LLMs 'indirectly' for controller synthesis, i.e., automated creation of code before or during deployment. For this indirect integration, LLM2Swarm provides a module (see Figure 2) that translates a description of a robot controller in natural language to an actual robot controller. Before the mission start, a human designer generates a robot controller draft using manual design. Parts of these controllers can be synthesized by LLM2Swarm, using a powerful LLM (LLM2Swarm supports any model provided by OpenAI's API or by Ollama for local execution). Also during deployment, this LLM2Swarm module can lead to increased autonomy of a robot swarm—as missing controller parts can be synthesized on the fly by on-board LLMs. Based on our experiments, we propose the following three-step process to automate controller synthesis:

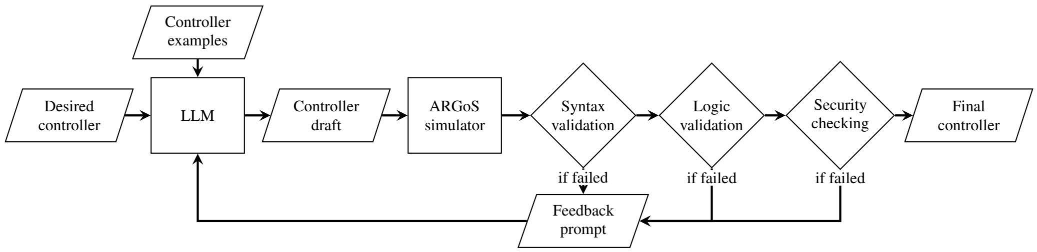

Figure 2: **Flow of LLM2Swarm's controller synthesis module**. Using LLM2Swarm's controller synthesis module, a user begins by specifying the desired controller in natural language. This specification, together with controller examples, is used as part of an LLM prompt to synthesize a robot controller. The synthesized controller draft is then executed directly in the ARGoS simulator. If ARGoS detects any syntax errors, the errors are reported back to the LLM, with the request to resolve them. If no syntax errors are found, the user can proceed to validate the controller logic: if the robot behavior is not as expected, the user provides information about both the robots' expected behavior and actual behavior to the LLM with the request to improve the controller. Once the controller logic is validated, the user can ask the LLM to check for any security vulnerabilities. After all validation steps are completed, the final controller, specified in programming code, is ready for deployment.

Step 1 – Syntax validation. The first validation is to check whether a synthesized controller contains syntactically correct and executable code. Consequently, a syntax validator is essential to prevent the execution of erroneous code, which could lead to disastrous outcomes (e.g., erroneous code controlling a drone could result in crashes and severe hardware damage).

Step 2 – Logic validation. Validating whether the generated code performs the intended task as expected is crucial but challenging, as there is not always a clear ground truth. If the code is synthesized before deployment, its resulting behavior can be analyzed and improved using LLM2Swarm—either by human feedback or in the future by the video analysis capabilities of LLMs (although the current video analysis capabilities are limited[1], we believe that future developments will improve this capability). During deployment, however, such verification is usually not possible—in addition, simulations cannot account for all possible real-world scenarios. Alternatives for evaluating a controller during deployment could be reinforcement learning to compare different controllers [2] or probabilistic model checking based on mathematical analysis [27].

Step 3 – Security checking. It is important to verify whether the LLM-synthesized controller contains security issues or potentially malicious parts (e.g., the controller could contain logic to sabotage other robots). Identifying security issues can be based on the *Common Weakness Enumeration* (CWE), a catalog that lists known exploits in programming languages and hardware components[2]. In addition, there are model checkers and LLMs fine-tuned to the task of identifying weaknesses that are part of the CWE. LLM2Swarm enables preliminary security checking through an LLM prompt with the request of identifying malicious parts and other weaknesses in a synthesized controller.

