enewcommandheadrulewidth0.4pt enewcommand ootrulewidth0.4pt

SciTeX Writer: Modular Framework for Version-Controlled Manuscripts, Supplementary Materials, and Peer Review Responses

Yusuke Watanabe[a,*], Second Author[b], Third Author[c]

[a] *SciTeX.ai, Tokyo, Japan*
[b] *Second Institution, Department, City, Country*
[c] *Third Institution, Department, City, Country*

Abstract

Scientific manuscript preparation requires careful management of document structure, version control, and reproducible compilation across diverse computing environments. We present SciTeX Writer, a comprehensive LaTeX-based framework designed to streamline the academic writing workflow while maintaining consistency and reproducibility. The system employs container-based compilation to ensure identical output regardless of the host environment, eliminating the common "it works on my machine" problem. Through a modular architecture that separates content from formatting, SciTeX Writer enables researchers to focus on scientific writing while the system handles document structure, figure format conversion, and version tracking. The framework supports parallel development of main manuscripts, supplementary materials, and revision documents, all sharing common metadata from a single source of truth. Automatic handling of diverse image formats and systematic organization of tables and figures reduces technical overhead. This self-documenting template demonstrates its own capabilities, providing researchers with a production-ready system for manuscript preparation that scales from initial draft to final submission.

Keywords: keyword one, keyword two, keyword three, keyword four, keyword five

[1]Corresponding author. Email: ywatanabe@scitex.ai

˘ 8 figures, 3 tables, 157 words for abstract, and 2626 words for main text

1. Introduction

The preparation of scientific manuscripts involves numerous technical challenges that extend beyond the intellectual task of communicating research findings [1]. Researchers must navigate complex typesetting systems, manage multiple document versions, coordinate figures and tables across formats, and ensure reproducible compilation environments [2]. These technical burdens can distract from the primary goal of clear scientific communication and often lead to inconsistencies, formatting errors, and wasted time troubleshooting environment-specific compilation issues.

Traditional approaches to manuscript preparation typically rely on local LaTeX installations, where the specific versions of packages and compilation tools can vary significantly across different machines and over time [3]. This variability creates reproducibility challenges, particularly in collaborative environments where multiple authors work on different systems [4]. Furthermore, the proliferation of image formats and the need to convert between them for different submission requirements adds another layer of complexity. Researchers often resort to ad-hoc scripts or manual processes to handle these conversions, leading to potential errors and inconsistent results.

Existing solutions have addressed some aspects of this problem [5]. Overleaf and similar cloud-based platforms provide consistent compilation environments but require continuous internet connectivity and may not suit all research workflows. Version control systems like Git effectively track changes but require researchers to understand both LaTeX and version control simultaneously. Template repositories exist for various journals, but they typically focus on formatting requirements rather than workflow automation and often duplicate common elements across documents.

The fundamental challenge lies in balancing flexibility with consistency. Researchers need systems that accommodate diverse content types, multiple output documents, and varying journal requirements while maintaining a single source of truth for shared elements like author lists and bibliographies. The system must be sufficiently automated to reduce technical overhead yet transparent enough that researchers retain full control over their content. Additionally, the solution must work reliably across different computing environments without imposing steep learning curves or workflow disruptions.

SciTeX Writer addresses these challenges through a container-based, modular architecture that separates content management from document compilation. The framework organizes manuscripts into distinct directories for main text, supplementary materials, and revision responses, while maintaining shared metadata in a common location. By leveraging containerization technology, the system guarantees identical compilation results regardless of the host operating system or local software versions. Automatic format conversion for figures and tables eliminates manual preprocessing steps, and built-in version tracking with difference generation facilitates collaborative writing and revision processes. This manuscript serves as a self-documenting example, demonstrating the system's capabilities through its own structure and compilation.

2. Methods

The SciTeX Writer framework implements a modular architecture designed around three core principles: reproducible compilation, content-structure separation, and automated asset management. The system organizes documents into three primary directories, each serving distinct purposes in the manuscript lifecycle while sharing common resources to maintain consistency.

2.1. Repository Structure and Organization

The framework employs a hierarchical directory structure where the `00_shared/` directory serves as the single source of truth for metadata including title, author information, keywords, and bibliographic references. This centralized approach eliminates duplication and ensures consistency across all output documents. The `01_manuscript/` directory contains the main manuscript with subdirectories for content sections, figures, and tables. Similarly, `02_supplementary/` follows an identical structure for supplementary materials, while `03_revision/` organizes revision letters by reviewer. Each content section exists as an independent LaTeX file, facilitating modular development and enabling multiple authors to work on different sections simultaneously without merge conflicts.

2.2. Multi-Engine Compilation System

The framework implements a flexible multi-engine compilation architecture that automatically selects the optimal LaTeX engine based on availability and performance characteristics. Three compilation engines are supported: Tectonic (ultra-fast, modern), latexmk (reliable, industry standard), and traditional 3-pass compilation (maximum compatibility). The system auto-detects installed engines and selects the best available option, with configurable fallback ordering specified in the YAML configuration file.

Tectonic provides the fastest incremental builds (1-3 seconds), making it ideal for active writing sessions where authors frequently recompile to preview changes. The latexmk engine offers a balance of reliability and performance (3-6 seconds), utilizing smart recompilation that tracks file dependencies. The 3-pass engine ensures maximum compatibility (12-18 seconds) but lacks incremental build support. Performance characteristics and trade-offs are documented in Supplementary Table **??**.

