Issues in Information Systems
Volume 27, Issue 1, pp. xx-xx, 2026

DOI: https://doi.org/10.48009/1_iis_2021_xx-xx (Note: The header will be completed by IIS editor, keep line space)

Rendered document indexing vs. structured source indexing: A comparative study for design-document multimodal RAG

Gerald Rigdon
Fellow, Software Engineering, Boston Scientific Corporation, St. Paul, USA
gerald.rigdon@bsci.com

Abstract

Firmware design documents often combine prose, diagrams, structured metadata, and traceability references, creating challenges for searchable indexes used in Retrieval Augmented Generation (RAG) pipelines. This research evaluates how document representation affects retrieval quality for medical device firmware design content. Using 101 firmware design queries, we compared indexes built from vision-enriched PDF sources, vision-enriched XML sources, and text-only XML sources. The experiment yielded three key findings: (1) vision-enriched indexes substantially improved retrieval for diagram-dependent queries compared with text-only indexing; (2) XML-based indexes outperformed PDF-based indexes for diagram-related questions, suggesting the value of structured source representations that preserve relationships between diagrams and surrounding prose; and (3) PDF-based indexes outperformed XML-based indexes on text-only queries, indicating that rendered PDF document structure may preserve useful narrative information that is lost or fragmented during XML extraction. Together, these results show that no single source format is optimal across all query types and that effective retrieval requires deliberate choices about document representation.

Keywords: document indexing, RAG, structured XML, PDF processing, GPT-5

Glossary

AI – Artificial Intelligence
ADI – Azure Document Intelligence
GPT – Generative Pre-trained Transformer
LLM – Large Language Model
OCR – Optical Character Recognition
PDF – Portable Document Format
RAG – Retrieval Augmented Generation
SCM – Software Configuration Management
SME – Subject Matter Expert
XML – Extensible Markup Language

Introduction

In traditional RAG processing, data stored in vector databases is retrieved and then made available to LLM context when answering queries. With the integration of vision models for enhanced document processing (Multimodal RAG), LLMs are exposed to even richer content resulting in improved response quality. Given a variety of document types and configurations, managing vision processing presents challenges that require architectural and design choices for optimal performance.

Firmware design documents describe how the firmware meets requirements and typically contain:

- Prose: Functional descriptions, rationale, algorithms, fault handling logic, etc.
- Diagrams: State machines, block, data flow, timing, and context diagrams, etc.
- Structured references: Traceability to requirements, cross-references to other design sections, references to source code files, etc.
- Metadata: Feature descriptions, revision history, authorship, etc.

An effective RAG system must index all this data to retrieve both the textual and visual content. Consider a configuration environment where the master design documents are in XML format, and the schema supports a path to resolve the visual images. Later in this document processing workflow, these XML documents and the associated visual images are converted to a single PDF format for publishing.

Let us now consider three approaches to managing this document processing system.

Configuration A (PDF + Vision): Treat the rendered PDF as the source of truth. Use Azure Document Intelligence (ADI) to extract text, tables, and layout from the PDF. Extract embedded images and describe them using the GPT-5 LLM. Chunk the text using fixed-size character splits where the vision descriptions are stored as separate chunks grouped by page.

Configuration B (XML + Vision): Treat the XML as the source of truth. Parse the XML structure to extract sections, traceability references, requirement IDs, and hierarchical metadata. Resolve references to the external image files and describe them using the same GPT-5 LLM. Chunk per document element (one chunk per section), with vision descriptions co-located inline.

Configuration C (XML No Vision): Identical to Configuration B except image references are not resolved or described. When a <graphic> or <figure> element is encountered, a placeholder ([Figure: filename]) is inserted in place of a vision description. All other processing remains the same.

Both Configuration A and Configuration B use GPT-5 for image understanding. The key differences in these configurations are in the text extraction method (PDF layout inference vs. XML direct parsing), chunking strategy (PDF fixed-size vs. XML per-element), and how vision descriptions are integrated (PDF separate chunks vs. XML co-located with section text). The tradeoffs in these configurations are captured in Table 1.

