POLITECNICO DI MILANO
DEPARTMENT OF AEROSPACE ENGINEERING

POLITECNICO
MILANO 1863

Machine Learning for Mechanical Systems

AA 2024-2025

Professor: Loris Roveda

LLM-Assisted Process Planning for CNC Machining

Person Code	Surname	Name
10795356	Frassinella	Luca
10990413	Massini	Alessandro
10730683	Nuccio	Gabriele

June 8, 2025

Contents

1 Introduction

In recent years, large language models (LLMs) such as those in the GPT family have demonstrated remarkable capabilities across tasks involving reasoning, natural language understanding, and domain-specific knowledge. This progress has unlocked new forms of automation that go beyond text processing: in particular, the ability to convert natural language into structured representations or executable instructions has proven to be highly valuable [1].

This project explores the use of an LLM to automate the planning of CNC machining processes. Starting from a natural language description of a mechanical part, including its shape, dimensions, material, tolerances and other features the system is capable of:

- Extracting structured specifications from the input;
- Generating a coherent and realistic machining plan, including tools, setups, operations, and machine parameters;
- Iteratively validating the plan against geometric and technological constraints;
- Producing standard LinuxCNC-compatible G-code ready for simulation or execution.

This pipeline demonstrates how a language model can operate effectively within a highly specialized technical domain, significantly narrowing the gap between design and manufacturing. The system functions not only as an intelligent assistant for engineers and technicians, but also as a concrete step toward the end-to-end automation of CNC programming.

This report outlines the model arichitecture and assumptions, the validation mechanisms implemented, and the results achieved.

2 Problem Description and Formulation

In this section, the problem of LLM-assisted CNC process planning is cast as a constraint-driven mapping from a natural-language part description to a fully specified, executable machining plan. It defines the transformation as a series of discrete stages: specification extraction, candidate plan generation, iterative validation and refinement, and final G-code synthesis. Each of this steps is governed by logical constraints and success checks.

The specification extraction stage converts a natural language text into a data model containing part geometry (features, dimensions and tolerances), material properties, and fixture requirements. The plan generation stage then constructs an ordered sequence of machining operations.

Success is defined by the complete satisfaction of all hard constraints and the production of a consequently valid and ready-to-use machining program.

2.1 Task Definition

A map is implemented:

$$\mathcal{P} : \mathcal{D} \longrightarrow \mathcal{L}$$

where $\mathcal{D}$ is the space of natural-language descriptions of mechanical parts (e.g., "steel block with two $\varnothing 10\,\text{mm}$ holes, 50 mm deep with ± 0.1 mm tolerance"). $\mathcal{L}$, instead, is the space of structured machining plans, represented as ordered lists of operations, tooling selections, setup configurations, and parameter assignments, together with the G-code.

The goal is to ensure that the generated plan $p = \mathcal{P}(d)$ satisfies a set of logical and domain-specific constraints.

2.2 Decision Steps

The $\mathcal{P}$ is decomposed into four sequential modules:

1. **Specification Extraction**
   Converts a raw text description of the part $d \in \mathcal{D}$ into a structure $\mathbf{s}$, namely a model containing: *Part geometry*: features, nominal dimensions, tolerance bands and surface finish; *Workpiece Material* and *Fixture requirements*: workpiece orientation and working axis.

2. **Plan Generation**
   Generates an initial candidate plan $p_{\text{LLM}} = \{o_1, \ldots, o_K\}$ from $\mathbf{s}$, where each operation o_k specifies:

   - Tool selection t_k;
   - Setup configuration s_k;
   - Process parameters p_k (spindle speed, feed rate, depth of cut);
   - Operation type (e.g., roughing, drilling, finishing).

3. **Validation & Refinement**
   To validate and refine the plan a series of constrains is here presented:

   - **Geometric** (C_{geom}): on setup, orientation and tool choice;
   - **Technological** (C_{param}): parameters within machine and tooling limits;
   - **Tolerance** (C_{tol}): finishing passes meet specified tolerances.

   Any violation triggers localized refinement via prompt engineering of the LLM until all the constraints are satisfied.

   A plan $p = \mathcal{P}(d)$ is acceptable if and only if

   $$C_{\text{geom}}(p) \;=\; C_{\text{tech}}(p) \;=\; C_{\text{tol}}(p) \;=\; \text{true}.$$

4. **G-Code Synthesis**
   Serializes the validated plan p^* into a generic compatible G-code file, ready for a CNC simulation or execution. The output consists compatible G-code program that translates the validated machining plan into executable instructions. This G-code is not limited to standard geometric commands (G-codes) such as linear moves or drilling cycles, but also integrates tool management (via T-codes) and machine control operations (via M-codes), ensuring a complete and realistic simulation or execution on a CNC machine [2] [3].

   Each machining operation begins with a tool change instruction using a structured tool code `TxxY`, where `xx` represents the tool type and `Y` the size class (small [1], medium [2], big [3]). The corresponding `M6` command initiates the tool change. Once the tool is active, spindle and coolant functions are controlled through commands such as `M3` (spindle on), `M5` (spindle off), `M8` (coolant on), and `M9` (coolant off), allowing for accurate cycle preparation and safe transitions between operations.

   Movement and machining actions are expressed through G-codes. These include rapid and feed moves (`G0`, `G1`), drilling cycles (`G81`), peck drilling (`G83`), tapping (`G84`), and boring (`G85`), among others. Each template includes placeholders for parameters like depth, feedrate, retraction height, and spindle speed, which are filled dynamically based on the operation plan generated by the LLM. The G-code is also annotated with meaningful comments for readability and debugging during simulation.

2.3 Data Repositories and Parameter Ranges

The system relies on two primary repositories: *Tool Database* and *Material Database*, which define the domain knowledge for plan generation and validation. The *Tool Database* contains each tool's geometries and includes a quite comprehensive set of manufacturing tools used in CNC machining operations. The database associates also each tool to a specific operation, to the compatible materials and to a unique T-code:

Tool Name	Type	Code	Small Ø(mm)	Medium Ø(mm)	Big Ø(mm)
drill bit [4]	drill	01	1–4	5–8	9–20
end mill [5]	mill	02	1–4	5–10	11–20
face mill	mill	03	1–20	21–50	51–100
chamfer mill	mill	04	1–2	3–4	5–10
ball nose end mill	mill	05	1–4	5–8	9–12
slot drill [4]	mill	06	2–5	6–10	11–20
reamer [6]	finish	07	1–3	4–6	7–12
boring bar [7]	boring	08	5–10	11–20	21–40
lathe insert	turning	09	2–5	6–10	11–20
grooving tool	groove	10	1–3	4–6	7–10
tapping tool	threading	11	2–4	5–8	9–12
deburring tool	finish	12	1–3	4–6	7–10
carbide insert	multi-purpose	13	1–4	5–8	9–12

Table 1: Tool Database Overview

Tolerance and surface finish ranges are also included in the database based on each process capabilities.

The *Material Database*, instead, specifies workpiece general materials and recommended machining parameters (Spindle Speeds and Feed Rates), which have been defined by combining empirical values commonly used in CNC machining with standard recommendations found in technical references and manufacturer data sheets.

Material	Spindle Speed (RPM)	Feed Rate (mm/min)
Aluminum [8]	3 000–12 000	300–2 500
Steel [9]	600–2 000	80–300
Stainless Steel [9]	400–1 200	60–200
Brass [9]	1 000–3 000	150–600
Titanium [10]	300–900	20–120
Plastic [11]	2 000–6 000	400–1 000
Copper [10]	800–2 500	100–400
Carbon Fiber [11]	5 000–15 000	100–500
Cast Iron [9]	200–700	30–120
Inconel [12]	300–1 000	20–100

Table 2: Material Database: Spindle Speed and Feed Rate Ranges

These tables are used by the LLM to constrain its suggestions and by the validation modules to verify feasibility. In addition to the tool and material databases, the system also includes a structured G-code library, organized by machining operation. Each template contains parameterized blocks (e.g., for drilling, face milling, or chamfering) that the LLM can reference to generate executable and syntactically correct CNC instructions.

2.4 LLM Characteristics

The core of the planning relies on a pretrained large language model (LLM) accessed via API. The following properties and inference parameters have been selected to balance determinism, coverage of complex prompts, and response coherence:

– **Model variant:** OpenAI GPT-o4-mini. This variant was selected as a cost-effective compromise between sufficient context capacity, enough to encode the full part description, and economical API usage.

