Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 128 (2024) 621–626

www.elsevier.com/locate/procedia

34th CIRP Design Conference

Leveraging Generative AI Prompt Programming for Human-Robot Collaborative Assembly

Christos Konstantinou[a], Dimitris Antonarakos[a], Panagiotis Angelakis[a], Christos Gkournelos[a], George Michalos[a], Sotiris Makris[a]*

[a]*Laboratory for Manufacturing Systems and Automation, Department of Mechanical Engineering and Aeronautics, University of Patras, Patras, 26504, Greece*

* Corresponding author. Tel.: +30-2610-910160; fax: +30-2610-997314. *E-mail address:* makris@lms.mech.upatras.gr

Abstract

In manufacturing, traditional robotic programming methodologies have often been focused on independent operation, offering limited capabilities for seamless human-robot collaboration. This paper introduces a paradigm shift in collaborative production systems by leveraging generative artificial intelligence (AI), specifically large language models (LLMs). Contrary to traditional methods that rely on pre-defined assembly instructions, this paper introduces a novel framework employing primitive knowledge of the production process, including product design and required assembly steps. By integrating LLMs and a behavior tree-based system control, this approach enables programmers to rapidly deploy collaborative assembly procedures by expediting the programming of robotic operations. The system also incorporates Natural Language Processing (NLP) technologies, which facilitate real-time alterations in assembly steps, leading to reduced overall production time. The framework's behavior tree-based control architecture allows for dynamic adaptability, offering optimized solutions across a range of assembly scenarios. The results of the framework's deployment suggest that this innovative programming paradigm significantly enhances both the adaptability and efficiency of collaborative manufacturing settings.

Keywords: Generative AI; Human Robot collaboration; Design informatics;

1. Introduction

In recent advancements in manufacturing [1], under the paradigm of Industry 5.0 (I5.0) [2] , the role of human-robot collaboration (HRC) [3] has significantly elevated, enhancing the efficiency of manufacturing processes by synergizing the adaptability of humans with the robustness and repeatability of robots, whilst upholding social values [4]. This evolution represents an extension from Industry 4.0, characterized by the initial integration of artificial intelligence (AI) in manufacturing [5,6]. Prior studies have investigated the utilization of natural language for controlling robots [7]. With the advent of Generative AI specifically Large Language Models (LLM) and their improved ability to interpret human language, behave as agents, formulate plans and use external tools [8], the barrier of seamless co-operation between robots and humans is lowered. Nonetheless, challenges persist in these systems' design abilities to comprehend task contexts, engage in logical reasoning, facilitate 'human-like' communication, and specifically generate HRC collaborative assembly plans that are grounded in the shopfloor environment and respect resource and tool constraints.

Some recent works have utilized LLMs as a task planners and used them to generate sequential or conditional task sequences in a zero-shot or few-shot fashion. The SayCan framework [9] combines a human high-level instruction and its corresponding robot primitive tasks as the prompt, rescoring matched admissible actions using a learned value function. The ProgPrompt [10] represents robot tasks as Pythonic programs and then takes the Pythonic code as the prompt, while also utilizing assertions to skip tasks based on environmental state. Similar works like [11] have proven that the robotic program generation is possible by constructing specific prompts including few shot learning making the system able to decompose natural language instructions to python code. Furthermore, promising results to LLM program generation are presented in [12], that using symbolic goals and a geometric feasibility planner, LLMs were able to create instructions to reach a high level goal. These examples prove that LLMs can be used in low level robot control, however they lack in the ability to obtain real manufacturing data and combining collaborative environments in a flexible way, generalizing a high-level control architecture. To overcome these limitations, other approaches enabled modular control using the behaviour trees (BTs). As stated in [13] BTs can be useful in robot programming tasks exhibiting characteristics of transparency and modularity. Also, it is shown that in the context of HRC, BTs can be used to improve the reactivity of the systems. As AI technologies progress, a lot more research is focusing on the integration of LLMs with such modular robotic architectures, since these transformer-based networks are well-suited for the task of creating strictly structured text files based on the specified logic. Relative works have utilized BT's for assembly operations [14] , while others have utilized LLM generative abilities to automatically create behaviour trees grounded in task domain [15] and [16]. Despite that, these works do not provide BT's suitable for HRC which requires online reactivity or respecting the assembly, shopfloor, and robot constraints. On the other hand, other works [17] have integrated smart ways of robot control using AR and VR technologies, including the HRC capabilities.

