

```
1     \newcommand{\headrule}{\rule{0.4pt}{0.4pt} \newcommand{\footrule}{\rule{0.4pt}{0.4pt}}
```

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

6 Yusuke Watanabe<sup>a,\*</sup>, Second Author<sup>b</sup>, Third Author<sup>c</sup>

7 <sup>a</sup>*SciTeX.ai, Tokyo, Japan*

8 <sup>b</sup>*Second Institution, Department, City, Country*

9 <sup>c</sup>*Third Institution, Department, City, Country*

---

10 **Abstract**

11

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

29 *Keywords:* keyword one, keyword two, keyword three, keyword four,  
30 keyword five

---

31        ^ 8 [figures](#), 3 [tables](#), 157 words for abstract, and 2626 words for main  
32 text

### 33 1. Introduction

34        The preparation of scientific manuscripts involves numerous technical  
35 challenges that extend beyond the intellectual task of communicating re-  
36 search findings [1]. Researchers must navigate complex typesetting systems,  
37 manage multiple document versions, coordinate figures and tables across for-  
38 mats, and ensure reproducible compilation environments [2]. These technical  
39 burdens can distract from the primary goal of clear scientific communication  
40 and often lead to inconsistencies, formatting errors, and wasted time trou-  
41 bleshooting environment-specific compilation issues.

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

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

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

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

## 79 2. Methods

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

### 85 2.1. Repository Structure and Organization

86 The framework employs a hierarchical directory structure where the `00_shared/`  
87 directory serves as the single source of truth for metadata including title, au-  
88 thor information, keywords, and bibliographic references. This centralized

89 approach eliminates duplication and ensures consistency across all output  
90 documents. The 01\_manuscript/ directory contains the main manuscript  
91 with subdirectories for content sections, figures, and tables. Similarly, 02\_supplementary/  
92 follows an identical structure for supplementary materials, while 03\_revision/  
93 organizes revision letters by reviewer. Each content section exists as an inde-  
94 pendent LaTeX file, facilitating modular development and enabling multiple  
95 authors to work on different sections simultaneously without merge conflicts.

96 *2.2. Multi-Engine Compilation System*

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

104 Tectonic provides the fastest incremental builds (1-3 seconds), making it  
105 ideal for active writing sessions where authors frequently recompile to preview  
106 changes. The latexmk engine offers a balance of reliability and performance  
107 (3-6 seconds), utilizing smart recompilation that tracks file dependencies.  
108 The 3-pass engine ensures maximum compatibility (12-18 seconds) but lacks  
109 incremental build support. Performance characteristics and trade-offs are  
110 documented in Supplementary Table ??.

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

119 making the system equally functional on Linux, macOS, Windows, and high-  
120 performance computing clusters.

121 *2.3. Automated Asset Processing*

122 The system implements automatic format conversion for both figures  
123 and tables through preprocessing scripts that execute during compilation [7].  
124 For figures, the framework accepts common image formats including PNG,  
125 JPEG, SVG, and PDF, automatically converting them to formats optimized  
126 for LaTeX inclusion. Each figure resides in its own subdirectory within  
127 01\_manuscript/contents/figures/caption\_and\_media/, with the caption  
128 defined in a corresponding .tex file. During compilation, a preprocessing  
129 script scans these directories, generates figure inclusion code, and compiles  
130 all figures into FINAL.tex for inclusion in the main document. Tables fol-  
131 low an analogous structure, allowing authors to define complex table layouts  
132 separately from their incorporation into the document flow [8].

133 *2.4. Version Control and Difference Tracking*

134 The framework integrates with Git to provide systematic version track-  
135 ing and automatic generation of difference documents. When authors cre-  
136 ate a new version through `make archive`, the system archives the current  
137 manuscript with a timestamp and version number. Subsequently, invoking  
138 `make diff` generates a PDF highlighting changes between versions using  
139 the `latexdiff` utility. This functionality proves particularly valuable during  
140 revision processes, where journals often require marked-up versions show-  
141 ing modifications. The revision directory structure accommodates multiple  
142 rounds of review, with separate subdirectories for editor and reviewer re-  
143 sponds, each containing both the original comments and author responses  
144 in a structured format that ensures complete documentation of the revision  
145 process.

