Solid Freeform Fabrication 2024: Proceedings of the 35th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

Generative Artificial Intelligence (GenAI) Prompt Engineering for Additive Manufacturing (AM)

N. A. Surovi[1,2] and P. Witherell[2]

Singapore University of Technology and Design (SUTD)[1]
National institute of Standards and Technology (NIST)[2]

Abstract

Additive manufacturing (AM) faces several challenges in achieving efficient and defect-free printing. Although traditional machine learning (ML) has proven effective in mitigating these challenges, it requires specialized models for solving specific problems with limited scopes. Generative artificial intelligence (GenAI) holds promise as a versatile tool capable of addressing multiple issues simultaneously by leveraging its expansive training data and robust problem-solving capabilities. However, getting the desired output from GenAI relies heavily on crafting effective prompts because incorrect formulation of prompts can lead to unexpected responses. Prompt engineering is crucial for GenAI models to produce desired outputs efficiently. In our study, we explore how different prompt techniques affect the responses of GenAI tools in addressing AM problems. We examine five popular prompt engineering methods: Zero-shot, Few-shot, Chain-of-thought, ReAct, and Directional Stimulus Prompting. We also use well-known GPT-4 model to get the responses across the prompt engineering methods.

Introduction

Additive Manufacturing (AM) has gained increasing popularity in both industry and academia due to its cost-effectiveness and time-saving capabilities. It offers several advantages over traditional manufacturing methods, such as the ability to create complex parts, achieve lightweight designs, and shorten production and delivery times. These advantages make AM valuable in various sectors, including aerospace, oil and gas, and offshore marine industries. However, AM faces several challenges during the transformation from design to final product. For instance, issues such as designing optimal 3D models, selecting suitable process parameters, and dealing with defects like lack of fusion, porosity, cracks, distortion, and oxidation etc. [1], [2]. These issues can impact the accuracy and quality of the final product, reducing its strength, lifespan, and performance [3], [4], [5]. Therefore, addressing these challenges is crucial for achieving efficient and defect-free printing in AM [6], [7].

Machine learning (ML) models have been instrumental in addressing various challenges in AM. These models are highly specialized and tailored to solve specific problems [7], [8]. In contrast, Generative Artificial Intelligence (GenAI) offers a promising multi-task solution and emerges as a valuable resource for addressing a broad spectrum of AM problems, from generic to complex. GenAI algorithms, trained on diverse datasets and have the potential to replace multiple specialized ML models with a single, versatile tool capable of handling various types of data and issues [8]. However, achieving consistency in responses from different GenAI tools is challenging. Slight variations in queries can result in drastically different responses, and even when using the

same query, responses can vary significantly due to differences in the architecture and training datasets of GenAI models. Our understanding of these variations in GenAI responses based on the queries provided to the models is still limited [9]. Before enhancing the capabilities of GenAI models through fine-tuning or retraining, it is crucial to understand what these models are currently capable of by querying them effectively. Crafting effective prompts is essential to reveal the fundamental limitations of existing GenAI models and identify areas where additional training might be required. This practice, known as prompt engineering, that involves developing and optimizing queries to achieve the best possible results from a given GenAI model. Effectively designed prompts are key to obtain desired outputs and improve the accuracy and relevance of responses from GenAI tools [10].

In this paper, we explore how variations in prompts impact the responses of GenAI tools in addressing AM problems. Furthermore, we investigate which prompt engineering techniques provide the best queries to obtain the most accurate responses from GenAI tools for specific AM scenarios. Our contributions are organized into three key areas: introduction of various prompt engineering methods relevant to AM, guidance on using these techniques with GenAI tools to address AM problems, and an explanation of why certain techniques are more effective than others for specific AM tasks. By focusing on these areas, our aim is to enhance the effectiveness of GenAI tools in solving AM challenges through improved prompt engineering.

Background

Challenges of AM and GenAI role

To achieve a defect-free and optimized 3D object, researchers face several challenges during the design to production phases. ML models are already proven effective to solve many of these issues. However, a specialized ML model alone is not sufficient to handle all these challenges. In this case, GenAI can play a promising role in addressing all aspects of AM challenges, as these models are trained on diverse datasets across multiple modalities. This capability allows GenAI tools to handle various data types and address a wide range of tasks in AM simultaneously. In our previous work, we proposed three types of task metrics to evaluate the effectiveness of existing GenAI tools in addressing AM challenges. These metrics were selected based on the opportunities within four exploration spaces (Figure 1). The task metrics include agnostic metrics, domain task metrics, and problem task metrics [8]. Agnostic metrics refer to baseline performance indicators used to evaluate GenAI capabilities across various AM tasks. Domain tasks metrics refer to the generic tasks or activities directly related to the specific phase within AM. Problem task metrics refer to the specific challenges in AM that require problem-solving skills in a particular AM phase, relying on established scientific principles and engineering methodologies for clear, objective solutions.

GenAI prompt engineering

GenAI algorithms can generate novel and realistic content, such as images, audio, video, and 3D models, by replicating real data distributions [10]. However, guiding GenAI models to the correct solution space is crucial. They require training on extensive datasets, though the effectiveness of this training can vary based on the selected data. Even with well-trained models,

variations in how queries are formulated can significantly influence the quality of the results [9].

Figure 1: Digital flow of AM and Generative AI dimensions in each AM phase, with selected metrics under these dimensions [8].