This work aims to enhance previous works with the integration of LLM agents that have acquired knowledge regarding the production system. These agents are deployed inside a framework that encapsulates runtime proprietary information about the production cell, resource capabilities with constraints and process plan requirements. It can generate a complete process execution program, which utilizes Behaviour Tree control systems having also HRC enabled capabilities addressing the intricate design challenges inherent in programming robotic cells. This control structure allows for the existence of only one tree to behave as the robot controller, improving loading times, monitoring of execution switching and robustness.

This paper is organized as follows: Section 2 presents the proposed approach and the overall system architecture. Section 3 details the implementation of each module, including integration with the user interface. Section 4 present the deployment of the solution in two distinct industrial use cases. Finally, Section 5 discusses the results and the impact of the solution.

2. Approach

To address the inherent complexities in industrial systems and facilitate efficient HRC, the proposed innovative approach leverages LLM capabilities. Central to this methodology is the use of *perplexity*, a key metric derived from LLMs, which serves as a measure of the model's uncertainty in predicting a sequence.

This metric establishes an abstraction layer that allows for correlating disparate data sources without the necessity for complex databases. This approach ensures that data entries are correlated based on contextual information, thereby offering a

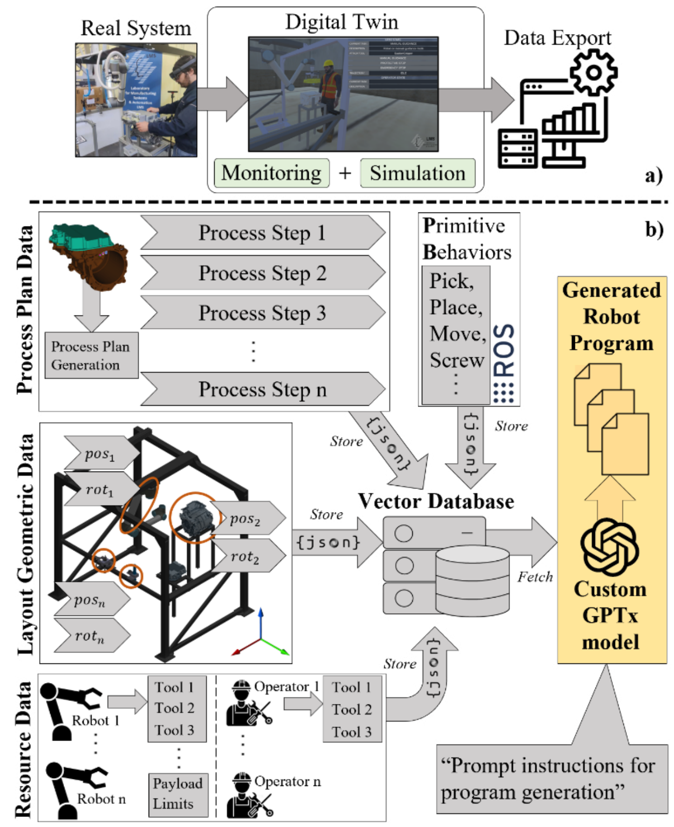


Fig. 1. a) Data export from the digital twin. b) Overall data driven architecture approach

more adaptable and efficient system for managing intricate and interconnected data.

