CATVis: A Collaborative Multi-Agent Workflow for Turbomachinery Simulation Data Visualization

Zhe Wang[1,2*], Zehao Lou[2,3*], Guanghui Zhao[2,3], Yu Dong[2], Guan Li[1,2], Pengyi Xu[4], Gaorong Liang[1], Jun Liu[1,2], Guihua Shan[1,2,3†]

[1] University of Chinese Academy of Sciences, Beijing, China
[2] Computer Network Information Center, Chinese Academy of Sciences
[3] Hangzhou Institute for Advanced Study, UCAS, Hangzhou, China
[4] Michigan Technological University, Houghton, MI, USA

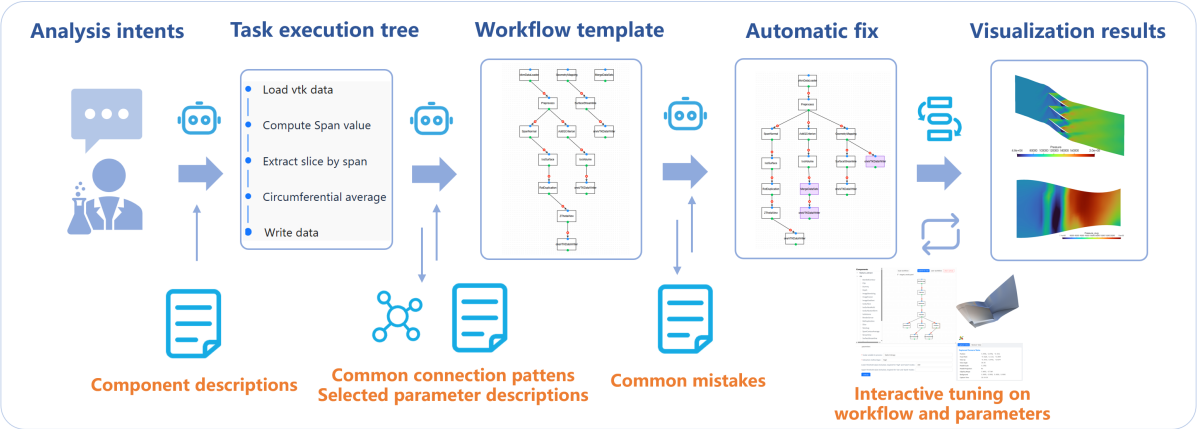

Figure 1: Overview of CATVis: (1) User intents are parsed into a task execution tree composed of domain-specific task components derived from turbomachinery visualization algorithms. (2) The execution tree is mapped to a structured workflow template represented as a DAG, guided by component descriptions and common connection patterns. (3) The generated workflow is automatically diagnosed and refined using a knowledge base of common error patterns. (4) An interactive interface enables human-in-the-loop refinement, allowing users to adjust workflow structure and parameters (e.g., camera settings).

Abstract

Recent advances in AI for Science have enabled natural language (NL) interfaces for scientific data analysis. In turbomachinery CFD post-processing, translating ambiguous high-level analytical goals (e.g., vortex identification) into precise visualization procedures supporting complex domain-specific analysis is challenging. We present CATVis, a **C**ollaborative multi-**a**gen**t** workflow system that bridges this gap by transforming NL intents into structured middle representation for visualization. Our approach reformulates domain-specific visualization procedures as composable workflow representations, and use multi agent to generate workflow representations via intent planning, template generation, and error-aware refinement, where each stage incrementally updates a shared structured representation. We evaluate the impact of external knowledge and workflow structuring on generation accuracy, demonstrating that the proposed approach significantly improves complex workflow generation correctness while reducing prompt complexity.

1 Introduction

AI for Science (AI4Science) has emerged as a new paradigm for accelerating scientific discovery, with LLM-based agents serving as a central component across a growing range of scientific software applications [20]. In particular, data agents have demonstrated promising capabilities in automating data analysis and visualization tasks, mapping natural language descriptions to visualization outputs, including scientific charts [15] and 3D visualizations [30]. The rapid advancement of pretrained LLMs, combined with improved knowledge organization techniques, has further accelerated this trend [23]. Despite this progress, generating complex, domain-specific visualization pipelines remains challenging.