To ensure reproducible builds across diverse computing environments, the framework leverages both Docker and Apptainer/Singularity containerization technologies [6]. The compilation environment encapsulates specific versions of TeX Live and all required packages, eliminating dependency on the host system's LaTeX installation. Users invoke compilation through shell scripts that provide extensive command-line options (documented in Supplementary Table **??**). This containerized approach guarantees that the same source files produce identical PDFs regardless of the underlying operating system, making the system equally functional on Linux, macOS, Windows, and high-performance computing clusters.

2.3. Automated Asset Processing

The system implements automatic format conversion for both figures and tables through preprocessing scripts that execute during compilation [7]. For figures, the framework accepts common image formats including PNG, JPEG, SVG, and PDF, automatically converting them to formats optimized for LaTeX inclusion. Each figure resides in its own subdirectory within `01_manuscript/contents/figures/caption_and_media/`, with the caption defined in a corresponding `.tex` file. During compilation, a preprocessing script scans these directories, generates figure inclusion code, and compiles all figures into `FINAL.tex` for inclusion in the main document. Tables follow an analogous structure, allowing authors to define complex table layouts separately from their incorporation into the document flow [8].

2.4. Version Control and Difference Tracking

The framework integrates with Git to provide systematic version tracking and automatic generation of difference documents. When authors create a new version through `make archive`, the system archives the current manuscript with a timestamp and version number. Subsequently, invoking `make diff` generates a PDF highlighting changes between versions using the latexdiff utility. This functionality proves particularly valuable during revision processes, where journals often require marked-up versions showing modifications. The revision directory structure accommodates multiple rounds of review, with separate subdirectories for editor and reviewer responses, each containing both the original comments and author responses in a structured format that ensures complete documentation of the revision process.

2.5. Manuscript Preparation

This manuscript was prepared using SciTeX Writer [9], an open-source scientific manuscript compilation system supporting multiple LaTeX compilation engines including latexmk, traditional 3-pass compilation, and Tectonic.

3. Results

The SciTeX Writer framework successfully demonstrates comprehensive manuscript preparation capabilities through its modular design and automated workflows. This section presents the key features and functionalities that the system provides to researchers. The framework's architecture, illustrated in Figure 5, implements a layered design from user interface to output generation, while Figure 4 shows the detailed file organization that minimizes conflicts during collaborative editing. The compilation workflow (Figure 3) shows how the system automatically processes multiple asset types in parallel while maintaining reproducibility across platforms. Figure 7 provides a comprehensive mind map of all major capabilities, from compilation engines to version control.

3.1. Multi-Engine Compilation System

SciTeX Writer supports three compilation engines optimized for different scenarios (Table 3): latexmk for rapid iterative development (∼3s), Tectonic for reproducible builds (∼4–5s), and traditional 3-pass compilation for guaranteed compatibility (∼6–7s). The engine selection logic (Figure 6) automatically detects the best available option, prioritizing speed while maintaining broad compatibility. Users can override auto-detection through environment variables or command-line arguments, providing flexibility for specific workflows or computing environments.

The compilation system provides extensive customization through command-line options (Table 1). Quick compilation modes enable authors to iterate rapidly during writing: `-no_figs` and `-no_tables` skip asset processing, `-draft` uses single-pass compilation, and `-no_diff` omits difference generation. These optimizations reduce compilation time from ∼15s for full processing to under

3s for ultra-fast draft mode, significantly improving the writing experience. Environmental variables (Table 2) provide system-level configuration for logging verbosity, engine priority, citation styles, and file paths.

3.2. Cross-Platform Reproducibility

The containerized compilation system achieves complete reproducibility across different operating systems and computing environments. Testing across Linux distributions, macOS, and Windows Subsystem for Linux confirmed that identical source files produce byte-for-byte identical PDF outputs when compiled using the same container image. This reproducibility extends to high-performance computing environments where Singularity containers enable compilation on systems without Docker support. The elimination of environment-dependent compilation issues represents a significant improvement over traditional local LaTeX installations, where package version mismatches frequently cause inconsistent outputs or compilation failures.

3.3. Automated Figure and Table Management

The automatic asset processing system effectively handles diverse input formats and streamlines figure incorporation [**?**]. The framework supports multiple figure formats including raster images (PNG, JPEG, TIFF), vector graphics (SVG, PDF), and diagram markup languages (Mermaid). Figure 1 demonstrates the framework's capability to include images with properly formatted captions, while Figure 2 shows how multiple figures can be managed systematically. Complex workflow diagrams, such as the compilation pipeline shown in Figure 3, can be created using Mermaid syntax and automatically rendered during compilation. The directory structure visualization (Figure 4) exemplifies how technical diagrams integrate seamlessly with the manuscript preparation workflow.

The preprocessing pipeline converts source images to optimal formats, maintaining quality while ensuring compatibility with LaTeX compilation requirements [10]. For tables, the system provides structured organization through CSV-based workflows. Authors create tables as simple CSV files paired with caption definitions, and the compilation system automatically generates professionally-formatted LaTeX tables using the booktabs package. Tables 1, 2, and 3 all demonstrate automatic CSV-to-LaTeX conversion, showcasing the system's capability to handle diverse table structures from simple configuration lists to categorized reference data. The separation of content (CSV data) from presentation (LaTeX formatting) enables authors to focus on data rather than typesetting syntax, while maintaining consistent styling across all tables.