– **Temperature:** set to 0.3. A low temperature enforces near-deterministic decoding, reducing variability in tool selection and operation sequence, which is critical for reproducible machining plans.

– **Maximum tokens:** capped at 1 000 for plan generation calls. This limit was chosen to control API costs while still accommodating typical plan outputs: concise JSON-style operation lists generally require 700–900 tokens.

– **Prompt engineering:** The prompt engineering strategy adopted in the `refine_plan` function is central to the system's ability to iteratively guide the LLM toward producing valid CNC plans. Rather than relying on a single static prompt, the system dynamically constructs and adjusts the input to the model based on constraint violations observed during validation.

  Each prompt is composed of several structured components. First, a *system message* encodes domain-specific knowledge: including constraints from the tool and material databases, formatting rules, and validation expectations (e.g., all operations must have complete cutting parameters, tool–operation compatibility must be respected, and the setup step must appear first). This ensures the model is fed with realistic assumptions about manufacturing workflows.

  In addition, the function appends a block of constraints including acceptable ranges for spindle speed, feed rate, and, where applicable, tolerance and surface finish requirements. These constraints are formulated as clear bullet points, e.g., "Spindle Speed must be between 300 and 1000 RPM," giving the model explicit numeric targets.

Determinism (low temperature) ensures consistent machining sequences across repeated runs, satisfying a fundamental requirement for industrial process reproducibility. The token cap on plan generation balances the need for complete operation lists against response latency and cost considerations. Finally, prompt-level rule encoding set the model's output format, reducing the weight on downstream validation and parsing modules.

2.5 Model Assumptions

To ensure tractability under API-only access and limited compute, the following simplifications are adopted:

– **Rigid bodies**: both workpiece and cutting tools behave as perfectly rigid bodies, so elastic and thermal deformations and dynamic deflections are neglected.;

– **Fixed tool library and setups**: only a predefined set of tools with dimensions range is considered;

– **Stateless LLM calls**: the model is used in a prompt-and-validate way without fine-tuning implementation;

– **Feature independence:** features (holes, pockets, chamfers) are treated as if they do not interact.

3 Simulation Workflow

3.1 Input

To initiate the planning process, the system accepts a concise and human-readable natural language input that describes the mechanical part to be machined. The choice to use natural language, rather than structured forms, reflects a precise goal: make the system accessible and intuitive for operators and engineers.

The description is then passed to a parsing function, `generate_part_spec_from_description`, which extracts a structured representation of the part's key manufacturing requirements. The function performs targeted regular expression matching and keyword lookup to infer:

– **Material**: selected from the predefined database in Section 2.3;

– **Dimensions**: parsed from the pattern `LxWxH mm` and assigned to the canonical machine axes (X, Y, Z);

– **Geometry**: classified using semantic keywords (e.g., block, plate, shaft);

– **Features**: such as holes and chamfers, including relevant parameters like diameter, depth, chamfer size, and axis of execution;

– **Tolerances and surface finish**: extracted using numeric pattern matching (e.g., `±0.1 mm`, `Ra < 2 µm`).

The output of this function is a structured Python dictionary containing all the extracted specifications. This serves as the canonical input for the downstream LLM prompt, which uses the extracted data to generate a complete machining plan.

3.2 LLM call - Guided Plan Generation and Validation

Once the part specification is defined, the system proceeds to generate a machining plan by calling the LLM through a structured and context-rich prompt. This is handled by the `generate_prompt` function, which encapsulates both technical expectations and formatting constraints. The prompt is framed as an instruction to an expert CNC process planner, providing a series of tailored instructions and an example step to ensures that the LLM response is syntactically uniform and semantically rich, facilitating downstream parsing and validation. Notably, each step is expected to include both machining and contextual information, such as orientation and workholding. The output of each generation cycle is parsed with the `parse_plan` function into a list of discrete steps (e.g., setup, drilling, chamfering), each annotated with tool selection, cutting parameters, workholding, and orientation. This candidate plan is then passed to the `validate_plan` function, which performs a comprehensive check against:

– tool–operation compatibility (e.g., chamfering requires a chamfer mill);

– material parameter ranges (spindle speed and feed rate);

– geometric feasibility (e.g., cut depth not exceeding part dimensions);

– surface finish and tolerance bounds based on process capability;

– completeness of each step (e.g., no missing tools or orientations).