Post-processing of Computational Fluid Dynamics (CFD) simulations is essential for understanding complex flow phenomena across a wide range of engineering applications. CFD solvers produce high-dimensional, unstructured datasets requiring sophisticated visualization pipelines for interpretation. Among these applications, turbomachinery simulation post-processing plays a critical role in validating the design of engine components, including compressors, turbines, and combustion chambers. Unlike general-purpose CFD visualization, turbomachinery post-processing relies on specialized techniques that exploit geometric information intrinsic to the rotating machinery. As examples shown in Figure 2, blade-to-blade surfaces at a constant spanwise height and meridional plane through the machine axis require coupling the flow solution with the underlying blade and passage geometry.

Commercial post-processing software, such as CFX-Post [1] and Tecplot Macro Language [25], provides dedicated turbomachinery modules that leverage geometric metadata during visualization. They target interactive use by domain experts and provide domain-specific scripting interfaces for visualization filters and operations. However, low-level macro languages (e.g., Tecplot [9]) are inherently state-dependent and structurally rigid; without a high-level abstraction layer, they force LLMs to manage abstract planning and intricate execution syntax simultaneously, increasing the difficulty of reliable generation. Open-source visualization platforms such as ParaView [2] provide Python-based scripting interfaces, and re-

*Authors contributed equally.
†Corresponding author: sgh@cnic.cn

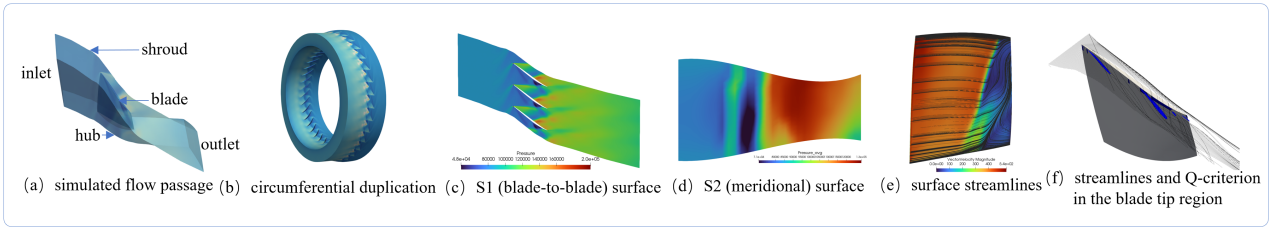

Figure 2: Visualization results for compressor simulation data, generated by our workflow system and rendered in ParaView.

cent studies [18, 13] have demonstrated the feasibility of integrating LLMs with the open source tool for automated visualization generation. However, these studies are developed on mature ParaView and VTK ecosystems, where abundant public resources enable LLMs to leverage existing knowledge learned during pretraining. In addition, they are not designed for complex turbomachinery post-processing workflows that involve multiple geometry-aware operations with dependency relationships, such as span-value computation, duplication, and blade-to-blade visualization. Since turbomachinery visualization libraries lack robust public documentation, rapidly changing APIs, and highly specialized operations, ensuring that LLMs accurately invoke these evolving interfaces remains challenging.

To address the difficulty of reliably generating workflows over low-level scripting interfaces, we abstract turbomachinery visualization algorithms into composable workflow tasks with structured composition rules. This abstraction reduces generation complexity and enables interactive human refinement. To further improve generation quality and support flexible context organization, we decompose workflow generation into multiple collaborative stages, where stage-specific contextual knowledge supports intent planning, workflow construction, and feedback-driven refinement. The main contributions of this paper are listed as follows:

(1) Workflow-centric abstraction of complex turbomachinery visualization processes. We abstract domain-specific turbomachinery procedures into composable units organized as directed acyclic graphs (DAGs). Each unit exposes structured attributes that describe its semantics and parameters. We further design an intermediate workflow representation that balances semantic clarity, human editability, and LLM generation efficiency.