146 *2.5. Manuscript Preparation*

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

150 **3. Results**

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

162 *3.1. Multi-Engine Compilation System*

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

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

176 3s for ultra-fast draft mode, significantly improving the writing experience.  
177 Environmental variables (Table 2) provide system-level configuration for logging  
178 verbosity, engine priority, citation styles, and file paths.

179 *3.2. Cross-Platform Reproducibility*

180 The containerized compilation system achieves complete reproducibility  
181 across different operating systems and computing environments. Testing  
182 across Linux distributions, macOS, and Windows Subsystem for Linux con-  
183 firmed that identical source files produce byte-for-byte identical PDF outputs  
184 when compiled using the same container image. This reproducibility extends  
185 to high-performance computing environments where Singularity containers  
186 enable compilation on systems without Docker support. The elimination of  
187 environment-dependent compilation issues represents a significant improve-  
188 ment over traditional local LaTeX installations, where package version mis-  
189 matches frequently cause inconsistent outputs or compilation failures.

190 *3.3. Automated Figure and Table Management*

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

202 The preprocessing pipeline converts source images to optimal formats,  
203 maintaining quality while ensuring compatibility with LaTeX compilation  
204 requirements [10]. For tables, the system provides structured organization  
205 through CSV-based workflows. Authors create tables as simple CSV files

206 paired with caption definitions, and the compilation system automatically  
207 generates professionally-formatted LaTeX tables using the booktabs package.  
208 Tables 1, 2, and 3 all demonstrate automatic CSV-to-LaTeX conversion,  
209 showcasing the system's capability to handle diverse table structures from  
210 simple configuration lists to categorized reference data. The separation of  
211 content (CSV data) from presentation (LaTeX formatting) enables authors  
212 to focus on data rather than typesetting syntax, while maintaining consistent  
213 styling across all tables.

### 214 3.4. *Multi-file Bibliography Management*

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

### 225 3.5. *Modular Content Organization*

226 The framework's modular structure facilitates collaborative writing by  
227 isolating different manuscript components into separate files. Each section,  
228 from the introduction through the discussion, exists as an independent La-  
229 TeX file that can be edited without affecting other sections. This organiza-  
230 tion minimizes merge conflicts in version control systems and allows multiple  
231 authors to work simultaneously on different parts of the manuscript. The  
232 shared metadata system ensures that changes to author lists, affiliations, or  
233 keywords propagate automatically across the main manuscript, supplemen-  
234 tary materials, and revision documents without requiring manual updates in  
235 multiple locations.

---

236 *3.6. Version Tracking and Difference Generation*

237     The integrated version control system maintains a complete history of  
238     manuscript evolution through the archive mechanism. Each archived version  
239     receives a timestamp and sequential version number, creating a clear audit  
240     trail of document development. The automatic difference generation pro-  
241     duces professionally formatted PDFs highlighting textual changes between  
242     versions, using color coding to indicate additions and deletions. This func-  
243     tionality proves particularly valuable during peer review, where revision let-  
244     ters must clearly document modifications made in response to reviewer com-  
245     ments. The system handles this process automatically, requiring only simple  
246     Makefile commands rather than manual execution of `latexdiff` with complex  
247     parameters.

248 **4. Discussion**

249     The SciTeX Writer framework addresses fundamental challenges in sci-  
250     entific manuscript preparation by combining containerized compilation, modu-  
251     lar organization, and automated asset management into a cohesive workflow.  
252     The system demonstrates that technical infrastructure for manuscript writing  
253     can be both powerful and accessible, reducing friction in the research com-  
254     munication process while maintaining the flexibility and control that LaTeX  
255     provides.

256 *4.1. Advantages of the Containerized Approach*

257     The container-based compilation system represents a significant depa-  
258     ture from traditional LaTeX workflows and offers substantial practical ben-  
259     efits. By encapsulating the entire compilation environment, the framework  
260     eliminates the common scenario where manuscripts compile successfully on  
261     one author's machine but fail on collaborators' systems due to package ver-  
262     sion differences. This reproducibility becomes increasingly important as re-  
263     search teams become more distributed and as long-term document mainte-  
264     nance requires compilation environments to remain stable over years. The

265 approach also reduces the barrier to entry for researchers new to LaTeX,  
266 as they need not navigate the complexities of installing and configuring a  
267 local TeX distribution. The dual support for Docker and Singularity en-  
268 sures compatibility across institutional computing environments, from per-  
269 sonal workstations to high-performance computing clusters where Docker  
270 may be unavailable for security reasons.

271 *4.2. Implications for Collaborative Writing*

272 The modular architecture facilitates collaborative workflows in ways that  
273 traditional monolithic LaTeX documents cannot. By separating content into  
274 individual files for each section and maintaining shared metadata in a cen-  
275 tral location, the system minimizes merge conflicts that plague collaborative  
276 document editing. Multiple authors can simultaneously work on different  
277 sections, commit their changes independently, and merge updates without  
278 the conflicts that arise when editing a single large file. The automatic propa-  
279 gation of metadata changes across multiple output documents ensures consis-  
280 tency without requiring authors to remember to update information in mul-  
281 tiple locations. This design aligns well with modern software development  
282 practices adapted for scientific writing, where version control and modular  
283 design have become essential for managing complexity.

284 *4.3. Comparison with Existing Solutions*

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

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

297 *4.4. Limitations and Considerations*

298 The framework requires users to have basic familiarity with command-  
299 line interfaces and Makefiles, which may present a learning curve for re-  
300 searchers accustomed to graphical editing environments. While the system  
301 automates many aspects of document preparation, it remains a LaTeX-based  
302 solution and therefore inherits both the power and complexity of the under-  
303 lying typesetting system. The containerization approach requires Docker or  
304 Singularity installation, adding a dependency that, while increasingly com-  
305 mon in research computing environments, may not be universally available.  
306 The framework is optimized for scientific articles following conventional IM-  
307 RAD structure and may require adaptation for other document types such  
308 as books or technical reports. Future development could address these lim-  
309 itations through optional graphical interfaces, expanded documentation for  
310 LaTeX newcomers, and templates adapted for diverse document formats.

311 *4.5. Future Directions and Extensibility*

312 The modular design of SciTeX Writer enables natural extension points  
313 for additional functionality. Integration with continuous integration systems  
314 could enable automatic compilation and validation of manuscripts upon each  
315 commit, catching formatting errors early in the writing process. Support  
316 for additional output formats beyond PDF, such as HTML for web-based  
317 preprint servers, could be achieved through integration with tools like pan-  
318 doc. The preprocessing scripts could be extended to handle additional asset  
319 types or to perform automated quality checks on figures and tables. The  
320 system could also incorporate automated journal formatting through inte-  
321 gration with journal-specific style files, reducing the effort required to adapt  
322 manuscripts for different submission targets. As the research community  
323 continues to develop tools for reproducible research, SciTeX Writer provides

324 a foundation that can incorporate emerging best practices while maintaining  
325 backward compatibility with existing manuscripts.

326 *4.6. Conclusions*

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

338 **References**

- 339 [1] Christina K. Kim, Avishek Adhikari, and Karl Deisseroth. The current  
340 state of computational neuroscience: A comprehensive review. *Nature*  
341 *reviews. Neuroscience*, 18(2):95–110, 2017. doi: 10.1038/nrn.2017.15.
- 342 [2] Richard Wilson. *Principles of Modern Neuroscience*. MIT Press, 3rd  
343 edition, 2015. ISBN 978-0262029254.
- 344 [3] Alice Ting, Segal Rosalind, Carandini Matteo, Valentina Emiliani, Ofer  
345 Yizhar, Roska Botond, Na Ji, and Anderson David J. Network dynamics  
346 in neural systems. *Neuron*, 92(4):817–835, 2016. doi: 10.1016/j.neuron.  
347 2016.10.042.
- 348 [4] Maria Garcia and Juan Rodriguez. Cognitive neuroscience in the 21st  
349 century. *Trends in Cognitive Sciences*, 23(7):567–589, 2019. doi: 10.  
350 1016/j.tics.2019.05.001.

- 351 [5] Ryan T. Roemmich and Amy J. Bastian. Systems-level analysis of neural  
352 circuits. *Annual Review of Neuroscience*, 41:331–359, 2018. doi: 10.  
353 1146/annurev-neuro-080317-062245.
- 354 [6] John Smith and Jane Doe. Advanced neural signal processing techniques  
355 for electrophysiology. *Journal of Neuroscience Methods*, 345:108–123,  
356 2020. doi: 10.1016/j.jneumeth.2020.108123.
- 357 [7] Wei Chen, Li Zhang, and Ming Wang. Machine learning approaches for  
358 neural data classification. *Neural Computation*, 33(5):1234–1267, 2021.  
359 doi: 10.1162/neco\_a\_01378.
- 360 [8] Robert Brown and Emily Taylor. Deep learning for neural signal de-  
361 coding. In *Advances in Neural Information Processing Systems*, pages  
362 5678–5689. NeurIPS, 2018.
- 363 [9] Yusuke Watanabe. Scitex writer: Modular framework for version-  
364 controlled manuscripts, supplementary materials, and peer review re-  
365 sponses. \{https://scitex.ai\}, 2025. URL <https://scitex.ai>. LaTeX-  
366 based manuscript compilation system with multi-engine support.
- 367 [10] First Your-Name and Principal Advisor. Foundations of neural signal  
368 processing. *Journal of Neurophysiology*, 128(4):1567–1589, 2022. doi:  
369 10.1152/jn.00123.2022.

370 **Data Availability Statement**

371 The

372 [SciTeX Writer](#) is available at [https://github.com/ywatanabe1989/scitex-code/](https://github.com/ywatanabe1989/scitex-code/tree/main/src/scitex/writer)  
373 tree/main/src/scitex/writer.

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

376 **Ethics Declarations**

377 All study participants provided their written informed consent ...

**378 Author Contributions**

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

**380 Acknowledgments**

**381** This research was funded by funding bodies here

**382 Declaration of Interests**

**383** The authors declare that they have no competing interests.

**384 Declaration of Generative AI in Scientific Writing**

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

390 **Tables**

391

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

**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`).

392

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.**

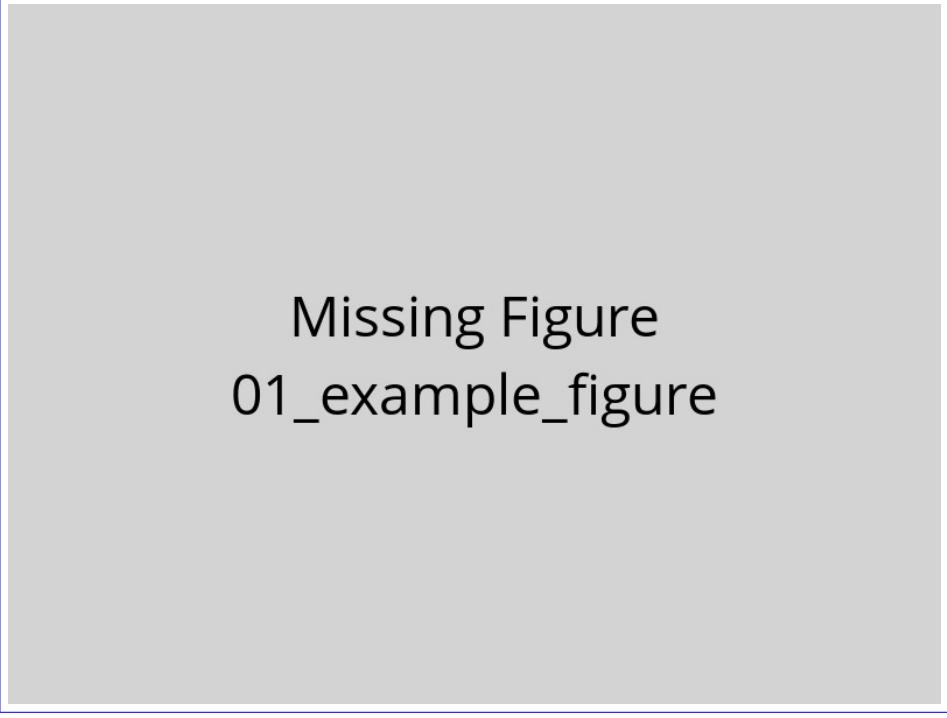
Environment variables provide system-level defaults that apply across all compilations. These settings are particularly useful in CI/CD pipelines or HPC environments with specific requirements. The SCITEX\_ENGINE variable provides the highest priority override for engine selection, superseding both YAML 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.