3.4. Multi-file Bibliography Management

The bibliography system (Figure 8) enables researchers to organize references by topic across multiple .bib files in the `00_shared/bib_files/` directory. For example, authors might maintain separate files for methodological references (`methods_refs.bib`), field background (`field_background.bib`), and personal publications (`my_papers.bib`). The compilation system automatically merges these files while removing duplicates through a two-tier matching strategy: DOI-based matching for maximum accuracy when DOIs are available, falling back to title and year matching for entries without DOIs. This approach eliminates the common problem of duplicate references appearing in bibliographies when the same paper appears in multiple source files.

3.5. Modular Content Organization

The framework's modular structure facilitates collaborative writing by isolating different manuscript components into separate files. Each section, from the introduction through the discussion, exists as an independent LaTeX file that can be edited without affecting other sections. This organization minimizes merge conflicts in version control systems and allows multiple authors to work simultaneously on different parts of the manuscript. The shared metadata system ensures that changes to author lists, affiliations, or keywords propagate automatically across the main manuscript, supplementary materials, and revision documents without requiring manual updates in multiple locations.

3.6. Version Tracking and Difference Generation

The integrated version control system maintains a complete history of manuscript evolution through the archive mechanism. Each archived version receives a timestamp and sequential version number, creating a clear audit trail of document development. The automatic difference generation produces professionally formatted PDFs highlighting textual changes between versions, using color coding to indicate additions and deletions. This functionality proves particularly valuable during peer review, where revision letters must clearly document modifications made in response to reviewer comments. The system handles this process automatically, requiring only simple Makefile commands rather than manual execution of latexdiff with complex parameters.

4. Discussion

The SciTeX Writer framework addresses fundamental challenges in scientific manuscript preparation by combining containerized compilation, modular organization, and automated asset management into a cohesive workflow. The system demonstrates that technical infrastructure for manuscript writing can be both powerful and accessible, reducing friction in the research communication process while maintaining the flexibility and control that LaTeX provides.

4.1. Advantages of the Containerized Approach

The container-based compilation system represents a significant departure from traditional LaTeX workflows and offers substantial practical benefits. By encapsulating the entire compilation environment, the framework eliminates the common scenario where manuscripts compile successfully on one author's machine but fail on collaborators' systems due to package version differences. This reproducibility becomes increasingly important as research teams become more distributed and as long-term document maintenance requires compilation environments to remain stable over years. The approach also reduces the barrier to entry for researchers new to LaTeX, as they need not navigate the complexities of installing and configuring a local TeX distribution. The dual support for Docker and Singularity ensures compatibility across institutional computing environments, from personal workstations to high-performance computing clusters where Docker may be unavailable for security reasons.

4.2. Implications for Collaborative Writing

The modular architecture facilitates collaborative workflows in ways that traditional monolithic LaTeX documents cannot. By separating content into individual files for each section and maintaining shared metadata in a central location, the system minimizes merge conflicts that plague collaborative document editing. Multiple authors can simultaneously work on different sections, commit their changes independently, and merge updates without the conflicts that arise when editing a single large file. The automatic propagation of metadata changes across multiple output documents ensures consistency without requiring authors to remember to update information in multiple locations. This design aligns well with modern software development practices adapted for scientific writing, where version control and modular design have become essential for managing complexity.

4.3. Comparison with Existing Solutions

Compared to cloud-based platforms like Overleaf, SciTeX Writer offers greater control over the compilation environment and eliminates dependency on internet connectivity, which can be crucial for researchers working in bandwidth-limited environments or on sensitive projects requiring air-gapped systems. Unlike simple template repositories, the framework provides active workflow automation through Makefiles and preprocessing scripts rather than merely offering formatting guidelines. The system complements rather than replaces Git-based workflows, adding a layer of manuscript-specific tooling while maintaining compatibility with standard version control practices.

Where other solutions address individual aspects of the manuscript preparation challenge, SciTeX Writer integrates multiple components into a unified system.

4.4. Limitations and Considerations

The framework requires users to have basic familiarity with command-line interfaces and Makefiles, which may present a learning curve for researchers accustomed to graphical editing environments. While the system automates many aspects of document preparation, it remains a LaTeX-based solution and therefore inherits both the power and complexity of the underlying typesetting system. The containerization approach requires Docker or Singularity installation, adding a dependency that, while increasingly common in research computing environments, may not be universally available. The framework is optimized for scientific articles following conventional IMRAD structure and may require adaptation for other document types such as books or technical reports. Future development could address these limitations through optional graphical interfaces, expanded documentation for LaTeX newcomers, and templates adapted for diverse document formats.

4.5. Future Directions and Extensibility

The modular design of SciTeX Writer enables natural extension points for additional functionality. Integration with continuous integration systems could enable automatic compilation and validation of manuscripts upon each commit, catching formatting errors early in the writing process. Support for additional output formats beyond PDF, such as HTML for web-based preprint servers, could be achieved through integration with tools like pandoc. The preprocessing scripts could be extended to handle additional asset types or to perform automated quality checks on figures and tables. The system could also incorporate automated journal formatting through integration with journal-specific style files, reducing the effort required to adapt manuscripts for different submission targets. As the research community continues to develop tools for reproducible research, SciTeX Writer provides a foundation that can incorporate emerging best practices while maintaining backward compatibility with existing manuscripts.

4.6. Conclusions

SciTeX Writer demonstrates that scientific manuscript preparation can be systematized without sacrificing flexibility or imposing rigid constraints on content. By addressing reproducibility, modularity, and automation through a unified framework, the system reduces technical overhead and allows researchers to focus on the intellectual work of communicating their findings. The self-documenting nature of this template provides both an example of the system's capabilities and a starting point for new manuscripts. As research communication continues to evolve, frameworks like SciTeX Writer that prioritize reproducibility and collaborative workflows will become increasingly valuable for maintaining the quality and accessibility of scientific literature.