(2) A collaborative multi-agent workflow generation framework. We propose a multi-agent visualization generation approach that constructs structured visualization workflow templates from user intents. The framework supports human-in-the-loop refinement through an interactive interface, allowing users to efficiently correct and improve generated workflows.

(3) Systematic evaluation of workflow generation and context strategies. We evaluate the ability of pretrained LLMs to translate visualization intents into executable workflow (DAG representations) based on published turbomachinery use cases, and analyze how different context organization strategies affect generation accuracy and workflow quality.

2 Background and Related Work

In this section, we first discuss the visualization of turbomachinery simulation data, and then present the multi-agent framework and the research gap for visualizing it.

2.1 Typical Turbomachinery Visualization

Visualization is an essential component of CFD post-processing [11], enabling domain experts to analyze complex flow behaviors through interactive visual exploration. Typical compressor simulations are performed on a single blade passage using periodic boundary conditions, with the computational domain defined by the hub, shroud, inlet, and outlet surfaces (Figure 2(a)). The full annular configuration is reconstructed by circumferentially replicating the simulated passage about the rotational axis (Figure 2(b)). The S1 surface (Figure 2(c)) and S2 surface (Figure 2(d)) provide sectional views that enable global inspection of flow patterns within the compressor. Surface streamlines (Figure 2(e)) are generated by mapping velocity fields onto blade surfaces and computing constrained streamline trajectories, enabling detailed analysis of flow evolution along blade geometry. In the blade tip region, vortex structures are identified by the Q-criterion combined with isosurface extraction (Figure 2(f)), revealing the spatial distribution of tip leakage vortices.

2.2 Research Trends in Turbomachinery Visualization

Turbomachinery visualization has been extensively studied over the past decades, particularly in flow visualization and vortex detection [21]. These advances have been incorporated into commercial software systems such as CFX-Post [1] and Tecplot [25], which provide specialized modules for domain-specific visualization, including spanwise slicing, meridional surface extraction, etc.

With the increasing scale of CFD simulations [8], research works focused on large-scale and in-situ visualization techniques to reduce data transfer overhead and improve efficiency of analysis [12, 6] and high fidelity rendering [19]. These approaches significantly reduce storage requirements and improve scalability for large-scale turbomachinery simulations. As simulation complexity continues to grow, visualization pipelines have evolved into multi-stage workflows that require repeated configuration of domain-specific operations. Traditional scripting-based approaches partially automate these workflows [10, 16], but still rely heavily on expert knowledge to design correct processing sequences.

Recently, CFD research has begun exploring AI-driven workflow automation. Large language models (LLMs) now assist in simulation setup, execution, analysis, and post-processing, as demonstrated by OpenFOAMGPT [17], ChatCFD [7], and MetaOpenFoam [4]. Since visualization-based post-processing is essential in CFD pipelines, these advances suggest that intelligent workflow generation can enhance turbomachinery visualization.

2.3 Agentic workflow for scientific visualization

Recent research has demonstrated the ability to generate visualization programs using LLMs [18, 3, 28]. These works leverage techniques such as retrieval-augmented generation (RAG), fine-tuning, and in-context learning to improve the reliability and efficiency of visualization code generation. Domain-specific systems, such as AuraGenome [27], further show how LLMs can facilitate visualization workflows tailored to specialized scientific fields. While LLM-based approaches show strong potential for visualization generation, complex workflows often require coordinated multi-step reasoning, motivating recent exploration of multi-agent approaches for scientific visualization. For example, LightVA [29] demonstrates how LLM-based agents can decompose high-level analysis goals into executable tasks through collaborative planning and execution. Recent studies [14] extend this idea to multi-agent collaboration for visualization generation and result explanation.

More broadly, agentic workflow typically organizes agents into components such as planners, memory modules, action spaces, and verification mechanisms [20]. Recent studies further highlight

the importance of suitable intermediate workflow representations and human-in-the-loop mechanisms, such as interactive workflow graphs [26] and human-guided refinement strategies [24].