As illustrated in Fig. 1, the comprehensive approach encompasses a set of data models, exported from a Digital Twin (DT) [18] system. This system simulates the industrial environment and monitors system data sources in real-time. Subsequently, these data are vectorized and stored inside a vector database. A GPT-based generative AI solution then utilizes this consolidated vectorized data to obtain the integrated application specific knowledge and later create the execution program. The execution component of the system, rooted in the Behavior Tree (BT) framework, fetches, and implements this program

The BT framework is instrumental in this approach, providing essential features like block interchangeability or modularity, reactivity, and the facility to create custom control and decorator nodes. Its unambiguous and comprehensible structure, storable in an XML format, enables even non-technical specialists to interact with and control robots in a workplace setting in a logical and intuitive manner. Following the approach of this solution, the required data models are:

Resource data model: This model encompasses a comprehensive representation of resources within the production system. It details specific toolsets for each resource category, along with crucial parameters such as maximum payload capacities and positional constraints. This data model is instrumental in analyzing and optimizing resource utilization in industrial operations.

Geometric Layout Data Model: Containing positions and rotations for every part and tool in the production system, this model is automatically generated by integrating production

CAD files.

Process Plan Data Model: It encompasses a series of instructions for assembling or disassembling products. The production steps are derived during the product design phase, utilizing 3D CAD file design systems and algorithms, such as assembly by disassembly, to automatically generate these instructions.

Primitive Behavior List: A foundational set of behaviors forms the building blocks for the LLM-augmented solution. These behaviors are systematically combined by the LLM to create the desired program. The selection and integration of these building blocks will be elaborated in the implementation section.

The knowledge augmentation phase depends on these vectorized data since this vector database retrieval structure is optimized for LLM fast and reliable lookup. In general, the knowledge base of LLMs can be enhanced through techniques such as fine-tuning and retrieval augmentation. Fine-tuning involves adjusting the model's weights based on a new set of vectorized input data that it hasn't seen before. This process demands a substantial amount of training data and can be time-consuming, often yielding only modest results, particularly when dealing with small training datasets [19]. In contrast, retrieval augmented LLMs do not exclusively rely on internal knowledge to generate responses; they extract relevant information from privately stored data.

After the knowledge augmentation phase, a flexible retriever mechanism is used to fetch these vectorized data enabling a general-purpose LLM to be applied in a range of diverse applications, including the HRC domain. The inherent capability of LLMs is ideal to abstract the interaction in a human-like way, without the need for extensive training or predefined scripts, making them versatile tools for various applications, including natural language understanding, content generation, and personalized assistance. Furthermore, this application is able to recall previous responses using its internal conversation memory.

The flexibility of the human language as interface for such complex systems enables the generation and execution of descriptive robotic programs that are relative to the data fetched from all these industrial oriented data models. As it will be described thoroughly in the case study section, this new way of robot programming is able to enhance the usability of complex industrial systems, minimizing the programming times, allowing for easy reconfigurability.

3. Implementation

3.1. Industrial Data Models

As presented in the approach section, there are 3 separate data models. The *process plan data*, the *layout geometric data*, and the *resource data*. Each of the models are described with a predefined JSON schema. The process plan data model primarily encompasses a list containing all the available process steps. These steps are characterized by two key fields: the *process type*, described as a verb (e.g., "*screwing*" or "*picking*"), and the *part name*. In the layout data model, a list enumerates the available products, with each product featuring a list of parts required for assembly. Each product part shares its name with the process plan data model and includes *position and rotation* attributes corresponding to the physical ones in the real industrial cell. The final data model encapsulates resource correlations. The data are organized into a list of two

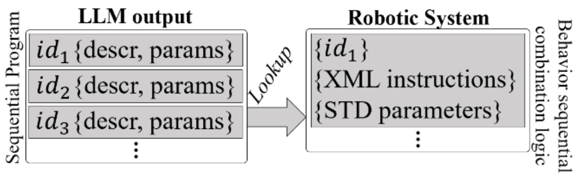


Fig. 2. Sequential program generation

objects with the names "*UR10*" and "*Operator1*," representing the available resources within the test environment. Each resource is classified as either "*robot*" or "*human*" and is further characterized by a *payload*, a *tool handling capability*, and a *behaviour set*. The tool handling capabilities entail a list of tools that the resource can manage, while the behaviour set comprises a list of primitive tasks that the resource can execute.

Notably these data models exhibit inherent correlations, and the role of the LLM lies in establishing comprehensive correlations utilizing the perplexity metric.