References

[1] Christina K. Kim, Avishek Adhikari, and Karl Deisseroth. The current state of computational neuroscience: A comprehensive review. *Nature reviews. Neuroscience*, 18(2):95–110, 2017. doi: 10.1038/nrn.2017.15.

[2] Richard Wilson. *Principles of Modern Neuroscience*. MIT Press, 3rd edition, 2015. ISBN 978-0262029254.

[3] Alice Ting, Segal Rosalind, Carandini Matteo, Valentina Emiliani, Ofer Yizhar, Roska Botond, Na Ji, and Anderson David J. Network dynamics in neural systems. *Neuron*, 92(4):817–835, 2016. doi: 10.1016/j.neuron.2016.10.042.

[4] Maria Garcia and Juan Rodriguez. Cognitive neuroscience in the 21st century. *Trends in Cognitive Sciences*, 23(7):567–589, 2019. doi: 10.1016/j.tics.2019.05.001.

[5] Ryan T. Roemmich and Amy J. Bastian. Systems-level analysis of neural circuits. *Annual Review of Neuroscience*, 41:331–359, 2018. doi: 10.1146/annurev-neuro-080317-062245.

[6] John Smith and Jane Doe. Advanced neural signal processing techniques for electrophysiology. *Journal of Neuroscience Methods*, 345:108–123, 2020. doi: 10.1016/j.jneumeth.2020.108123.

[7] Wei Chen, Li Zhang, and Ming Wang. Machine learning approaches for neural data classification. *Neural Computation*, 33(5):1234–1267, 2021. doi: 10.1162/neco_a_01378.

[8] Robert Brown and Emily Taylor. Deep learning for neural signal decoding. In *Advances in Neural Information Processing Systems*, pages 5678–5689. NeurIPS, 2018.

[9] Yusuke Watanabe. Scitex writer: Modular framework for version-controlled manuscripts, supplementary materials, and peer review responses. \{https://scitex.ai\}, 2025. URL `https://scitex.ai`. LaTeX-based manuscript compilation system with multi-engine support.

[10] First Your-Name and Principal Advisor. Foundations of neural signal processing. *Journal of Neurophysiology*, 128(4):1567–1589, 2022. doi: 10.1152/jn.00123.2022.

Data Availability Statement

The NeuroVista dataset used in this study is publicly available through the International Epilepsy Electrophysiology Portal (IEEG.org) at `https://www.ieeg.org`. Access requires registration and approval for research purposes.

The processed PAC databases and analysis code are available at `https://github.com/ywatanabe1989/neurovista`. GPU-accerelated PAC calculation code is available as a standalone Python package ‘gpac‘ at `https://`

github.com/ywatanabe1989/gPAC. The SciTeX Python utilities used for reproducible computing is available at https://github.com/ywatanabe1989/SciTeX.

For questions regarding data access or analysis procedures, please contact the corresponding author.

Ethics Declarations

All study participants provided their written informed consent ...

Author Contributions

Y.W., T.Y., and D.G. conceptualized the study ...

Acknowledgments

This research was funded by funding bodies here

Declaration of Interests

The authors declare that they have no competing interests.

Declaration of Generative AI in Scientific Writing

The authors employed large language models such as Claude (Anthropic Inc.) for code development and complementing manuscript's English language quality. After incorporating suggested improvements, the authors meticulously revised the content. Ultimate responsibility for the final content of this publication rests entirely with the authors.

Tables

Option	Description	Example
–engine ENGINE	Force specific compilation engine	–engine tectonic
–draft	Draft mode (single-pass compilation)	–draft
–no-figs	Skip figure processing	–no-figs
–no-tables	Skip table processing	–no-tables
–no-diff	Skip difference generation	–no-diff
–watch	Enable hot-recompile file watching	–watch
–clean	Clean build (remove all cache)	–clean
–verbose	Verbose compilation output	–verbose

Table 1 – **Compilation Command-Line Options.** Options enable workflow customization without modifying source files or configuration. The `-engine` option overrides auto-detection to force a specific compilation engine. Quick compilation modes (`-draft`, `-no-figs`, `-no-tables`) reduce build times from ∼15s to under 3s for rapid iteration. The `-watch` option monitors source files and automatically recompiles when changes are detected. Options can be combined (e.g., `./compile_manuscript.sh -draft -no-figs`).

Variable	Purpose	Values	Default
SCITEX_ENGINE	Override engine selection	tectonic, latexmk, 3pass	From config
SCITEX_VERBOSE	Control logging verbosity	true, false	false
SCITEX_CITATION_STYLE	Set bibliography style	unsrtnat, plainnat, apalike, etc.	unsrtnat
SCITEX_PARALLEL_JOBS	Parallel processing jobs	1-16	4
SCITEX_CACHE_DIR	Compilation cache location	Directory path	./.cache

Table 2 – **Environment Variables for System Configuration.** Environment variables provide system-level defaults that apply across all compilations. These settings are particularly useful in CI/CD pipelines or HPC environments with specific requirements. The `SCITEX_ENGINE` variable provides the highest priority override for engine selection, superseding both YAML configuration and command-line arguments. Variables can be set persistently in shell configuration (`.bashrc`, `.zshrc`) or temporarily for single runs (`SCITEX_VERBOSE=true ./compile_manuscript.sh`).