In summary, despite recent progress in LLM-driven visualization and agentic workflows, they mainly focus on improving interaction mechanisms without explicitly bridging domain-specific CFD simulation data visualization queries and scientific visualization workflow generation. Reliably generating such complex workflows via an Agent-based approach for turbomachinery CFD post-processing remains a major challenge. In particular, there is a lack of structured intermediate representations that facilitate both reliable automated generation and human-in-the-loop interactive refinement.

3 METHOD

This section presents the design of the proposed workflow generation framework. We first describe the workflow representation used to organize visualization operations. Next, we describe how we prepare context knowledge to support LLM reasoning. Finally, we present the multi-stage workflow for template generation.

3.1 Visualization Workflow Representation

Our visualization workflow is represented as a directed acyclic graph (DAG), where nodes represent visualization filters and edges denote data dependencies between operations. Each node encapsulates a specific visualization or data-processing operation, and its parameters are organized using a standardized schema (e.g., PyDantic), which can be serialized to structured formats such as YAML or JSON. This representation bridges domain-specific visualization procedures and LLM-based generation.

The workflow composer parses workflow templates into executable DAGs and schedules node execution according to dependency relationships. Domain-specific filters for turbomachinery visualization are integrated into the workflow, including span computation based on hub and shroud geometry, vortex-related scalar computation (e.g., Q-criterion), and streamline generation. Figure 2 illustrates representative visualization results using the presented workflow system. In addition to domain-specific operations, the workflow incorporates commonly used visualization filters such as contouring, slicing, and offscreen rendering. The current library contains more than 50 filters. We extend the visualization algorithms introduced in GPVis [22] and reorganize them into a modular workflow-compatible library.

All workflow tasks inherit from a unified parent task class, enabling consistent parameter management and automatic data connection. The graph-based structure increases flexibility for humans to verify the correctness of dependencies in the LLM-generated workflow tasks, while the structured parameter representation makes the workflow interpretable by both machines and users. Users can further modify parameters through the graphical interface when needed. A detailed workflow example and its visualization results are provided in the supplementary materials.

3.2 Contextual Construction and Prompt Generation

To support reliable workflow generation, we prepare multiple types of domain-specific knowledge as contextual input to the LLM. This contextual information helps the LLM understand workflow construction rules and data details and generate executable visualization pipelines from user queries.

We organize the context knowledge into five categories: (1) workflow examples: representative workflow instances that demonstrate common visualization procedures; (2) workflow rules: structural constraints defining task dependencies, data connections, field usage, and valid operator sequences; (3) task descriptions: semantic descriptions of each visualization operation, including its purpose and applicable scenarios; (4) parameter specifications: detailed definitions of task parameters, automatically extracted from Python docstrings and PyDantic schemas, and (5) dataset descriptions: metadata describing dataset structure, including field names, block organization, and geometric components (e.g., blade, hub, and shroud regions). This structured contextual information enables the LLM to reason effectively about workflow parameters. Section 2 of the supplementary materials further illustrates the strategy for preparing contextual information to support multi-agent tasks with different objectives.

3.3 Multi-stage Workflow template generation

We adopt a multi-stage framework in which multiple agents collaboratively construct executable visualization workflows through progressive refinement (described in Figure 1). Each stage is modeled as an independent agent with a specialized role, including intent interpretation, workflow construction, and error-aware refinement. Agents shared a structured workflow representation that is incrementally updated to ensure consistency and correctness.

The intent agent first parses the user's natural language input into high-level visualization tasks. These tasks are expanded into a task execution tree based on predefined task components, where candidate visualization units are identified to guide downstream generation. The workflow generation agent then maps the execution tree into structured workflow templates by instantiating operators and establishing data dependencies. To ensure completeness, predefined connection patterns derived from prior examples are applied to insert necessary intermediate operations when dependencies are missing. The refinement agent further improves the generated workflows through error-aware feedback based on predefined function calls to provide suggestions, including common failure patterns collected during development. Section 2 of the supplementary materials provides details of context preparation for each agent.