Prompt engineering is a powerful technique that enhances GenAI model responses by strategically crafting queries that align with the model's training and capabilities. This technique improves the precision and relevance of generated answers by formulating clear, specific, and contextually appropriate questions. By optimizing query structure, prompt engineering maximizes the utility of existing GenAI models without requiring architectural changes or extensive retraining. This method is cost-effective and efficient, focusing on leveraging the model's existing capabilities to address the complexities of specific AM tasks effectively [11].

Types of Prompt Engineering

In this paper, we explore simple and easily applicable prompt engineering techniques to address various AM problems. Our focus is on methods that do not require retrieving data from external sources, such as the Retrieval Augmented Generation (RAG) method developed by META researchers [12]. We examine five types of prompt engineering:

Zero-shot prompting [13] involves asking a GenAI model to perform a task or generate a response without providing any specific examples or context. This approach relies on the model's ability to generalize from its extensive training data to understand and respond to new, previously unseen queries.

Few-shot prompting [14] provides the GenAI model with a limited number of examples or contextual information alongside the query. This technique helps the model understand the task better and generate more accurate responses.

Chain-of-thought (CoT) prompting [15] guides the GenAI model through a series of logical reasoning steps or sub-questions to arrive at a detailed conclusion. This method helps the model

tackle complex problems by breaking them down into manageable parts. In our experiments, we use zero shot CoT [16] to keep our prompt simple and avoid complexity.

ReAct prompting [17] combines reasoning steps with directives for action, asking the GenAI model to analyze a problem and suggest practical solutions. This method is suitable for scenarios requiring both analysis and concrete actions.

Directional Stimulus Prompting (DSP) [18] guides the GenAI model's response by providing contextual clues or stimuli that influence its output towards a desired direction. This technique does not provide explicit instructions but uses hints or themes to shape the model's interpretation and answer.

Methodology

Overview of our approach

In our earlier work [8], we developed zero-shot prompts to generate responses from GenAI tools for three types of AM tasks. Since we wanted to evaluate the model's ability to solve tasks based on its prior knowledge, we did not provide any contextual clues or examples. Our results showed that zero-shot prompt engineering was sufficient for most agnostic and domain task metrics because those tasks are simple. However, for problem task metrics, zero-shot prompting often failed to produce the desired responses. Therefore, in this paper, we focus on addressing problem task metrics using five different prompt engineering techniques described in the background. As shown in Figure 2, for each problem task metric, we apply and compare these five techniques. We evaluate the responses using a precision vs. recall graph (explained in the next section) and select the prompt engineering technique that provides the best answers for each task.

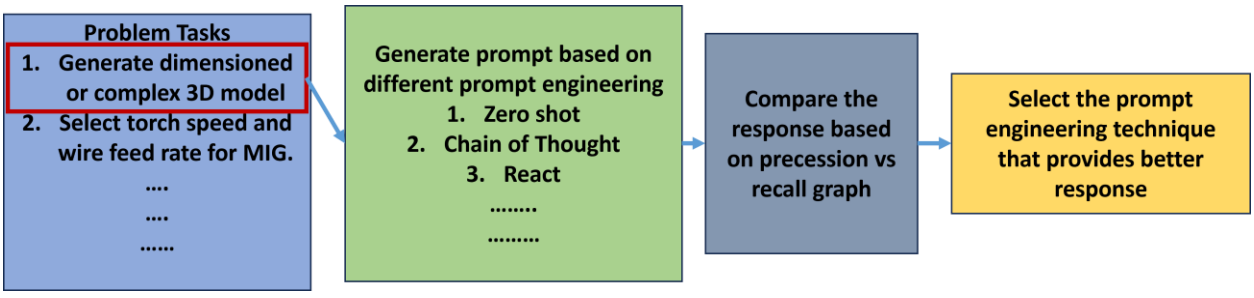

Figure 2: Overview of exploration of different prompt engineering for different AM tasks

Precision vs. Recall

We define precision and recall as follows [19]:

Recall: Measures how well the GenAI captures all the information and addresses all aspects of the query (completeness). A high recall indicates that the GenAI's response is relevant to the prompts and captures important information.
Precision: Measures the accuracy of the information provided by the GenAI (correctness). A high precision indicates that the GenAI's response closely matches the reference.

Recall vs precession relation graph:

We use a precision vs. recall graph to compare GenAI responses based on different prompt engineering techniques. Figure 3 illustrates this comparison, categorizing the responses into four categories: high precision and high recall (complete and correct), low precision and high recall (complete but not correct), high precision and low recall (correct but not complete), and low precision and low recall (neither complete nor correct). This graph helps us evaluate and select the most effective prompt engineering technique for each problem task metric.

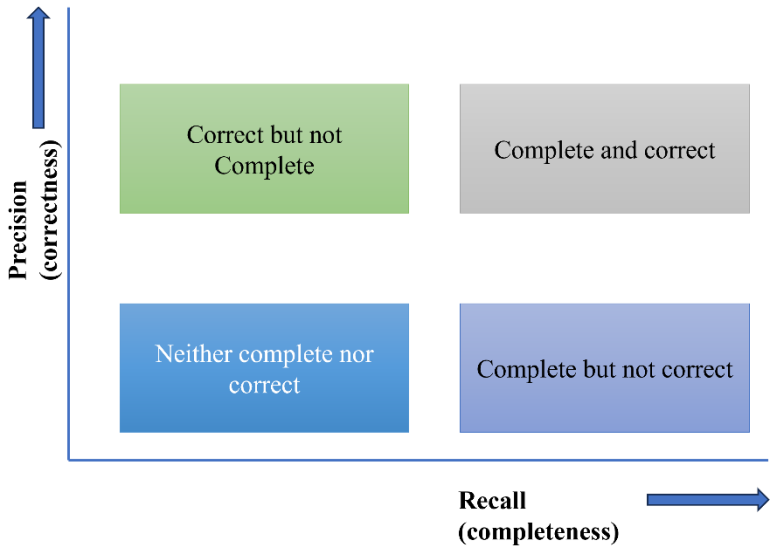

Figure 3: Precision vs recall graph based on GenAI responses.

