Interactive GUI for .cli tool path visualization and gaussian thermal overlay in additive manufacturing

July 17, 2025

---

1 Introduction

Additive manufacturing (AM), commonly known as 3D printing, has revolutionized modern manufacturing with its ability to fabricate complex geometries from digital models. Among its methods, laser-based powder bed fusion (PBF) stands out for high precision. This process relies on scan instructions encoded in `.cli` files to direct the laser tool path for each layer.

This project introduces a Python-based GUI that parses `.cli` files, visualizes scan paths layer-wise, and overlays thermal input using a Gaussian heat source model. Such visualization aids in understanding thermal gradients, residual stresses, and energy distribution.

Python was chosen for its scientific computing strengths, with Tkinter for GUI design and Matplotlib for visualization.

2 Background and Literature Context

`.cli` (Common Layer Interface) files are used in AM to define layer-wise scan instructions, including metadata like power, speed, and focus [1]. Accurate interpretation helps predict heat-affected zones and optimize process parameters.

The Gaussian heat source model is widely used due to its simplicity and practical approximation of real laser behavior. Though more advanced models like Goldak's double ellipsoid exist, they are computationally intensive.

Open-source tools in Python offer affordable and customizable alternatives to commercial AM software [2].

3 System Design and Architecture

The system is divided into three core components: parser module, heatmap model, and GUI rendering interface.

Parser Module

This module reads `.cli` files line-by-line and extracts data tags like `$$LAYER`, `$$HATCHES`, `$$POWER`, `$$SPEED`, and `$$FOCUS`. It organizes data into a structured dictionary by layers and scales coordinates using the `$$UNITS` tag.

```
with open(filepath, 'r') as file:
    for line in file:
        line = line.strip()

        if line.startswith("$$UNITS/"):
            units = float(line.split("/")[1])

        elif line.startswith("$$LAYER/"):
            current_layer = int(line.split("/")[1])
            layers[current_layer] = {'paths': [], 'power': None, 'speed': None, 'focus': No

        elif line.startswith("$$POWER/") and current_layer is not None:
            layers[current_layer]['power'] = float(line.split("/")[1])

        elif line.startswith("$$SPEED/") and current_layer is not None:
            layers[current_layer]['speed'] = float(line.split("/")[1])

        elif line.startswith("$$FOCUS/") and current_layer is not None:
            layers[current_layer]['focus'] = float(line.split("/")[1])

        elif line.startswith("$$HATCHES/") and current_layer is not None:
            parts = line.split(",")[2:]  # Skip $$HATCHES and count
            coords = list(map(int, parts))
            points = [(coords[i] * units, coords[i + 1] * units) for i in range(0, len(coor
```

Figure 1: File Parser (Source: Obtained from Jupyter Notebook)

Heatmap Model

The heatmap model simulates Gaussian thermal input centered around scan paths. Each point's intensity is computed using a 2D Gaussian kernel based on scan path center and user-defined σ.

```
def gaussian_heatmap(x, y, x0, y0, sigma=0.4):
    return np.exp(-((x - x0) ** 2 + (y - y0) ** 2) / (2 * sigma ** 2))
```

Figure 2: Heatmap Model (Source: Obtained from Jupyter Notebook)

GUI Rendering

Tkinter creates a user-friendly interface, allowing file selection, layer navigation, and interactive plotting. Matplotlib plots are embedded in the window with real-time updates.

4 Implementation Details

The workflow begins with the `parse_cli()` function, which extracts and stores hatch data and metadata. The GUI allows users to upload `.cli` files and select layers via

dropdown.

The `gaussian_heatmap()` function uses NumPy to simulate a 2D Gaussian heat distribution. Visual output is rendered using `contourf()` from Matplotlib.

Modular functions ensure extensibility, and error-handling blocks manage missing or malformed data gracefully.

5 Features and User Interaction

Key features of the application include:

- File upload dialog for `.cli` files.
- Layer-wise dropdown menu with auto-detection.
- Color-coded scan path and Gaussian heat overlay.
- Metadata display: power, speed, focus.
- Real-time plot refresh on layer change.