Finally, users can interactively inspect and adjust the generated workflows through the interface. Parameters that are difficult to specify in natural language (e.g., camera settings) can be refined with visual feedback, and users may modify workflow topology by editing nodes and their associated parameters.

4 EVALUATION

We carefully select five representative visualization use cases derived from published turbomachinery studies. Table 2 in the supplementary materials provided detailed descriptions for these tasks. For each case, we analyze the visualization results reported in the corresponding literature, and reconstruct equivalent reference workflows using our system and simulation data based on the NASA Rotor37 [5]. We conduct two experiments with these use cases: one evaluating the effectiveness of external knowledge, and the other evaluating the proposed multi-agent workflow for visualization generation. To ensure consistent evaluation, we use DeepSeek-V3 as the base LLM.

4.1 External Knowledge Ablation

This experiment aims to investigate how different combinations of external knowledge affect the quality of workflow template generation from natural language descriptions. Table 1 in the supplementary materials shows five configurations with increasing levels of external information included in system prompt. **C1** provides only simple workflow examples (e.g., small pipelines with 3-5 nodes), serving as the baseline to assess how well the model can generalize from minimal structural demonstrations without additional semantic. **C2** additionally introduces construction rules that define how operators compose valid workflows, giving the model explicit structural constraints. **C3** further adds semantic descriptions for each operator, enabling the model to better understand their capabilities. **C4** further adds parameter-level descriptions that specify the name and type of arguments accepted by each operator. **C5** additionally adds data-level descriptions, including geometric properties and available field variables, ensuring generated workflow parameter consistency with the input data.

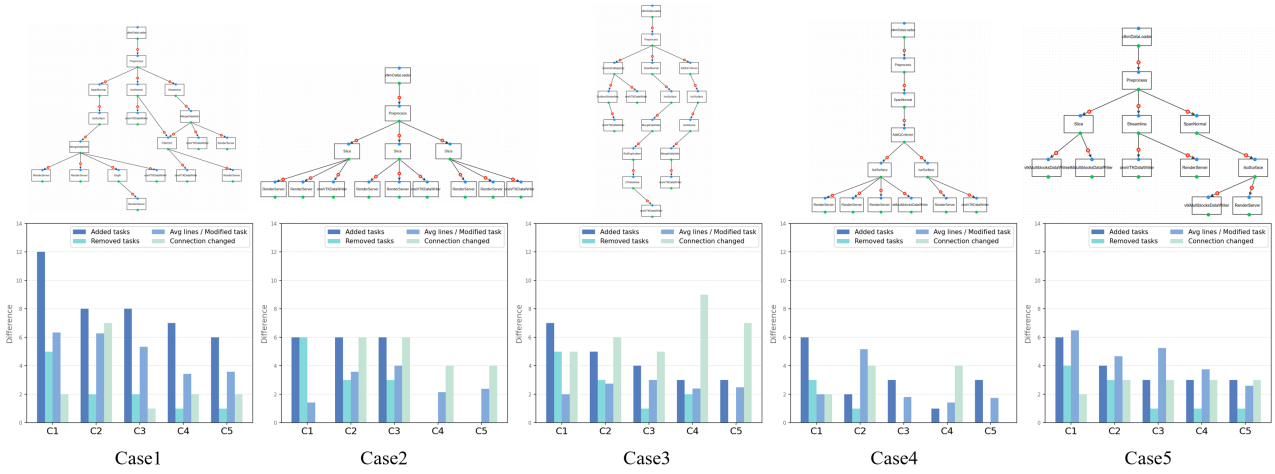

Figure 3: Comparison of workflow generation errors across five turbomachinery cases under different external knowledge configurations.