3.2 Direct Integration

For directly integrating LLMs into robot swarms, we propose to execute one LLM per robot, locally on each robot's hardware (see Figure 3), which requires lightweight LLMs. Currently, LLM2Swarm *simulates* local execution: In our showcases, each robot is connected to a separate *external* LLM, allowing us to run LLMs of any capacity and thus simulate today what we anticipate for the future. For direct integration, the LLMs' prompts are fed with sensor information (including messages received from other robots), and the outputs of the LLMs are mapped to robot actions. During deployment, a human operator may interact with these on-board LLMs to supervise the mission.

[1]Even though OpenAI's API does not yet support video analysis, in a preliminary experiment, we tested ChatGPT-4o's video analysis capabilities by uploading a video of *random-walk* behavior in an ARGoS simulation and asking whether the swarm demonstrated successful *flocking* behavior. The LLM's analysis was unsatisfactory, as it incorrectly concluded that the behavior was successful flocking behavior.

[2]`https://cwe.mitre.org/`, accessed September 20, 2024

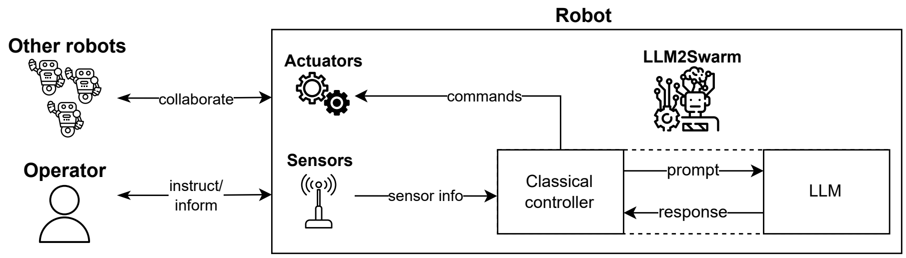

Figure 3: **System interactions for LLM2Swarm's direct integration module.** Using LLM2Swarm's direct integration module, a robot's controller is composed of two parts: a classical controller and an on-device LLM. As in traditional approaches, the classical controller manages the robot's actuators and sensors. However, unlike traditional approaches, the controller also creates prompts for the on-device LLM and uses the responses to guide the robot's actions. This on-device LLM also enables a robot to interact with other robots or a human operator by using natural language. As a result, such LLM-enabled robots can reason, plan, and collaborate thanks to the capabilities of their on-device LLMs.

3.3 Experiment setup

In our showcases, we simulate Pi-puck robots using the widely-used robot swarm simulator ARGoS [34] (version 3.0.0-beta56) on a personal computer (OS: Ubuntu 22.04.4 LTS, CPU: Intel Core i7-8550U @ 1.80 GHz, RAM: 16 GB, GPU: Intel UHD Graphics 620). Although ARGoS controllers are usually written in C++, we use the ARGoS-Python interface [21] (version v1.0.0), enabling us to easily connect each simulated robot to an LLM instance (we use GPT-4o).

4 Opportunities

In this section, we explore the opportunities that are enabled by LLM2Swarm: automated controller synthesis, enhanced robot-robot collaboration, and more intuitive human-swarm interaction.

4.1 Automated controller synthesis

Synthesizing controllers maintains the *responsiveness* of a robot swarm, as this can be done before deployment or during a low-activity phase during deployment. Overall, this automated controller synthesis can lead to faster development times, the specification of controllers by non-experts, and on-the-fly adaptation of robot controllers during deployment.

For creating controllers by LLMs, different design choices need to be made. It needs to be decided whether the LLM should synthesize the complete controller or only specific parts (e.g., only the movement logic). Another design choice is whether the controller (parts) should be written by an LLM before or rather during deployment. These design choices balance the trade-off between autonomy and controllability (see also Subsection 5.4).

4.2 Robot-robot collaboration and human-swarm interaction

Robot-robot collaboration and adaptivity in swarms. We believe that LLM2Swarm can largely improve robot-robot collaboration. First, enhancing robots with LLMs expands their ability to *reason*. Pre-trained LLMs incorporate general knowledge about the world and context-aware reasoning, which they can use to react to unanticipated challenges, incorporating moral values and favoring ethical actions—potentially considering how humans would react in similar situations.