3.2. Selection of Primitive Behaviours

Primitive behaviours provide a foundation for more complex actions. They are used as building blocks, and the LLM is responsible to consecutively combine them to create the final program, that is ready for execution. Every complex task can be analyzed into smaller sub-tasks. Upon analyzing the use case, it was concluded that all the robot tasks can be generated through the use of the following primitives: **Pick**, **Place, Move, Screw, Manual Guidance** and **ToolChange**. Each of these tasks can be interpreted as a computer parametrized function. As shown in Fig. 2, the required parameters are customized for each of the primitive actions: $Pick \rightarrow part\_name$, $Place \rightarrow position$, $Move \rightarrow position$, etc. All these parameters are subsets of the information stored in the vector database. As it will be described in the section 3.3, the LLM will be responsible for generating all of these parameters. To further minimize the errors [20] generating these complex XML based instructions, a pointer semantic architecture was utilized. This architecture as seen in Fig. 2, translates the generated id task sequence from the LLM output to a complete robotic sequential behavior program. Each of the ids has one-to-one correlation with the primitive behaviors.

The LLM demonstrates the ability to construct individual tasks using primitives and further combine these primitives in logical sequential steps. This process is ultimately aimed at assembling or disassembling the products stored within its database.

3.3. Knowledge Augmentation with Vector DB

In this study, the knowledge augmentation of the LLM can be achieved through the storage of the data in a vector database as shown in Fig. 3.

The data are first passed through an embedding model which creates the data's word embeddings.

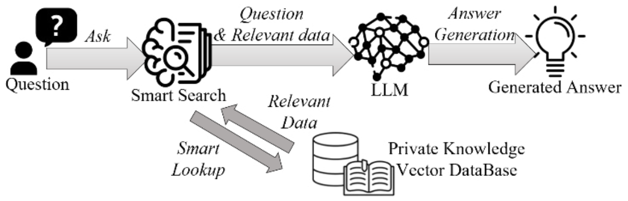


Fig. 3. LLM Vector DB embedding

Word embeddings are vector representations of words where each dimension of the vector represents a different aspect of the word's meaning. These preprocessed data are then stored into a PostgreSQL. To establish a connection between the LLM and a Vector Database and efficiently fetch the vectorized embedded data, the LangChain [21] framework has proven to be a valuable asset. This open-source framework can seamlessly integrate LLMs, such as OpenAI's GPTx models, with external components, facilitating the development of such custom LLM-driven applications.

3.4. Interaction Interfaces

The final aspect of the system's implementation is regarding the interaction with the user. The interface was designed as a Unity editor plugin, using the Unity open SDK, within the existing DT software. The selection of the Unity development platform for creating the simulation and visualization DT software was beneficial due to the availability of open-source plugins suitable for industrial and robotic applications. Given the necessity for direct robot communication and control, the system's core is constructed on top of the ROS [22] framework, making use of valuable assets like motion planning and collision avoidance. The framework also leverages a behaviour tree structure for intelligent system control. The developed plugin follows a chat-bot structure, enabling quality assessment of the generated robot programs (LLM responses) before direct execution. This quality assessment and the communication aspect a detailed diagram is presented in Fig. 4. One additional advantage of a ROS-BT framework is the lack of compilation need due to the nature of the XML instructions that can be executed immediately by the system.

4. Case Study

The presented work for seamless programming of a collaborative assembly has been deployed and tested in two distinct use cases drawn from the automotive and machinery industries. The first use case revolves around the assembly of an electric motors, while the second pertains to the assembly of an industrial air compressor unit.

The required data for the knowledge augmentation of LLM were created on the design and construction phase of the assembly station. These data are used also for the construction of the Digital Twin of the system. Both the physical and digital environments are depicted in Fig. 5. The selection of these two use cases was deliberate, aiming to showcase the versatility of data sources and thoroughly evaluate the performance of this programming solution. As will be explored later, the results obtained are promising, indicating the potential for extending this solution to diverse manufacturing sectors.