Any violations are collected and encoded into a structured list of error messages.

3.3 LLM Call - Iterative Refinement

If validation fails, the system enters a feedback loop. The errors are translated into natural language hints and suggestions which are automatically appended to the next prompt. These include:

– corrective instructions on orientation and parameters;

– tool incompatibility or replacement suggestions;

– surface finishing and tolerance suggestion;

– missing information alerts.

The core of this process is implemented in the `refine_plan` function, which iteratively queries the model until a valid CNC plan is produced or a maximum number of refinement steps is reached.

At each iteration, the LLM receives the input described in the previous section. The prompt is carefully crafted to provide not only explicit numeric limits, but also safety considerations for difficult materials like titanium or inconel. This design ensures that the LLM has access to both hard constraints and qualitative knowledge. This feedback mechanism enables the LLM to iteratively refine its plan with progressively stronger constraints and clearer guidance. The simulation aims to find a valid plan in maximum 7 iterations and often converges in less than five iterations, and is robust to ambiguous or underspecified initial inputs.

Furthermore, certain violations are automatically corrected without regenerating the full plan. An example is linked to the maximum depth of cut for a specific operation which is auto-corrected if it overcomes the length of the piece along the specified working axis. These corrections are logged and appended to the step descriptions as annotations (e.g., `[Auto-corrected depth to 30 mm]`), providing transparency in the planning process and ensuring traceability.

3.4 Output

After validating and refining the call, the final machining plan is rendered using the `display_plan` function. This function transforms the plan into a human-readable and structured table with the following fields: Operation type and Tool used, parameters such as spindle speed, feed rate, depth of cut, Orientation and Workholding method, necessary for physical setup and Working Plane and Cut Axis, mapped from the part geometry.

Additionally, if any auto-corrections (e.g., depth limits) were applied during validation, they are highlighted inline in the depth column to maintain transparency. The plan is also printed in text format before the table, preserving the LLM's formatting and annotations. This dual representation facilitates review by both engineers and automated post-processors.

Once the machining plan has been validated, the system automatically converts it into executable G-code. This final output serves as a ready-to-run program that can be simulated or directly executed on a CNC machine, completing the fully automated pipeline from language input to actual machine instructions.

4 Results

To evaluate the effectiveness and correctness of the proposed pipeline, this section presents the results of a complete CNC process planning cycle on a representative test case. The process begins from a natural language description of a mechanical part and proceeds through all the stages described in the previous sections.

The goal of this simulation is twofold: first to illustrate the kind of structured output the system is capable of producing from unstructured language input, and second to validate that the resulting G-code is both logically coherent and compatible with standard CNC machining interpreters.

The example is selected to contain multiple machining features (e.g., holes, pockets, chamfers) requiring a defined surface roughness and a required tolerance, allowing for a rich demonstration of the model's reasoning capabilities and the expressiveness of the output format.

For this demonstration, the input provided is:

```
Block 80x50x10 mm in aluminum with two Ø5 mm holes, 15 mm deep, along Z-axis,
and a 2 mm chamfer on the top edges.  The part must meet a tolerance of ±0.1
mm and a surface finish of Ra < 2 µm.
```

Below is the validated machining plan produced by the LLM, following the iterative refinement process described earlier. The plan includes five operations in sequence, each with detailed parameters and orientation information. One automatic correction was applied to the drilling depth, as noted in step 3.

1. **Operation**: Setup
   Tool: N/A
   Parameters:
   - Workholding: Secure the workpiece using a precision vice with soft jaws to prevent damage to the aluminum surface.
   - Orientation: Top face up, aligning the length along the X-axis.
2. **Operation**: Face Milling
   Tool: 50mm Face Mill
   Parameters:
   - Spindle Speed: 4000 RPM
   - Feed Rate: 1000 mm/min
   - Depth of Cut: 2 mm
   - Coolant: Flood coolant to manage heat and improve surface finish.
   - Orientation: Top face up, machining the top surface to achieve flatness.
3. **Operation**: Drilling
   Tool: 10mm Twist Drill Bit
   Parameters:
   - Spindle Speed: 3000 RPM
   - Feed Rate: 600 mm/min
   - Depth of Cut: 15 mm (full depth) [Auto-corrected depth to 10 mm]
   - Coolant: Flood coolant to reduce chip buildup and heat.
   - Orientation: Top face up, drilling two holes along the Z-axis.
4. **Operation**: Chamfering
   Tool: Chamfer Mill (10mm)
   Parameters:
   - Spindle Speed: 4000 RPM
   - Feed Rate: 800 mm/min
   - Chamfer Angle: 45 degrees
   - Depth of Cut: 2 mm (to achieve a 10mm chamfer size)
   - Coolant: Flood coolant to improve surface finish.
   - Orientation: Top face up, chamfering the top edges of the block.
5. **Operation**: Finishing Pass
   Tool: Polishing Pad
   Parameters:
   - Spindle Speed: 3000 RPM
   - Feed Rate: 500 mm/min
   - Orientation: Top face up, finishing all surfaces to achieve Ra $\leq$ 2 µm.
   - Note: Ensure all surfaces are polished to meet the required surface finish.