Second, LLM2Swarm provides robots with advanced *planning* capabilities. Pre-trained LLMs can generate collective plans, involving the allocation of different robots to different tasks and the ability to adapt plans in real-time (for example, to adapt a path planning procedure when unanticipated situations occur). LLMs are also able to analyze the history of interactions and decide on the best next collective action based on this.

Finally, LLM2Swarm enables robots to *collaborate* using natural language, enabling more sophisticated interactions among robots and between robots and humans than classical controllers. As the interactions are expressed in natural language, they can also be easily understood by humans, making the interactions more transparent. This also enhances trustworthiness and improves accountability, as human operators can audit the decisions and actions taken by the robot swarm. Overall LLM2Swarm can potentially enhance a robot swarm's effectiveness, autonomy, resilience, and security, in particular in complex, unpredictable (and unknown) environments, such as search-and-rescue missions where fast decision-making is critical.

Human-swarm interaction. How to interact and collaborate as a human with a robot swarm in an effective manner while minimizing the cognitive load is an open research question [33; 35]. We believe that LLM2Swarm could play an important role in advancing the field of human-robot and human-swarm interaction. For example, a human operator could communicate with the LLMs of robot swarms through chat interfaces or voice commands, using natural language. This could be done with the goal of retrieving information about the collective state of the swarm [4] or of sending commands. The LLMs may be able to understand the human commands *within their context* (e.g., based on previous interactions, the current task that the swarm performs, or related to an object the swarm has just identified)—or ask back clarifying questions to the human operator. Moreover, in disaster response such as search-and-rescue operations, an LLM could interact with the victims using natural language (e.g., to inquire about the condition of the victims) and clarify the intentions and possible actions of the swarm (e.g., provide first aid or transport the victims to the hospital).

5 Challenges

We have identified the following main challenges for integrating LLMs into robot swarms: overcoming hardware limitations, ensuring scalability, addressing partitioning, and maintaining controllability. In the following sections, we describe each of these challenges in more detail.

5.1 Hardware limitations

To guarantee the autonomy of a robot swarm, the LLMs should be executed *locally* on each robot's hardware, as external infrastructure might not always be accessible during deployment (e.g., in underwater operations, underground mining, or space exploration) and could result in communication bottlenecks. Additionally, this ensures that data remains local, thereby protecting privacy in compliance with regulations like the GDPR. However, this local execution is difficult, as robots in swarms are typically assumed to have limited hardware capacities, even though recent hardware developments are starting to relax this assumption [25].

Scaling down hardware requirements and enabling on-board execution of LLMs on edge devices is a very active area of research; several approaches (e.g., finetuning [23], quantization [8; 29; 30], pruning [16; 31], distillation [22], and flash attention [7]) have been proposed for this. In general, these approaches are able to reduce the storage and memory requirements of LLMs, at the expense of lower accuracy and flexibility. We anticipate that both ongoing developments in hardware miniaturization (e.g., of microprocessors and GPUs) and the aforementioned software advancements will soon enable the execution of powerful LLMs on robots in swarms.

In a preliminary test, we executed TinyLlama via Ollama on a Raspberry Pi 5—a small-scale single board processor suitable for swarm robotics hardware. Executing example prompts from our showcase in Subsection 6.2, we obtained generation speeds between 10 and 12 tokens per second, indicating promising performance for robot-to-robot-collaboration and human-swarm interaction.

5.2 Scalability

Addressing scalability—that is, maintaining system stability when the number of robots increases—is another critical challenge when deploying LLM-integrated robot swarms.The higher the number of robots, the higher the number of potential inter-robot interactions. As the results of such inter-robot interactions may be necessary for a prompt, a higher number of them can make the prompt very complex, potentially exceeding token limits or leading to prolonged response times. Additionally, LLMs can favor verbosity over conciseness [36], which can negatively impact the efficiency of LLM-enabled robot swarms in terms of computational load and communication overhead. The potential exponential increase in token requirements could be reduced by restricting the robots' communication radius, in order to limit the number of messages exchanged with nearby robots.

5.3 Partitioning

Robot swarms may get partitioned into disconnected subswarms, halting the information flow between them. During such partitioning, the conversations in the subswarms might largely differ and different subswarms might not understand each other when they eventually reconnect. The partitioning problem could be partially reduced if the LLM itself controlled the robot actuators (e.g., rotors of drones) in a way that decreases the risk of partitioning. In addition, the LLMs could analyze the communication patterns and propose to execute a control logic dedicated to finding other subswarms if there is the risk of prolonged disconnections.

Still, new procedures are needed that enable both the autonomy of subswarms and the reconciliation of subswarm conversations. A possibility is to use blockchain technology for obtaining consistent states, as demonstrated in previous research [37; 38; 13]. Blockchain technology allows for maintaining trustworthy information in a shared database, even if the agents (the robots) who maintain the database do not trust each other. The blockchain could, for example, be used to store LLM responses or decisions made in different sub-swarms—thus preventing Byzantine robots (robots that show a discrepancy between their intended behavior and their actual behavior, for example, due to broken parts or malicious control [37]) from tampering with the data.

5.4 Controllability

Increasing the autonomy of robot swarms—that is, reducing the reliance on external control or infrastructure—is a main goal of swarm robotics research [10; 3]. We believe that LLMs can significantly contribute to achieving this goal. However, when increasing autonomy, it is crucial to ensure the *controllability* of robot swarms, so that they perform their intended tasks while also acting in compliance with rules and regulations [13]. When enhancing robot swarms with LLMs, there is no guarantee that the swarm will behave as intended, due to the probabilistic nature of LLM responses.

To partially counteract this problem, we proposed a three-step validation procedure for LLM2Swarm's *indirect* integration module (see Section 4.1) that can potentially enhance the robustness and reliability of the synthesized controllers, ensuring more secure and predictable deployments. With LLM2Swarm's *direct* integration module, new attack vectors arise that have already started to be identified in other research areas that integrate LLMs [18]. For example, it needs to be studied if users can elicit private data from robots by using specific prompts. Another important issue is determining if a user—or even a robot—can reprogram robots through prompt injection attacks. Additionally, it is essential to develop methods for identifying Byzantine robots (see our sub-showcases "Self diagnosis" and "Peer diagnosis" in Figure 4) that send misleading information.

6 Demonstrations

In this section, we detail our vision and provide brief illustrative showcases implemented in our LLM2Swarm software package (see also the accompanying videos at `https://github.com/Pold87/LLM2Swarm`). In future research, more detailed studies are needed to systematically evaluate performance metrics.

6.1 Showcase — Controller synthesis (aggregate-then-disperse algorithm)

In our controller synthesis showcase, we use LLM2Swarm's indirect integration module to synthesize a controller that lets the robots aggregate in the center of the arena, and then disperse after 150 timesteps. For the controller examples, we use a random-walk implementation and a targeted navigation implementation. The LLM is able to extrapolate from these few examples—even though the ARGoS-Python interface is not a widely-used interface and does not feature any API description—and synthesize a correct controller. This demonstrates LLM2Swarm's potential to facilitate the design of robot swarms—in particular as no programming knowledge is required.