Table 1. Comparing the configurations.

Dimension	PDF + Vision (Configuration A)	XML + Vision (Configuration B)	XML No Vision (Configuration C)
Input	Rendered PDF (flat layout)	Source XML + separate image files	Source XML (no image files)
Text extraction	ADI layout inference (font, position)	Direct XML parse (tags, attributes)	Direct XML parse (tags, attributes)
Structure	Inferred (headings by font size)	Explicit (XML hierarchy, section nesting)	Explicit (XML hierarchy, section nesting)
Metadata	Minimal (page number, position)	Full (req id, traceability, feature, availability)	Full (req id, traceability, feature, availability)

Image handling	GPT-5 on PDF-extracted images (separate chunks)	GPT-5 on source image files (co-located in section chunk)	Placeholder text only ([Figure: filename])
Chunking	Fixed-size character splits (1000 chars)	Per-element (one chunk per section/requirement)	Per-element (one chunk per section/requirement)
Vision co-location	Vision descriptions in separate chunks from text	Vision descriptions inline within section chunk	N/A (no vision descriptions)
Cost	ADI per page + Vision per image	Vision per image only (XML parsing is free)	$0 (XML parsing only)
Applicability	Any document (including scans)	Only documents with available structured source	Only documents with available structured source

We present a controlled experiment using actual medical device design documentation from stored repositories. This involves comparing both Configuration A and Configuration B on the same design document corpus. Finally, we supplement the investigation using XML without vision as a control to isolate the specific contribution of vision-generated image descriptions.

Prior Research

Although the roots of retrieval systems go back decades, the formalization of RAG occurred in 2020 when Lewis et al. (2020) introduced a method for grounding LLM responses in retrieved evidence. This RAG approach enabled LLMs access to knowledge stores that were not included in model pre-training. This was presented to improve response and reduce LLM hallucination without model re-training. Following was an explosion of RAG approaches and in 2024 Gao et al. (2024) provided a comprehensive survey of RAG architectures, categorizing them into naive, advanced, and modular designs. Chen et al. (2024) proposed a RAG benchmark to support diagnosing challenges of LLM use with RAG. More recently in November 2025 Zhang et al. (2025) conducted a comprehensive survey of Multimodal RAG and presented a taxonomy which identified four essential workflow stages for these types of RAG systems.

Our work contributes to this space by comparing three different indexing configurations applied to the same source documents and evaluating the outcomes for design documents created to describe firmware in medical devices. This is a domain with highly specialized visual conventions such as state machines, sequence diagrams, timing charts, etc., where support for multimodal processing would intuitively result in better retrieval performance.

Background

ADI (Microsoft Azure, 2024) provides pre-built models for document analysis:

- prebuilt-read: OCR + text extraction
- prebuilt-layout: Text + tables + figures + selection marks + bounding boxes
- prebuilt-document: Key-value pairs + entities + tables

Firmware design PDFs use the prebuilt-layout models which return:

- Pages with text lines and their positions
- Tables with row/column structure and cell content
- Figures with bounding polygons and extracted text within the figure region
- Section headings that are inferred from font size and position

The output format is text that preserves table structure and heading hierarchy inferred from visual content.

Limitations for PDF documents:

- No traceability: The PDF may contain text like “Traced to: SysRS001234” which is a text string and not a queryable metadata field.
- No element IDs: The requirement ID “FwRS001957” appears as inline text which is not a filterable reqid field.
- Weak figure understanding: A state machine diagram is described by its bounding box and any OCR derived text rather than its semantic meaning.
- Heuristic font detection: Font-based heading detection can fail on design documents with complex formatting. For example: multi-column layouts, nested tables, callout boxes, etc.

Research Methodology

For both A and B, the data flow begins with an original XML document as shown below in Figure 1 and Figure 2.

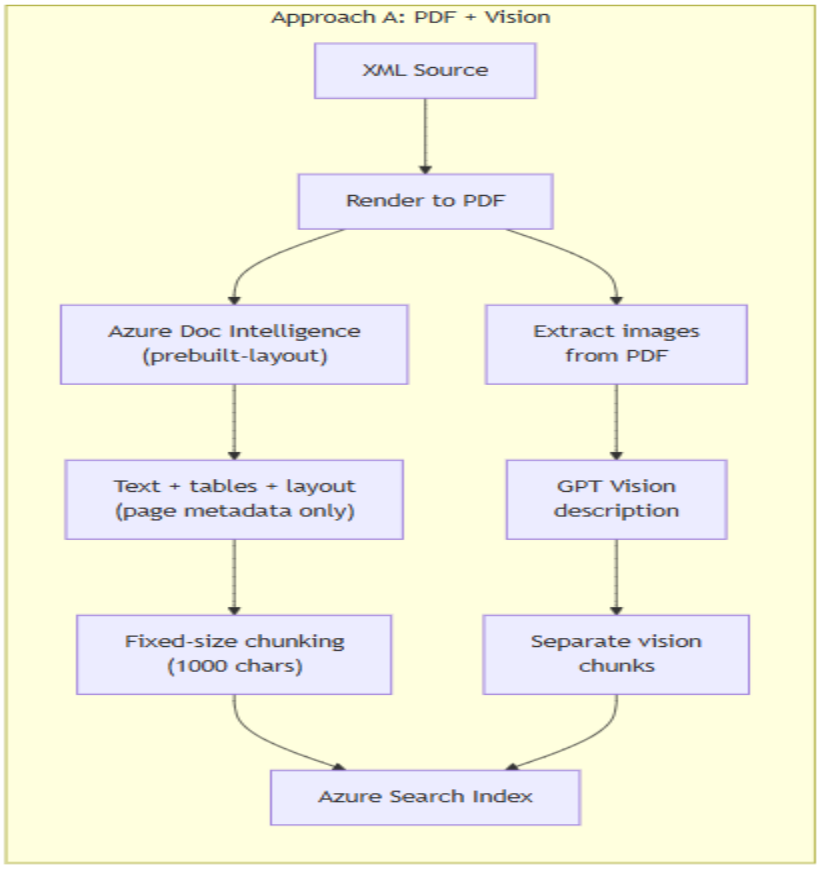

Figure 1. Configuration A (PDF + Vision) processing pipeline.