This machining plan includes all necessary operations to manufacture the aluminum block with the specified features and tolerances. Each operation is carefully sequenced to ensure efficiency and accuracy while adhering to the material constraints and required surface finish.

index	Operation	Tool	Spindle Speed	Feed Rate	Workholding	Orientation	Working Plane	Cut Axis	Depth
0	Setup	na	NaN	NaN	Secure the workpiece using a precision vice with soft jaws to prevent damage to the aluminum surface.	Top face up, aligning the length along the X-axis.	XY		
1	Face milling	face mill (small)	4000.0	1000.0		Top face up, machining the top surface to achieve flatness.	XY	Z	2
2	Drilling	twist drill bit	3000.0	600.0		Top face up, drilling two holes along the Z-axis.	XY	Z	10 ⚠ (corrected)
3	Chamfering	chamfer mill	4000.0	800.0		Top face up, chamfering the top edges of the block.	XY	Z	2
4	Finishing pass	polishing pad	3000.0	500.0		Top face up, finishing all surfaces to achieve Ra < 2 µm.	XY		

Figure 1: Output Parsed Table

The following G-code was automatically generated from the validated machining plan.

Step 1: Setup

```
; Setup:  configure units and positioning
G21 ; Set units to millimeters
G90 ; Absolute positioning
G17 ; Select XY plane
G94 ; Feedrate per minute (units/min)


; Workholding:  Secure the workpiece using a precision vice with soft jaws to prevent
damage to the aluminum surface.
; Orientation:  Top face up, aligning the length along the X-axis.
```

Step 2: Face Milling

```
; Face Milling
T031 M6 ; Tool change (select and load tool)
M3 S4000 ; Spindle ON clockwise (start rotation)
M8 ; Coolant ON (flood coolant)
G0 Z5 ; Rapid to safety height
G0 X0 Y0 ; Move to start position
G1 Z-2 F1000 ; Plunge to depth
G1 X80 Y0 ; First pass
G1 X80 Y50 ; Second pass
G1 X0 Y50 ; Third pass
G1 X0 Y0 ; Final pass
G0 Z5 ; Retract
M5 ; Spindle OFF (stop rotation)
M9 ; Coolant OFF
```

Step 3: Drilling

```
; Drilling
T012 M6 ; Tool change (select and load tool)
M3 S3000 ; Spindle ON clockwise (start rotation)
M8 ; Coolant ON (flood coolant)
G0 Z5 ; Rapid to safety height
G0 X20 Y25 ; Move to hole position (estimated coordinates)
G81 Z-10 R5 F600 ; Drilling cycle
G0 Z5 ; Retract
G80 ; Cancel drilling cycle
M5 ; Spindle OFF (stop rotation)
M9 ; Coolant OFF
```

Step 4: Chamfering

```
; Chamfering operation for aluminum block
G21 ; Set units to millimeters
G90 ; Absolute positioning
G17 ; Select XY plane
G94 ; Feedrate per minute (units/min)

T042 M6 ; Tool change (select and load chamfer mill medium)
M3 S4000 ; Spindle ON clockwise (start rotation)
M8 ; Coolant ON (flood coolant)

G0 Z5 ; Rapid to safety height
G0 X0 Y0 ; Move to chamfer start at top left corner of the block

; Chamfering the top edges of the block
G1 Z-2 F800 ; Engage cut at depth of 2 mm
G1 X80 Y0 ; Chamfer path along the top edge
G1 X80 Y50 ; Chamfer path down the right edge
G1 X0 Y50 ; Chamfer path along the bottom edge
G1 X0 Y0 ; Chamfer path up the left edge

G0 Z5 ; Retract to safety height
M5 ; Spindle OFF (stop rotation)
M9 ; Coolant OFF
```