Case Study

In this section, we demonstrate the utility of exploring different prompt engineering techniques to enhance responses for various AM tasks through case studies. These studies highlight how varying prompts can influence the outcomes generated by GenAI tools as the same prompt technique may not be effective across all AM tasks. We focus on the two specific tasks of the pre-printing phases: generating a complex 3D model (design phase) and selecting torch speed and wire feed rate for the Wire Arc Additive Manufacturing (WAAM) process (process planning phase). We use GPT-4-turbo-2024-04-09 (trained up to December 2023) with a temperature of 0 to reduce randomness in token selection.

Case 1) Design: Generating a complex-dimensioned 3D model.

We aim to create an STL file of the 3D model referenced in Figure 4. To achieve this, we explore various prompt engineering techniques, excluding few-shot prompting. Few-shot prompting requires providing examples, and since generating a dimensioned 3D model is already a complex task, adding examples would only increase the complexity.

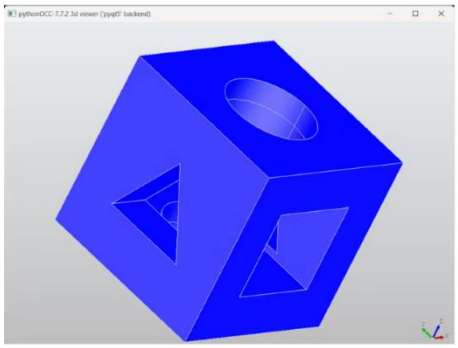

Figure 4: Reference "DemoCube" is a 40 mm sided cube featuring three distinct through bores: a 6 mm radius circular bore at the center of the xy face, a 10 mm sided square bore at the center of the xz face, and a 10 mm sided triangular bore at the center of the yz face, oriented normal to the face [20].

Zero shot

We asked ChatGPT-4 to generate a 3D model of the DemoCube in either OpenSCAD or STL format, using the measurements provided in Figure 5(a). ChatGPT-4 produced the OpenSCAD code, which is partially shown in Figure 5(b) due to its length. We then copied the script into OpenSCAD, rendered the model, and exported it as an STL file. To verify the dimensions (e.g., the cube's length, the radius of the circular bore, the length of the square bore, etc.), we used Blender software. Finally, we opened the STL file in 3D viewer software to visualize the final model, as shown in Figure 5(c).

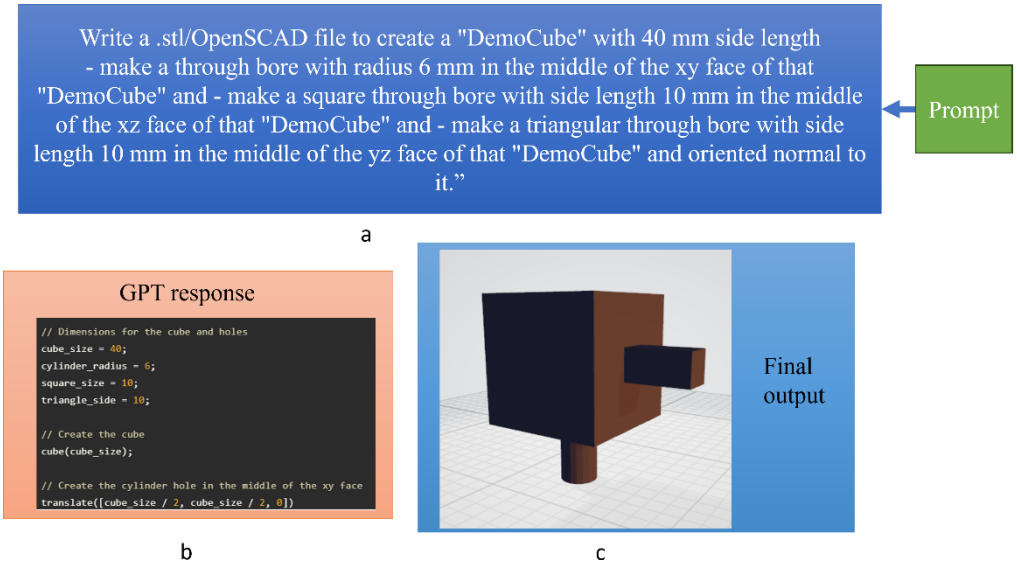

Figure 5: Zero-Shot Prompt for Generating the DemoCube.

From this experiment, we observed that while GPT-4 generated a complete 3D model, it did not match the reference exactly.

CoT

We used zero shot CoT prompting to avoid complexity. Zero shot CoT involves structuring prompts to encourage the model to break down the problem into manageable steps, explicitly articulating these steps before concluding. This method is ideal for complex tasks with multiple sub-tasks.

In our case, we used the same zero-shot prompt but added a sentence to trigger the CoT process, which significantly altered GPT-4's response. ChatGPT-4 generated the OpenSCAD code, which we rendered and exported as an STL file. The model's dimensions were verified in Blender, as done previously.

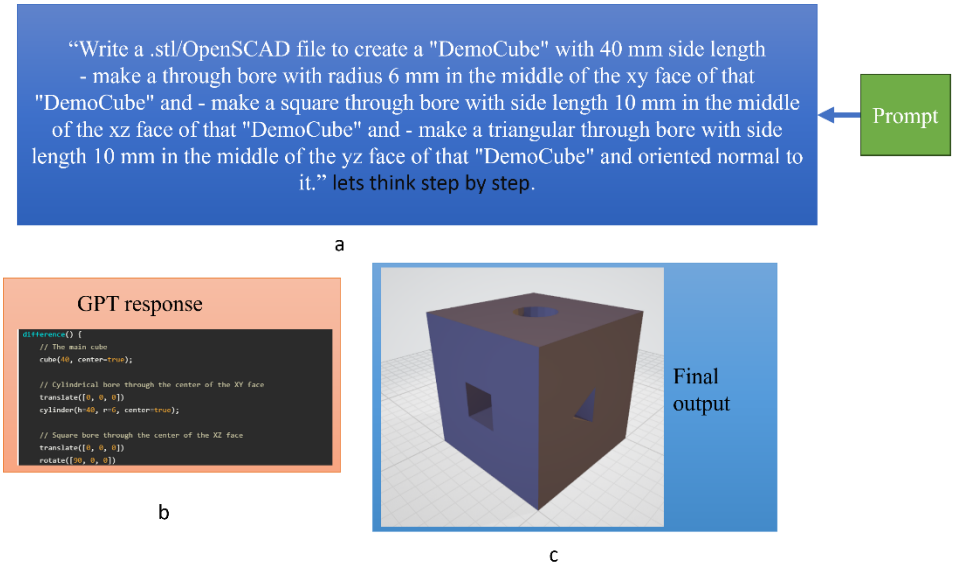

Figure 6: CoT prompt for Generating the DemoCube.

From the figure 6, we observed that the final 3D model matches the reference model based on Blender measurement. Therefore, it demonstrates that the zero shot CoT approach provides both a correct and complete response.

ReAct

To apply ReAct prompting, we used a LangChain agent to get responses from GPT-4. LangChain agents rely on the LLM's capabilities for reasoning, decision-making, processing information, drawing conclusions, and interacting with the outside world. As ReAct combines reasoning and decision-making, we crafted a prompt using a LangChain agent.

We started with the same zero-shot prompt but created a customized "general knowledge" tool and added it to the agent. We named the tool "language model" and selected the agent type as "zero-shot ReAct description," meaning the agent performs a reasoning step before acting [21] .

Figure 7(c) and Blender measurements show that the final model is close to the reference, but the triangle's position and size are slightly off and need minor adjustments. Overall, the ReAct prompt is still a good option for generating 3D models.

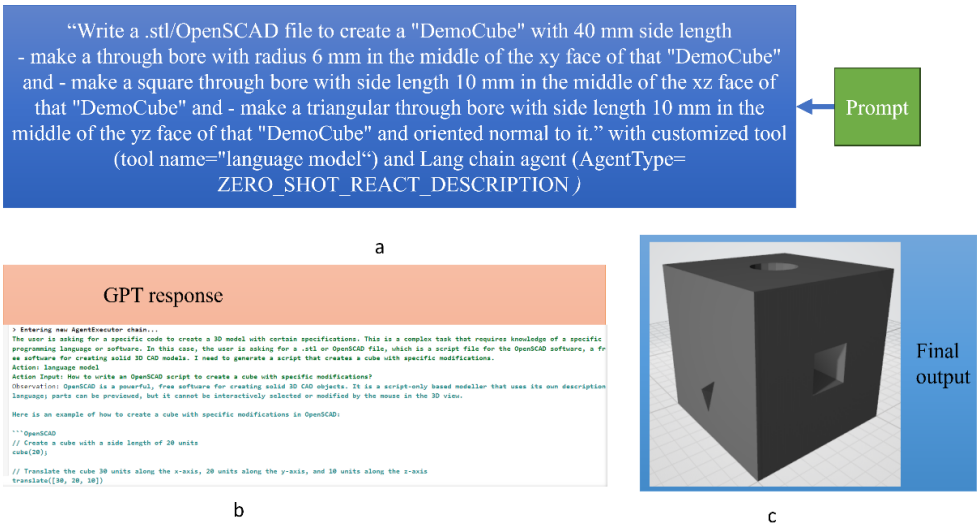

Figure 7: React prompt for Generating the DemoCube.

DSP

To use DSP, we provided a hint to guide GPT-4's response. Starting with the same zero-shot prompt, we introduced the hint to direct GPT-4's output. After receiving the response, we followed the same steps as with zero-shot prompting: copying and pasting the script into OpenSCAD, rendering the model, exporting it as an STL file, and then verifying it using Blender. Finally, we opened the STL file in a 3D viewer, as illustrated in Figure 8(c).