We quantitatively compare generated workflows of C1-C5 with reference workflows using **four metrics**: the number of added tasks, deleted tasks, modified connections, and parameter mismatches. Figure 3 shows the performance of the LLM across five complex turbomachinery visualization cases under five external knowledge configurations, where lower values indicate closer agreement with the ground truth (user prompt of each case are listed in supplementary materials). Based on quantitative trends and qualitative log analysis, we summarize the following key observations:

Across all cases, adding structural rules and operator descriptions gradual reduces added and removed tasks from C1 to C5. In several cases (e.g., Case2 and Case4), structural errors are fully eliminated under C4 and C5, indicating that the generated workflows become increasingly consistent with the ground truth topology. Despite the improvements observed in C5, parameter differences tend to plateau rather than reaching zero. The remaining discrepancies mainly occur in rendering-related parameters, such as camera position, focal point, and view direction, which are inherently difficult to maintain consistently in one-shot generation. We also observe that performance improvements come at the cost of significantly larger context inputs, including irrelevant operator input parameter descriptions. In practice, when we update the configuration associated with a specific operator, we only need to provide information associated with this operator instead of the whole context. These observations motivate a design choice that decomposes context information into multiple stages aligned with different generation goals, as evaluated in the next experiment.

4.2 Multi-Agent Workflow Generation

This experiment evaluates the proposed multi-agent visualization generation approach. We use high-level visualization instructions that reflect realistic user intents, without explicitly specifying workflow construction details.

The baseline configuration adopts C5 from Experiment 4.1. We then apply the proposed multi-agent framework to generate workflows from the user query. As described in Section 3.3, the process is decomposed into three stages: intent planning (Stage 1), workflow template generation (Stage 2), and self-correction (Stage 3).

Due to ambiguity in the input instructions, the generated workflows may vary across runs. We therefore conduct three independent runs to evaluate the stability of the generated results. For each run, the LLM first generates a workflow template, which is then manually refined into a runnable version using a dedicated visualization interface. We compare the generated workflows with the refined versions to quantify the amount of correction required, following the same metrics as the previous experiment. Detailed results and the visualization interface are provided in the supplementary materials.

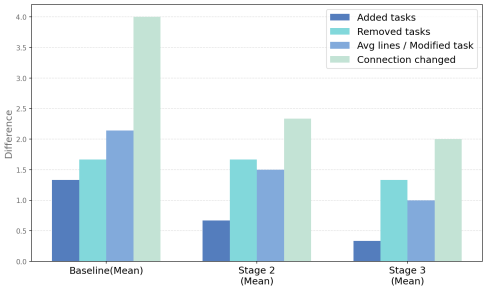

Figure 4: Workflow template differences: baseline vs. multi-stage generation (lower bounds truncated at 0).

Figure 4 shows detailed differences between generated and corrected workflows for the baseline, Stage 2, and Stage 3 averaged over three runs. Detailed experiment number are listed in the supplementary materials. The results indicate that both Stage 2 and Stage 3 outperform the baseline across all runs, and Stage 3 achieves the best performance. Test 3 in Stage 3 even produces a near-correct workflow, with no added or removed nodes, and only two camera parameters are adjusted. This improvement is attributed to the error-aware refinement mechanism, which captures common failure patterns and reduces repeated mistakes. Furthermore, the multi-agent decomposition reduces the need for extensive operator descriptions in the input, resulting in more compact prompts and lower context overhead. For example, the total token usage decreases from 23,500 to 14,000 approximately when using the multi-stage approach to generate a workflow template.

5 Conclusion and Future Work

This paper presents an intelligent workflow generation approach for turbomachinery visualization that integrates structured representations, domain-aware context knowledge, and multi-stage agent reasoning. The proposed method translates natural language intents into intermediate representations and supports interactive refinement. Results show improved reliability and reduced manual effort. Future work will extend the framework to more turboachinery visualization scenarios, such as turbines and combustors, standardized workflow interfaces, and improved robustness through better context knowledge management and multi-pass verification.

Acknowledgment

This work was supported by the Beijing Natural Science Foundation (No.4254090) and the Youth Fund of Computer Network Information Center of Chinese Academy of Sciences (No.25YF07). The authors gratefully acknowledge the experts at the Digital Twin Research Center, Institute of Engineering Thermophysics, Chinese Academy of Sciences, and Zhonghao Wang (Institute of Engineering Thermophysics) for providing constructive suggestions and assistance for the user study.