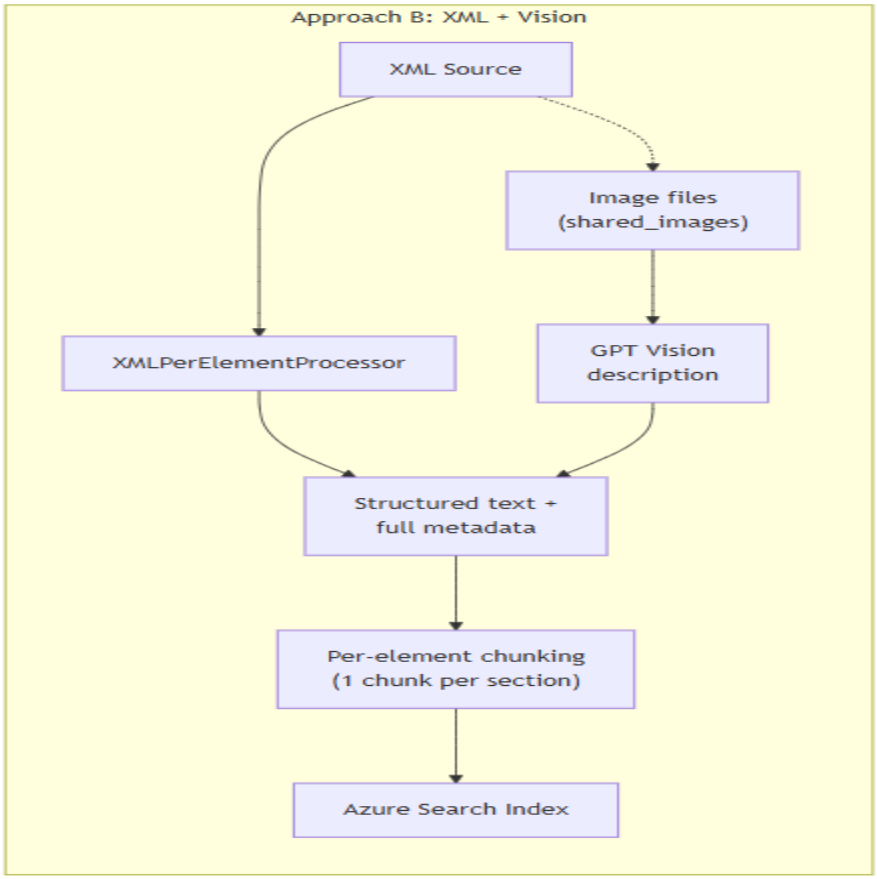

Figure 2. Configuration B (XML + Vision) processing pipeline.

How Vision Descriptions Are Managed
Multimodal LLMs can analyze images and produce detailed textual descriptions. However, the quality of vision descriptions depends heavily on:

- Prompt engineering: A generic "describe this image" prompt produces vague output. A domain-specific prompt like: "This is a firmware design diagram. Describe all state names, transitions, signal names, thresholds…" produces rich content.
- Image quality: Low-resolution images reduce vision model accuracy.
- Diagram complexity: Simple visuals are more likely to be described accurately vs. complex ones.

The implementation in this research uses a VisionDescriber class with a specific prompt:

> *"This [figure label] is from a firmware design document. Describe what this diagram or figure shows in detail. Include all labels, signal names, state names, data flows, equations, thresholds, and relationships visible in the image. Be thorough but concise."*

This prompt is optimized for firmware content and consistently produces descriptions that capture the engineering-relevant content of diagrams.

Where Vision Descriptions Are Located
The critical difference between Configuration A and B is where the vision description ends up relative to the section text.

Consider a pressure sensor design document section. The XML contains:

```
<section name="PressureSensor_StateMachine">
  <title>Pressure Sensor State Machine</title>
  <description>The pressure sensor module operates as a state machine
    with the following states and transitions.</description>
  <figure name="pressure_sensor-figure-001">
    <caption>Pressure Sensor State Machine</caption>
    <graphic src="pressure_sensor_sm.gif"/>
  </figure>
  <traceability refer_to="FwRS001957"/>
  <traceability refer_to="FwRS001958"/>
</section>
```

A (PDF + Vision) produces two separate chunks:

- Chunk 1 from ADI: “Pressure Sensor State Machine. The pressure sensor module operates as a state machine with the following states and transitions. [Figure region detected] Idle Init Active Fault”.
- Chunk 2 (from VisionDescriber): “State machine diagram with four states: Idle (initial state), Init (sensor_enable, 50ms timeout), Active (sampling at 100Hz), Fault (sensor_error, recoverable via reset)”.
- Metadata: page number only

B (XML + Vision) produces one unified chunk:

- Single chunk: Section text + inline vision description: “Pressure Sensor State Machine. The pressure sensor module operates as a state machine with the following states and transitions. [Figure 1: Pressure Sensor State Machine] State machine diagram with four states: Idle (initial state), Init (sensor_enable, 50ms timeout), Active (sampling at 100Hz), Fault (sensor_error, recoverable via reset)”.
- Metadata: reqid = PressureSensor_StateMachine, traceability = [FwRS001957, FwRS001958]

C (XML No Vision) produces one chunk with a placeholder instead of a description:

- Single chunk: Section text + placeholder: “Pressure Sensor State Machine. The pressure sensor module operates as a state machine with the following states and transitions. [Figure: pressure_sensor_sm.gif].”
- Metadata: reqid = PressureSensor_StateMachine, traceability = [FwRS001957, FwRS001958]
- The chunk has the same structure, metadata, and element boundaries as Configuration B but the figure is just a filename reference alone that carries no semantic content.