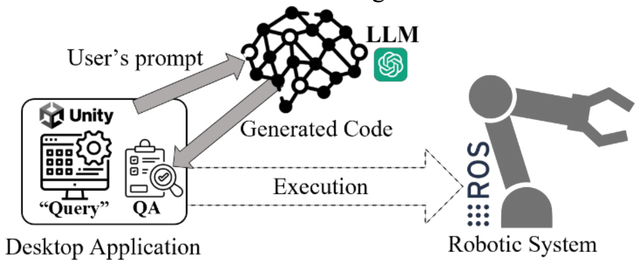


Fig. 4. Generative AI feedback system

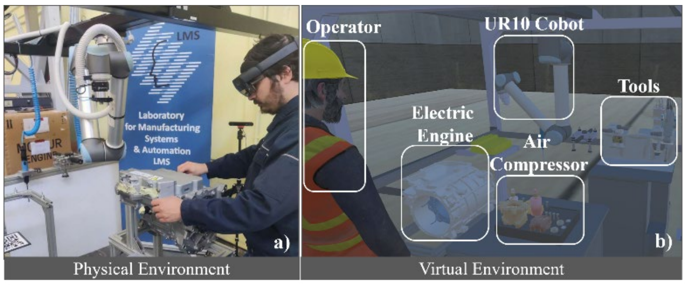
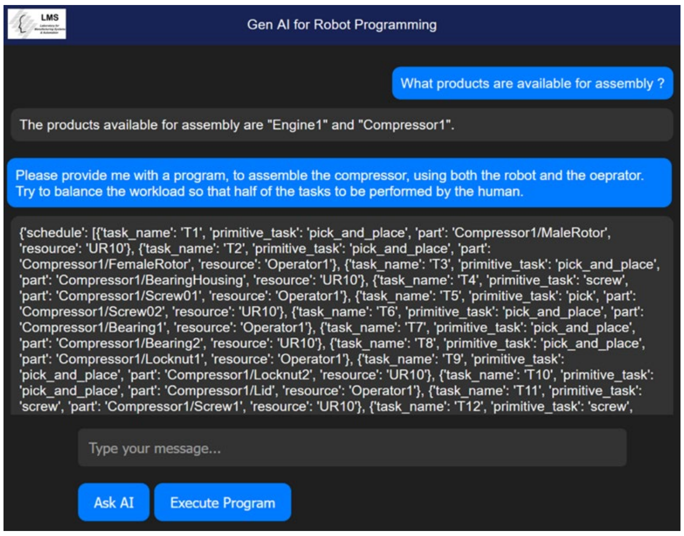


Fig. 5. Physical and Virtual Assembly environment

Fig. 6 demonstrates the system's capability to comprehend and correlate various data sources. This ability is crucial for generating effective HRC assembly instructions and is a testament to the intrinsic strengths of the generative AI networks employed. Furthermore, it is able to recognize natural human language and concepts, such as the human robot collaboration, and create programing solutions that try to include such information.

In this specific example, it creates a balanced schedule between the robot's and human's workload, assigning different task in an alternate way. The way each task assigned to an available resource is derived from the capability and availability of each resource based on semantic correlation of the augmentation data. This is critical for optimizing productivity and ergonomics in HRC scenarios. It is also able to reconstruct the process plan tasks using only the known primitives. The system's capability to incorporate new tasks, such as tool changing, highlights its adaptability and responsiveness to dynamic manufacturing requirements. This flexibility is a significant advancement over traditional, more rigid systems.

The vector knowledge augmentation always follows a given JSON output ontology, without using the few-shot learning approach presented in [23], giving more precise and promising results. Continuing with the experimentation of this program generation, a unified and a valid method of comparison was established using the following parameters: The "task level abstraction", "reusability and flexibility", "teaching and interaction".



What products are available for assembly ?

The products available for assembly are "Engine1" and "Compressor1".