References

[1] Ansys, Inc. Turbo Selector. Ansys Help, Version 24.2, 2024. Accessed: March 18, 2026. 1, 2

[2] U. Ayachit. *The paraview guide: a parallel visualization application*. Kitware, Inc., 2015. doi: 10.5555/2789330 1

[3] A. Biswas, T. L. Turton, N. R. Ranasinghe, S. Jones, B. Love, W. Jones, A. Hagberg, H.-W. Shen, N. DeBardeleben, and E. Lawrence. Vizgenie: Toward self-refining, domain-aware workflows for next-generation scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 32:1021–1031, 2026. doi: 10.1109/TVCG.2025.3634655 2

[4] Y. Chen, X. Zhu, H. Zhou, and Z. Ren. Metaopenfoam 2.0: Large language model driven chain of thought for automating cfd simulation and post-processing. *arXiv preprint arXiv:2502.00498*, 2025. doi: 10.48550/arXiv.2502.00498 2

[5] J. D. Denton. Lessons from rotor 37. *Journal of Thermal Science*, 6(1):1–13, 1997. doi: 10.1007/s11630-997-0010-9 3

[6] S. Dutta, C.-M. Chen, G. Heinlein, H.-W. Shen, and J.-P. Chen. In situ distribution guided analysis and visualization of transonic jet engine simulations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):811–820, 2017. doi: 10.1109/TVCG.2016.2598604 2

[7] E. Fan, K. Hu, Z. Wu, J. Ge, J. Miao, Y. Zhang, H. Sun, W. Wang, and T. Zhang. Chatcfd: A large language model-driven agent for end-to-end computational fluid dynamics automation with structured knowledge and reasoning. *Advanced Intelligent Discovery*, 2(3):e202500174, 2026. doi: 10.1002/aidi.202500174 2

[8] Y. Fu, W. Shen, J. Cui, Y. Zheng, G. Yang, Z. Liu, J. Zhang, T. Ji, F. Xie, X. Lv, H. Liu, X. Liu, X. Liu, X. Song, G. Tao, Y. Yan, P. Tucker, S. Miller, S. Luo, S. Koric, and W. Zheng. Towards exascale computation for turbomachinery flows. In *SC23: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. Association for Computing Machinery, 2023. doi: 10.1145/3581784.3627040 2

[9] O. Heidbach, M. Ziegler, and D. Stromeyer. Manual of the tecplot 360 add-on geostress v2. 0. 2020. doi: 10.2312/wsm.2020.001 1

[10] M. M. Hossain, B. Roy, C. K. Roy, and K. A. Schneider. Vizsciflow: a visually guided scripting framework for supporting complex scientific data analysis. *Proceedings Of The ACM On Human-Computer Interaction*, 4(EICS):1–37, 2020. doi: 10.1145/3394976 2

[11] G. Ilieva. Cfd—a powerful visualization tool in turbomachinery applications. In *Advances in Visualization and Optimization Techniques for Multidisciplinary Research: Trends in Modelling and Simulations for Engineering Applications*, pp. 283–305. Springer, 2019. doi: 10.1007/978-981-13-9806-3_10 2

[12] M. List, M. Turner, D. Galbraith, J. Nimersheim, and M. Galbraith. High resolution, parallel visualization of turbomachinery flowfileds. In *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, p. 5043. American Institute of Aeronautics and Astronautics, 2007. doi: 10.2514/6.2007-5043 2

[13] S. Liu, H. Miao, and P.-T. Bremer. Paraview-mcp: An autonomous visualization agent with direct tool use. In *2025 IEEE Visualization and Visual Analytics (VIS)*, pp. 61–65. IEEE, 2025. doi: 10.1109/VIS60296.2025.00018 2

[14] X. Lu, G. Li, Y. Dong, R. Peng, Z. Wang, D. Tian, and G. Shan. Agentic scientific visualization generation and caption semantic alignment. *Information Visualization*, 25:213–231, 2026. doi: 10.1177/14738716261434841 2

[15] P. Maddigan and T. Susnjak. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, 11:45181–45193, 2023. doi: 10.1109/ACCESS.2023.3274199 1