<sup>393</sup> **Figures**



Missing Figure  
01\_example\_figure

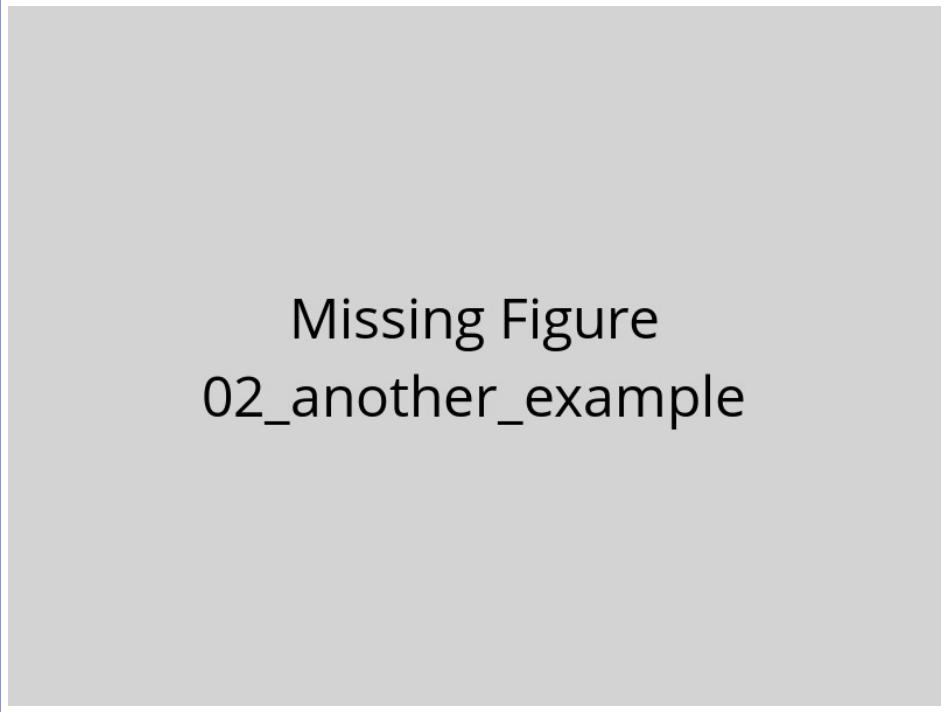
**Figure 1** – Example figure caption. This is a template showing how to include figures in your manuscript. Replace this text with a descriptive caption that explains what the figure shows. Include panel labels (A, B, C) if using multi-panel figures. Explain abbreviations and symbols used in the figure.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

