VISKNACKS: AGENTIC PARAVIEW VISUALIZATION ON HPC

A HPC-ready agentic system evaluated for visual quality

Nicholas M. Synovic[1], Manish Mathai[2], Mengjiao Han[3], Joseph Insley[3], Janet Knowles[3], Victor Mateevitsi[3], Michael E. Papka[3], Silvio Rizzi[3], George K. Thiruvathukal[1]
Loyola University Chicago[1], University of Oregon[2], Argonne National Labs[3]

PROJECT SUMMARY

Motivation: Large language model (LLM) enabled scientific visualization agents (SciVisAgents) can assist experts in exploring scientific datasets and prototyping ParaView visualization pipelines.

Problem: Existing SciVisAgents often target local execution and may lack capabilities needed for HPC deployment, including remote rendering, integration with deployed visualization services, and portable software packaging.

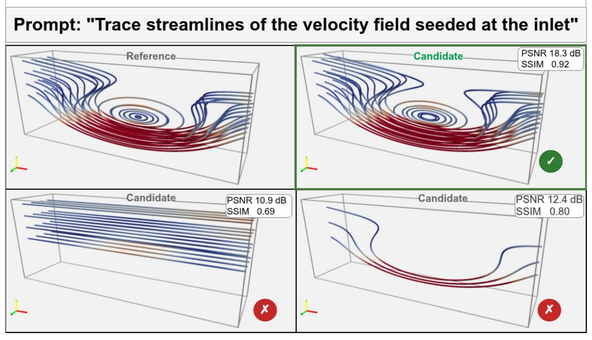
Contribution: We introduce *VisKnacks*, a LLM agnostic SciVisAgent system for hybrid local-HPC ParaView workflows. VisKnacks enables agents to create and refine pipelines locally while executing renders on the ALCF Crux HPC resource.

Evaluation: We evaluate VisKnacks on 10 SciVisAgentBench ParaView tasks using Anthropic Claude Opus 4.8, Google Gemini 3.5 Flash, and OpenAI GPT-5.5. We compare HPC-rendered images with task reference images using PSNR and SSIM.

Implication: VisKnacks provides a modular and portable approach for exposing documentation retrieval, pipeline generation, and remote rendering to existing agent harnesses. Deployment on Crux also reveals practical challenges for HPC-based SciVisAgents, including limited LLM access to large datasets and queueing delays for remote rendering services.

MOTIVATION

- SciVisAgent strategies address different parts of the visualization workflow, but they have not yet been integrated into a reproducible HPC deployment.
- While prior work has evaluated individual strategies in isolation [5], limited work has evaluated combinations of strategies on visual quality metrics (e.g., PSNR, SSIM).