[16] K. Naha and H. M. Jamil. Abstract visual scientific workflow design using visflow 2.0. In *Proceedings of the 37th International Conference on Scalable Scientific Data Management*, pp. 1–6, 2025. doi: 10.1145/3733723.3733746 2

[17] S. Pandey, R. Xu, W. Wang, and X. Chu. Openfoamgpt: A retrieval-augmented large language model (llm) agent for openfoam-based computational fluid dynamics. *Physics of Fluids*, 37(3):035120, 2025. doi: 10.1063/5.0257555 2

[18] T. Peterka, T. Mallick, O. Yildiz, D. Lenz, C. Quammen, and B. Geveci. Chatvis: Large language model agent for generating scientific visualizations. In *2025 IEEE 15th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 22–32. IEEE, 2025. doi: 10.1109/LDAV68558.2025.00007 2

[19] A. Prabhakar, C. Goddard, D. Amirante, I. Z. Reguly, A. Gerstenberger, J. F. Suhrmann, S. A. Jarvis, L. Lapworth, and G. R. Mudalige. Virtual certification of gas turbine engines - visualizing the dlr rig250 compressor. 2022. 2

[20] S. Ren, P. Jian, Z. Ren, C. Leng, C. Xie, and J. Zhang. Towards scientific intelligence: A survey of llm-based scientific agents. *ArXiv*, abs/2503.24047, 2025. doi: 10.48550/arXiv.2503.24047 1, 2

[21] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proceedings of Seventh Annual IEEE Visualization'96*, pp. 381–384. IEEE, 1996. doi: 10.1109/VISUAL.1996.568137 2

[22] G. Shan, J. Liu, G. Li, Y. Gao, T. Xu, and D. Tian. Gpvis: A scientific visualization system for large scale data. *Frontiers of Data and Computing*, 1(1):46–62, 2019. doi: 10.11871/jfdc.issn.2096.742X.2019.01.006 3

[23] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards natural language interfaces for data visualization: A survey. *IEEE transactions on visualization and computer graphics*, 29(6):3121–3144, 2022. doi: 10.1109/TVCG.2022.3148007 1

[24] Y. Tang, Y. Xie, Y. Feng, T. Lan, J. Lao, Y. Cheng, and W. Chen. Vividoc: Generating interactive documents through human-agent collaboration. *arXiv preprint arXiv:2603.27991*, 2026. doi: 10.48550/arXiv.2603.27991 3

[25] Tecplot, Inc. Tecplot 360 2020 r2 scripting guide, 2020. https://www.tecplot.com/docs/pytecplot/. 1, 2

[26] J. Wang and G. Li. Aop: Automated and interactive llm pipeline orchestration for answering complex queries. CIDR, 2025. 3

[27] C. Zhang, Y. Dong, Y. Wang, Y. Han, G. Shan, and B. Tang. Auragenome: An llm-powered framework for on-the-fly reusable and scalable circular genome visualizations. *IEEE Computer Graphics and Applications*, 45:78–92, 2025. doi: 10.1109/MCG.2025.3581560 2

[28] G. Zhao, Z. Wang, Y. Dong, G. Li, and G. Shan. Toward reliable scientific visualization pipeline construction with structure-aware retrieval-augmented llms. *Information Visualization*, 25:373–390, 2026. doi: 10.1177/14738716261434848 2

[29] Y. Zhao, J. Wang, L. Xiang, X. Zhang, Z. Guo, C. Turkay, Y. Zhang, and S. Chen. Lightva: Lightweight visual analytics with llm agent-based task planning and execution. *IEEE Transactions on Visualization and Computer Graphics*, 31(9):6162–6177, 2025. doi: 10.1109/TVCG.2024.3496112 2

[30] Y. Zhu, L. Wang, C. Yang, X. Lin, B. Li, W. Zhou, X. Liu, Z. Peng, T. Luo, Y. Li, et al. A survey of data agents: Emerging paradigm or overstated hype? *arXiv preprint arXiv:2510.23587*, 2025. doi: 10.48550/arXiv.2510.23587 1