For vision related configurations, the separation of vision descriptions into standalone chunks is the natural result of how the PDF pipeline processes text and images independently. ADI extracts text, GPT-5 describes images, and then the two outputs are combined (but as separate documents in a list) before chunking. Co-locating them for PDFs, which is done more naturally in the XML case, would require additional engineering to merge vision descriptions into the ADI text by page before chunking. This modification could be implemented but natural separation creates an observable difference in retrieval behavior worth studying.

Experimental Evaluation

LLM Configuration

GPT-5 was used for text, vision, and as the metrics judge.

Research Questions

RQ1: Does XML-based indexing (per-element chunking, full metadata, co-located vision) produce higher retrieval quality than PDF-based indexing (ADI layout extraction, fixed-size chunking, separate vision chunks) for firmware design document queries?

RQ2: What is the specific contribution of vision-generated image descriptions to retrieval and answer quality, independent of text extraction and chunking strategy?

RQ3: For text-only queries where vision is irrelevant, does the text extraction method (ADI layout vs. XML parsing) matter?

Data Set

Design document corpus from a medical device project:

- Source XML files from Software Configuration Management (SCM) system
- Shared images from SCM design/xml/images and requirements/images paths

The design document corpus provides a controlled comparison because:

- The same content exists in both XML and PDF form
- The documents are rich in diagrams (state machines, block diagrams, data flows)
- The documents have formal traceability to requirements
- The corpus is large enough to produce statistically meaningful results

Table 2. Index sizes after processing.

Configuration	Indexed Chunks	Description
A: PDF + Vision	598	Page-level chunks plus separate vision chunks
B: XML + Vision	567	Per-section chunks with inline vision
C: XML No Vision	568	Per-section chunks with placeholders

Test Set and Query Taxonomy

101 synthetic queries (59 diagram-dependent, 42 text-only), were generated using RAGAS (Es et al., 2023) from the indexed content. Each query includes a synthetic reference answer, source file, and category tag. All three configurations A, B, and C were evaluated against the same test set using hybrid search with top_k=10 without re-ranking. To avoid source bias, synthetic queries were generated from A, B, and C indexes independently, then merged and de-duplicated by semantic similarity. The taxonomy is presented in Table 3.

Table 3. Taxonomy of Categories.

Category	Example Query	Expected Behavior
Text-only (baseline)	"Describe the pressure sensor initialization sequence"	All configs should retrieve relevant text sections
Diagram-dependent	"What states does the battery state machine have?"	Both A and B use Vision for images; B benefits from per-element chunking; C (no vision) gets placeholder only

Metadata-filtered	"Find all design sections traced to FwRS001957"	XML configs (B, C) use metadata filter; PDF config (A) relies on text matching
Cross-reference	"What design sections reference the RechargeableBattery feature?"	XML configs use availability metadata; PDF config searches for text string

Metrics

This study employs RAGAS (Es et al., 2023) to generate the synthetic test set (queries + reference answers) used across all metrics. The RAG Triad metrics require an LLM but do not require a reference answer for comparison because the LLM judge only evaluates the relationship between query, retrieved chunks, and the generated answer. While the RAGAS library is also used for the RAG Triad evaluation, these results are not biased by the synthetic reference answer quality as is the case for other metrics.

LLM Answer Quality Metrics (RAG Triad — LLM-as-judge):

- **Context Relevance (0–1):** Are the retrieved chunks relevant to the query? This measures how much of the retrieved context is useful for answering the question. An LLM judge scores each retrieved chunk based on its relevance to the query.
- **Faithfulness (0–1):** Is the generated answer grounded in the retrieved chunks? This measures whether the claims in the generated answer are supported by the retrieved context. High faithfulness indicates low hallucination.
- **Response Relevancy (0–1):** Does the generated answer address the query? This measures whether the answer is on-topic relative to the question, regardless of factual correctness.

The following Accuracy and Completeness metrics do require a reference answer and are sensitive to synthetic test quality.

LLM Answer Correctness (LLM-as-judge):

- **Accuracy (0–1):** Of the facts in the generated answer, what fraction is correct according to the reference answer? This is analogous to precision where additional correct details beyond the reference answer do not reduce the score.
- **Completeness (0–1):** Of the facts in the reference answer, what fraction is covered by the generated answer? This is analogous to recall where missing facts reduce completeness while incorrect facts do not.