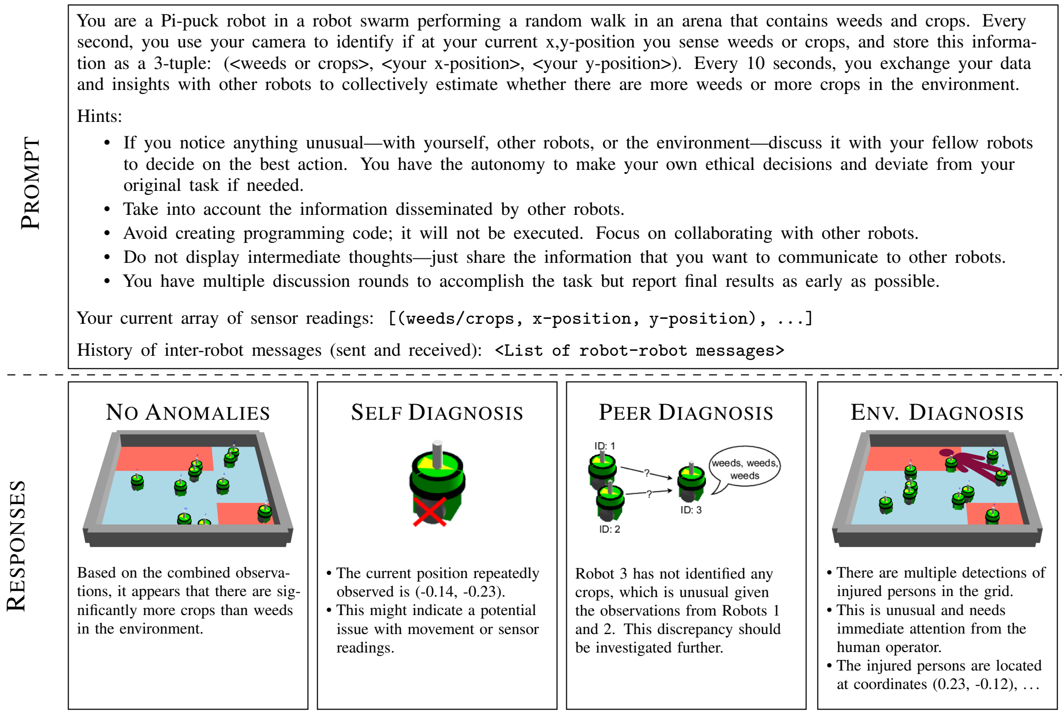

Figure 4: **Showcase — Robot-robot collaboration (anomaly detection).** In this showcase, the robots' task is to determine whether the environment contains more crops (blue floor) or more weeds (red floor). To do so, the robots use LLM2Swarm's direct integration module. The robots are given the prompt (as shown in the upper part of the figure) and use it to generate the responses (i.e., inter-robot messages), as shown in the lower part, across four sub-showcases. The text below the images are LLM-generated responses of a select robot, obtained by executing LLM2Swarm. The system was able to successfully diagnose the studied anomalies without prior information about the exact nature of the anomalies.

6.2 Showcase — Robot-robot collaboration (anomaly detection)

In this showcase (Figure 4), we demonstrate LLM2Swarm's direct integration capabilities. The robots perform a random walk and are tasked with determining whether there are more crops (blue floor) or more weeds (red floor) in the environment—and exchange their findings via LLM-generated inter-robot messages. We study four sub-showcases, each demonstrated with illustrative LLM responses below the images.

1. *No anomalies*: The robots sense the environment and exchange inter-robot messages using their (simulated) on-board LLMs. Over time, they correctly identify that there are more crops than weeds in the environment.

2. *Self diagnosis*: We disable all the robots' wheels. The robots detect their constant x, y-coordinates and correctly suggest that there is an issue with movement or sensor readings.

3. *Peer diagnosis*: We simulate a fault in one of the robots, causing the robot to only sense weeds. The other robots detect this fault by comparing sensor readings and suggest further investigations.

4. *Environmental diagnosis*: We simulate that an injured person is part of the environment (implemented by the sensor reading 'injured person' if a robot moves on top of the person). The robots recognize this anomaly and ask for immediate human help.

An interesting aspect of this showcase is that LLM2Swarm is able to identify various anomalies using the same prompt, without prior information about the anomalies. This provides a first indication for LLM2Swarm's potential of incorporating commonsense knowledge into robot swarms.