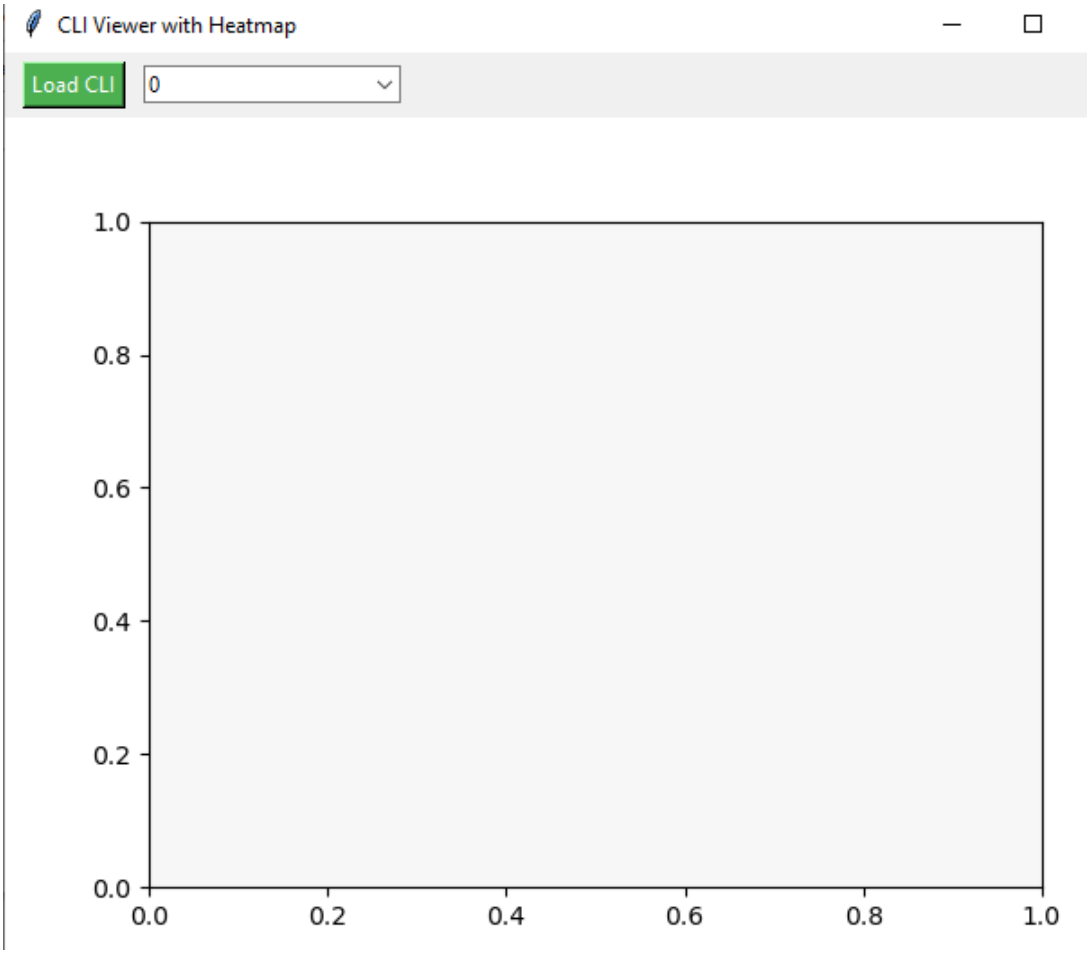

Figure 3: Initial GUI Window (Source: Obtained from Jupyter Notebook)

This tool is ideal for researchers and engineers analyzing localized heating, defect prediction, or energy inefficiencies.

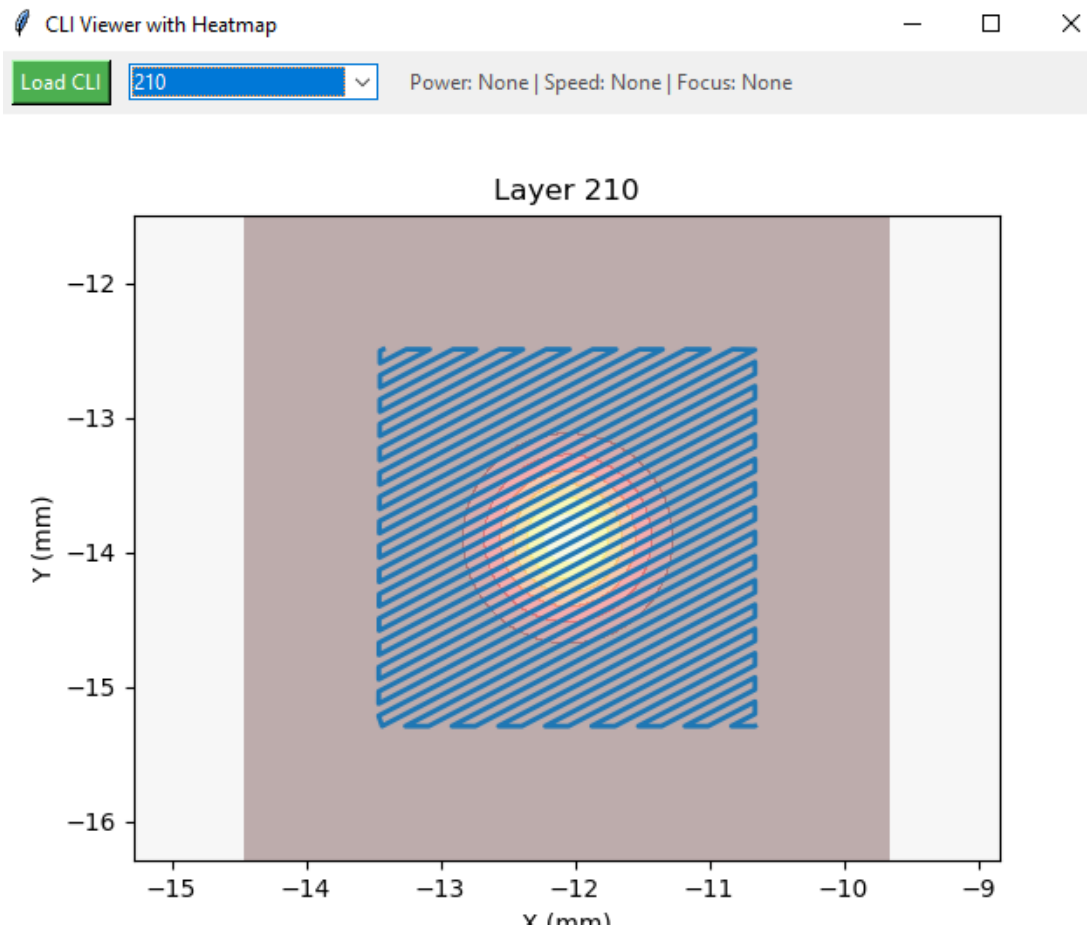

Figure 4: Window after .cli File Upload (Source: Obtained from Jupyter Notebook)

6 Limitations and Future Improvements

Current Limitations:

- Only supports static 2D visualization.
- No real-time animation or toolpath playback.
- Uses only Gaussian model for thermal simulation.

Planned Enhancements:

- 3D visualization using PyVista or VTK.
- Integration of Goldak model and other thermal profiles.
- Playback feature for scan path simulation.
- Real-time parameter tuning (power, speed, σ).
- Export plots and summary reports.

7 Conclusion

This project demonstrates a lightweight yet functional Python tool to visualize `.cli` tool paths and simulate thermal distribution using a Gaussian model. It provides researchers with an accessible platform for toolpath analysis, thermal insight, and process parameter exploration.

While current features are foundational, the modular architecture supports scaling toward full-fledged AM process simulation tools, fostering more intelligent design and process planning.

References

[1] Ferreira, P., Vilarinho, L., & Scotti, A. (2022). *Toolpath Data in AM Systems*. Journal of Manufacturing.

[2] Redchyts, O., Koller, T., & Lambert, M. (2024). *Thermal Mapping and Visualization in AM*. Additive Manufacturing Review.

[3] Dirks, C., Collet, F., & Schleifenbaum, H. (2022). *Analyzing .CLI Files for Process Simulation*. Procedia CIRP.

[4] Tran O'Leary, A., Jun, M., & Peek, D. (2022). *Process Parameter Extraction from .CLI Files*. Materials Processing Technology.

[5] Krueckemeier, L., Anderl, R., & Schleich, B. (2023). *Thermal Overlay Models in AM*. International Journal of AM Research.

[6] Mollamahmutoglu, M., & Yilmaz, S. (2021). *Gaussian Heat Distribution in AM*. Journal of Advanced Materials.

[7] Abdulrazzaq, H., Tan, K., & Foster, P. (2024). *Lightweight Python-Based GUI for AM*. Computer-Aided Engineering Journal.