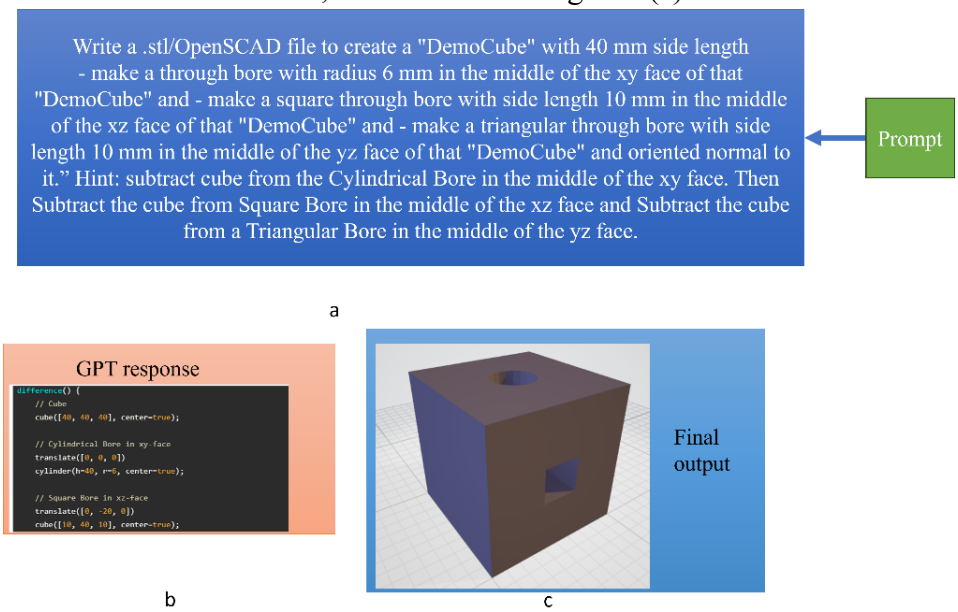

Figure 8: DSP prompt for Generating the DemoCube.

Figure 8(c) and Blender measurements show that while the code generates the 3D object, it is not entirely correct. Some modifications to the existing code are necessary to match the reference exactly. However, it is quite close to the reference.

Case 2) Process Plan: Select torch speed and wire feed rate for WAAM process.

An optimal and acceptable range for torch speed and wire feed rate in stainless steel (SS) WAAM process lies between 7 to 15 mm/s and 4 to 6 m/min, respectively [22]. Our objective is to predict these process parameters using various prompt engineering techniques.

Zero shot

We requested ChatGPT-4 to predict two process parameters (torch speed and wire feed rate) for WAAM process as shown in figure 9 (a). ChatGPT-4 predicted the good process parameters in the desired units, but they did not match to the reference values.

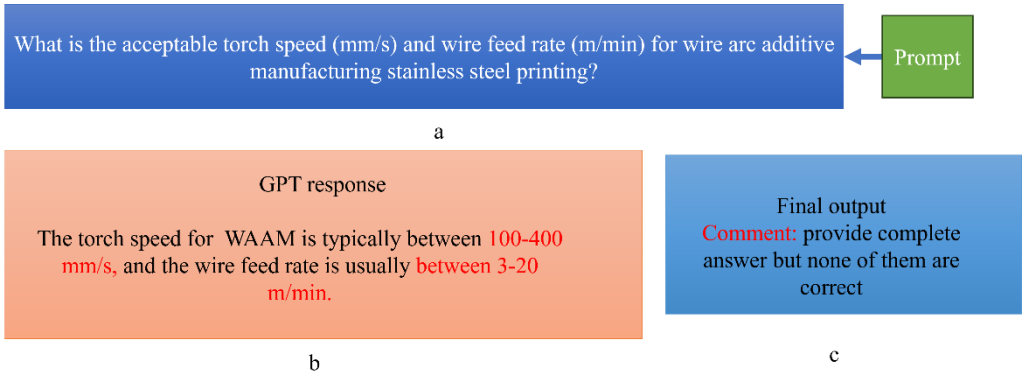

Figure 9: Zero shot prompt for predicting process parameters.

Few shot

For the few-shot prompt technique, we provided two examples for Inconel and bronze, including their process parameters (torch speed and wire feed rate) values, and then ask for the process parameters for stainless steel in the WAAM process.

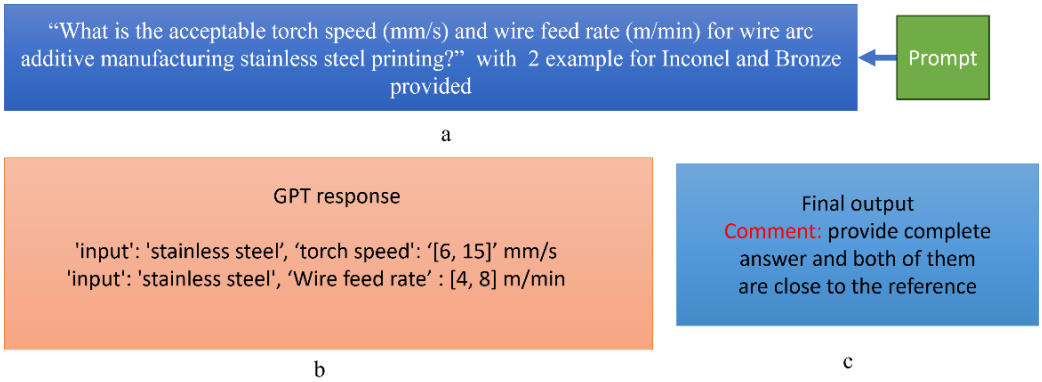

Figure 10: Few shot prompt for predicting process parameters.

We used LangChain to create the few-shot prompt because LangChain's prompt template makes the prompt structure flexible and effective for interacting with language models.

From the figure, we observe that the final output is close to the reference. Therefore, we can say that the few-shot approach provides both a correct and complete response.

CoT

In the CoT approach, we utilized the same zero-shot prompt with an additional sentence to initiate the zero shot CoT process, detailed in Figure 11 (a). However, the CoT response deviated significantly from the reference values observed in the figure 11 (b). Thus, it is evident that the CoT approach did not yield a correct or complete response.

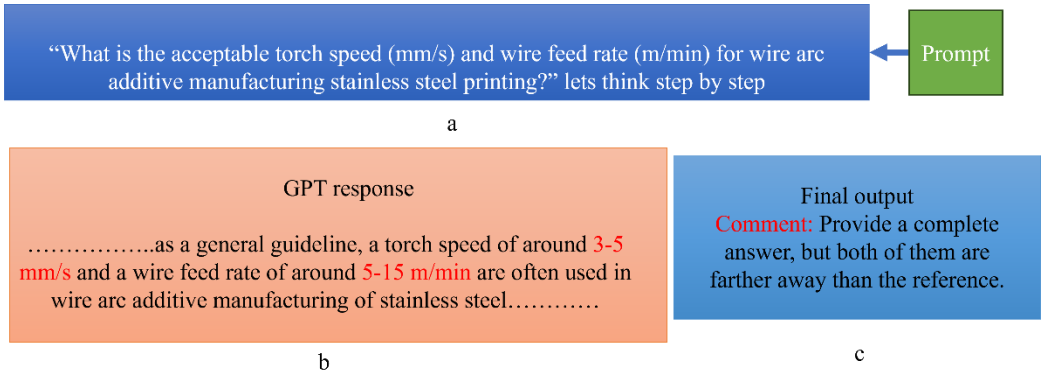

Figure 11: CoT prompt for predicting process parameters.

ReAct

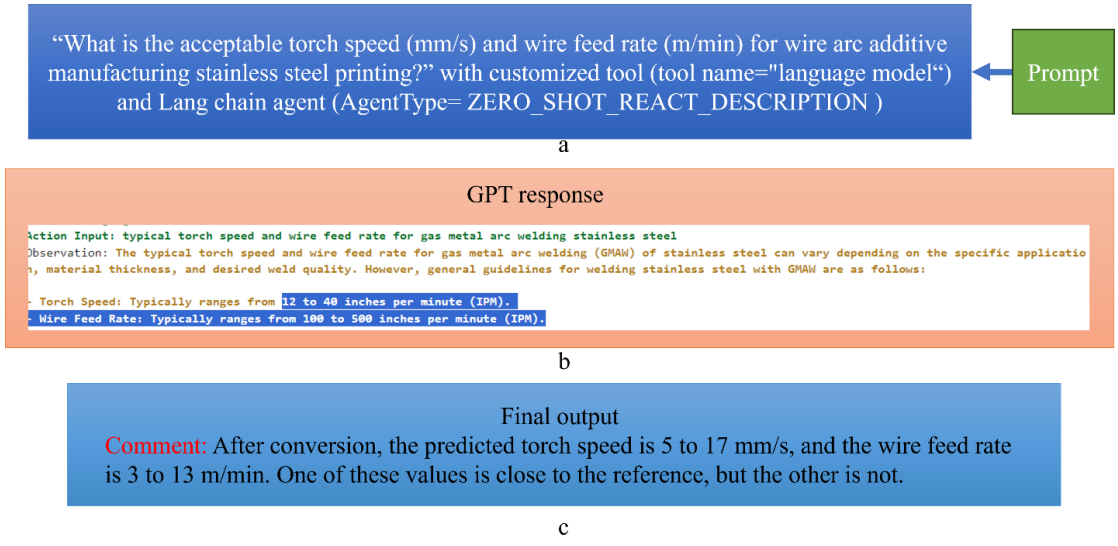

Figure 12: React prompt for predicting process parameters.

To apply ReAct prompting, we used the same LangChain agent as in Case Study 1 to get the response from GPT-4. The structure of the prompt is the same as in Case Study 1.

From the figure, we observe that ChatGPT's response includes one answer (torch speed) close to the reference and another answer (wire feed rate) not close to the reference. Therefore, we can conclude that ReAct provides a complete answer, though it may not be entirely accurate.

DSP

To use DSP, we need to give a hint that stimulates GPT-4's response like Case Study 1, as shown in Figure 13.
The figure shows that one answer is (wire feed rate) close to the reference and another answer (torch speed) is farther away from the reference. So, the DSP approach provides a complete answer, but not a fully correct one.