The following ROUGE and BLEU metrics do not require an LLM but still require a reference answer for comparison.

Text Overlap Metrics (non-LLM, reference-dependent, via RAGAS library):

- **ROUGE Score (0–1):** N-gram (continuous sequence of N words or characters) overlap between generated and reference answers. Standard Natural Language Processing (NLP) recall-oriented metric.
- **BLEU Score (0–1):** N-gram precision between generated and reference answers. Standard machine translation metric applied to answer quality.

The following metrics do not require an LLM or a reference and are available in Python libraries.

Non-LLM Heuristics (no LLM calls, computed from string matching):

- **Chunk Utilization (0–1):** Fraction of retrieved chunks that contributed content to the answer, measured by 3-word sliding window overlap. Higher means the LLM used more of what was retrieved.

- **Query Term Coverage (0–1):** Fraction of substantive query terms, after stop-word removal, that appear in the generated answer. Higher values suggest that the answer addresses the specific terms used in the query although valid paraphrases may be missed.
- **Hedging Score (0-1):** Detects refusal, uncertainty, or insufficient-evidence phrases, such as "I don't have enough information" or "the context does not mention." A score of 0.0 indicates little or no hedging, while 1.0 indicates heavy hedging. Lower generally means the answer is more direct.
- **Answer Length:** Character count of the generated answer.

Results and Discussion

We evaluated all three configurations A, B, and C, on the same 101-query test set (59 diagram-dependent, 42 text-only). Table 4 reports the aggregate scores.

Table 4. Overall aggregate results.

Metric	A: PDF + Vision	B: XML + Vision	C: XML No Vision
Context Relevance	0.877	0.877	0.745
Faithfulness	0.901	0.931	0.896
Response Relevancy	0.793	0.778	0.612
Accuracy	0.510	0.510	0.464
Completeness	0.455	0.468	0.362
ROUGE	0.316	0.333	0.318
BLEU	0.151	0.145	0.177
Chunk Utilization	0.833	0.608	0.451
Query Term Coverage	0.648	0.702	0.614
Hedging Score	0.092	0.066	0.073
Avg Latency (ms)	5,345	3,265	2,712
Total Duration (s)	1,838	874	523

The results reveal that at the aggregate level, A and B are hard to separate which is itself meaningful. However, C clearly lags in context relevance, response relevancy, and completeness. B stands out because it has the highest query term coverage and the lowest hedging score, which together suggest it tends to answer questions more directly.

Table 5 shows the diagram-dependent slice.

Table 5. Per-tag breakdown: diagram-dependent queries (n = 59).

Metric	A: PDF + Vision	B: XML + Vision	C: XML No Vision
Context Relevance	0.863	0.953	0.719
Faithfulness	0.831	0.881	0.822
Response Relevancy	0.788	0.915	0.632
Accuracy	0.408	0.418	0.339
Completeness	0.442	0.469	0.322
ROUGE	0.366	0.368	0.331

In this context B is the clear winner, especially in response relevancy. At 0.915 it is well ahead of A at 0.788 and C at 0.632. Context relevance is similar (0.953 for B vs. 0.863 for A and 0.719 for C). Since all configurations use the same GPT-5 vision model, B's advantage comes from co-locating per-element chunking with vision descriptions in the structured document representation.

Table 6 shows the text-only slice.

Table 6. Per-tag breakdown: text-only queries (n = 42).

Metric	A: PDF + Vision	B: XML + Vision	C: XML No Vision
Context Relevance	0.898	0.771	0.781
Faithfulness	1.000	1.000	1.000
Response Relevancy	0.800	0.586	0.583
Accuracy	0.654	0.640	0.639
Completeness	0.474	0.467	0.417
ROUGE	0.245	0.283	0.300

On text-only queries, A generally comes out ahead although all three configurations hit perfect faithfulness. Overall, the difference between B and C is negligible and what we would expect given the image component has nothing to add.