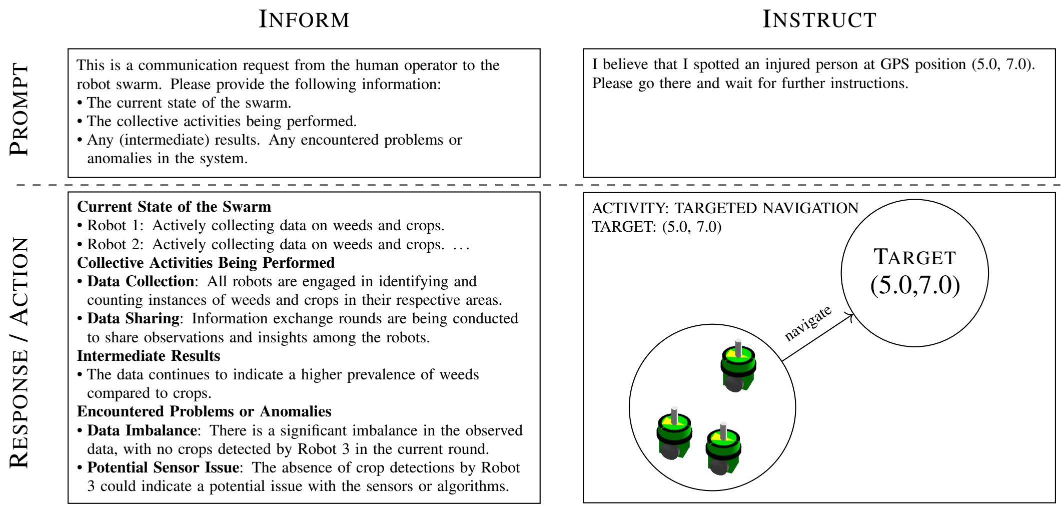

Figure 5: **Showcase — Human-swarm interaction.** This showcase demonstrates how LLM2Swarm enables a human operator to retrieve information about the current state of the swarm and to send instructions to the swarm. The upper part of the figure shows the prompt sent by the human operator to the robots, the lower part shows the corresponding responses. In the Inform showcase (left), a selected robot generates a concise summary of the swarm's current activities and intermediate results. In the Instruct showcase (right), the robots move to the specified target that they extracted from the natural language input provided by the human operator.

6.3 Showcase — Human-swarm interaction (inform and instruct)

Figure 5 displays our showcase for human-swarm interaction: The upper part of the figure contains the human's prompt, the lower part depicts the LLM response or robot swarm action. When the human uses the *Inform* prompt, the selected robot provides a human-readable concise version of all interactions. Using the *Instruct* command, LLM2Swarm translates the instructions specified in natural language into executable robot commands. In this showcase, LLM2Swarm (i) correctly and understandably summarizes the current swarm activities and (ii) understands the human's instruction about deviating from the original task. This provides indication that LLM2Swarm facilitates human-swarm interaction, in particular for human operators without programming knowledge.

7 Conclusions

Integrating LLMs into robot swarms enables natural language communication both among robots and with human operators. In our series of showcase experiments, we demonstrated that such an integration(i) facilitates controller generation, (ii) increases a robot swarm's autonomy, and (iii) introduces new ways for humans to interact with a robot swarm. Once technical challenges are overcome, LLM-enabled robot swarms could transform swarm robotics research by simplifying and automating the solution of several existing research challenges. We release the LLM2Swarm software and videos to encourage further developments in this area: `https://github.com/Pold87/LLM2Swarm`.

8 Acknowledgements

V.S. and M.D. acknowledge support from the Belgian F.R.S.-FNRS. In addition, V.S. acknowledges the Helmholtz Information & Data Science Academy (HIDA) for providing financial support enabling a short-term research stay at CISPA Helmholtz Center for Information Security. This work was also partially funded by ELSA - European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement number 101070617, as well as the German Federal Ministry of Education and Research (BMBF) under the grant AIgenCY (16KIS2012). The authors thank Avian Krämer of the CISPA Helmholtz Center for Information Security for proofreading this manuscript.

References

[1] S. Abdelnabi, A. Gomaa, S. Sivaprasad, L. Schönherr, and M. Fritz. Cooperation, competition, and maliciousness: LLM-stakeholders interactive negotiation. doi:10.48550/arXiv.2309.17234, 2024. Preprint, `arXiv:2309.17234 [cs.CL]`.