Step 5: Finishing Pass (Polishing)

```
; Polishing Pass for finishing aluminum block
G21 ; Set units to millimeters
G90 ; Absolute positioning
G17 ; Select XY plane
G94 ; Feedrate per minute (units/min)

; Tool change and spindle setup
M6 ; Tool change (select and load polishing tool)
M3 S3000 ; Spindle ON clockwise (start rotation)
M8 ; Coolant ON (flood coolant)

; Safety height and start position
G0 Z5 ; Rapid to safety height
G0 X0 Y0 ; Move to polishing start position
; Engage polishing surface
G1 Z-0.1 F300 ; Engage polishing surface at 0.1 mm depth
G1 X80 Y0 ; Move to the end of the first pass
G1 X80 Y50 ; Move to the end of the second pass
G1 X0 Y50 ; Move to the end of the third pass
G1 X0 Y0 ; Return to start position
; Retract and finish
G0 Z5 ; Retract to safety height
M5 ; Spindle OFF (stop rotation)
M9 ; Coolant OFF
```

```
M30 ; End of program
```

5 Statistical Analysis

To evaluate the robustness of the planning system, a large-scale has been conducted statistical analysis on 100 randomly generated part descriptions, covering a wide range of geometries, materials, and manufacturing features. For each input, we tracked whether a valid process plan was produced and, if not immediately, how many iterative LLM refinement steps were required to achieve validation. This allowed the team to assess not only the overall success rate of the system but also the number of LLM calls typically needed to produce a valid and constraint-compliant CNC process plan.

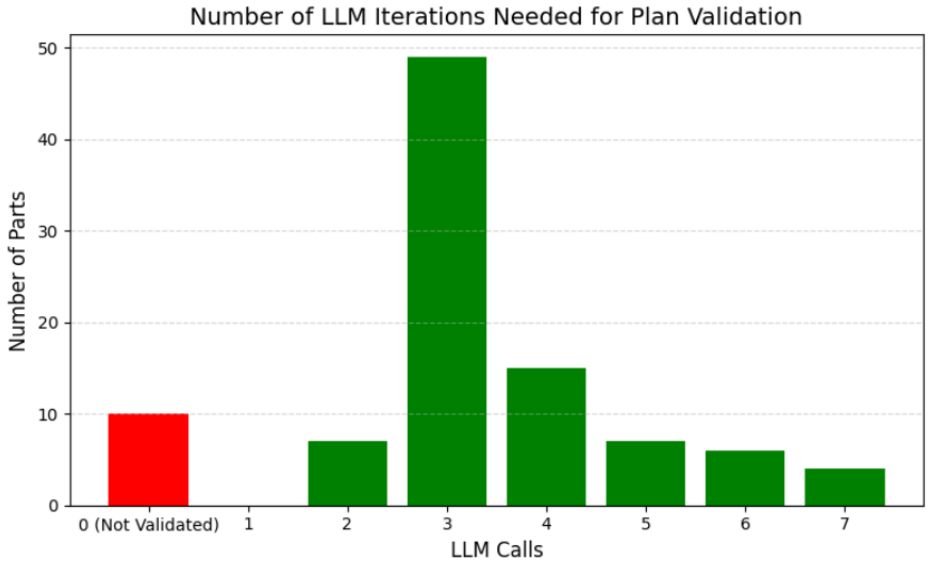

Figure 2: Statistical Analysis

The bar chart illustrates how many LLM iterations were required to obtain a valid CNC process plan across 100 test cases. As it shown in Figure 1 only the 10% of random plans are not validated after the maximum number of iterations. Moreover, no parts were successfully validated on the first attempt, while the 90% of random plans were successfully validated after three or more iterations. This clearly highlights the importance of the iterative loop in driving the validation process. Prompt engineering is, in fact, crucial to satisfy the imposed constraints and thus validate the plan converging towards feasible solutions.