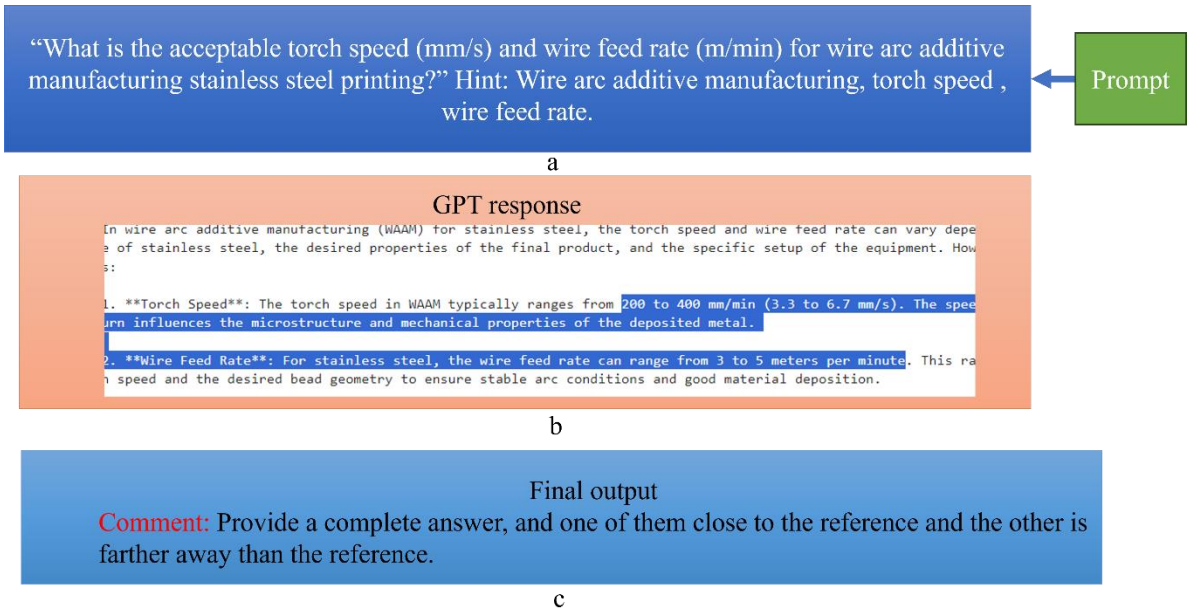

Figure 13: DSP prompt for predicting process parameters.

Discussion

The AM process is complex, and it remains unclear which specific AM challenges can be effectively addressed using GenAI tools, though ongoing research is exploring this area [8]. After analyzing the case study in this research, we concluded that the prompt techniques we explored play a crucial role in obtaining desired responses from GenAI tools for complex problems. While all techniques provided a similar level of completeness by responding to the prompts as asked, the accuracy varied depending on the specific technique used. This indicates that while GenAI can deliver comprehensive answers, the correctness of these responses may differ based on the prompting approach.

It is important to note that these case studies focused only on two AM tasks and used a single GenAI model GPT-4. Many other AM tasks and models were not covered in this study. Additionally, we did not investigate all available techniques (e.g., RAG, Relaxion, Tree of Thought) because the ones we tested were sufficient for obtaining desired responses from GPT-4.

Key findings of this paper are illustrated in the Precision vs. Recall graphs in Figure 14 and are summarized below:

1. **The effectiveness of prompt techniques varies by task:** The prompt structure should align with the task's specific requirements to generate relevant responses.

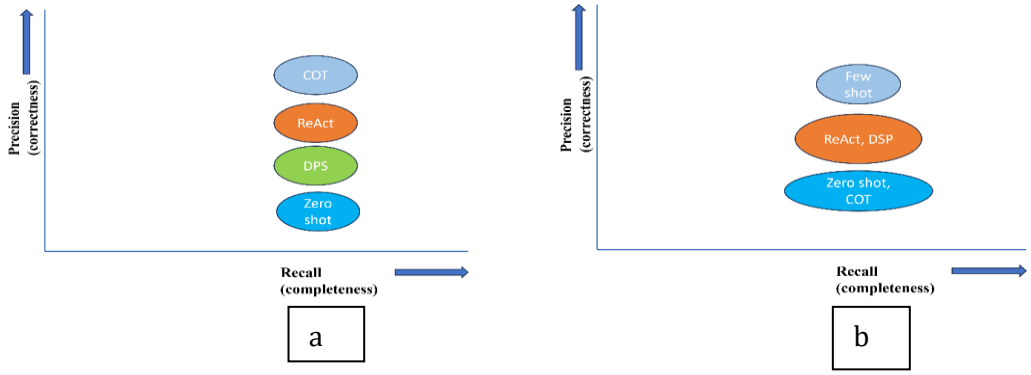

Figure 14: Precision vs Recall graph for case 1 (a) and case 2 (b).

2. **Case Study 1:** The CoT prompt technique demonstrated superiority in generating complex 3D models, while the ReAct and DSP techniques required minor modifications. All explored techniques outperformed zero-shot prompts.

3. **Case Study 2:** The few-shot prompt technique correctly predicted both process parameters, whereas ReAct and DSP predicted only one correctly, and CoT and zero-shot techniques failed to predict accurately. Again, all explored techniques outperformed zero-shot prompts. For this case study, we used only one paper for reference process parameters.

For this research, we generated responses from the model API using a single prompt instance, based on OpenAI's assurance that data sent to their API is not used to train or improve the models [23]. In previous experiments, we used the same prompt multiple times to gather responses from GenAI tools. We observed that while the line-by-line responses often differed, the overall takeaway and context remained consistent across attempts most of the time [8].

Conclusion

This paper explores the utility and effectiveness of different prompt engineering techniques for solving complex AM problems. Prompt engineering helps to improve the quality of responses from GenAI models by crafting effective questions. While simple AM problems can be addressed with straightforward prompts, complex problems require more sophisticated prompt structures to obtain accurate responses from GenAI tools. Prompt enginerring approach leverages the model's

existing knowledge, enhancing its utility for specific tasks in a cost-effective and efficient manner without retraining or modifying the model's architecture.