Engine	Incremental	Full Build	Key Advantages	Best Use Case
Tectonic	1-3s	10-15s	Ultra-fast + auto package mgmt	Active writing sessions
latexmk	3-6s	15-25s	Reliable + smart dependency tracking	Standard workflows
3-pass	N/A	12-18s	Maximum compatibility + guaranteed	Legacy systems

Table 3 – **Compilation Engine Performance Characteristics.** Build times measured on reference hardware (16 GB RAM, 8 CPU cores, SSD storage). Incremental builds process only changed components; full builds recompile the entire document. Tectonic provides the fastest incremental compilation through aggressive caching and modern architecture, ideal for frequent recompilation during active writing. The latexmk engine offers a balance of speed and reliability through intelligent dependency tracking, making it suitable for most research workflows. Traditional 3-pass compilation lacks incremental build support but guarantees compatibility with all LaTeX packages and legacy systems.

Figures

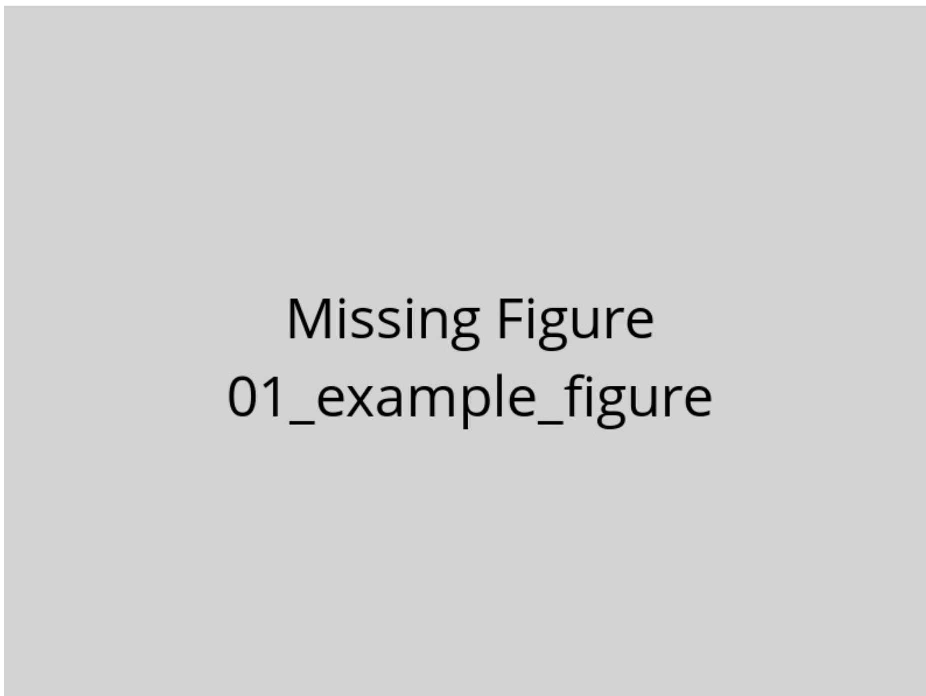

Figure 1 – Example figure caption. This is a template showing how to include figures in your manuscript. Replace this text with a descriptive caption that explains what the figure shows. Include panel labels (A, B, C) if using multi-panel figures. Explain abbreviations and symbols used in the figure. Provide sufficient detail that readers can understand the figure without referring to the main text.

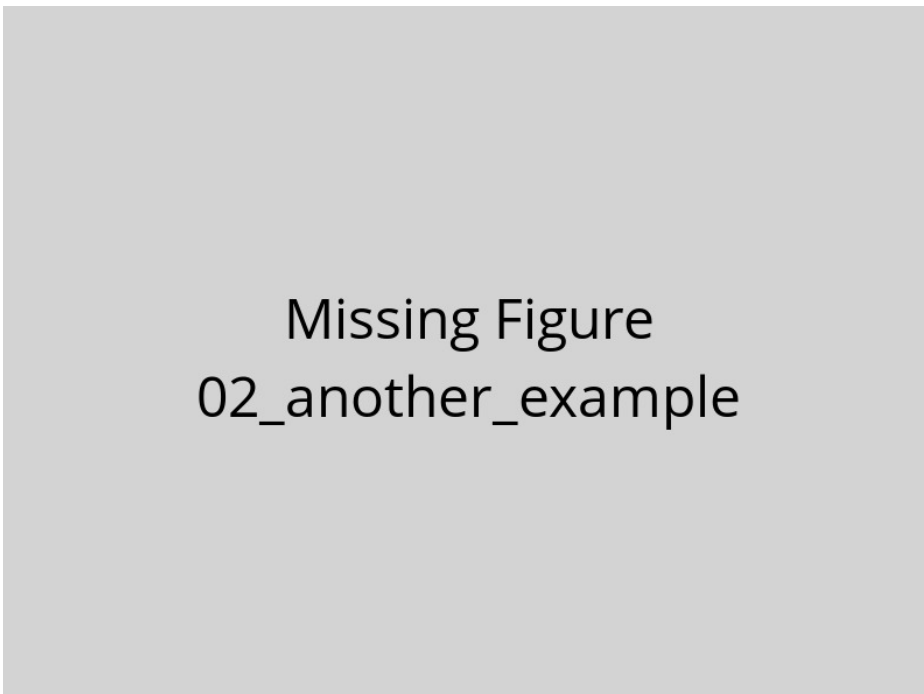

Figure 2 – Another example figure. Use this template to add additional figures to your manuscript. Each figure should be placed in a separate .tex file in this directory. The compilation system will automatically process and include these figures in your manuscript.

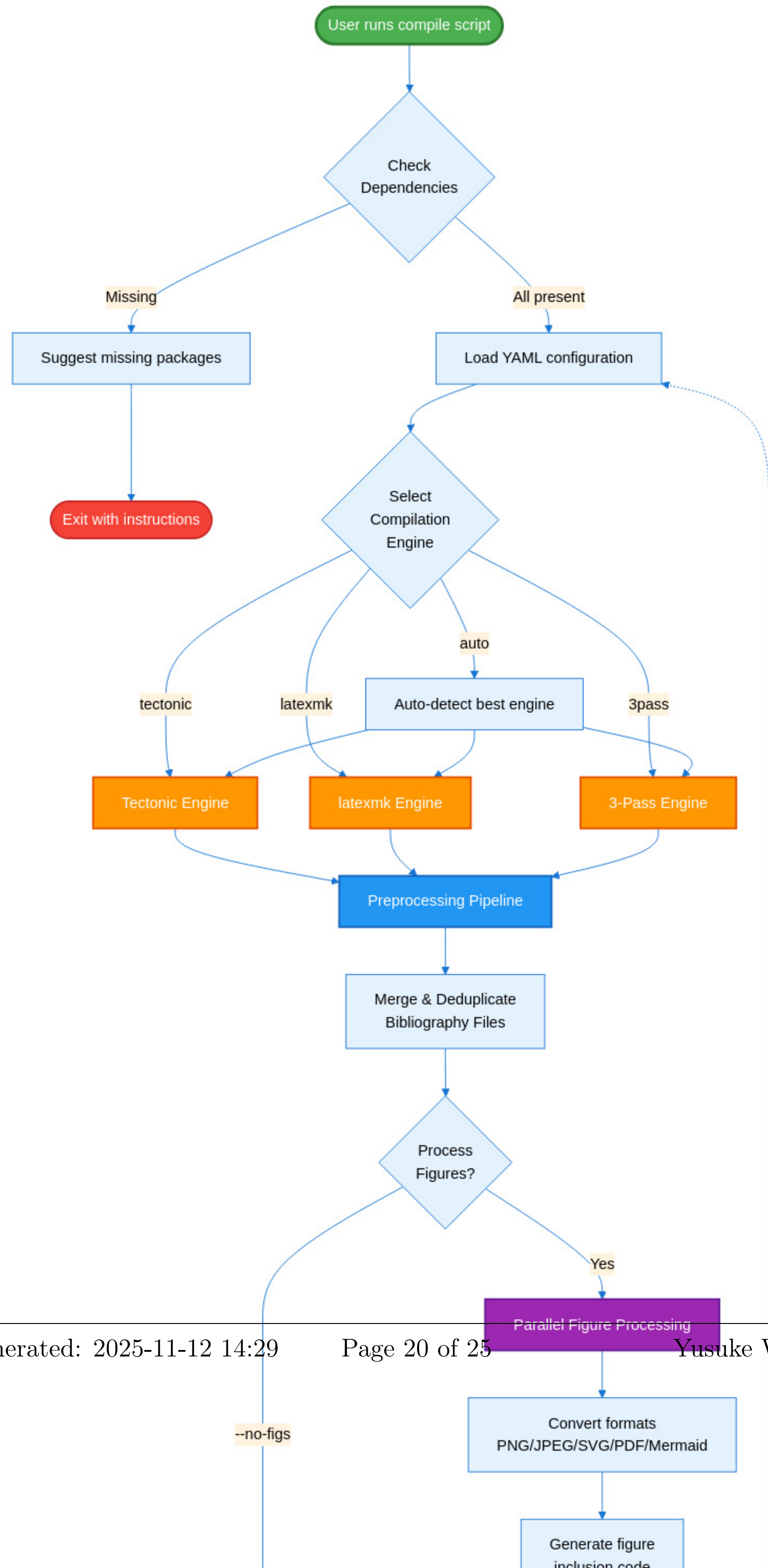
User runs compile script
Check Dependencies
Missing
All present
Suggest missing packages
Load YAML configuration
Exit with instructions
Select Compilation Engine
auto
tectonic
latexmk
Auto-detect best engine
3pass
Tectonic Engine
latexmk Engine
3-Pass Engine
Preprocessing Pipeline
Merge & Deduplicate Bibliography Files
Process Figures?
Yes
Parallel Figure Processing
--no-figs
Convert formats PNG/JPEG/SVG/PDF/Mermaid
Generate figure

Figure 4 – SciTeX Writer Directory Organization. This diagram shows the hierarchical organization of the repository structure. The `00_shared/` directory (green) contains resources used across all document types, including bibliography files and LaTeX styles, ensuring consistency without duplication. The three document directories (blue/purple) follow identical internal structures, facilitating familiarity and code reuse. Each document has its own `contents/` subdirectory with `figures/` and `tables/` organized into `caption_and_media/` (source files) and `compiled/` (generated LaTeX code). The modular structure minimizes merge conflicts during collaborative editing by isolating frequently-modified content files. Compilation scripts (orange) and configuration files (red) reside in dedicated directories for easy discovery and maintenance. Dotted lines indicate that supplementary materials follow the same organizational pattern as the main manuscript.