Please provide me with a program, to assemble the compressor, using both the robot and the oeprator. Try to balance the workload so that half of the tasks to be performed by the human.

```
{'schedule': [{'task_name': 'T1', 'primitive_task': 'pick_and_place', 'part': 'Compressor1/MaleRotor',
'resource': 'UR10'}, {'task_name': 'T2', 'primitive_task': 'pick_and_place', 'part':
'Compressor1/FemaleRotor', 'resource': 'Operator1'}, {'task_name': 'T3', 'primitive_task': 'pick_and_place',
'part': 'Compressor1/BearingHousing', 'resource': 'UR10'}, {'task_name': 'T4', 'primitive_task': 'screw',
'part': 'Compressor1/Screw01', 'resource': 'Operator1'}, {'task_name': 'T5', 'primitive_task': 'pick', 'part':
'Compressor1/Screw02', 'resource': 'UR10'}, {'task_name': 'T6', 'primitive_task': 'pick_and_place', 'part':
'Compressor1/Bearing1', 'resource': 'Operator1'}, {'task_name': 'T7', 'primitive_task': 'pick_and_place',
'part': 'Compressor1/Bearing2', 'resource': 'UR10'}, {'task_name': 'T8', 'primitive_task': 'pick_and_place',
'part': 'Compressor1/Locknut1', 'resource': 'Operator1'}, {'task_name': 'T9', 'primitive_task':
'pick_and_place', 'part': 'Compressor1/Locknut2', 'resource': 'UR10'}, {'task_name': 'T10', 'primitive_task':
'pick_and_place', 'part': 'Compressor1/Lid', 'resource': 'Operator1'}, {'task_name': 'T11', 'primitive_task':
'screw', 'part': 'Compressor1/Screw1', 'resource': 'UR10'}, {'task_name': 'T12', 'primitive_task': 'screw',
```

Fig. 6. HRC assembly instruction program generation from simple natural language instructions

Task-Level Abstraction: The proposed method abstracts the programming at a higher level, focusing on defining tasks and decomposing them into primitive behaviours. This simplifies the programming process, since the work is performed with higher-level commands. Table 1 shows some examples in the task decomposition made from the LLM. Traditional programming methods for robots often involve low-level coding of individual joint movements or trajectories. Teaching by demonstration as presented in [24] might also involve manually guiding the robot through the desired motions. This simplification of the programming process is a core advantage of this approach.

Table 1: Examples of task decomposition

High Level Assembly Tasks	Primitive Decomposition
"Bring the Inverter in front of the operator"	1. $tool\_change\{gripper\_1\}$ 2. $pick\ \{inverter\}$ 3. $move\{collab\_area\_1\}$
"Place the female rotor in the main housing"	1. $tool\_change\{gripper\_2\}$ 2. $pick\{rotor\_1\}$ 3. $place\{rotor\_1, housing\}$
"Fix inverter in place using the 2 M8 screws"	1. $tool\_change\{scrd\_m8\}$ 2. $pick\{screw01\}$ 3. $screw\{screw01\}$ 4. $pick\{screw02\}$ 5. $screw\{screw02\}$

Reusability: The task decomposition into primitive building blocks can lead to more efficient and modular programming reusing these behaviours into more complex tasks. Traditional methods required more complex coding procedures and required significant effort for a new task adaptation. In the proposed approach the time to add a new product to the database, until the final output of an HRC program was measured. The addition of the process plan, layout data, and resource data, adding the wait time for the program generation, took an average time of ~10 minutes. This time is regarding only the scope of the programming tool and not the actual physical setup.

Teaching and Interaction: This proposed method encapsulates the natural language understanding that can be used in programming and interaction. This aspect ensures a natural interaction with complex systems, based on the minimal amount of information. Similar researches have shown [25] that higher level programming ways lead to more efficient robot programming, and also allowing people with little or no technical skills to interact with such systems. On the other hand, current programming approaches lack in the context of additional intelligence either in the form of a production data inclusion, or in the form of generative AI solutions and require extensive system knowledge.