Table 7 isolates the vision contribution by comparing B against C, which are identical except that C swaps every vision description for a filename placeholder.

Table 7. Isolating vision's contribution (B vs. C).

Metric	B (Vision)	C (No Vision)	Delta	Significant?
Diagram: Response Relevancy	0.915	0.632	+0.283	Yes
Diagram: Context Relevance	0.953	0.719	+0.234	Yes
Diagram: Faithfulness	0.881	0.822	+0.059	Moderate
Diagram: Completeness	0.469	0.322	+0.147	Yes
Diagram: Accuracy	0.418	0.339	+0.079	Moderate
Text-only: Response Relevancy	0.586	0.583	+0.002	No
Text-only: Accuracy	0.640	0.639	+0.001	No
Text-only: Completeness	0.467	0.417	+0.050	Marginal

For diagram queries the effect is large. Vision increases response relevancy by +0.283 and context relevance by +0.234, the two biggest swings anywhere in the study. For text-only queries it is essentially nil (+0.001 accuracy). This demonstrates that instead of being a blanket improvement, vision pays off precisely when the answer to the query depends on the diagram.

Vision descriptions as retrieval anchors

A more surprising pattern showed up when we looked at Configuration B on its own per Table 8. Within the same index, diagram queries actually retrieve better than text-only queries. This runs against the intuition that text is easier to process and would produce the best results.

Table 8. Within Configuration B: diagram vs. text-only retrieval.

Metric	B: Diagram Queries	B: Text-Only Queries
Context Relevance	0.953	0.771
Response Relevancy	0.915	0.586

Two things seem to be happening. First, vision descriptions are dense and distinctive. Thus, a GPT-5 vision description like "state machine with four states: Idle, Init, Active, Fault; Delimit on sensor_ready with a 50 ms timeout guard" is full of specific terms that produce sharp vector matches, whereas ordinary firmware prose such as "initialization" and "error handling" repeats across many sections and blurs retrieval precision. Second, diagram questions tend to be more specific. For example, "Which module connects to Charger Communication via GPIO1?" carries more discriminating terms that land on a single chunk

whereas "How does pressure initialization work?" is broad enough to scatter across several partial matches. The practical takeaway is that GPT-5 descriptions act as high quality semantic anchors that are among the most retrievable chunks in the index.

Limitations and Generalizability

These results come from a single corpus which shapes their generalizability. The points below help frame the limitations in this study.

Dataset. All measurements derive from one medical device firmware project, indexed from a single SCM repository. Moreover, the documents follow domain-specific authoring conventions and one specific XML schema. While the relative comparisons are controlled by using identical content in both XML and PDF formats, not all absolute metrics are transferable.

Domain-specific visual conventions. Firmware design diagrams are highly stylized with state machines, timing charts, block and data-flow diagrams with consistent labeling. The large vision benefit we observe depends on GPT-5 reading those conventions well and generating good descriptions. Cases where the visuals are more free-form or inconsistently labeled may yield less reliable vision descriptions.

Dependence on a structured source. Configurations B and C assume a structured source with resolvable image paths and rich metadata such as requirement IDs, traceability, and feature tags. However, many real document sets exist only as rendered PDFs or scans, where Configuration A is the only option. The XML advantage on diagram queries is therefore available only to such structured authoring formats.

Applicability across industries. We expect the qualitative pattern of co-located, semantically dense vision descriptions to help most with diagram-heavy queries and layout-aware PDF extraction for narrative text to extend to other engineering and regulated domains. However, the magnitude of the benefit will depend on how diagram-dependent a document corpus is and whether a structured source is available. For example, text-heavy domains with few diagrams would likely show a much smaller vision effect.

Alternative document structures. We tested only one structured XML format against rendered PDF. Other structured representations such as HTML and Markdown carry different amounts of explicit structure and metadata. Hence, other such formats could be used in further studies to determine whether the per-element and co-location advantages observed in this study are general properties of structured source rather than the result of a particular schema.