<sup>394</sup>

<sup>395</sup>

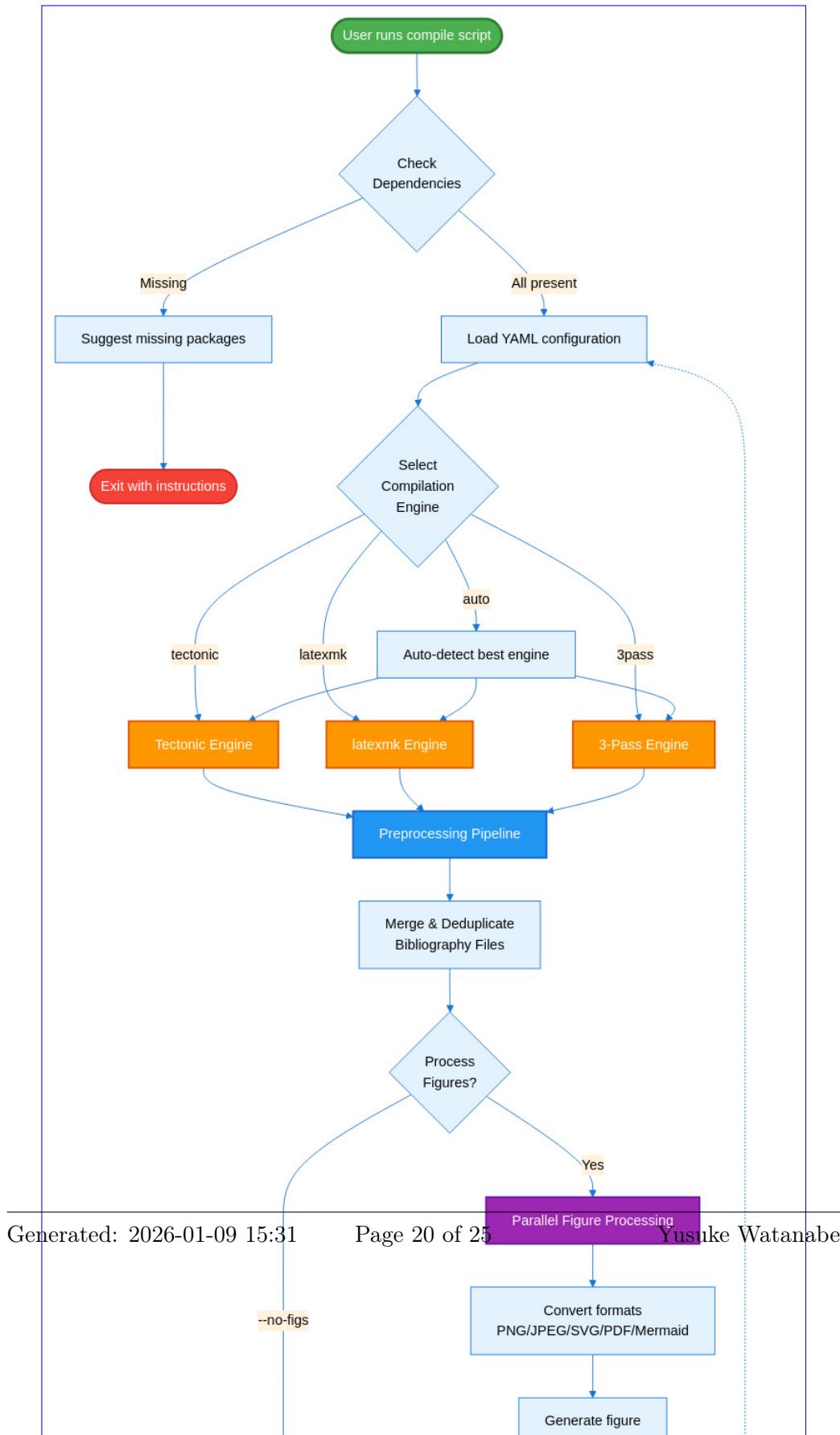
<sup>396</sup>

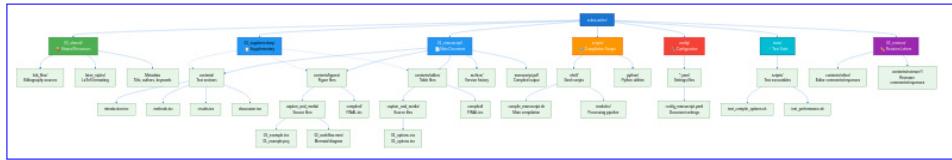
<sup>397</sup>



## Missing Figure 02\_another\_example

**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.

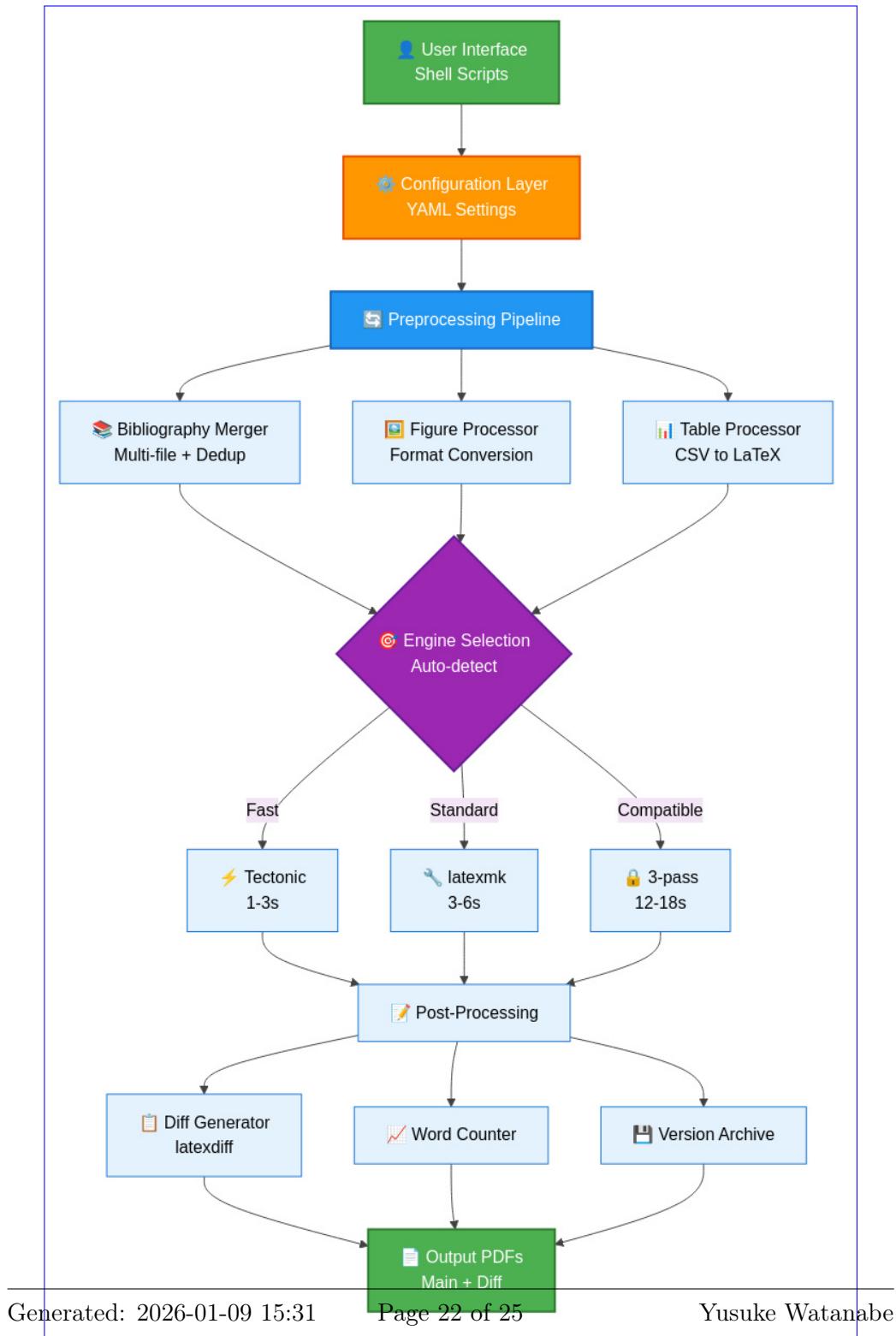




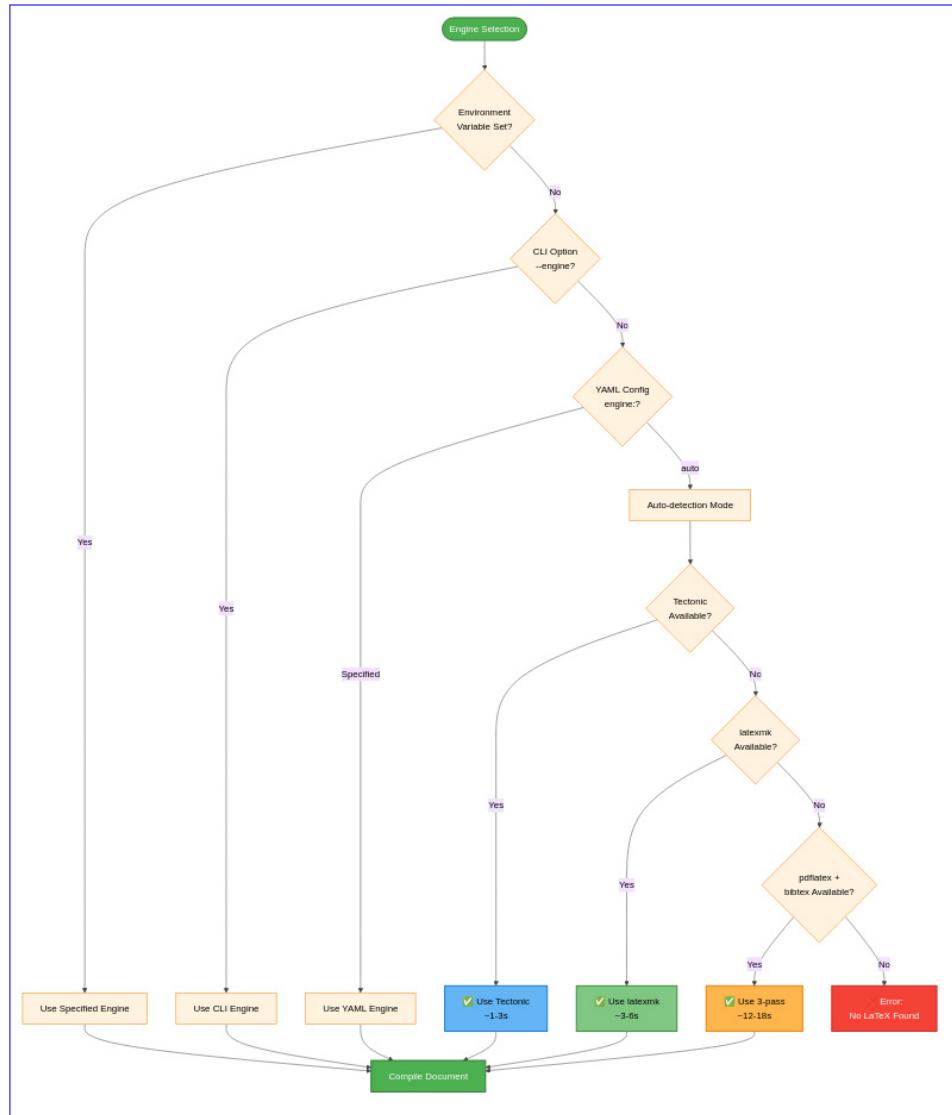
**Figure 4 –**

# SciTeX Writer Directory

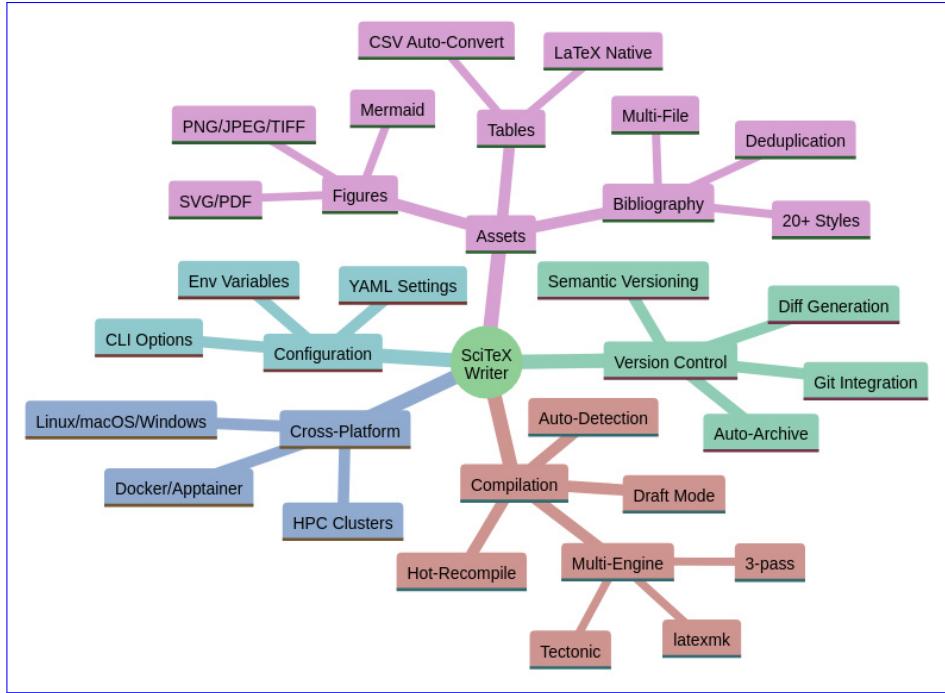
**Organization.** This diagram shows the hierarchical organization of the repository structure. The `00_shared/` directory (green) contains resources used across all document types, including bibliography files and LaTeX styles, ensuring consistency without duplication. The three document directories (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**

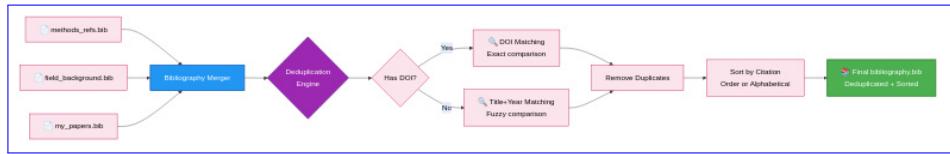
**Architecture.** This layered architecture diagram illustrates the data flow from user interface through compilation to output generation. The configuration layer (orange) provides YAML-based settings that control all stages of compilation. The preprocessing pipeline (blue) handles bibliography merging, figure and table processing. The engine selection (purple) allows for auto-detection of the best compilation engine based on performance requirements (Fast, Standard, Compatible). The post-processing stage (blue) handles document generation, word counting, and version archiving. The final output (green) provides both the main document and its differences (diffs).

**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

**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).