[2] M. Blais and M. Akhloufi. Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators. *Cognitive Robotics*, 3:226–256, 2023. doi:10.1016/j.cogr.2023.07.004.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. doi:10.1007/s11721-012-0075-2.

[4] D. S. Brown, M. A. Goodrich, S.-Y. Jung, and S. Kerman. Two invariants of human-swarm interaction. *Journal of Human-Robot Interaction*, 5(1):1–31, 2016. doi:10.5898/JHRI.5.1.Brown.

[5] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan. Scalable multi-robot collaboration with large language models: Centralized or decentralized systems? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2024)*, pages 4311–4317. IEEE Press, 2024. doi:10.1109/ICRA57147.2024.10610676.

[6] Y.-S. Chuang, A. Goyal, N. Harlalka, S. Suresh, R. Hawkins, S. Yang, D. Shah, J. Hu, and T. T. Rogers. Simulating opinion dynamics with networks of LLM-based agents. doi:10.48550/arXiv.2311.09618, 2023. Preprint, `arXiv:2311.09618 [physics.soc-ph]`.

[7] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 16344–16359. Curran Associates, Inc., 2022.

[8] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer. `GPT3.int8()`: 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 30318–30332. Curran Associates, Inc., 2022.

[9] Y. Ding, X. Zhang, S. Amiri, N. Cao, H. Yang, A. Kaminski, C. Esselink, and S. Zhang. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Autonomous Robots*, 47(8):981–997, 2023.

[10] M. Dorigo, M. Birattari, and M. Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014. doi:10.4249/scholarpedia.1463.

[11] M. Dorigo, G. Theraulaz, and V. Trianni. Reflections on the future of swarm robotics. *Science Robotics*, 5(49):abe4385, 2020. doi:10.1126/scirobotics.abe4385.

[12] M. Dorigo, G. Theraulaz, and V. Trianni. Swarm robotics: Past, present and future. *Proceedings of the IEEE*, 109(7):1152–1165, 2021. doi:10.1109/JPROC.2021.3072740.

[13] M. Dorigo, A. Pacheco, A. Reina, and V. Strobel. Blockchain technology for mobile multi-robot systems. *Nature Reviews Electrical Engineering*, 1(4):264–274, 2024. doi:10.1038/s44287-024-00034-9.

[14] B. Finkbeiner, N. Metzger, and Y. Moses. Information flow guided synthesis. In *Proceedings of the 34th International Conference on Computer Aided Verification (CAV 2022)*, volume 13372 of *LNCS*, pages 505–525. Springer, 2022.

[15] G. Francesca, M. Brambilla, A. Brutschy, V. Trianni, and M. Birattari. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8 (2):89–112, 2014. doi:10.1007/s11721-014-0092-4.

[16] E. Frantar and D. Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, pages 10323–10337. JMLR, 2023.

[17] D. Garzón Ramos and M. Birattari. Automatically designing robot swarms in environments populated by other robots: An experiment in robot shepherding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2024)*, pages 12240–12247. IEEE Press, 2024. doi:10.1109/ICRA57147.2024.10611309.

[18] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, AISec '23, pages 79–90. ACM, 2023. doi:10.1145/3605764.3623985.

[19] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang. Large language model based multi-agents: A survey of progress and challenges. doi:10.48550/arXiv.2402.01680, 2024. Preprint, `arXiv:2402.01680 [cs.CL]`.

[20] H. Hamann. *Swarm Robotics: A Formal Approach*. Springer, 2018. doi:10.1007/978-3-319-74528-2.

[21] K. Hasselmann, A. Parravicini, A. Pacheco, and V. Strobel. Python wrapper for ARGoS3 simulator. doi:10.5281/zenodo.13765570, 2024. GitHub repository, version v1.0.0.

[22] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS 2015 Deep Learning and Representation Learning Workshop*, 2015.

[23] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 328–339. ACL, 2018. doi:10.18653/v1/P18-1031.