5. Conclusions

This paper introduced a novel programming method for robotic applications in industrial environments, with a specific focus on enhancing collaborative scenarios facilitating a fast system simulation during the design stage and easing programming during physical commissioning. Central to the proposed approach is the utilization of LLMs, which have demonstrated exceptional capability in simplifying complex data structures and aiding in logical reasoning and content generation. Through this methodology, it is shown that leveraging the inherent abilities of LLMs allows for the logical combination of disparate information, transforming this into cohesive, simplified solutions based on fundamental building block instructions. This process effectively decomposes and restructures complex tasks into manageable components also enabling quick repurposing or reconfiguration of the system for diverse manufacturing processes.

A pivotal aspect of this method is its seamless integration with digital twin solutions. This integration enables not only effective simulation but also efficient execution within industrial settings, enhancing both the planning and implementation phases of robotic programming. By embedding this programming approach within digital twin technology, a level of synchronization and realism that significantly enhances the feasibility and applicability of this solution in real-world scenarios was achieved.

The implications of this programming approach for industrial environments are profound. By abstracting task creation and leveraging natural language, this method paves the way for more intelligent, flexible, and efficient collaborative applications. While this study has provided valuable insights into the HRC program generation, there are several avenues for future research that need exploration. The system's augmentation database can be enhanced using bigger and more diverse amount of data to test the ability of LLM agents to interpret specific instructions. Additionally, advances in image recognition systems such as [26] can be integrated within the proposed system to augment the database with real time scene information and increase the overall reactivity of the system.

Finally, this smarter programming technique has the potential to reshape the foundations of collaborative robotics, opening new avenues for research and application in various industrial contexts. The flexibility and adaptability of the proposed method suggest a future where programming and operational efficiency in industrial robotics are significantly enhanced, leading to more advanced and seamless human-robot collaboration.

Acknowledgments

This research has been supported by the EC funded projects "ASSISTANT: Learning and robust decision Support systems for agile manufacturing environments" (Grant Agreement: 101000165) and "RENEE: Flexible remanufacturing using AI and advanced Robotics for circular value chains in EU industry".

References

[1] G. Chryssolouris, Manufacturing systems: theory and practice, 2nd ed, Springer, New York, 2006.

[2] F. Yavari, N. Pilevari, Industry revolutions development from Industry 1.0 to Industry 5.0 in manufacturing, Journal of Industrial Strategic Management 5 (2020) 44–63.

[3] S. Makris, Cooperating Robots for Flexible Manufacturing, Springer International Publishing, Berlin, Heidelberg, 2021. https://doi.org/10.1007/978-3-030-51591-1.

[4] Y. Ye, H. You, J. Du, Improved Trust in Human-Robot Collaboration with ChatGPT, (2023). https://doi.org/10.48550/arXiv.2304.12529.

[5] G. Chryssolouris, K. Alexopoulos, Z. Arkouli, A Perspective on Artificial Intelligence in Manufacturing, Springer International Publishing, Cham, 2023.

https://doi.org/10.1007/978-3-031-21828-6_4.

[6] S. Makris, K. Alexopoulos, G. Michalos, Z. Arkouli, A. Papacharalampopoulos, P. Stavropoulos, A. Fernández-Martinez, S. Muiños-Landin, K. Gadeyne, B. Meyers, P. Betinelli, F. Gosselin, C. Vienne, S. Kchir, B. Vieru, G. Gallou, M. Penalva, F. Boto, J. Outón, R. Virkkunen, ARTIFICIAL INTELLIGENCE IN MANUFACTURING White paper Prepared by the Artificial Intelligence in Manufacturing Network -AIM-NET, (2023).

[7] G. Michalos, N. Kousi, P. Karagiannis, C. Gkournelos, K. Dimoulas, S. Koukas, K. Mparis, A. Papavasileiou, S. Makris, Seamless human robot collaborative assembly – An automotive case study, Mechatronics 55 (2018) 194–211. https://doi.org/10.1016/j.mechatronics.2018.08.006.

[8] J. Ruan, Y. Chen, B. Zhang, Z. Xu, T. Bao, G. Du, S. Shi, H. Mao, X. Zeng, R. Zhao, TPTU: Task Planning and Tool Usage of Large Language Model-based AI Agents, (2023). http://arxiv.org/abs/2308.03427 (accessed September 13, 2023).