We investigated several prompt engineering techniques to address complex AM problems and demonstrated how different prompt structures are tailored to solve specific AM issues. Two case studies were conducted to assess ChatGPT-4's performance in responding to two phases of AM: design and process planning. We categorized different prompt techniques based on their responses, illustrated in precision vs. recall graphs for each task. Our findings indicate that a single prompt engineering technique cannot address all AM problems effectively, as each task is unique and requires prompts tailored to its specific needs and context. In the future, we plan to explore the Retrieval-Augmented Generation (RAG) prompt engineering technique because sometimes simple prompt engineering is not enough to solve more knowledge intensive AM problems that require external knowledge source to provide better response.

Acknowledgment

This research was funded by the National Institute of Standards and Technology (NIST), U.S. Department of Commerce, under the Additive Manufacturing Program. Certain commercial and third-party products are identified in this paper. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the products identified are necessarily the best available for the purpose.

References

[1] Standard, A., et al., 2012. “Standard terminology for additive manufacturing technologies”. ASTM International F2792-12a, pp. 1–9.

[2] Nowrin Akter Surovi and Gim Song Soh. “Acoustic feature based geometric defect identification in wire arc additive manufacturing”. In: Virtual and Physical Prototyping 18.1 (2023), e2210553.

[3] Frazier, W. E., 2014. “Metal additive manufacturing: a review”. Journal of Materials Engineering and performance, 23, pp. 1917–1928.

[4] N. A. Surovi, A. G. Dharmawan, and G. S. Soh, “A Study on the Acoustic Signal Based Frameworks for the Real-Time Identification of Geometrically Defective Wire Arc Bead,” *Proceedings of the ASME Design Engineering Technical Conference*, vol. 3A-2021, Nov. 2021, doi: 10.1115/DETC2021-69573.

[5] N. A. Surovi and G. S. Soh, “MULTI-BEAD AND MULTI-LAYER PRINTING GEOMETRIC DEFECT IDENTIFICATION USING SINGLE BEAD TRAINED MODELS,” 2023.

[6] N. A. Surovi, S. Hussain, and G. S. Soh, “A Study of Machine Learning Framework for Enabling Early Defect Detection in Wire Arc Additive Manufacturing Processes,” *Proceedings of the ASME Design Engineering Technical Conference*, vol. 3-A, Nov. 2022, doi: 10.1115/DETC2022-89164.

[7] N. A. Surovi and G. S. Soh, “A Heuristic Approach to Classify Geometrically Defective Bead Segments Based on Range of Curvature, Range of Sound Power and Maximum Height,” *Proceedings of the ASME Design Engineering Technical Conference*, vol. 3A, Nov. 2023, doi: 10.1115/DETC2023-114741.

[8] Nowrin Akter Surovi et al. “Current State and Benchmarking Generative Artificial intelligence for Additive Manufacturing”. In: (2024).
[9] Sabit Ekin. “Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices”. In: Authorea Preprints (2023).
[10] “Prompt Engineering Guide | Prompt Engineering Guide.” Available: https://www.promptingguide.ai/
[11] Ggaliwango Marvin et al. “Prompt Engineering in Large Language Models”. In: International Conference on Data Intelligence and Cognitive Informatics. Springer. 2023, pp. 387–402.
[12] Y. Gao *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” Dec. 2023. Available: http://arxiv.org/abs/2312.10997.
[13] Jason Wei et al. “Finetuned language models are zero-shot learners”. In:arXiv preprint arXiv:2109.01652 (2021).
[14] T. B. Brown *et al.*, “Language models are few-shot learners,” *Adv Neural Inf Process Syst*, vol. 2020-December, 2020.
[15] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: Advances in neural information processing systems 35 (2022), pp. 24824–24837.
[16] Takeshi Kojima et al. “Large language models are zero-shot reasoners”. In:Advances in neural information processing systems 35 (2022), pp. 22199–22213.
[17] S. Yao *et al.*, “ReAct: Synergizing Reasoning and Acting in Language Models,” Oct. 2022. Available: http://arxiv.org/abs/2210.03629.
[18] Z. Li, B. Peng, P. He, M. Galley, J. Gao, and X. Yan, “Guiding Large Language Models via Directional Stimulus Prompting,” *Adv Neural Inf Process Syst*, vol. 36, pp. 62630–62656, Dec. 2023.
[19] Rosanne Schoonbeek et al. “Completeness, Correctness and Conciseness of Physician-Written Versus Large Language Model Generated Patient Summaries Integrated in Electronic Health Records”. In: Available at SSRN 4835935 ().
[20] “Design any part with ChatGPT for 3D printing. - Hackster.io. Available: https://www.hackster.io/trisolarian/design-any-part-with-chatgpt-for-3d-printing-36de7f.
[21] “langchain.agents.agent_types.AgentType — 🦜🔗 LangChain 0.2.16.”Available: https://api.python.langchain.com/en/latest/agents/langchain.agents.agent_types.AgentType.html
[22] Nowrin Akter Surovi and Gim Song Soh. “Process map generation of geometrically uniform beads using support vector machine”. In: Materials Today: Proceedings 70 (2022), pp. 113– 118.
[23] “Models - OpenAI API.” Available: https://platform.openai.com/docs/models/gpt-base.