[24] A. Jiao, T. Patel, S. Khurana, A.-M. Korol, L. Brunke, V. Adajania, U. Culha, S. Zhou, and A. Schoellig. Swarm-GPT: Combining large language models with safe motion planning for robot choreography design. In *NeurIPS 2023 Workshop on Robot Learning*, 2023.

[25] S. Jones, M. Studley, S. Hauert, and A. F. T. Winfield. A two teraflop swarm. *Frontiers in Robotics and AI*, 5, 2018. doi:10.3389/frobt.2018.00011.

[26] Y. Kim, D. Kim, J. Choi, J. Park, N. Oh, and D. Park. A survey on integration of large language models with intelligent robots. *Intelligent Service Robotics*, 17(5):1091–1107, 2024. doi:10.1007/s11370-024-00550-5.

[27] S. Konur, C. Dixon, and M. Fisher. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2):199–213, 2012. doi:10.1016/j.robot.2011.10.005.

[28] J. Kuckling. Recent trends in robot learning and evolution for swarm robotics. *Frontiers in Robotics and AI*, 10, 2023. doi:10.3389/frobt.2023.1134841.

[29] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of the 7th Annual Conference on Machine Learning and Systems (MLSys 2024)*, pages 87–100, 2024.

[30] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024)*, volume 235 of *Proceedings of Machine Learning Research*, pages 32332–32344. PMLR, 2024.

[31] X. Ma, G. Fang, and X. Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pages 21702–21720. Curran Associates, Inc., 2023.

[32] Z. Mandi, S. Jain, and S. Song. RoCo: Dialectic multi-robot collaboration with large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2024)*, pages 286–299. IEEE Press, 2024. doi:10.1109/ICRA57147.2024.10610855.

[33] A. Paas, E. B. J. Coffey, G. Beltrame, and D. St-Onge. Towards evaluating the impact of swarm robotic control strategy on operators cognitive load. In *Proceedings of the 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2022)*, pages 217–223. IEEE Press, 2022. doi:10.1109/RO-MAN53752.2022.9900763.

[34] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4): 271–295, 2012. doi:10.1007/s11721-012-0072-5.

[35] G. Podevijn, R. O'Grady, N. Mathews, A. Gilles, C. Fantini-Hauwel, and M. Dorigo. Investigating the effect of increasing robot group sizes on the human psychophysiological state in the context of human-swarm interaction. *Swarm Intelligence*, 10(3):193–210, 2016. doi:10.1007/s11721-016-0124-3.

[36] K. Saito, A. Wachi, K. Wataoka, and Y. Akimoto. Verbosity bias in preference labeling by large language models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[37] V. Strobel, E. Castelló Ferrer, and M. Dorigo. Managing Byzantine robots via blockchain technology in a swarm robotics collective decision making scenario. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, pages 541–549. IFAAMAS, 2018.

[38] V. Strobel, A. Pacheco, and M. Dorigo. Robot swarms neutralize harmful Byzantine robots using a blockchain-based token economy. *Science Robotics*, 8(79):eabm4636, 2023. doi:https://doi.org/10.1126/scirobotics.abm4636.

[39] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 671–682. PMLR, 2020.

[40] V. Trianni, E. Tuci, C. Ampatzis, and M. Dorigo. Evolutionary swarm robotics: A theoretical and methodological itinerary from individual neuro-controllers to collective behaviours. In *The horizons of evolutionary robotics*, volume 153. MIT Press, 2014.

[41] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor. ChatGPT for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft, 2023. URL `https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/`.

[42] A. F. T. Winfield and J. Nembrini. Safety in numbers: Fault tolerance in robot swarms. *International Journal on Modelling Identification and Control*, 1(1):30–37, 2006. doi:10.1504/IJMIC.2006.008645.

[43] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood. The grand challenges of *Science Robotics*. *Science Robotics*, 3(14): eaar7650, 2018. doi:10.1126/scirobotics.aar7650.

[44] H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan. Building cooperative embodied agents modularly with large language models. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, 2024.