RAG for Microscopy Data Analysis

Ganesh Narasimha[1], Zijie Wu[1]
[1]Center for Nanophase Materials Sciences, Oak Ridge National Laboratory, USA - 37831

Abstract

Automated microscopy platforms are central to accelerating the development of self-driving laboratories. Microscopy control and data analysis are often implemented in customized codebases tailored to specific instruments, modalities, and laboratory workflows, frequently maintained in isolated frameworks. This limits interoperability, impedes cross-platform evaluation, and hinders using domain-specific analysis instruments and datasets. Here, we present a retrieval-augmented generation (RAG) framework that interprets application-specific microscopy software and generates analysis code directly from user prompts. The RAG backbone employs OpenAI's retrieval and embedding model for creating contextual databases. We demonstrate the approach on multimodal workflows spanning scanning tunneling microscopy (STM) and scanning transmission electron microscopy (STEM), and evaluate efficacy on *sidpy*, a scanning probe microscopy analysis repository. We built a conversational RAG agent that interprets the chat-history that is unique to users and topics. Finally, we develop an interactive graphical user interface (GUI) app through which users can query the agent, review generated code, and export it for downstream testing and validation.

Methodology:

We developed a retrieval-augmented generation (RAG) workflow that interfaces with custom codebases to support analysis of multimodal microscopy data. In a RAG system, a user query is used to retrieve the most relevant code, documentation, and examples from an indexed database. The retrieved material is then provided as context to a language model, which generates outputs grounded in the local repository rather than relying only on general training knowledge. The schematic of the RAG workflow is shown below.

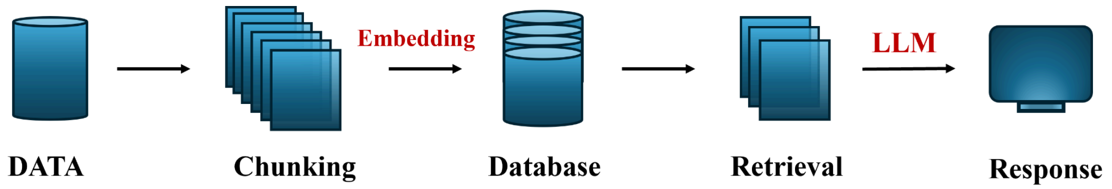
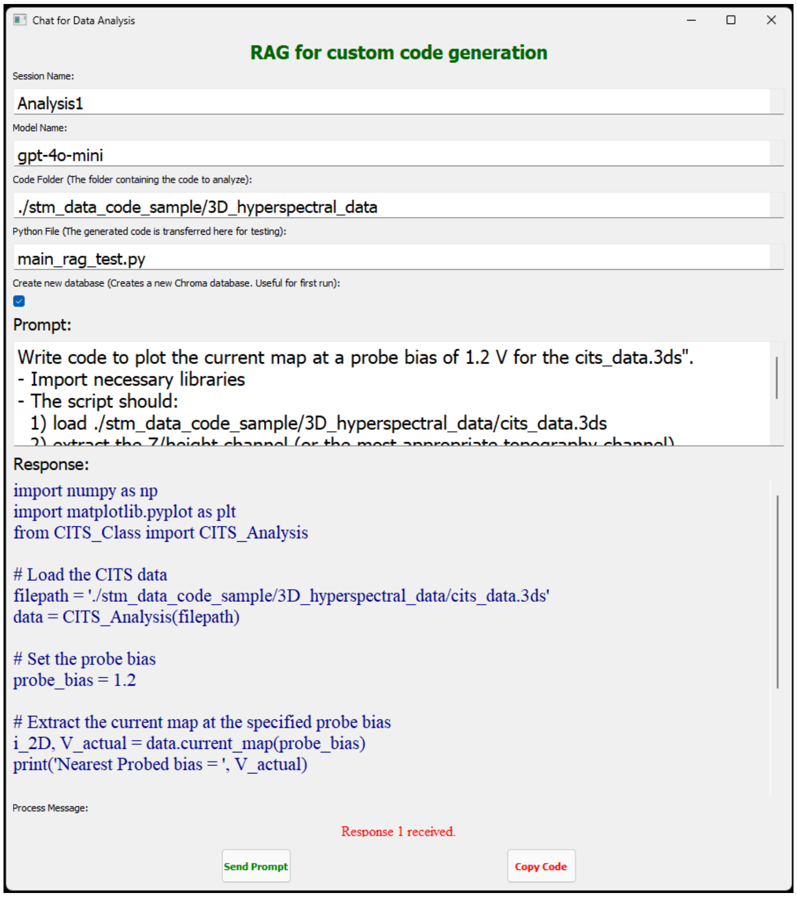