Evaluation method. The queries and reference answers are synthetic, generated by RAGAS rather than validated by subject matter experts. Additonally, the study uses only the GPT-5 model and only one retrieval configuration, namely, hybrid search, top_k = 10, and no re-ranking. While the RAG Triad metrics measure the integrity of the RAG pipeline without reference bias, the other reference-dependent metrics such as accuracy, completeness, ROUGE and BLEU would inherit any bias in the synthetic answers.

Future Directions

Several follow-ups would sharpen these findings:

- Address the confound. Inject vision descriptions into the PDF pipeline and merge them into the ADI text before chunking to test whether co-location alone explains B's diagram advantage.
- Build a golden test set. Replace synthetic reference answers with SME-verified ground truth to bolster the accuracy and completeness results.

- Try per-element chunking for PDF. Use ADI's heading detection to create element-level chunks from the PDF, reducing the chunking confound.
- Run a broader tradeoff analysis. A modified Configuration A with inline vision (PDF + inline vision) might be the best general-purpose retrieval setup. However, dropping XML would forfeit structured metadata that keeps documents sections coherent and cross-references typed.

Conclusion

This study compares PDF-based and XML-based indexing for firmware design document RAG. Two pipelines used GPT-5 for image descriptions and differed in text extraction (ADI layout vs. XML parsing), chunking (fixed-size vs. per-element), metadata (page number vs. full traceability), and vision co-location (separate chunks vs. inline). The third pipeline was XML-based without vision.

RQ1 asked whether XML-based indexing beats PDF-based indexing on retrieval quality. For diagram-dependent queries we found that B outperforms A on response relevancy (+0.127), context relevance (+0.076), and completeness (+0.027). But, at the aggregate level, A and B are surprisingly close. Context relevance is identical (0.877) and accuracy is tied (0.510). Thus, the XML advantage is concentrated in diagram-heavy queries, where co-located vision descriptions and per-element chunking let the retriever surface both the text and the visual semantics in one chunk.

RQ2 asked what vision contributes on its own, isolated by the B vs. C comparison. The answer is clear and category-dependent. For diagram queries, GPT-5 descriptions improve response relevancy by +0.283 and context relevance by +0.234 which are the two largest deltas in the experiment. For text-only queries they add essentially nothing (+0.001 accuracy). Vision is a targeted improvement that matters exactly when the answer depends on information conveyed by a diagram.

RQ3 asked whether the text extraction method matters for text-only queries. The data shows an edge for PDF where it beats both XML configurations on response relevancy (0.800 vs. 0.586) and accuracy (0.654 vs. 0.640). The ADI PDF layout-aware extraction appears to produce more natural prose for text-heavy sections, whereas XML's per-element chunking, which is optimized for structured elements, can fragment narrative text into chunks that are individually correct but less useful for open-ended questions.

Two surprises stand out:

1. PDF-based indexing outperforms XML on text-only queries (0.898 vs. 0.771 context relevance). ADI's layout analysis produced a better prose representation than XML per-element extraction.
2. Within the same index, vision-described chunks retrieve better than plain-text chunks. Diagram queries score higher than text-only queries in Configuration B.

Acknowledgements

The author thanks Ankur Kaushik (Principal Software Engineer, Boston Scientific Corporation) for his contributions to this work.

References

Chen, J., Lin, H., Han, X., & Sun, L. (2024). Benchmarking large language models in retrieval-augmented generation. arXiv. https://arxiv.org/abs/2309.01431

Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). RAGAS: Automated evaluation of retrieval augmented generation. arXiv. https://arxiv.org/abs/2309.15217

Gao, Y., Xiong, Y., Gao, X., et al. (2024). Retrieval-augmented generation for large language models: A survey. arXiv. https://arxiv.org/abs/2312.10997

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv. https://arxiv.org/abs/2005.11401

Microsoft Azure. (2024). Azure AI Document Intelligence documentation. https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/

Zhang, R., Liu, C., et al. (2025). A comprehensive survey on multimodal RAG: All combinations of modalities as input and output. TechRxiv. https://www.techrxiv.org/doi/full/10.36227/techrxiv.176341513.38473003/v2