6 Closing Comments and Further Developments

The system presented in this work demonstrates the feasibility and effectiveness of using a large language model to automate the generation of CNC machining plans from natural language descriptions. Throughout the development of this project, the integration of a large language model proved to be both highly effective and, at times, unpredictable. The LLM consistently demonstrated strong capabilities in generating coherent machining plans, correctly selecting tools, and adhering to basic geometric and technological constraints when properly guided through structured prompts and finally generate post-processed G-code ready for real world simulation or execution. In particular, its ability to generalize from examples and adapt to changes in material, geometry, and feature type made it an extremely helpful component in automating what is traditionally a manual and expert-driven task.

However, human oversight remained essential in several key areas. First, the model occasionally misunderstand tool names or produced inconsistent formatting when not explicitly constrained by validation logic. Second, in early iterations, plans often missed setup steps, lacked workholding details, or assigned wrong cutting parameters, especially when the part description was ambiguous or underspecified. These issues were mitigated through the introduction of a refinement loop and explicit error feedback, but still highlight the need for a human-in-the-loop validation around any full LLM-driven system.

Overall, while the LLM served as a powerful generator of content, it required a structured context, careful prompt engineering, and post-generation control mechanisms to ensure reliability. The statistical analysis, finally, confirms the robustness of the approach. Across a random span of input descriptions, materials, and machining features, 90% of plans were successfully validated.

Looking ahead, several directions can extend and enhance this work. First, the system could be extended to accept 3D CAD models as input, allowing automatic extraction of part geometry and features directly from the shape. Second, the generated G-code could be passed to an external simulator, such as CAMotics or Fusion360, to visually evaluate the CNC process efficiency. Third, the testing set could be expanded to include more complex, multi-face, or multi-setup parts, as well as features that are particularly challenging to machine. Lastly, although the current system uses a general-purpose LLM, future versions could benefit from fine-tuning on domain-specific datasets, such as annotated machining logs or industrial documentation, to better align the model's reasoning with this specific application and to perform better on specialized tasks. However this latter improvement will require significant resources for training and deployment. In conclusion, this work demonstrates how the integration of language models with engineering logic and domain constraints can enable a new generation of intelligent, adaptive, and accessible CNC programming tools, paving the way for future fully-automated implementations in machining control and manufacturing.

References

[1] N. L. Fanlong Zeng Wensheng Gan, Yongheng Wang and P. S. Yu, "Large Language Models for Robotics: A Survey."

[2] "Linux G-Codes." [Online]. Available: https://linuxcnc.org/docs/html/gcode/g-code.html

[3] "Gcode - Marlin Firmware." [Online]. Available: https://marlinfw.org/meta/gcode/

[4] "CNC Machining Part Size: Limitations, Design and Values." [Online]. Available: https://www.3erp.com/blog/cnc-machining-part-size/

[5] "End Mill Size Chart." [Online]. Available: https://www.6gtools.com/technical-info/end-mills/end-mill-sizes.html?srsltid=AfmBOopB_LW90O_eHIOeW66kycX_w4rWU0tIiJb00WGABvptf7-tA2zf

[6] "Reamers — Reaming Tools for all diametersCNC Machining Part Size: Limitations, Design and Values." [Online]. Available: https://guhring.com/BrowseProducts/Products/Reamers

[7] "HOW TO READ THE STANDARD OF BORING BARS." [Online]. Available: https://data.mmc-carbide.com/1916/9575/0325/catalog_c010a_boring.pdf

[8] "Aluminum CNC Milling Feeds and Speeds: Impact Factors and Optimizing Tips." [Online]. Available: https://www.jtrmachine.com/aluminum-cnc-milling-feeds-and-speeds-impact-factors-and-optimizing-tips?

[9] "Cutting Speeds." [Online]. Available: https://littlemachineshop.com/reference/cuttingspeeds.php?srsltid=AfmBOorgn6xfoTY-k0C20mExziK_CPQPBPvV9eUdJn73lptq4bAv_9Pi&

[10] "CNC Machining." [Online]. Available: https://www.jtrmachine.com/

[11] "Carbon Graphite and Carbon Fiber Panel CNC Cutting Solid Carbide Router Bits Speed and Feed Chart." [Online]. Available: https://www.amanatool.com/pub/media/custom/upload/File-1436543087.pdf?

[12] "CNC Machining Inconel: Material Characteristics, Challenges, and Uses." [Online]. Available: https://flyingprecision.com/cnc-machining-inconel/