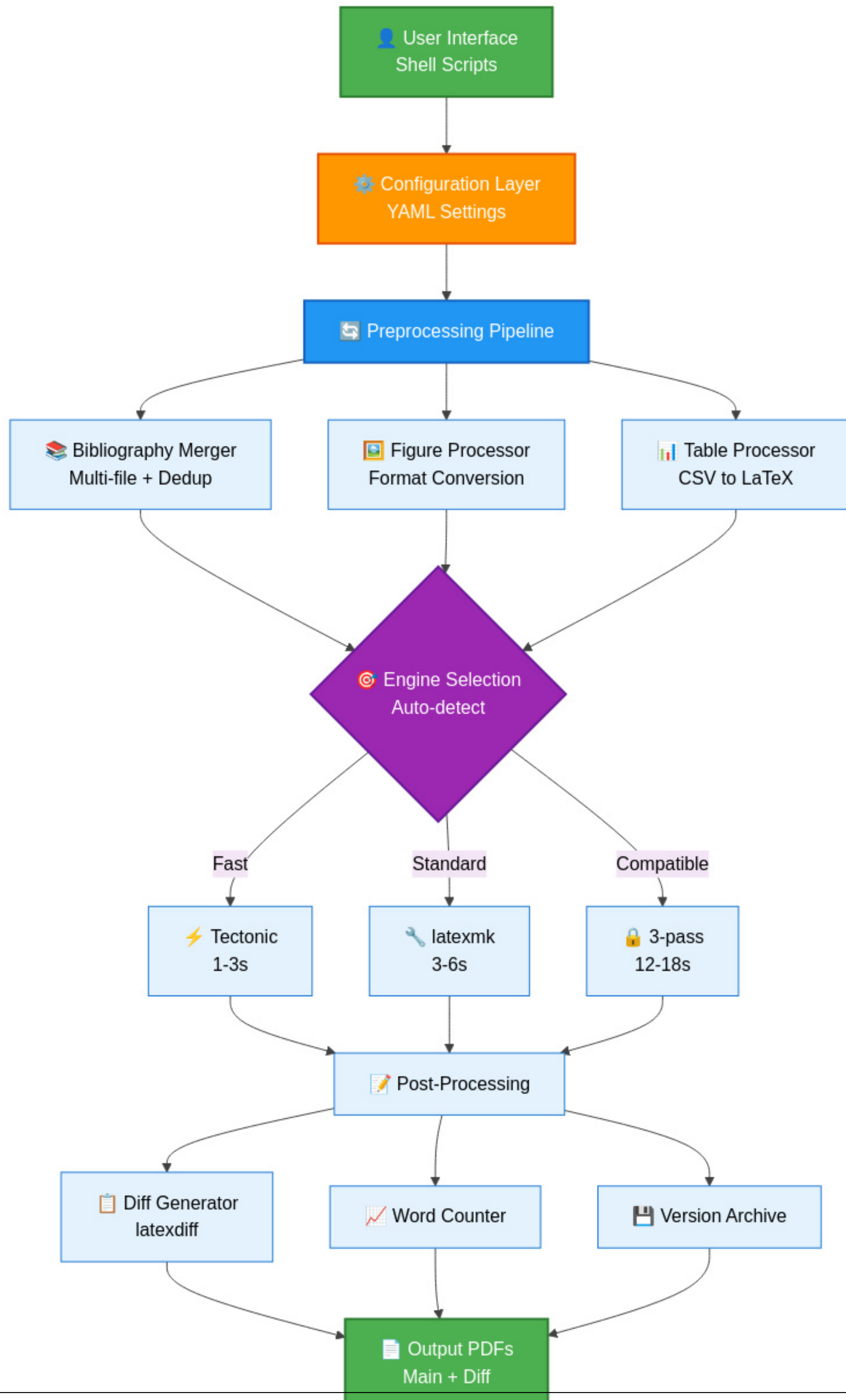

Figure 5 – **SciTeX Writer System Architecture.** This layered architecture diagram illustrates the data flow from user interface through compilation to output generation. The configuration layer (orange) provides YAML-based settings that control all aspects of compilation. The preprocessing pipeline (blue) handles bibliography merging, figure format