[9] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R.J. Ruano, K. Jeffrey, S. Jesmonth, N.J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, A. Zeng, Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, (2022). http://arxiv.org/abs/2204.01691 (accessed September 8, 2023).

[10] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, A. Garg, ProgPrompt: Generating Situated Robot Task Plans using Large Language Models, (2022). http://arxiv.org/abs/2209.11302 (accessed September 12, 2023).

[11] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, H. Li, Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model, (2023). https://doi.org/10.48550/ARXIV.2305.11176.

[12] K. Lin, C. Agia, T. Migimatsu, M. Pavone, J. Bohg, Text2Motion: from natural language instructions to feasible plans, Auton Robot 47 (2023) 1345–1365. https://doi.org/10.1007/s10514-023-10131-7.

[13] M. Iovino, E. Scukins, J. Styrud, P. Ögren, C. Smith, A survey of Behavior Trees in robotics and AI, Robotics and Autonomous Systems 154 (2022) 104096. https://doi.org/10.1016/j.robot.2022.104096.

[14] J. Styrud, M. Iovino, M. Norrlöf, M. Björkman, C. Smith, Combining Planning and Learning of Behavior Trees for Robotic Assembly, (2021). http://arxiv.org/abs/2103.09036 (accessed September 13, 2023).

[15] A. Lykov, D. Tsetserukou, LLM-BRAIn: AI-driven Fast Generation of Robot Behaviour Tree based on Large Language Model, (2023). http://arxiv.org/abs/2305.19352 (accessed September 8, 2023).

[16] Y. Cao, C.S.G. Lee, Robot Behavior-Tree-Based Task Generation with Large Language Models, (2023). http://arxiv.org/abs/2302.12927 (accessed September 6, 2023).

[17] K. Lotsaris, C. Gkournelos, N. Fousekis, N. Kousi, S. Makris, AR based robot programming using teaching by demonstration techniques, Procedia CIRP 97 (2021) 459–463. https://doi.org/10.1016/j.procir.2020.09.186.

[18] E. Tzavara, P. Angelakis, G. Veloudis, C. Gkournelos, S. Makris, Worker in the Loop: A Framework for Enabling Human-Robot Collaborative Assembly, IFIP Advances in Information and Communication Technology 630 IFIP (2021) 275–283. https://doi.org/10.1007/978-3-030-85874-2_29.

[19] B. Chen, F. Yi, D. Varró, Prompting or Fine-tuning? A Comparative Study of Large Language Models for Taxonomy Construction, (2023). http://arxiv.org/abs/2309.01715 (accessed November 1, 2023).

[20] L. Zhong, Z. Wang, Can ChatGPT replace StackOverflow? A Study on Robustness and Reliability of Large Language Model Code Generation, (2023). http://arxiv.org/abs/2308.10335 (accessed November 1, 2023).

[21] LangChain, (n.d.). https://python.langchain.com/docs/get_started/introduction.

[22] M. Quigley, B. Gerkey, K. Conley, ROS: an open-source Robot Operating System, IEEE International Conference on Robotics and Automation (2009).

[23] A. Koubaa, ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS, (2023). https://doi.org/10.20944/preprints202304.0827.v2.

[24] J. Bautista-Ballester, J. Vergés-Llahí, D. Puig, Programming by Demonstration: A Taxonomy of Current Relevant Methods to Teach and Describe New Skills to Robots, in: M.A. Armada, A. Sanfeliu, M. Ferre (Eds.), ROBOT2013: First Iberian Robotics Conference, Springer International Publishing, Cham, 2014: pp. 287–300. https://doi.org/10.1007/978-3-319-03413-3_21.

[25] G. Biggs, B. MacDonald, A Survey of Robot Programming Systems, (n.d.). https://www.societyofrobots.com/robottheory/Survey_of_Robot_Programming_Systems.pdf.

[26] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M.G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W.E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, B. Zitkovich, RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, (2023). https://doi.org/10.48550/ARXIV.2307.15818.