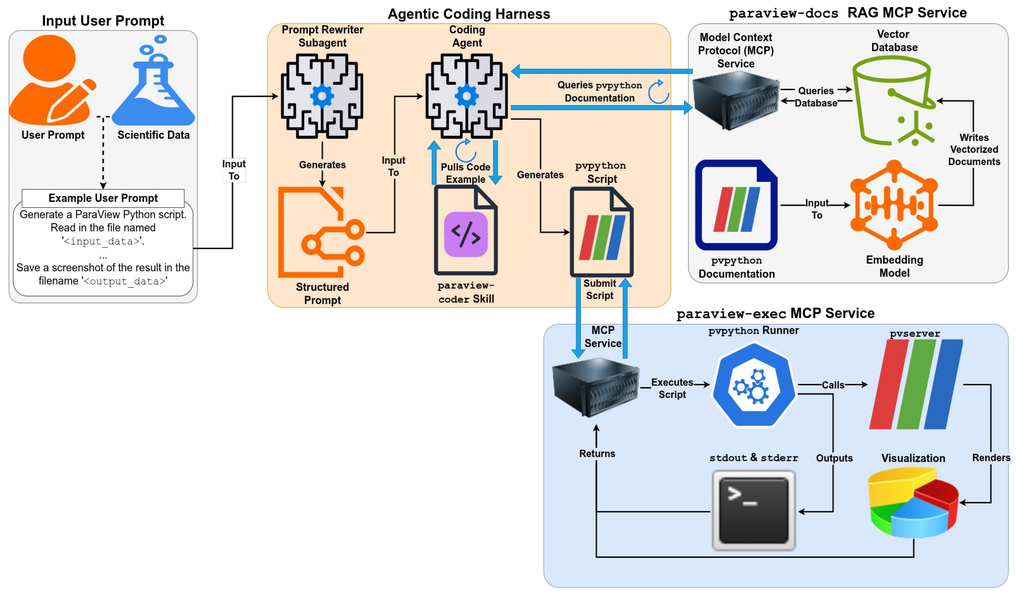
STRATEGIES

- **Agent Skills:** Provides common ParaView code patterns [1].
- **Coding Harness:** Orchestrates strategies into a cohesive workflow.
- **Model Context Protocol (MCP):** Connects the agent to external services for documentation lookup and remote ParaView execution on HPC resources [2].
- **Retrieval Augmented Generation (RAG):** Retrieves relevant ParaView documentation to support version aware code generation [3].
- **Subagents:** Decomposes the visualization task into intermediate steps, converting user requests into structured pipeline actions [4].

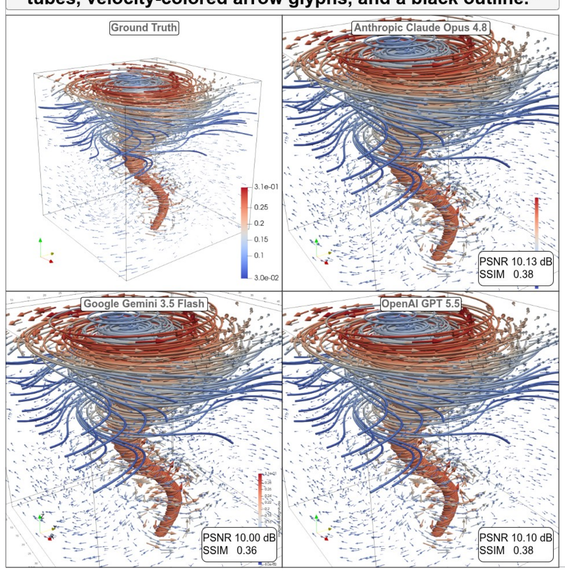
SYSTEM OVERVIEW

Our system operates in **six** discrete steps:

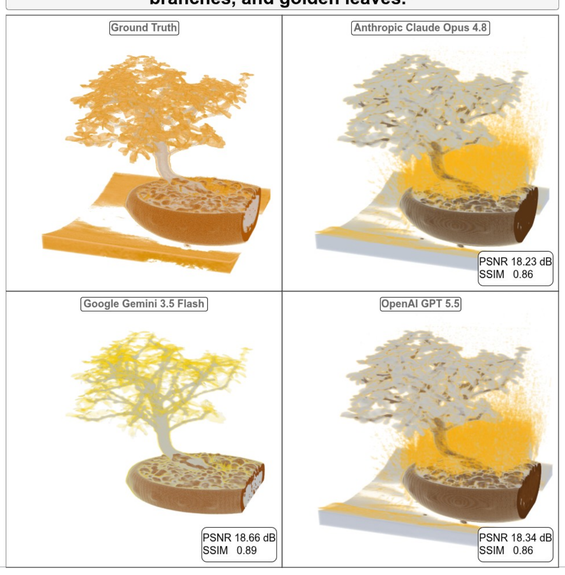
1. **Prompt:** User requests a visualization in natural language.
2. **Reformulate:** A subagent converts the request into ParaView [6] pipeline steps.
3. **Generate:** The coding agent writes ParaView code referencing the `paraview-coder` skill templates.
4. **Consult Docs:** A MCP service retrieves Python `paraview.simple` module API documentation when skill templates are insufficient.
5. **Execute:** The code runs on a HPC resource through hosting the `paraview-exec` MCP service.
6. **Iterate:** Rendered visualizations, `stdout`, and `stderr` guide iterative refinement of ParaView pipeline code.