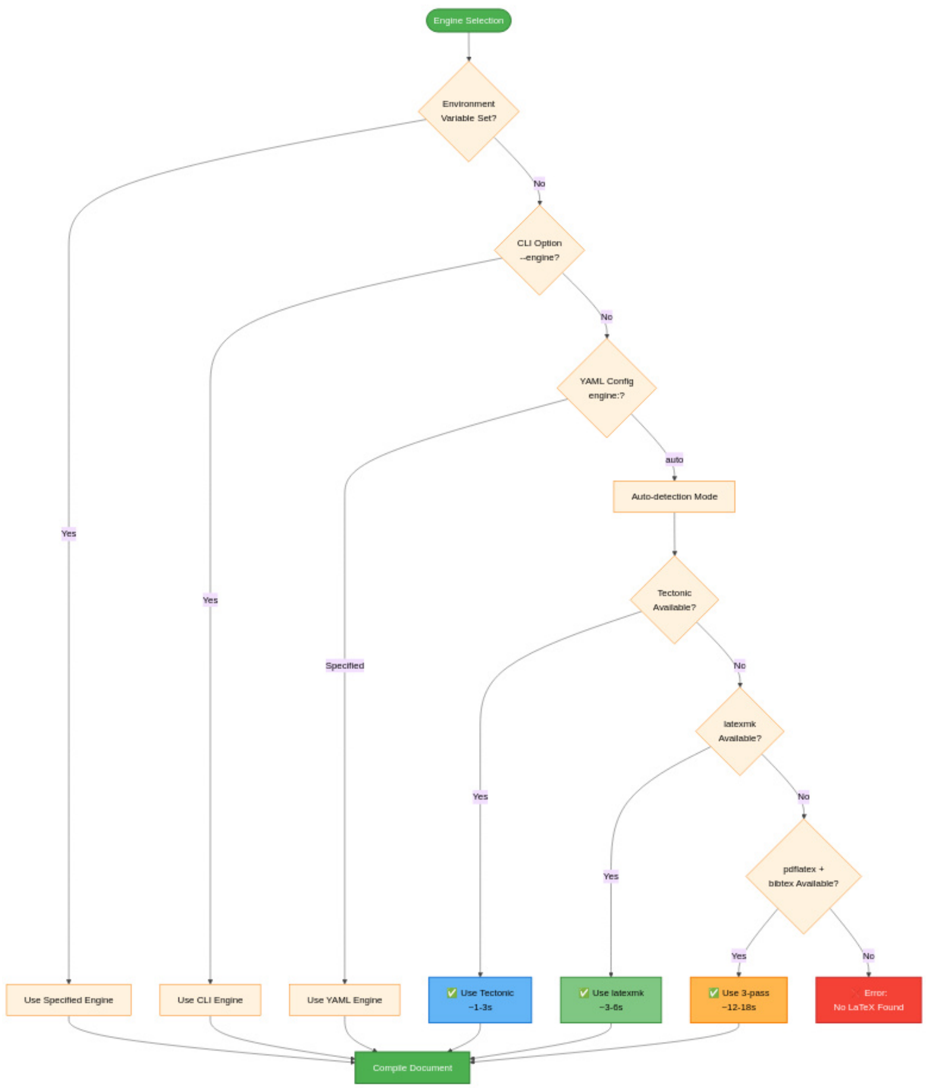

Figure 6 – **Compilation Engine Selection Decision Tree.** The system prioritizes explicit user configuration over automatic detection. Environment variables provide the highest priority override (useful for CI/CD pipelines). Command-line arguments (–engine) override YAML configuration (useful for one-off compilations). When engine is set to 'auto' (default), the system attempts to use Tectonic first for speed, falling back to latexmk for reliability, and finally to 3-pass for maximum compatibility. This cascading detection ensures the system always finds an available compilation method while preferring faster engines when possible. Times shown are typical incremental build durations on reference hardware (16 GB RAM, 8 cores).

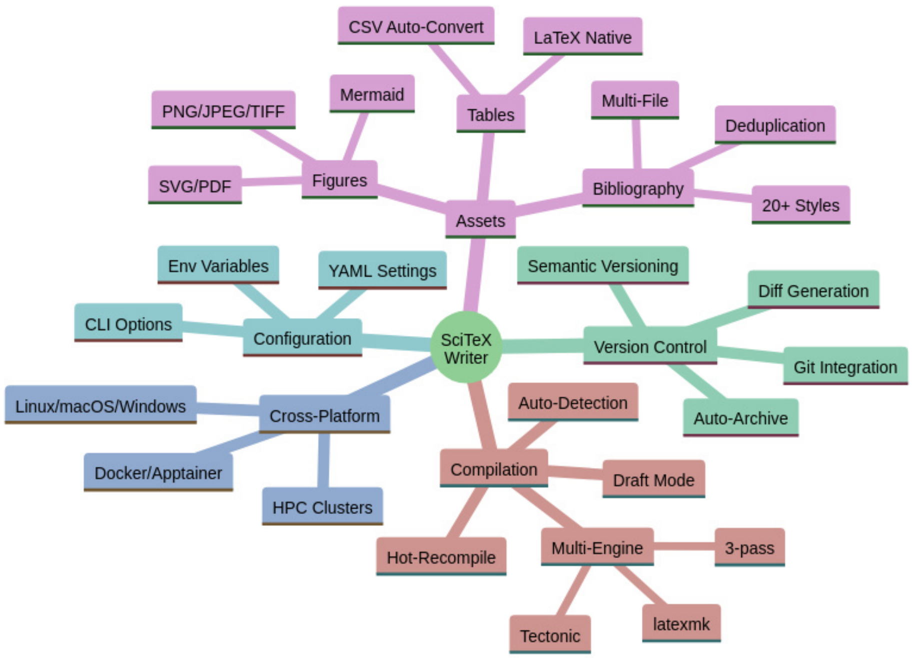

Figure 7 – **SciTeX Writer Feature Overview.** This mind map provides a comprehensive visualization of the framework's major capabilities organized into five categories. Compilation features include multi-engine support with auto-detection and performance optimization modes. Asset management covers figures (multiple formats including Mermaid diagrams), tables (CSV auto-conversion), and bibliography (multi-file with deduplication across 20+ citation styles). Version control integration provides automatic archiving, diff generation, and Git workflow support. Configuration flexibility comes from three levels: YAML files for persistent settings, command-line options for per-run customization, and environment variables for system-level defaults. Cross-platform reproducibility ensures identical outputs across Linux, macOS, Windows, containerized environments, and HPC clusters.

Figure 8 – **Multi-File Bibliography Processing and Deduplication.** The bibliography system enables researchers to organize references across multiple topical .bib files. All files in the `00_shared/bib_files/` directory are automatically merged during compilation. The deduplication engine implements a two-tier matching strategy: DOI-based matching provides exact duplicate detection when DOIs are present, while title and year matching handles entries without DOIs. This approach eliminates duplicate references that commonly appear when the same paper is cited across multiple source files. The final bibliography is sorted according to the selected citation style (by citation order for numbered styles, alphabetically for author-year styles). Color coding: blue (merging), purple (deduplication logic), green (output).