Figure 1: Flowchart of the retrieval augmented generation (RAG).

We first applied the workflow to scanning tunneling microscopy (STM) datasets. Our data comprise multimodal measurements on the semimetal $EuZn_2As_2$ [1,2]. We evaluated code generation for both (i) topographic analysis and (ii) hyperspectral current imaging via current imaging tunneling spectroscopy (CITS). In these tests, gpt-4o-mini was effective at identifying relevant routines and understanding the overall code layout. The newer gpt-5 model performed better in resolving path dependencies and adding robust import and file handling, which reduced execution failures.

Next, we implemented a conversational RAG mode by storing and reusing chat history as part of the retrieval context. This enabled the agent to retain dataset-specific details and incorporate them consistently when generating follow-on analysis code. Example notebook describes conversational RAG [link]. We also developed a GUI-based application for interactive use. The interface allows users to select a model, organize conversations into separate themes, and maintain consistent context through a persistent chat database. A snapshot of the GUI is shown below.



RAG for custom code generation

Session Name:
Analysis1

Model Name:
gpt-4o-mini

Code Folder (The folder containing the code to analyze):
./stm_data_code_sample/3D_hyperspectral_data

Python File (The generated code is transferred here for testing):
main_rag_test.py

Create new database (Creates a new Chroma database. Useful for first run):
☑

Prompt:

```
Write code to plot the current map at a probe bias of 1.2 V for the cits_data.3ds".
- Import necessary libraries
- The script should:
  1) load ./stm_data_code_sample/3D_hyperspectral_data/cits_data.3ds
  2) extract the Z/height channel (or the most appropriate topography channel)
```

Response:

```
import numpy as np
import matplotlib.pyplot as plt
from CITS_Class import CITS_Analysis

# Load the CITS data
filepath = './stm_data_code_sample/3D_hyperspectral_data/cits_data.3ds'
data = CITS_Analysis(filepath)

# Set the probe bias
probe_bias = 1.2

# Extract the current map at the specified probe bias
i_2D, V_actual = data.current_map(probe_bias)
print('Nearest Probed bias = ', V_actual)
```

Process Message:
Response 1 received.

Figure 2: Snapshot of the GUI-App for interactive RAG based code generation.

Finally, we assessed whether the agent could generate reliable, production-ready code for a larger and more complex repository by building a separate RAG database for sidpy [3], an open-source package providing a universal data structure for common microscopy data. We built a chromaDB on sidpy's code repository and asked it to generate codes for several prompts [link] to analyze scanning transmission electron microscopy (STEM) data such as HAADF (High-Angle Annular Dark-Field) images. The RAG agent produced comprehensible and executable python code; it is especially worth noting that while the agent did not fully understand the "plot" functionality of the dataset in the initial attempt, an extra hint in the prompt indicating the correct way to invoke plot functions helped the agent produce the correct plot prompt, as shown in [link].

References:

1.Narasimha, Ganesh, et al. *npj Computational Materials* 11.1 (2025): 189.

2. Kong, Dejia, et al. *Communications Physics* 8.1 (2025): 287.

3. Vasudevan, Rama Krishnan, et al. *Advanced Theory and Simulations* 6.11 (2023): 2300247.