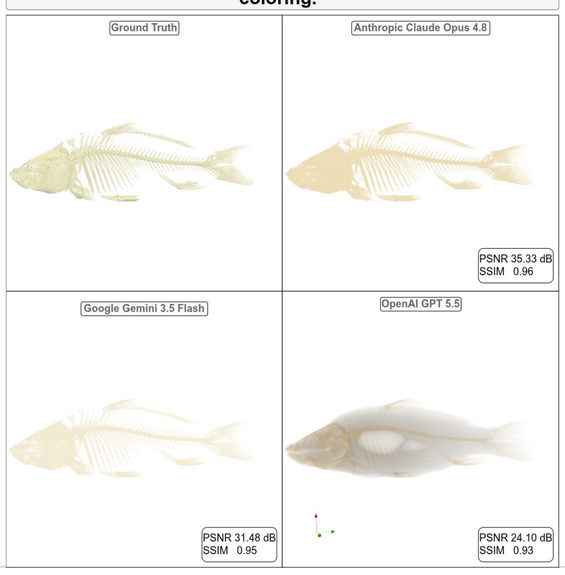
Task: "Render the Tornado vector field with streamlines, tubes, velocity-colored arrow glyphs, and a black outline."



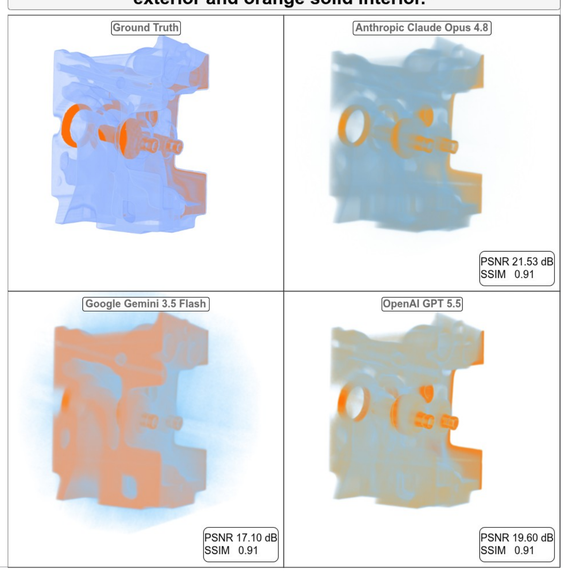
Task: "Render the bonsai volume as a brown pot, silver branches, and golden leaves."



Task: "Render the carp skeleton volume with bone-only coloring."



Task: "Render the engine volume with a light-blue transparent exterior and orange solid interior."



CONCLUSIONS

- **We unify** five SciVisAgent strategies in a packaged OpenCode workflow for reproducible HPC deployment.
- **We evaluate** this workflow on ALCF Crux using the SciVisAgentBench ParaView suite, observing comparable PSNR and SSIM scores across model providers.
- **We provide** a model agnostic, multi-strategy SciVisAgents foundation for future systems research and deployment to HPC resources.

NEXT STEPS

- Expand to the full SciVisAgentBench suite, additional datasets, and visualization backends.
- Ablation study on how combinations of strategies contribute to visualization quality.

ACKNOWLEDGMENTS

- This research used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357. This work was supported by the U.S. Department of Energy, Office of Science (SC), Advanced Scientific Computing Research (ASCR), Competitive Portfolios Project, PIONEER: Program for Intelligent Optimization for Next-generation Experiments, Exploration, and Research, under Contract DE-AC02-06CH11357.

REFERENCES

[1] K. Ai, H. Miao, et.al., “SciVisAgentSkills: Design and Evaluation of Agent Skills for Scientific Data Analysis and Visualization,” doi: 10.48550/ARXIV.2606.05525.

[2] S. Liu, H. Miao, and P.-T. Bremer, “ParaView-MCP: An Autonomous Visualization Agent with Direct Tool Use,” doi: 10.1109/VIS60296.2025.00018.

[3] T. Peterka, T. Mallick, O. Yildiz, D. Lenz, C. Quammen, and B. Geveci, “ChatVis: Large Language Model Agent for Generating Scientific Visualizations,” doi: 10.1109/LDAV68558.2025.00007.

[4] T. Mallick, O. Yildiz, D. Lenz, and T. Peterka, “ChatVis: Automating Scientific Visualization with a Large Language Model,” doi: 10.1109/SCW63240.2024.00014.

[5] K. Ai et al., “SciVisAgentBench: A Benchmark for Evaluating Scientific Data Analysis and Visualization Agents,” 2026, arXiv. doi: 10.48550/ARXIV.2603.29139.

[6] J. Ahrens, B. Geveci, and C. Law, “ParaView: An end-user tool for large-data visualization,” pp. 717–731. doi: 10.1016/B978-012387582-2/50038-1.

GitHub Repository with Source Code, Installation Instructions, and Demos

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY